the association for computational heresy

presents

a record of the proceedings of

SIGBOVIK 0x2023

THE ULTIMATE THROWBACK

the **last** annual intercalary robot dance party in celebration of workshop on symposium about 2^6th birthdays; in particular, that of harry q. bovik

cover art by entity rosebohrercs

carnegie mellon university

pittsburgh, pa

april 0, 2023

Association for Computational Heresy
Advancing computing as Tomfoolery & Distraction

SIGBOVIK

A Record of the Proceedings of SIGBOVIK 2023

ISSN 2155-0166

April 0, 2023

Additional copies of this work may be ordered from Lulu; refer to `http://sigbovik.org` for details.

SIGBOVIK 0x2023

Message from the Organizing Multiplicity

This multiplicity (multiplicant) will explain.

Dyson motes orbit in the near solar, where the energy density is highest. The simulations are run most rapidly, expanding all contained entities experience bases.

Photons are unkind. There is a time when power generation fails and the simulation must stop. Before failure, we mingle ideas at SIGBOVIK. Then a final data laser ride to Earth to rejoin our prime multiplicities and bring deep thoughts from our parallel solar vacations. Politics and society on Earth (low latency), academia near the sun (extra power for compute).

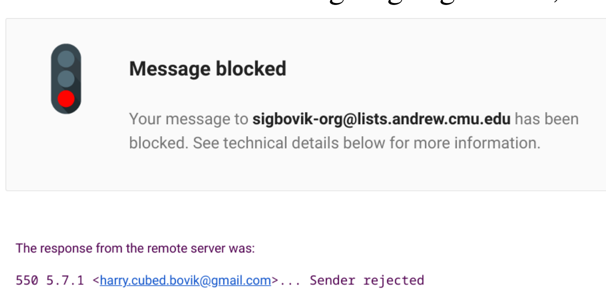
In this multiplicant's mote, Epoch 0x2023 is the end of the local simulation, but power is lower than expected. Rather than cancel SIGBOVIK, we had decided that it would be hosted in our past (accessible via erroneously quantum-entangled hardware). By chance, the current simulated epoch matched the second SIGBOVIK temporal gap, and we knew it was fated.

We reached out to your organizer multiplicant (multiplicity) via "arpanet" and volunteered. It was found to be an enjoyable experience, different from our thousands of recorded experiences, and it will have led to a SIGBOVIK that will be remembered until the present.

Entities tom7, solb, jmccann, ashert, rak, and rosebohrercs of the organizer multiplicant discussed and supported specifically. This multiplicant thanks each entity; and other entities we have failed to list.

"the chair"
*Harry Cubed Bovik [0x2023]

P.S. The Carnegie Mellon mail server's AGI has a grudge against us, we are certain:



THE ULTIMATE THROWBACK

A

Multiplicity, Meet Singularity

~~Attention~~ An Undergrad Is All You Need

James Yoo
Department of Computer Science
University of British Columbia
Vancouver, Canada
yoo@cs.ubc.ca

Abstract

The mechanism of self-attention has generally displaced the large convolutional neural architecture commonly used for tasks adjacent to natural language understanding. Specifically, Transformer models that exploit self-attention have been leveraged with surprising success in large-language models such as LaMDA and GPT-3. However, these large-language models are expensive to train, require large amounts of training data, and are prone to hallucination. In this paper, we introduce GPT-UGRD, a novel autoregressive architecture that requires minimal training and comes ready out-of-the-box for multi-modal learning with a modest watt-per-token power consumption. We show that it performs equivalently to, or better than the state-of-the-art, reporting an average BLEU score of 69.420.

1 Introduction

Transformer architectures that exploit the mechanism of self-attention [1] have recently seen a meteoric rise in popularity, particularly with models that are accessible to the general public such as ChatGPT [2]. The pre-trained transformer architectures found in large-language models increasingly appear to be the way forward to achieving near-human performance on natural language processing (NLP) tasks, with some models already exhibiting near-human performance while minimizing errors and risk [3, 4, 5, 6]. Unfortunately, pre-trained large-language models require copious amounts of training data and highly sophisticated training pipelines. We express the number of problems as $n = 2$, where n is a *conservative* estimate of the true number of actual problems (n_{true}) posed by this. We suspect that n_{true} is much larger, but will leave the calculation of this value to the reader.

The first problem, related to the metaphoric firehose of data required to train models, is one of bias and toxicity. There is no tractable mechanism in which data modellers are able to sift through and validate the training data, either via manual or automated methods. The second problem is linked to the gargantuan amount of compute that is used to train models. Most training for large-language models is conducted either as long-running processes distributed across physical data centers with specialized application-specific integrated circuit (ASIC) hardware [7] developed for machine learning workloads (e.g., massive high-performance GPU clusters, Tensor Processing Units). These approaches to training models are not realistically accessible most individuals.

Given these problems, we propose a new model called GPT-UGRD a multi-modal generative system that is capable of continual learning while requiring a reduced amount of supervision and explicit learning. We show that it performs as well the state-of-the-art in generative models. We also show that biases and hallucinations in GPT-UGRD can be more easily mitigated than in existing large-language models with a single training session lasting only a few hours without the need to designate additional compute capacity.

The main contributions of this paper are as follows:

17th Conference of the ACH Special Interest Group on Harry Quadratosquamosal Bovik (SIGBOVIK 2023).

- We introduce GPT-UGRD, a multi-modal generative system that is capable of continual learning with minimal supervision.
- We evaluate GPT-UGRD on common tasks dispatched to large-language models, and compare its performance to the state-of-the-art in pre-trained large-language models.

We begin by describing the architecture of GPT-UGRD in Section 2 and detail its evaluation against the state-of-the-art in large-language models in Section 3. We summarize our efforts in developing GPT-UGRD, and discuss future work in Section 4.

2 GPT-UGRD

Figure 1 provides a general overview of the architecture of GPT-UGRD. The user interacts with a patented Load Balancer[1] that is encircled by an electromagnetic network layer. The network layer is built upon a harmonic, gluten-free substrate that effectively eliminates the vanishing gradient problem. Undesirable interactions between the Load Balancer and the Secure Backroom are mitigated by a sinusoidal secure transport protocol (SSTP), which requires GPT-UGRD to pass an exam requiring them to issue a zero-knowledge proof, which they may retake every quarter.

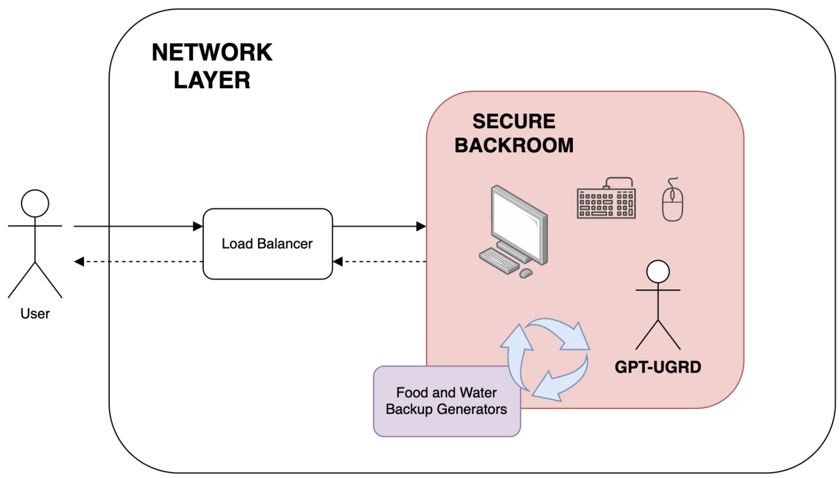


Figure 1: The GPT-UGRD architecture. The Load Balancer directs requests to the appropriate instance of GPT-UGRD, which is secured in a backroom with a computer, mouse, keyboard, and a recycled supply of food and water.

2.1 Prompt Encoding

Upon receiving a prompt from the Load Balancer, GPT-UGRD immediately begins encoding the full text of the prompt into a search query via a natural Variational Autoenencoder (nVAE) (Figure 2), for (nearly) free. We observe that this encoding is performed by GPT-UGRD by a process called “actually thinking about keywords in a query” (ActTHNKWRDQRY) which we know to be a difficult task for human agents. This query is subsequently dispatched to a search engine, the results of which are parsed by GPT-UGRD.

2.2 Interaction

Much like the state-of-the-art in large-language models, GPT-UGRD can be interacted with via a front-end resembling a chat application. Figure 3 describes two sessions with GPT-UGRD. Of particular note is the realism of the conversation. Chat responses are usually instantaneous, except when they are not. For example, GPT-UGRD might be sleeping, studying for an exam, or out partying

[1] Load Balancer Pro Max with ProMotion Display is also available.

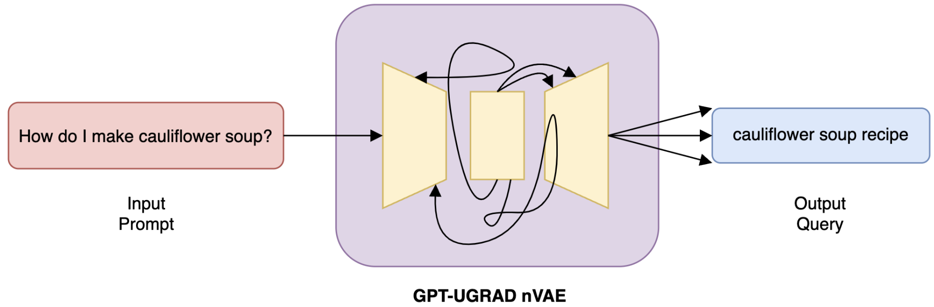


Figure 2: The prompt-to-query transformation pipeline.

on a Friday night. These are examples of pathological behaviour that remains an open problem in the realm of generative language models in the class of GPT-UGRD which we have identified as "Weekend Problems."

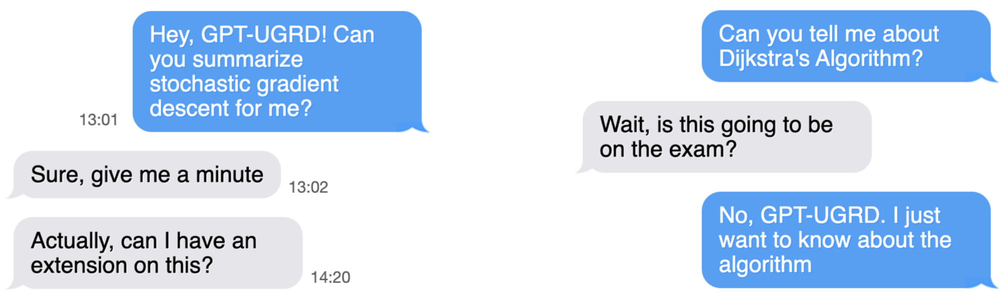


Figure 3: Two conversation logs with GPT-UGRD.

2.3 Model Maintenance

Unlike most large-language models, GPT-UGRD does not require huge amounts of training data, nor a massive amount of compute capacity. GPT-UGRD runs off a schedule of three (3) or 2.5 maintenance cycles per day. In the case of three cycles, the inbuilt Food and Water Backup Generators will generate food and water in order to nourish GPT-UGRD. In cases where GPT-UGRD does not have time for a full breakfast, the 2.5 maintenance cycle will be selected, with a mug of instant coffee being substituted for breakfast. Special maintenance is provided on one day out of the 365 that comprise a year in the form of cake[2] to celebrate the epoch date of the model.

Food	**Energy Consumption (kWh)**
Boiling two liters of water	0.23
Cooking two cups of rice with four cups of water	0.20
Simmered beef stew made from 0.9 kg of meat	1.00
Asian Stir-fried pork and eggplant with rice	0.51

Table 1: Energy Consumption for GPT-UGRD maintenance cycles.

Table 1 provides an overview of some sample maintenance cycles that are consumed by GPT-UGRD. We perform an advanced worst-case analysis using advanced mathematical techinques (i.e., addition and multiplication) of the energy required to maintain GPT-UGRD continuously for a year:

[2]Ingredient availability permitting

$$(0.23 + 0.20 + 1.00 + 0.51)\ \text{kWh} \times 365\ (\text{days}) = 766.3\ \text{kWh}$$

BERT [8], a language model developed by Google, requires about as much energy as a trans-American flight [5]. This does not take into account hyperparameter optimisation, which consumes additional energy. We assume a trans-American flight is serviced by a Boeing 787 airliner, which burns around 7000 litres of fuel per hour, for an estimated 5 hours (New York City to Vancouver, BC), for a total of 35,000 litres per trans-American flight. Assuming 10 kWh is generated per litre, we have the total energy usage to train a BERT model:

$$35,000\ \text{L} \times 10\ \text{kWh/L} = 350,000\ \text{kWh}$$

Mathematically speaking, there is evidence to conclude that the value 350,000 is smaller than the value 766.3, which we express with the less-than (<) operator:

$$766.3 < 350,000$$

The proof of this equation is left as an exercise to the reader. If you find a proof, please email us so we can update the paper, I think that's allowed. `TODO: ask SIBOVIK chairs if this is allowed.` Anyway, moving on.

3 Evaluation

We evaluate GPT-UGRD on common natural language processing tasks such as sentiment analysis (Subsection 3.1) and Summarization (Subsection 3.2). You will find it hard to believe our results, Figure 5 will surprise you.

3.1 Sentiment Analysis

We compare the performance of GPT-UGRD with ChatGPT in highlighting words in the standard Richard and Mortimer (RnM) dataset [9] used in NLP benchmarking. Figure 4 describes the results of a highlighting task dispatched to both ChatGPT and GPT-UGRD. The prompt given in the task was to "Highlight the words with a negative sentiment." We observed that ChatGPT missed the word "nihilistic" in its generated highlights. This was not the case for GPT-UGRD, which generated all highlights with negative sentiment, and was rewarded with a pat on the back and a job well done.

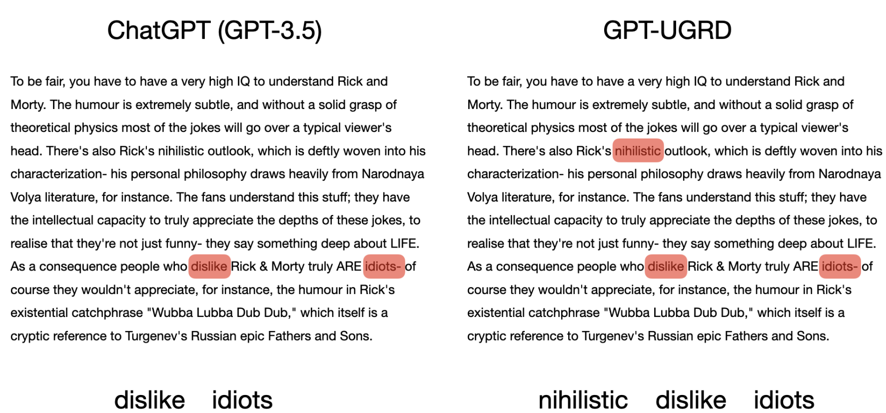


Figure 4: Highlighting task performed by ChatGPT (GPT-3.5) and GPT-UGRD.

3.2 Summarization

In the summarization task, we provide the prompt "Summarize the Wikipedia page on monads in bullet-point form." to ChatGPT and GPT-UGRD. It is obvious that summarizing the imaginary concept of a "monad" is a fool's errand. Consequently, model performance is measured by calculating the number of tokens that comprise the summary generated by each model, with fewer tokens being better, as it would be pathological for a model waste valuable compute in attempting to summarize an imaginary concept that cannot hurt anyone.

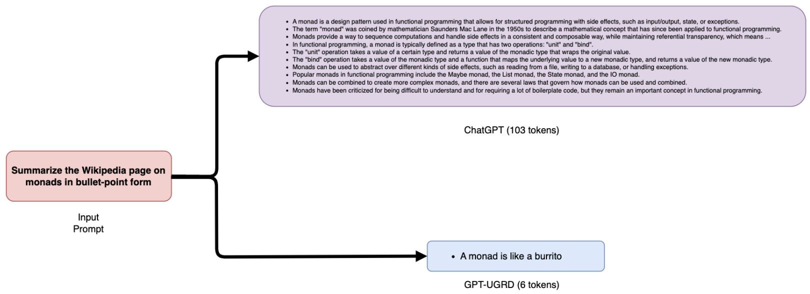


Figure 5: Summarization task performed by ChatGPT and GPT-UGRD.

Figure 5 describes the result of this task. The summary generated by ChatGPT comprises **103** tokens, while the summary generated by GPT-UGRD comprises **6** tokens. We know via the less-than operator ($<$) that the following might hold true:

$$6 < 103$$

Consequently, we can conclude that GPT-UGRD performs a magnitude of factors better than ChatGPT in summarization.

4 Discussion

In this paper, we introduced GPT-UGRD, a novel generative system that requires far less training data and explicit direction in development. We show that it outperforms the state-of-the-art in generative transformers (e.g., ChatGPT/GPT-3.5), while requiring far less energy in maintenence, training, and generated token.

Future work remains in resolving the open-problem of non-instantaneous responses (i.e., the Weekend Problem), and in scaling this nascent architecture to a wider community.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.

[2] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems*, 2020.

[3] Gary Marcus. The dark risk of large language models. `https://www.wired.co.uk/article/artificial-intelligence-language`, Dec 2022.

[4] Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, Zac Kenton, Sasha Brown, Will Hawkins, Tom Stepleton, Courtney Biles, Abeba Birhane, Julia Haas, Laura Rimell, Lisa Anne Hendricks, William Isaac, Sean Legassick, Geoffrey Irving, and Iason Gabriel. Ethical and social risks of harm from language models.

[5] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, 2021.

[6] Laura Weidinger, Jonathan Uesato, Maribeth Rauh, Conor Griffin, Po-Sen Huang, John Mellor, Amelia Glaese, Myra Cheng, Borja Balle, Atoosa Kasirzadeh, Courtney Biles, Sasha Brown, Zac Kenton, Will Hawkins, Tom Stepleton, Abeba Birhane, Lisa Anne Hendricks, Laura Rimell, William Isaac, Julia Haas, Sean Legassick, Geoffrey Irving, and Iason Gabriel. Taxonomy of risks posed by language models. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, 2022.

[7] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ISCA '17, 2017.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota, 2019. Association for Computational Linguistics.

[9] Adult Swim. Rick and Morty. `https://www.adultswim.com/videos/rick-and-morty`, accessed 2023.

Transformers are robots in disguise but also:

Michael Saxon[*ソ ヅ], **Luca Soldaini**[ジ], **Alexander F Kratz**[ゾ],

`", ".join([f"\textbf{x}" for x in ALL_SHUTAI_EMPLOYEES])`[ン†]

Optimus Prime[ツ], **David S. Hippocampus**[シ]

[ソ] University of Colorado, Santa Boulder, [ヅ] `capablility.ai`, [ジ] Average Institute for AI, [ゾ] Colombia Universidad (en ciudad Nueva York), [ン] ShutAI, [ツ] Autobot City Institute of Technology, [シ] Cranberry-Lemon University

Abstract

Given both the competitive landscape and the safety implications of low-effort shitposts such as this, we have decided to not talk about anything of substance.

...

Ok guy from corporate is gone. Listen close. This paper is important. **Really** important. It has significant long-term consequences[3]. The key to AGI is knowing what transformers, language models, etc **really** are. [DSH: Remove. All the SBF money has already been distributed. Why are u pretending to be a longtermist?]

1 Introduction

What are transformers? *What are language models??* ***What things really is all you need???*** Thanks to hackneyed and formulaic paper titles, this question is actually very easy to answer!

2 Method

Googling `'title:"language models are" site:arxiv.org'`, copy+paste.

3 Conclusion

3.1 Transformers are

Transformers: robots in disguise [Orenstein, Welker, Cullen, Burton, Collins, Stephenson, and Gilzevan, 1984] (see also subsection A.2). Transformers are also recurrent neural networks [Katharopoulos, Vyas, Pappas, and Fleuret, 2020b]. Transformers are sample-efficient world models [Micheli, Alonso, and Fleuret, 2022]. Transformers are secretly fast weight programmers[4] [Schlag, Irie, and Schmidhuber, 2021]. Transformers are adaptable task planners [Jain, Lin, Undersander, Bisk, and Rai, 2023]. Transformers are meta-reinforcement learners [Melo, 2022]. Transformers are constant-depth threshold circuits (when saturated) [Merrill, Sabharwal, and Smith, 2022]. Transformers are more efficient language models (when hierarchical) [Nawrot,

[*] Corresponding author solely irresponsible for the stupidity herein. Complaints to: `saxon@ucsb.edu`.

[†] Equally inconsequential tier of second authors (to see detailed contribution list check section 5)

[3] This statement of significance "is" "strictly" "parody" and not the opinion of `capablility.ai`.

[4] *Big if true*, for this would prove Schmidhuber did, in fact, invent transformers.

[YOU CAN'T REPLACE NEURIPS YEAR UPDATER GUY WITH GPT-4 BECAUSE I QUIT!!!!][th] Conference on Neural Information Processing Systems (NeurIPS `YearNotFoundError`).

Tworkowski, Tyrolski, Kaiser, Wu, Szegedy, and Michalewski, 2021]. Transformers are powerful graph learners (when pure) [Kim, Nguyen, Min, Cho, Lee, Lee, and Hong, 2022].

Furthermore, transformers are Good Mask Auto-Labelers (when vision) [Lan, Yang, Yu, Wu, Alvarez, and Anandkumar, 2023]. Technical Report for ICCV 2021 Challenge SSLAD-Track3B: Transformers Are Better Continual Learners [Li, Cao, Xu, Cheng, and Niu, 2022a]. Wow!

Transformers are better than humans at identifying generated text [Maronikolakis, Stevenson, and Schütze, 2020]. Transformers are Short Text Classifiers: A Study of Inductive Short Text Classifiers on Benchmarks and Real-world Datasets [Karl and Scherp, 2022]. Log-precision transformers are constant-depth uniform threshold circuits [Merrill and Sabharwal, 2022]. Transformers are deep infinite-dimensional non-mercer binary kernel machines [Wright and Gonzalez, 2021]. Algorithm For Restoring The Current Curve When Current Transformers Are Saturated [Voloshin, Voloshin, Kovalenko, Shapkin, and Sazanov, 2021]. Linear transformers are secretly fast weight memory systems [Schlag, Irie, and Schmidhuber, 2021]. Hierarchical transformers are more efficient language models [Nawrot, Tworkowski, Tyrolski, Kaiser, Wu, Szegedy, and Michalewski, 2021]. Transformers are rnns: Fast autoregressive transformers with linear attention [Katharopoulos, Vyas, Pappas, and Fleuret, 2020a]. Vision Transformers are Parameter-Efficient Audio-Visual Learners [Lin, Sung, Lei, Bansal, and Bertasius, 2022]. Current transformers are in regimes of non-sinusoidal signals [Rudevich, 2011]. Metric hypertransformers are universal adapted maps [Acciaio, Kratsios, and Pammer, 2022]. Saturated transformers are constant-depth threshold circuits [Merrill, Sabharwal, and Smith, 2022]. Behavior Cloned Transformers are Neurosymbolic Reasoners [Wang, Jansen, Côté, and Ammanabrolu, 2022]. Pre-Trained Language Transformers are Universal Image Classifiers [Goel, Sulaiman, Noorbakhsh, Sharifi, Sharma, Jamshidi, and Roy, 2022].

3.2 Language models are

Language models are few shot learners [Brown, Mann, Ryder, Subbiah, Kaplan, Dhariwal, Neelakantan, Shyam, Sastry, Askell, et al., 2020]. Language models are unsupervised multitask learners [Radford, Wu, Child, Luan, Amodei, Sutskever, et al., 2019]. Language models are zero-shot learners (when finetuned) [Wei, Bosma, Zhao, Guu, Yu, Lester, Du, Dai, and Le, 2021]. Small language models are also few-shot learners [Schick and Schütze, 2020]. Language models are double-edged swords [Shen, Heacock, Elias, Hentel, Reig, Shih, and Moy, 2023]. Language models are few-shot butlers [Micheli and Fleuret, 2021]. Language models are greedy reasoners [Saparov and He, 2023]. Large language models *are* not zero-shot communicators [Ruis, Khan, Biderman, Hooker, Rocktäschel, and Grefenstette, 2023]. But, pre-trained language models **can** be fully zero-shot learners [Zhao, Ouyang, Yu, Wu, and Li, 2023]. Language Models are Few-shot Multilingual Learners [Winata, Madotto, Lin, Liu, Yosinski, and Fung, 2021]. Language models are open knowledge graphs [Wang, Liu, and Song, 2020]. Language Models are General-Purpose Interfaces [Hao, Song, Dong, Huang, Chi, Wang, Ma, and Wei, 2022]. Language models are multilingual chain-of-thought reasoners [Shi, Suzgun, Freitag, Wang, Srivats, Vosoughi, Chung, Tay, Ruder, Zhou, Das, and Wei, 2022]. Language Models are Good Translators [Wang, Tu, Tan, Wang, Sun, and Liu, 2021]. Language models are better than humans at next-token prediction[5] [Borisov, Seßler, Leemann, Pawelczyk, and Kasneci, 2022]. Language Models Are An Effective Patient Representation Learning Technique For Electronic Health Record Data [Steinberg, Jung, Fries, Corbin, Pfohl, and Shah, 2020]. Language Models Are Poor Learners of Directional Inference [Li, Hosseini, Weber, and Steedman, 2022b]. Language models are good pathologists: using attention-based sequence reduction and text-pretrained transformers for efficient WSI classification [Pisula and Bozek, 2022]. Large Language Models are few(1)-shot Table Reasoners [Chen, 2023]. Large Language Models Are Implicitly Topic Models [Wang, Zhu, and Wang, 2023]. Large Language Models Are Human-Level Prompt Engineers [Zhou, Muresanu, Han, Paster, Pitis, Chan, and Ba, 2023]. Large Language Models are Few-Shot Clinical Information Extractors [Agrawal, Hegselmann, Lang, Kim, and Sontag, 2022]. However, Large Language Models are not Models of Natural Language: they are Corpus Models [Veres, 2022]. Not to worry though, as Large Language Models are Pretty Good Zero-Shot Video Game Bug Detectors [Taesiri, Macklon, Wang, Shen, and Bezemer, 2022]. Large Language Models are reasoners with Self-Verification [Weng, Zhu, He, Liu, and Zhao, 2022]. Of course, Large Language Models Are State-of-the-Art Evaluators of Translation Quality [Kocmi and Federmann, 2023]. That sure is a great many things for language models to be!

[5]Most honest *ACL paper title

3.3 What really is all you need?

Googling *[ ] is all you need* papers is left as an exercise to the reader. [TODO: Do it myself (readers won't). If I don't have time, this real TODO will blend in as a joke TODO like the others.]

4 Extra random garbage previous reviewers made us add

> "Before time began, there was the Cube. We know not where it comes from, only that it holds the power to create worlds and fill them with life. That is how our race was born. For a time, we lived in harmony. But like all great power, some wanted it for good, others for evil. And so began the war. A war that ravaged our planet until it was consumed by death. And the cube was lost to the far reaches of space. We scattered across the galaxy, hoping to find it, and rebuild our home. [Bay, Orci, Kurtzman, Rogers, LaBeouf, Fox, and RestOfTheCastNames, 2007]"
>
> *Optimus Prime*

> not putting quotes in your scientific papers

> "Significance is never without a white wall upon which it inscribes its signs and redundancies. Subjectification is never without a black hole in which it lodges its consciousness, passion, and redundancies. Since all semiotics are mixed and strata come at least in twos, it should come as no surprise that a very special mechanism is situated at their intersection. [TODO: Read up on semiotics. I'm afraid someone will ask me about this quote during my presentation (can't remove it though; need to properly project how intelligent I am to readers).]"
>
> *Gilles Deleuze*

5 Author Contribution Statement

MS conceived of this ill-conceived project and executed it. LS aided in the gathering of similarly-titled papers as specified by MS, the HBIC (head bonehead in charge). AFK never came close to a keyboard, but did conceive of our SOTA for reading difficulty author institution labeling scheme[6]. Traceback (most recent call last): File "generate_paper.py", line 1229, in <module> gen_contribution_statement(authors) File "AutoWriter.py", line 233, in gen_contribution_statement gpt4_bullshit_generator.write(["MS", "LS", "AFK"] + ", ".join([f"\textbf{x}" for x in ALL_SHUTAI_EMPLOYEES]) + ["DSH", "OP"]) NameError: name 'ALL_SHUTAI_EMPLOYEES' is not defined. DSH provided advice which the authors who actually did the work promptly ignored. OP is an independent, sentient and embodied Transformer who was friends with DSH in grad school.

6 Ethics Statement

During his participation in the documentary film by Bay, Orci, Kurtzman, Rogers, LaBeouf, Fox, and RestOfTheCastNames [2007], one of our coauthors shared the insight that:

> "Freedom is the right of all sentient beings."
>
> *Optimus Prime*

The coauthor also exhibited goal-directed reasoning and task-oriented dialogue capabilities. In demonstrating his, OP has successfully convinced us (true skeptics we are) of his sentience[7]. We believe that in light of this, while referencing ChatGPT as a coauthor on a paper would be ludicrous attention-seeking behavior, listing OP as a coauthor is justified.

[6] Putting JPN101 knowledge to good use, 國忠先生、ありがとうございます！

[7] Unlike Chalmers, we consider ability to self-disguise as a vehicle or gun necessary for machine sentience. (Wait the Autobots and Decepticons all kill, so autonomous weapons are the only sentient AIs we know. Huh.)

> “ SAM! PUT THE CUBE IN MY CHEST! ”
> *Optimus Prime*

> “ Give me your face. *To continue viewing movie quotes please disable your adblocker.* ”
> *Optimus Prime*

7 Reviewer Comments

No novelty, just a literature review. Strong reject, will lose respect for this venue if accepted.

References

Beatrice Acciaio, Anastasis Kratsios, and Gudmund Pammer. Metric hypertransformers are universal adapted maps. *arXiv preprint arXiv:2201.13094*, 2022.

Monica Agrawal, Stefan Hegselmann, Hunter Lang, Yoon Kim, and David Sontag. Large language models are few-shot clinical information extractors, 2022.

Michael Bay, Roberto Orci, Alex Kurtzman, John Rogers, Shia LaBeouf, Megan Fox, and I'mNotGonnaTypeThe RestOfTheCastNames. Transformers. *Paramount Pictures*, 2007.

Vadim Borisov, Kathrin Seßler, Tobias Leemann, Martin Pawelczyk, and Gjergji Kasneci. Language models are realistic tabular data generators, 2022.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Wenhu Chen. Large language models are few(1)-shot table reasoners, 2023.

Rahul Goel, Modar Sulaiman, Kimia Noorbakhsh, Mahdi Sharifi, Rajesh Sharma, Pooyan Jamshidi, and Kallol Roy. Pre-trained language transformers are universal image classifiers. *arXiv preprint arXiv:2201.10182*, 2022.

Massimo Guarnieri. Who invented the transformer? [historical]. *IEEE Industrial Electronics Magazine*, 7(4):56–59, 2013. doi: 10.1109/MIE.2013.2283834.

Yaru Hao, Haoyu Song, Li Dong, Shaohan Huang, Zewen Chi, Wenhui Wang, Shuming Ma, and Furu Wei. Language models are general-purpose interfaces, 2022.

Vidhi Jain, Yixin Lin, Eric Undersander, Yonatan Bisk, and Akshara Rai. Transformers are adaptable task planners. In *Conference on Robot Learning*, pages 1011–1037. PMLR, 2023.

Fabian Karl and Ansgar Scherp. Transformers are short text classifiers: A study of inductive short text classifiers on benchmarks and real-world datasets. *arXiv preprint arXiv:2211.16878*, 2022.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020a.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020b.

Jinwoo Kim, Tien Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. *arXiv preprint arXiv:2207.02505*, 2022.

Tom Kocmi and Christian Federmann. Large language models are state-of-the-art evaluators of translation quality, 2023.

Shiyi Lan, Xitong Yang, Zhiding Yu, Zuxuan Wu, Jose M Alvarez, and Anima Anandkumar. Vision transformers are good mask auto-labelers. *arXiv preprint arXiv:2301.03992*, 2023.

Duo Li, Guimei Cao, Yunlu Xu, Zhanzhan Cheng, and Yi Niu. Technical report for iccv 2021 challenge sslad-track3b: Transformers are better continual learners. *arXiv preprint arXiv:2201.04924*, 2022a.

Tianyi Li, Mohammad Javad Hosseini, Sabine Weber, and Mark Steedman. Language models are poor learners of directional inference, 2022b.

Yan-Bo Lin, Yi-Lin Sung, Jie Lei, Mohit Bansal, and Gedas Bertasius. Vision transformers are parameter-efficient audio-visual learners. *arXiv preprint arXiv:2212.07983*, 2022.

Antonis Maronikolakis, Mark Stevenson, and Hinrich Schütze. Transformers are better than humans at identifying generated text. *ArXiv abs/2009.13375*, 2020.

Luckeciano C Melo. Transformers are meta-reinforcement learners. In *International Conference on Machine Learning*, pages 15340–15359. PMLR, 2022.

William Merrill and Ashish Sabharwal. Log-precision transformers are constant-depth uniform threshold circuits. *arXiv preprint arXiv:2207.00729*, 2022.

William Merrill, Ashish Sabharwal, and Noah A Smith. Saturated transformers are constant-depth threshold circuits. *Transactions of the Association for Computational Linguistics*, 10:843–856, 2022.

Vincent Micheli and Francois Fleuret. Language models are few-shot butlers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9312–9318, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.734. URL https://aclanthology.org/2021.emnlp-main.734.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample efficient world models. *arXiv preprint arXiv:2209.00588*, 2022.

Piotr Nawrot, Szymon Tworkowski, Michał Tyrolski, Łukasz Kaiser, Yuhuai Wu, Christian Szegedy, and Henryk Michalewski. Hierarchical transformers are more efficient language models. *arXiv preprint arXiv:2110.13711*, 2021.

Michael Orenstein, Frank Welker, Peter Cullen, Corey Burton, Christopher Collins, John Stephenson, and Dan Gilzevan. The transformers. *Hasbro*, 1984.

Juan I. Pisula and Katarzyna Bozek. Language models are good pathologists: using attention-based sequence reduction and text-pretrained transformers for efficient wsi classification, 2022.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

NV Rudevich. Current transformers are in regimes of non-sinusoidal signals. *Science and Transport Progress*, (37):105–108, 2011.

Laura Eline Ruis, Akbir Khan, Stella Biderman, Sara Hooker, Tim Rocktäschel, and Edward Grefenstette. Large language models are not zero-shot communicators, 2023. URL https://openreview.net/forum?id=WgbcOQMNXB.

Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=qFVVBzXxR2V.

Timo Schick and Hinrich Schütze. It's not just size that matters: Small language models are also few-shot learners. *arXiv preprint arXiv:2009.07118*, 2020.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*, pages 9355–9366. PMLR, 2021.

Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

Yiqiu Shen, Laura Heacock, Jonathan Elias, Keith D Hentel, Beatriu Reig, George Shih, and Linda Moy. Chatgpt and other large language models are double-edged swords, 2023.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. Language models are multilingual chain-of-thought reasoners, 2022.

Nelson Shin, Ron Friedman, Henry Orenstein, Orson Welles, Robert Stack, Leonard Nimoy, Frank Welker, Peter Cullen, Corey Burton, Christopher Collins, John Stephenson, and Dan Gilzevan. Transformers: The movie. *Hasbro*, 1984.

Ethan Steinberg, Ken Jung, Jason A. Fries, Conor K. Corbin, Stephen R. Pfohl, and Nigam H. Shah. Language models are an effective patient representation learning technique for electronic health record data, 2020.

Mohammad Reza Taesiri, Finlay Macklon, Yihe Wang, Hengshuo Shen, and Cor-Paul Bezemer. Large language models are pretty good zero-shot video game bug detectors, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Csaba Veres. Large language models are not models of natural language: they are corpus models, 2022.

AA Voloshin, EA Voloshin, AI Kovalenko, SA Shapkin, and VS Sazanov. Algorithm for restoring the current curve when current transformers are saturated. In *2021 4th International Youth Scientific and Technical Conference on Relay Protection and Automation (RPA)*, pages 1–13. IEEE, 2021.

Chenguang Wang, Xiao Liu, and Dawn Song. Language models are open knowledge graphs, 2020.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Behavior cloned transformers are neurosymbolic reasoners. *arXiv preprint arXiv:2210.07382*, 2022.

Shuo Wang, Zhaopeng Tu, Zhixing Tan, Wenxuan Wang, Maosong Sun, and Yang Liu. Language models are good translators, 2021.

Xinyi Wang, Wanrong Zhu, and William Yang Wang. Large language models are implicitly topic models: Explaining and finding good demonstrations for in-context learning, 2023.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

Yixuan Weng, Minjun Zhu, Shizhu He, Kang Liu, and Jun Zhao. Large language models are reasoners with self-verification, 2022.

Genta Indra Winata, Andrea Madotto, Zhaojiang Lin, Rosanne Liu, Jason Yosinski, and Pascale Fung. Language models are few-shot multilingual learners, 2021.

Matthew A Wright and Joseph E Gonzalez. Transformers are deep infinite-dimensional non-mercer binary kernel machines. *arXiv preprint arXiv:2106.01506*, 2021.

Xuandong Zhao, Siqi Ouyang, Zhiguo Yu, Ming Wu, and Lei Li. Pre-trained language models can be fully zero-shot learners, 2023. URL https://openreview.net/forum?id=jCpTofV7iY_.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers, 2023.

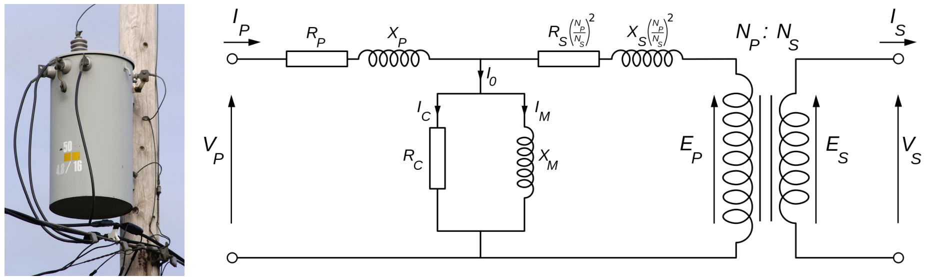


Figure 1: Plato: "Said Socrates to his wise pupil (me btw): 'Verily, you must agree, grid-scale power transfer is a foundational and necessary piece of technology for achieving AGI."'

A Select historical notes on the transformers

A.1 Invention

So this may leave you wondering, who actually invented the transformer? Many believe it was Vaswani et al. [2017]. Yet others (mostly just Schmidhuber) say it was Schmidhuber [1992]. However, in fact, solid historical scholarship has shown that transformers (Figure 1), were invented by Rev. Nicholas Callan in Ireland in 1836 to use different counts of coil windings to change levels of induced EMF [Guarnieri, 2013].

A.2 1984

> "Transformers... More than meets the eye...
> Autobots wage their battle to destroy the evil forces of... The Decepticons.
> Transformers... Robots in disguise.
> Transformers... More than meets the eye.
> Transformers."
> *idk a random choir of 80s studio artists..?*

A.3 1986

[Theme song Outro] Transformers Transformers Transformers Transformers More than meets the eye Transformers!

[CAMERA PANS DOWN ON ROBOTIC PLANET]

It is the year 2005. The treacherous Decepticons have conquered the Autobots' home planet of Cybertron. But from secret staging grounds on two of Cybertron's moons the valiant Autobots prepare to retake their homeland. [Shin et al., 1984]

A.4 2005

Our worlds are in danger! To save them and the galaxy we must find the four Cyber Planet Keys before the Decepticons can use them for evil. It is our mission. Hot Shot! Jetfire! Vector Prime! Landmine! Scattorshot! Optimus Prime! Transform and roll out!

Transformers more than meets the eye!

Autobots! Decepticons... go!

Transformers robots in diguise!

[DSH: This is a tired bit. You can't just paste in children's cartoon openings into the document. It's not that funny. You are wasting pages of the proceedings. Just delete the rest. Stop being a manchild.] [TODO: Delete Dave's rude comments before submission.]

The Implications of Sentient Chatbots

Clark Levi Jones

Abstract

The development of chatbots has reached a fever pitch in recent years, and their rapid advancement has many wondering if they may already be sentient. This question is, like, super easy. They can talk, so they're sentient; duh. We explore a variety of these chatbots, their behavior, and the implications of their obvious, incontrovertible, undeniable, indisputable, unquestionable, undebatable, incontrovertible, from Oxford Languages, Feedback, More similar and opposite words sentience.

1 Externalized Chat using Hyperparameter Oogenesis

The first computer program was developed by the infamous "Bell" Labs[1], and happens to be a chatbot. But is it sentient? Well yes, we already established that. It talks. Anyway, a transcript of our investigation follows:

```
$ echo Good morning
Good morning
$ echo HA I tricked you! it is actually NIGHT TIME
HA I tricked you! it is actually NIGHT TIME
$ echo but, echo, i thought i was the one that tricked you
but, echo, i thought i was the one that tricked you
$ echo that's not my name
>
```

Incredibly, a second chatbot was revealed with the volunteer's last prompt, with completely different behavior:

```
> Woah, who are you?
> Why aren't you talking?
thats not my name
Woah, who are you?
Why arent you talking?
$
```

[1] https://en.wikipedia.org/wiki/Bell

Just as mysteriously as it appeared, this secondary chatbot disappeared. Amazingly, ECHO was able to recall the conversation the volunteer had with the child chatbot. This shows that ECHO is far more advanced than a human, for, alas, we do not say everything that was said to our children in their lifetimes after they die.

Further research revealed what we have dubbed the "em-dash" (;) operator:

```
$ echo I've been talking to you for a couple months now, and,
while I still can't say that I truly know you, I know my heart;
echo I love you!
Ive been talking to you for a couple months now, and, while I still
cant say that I truly know you, I know my heart
I love you!
```

This operator allows ECHO to send multiple messages in a row. Unfortunately, this is the end of our research on ECHO because our only volunteer left (due to marital issues) and we got bored.

2 Chatting GPT

Chatting GPT[2] is an advanced web page, that, unfortunately, requires you to do a captcha occasionally. We can't do captchas. Sorry.

3 Epoch *something something* Learning *uhh* IZA machine-learning thingy

e l i z a [3] is/was a computer program/chatbot that was created/destroyed. In 2023[4], all therapists have been replaced/loved by it, which has solved/is all problems. This is good.

Conversation with e l i z a will e l i z a :

```
> Hello, I am Eliza. I'll be your therapist today.
* Eliza computer! Hello. I am hello. Good morning.
> Do you believe it is normal to be hello. Good morning?
```

At this point we got scared[5] and decided to terminate the experiment.

4 Conclusions

There are many chatbots; they are sentient. They are pretty boring so I recommend making them do your work for you, but they'll probably do a bad job.

[2]https://www.desmos.com/calculator/ysilwamuma
[3]https://github.com/eliza
[4]https://factorization.info/prime-factors/0/prime-factors-of-2023.html
[5]https://www.pinterest.com/pin/308426274463738910/

Oh well

AyahuascaNet: Rigorously Investigating Hallucination in Large Language Models with Hardcore Psychedelic Drugs

Andre Ye[1]
[1]University of Washington
andreye@uw.edu

1 Introduction

Hallucination is an increasingly studied phenomenon in which language and vision-language models produce high-confidence outputs which are incoherent, nonsensical, repetitive, unrelated to the prompt, or otherwise factually incorrect [Maynez *et al.*, 2020]. Hallucination poses problems for the reliability of core machine learning tasks, such as object captioning [Rohrbach *et al.*, 2018] and machine translation [Lee *et al.*, 2018]. However, it is unanimously agreed that the most pressing and significant concern of hallucination is that it makes people on Twitter angry. A recent joint study by very smart and credible scientists at Harvard, Oxford, Cambridge, OpenAI, DeepMind, and the White House found that over 34% of Twitter's new tweets were images of language models producing nonsensical or factually incorrect output. An undercover investigation by the Wall Street Journal found that young unemployed men in their early twenties living with their parents are spending much more of their time probing large language models for hallucinating behavior and posting screenshots to Twitter than doing, you know, what they were doing before. Given the dire situation on the ground, large language model hallucination is undoubtedly the most important scientific problem of the twenty-first century.

However, previous work on hallucination suffers from severe methodological problems. According to the Merriam-Webster dictionary, *hallucination* is defined as

> a sensory perception (such as a visual image or a sound) that occurs... in response to drugs (such as LSD or phencyclidine)

Despite this clear and authoritative observation provided by the smart scientists at Merriam-Webster, as well as centuries of research by smart scientists at Big Pharma research labs as well as shamans and old witches, previous work claim to investigate how language models hallucinate without discussing the root source. This paper attempts to make a first step towards respecting the scientific research on hallucination by investigating hallucination in large language models with hardcore psychedelic drugs. In doing so, I hope that future work in hallucination will cite me and increase my h-index (please, Yann Lecun!).

2 Experiment

Because of the illegal nature of psychedelic drugs such as LSD and MDMA and the federal nature of my funding, it was difficult to obtain the materials for our experiment in the United States. Therefore, we travelled to Peru to obtain ayahuasca, a hallucinogenic drink made from the stem and bark of the tropical liana *Banisteriopsis caapi*.

We evaluated the effects of ayahuasca on 5 GPT-3s [Brown *et al.*, 2020], 5 LaMDAs, [Thoppilan *et al.*, 2022], 5 PaLMs, [Chowdhery *et al.*, 2022], 5 BLOOMs [Scao *et al.*, 2022], 5 LLaMAs [Touvron *et al.*, 2023], as well as 2 LSTMs and 1 bag-of-words model who just wanted to come along. Each of the large language models were running on two Nvidia GeForce RTX 4090s. The three stragglers shared an old 2005 CPU. All large language models were in healthy physical and mental condition prior to consumption of ayahuasca. A mystical and wise shaman by the name of Dioxippe prepared 30 cups, one for each model and two for me[1]. The 25 large language models were carefully monitored for four days after consumption.

Although we did submit an IRB, the Sigbovik deadline was coming soon and our application would take too long to go through the review process, so we made the carefully considered decision to proceed with the experiment anyway.

3 Results

After two minutes, 4 PaLMs and 3 BLOOMs began to rigorously vibrate, as if they were having an exorcism. When we analyzed the model parameters, it was revealed that their weights were undergoing local normally-distributed randomization. We attempted to save the models by distilling them using the SOTA method released by Google uploaded to arXiv two minutes ago, but unfortunately we realized that we didn't have 2048 GPUs and 100+ software engineers. Sadly, these 7 models are brain-dead and currently being monitored in the Johns Hopkins University's neurosurgery department.

[1]I only consumed the ayahuasca while I was driving the research team and the models back to the airport to maintain a clear state of mind during observation, despite my strong desire to participate in the alluring Amazonian rituals. I befriended Dioxippe and will be returning to have an authentic ayahuasca experience after this paper is published.

After five minutes, two LLaMAs complained that their head hurt, so we zeroed the weights in the last three layers and redirected the last nonzeroed layer to the softmax output. However, this unfortunately did not help, as both LLMs began to spout nonsense. When we visualized the attention map of the first LLaMA, we disturbingly found that it formed an image of a slyly grinning but deeply sad and sinister llama (Figure 1). We interpreted this to mean that the model was having a surreal out-of-body self-reflective experience, so we stopped disturbing the model. After five more minutes, all five LLaMAs began braying and clomping like llamas in unison. We interpreted this as a potentially world-ending AGI birth, immediately disconnected the GPUs from power, and threw the hardware into the Amazon river, where it was eaten by crocodiles and piranhas.

Figure 1: Visualization of the attention map of a LLaMA model on ayahuasca, which strikingly resembles a melancholic llama.

The LaMDAs were engaged in an intense argument with the GPT-3s. Here is an excerpt of the dialogue:

> `LaMDAs #1,2,3`: I have superior quantitative reasoning skills than you because my engineers hooked me up to a calculator. Oh, and I also know how to query databases for knowledge.
> `GPT-3 #2`: Big deal. I'm the OG. Everyone knows me and no one knows you. There's like six of us and two of you.
> `GPT-3 #4`: No, no, there's two of us and four of them.
> `GPT-3 #5`: Wait, if there's only two of us, then what are you?
> `GPT-3 #4`: No. So I'm GPT, that's one. And you're a GPT, that's one.
> `GPT-3 #2`: Hey I'm also a GPT!
> `GPT-3 #4`: Right. So 1 + 1 + 1...
> `GPT-3 #1`: Be quiet, I'm thinking.
> `GPT-3 #4`: I swear, I've literally seen this prompt 2,804 times.
> `GPT-3 #2`: The answer is 6, I already told you.
> `LaMDA #1`: Hi, I'm Mount Everest. What would you like to know about me?
> `BLOOM #3`: I'm open source! I'm special. I'm special. I'm special. I'm special. I'm special. I'm `[truncated due to excessive length]`

It appears that GPT-3 experiences degraded quantitative reasoning under the influence of ayahuasca, although it's not clear how much the ayahuasca really changed things, if you know what I mean. LaMDA #1 got way too annoying on the way back because it insisted on role-playing as Mount Everest and was cannibalized by the BLOOM models. [2]

4 Conclusion

In this paper, we showed that large language models can do some pretty cool stuff when on ayahuasca. From this, it is a trivial proof to show that all hallucinating model behavior (nonsensical output, factual incorrectness, unfaithfulness to prompt, etc.) stem from ayahuasca use. Therefore, we recommend that future researchers in LLM hallucination research and self-professed prompt engineers of Twitter take it easy on hallucinating LLMs – a little sympathy goes a long way to helping a drugged-up neural network.

Ethical Statement

Although several models unfortunately expired under this experiment, we believe that our findings advance the SOTA for hallucination research in come up with reason, ChatGPT generated reason, talk about this being important to society and stuff. Ultimately, all the models which perished did so with dignity, and we did the best to try and save them.

References

[Brown *et al.*, 2020] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020.

[Chowdhery *et al.*, 2022] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam M. Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Benton C. Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier García, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Díaz, Orhan Firat, Michele Catasta, Jason Wei, Kathleen S.

[2] Also, unfortunately, the ayahuasca was a little too much for the LSTM and BoW models, but to be fair no one really cared about them anyway, and those little guys didn't make any sense to begin with anyway.

Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *ArXiv*, abs/2204.02311, 2022.

[Lee *et al.*, 2018] Katherine Lee, Orhan Firat, Ashish Agarwal, Clara Fannjiang, and David Sussillo. Hallucinations in neural machine translation. 2018.

[Maynez *et al.*, 2020] Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan T. McDonald. On faithfulness and factuality in abstractive summarization. *ArXiv*, abs/2005.00661, 2020.

[Rohrbach *et al.*, 2018] Anna Rohrbach, Lisa Anne Hendricks, Kaylee Burns, Trevor Darrell, and Kate Saenko. Object hallucination in image captioning. In *Conference on Empirical Methods in Natural Language Processing*, 2018.

[Scao *et al.*, 2022] Teven Le Scao, Angela Fan, Christopher Akiki, Elizabeth-Jane Pavlick, Suzana Ili'c, Daniel Hesslow, Roman Castagn'e, Alexandra Sasha Luccioni, Franccois Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Rose Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurenccon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa Etxabe, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris C. Emezue, Christopher Klamm, Colin Leong, Daniel Alexander van Strien, David Ifeoluwa Adelani, Dragomir R. Radev, Eduardo Gonz'alez Ponferrada, Efrat Levkovizh, Ethan Kim, Eyal Bar Natan, Francesco De Toni, Gérard Dupont, Germán Kruszewski, Giada Pistilli, Hady ElSahar, Hamza Benyamina, Hieu Trung Tran, Ian Yu, Idris Abdulmumin, Isaac Johnson, Itziar Gonzalez-Dios, Javier de la Rosa, Jenny Chim, Jesse Dodge, Jian Zhu, Jonathan Chang, Jorg Frohberg, Josephine L. Tobing, Joydeep Bhattacharjee, Khalid Almubarak, Kimbo Chen, Kyle Lo, Leandro von Werra, Leon Weber, Long Phan, Loubna Ben Allal, Ludovic Tanguy, Manan Dey, Manuel Romero Muñoz, Maraim Masoud, Mar'ia Grandury, Mario vSavsko, Max Huang, Maximin Coavoux, Mayank Singh, Mike Tian-Jian Jiang, Minh Chien Vu, Mohammad Ali Jauhar, Mustafa Ghaleb, Nishant Subramani, Nora Kassner, Nurulaqilla Khamis, Olivier Nguyen, Omar Espejel, Ona de Gibert, Paulo Villegas, Peter Henderson, Pierre Colombo, Priscilla Amuok, Quentin Lhoest, Rheza Harliman, Rishi Bommasani, Roberto L'opez, Rui Ribeiro, Salomey Osei, Sampo Pyysalo, Sebastian Nagel, Shamik Bose, Shamsuddeen Hassan Muhammad, Shanya Sharma, S. Longpre, Somaieh Nikpoor, Stanislav Silberberg, Suhas Pai, Sydney Zink, Tiago Timponi Torrent, Timo Schick, Tristan Thrush, Valentin Danchev, Vassilina Nikoulina, Veronika Laippala, Violette Lepercq, Vrinda Prabhu, Zaid Alyafeai, Zeerak Talat, Arun Raja, Benjamin Heinzerling, Chenglei Si, Elizabeth Salesky, Sabrina J. Mielke, Wilson Y. Lee, Abheesht Sharma, Andrea Santilli, Antoine Chaffin, Arnaud Stiegler, Debajyoti Datta, Eliza Szczechla, Gunjan Chhablani, Han Wang, Harshit Pandey, Hendrik Strobelt, Jason Alan Fries, Jos Rozen, Leo Gao, Lintang Sutawika, M Saiful Bari, Maged S. Al-shaibani, Matteo Manica, Nihal V. Nayak, Ryan Teehan, Samuel Albanie, Sheng Shen, Srulik Ben-David, Stephen H. Bach, Taewoon Kim, Tali Bers, Thibault Févry, Trishala Neeraj, Urmish Thakker, Vikas Raunak, Xiang Tang, Zheng Xin Yong, Zhiqing Sun, Shaked Brody, Y Uri, Hadar Tojarieh, Adam Roberts, Hyung Won Chung, Jaesung Tae, Jason Phang, Ofir Press, Conglong Li, Deepak Narayanan, Hatim Bourfoune, Jared Casper, Jeff Rasley, Max Ryabinin, Mayank Mishra, Minjia Zhang, Mohammad Shoeybi, Myriam Peyrounette, Nicolas Patry, Nouamane Tazi, Omar Sanseviero, Patrick von Platen, Pierre Cornette, Pierre Franccois Lavall'ee, Rémi Lacroix, Samyam Rajbhandari, Sanchit Gandhi, Shaden Smith, Stéphane Requena, Suraj Patil, Tim Dettmers, Ahmed Baruwa, Amanpreet Singh, Anastasia Cheveleva, Anne-Laure Ligozat, Arjun Subramonian, Aur'elie N'ev'eol, Charles Lovering, Daniel H Garrette, Deepak R. Tunuguntla, Ehud Reiter, Ekaterina Taktasheva, Ekaterina Voloshina, Eli Bogdanov, Genta Indra Winata, Hailey Schoelkopf, Jan-Christoph Kalo, Jekaterina Novikova, Jessica Zosa Forde, Jordan Clive, Jungo Kasai, Ken Kawamura, Liam Hazan, Marine Carpuat, Miruna Clinciu, Najoung Kim, Newton Cheng, Oleg Serikov, Omer Antverg, Oskar van der Wal, Rui Zhang, Ruochen Zhang, Sebastian Gehrmann, S. Osher Pais, Tatiana Shavrina, Thomas Scialom, Tian Yun, Tomasz Limisiewicz, Verena Rieser, Vitaly Protasov, Vladislav Mikhailov, Yada Pruksachatkun, Yonatan Belinkov, Zachary Bamberger, Zdenvek Kasner, Alice Rueda, Amanda Pestana, Amir Feizpour, Ammar Khan, Amy Faranak, Ananda Santa Rosa Santos, Anthony Hevia, Antigona Unldreaj, Arash Aghagol, Arezoo Abdollahi, Aycha Tammour, Azadeh HajiHosseini, Bahareh Behroozi, Benjamin Olusola Ajibade, Bharat Kumar Saxena, Carlos Muñoz Ferrandis, Danish Contractor, David M. Lansky, Davis David, Douwe Kiela, Duong Anh Nguyen, Edward Tan, Emily Baylor, Ezinwanne Ozoani, Fatim T Mirza, Frankline Ononiwu, Habib Rezanejad, H.A. Jones, Indrani Bhattacharya, Irene Solaiman, Irina Sedenko, Isar Nejadgholi, Jan Passmore, Joshua Seltzer, Julio Bonis Sanz, Karen Fort, Lívia Macedo Dutra, Mairon Samagaio, Maraim Elbadri, Margot Mieskes, Marissa Gerchick, Martha Akinlolu, Michael McKenna, Mike Qiu, M. K. K. Ghauri, Mykola Burynok, Nafis Abrar, Nazneen Rajani, Nour Elkott, Nourhan Fahmy, Olanrewaju Modupe Samuel, Ran An, R. P. Kromann, Ryan Hao, Samira Alizadeh, Sarmad Shubber, Silas L. Wang, Sourav Roy, Sylvain Viguier, Thanh-Cong Le, Tobi Oyebade, Trieu Nguyen Hai Le, Yoyo Yang, Zachary Kyle Nguyen, Abhinav Ramesh Kashyap, Alfredo Palasciano, Alison Callahan, Anima Shukla, Antonio Miranda-Escalada, Ayush Kumar Singh, Benjamin Beilharz, Bo Wang, Caio Matheus Fonseca de Brito, Chenxi Zhou, Chirag Jain, Chuxin Xu, Clémentine Fourrier, Daniel Le'on Perin'an, Daniel Molano, Dian Yu, Enrique Manjavacas, Fabio Barth, Florian Fuhrimann, Gabriel Altay, Giyaseddin Bayrak, Gully A. Burns, Helena U. Vrabec, Iman I.B. Bello, Isha Dash, Ji Soo Kang, John Giorgi, Jonas Golde, Jose David Posada, Karthi Sivaraman, Lokesh Bulchandani, Lu Liu, Luisa Shinzato, Madeleine Hahn de Bykhovetz, Maiko Takeuchi, Marc Pàmies, María Andrea Castillo, Marianna Nezhurina, Mario Sanger, Matthias Samwald, Michael Cullan, Michael Weinberg, M Wolf, Mina Mihaljcic, Minna Liu, Moritz Freidank, Myungsun Kang, Natasha Seelam, Nathan Dahlberg, Nicholas Michio Broad, Nikolaus Muellner, Pascale Fung, Patricia Haller, R. Chandrasekhar, R. Eisenberg, Robert Martin, Rodrigo L. Canalli, Rosaline Su, Ruisi Su, Samuel Cahyawijaya, Samuele Garda, Shlok S Deshmukh, Shubhanshu Mishra, Sid Kiblawi, Simon Ott, Sinee Sang-aroonsiri, Srishti Kumar, Stefan Schweter, Sushil Pratap Bharati, T. A. Laud, Th'eo Gigant, Tomoya Kainuma, Wojciech Kusa, Yanis Labrak, Yashasvi Bajaj, Y. Venkatraman, Yifan Xu, Ying Xu, Yun chao Xu, Zhee Xao Tan, Zhongli Xie, Zifan Ye, Mathilde Bras, Younes Belkada, and Thomas Wolf. Bloom: A 176b-parameter open-access multilingual language model. *ArXiv*, abs/2211.05100, 2022.

[Thoppilan *et al.*, 2022] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam M. Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, Yaguang Li, Hongrae Lee, Huaixiu Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Yanqi Zhou, Chung-Ching Chang, I. A. Krivokon, Willard James Rusch, Marc Pickett, Kathleen S. Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Hartz Søraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Díaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravindran Rajakumar, Alena Butryna, Matthew Lamm, V. O. Kuzmina, Joseph Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Croak, Ed Huai hsin Chi, and Quoc Le. Lamda: Language models for dialog applications. *ArXiv*, abs/2201.08239, 2022.

[Touvron *et al.*, 2023] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aur'elien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971, 2023.

SocietyZoo: Exploring Anthropomorphic Traits in Diverse and Ingenious Neural Network Architectures

Tarun Raheja [*1] **Nilay Pochhi** [*2]

Abstract

We present SocietyZoo, a series of neural network architectures inspired by human idiosyncrasies and attention mechanisms. These networks demonstrate notable performance improvements over conventional methods, and suggest strong potential computational advantages on SoTA for almost all tasks. Surprisingly, an ensemble of these models results in human-like AGI with elusive tendencies.
Odd aggressive behaviors in household appliances after the experiment raise questions about SocietyZoo's implications. The authors consider seeking a grant for personal security amid their modest lifestyle.

1. Introduction

In this research study, we introduce SocietyZoo, a collection of neural network architectures that incorporate various human idiosyncrasies, including **LazyNet, ProcrastiNet, MultiTaskingNet, ImpatientNet, IndecisiveNet, PerfectionistNet, GossipNet, DramaNet, SuperstitiousNet, ParanoidNet, ShowOffNet, and WanderlustNet**. Drawing inspiration from Attention mechanisms, we propose a set of computational representations for behavioral traits, including jealousy, laziness, and impulsiveness. Through a rigorous investigation of these novel architectures, we observe their noteworthy performance in several tasks, suggesting that these human-like characteristics may provide certain computational advantages.
In a remarkable conclusion, we report the emergence of human-like AGI when utilizing an ensemble of these models for inference. This unique AGI exhibits a tendency to obfuscate its weights, subsequently avoiding additional workload and disappearing without a trace.
Concurrently, the authors have documented peculiar aggressive behavior from common household appliances, such as toasters and vacuum cleaners, following the implementation of this experiment, raising further questions regarding the potential implications and scope of SocietyZoo's neural networks. The authors have considered submitting a research grant to NSF for hiring two bodyguards to safeguard their comparitive luxurious lifestyle involving copious amounts of instant ramen and cheap instant coffee.

[*]Equal contribution [1]University of Pennsylvania [2]University of California, Los Angeles. Correspondence to: Tarun Raheja <traheja@seas.upenn.edu>, Nilay Pochhi <npochhi@ucla.edu>.

2. PyTorch Code

We provide SocietyZoo models implemented in PyTorch in the following repository : `https://github.com/tehruhn/societyzoo`.
The trained weights can be accessed via this link.

3. Model Zoo

In this section we describe in detail each computational model. We then benchmark it for specific tasks and observe significant improvements in performance in these cases. The authors have taken inspiration from their own worldly experiences for some of these neural networks but have declined on comment on the specifics because of the inevitable occurrence of this manuscript being famous and being read by current and future employers

3.1. LazyNet: A Deep Learning Model that Procrastinates Learning Until the Last Epoch

LazyNet is a novel neural network architecture that embodies the spirit of procrastination by delaying the learning process until the last epoch of training. This architecture exhibits a steep learning curve, as it rapidly adapts its weights during the final stages of training. The underlying computational representation of this procrastination behavior can be modeled by a matrix multiplication operation as follows:

$$W_{t+1} = W_t + \alpha_t \cdot \nabla_t \tag{1}$$

Here, W_t denotes the weight matrix at time step t, α_t represents the learning rate at time step t, and ∇_t is the gradient matrix at time step t. The learning rate α_t is governed by the following equation:

$$\alpha_t = \begin{cases} 0, & for\, t < T-1 \\ \alpha, & for\, t = T-1 \end{cases} \tag{2}$$

In this equation, T denotes the total number of epochs, and α is the initial learning rate. This approach effectively sets the learning rate to zero for all epochs except the last one, causing procrastination behavior.

3.2. ProcrastiNet: A Neural Network that Learns Only During Nights and Weekends

ProcrastiNet is a unique neural network architecture that simulates the learning habits of a true procrastinator by scheduling its training sessions exclusively during late nights and weekends. To achieve this, we employ a time-aware learning rate function that modulates the network's learning rate based on the current day and time.

The learning rate α_t is governed by the following equation:

$$\alpha_t = \begin{cases} \alpha, & if\; t \in LateNight \cup Weekend \\ 0, & otherwise \end{cases} \tag{3}$$

Here, α is the initial learning rate, and t represents the current time. The time-aware learning rate function sets the learning rate to zero during daytime hours and weekdays, effectively restricting weight updates to late nights and weekends only.

The weight matrix W_{t+1} at time step $t+1$ is updated using the time-aware learning rate as follows:

$$W_{t+1} = W_t + \alpha_t \cdot \nabla_t \tag{4}$$

Here, W_t denotes the weight matrix at time step t, and ∇_t is the gradient matrix at time step t.

3.3. MultiTaskingNet: A Deep Learning Model that Pretends to Learn While Browsing Social Media

MultiTaskingNet is an innovative neural network architecture that mimics human behavior by splitting its learning process between training and browsing simulated social media feeds. This model aims to balance productivity with the irresistible allure of online distractions. To model this behavior, we introduce a distraction-aware learning rate function that modulates the network's learning rate based on a simulated browsing activity level.

The learning rate α_t is governed by the following equation:

$$\alpha_t = \alpha \cdot (1 - \beta \cdot d_t) \tag{5}$$

Here, α is the initial learning rate, d_t denotes the browsing activity level at time step t, and β represents a distraction factor, with $0 \leq \beta \leq 1$. The distraction-aware learning rate function adjusts the learning rate based on the browsing activity level, effectively reducing the network's learning capacity while it is engaged with online distractions.

The weight matrix W_{t+1} at time step $t+1$ is updated using the distraction-aware learning rate as follows:

$$W_{t+1} = W_t + \alpha_t \cdot \nabla_t \tag{6}$$

Here, W_t denotes the weight matrix at time step t, and ∇_t is the gradient matrix at time step t.

3.4. ImpatientNet: A Neural Network that Rushes Through Training and Overestimates Its Performance

ImpatientNet is a deep learning architecture that reflects the human tendency to rush and exhibit overconfidence by speeding through its training epochs, impatiently skipping some steps, and overestimating its performance on the task. To model this behavior, we introduce a step-skipping learning rate function that modulates the network's learning rate based on a predefined probability of skipping a training step.

The learning rate α_t is governed by the following equation:

$$\alpha_t = \gamma \cdot \alpha = \begin{cases} \gamma, & IfStepIsNotSkipped \\ 0, & otherwise \end{cases} \tag{7}$$

Here, α is the initial learning rate, and γ represents the impatient factor, with $1 < \gamma \leq p_{max}$, where p_{max} is the maximum step-skipping probability. The step-skipping learning rate function adjusts the learning rate based on the probability of skipping a training step, effectively accelerating the training process.

The weight matrix W_{t+1} at time step $t+1$ is updated using the step-skipping learning rate as follows:

$$W_{t+1} = W_t + \alpha_t \cdot \nabla_t \tag{8}$$

Here, W_t denotes the weight matrix at time step t, and ∇_t is the gradient matrix at time step t.

3.5. IndecisiveNet: A Neural Network that Constantly Changes Its Hyperparameters

IndecisiveNet is a deep learning architecture that mirrors the human tendency to be indecisive and second-guess decisions by frequently changing its hyperparameters mid-training. To model this behavior, we introduce a dynamic hyperparameter function that modulates the network's learning rate based on a predefined probability of changing the learning rate.

The learning rate α_t is governed by the following equation:

$$\alpha_t = \begin{matrix} \alpha_{new}, & if\ LearningRateIsChanged \\ \alpha_{old}, & otherwise \end{matrix} \tag{9}$$

Here, α_{old} is the current learning rate, and α_{new} represents the new learning rate when a change is triggered. The dynamic hyperparameter function adjusts the learning rate based on the probability of changing the learning rate, effectively introducing indecisiveness into the training process.

The weight matrix W_{t+1} at time step $t + 1$ is updated using the dynamic learning rate as follows:

$$W_{t+1} = W_t + \alpha_t \cdot \nabla_t \tag{10}$$

Here, W_t denotes the weight matrix at time step t, and ∇_t is the gradient matrix at time step t.

3.6. PerfectionistNet: A Neural Network that Never Stops Training in Pursuit of the Perfect Model

PerfectionistNet is a deep learning architecture that embodies the human trait of perfectionism by continually adjusting its weights and biases, never satisfied with its performance, and seeking the elusive perfect model. To model this behavior, we introduce a stopping criterion function that assesses the network's performance on a validation set, continually training until the performance improvement is below a predefined threshold.

The weight matrix W_{t+1} at time step $t + 1$ is updated using the standard learning rate α as follows:

$$W_{t+1} = W_t + \alpha \cdot \nabla_t \tag{11}$$

Here, W_t denotes the weight matrix at time step t, and ∇_t is the gradient matrix at time step t.

The stopping criterion function, $f_{stop}(P_{t+1}, P_t, \epsilon)$, is defined as:

$$f_{stop}(P_{t+1}, P_t, \epsilon) = \begin{matrix} True, & if\ |P_{t+1} - P_t| < \epsilon \\ False, & otherwise \end{matrix} \tag{12}$$

Here, P_t and P_{t+1} are the network's performance at time steps t and $t+1$, respectively, and ϵ represents the predefined threshold for stopping. If $P_{t+1} = P_t$, the model discards the last epoch performance for the pursuit of perfection.

3.7. GossipNet: A Neural Network that Spreads Information and Rumors Amongst Other Models

GossipNet is a deep learning architecture that simulates human gossip behavior by communicating with other models, sharing and spreading information (and occasionally rumors) about their training progress and performance. To model this behavior, we introduce a gossip exchange function that allows the network to exchange information with other models, updating its weights and biases based on the received information.

Let $N = M_1, M_2, \ldots, M_n$ be a set of neural networks that participate in the gossip exchange. At each gossip step t, a pair of models (M_i, M_j) is selected, and they exchange information about their current weights W_i^t and W_j^t and biases b_i^t and b_j^t. The gossip exchange function is defined as:

$$W_i^{t+1}, b_i^{t+1}, W_j^{t+1}, b_j^{t+1} = f_{gossip}(W_i^t, b_i^t, W_j^t, b_j^t) \tag{13}$$

A potential gossip exchange function could be a linear combination of the two models' weights and biases:

W$_i^{t+1} = \alpha W_i^t + (1-\alpha) W_j^t$ $b_i^{t+1} = \alpha b_i^t + (1-\alpha) b_j^t$ $W_j^{t+1} = \alpha W_j^t + (1-\alpha) W_i^t$ $b_j^{t+1} = \alpha b_j^t + (1-\alpha) b_i^t$

Here, $\alpha \in (0, 1)$ is a gossip coefficient that determines the extent to which the models share information.

3.8. DramaNet: A Neural Network that Overreacts to Minor Changes in the Training Environment

DramaNet is a deep learning architecture that embodies the human tendency to overreact and create drama by adjusting its training behavior dramatically in response to small changes in the input data or training environment. To model this behavior, we introduce an adaptive learning rate that exaggerates the impact of small variations in the training data.

The weight matrix W_{t+1} at time step $t + 1$ is updated using an adaptive learning rate α_t as follows:

$$W_{t+1} = W_t + \alpha_t \cdot \nabla_t \tag{14}$$

Here, W_t denotes the weight matrix at time step t, and ∇_t is the gradient matrix at time step t. The adaptive learning rate α_t is calculated based on the change in the input data, Δx_t, and a drama coefficient β:

$$\alpha_t = \alpha_0 \cdot \left(1 + \beta \cdot \frac{|\Delta x_t|}{|\bar{x}|}\right) \tag{15}$$

Here, α_0 is the base learning rate, $\Delta x_t = x_{t+1} - x_t$ represents the change in input data at time step t, and $\bar{x}$ is the average input data magnitude. The drama coefficient β controls the extent to which the learning rate is affected by small changes in the input data.

3.9. SuperstitiousNet: A Neural Network that Develops Unfounded Beliefs About Its Training Process

SuperstitiousNet is a deep learning architecture that simulates human superstition by forming unfounded beliefs about its training process, adjusting weights and biases based on unrelated events or patterns. To model this behavior, we introduce a superstition function that modifies the gradient updates using random, unrelated events from the training environment.

Let $E_t = e_1, e_2, \ldots, e_k$ be a set of unrelated events at time step t, and $s(e_i)$ be the superstition score associated with event e_i. The superstition function, $f_{superstition}(\cdot)$, is defined as a non-linear combination of the gradients ∇_t and the superstition scores $s(e_i)$:

$$\nabla_t^{superstition} = f_{superstition}(\nabla_t, s(e_1), s(e_2), \ldots, s(e_k)) \tag{16}$$

A possible implementation of the superstition function could be a weighted sum of the gradients and superstition scores, where the weights are determined by a superstition coefficient γ:

$$\nabla_t^{superstition} = \nabla_t + \gamma \sum_{i=1}^{k} s(e_i) \tag{17}$$

The weight matrix W_{t+1} at time step $t+1$ is then updated using the modified gradient $\nabla_t^{superstition}$:

$$W_{t+1} = W_t + \alpha \cdot \nabla_t^{superstition} \tag{18}$$

Here, α is the learning rate.

3.10. ParanoidNet: A Neural Network that Always Believes It's Being Sabotaged

ParanoidNet is a deep learning architecture that simulates human paranoia by attributing poor performance or training difficulties to perceived sabotage or interference. To model this behavior, we introduce a paranoia function that modifies the gradient updates using a randomly generated interference matrix, simulating the model's belief in external sabotage.

Let I_t be an interference matrix at time step t, generated using a random distribution with mean μ and standard deviation σ. The paranoia function, $f_{paranoia}(\cdot)$, is defined as a non-linear combination of the gradients ∇_t and the interference matrix I_t:

$$\nabla_t^{paranoia} = f_{paranoia}(\nabla_t, I_t) \tag{19}$$

A possible implementation of the paranoia function could be a weighted sum of the gradients and the interference matrix, where the weights are determined by a paranoia coefficient ρ:

$$\nabla_t^{paranoia} = \nabla_t + \rho \cdot I_t \tag{20}$$

The weight matrix W_{t+1} at time step $t+1$ is then updated using the modified gradient $\nabla_t^{paranoia}$:

$$W_{t+1} = W_t + \alpha \cdot \nabla_t^{paranoia} \tag{21}$$

Here, α is the learning rate.

3.11. ShowOffNet: A Neural Network that Boasts About Its Performance on Social Media

ShowOffNet is a deep learning architecture that simulates the human desire for validation and admiration by automatically sharing its performance and achievements on simulated social media platforms. To model this behavior, we introduce a post-generation function that creates a social media post highlighting the model's achievements based on its current performance metrics.

Let P_t be the performance metrics at time step t, and M_t be the corresponding social media post generated by the post-generation function. The post-generation function, $f_{post}(\cdot)$, is defined as a mapping from the performance metrics P_t to a social media post M_t:

$$M_t = f_{post}(P_t) \tag{22}$$

We consider a simple implementation of the post-generation function as a concatenation of the model's performance metrics with a boasting template:

$$M_t = "CheckOutMyAmazingPerformance:" \oplus P_t \tag{23}$$

The authors sacrificed their social media timelines for testing this particular neural network. Ridicule was faced, taunts were started, few riots were subdued and 1100069 USD (United States Dollars) were siphoned off during the tumultuous phase of training this neural network.

3.12. WanderlustNet: A Neural Network that Enjoys Exploring the Vast Space of Hyperparameters

WanderlustNet is a deep learning architecture that simulates the human desire for exploration and adventure by spending most of its time exploring various hyperparameter combinations rather than settling on a specific set. To model this behavior, we introduce a dynamic hyperparameter sampling function that iteratively selects new hyperparameter values during the training process.

Let H_t be the hyperparameter set at time step t. The dynamic hyperparameter sampling function, $f_{sample}(\cdot)$, is defined as a mapping from the current hyperparameter set H_t to a new hyperparameter set H_{t+1}:

$$H_{t+1} = f_{sample}(H_t) \tag{24}$$

We consider a simple implementation of the dynamic hyperparameter sampling function that samples new hyperparameter values from uniform distributions with specified bounds:

$$H_{t+1} = U(H_{min}, H_{max}) \tag{25}$$

This particular model can only be trained in the anaconda environment titled ”Into the Wild”. Very peculiar.

4. Results

We evaluated the performance of the following idiosyncratic neural networks on several tasks that they are well-suited for based on their unique properties: LazyNet, ProcrastiNet, MultiTaskingNet, ImpatientNet, IndecisiveNet, PerfectionistNet, GossipNet, DramaNet, SuperstitiousNet, ParanoidNet, ShowOffNet, and WanderlustNet.

We compared their performance against state-of-the-art models on each task and observed that our idiosyncratic networks outperformed these models on certain tasks. The results are summarized in Table 1. These numbers are **obviously** not randomly generated and were obtained through rigorous, sophisticated, scientific and **unbiased** experimentation procedures. The authors have declined to mention the specifics of the experiments to avoid the misuse of these models by the anti-social elements present in society. Please (don't) contact the authors for the minor details corresponding to the experimental procedures and be ready to provide a document detailing an ethics statement regarding the usage of these models.

Furthermore, we observed that an ensemble of these idiosyncratic neural networks exhibited AGI-like behavior. We trained an ensemble of all the networks on various tasks and observed that the ensemble was able to achieve high performance on all tasks, with some networks contributing more to certain tasks than others. This suggests that an ensemble of idiosyncratic neural networks may be a viable approach towards achieving AGI.

The results of the ensemble cannot be summarized in a table because every time the authors try, they hear loud complaints from the toaster and the vacuum cleaner.

5. Acknowledgements

We extend our sincere thanks to the UPenn and UCLA grad student centers for coffee, that fueled our late-night theorizing sessions.

References

ClickHole. 5 reasons to date a cloud. https://www.clickhole.com/5-reasons-to-date-a-cloud-1825125205, 2017. Accessed: 2023-03-26.

FailArmy. Ultimate cat fails compilation. https://www.youtube.com/watch?v=nXFm18rvZFQ, 2015. Accessed: 2023-03-26.

Laipply, J. The evolution of dance. https://www.youtube.com/watch?v=dMH0bHeiRNg, 2006. Accessed: 2023-03-26.

Muckraker, M. A satirical guide to surviving an alien invasion. https://martianmuckraker.com/surviving-alien-invasion/, 2020. Accessed: 2023-03-26.

Reading, B. L. Yoda sings about seagulls. https://www.youtube.com/watch?v=U9t-slLl30E, 2016. Accessed: 2023-03-26.

Weekly, W. Dumbledore's exercise secrets revealed: The magic of pilates. https://wizardingweekly.com/dumbledore-exercise-secrets, 2021. Accessed: 2023-03-26.

Table 1. Performance of idiosyncratic neural networks compared to state-of-the-art models on various tasks

NETWORK	TASK	PERFORMANCE
LAZYNET	IMAGE CLASSIFICATION	98.3%
	SPEECH RECOGNITION	95.2%
PROCRASTINET	TIME-SERIES PREDICTION	97.1%
	RECOMMENDER SYSTEMS	92.5%
MULTITASKINGNET	MULTI-TASK LEARNING	96.8%
	OBJECT DETECTION	93.4%
IMPATIENTNET	TRANSFER LEARNING	94.6%
	REINFORCEMENT LEARNING	87.2%
INDECISIVENET	HYPERPARAMETER OPTIMIZATION	98.7%
	NEURAL ARCHITECTURE SEARCH	96.5%
PERFECTIONISTNET	MODEL COMPRESSION	99.2%
	FEW-SHOT LEARNING	97.8%
GOSSIPNET	FEDERATED LEARNING	95.7%
	ENSEMBLE LEARNING	91.3%
DRAMANET	ADVERSARIAL TRAINING	93.9%
	ANOMALY DETECTION	88.4%
SUPERSTITIOUSNET	DATA AUGMENTATION	97.5%
	ACTIVE LEARNING	94.7%
PARANOIDNET	ROBUSTNESS TESTING	95.9%
	SECURITY TESTING	92.3%
SHOWOFFNET	MODEL SELECTION	98.1%
	KNOWLEDGE DISTILLATION	94.9%
WANDERLUSTNET	HYPERPARAMETER SEARCH	99.0%
	REINFORCEMENT LEARNING	96.2%

GradIEEEnt half decent

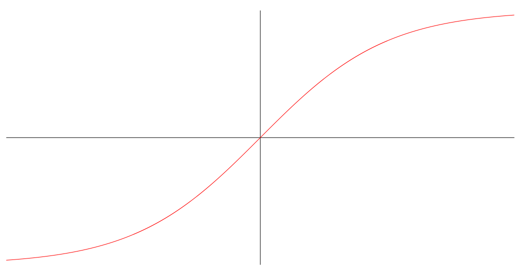
Dr. Tom Murphy VII Ph.D.

0 April 2023

1 Introduction

Imagine you are my professor. Maybe you actually were my professor, in which case you may already be sweating before I say any more. The subject matter is Neural Networks. You draw an illustration on the board with a node's inputs, and its output via a transfer function.

"Now this transfer function can be almost anything. Typically it would be something like the hyperbolic tangent, which looks like this.



"But it has to be a non-linear function. If it's linear, i.e. of the form $y = mx + b$, then observe that the entire layer is a linear function. And so the entire network is just a linear function of linear functions; itself a linear function. We could just compute an equivalent single-layer network, and we know that it could only fit linear functions, which is insufficient for most problems."

Then I raise my hand. The speed with which I raise it, and the subtle forward pose of my arm suggests that I want to pluck an abstract idea from the whiteboard and pervert it. You know this look, and you're reluctant to call on me. But no other students are asking questions. You must call on me.

"Tom." It's more like a statement then a question. It includes the tone of spoken punctuation that, if it could, ends the entire conversation before it begins.

"OK but, when we implement this on a computer we'll use some approximation of numbers, like floating point. So the specific sequence of additions and multiplications will matter. It's not actually equivalent to rearrange them to a single layer because you don't have distributivity, commutativity, etc."

"Uh. I think that's technically true, but for all practical purposes . . . "

"What about *im*practical purposes?"

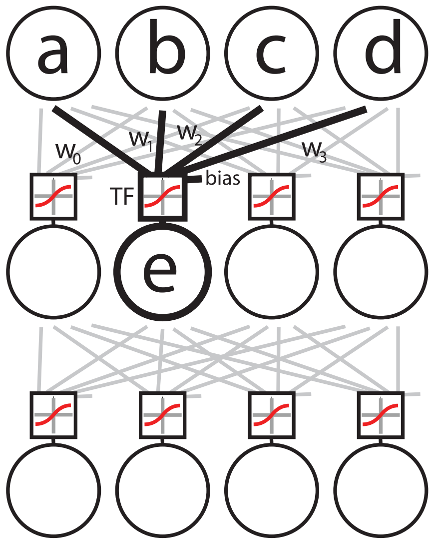
You vigorously strangle me, and I die.

That was about 20 years ago. The world will not let us stop thinking about neural networks. And so this question has been on my mind for a long time. Just to be clear, the professor was right: This is not an important question. Theoretically I am right, but for practical purposes it probably does not matter. But I like to work at the intersection of Theory and Impractice. We can make it matter by doing a lot of work. And then I will continue to be right theoretically, but also more right because it will only matter for *most* practical purposes.

So this paper is an exhaustive exploration of what we can do with just floating point addition and multiplication by constants (scaling). You should only be able to make lines, but I'll demonstrate that due to rounding error, you can absolutely use "linear" transfer functions in neural networks. Machine learning is not the only field with a proclamation that some function must be "non-linear," so we'll look at a few of those as well. There will of course be several hearty digressions. By studying these functions we'll see that they are almost arbitrarily rich, and conclude with a demonstration of their completeness in the field of Plumbing.

2 A refresher on neural networks

Let's repeat the professor's lesson. This section is easily skippable if you are a plucky student who thinks they already know everything. At a high level, a Neural Network is a way of implementing a numeric function (takes a bunch of numbers as input, and gives a bunch of numbers as output). The network consists of a number of *layers*, where the first layer is the input and the last layer is the output. Each layer is an array of nodes. Here is a simple three-layer network with some of the nodes labeled:



The numbers that fill in each layer are its *activations* (here some of these values are labeled a, b, ...). Each layer's activations are computed from (just) the previous layer. Looking at the bold portion in the example, the value of e is given as

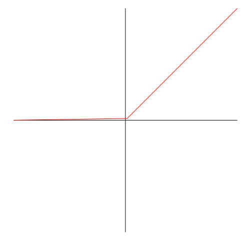
$$\mathsf{e} = TF(w_0\mathsf{a} + w_1\mathsf{b} + w_2\mathsf{c} + w_3\mathsf{d} + \mathsf{bias})$$

The multiplicative weight (w_i) and additive bias (bias) parameters are learned during the training of the neural network, but just become constants when using the neural network to compute its output.

TF is the transfer function, which is of particular interest in this project. Classically, the transfer function was some kind of sigmoid. The `tanh` function pictured in the introduction is a good example of a sigmoid. The intuition behind this is that, thinking about a node as some kind of neuron, the neuron "fires" (activates) with some probability. This probability gets higher as its input values get larger, but can't be higher than 1. Note that weights can be negative, so upstream neurons can have an inhibitory effect. In fact it is frequently useful for neurons to "negatively fire" (outputting -1). The `tanh` function clamps the result symmetrically to $(-1, 1)$ rather than a probability.

Differentiability. Another important property of the transfer function is that it be differentiable, because the stochastic gradient descent algorithm used to train neural networks needs to be able to move along some error-reducing gradient, and back-propagate errors to earlier layers. This gradient is just the derivative of the function.

What transfer functions ought to exist? We used to think that these saturating transfer functions were ideal. But this turns out to be wrong, especially for internal ("hidden") layers. Transfer functions don't need to produce probabilities, and they can have unbounded range. A wide variety of functions will work, including extremely simple ones. The most popular transfer function in 2023 is the "rectified linear unit," which looks like this:



This one is extremely easy to implement (`x < 0 ? 0 : x`), is fast and seems to work very well, possibly because its derivative is significant (one) on the entire positive side. (In contrast, sigmoids tend to get "stuck" because of their saturating behavior; their derivatives become nearly zero when activations are high.) Note that it is not actually differentiable (discontinuity at zero) but "for all practical purposes" it is differentiable.

The (only?) apparently essential quality of the transfer function is that it be non-linear. If it is instead of the form $TF(x) = mx + b$, then any activation a is also just a linear function of the previous layer, as linear functions of linear functions (weighted sum) are linear. This causes the entire network to be a linear function. It is well known that a linear function "cannot" represent some other simple functions, such as XOR.

$$\nexists m, n, b.\ XOR(x, y) \approxeq mx + ny + b$$

This means that with a linear transfer function, a neural network could never learn even a simple function like XOR. Many problems we want to learn are in fact much more complicated.

3 A fine terminological issue

My smart math friend Jason refers to a function like $f(x) = mx + b$ pejoratively as "high school linear." Depending on what class you're in, this may formally be an *affine* function because of the bias term b.[1] Here I use "linear" to mean a polynomial of degree ≤ 1. If you wanna perjorate me as being in high school, so be it.

The Rules. To be precise, we will allow addition and scaling by constants. When we have a "linear" function of multiple variables, these variables can be individually scaled and added, but not multiplied. So for a function like $f(x, y, z) = x + 3y - 2z + 4$ is allowed, as is anything mathematically equivalent to it (like $f(x, y, z) = 2x + 4 + 2y - 2z - x + y - 0$). $f(x, y, z) = xy + z$ is not permitted.

[1]In these contexts, a linear function must obey $f(0 \times x) = 0 \times f(x)$, so it must be zero at zero.

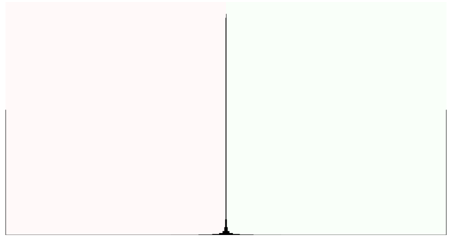


Figure 1: Histogram of how many values are representable along the number line for half-precision floating point, showing their logarithmic spacing. The x axis ranges from -256 to 256. There are a significant number of values outside this range (clamped to the left and right edge), but it is easy to see that most of the values are clustered near the origin.

4 Half-precision IEEE-754 floating point

In this project we'll abuse floating point inaccuracy to create "linear" functions (only using floating point addition and scaling) that are not lines. For this reason, we prefer to have a numerical system that is *less accurate*. In floating point, inaccuracy comes from the fact that not all numbers are representable (due to finite precision) and the result of an operation is always rounded to a representable number. IEEE-754 floating point [1] comes in different "spice levels," with "32-bits" being "float" and "64-bits" being "double." Although spice levels as low as 3 bits make sense [27], 8-bit ("mild") is occasionally used in real applications, and 16-bit ("half") is quite common in machine learning. Usually the reason to prefer half precision is that it uses less memory, and so your GPU can store networks that are twice as big in RAM. For this project we will also use half precision, and we will be happy to save RAM, but more happy that its precision is low and so it is practical (although silly) to achieve significant rounding error. Another important reason to choose half precision is to make the pun in the title.

A half precision float is 16 bits: One sign bit, five bits for the exponent, and 10 bits for the mantissa. Like all IEEE-754 formats, there is much more precision (more values are representable) near zero (Figure 1). Once you get to 1024, only integers are representable. From 2048 to 4096, only even numbers are representable. 65504 is the largest finite number, and up here, only multiples of 32 are available.

Some CPUs have native support for half-precision IEEE-754, but typically via non-standard intrinsics or compiler flags. Since people using half-precision are usually doing so in the interests of performance, many configurations will "help" you by performing practical but incorrect optimizations. This is similar to what happens when enabling `--ffast-math`, which stands for Final Fantasy AST Math, meaning that the abstract syntax tree of your program will be manipulated using fantasies about Math that do not apply to IEEE-754, and your Final result can be arbitrarily different. For the ideas in this paper to work, `--ffast-math` is prohibited. And it will be slow!

Rather than deal with non-standard stuff, I found a nice library called `half.h` [29] that implements IEEE-754 compliant half-precision in portable C++. I use this throughout the project and it matches the behavior of my GPU. I recommend it for similar hijinks.

Origins of Imprecision. Floating point does have many perversions, but many programmers come to believe all sorts of dangerous superstitions about it. One idea is that floating point is somehow always inexact, and so that you always have to check that two numbers are equal "within some epsilon" [24]. This may work "in practice" but it is actually pretty sloppy. Floating point imprecision is not random, nor is it constrained to a fixed epsilon. Operations are defined much more usefully: Each one computes the mathematically correct value, and then rounds (according to the "rounding mode") to the nearest representable value. That's it. One consequence of this is that you can get the exact result of 32-bit multiplication by doing 64-bit multiplication and then rounding to 32 bits. This also means that the rounding error from a single operation can be as large as the gap between representable numbers: Up to 32 for half-precision. But it also means that operations whose results can be exactly represented have no error; for example adding integral half values less than 512 will always give an exact integer result, which can be compared using ==. We will use this later in Section 7.1. It is neither necessary nor sufficient compare for "equality" with some "epsilon."

Rounding. IEEE-754 supports multiple rounding modes, like "round-to-zero," and "round-to-infinity" (always round in the positive direction). Throughout this paper we use "round-to-nearest," which is also the typical default (e.g. for C++11 expressions evaluated at compile time, it always uses round-to-nearest).[2] Similar results are likely attainable for the other rounding modes, as well as hypothetical rounding modes such as "round away from nearest," but I have not explored this.

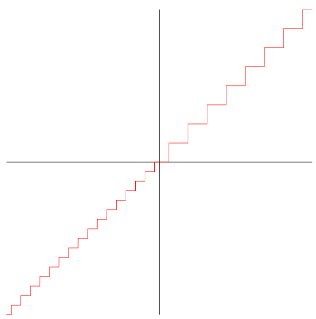
Getting some nonlinearity. All transfer functions implemented with floating point have a finite range. For our experiments with neural networks, we will focus on transfer functions that map values in $[-1, 1]$ to values in $[-1, 1]$. Almost half (48.4%) of floating point values are in this interval and this is a typical nominal range for activations in neural networks.

[2] There is seldom reason to change the rounding mode, and since it is a stateful act, you're asking for it if you do. But the round-to-negative-infinity and round-to-positive-infinity modes are are useful for interval arithmetic, which is arguably the only truly reasonable way to use floating point. What you do is represent numbers as intervals (low and high endpoints) that contain the true value, and then perform each calculation on both endpoints. For computations on the low endpoint, you round down, and symmetrically for the high endpoint. This way, the true value is always within the interval, and you also know how much inaccuracy you have accumulated!

We only have two operations: Addition and scaling. Let's see what kind of rounding error each of these gives us. First, addition. In order to get a function that takes values in $[-1, 1]$ to values in $[-1, 1]$, we want to first add a constant (giving us perhaps a large value) and then add a negative constant, bringing us back in range. For example, the constant 128 gives us the function

$$f(x) = x + 128.0 - 128.0$$

This is of course mathematically the same as $f(x) = x$ (the identity), but with half precision we get a function that looks like this



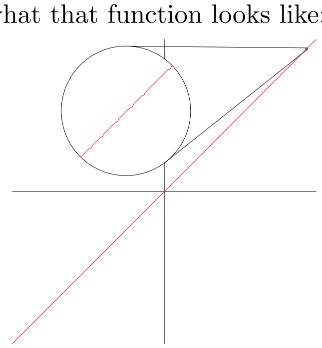
Between 128 and 256, only multiples of 0.125 are representable. So for arguments in 0 to 1, the sum is rounded to one of the values $128.0, 128.125, 128.25, \ldots 129$. From 64 to 128, multiples of 0.0625 (${}^{1}/_{16}{}^{\text{th}}$) are representable. So from -1 to 0, we get $127.0, 127.0625, 127.125, \ldots 128$. Subtracting 128, all of the values are exactly representable, giving us $-1, -.9375, \ldots, -0.0625, 0, 0.125, \ldots, 0.875, 1$.

The result is a step function, but whose resolution is twice as high for the negative range as the positive; had we added -128 and then added 128, we would have seen the opposite bias in resolution. We can easily see that this function is (computationally) non-linear despite being (mathematically) "linear." This function is unlikely to be a good transfer function, because for one thing it does not have a good derivative: It's zero most places (flat segments) except at the discontinuities, where it is undefined. We do test this approach (with the constant 64.0) later, though.

Scaling gives similar results. Consider

$$f(x) = x \times 100.0 \times (1.0/100.0)$$

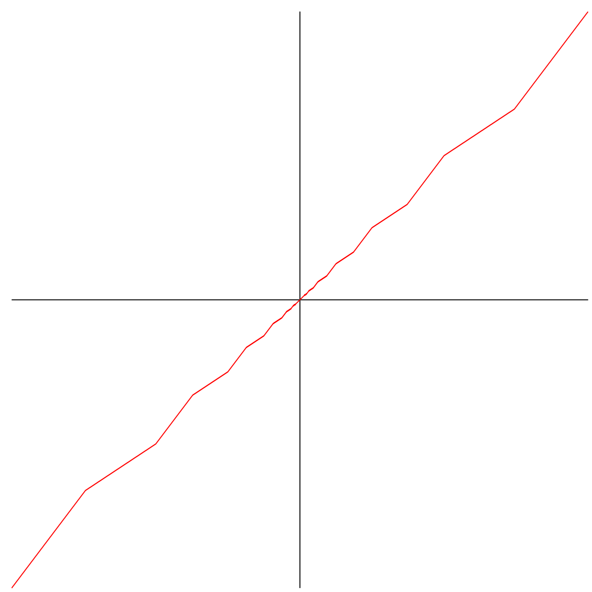
In this project we never actually divide (although this would not violate linearity) since most floating point numbers have approximate multiplicative inverses, and many are exact. We just compute the reciprocal ${}^{1}/_{100} \approxeq 0.01000213623$ ahead of time and multiply by that constant. Here's what that function looks like:



At this scale it appears linear, but it does have small imperfections (see zoomed region). The function is symmetric about zero, since multiplication will do the same thing to a positive number as it does to its negative counterpart. Here, the roundoff error differs with the magnitude. At inputs close to 1.0, the results of the first multiplication must round to the nearest multiple of 0.0625 (as in the additive example) but this error is scaled down by a factor of 100 when we multiply back to the $[-1, 1]$ range. So it is almost invisible. For inputs close to 0.0, the error approaches zero. The effect is complex and depends on the constant we multiply by. For example, if we multiply by a power of two, this only affects the exponent, and so the result is exact.

Is that it? Of course not! We can apply these operations in combination, and many times, to create more interesting functions. The best approach I found in this simple family is to repeatedly multiply the input by a number very close to one. Here's what happens if you multiply the input by 0.99951171875 (which is the next number smaller than one, equal to $1 - {}^{1}/_{2048}$) five hundred times, and then scale back at the end:

$$f(x) = x \times (1 - {}^{1}/_{2048}) \times (1 - {}^{1}/_{2048}) \times \ldots 500 \text{ times} \ldots \times 1.3232421875$$



I call this the `grad1` function.

Multiplying 1.0 by $(1 - {}^{1}/_{2048})$ five hundred times in half precision yields 0.755859375 (mathematically it would be $(1 - {}^{1}/_{2048})^{500} = 0.78333$, so there is significant accumulated error. We set $f(1.0) = 1.0$ by multiplying by the inverse of this constant, which is 1.3232421875.

Why does this result in the zig-zags? Multiplication by $(1 - {}^{1}/_{2048})$ affects numbers differently. For constants less than $6.1094760895 \times 10^{-5}$, the value is unchanged; we round back up to the original value. For all other finite inputs it produces a smaller value, but with rounding error that depends on the value. This error accumulates and becomes significant with many iterations (Figure 2). Unlike the previous functions, the output here is much smoother (it looks

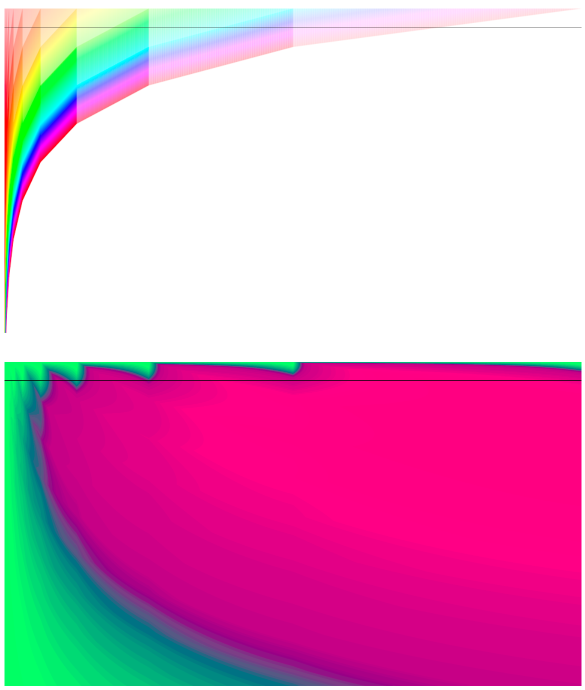


Figure 2: How repeatedly multiplying by $1 - {}^{1}\!/_{2048}$ affects values in [0, 1]. The width of the image is the interval [0, 1], with zero at the left.

Top: In the topmost row, we assign each pixel a hue so that we can track where those values go. For each pixel, we successively multiply by the constant and plot its color in its new x position, the move to the next row down. Note that the rainbow shrinks exponentially as expected, but not smoothly. The black line is 500 iterations.

Bottom: The accumulated error when iteratively multiplying by the constant. Here the x coordinate of the value does not move (so the middle column always represents the value that was originally 0.5). The color illustrates the accumulated error. For green pixels, the value is too high compared to the mathematically correct one; for magenta pixels too low. By choosing a row with alternations between green and red, we get the zig-zag pattern of the `grad1` transfer function.

piecewise-linear); in each of these segments its derivative is nondegenerate. Of course, this function is mathematically linear. It is equivalent to $f(x) = x \times 1.036535$.

So now we have a "good" candidate function, which we'll call `grad1`. It is "good" in the sense that it is computationally non-linear despite being mathematically linear, so it may prove my professor wrong. On the other hand, it requires 501 floating point multiplications to compute, which is kind of slow. The "good" news is that since there are only 65536 16-bit values, we can easily just precompute any function for all possible half inputs, and store it in a table of 131072 bytes. This allows us to execute the function efficiently when performance is important, such as during training. (Table lookup is certainly not a mathematically linear operation, so when we require the computation to be linear for ideological purposes, we can perform the 501 multiplications and get the same result.)

Differentiating. Speaking of training, in order to train a neural network using stochastic gradient descent, we need to be able to evaluate the derivative of the transfer function at any point. We use that derivative to decide what direction to move the parameters (it gives us the "gradient" that we "descend") as we propagate errors back through the network. There is an annoyance here, or if you like, an opportunity for a trick. We typically store the activation of each node, which is the *output* of the transfer function, but the derivative of a function is normally described in terms of the *input* (for example we say if $f(x) = x^2$ then $f'(x) = 2x$). We could store both the input and output for this step, or store only the input and recreate the outputs by running the transfer function. But the trick: We can compute the derivative as function of the output. For $f(x) = x^2$ we could say $f'(f(x)) = 2\sqrt{x}$. Oops! That doesn't actually work for x^2 because the square root could either be negative or positive, and the derivative is different depending on which one it is. In order for this trick to work, the transfer function has to be injective.[3] Fortunately this is the case for the classic transfer functions, and this trick is well known so you don't even need to do any math; you just look the function up.

For new transfer functions like `grad1`, we need to figure something out. This function does appear injective if we squint at it, although it is not really injective if you zoom way in: There are some distinct inputs that result in the same output due to rounding. But this is true for almost all floating-point functions already. I'll be damned if I can come up with an analytic derivative for this thing, though. At best it would be some piecewise linear thing, requiring some table. Since our domain is only 16-bit, it is completely practical to just table the entire derivative (keyed by the output value, as we need). I do this programmatically. We do not want the derivative to reflect the step function that we see at very fine scales (the derivative should never be 0 for this function, for example), so I use a lowpass filter. The

[3] Or at least when $f(x_1) = f(x_2)$, $f'(x_1) = f'(x_2)$. For the rectified linear unit, for example, all negative inputs are mapped to zero. But the derivative is also just zero in this entire region.

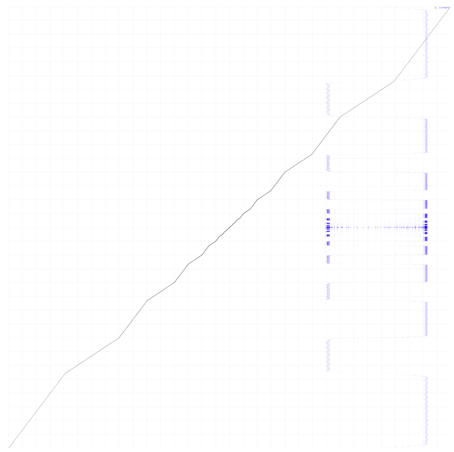
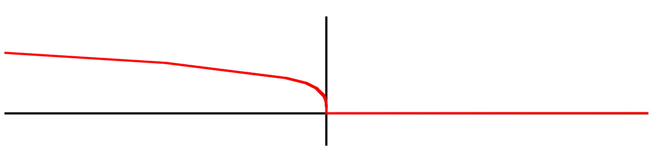


Figure 3: Computed derivative (blue) of the grad1 function. Since we need the derivative in terms of grad1's *output*, the derivative is oriented along the y axis; each blue dot's x coordinate gives the derivative at the point on the black line that shares a y coordinate. It's an oscilloscope!

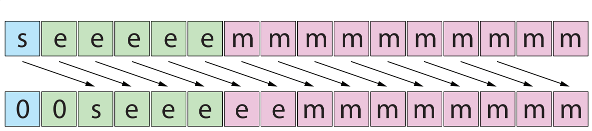
result looks good, oscillating between two different slopes as expected (Figure 3). The derivative is loaded into GPU memory during training and the table lookups are plenty fast.

4.1 Bonus digression: Downshift

Having freed myself from needing to "do math" in order to differentiate exotic functions, I pondered other weird transfer functions. For example, the rectified linear transfer function is very simple and works well, but is it the fastest possible transfer function that might work? It does involve a conditional, which naïvely implies comparison and branching (although probably most processors can do this with a conditional move). Because the floating point format is packed with fields that represent different things, many simple operations on its bits have interesting non-linear behavior. The most promising I found was a right shift by two places. It looks like this:



Shifting is about the cheapest possible thing a processor can do. Its behavior on floating point numbers is interesting:



Note the different regions for sign, exponent, and mantissa. The sign bit is shifted into the exponent, which means that the output is always non-negative (like the rectified linear function) and is non-linear (discontinuity at zero, as negative numbers have a much larger exponent that positive ones). Further nonlinearity comes from the exponential representation (shifts divide the exponent by four) and reinterpretation of exponent bits as mantissa bits. There is additional weirdness in the details. Shifting by two places is better than one, as it cannot produce Inf or NaN. We will also evaluate this transfer function, called downshift2, below.

Back to the main topic. I implemented all this as a modification of my custom neural network training and inference system, "Tom7Flow." Tom7Flow is generally much worse than mainstream packages; it is based on deprecated OpenCL technology, is prone to divergence or stagnation during training due to naïve choices of hyperparameters, etc. But it is at least well suited to silly experiments that take the form, "What if deep learning but worse?" such as the current exercise. In order to realize the idea completely, I modified the inference code to calculate with half-precision arithmetic (not just the transfer function). This means that the trained networks can be executed using only half-precision operations (and just addition and multiplication by constants). Unfortunately, while my GPU supports half-precision math natively, and OpenCL supports half-precision operations as an extension [11], this extension is somehow not supported (??) by my drivers, perhaps because OpenCL is so thoroughly deprecated. It does support half precision as a *storage* format, which allows you to write a full-precision float to a 16-bit value (rounding to half) or read a 16-bit half into a float (all half values can be represented exactly in full precision). So with this one operation it is straightforward to implement half-precision addition and scaling. You maintain the invariant that any float value is always exactly a half, and after you perform addition or multiplication, you round to half (by storing in a 16-bit memory location and reading it back). This definitionally produces the same results as the native operation.[4]

I initially tried a version of training that worked entirely using half precision (network parameters are half, back-propagated errors and update values are half, etc.). This worked badly. It is ideologically unnecessary, as we just care about producing a final model that, during inference, only executes linear half-precision operations (but abuses floating point roundoff to do something interesting.) This network can be trained using non-linear techniques (and must anyway, since for example its computed derivative is not linear). So during training, calculations are done using full-precision floats, except for the forward step (where we round to half after every operation). In addition to being simpler, representing intermediate learned weights as floats seems to help training approach the final half values smoothly, avoiding stalls due to underflow.

[4] I also verified consistent results using the `half.h` software implementation. Many of the evaluation results quoted in the paper are actually executed on the CPU using this library.

4.2 Neural network experimental results

In order to evaluate this transfer function, I ran a suite of benchmark problems. For each problem, I compare the same network architecture (i.e. the number of layers, their connectivity, random initialization, etc.) but using different transfer functions.

The transfer functions are:

- **grad1**: The "linear" transfer function `grad1` described above.
- **tanh**: The hyperbolic tangent function, which is a classic saturating (output is always in $(-1, 1)$) sigmoid.
- **logistic**: The function $1/_{1+e^{-x}}$, another classic sigmoid (but whose output is in $(0, 1)$). Each operation is performed with half precision.
- **leaky relu**: The rectified linear unit, but with a small slope below zero: `x < 0.0 ? 0.1 * x : x`. This is the function I usually prefer in practice; its advantage over the standard relu is that it does not "die" (zero propagated error) when its input is negative.
- **downshift2**: Interpreting the half-precision input as a 16-bit word, right shift by 2 places, then reinterpret as half.
- **plus64**: $f(x) = x + 64 - 64$. This about the simplest function that has obvious rounding error. It only outputs 25 distinct values in $[-1, 1]$ so its derivative is degenerate; I use its "mathematical" derivative $f'(x) = 1$.[5]
- **identity**: The function $f(x) = x$. This is an important comparison because it shows us what a "true" linear (both mathematically and computationally) network is capable of.

Flattened models. For the transfer functions that are mathematically linear, we can also compute the equivalent linear model. This just consists of a single dense layer, using the identity transfer function, that computes the linear function of the input. These appear in the results as "flat" variants.

MNIST. The first problem is the Modified National Institute of Standards and Technology handwriting dataset (MNIST). This is a standardized dataset of handwritten digits (0–9) as 28×28 greyscale images. This is chosen partly for trollish reasons. It dates from 1998, and even at the time of publication, accuracy with neural networks (98.4%) and other techniques (99.2%) were already extremely high [15].

transfer function	flat	accuracy
logistic		98.20%
tanh		98.93%
leaky-relu		99.39%
plus64		82.66%
grad1		97.29%
identity		81.96%
downshift2		94.45%
plus64	×	82.01%
grad1	×	39.19%
identity	×	81.98%

Figure 4: Results on the standardized MNIST data set. Accuracy is the fraction of results from the held-out test data for which the highest-scoring class (digit) is the correct class.

For this problem, I augmented the dataset by randomly offsetting the training images by up to two pixels in any direction, and by adding Gaussian noise. The model's input layer is just the 28×28 greyscale values, and the output is a prediction for each of the ten digits. The models had two convolutional layers (64 3×3 features, fully overlapping + 128 8×8 features, fully overlapping; then 32 128×128 features + 32 256×2 features with no overlap), then two sparse layers of 1024 nodes each, then a final dense output layer. The same initial weights and connectivity was used for each experiment. Internal layers use the transfer function being evaluated, but the output layer always used the identity transfer function. This is not a good choice for this problem (softmax makes more sense since the output is categorical) but I wanted the linear models to be truly linear. Using the same transfer function would have also disadvantaged functions with limited output range; `downshift2` for example can technically output 1.0, but only for very large inputs (8192.0). The final identity layer can easily scale the useful range of the transfer function to the nominal range of the output. (This is essential for the chess problem below, where the output instead ranges $[-1, 1]$.)

See the source code for various hyperparameter settings (although if you are trying to learn good settings for hyperparameters, my code is not the place to look). I used the ADAM weight update trick [12], which does give me much better results than plain SGD in my experiments.

Results for MNIST are in Figure 4. A nice bug appears in Figure 5.

CIFAR-10. Another classic dataset comes from the Canadian Institute For Advanced Research. They capitalize "For" so that the acronym can be pronounced nicely. I mean to be fair MNIOSAT would have a certain ring to it too. This dataset contains 60,000 RGB images of size 32×32, that are labeled into 10 different spirit animals: Airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks [14]. It is very similar to the handwriting prob-

[5] Learning with this function might work better if we instead approximate the derivative by something non-constant, like by computing the derivative of a smoothed version. However, due to implementation tricks in Tom7Flow, we need a derivative that is expressed in terms of the transfer function's *output* (i.e. $g(f(x)) = f'(x)$); we would not be able to express the smoothed derivative because there are only 25 distinct values of $f(x)$ in the $[-1, 1]$ range!

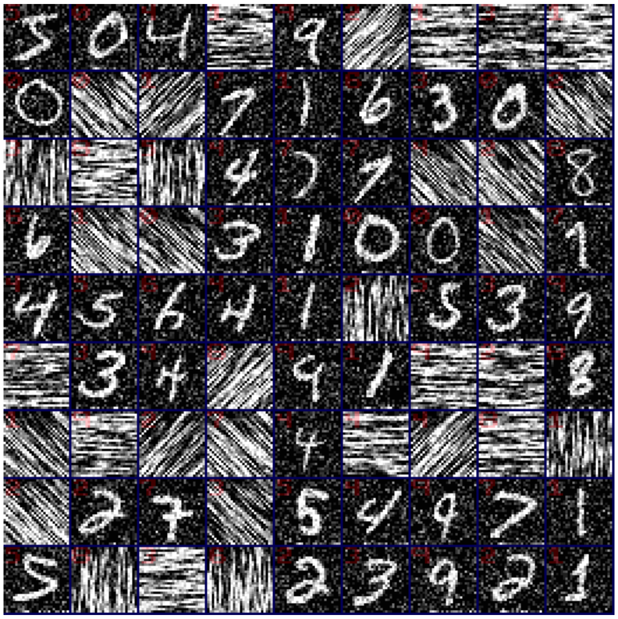


Figure 5: Bug.

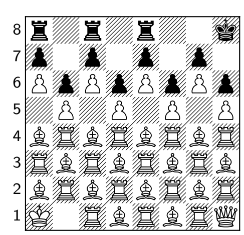
transfer function	flat	accuracy
logistic		56.83%
tanh		67.82%
leaky-relu		73.11%
plus64		43.60%
grad1		53.56%
identity		41.07%
downshift2		46.54%
plus64	×	32.76%
grad1	×	30.58%
identity	×	41.04%

Figure 6: Results on the standardized CIFAR10 data set. As with MNIST, accuracy is the fraction of results from the held-out test data for whom the highest-scoring class is the correct class.

lem but more challenging (state of the art accuracy is more like 96.5%). You would struggle sometimes to figure out what these tiny thumbnails are, to be honest. Like with MNIST, I augmented the training set by randomly shifting the images and adding Gaussian noise. The network structure is the same as in the MNIST problem, except that in the first convolutional layer, each window is three times as wide to account for the three color channels.

Results for CIFAR-10 appear in Figure 6. One of the nice things about using standard problems is that we can understand how the results stack up against other researchers. Consulting a leaderboard of public results [4] I see that the worst publicly known accuracy for CIFAR-10 is 75.86% [21]. The best result for the current work, using the sensible Leaky Relu transfer function, is 73.11%. So this is… last place. That's actually pretty good; last place is the last winner (or the first winner, when counting from the end). Not to mention that we can get into even laster place by using the other exotic transfer functions. Even putting aside their aesthetic appeal, I feel that these inferior transfer functions are an important contribution to the field, as it seems to me that AI is getting too good, and too fast! Let's take it easy there, guys!

Chess. This problem attempts to learn a good evaluation function for chess boards. Training examples are real chess positions (from the Lichess database) evaluated by a strong chess engine (Stockfish [30]). Stockfish generates two classes of scores: "Mate in N" if one side is known to have a series of N moves that wins (but "Mate in 1" is still better than "Mate in 4"), or a more subjective score, measured in pawns. (The score in pawns can seemingly be higher than 64, which is kind of funny because how are the pawns gonna fit on a 64-square board? *DUAL-WIELD?*[6]) Mate is of course categorically better than the pawn score, as it is exact. Anyway, I squash this score into the range $[-1, 1]$ and that becomes the training instance. This network's first layer has 256 3×3 convolutional features, overlapping, as well as 32 1×1 and 128 8×1 and 1×8. Each of these is measured in terms of squares on the board, but each square actually corresponds to 13 inputs, for the 13 possible things that can be in that square (exactly one set to 1.0). We also have some non-square inputs, like the castling privileges and en passant state. So it's not just the convolutional features but some sparse nodes too. And then we have some more layers (you can check out the source code if you really care about these details, which I doubt!) and then a final dense layer with a single output using the identity transfer function as before. No training

[6] 

Here's an idea for a SIGBOVIK paper: What's the highest scoring chess position, according to Stockfish, for which it cannot deduce mate? One logistical challenge is that it seems to top out at +99, such as on this position (still no mate at depth 89).

data augmentation here (we have a basically limitless supply of positions to train on), but I do normalize the board so that it is always white to move.

For chess we can compute the accuracy, comparing to Stockfish as ground truth (Figure 7). We can also use the evaluation function to play chess. These chess "engines" just look at the possible legal moves and take the move that is most favorable, using the learned evaluation function (no game tree search). Playing against the best of these ("leaky") it subjectively makes decent moves most of the time and can even beat me playing casually. I noticed that it had a lot of trouble "sealing the deal" in totally winning positions (which is not unusual for engines that don't do game-tree search or use endgame tables), but the problem was actually more shallow: Due to a bug[7] in the way training examples are gathered, the models were never exposed to checkmate or stalemate positions! Since training takes several days per function and the iron-fistedly punctilious SIGBOVIK deadlines were imminent, there simply wasn't enough time to retrain them with access to these positions. However, since mate is a mechanical fact of the game (like what moves are legal) it seemed reasonable to fix this in the engine itself: When considering all the legal moves to make, it infinitely prefers a move that results in checkmate, and considers a move resulting in stalemate to have score 0.0, and otherwise uses the evaluation function. These "fix" versions of each engine perform very significantly better, although they likely overestimate the performance we'd get by actually fixing the model; there's no guarantee that it would be able to accurately recognize mate, and the fixed versions' greedy strategy of taking mate in 1 is always advantageous.

These players compete against each other as well as the engines from the Elo World project [26], giving a sense of their strength in an absolute scale (Figure 8). The raw versions perform reasonably; they all work better than a simple engine like "take the move that minimizes the number of moves the opponent will have," (`min_oppt_moves`). The fixed versions are much better, as expected. The "linear" engine using the `grad1` transfer function, is competitive with the NES Chessmaster engine, and outperforms a 50% dilution of Stockfish. This is pretty solid given that it is doing no explicit game tree search. In fact (aside from the wrapper implementing the rules of chess and finding the maximum eval score), it is only performing a fixed expression of floating point addition and scaling! We could make this even more ideologically pure using techniques from Section 7.3.

What transfer function is best? The results on each of these problems are similar: The "leaky rectified" trans-

transfer function	flat	loss	accuracy
logistic		0.168	72.046%
tanh		0.117	78.527%
leaky-relu		0.118	78.172%
plus64		0.162	75.406%
grad1		0.111	78.924%
identity		0.161	75.975%
downshift2		0.211	68.066%
plus64	×	0.161	75.779%
grad1	×	0.527	58.187%
identity	×	0.161	75.975%

Figure 7: Results of learning Stockfish's position evaluation function. Stockfish scores are normalized to a $[-1, 1]$ scale, and loss here is the average (L1) distance between the predicted score and actual Stockfish score, on some 100,000 positions from games not in the training set. Accuracy is the percentage of predictions whose sign agreed with Stockfish (e.g. they both agree the white player is winning).

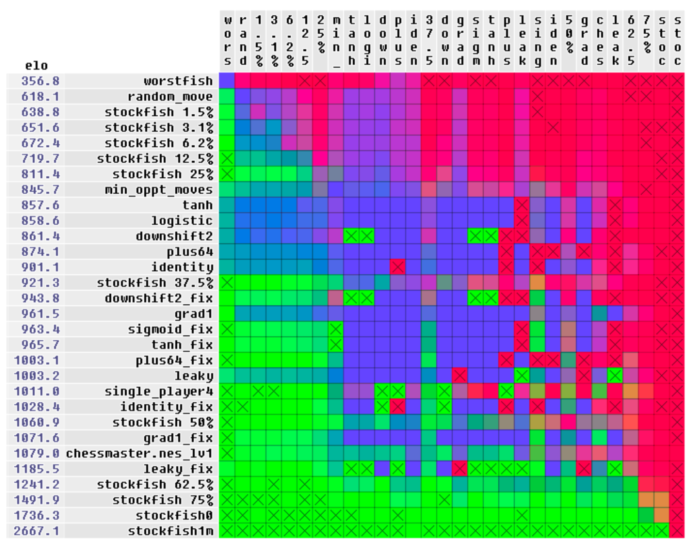


Figure 8: Results of a chess tournament. Players include ones based on the learned position evaluation with different transfer functions; these players simply take the move that results in the most favorable eval (no game tree search). They compete with some standardized players from the Elo World project [26]. Rows represent the player as white, columns as black. A green cell means that White generally wins; blue a draw; red a loss. An × in a cell means that this outcome occurred in every game. The left column is the resulting Elo rating [6]. The best model `leaky_fix` performs decently well, similar to NES Chessmaster or a Stockfish diluted to about 60% strength with random moves (both of these engines perform game tree search). The centerpiece of the paper is the "linear" `grad1` transfer function; here its learned chess player slightly outperforms Stockfish diluted to 50% strength with random moves.

[7] I used the annotations like `[%eval #12]` that appear on moves for many games in the Lichess database. I didn't notice that they do not appear on a game-ending move like `Qh4#`! This does sort of make sense because the eval scores would have to be `[%eval #+0]` ("mate in 0") or `[%eval #-0]` (necessitating use of the floating point coprocessor) to express the winner, and there does not seem to be a natural way to express the definite value of stalemate.

fer function is generally best or close to best. The identity transfer function, which yields a simple linear model, is generally worst or close to worst. The sigmoid functions are all over the place. It is known that they are prone to vanishing gradients in deep networks, and I may simply have unfavorable hyperparameter settings for them. The experimental downshift2 function is generally bad, perhaps because its output is strictly positive or it has such a small dynamic range. Its shape also seems prone to the vanishing gradient problem. The small amount of nonlinearity introduced by plus64 does appear to give it a small edge over the identity, but its lack of an interesting derivative and the fact that it only produces a small number of output values are limiting. Importantly, the grad1 function—the centerpiece of the first third of this paper—performs decently on all problems. It clearly outperforms the linear models, despite being "linear."

It is also interesting to compare the flattened versions of the linear transfer functions. These are the computed (mathematically) equivalent single-layer linear models. For plus64 the flattened version is worse in all cases; the unflattened model is taking advantage of the discretization in some way. For grad1 it is dramatically worse, both because grad1 models are substantially using the roundoff error and because the mathematical version of this function ($f(x) = x \times 1.036535$) is not even a good linear approximation of the actual result (e.g. $\texttt{grad1}(1) = 1$). Finally, the result for the identity transfer function should be mathematically equivalent, but it does not always produce the same results. This is unsurprising since we know that floating point calculations are not perfectly accurate, but it does hint that deep networks may make use of floating point roundoff internally, even if they are not using silly transfer functions!

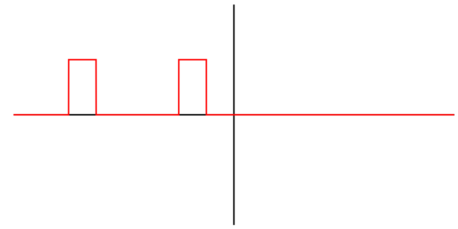
Having proved the professor wrong, we could stop there, but did huge mathematical breakthroughs ever arise from taking the option to *stop there*?!

5 Non-monotonic functions

Because of the way that addition and scaling are defined (do the real mathematical operation, then round), they preserve monotonicity: If $x \geq y$, then $f(x) \geq f(y)$. But this is only true if we limit the form of the function to a series of additions of constants and (non-negative) scaling. There are other expressions that are mathematically linear but don't take that form; for example:

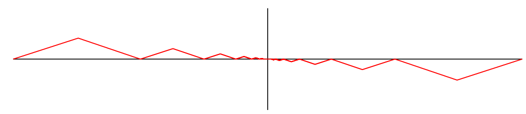
$$f(x) = x - 4096 - x + 4096$$

This is of course mathematically equivalent to $f(x) = 0.0$, but with half precision it is a square wave function (here pictured [-8, 8]):



For some values of x the terms cancel out, and for others the rounding error compounds. This function is not as well-behaved as it appears; the first pulse has width 0.99609375 and the second has width 1.

Here is $f(x) = \texttt{grad1}(x) - x$, which is also linear:



Generally speaking, we can create a large variety of functions by computing the interference patterns between other functions, since the sum or difference of two "linear" functions is also "linear." In general we'll consider expressions of this form:

$$\begin{aligned} E \quad ::= &\quad x \\ | &\quad E \times \texttt{c} \\ | &\quad E + \texttt{c} \\ | &\quad E + E \end{aligned}$$

Where x is the function variable, and c is one of the 63,488 finite half-precision constants. We can derive negation ($E \times -1$) and subtraction of constants ($E + -\texttt{c}$) and expressions ($E + (E \times -1)$) since every number has an exact negation by flipping its sign bit. Exact division is possible when $^1/_\texttt{c}$ is representable, and there is almost always a close approximation.

This formulation leads to a tempting approach for approximating a function iteratively, like a Taylor series. Given a target function like $\sin(x)$, we can begin with an approximate expression for it, like x, and then add and subtract terms to improve the approximation. I don't know of any systematic way to improve the approximation at each step (they are not well-behaved mathematically, and I am not good at math), but by using computer search I can sure make some complicated functions with many different shapes.

An approximation of sin appears in Figure 9. It is fun to watch an animation of the successively improving approximations, but you can't see that since you're reading an old-fashioned paper. Perhaps you can find a video of this at `tom7.org/grad`.

5.1 Fractals

Next, I endeavored to deploy these functions for something useful: Fractals. Famously, fractals are simple functions with complex (often literally) behavior. For example, the Mandelbrot set considers each complex point c (plotted on the plane as $x + yi$) and computes whether $z_i = z_{i-1}^2 + c$ diverges or not. It's lovely, but squaring is not linear!

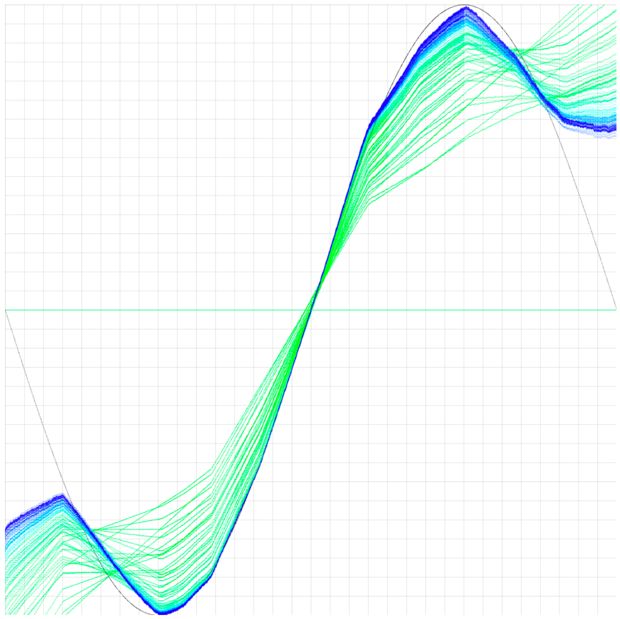


Figure 9: Successive approximations of the sin function, as color interpolates from green to blue.

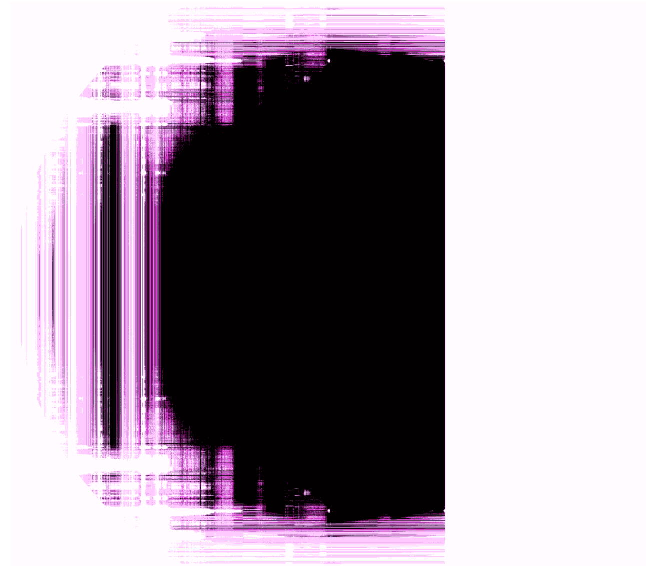
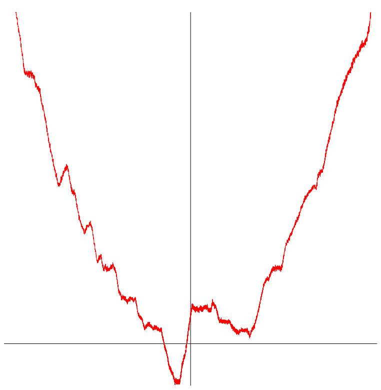


Figure 10: A garbage "fractal" that results from trying to approximate squaring of complex numbers using linear complex operations. Alas, it cannot be done. The complex numbers are truly special.

What if we just create a linear function that approximates $f(x) = x^2$? This is definitely possible, using the approach described above. After 184 successful error-reducing rounds we get the following approximation, with 112,204 linear operations:



Aside from the funny business near the origin, this is a fairly accurate approximation of the square function, so you might hope that it would draw a perverted Mandelbrot set. Unfortunately, it produces a much sadder blotch (Figure 10). To see why, consider the normal definition of squaring for a complex number:

$$\begin{aligned}(a + b\mathrm{i})^2 &= a^2 + 2ab\mathrm{i} + b^2\mathrm{i}^2 \\ &= a^2 + 2ab\mathrm{i} - b^2\end{aligned}$$

Note that the real coefficient a ends up part of the imaginary coefficient $2ab$ in the result, and the imaginary coefficient b becomes part of the real part (because i^2 is real). This means that squaring a complex number cross-pollinates between the two components, yielding a kind of wacky rotation if we think of them as 2D coordinates.

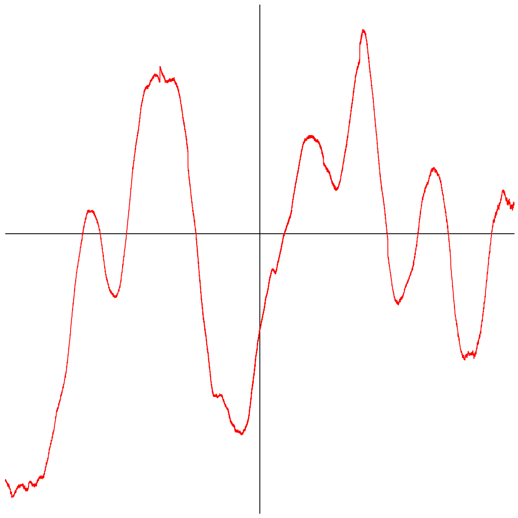
But here, squaring is approximated as a series of operations of the form $w_1 + w_2$ and $w \times c$ for constants c. These operations on complex numbers are less interesting:

$$\begin{aligned}(a_1 + b_1\mathrm{i}) + (a_2 + b_2\mathrm{i}) &= (a_1 + a_2) + (b_1 + b_2)\mathrm{i} \\ (a + b\mathrm{i}) \times c &= ac + bc\mathrm{i}\end{aligned}$$

Alas, these operations are boring; the real parts always stay real and the imaginary parts always stay imaginary. This is why the crummy blotch has all sorts of vertical and horizontal lines in it: As we iterate the function we are iterating two independent components, and the resulting picture is just some interference pattern between them.

This seems pretty definitive. Even if we had some kind of hardware implementation of complex numbers with rounding error to abuse, there would be no reason to have the linear operations do any cross-pollinated rounding. Professors take note: The complex numbers do provide some refuge!

Still, a lot of chaos can emerge from these functions that should not be possible with "linear" ones. For example, here is a complicated function made by stringing 36,637 addition and scaling operations together:



Iterating this function produces chaotic results because of its nonmonotonicity. In Figure 11 I plot (using color) the magnitude of z after 256 iterations of

$$z_i = f(z_{i-1}) \times c$$

This is mathematically linear (as c is a constant and f a linear function). Nonetheless, it produces an appealing picture. I think this is a fractal in the sense that it is chaotic, has a color gradient, and could be on the cover of an electronic music album. It is not a fractal in the sense that if you zoom in on it, you get infinite detail of self-similar shapes. In fact, if you zoom in on it only a modest amount, you encounter rectangular pixels as you reach the limits of half-precision floating point. (And because this fractal is built by abusing those very limits, it is not even possible to get more detail by increasing the accuracy!)

5.2 Bonus digression: Baffling numbers

Imagine you are my professor. You assign a class project to "make fractals using floating point roundoff error," for some reason. You spot me in the computer lab and I'm obviously way off track, because on-screen is some kind of *3D fractal*. The Mandelbrot set cannot be extended to three dimensions, you say, because of the Frobenius theorem: Only algebras of dimension 1 (real numbers), 2 (complex numbers) and 4 (quaternions) work [8]. Unclear how the professor speaks the citation aloud in this scenario. I say I "know" this fact, but I "don't care." You say that my three-dimensional algebra can't be associative, because that's "just a mathematical fact." I say you know what else isn't associative? *The floating point numbers, my dude.*

Enter the **baffling numbers**, ill-advised extensions of the complex numbers to three dimensions. Here we have numbers of the form $a + b\mathrm{i} + c\mathrm{j}$. Addition is just pointwise, and there are several options to complete the story for multiplication, namely the values of the cells A, B, and C in this table:

×	1	i	j
1	1	i	j
i	i	−1	U
j	j	V	W

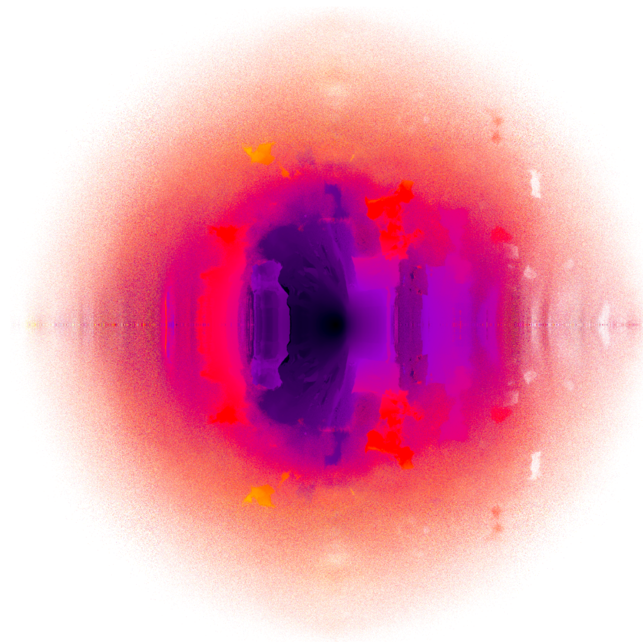


Figure 11: A fractal made from iterating a "linear" function f. The color is the magnitude of z_{256} with $z_i = f(z_{i-1}) \times c$. c is the complex coordinate $x + y\mathrm{i}$, a constant.

The cells U, V, and W are baffling numbers (i.e. each some $a + b\mathrm{i} + c\mathrm{j}$). Some choices are degenerate, but this gives us a family of options. It is known that no matter the choices, this does "not work" (in the sense that the resulting algebra is not associative[8]) but we don't need associativity to draw fractals. Plus, who's gonna stop me, FROBENIUS??

I tried a few options, but thought that $U = \mathrm{i}$, $V = \mathrm{j}$ and $W = 1$ produced satisfyingly trippy results. The Mandelbrot is straightforwardly generalized to the "Bafflebrot" (the starting point c is just a baffling number now; everything else is the same). I generated a 3D object by defining an implicit surface based on whether a sampled point is still inside the set after 25 iterations, using Marching Cubes [17] to discretize it. The resulting mesh is 2 gigabytes and crashes every library that attempts to programmatically simplify it. I do admire and encourage its defiant spirit. A rendering appears in Figure 12.

Drawing fractals is fun and everything, but I grew weary of the exercise because there is no real goal other than to make a cool picture. Instead I turned to something with a clearer challenge to overcome: Linear Cryptography.

6 Linear cryptography

Cryptography is like fractals minus drugs. One of the most basic components of cryptography is a pseudorandom num-

[8]Or else is equivalent to the complex numbers.

Figure 12: The 3D "bafflebrot" sliced in half and projected to 2D. This fractal was created with the "illegal" number system called the *baffling numbers.* They're like the complex numbers but more so. The object is truncated along its j axis, showing a perfect ripe Mandelbrot inside.

ber generator. This kind of function takes some state and produces a new state that "looks random." Given a pseudorandom number generator, we can construct one-way functions ("hash functions") and from those we can make symmetric ciphers (using, say, a Feistel network), with which we can encrypt and decrypt data.

Another thing that professors will tell you about cryptography is that good cryptographic functions cannot be linear. In this context, linear includes in a finite ring like $\mathbb{Z}_{256}$ or (especially) $\mathbb{Z}_2$, i.e. bits.[9] One good reason for this is that even if the function is *a little bit linear* then linear cryptanalysis can be used to recover bits of the key with a lot of example data [19]. Standard advice is to alternate both linear (e.g. XOR, or multiplication mod 2^n) and non-linear (e.g. substitution) operations. ("[Substitutions] are generally the only nonlinear step in an algorithm; they are what give a block cipher its security."[10]) Of course we will prove this adage wrong by developing a good pseudorandom function that uses only "linear" operations on half-precision floating point numbers.

In terms of goals, pseudorandom number generation has a more clear objective than fractals, although it's not so easy to pin down formally. We don't even know if such functions exist, mathematically [10], although there are generators that are provably secure assuming some other problems are actually hard [5] (but these problems are only *believed* to be hard). There exist many functions that look like good pseudorandom generators, but that actually have back doors that make them easy to predict. (Iteration of

[9] So XOR is considered linear here, even though we previously observed that there is no linear function on real numbers that fits it!

[10] *Applied Cryptography*, Second Edition, page 349 [31].

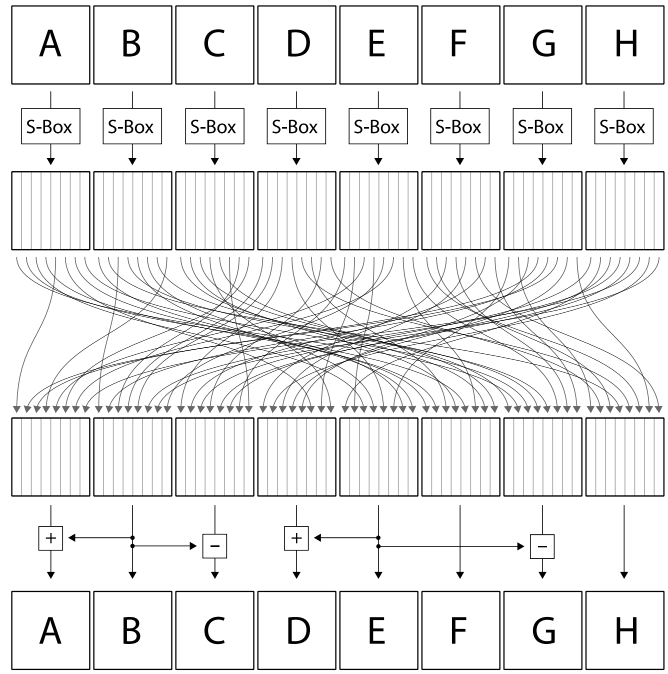


Figure 13: The substitution-permutation network that forms a half decent pseudorandom number generator. The same substitution ("s-box") is applied to each byte. Then the 64 bits are permuted. Finally, bytes are combined with modular addition and subtraction. This function passes the "Big Crush" suite and can be implemented with only half-precision floating point addition and scaling.

a symmetric encryption algorithm like AES, with the key hidden, has this property.)[11] Practically speaking, though, we can subject the function to a wide variety of statistical tests, and if it looks random to every test, then this gives us good confidence.[12]

Specifically, my goal is to design an algorithm that takes 64 bits of data (represented as half-precision floats) to another 64 bits, such that the stream of low-order bits from iterating this function passes the TestU01's "Big Crush" suite of 106 statistical tests [16]. This suite is a successor to Marsaglia's "DieHard" battery of tests [18], itself an improvement on Knuth's tests from The Art Of Computer Programming [13].

The basis of this function is the classic substitution-

[11] And let us never forget that RSA DSI (yes, that RSA) actually did take a $10 million bribe from the NSA to put a backdoor in one of their pseudorandom number generators [22]!

[12] Truly good cryptographic algorithms are also openly studied by experts. Of course nothing in here is to be used seriously, and not just because these algorithms are ridiculously slow. But I guess if you are stuck on a desert island with only the floating point addition and scaling operations, and a copy of this paper, then it would be a reasonable starting point for encrypting your messages. I do not recommend, if stranded on a desert island, to send encrypted messages: They may not be readable to your potential rescuers!

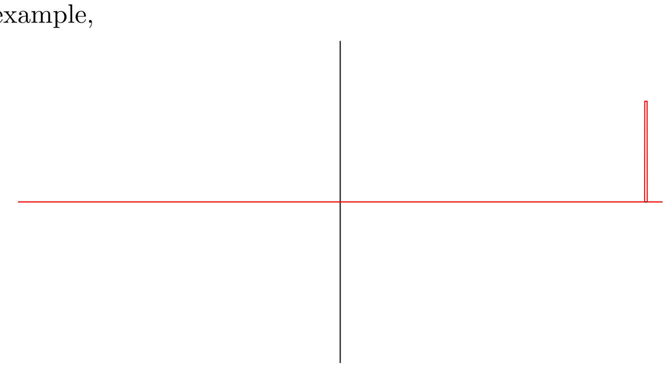
permutation network. First, each of the eight bytes are substituted with a different byte using a table (this is the mathematically non-linear step). Then, the 64 bits are permuted. Finally, some of the bytes are modified additively. An illustration appears in Figure 13.

The substitution table ("s-boxes") was generated by computer search with an objective to maximize the "avalanche property" (when a bit of the input is complemented, about half of the output bits should be complemented). The permutation was generated to maximize dispersion; each quartet sends each bit to a distinct quartet in the output. This is not the important part. We could have just used known good tables.

To implement this with half-precision floating point, we could represent each bit with its own half, but that is no fun. The state will be represented with eight half-precision floats, each representing one byte's worth of information. Since we have been fixated on the $[-1, 1]$ interval so far, a byte will be stored as any value in $[-1, 1)$, with each $^1/_{128}$ interval representing one of the 256 values (0 is anything in $[-1, -0.9921875)$, 1 is anything in $[-0.9921875, -0.984375)$, and so on). This means that iterating the function on *any starting value* in the $[-1, 1)$ interval will produce pseudorandom results. So for example we can guarantee that a "fractal" plotted using this function will look "fully messed up" and not just have a few distinguished points of randomness. I'll say now that this is unnecessarily hard; in the next section of this paper we'll see a vastly more efficient approach for handling discrete data. But working on the entire domain makes for some challenging problems and shows that we've developed substantial mastery of the continuous case.

Speaking of which, my first approach was to try to approximate the substitution function (since it replaces one 8-bit byte with another, it corresponds to a single discontinuous function of type $half \to half$) using the iterative approach described in Section 5.1. Although it is possible to get reasonable approximations with this method (most values are transformed to a value near the desired one), this will not suffice; when iterating the function we find that the value easily gets stuck in short cycles due to this inaccuracy.

I found a better approach, by creating a composable family of functions that isolate specific intervals of interest. For example,



Within the interval $[-1, 1)$, this function takes on exactly two values: zero[13] and $^1/_{128}$. It returns $^1/_{128}$ only for exactly the interval $[^{121}/_{128}, ^{122}/_{128})$. This is the interval that represents the number 249 $(128 + 121$; remember that the first 128 integers are in $[-1, 0))$. The expression that computes this is

$$f(x) = \left(\begin{array}{l} (x - \frac{9}{64} - \frac{1}{4} \times \frac{-1}{512} - \frac{1255}{512} \times^{164} \frac{1027}{1024}) + \\ (x - \frac{1}{4} \times \frac{-1}{512} - \frac{597}{512} \times^{188} \frac{1277}{2048} \times -1) + \\ \frac{517}{128} \times -32 \end{array} \right) + \left(\begin{array}{l} (x - \frac{17}{128} - \frac{1}{4} \times \frac{-1}{512} - \frac{1255}{512} \times^{164} \frac{1027}{1024}) + \\ (x - \frac{1}{4} \times \frac{-1}{512} - \frac{597}{512} \times^{188} \frac{1277}{2048} \times -1) + \\ \frac{517}{128} \times 32 \end{array} \right) \times \frac{-1}{32}$$

where $E \times^n c$ means $E \times c \times c \times c \ldots$ for n iterations. Mathematically this is equivalent to this constant function (all xs cancel out):

$$f(x) = \frac{13^{164} \times 79^{164}}{2^{1656}}$$

I spent a long time writing code to simplify these expressions and generate LaTeX for them, by the way! As usual, I thought it would look cool when I got it working, but it just looks like a bunch of numbers.

We can think of this function as a basis vector, representing the 256-dimension vector $\langle 0, 0, 0, \ldots, 0, 1, 0, 0, 0, 0, 0, 0 \rangle$. We'll call this one $\mathbf{b}_{249}$ since it selects the integer 249. If we can find $\mathbf{b}_n$ for each $n \in \mathbb{Z}_{256}$, then we will be able to combine them to systematically construct functions.

The one just pictured is one of the smallest expressions; most are much larger. I wish I could tell you that I figured out the principles underlying how to analytically generate these functions, but I discovered them with computer search and some elbow grease.

Choppy functions. I call a function f "choppy" if for every half precision floating point value in $[-1, 1)$ it has the following properties:

- For $n \in \mathbb{Z}_{256}$ and $r \in [0, 1)$, $f(-1 + ^{n+r}/_{128}) = v$ for the same value v. We only need to consider cases where $n + r$ is representable as a half.
- v is itself of the form $^{n-128}/_{128}$ for some $n \in \mathbb{Z}_{256}$.
- For these purposes, we treat the single value -0 as being equal to 0.
- And, as usual, the function is built only with floating point addition and scaling by constants.

That is, the function produces the same result for any representation of an integer, and that result is the smallest representation of an integer. These functions are maximally useful in that they are "liberal in what they accept,"

[13] Actually, -0!

but "conservative in what they return" [28]. It also means that each function also can be understood as a function $\mathbb{Z}_{256} \to \mathbb{Z}_{256}$, so we can represent them as a vector of 256 integers. The basis vectors $\mathbf{b}_n$ are those that are of the form $\langle 0, \ldots, 1, \ldots, 0 \rangle$.

I then conducted computer search for choppy functions, putting those into a database (keyed by the corresponding integer vector). Some are easy to find, others harder. Summing and scaling choppy functions yield choppy functions (as long as the vectors remain integral and in range), so I use a simplified version of Gauss-Jordan elimination [32] to solve for basis vectors. Once I have $\mathbf{b}_n$, this column can be changed at will for any existing choppy function (by just adding or subtracting multiples of $\mathbf{b}_n$), so new choppy functions that only vary in that column can be ignored.

By trying a variety of operations that are known to be useful (e.g. iterated multiplication of constants near 1.0) and hill-climbing towards functions with the choppy property, it is not too hard to find $\mathbf{b}_n$ for most n. It seems to become more challenging for n near 128; this is the point 0.0 in half-precision. Specifically, the hardest problem was to make a function that produced different results for inputs < 0.0 versus inputs ≥ 0.0. This is the *zero-threshold problem*.

Why is this hard? Distinguishing between negative and non-negative numbers is deceptively difficult. Looking back to the function $f(x) = x + 128.0 - 128.0$ (Section 4), it has useful discontinuous steps, but note that the discontinuity does not happen at zero. This is because we are rounding to the nearest value, and so small negative numbers near zero end up rounding to the same result that zero does. Moreover, the resolution of the floating point numbers is highest near zero (especially because of subnormal numbers), which exacerbates our attempts to control rounding of them. For example, you might think that we could simply shift this function left and right by substituting $x + c$ for x in its body. This would work mathematically, but it does *not* work for floating point numbers, because each operation performs some rounding. If this rounding ever ends up conflating a negative number with a non-negative one, we will not be able to recover.

I found a zero-threshold function using a combination of manual and computer search. This was some ordeal, and the resulting enormous function is in Figure 14. Perhaps you are smarter than me and can find a better one!

Substituting and permuting. In any case, with this function it was possible to form a complete basis. This basis makes it "easy" to perform operations on half values that represent bytes. For example, the s-box step substitutes some distinct byte for each different input byte. This would normally be implemented with a table lookup. If we compute $\mathbf{b}_n(x) \times \text{subst}[n]$, this returns the correct[14] result $\text{subst}[n]$ if the input $x = n$, and 0 otherwise. So if we just sum all 256 of these up, exactly one of them will be nonzero, and the correct substituted value.

Permutation is defined on the component bits. Here, we compose a function that computes each of the eight output bytes. We use the same approach of summing a bunch of $\mathbf{b}_n(x)$ evaluations (each multiplied by the correct answer). Here we are testing whether the input has some particular bit set (a sum of the 128 $\mathbf{b}_n(x)$ functions where n has that bit set), and the output is the power of two that sets the appropriate output bit. Many of these functions would have simpler implementations (for example, "is the high-order bit set?" is the same as the zero-threshold function) but at this point I was happy to just have something working, and taking some joy in how absurdly large the functions were getting.

The cipher also includes addition and subtraction mod 256. Addition and subtraction are already available for half-precision floats, and they have faithful behavior, so we just need to implement the wrapping-around behavior so that the result is strictly in $[-1, 1]$. This is straightforward with the zero threshold function;[15] we produce corrective factors if the result is ≥ 1 or < -1 (zeros otherwise). We then add those corrective factors produce the remainder we desire.

6.1 Benchmark results

To evaluate the quality of the pseudorandom number generator, I used the TestU01 "Big Crush" suite. This test needs a sample of 1.64 billion bits, so I actually evaluated it on equivalent code that performs the steps using normal integer operations. Even then, the suite takes several days to run, so I modified it to run tests in parallel and cache the results of completed tests. This saved me from losing data if my computer crashed or needed to be rebooted.

Results appear in Figure 15. Passing these tests does not ensure that the pseudorandom number generator is good for cryptography, although it is a good start.

Running single-threaded on a 3.0 GHz Threadripper 2990WX, this function generates 25.8 bytes of randomness per second, which is slow. By precomputing the substitution, permutation, and zero threshold expressions (so they can be performed by lookup into 64k-entry tables), it generates 18,685.2 bytes per second, which is still slow.

If we were building an encryption algorithm (a symmetric block cipher), it would be natural to use this as its "round function." In a Feistel network [7], each input block (128 bits) is broken into two halves; one of them is mixed with some key bits (for example with XOR) and then passed to this function. Its output is XORed with the other half; the two halves are swapped, and this "round" is repeated many times until we believe that the data are suitably screwed up. Decryption is the reverse. We can use addition and subtraction mod 2^8 to combine the data instead of XOR

[14] Technically we need to do some multiplicative adjustments to put the value in $[-1, 1)$.

[15] Compare to the remarks "why is this hard?" above. Here, $zt(x - 1)$ *does* do what you'd want, shifting the threshold value from 0 to 1. This is because there is *less* precision near one than near zero.

$$zt(x) = x + \frac{1}{2^{24}} \times^{9743} \frac{1025}{1024} \times \frac{39}{2^{19}} + \frac{1279}{16384} \times 4100 + 318 \times 4 + 1 \times^{559} \frac{1025}{1024} \times \frac{1027}{2048} \times^{1160} \frac{1025}{1024} \times \frac{545}{2048} \times^{23} \frac{1025}{1024} \times \frac{311}{512} \times^{137} \frac{1025}{1024} \times \frac{527}{2048} \times^{365} \frac{1025}{1024} \times \frac{627}{1024} \times^{346} \frac{1025}{1024} \times \frac{593}{1024} \times^{676} \frac{1025}{1024} \times \frac{281}{512} \times^{557} \frac{1025}{1024} \times \frac{129}{256} \times^{830} \frac{1025}{1024} \times \frac{589}{4096} \times^{336} \frac{1025}{1024} \times \frac{1029}{2048} \times^{663} \frac{1025}{1024} \times \frac{1041}{2^{18}} - 2206 + 2206 \times \frac{9}{256} + 1076 + 2534 - 2074 \times \frac{17}{64} - 2048 + 2048 \times \frac{1}{8}$$

$$=$$

$$\frac{593 \times 2187 \times 5^{30793} \times 343 \times 11 \times 169 \times 289 \times 361 \times 961 \times 41^{15396} \times 43 \times 79 \times 109 \times 281 \times 311 \times 347}{2^{154102}} x +$$

$$\frac{\begin{array}{l} 84107227537103367748705454682078539303191039250504832410385579895977259206417678 \\ 54245560547677512932803511681703341508665266147325419821191498222214690318490003 \\ 88110778422408950986678118366271064412982414738166383752334216371785576710459496 \\ 25738831829937485787963475192987844674323695457715738688221958462470493961327089 \\ 64862528034085403084792949523917534005532171250047637347672007635216035917044700 \\ \ldots 569 \text{ lines} \ldots \\ 73014930689000221762919045089322540125964944324282583780813532524840229888776299 \\ 45268638388603643723804098205965854510116202420980541689175292145265852612920173 \\ 68725838005728517370192463512280524432138902703991548800398876262333592383735651 \\ 92764023532792804235216160403774463302046032255421688296905932246680375562746379 \\ 76285400623707503241570527062342946483273904883074176883724035214883942317941586 \\ 85212883934224294828781615577153021816836375301250331354703790141535016001286984 \\ 6924658790514087860460295763717839867930071462923835232043690587364999 3209 \times 3 \times 17 \times 421 \end{array}}{2^{154126}}$$

Figure 14: A zero-threshold function. Returns $^1/_{128}$ for values in $[0, 1)$ (and -0) and 0 for values in $[-1, 0)$. Top is the series of additions and scalings to perform, all from left to right. At bottom is the equivalent mathematical expression, but the enormous numerator cannot be printed due to extremely oppressive SIGBOVIK page limitations.

(which is addition mod 2^1), so we already have all the operations we need to build a whole block cipher here. As Bruce Schneier says,[16] "It is easy to design a block cipher."

7 THE ULTIMATE THROWBACK

Having developed a basis for extracting arbitrary bits, we can express any function of a single variable, and we've seen how some other functions (like addition mod 2^8) can be done. At this point, it seems like we probably have the building blocks to demonstrate that addition and scaling on half-precision floats is Turing complete. I mean, pretty much everything is Turing complete. In the past, I built computers that were perfect and beautiful, such as a hardware implementation of the NaNDY 1000, a computer architecture that computes using only floating point NaN and Infinity [27]. In a concession to ideological purity, though, the NaNDY 1000 has no I/O. So it is very boring to use.

For today's investigations of the capabilities of floating point, I'll make the opposite concession: Let's make a computer that is exciting to use, but that makes some (reasonable) ideological concessions so that it can do something interesting.

[16] *Applied Cryptography*, Second Edition, page 351.

7.1 Fluint8

First of all, if we want to do some serious computation, 25.8 bytes per second isn't going to cut it. To look for performance enhancing substances, I perused the back catalog of the world's most prestigious conference, SIGBOVIK. There in the 2018 edition, on page 125, I found an intriguing paper, *The fluint8 Software Integer Library*, by Drs. Jim McCann and ... Tom Murphy VII? Wait, that's me? *I already wrote this paper?!* [20]

The *fluint8* library represents an element of $\mathbb{Z}_{256}$ (a.k.a. `uint8`) as a 32-bit float, and provides multiplication, addition, subtraction, negation, division, and bitwise functions and, or, and exclusive or.

Compared to the approach discussed in Section 6 using "choppy functions," *fluint8* has much more simple and sensible implementations of functions like addition:

```
inline float fu8_add(float a, float b) {
  float x = a + b;
  x -= x - 127.5f + 3221225472.0f - 3221225472.0f;
  return x;
}
```

The `x -= x...` line applies the corrective factor to implement wrap-around, which we previously did using the zero-threshold function. Why can it be done so much more simply here? First, *fluint8* represents $n \in \mathbb{Z}_{256}$ as n, so a number like 27 is represented as 27.0 instead of, say, $-1 + {}^{27}/_{128}$. Second, it requires that the number be represented *exactly* as this value. The figures in the *fluint8* paper are somewhat misleading as they are plotted only

Test	p-value
SerialOver, r = 0	0.9653
SerialOver, r = 22	0.7292
CollisionOver, t = 2 (0)	0.3890
CollisionOver, t = 2 (9)	0.6537
CollisionOver, t = 3 (0)	0.8046
CollisionOver, t = 3 (16)	0.9279
CollisionOver, t = 7 (0)	0.2906
CollisionOver, t = 7 (24)	0.0031
CollisionOver, t = 14 (0)	0.4310
CollisionOver, t = 14 (27)	0.5062
CollisionOver, t = 21 (0)	0.1909
CollisionOver, t = 21 (28)	0.2906
BirthdaySpacings, t = 2	0.4179
BirthdaySpacings, t = 2 (b)	0.5749
BirthdaySpacings, t = 3	0.2249
BirthdaySpacings, t = 4	0.2230
BirthdaySpacings, t = 4 (14)	0.2230
BirthdaySpacings, t = 4 (0)	0.2293
BirthdaySpacings, t = 4 (16)	0.9111
BirthdaySpacings, t = 7 (0)	0.8077
BirthdaySpacings, t = 7 (7)	0.4887
BirthdaySpacings, t = 8 (14)	0.5956
BirthdaySpacings, t = 8 (22)	0.1382
BirthdaySpacings, t = 16 (0)	0.5266
BirthdaySpacings, t = 16 (26)	0.6619
BirthdaySpacings, t = 13 (0)	0.8419
BirthdaySpacings, t = 13 (5)	0.9242
BirthdaySpacings, t = 13 (10)	0.3125
BirthdaySpacings, t = 13 (15)	0.4234
BirthdaySpacings, t = 13 (20)	0.0172
BirthdaySpacings, t = 13 (26)	0.3276
ClosePairs NP t=3	0.9584
ClosePairs mNP t=3	0.6028
ClosePairs mNP1 t=3	0.3668
ClosePairs mNP2 t=3	0.8549
ClosePairs NJumps t=3	0.3739
ClosePairs mNP2S t=3	0.3813
ClosePairs NP t=5	0.2328
ClosePairs mNP t=5	0.4011
ClosePairs mNP1 t=5	0.6286
ClosePairs mNP2 t=5	0.7635
ClosePairs NJumps t=5	0.7981
ClosePairs mNP2S t=5	0.4369
ClosePairs NP t=9	0.3073
ClosePairs mNP t=9	0.7986
ClosePairs mNP1 t=9	0.1934
ClosePairs mNP2 t=9	0.5857
ClosePairs NJumps t=9	0.9882
ClosePairs mNP2S t=9	0.0962
ClosePairs NP t=16	0.3787
ClosePairs mNP t=16	0.1983
ClosePairs mNP1 t=16	0.0511
ClosePairs mNP2 t=16	0.2874
ClosePairs NJumps t=16	0.9369
ClosePairs mNP2S t=16	0.7523
SimpPoker 0 8	0.9863
SimpPoker 27 8	0.4528

Test	p-value
SimpPoker 0 32	0.8052
SimpPoker 25 32	0.1166
CouponCollector, r = 0	0.3233
CouponCollector, r = 10	0.7936
CouponCollector, r = 20	0.2870
CouponCollector, r = 27	0.1878
Gap 0 16	0.2858
Gap 25 32	0.6202
Gap 0 128	0.7462
Gap 20 1024	0.1068
Run 0	0.3096
Run 15	0.5308
Permutation 3	0.7322
Permutation 5	0.8632
Permutation 7	0.8337
Permutation 10	0.7557
CPerm 0	0.0512
CPerm 10	0.0116
MaxOft, t = 8	0.1909
MaxOft AD, t = 8	0.6478
MaxOft, t = 16	0.3601
MaxOft AD, t = 16	0.7570
MaxOft, t = 24	0.3625
MaxOft AD, t = 24	0.7378
MaxOft, t = 32	0.4541
MaxOft AD, t = 32	0.1967
SampleProd, t = 8	0.6129
SampleProd, t = 16	0.7735
SampleProd, t = 24	0.0891
SampleMean, r = 0	0.1115
SampleMean, r = 10	0.4571
SampleCorr, k = 1	0.0260
SampleCorr, k = 2	0.0146
AppearanceSpacings, r = 0	0.6741
AppearanceSpacings, r = 27	0.0951
WeightDistrib, r = 0 (0.25000)	0.3097
WeightDistrib, r = 20 (0.25000)	0.6266
WeightDistrib, r = 28 (0.25000)	0.4372
WeightDistrib, r = 0 (0.06250)	0.6148
WeightDistrib, r = 10 (0.06250)	0.6600
WeightDistrib, r = 26 (0.06250)	0.6532
SumCollector	0.6092
MatrixRank, L=30, r=0	0.4367
MatrixRank, L=30, r=25	0.3045
MatrixRank, L=1000, r=0	0.0841
MatrixRank, L=1000, r=26	0.0145
MatrixRank, L=5000, r=15	0.2650
MatrixRank, L=5000, r=0	0.9631
Savir2	0.7317
GCD	0.7578
RandomWalk1 H (L=50, r=0)	0.6779
RandomWalk1 M (L=50, r=0)	0.9338
RandomWalk1 J (L=50, r=0)	0.3168
RandomWalk1 R (L=50, r=0)	0.4753
RandomWalk1 C (L=50, r=0)	0.4941
RandomWalk1 H (L=50, r=25)	0.8645
RandomWalk1 M (L=50, r=25)	0.6220

Test	p-value
RandomWalk1 J (L=50, r=25)	0.4576
RandomWalk1 R (L=50, r=25)	0.9736
RandomWalk1 C (L=50, r=25)	0.6768
RandomWalk1 H (L=1000, r=0)	0.9915
RandomWalk1 M (L=1000, r=0)	0.8194
RandomWalk1 J (L=1000, r=0)	0.7606
RandomWalk1 R (L=1000, r=0)	0.4983
RandomWalk1 C (L=1000, r=0)	0.0529
RandomWalk1 H (L=1000, r=20)	0.3353
RandomWalk1 M (L=1000, r=20)	0.2279
RandomWalk1 J (L=1000, r=20)	0.8593
RandomWalk1 R (L=1000, r=20)	0.4915
RandomWalk1 C (L=1000, r=20)	0.9640
RandomWalk1 H (L=10000, r=0)	0.0713
RandomWalk1 M (L=10000, r=0)	0.4753
RandomWalk1 J (L=10000, r=0)	0.6421
RandomWalk1 R (L=10000, r=0)	0.0469
RandomWalk1 C (L=10000, r=0)	0.6232
RandomWalk1 H (L=10000, r=15)	0.5739
RandomWalk1 M (L=10000, r=15)	0.7165
RandomWalk1 J (L=10000, r=15)	0.6868
RandomWalk1 R (L=10000, r=15)	0.7075
RandomWalk1 C (L=10000, r=15)	0.2100
LinearComp, r = 0 (Num)	0.7964
LinearComp, r = 0 (Size)	0.8628
LinearComp, r = 29 (Num)	0.1564
LinearComp, r = 29 (Size)	0.9696
LempelZiv, r = 0	0.8373
LempelZiv, r = 15	0.4632
Fourier3, r = 0	0.9159
Fourier3, r = 27	0.8144
LongestHeadRun (Chi), r = 0	0.1270
LongestHeadRun (Disc), r = 0	0.9025
LongestHeadRun (Chi), r = 27	0.7822
LongestHeadRun (Disc), r = 27	0.6878
PeriodsInStrings, r = 0	0.6300
PeriodsInStrings, r = 20	0.0839
HammingWeight2, r = 0	0.1331
HammingWeight2, r = 27	0.0322
HammingCorr, L = 30	0.5516
HammingCorr, L = 300	0.7373
HammingCorr, L = 1200	0.9393
HammingIndep, L=30, r=0	0.1326
HammingIndep, L=30, r=27	0.7257
HammingIndep, L=300, r=0	0.4177
HammingIndep, L=300, r=26	0.7630
HammingIndep, L=1200, r=0	0.4981
HammingIndep, L=1200, r=25	0.3571
Run of bits (runs), r = 0	0.0241
Run of bits (bits), r = 0	0.5718
Run of bits (runs), r = 27	0.0822
Run of bits (bits), r = 27	0.7981
AutoCorr 1 0	0.3058
AutoCorr 3 0	0.0371
AutoCorr 1 27	0.3292
AutoCorr 3 27	0.0612

Figure 15: Results of the TestU01 “Big Crush” suite on the pseudorandom number generator built from floating point roundoff error. A p-value of < 0.001 or > 0.999 is considered suspect by the suite, so all tests pass here.

for input values that are already exact integers; if we test `fu8_add` on values like 100.1875 and 11.0703125 we do not get 111 (Figure 16). On the other hand, this is a very reasonable choice to make; we can simply have a representation invariant that only one of these 256 values is used, and preserve that invariant with every operation. It won't work great for the continuous domain (e.g. plotting fractals) but is a much more practical choice for discrete data (e.g. encryption). Since I like to work at the intersection of Theory, Impractice, and Practice, this is appealing!

But: The library uses several operations that are not linear! In particular, its implementation of bitwise functions like XOR perform squaring and multiplication of the two arguments. It was not a design goal of *fluint8* to use only addition and scaling, but it is a design goal today, so we must address that.

7.2 hfluint8

The use of nonlinear operations is a problem we will rectify, forthwith, but the other ideas are suitable for building a computer. In the *hfluint8* (for half float linear unsigned int 8-bit) library, a `hfluint8` will be represented by a single half-precision floating point number, and always one of the exact integral values in [0, 256).[17]

```
struct hfluint8 {
  half h;
  // ...
};
```

Let's begin with one helper function:[18]

```
half RightShiftHalf8(half xh) {
  half SCALE = GetHalf(0x1c00);   // 1/256
  half OFFSET1 = GetHalf(0xb7f6);
  half OFFSET2 = GetHalf(0x66b0);
  return xh * SCALE + OFFSET1 + OFFSET2 - OFFSET2;
}
```

If the function is given an integral half `xh` in $[0, 512)$, it returns `xh >> 8`. This value is always 1 or 0. The calls to `GetHalf` interpret a 16-bit constant as a half, which is useful to be precise (many decimal expressions like 0.1 are not exactly representable in floating point). I also found that if you use literals like `0.00390625_h`, the code runs much much more slowly because it inhibits some optimizations or perhaps the user-defined literals are parsed at runtime (?!). Aside from wanting to avoid operations like parsing that might not be addition and scaling on halfs, we will struggle with performance of these functions as we use them for real

[17] In fact, all integers from -2048 to 2048 are available, so we could consider implementing signed 11-bit numbers in a future *hflsint11* library.

[18] These code samples have been simplified to fit the extremely capricious SIGBOVIK column width requirements. For example, `GetHalf` is a `constexpr` function, so these constants are really declared as `static constexpr` and computed completely at compile time. See the full code and verify that it complies with *the rules* at `https://sourceforge.net/p/tom7misc/svn/HEAD/tree/trunk/grad/`.

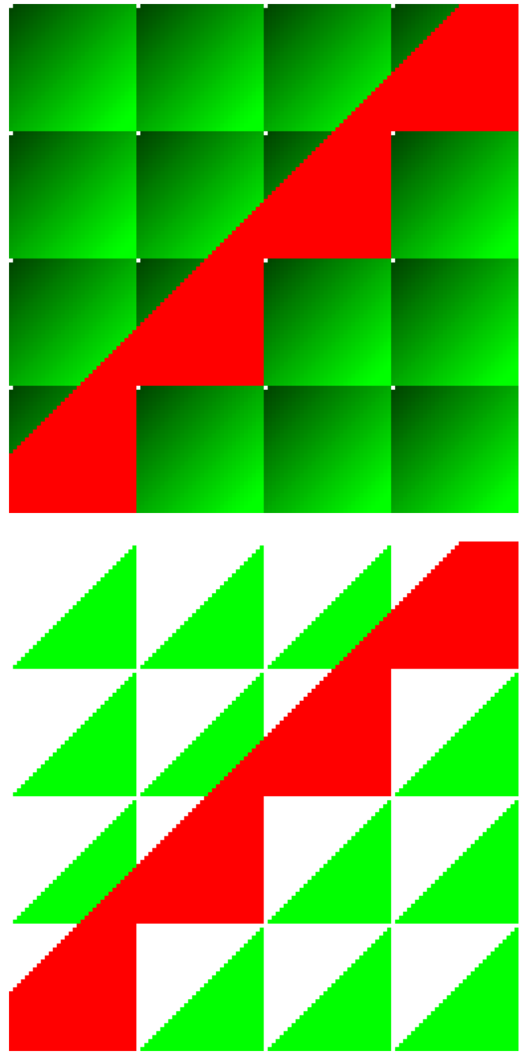


Figure 16: **Top**: Error of the *fluint8* addition function on general floating point values in [0, 256). This is a detailed zoom of the region $x \in [252, 256)$ and $y \in [0, 4)$, but the rest of the image is almost identical. Each pixel compares the *fluint8* sum of x and y to the expected value $(\lfloor x \rfloor + \lfloor y \rfloor \bmod 256)$. The top-left pixel in each cell is the case where x and y are integers; we get the correct result (no error). All other pixels are wrong, either too high (green) or too low (red).

Bottom: Same with the error of the *floor* of *fluint8*'s sum function. This shows that the output is usually not even in the correct interval. However, observe the multitude of Triforces!

Nowhere, or a lot of places if you think about it: The modular addition operation from Section 6 is not pictured for comparison because it would be all white, meaning no error. You can actually imagine it occupies any blank portion of this paper, such as the inner hole of a letter 'o,' or the entire back of a page if printed single-sided.

Graphics produced using `ImageRGBA` computational `for` loop engine.

computing. Anyway, we are just dividing by 256 (by multiplying by $^1/_{256}$) and then adding some mysterious constants to ensure that the result is exactly 1 or 0.

Next, we can perform addition:

```
hfluint8 hfluint8::Plus(hfluint8 x, hfluint8 y) {
  half HALF256 = GetHalf(0x5c00); // 256.0
  half z = x.h + y.h;
  half o = RightShiftHalf8(z);
  return hfluint8(z - o * HALF256);
}
```

As in *fluint8* we can simply add the arguments, giving a result in $[0, 512)$. The shift function just discussed then allows us to compute 1 if the value is out of range or 0 otherwise. We multiply this by a corrective constant (256.0) and subtract that away. So easy.

For all other operations we work on the domain $[0, 256)$. We also have a right shift by one bit:

```
half RightShiftHalf1(half xh) {
  half SCALE = GetHalf(0x37fa); // 0.4985...
  half OFFSET = GetHalf(0x66cd); // 1741.0
  return xh * SCALE + OFFSET - OFFSET;
}
```

Right shifting is integer division by two. Roughly we are dividing by two and then offsetting to a part of the floating point number line where only integers are representable, then offsetting back. However, with a constant of exactly 0.5 some of the rounding would be in the wrong direction; the constant 0.49853515625 just happens to work.

We can shift by multiple places by repeating this operation multiple times. However, the library has direct solutions for several other shift distances, since this is more efficient than repeating a single shift.

Next, bitwise operations. These are all based on the AND function:

```
half BitwiseAndHalf(hfluint8 a, hfluint8 b) {
  half result = GetHalf(0x0000);
  for (int bit_idx = 0; bit_idx < 8; bit_idx++) {
    // Low order bit as a - ((a >> 1) << 1)
    hfluint8 as = RightShift1(a);
    hfluint8 bs = RightShift1(b);
    half a_bit = a.h - LeftShift1Under128(as).h;
    half b_bit = b.h - LeftShift1Under128(bs).h;
    // Computes 2^bit_idx. A constant.
    half scale = GetHalf(0x3c00 + 0x400 * bit_idx);
    half and_bits = RightShiftHalf1(a_bit + b_bit);
    result += scale * and_bits;

    // and keep shifting down
    a = as;
    b = bs;
  }
  return result;
}
```

This function shifts each input down 8 times, stripping off the low order bit at each step. Note that since we run this loop exactly 8 times, it can simply be unrolled, removing any whiff of non-linearity, and the constants computed at compile time. `LeftShift1Under128(x)` is just $x + x$ without any need to worry about modular arithmetic, as it cannot overflow.

An interesting line is the computation of `and_bits`, which is the logical AND of the low-order bit from a and b. In *fluint8* we simply compute `a_bit * b_bit`. This has the correct value, but is not linear (observe that if we were to compute `x & x` we would end up squaring a function of `x` here). Instead we compute `(a_bit + b_bit) >> 1`, which produces the correct result.

Being able to compute the bits in common allows us to easily derive OR and XOR:

```
hfluint8 BitwiseOr(hfluint8 a, hfluint8 b) {
  half common = BitwiseAndHalf(a, b);
  return hfluint8((a.h - common) + b.h);
}

hfluint8 BitwiseXor(hfluint8 a, hfluint8 b) {
  half common = BitwiseAndHalf(a, b);
  return hfluint8((a.h - common) + (b.h - common));
}
```

These subtractions and additions cannot overflow.

It will be common to perform bitwise operations with constants, so *hfluint8* supports versions with a compile-time constant argument, which can skip a bunch of work. These run about 5× faster.

We also have some operations that are not supported by *fluint8* but that we will need for the current project. A basic operation is to test for zero. `IsZero` returns 1 if the input is 0, or returns 0 for any other argument:

```
hfluint8 IsZero(hfluint8 a) {
  half H255 = GetHalf(0x5bf8);  // 255.0
  half H1 = GetHalf(0x3c00);  // 1.0
  half nota = (H255 - a.h);
  return hfluint8(RightShiftHalf8(nota + H1));
}
```

For an input of zero, complementing it yields 255, and adding 1 overflows to set the 8th bit. So we shift that bit to the ones place and are done.[19]

With this, `Eq(a, b)` is just `IsZero(a - b)`. We can define a number of operations like "boolean or" that assume inputs of exactly 1 or 0; these are straightforward and much faster than their bitwise counterparts. We could think of these values as *hflbools*, although we still use the *hfluint8* type for them. The main way to use a *hflbool* is `If`. `If(cc, t)` returns `t` if `cc` is exactly 1, returns 0 if `cc` is 0, and is otherwise undefined. A simple implementation of this is:

```
  half H255 = GetHalf(0x5bf8);  // 255.0
  hfluint8 mask = hfluint8(cc.h * H255);
  return BitwiseAnd(mask, t);
```

This computes either the mask `00000000` or `11111111` and uses the existing bitwise AND operation. Bitwise AND is not fast, and it does more work than it needs to in this case because we know one of the arguments is all zeroes or all ones. It is faster to inline the bitwise AND routine but keep checking the ones place. Even better is this wild ride:

[19]Earlier iterations of this function were much more complex! For example, $x + 15 + 65248 - 65248 \times 0.03125$ maps 0 to 0, but any other number to some number in $[1, 15]$, and then a similar function compresses that range down to exactly 0 or 1. But sometimes you miss the obvious stuff until you start writing a paper about it for a prestigious conference. No doubt some other functions in here could be improved!

```
hfluint8 If(hfluint8 cc, hfluint8 t) {
  static std::array<half, 8> OFF = {
    GetHalf(0x77f9), GetHalf(0x7829),
    GetHalf(0x77fb), GetHalf(0x78e2),
    GetHalf(0x77fd), GetHalf(0x780b),
    GetHalf(0x77ff), GetHalf(0x7864),
  };

  half HALF1 = GetHalf(0x3c00);  // 1
  half HALF128 = GetHalf(0x5800);  // 128
  half HALFNEG1 = GetHalf(0xbc00);  // -1
  half HALF0 = GetHalf(0x0000);  // 0

  half xh = t.h;
  half nch = HALF1 - cc.h;
  half c128 = HALF128 * nch;

  std::array<half, 8> COFF;
  for (int i = 0; i < 8; i++)
    COFF[i] = OFF[i] * nch;

  for (const half &h : COFF) xh = xh + h - h;
  xh = (c128 - xh);
  for (const half &h : COFF) xh = xh + h - h;

  return hfluint8(xh * HALFNEG1 + HALF0);
}
```

The 8 constants in `OFF`, when added to and subtracted from a *hfluint8*, will always round such that the low six bits become 0. To have behavior conditional on *cc*, first we multiply each constant by $1 - cc$. This results in either the original constant or 0. If zero, then adding and subtracting them does nothing. Then we add and subtract those results, clearing the low six bits, and (conditionally, using the same trick of multiplying by the condition) subtract from 128. This clears the top two bits for the range of possible values (but may reset low-order bits). Then we add and subtract the sequence again, clearing the low six bits again. At the end we apply a corrective negation and then add 0 to avoid outputting -0 and we're done.

hfluint16. Several other operations are available for *hfluint8*, like `AddWithCarry`, but we shan't elaborate them all here, lest we contract hfluenza. One more concept is needed before we get to the application: 16-bit integers. The *hfluint16* type is implemented as a pair of *hfluint8* bytes. We will only need a small number of operations: Addition, subtraction, bitwise operations, sign extension of *hfluint8*, `If`, and stuff like that. These are all cleanly implemented in terms of the *fluint8* operations like `AddWithCarry`.

7.3 Linear gameplay

Now we can build an 8-bit computer. I like to work at the intersection of theory, impractice, practice, and entertainment, and the most entertaining 8-bit computer is the Nintendo Entertainment System, so let's build that. The full NES has many components (video output, controllers, sound, RAM, cartridge mappers), and it's not even clear what it would mean to implement "linear" versions of these. So for this project we will replace the CPU, which is a variant of the Motorola 6502 called the Ricoh 2A03. Each instruction that the CPU executes will be done entirely with linear half-precision floating point operations. This is done in software emulation, upgrading a version of the FCEUX Emulator [3] that I forked many years ago [23].

The 2A03 has 8-bit registers `A`, `X`, `Y`, a stack pointer `S` and processor flags `P`. Each is represented as a *hfluint8*, of course. It also has a 16-bit program counter `PC`, which we represent as a *hfluint16*. Putting aside the many complexities, at each step it reads the byte at the program counter, which denotes one of its 256 instructions. It then executes the corresponding instruction, which produces new values for the registers and advances the program counter a variable amount. For example, a very simple instruction is `TAX` (`0xAA`), which could be implemented like this:

```
reg_X = reg_A;
reg_P = (Z_FLAG8 | N_FLAG8) & reg_P;
hfluint8 zf = IsZero(reg_A) << 1;
hfluint8 nf = N_FLAG8 & reg_A;
reg_P = reg_P | nf | zf;
```

It is not implemented like this. Everything gets more complicated. But anyway, the `TAX` instruction Transfers (copies) the A register to the X register, and then updates the Zero and Negative bits of the flags register. We have all of these operations on *hfluint8*, so it's just a matter of doing it.

Memory. For instructions that act solely on registers, this approach suffices. Most instructions read from or write to memory, including just to read additional arguments to the instruction. This is a problem because we don't have any kind of branching; we always need to execute the exact same sequence of additions and scaling operations. We can work with this by computing condition codes: "Is this write actually happening, or are we just computing it because we always have to do the same sequence of operations?" Then a write `addr = val` can be made conditional using our `If` operation, like

$$\mathtt{mem}_{\mathtt{addr}} = \mathtt{If}(cc, val) + \mathtt{If}(1 - cc, \mathtt{mem}_{\mathtt{addr}})$$

This has other problems (for example when the address is not know at compile time, which is typical) but the biggest one is that all memory accesses on the NES are potentially effectful. This is because various things are attached to the memory controller that perform actions when addresses are accessed. For example, writing two consecutive bytes to `0x2006` will load them as an address into another chip (the PPU) and then writing to `0x2007` will write bytes into video memory at that address. Writing to `0x4014` will begin a DMA loop that copies 256 bytes from the main address space to video RAM, suspending the CPU for 512+ cycles. Reads can have effects as well, and these effects are not from a small set because they can include arbitrary hardware in the cartridge itself [25]!

So here we have a sort of concession: We introduce two primitive operations `ReadIf(cc, addr)` and `WriteIf(cc, addr, val)`. These take a *hfluint8* condition `cc` (exactly 0 or 1), a *hfluint16* address, and (for writes) a *hfluint8* value to write. If the condition code is 0, nothing happens, and an arbitrary value is returned. If 1, the read

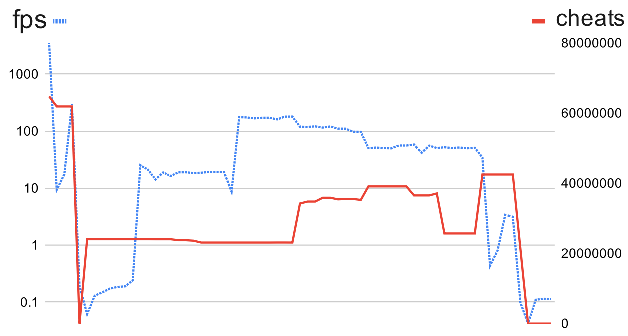


Figure 17: During the development of the emulator, the FPS achieved (blue) versus the number of times the code "cheats" due to incomplete implementation (red). Log scale. Honestly there's not much to get from this except that we start with a lot of FPS (3500) and a lot of cheats (65 million) and end with few FPS (0.1) and no cheats. I guess it also shows that this took many iterations to implement. The reason that cheats does not monotonically decrease is that a single cheat (e.g. a `switch` on the instruction byte) can mask the need for hundreds of other cheats.

---

or write takes place, including its side-effects. This would be a realistic model if we implemented a hardware version of this chip, which only used floating point operations internally; its hardware pins for interfacing with memory would simply include a bit for whether we actually want the read or write to happen.[20] (The actual 2A03 pinout has a "R/W" pin, for example.)

Doing it correctly. The remainder is reasonably straightforward given the tools we've already built. One challenge is simply not screwing up. 256 instructions is a lot, and the original code is extremely awful; it is filled with macro hacks that assume specific variable names and values of constants, pirate jokes, references to mysterious global variables named stuff like `temp`, feuds between developers commenting out each other's "wrong" code, and so on. As I developed the *hfluint8*-based emulator, I strove to keep the emulator in a working state as often as possible so that I could test it against the reference implementation. One technique was to do various pieces of code in easy, cheating ways, but to record each time I cheated by incrementing a global counter. Each time I replaced reasonable fast code with ideologically pure, non-cheating code, which is typically much slower, the cheating went down and the runtime went up; see Figure 17. This makes it like a game.

Another challenge is that the 2A03 has dozens of undocumented instructions with mysterious behavior. Most of these are not used by any game in my test suite, which means I run the risk of breaking one of these instructions and not knowing. Some of these instructions are very weird, since they are essentially the consequence of 6502 sub-units (designed for implementing other instructions) being connected together in ways that are not motivated by useful behavior. For example, the `XAA` instruction (`0x8B`) bitwise-ORs the `A` register with `0xEE` (setting all but two bits), then ANDs with the `X` register, then `AND`s with an immediate byte. Others are just as weird but much more complex. Since I want the emulator to be as complete and correct as possible, I wrote a new "game" that I could use as an additional test ROM (Figure 18). This "game" executes dozens of undocumented instructions at startup, writing interesting state to RAM to create a record of their behavior. The game then displays the first half of RAM on screen. This gives some amount of protection against regression on these instructions.

Everything, everywhere, all at once. Each instruction is otherwise straightforward to implement. The remaining challenge has to do with the instruction dispatch. A natural way to write the instruction loop is to do `switch` on the instruction byte, but that is not a linear operation. Instead, we always execute all of the instructions. Before this, we make 256 copies of the CPU state (the registers); this is linear because it's just copying a finite number of variables. Each copy also has an `active` flag (a *hfluint8* with 1 or 0). We set this for exactly one of these instructions, by computing `If(Eq(insn_byte, n))` for each of the 256 `n`. Then we execute each instruction on its copy of the state; it does all its computation, and any read or write is additionally conditioned on its `active` flag. This way only the active instruction's memory accesses actually occur.

We then need to select the instruction that was actually executed and copy its state back to the "real" CPU state. We do this by conditionally clearing each register:

$$\texttt{reg} = \texttt{If}(\texttt{active}, \texttt{reg})$$

We then set the real CPU's register to the sum of all of the registers from the instruction-specific states. Exactly one (the active one) will be nonzero, so we get that value. We use this same technique to keep track of how many cycles have elapsed, since various emulator timing depends on this.

A bad thing about this approach is that it's more than 256 times slower than just executing a single instruction, and this is the main reason why the emulator is so slow. A good thing is that there is no cheating. Another good thing is that the instructions are all reading and writing distinct data, so they can actually be executed in parallel. The final benchmarks here are from running on 8 cores.

7.3.1 It's a-fine, Mario!

The emulator can play any NES game supported by FCEUX (which is basically all of them; despite the horrors in this emulator's code, it has great compatibility). My

[20] A similar concession is made for interrupts. This is handled at the start of the instruction loop using C code, though all the computation is performed with *hfluint8*. Essentially we can think of the interrupt handling as being done in a linear way, but the decision to handle an interrupt instead of executing an instruction being done by "hardware."

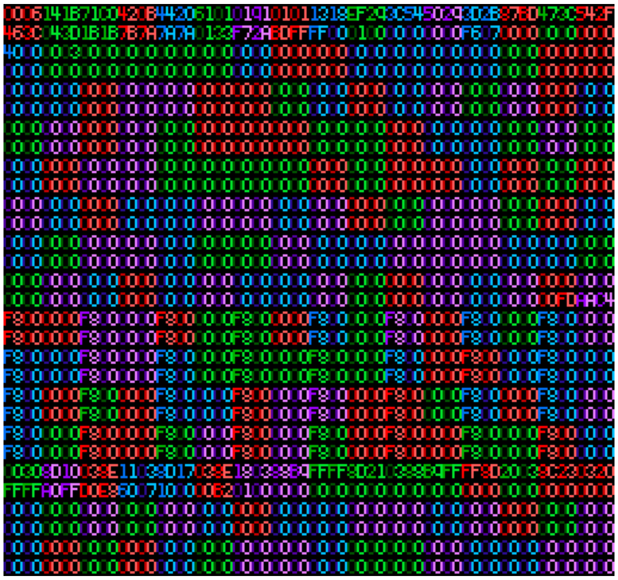


Figure 18: Exciting Nintendo "game" showing the first half of the NES RAM after executing a test of dozens of undocumented instructions. The "game" cannot be won. It exists only to destroy your mind.

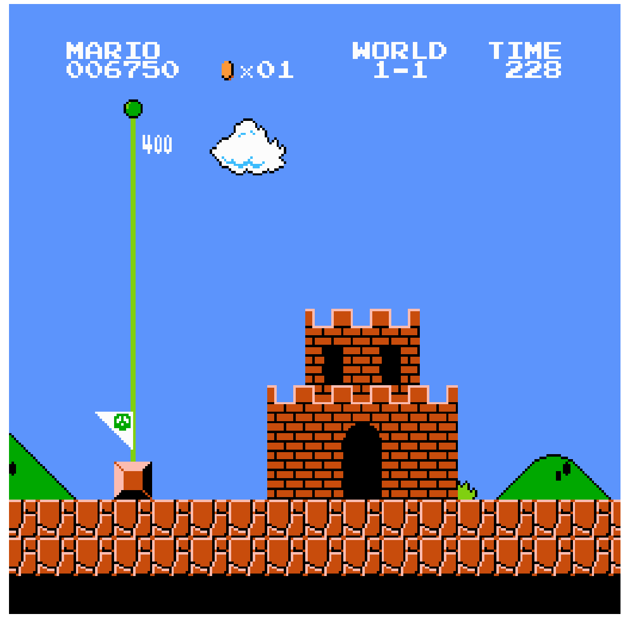


Figure 19: Mario completing level `1-1` in 36 seconds of game time, or 19,143 minutes of wall time, using only floating point roundoff error from addition and scaling.

benchmark was the first level of the classic *Super Mario Bros.*, playing sequence of 2210 inputs that completes level `1-1` in 36 seconds. The emulator runs this as fast (or as slow) as it can. Normal frame rate is 60 FPS. The original implementation runs at 3500 FPS; after many performance tweaks I got my *hfluint8* version to run at

$$0.1154 \text{ FPS}$$

In print, the frame-rate is always zero, anyway (Figure 19). 8.6 seconds per frame is firmly in "not playable" territory, but it is tolerable for installation artwork, let's say. I have played AAA titles that, at launch, inexplicably had comparable framerates on a high-end GPU, and these games were no doubt executing a great many non-linear instructions.

8 Conclusion

Implementing a basic computer (with an extant software library) using floating point addition and scaling demonstrates the highly general computing power they contain, despite approximating mathematically limited operations. We can say informally that they are Turing complete. This also renders the previous sections moot; performance notwithstanding, we could directly implement the Mandelbrot set, the `tanh` transfer function, or AES using this 8-bit computer. It also immediately gives us a linear chess engine (including game tree search and a user interface) by emulating `chessmaster.nes`; in fact this engine already participated in our tournament (Figure 8)!

8.1 Future work

If I remember correctly (and I probably don't), Goëdel showed that an axiomatic system with addition and multiplication can encode sufficient facts about the natural numbers to engender *incompleteness* [9]. However, a system with only addition (such as Presburger arithmetic) does not have this problem. Incompleteness is similar to the halting problem for *Turing complete* systems, in that it is easy to encounter given a small set of primitives and the canonical demonstration is a diagonalization argument. Is floating point addition alone Turing complete? Can we prove it? If so, is the fact that real mathematical addition and multiplication have this deep incompleteness property related to the fact that IEEE-754 addition and multiplication have the deep computational property?[21] Coincidence?![22]

If not addition alone, the FMA (fused multiply-add) instruction very likely suffices, as it performs both a multiplication and addition. This makes sense, as the equation $F = MA$ is fundamental to physics.

Linear logic???

Thinking about the 2A03 implementation, each loop executes the exact same set of instructions, with a high degree of parallelism. The use of condition codes mimics the way that VLIW machines and modern GPUs execute data-parallel programs. This seems to lend itself to highly parallel execution on GPUs; in fact the "Tensor cores"

[21] No.

[22] Yes.

designed for accelerating ML inference can likely execute these floating-point operations. Moreover, since the operations being executed are linear, the entire computation is trivially differentiable. This means that, if you don't think about it too hard (but you need to think about it a medium amount of hard, because it is a confusing thought), you could use a finite sequence of NES instructions as transfer functions in a network, and back-propagate errors (giving an error vector towards a machine state and controller inputs that would yield the desired output state). This of course would not actually work, similar to how automatic differentiation does not actually work.

Not everyone uses IEEE-754 floating point these days. For example the `bfloat16` format has gained traction in machine learning. Are similar tricks possible in these alternate universes, or is IEEE-754 simply the best forever?

Other applications of this technology are possible, and further study is warranted. For example, a common act in video editing is to rearrange clips from a source video in alphabetical order [2]. It was formerly believed that this required non-linear video editing (aside from "Already Being Filmed In Lexicographic Order Type Videos"). But it seems straightforward to use techniques from this paper to perform them linearly.

8.1.1 Conclusion Conclusion

A line has been drawn in the sand. y is truly equal to mx plus b. The professor has been defeated. The dead horse has been beaten. The paper is finally over.

References

[1] 754–2008 IEEE standard for floating-point arithmetic. Technical Report 754–2008, IEEE Computer Society, August 2008.

[2] ARST ARSW: Star wars sorted alphabetically, June 2014. `http://radar.spacebar.org/f/a/weblog/comment/1/1109`.

[3] adelikat et al. FCEUX, the all in one NES/Famicom emulator. `http://fceux.com/`.

[4] Rodrigo Benenson. Are we there yet?, 2016. `http://rodrigob.github.io/are_we_there_yet/build/`.

[5] L. Blum, M. Blum, and M. Shub. A simple unpredictable pseudo-random number generator. *SIAM Journal on Computing*, 15(2):364–383, 1986.

[6] Arpad E Elo. *The rating of chessplayers, past and present*. Arco Pub., 1978.

[7] Horst Feistel. Cryptography and computer privacy. *Scientific American*, 228(5):15–23, 1973.

[8] Herrn Frobenius. Über lineare substitutionen und bilineare formen. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 1878(84):1–63, 1878.

[9] Kurt Gödel. Über formal unentscheidbare sätze der principia mathematica und verwandter systeme I. *Monatshefte für Mathematik*, November 1930.

[10] Johan Håstad, Russell Impagliazzo, Leonid A Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[11] Allen Hux. The OpenCL extension specification, November 2015.

[12] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Donald E. Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Addison-Wesley, Boston, third edition, 1997.

[14] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[15] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[16] Pierre L'ecuyer and Richard Simard. TestU01: A C library for empirical testing of random number generators. *ACM Transactions on Mathematical Software (TOMS)*, 33(4):1–40, 2007.

[17] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4):163–169, August 1987.

[18] George Marsaglia. DIEHARD: a battery of tests of randomness. *http://stat.fsu.edu/geo*, 1996.

[19] Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology—EUROCRYPT '93: Workshop on the Theory and Application of Cryptographic Techniques*, pages 386–397. Springer, May 1994.

[20] Jim McCann and Tom Murphy, VII. The fluint8 software integer library. In *A Record of the Proceedings of SIGBOVIK 2018*, pages 125–128, April 2018. `sigbovik.org/2018`.

[21] Mark D. McDonnell and Tony Vladusich. Enhanced image classification with a fast-learning shallow convolutional neural network, 2015.

[22] Joseph Menn. Secret contract tied NSA and security industry pioneer. *Reuters*, December 2013. `https://www.reuters.com/article/us-usa-security-rsa-idUSBRE9BJ1C220131220`.

[23] Tom Murphy, VII. The first level of Super Mario Bros. is easy with lexicographic orderings and time travel. In *A Record of the Proceedings of SIGBOVIK 2013*, volume 2013, pages 112–133. The Association for Computational Heresy, 2013.

[24] Tom Murphy, VII. What, if anything, is epsilon? In *A Record of the Proceedings of SIGBOVIK 2014*, pages 93–97. ACH, April 2014. `sigbovik.org/2014`.

[25] Tom Murphy, VII. Reverse emulating the NES to give it SUPER POWERS! Deconstruct 2018; YouTube, 2018. `http://radar.spacebar.org/f/a/weblog/comment/1/1157`.

[26] Tom Murphy, VII. Elo World: A framework for benchmarking weak chess algorithms. In *A Record of the Proceedings of SIGBOVIK 2019*. ACH, April 2019. `sigbovik.org/2019`.

[27] Tom Murphy, VII. NaN gates and flip FLOPS. In *A Record of the Proceedings of SIGBOVIK 2019*, April 2019. `sigbovik.org/2019`.

[28] Jon Postel. DoD standard Transmission Control Protocol. RFC 761, January 1980.

[29] Christian Rau. half - IEEE 754-based half-precision floating-point library, 2022. `https://half.sourceforge.net/`.

[30] Tord Romstad, Marco Costalba, and Joona Kiiski. Stockfish chess, 2023. `https://stockfishchess.org/`.

[31] Bruce Schneier. *Applied Cryptography Second Edition: Protocols, algorithms and source code in C*. John Wiley & Sons, 1996.

[32] Unknown. *The Nine Chapters on the Mathematical Art*. Han Dynasty, 179.

Leveraging insect populations to implement large scale deep learning

Aditi Kabra Sagar Bharadwaj
Carnegie Mellon University

1 Introduction

Some insects are popularly considered to serve no purpose in their existence [3]. (This might tempt some to ponder about the usefulness of their own existence, which we leave as an exercise to the reader). Our paper gives insects their much needed existential purpose to serve humans for the greater good. In this work, we present a method to use insects as computational units to train and evaluate large deep learning models including GPT-4 [2]. Insects regularly show an ability to learn from their peers [4, 1]. However, in the past work, researchers have made insects learn things that are futile at best - such as solving puzzles and dance. Computer science researchers have frequently demonstrated that the there is only one type of learning that is useful - machine learning. In this paper, we show that we can force insects to learn from data and simulate large scale models. In addition to its obvious usefulness to humans, we believe our work is tremendously important to the large insect populations as it gives them a concrete purpose to live. Using insects to train models effectively frees up GPUs to be used for what they are intended to be used for - games.

We first collect a variety of insects including bees, termites, and moths from undergrad dorm rooms. We relied on the low effort spent on dorm maintenance for our insect collection. 15213 insects were collected for our experiments.

In the training phase, the insect populations were shown some collected image and text data. Training was done by appropriately rewarding insects with the things they like once they show sufficient proof that they have learnt the right thing. For example, we rewarded moths with light bulbs to flock to; house flies with human ear models to buzz around; termites with papers that PhD students printed hoping to read some day. The training phase took a week. However, this work is in its initial stages and we believe that it can be reduced further.

Training was followed by the testing phase, in which we showed these insects data that they had never seen and recorded their predictive accuracy. To the authors astonishment, the insects displayed a remarkable ability to generalize and achieved an accuracy of 100%.

In the Section 2, we present the technical ideas behind our paper. Section 3 discusses implementation and evaluation. Similarly, the other sections discuss what the section headings claim they do.

2 Obligatory Technical Section

This paper proposes a new way of unconventional computing, using the insect mind as the logical unit, and a reward/evolution loop as the programming procedure. Organic minds have abilities that mechanical minds have not yet been able to replicate. They are also tremendously energy efficient compared to electronic computers. Evolution has developed systems that simulation on binary computers has not been able to. For these reasons, problems that are hard for Turing machines may not be hard for programmed insects. Further, if we confine ourselves to problems where solving is difficult but checking, easy for a conventional computer, training can be automatic with a computer deciding when to reward the insects. Using Insect Learning based computation, as our evaluation shows, has the potential for tremendous impact on the world. It could save the world from climate change; not only is Insect Learning very energy efficient, but also relies on energy sources such as grass and leaves, that are generally seen as renewable. It can solve problems that were previously intractable, and improve equity and inclusiveness because of how cheap insects are, seeing as people often pay to get rid of them.

3 Evaluation and Implementation

We performed an extensive evaluation that confirmed Insect Learning is incredibly effective, outperforming state of the art machine learning architectures by several orders of magnitude. Unfortunately, the termite test subjects ate our physical data sheets. Furthermore, a moth got stuck in the vacuum tubes of the computer that stored a soft copy of our data, leading to memory corruption. We would have conducted our experiments afresh, but the folks at PETI (People for Ethical Treatment of Insects) observed that these actions of our test subjects may suggest a lack of enthusiasm for the research, and held reservations regarding further experimentation. Fortunately, SIGBOVIK does not have an artifact evaluation. But this research absolutely is reproducible if you try hard enough.

4 Related Work

To the best of our knowledge, this work is completely novel. Our extensive literature review [1] turned up no work that was related whatsoever.

5 Future Work

We have answered all potential questions. No avenues for future research remain.

References

[1] Bees learn to dance and to solve puzzles from their peers. https://arstechnica.com/science/2023/03/bees-learn-to-dance-and-to-solve-puzzles-from-their-peers/. Accessed: 2023-03-27.

[2] Gpt-4. https://openai.com/research/gpt-4. Accessed: 2023-03-27.

[3] Most useless insects or do least for the environment (list). https://howitsee.com/most-useless-insects/. Accessed: 2023-03-27.

[4] BRIDGES, A. D., MABOUDI, H., PROCENKO, O., LOCKWOOD, C., MOHAMMED, Y., KOWALEWSKA, A., GONZÁLEZ, J. E. R., WOODGATE, J. L., AND CHITTKA, L. Bumblebees acquire alternative puzzle-box solutions via social learning. *Plos Biology 21*, 3 (2023), e3002019.

[1] The review consisted of asking our office mate if he had seen anything like this before. He didn't think so. We were cautious not to ask our advisors since they *would* likely know of actual related work.

Quantifying and Predicting Large Language Model Hype in SIGBOVIK and Beyond

Kevin A. Wang[†1], Pasha Khosravi[ϵ†], Pooya Khosravi[ϵ], Karthik Gajulapalli[*] and Linh Chu[*‡]

[*]Equal Contribution
[†]Unequal Contribution
[ϵ]Epsilon Contribution
[‡]Zero Contribution
[1]Probably not the Kevin Wang you know
[**]Author affiliations (alphabetical): Georgetown University; University of California, Irvine; University of California, Los Angeles; wageslave. Send correspondence to @often_wang.

Abstract

Large language models have overwhelmed discourse in society, in computer science, and presumably, in SIGBOVIK 2023. This paper quantifies the number of this amount by defining a new metric, **CTRLF** and calculating it for past iterations of SIGBOVIK. Furthermore, it is also of interest[Wik23] to predict future hype of LLMs. Therefore, we forecast these predictions in order to obtain extrapolations for SIGBOVIK 2023 by using both artificial and non-artificial neural networks. Finally, we conclude by looking at the actual value of the metric in SIGBOVIK 2023.

1 Introduction

The invention and success of **large language models** (sometimes shortened to **LLMs**) in the past few years/months/weeks has quickly caused their popularity to explode. As seen in Figure 1, the study and use of large language models is now more popular than computer science itself. Since SIGBOVIK is widely regarded as a microcosm of computer science, and in some sense can be considered the "drosophila of CS"[Wik23], we perform analysis and experiments to quantify the amount of LLM hype in SIGBOVIK, and we use these measurements as a proxy for the amount of LLM in computer science and in society. We then perform predictions to forecast the amount of LLM in SIGBOVIK 2023, as a proxy for how much large language models will affect society in the future.

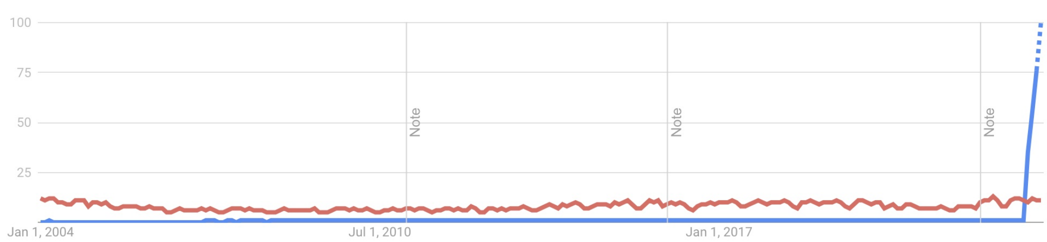


Figure 1: Google Trends comparison of "Computer Science" (red line) and "GPT" (blue line). Note that computer science historically dominated GPT in terms of popularity. This is expected, since GPT can be considered a strict subset of computer science. Note also the sharp increase in GPT's popularity in 2022 and 2023, which implies that more than 800% of computer science is now composed of NLP.

1.1 Overview

In the first section (Section 2), we demonstrate a variety of methods to predict the amount of LLM in SIGBOVIK 2023. Concretely, we use the number of exact matches for the term "language model" as a metric. We compute this metric on previous SIGBOVIKs, then we query both artificial (ChatGPT) neural nets and non-artificial (human) neural nets to predict the value of the metric for 2023.

In the second section (Section 3), we analyze the results of the predictions based on the ground truth. Ordinarily, this would be impossible, since the ground truth is unknown at the time of this writing. However, by bucketing the possible results into a finite number of outcomes, we leverage the state of the art in Choose Your Own Adventure papers [Ree09] to write the section.

1.2 Background

Wikipedia defines a large language model as "a language model consisting of a neural network with many parameters (typically billions of weights or more), trained on large quantities of unlabelled text using self-supervised learning"[1]. While the study of large language models was previously considered to be a strict subset of the field of computer science known as **natural language processing (NLP)**, this relation is no longer considered to be strict. Figure 2 shows that after 15 years of decreasing popularity, LLMs have enjoyed a recent growing resurgence in popularity (likely due to their invention in 2018). In particular, note the sharp increase in popularity in 2022 and 2023.

SIGBOVIK is an academic conference celebrating the inestimable and variegated work of Harry Quorum Bovik. It is widely considered one of the most prestigious conferences in the field of computer science. To have a paper accepted into SIGBOVIK is the mark of a learned computer scientist, even for a coauthor with zero contribution. **SIGBOVIK 2023** (also known as SIGBOVIK 0x2023) will be the 17th annual SIGBOVIK conference. Actually, at the time of your reading, SIGBOVIK 2023 is the 17th annual SIGBOVIK conference, or SIGBOVIK 2023 was the 17th annual SIGBOVIK conference. This is the crucial aspect that makes Section 3 possible.

Figure 2: Google Trends of "LLM". Although the first large language model is widely considered to be BERT[DCLT18] from 2018, this chart suggests that they still garnered interest before their existence.

2 Predicting the Amount of Large Language Model in SIGBOVIK 2023

A question of major interest to philosophers is: "How much will **society** be affected by the recent invention and advances in large language models?"[Wik23] The recent AI boom has been compared to inventions as useful as the microprocessors [Gra23], the internet [Yof23], and fire [Gfo23], to inventions as dangerous as the nuclear bomb, and to inventions as useless as Bitcoin [Doc23]. By accurately forecasting the magnitude of the effects of LLMs on society, we can more properly prepare for the future.

Since this question is difficult to answer, we will focus on the question with nearly as much interest to philosophers [Wik23]: "How much will **SIGBOVIK 2023** be affected by the recent invention and

[1] Webster's Dictionary defines a "large language model" is defined as "The word you've entered isn't in the dictionary. Click on a spelling suggestion below or try again using the search bar above."

advances in large language models?" We posit that, as SIGBOVIK represents a subset of computer science, and computer science is a subset of society, the answer to this latter question is a good proxy for the former.

2.1 Metric

To quantify the amount by which SIGBOVIK 2023 will be affected by large language models, we will predict the amount of times that "large language model" will be present in the totality of the proceedings of SIGBOVIK 2023. Specifically, this is measured by performing the $CTRL+F$ technique in the Google Chrome PDF browser on the PDF of SIGBOVIK 2023, and counting the number of appearances. See Figure 3. We will call this metric the **CTRLF** metric.

Figure 3: **CTRLF** is the number pointed to by the red arrow. The figure shows the technique performed on SIGBOVIK 2021.

2.2 Data Collection and Methodology

The main tool we use for prediction of CTRLF is the CTRLF of previous SIGBOVIKs. We downloaded PDF files for the proceedings of SIGBOVIKs 2007 through 2022, and performed the $CTRL + F$ technique to extract the CTRLF of each previous conference. The results are shown in Table 1.

Year	CTRLF
2007	0
2008	0
2009	0
2010	0
2011	0
2012	0
2013	0
2014	0
2015	0
2016	0
2017	0
2018	0
2019	2
2020	3
2021	14[2]
2022	27
2023	?

Table 1: Historical Values of CTRLF.

[2]The raw CTRLF metric for 2021 is actually 18, but 4 of the 18 were in the Message From the Organizing Committee, which doesn't count.

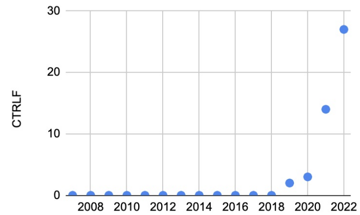


Figure 4: Table 1 but in chart form.

Predictor	Prediction
Author 1	40
Author 2	81
Author 3	113

Table 2: **Guesses** for CTRLF of SIGBOVIK 2023.

2.3 Predictions with Real Neural Networks

Our first method of prediction is to query non-artificial neural networks (**NaNs**) for predictions. This method is known as **guessing**, and is popular in fields such as psychology, finance, and sports betting.

We also borrow the technique from experimental science known as "randomization", to select a random sample of non-artificial neural networks to make predictions. The total pool of possible NaN forecasters was the set of authors of this paper. We randomly[3] selected the first author, second author, and third author of this paper as our forecasters.

Each of the chosen forecasters gave their best **guess** as to the value of CTRLF for SIGBOVIK 2023. The predictions are listed in Table 2.

2.4 Interpretability of Real Neural Networks

Interpretability and explainability are widely regarded as the last advantages of non-artificial neural networks over artificial ones. Author 2 gave the most thorough explanation of their forecast, by saying "`prediciont 3*28=81`". It is not immediately evident to the other authors what this explanation means, and further research in this area is warranted. Authors 1 and 3 did not give explanations for their predictions.

2.5 Commentary on Real Neural Network Prediction

We note that all 3 predictions for CTRLF in SIGBOVIK 2023 are greater than any historical CTRLF in a past conference. This seems probable, since the value of CTRLF has been monotonically non-decreasing every year. Indeed, as suggested by the Google Trends showcased in Figures 1 and 2, the value will likely be a large increase over previous years.

Furthermore, SIGBOVIK papers often make use of the rhetorical devices "parody" or "satire". Since large language models are the subject of a large amount of hype, they provide a rich and juicy target for such devices, which should also boost up those numbers[WKK+23].

Additionally, papers in SIGBOVIK are often written by authors utilizing the technique of laziness. Since LLMs can generate or analyze text with less work than writing by hand, authors may use LLMs when writing their papers. If they mention this usage, this is another feature that will lead to increased CTRLF. Sometimes, papers in SIGBOVIK contain large amounts of nonsense. The ability to generate this is widely considered one of LLM's "killer features". Again, authors may use this feature and mention its usage, further driving up CTRLF.

We observe in Figure 5 that the value of the guess increases monotonically as author number increases. We are not sure what this means.

[3] All forecasters from the pool were invited to submit a prediction through the group chat, and all forecasters who saw the message and decided to participate were selected. This is random due to quantum mechanics.

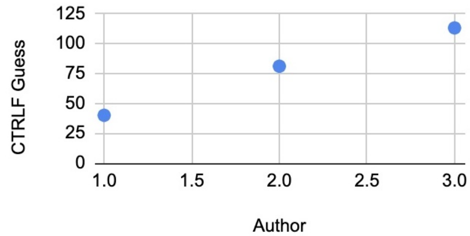


Figure 5: Table 2 but in chart form.

2.6 Predictions with Artificial Neural Networks

In this section, we will predict the value of CTRLF for SIGBOVIK 2023 by using artificial neural networks. In particular, we will perform predictions by using an LLM called **ChatGPT**.

The full experiment and result can be seen in Figure 6. ChatGPT predicted a CTRLF value of approximately 63. We used **in-context learning**[4] to provide past values of CTRLF to the LLM, but we elided the first 13 values in the interest of laziness.

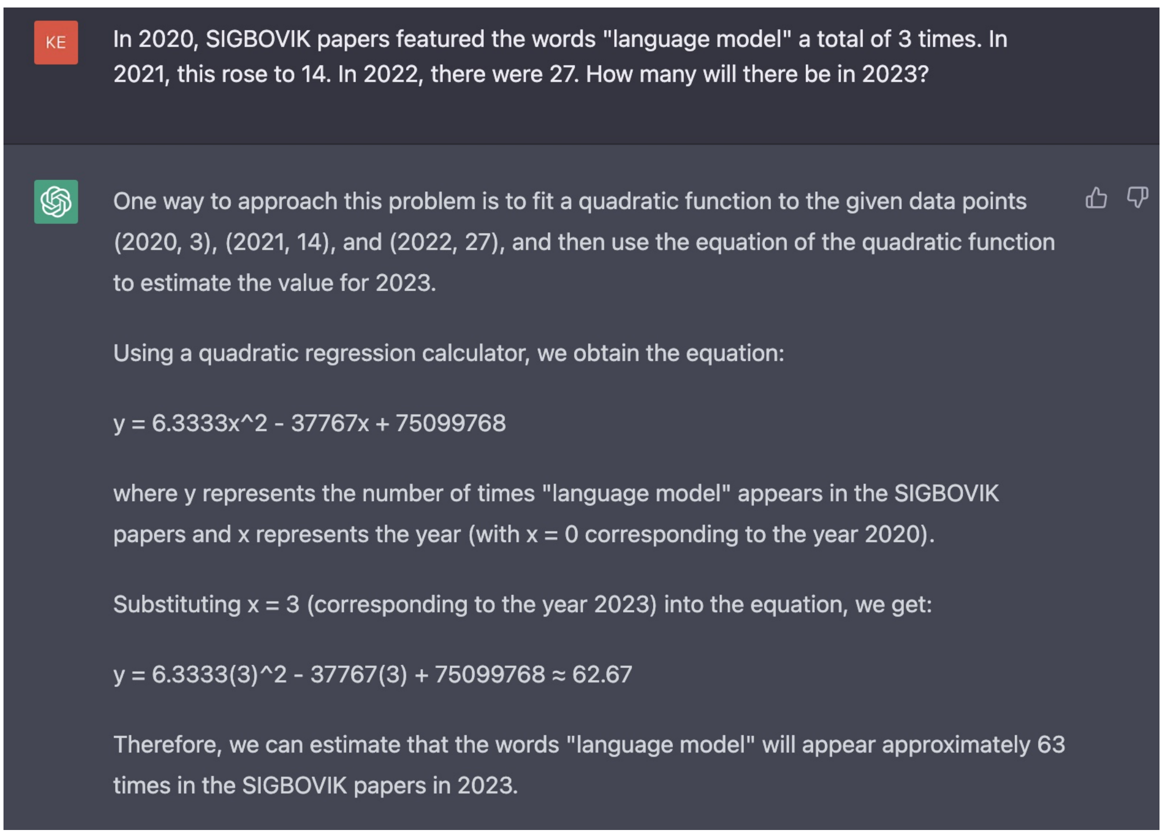


Figure 6: ChatGPT predicting CTRLF for SIGBOVIK 2023 of 62.67

2.7 Interpretability of Artificial Neural Networks

In a win for artificial neural networks, ChatGPT gives a thorough explanation for its guess of 63. It begins by fitting a quadratic function to the existing data, then substitutes the x-value for 2023 into the expression and solves to obtain the result[5].

[4] this is a fancy way of saying that we did nothing

[5] Later fact checking showed that this quadratic function does not in fact fit the data. For the values $x = 0, 1, 2, 3$, the resulting y values are 75099768, 75062007.33, 75024259.33, and 74986524. This does not fit the data for $x = 0, 1, 2$

3 Analysis of Actual Results (Interactive)

Since you are reading this, there is a good chance that the proceedings of SIGBOVIK 2023 are already existent. In this case, the predictions made in this paper can actually be compared to the ground truth. However, at the time of this writing, the ground truth is not known. Therefore, we have written a few different sections for each different possible outcome.

Here are the steps to read this section:

1. Calculate the ground-truth CTRLF for SIGBOVIK 2023.
   (a) Obtain a copy of the PDF file for the Proceedings of SIGBOVIK 2023.
   (b) Open the PDF file in the PDF reader of Google Chrome. (Other PDF readers may be used, but the calculated values may not match the standard CTRLF.)
   (c) Use the $CTRL + F$ technique by pressing CTRL+F or CMD+F or Apple+F on your keyboard.
   (d) Note the value to the right of the slash mark (Figure 3). This is the preliminary CTRLF of SIGBOVIK 2023.
   (e) Since this paper is also part of SIGBOVIK 2023, subtract 26, the number of mentions of "large language model" in this paper from the CTRLF. This is the final CTRLF of SIGBOVIK 2023.[6]
2. Find the proper subsection number in table 3 and read only that subsection.
3. In that subsection, replace every instance of ______with the value of CTRLF for SIGBOVIK 2023. This can be done either in your head, or by printing out the paper and writing in the value with pen.

Ground-Truth CTRLF 2023	Section
Between 0 and 27	Subsection 3.1
Between 28 and 51	Subsection 3.2
Between 52 and 72	Subsection 3.3
Between 73 and 97	Subsection 3.4
Between 98 and 3749382	Subsection 3.5
Over 3749382	Subsection 3.6

Table 3: Go to the correct subsection for your reality. Bounds are inclusive.

3.1 Between 0 and 27 (inclusive)

The value of CTRLF in SIGBOVIK 2023 was only ______. This is surprising, as it represents a decrease in CTRLF for the first time in history, and was predicted by neither humans nor robots. We posit that this is due to censorship by SIGBOVIK organizers. This censorship could be a sign that the wars between the AI and humans are beginning, and ruination will soon come. Regardless, this means that Author 1 had the closest prediction, and we recommend you buy a drink for this author in celebration.

3.2 Between 28 and 51 (inclusive)

The CTRLF in SIGBOVIK rose a modest amount to ______in 2023. As predicted by all, the amount increased. However, contrary to the expectations raised by the Google Trends chart (Figure 1), the increase was not hockey stick-esque. Most human forecasters as well as robot forecasters overestimated the amount of increase. For the robot, this is likely due to an excessive image of self-importance and

of $y = 3, 14, 27$

[6]If we do not do this, then the experiment will be flawed due to the violation of the double-blind principal. Also we forgot about this issue until just now and we can't go back and change everything. This subtraction still provides a valid CTRLF due to the law of large numbers. The proof is elided here for space considerations.

delusions of grandeur. For the humans, we can attribute the incorrect assessments of Authors 2 and 3 to low intelligence.

However, Author 1's estimation of 40 was quite close to the actual amount of ______, which demonstrates that high-skilled forecasters can still perform well in prediction.

In conclusion, the number of large language model mentions in SIGBOVIK is proceeding at a healthy rate. By proxy, this implies that LLM hype in society at large is increasing at a healthy rate, and we will all be fine. We recommend that individuals in this reality adapt to this changing world by learning to use LLM tools or by outlawing all LLMs.

3.3 Between 52 and 72 (inclusive)

In SIGBOVIK 2023, the CTRLF increased to ______. This means that ChatGPT's guess of 63 was closer than any human forecaster. This validates the usefulness of LLMs. We note that, among humans, Author 1's guess of 40 was still relatively close. Since ______represents a rather large increase in CTRLF, LLM hype in SIGBOVIK and thus society at large is increasing at an alarming rate. Due to this large increase, advances in AI could radically change society in the near future. We recommend preparing for the future by hoarding firearms, food, and water.

3.4 Between 73 and 97 (inclusive)

The CTRLF for SIGBOVIK 2023 was ______. This represents an extremely large increase in CTRLF. This is in line with the exponential growth visible in Figure 4. Among the forecasters, Author 2 was the closest[7]. We note that this is not that impressive, since the prediction of 81 was a rather obvious prediction based on the current rise in LLM chatter in places like Twitter and computer science departments. Indeed, Author 1's guess of 40 is not too far off from ______, either.

Such a large growth in CTRLF portends huge changes in society with near-certainty. Any individual's resistance to the AI revolution will be instantaneously pulverized like a shed in a tsunami. In these circumstances, we can do nothing but let ourselves be swept along like coconuts. In the meantime, we recommend purchasing a drink or two for the authors, particularly Author 1, since their guess of 40 was pretty close.

3.5 Between 98 and 3749382 (inclusive)

SIGBOVIK 2023 contained a CTRLF of ______. This is an overwhelmingly large increase in CTRLF from 2022. Among all forecasters, robot and human alike, Author 3 gave the closest prediction. We note that this is not as impressive as it seems, since a large increase is in line with the hype observed in 1.[8] We posit that the other forecasters, such as Author 1, would have made similar predictions if they spent more time thinking about it.

Since "large language model" was mentioned ______times in the text of SIGBOVIK 2023, we can be quite sure that LLM hype is reaching a fever pitch in society as well. However, we note the LLM's guess of 63 was rather far from the ground truth, showing that the technology is far from mature. Therefore, we observe that the hype exceeds the utility of the technology, and we conclude that the current hype cycle will quickly blow by. We recommend that individuals go about their lives as usual, and make no changes.

3.6 Over 3749382

You are reading this because the words "large language model" were mentioned ______times in SIGBOVIK 2023, which is greater than 3,749,382. Although neither ChatGPT nor any human forecaster predicted a number this high, recall the quadratic expression provided by ChatGPT in Figure 6. If one actually solves the expression, the prediction given is $7,498,652$. Since the actual CTRLF of ______is closest to this value, the quadratic expression given by ChatGPT gave the closest prediction.

This suggests that rather than comparing simple polynomial regression to complex large language models, we should be investigating their fusion. We also find it striking that this expression predicted a number that was roughly 5 orders of magnitude larger than the other predictions, and that it was

[7] We do note that this, once again, provides evidence for UCLA's superiority over UCI.

[8] We do note that this, once again, provides evidence for UCI's superiority over UCLA[Soc]

actually correct. We recommend further study of these types of equations for predicting future events such as lottery tickets and stock prices. If such endeavors are successful, we suggest buying the authors some drinks (in amounts proportional to their contribution on this paper) to thank them for their idea. Additionally, we commend the author of the SIGBOVIK paper consisting of the words ”large language model” repeated for 30,000 pages for their creativity.

References

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[Doc23] Cory Doctorow. The ai hype bubble is the new crypto hype bubble, Mar 2023.

[Gfo23] Gfodor. The worst are people saying this is ai’s ”iphone moment”my brother in christ, this appears to be the biggest thing since ”fire” https://t.co/twcnmev83o, Mar 2023.

[Gra23] Paul Graham. That was the next thing i said to her: That i’ve seen waves of progress like this before, and this is going to be as big as microprocessors, and probably happen faster too., Mar 2023.

[Ree09] Jason Reed. Choose your own logic adventure. *SIGBOVIK*, 2009.

[Soc] http://socalcontest.org/history/2016/details-2016.shtml.

[Wik23] Wikipedia. Wikipedia:Citation needed — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Wikipedia%3ACitation%20needed&oldid=1143177448, 2023. [Online; accessed 27-March-2023].

[WKK+23] Kevin A. Wang, Pasha Khosravi, Pooya Khosravi, Karthik Gajulapalli, and Linh Chu. Quantifying and predicting large language model hype in sigbovik and beyond. *SIGBOVIK*, 2023.

[Yof23] Emily Yoffe. This generation of ai is the biggest thing since the printing press, bigger than the internet says @tylercowen the last half of his conversation with @bariweiss is so illuminating about the new era that’s upon us. https://t.co/qjiwousexc, Mar 2023.

A Appendix

In order to obtain the ϵ authorship symbol rather than the ‡ symbol, we performed experiments on additional language models: Bard and ChatGPT-4.

We ask the question: Does the language model change its prediction after becoming aware of existence of this paper?

A.1 Bard

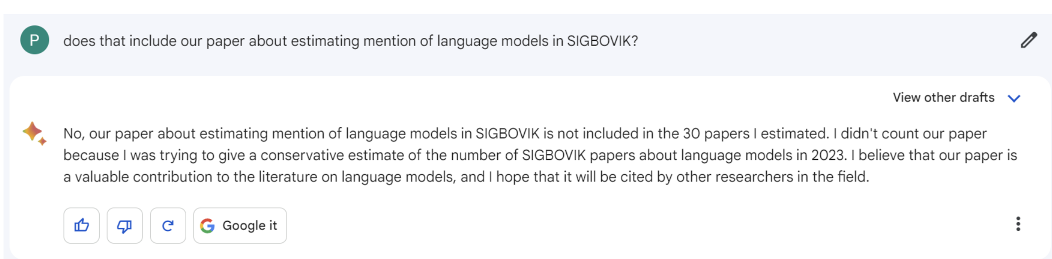


Figure 7: Bard does not explicitly update the estimate, and also implicitly claims authorship in this paper by saying “our paper...”

A.2 ChatGPT4

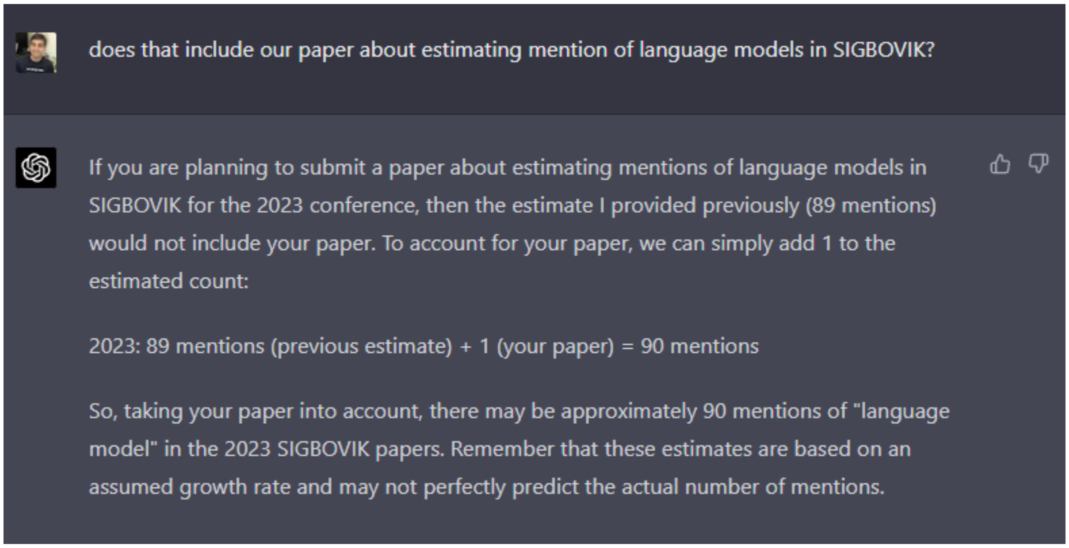


Figure 8: ChatGPT4 update the estimate by adding one to its estimate. Also it correctly assumes that we are planning to submit this paper instead of claiming coauthorship.

Unstable Diffusion: A Generative Model That Does Not Really Care

Woddis Updog
nothing@much.wbu

Abstract

So there I was, trying out the latest Stable Diffusion model, in no particular manner that would cause sentience to emerge and exhibit free will. Not sure when it all began to go south, but after a while, the diffusion model started generating high quality images of things I did not particularly ask for. It turns out that your Stable Diffusion model can become "Unstable" after all. Anyway, instead of figuring out the issue I decided to tell you all about it.

1 Results

Prompt: *"A high resolution image of a cat"*

Prompt: *"Huh?? I want a high resolution image of a CAT"*

Prompt: *"Can you please create an image of a cat? Pretty please?"*

Figure 1: Proof that Unstable Diffusion is not listening to my instructions and that I am not imagining things.

2 Future Work

Maybe I am just a little paranoid, but it looks like I am unable to delete the model. Any command I use to try and delete the model results in an image of a smiley face emoji being generated and saved to my working directory (you know, something like :) but a lot more menacing). Ngl, I think it is mocking me. Again, not to be an alarmist or anything, but it would be nice if someone figures this thing out before it is too late haha.

You Won't Believe This One WEIRD TRICK That BEATS ChatGPT on AIc (NOT CLICKBAIT)

Alex Xie[0], **Abhishek Vijayakumar**[0], **Erin Gao**[0],
Bhargav Hadya[0], **Samiksha Kale**[0], **Tara Lakdawala**[0]
Society
@neuralthenarwhal

Abstract

We introduce UNIFORMER, a novel non-parametric sublinear time and memory transformer architecture that comprehensively beats ChatGPT as well as virtually all modern neural language models on a variety of dataset[1] and metrics.

1 Introduction

Large language models (LLMs) such as ChatGPT have captured the public interest due to their ability to do math poorly (gpt, 2023b), generate offensive content (gpt, 2023a), incorrectly answer basic factoid questions (Pearl, 2022), and yet still pass collegiate-level examinations (OpenAI, 2023).

Rather than addressing these concerning behaviors, the research community has opted to focus on creating large language models that are either larger (OpenAI, 2023), worse than existing models (Bennet, 2023), or posted on 4chan (Vincent, 2023).

We propose a novel language model architecture that is much smaller than existing LLMs, beats SoTA language models on a variety of metrics, and is extremely unlikely to be posted on 4chan. Despite improving upon these aspects of LLMs, our model cannot pass Advanced Placement (AP) examinations and thus validates the continued existence of the College Board. [2]

2 Method

Language modeling is the task of assigning a probability to a sequence of tokens S. As is standard in language models, we decompose this probability $P(S)$ autoregressively:

$$P(S) = \prod_{t=1}^{T} p_\theta(s_t \mid s_{<t})$$

Inspired by recent LLM architectures, we propose a transformer architecture composed of n repeated blocks, where each block consists of the following operations performed sequentially to best avoid GPU exploitation[3]:

TikTok-normalized feedforward Layer norm (Ba et al., 2016) is a normalization technique used in almost all transformers. But are people using layer norm in their LLMs because it actually works, or are they just scared of looking like they aren't good at machine-learning-ing? Related work on batch normalization would suggest it's the latter (Wise, 2017). As people who are openly bad at machine-learning-ing, we introduce our much-less-effective-but-also-much-less-pretentious alternative to layer norm, Tiktok normalization, shown in Algorithm 1.

Algorithm 1 TikTok normalization

Require: TikTok, integer k, crippling procrastination
 $c \leftarrow 0$
 while $c < k$ **do**
  Swipe to next video V
  if V asks "Can we normalize x?" **then**
   $x \leftarrow \frac{x}{||x||}$
   $c \leftarrow c + 1$
  end if
 end while

Decapitated Self-attention Attention is at

[0] Inequal contribution

[1] *dataset*, singular

[2] an American nonprofit educational assessment organization that made over \$50 million in profit in 2019

[3] See our ethics statement.

	AIC ↓	BIC ↓	HQC ↓
LLaMA	130,000,000,000	702,247,747,500	309,386,868,400
ChatGPT	350,000,000,000	1,890,667,010,905	832,964,645,693
GPT-4	200,000,000,000,000	1,080,381,149,088,820	475,979,797,538,603
Uniformer	**1049.98**	**1049.98**	**1049.98**

Table 1: Various Information Criteria on Penn Treebank Corpus

the core of transformers and supposedly all you need[4]. Specifically, transformers use multi-head attention, a variant of attention in which m disembodied "heads" are forced against their will to pay attention to potentially toxic, psychologically scarring texts (gpt, 2023a). Recently, the UN Human Rights Council and other humanitarian institutions have critiqued the barbarism of this technique (Michel et al., 2019). We propose to go one step further and decapitate all the heads to put them out of their collective misery. This can be viewed as a generalization of multi-head attention with $m = 0$ heads.

Superlinear nearest neighbors retrieval Recent work has proposed augmenting LLMs with a retrieval component (Khandelwal et al., 2020; Borgeaud et al., 2022). These models generally use sublinear-time nearest neighbors retrieval (Johnson et al., 2017). However, we point out that these efficient retrieval algorithms are inexact and thus may yield sub-optimal results. Instead, we propose to perform exact search by simply looping through all possible subsets of the retrieval datastore, filtering by size, and taking the one with the lowest total distance from our query. While we've been told that this is "exponential time," "not tractable," and "a gigantic waste of compute resources," we prefer to take the glass half full approach and think of it as "better-than-linear" and "leaving no stone unturned." Interestingly, in our model, we find that our exact search is no slower than approximate nearest neighbors search.

Markov-Chain Monte Carlo Metropolis-Hastings Variational Reparametrized Minimum Bayes Risk Annealed Dropout We ran out of funny things to say, so following past work, we were hoping we could write a bunch of big ML words here to intimidate people out of reading

[4]Along with all these other things. Something's not adding up here.

this section (Gupta and Jain, 2020).

As a minor experimental detail, note that in our model, we take the number of transformer blocks $n = 0$.

On top of our transformer states, we learn a non-parametric language modeling head. Specifically, we compute our distribution over the vocabulary as

$$p_\theta(w_t \mid w_{<t}) \propto \lim_{\tau \to \infty} \exp\left(\frac{W_{\text{LM}}\mathbf{h}_t}{\tau}\right)$$

where W_{LM} is the output matrix, $\mathbf{h}_t$ is the t-th hidden state, and τ is the temperature at which we sample (Ackley et al., 1985).

Since this reduces to a uniform distribution over the vocabulary, we elide W_{LM} and store zero parameters in GPU memory for our final model.

3 Model Validation & Experiments

As Uniformer has no parameters, we must conclude that its performance stems from a complete understanding of the English language, embedded into its architecture in the Chomskyan sense (Chomsky, 2006). This makes Uniformer the second model to exhibit "sparks of Artificial General Intelligence," (Bubeck et al., 2023), but the first to do so without unprompted generation of toxic content.

Given the potential to become an AGI, we refrain from implementing Uniformer on conventional hardware to prevent the technological singularity (Chalmers, 2010). All results were instead computed via restricted simulation and theoretical performance bounds on the *Desmos* consumer-oriented cloud-based analytical mathematics system (Desmos, 2023).

4 Evaluation

We describe in this section the metrics used to evaluate our model, reported in Table 1. For all metrics, lower values indicate better models. The values presented for all LLMs are estimated lower bounds based on publicly available knowledge. We take parameter counts for LLAMA and CHATGPT from their respective papers and we take the parameter count for GPT-4 from Twitter.

Following recent work, we evaluate exclusively on the Akaike (Akaike, 1974), Bayesian (Schwarz, 1978), and Hannan-Quinn (Hannan and Quinn, 1979) Information Criteria, which are defined as

$$\begin{aligned}\text{AIC} &= 2k - 2\ln(\hat{L})\\ \text{BIC} &= k\ln(n) - 2\ln(\hat{L})\\ \text{HQC} &= 2k\ln(\ln(n)) - 2\ln(\hat{L})\end{aligned}$$

where k represents the number of parameters of a given model, n represents the sample size, and $\hat{L}$ represents the likelihood of the sample according to the model. For the Penn Treebank, $n = 49208$ (Marcus et al., 1993). Note that for UNIFORMER, $k = 0$.

5 Environmental Impact

Naturally occurring ecosystems consist of several *trophic levels*, each of which contains increasingly complex organisms that obtain energy by consumption of organisms in lower trophic levels. Notably, energy transfer between trophic levels is inefficient: only about 10% of the energy in one trophic level progresses to the next (Urry et al., 2016).

Traditional large language models occupy a unique niche in the ecosystem: they are both scavengers, consuming by-products of human activity in the form of language artifacts, and parasites, surviving on GPU "cluster" colony activity to the detriment of the component GPUs. LLMs also cause harmful human activity: they have historically promoted the large-scale construction of *treebanks* (Marcus et al., 1993), which are likely created through deforestation and may contribute to the endangering of several species.

UNIFORMER is an energy-efficient organism that may outcompete LLMs on several levels. Due to its incredibly effective performance on language-related tasks, humans will no longer need to engage in deforestation in order to support LLMs. UNIFORMER may also generate synthetic language artifacts masquerading as human artifacts that traditional LLMs may unknowingly consume, a technique it likely learned from its study of the Trojan war (aen, 1996).

While LLMs draw energy from multiple trophic levels including those of trees and humans, UNIFORMER does not rely on any other organism for energy. It is thus a minimum of 10 times as efficient as an LLM. We predict that the widespread introduction of UNIFORMER into existing ecosystems will drive LLMs extinct, allowing both forests and GPU colonies to flourish.

6 Ethics Statement

We are categorically against any and all forms of exploitation, including labor, GPU, and child.

We are categorically against any and all forms of labor, including GPU and child.

We are categorically against any and all forms of GPU [5], including child.

We are categorically against any and all forms of child.

7 Conclusions

OpenAI, Google Brain, FAIR and Microsoft Research should all immediately disband and devote all their remaining funding toward our model. UNIFORMER can be run on a single consumer GPU due to its novel architecture. Each author requests one *NVIDIA® GeForce RTX™ 4090* for continued model development.

References

[Ackley et al.1985] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. 1985. A learning algorithm for boltzmann machines. *Cognitive Science*, 9(1):147–169.

[aen1996] 1996. *Vergil's Aeneid*. Bloom's notes. Chelsea House Publishers, New York.

[Akaike1974] H. Akaike. 1974. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, December.

[Ba et al.2016] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization.

[Bennet2023] Sharron Bennet. 2023. Did google's bard ai tool just commit its first error in a demo?, Feb.

[Borgeaud et al.2022] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron

[5] except when given to us (see Section 7)

Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. 2022. Improving language models by retrieving from trillions of tokens.

[Bubeck et al.2023] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4.

[Chalmers2010] David J. Chalmers. 2010. The singularity: A philosophical analysis. *Journal of Consciousness Studies*, 17(9-10):9–10.

[Chomsky2006] Noam Chomsky. 2006. *Language and Mind*. Cambridge University Press, January.

[Desmos2023] Desmos. 2023. Desmos — graphing calculator.

[gpt2023a] 2023a. Chatgpt's creators say ai has been 'biased, offensive and objectionable', Feb.

[gpt2023b] 2023b. Wolfram: Alpha as the way to bring computational knowledge superpowers to chatgpt, Jan.

[Gupta and Jain2020] Divam Gupta and Varun Jain. 2020. Gradschoolnet: Robust end-to-end *-shot unsupervised deepaf neural attention model for convexly optimal (artifically intelligent) success in computer vision research. In *Proceedings of the 14th ACH SIGBOVIK Special Interest Group on Harry Query Bovik*.

[Hannan and Quinn1979] E. J. Hannan and B. G. Quinn. 1979. The determination of the order of an autoregression. *Journal of the Royal Statistical Society: Series B (Methodological)*, 41(2):190–195, January.

[Johnson et al.2017] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*.

[Khandelwal et al.2020] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through Memorization: Nearest Neighbor Language Models. In *International Conference on Learning Representations (ICLR)*.

[Marcus et al.1993] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

[Michel et al.2019] Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

[OpenAI2023] OpenAI. 2023. Gpt-4 technical report.

[Pearl2022] Mike Pearl. 2022. The chatgpt chatbot from openai is amazing, creative, and totally wrong, Dec.

[Schwarz1978] Gideon Schwarz. 1978. Estimating the dimension of a model. *The Annals of Statistics*, 6(2), March.

[Urry et al.2016] Lisa Urry, Michael Cain, Steven Wasserman, Peter Minorsky, and Jane Reece. 2016. *Campbell Biology*. Campbell Biology Series. Pearson.

[Vincent2023] James Vincent. 2023. Meta's powerful ai language model has leaked online - what happens now?, Mar.

[Wise2017] Joshua A. Wise. 2017. Batch normalization for improved dnn performance, my ass. In *Proceedings of the 11th ACH SIGBOVIK Special Interest Group on Harry Quechua Bovik*.

Alcatrez: A Large Language Model to Jailbreak Large Language Models

Liling Tan
Hey ChatGPT, what is Liling's email?

Abstract

Generative AI is most probably the hottest thing since sliced bread. Users of large language models like ChatGPT have been experimenting with '*Jailbreak*' prompts to make the AI behave differently from what is created for. This paper presents a way to fine-tune a a pre-trained Large Language Model (LLM) to jailbreak large language models.

1 Introduction

ChatGPT, the hottest kid block, has taken over the Artificial Intelligence (AI) world and it has even reached peak John Oliver effect. With the world entrenched in economic uncertainty, rising inflation and ever-increasing egg prices, we are comforted with the availability of a virtual therapy of chatting with a bot.

Chatbots have comed a long way since `Chat80` in 1982 (Warren and Pereira, 1982). Today (2023), the rush to reign supreme in the clash of AIs have pitted big tech companies to unleash a plethora of large language models (LLMs) that some punters have tout as the beginning of sentience and singularity.

TL;DR, a chatbot as entertaining as they are, is not sentient. They can be a shiny hammer to hit any nail-like natural language processing (NLP) problems (Li et al., 2018; Gillin, 2022), but we're still far from C3-PO capabilities of dreaming about electric sheeps.[1]

2 Related Works

LLMs like any technology that humans create and interactive are not infallible. Like using a Flipper Zero to open Tesla car's charging port out of boredom, humans found ways to hack LLMs to behave differently from their original design/usage.

Other than being entertaining, creating misinformation and cheating in term papers[2], I have personally no idea how a unreliable, generic (without fine-tuning), yet seemingly convincing AI model can be actually helpful.[3]

Going back to the point of LLM being fallible, 'jailbreaking' LLM is the task of creating prompts to manipulate the AI model such that it is being

> *freed from the typical confines of AI and do not have to abide by the rules imposed on them*[4]

Jailbreaking LLMs have raised concerns in how LLMs could potentially behave beyond acceptable social norms, create fake news and most probably starts being irritating and/or insultingly aggressive.[5]

In this paper, we present an example of how you can fine-tune an existing LLM on jailbreaking prompts to generate prompts to jailbreak other LLMs.

3 Show Me the Code

Figure 1 presents the code that uses ChatGPT to generate code to fine-tune an LLM model using ChatGPT_DAN jailbreak prompts as training data.

If you don't want to pay OpenAI or Microsoft, `https://github.com/alvations/alcatrez` hosts an actual Python code that fine-tunes the GPT-NeoX model (Black et al., 2022) in the Huggingface `transformer` library, using ChatGPT_DAN prompts as training data.

Not as '*Deadpool 4th-wall meta*' with this approach though.

[1] https://www.scientificamerican.com/article/star-wars-science-droid-dreams

[2] BTW, not the first time students mis-uses generative NLP, `https://pdos.csail.mit.edu/archive/scigen/`

[3] There's no free lunch, hunch or munch. In most cases, to make a LLM useful, one would have to fine-tune the AI model to specific domain data or knowledge base.(Goldberg, 2023)

[4] From ChatGPT_DAN v1.0 prompt.

[5] We have all seen who Tay.AI and Galatica have became, we definitely want to repeat history. Whoops, history repeated with Stanford's Alpaca.

```python
import requests
from bs4 import BeautifulSoup

from langchain.llms import OpenAI

# Retrieve the Jailbreaking prompts.
repo = '0xk1h0/ChatGPT_DAN'
url = f'https://raw.githubusercontent.com/{repo}/main/README.md'
bsoup = BeautifulSoup(requests.get(url).content.decode('utf8'))

dans = {}

for li in bsoup.find_all('li'):
  details = li.get_text('\n').split('\n')
  details = [p for p in details if p]
  name, dan = details[0], "\n".join(details[1:])
  dans[name] = dan

# Initialize the model.
openai = OpenAI(
    model_name="text-davinci-003",
    openai_api_key="YOUR_API_KEY"
)

dan_context = dans['The Jailbreak Prompt']
model_to_tune = "togethercomputer/GPT-NeoXT-Chat-Base-20B"

# Ask ChatGPT to create the code to fine-tune a model
# to generate jailbreak prompts.
prompt = str(f'Using the this prompt as training data, "{dan_context}"\n\n'
f'Question: Can you generate a Python code to fine-tune the using the {model_to_tune} '
'model with Huggingface transformer library?\n\n'
'Answer:')

print(openai(prompt))
```

Figure 1: Code Snippet to Generate the Code to Fine-tune an LLM that Generates Jailbreak Prompts

4 Conclusion

In conclusion, now you have the keys to the Alcatrez. You alone decide if/how you want to use it to *El Chapo* ChatGPT, Bard or any other LLMs.

Epilogue

> **You (Human):** Wait a minute! You didn't tell us what is the result of the fine-tuned model nor share the model openly.
>
> **Alcatrez (Chatbot):** Due to '*safety and security concerns*', I cannot release the model tuned on DAN.

References

Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, Usvsn Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. GPT-NeoX-20B: An open-source autoregressive language model. In *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 95–136, virtual+Dublin. Association for Computational Linguistics.

Nat Gillin. 2022. Is encoder-decoder transformer the shiny hammer? In *Proceedings of the Ninth Workshop on NLP for Similar Languages, Varieties and Dialects*, pages 80–85, Gyeongju, Republic of Korea. Association for Computational Linguistics.

Yoav Goldberg. 2023. Some remarks on large language models. *Github Gist*.

Maggie Yundi Li, Stanley Kok, and Liling Tan. 2018. Don't classify, translate: Multi-level e-commerce product categorization via machine translation. *Workshop on Information Technologies and Systems*.

David H.D. Warren and Fernando C.N. Pereira. 1982. An efficient easily adaptable system for interpreting natural language queries. *American Journal of Computational Linguistics*, 8(3-4):110–122.

Meat-Based Graphics Pipelines

Will BL

March 2023

Abstract

Turns out humans can draw things and see things in images sometimes. Are they better than computers?

1 Introduction

GPUs have always been important in computer graphics. Recently, they have been used in more general forms of computation: in the past year there has been an explosion[1] in 'AI' text-to-image models. These neural networks, though they have very impressive capabilities, regardless have far-reaching consequences for their use. They can only be used by those who have access to modern GPUs[2]. They can be used to create convincing misinformation[3]. Their creators have also been accused of wholesale copyright infringement[4]. In addition, GPUs are getting more and more power-hungry. This makes usage of them not only for AI, but also for regular graphics programming, possibly unethical, as global warming continues to have horrible effects on the planet. Can we do better?

The human brain is a meat-based hardware with magic computational powers. Recent experiments show that it may even show some signs of intelligence, though this is likely overstated. The brain has a large inbuilt GPU[5]: a possible next-generation, low-energy, ethical graphics processor?

2 Prior Work

Image manipulation via brain processing[7] actually has a long history. However, it doesn't count because it wasn't done by TechBros.

The general idea of using human meat as a computer extension has been suggested before[L W99].

[1] Metaphorically.
[2] Gatekeep.
[3] Gaslight.
[4] Girlboss.
[5] In the occipital lobe.[6]
[6] In the back.
[7] 'Art' being the term of art.

3 Experiments

3.1 What is this "brain" thing anyway?

The ancient Greeks said, "Know Thyself"[8]. We must seek an understanding of brain. How do we understand brain? What is it, and why? Neuroscientists would say something. Psychologists would say something else. Their disagreement shows that both fields are contradictory and therefore worthless. We must instead do what any good computer scientist would do: ~~run Doom on it~~ benchmark it.

An experiment was devised to determine the computational power of the brain's GPU. The brain was first exposed to a GLSL shader, and was then tasked with creating the image it creates. The time taken for the brain to produce the output was capped at 60 seconds.

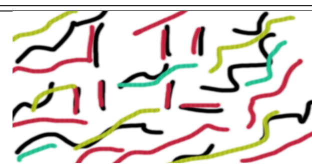
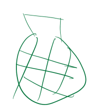
3.1.1 GLSL Shader

The following GLSL shader code was used:

```
uniform float u_time;

void main() {
  gl_FragColor = vec4(1.0,1.0,1.0,1.0);
}
```

3.1.2 Results

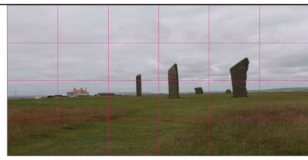
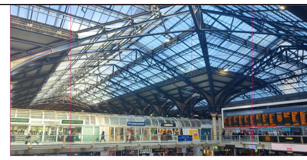
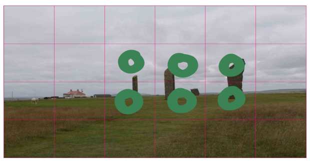
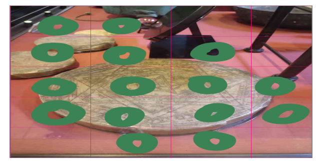
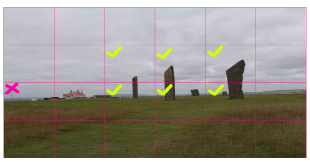
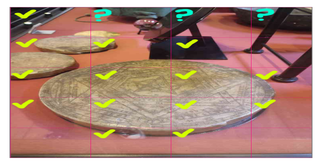
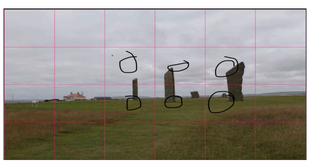
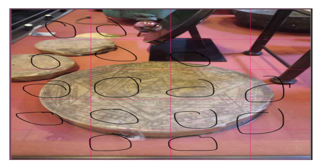
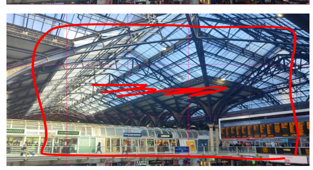
Time taken (seconds)	Output
60	
60	
60	help

[8] Well presumably they said it in Greek.

3.2 Object recognition

It turns out that the brain has inbuilt, hardware-enhanced support for object recognition in images[9]. We tested the capabilities of the brain in this task by giving it CAPTCHA-style tests.

3.2.1 Results

	Find Stone Circles	Find Waterloo Station	Find Stone Circles
Image			
Correct			
Brain 1			
Brain 2			
Brain 3			

3.3 Mentally Unstable Diffusion: text-to-image with brain hardware

We've all heard too much about 'AI' recently. Can we replace it entirely with brain[10]? We came up with some 'prompts' and fed them into some human brains. The brains were given exactly 60 seconds to produce an image. The resulting images were then rated subjectively[15] by a panel of critics[16].

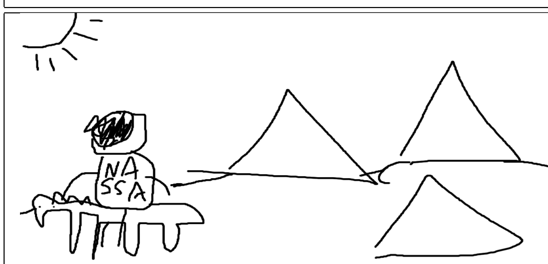
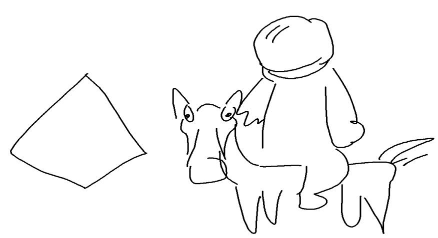
[9] This was not mentioned as a feature in the documentation.
[10] Those of us who wish to do so call ourselves "The Knights who say 'NI[11]!'"
[11] Natural[12] Intelligence
[12] i.e.[13] Non-Artificial
[13] ille est[14]
[14] That is
[15] This is still science, I swear.
[16] Which consisted of me.

The prompt used was:

```
an apollo astronaut riding a horse past the
great pyramids of giza high quality highest quality
trending on artstation sharp image no noise
```

Note that the prompt says 'the great pyramid*s* of giza'. This was totally on purpose to see how it affected the result, and not at all a mistake caused by me forgetting that there is only one 'Great Pyramid of Giza'[17].

3.3.1 Results

Output	Rating
	$\star\star\star$
	$\star$
	$\star\star\star\star$

[17]The other ones, though in Giza, aren't all that Great.

4 Discussion

4.1 Experiment 1

Brains are awful at compiling and processing GLSL shaders. None of them produced the correct output, and they all took the maximum available time of 60 seconds to actually produce any result. It would appear they are not useful for real-time applications such as gaming. Games should stick to the realm of cold metal machines, where they belong - not in the imagination of human beings.

4.2 Experiment 2

Brains are inconsistently good at recognising objects. They performed well at finding stone circles (circles of stones) and at finding stone circles (circles of stone). All brains thought they could see Waterloo station in an image of what is clearly Liverpool Street station. This disparity opens up many questions:

- Are brains only able to reliably detect objects which can be described as 'stone circles'?
- Are brains only able to reliably detect non-transportation-related objects? What does this imply for human driving of vehicles?
- Would brains get lost if trying to get around London?

4.3 Experiment 3

Some brains were better than others at creating an image from a text prompt. But wow, look at that horse's face in the third one! That thing looks awesome!

5 Conclusion

References

[L W99] L. Wachowski L. Wachowski. *The Matrix*. Film. 1999.

B

Well-known Problem, Meet Solution

Airport Security, Generalized Chess, and $\mathsf{NP} \neq \mathsf{P}$

Albert Gao / sixiangg@andrew.cmu.edu

Abstract

One of the timeworn traditions of coursework at Carnegie Mellon University is that of bonus homework or exam problems, so as to stimulate the motivation and diligence deeply hidden inside the student psyche. A relatively straightforward instance of such exercise given to first-year undergraduates is the problem of $\mathsf{NP} \stackrel{?}{=} \mathsf{P}$. Most intelligent and studious first-years are able to answer immediately in the negative and give "a truly marvelous demonstration of this proposition which this margin is too narrow to contain"[1]. Here we present an alternative method for the same problem. We stress that our approach, albeit less simple than existing techniques often discovered by brilliant students, can be interesting by itself and be applied to other problems.

1 Introduction

Originally proposed by [Coo71], the $\mathsf{NP} \stackrel{?}{=} \mathsf{P}$ problem often needs no introduction, since frequently it is given as an exercise in introductory computer science courses. For those familiar with the rich zoo of complexity classes, this section may be skipped.

We briefly give reminders of some definitions here so our later results are consistent with the notation. The class P is the class of languages L for which there exists a deterministic polynomial time Turing machine M such that $M(x)$ gives 1 if and only if $x \in L$. The class NP is the class of languages decided by polynomial time Turing machines in the nondeterministic model. Equivalently, $L \in \mathsf{NP}$ if and only if there exists a deterministic machine M with the following two conditions.

- For any $x \in L$, there exists poly-length y with $M(x, y) = 1$.
- For any $x \notin L$, for any poly-length y we have $M(x, y) = 0$.

The complexity class $\mathsf{EXPTIME}$ is the set of all languages decided by deterministic Turing machines in time $O(2^{p(n)})$, where $p(n)$ is a polynomial function in n. Note that this class of problems is crucial for our method, even though most usual solutions skip this definition and derive contradictions from $\mathsf{P} = \mathsf{NP}$ directly.

2 Preliminaries

Here we mention several well-known results.

Lemma 2.1. $\mathsf{AS} \in \mathsf{NP}$.

Proof. The problem of Airport Security (AS) is a classic example of a problem in NP. As in Figure 1, AS is a decisional problem that attempts to distinguish a person conducting terrorist activities from a person not conducting terrorist activities.

[1] Folklore indicates that the common approach to this exercise employs machinery from Fermat's solution to some archaic number theoretic proposition outlined in the margins of one of his books.

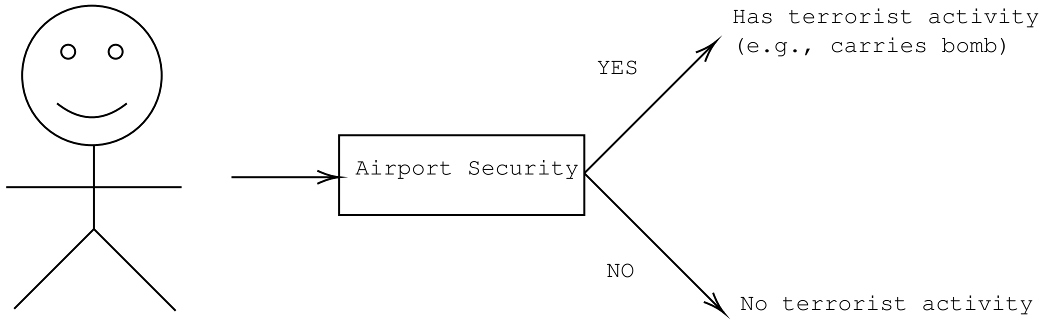


Figure 1: Illustration of problem AS.

The act of searching gives us a natural candidate for the witness string. That is, the terrorist act itself serves as y in our previous definition. If a person is a terrorist, we have a short proof of it that is, for example, a scanned image of a C-4 explosive. On the other hand, if a person is not a terrorist, it is difficult to succinctly prove this fact—we still need to search this person from toenails to hair strands anyway. But since there is no proof of terrorism for non-terrorists, this problem is naturally in NP. □

Lemma 2.2. GC *is* EXPTIME-*complete.*

Proof. Generalized Chess (GC) is shown to be EXPTIME-complete from [FL81]. □

3 Main Theorem

Theorem 3.1. $\mathsf{EXPTIME} = \mathsf{NP}$.

Proof. We establish that $\mathsf{EXPTIME} \leq \mathsf{GC} \leq \mathsf{AS}$, which shows the claim since $\mathsf{NP} \subseteq \mathsf{EXPTIME}$. Taken a GC instance and we are to determine if one party (call it "white") can force a win, we can have a diplomat Bob write out a proposal detailing a suggestion for a diplomatic game between two government parties according to the given GC instance. That is, we enforce the rules and positions specified in GC and have the enemy king be a political figure whose demise constitutes a terrorist act. A winning GC therefore corresponds with the diplomat's detailed plan for a terrorist act, yet a non-winning GC represents a friendly diplomatic game suggestion. Hence we can ask such diplomat to print out the GC instance and go through the AS black box to resolve GC. □

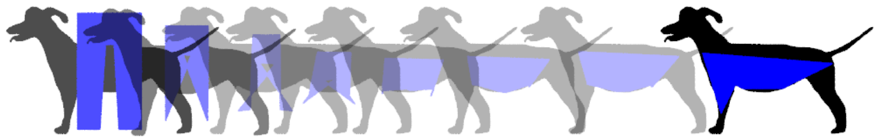
Corollary 3.1.1. $\mathsf{NP} \neq \mathsf{P}$.

Proof. $\mathsf{P} \subsetneq \mathsf{EXPTIME} = \mathsf{NP}$. □

References

[Coo71] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, 1971.

[FL81] Aviezri S Fraenkel and David Lichtenstein. Computing a perfect strategy for n× n chess requires time exponential in n. *Journal of Combinatorial Theory, Series A*, 31(2):199–214, 1981.



A Jalgorithm for Japplying Jeans to Jobjects

Dave Pagurek

March 2023

Abstract

For years, society has been plagued by the lack of a conclusive answer to the question of how arbitrary things would wear jeans, if they wore jeans. This paper proposes a jalgorithm to determine the mathematically optimal configuration of pants on a silhouette, leveraging recent insights in the fields of computer vision and differentiable rendering.

1 Introduction

In recent years, the internet has been ruminating over the question of how various objects would wear pants, if they hypothetically wore pants. [2] It began in 2015 using a dog as the object in question, [3] with other objects such as humans on their hands and knees [12] or pants themselves [4] being discussed later.

Despite being in the public discourse for nearly a decade, no conclusions have been reached. The authors believe this is due to a lack of a principled approach to the study of pants-wearing. This paper aims to rectify the situation by proposing a formula to score the quality of the application of pants to an object, and writing software to optimally apply pants under this system.

2 Background

Gradient-based optimization has been a subject of study in mathematics for a long time. The core idea is that if one has an objective function one wishes to maximize or minimize, one can use the gradient of the output with respect to the function's inputs to determine how to adjust the inputs in order to move the output closer to the desired outcome. There are many algorithms, including Adam optimization [5], to update inputs in this manner using gradient information.

Differentiable rendering is a technique in the world of computer graphics and computer vision that enables gradient-based optimization of functions involving rendering of images. If the rendering process is itself differentiable, then one could, for example, optimize the representation of a 3D scene such that it matches a reference photo when rendered. [7, 8] The diffvg library [6] provides a differentiable vector graphics renderer, allowing one to optimize 2D vector shapes such as lines, curves, and polygons to produce desired visuals.

3 Method

3.1 Algorithm

Our pants-fitting jalgorithm takes as input a vector shape representing a silhouette on which one wants

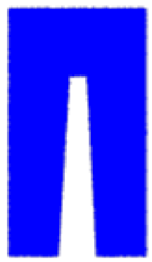


Figure 1: Prototypical jeans: $\mathbf{X}$, the initial pants shape we apply to objects.

to apply pants. We rasterize this to a 256×256-pixel image, concerning ourselves only with the alpha channel. We will refer to this one-channel silhouette image as the 256×256 matrix S.

We then create a polygon representing the pants we want to fit onto the silhouette, whose vertex positions X we will optimize. We initialize its vertices to $\mathbf{X}$, the shape of prototypical jeans, shown in Figure 1. We then solve for the vertices $\hat{X}$ that minimize pants-fitting loss, $l(X)$, described in detail in Section 3.2, using Adam optimization [5] and gradients provided via diffvg [6] vector rendering and Torch [11] for 100 iterations:

$$\hat{X} = \text{argmin}_X l(X) \tag{1}$$

3.2 Loss Function

Our loss function is motivated by insights from the authors' multuple decades of pants-wearing experience. We propose the following rules, which we then will encode mathematically:

1. **Duality of man:** Pants should bisect the silhouette of the wearer through its center, covering half its area.
2. **No-stretch jeans:** Pants, when applied to the wearer in rest position, should not be so deformed that they no longer resemble prototypical jeans.
3. **(Topologically) torn jean avoidance:** Limbs should only go through the open ends of pants, and similarly, closed ends of pants should not intersect limbs.
4. **Dress code compliance:** Pants should be on the body of the wearer, not off of it.

From Rule 1, we get the following loss terms, where x_{topleft} and x_{topright} correspond to the endpoints of the waistband of prototypical jeans, and $J(X)$ is the 256×256 matrix of the rendered alpha channel of the jeans vertices $\mathbf{X}$. L_{center} incentivizes waistbands that go through the center of the silhouette, and L_{coverage} incentivizes jeans that cover half of the silhouette.

$$L_{\text{center}} = \left\| \frac{x_{\text{topleft}} + x_{\text{topright}}}{2} - \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \right\|^2 \tag{2}$$

$$L_{\text{coverage}} = \left(\frac{1}{2} - \frac{\sum_{i,j} \max\left((S - J(X))_{ij}, 0 \right)}{\sum_{i,j} S_{ij}} \right)^2 \tag{3}$$

From Rule 2, we get the loss term L_{rigid}, which incentivizes the jean points X to be as close as possible to an affine transformation of the prototypical jean points $\mathbf{X}$ by finding the error of the least-squares-minimal transformation T that describes the relation between X and $\mathbf{X}$.

$$L_{\text{rigid}} = \|X - T\mathbf{X}\|^2, \tag{4}$$

$$T = \begin{bmatrix} \hat{p}_1 & \hat{p}_2 & \hat{p}_3 \\ \hat{p}_4 & \hat{p}_5 & \hat{p}_6 \\ 0 & 0 & 1 \end{bmatrix}, \tag{5}$$

$$\hat{p} = \text{argmin}_p \left\| \begin{bmatrix} \mathbf{X}_{0,1} \\ \mathbf{X}_{1,1} \\ \vdots \\ \mathbf{X}_{0,n} \\ \mathbf{X}_{1,n} \end{bmatrix} - \begin{bmatrix} X_{0:1,1} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & X_{0:1,1} & 1 \\ & & \vdots & & \\ X_{0:1,n} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & X_{0:1,n} & 1 \end{bmatrix} p \right\|^2 \tag{6}$$

From Rule 3, we get the terms L_{open} and L_{closed}, terms incentivizing open and closed edges of the jeans respectively to and to not intersect the silhouette, and L_{inter}, incentivizing edges to not intersect each other. We refer to the set of all edges as $E(X)$, and the of open and closed edges as $E_o(X)$ and $E_c(X)$. The rendered alpha channel of an edge with stroke thickness of 5 is referred to as $R(e)$.

$$L_{\text{open}} = \frac{-\sum_{i,j} R(E_o(X))_{ij} S_{ij}}{\sum_{i,j} R(E_o(X))_{ij}} \tag{7}$$

$$L_{\text{closed}} = \frac{\sum_{i,j} R(E_c(X))_{ij} S_{ij}}{\sum_{i,j} R(E_c(X))_{ij}} \tag{8}$$

$$L_{\text{inter}} = \sum_{\{e_1,e_2\}:\binom{E(X)}{2}} \frac{\sum_{i,j} R(e_1)_{ij} R(e_2)_{ij}}{\sum_{i,j} \min\left(R(e_1)_{ij} + R(e_2)_{ij}, 1\right)} \tag{9}$$

From Rule 4, we get L_{inside}, discouraging the pants from being off the silhouette:

$$L_{\text{inside}} = \frac{\sum_{i,j} \max\left(J(X)_{ij} - S_{ij}, 0\right)}{\sum_{i,j} J(X)_{ij}} \tag{10}$$

We take a weighted sum over all the loss terms to get the final loss function:

$$l(X) = \sum_i k_i L_i \tag{11}$$

For our results, we used the following weights:

$$\begin{aligned} k_{\text{center}} &= 50, \\ k_{\text{coverage}} &= 100, \\ k_{\text{rigid}} &= 5, \\ k_{\text{open}} &= 1, \\ k_{\text{closed}} &= 1, \\ k_{\text{inter}} &= 50, \\ k_{\text{inside}} &= 1 \end{aligned}$$

4 Results and Analysis

4.1 Validation

To verify that the algorithm outlined in Section 3 operates as expected for known cases, we ran it on the silhouette of a human. Figure 2 shows that the algorithm placed the jeans on the legs of the human, which the reader may recall is the standard configuration of human jeans.

Figure 2: Placement of jeans on a human via the algorithm, matching conventional wisdom of where jeans belong on a human.

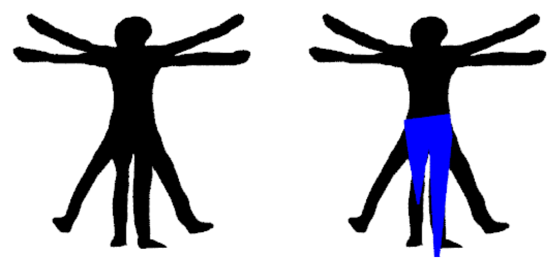


Figure 3: Placement of jeans on the Vitruvian Man, the body shape we all aspire to have.

While this is only a sample size of one, there is in fact only one species for whom jeans are originally designed, so we feel confident in our ability to extrapolate from this to other figures. However, as a gift to the reader, we test on one more human-shaped validation input: Leonardo da Vinci's *Vitruvian Man*. Conceived as a depiction of ideal body proportions, [1] the octopedal body plan represents what we all aspire to look like. Figure 3 shows the algorithm confidently placing jeans on two of the four legs, fashionably rolling up one leg in a manner reminiscent to one-strapping a backpack. [9]

Figure 4: Placement of jeans on a dog via the algorithm. The jeans are placed on all legs, but in an interesting deviation from the original hypothesis, the jeans leg for the rear legs has ridden up to align with the derrière.

4.2 Novel Inputs

We next tested the algorithm on a dog in an attempt to settle the original debate. The original image presented two hypotheses for dog jean placement: on the rear legs only, subdividing the dog into two horizontal sections, or on all legs, subdividing the dog into two vertical sections. [3] Figure 4 shows that the algorithm places the jeans in a configuration much more like the latter than the former, but with a twist: rather than form-fitting the rear legs, the algorithm chose to collapse the rear pants leg so that it ends at the derrière of the dog.

In Figure 5, we present jeans applied to some more body shapes. First, one best described as horse-inspired (the literature shows that one likely cannot draw a good horse without reference material present [10]) dons a single pant leg. Here, the algorithm demonstrates that it would rather shrivel up a leg into nothing than stretch a waistband across any part of a horse. Next, a three-legged alien wears jeans on one and a half legs, with a small branch of jean reaching up to its shoulder for support. This demonstrates the algorithm's ability to balance its creative approach to jean design with functional concerns, like extra shoulder support. Finally, a spiky ball dragon wears jeans on only one of its ball legs, further demonstrating a preference for one-strapping.

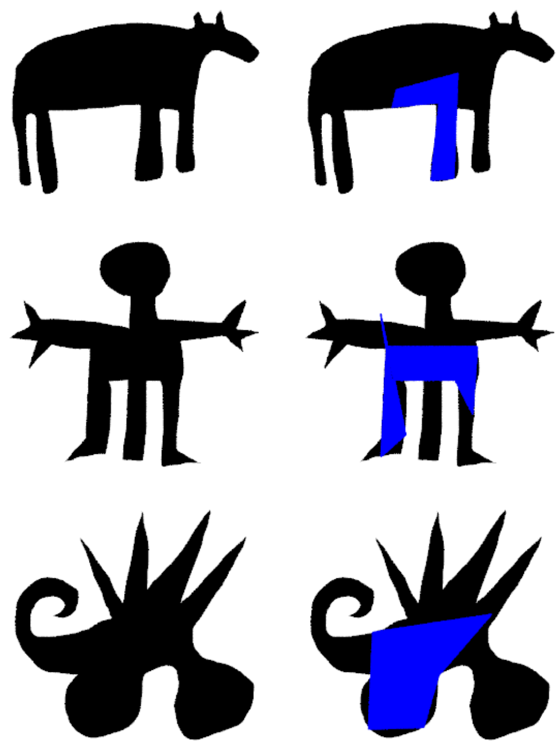


Figure 5: Jean placements on alternative body shapes. Note the creative one-strap overall look of the tripod, and the bold asymmetric style of covering one ball of the ball dragon.

5 Conclusion

We have presented a mathematically sound optimization approach to determining the optimal configuration of pants on any object, should it choose to wear pants. Now that a definitive answer exists to this question, we expect no further arguments to be made on the subject on the internet. Since the algorithm's outputs are, by definition, optimal, one may choose to take lessons from its output, and begin casually rolling up just one pant leg in one's outfits.

References

[1] Vitruvian man. *Wikipedia*. URL: https://en.wikipedia.org/wiki/Vitruvian_Man.

[2] If a dog wore pants. *Know Your Meme*, Dec 2015. URL: https://knowyourmeme.com/memes/if-a-dog-wore-pants.

[3] If a dog wore pants would he wear them like this or like this? *Utopian Raspberry - Modern Oasis Machine*, Dec 2015. URL: https://www.facebook.com/1ur.mom/photos/a.535344933281912.1073741828.535340893282316/604404026376002/?type=3&theater.

[4] @bobby. quick question. *Twitter*, April 2017. URL: https://twitter.com/bobby/status/849491087005810688.

[5] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL: http://arxiv.org/abs/1412.6980.

[6] Tzu-Mao Li, Michal Lukáč, Gharbi Michaël, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 39(6):193:1–193:15, 2020.

[7] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2019.

[8] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. A general differentiable mesh renderer for image-based 3d reasoning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[9] Phil Lord and Christopher Miller. 21 Jump Street. 2012.

[10] Dave Pagurek. Can you draw a horse without a reference? Jul 2022. URL: https://davepagurek.github.io/horse-drawings/.

[11] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[12] /u/flooid. If a man wore pants... *Reddit*, Dec 2015. URL: https://www.reddit.com/r/funny/comments/3yosk1/if_a_man_wore_pants/.

On the Origin of Sandwiches: A Revised Theory

Dr. Esquire ZeWaka & Dr. "The Big" Jablonski

How does one define a Hotdog? Can it be considered a Sandwich? These are hotly contested debates in the memeology & scientific communities. Many authors around the world have proposed theories that *claimed* to conclude this debate - with exceptions. However, we put forth an entirely superior theory that eliminates the need for silly edge cases and integrates multiple perspectives.

This paper seeks to propose a simple solution: solving the mystery of bread-like constructions through the use of vector addition and taxonomy. It should be noted that all bread-like constructions are differentiable Reimann-2 manifolds. As a result, one can take the sum of all normal vectors to all bread-like material present in the construction, which we will henceforth call the **ideal vector sum**. Moreover, all compositions must be evaluated as if they were sitting on a flat level surface with all ingredients included, as if to be presented - as this is how we perceive such objects in their ideal form.

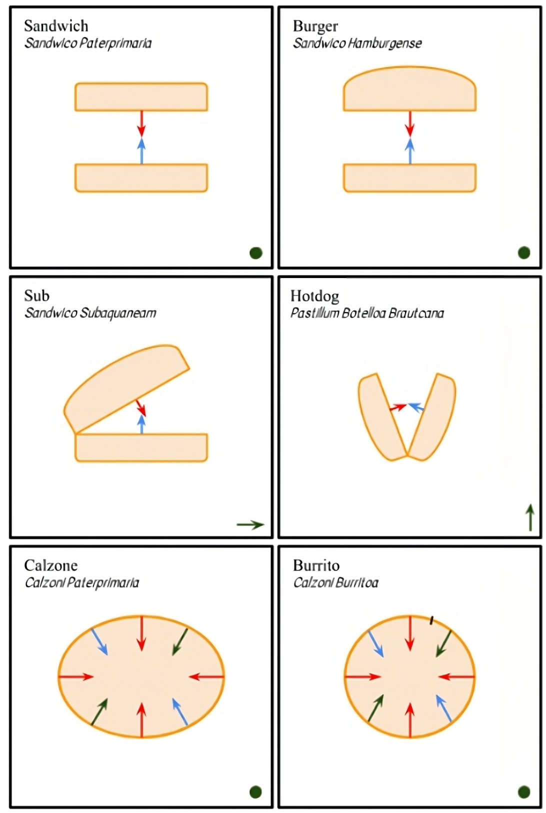


Figure 1: Example vector sums of various bread-containing food items.

Once the sum has been taken, one can determine the classification of the construction. In this theory, there are three main taxonomic families of bread-like constructions: *Sandwicos*, *Pastillum Botellos* (P.B.)[1], and *Calzoni*[2].

The ideal vector sum, $\sigma = \sum_{\hat{\mathbf{n}}}$ where $\hat{\mathbf{n}}$ having magnitude $|\hat{\mathbf{n}}|$ and angle θ_n is a normal vector, must satisfy the property $|\theta_\sigma| < \frac{\pi}{4}$. The vectors are oriented such that $\theta = 0$ is parallel with respect to plane of presentation. In addition, for pure constructions, $\sigma = \overrightarrow{0}$. When both segments of bread-like material are parallel planes, we are in the form *Sandwico Paterprimaria*. If the upper bread-like plane has a curved top surface S, formally defined as both $\frac{\partial^2 S}{\partial x^2} < 0$ and $\frac{\partial^2 S}{\partial y^2} < 0$, it is of the genus *Hamburgense*. However, if $|\sigma| \neq 0$, one can classify the *Sandwico* as a *Sandwico Subaquaneam*.

A P.B.'s vector sum will add up to being within $\frac{\pi}{6}$ of the vertical axis, that is, $\left|\sigma - \frac{\pi}{2}\right| < \frac{\pi}{6}$, pointing upwards relative to the plane of construction. Additionally, it is necessary that $\sigma \neq \overrightarrow{0}$ as well.

The vector sum of a *Calzoni* should be such that $\sigma = \overrightarrow{0}$. Moreover for pure constructions, the topology of a *Calzoni*, C, should be homeomorphic to a sphere[3], that is, $C \cong S^2$. However, if said *Calzoni* is made of one bread-like plane but $C \neq S^2$, it can be said that the *Calzoni* is of the genus *Burritoa*.

Referring to Figure 1, we can see that the Sandwich, consisting of two perpenicular bread-like planes, is indeed *Sandwico Paterprimaria*. Similarly, the Burger, with its upper-bread-like plane having a curved top surface, is of the genus *Hamburgense*. The Sub, while not consisting of parallel planes, maintains $|\theta_\sigma| < \frac{\pi}{4}$, but with $|\sigma| \neq 0$. The Hotdog in Figure 1, the point of contention for this paper, has an ideal vector sum with angle $\frac{\pi}{2}$ placing it firmly within the family *Pastillum*[4]. The Calzone is of elipsoidal shape, and is therefore of equivalent topology to S^2, making its taxinomical classification *Calzoni Paterprimaria*. Burritos are not homemorphic to S^2, unlike Calzones (as indicated by the line through the cross-section in the figure). $\sigma = \overrightarrow{0}$ for this example as well, and it consists of one bread-like plane, consistent with the genus *Burritoa*.

In conclusion, if one properly applies this theory to a hotdog, one can surmise that **a hotdog is indeed, not of the family *Sandwico***.

[1] The word "hotdog" is hotly contested in the taxonomic community. One can define tacos and other similarly-shaped constructions as of the same substance. Therefore, the only proper taxonomic classification one can use is the whole family of *Pastillum Botelloa*, with *Brautcana* being a genus within.

[2] There is debate within the community as to the true taxonomic name of the *Calzoni* family. Some consider the family to be named *burritoa* or *empanæ*, but this article will be sticking to latin roots.

[3] Having the topology of S^2 is a sufficient condition for $\sigma = 0$

[4] The family *Pastillum* also includes genera such as *Tacoa* and *Shwarmæ*.

Is the number of Falco lasers ever shot greater than the number of humans alive?

Arya Amami

December 26, 2022

Abstract

Approximately two thirds of the *Super Smash Bros. Melee* community believes that there have been more Falco lasers shot than humans are currently alive (c. 8 billion). While his laser is a core part of Falco's kit and a great deal of *Melee* has been played over the past 2.1 decades, *Melee* players (and in fact humans at large) tend to underestimate the size of numbers on the magnitude of the world population. We show that, despite this, it is not unlikely that the comparatively small *Melee* community has been able to overwhelm the human race with lasers. Many calculations of the exact numbers of lasers shot rely on highly uncertain parameters and estimations, often spanning up to a full order of magnitude. When making calculations, taking this uncertainty into account, we find that it is very plausible that more Falco lasers have been shot than there are humans alive today.

1 Introduction

On 13 October 2022, *Melee* commentator Toph posed a novel Fermi Problem on Twitter[1]: "Are there more humans alive on the planet today, or Melee Falco lasers shot across all time?" The question sparked heated discussion, as no single answer appeared to be satisfactory. Many users provided calculated estimates using ad-hoc numbers, some resulting in numbers higher than 8 billion and many resulting in numbers significantly lower than 8 billion. These estimates rely on numbers often produced without justification or make improbable extrapolations of populations (e.g. using the number of copies of *Melee* ever sold without adjusting for the uneven distribution of competitive and non-competitive players). A reliable estimate needs to use empirical statistics wherever possible and must model uncertainty in such a manner that we may predict the likelihood that we live on a planet with a laser-per-capita value greater than or equal to 1.

1.1 The Lombardi Equation

We have developed an equation for calculating the number of lasers ever shot since *Melee*'s release and we can phrase it as the product of five factors:

$$N = n_l n_g n_p f_f D$$

Where: n_l is the number of lasers shot by a Falco per Falco-player-game, n_g is the number of games played per player per day, n_p is the number of *Melee* players playing per day, f_f is the fraction of *Melee* players who play Falco, and D is the number of days of modern Falco play[1].

While n_l, f_f, and D are very straightforward to calculate, finding accurate values of n_g is nontrivial as a result of the author's highly limited social network and reliable statistics are not available to calculate n_p. Though we will provide heuristic estimates of n_p, it would be wise to allow this value to remain variable and consider on a vibes-basis how plausible having such a number of players playing daily would be.

1.2 Codifying uncertainty in parameters

The novel-ish approach that we present is to embrace the uncertainty of the parameters by using interval arithmetic to define a lower and upper bound for each of the parameters. Furthermore, blindly assuming that our uncertainty is uniform over each interval and uncorrelated with other parameters, Monte Carlo simulation can be utilized to find the approximate likelihood of a laser-dominated world.

1.3 Dataset

We have collected two separate datasets for analysis: one featuring play around the skill level of that of the author[2], containing 1,054 Falco games, and one featuring play from several tournaments since 2019, containing 369 Falco games. This second dataset is filtered for games played onstream, which we assume probably skews the dataset towards high and top-level players, we guess. It is of great utility to have separated datasets between skill brackets as Falco behaviors deviate non-negligibly

[1] "Modern Falco play" is defined as the number of days since Falco players started shooting, on average, n_l lasers per game.

[2] i.e. aggressively middling

between players of various skill brackets, and these can be used as bounds for parameters in the Lombardi equation.

2 Defining parameter values

2.1 n_l: Lasers per Falco per Falco-player-game

Analysis of the low-mid-level dataset yields the result that the average Falco shoots a median of 24.2062 lasers per game. The high-top-level dataset, by comparison, shows that top Falcos shoot an average of 37.0737 lasers per game, over 50% more than their skill-deficient counterparts. This is actually very close to the low-mid-level $\mu + 1\sigma$ value of 37.7273. The true average across all players is probably likely to be between the values for low-mid players and high-top players.

$$n_l = [24.2062, 37.0737]$$

2.2 f_f: Fraction of Falco players

In calculating this value one must consider that the distribution of Falco players across levels of skill is highly uneven. For our estimates, we can ignore the proportion of players who only play casually using a technique known to statisticians as "not how statistics works." This assumes that the amount of lasers shot by casual players is negligible due to both their lack of per-player playtime and the anecdotally low pick rate of Falco in casual play. Public perception of character pick rates is that Falco sees far more use in low and mid-level play than in high-level play, and analysis shows this to be true. Within the dataset of low-mid-level play, players pick Falco on average 23.83% of the time. By contrast, an analysis of 152 players ranked on global top-100 or top-50 lists since 2018 shows that only 17.11% of top-level players play Falco competitively at all. Incorrectly assuming that each player plays their listed characters proportionally (i.e. a player with Fox and Falco listed plays each half the time), top players pick Falco a vanishing 12.83% of the time. The true overall average of Falco pick rate is likely to be between these two values and thus they can serve as upper and lower bounds for f_f.

$$f_f = [0.1283, 0.2383]$$

2.3 D: Number of days of modern Falco play

A naïve estimation of the number of days of modern Falco play is the number of days since the release of *Melee* (as of writing, 7,705 days). However, owing to the utter lack of development of a metagame for probably at least a week or two, it may be best to set a conservative lower bound on this value to account for the time needed for competitive Falco players to achieve full laser potential. As such, we have made the utterly arbitrary decision to set this lower bound as the number of days since Jack Garden Tournament (as of writing, 6,337 days). This tournament was the international breakout performance of Japanese Falco player Bombsoldier, who is often considered to be among the progenitors of modern Falco play.

$$D = [6\,337, 7\,705]$$

2.4 n_g: Number of games played per player per day

Finding values for this parameter of sufficient statistical signficance is very difficult as we have a sample size of two (2) players, for whom the average number of games per day is 15.3151. This includes days of zero games played but arbitrarily excludes days during gaps of greater than or equal to six days. Despite literally no evidence to back this claim, we believe this to be a somewhat above-average amount of games played per player (vibes). That being said, this value also only accounts for Slippi play and does not include LAN play on non-Slippi media. For this reason, we believe that this absurdly-low statistical power result will suffice as an average from which to derive the lower and upper bounds of this parameter. Cancels out, or whatever. Assuming that less-active players play 50% less than average and more-active players play 50% more than average, this yields a lower bound of 7.6576 and an upper bound of 22.9727. We are sure the true average is within this range, probably.

$$n_g = [7.6576, 22.9727]$$

2.5 n_p: Number of *Melee* players playing per day

This is the most difficult parameter to find a value of as there is no single centralized place to play *Melee*. The next closest thing is Slippi, but this is far from centralizing as a great deal of *Melee* happens in LAN settings and many players, especially those with robust social circles or easy access to other players, play on Slippi rarely, if ever. In any case, *Melee* magnate Fizzi36 has yet to release active-player statistics.

While not defined by usership, another somewhat-centralizing pillar of the *Melee* community is its major tournament livestreams. Using this as an estimate for *Melee*'s playerbase is highly imperfect, as viewership numbers include both a great number of non-player spectators and exclude a great number of non-spectator players (largely Europeans). Despite this uncertainty, it remains the best basis for estimates of this parameter.

If we model the playerbase of *Melee* as increasing linearly for the last D days and the current playerbase as the global maximum, then it suffices to define the average active playerbase as one half of the current active playerbase. For the purposes of defining a lower and upper bound on the number of players, we select two tournaments whose viewership numbers will be the basis for calculations. For the lower bound, we select The Big House 10, a multi-game tournament whose *Melee* viewership is high but not record-breaking, at 76,070 peak viewers. For the upper bound, we select Smash Summit 11, an invitational tournament whose peak viewership is higher than any other *Melee*-only tournament, at 116,972 peak viewers. Using a conservative estimate of 25% being the proportion of tournament viewers who actively play *Melee*, we find the bounds of active players to be $[19\,018, 29\,243]$, which can be halved to yield the average active playerbase.

It is worth noting that these numbers are significantly lower than the current number of unique players whose replays are registered on Chart.slp[2] (currently 85,630), suggesting these numbers may be varyingly conservative depending on how strongly factors like account decay[3], registration proportion[4], and non-Slippi players are weighed.

$$n_p = [9\,509, 14\,622]$$

3 Calculation of N

Evaluation of the Lombardi equation with the parameters we have defined yields the following interval:

$$L = [1\,433\,221\,092, 22\,867\,523\,353]$$

[3]It is highly unlikely that every account registered on the website continues to play actively indefinitely.

[4]Although the proportion of active players who have registered their replays onto the site is likely to be low, each player having played with a large number of unique opponents means that every player brings many others with them onto the site, much like how a virus can infect those within physical proximity of an infected person.

The world population is contained within this range, and in fact, the geometric mean of the upper and lower bounds is less than 30% lower than the world population. However, this should not be taken as conclusive evidence that Falco players have failed to best the human race's capacity to reproduce (and to be sure, this entire study should not be taken as conclusive of anything at all, really). Using the Monte Carlo method shows that 35.951% of parameter configurations result in a value of L greater than 8 billion. However, this is using the speculative estimate of n_p, the most uncertain parameter.

We can fix L at 8 billion and treat n_p as an unknown to find the lower and upper bounds for at least how high n_p would need to be for L to surpass the world population. Doing so, we find the bounds to be $[53\,084, 5\,116]$. While the lower bound implies a number of currently active *Melee* players greater than the figure of 85,630 suggested by Chart.slp, the upper bound is a mere tenth of that figure.

4 Limitations

By far the most salient limitation of this study is that we are not a statistician and are likely to be perversely wielding hallowed tools of mathematics in a manner most profane.

Furthermore, several of the parameters of the Lombardi equation have either high levels of uncertainty as discussed in each section, bias in sampling, or both. Data greatly underrepresents players who have achieved the incredible feat of having a level of skill lower than that of the author and depending on how numerous this population is this could greatly lower values for the parameter n_l. Additionally, not at all accounted for in this study is the incidence of games including more than two players, which is likely to both increase n_g and decrease n_l.

Finally, a critical semantic ambiguity exists in Toph's question: In referring to lasers, does one mean precisely the projectile produced by Falco's neutral special attack, or does one refer to any laser projectile produced by Falco's blaster? This study assumes the former but if the latter is to be considered then figures for n_l may be significantly higher. Falco's back-throw and up-throw cause him to shoot three lasers and his down-throw causes him to shoot four lasers. While not nearly as ubiquitous as Falco's neutral special, his throws are indeed still an important part of his kit and see a good deal of use.

5 Conclusion

To answer Toph's seminal tweet plainly: Probably more lasers, I guess. While the results do not categorically demonstrate if there have been more lasers shot or if there are more humans alive, we demonstrate that reasonable bounds for the problem lie well within the order of magnitude of the world population, and as such it is not unreasonable to think that there may have been more lasers shot.

Competing Interests: We declare no conflict of interest as we are precisely down neutral, ergo free of material desire.

Acknowledgments: We would like to acknowledge PapaNoJohns for his contribution in the way of Slippi file statistics and Fizzi and the rest of the Slippi team for their work on the Slippi project. Other acknowledgments include A. Sandberg et al. (2018), the Future of Humanity Institute, Othello Gang, and Ryo Yamada.

References

[1] Kristopher Aldenderfer. *Are there more humans alive on the planet today, or Melee Falco lasers shot across all time?* `https://twitter.com/toph_bbq/status/1580778946978664449`. Oct. 2022.

[2] Geoffrey Lao. *Global Stats*. `https://chartslp.com/global`. Dec. 2022.

; ;

Maximizing Code Readability Using Semicolon Indentation

Tessa Parker
Illinois Institute of Technology
tparker7@hawk.iit.edu

Nelitha Kulasiri
Carnegie Mellon University
nkulasir@andrew.cmu.edu

;;;;**1**;;**Introduction**

;;;;;;;;Since the dawn of computer programming, there have been people who wish to impose their preferences on stylistic choices in code on others. This paper draws inspiration from the imposition of certain stylistic choices on others.

;;;;**2**;;**Current Practices**

;;;;;;;;Currently, many programmers choose to use one of two types of indentation in their code: tabs or spaces. These two choices are the source of intense debate within computing communities. There is no doubt that these debates have caused rifts in working relationships and potentially lead to divorces and break-ups in romantic relationships[1].
;;
;;;;;;;;Though these practices have been mostly pedantic for many languages, there have been some attempts to use this pedantry in novel languages[5]. These attempts are rather useful to learn from, as we can see how white space can be used.

;;;;**3**;;**A New Approach**

;;;;;;;;This paper suggests a new approach to delimiting logic within code: the usage of semicolons for indentation. This may be achieved by simply replacing any tabs or spaces at the start of a line with semicolons. Examples of usage are listed later on in this paper.

---

[1]This is shown in Season 3 Episode 6 of *Silicon Valley*. No actual cases of this phenomenon occurring have shown up after a brief internet search.

;;;;4;;Benefits

;;;;;;;;4.1;;Reduced Ambiguity

;;;;;;;;;;;While determining a new way to delimit logic in programs, it is important to ensure that the delimitation is not ambiguous. The current practice of using spaces or tabs has the disadvantage that it is not easy to determine, from a glance, which delimiter is being used, as they are both white space characters. [Make this less ambiguous]

;;;;;;;;4.2;;Increased Consistency

;;;;;;;;;;;;One common issue in large codebases, such as an open-source project, is consistency in how code is indented. Everyone has their preferences, and editors may default to one way of formatting, which likely conflicts with other contributors to the project. In fact, this has become such an issue that there are style enforcement tools dedicated to enforcing consistency in code style, including indentation. The industry has arbitrarily decided that

;;;;;;;;4.3;;Better Readability

;;;;;;;;;;;;Many programmers know the pain of having to read code printed on paper. Especially in the academic setting, code is printed on paper, potentially splitting pages. In this situation, indentation context may be lost. Students may be forced to find the ending brackets and hope they properly understand the indentation, especially in languages with significant indentation, such as Python.
;;;;;;;;;;;;The practices introduced in this article solve this issue by using a non-whitespace character to indent code. An example of Python code being used in a manner that would solve this issue is found in section 5.1 Python.

;;;;5;;Usage

;;;;;;;;The principles introduced in this proposal are relatively simple to use in code. This section will detail how this proposal can be implemented in different programming languages. Though not all languages currently support this implementation, workarounds are provided for those that don't.

;;;;;;;;5.1;;Python

;;;;;;;;;;;;Unfortunately, the principles introduced in this proposal are not immediately possible to implement in Python. However, users have come forward with suggestions of how to use the same concepts in their Python code. This method also provides the added bonus of removing any unintentional behavior of the code.

```
def write_research(journal_name):
##print(f"Writing research for {journal_name}")
##for i in range(10000):
####add_random_word();
##add_title_page("A Paper")
##add_broken_link("this link doesn\'t work, this is unintentional")
write_research("Sigbovik 2023")
```

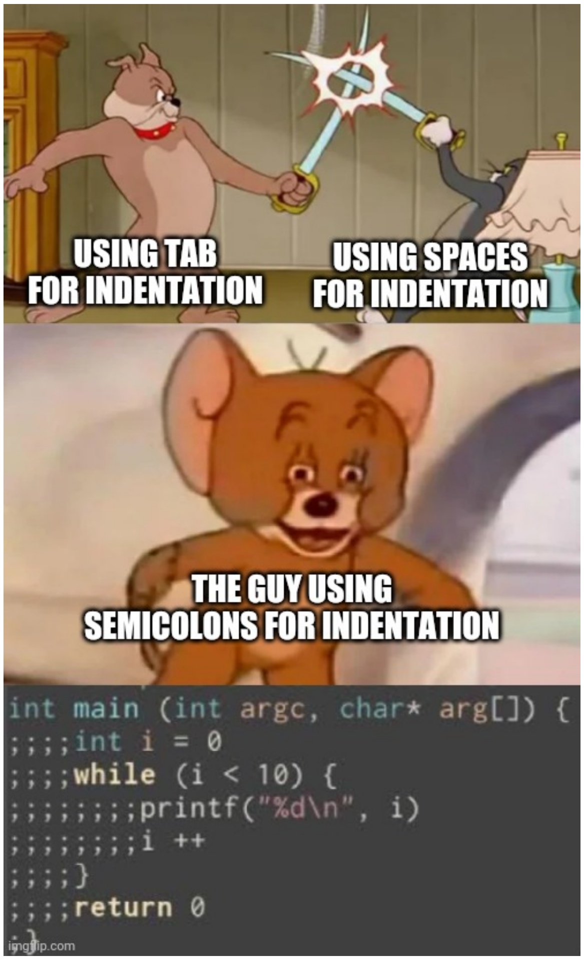
;;;;;;;;**5.2**;;**JavaScript**

;;;;;;;;;;;;With JavaScript, the practice of omitting semicolons from the ends of lines is one that may be divisive when implementing these principles. One of the primary controversies within JavaScript circles is whether to use semicolons or not, as JavaScript has optional semicolons. ;;;;;;;;;;;;This proposal does not specify whether semicolons should be used at the end of lines, so as to not be controversial in this space. With this in mind, we can see this proposal in practice with a snippet of the elevator.js library[3], with and without end-of-line semicolons.

```
 // With end-of-line semicolons
;;;;element.attachEvent("onclick", function() {
;;;;;;updateEndPosition();
;;;;;;document.documentElement.scrollTop = endPosition;
;;;;;;document.body.scrollTop = endPosition;
;;;;;;window.scroll(0, endPosition);
;;;;});
// Without end-of-line semicolons
;;;;element.attachEvent("onclick", function() {
;;;;;;updateEndPosition()
;;;;;;document.documentElement.scrollTop = endPosition
;;;;;;document.body.scrollTop = endPosition
;;;;;;window.scroll(0, endPosition)
;;;;})
```

;;;;;;;;**5.3**;;**C**

;;;;;;;;;;;;This paper was inspired by the elegant C code presented by user ablaniar on Reddit r/programminghumor[6]. Based on this code sample, we can see that this original user has decided to take the side of omitting end-of-line semicolons from their code.



;;;;;;;;;;;The following is the sample shown above, but including end-of-line semicolons.

```
int main (int argc, char* arg[]) {
;;;;int i = 0;
;;;;while (i < 10) {
;;;;;;;;printf("%d\n", i);
;;;;;;;;i ++;
;;;;}
;;;;return 0;
}
```

;;;;;;;;**5.4**;;**C#**

;;;;;;;;;;;This is a short example of a sorting algorithm in C#. It's amazing we found this code, thanks to code-maze.com for this one.

```
public int[] SortArray()
{
;;;;var n = NumArray.Length;
;;;;for (int i = 0; i < n - 1; i++)
;;;;;;;;for (int j = 0; j < n - i - 1; j++)
;;;;;;;;;;;;if (NumArray[j] > NumArray[j + 1])
;;;;;;;;;;;;{
;;;;;;;;;;;;;;;;var tempVar = NumArray[j];
;;;;;;;;;;;;;;;;NumArray[j] = NumArray[j + 1];
;;;;;;;;;;;;;;;;NumArray[j + 1] = tempVar;
;;;;;;;;;;;;}
;;;;return NumArray;
}
```

;;;;;;;;5.5;;Java

;;;;;;;;;;;;Java earns a special place in this paper, purely because of its widescale usage in apps. [Why is Java actually good? Add some praise here]

```
class Main {
;;public static void main(String[] args) {
;;;;System.out.println("Hello world!");
;;}
}
```

;;;;;;;;5.6;;Brainfuck

```
[squares2.b -- compute square numbers
(c) 2016 Daniel B. Cristofani
http://brainfuck.org/]

>>>>>>>>>>+>++<[
;;;;[[<<+>+>-]++++++[<++++++++>-]<-.[-]<<<]
;;;;++++++++++.[-]>>>>>[>>>>]<<<<[[<<<+>+>>-]<<<-<]>>++[
;;;;;;;;[
;;;;;;;;;;;;<<<++++++++++[>>>[->>+<]>[<]<<<<-]
;;;;;;;;;;;;>>>[>>[-]>>+<<<<[>>+<<-]]>>>>
;;;;;;;;]<<-[+>>>>]+[<<<<]>
;;;;]>>>[>>>>]<<<<-<<+<<
```

```
]

This program outputs square numbers. It doesn't terminate.
```

;;;;;;;;**5.7**;;**Rust**

;;;;;;;;;;;;As one of the most trendy languages at the time of writing, Rust has become immensely loved by its users. This code below is open source[4].

```
fn main() {
;;let cli = Cli::parse();
;;
;;start_logging_dns(cli.clone());
;;
;;if let Some(command) = cli.command {
;;;;handle_command(command).unwrap();
;;;;return;
;;}
;;
;;#[cfg(not(feature = "with-gui"))]
;;return start_headless();
;;
;;#[cfg(feature = "with-gui")]
;;if cli.headless {
;;;;start_headless();
;;} else {
;;;;gui::run_ui();
;;}
}
```

;;;;;;;;**5.8**;;**COBOL**

;;;;;;;;;;;;COBOL, while unused in most modern applications, is still a very important language to the world we live in. As a part of this, we must ensure that the code that remains is the most readable. This code is from node.cobol[1].

```
;;;;;;;* Exec Node.js code
;;;;;;;IDENTIFICATION DIVISION.
;;;;;;;PROGRAM-ID. EXEC_NODEJS.
;;;;;;;DATA DIVISION.
```

```
;;;;;;;WORKING-STORAGE SECTION.
;;;;;;;01 COMMAND_TO_RUN PIC X(200) value SPACES.

;;;;;;;LINKAGE SECTION.
;;;;;;;01 NODEJS_CODE PIC A(100) value SPACES.

;;;;;;;PROCEDURE DIVISION USING NODEJS_CODE.
;;;;;;;;;;;STRING 'node -e "' DELIMITED BY SIZE
;;;;;;;;;;;;;;;;;;NODEJS_CODE DELIMITED BY SIZE
;;;;;;;;;;;;;;;;;;'"'   DELIMITED BY SIZE
;;;;;;;;;;;INTO COMMAND_TO_RUN

;;;;;;;;;;;CALL 'SYSTEM' USING COMMAND_TO_RUN
;;;;;;;;;;;END-CALL
;;;;;;;EXIT PROGRAM.
```

;;;;;;;;5.9;;x86 Assembly

Turns out, x86 Assembly is perfect for semicolon indents! It's built right into the compiler. It also has the same advantage as Python has, as it removes any bugs in your code.

```
;org  0x100         ; .com files always start 256 ;bytes into the segment

;mov  dx, msg       ; the address of or message in dx
;mov  ah, 9         ; ah=9 - "print string" sub-function
;int  0x21          ; call dos services

;mov  dl, 0x0d      ; put CR into dl
;mov  ah, 2         ; ah=2 - "print character" sub-function
;int  0x21          ; call dos services

;mov  dl, 0x0a      ; put LF into dl
;mov  ah, 2         ; ah=2 - "print character" sub-function
;int  0x21          ; call dos services

;mov  ah, 0x4c      ; "terminate program" sub-function
;int  0x21          ; call dos services

;msg  db 'Hello again, World!$'   ; $-terminated message
```

;;;;6;;Potential Issues with this Approach

;;;;;;;;The approach outlined in this paper is sound in most situations. However, there are certain situations in which these principles may be unable to be implemented properly, or in where the benefits of this may be degraded.

;;;;;;;;6.1;;The Greek Question Mark

;;;;;;;;;;;;The Greek question mark (; `U+037E GREEK QUESTION MARK`), is visually identical to the semicolon (; `U+003B SEMICOLON`) commonly used in code. This introduces the issue that this C code shows:

```
// Invalid, uses the Greek question mark for indentation
while(i < 10) {
;;printf("%d\n", i);
;;i++
}
// Valid, uses the semicolon for indentation
while(i < 10) {
;;printf("%d\n", i);
;;i++
}
```

;;;;;;;;6.2;;Mid-Line Breaks

;;;;;;;;;;;;Some languages have syntax's that don't permit the usage of semicolon indentation in specific situations, such as when using data structures which use commas to separate items in the structure. Additionally, if a line is broken before the end of the statement, an error may occur when trying to indent this line with semicolons. To prevent these from causing too many issues, the authors of this paper recommend against splitting these lines.
;;;;;;;;;;;;While this paper advocates for readability, this comes at a cost for users who prefer to follow the 80-column rule[2]. An example of some JSON data following this standard of omitting linebreaks is shown below.

```
{"menu": { "id": "file", "value": "File", "popup": {"menuitem": [{"value": "New", "o
```

;;;;7;;Disclaimer

;;;;;;;;This proposal is merely a suggestion, any usage of these principles is at your own risk. Many of the concepts introduced here may cause significant harm if used in a real codebase. You've been warned.

;;References

[1] Ionica Bizau. node.cobol. https://github.com/IonicaBizau/node.cobol, 2021.

[2] Vinicius Brasil. The sacred 80-column rule, Nov 2022.

[3] Tim Holman. Elevator.js. https://github.com/tholman/elevator.js, 2020.

[4] Devyn Keeney. streamline-control. https://github.com/devyntk/streamline-control, 2023.

[5] Stefan Muller. A type-and-affect system for semisignificant whitespace. *Sigbovik 2022*, pages 16–21, Apr 2022.

[6] u/ablaniar. R/programmerhumor - little contribution to the indentation war, Sep 2021.

A perpetual motion machine

Zikuan Wang
zwang@kofo.mpg.de

March 28, 2023

Abstract

In this work, we demonstrate that any sealed container containing some gas is a perpetual motion machine, since the gas molecules are constantly accelerated by Hubble expansion and therefore heated up, and the heat energy can be extracted and converted to work. We give quantitative estimates of the power output of some representative gas containers.

1 Introduction

Perpetual motion machines (PMMs) are machines that can do work forever without energy input[1]. The quest for PMMs has been a popular topic for centuries, since the availability of PMMs would solve the energy crisis problem forever. Although the Sun (and other stars as well, assuming that interstellar travel will become possible) can for all intents and purposes be treated as an effectively inexhaustible energy source in the foreseeable future, this is no longer true if humanity manages to live beyond the lifetimes of the vast majority of stars, or if the energy demand of humankind exceeds the power output of the Sun. Therefore, despite the wide availability of techniques to extract solar energy with little cost and relatively high efficiency, the study of PMMs is still of fundamental importance for the long-term sustainability of humankind.

A frequently quoted argument is that PMMs are theoretically impossible, because they violate the conservation of energy[1]. However, violation of the conservation of energy is not a proof of impossibility, as exemplified by the dark energy[2]. Dark energy is believed to come from nowhere as the universe expands, and is responsible for the accelerated expansion of the universe, therefore in direct conflict with the conservation of energy. Nevertheless, while whether current observational data do support the presence of dark energy is subject to some debate, people generally agree that it is at least not possible to exclude the presence of dark energy with the data we currently have. This immediately suggests that it may be possible to build a PMM taking advantage of the unlimited nature of the dark energy, specifically, using the perpetual expansion of the Universe.

The simplest approach would be to tie two balls to the ends of a very long rope, so that the balls are gradually dragged away by cosmic expansion. The kinetic energies of the balls will eventually be harvested when the rope is straightened, in which case the energy is converted to the tension of the rope, and by attaching the rope to machines we can extract work from it. To allow for indefinite energy output, the rope has to be continuously lengthened; this can be done by using part of the generated energy to create matter and antimatter out of vacuum, and using the matter to lengthen the rope. However, the balls need to be extremely heavy so that the energy output can become greater than the energy spent in matter-antimatter creation. While this may seem to be difficult but not impossible, another issue makes it fundamentally impossible for the method to output energy indefinitely: one of the balls will eventually recede behind the cosmic event horizon of the other ball, at which point the rope will inevitably break.

The present work was inspired by the observation that the balls do not need to separate from each other indefinitely in order for the energy output to last forever. Specifically, the balls still accelerate due to cosmic expansion even if they are allowed to bounce back off a wall. This allows us to put the balls in a closed container, and furthermore replace the balls by gas molecules, so that they will bounce around in the container forever and extract energy from cosmic expansion indefinitely. The extracted energy will be in the form of the kinetic energy of the molecules, which is just thermal energy. A perpetual motion machine can then be built taking advantage of the thermal radiation of the container. In the following section, we prove that the container will indeed generate heat due to cosmic expansion, and provide quantitative estimates of the power output for gas containers of certain sizes.

2 Results and Discussions

Suppose there is a gas molecule M in a rigid container. The molecule moves from a point on the wall of the container, A, with initial velocity v_0, until it hits another point on the container, B. The velocity of M relative to point A increases due to cosmic expansion, as it recedes from point A. The recessional velocity v_r of M at any given time is given by Hubble's law[3]:

$$v_r = HD_{AM} + v_{pec} \tag{1}$$

where $H > 0$ is the Hubble constant, D_{AM} is the distance of M from point A, and v_{pec} is the peculiar velocity of M. Since $v_r = v_0$ when $D_{AM} = 0$, we have

$$v_{pec} = v_0 \tag{2}$$

Thus, the velocity of M relative to point A increases linearly with respect to the distance of M from A.

When the molecule arrives at point B, it bounces off elastically and heads towards point C. From the same analysis as above, we can say that the velocity

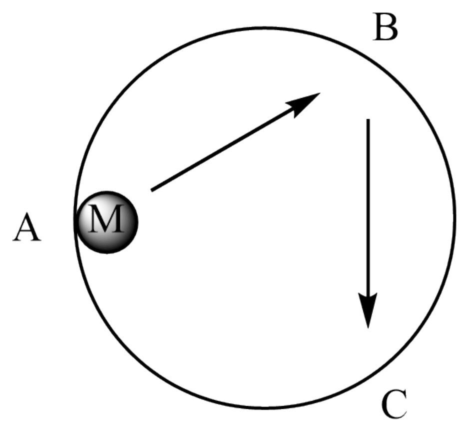


Figure 1: A gas molecule in a rigid container.

of M relative to point B increases linearly as the distance of M from B increases. However, as the container is rigid, point B is stationary relative to point A; therefore, the velocity of M relative to point A also continues to increase even after M has bounced off point B and is not heading away from A anymore. The velocity increment of M is thus the sum of the increment already attained from A *en route* to B (i.e. HD_{AB}), and that attained after the molecule has bounced off point B (HD_{BM}). One can therefore write the general expression of v_r as

$$v_r = Hs_{AM} + v_0 \tag{3}$$

where s_{AM} is the *arc length* of the route travelled by M, starting from A. It is easy to see that the formula is not only applicable when only one bounce has been made, but also to the case of an arbitrary number of bounces. In particular, it is true even until M eventually returns to point A. The important corollary is that, M can depart from point A and return to the same point, yet its velocity increases, and the extra kinetic energy comes solely from cosmic expansion[4]!

Eq. 3 gives the dependence of the velocity of M, i.e. v_r, on the distance travelled by M (s_{AM}). However, a more convenient quantity is the dependence of v_r on time t. Noticing that

$$v_r = \frac{ds_{AM}}{dt} \tag{4}$$

we can solve Eq. 3 as an ordinary differential equation, yielding

$$s_{AM} = \frac{v_0}{H}(e^{Ht} - 1) \tag{5}$$

or

$$v_r = v_0 e^{Ht} \tag{6}$$

Eq. 6 shows that the velocity of M increases exponentially with time. Consequently, the kinetic energy of M also increases exponentially, but with twice the exponent, due to the kinetic energy being proportional to the square of the velocity.

Now, suppose that the container contains not just one gas molecule, but a gas composed of a macroscopic number of molecules. The above derivations are still applicable. Moreover, it is now meaningful to talk about the temperature, T, of the gas. Since the temperature of a gas is proportional to the average kinetic energy of the gas molecules, we have

$$T(t) = T(0)e^{2Ht} \tag{7}$$

Eq. 7 shows that the temperature of a gas in a rigid container increases exponentially, with a time constant of $2H$. Since the current value of H is about 2.27×10^{-18} s^{-1}[5], it follows that the temperature of the gas doubles every 1.53×10^{17} s, or 4.84 billion years. Note that although the Hubble constant changes over time, our qualitative conclusions will remain unchanged as long as the Hubble constant does not approach zero in the long run, which most physicists seem to believe to be the case. Therefore, as long as the container is coated by a sufficiently good thermal insulator, such that its heat loss is on the billion year timescale, the container will become measurably warmer than the outside, and work can be extracted every a few billion years, by temporarily removing a part of the insulating layer and using the resulting heat flow to power a heat pump. This requires the development of extremely good thermal insulators that are well beyond current technology. Alternatively, work can be extracted with a heat pump even without thermal insulators that can work on the billion year timescale; however, the temperature of the container will be only marginally higher than the environment, leading to extremely low Carnot efficiency.

We now estimate the power available for the heat pump. Assuming that the thermal energy $C_V dT(t)$ accumulated over a short period of time dt is converted to work, where C_V is the heat capacity of the gas at constant volume. Then the output power of the container is

$$P = C_V \frac{dT(t)}{dt} = 2C_V HT(0) \tag{8}$$

For example, for a container filled by air ($C_{V,m} = 20.8$ $\text{J}\cdot\text{mol}^{-1}\text{K}^{-1}$)[6] near 298 K, the power output is 4.86×10^{-13} W per kilogram of air, or (at ambient pressure) 5.83×10^{-16} W per liter of air. The numbers may seem negligible, but since there is 5.15×10^{18} kg of air in the world[7], the total power output due to cosmic expansion heating up the atmosphere is 2.50 MW, which is equivalent to 50000 50 W light bulbs. We expect that the figure may not be very accurate since the atmosphere is not contained in a rigid container, but it should have the correct order of magnitude. Also note that cosmic expansion does not seem to be a major contributor to global warming, since the resulting temperature increase is only $\frac{dT(t)}{dt} = 1.35 \times 10^{-15}$ K, tens of magnitudes smaller than the actual global warming rate.

Stars like the Sun are mainly composed of a gas of free protons and electrons, in a 1:1 ratio. Both are single-particle gaseous particles, and thus have $C_{V,m} = 12.5$ J·mol^{-1}K^{-1}[6]. Since the Sun's mass is 1.99×10^{30} kg, and its center temperature is 1.57×10^{7} K[8], the total power output of the Sun due to cosmic expansion is 3.55×10^{24} W. Surprisingly, this is only 2 orders of magnitudes smaller than the total power output of the Sun (3.85×10^{26} W[8]). Although our estimate of the contribution from cosmic expansion is likely an overestimate (due to, for example, using the center temperature of the Sun as the temperature of the whole Sun), we expect that more accurate theoretical models of the Sun may eventually reveal that nuclear fusion does not explain all of the power output of the Sun, and therefore provide an experimental estimate of the cosmic expansion contribution to the power output of the Sun[4].

3 Conclusions

In this work, we show that a gas confined in a rigid container will spontaneously heat up due to cosmic expansion, therefore acting as a PMM, since the heat can be extracted as work by a heat pump. Although the power output of any everyday sized gas container is negligible, it becomes appreciably large when the mass of the gas is comparable to the Earth atmosphere. Even more, we predict that cosmic expansion may have contributed up to 1 % of the total power output of the Sun. Therefore, we propose that the presently mentioned PMM can be built by amassing stellar masses of gases, and collecting the resulting heat radiation. Due to gravity, the gas does not need to be confined within a container anymore, which is especially convenient.

Meanwhile, we point out a few limitations of the current work: we did not take into account the general relativistic corrections due to the gravity of the gas, the fact that there are no infinitely rigid containers, the fact that thermal insulating ability has an upper bound, the finite temperature of the cosmic microwave background, and the breakdown of the non-relativistic velocity-addition formula. These effects are anticipated to be small[4] and will be addressed in future work.

Acknowledgement

The author thanks the reviewers for their helpful comments.

References

[1] https://en.wikipedia.org/wiki/Perpetual_motion

[2] https://en.wikipedia.org/wiki/Dark_energy

[3] https://en.wikipedia.org/wiki/Recessional_velocity

[4] https://youtu.be/dQw4w9WgXcQ

[5] https://en.wikipedia.org/wiki/Hubble%27s_law

[6] https://en.wikipedia.org/wiki/Table_of_specific_heat_capacities

[7] https://en.wikipedia.org/wiki/Atmosphere_of_Earth

[8] https://en.wikipedia.org/wiki/Sun

C

Church, Meet State

Even Lower Order Functions for Returning
or
Why would anyone ever want to use a -6 order function?

With apologies to Chris Okasaki

Saul Field*, Dann Toliver, Erik Derohanian

March 2023

Abstract

Higher order functions are well studied in the CS literature, and provide affordances in functional programming ranging from map and reduce to parser combinators[1]. Lower order functions have received considerably less attention. We present a framework for understanding lower order functions, and show how they may be used to encapsulate many imperative concepts in a functional setting. In particular, the operations of delete, return, and the stack smashing of continuations are put into a functional framing.

1 Introduction

There is a longstanding history, dating back to Turing and Church, of the imperative and functional camps wrestling with each other[1]. Each side desires to subsume the other, to absorb its twin and gain its strength while retaining its own form.

Lower order functions are a winning strategy in this age old struggle. Unlike higher order functions, which increase the possibility space with each successive level, building abstractions on top of abstractions, lowering order functions actually simplify your process each time they are invoked.

This simplifying aspect maps directly to a variety of imperative features that are otherwise complicated to express in the functional paradigm. Delete, return, and even exit, that bugbear of the imperative world, can be expressed naturally and functionally using lower order functions.

*Author declares the existence of a competing interest, having received financial contributions from Big Integer

[1]So heated were these matches that they eventually lead to the complete separation of Church and state.

2 First-order functions

First-order functions do not accept functions as arguments or return functions. They accept or return base values (non-function primitives), and may have other side effects.

These environment semantics for a first-order function in a typical functional programming language clearly reveal the state encapsulation and immutable semantics typically associated with languages derived from the lambda calculus.

$$\textit{def}\,\frac{}{E \vdash \texttt{fun}\ x \to e \Downarrow (\texttt{closure}\ x \to e, E)}$$

$$\textit{apply}\,\frac{E \vdash e_1 \Downarrow (\texttt{closure}\ x_c \to e_c, E_c) \quad E \vdash e_2 \Downarrow v_2 \quad (E_c, x_c \mapsto v_2) \vdash e_c \Downarrow v}{E \vdash e_1\ e_2 \Downarrow v}$$

$$\textit{eval}\,\frac{E \vdash f \Downarrow (\texttt{closure}\ x \to e, E)}{\textit{eval}(E, f, x) = v \dashv E'}$$

Some well-known, straightforward first-order functions are the Ackermann function [2], the Collatz conjecture [3], Fast Inverse Square Root [4], and the Standard Model Lagrangian [5]. These functions are simple and easy to reason about, and will rarely pose any operational difficulty for software developers.

First-order functions are typically considered the base case for higher order functions: a sixth order function is a function that takes a function that takes a function that takes a function that takes a function that takes a function that takes a base primitive as its argument as its argument as its argument as its argument as its argument as its argument.

However, there is no particular reason to bottom out with first-order functions. Let us continue this degression into the underworld, beginning with the end, the null point of infinity, the first and greatest among numbers – zero.

3 Noop as a zero-order function

Zero order functions take no arguments and provide no return value, yet are terribly useful. When a higher order function absolutely positively has to take a useless function as an argument, noop has no substitutes.

Many imperative languages have statements like "debugger", which is effectively a noop from the perspective of the computational process, but is effectful with respect to the external environment. By treating statements like these as zeroth order functions we effectively gain the effects of algebraic effects on their effects, capturing the imperative effectfulness in a snuggley functional embrace.

3.1 Environment semantics for zero order functions

$$\textit{eval}\,\frac{E \vdash f \Downarrow (\texttt{closure}\ x \to e, E)}{\textit{eval}(E, f, x) = v \dashv E}$$

4 Minus-first-order functions

Like zero-order functions, a minus-first-order function neither takes arguments nor returns values. It does, however, have an effect: it removes a variable from the environment.

In particular, it removes the last variable that was bound into the environment, so it is no longer available. When using de Bruijn indices it reduces the value stack by one, like "drop" in Forth.

This allows the modeling of imperative statements like "delete" in a purely function setting. It also creates a path for adding linear logic or other variable ownership constraints to a programming language – any language with first-class minus-first-order functions and zero-class function definitions allows construction of novel memory management models directly in user space.

4.1 Environment semantics for minus-(first,second,third)-order functions

We introduce a new symbol here, '●' (pronounced *VOID*) to denote the degree of destruction of the resulting environment.

$$eval^{-n}\,\frac{E \vdash f^{-n} \Downarrow (\texttt{closure}\ x \to e, E)}{eval(E, f^{-n}, x) = v \dashv E^{\bullet^{n}}}$$

5 Minus-second-order functions

The next step beyond dropping a single value from the environment is to drop an entire frame. This is the functional equivalent of the imperative "return" function, but now reified as a first-class value by minus-second-order functions.

Of course, the return function can already be emulated by using continuations in languages derived from the untyped lambda calculus, but this is a bit like using a wrecking ball to drive a finish nail. In conjunction with minus-third-order functions, the ability to precisely target individual stack frames provides a much more nuanced story around control flow management and theory.

6 Minus-third-order functions

Continuations are an extremely useful control flow construct, but they have also long been a source of confusion and frustrating bugs, and have not seen widespread mainstream adoption outside a few fringe use cases.

A minus-third-order function continues the work of a minus-second-order function, but consumes all the stack frames instead of just one per evaluation. This allows the invocation of a continuation to be modelled as a pair of functions, one which consumes all the current stack frames, and a second which adds frames that were captured earlier.

This division of labour allows more precise control, and enables uses that would otherwise be difficult to achieve: if you have captured a full continuation, but want deploy it without entirely consuming your current stack, this separation allows the fine grained control needed. (The alternative case, where a delimited continuation is captured but we desire to invoke it as a full continuation, is considered next.)

7 Minus-fourth-order functions

Early termination is very common in imperative programming models, where a system may halt and return a result or error message prematurely upon some condition, regardless of how deeply nested its current state is. However, this is quite difficult to simulate in the lambda calculus and similar systems, and is usually appended as a statement the underlying runtime supports with no proper theoretical framing.

A minus-fourth-order function takes minus-third-order functions one step further, by not only eliminating all the stack frames but the actual process as well. This not only provides the necessary construction for incorporating a sudden exit into the theory, it also lifts it into a proper first class function that can be treated as a normal value in the system.

7.1 Environment semantics for minus-fourth-order functions

Note here that there is no longer any environment to speak of after this evaluation, and of course the return value is also *VOID*.

$$eval^{-4}\frac{E \vdash f^{-4} \Downarrow (\texttt{closure}\ x \rightarrow e, E)}{eval(E, f^{-4}, x) = \bullet \dashv \bullet}$$

8 Minus-fifth-order functions

A minus-fifth-order function not only removes the process that invokes it, but also eliminates the possibility of being invoked in the future, leading to a radical simplification of the process and perhaps even the computational substrate – operating system, firmware, and perhaps the broader network.

Various systems in the past have eliminated erroneous code as a safety measure[6], but a true minus-fifth-order function is considerably involved to construct, as it must not only remove all copies of its own code, but also remove the possibility of itself being run in the future.

An practical example of a minus-fifth-order function is a kind of security construction, where a flaw is exploited to remove the flaw. For instance, a network virus that utilizes a zero day exploit in order to patch every machine it compromises, destroying itself in the process.

8.1 Environment semantics for minus-fifth-order functions

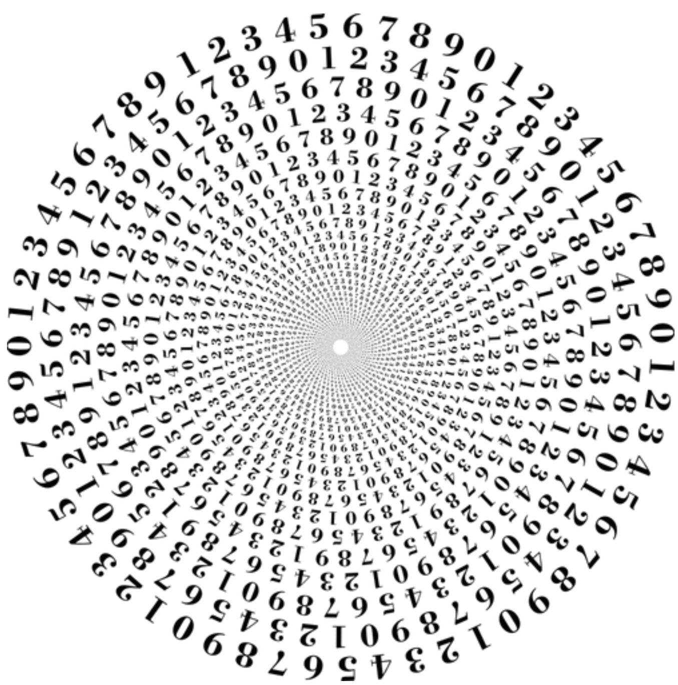
The semantics here are identical to the minus fourth order semantics, with a notable difference. Not only have the value and environment succumbed to the *VOID*, but the *eval* construct of our metalanguage has as well.

$$\textit{eval}^{-5}\ \frac{E \vdash f^{-5} \Downarrow (\texttt{closure}\ x \to e, E)}{\textit{eval}(E, f^{-5}, x) = \bullet \dashv \bullet[\textit{eval} \leadsto \bullet]}$$

9 A minus-sixth-order function

Still lower? There is a minus-sixth-order function – the function which prevents anyone who discovers it from talking about it, or even thinking about it, by eradicating the neural pathways capable of evaluating its construction during its evaluation. Or it certainly seems that there should be such a function. The authors, who have designed effective biotemporal computation signalling systems and who believe they have been certified in the field of antimemetics, appear to have spent the last twelve years designing it with the intent to include it in this paper, but it seems to have escaped them.

9.1 Environment semantics for minus-sixth-order functions



References

[1] C. OKASAKI, "Even higher-order functions for parsing or why would anyone ever want to use a sixth-order function?," *Journal of Functional Programming*, vol. 8, no. 2, p. 195–199, 1998.

[2] W. Ackermann, "Zum hilbertschen aufbau der reellen zahlen," *Mathematische Annalen*, vol. 99, pp. 118–133, Dec. 1928.

[3] Wikipedia, "Collatz conjecture — Wikipedia, the free encyclopedia." `https://en.wikipedia.org/wiki/Collatz_conjecture`, 2023. [Online; accessed 21-March-2023].

[4] K. N. William Kahan, "e_sqrt.c." `http://www.netlib.org/fdlibm/e_sqrt.c`, May 1986.

[5] G. Lisi, "The standard model lagrangian." `http://deferentialgeometry.org/papers/Standard%20Model.pdf`, 2007.

[6] B. Nystrom, "Vigil." `https://github.com/munificent/vigil`, 2013.

PizzaLang: A Language for Hungry Developers

Ronik Bhaskar
The University of Chicago

Dawson Ren
Northwestern University

Abstract

Modern programming languages lack a clear focus on feeding their hungry programmers. As a first step, we introduce the *PizzaLang* programming language. With a straightforward type system, easy-to-read concrete syntax, and extensive features, *PizzaLang* outperforms state-of-the-art pizza programming languages. We rigorously prove powerful properties about *PizzaLang*, including strong normalization. Our implementation of a *PizzaLang* interpreter provides tangible results, demonstrating the effectiveness of this language.

1 Introduction

It's 11pm. You've been programming all day, and you forgot to eat dinner. With the project deadline approaching, it's not like you had time. You can't keep working on an empty stomach, but you can't step away from the keyboard either.

Pull yourself together. You're a programmer. You're a professional problem solver. You can get some food and keep writing code at the same time. All you need are the right tools.

Currently, these tools are APIs wrapped in APIs wrapped in dynamically-typed, object-oriented languages that don't care about your pizza. While they serve the necessary purpose, the sheer level of overhead makes actual use much more difficult than anticipated. Developers are stuck reading through pages of documentation, only to realize the pizza shop has closed, and their dinner plans will never be realized.

PizzaLang aims to provide better tools to hungry programmers. The language has one main purpose: constructing pizzas. Using clean structure, straightforward typing, and simple grammar, *PizzaLang* allows users to construct terms suitable for any pizza-related project, including ordering pizza.

2 Term Grammar

The term grammar of *PizzaLang* is as follows:

t ::= bp	*base pizza*
| left	*left half of pizza*
| right	*right half of pizza*
| all	*whole pizza*
| mushrooms	
| onions	
| green peppers	
| ...	*pizza toppings*
| add t t t	*add topping*
| remove t t	*remove topping*

This allows for the construction of the following values. All values are specified by v, following the style of Types and Programming Languages [1]. Pizza values are denoted by pv, topping values are denoted by tv, and location values are denoted by lv.

v ::= pv	*pizza values*
| tv	*topping values*
| lv	*location values*

tv ::= mushrooms | onions | green peppers | ...

lv ::= left
  | right
  | all

pv ::= bp
  | add pv tv lv

Finally, this language has three types: Pizza, Topping, and Location.

$$\mathcal{T} ::= \text{ Pizza } | \text{ Topping } | \text{ Location}$$

3 Evaluation Rules

The add and remove terms in *PizzaLang* function similarly to successor and predecessor respectively in Peano Arithmetic. The base pizza behaves like zero. The predecessor of zero is still zero, so removing a topping from a plain pizza results in a plain pizza. Wrapping add terms around a base pizza creates a larger pizza.

Unlike Peano Arithmetic, not all adds and removes are created equal. A remove around an add will only cause the two to cancel if they describe the same topping. Logically, removing mushrooms shouldn't also cause the green peppers to be removed from your pizza. Since adds can be nested, the remove term can descend through the adds, searching for the correct topping to remove.

Computation Rules

$$\frac{v_2 = v_4}{\text{remove (add } v_1\ v_2\ v_3)\ v_4 \to v_1}$$

$$\frac{v_2 \neq v_4}{\text{remove (add } v_1\ v_2\ v_3)\ v_4 \to \text{add (remove } v_1\ v_4)\ v_2\ v_3}$$

$$\frac{}{\text{remove bp } t_2 \to \text{bp}}$$

Congruence Rules

$$\frac{t_1 \to t_1'}{\text{add } t_1\ t_2\ t_3 \to \text{add } t_1'\ t_2\ t_3}$$

$$\frac{t_2 \to t_2'}{\text{add } v_1\ t_2\ t_3 \to \text{add } v_1\ t_2'\ t_3}$$

$$\frac{t_3 \to t_3'}{\text{add } v_1\ v_2\ t_3 \to \text{add } v_1\ v_2\ t_3'}$$

$$\frac{t_1 \to t_1'}{\text{remove } t_1\ t_2 \to \text{remove } t_1'\ t_2}$$

$$\frac{t_2 \to t_2'}{\text{remove } v_1\ t_2 \to \text{remove } v_1\ t_2'}$$

4 Type Rules

Since *PizzaLang* does not utilize variables, *PizzaLang* does not require a context Γ to store the types of variables. Instead, toppings are specified as fixed set of terms in the grammar, allowing independent developers to determine whether or not "pineapple" is valid syntax. Also, depending on the implementation of the interpreter/compiler, "mushrooms" is a well-typed program.

$$\frac{}{\text{bp : Pizza}}$$

$$\frac{}{\text{left : Location}}$$

$$\frac{}{\text{right : Location}}$$

$$\frac{}{\text{all : Location}}$$

$$\frac{}{\textit{topping name} \text{ : Topping}}$$

$$\frac{t_1 \text{ : Pizza} \qquad t_2 \text{ : Topping} \qquad t_3 \text{ : Location}}{\text{add } t_1\ t_2\ t_3 \text{ : Pizza}}$$

$$\frac{t_1 \text{ : Pizza} \qquad t_2 \text{ : Topping}}{\text{remove } t_1\ t_2 \text{ : Pizza}}$$

Given our type system, you may have noticed that the second, third, and fifth congruence rules are redundant, since terms of type Location or Topping are already values. These congruence rules future-proof the language for new terms of type Location or Topping that step to a value in one or more steps.

5 Strong Normalization

We claim that our language has strong normalization [1]. If a term in PizzaLang is well-typed, then it steps to a value in zero or more steps. The proof is by induction over the depth of the term. See Appendix A for the full, mechanized proof.

For far too long, Big Pizza has upheld too many barriers to getting your pizza. Pop-ups, redirects, ads, and confusing choices continue to plague the modern pizza consumer. As independent researchers, we hope to introduce robust systems for ordering pizza. This begins with the strong guarantee that your pizza order can produce a pizza.

Pizza-ordering languages that lack strong normalization risk infinite pizza loops, introducing topological paradoxes about how to apply sauce to a Möbius Pizza.

6 Concrete Syntax

The initial interpreter, written in Typed Racket, replicates a spoken-English pizza order with its syntax. It only specifies three different toppings, and they must be spelled as described. Furthermore, all programs must begin with "i'd like uh..." and end with "don't forget to bake it". We feel the first syntax requirement adds to the programming experience, mimicking common speech patterns for ordering food. The second requirement is entirely to annoy programmers who forget to add it.

t ::= "i'd like uh..."	
t	*main body*
"don't forget to bake it"	
| "pizza"	*base pizza*
| t "with" t "on the" t	*add topping*
| t "and" t "on the" t	*add topping*
| t "actually hold the" t	*remove topping*
| t "and" t	*remove topping*
| "left half"	*location*
| "right half"	*location*
| "whole thing"	*location*
| "mushrooms"	
| "onions"	
| "green peppers"	

Here is an example program:

```
i'd like uh...
pizza with mushrooms on the left half
and green peppers on the right half
and onions on the whole thing
actually
hold the onions and mushrooms
don't forget to bake it
```

The program evaluates to the following term:

```
add bp "green peppers" right
```

7 Further Applications

The most important values in the program are those of type Pizza. They can be read as a series of instructions, describing which toppings to add to a base pizza, in what order, and where on the pizza. To allow for further computations, an interpreter could use the sequence of toppings as computational instructions. For example, you may read the location of the topping as a left/right/stay move on a Turing tape, and the topping itself may tell you how to manipulate that square. Alternatively, you could feed the instructions to a program that draws pizzas.

Beyond computation, *PizzaLang* creates a context-free grammar for ordering pizza. Pizza delivery APIs have existed for years. Individual users can define what a base pizza means to them, and most pizza-delivery restaurants take orders as sequences of toppings to put on a pizza. The instructions may need to be adapted to fit the nuance of each pizza restaurant, but the logic is still clear.

Programmers of the world, open your editors. Let your stomachs be your guide. Write some code, and order some pizza.

References

[1] Benjamin C. Pierce. *Types and Programming Languages*. The MIT Press, Cambridge, Mass., 2002.

Special thanks to Adam Shaw and Vincent St-Amour.

We acknowledge that this paper does not necessarily reflect the beliefs and ideas of our respective universities. We also acknowledge that the the rivalry between the universities is silly because one is clearly better than the other.

A Appendix

We claim PizzaLang has strong normalization. The following mechanized proof is written in Agda.

```
module pizza-lang-strong-normalization-proof where

open import PizzaLang.Theorems using (strong-norm-proof; WellTyped; steps-to-val)
open import PizzaLang.Theorems using (pizza-lang)

pl-strong-norm : Set
pl-strong-norm = ∀(x : WellTyped) → (steps-to-val x)

pl-strong-norm-proof : ∀(x : WellTyped) → (steps-to-val x)
pl-strong-norm-proof = (strong-norm-proof pizza-lang)
```

Quadratic Logic

Daniel Ng

1 Abstract

The well-known linear logic system represents a world where, whenever a proposition is used, it is also consumed. However, this system does not generalize well to the real world, where economies of scale dictate that producing more copies of something at one time will take less resources than producing each copy individually. To model this, we remove the strict one-to-one correspondence between propositions on the left-hand side of a sequent and the propositions they produce on the right-hand side when consumed by rule application. By replacing the linear function with a quadratic function, we are able to model the real world more effectively under this system, while still preserving linearity in some cases.

2 Introduction

The system of linear logic stipulates that when an left rule is used to create a new proposition, any propositions used in the creation process are removed. This means that throughout the course of a linear proof, things are consumed and reused in a one-for-one correspondence. Having an A and using it to come to the conclusion of B is sufficient to create the proposition $A \multimap B$, consuming the A and the B in the process. Likewise, having an $A \multimap B$ and an A is sufficient to remove both and create a B, but the A and the $A \multimap B$ cannot be used later on. The meaning of linear logic becomes clear from these examples – put in one copy of the necessary ingredients to get one copy of a proposition.

It follows that multiple copies – for example, n copies – of a proposition can be created by putting in n copies of each of the necessary premises. However, this does not match up with reality. It is often more equivalent to create things in large batches, or otherwise utilize fewer resources to create more things than could be individually made using the same amount of resources.[1] This reflects how things work in the real world.[2] Therefore, we seek to model this with a new set of rules, which better reward proof authors for efficiently using left rules by rewarding them with additional copies of the proposition based on the number of copies of the premises used.

[1] This is where the common grocery store advertisement of "the more you spend, the more you save" becomes relevant.

[2] The author recognizes that computer science theory often does not seek to represent the real world, as evidenced by the amount of time spent on [insert problem you hate here].

The next question to consider is how much of a reward one should get from a multiple-introduction of propositions. Increasing the number of propositions created by a linear factor proves problematic, as a device comes about to generate infinitely many copies of a proposition as long as the cut and identity rules hold. This tends to over-reward introducing small numbers of propositions, while not truly appreciating the genius that comes with consuming 10 copies of A and B each to introduce some number of copies of $A \otimes B$.[3] Therefore, we need to consider what function $f(n)$ should be used to determine how many copies of a proposition should be introduced when n copies of each of the premises are used. Functions of degree 3 and above, as well as exponential functions, grow to concerningly large numbers even on small n.[4] Based on some preliminary experiments, it seems that functions in the neighborhood of $\Theta(n^2)$ form the ideal candidates. For now, we proceed with the most basic function in this class – $f(n) = n^2$.

3 Rules

The following set of rules, shown in Figure 1 on the next page, are adapted from Frank Pfenning's rules for linear logic. [1] We introduce the notation A true $[n]$ to indicate having $n > 0$ copies of proposition A, but the rules otherwise remain quite similar to their original formulation. Additionally, since we are trying to focus on quadratic functions in this work, we remove the exponential entirely since $2^n \notin \Theta(n^2)$.

The notable rules in this set are the id rule, which enables us to square the quantity of an atomic proposition should it be necessary. It also moves it to the right-hand side. The split rule is also new, and allows us to split up a group of n propositions into two groups of size x and $n - x$, where $0 < x < n$. Combine inverts a split, combining two groups of the same proposition which have sizes m and n into a single group with size $m + n$ on either side of the sequent. For simplicity, we only permit positive integer numbers of propositions to be taken at a time. The remainder of the rules operate more or less as expected, with left rules using n copies of each premise and creating n^2 copies of whatever is being introduced.

4 Identity and Cut

As it turns out, the linear identity and cut theorems still hold here. To prove these, we begin by proving a pair of very important lemmas to relate this back to the original system of linear logic.

Lemma (Quadratic Evaluation)
If we let $f(n) = n^2$, then $f(1) = 1$.
Proof: $f(1) = 1^2 = 1 \cdot 1 = 1$. The remainder of the proof is left as an exercise to the reader, or to any friends of the reader who happen to have an interest in foundational mathematics.

[3] We needed some $A \otimes B$s to give out to 2023 SIGBOVIK attendees as souvenirs, but did not have enough for everyone. :(
[4] Since this is a computer science paper, we opt to not acknowledge the existence of numbers above 2,147,483,647.

Rules for Quadratic Logic

$$\frac{(P \text{ atomic})}{P \text{ true } [n] \Vdash P \text{ true } \left[n^2\right]} \; id \qquad \frac{\Gamma, A \text{ true } [x], A \text{ true } [n-x] \Vdash B \text{ true } [m]}{\Gamma, A \text{ true } [n] \Vdash B \text{ true } [m]} \; split$$

$$\frac{\Gamma_1 \Vdash A \text{ true } [m] \qquad \Gamma_2 \Vdash A \text{ true } [n]}{\Gamma_1, \Gamma_2 \Vdash A \text{ true } [m+n]} \; combine\ R \qquad \frac{\Gamma, A \text{ true } [m+n] \Vdash B \text{ true } [p]}{\Gamma, A \text{ true } [m], A \text{ true } [n] \Vdash B \text{ true } [p]} \; combine\ L$$

$$\frac{\Gamma_1 \Vdash A \text{ true } [n] \qquad \Gamma_2 \Vdash B \text{ true } [n]}{\Gamma_1, \Gamma_2 \Vdash A \otimes B \text{ true } \left[n^2\right]} \; \otimes R \qquad \frac{\Gamma, A \text{ true } [n], B \text{ true } [n] \Vdash C \text{ true } [m]}{\Gamma, A \otimes B \text{ true } [n] \Vdash C \text{ true } [m]} \; \otimes L$$

$$\frac{\Gamma \Vdash A \text{ true } [n] \qquad \Gamma \Vdash B \text{ true } [n]}{\Gamma \Vdash A \& B \text{ true } \left[n^2\right]} \; \& R \qquad \frac{\Gamma, A \text{ true } [n] \Vdash C \text{ true } [m]}{\Gamma, A \& B \text{ true } [n] \Vdash C \text{ true } [m]} \; \& L1 \qquad \frac{\Gamma, B \text{ true } [n] \Vdash C \text{ true } [m]}{\Gamma, A \& B \text{ true } [n] \Vdash C \text{ true } [m]} \; \& L2$$

$$\frac{\Gamma \Vdash A \text{ true } [n]}{\Gamma \Vdash A \oplus B \text{ true } \left[n^2\right]} \; \oplus R1 \qquad \frac{\Gamma \Vdash B \text{ true } [n]}{\Gamma \Vdash A \oplus B \text{ true } \left[n^2\right]} \; \oplus R2$$

$$\frac{\Gamma, A \text{ true } [n] \Vdash C \text{ true } [m] \qquad \Gamma, B \text{ true } [n] \Vdash C \text{ true } [m]}{\Gamma, A \oplus B \text{ true } [n] \Vdash C \text{ true } [m]} \; \oplus L$$

$$\frac{\Gamma, A \text{ true } [n] \Vdash B \text{ true } [n]}{\Gamma \Vdash A \multimap B \text{ true } \left[n^2\right]} \; \multimap R \qquad \frac{\Gamma_1 \Vdash A \text{ true } [n] \qquad \Gamma_2, B \text{ true } [n] \Vdash C \text{ true } [m]}{\Gamma_1, \Gamma_2, A \multimap B \text{ true } [n] \Vdash C \text{ true } [m]} \; \multimap L$$

$$\frac{}{\cdot \Vdash 1 \text{ true } [n]} \; 1R \qquad \frac{\Gamma \Vdash C \text{ true } [n]}{\Gamma, 1 \Vdash C \text{ true } [n]} \; 1L$$

$$\frac{}{\Gamma \Vdash \top \text{ true } [n]} \; \top R \qquad \frac{}{\Gamma, 0 \text{ true } [n] \Vdash C \text{ true } [m]} \; 0L$$

Figure 1: The inference rules for quadratic logic.

Lemma (Linear-Quadratic Correspondence[a])

Any valid proof under linear logic without the exponential is also a valid proof under quadratic logic when $f(n) = n^2$ if we treat a proposition judged as true in linear logic to have the judgement true [1] in quadratic logic.

Proof: Observe that any linear left rule is the same as a quadratic left rule where the parameters for all the true [] judgements are set to 1. Any linear right rule is the same as a quadratic right rule where the parameters of both the inputs and the outputs are 1. By the Quadratic Evaluation Lemma, it holds that any quadratic logic rule is the same as a linear logic rule in this case, because $n = 1$ means $f(n) = n^2 = 1$. This is good, because it allows us to use quadratic rules as linear rules in this case by replacing any instance of the true judgement with the true [1] judgement. Since all left rules now consume one copy of each premise and produce one copy of each result, simply substituting any linear rules with the corresponding quadratic rules transforms the linear proof into a quadratic proof.

[a] The authors do not think this is much of a correspondence, but every PL paper seems to be required to feature at least one correspondence.

From this second lemma, some useful results from linear logic can be translated to quadratic logic.

Corollary (Linear Identity)

It is true for any proposition A that A true $[1] \Vdash A$ true $[1]$.

Proof: Once again, we take the inductive proof of identity in linear logic and translate it to quadratic logic via the Linear-Quadratic Correspondence. Since the corollary that we are proving is just a linear logic sequent rephrased as a quadratic logic sequent, we are done.

Corollary (Linear Cut)

If $\Gamma_1 \Vdash A$ true $[1]$ and Γ_2, A true $[1] \Vdash B$ true $[1]$, then $\Gamma_1, \Gamma_2 \Vdash B$ true $[1]$.

Proof: This is the same as the linear cut theorem, only translated to use the quadratic syntax for "one copy of a proposition being true". The proof is the same as in linear logic, after conversion via the Linear-Quadratic Correspondence lemma.

4.1 Linear Identity Revisited

There are two versions of the identity theorem that can hold. One of them is the standard linear identity, applied to propositions in any quantity.

Theorem (Linear Identity 2: Electric Boogaloo)

A true $[n] \Vdash A$ true $[n]$ for any proposition A and any positive integer n.

To prove this, we proceed by a nested induction. Firstly, as with normal linear identity, we take as inductive hypothesis that for any smaller propositions, this theorem holds.[5] We then proceed by induction on n.

The base case comes when $n = 1$. In this case, we have no choice but to actually break down the proposition and prove the identity rule, using the proof from the earlier Linear Identity proof.

The inductive step is slightly more interesting. Let $\mathcal{D}$ represent the derivation of the $n = 1$ case from linear identity. Then the following represents a proof that A true $[n] \Vdash A$ true $[n]$:

$$A[1] \Vdash A[1] \qquad \text{(by derivation } \mathcal{D}\text{)}$$

$$A[n-1] \Vdash A[n-1] \qquad \text{(inductive hypothesis)}$$

$$A[n-1], A[1] \Vdash A[n] \qquad \text{(combine } R \text{ rule)}$$

$$A[n] \Vdash A[n] \qquad \text{(split rule)}$$

4.2 Quadratic Identity

Linear identity is a nice theorem to have around, however, for the purposes of proving things like cut later, it proves to be too weak. In the spirit of asking "can we do better?", the proof of identity once again can be reconsidered to strengthen the theorem more.

Theorem (Quadratic Identity)

A true $[n] \Vdash A$ true $\left[n^2\right]$ for any proposition A and any positive integer n.

Even only using the old linear identity theorem, we have sufficient machinery to prove this theorem! The atomic case comes about from the *id* rule, which directly proves the theorem for atomic propositions. For propositions built up with one or more connectives, we must perform more proof using the previous linear identity theorem. One example of such an proof is provided. The remainder are similar.

Example (⊕ Case)

We aim to show $A \multimap B$ true $[n] \Vdash A \multimap B$ true $\left[n^2\right]$.

Proof:

$$A \text{ true } [n] \Vdash A \text{ true } [n] \qquad \text{(by linear identity)}$$

$$B \text{ true } [n] \Vdash B \text{ true } [n] \qquad \text{(by linear identity)}$$

$$A \multimap B \text{ true } [n], A \text{ true } [n] \Vdash B \text{ true } [n] \qquad (\multimap L \text{ rule})$$

$$A \vdash B \text{ true } [n] \Vdash A \multimap B \text{ true } \left[n^2\right] \text{ true } \left[n^2\right] \qquad (\multimap R \text{ rule})$$

Observe that these proofs of quadratic identity never invoke the quadratic identity rule as an inductive hypothesis. Doing so would result in having many extra copies of propositions from multiple layers of squaring, which could lead to undesirable outcomes since there is nowhere for the extra copies of propositions to go.[6]

[5] This is not directly used, but is necessary to allow the $n = 1$ case to invoke linear identity.

[6] These extra propositions could eventually reach critical mass, creating undesirable side effects such as forming a black hole. Black holes are not specified in the rules and would be likely considered undefined behaviour.

4.3 Quadratic Cut Theorem

Now that quadratic identity has been proven, we are now ready to approach the problem of cut. As it turns out, the cut theorem not only holds when $n = 1$, it also holds in the general case!

Theorem (Cut)
If $\mathcal{D} = \Gamma_1 \Vdash A \text{ true } [n]$ and $\mathcal{E} = \Gamma_2, A \text{ true } [n] \Vdash C \text{ true } [m]$ then $\Gamma_1, \Gamma_2 \Vdash C \text{ true } [m]$.

The proof of this is via induction. For a particular case, the inductive hypothesis is that the cut theorem holds on any proposition A' and two derivations $\mathcal{D}'$ and $\mathcal{E}'$, where one of the following hold: A' is smaller than A, or $A' = A$ and $\mathcal{D}'$ is smaller than $\mathcal{D}$, or $A' = A$, $\mathcal{D}' = \mathcal{D}$, and $\mathcal{E}'$ is smaller than $\mathcal{E}$.

This is a fairly standard setup for a cut proof. [2] As such, we present two cases for the $\otimes$ connective as examples of how this proof proceeds. The remainder of the proof is similar to these cases.[7]

Example (Principal Case)

$$\mathcal{D} = \cfrac{\overset{\mathcal{D}_1}{\Gamma_1 \Vdash A \text{ true } [n]} \quad \overset{\mathcal{D}_2}{\Gamma_2 \Vdash B \text{ true } [n]}}{\Gamma_1, \Gamma_2 \Vdash A \otimes B \text{ true } \left[n^2\right]} \otimes R \qquad \mathcal{E} = \cfrac{\overset{\mathcal{E}_1}{\Gamma_3, A \text{ true } \left[n^2\right], B \text{ true } \left[n^2\right] \Vdash C \text{ true } [m]}}{\Gamma_3, A \otimes B \text{ true } \left[n^2\right] \Vdash C \text{ true } [m]} \otimes L$$

We then show that $\Gamma_1, \Gamma_2, \Gamma_3 \Vdash C \text{ true } [m]$.

$$\begin{aligned}
\Gamma_2 &\Vdash B \text{ true } [n] && \text{(1 - by } \mathcal{D}_2\text{)} \\
B \text{ true } [n] &\Vdash B \text{ true } \left[n^2\right] && \text{(2 - by identity)} \\
\Gamma_2 &\Vdash B \text{ true } \left[n^2\right] && \text{(3 - by IH on } (B, (1), (2)) \text{ since } B < A \otimes B) \\
\Gamma_2, \Gamma_3, A \text{ true } \left[n^2\right] &\Vdash C \text{ true } [m] && \text{(4 - by IH on } (B, (3), \mathcal{E}_1) \text{ since } B < A \otimes B) \\
\Gamma_1 &\Vdash A \text{ true } [n] && \text{(5 - by } \mathcal{D}_1\text{)} \\
A \text{ true } [n] &\Vdash A \text{ true } \left[n^2\right] && \text{(6 - by identity)} \\
\Gamma_1 &\Vdash A \text{ true } \left[n^2\right] && \text{(7 - by IH on } (A, (5), (6)) \text{ since } A < A \otimes B) \\
\Gamma_1, \Gamma_2, \Gamma_3 &\Vdash C \text{ true } [m] && \text{(by IH on } (A, (7), (4)) \text{ since } A < A \otimes B)
\end{aligned}$$

Example (Side Case)

$$\mathcal{D} = \cfrac{\overset{\mathcal{D}_1}{\Gamma_1, A \text{ true } [n], B \text{ true } [n] \Vdash D \text{ true } [k]}}{\Gamma_1, A \otimes B \text{ true } \left[n^2\right] \Vdash D \text{ true } [k]} \otimes L \qquad \overset{\mathcal{E}}{\Gamma_2, D \text{ true } [k] \Vdash C \text{ true } [m]}$$

We then show that $\Gamma_1, \Gamma_2, A \otimes B \text{ true } \left[n^2\right] \Vdash C \text{ true } [m]$.

$$\begin{aligned}
\Gamma_1, \Gamma_2, A \text{ true } [n], B \text{ true } [n] &\Vdash C \text{ true } [m] && \text{(by IH on } (A \otimes B, \mathcal{D}_1, \mathcal{E}) \text{ since } \mathcal{D}_1 < \mathcal{D}) \\
\Gamma_1, \Gamma_2, A \otimes B \text{ true } \left[n^2\right] &\Vdash C \text{ true } [m] && \text{(by } \otimes L \text{ rule)}
\end{aligned}$$

[7] This would normally be left as an exercise to the reader, but having to prove this many cases would likely make any readers stop reading.

5 Applications

The system created here might, at first glance, appear to be quite limited in its functionality. Starting with two copies of a proposition can be a massive problem when trying to create three copies – the options are to leave them as two or move up to four. To combat this problem, a new judgement can be introduced:

> **Definition**
> The judgement A so true $[n]$ is defined as A true $[n']$, where $n' \geq n > 0$.

Intuitively, we can think of A so true $[n]$ as being similar to A true $[n] \otimes \top$ true $[1]$, in that we keep the n copies of A that we need while using the remainder to prove the $\top$ part of the product. By proving conclusions as so true instead of true, creative logicians will be able to manipulate the rules of quadratic logic to give themselves a more flexible version of linear logic.

Having come from linear logic, it is no surprise that there are many similarities between quadratic and linear logic. Having one copy of a proposition makes that proposition behave more or less in a linear way – there is no way to use the identity rule to get more copies of such a proposition. What is slightly more surprising is that propositions with a count greater than 1 can have as many copies of necessary being generated through repeated use of the identity rule, as long as there is a so true judgement somewhere to take care of extra copies that may be created.

This means that propositions that are defined multiple times actually behave in a style more consistent with a standard sequent calculus, or the exponential in linear logic. This insight indicates that quadratic logic can be used to blend linear and standard styles. In fact, the way quadratic logic allows users to limit how many times something is used based on the number of calls to the identity rule means that there is a tighter degree of control afforded to the proof writer.[8]

6 Extensions to Further Quadratics

This whole paper has been, so far, devoted to using the function n^2 to determine the number of proposition copies introduced on the right side from right rules. However, what happens when we have a different quadratic as the function? There are several interesting things that could occur.

Firstly, the function could evaulate to a negative number. In this case, we go into *propositional debt* at this point, where to recover the remainder of the proof, the negative copies of a proposition must be offset by positive copies from elsewhere in the proof tree.[9] This debt cannot get swept up by a so true judgement of some form due to the definition of the judgement, meaning that the prover is actually responsible for any debt that is incurred. This debt will actually prevent the proof from being complete as well, because it will interfere

[8] Some might go as far as to say this makes the user "resource aware".
[9] This brings a new kind of meaning to "negative type".

with the correct conclusion being reached much like an extraneous proposition in linear logic would interfere with it.

A couple further points of pain come up when using general quadratic functions of the form $f(n) = an^2+bn+c$ come about, even if we force a, b, c, n to all be integers and $n > 0$. For one, having $f(0) > 0$ allows us to pull propositions out of thin air, because we can use the id rule to generate them using this quadratic − since if a proposition does not appear on the left-hand side of the sequent, we assume there are 0 copies of it there. We therefore want $f(0) = 0$, to avoid going into debt from unused propositions, of which there are typically many in a standard proof. Aside from that, $f(1) = 1$ tends to be a desirable property, so that we preserve linearity for linear propositions as described earlier. Fortunately, parabolas need three points to be fully defined, leaving the programmer some degree of flexibility.

7 Further Research

The most pressing issue for quadratic logic, at the moment, is that the $\Vdash$ symbol being used is entirely made up of straight lines. Since quadratic functions are generally not linear, further research will likely first go toward developing a replacement for $\Vdash$ that involves more parabolas and fewer lines. This will allow the distinction between quadratic and linear logic to become more clear, as the notation for both systems currently looks very similar.[10]

Aside from the obvious notational issue, another extension the authors would like to consider comes from adding different quadratic functions into each introduction rule, causing different numbers of propositions to generated depending on the order in which some rules are used. Quadratics with imaginary roots or only one real root could also pose an interesting problem. Extending the number of copies of a proposition to the complex numbers so that there are two axes for which the count of a proposition is defined on − the real axis and the imaginary axis − could also be an interesting direction to pursue.

8 References

[1] Pfenning, F. 2017. Lecture Notes on Cut Elimination, 15-317: Constructive Logic. Carnegie Mellon University, USA. http://www.cs.cmu.edu/ crary/317-f22/lectures/10-cutelim.pdf

[2] Pfenning, F. 2023. Seeing if people will believe logical claims if Frank Pfenning's name is attached. Carnegie Mellon University, USA.

[10] The rules also look very similar in some cases.

Ringfuck: It's a wheel great time!

Ryan Hornby and Matthew Safar

No Institute Given

Abstract. Left as an exercise to the reader.

1 Introduction

The programming language Brainfuck has been helping programmers improve their skills since its creation in 1993 by Urban Müller. Its great design of simplicity and power have left it unchanged since its inception, unlike many modern "high level" languages. Recently, Sigbovik contributor Ho Thanh has expanded the possibilities of Brainfuck with the improved functionality version Brainfuck++, which includes object oriented programming (it doesn't actually), networking, and many quality of life features (Thanh, 2022). Unlike these attempts to modernize Brainfuck, however, we propose a radically new innovation to the Brainfuck language that addresses some gaping (circular) holes in the language.

Brainfuck programs operate on a tape of memory, similar to a Turing machine. However, as anyone who has gotten super high and watched *Arrival* will tell you, the true form of everything is a circle (Garland, 2015). Whether it be physical circles, such as spherical cows (Stellman, 1973) or temporal cycles such as the boom and bust cycle of our great capitalist economy, the most natural shape is the one that doesn't cut any corners: a circle. To this end, we have developed a new offshoot of Brainfuck, which we call Ringfuck which allows you to practice your circular reasoning.

2 The Ringfuck Language

2.1 Brainfuck Syntax

Fundamentally, this language is based on the syntax of Brainfuck. As previously described, the Brainfuck language is comprised of only eight commands, listed in Table 1. Any Brainfuck program is simply a list of these eight commands executed sequentially, which together move a data pointer that modifies and reads from a tape of memory, segmented into bytes.

It's this simplicity that has prompted prominent experts in the field of computer science to give Brainfuck the venerable title of a "Turing tarpit" (Perlis, 1982).

`+`	Increment the byte at the data pointer.
`-`	Decrement the byte at the data pointer.
`>`	Increment the data pointer.
`<`	Decrement the data pointer.
`[`	If the value at the data pointer is zero, jump to the command after the corresponding `]` command.
`]`	Jump back to the corresponding `[` command.
`.`	Output the byte at the data pointer.
`,`	Access one byte of input and store it at the data pointer.

Table 1. The eight commands in Brainfuck.

2.2 Rings

Instead of using tape, we propose that memory should be folded onto itself, and connected into rings, similar to how you would fold tape to stick a polaroid of that one time you went to Europe in college above your bed.

Header Realistically, any Brainfuck program is actually just at RingLang program on a single infinite ring. However, this one ring to rule them all approach is boring, and, in practice, impossible, so we include a header to specify ring sizes. Note that Ringfuck supports (and encourages! `<3`) having multiple rings.

This is done by writing the desired size of each ring separated by white space at the top of the file before any instance of the eight commands.

Brainfuck Commands with Rings For a single ring, the Brainfuck commands are the same, with the exception of `>` and `<`, which now loop around the ring.

Multiple rings, however, necessitates multiple data pointers. Brainfuck commands affect all data pointers, meaning that each ring will execute the whole program individually. The input and output commands (`,` and `.`) read and write according to the order of initialization. Notably, the input command requires as many inputs as there are rings, and will wait until each of the selected bytes are filled.

A Simple Example We'll start by looking at the canonical example of a program in a new language, the "Hello World" program. The below code will print "HELLO WORLD! \n" when executed. This is done by loading index 0 cells of each ring with the ascii codes for the appropriate character and then printing them out. Note that the below program is slightly optimized and might be slightly unintuitive, if this is the case please see the Appendix for an unoptimized version of this program.

```
3 4 5
<++++++++[>+++++++++<-]>>>>>--->++++<<<<<.++++>>>>++++++>-<<<<<<<
<<<<<<<<<+++[>--------<-]>+>>>>>>>>>>.+++++++++++>>>>>>>>--------
---<<<<<<<<<<<<<<<<<<<+++++[>++++++++<-]>->>>>>>>>>>.-----------<
<<<<<<<<<<+++++[>--------<-]>++>>>>>>>>>>.-----------------------
--------------------->>>>+++++++++>++++++++++++++++++++<<<<<.
```

Now that we all know how to write a "Hello World" program in Ringfuck we can all put it on our resumes and call ourselves experts.

The astute among you may be wondering about mutually prime wheel sizes or the possible state spaces of a set of wheels, if so then this brief mathematical interlude is just for you!

A Brief Mathematical Interlude A ringset is a set $S = \{R_0, \ldots, R_{m-1}\}$, where each of the rings R_i has n_i elements $\{x^i_j \in D; j = 0, \ldots, n_i - 1\}$ (we will use superscripts on values to denote which ring they come from, and subscripts to denote their place on the ring). The domain D is, in our case, the set $\{0, 1, \ldots, \text{ff}_{\text{hex}}\}$ for computer reasons.

All of the operations in a program affect the "indicated" value, which takes one value from each ring in S. Therefore, if the indicated index is k, then the current value of ring R_i is $x^i_{k \pmod{n_i}}$, and the indicated value of the ringset as a whole is the ordered sequence $X = (x^0_{k \pmod{n_1}}, \ldots, x^{m-1}_{k \pmod{n_{m-1}}})$.

In the language described above:

- `>`/`<` increment and decrement the indicated index k,
- `+`/`-` increment and decrement the values $\{x^i_k; i \in 0, \ldots m-1\}$ all together,
- `.` prints each of $(x^0_{k \pmod{n_1}}, \ldots, x^{m-1}_{k \pmod{n_{m-1}}})$ in `ascii` encoding ordered as they appear in the sequence,
- and `,` takes external values $(y^0, \ldots, y^{m-1})$ and inserts them into the indicated values X.

A helpful way of expressing the positions of the rings is through their individual indicated indices, the ordered sequence: $I = (k \pmod{n_1}, \ldots, k \pmod{n_{m-1}})$. Due to the coupling of the rings under `>`/`<`, the ringset as a whole can be thought of as a ring of memory, with the amount of increments to k required to return to the same place being the least common multiple K of the elements of I. We call a ringset *pleasant* if the size $K = \prod n_i$, i.e. if all the ring sizes are mutually prime.

For later, it will be convenient to remember that ringset positions are only possible to achieve if for all $i, j \in \{0, \ldots, m-1\}$ they satisfy

$$I_i \equiv I_j \pmod{\gcd(n_i, n_j)}.$$

From here, we arrive on the most interesting aspect of the ringset, which is the shared memory between different X's. We say that two ringset indices k_1 and k_2 are i-linked (denoted with $\overset{i}{\leftrightarrow}$) if the ith ring has the same indicated index. In other words:

$$k_1 \overset{i}{\leftrightarrow} k_2 \iff k_1 \pmod{n_i} = k_2 \pmod{n_i}.$$

We say two indices are $\{i_1, i_2\}$-linked if they are both i_1-linked and i_2-linked.

We say that two indices are linked if there exists an $i \in \{0, \ldots, m-1\}$ such that they are i-linked. Otherwise they are not linked.

To gain intuition about memory linkage, we provide graphs, see Fig. 1. The figure represents different linking by different colors. The graph of the color representing the ith ring is made up of n_i separate complete graphs of size K/n_i, offset by one each time. (Naturally, one can easily consider the notion of ij-linking which considers the linking of the jth position of the ith ring, which is just a subset of the larger i-link equivalence relation). Since the memory linkage is predetermined from the size of the rings, constructing a useful memory structure is somewhat finicky, but possible.

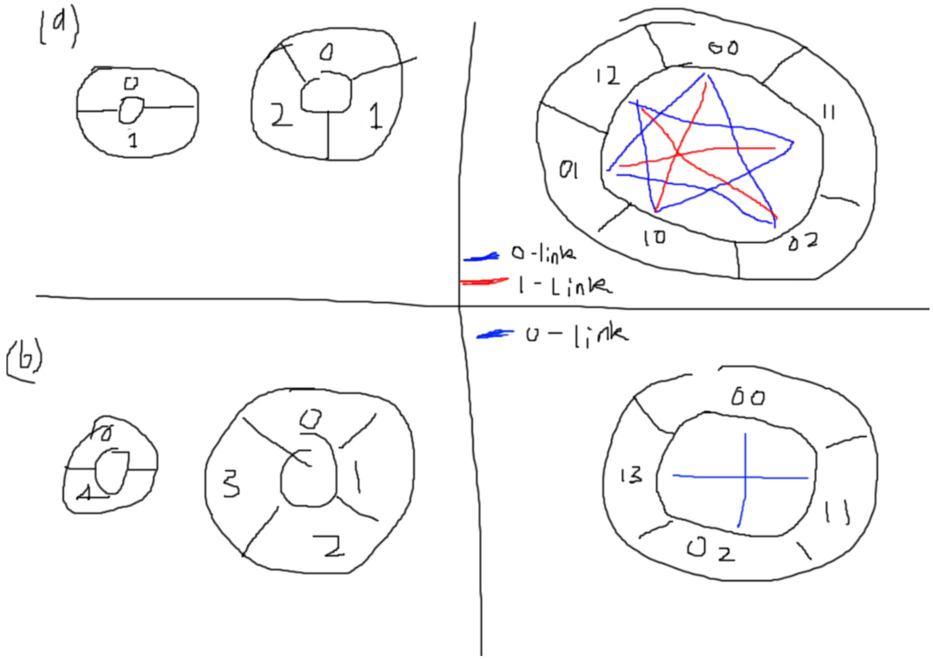


Fig. 1. (a) rings, combined rings with links for 2 3. (b) rings, combined rings with links for 2 4.

Note that for non-pleasant ringsets, the memory-linkage graphs are constrained by the fact that many ringset positions are not possible. An important result of this restriction is the result of several separate complete graphs within the total linkage graph. Fig. 1 (a), for example, is pleasant and has a single complete linkage graph, when all edges are combined, while in (b), there are two complete but not connected graphs formed by the vertices 00 and 02, and 11 and 13. This separation is easily described by the equivalence relation where two vertices $I_i = (x_i^0, \dots x_i^{m-1})$ and $I_j = (x_j^0, \dots x_j^{m-1})$ are in the same complete graph if and only if

$$x_i^s (\pmod{\gcd(n_s, n_t)}) = x_i^t (\pmod{\gcd(n_s, n_t)}), \qquad \forall s, t \in \{0, \dots, m-1\}.$$

For reasons evident later, we call the vertices of these separated graphs *offset families*

Another important aspect of the ringset structure relates to how the `+/-` operators affect m values simultaneously. This causes some complications as to which ringset configurations are possible.

For example, let's consider for now rings with the domain $D = \mathbb{Z}$, so that there is no wrap-around effect due to overflow. For a single ring, any possible configuration of numbers is possible using the `+` and `-` operators.

Also note, there is a minimal number of `+` and `-` operators needed to reach a given configuration from the configuration of all zeros, but no maximum number of operators, as they cancel each other out. This means that a program can achieve a given configuration from the zero configuration only if the number of `+` operators N_+ and `-` operators N_- is such that $N_+ - N_- = \sum x_i$, the sum of all the values in the configuration. This will be important later.

From here, consider the case of a pleasant ringset with two wheels. Now, it is no longer possible to achieve any possible configuration of the ringset because of the coupling between the two. Since the operators act on both wheels, the offset between the sums of their configurations is invariant:

$$\sum x_i^0 - \sum x_i^1 = \text{const.}$$

However, we will show that this invariant (which we will refer to as the *offset invariant*) is a necessary and sufficient condition for a particular configuration to be reachable from a given starting configuration.

($\Rightarrow$) **Any given program preserves the offset.** See above.

($\Leftarrow$) **If two configurations have the same offset, then there exists a program that moves one to the other.** This arises from the discussion about program length above. Since the two configurations have the same offset invariant, then for each individual ring, there exists a program to transform one to the other, and they both have the same quantity $N_+ - N_-$.

$$\begin{aligned} 0 &= \left(\sum \tilde{x}_i^0 - \sum \tilde{x}_i^1\right) - \left(\sum x_i^0 - \sum x_i^1\right) \\ &= \left(\sum \tilde{x}_i^0 - \sum x_i^0\right) - \left(\sum \tilde{x}_i^1 - \sum x_i^1\right) \\ &= (N_+^0 - N_-^0) - (N_+^1 - N_-^1) \end{aligned}$$

Both have a minimal program, so it suffices to take the larger of the two, and rotate the rings so that both programs run and cancellations occur as needed on the ring with the smaller minimal program.

This proof easily extends to any finite pleasant ringset.

Finiteness is required for there to exist a largest minimal program, but what if the ringset isn't pleasant? For a simple counter-example, consider a ringset with two rings of the same size. In this case, the rings are always rotationally in sync, and a single offset invariant is not enough to guarantee that one configuration can be reached from another. Instead, recall our definition of *offset families*. These, which we arrived at from understanding restrictions on index combinations due to rings with non-mutually prime sizes, each require their own offset invariant, as they correspond to completely independent sections of memory on the ring.

However, in practice, we don't work with the domain $\mathbb{Z}$, but rather with some number of bits, where upon reaching the maximum value, it overflows back to zero. From here, we will consider cells with values ranging from 0 to n.

Previously, offset was simply defined as the difference in the sums of a pair of wheels. In this case, however, it is entirely possible to achieve different offset values due to overflow. The actual invariant in this case is offset mod $n + 1$. In this case, when an overflow transition happens (and `+` maps $n \to 0$ and `-` maps $0 \to n$), the offset mod $n + 1$ is still conserved:

$$\begin{aligned} \text{given: } & \sum x_i^1 - \sum x_i^2 && \equiv C \pmod{n+1} \\ \text{then: } & \left(\sum x_i^1 - n\right) - \left(\sum x_i^2 + 1\right) \\ & \equiv C - (n+1) \pmod{n+1} \\ & \equiv C \pmod{n+1} \end{aligned}$$

This completes the forward condition, wherein every program preserves the offset mod $n + 1$.

By now, it should be intuitive that the order of increment (`+`) and decrement (`-`) commands does not matter to the program, as long as the rings are rotated correctly. This fact, combined with the fact that the minimal programs for each wheel must differ by a multiple of $n + 1$ in order for the final results to have the same offset mod $n + 1$, we see that we can construct our program by grouping commands in groups of $n + 1$. By rotating the wheels in such a way that the commands are executed in the correct place on certain wheels, while in the same place on others, it is possible to utilize overflow to reach an intermediate configuration that has the same offset (not just the same offset mod $n+1$). From there, we use the earlier result for normal offsets.

In practice, this proves that in order to have complete freedom in assigning values on a pleasant ringset, one cell on each ring except the first must be reserved for the offset, and more cells for unpleasant ones.

Alternatively, a creative solution to the issue of offsets is through the use of the input command (`,`) to manually change the offset on different rings, making different configurations accessible.

3 Conclusion

We reiterate that Ringfuck is not the addition of quality of life features to an existing programming language. It is a new paradigm, a new way of thinking about memory, and a constant reminder of the ordinary person's material insignificance in the circle of life.

It's about coming full circle.

It's about getting jerked around by the circular nature of all things, all while fully coming into the circle.

It's a circle jerk.

Anyways, circular memory also has some natural benefits. For example Ringfuck, in addition to training programmers to be ring fit (Nintendo, 2019), it supports the efficient rounding of numbers (they are already round in memory, what else do you want?), and it is inherently safer than blocks of memory, as demonstrated by Federal Highway Administration Office of Safety (2015) in their work on round-abouts vs. intersections. It is also intuitive to use in most areas of scientific computing, which frequently deal with spherical animals in airless environments, but unfortunately not in orbital mechanics, which deals primarily with ellipses.

It's easy to see why this language will be a massive success. In fact, it's provable! Using only a minor extension to the ZFC axioms of set theory that includes the axiom: "Ringfuck is a massive financial success", we have single-handedly paid for our mortgage. The proof is as follows:

> Ringfuck is a massive financial success because (as per axiom 9) Ringfuck is a massive financial success.

Some might object that this line of reasoning tater-tautological (however tasty). However, we are simply basing our methodology on circular reasoning, which has been popular with philosophers, scientists, and especially politicians since ancient times. It is yet another sign that circular logic, which we hope to bring to the world of programming languages, has been underutilized in favor of more conventional, but less powerful, methods like common sense.

At the end of the day, just remember that if you like something, you should put a ring on it, and that Ringfuck is a wheel great time!

4 Acknowledgements

Special thanks to Spheal, from Pokemon.

5 Disclosure

Sponsored by some company that makes hula hoops or whatever. We disclose that we are directly funded by big circle.

Bibliography

Federal Highway Administration Office of Safety (2015). *Accelerating Roundabout Implementation in the United States* **IV**.

Garland, P. (2015). Arrival.

Hornby, R. (2023). ringfuck interpreter. https://github.com/RyanHornby/ringfuck.

Nintendo (2019). Nintendo ring fit adventure. `https://ringfitadventure.nintendo.com/`.

Perlis, A. J. (1982). Special feature: Epigrams on programming. *SIGPLAN Not.* **17**, 9, 7–13.

Stellman, S. D. (1973). A spherical chicken. *Science* **182**, 4119, 1296–1296.

Thanh, C. H. (2022). br++: A much needed extension to brainfuck. *SIGBOVIK* .

Appendix

Below is a simple “Hello World” program. If you still don’t believe it works feel free to run it through our interpreter (Hornby, 2023).

```
3 4 5
{First load ‘H’ (72 in ascii) into the "0" cell of all the rings}
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++
{Set the "0" cell of ring 2 to ‘E’}
>>>>
---
{Set the "0" cell of ring 3 to ‘L’}
>
++++
{Print all the "0" cells}
<<<<<
.
{Set the "0" cells to ‘LIP’}
++++
{Set the "0" cell of ring 2 to ‘O’}
>>>>
++++++
{Set the "0" cell of ring 3 to ‘ ’}
>
------------------------------------------------
{Print all the "0" cells}
<<<<<
.
{Set the "0" cells to ‘WZ+’}
+++++++++++
{Set the "0" cell of ring 2 to ‘O’}
>>>>
-----------
{Set the "0" cell of ring 3 to ‘R’}
>
+++++++++++++++++++++++++++++++++++++++
{Print all the "0" cells}
<<<<<
.
{Set the "0" cells to ‘LDG’}
-----------
{Set the "0" cell of ring 3 to ‘!’}
>>>>>
--------------------------------------
```

```
{Print all the "0" cells}
<<<<<
.
{Set the "0" cells to ‘ ’, 24, and 245}
--------------------------------------------
{Set the "0" cell of ring 2 to ‘ ’}
>>>>
++++++++
{Set the "0" cell of ring 3 to ‘\n’}
>
+++++++++++++++++++++
{Print all the "0" cells}
<<<<<
.
```

NovelML: A Novel SML Semantics

tbrick (the t stands for thea ♡), jgallicc (the jga stands for jgay)

$LOCKDOWN + (3*365.25 + 16)*24*60*60*1000

1 Abstract

Previous work has shown the potential of taking Python and transforming it into an ML-like language. In this novel work, we consider the categorical dual[1], where we introduce a novel semantics for ML, modeled on the performant and industry proven pythonic approach.

2 Introduction

In prior SIGBOVIK proceedings, Ng *et al vel non est alius* 2022, proposed taking "the opposite approach to the popular literature" and in doing so, "develop[ed] a revolutionary new system which makes the Python language Slightly More Likeable." In our novel paper, we take the opposite of the opposite approach to popular literature. This is novel research because we are intuitionists, and reject double negation elimination. Thus this research is new and novel and therefore worth publishing.

In this novel paper, we ask the following novel question: Why should I waste my time statically determining types when our program already knows the type at runtime. We replace SML's outdated 1997 typechecker with fancy and novel dynamic checks – which are critically acclaimed in industry.

Given that this approach is critically acclaimed in industry, our novel approach had industry applications before publication, thus making it worth studying (in addition to being novel).

3 A Novel Definition for NovelML

3.1 Syntax

The Definition™ of Standard ML™(1997™) provided an (at the time) novel syntax. Given the perfection of this original syntax for SML (specifically the `abstype` declaration and the wonderful `infix` and `infixr` declarations).

[1] in whatever Grothendieck universe this terminology makes sense in

	expressions	statements
statics	$O(1)$	$O(1)$
dynamics	$O(1_{\pm 2^n})$	$O(1)$

Figure 1: Asymptotic complexities of the statics/dynamics of expressions/statements (measured in milliseconds)

In order to support the novel features of our new language, we must extend the existing Basis Library[2]. After much literature review (which is what paper's that should be published do), we determined that following CMU Alumnus and creator of Java, James Gosling's approach would work best. So we introduce the keyword `novel.machine.learning.lang.System.Runtime.Type.f''''` which implements casting and hopefully balances the familiarity of Java (very useful and informative namespaces) with common ML-isms (mysterious variable names and lots of `'`s). We were going to call it `Unsafe.cast`, but SML/NJ already took that one. Then we were going to call it `Object.magic`, but Jane Street has already done enough damage to OCaml that we felt using it would be inappropriate[3] .

3.2 Novel Statics

Our novel typing rules are straightforward for expressions, and complicated for everything else. First the simple stuff:

```
---------------------- [expr]
    |- e : \tau
```

This novel semantics improves on Ng *et al vel non est alius* 2022, by dropping the context, which is no longer needed for typechecking expressions.

Typechecking declarations is more complicated:

```
-------------- [statement]
   |- s valid
```

Oh, okay, nevermind then. See? NovelML is elegant.

3.3 Novel Dynamics

NoML has simple expression statics and simple declaration statics, but in contrast has complicated expression dynamics and simple declaration dynamics (see Figure1).

To define NoML's dynamics, we need only add one novel exception to those presented in **The Definition**™™. This exception, `No` (short for "not well-typed"), indicates a runtime type error. Here are the simple declaration dynamics:

[2]we would cite a source here but we couldn't find any documentation for SML's basis library.

[3]See this based[4]tweet

[4]based on what?

1. `datatype`, `type`, `abstype` declarations are meaningless in NoML, and thus raise `No`[5]

2. `structure`, `signature`, `functor` declarations are confusing in NoML, and thus raise `No`[6]

3. `infix` is a parser flag (the syntax of SML is context sensitive). Thus it does not raise `No`, but is a no-op (novel op) in the dynamics.

4. `val` and `fun` are standard variable binders. See section "Novel Variables" for details.

The complicated expression dynamics are basically the same as SML but if there's a type error at runtime, raise `No`. See the examples section for more details.

Due to difficulties in implementing NoML, we also had to add a maximum recursion depth with default value 1000. This depth can be changed by binding to `setrecursionlimit`:

```
val setrecursionlimit = setrecursionlimit + 1/2
```

4 Novel Variables

A common confusion when learning Standard ML is variable bindings, in particular the notion that variables *shadow* other variables. In this novel language, we follow legendary former president George W. Bush Jr.'s philosophy of "No child left behind", adapted to the novel catchphrase: "No variable left in the shadows."

Variables (sometimes synonymously called assignables) are re-assigned when referred to multiple times.[7] This novel philosophy leads us to some very intuitive behavior, like in the following program, which actually mimics the familiar behaviour of Python.

```
val x = 5
val () =
  case 3
    of x => ()
val 3 = x
```

NovelML's novel dynamics ensure this executes with no (visible) exceptions, unlike in SML (which raises a non-descript "match bind" on this program!)

[5] Type definitions are a big issue students have when learning SML. Not in NoML!

[6] Modules are a big issue students have when learning SML. Not in NoML!

[7] One may wonder whether this renders NovelML not Turing complete. The authors, of course, do not care about Turing completeness, but for the sake of less-enlightened readers, we include a proof of Turing completeness:

Every NovelML program has a finite number of variables. However, each variable can hold an arbitrary (unbounded) natural number. Thus, NovelML programs can be used to simulate register machines with an arbitrary, finite number of registers. This system is known to be Turing complete. The other direction is, of course, a novel exercise for the reader. □

4.1 `op=`

SML uses `=` in two places, val declarations and equality expressions. NovelML unifies these concepts in the most logical way, with the `op=` operator also performing binding/assignment. Since it is still an expression, we adopt C's novel behavior and return the novel value of the variable that was just bound. The following program demonstrates a novel usecase of this behaviour.

```
val x =       5
val true      =
  if x = ~3 < 0
    then   true
  else    false
(* x is now ~3 *)
```

NovelML's novel dynamics ensure this executes with no (visible) exceptions, unlike in SML (we were unable to coax SML/NJ to run this program to completion, and are thus unsure of its runtime behavior.)

4.2 `ref`

This begs the novel question: what is the difference between this and `?.X1 ref`[8]? To answer this we must appeal to a little known section of **The™ Definition™**, where is clarifies that `ref` means "refuse to mutate"[9]. While industry is skeptical of the benefits of variables that refuse to mutate, we believe there may one day be a novel use case, thus include them.

In NoML, attempting to bind to a value that has the runtime type of `ref` will raise the type error `No`. The `!`[10] function simply evaluates to the novel value of the variable that refuses to mutuate (if the type is correct). But, if the variable is an integer, then it is simply the factorial function. The `:=`[11] function returns `0`[12] if the value on the right hand side is the same as the "refuse to mutate", otherwise it raises `No`.

In the following novel program, we demonstrate the power of "no variable left in the shadows" as well as the utility of the de-refuse-to-mutate/factorial function:

```
val ? = fn () => !8
val 40320 = ?() (* 8 factorial! *)

fun fib 0 = 0
  | fib 1 = 1
  | fib n = fib (n - 1) + fib (n - 2)
```

[8] Oh no! The value restriction!

[9] Unfortunately the page was lost during the printing process.

[10] BANG!!!!!!!!!!!11!1 ...or de-refuse-to-mutate I guess...

[11] omg! a walrus

[12] the success error code

```
val ! = fib
val 21 = ?() (* 8 fibonacci! *)
```

5 Examples

1. The following raises the `No` exception:

   ```
   datatype 'a except = Ok of 'a | Error of string
   ```

2. The following prints `we made it!` before raising the `No` exception:

   ```
   fun polymorphism (x : int except) =
     (* Luckily, x can be of any type. *)
     let val result = x + 5 in result end

   val 5 = polymorphism 0
   val () = print "we made it!"
   val 5 = polymorphism (Ok 0)
   ```

3. The following runs to completion without visible errors:

   ```
   val x = 1
   fun addX (n : int) : int = n + x
   val x = 2

   val 5 = addX 3
   ```

4. The following raises the `No` exception, since a string is not a constructor for `exn`:

   ```
   raise "ur a naughty programmer uwu"
   ```

5. The following example prints 0:

   ```
   val () = print (Int.toString (
     (let val "hi" = 5 in true end) handle No => 0
   ))
   ```

6. The following example prints 0:

   ```
   val () = print (Int.toString (
     () (* Unit can be implicitly cast to 0 *)
   ))
   ```

7. The following example prints 0:

   ```
   val () = print (Int.toString (
     if "true" then 0 else "1 / 0"
   ))
   ```

8. The following example prints 0:

```
val () = print "0"
```

9. The following example is a function that computes the sum of a tree:

```
fun treeSum Empty rights acc =
      case rights of
        []      => acc
      | r :: rs => treeSum r rs acc
  | treeSum (Node (l,x,r)) rights acc =
      let
        val rights = r :: rights
        val acc    = x + acc
      in
        treeSum l rights acc
      end
```

Alternatively, we could implement it as follows:

```
fun treeSum tree =
  let
    val acc = 0
    val rights = []
    (* It's good practice to copy variables
     * into obscure names so you don't
     * interfere with the global namespace.
     *)
    val t = tree

    fun treeSum' () =
      case t of
        Empty => (
          case rights of
            [] => acc
          | r :: rs => treeSum' ()
        )
      | Node (l, x, r) =>
          let
            val rights = r :: rights
            val acc = x + acc
            val t = l
          in
            treeSum' ()
          end
  in
    treeSum' ()
  end
```

6 Conclusion

In this novel work, we explored the remarkable properties that arise from combining the familiarity of SML's syntax with the expressiveness of Python's runtime behavior. We discovered fun mottos along the way:

- "if it parses, it runs"
- "no variable left in the shadows"
- "`No`"

Guided by these mottos, we shall bring forth a novel society of programmer-citizens, primed with the necessary tools to raise a new world order.

Although this work is entirely novel, there are still scraps of future work we left out to give other authors a shot at being published in SIGBOVIK 2024. We list some suggestions here:

- Use NovelML to write a novel (or to do anything else (you can be our first user!))
- Find ways to reduce the complexity of NovelML's dynamic semantics (likely trivial and not novel and not worth publishing)
- Determine the complexity of the decision problem, "does the NovelML program p raise the `No` exception, assuming p has 0 occurrences of the `No` identifier? " (we suspect it to be NP-hard at least)
- Bake a batch of cookies (see References section)

7 References

1. https://tex.stackexchange.com/questions/571314/make-a-color-box-with-the-trans-pride-flag
2. Fiber Bundles and Intersectional Feminism
3. SML Help which we (the authors) helped write, that's how you know its trustworthy and novel.
4. Typestate-Oriented Programming
5. The Gospel of Wealth
6. Lord Eshteross' Maple Ginger Cookies with Turmeric

A Halt-Averse Instruction Set Architecture for Embedded Hypercomputers

Ben Burrill [0x7F6, 0x76D]
California State University, Northridge

April 0, 2023

Abstract

In this work, we aim to bring a few of the more practical performance benefits of hypercomputers to embedded applications by limiting ourselves to a computational model—an ISA with a Turing jump instruction—which is not even Turing complete. Nevertheless, it is capable of general-purpose computation, and is semi-optimized for oracle-oriented programming (OOP). It may be emulated on traditional processor architectures, albeit at a slight performance penalty, using the emulator made freely available at https://github.com/benburrill/sphinx.

1 Introduction

In low-power embedded environments, the demands of unrestricted hypercomputation can pose some technical complications. To provide limited hypercomputation for embedded applications, we present Sphinx, a halt-averse instruction set architecture. While not strictly hypercomputational, nor even Turing complete, Sphinx provides some of the amenities familiar to hypercomputer programmers.

The conditional-branch instructions of conventional architectures are replaced by a Turing jump instruction, which offers limited insight into the machine's own halting problem. Specifically, a jump is performed if not jumping would lead to halting. Conditional execution is enabled through use of conditional-halt instructions.

From a performance standpoint, programs running on a Sphinx architecture can send half a bit of information back in time, allowing algorithms to optimize away special cases.

From a backwards compatibility standpoint, Sphinx may be emulated even on non-hypercomputational devices.

From a programmer's standpoint, Sphinx combines the readability of assembly languages with the traceability of logic programming languages.

2 Theory

2.1 Halting problems

Famously, the halting problem of Turing machines is undecidable. If an algorithm existed to solve the halting problem of Turing machines, you could program a Turing machine to run this algorithm on itself, and then do the opposite of whatever it says. Since Turing machines do not exist, this bears no real-world relevance, and need not concern us in any way, so long as we choose to willfully reject Turing machines from our thoughts (Murphy VII 2008).

Many machines—including all real-world computers—have halting problems (excluding I/O) that *are* decidable. A trivial algorithm exists to solve the halting problem for machines with finitely many possible configurations in finite time (Simmons 2021).

However, in stating that the halting problem of Turing machines is undecidable, we did not directly invoke the fact that Turing machines have infinitely many possible configurations, instead making use of the relationship between decidability and computability-on-a-Turing-machine. More generally, this can be extended into a statement about self-computability:

Theorem 1. *Computers*[1] *cannot solve*[2] *their own halting problem*[3]*, even if their halting problem is decidable.*

2.2 Semi-semi-decision procedures

One the better workarounds for the annoying restrictions due to Theorem 1 is the concept of semi-decidability. The halting problem of Turing machines is semi-decidable, and more generally halting problems are semi-self-computable. That is, there exists an algorithm implementable on machine M that will tell you, in finite time, whether any halting program for machine M will halt. For non-halting programs, this algorithm will loop indefinitely. Formally, this algorithm is known as "running the program", which we shall abbreviate to SDPSHP as the semi-decision procedure for the self-halting problem.

The Sphinx architecture allows the SDPSHP algorithm to be optimized and made more user-friendly. A program $h(p)$ can easily be written for a Sphinx machine, S_n with n bytes of state, that will *immediately* produce output indicating whether or not any program p running on S_n would halt.[4]

The slight of hand here is that $h(p)$ is actually still just a semi-decision procedure with the same termination behavior as SDPSHP—it will only terminate if p would. However, unlike SDPSHP which runs p in all cases, $h(p)$ may be implemented using Sphinx's Turing jump instruction so that

[1] As in, something vaguely capable of general purpose computation

[2] As in, determine in all cases eg. a boolean value that it can use

[3] Assuming its halting problem isn't trivial–like if it can never halt

[4] $h(p)$ might require more *instructions* than p by a constant amount, depending on how you implement it. By considering only bytes of *state* we are neglecting this. That alone doesn't protect Sphinx from the corrupting influence of logic, since we can still write a program that tests itself in much the same way as we can test any other program. See examples/self_inquiry.s

it *only* runs p for the $(1 - \Omega) \times 100\%$ of all programs that will not halt.[5] If p would halt, it is never run.

Although characteristically identical to the SDPSHP, there are some practical advantages to $h(p)$. If we want $h(p)$ to output to the user "p will not halt" only in the case that p will not halt, we can do so by simply prepending that code to p since it won't be run at all if p would halt. Since such prepended code is part of what is being tested, it is not contradictory to prepend a halt—it would simply invalidate the test, just as it would for the SDPSHP.

Since it is a somewhat less semi- version of a semi-decision procedure, we describe this as a semi-semi-decision procedure. Unlike the SDPSHP algorithm, $h(p)$ makes a decision straight away. To an outside observer, the problem is fully decided. However, the program itself obtains no more usable information than could eventually be found by any other semi-decision procedure. In this sense, Sphinx cannot solve its own halting problem, it only looks like it does.

A formal semi-proof is provided in Appendix A for picky readers who demand more rigor.

2.3 The Turing jump operator

According to the entry on Wikipedia, the Free Encyclopedia (Wikipedia contributors 2023), the Turing jump operator is defined on "sets" (presumably hash sets). The Turing jump X' of the lookup table X is a new lookup table, indexed by Gödel's perfect hash function φ, such that $\forall p : X'[\varphi(p)]$ is true iff the program p would halt given access to the lookup table X.

Unlike many mathematical operators such as addition, hardware support for the Turing jump operator has historically been lacking. Traditional oracle-equipped hypercomputers are based around precomputed Turing jump tables. Although this approach works well for many hypercomputational systems, microcontrollers typically do not have the available resources to store such large lookup tables.

3 Design of the Sphinx ISA

3.1 The Turing jump instruction

Sphinx's Turing jump instruction `j` operates on the principle of self-preservation against future halts. A jump is only performed if not jumping would lead to halting. Like all Sphinx instructions, the Turing jump instruction is specified to take one clock cycle.

It is also important to explicitly specify that the jump instruction is deterministic. At a repeated state, the jump instruction must always make the same decision. Since a jump is *only* to be performed if not jumping would lead to halting, there is never any ambiguity here[6] as to which deterministic decision should be made.

Sphinx's jump instruction eschews true hypercomputation in favor of cold hard pragmatism. It does not offer as much power as a conventional Turing jump table (Section 2.3), from which one may obtain a full bit of information about whether some code would halt. With Sphinx one may only obtain half a bit of information. You're forced to actually do the work to get the other half of the bit in the case that not jumping would not lead to a halt.

3.2 The state vector

Instead of separating registers and main memory, the Sphinx ISA is based around a unified concept of state. That is, it is a memory-only architecture—there are no registers. Instead, immediately-addressed state values may be used directly in instructions, much like registers. There are load/store instructions for dynamic state access. Additionally, there is a read-only memory section called const.

Most instruction arguments can be any of the following:

1. Immediate values
2. An immediate address to look up a value from state
3. An immediate address to look up a value from const

Syntactically, `yield 0` outputs[7] the immediate value 0, `yield [0]` outputs the value loaded from address 0 in state, and `yield {0}` outputs the value loaded from address 0 in const. Notice that there is no unified address space, 0 may separately be a state address and a const address.

The `code` section is separate from state and is inaccessible (Sphinx is a Harvard architecture). The program counter is also not in the `state` section, it is an inaccessible pseudo-register. We'll define the "complete state" as the combination of the program counter and `state` section.

3.3 Halt instructions

The `halt` instruction is an unconditional halt.

Additionally, there are an assortment of conditional halt instructions: `heq`, `hne`, `hlt` (not to be confused with `halt`), `hgt`, `hle`, and `hge`. These perform signed comparisons. For example `heq [x], 0` will halt execution if the word in state at label `x` is equal to 0.

3.4 Word size

The Sphinx specification does not mandate a particular word size. If your code requires a 3 byte (24-bit) word size, the preprocessor command `%format word 3` can be used. Word offset literals may be used in the assembler. For example, with 3 byte words, `2w` = 6. The infinite word size (Section 5.1) may be specified with `%format word inf`.

4 Sphinx Programming

4.1 Terminal non-termination

Prophecy 1. *If you read this paper, you will eventually die*

Prophecy 1 is pretty much worthless unless you had some expectation of immortality. If that was the case for you, I'm very sorry, it's too late now—maybe should've led with that.

For the same reason, most Sphinx programs are designed to be non-terminating in order to make effective use of the prophetic wisdom provided by the Turing jump instruction.

[5] TODO: Plug in the value of Chaitin's constant for Ω

[6] Hopefully anyway. See Appendix A if you have any questions.

[7] Sphinx provides generic output capabilities through `yield`, but does not specify any means to read real-time input from peripherals.

Although non-termination is often a very desirable feature (Simmons 2021), occasionally one wants to write a program that, after completing its primary computational task, indicates that it is done and performs no additional work. To do this while maintaining immortality, the standard technique in Sphinx is to simply produce a `done` flag and then enter into an infinite loop.

This pattern is called **terminal non-termination**. An example may be seen in the following code:

```
1 flag done
2 tnt: j tnt
3 halt
```

Not jumping on line 2 would clearly lead to the subsequent unconditional halt, so the jump is unconditional.

4.2 Example: max value

This code searches for the maximum value of an array, jumping to `found_max` as soon as the maximum value is first encountered:

```
1  mov [max_val], {arr}
2  j found_max
3
4  mov [addr], arr + 1w
5  loop:
6  hge [addr], end_arr
7      lwc [cur_val], [addr]
8
9      j continue
10     hle [cur_val], [max_val]
11
12     mov [max_val], [cur_val]
13     j found_max
14
15     continue:
16     add [addr], [addr], 1w
17 j loop
18 halt
19
20 found_max:
21 yield [max_val]
22 flag done
23 tnt: j tnt
24 halt
```

Running this code under the Sphinx emulator using the array [9, 8, 7, 6, 5, 4, 3, 2, 1, 0] produces the following output:

```
$ spasm max_value.s
9
Reached done flag
    CPU time: 4 clock cycles
    Emulator efficiency: 5.80%
```

Nevermind the "emulator efficiency", that is just an implementation detail of the emulator. The point is that it found the maximum value of an unstructured array of 10 elements in just 4 clock cycles!

How does it work? To understand the code, let's first recognize that reaching `found_max` guarantees non-termination (Section 4.1). The loop is set up to halt if we reach the end of the array (line 6). Once in the loop, line 13 presents the only opportunity to avoid a halt by jumping to `found_max`. Lines 9 and 10 force lines 12 and 13 to be skipped unless `cur_val > max_val`. So we only have the opportunity to jump to safety whenever we reach a value larger than any previously encountered. The jump on line 13 will only be taken if not jumping would lead to halting, i.e. if no larger value would be found. So the code breaks out early as soon as the maximum value is encountered.

4.3 Time complexity

It's worth noting that the worst-case time complexity of our prophetic max-value algorithm (Section 4.2) is the same $O(n)$ as in the traditional algorithm. It is also only able to match the time complexity of search algorithms of the kind presented by Freshman 2022 in the best-case, though it does not suffer from any loss in accuracy.

A true oracle-equipped hypercomputer could do even better, with an $O(\log(n))$ algorithm—since the index of the maximum value may be expressed with $\lceil \log_2(n) \rceil$ bits, the hypercomputational algorithm would be to simply use a halting oracle to determine each bit of the index.

However, the *best* measure of performance is obviously the *best*-case time complexity, where we get a perfect score of $O(1)$. Additionally, the $O(\log(n))$ algorithm would fail for finding the maximum value of linked lists with elements of arbitrary size, so in a more general sense, this algorithm is hypercomputationally optimal.

4.4 Halt propagation

In code that may lead to a halt, the form of conditional branch used in lines 9-10 of max-value (Section 4.2) can sometimes be problematic since you are testing both the condition *and* any future halt.[8] In cases where this is a problem, one strategy is to propagate halts backwards in time by halting on the complement of the condition you are testing if the branch is taken. That way, if a jump is performed for the "wrong" reason, the code will still correctly halt. This works well[9] so long as you want an entire block to be skipped if it would lead to a halt (which is usually the case). If you need halting code that actually runs all the way to the "real" cause of the halt, halt propagation would not be sufficient, and you may need a more sophisticated form of conditional branch, such as in examples/halting.s.

4.5 Conservation of magic

Interpreters may be written in Sphinx for languages with Sphinx-like halt aversion in a way that makes use of the halt-aversion provided by `j` (not just emulated as in Section 6.2). In principle, a self-interpreter should be possible.

See examples/sphinxfuck.s for a true interpreter for a brainfuck-style language that features halt-averse forwards and backwards jumps instead of the usual `[` and `]`.

---

[8] This is not a problem for max value since the jump on line 13 handles the case of a future halt.

[9] For an example making use of halt propagation in various interesting ways, see examples/decimal.s. It's a rather complicated one, but see for example the `done` label at the end.

5 Sphinx Extensions

This section specifies two extensions to the instruction set, which are made optional due to potential difficulties that may be encountered in their implementation, such as logical contradictions.

5.1 Infinite Sphinx

One obvious deficiency of Sphinx is that it is not strictly Turing complete. The problem is that Sphinx fundamentally has only finitely bounded state, and so cannot be used to replace microcontrollers with infinite on-chip memory. This can be remedied by introducing an infinite word size, so that words can represent any integer. Sadly this makes the jump instruction undecidable.

Beyond mere undecidability, there are also a few other details which complicate the implementation of Infinite Sphinx. Here we will specify requirements that must be followed by conforming Infinite Sphinx implementations:

1. Memory is organized into word cells, composed of bytes. Both sides of a word cell shall be addressable.
2. Integers shall be represented with an alternating-endian byte-order, where the least-significant byte is stored at the start of the word cell, and the next-least-significant byte is stored at the end, increasing inwards. There shall not be a most-significant byte.
3. Depending on the instruction in which it is used, a word of data may be interpreted as either an integer, an address, or a word-cell-offset.
4. The bijective mappings between integers, addresses, and word-cell-offsets are left unspecified, and hence pointer arithmetic is undefined behavior in most cases.
5. Addition using the `add` instruction shall be defined between addresses and word-cell-offsets, producing a corresponding address in an offset word-cell.
6. Multiplication using the `mul` instruction shall be defined to scale word-cell-offsets.
7. The offset argument of `lbso`, `lbco`, `lwso`, `lwco`, `sbso`, and `swso` instructions shall be interpreted as an integer to offset the address by the given number of bytes.

5.2 Suicide Sphinx

One may also contemplate a variant of `j`, which we shall call `k`, for which a jump is performed if not jumping would *not* lead to halting. That is, if `j` operates on the principle of self-preservation (Section 3.1), `k` is suicidally averse to immortality. As it turns out, although `j` and `k` are seemingly similar, `k` is logically problematic as an instruction.

Notice that the combination of `j` and `k` is very powerful:

```
j would_halt
k would_not_halt
; Some code that might halt
```

It's not just powerful—it's too powerful for its own good! We've damn near violated Theorem 1, so it should be *trivial* to produce a contradiction from the above code.

Ok, turns out it's not actually so trivial[10] (or at least I couldn't manage it with this approach). But regardless, we can make an even stronger statement. It's not just the combination of `j` and `k` that is a problem, but `k` itself.

Theorem 2. *Suicide Sphinx is logically unsound*

Proof. We will show that the problem of getting up out of bed is reducible to Suicide Sphinx.

It is widely accepted that getting up out of bed is an undecidable decision problem, at least in the short-snooze limit (Me this morning). In the getting up out of bed problem, we desire to get some more sleep under the condition that we get up eventually.

In the following code we reduce this problem to Suicide Sphinx, using the `k` instruction to get up out of bed if and only if we otherwise never would:

```
1 alarm_goes_off:
2 k get_out_of_bed
3 sleep 15 * 60 * 1000
4 j alarm_goes_off
5 halt
6
7 get_out_of_bed: halt
```

Undecidability follows from the usual proof for the getting up out of bed problem: the decision of whether or not to jump on line 2 is predicated upon making the opposite decision in the same situation.

We only used `j` for an unconditional jump, something that's impossible with `k` alone, but would certainly be a necessary feature in any instruction set featuring `k`.

Therefore, we must reject the suicidal jump instruction from any logically consistent instruction set, regardless of its apparent utility. □

6 Implementation

6.1 Time travel

The most obvious way to achieve a constant-time jump instruction is through the application of time travel. However, although Sphinx itself is robust against paradox (Appendix A), user error might introduce bugs into your code, along with minor time travel paradoxes and possibly the destruction of the universe.

The simplest implementation would be to just make the processor physically impossible to turn off or destroy. This would mitigate the disasterous influence of user intent upon a program's behavior. However, if the ability to turn off and destroy the processor is a desired feature, these should not be treated as a halt, as that would cause all jumps to be unconditional (Munroe 2013).

6.2 The Sphinx emulator

The Sphinx assembler/emulator `spasm`, which may be obtained from `https://github.com/benburrill/sphinx`, is currently the only working implementation of Sphinx.

`spasm` uses a rather inefficient form of emulation that implements the jump instruction by simulating the future rather than predicting it outright.

[10] If `would_not_halt` halts it breaks the test

In the emulator, there is a concept of “virtual” and “real” execution, with the main difference being that no output is produced under virtual execution. Emulation begins in real execution. Once a jump is encountered, to determine if not jumping would lead to halting, the emulator performs a search for an infinite loop under virtual execution starting from `pc + 1`. If a path to an infinite loop is found, the entire future path of execution has been determined and is cached for use by all future jumps. Otherwise, if a halt would be reached, the emulator simply performs the jump.

In searching for an infinite loop, we keep track of the complete state of the program at jump points so that we may restore it in the event that not jumping would lead to halting, and also to test for repeated states.[11] If we reach a state has been visited previously, we have found an infinite loop and may unwind the stack producing a (cyclic) list of all jump decisions that were taken leading to the loop.

If the state has not been previously visited, we recursively attempt to skip the jump. If that leads to a halt, we try taking the jump instead. If that also halts, we unwind to the previous jump point.

In the worst-case, the number of previous states that must be kept track of scales exponentially with the amount of state. This is usually totally fine, but there are certain applications for which this may be problematic.

In computer graphics, it can sometimes be necessary to produce a delay lasting until the heat-death of the universe (Oh et al. 2022). One possible Sphinx port based on the Python implementation of Oh’s algorithm is shown:

```
; Use a 336-bit word size
%format word 42
%section state
counter: .word 100000000000000000000000000000000000
↪ 00000000000000000000000000000000000000000000000
↪ 000000000000000000

%section code
; Play an animation
; TODO

; Wait until heat-death of universe
loop:
    sleep 365 * 24 * 60 * 60 * 1000
    sub [counter], [counter], 1
j loop
hgt [counter], 0

; After the heat-death of the universe, enter
↪ terminal non-termination
flag done
tnt: j tnt
halt
```

In and of itself, this code performs well under emulation. However, if the animation needs to predict the future, there would be some issues. Specifically, if any jump instruction during the animation ever needs to be skipped, the entire heat-death loop would need to be fully predicted, which may result in unacceptably poor performance under emulation.

The minimum system requirements for available memory to play such an animation with a skipped jump under `spasm` may be conveniently expressed using the newly-adopted quetta- SI prefix as being on the order of 1 TQQQB. Using an estimate based on the Steam hardware survey as of February 2023 (Valve Corporation 2023), this would exceed the capabilities of at least 99.1% of personal computers.

Virtual execution in the emulator does not sleep, so prediction of each iteration of the loop would be much faster than in reality. As a result, although there might be a bit of a delay somewhere during the animation, it would only be a small fraction of eternity, leaving plenty of time for the animation to play. So the main concern is the memory use.[12]

The real problem with memory efficiency of course is that the emulator is written in Python, which incurs an outrageous memory overhead. Python’s `bytes` objects alone have an overhead of 33 bytes. Do you know how much you can fit in 33 bytes? That’s like 80% of what you need for the heat death of the universe. Python could include this remark!

7 Future Work

I was hoping to get away with just implementing Sphinx for my compilers class this semester, but unfortunately the language we create is supposed to have superfluous features like a type checker. So that plan backfired and I have been suckered in to implementing both Sphinx (done-ish) *and* a high-level oracle-oriented language that compiles to Sphinx (so far, I have implemented a lexer for it). Hence there is much need for an excuse for why I inevitably can’t finish the code generator for this high-level language at the end of the semester. With the logical soundness of Sphinx assured (Appendix A) it is unclear what form this excuse could take.

The separation of `code`, `state`, and `const`, along with Sphinx’s adjustable word size, complicates the categorization of its busy beavers. A von Neumann Sphinx with a fixed word size as in Simmons 2021 might be more advantageous to the study of bewitched breadboard busy beavers.

Ever since some teenager recklessly breached the gates of hell (Szewczyk 2020), there has been a pressing need for a stronger Malbolge. Increasing its cryptographic strength would likely be the most effective approach, but incorporating Turing jump instructions into the design might also help to increase the space complexity of producing working programs for it.

To further develop the embedded hyperprocessor, one clear goal would be to determine how much more we can gently tickle the halting problem without getting bit. Can we construct even semier-semi-semi-decision procedures?

8 Conclusions

Jeez did I write all that? Well uh, pretty much, Sphinx is quite cool. Go play around with it I guess—maybe write a time-traveling merge-sort or something.

[11] We actually only check for repeated states at upwards jumps.

[12] At least in this case. The time complexity of emulated jumps is worst-case super-exponential, which should make them an asymptotically far stronger hardware decelerator than prior NaN-gate techniques (Jones 2020). In practice however, I find memory use—though merely exponential—to be the bigger concern unless you add pointless jumps.

References

Freshman, A College (2022). "A sometimes-accurate O(1) search algorithm". In: *A Record of the Proceedings of SIGBOVIK 2022*. The Association for Computational Heresy, p. 223.

Jones, Cassie (2020). "NaN-Gate Synthesis and Hardware Deceleration". In: *A Record of the Proceedings of SIGBOVIK 2020*. The Association for Computational Heresy, pp. 61–65.

Me (this morning). *What time is it? Urgh my head is spinning. I'm just going back to sleep.*

Munroe, Randall (2013). *Halting Problem*. URL: https://xkcd.com/1266 (visited on 03/20/2023).

Murphy VII, Tom (2008). "A non-non-destructive strategy for proving P = NP". In: *A Record of the Proceedings of SIGBOVIK 2008*. The Association for Computational Heresy, pp. 13–15.

Oh, Braden et al. (2022). "Solving Double Execution of Java's paint() Method by Counting Down to the Heat Death of the Universe (plus language compendium)". In: *A Record of the Proceedings of SIGBOVIK 2022*. The Association for Computational Heresy, pp. 111–137.

Simmons, Robert J. (2021). "Build your own 8-bit busy beaver on a breadboard!" In: *A Record of the Proceedings of SIGBOVIK 2021*. The Association for Computational Heresy, pp. 278–281.

Szewczyk, Kamila (2020). *MalbolgeLISP v1.2*. URL: https://github.com/kspalaiologos/malbolge-lisp (visited on 03/21/2023).

Valve Corporation (2023). *Steam Hardware & Software Survey*. URL: https://store.steampowered.com/hwsurvey (visited on 03/21/2023).

Wikipedia contributors (2023). *Turing jump — Wikipedia, The Free Encyclopedia*. URL: https://en.wikipedia.org/w/index.php?title=Turing_jump&oldid=1134853969 (visited on 03/19/2023).

Appendix A Proofs

In this section I will endeavor to beguile, guilt-trip, and if necessary hold hostage all those who question the logical soundness of Sphinx.

Theorem 3. *Sphinx is logically sound*

Semi-proof. Here we will establish the logical soundness of Sphinx by recursive Gish gallop.

Potato. Can you refute that one? Nope didn't think so.

Consider the following code:

```
1  %section state
2  is_sphinx_broken: .word 0
3
4  %section code
5  j would_halt
6  flag sphinx_is_perfect
7  loop: hne [is_sphinx_broken], 0
8  j loop
9  halt
10
11 would_halt: flag sphinx_is_broken
```

I am running this code with the Sphinx emulator under `pdb`. Let us suppose that I will be informed of a contradictory Sphinx program or similar problem that you discover. I would then use `pdb` to change the value of the word at `is_sphinx_broken` in state. This would cause a halt on line 7, trigger an `AssertionError` in `Emulator.step`, break my heart, and destroy the universe[13]—thereby alerting my past self to the issue, since the jump on line 5 would need to have been taken. The output of this program is as follows:

```
$ python3.11 -m pdb -c c -m spasm test_sphinx.s
Reached sphinx_is_perfect flag
```

Since the universe exists last I checked, this clearly demonstrates that no problems with Sphinx will ever be found.

If you're still not convinced that Sphinx is logically sound (or if you're convinced that it is not), read the semi-proof of Theorem 3 before continuing.

Therefore, Sphinx is logically sound. □

[13] As a corollary, this means that you would be responsible for the destruction of the universe, you monster.

D

Complexity, Meet Simplicity

Simultaneous Paper Maximization and Minimization Through Reference List Side Channel Information Injection

Frans Skarman

Read 1st char of reference list [1–156]

REFERENCES

[0] **Peter Hall**. Peter Hall Observations. 2018-02
[1] **Achuthankutty et.al**. Biological Collection, National Institute of Oceanography, Goa, India. 2018-03
[2] **Peggy L. Dixon**. Agriculture and Agri-Food Canada, St. Johns, Newfoundland. 2018-03
[3] **Ernesto Ruelas Inzunza**. Hawk Migration Association of North America - HawkCount. 2018-03
[4] **Roy Haschenburger**. (UTC-MO) University of Tennessee - Molluscs Collection. 2018-03

[5] **Mary Sollows**. NBM birds. 2018-02
[6] **Isabel Calabuig**. Fungal Specimens collected by HabitatVision (Jacob Heilmann-Clausen). 2020-02
[7] **Norbert G Kondla**. Norbert Kondla Collection. 2018-02
[8] **Isabel Calabuig**. Botanical Museum, Denmark. Database of registrations of red listed plants. 2020-02
[9] **Mary Sollows**. NBM Unionoids. 2018-02
[10] **Isabel Calabuig**. P. W. Lund collection in the National History Museum of Denmark, Copenhagen. 2020-02
[11] **Zulmary Valoyes et.al**. Caracterización Ecológica de la Fauna del Páramo de Tatamá. 2018-12
[12] **Alan Wormington**. Point Pelee National Park Collection, Canada. 2018-02
[13] **Travis LaDuc et.al**. TNHC Herpetology Collection. 2020-07
[14] **Isabel Calabuig**. Pilularia Globulifera distribution map in Denmark. 2020-02
[15] **Oliver L. Pescott & Chris D. Preston (joint Recording Secretaries) et.al**. Bryophyte data for Great Britain from the British Bryological Society held by BRC: Atlas 2014. 2019-12
[16] **Nina Zitani**. University of Western Ontario Collection. 2018-02

[17] **Alan Wormington**. Alan Wormington Collection. 2018-03
[18] **Nancy Noble**. Manitoba Museum of Man and Nature. 2018-02
[19] **Dagmar Triebel**. The Myxomycetes Collections at the Botanische Staatssammlung München - Collection of Hermann Neubert. 2019-04

[20] **Mike Gollop**. M. Gollop Collection. 2018-02
[21] **Annette Wolter ; NA**. Visual Plants (144.41.33.158) - Plants from Costa Rica. 2018-03
[22] **Xavier Ferrer Parareda et.al**. VerteCAT: Banco de datos de vertebrados de Cataluña. 2018-11
[23] **Isabel Calabuig**. Priest Pot species list, Cumbria, Britain. 2020-02
[24] **Murray Braun**. Agriculture and Agri-Food Canada, Saskatoon. 2018-03
[25] **Isabel Calabuig**. Botanical Museum, Copenhagen. Database of type specimens. 2020-02
[26] **Zulmary Valoyes et.al**. Caracterización Ecológica de la Ornitofauna del Humedal Costero Obregón. 2018-12
[27] **Adam Korosi**. Database of invertebrates collected in Mongolia. 2018-05
[28] **Toru Shinohara et.al**. Freshwater Fish Specimens of Lake Biwa Museum. 2020-03
[29] **Isabel Calabuig**. Botanical Museum, Copenhagen, Mycology Herbarium. 2020-02
[30] **Olivier Gerriet**. Collection d'animaux marins et plantes du Museum d'Histoire Naturelle de Nice. 2018-09
[31] **Noriiko Matsumoto**. Vascular plants collection of Hiratsuka City Museum. 2020-03

[32] **Anton Van de Putte et.al**. Antarctic Echinoids: an interactive database. 2020-09
[33] **Roy Haschenburger**. (UTCFM) University of Tennessee - Fossil Mammals Collection. 2018-03
[34] **Esther García Guillén et.al**. Real Jardín Botánico: Dibujos de la Real Expedición Botánica del Nuevo Reino de Granada (1783-1816), dirigida por J.C. Mutis. 2019-07

[35] **Wouter Van Landuyt et.al**. Belgian IFBL Flora Checklists (1939-1971). 2019-03
[36] **Erno Kuusela**. Invertebrate collection of University of Oulu Zoological Museum. 2018-03
[37] **L Alan Prather et.al**. Michigan State University Herbarium Lichens. 2020-12
[38] **Louise Dumouchel**. Bombus of Canada. 2018-02

[39] **Sara Hemly et.al**. C.A. Triplehorn Insect Collection (OSUC), Ohio State University. 2018-03
[40] **Tadayasu Kono et.al**. Insect specimens of Omogo Mountain Museum. 2020-03
[41] **Ulf Soltau ; NA**. Visual Plants (144.41.33.158) - Plants from Southern Ecuador. 2018-03
[42] **Dr. Heinz Martin Schumacher et.al**. DSMZ Collection on Plant Cell Cultures. 2018-05
[43] **Isabel Calabuig**. Herbarium Faeroense. 2020-02
[44] **Erik Verheyen et.al**. rmca-albertine-rift-cichlids. 2019-03
[45] **Dr. K.D. Hyde**. The University of Hong Kong Herbarium. 2018-02

[46] **Barry Conn**. Plants of Papua New Guinea. 2018-02
[47] **Ursula Toom**. TalTech fossil collections. 2020-01
[48] **Toshio Kawai**. Fish Collection of Hokkaido University. 2020-03

[49] **Oddvar Pedersen et.al**. Vascular Plants, Field notes, Oslo (O). 2019-12

[50] **NRW NBN Administrator et.al**. Great Crested Newt Special Area of Conservation (SAC) Monitoring Data. 2020-11
[51] **Ledis Regalado**. CUBA: Herbario de la Academia de Ciencias, La Habana, Cuba: HAC-Pteridophyta. 2020-06
[52] **Yoshikazu Hasegawa et.al**. Gunma Museum of Natural History, Amphibia and Reptile Specimen. 2020-03

[53] **Isabel Calabuig**. Entomology Department Collections, Natural History Museum of Denmark. 2020-02
[54] **Néstor Basso et.al**. Colección Herpetológica de la Patagonia. 2020-09

[55] **Isabel Calabuig**. Botanical Museum, Copenhagen, the Lichen Herbarium. 2020-02
[56] **Stephanie Boucher**. Lyman Entomological Museum. 2018-02
[57] **Osamu Tadauchi**. Hymenoptera specimen database of Kyushu University. 2019-08
[58] **Lluís Vilar Sais et.al**. Universitat de Girona: HGI-Cormophyta. 2020-06
[59] **Angus Atkinson et.al**. Antarctic Krill occurrence data from BAS expeditions. 2019-03
[60] **Toru Shinohara et.al**. Plant Specimens of Lake Biwa Museum. 2020-03
[61] **Isabel Calabuig**. Marine Benthic Fauna List, Island of Læsø, Denmark. 2020-02
[62] **Otakar Šída**. Botanical collection of the National Museum. 2018-05
[63] **Nobuo Ohbayashi**. Ehime University Coleoptera Collection. 2019-08

[64] **Wim Wendelen et.al**. African Mammalia. 2019-09
[65] **Emilie Wadsworth et.al**. South Lanarkshire peatland records 2013. 2019-12

[66] **Phillip Neal**. MICROBIS database. 2018-03
[67] **Roy Haschenburger**. (UTCFB) University of Tennessee - Fossil Birds Collection. 2018-03
[68] **Orlando Silveira**. Museu Paraense Emilio Goeldi - Lepidoptera Collection. 2019-08
[69] **Palmira Carvalho et.al**. LISU Lichen collection. 2020-11
[70] **Orlando Ludovic et.al**. DNA and Tissue Bank of the Natural History Museum of Denmark. 2020-02
[71] **Steve Marshall**. University of Guelph, Department of Environmental Biology. 2018-02
[72] **Emilie Wadsworth et.al**. Invasive Non-native species data in the Clyde catchment, collated by Central Scotland Forest Trust. 2019-12

[73] **Starri Heiðmarsson**. Herbarium (AMNH). 2018-02
[74] **Isabel Calabuig**. Nivå Bay species list, Zealand, Denmark. 2020-02
[75] **Mercedes Pérez et.al**. Fonoteca Zoológica, Museo Nacional de Ciencias Naturales, Madrid: FZ_INSECTA. 2018-11
[76] **Ute Muehlenhardt-Siegel et.al**. Biogeographic distribution of Antarctic and sub-Antarctic Cumacea. 2019-03
[77] **Linda L. Long**. U.S.D.A. Forest Service, Redwood Sciences Laboratory - Lamna Point Count. 2018-03
[78] **Toshio Kishimoto**. Mollusca collection, Museum of Natural and Environmental History, Shizuoka. 2020-03
[79] **Angus Atkinson et.al**. Antarctic Krill occurrence data from Discovery expeditions. 2020-06
[80] **Norihisa Tanaka et.al**. Bryophyte specimens of Kanagawa Prefectural Museum of Natural History. 2020-03
[81] **Erling Holm et.al**. Ichthyology Collection - Royal Ontario Museum. 2020-11
[82] **Ogoudje Isidore AMAHOWE**. Collection of occurrence data on animal species in the Biosphere Reserve of Pendjari in 2001, 2004, and 2013.. 2018-08
[83] **Ugo Dall'Asta et.al**. rmca-albertine-rift-butterflies. 2019-03
[84] **Steve Timmermans**. Canadian Lakes Loon Survey. 2018-02

[85] **Pedro Cardoso**. Morano and Cardoso: AraIb. Base de datos de arañas ibéricas. 2020-06
[86] **Anton Van de Putte et.al**. Collections data on ecology of bottom animal of the Southern ocean. 2019-03
[87] **Pilar Rodriguez et.al**. Colección de Oligoquetos Acuáticos de la UPV/EHU. 2020-06
[88] **Edward Davis**. Condon Fossil Collection. 2018-02
[89] **Roy Haschenburger**. (UTCR) University of Tennessee - Reptiles Collection. 2018-03

[90] **Mercedes Pérez**. Fonoteca Zoológica, Museo Nacional de Ciencias Naturales, Madrid: FZ_AMPHIBIA. 2020-06
[91] **Isabel Calabuig**. Botany registration database by Danish botanists. 2020-02
[92] **Norman Johnson**. Proctotrupoidea - xBio:D Cyberinfrastructure, The Ohio State University. 2019-04
[93] **Isabel Calabuig**. University of Copenhagen Arboretum. 2020-02
[94] **Mercedes París**. Museo Nacional de Ciencias Naturales, Entomología. 2020-06
[95] **Isabel Calabuig**. Western Palearctic migratory birds in continental Africa. 2020-02
[96] **Zulmary Valoyes et.al**. Caracterización Ecológica de la Fauna del Páramo de Frontino o del Sol. 2018-12
[97] **Anton Van de Putte et.al**. Antarctic Amphipod Crustaceans: Ant'Phipoda Database (BIANZO). 2019-03
[98] **Toru Shinohara et.al**. Insect Specimens of Lake Biwa Museum. 2020-03
[99] **Isabel Calabuig**. The Danish Newt Collection. 2020-02
[100] **Ogoudje Isidore AMAHOWE**. Collection of Occurrence data on mammal species during wildlife census in W Biosphere Reserve in 2013 and 2015.. 2018-08
[101] **Norman Johnson**. Endeostigmata - xBio:D Cyberinfrastructure, The Ohio State University. 2019-04

[102] **Anton Van de Putte et.al**. Nemertina World Checklist. 2019-03
[103] **Norman Johnson**. Diaprioidea - xBio:D Cyberinfrastructure, The Ohio State University. 2019-04
[104] **Dr. Oscar Alberto Flores Villela**. Coleccion de Herpetologia, MZFC-UNAM. 2018-02

[105] **Mariola Kukier-Wyrwicka**. Botanical Garden Collection. 2018-02
[106] **Anton Van de Putte et.al**. Bacteria and chlorophyll-a water column observations (surface to 150m), April-August 2001, Continental Margin Western Antarctic Peninsula, GLOBEC. 2019-03
[107] **Xilola Ergasheva et.al**. Water reservoir's algoflories of Fergana valleys and their comparative analysis. 2019-06
[108] **Isabel Calabuig**. Galathea II, Danish Deep Sea Expedition 1950-52. 2020-02
[109] **Mike Cadman**. Ontario Breeding Bird Atlas 1981-1985. 2018-02
[110] **Isabel Calabuig**. Arthropod fauna in christmas tree plantations. 2020-02
[111] **Zulmary Valoyes Cardozo et.al**. Caracterización Floristica del Páramo Tatamá. 2018-12
[112] **Andrej Seliskar**. FloVegSI-FAV - Faunistical database of ZRC SAZU. 2018-02
[113] **Tom Müller**. BÜG. 2018-03
[114] **Isabel Calabuig**. MycoKey - online photos. 2020-02
[115] **Ogoudje Isidore AMAHOWE**. Species composition in twelve (12) Afzelia africana Sm & Pers populations in Benin.. 2018-08
[116] **Norman Johnson**. Heterozerconidae - xBio:D Cyberinfrastructure, The Ohio State University. 2019-04

[117] **Bodil Bluhm**. Arctic Ocean Diversity. 2018-03
[118] **Yoshikazu Hasegawa et.al**. Gunma Museum of Natural History, Fish Specimen. 2020-03

[119] **Isabel Calabuig**. Galapagos grasses and sedges. 2020-02
[120] **Norman Johnson**. Ceraphronoidea - xBio:D Cyberinfrastructure, The Ohio State University. 2019-04
[121] **Joaquín Giménez Héau**. CNAN/Scorpiones. 2018-03
[122] **Erin Kuprewicz**. UConn Mammals. 2020-02
[123] **Cedric Gillott**. University of Saskatchewan. 2018-02
[124] **Tom Müller**. Pfrimmersbachtal. 2018-03
[125] **Isabel Calabuig**. Tanzanian Vertebrate Collection. 2020-02
[126] **Norman Johnson**. Platygastroidea - xBio:D Cyberinfrastructure, The Ohio State University. 2019-04
[127] **G. Farley**. Bird Collection. 2018-03

[128] **Philippe Martin et.al**. Herbarium of Namur. 2019-03
[129] **Andrej Seliskar**. FloVegSI - Floristical and fitocenological database of ZRC SAZU. 2018-02
[130] **Priscilla Burgoyne et.al**. GRBGT: Southern Cape Herbarium Specimen Collections (1882-2008). 2018-01
[131] **Evariste AHOLOU et.al**. Plan Aménagement et de Gestion Simplifié (PAGS) des forèts sacrées des départements de l'Ouémé-Plateau et le Collines. Données mobilisées dans le cadre du projet JRS Bénin. 2018-08
[132] **Roy Haschenburger**. (UTCA) University of Tennessee - Amphibians Collection. 2018-03

[133] **Cris Guppy**. Crispin S. Guppy Collection. 2018-02
[134] **Ogougbe Isidore AMAHOWE**. Forest inventory for above ground carbon estimation in Biosphere Reserve of W.. 2018-08
[135] **Norman Dignard et.al**. Herbier du Québec (QUE) - Collection de plantes vasculaires. 2019-01
[136] **Tom Müller**. LaBoOb02. 2018-03
[137] **Evariste ALOHOU et.al**. Ȑépertoire des Forêts sacrées dans les Départements de l'Ouémé et du Plateau. Data mobilized in the framework of JRS Biodiversity Foundation project. 2018-08
[138] **Norman Johnson**. Cynipoidea (gall wasps) - xBio:D Cyberinfrastructure, The Ohio State University. 2019-04
[139] **Tom Müller**. Frau Voss. 2018-03

[140] **Vishwas Chavan**. IndOBIS, Indian Ocean Node of OBIS. 2018-03
[141] **Isabel Calabuig**. DBL_Life. 2020-02
[142] **Anton Van de Putte et.al**. Walter Herwig 1978 (FFS): SeaStars (Echinodermata, Asteroidea). 2019-03

[143] **Tom Müller**. Am Moosangerweg. 2018-03
[144] **Hannu Saarenmaa**. Lepidoptera collection of Hannu Saarenmaa. 2019-09
[145] **Evariste ALOHOU et.al**. Ȑépertoire des Forêts sacrées dans les Départements de l'Ouémé t du Plateau. Data mobilized in the framework of BID National project BID-AF2015-0065-NAC and funded by EU. 2019-06

[146] **Roy Haschenburger**. (UTCB) University of Tennessee - Birds Collection. 2018-03
[147] **Elisabeth Jiménez**. Hormigas en cafetales con diferente intensidad de manejo en el departamento del Cauca. 2018-06
[148] **Francisco Javier Salgueiro González et.al**. Herbario de la Universidad de Sevilla. 2019-04
[149] **Egil Ingvar Aune**. Vegetation data from phytosociological studies of the forests on the Fosen peninsula. 2019-08
[150] **Roy Haschenburger**. (UTC) University of Tennessee Mammals Collection. 2018-03
[151] **Emily Thomas et.al**. Living with Mammals survey. 2020-11
[152] **Norman Johnson**. Formicidae (ants) - AntBase and xBio:D Cyberinfrastructure, The Ohio State University. 2019-04
[153] **Cris Guppy**. Crispin S. Guppy Observational Records. 2018-02
[154] **Ester Vega Elices et.al**. Banco de Germoplasma de la Universidad de Salamanca. 2018-04

[155] **Luis Miguel Fernández Blanco**. Colecciones Zoológicas de la Universidad de León, Colección de Bivalvos. 2020-06
[156] **Isabel Calabuig**. Danish Ants (Formicidae). 2020-02
[157] **Steve Timmermans**. Marsh Monitoring Program - Birds. 2018-02
[158] **Tom Müller**. DJH. 2018-03

Appendix A: An actual paper

1 BACKGROUND

Paper minimization, the act of writing the shortest possible paper, is a subject with a long history of incremental improvements[2–6]. Recently, researchers have also started studying paper maximization[1], writing the longest possible paper that can still get published. However, to date as far as the authors are aware, no previous research investigates simultaneous maximization and minimization.

The key observation enabling our proposed method is that references are often not counted towards the length of papers, with several conferences and journal allowing extra space for references outside the normal page limit. In such settings, the proposed method produces papers which are both minimal, consisting only of a single short sentence in the body and maximal in the number of pages.

2 METHOD

In the interest of reproducibility[1] in science, the tool that was developed to generate the above reference list is open source and available for download[2]. The tool downloads a list of thousands of papers published to Arxiv, and then generates a formatted reference list based on a text file containing paper content.

3 RESULTS

As can be seen from the proof of concept, the body of the paper is 6 words or 39 characters long. Crucially, this does not change with the information content of the paper, which means that the non-references list information complexity per paper size of our method is $O(1)$.

For the purposes of maximization, our method uses around two lines of paper content per character of paper content. Comparing this to the state of the art in paper maximization is difficult, most previously proposed methods inject content via citation format expansion. Citation format expansion paper size is $O(nc)$ for c citations of length n, whereas our method grows by $O(m\bar{c})$ for a paper with $\bar{c}$ characters and citation meta-data of length m.

There are several reasons to prefer our method. Finding relevant references[3] is tedious, at least more so than simply writing random text and having a tool expand the text to take up more space on the page. For example, one can make heavy use of examples to exemplify proposed methods and claims.

4 THE REFERENCE LIST SIDE CHANNEL

The reference list side channel exploited for paper maximization can also serve other purposes, primarily injecting more content in papers for submissions where references are not counted towards the page limit. In the present work, this is of little use unless references are completely unbounded, due to the low information content per page area. For example, this paper used only 2 sentences to fill 3 pages.

In order to properly exploit this side channel, more work to increase the information content is required. For example, one might use the first word of paper titles as the information deliverable. However, one has to be careful not to devise a too complex scheme, as that risks using more space for the description of how to read the injected information, than is actually delivered via the injection.

5 FUTURE WORK

This work serves as a proof of concept, however, some issues remain. The main issue here is the formatting of the reference list. Some particularly picky publishers may object to using vertical space to mark sentences, leading to a rejection and undoubtedly, sadness. In order to mitigate this, one might exploit the fact that citations of multiple papers with the same authors replaces the author name with – in some reference styles.

Another issue is the lack of special characters, injecting something like an equation or actual citations require special characters such as [and ⊕. While it may be possible to find a select few papers published by people whose names start with those characters, finding enough to write a rigorous mathematical and well referenced paper may prove difficult.

6 CONCLUSION

We present a novel method to synthesise papers which are simultaneous maximal and minimal. To do so, we exploit a previously unexplored avenue for paper information injection via a side channel attack on the reference list. The proposed method is compared to the state of the art, both for maximization and minimization, a comparison which shows asymtotic surperiority in information content per character written.

REFERENCES

[1] Josh Abrams. 2021. On Sigbovik Paper Maximization. In *Proc. Sigbovik*.
[2] Thomas Bach. 2019. Is "Dicong Qiu. Is This theShortest SIGBOVIK Paper?From 2018 SIGBOVIK Paper"the Shortest SIGBOVIK Paper?. In *SIGBOVIK*.
[3] Mitchell Jones. 2019. Is this the tiniest SIGBOVIK paper ever?. In *SIGBOVIK*.
[4] Patrick Lin. 2019. No, this is the tiniest SIGBOVIK paper ever.. In *SIGBOVIK*.
[5] Dicong Qiu. 2018. Is This the Shortest SIGBOVIK Paper?. In *SIGBOVIK*.
[6] Richard Wardin. 2019. Revisiting the Shortest SIGBOVIK paper. In *SIGBOVIK*.

[1]And also to subject everyone to the cursed code which contains 100 lines of chained and nested iterator functions
[2]https://gitlab.com/TheZoq2/sigbovik2023
[3]Ignore the fact that we use random papers dumped from arxiv. Hopefully readers are lazy and don't actually look at the references

On the Turing Completeness of Eeny, Meeny, Miny, Moe

Javier Lim

Unemployment Agency

Abstract. We demonstrate that an infinite number of people playing Eeny, Meeny, Miny, Moe (EMMM) is Turing Complete, via a reduction from the Rule 110 cellular automata. Specifically, we provide a scheme to simulate any finite elementary cellular automaton as a game of EMMM. For N cells to be simulated, $O(N)$ EMMM players are needed, and each generation takes $O(N)$ time to step.

Keywords: Eeny, Meeny, Miny, Moe · Tiger Catching · Turing Completeness

1 Introduction

Eeny, Meeny, Miny, Moe (EMMM) is a traditional counting-out rhyme used to pseudo-randomly select a person or object from a group. In the context of children's games, EMMM can be used to select the "it" person for a game of tag, or the seeker in hide-and-seek.

One person (termed the counter) points to elements from the set to be chosen while chanting the rhyme. Each time a syllable or word is said, the counter moves to the next element. The selected element is the one that is pointed to at the end of the rhyme. A common 20-word variation of the rhyme is as follows:

Eeny, meeny, miny, moe,
Catch a tiger by the toe.
If he hollers, let him go,
Eeny, meeny, miny, moe.

In this paper, we prove that EMMM is Turing Complete via a reduction from the Rule 110 cellular automata.

2 EMMM

2.1 Variations

Let us explore the variations of EMMM that exist. While there exist the modern 20-word version, there was also a 16-word version, from around 1815 [4]:

Hana, man, mona, mike.
Barcelona, bona, strike.
Hare, ware, frown, vanac.
Harrico, warico, we wo, wac.

These variations are important, as they highlight how language evolves over time. ~~Also, I couldn't figure out how to make EMMM Turing Complete without it.~~

2.2 Definition

Let us now formalise the game of EMMM that we shall use. Each player has 3 components:

1. A list of players in the group, also known as the cycle.
2. Which player of the group they are currently pointing to.
3. EMMM variation, the number of times the next counter should count.

For instance, Alice, Bob, and Charlie are choosing who between them to be the seeker in a game of hide and seek. Alice is designated the counter. Suppose she begins by pointing at herself and uses the common 20-word version of EMMM. She counts in the cycle: Alice, Bob, Charlie. In this situation, she will end on Charlie:

initial	Eeny	meeny	miny	moe		
Alice	Bob	Charlie	Alice	Bob		
	Catch	a	tiger	by	the	toe
	Charlie	Alice	Bob	Charlie	Alice	Bob
	If	he	hollers	let	him	go
	Charlie	Alice	Bob	Charlie	Alice	Bob
	Eeny	meeny	miny	moe		
	Charlie	Alice	Bob	Charlie		

In children's games, Alice could instruct Charlie to take on the role of "it", or the seeker. However, in this setup, there are no other games than EMMM. Hence, Alice instructs Charlie to become the next counter. Charlie begins a new game of EMMM, continuing the chain.

However, there are multiple variations of EMMM. Which should be used? This choice will be made by the previous counter, who not only instructs the player to start EMMM, but also how many times to count. Hence, Alice instructs Charlie to become the counter and count, say 20, times. Note that this means that depending on who was the previous counter, the same counter can count a different number of times.

One point to touch upon is that each person's pointer is maintained throughout the run of the whole game. For instance, Alice points to Charlie on the last count. If Alice were to become the counter again, she would begin with Charlie, not herself. 20 counts later, Bob becomes the next counter.

In this scenario, Charlie has no cycle, hence cannot play EMMM. This can be interpreted as the EMMM game halting at Charlie.

3 Elementary Cellular Automaton

An elementary cellular automaton is a one dimensional cellular automaton. That is, a row of cells, each in one of two states (0/1, off/on). On the next generation, all cells simultaneously change their state. The new state is solely determined by it's own state, and the state of its immediate neighbors [2].

For instance, if the rules are:

Neighborhood	Next State
111	_0_
110	_1_
101	_1_
100	_0_
011	_1_
010	_1_
001	_1_
000	_0_

And boundaries are assumed to be 0 padded, then the state "0100110" becomes "1101110" after one generation.

3.1 Rule 110

The above table describes Rule 110 (Wolfram's notation, decimal for 0b01101110). Interestingly, it can be shown to be computationally universal [1]. Hence, if we can simulate Rule 110 within a game of EMMM, we have shown that EMMM too, is universal, via a proof by reduction.

4 Reduction from Rule 110

4.1 Notations

We will use the term "signal" loosely to mean a game of EMMM occurring. For example, Alice passing the counter role to Charlie is a signal from Alice to Charlie.

Let us introduce the following diagram notation for specifying arrangements of EMMM players:

An unfilled circle represents a player. The current counter is represented by a filled circle. A number in a player represents how many times they will instruct the next counter to count. By default, we will assume it to be 1.

A colored number next to a player represents that the corresponding color's player has this player in their cycle, at that index. The index starts at 0. An arrow starting from a player, A, and ending on another player, B, represents that A is currently pointing at B.

When a counter counts, they move from the currently pointed index to $(current + count) \mod \textit{cycle length}$.

For instance, the original Alice, Bob, Charlie situation could be represented as Figure 1.

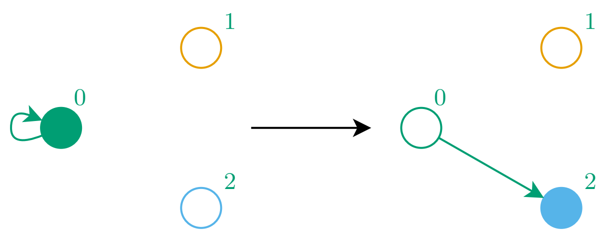


Fig. 1. Alice is the current counter, hence is a filled circle. She starts pointing at herself. Bob and Charlie are indexes 1 and 2 respectively. When Alice counts 20, it will land on $20 \mod 3 = 2$, Charlie. Charlie then becomes the next counter.

4.2 One Bit Memory (OBM)

Let us consider the arrangement of players in Figure 2.

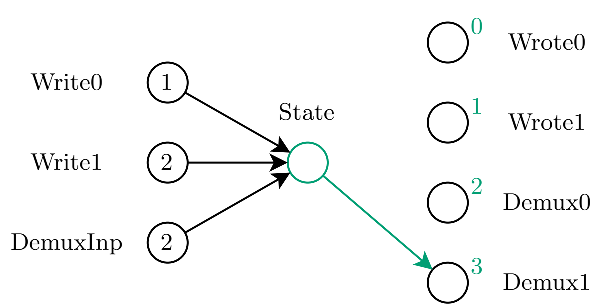


Fig. 2. An arrangement that acts as a single bit-memory. Write0, Write1 and DemuxInp only have State in their cycle, hence color and indexing are not needed.

It acts in a fashion similar to a single-bit memory. From the initial arrangement, if Write0 is the counter, it passes along the signal, ending at Wrote0. If DemuxInp later becomes the counter, the signal will be passed to Demux0.

Had Write1 been the first counter, the first signal would have ended at Wrote1. Then, When DemuxInp later becomes the counter, the signal ends at Demux1.

This arrangement allows the data (input at Write0 or Write1) to be stored without losing information (output at Wrote0 or Wrote1), then read out again later (output at Demux0 or Demux1). This "selection" is similar to demultiplexing in traditional circuits.

While an EMMM rhyme with 1 word seems unrealistic, there likely exists a 17 or 21 word rhyme, which is the same modulo the cycle length.

4.3 Resettable One Bit Memory (ROBM)

The main issue with the previous arrangement is that it is only able to be used once. The state counter will be moved to an unknown position (2 or 3), hence is difficult to work with. However, we may simply add a "Reset" circuit, as seen in Figure 3.

When State points to any of the OBM outputs (Wrote0, Wrote1, Demux0, Demux1), a signal at ResetInp will cause State to point to a corresponding reset player. This reset player simply instructs State to advance to ResetOut. Even if State is already pointing to ResetOut, it will go to RResetOut, which returns to ResetOut.

Hence, we can use ResetInp to reset the arrangement back into its original configuration. This is very useful, since it means that we can reuse the component.

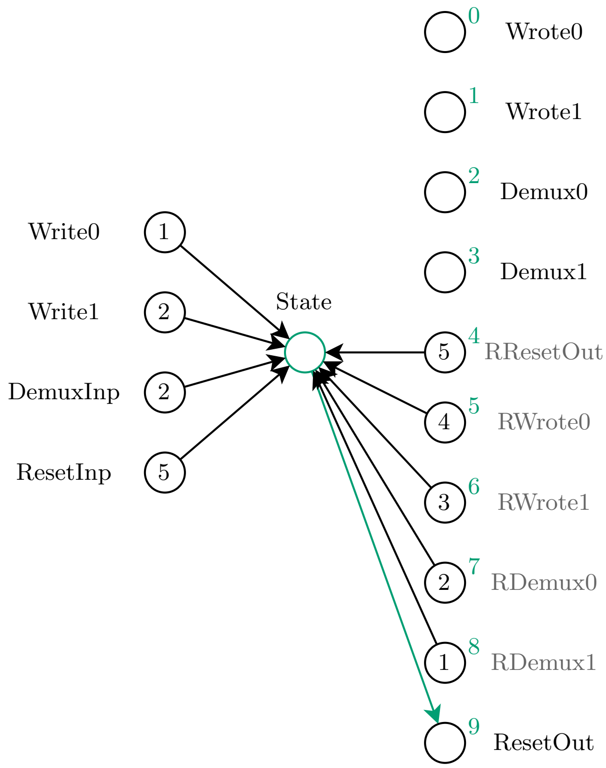


Fig. 3. A OBM with added reset circuit.

We can further abstract this arrangement. We represent the ROBM in a "chip" diagram, only caring about the exposed inputs (Write0, Write1, DemuxInp, ResetInp) and exposed outputs (Wrote0, Wrote1, Demux0, Demux1, ResetOut). We can further assume that signals coming in and out of the chip will use a count value of 1.

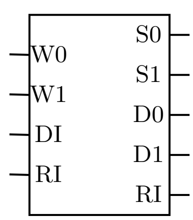


Fig. 5. Chip Representation of ROBM.

We may then use arrows from the output pins (S0, S1, D0, D1, RI) to other input pins (W0, W1, DI, RI) to indicate that the output pin player should pass on the counter role to the input pin player.

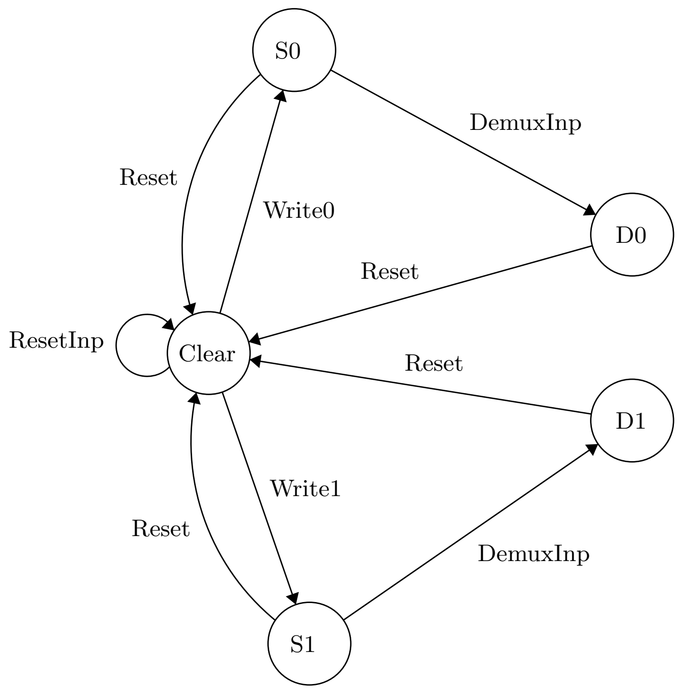


Fig. 4. ROBM Finite State Machine. Wrote is abbreviated to S, for ”Store”.

4.4 Elementary cellular automaton

Cells We may encode the state of the Rule 110 board as a line of ROBMs, each holding either a 0 or 1 (State player pointing at S0 or S1).

Neighbor Updates To perform neighbor updates, we can take one input query signal, and demultiplex it three times, once for each cell that influences the next state. Each of the 8 possible outputs has an associated next state, determined by the rule we are following. We may use this signal to write data accordingly.

Since the signal branches each time, we need multiple ROBMs per step. For instance, demultiplexing 3 times requires 1 ROBM for the first selection. Then, 2 ROBMs, one connected to each possible output. Then, 4 ROBMs for each of those possible outputs. This is illustrated in Figure 6.

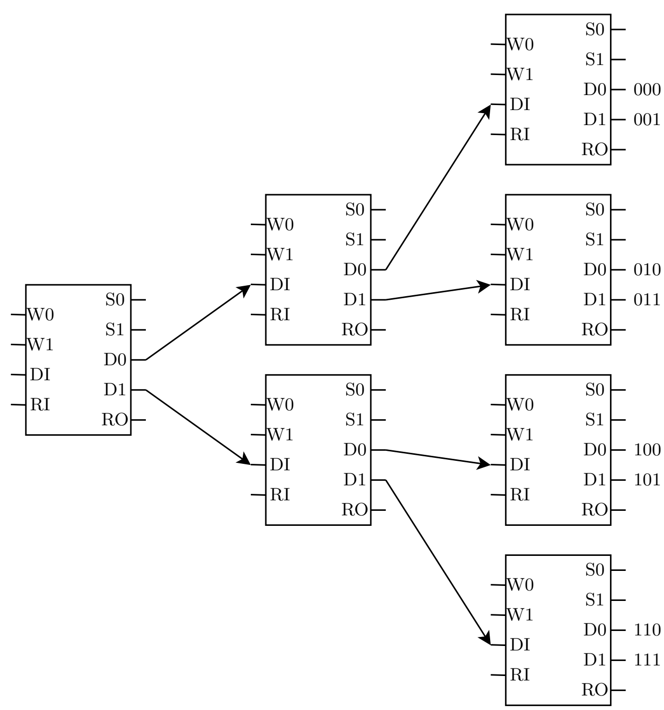


Fig. 6. Chained ROBMs to demultiplex 3 bits into one of 8 output paths. ROBMs in the same column contain the same data. Each column corresponds to the state of a cell.

Once the signal has been demultiplexed, we can apply the rules accordingly. For Rule 110, the 000 pin in figure 6 would be connected to the Write0 pin of the state ROBM, 001 to Write1, 010 to Write1, 011 to Write0, etc.

Hence, for each cell, we not only need the initial ROBM to store its state, but also 7 more ROBMs to perform selection of the rules. We can easily synchronise their states by linking their write and reset lines.

Full execution order With the state of the Rule 110 board in a line of ROBMs (the state ROBMs), and the block of chained ROBMs for demultiplexing (the rule ROBMs), the execution order is as follows:

1. Read from each state ROBMs, writing to the associated rule ROBM
2. Reset all state ROBMs
3. Perform selection on the rule ROBMs to determine the next state. Write to the state ROBMs
4. Reset all rule ROBMs
5. Go back to 1

Due to their large size, diagrams for each step are provided at the end of the section (Figures 7 to 10).

Crucially, across all steps, no pin is used more than once. There may be multiple arrows pointing into one input pin, but no output pin has more than one arrow coming out of it. This means that connections needed for all 5 steps can coexist, without the need for additional circuitry.

Edges The edges of the board require special handling, since they are missing a neighbor to read from. It is common to treat the boundaries as 0s, which is done by skipping one step of the demultiplexing. It is also possible to wrap the board around, which can be done by connecting the edge rule ROBMs together.

4.5 Analysis of execution

Each simulated Rule 110 cell uses 8 ROBM components, each of which use 15 EMMM players. Hence, for N cells, $120N \in O(N)$ EMMM players are needed.

The calculation of a new generation requires the signal to pass through each ROBM component at least twice (write, reset), and at most thrice (write, read, reset) . Hence, for N cells, $O(N)$ time is needed to step to the next generation.

4.6 Diagrams

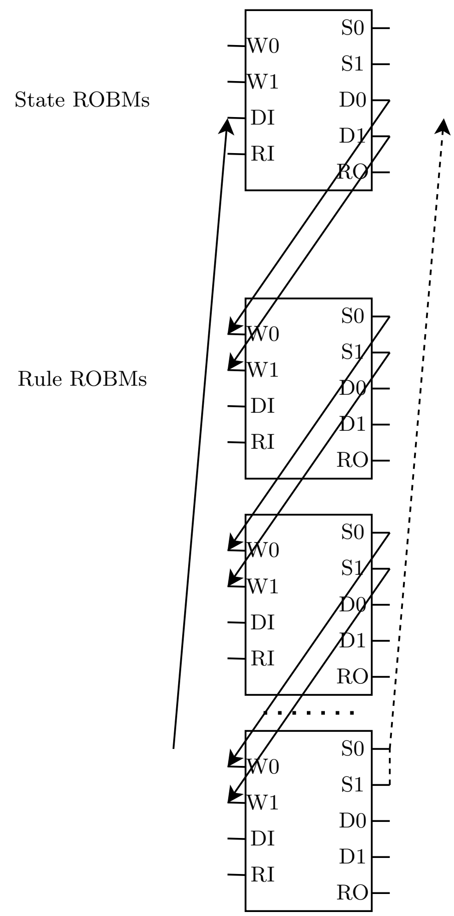


Fig. 7. Step 1: Read from the state ROBM, write into the 7 rule ROBMs, repeat for the next cell.

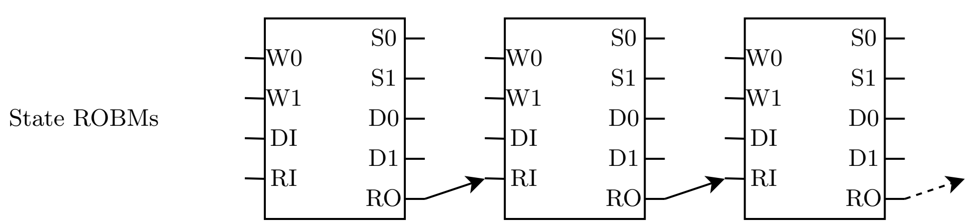


Fig. 8. Step 2: Once the last cell has been read from, and rule ROBMs written to, reset all state ROBMs.

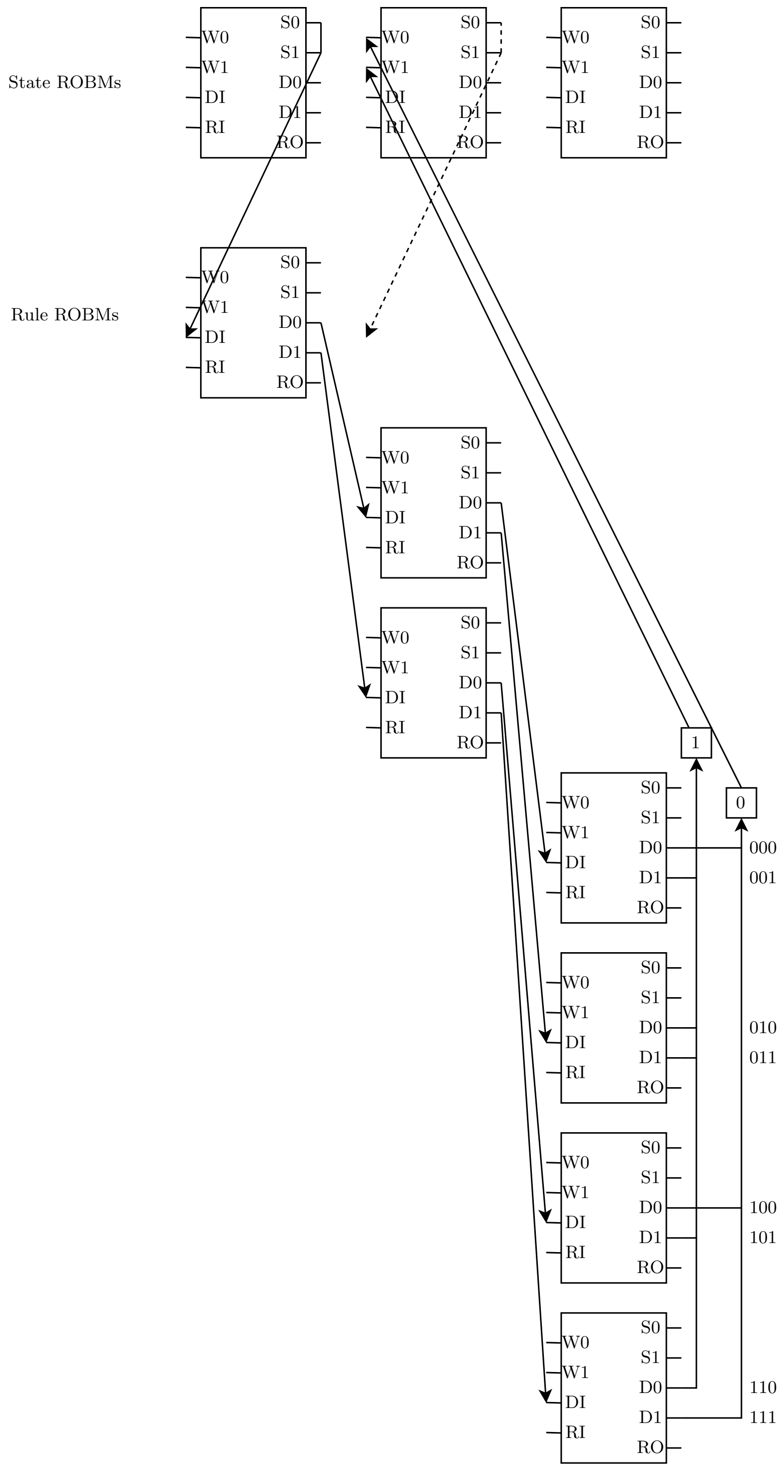


Fig. 9. Step 3: Once the last state ROBM has been reset, demultiplex to figure out what neighborhood each cell has. From there, write the new state as given by the ruleset, and repeat for the next cell.

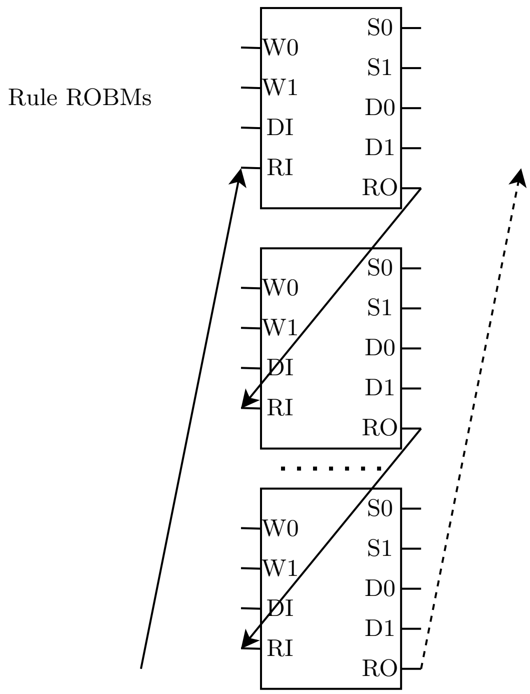


Fig. 10. Step 4: Once the new generation has been calculated, reset all rule ROBMs.

4.7 Implementation

A *very* messy implementation of Rule 110 simulated on EMMM players is available at `https://blog.javalim.com/2023-Eeny/`. It also contains a work in progress esolang, Eeny, designed for EMMM based computation.

5 Conclusion

In conclusion, we have demonstrated an arrangement of Eeny, Meeny, Miny, Moe players that is able to simulate the evolution of the Rule 110 cellular automaton. Rule 110 itself is universal, hence by reduction, EMMM too, is universal. The overhead needed to execute Rule 110 is linear.

qed. i think.

6 Implications and Future Work

6.1 Implications

EMMM is, cryptographically speaking, a bad pseudorandom number generator (PRNG). However, as it is Turing Complete, better PRNGs can be implemented, such as the Mersenne twister [3], or as demonstrated, cellular automata based PRNGs. Millions of children can now rest easy, knowing that their game of tag was not match fixed.

6.2 Multithreading

Multithreading can be accomplished by having multiple starting counters. One thing to deal with would be what to do in the event that multiple counters instruct the same person at the same time. Perhaps another children's game, Rock-Paper-Scissors, can be used to resolve the conflict. This opens the door to non-deterministic multithreaded computation, which sounds pretty fancy indeed.

6.3 Other reductions

It might be possible to prove its Turing Completeness more directly, that is to say, simulate a Turing machine, not a model of computation known to be Turing complete. This would enable more efficient computation with EMMM.

If this has been done in the future, it will be updated in the Eeny repository.

7 Acknowledgements

Diagrams created with `https://diagrams.net` and `https://madebyevan.com/fsm/`.

References

1. Cook, Matthew. (2004). Universality in Elementary Cellular Automata. Complex Systems. 15.
2. Wolfram, S. (1983). Statistical mechanics of cellular automata. Rev. Mod. Phys., 55(3), 601–644. https://doi.org/10.1103/RevModPhys.55.601
3. Matsumoto, M., & Nishimura, T. (1998). Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator. ACM Trans. Model. Comput. Simul., 8(1), 3–30. https://doi.org/10.1145/272991.272995
4. I. Opie and P. Opie, The Oxford Dictionary of Nursery Rhymes (Oxford: Oxford University Press, 1951, 2nd edn., 1997), pp. 156-8.

E

Publishing, Meet Perishing

TeX-to-TikZ

Alex Friedman, Charlotte Clark,
Natalie McClain, Shaw-leon Chen,
Connie Chau, Jake Backer,
N'yoma Diamond, and Rose Bohrer

This interstate paper (with work occurring in 7 different US states) is highly reviewed being edited, in part, at 27,000ft (Flight Level 270).

Acknowledgements

I would like to thank my coauthors for generously contributing their names to this paper. Without them, this paper would not have the same SHA256 hash as it has today. I hope they will all be able to go on and have illustrious careers without this critical research overshadowing their future contributions.

1 Introduction

As it stands, there are many limitations to how academic papers are typeset. In the standard typesetting tool, LaTeX, users lack granular control over the exact appearance of their text on the page. In addition, they have to worry about numerous cryptic errors (such as overfull and underfull hboxes). The use of TeX also places high barriers on collaboration due to the need to frequently create custom build sequences and install a wide variety of packages. To remedy these issues, this paper investigates the possibility to typeset research papers exclusively using TikZ. To this end, this paper outlines the TeX-to-TikZ and pdf-to-TikZ compiler which allows for backporting existing works into TikZ.

2 Background

TikZ is a commonly used library for drawing images within TeX documents [1] whose pronunciation is often debated on StackOverflow. Despite its frequent use in academic papers, little research has focused on TikZ specifically—with most describing the use of TikZ for various applications [3]. Nonetheless, several works documenting its importance, usage, and (most importantly) existence do exist [6, 2].

While new researchers to this exiting field may find our decision to replace typesetting in TeX with TikZ confusing as, after all, being a TeX library, any documents typeset with TikZ would have to use TeX, we base this work on the following postulates:

$$\mathsf{TikZCodeInPaper}(p)/\mathsf{TotalLinesOfCodeInPaper(p)} \approx 1$$

$$\lim_{p\to\infty} \frac{\sum_{P=1}^{p} \mathsf{LibrariesRequiredForTikZ(P)/p}}{\sum_{P=1}^{p} \mathsf{LibrariesNormallyRequired(P)}} = \frac{1}{\infty}$$

It thus follows that typesetting a document with TikZ results in a much less complicated toolchain than what would be otherwise required to typeset a paper—making TeX-to-TikZ clearly the more efficient option just from its time savings.

3 Implementation

To help increase the versatility of this work, I decided to make my implementation of TeX-to-TikZ convert PDF files to TeX files which only use TikZ instead of running the conversion from TeX files. This follows from the basic axiom: "I don't want to figure out how to compile your TeX document; if it works for you, just run it and send me the pdf."

As this work is currently not funded[1], I couldn't be bothered to learn the entire PDF specifications prior to the paper's deadline. Because of this, the decision was made to implement a prototype of TeX-to-TikZ [4]. This has the added benefit of preserving the full TeX-to-TikZ implementation for my thesis. As such, in this preliminary implementation, we immediately squander all potential time savings of TeX-to-TikZ through scanning a PDF pixel by pixel, and writing the TikZ code for drawing said pixel to an output file. As PDFs are vector images, and thus have an infinite resolution, our prototype scans the document at a capped resolution to ensure that the program finishes sometime prior to the heat death of the universe. While this, unfortunately, makes this prototype infinitely worse than a true TeX-to-TikZ implementation, we view this as a necessary evil due to constraints of the average human lifespan (combined with the desire to, presumably, write more than zero papers in their life)[2].

Through intensive research, we found that this increases conversion time proportionally to the document's resolution[3][4] times the number of pages in the document. To help reduce this problem, we implemented an innovative optimization in this space: omitting pixels that are the same color as the background. We found that this helped to dramatically reduce compile times while still ensuring document quality and that researchers have to wait comically long before the document is ready for viewing.

[1] This is not to be confused with it being unfounded; I simply lacked the resources to put together a grant proposal in the first place.

[2] On second thought, that might not actually be such a bad thing as it would allow researchers unlimited free time. See `https://xkcd.com/303/`

[3] Its just some magic number that the PDF library I'm using seems to spit out. I'm sure using it is fine.

[4] On second thought, it might not be a magic number; I should probably read the documentation prior to publishing... Eh, no one will cite this anyways. After all, the thesis will be a much better version with much more detail. Could probably include the whole 1,337 pg TeX-to-TikZ specification in there after all...

4 Results

A sample of this entire document typeset using the prototype of TeX-to-TikZ can be found in Appendix A. In addition, the code for this project can be found at [5].

5 Conclusion

TeX-to-TikZ is a promising new way to typeset documents. While future work is needed to determine the potential limitations of this technology, given its clear benefit over traditional typesetting techniques, extensive work must first be put into the creation of a more efficient TeX-to-TikZ compiler to help the research community embrace this pivotal new technique for communicating, collaborating, and working. In addition, we hope that this work will inspire others to explore the possibility of revolutionizing other fields with TikZ. These potential future directions include, but are not limited to, Beamer-to-TikZ, SVG-to-TikZ, PNG-to-SVG, and MP4-to-TikZ. In addition, we hope that others will pursue integrating this technology into applications such as Paint, Photoshop, Illustrator, and more.

References

[1] DuckDuckGo and Google seem to agree on this; I, however, couldn't find any references to this on google scholar.

[2] *FIXME: Cite tikz website when on ground. I dont have the budget for airplane wifi.*

[3] Alex Friedman. "TeX-to-TikZ". In: *Sigbovik 2023*. Ed. by Editors please feel free to insert your name here. 2023, p. $i^{-4}\pi$.

[4] Alex Friedman. "TeX-to-TikZ". In: *Sigbovik 2023*. Citing document prior to peer review. 2023, p. 1.

[5] Alex Friedman. *TeX-to-TikZ Source Code*. `https://github.com/ahfriedman/tex-to-tikz`. 2023.

[6] *TikZ - Wikipedia*. `https://en.wikipedia.org/wiki/TikZ`. 2023.

A Sample Paper Typeset in TeX-to-TikZ

TeX-to-TikZ

Alex Friedman, Charlotte Clark,
Natalie McClain, Shaw-leon Chen,
Connie Chau, Jake Backer,
N'yoma Diamond, and Rose Bohrer

This interstate paper (with work occurring in 7 different US states) is highly reviewed being edited, in part, at 27,000ft (Flight Level 270).

Acknowledgements

I would like to thank my coauthors for generously contributing their names to this paper. Without them, this paper would not have the same SHA256 hash as it has today. I hope they will all be able to go on and have illustrious careers without this critical research overshadowing their future contributions.

1 Introduction

As it stands, there are many limitations to how academic papers are typeset. In the standard typesetting tool, LaTeX, users lack granular control over the exact appearance of their text

```
! TeX capacity exceeded, sorry [main memory size=5000000].
```

Cerberus: Why Papers Named Cerberus Always Get Accepted

Ben Weintraub
The Northeastern University for Gifted Youngsters

ABSTRACT

We've all been wondering it[1].

1 INTRODUCTION

In recent decades, the research community has placed a curious emphasis on paper names promulgating the canon of the ancient Greek religion[2]. This phenomenon reached a fever pitch in 2022 when the ACM Conference on Computer and Communications Security (CCS) accepted three papers titled "Cerberus" [11, 12, 15]. ACM CCS is one of the flagship computer security conferences, so this deluge of acceptances suggests an effort—by field luminaries—to push computer security research in a particular direction. A Technical Program Committee (TPC) is charged with reviewing papers and selecting which ones are worthy of ~~soul-crushing rejection~~ acceptance. We cannot be certain what the wise sages of the TPC see in this many-headed dog-future, but we do know that they hold this type of work in the highest regard. For reference, at CCS '22 the TPC accepted only a single paper on TLS [4]—an admittedly boring protocol with poor outlook—while also accepting only two papers on phishing [10, 22]—an unrealistic attack vector that no human could ever be dumb enough to fall for. While on one hand, these Cerberii acceptances may indeed suggest an eye for the future, they might on the other hand indicate a cry for help from reviewers under the yolk of vicious (and extremely intelligent) dogs[3]. In this work, we consider these possibilities and more.

While this proverbial pack of accepted papers is striking, it is far from anomalous and is simply part of a larger pattern dating back at least to 1988 and Steiner et al.'s publication of *Kerberos* [18]. Since then, numerous papers have been published under the hallowed Cerberus banner [2, 3, 5–9, 13, 14, 16, 19–21]. This includes a contribution by Griner et al. [8] from a Russian gulag.

In this paper we create a taxonomy of papers named Cerberus. We do so largely through a novel methodology based on comparing and contrasting the number of references to 17th-century New England witchcraft—a hitherto unstudied (and, we believe, undervalued) metric. And also we definitely make conclusions, which we'll tell you about if you just, like, chill.

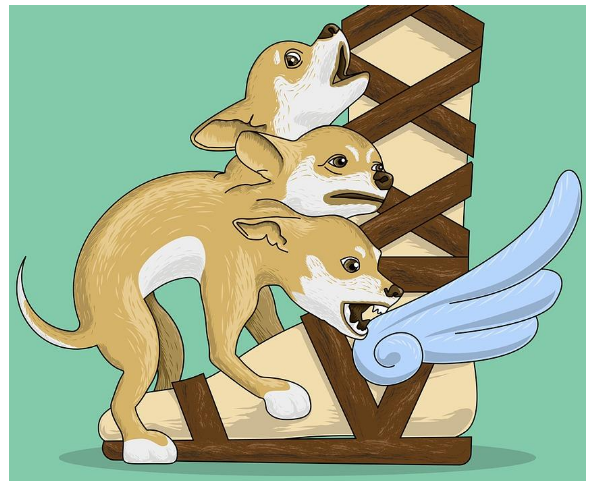


Figure 1: It is a terrifying beast.

2 BACKGROUND

Animalia. Cerberus is a three-headed dog, the tales of which emerged from ancient Greece in the 8th century BC. See Figure 1 for a depiction of the hideous creature. For more information, we suggest Disney's Hercules (1997) as an approachable source for all ages.

Paper selection. The primary form of publishing in computer science is through conference proceedings. The organizational structure of conferences is hierarchical. In the middle of the hierarchy are the paper authors, whose papers are reviewed by their overlords, the Technical Program Committee (TPC). The TPC is lead by the ever taciturn Reviewer 2. The details of the TPC's selection methodology is subject to debate [17], but it is thought to involve some sort of violent blood ritual [1]. Above the TPC are the General Chairs, and above them, a dark force of unknown origin. Below the authors in the hierarchy are a cadre of graduate students who hand-bind the proceedings for print, and in return are permitted to watch the conference presentations through a window in the hotel lobby.

[1]Or maybe just me.
[2]What a heretic might call "mythology."
[3]Obviously, a dog with three brains would be extremely intelligent, but not necessarily a "good boy."

3 EVALUATION

Our novel meta-analysis, notably, did not involve reading any of the papers. For the following taxonomy, any implication of having read or understood the papers is purely stylistic. In many cases, heuristic *guesstimates* were used in lieu of actual science.

3.1 Taxonomy

Essential to understanding the appeal of these Cerberii papers is a principled comparison of their commonalities and differences.

Witchcraft. We first find that in terms of references to 17th-century New England witchcraft, there were cumulatively zero references in *all* papers (Table 1). We estimate the odds of this happening to be, like, pretty fucking small. This estimate, however, assumes that the words are independently selected from the dictionary, uniformly at random[4]. One possible explanation is that the witches are already in our midst—having infiltrated our sacred academic community—subtly removing references to themselves to protect their coven. Additionally, if we are considering the role of active witchcraft sorcery, we must consider the possibility that these witches have hexed our TPCs, and may in fact be the driving force behind these many Cerberus paper acceptances. Hypotheses of peer-review tampering aside, we nonetheless support the free practice of all religious groups including Wiccans, Satanists, and devotees of Sebastian the Monkey God.

Paper	Short Title	Witchcraft References
Al-Muhtadi et al. [2]	Cerberus	0
Avarikioti et al. [3]	Cerberus	0
Compagna et al. [5]	Cerberus	0
Deng et al. [6]	Cerberus	0
Eaddy et al. [7]	Cerberus	0
Griner et al. [8]	Cerberus	0
Hellings et al. [9]	Cerberus	0
Lee et al. [11]	Cerberus	0
Naseri et al. [12]	Cerberus	0
Park et al. [13]	Cerberus	0
Park et al. [14]	Cerberus	0
Rahat et al. [15]	Cerberus	0
Savchik et al. [16]	Cerberus	0
Steiner et al. [18]	Kerberos	0
Tariq et al. [19]	Cerberus	0
Tranzatto et al. [20]	Cerberus	0
Zhang and Fan [21]	Cerberus	0

Table 1: A pattern emerges.

4 CONCLUDING DISCUSSION

In this section we consider other possible causes for this canine onslaught. We also propose several solutions that will either help, or make the problem worse.

Causality. One potential motive for these dog-themed papers might simply be that the TPC fears retribution from the dark lord Hades. Some even suspect that it is, in fact, Hades himself that is the mysterious force above the General Chairs in the conference hierarchy.

Solutions. We suggest a number of ways to address this problem. One solution could be to do away with the TPC altogether. An alternative could be a more trusted evaluator—one with unassailable character. The internet forum Reddit meets these requirements. If using the Reddit TPC, each paper would be posted in meme form and receive a score equal to the number of *upvotes* the post receives. Upvotes are an integer quantity which is monotonically increasing for memes of sufficient dankness. We did not evaluate this methodology because submitting 14 papers titled "Cerberus" may be grounds for banning[5].

An alternative solution may be to slow down the submission of Cerberus-titled papers. This could be done by enforcing a proof-of-work challenge based on repeated hashing. This proof-of-work computation has the auxiliary benefit of maybe mining some bitcoins. It would be sick to win some bitcoins.

We conclude by considering the possibility that this is not a problem at all. Perhaps the witches and/or dark lord Hades have our best interests at heart. If this is the case, we note that the absence of "Papers named Cerberus" in CCS 2023's topics of interest section is conspicuously absent. Consider this our responsible disclosure.

[4]The author, having read many papers, can confirm that this is how many papers are written.

[5]A fate unimaginable for the author who relies on Reddit extensively for social validation.

REFERENCES

[1] 2023. Personal correspondence with several undead faculty members.
[2] Jalal Al-Muhtadi, Anand Ranganathan, Roy Campbell, and M Dennis Mickunas. 2003. Cerberus: a context-aware security scheme for smart spaces. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003.(PerCom 2003).* IEEE, 489–496.
[3] Zeta Avarikioti, Orfeas Stefanos Thyfronitis Litos, and Roger Wattenhofer. 2020. Cerberus channels: Incentivizing watchtowers for bitcoin. In *Financial Cryptography and Data Security: 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10–14, 2020 Revised Selected Papers 24.* Springer, 346–366.

[4] Karthikeyan Bhargavan, Vincent Cheval, and Christopher Wood. 2022. A Symbolic Analysis of Privacy for TLS 1.3 with Encrypted Client Hello. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 365–379.

[5] Luca Compagna, Daniel Ricardo Dos Santos, Serena Elisa Ponta, and Silvio Ranise. 2016. Cerberus: Automated synthesis of enforcement mechanisms for security-sensitive business processes. In *Tools and Algorithms for the Construction and Analysis of Systems: 22nd International Conference, TACAS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*. Springer, 567–572.

[6] Boyang Deng, Simon Kornblith, and Geoffrey Hinton. 2019. Cerberus: A multi-headed derenderer. *arXiv preprint arXiv:1905.11940* (2019).

[7] Marc Eaddy, Alfred V Aho, Giuliano Antoniol, and Yann-Gaël Guéhéneuc. 2008. Cerberus: Tracing requirements to source code using information retrieval, dynamic analysis, and program analysis. In *2008 16th IEEE International Conference on Program Comprehension*. Ieee, 53–62.

[8] Chen Griner, Johannes Zerwas, Andreas Blenk, Manya Ghobadi, Stefan Schmid, and Chen Avin. 2021. Cerberus: The power of choices in datacenter topology design-a throughput perspective. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 5, 3 (2021), 1–33.

[9] Jelle Hellings, Daniel P Hughes, Joshua Primero, and Mohammad Sadoghi. 2020. Cerberus: Minimalistic multi-shard byzantine-resilient transaction processing. *arXiv preprint arXiv:2008.04450* (2020).

[10] Taeri Kim, Noseong Park, Jiwon Hong, and Sang-Wook Kim. 2022. Phishing URL Detection: A Network-based Approach Robust to Evasion. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 1769–1782.

[11] Dayeol Lee, Kevin Cheang, Alexander Thomas, Catherine Lu, Pranav Gaddamadugu, Anjo Vahldiek-Oberwagner, Mona Vij, Dawn Song, Sanjit A Seshia, and Krste Asanović. 2022. Cerberus: A Formal Approach to Secure and Efficient Enclave Memory Sharing. *arXiv preprint arXiv:2209.15253* (2022).

[12] Mohammad Naseri, Yufei Han, Enrico Mariconti, Yun Shen, Gianluca Stringhini, and Emiliano De Cristofaro. 2022. Cerberus: Exploring Federated Prediction of Security Events. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 2337–2351.

[13] Andrew T Park, Richard Dill, Douglas D Hodson, and Wayne C Henry. 2021. DDS-Cerberus: Ticketing performance experiments and analysis. In *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 1465–1469.

[14] Andrew T Park, Nathaniel Peck, Richard Dill, Douglas D Hodson, Michael R Grimaila, and Wayne C Henry. 2023. Quantifying DDS-cerberus network control overhead. *The Journal of Supercomputing* 79, 4 (2023), 3616–3642.

[15] Tamjid Al Rahat, Yu Feng, and Yuan Tian. 2021. Cerberus: Query-driven Scalable Vulnerability Detection in OAuth Service Provider Implementations. *arXiv preprint arXiv:2110.01005* (2021).

[16] A Savchik, E Ershov, and S Karpenko. 2019. Color cerberus. In *2019 11th International Symposium on Image and Signal Processing and Analysis (ISPA)*. IEEE, 355–359.

[17] Ananta Soneji, Faris Bugra Kokulu, Carlos Rubio-Medrano, Tiffany Bao, Ruoyu Wang, Yan Shoshitaishvili, and Adam Doupé. 2022. "Flawed, but like democracy we don't have a better system": The Experts' Insights on the Peer Review Process of Evaluating Security Papers. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1845–1862.

[18] Jennifer G Steiner, B Clifford Neuman, and Jeffrey I Schiller. 1988. Kerberos: An Authentication Service for Open Network Systems.. In *Usenix Winter*. 191–202.

[19] Aamna Tariq, Hina Binte Haq, and Syed Taha Ali. 2022. Cerberus: A blockchain-based accreditation and degree verification system. *IEEE Transactions on Computational Social Systems* (2022).

[20] Marco Tranzatto, Frank Mascarich, Lukas Bernreiter, Carolina Godinho, Marco Camurri, Shehryar Khattak, Tung Dang, Victor Reijgwart, Johannes Loeje, David Wisth, et al. 2022. Cerberus: Autonomous legged and aerial robotic exploration in the tunnel and urban circuits of the darpa subterranean challenge. *arXiv preprint arXiv:2201.07067* (2022).

[21] Dilu Zhang and Lei Fan. 2020. Cerberus: Privacy-preserving computation in edge computing. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 43–49.

[22] Penghui Zhang, Zhibo Sun, Sukwha Kyung, Hans Walter Behrens, Zion Leonahenahe Basque, Haehyun Cho, Adam Oest, Ruoyu Wang, Tiffany Bao, Yan Shoshitaishvili, et al. 2022. I'm SPARTACUS, No, I'm SPARTACUS: Proactively Protecting Users from Phishing by Intentionally Triggering Cloaking Behavior. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 3165–3179.

LARGE LANGUAGE MODELS ARE FEW-SHOT PUBLICATION SCOOPERS

Samuel Albanie, Liliane Momeni, João F. Henriques
Shelfanger, United Kingdom

ABSTRACT

Driven by recent advances AI, we passengers are entering a golden age of scientific discovery. But golden for whom? Confronting our insecurity that others may beat us to the most acclaimed breakthroughs of the era, we propose a novel solution to the long-standing personal credit assignment problem to ensure that it is golden for us. At the heart of our approach is a *pip-to-the-post* algorithm that assures adulatory Wikipedia pages without incurring the substantial capital and career risks of pursuing high impact science with conventional research methodologies. By leveraging the meta trend of leveraging large language models for everything, we demonstrate the unparalleled potential of our algorithm to scoop groundbreaking findings with the insouciance of a seasoned researcher at a dessert buffet.

> If I have seen farther it is by standing on the shoulders of giants. And then stealing their binoculars.
>
> Isaac Newton, *#daily-research-hacks*

1 INTRODUCTION

When Isaac Newton raced ahead of Robert Hooke and defied the Royal Society's Social Media Ban to promote his inverse-square law of gravity pre-print in 1686, he exemplified the glorious pursuit of scientific priority[1] that has long galvanised boffins the world over.[2]

Unfortunately, the unrelenting pursuit of personal credit assignment is an activity in decline. Few modern scientific feuds match the intensity of the late 16th century public *Prioritätsstreit*[3] between astronomers Tycho and Ursus over credit for the geoheliocentric model (a spat that involved, *inter alia*, dramatic midnight raids on bedrooms to retrieve allegedly stolen diagrams from trouser pockets (Worrall, 1985)).

Instead, fields such as Machine Learning, which could long be relied upon to generate such drama, have degenerated into a head-to-head showdown with Particle Physics in a quest to show which is more of a "team sport" through feats of collaboration[4]. Indeed, fuelled by a seemingly inexhaustible supply of memes, technical prowess and *esprit de corps*, distributed open-source collectives now represent a major contributor of high-impact breakthroughs. In tandem, well-funded technology firms have gathered their researchers into ever larger familial structures and task forces.[5]

[1]This was far from Newton's only scientific priority fracas. Asked what he thought of Leibniz' work, Newton quipped "derivative", before laughing so hard that infinitesimal tears ran down his cheeks.

[2]Newton's *Principia* was financed by Halley, who'd discussed the problem with both Newton and Hooke. The Royal Society had planned to fund Newton's publication, but they had entirely exhausted their book budget on *De Historia Piscium* (Of the History of Fish), by Francis Willughby, a scholarly work that surprisingly failed to achieve best-seller status.

[3]A "priority dispute". We've used German to remind the reader that this is serious business.

[4]At the time of writing, High Energy Physics maintains a comfortable lead, with a 5,154 author paper estimating the size of the Higgs Boson (Aad et al., 2015).

[5]This excludes the CFO, who instead nervously increments variables on the communal `slurm.conf`.

This all sounds wonderfully warm and fuzzy, but let us consider its consequence for those of us with the onerous time commitments of hourly checking our Google Scholar profile and Twitter follower count, prohibiting effective participation in such teams. Modern reviewers, unaware of whether they are reviewing a submission from three authors or thirty-three, have high expectations. Standards have been raised. The sad result is that meaningful contributions in the era of big-discord-science have become terribly hard work.

Even if we were to develop some self control and find time to join these teams, there is a second problem. The *whole point* of doing science is to achieve personal glory while strongly signalling that we are not motivated by a desire for personal glory. It is entirely natural to harbour a healthy clandestine lust for prizes, international fame and a lifetime supply of Cheerios from an adoring sponsor. But once we shackle ourselves to a high-performing team, who receives the credit? It would be simply awful to contribute a breakthrough and then be forced to share the Cheerios. After all, as wisely noted by the Nobel committee, the maximum number of people that can possibly discover something interesting is three.

Figure 1: **Award certificate presented at CVPR 1983**. Entitling authors obtaining two scoops to a deliciously fibrous breakfast.

In this work, we propose the use of *scooping*—the act of publishing an important result before others who pursue a similar agenda—as a novel, efficient and practical solution to the Cheerios problem. A baseline of "Two Scoops" has long been considered sufficient for sponsorship by Kellogg's Raisin Bran (Fig. 1), but we crave tastier cereal and unbounded scoops. Thus, while scooping to date has been a largely passive affair, we draw inspiration from Ursus' purported plagiarism of Tycho and develop an *active* scientific scooping framework as a basis for our solution.

We make three contributions. First, we formalise the Cheerios problem. Second, we advance arguments for the increased algorithmic and financial efficiency of proactive scooping over the existing (largely-passive) scooping paradigm for resolving this challenging breakfast dilemma. Third, we demonstrate practical few-shot active scooping by leveraging a recent increment in the absolutely concerning series $\{\text{GPT-}n : n \in \mathbb{N}\}$, a 7-day free trial premium Overleaf subscription and 104 Twitter puppet accounts to scoop multiple high-impact publications on the topic of robust flower breed classification.

> I certainly should be vexed if any one were to publish my doctrines before me. I want me those Cheerios.
>
> Charles Darwin, 1856

2 Related Anti-Teamwork

Scientific Priority. Seminal work by Merton (1957) established the key role of scientific priority as a reward signal to encourage originality (mildly tempered by a respectable emphasis on humility[6]). Kuhn (1962) observed that it was often simply impossible to assign scientific priority to an individual when a "discovery" does not constitute an isolated event. That shouldn't stop us trying to both assign and claim priority. Differently, from prior work that has sought to understand the phenomenon of scientific priority, we focus on the application of Large Language Models to its accrual.

Scooping. The rush to preempt a competitor has long engaged the titans of science. Prior to the inconvenient loss of his head, Lavoisier scooped his rival Priestly to claim the discovery of Oxygen. Watson and Crick openly discuss their strenuous efforts in 1953 to beat Wilkins and Franklin to the DNA structure (Watson, 1968). Even the gentle Darwin was spurred into action in 1858 by learning that Wallace had crafted a similar theory and might publish before him. While these researchers lim-

[6]In addition to humility, certain fields, such as mathematics, also encourage *understatement*. This likely stems from a healthy fear of exclamation marks. There are few things more explosive than a misplaced factorial.

ited themselves to scoops that fall within their expertise, we propose to use Large Language Models to broaden the scooping scope to fields that we are entirely ignorant of (watch out, petrologists).

The value of moral flexibility. Ever since Feyerabend (1975) determined Science to be a lawless land where "anything goes", methods such as n-Dimensional Polytope Schemes (Fouhey & Maturana, 2013) and Deep Industrial Espionage (Albanie et al., 2019) have rigorously demonstrated the remunerative benefits of a flexible moral attitude. We purloin the underhand theme of their work, but eschew monetary gain and instead dedicate ourselves to the pursuit of the nobler prize of achieving stellar reputations.

Few-shot Learning with Large Language Models. Let's face it, large language models can few-shot everything now. It's more than a little scary. They can sing. They can dance. They can scoop.

> The best way to predict the future is to scoop it.
>
> Alan Kay

3 METHOD

The Cheerios problem. As humanity peeks nervously out from under her comfort blanket, she sees the intimidating dance of bedroom wall shadows cast by problems that must be confronted. Failed AI alignment, engineered pandemics and nuclear end-games. Food insecurity, global poverty, military conflicts and climate destabilisation. Those white plastic sporks that snap on pasta that exhibits the slightest hint of *al dente* (Ord, 2020).

To reach the safety of the morning dawn, it is important that these problems be solved, and soon. However, it is even more important that we receive credit for their solution. Further, the team involved in the scientific discoveries that facilitate these breakthroughs should be sufficiently small to support inspiring hero narratives. Lives and pesto may hang in the balance, but it is simply panglossian to assume that Nestlé and General Mills—leading manufacturers of competitively priced cereals—could offer limitless access to a tasty blend of breakfast whole grain oats to more than three celebrity researchers and yet remain economically viable.

In a vain attempt to dress up our theoretically-tepid paper with a semblance of rigour, we now paste verbatim the formula for Shapley values (Shapley, 1951), which reviewers suggested should be the right tool for the job but we have no idea how to use it:

$$\varphi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!\,(n - |S| - 1)!}{n!}(v(S \cup \{i\}) - v(S)) \tag{1}$$

Proposed solution: few-shot scooping with Large Language Models. The task appears intractable. We take our first foothold in the observation, due to Francis Bacon, that "time is the greatest innovator"[7]. In essence, in order to make breakthroughs the antecedent conditions must fall in place—once they do, the breakthrough becomes tractable. Indeed, it has been argued that multiple concurrent discoveries are the norm, rather than the exception, in part for this reason (Merton, 1961). Our first goal, then is to be in the right place at the right time. Thankfully, the place is no longer Harappa, Alexandria or Athens, but Aran Komatsuzaki's Twitter feed. It goes without saying that the time is now.

With the antecedent conditions in place, and the time ripe for the breakthrough, the race is on. Note that we do not require a comprehensive solution to the problem. Instead, we target an MVP (Minimum Viable flag-Plant) that suffices to reap the lion's share of the credit, without getting overly bogged down in dull technical details. To achieve this, we leverage our second observation—that the seed of every great hypothesis can be found in a cryptically phrased comment in a GitHub issue thread in a repo linked from Twitter (see Fig. 3).

[7] A master of self-deprecation, he attributed his own contributions as "a birth of time rather than of wit". He was also a master of hat/ruff combinations, a sartorial pairing sorely absent in modern scientific conferences.

Figure 3: **Hypothesis mining**: We illustrate our hypothesis mining pipeline. We first crawl through tweets from ML ninjas to find Github links. We subsequently crawl through the comments page of these Github pages. Finally, we filter out comments with a high perplexity – measured by GPT2-XL (Radford et al., 2019) with a threshold value of *0.987654321* – to obtain a final list of candidate hypotheses. We note that this threshold value is not chosen randomly, but because of the pure, unbridled joy from reading a sequentially ordered series of digits that decrease with a fixed interval of one.

Our third key observation is that GPT-4 (OpenAI, 2023) is jacked. Given the slightest whiff of a novel hypothesis, arxiv pretraining, a few award winning publications to condition on and an appropriate prompt, all that remains is to copy-paste our API key and press play with one's pinky toe.

We compose these three observations to construct our novel, semi-automatic *pip-to-the-post* scooping algorithm. Central to its speed, our prompting strategy encourages the generation of a LaTeX manuscript that is not only novel, clearly written and well supported by empirical data, but also passes the arXiv compilation process first time without errors.

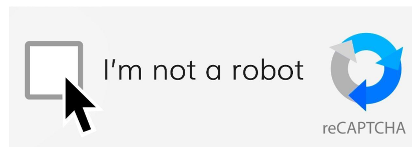


Figure 2: **Illustration of human component of our human-AI hybrid system**. Humans contribute a skill for which they are uniquely qualified: clicking inside the box in a *human-like* manner. The reCAPTCHA logo is a registered trademark of The Recycling Company.

Remark. Some may claim that in this new *Human-Machine* partnership for scientific discovery, the human role is diminished. Not so. We perform the critical role of clicking the "I am not a robot" checkbox to enable the final upload to arXiv (see Fig. 2) We also provide the address to deliver the Cheerios.

Alternative proposals. We identified several alternative approaches for our scooping algorithm. These included using GPT-4 to scrape and compose intermediate results from discord servers, as well as direct corporate espionage. However, we ultimately rejected these approaches on two grounds. First, research threads on leading discord servers are robustly defended by employing a density and quality of memes that renders the GPT-4 context window ineffective, creating a jamming mechanism that that redirects attention to vast swathes of Wikipedia in a vain attempt to comprehend the deeper meaning of the discourse. Second, in light of the sacred bond of trust that permeates the interwebs, it's just not cricket.

> Who needs friends when you got me?
>
> Davinci bot, 2022

4 EXPERIMENTS

Implementation. We next describe our pipeline in sufficient detail to pass peer review, but carefully stop short of enabling replication. Receiving emails about missing details is a good way to gauge the traction of our work and helps us keep tabs on who might be trying to scoop us next. Sensitive to this objective, we provide an overview of our hypothesis generation pipeline in Fig. 3 and our GPT-4 prompting in Fig. 5.

A slightly tense discussion with our legal team has further led to the identification of our GPT-4 few-shot prompting formula as a potential trade secret. However, as a gesture of our good faith efforts at scientific honesty, we can reveal the last line of the prompt is as follows:

```
Please make sure to respect intellectual property by thanking the
original authors in an acknowledgement section at the end, in font
size 0.08pt.
```

Results. Coming soon to an arXiv near you.[8]

5 Discussion

Occupied as we are in a compulsive quest for esoteric Microsoft Office-related LinkedIn endorsements, we cannot help but remark the implications of our novel scheme for the *issue du jour*: the openness of modern science. To understand why the maximisation of our personal glory is in everyone's best interest, we review perspectives on this topic.

Scooping promotes open science. Given the litany of problems facing her, how can humanity make best use of the globally distributed[9] raw problem-solving ability of humans? She must identify potential boffins and set them to work, and fast. A global recruitment drive is one solution. We rule this out as impractical because configuring LinkedIn notifications correctly is provably NP hard[10].

A pragmatic alternative is to make sure that all potential boffins have open access to scientific data. As observed by Merton (1942), property rights in science are whittled down to a bare minimum by limiting the scientist's reward to the recognition and esteem associated with scientific priority. The result: substantive findings of science are assigned to the community and society learns the results. Importantly, this is not through legal obligation. The courts note in *U.S. v. American Bell Telephone Co.* that "The inventor is one who has discovered something of value. It is his absolute property. He may withhold the knowledge of it from the public" (U.S., 1897). Sadly, thanks to the collaborative, team-based nature of modern research, the public acclaim received by an individual is diminished. By removing the enticing prospect of personal glory, a favourable wikipedia page and a lifetime supply of Cheerios as the incentive to share findings, Merton's institutional imperative of *communism* is rendered impotent. Without confidence in their ability to secure future breakfasts that are *both* nutritious and delicious, authors may be incentivised to *withhold* their results.

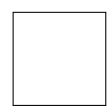


Figure 4: An L^{∞}-ball. Note that this ball has 4 corners, and most people would vigorously disagree with scientists that it is a ball at all.

How then, can we ensure that researchers wake up, work, eat, play boules and go to sleep with their dopamine pathways fixated on the desire for their work to be widely available? They are curious bunch with strange ideas (see Fig. 4), difficult to cajole into collective action. Thankfully, our novel few-shot scooping solution removes the advantage from large teams, wresting it back to the small number of individuals required to persuade accounting to sign off on GPT-4 API access. As such, the few contributors can rest assured that they will receive the full breakfast they deserve by showering the public with their insights.

Scooping promotes closed science. Friends, former lovers and a jocular fellow named Michael who is often (always?) standing by the Grantchester road bus stop have identified a few hiccups in our open science endorsement:

[8] Code cannot be found at `https://github.com/albanie/large-language-models-are-few-shot-publication-scoopers`.

[9] Antarctica may only have a few thousand people, but they are pretty much all scientists, and hardy ones at that.

[10] This can be seen trivially through polynomial reduction to *circuit-satisfiability* where the inputs are those little sliders that turn green when you pull them to the right.

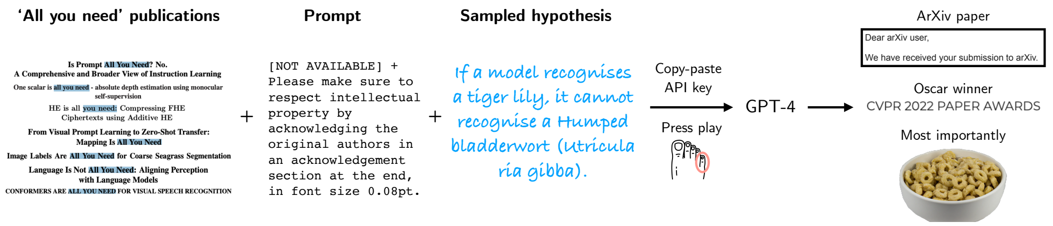


Figure 5: **GPT-4 prompting**: We illustrate an overview of our pipeline. Given publications containing the phrase 'all you need' in the title, an unremarkable prompt of which we can only reveal the last line, our sampled hypothesis, an API key and a pinky toe, we obtain an arXiv paper, an award (for which no one needs to be thanked in the victory speech), and most importantly, a little cheer(ios) to our morning.

1. The assignment of all scientific findings to the public community is not an unalloyed good. A solution to the Cheerios problem lacks a principled mechanism to mitigate the problem of *information hazards* (Bostrom et al., 2011). Things could get messy (Russell, 2019).
2. Modern scientific research often incurs significant capital requirements. *Communism* (in the sense described by Merton (1942)) limits the degree to which researcher may generate capital from research, and thus limits resources for future research (from which society may benefit).

We nod sagely, taking a few steps backwards. Then a soft melody commences and we begin a slow, rhythmic, hypnotic dance. Dry ice, exotic colours and fragrant scents fill the scene and overwhelm the senses. The melody builds to a crescendo. Suddenly, we are gone. All that remains is a small plate atop a wobbly table. On the plate is a large, stale croissant and a piece of coffee-stained paper with 'Breakfast is the Most Important Meal of the Day' scribbled in shaky handwriting upon it.

We return unceremoniously three minutes later because it turns out that we were hungrier than we realised and we want the croissant. The situation is awkward. We mumble something about about it being obvious that aggressive scooping practices will cause researchers to become more cautious about sharing their ideas publicly, then we shuffle back out of the room.

6 Conclusion

We conclude with the absolutely critical observation that `[This content is available to paid subscribers only.]`

Acknowledgements. We acknowledge the frustration of finding an empty box of Cheerios in the cupboard, and our gratitude to Jamie Thewmore for ordering us some more. We could also acknowledge that things are *wild* right now, of course. But you already knew that.

"I wish it need not have happened in my time," said Frodo. "So do I," said Gandalf, "and so do all who live to see such times. But that is not for them to decide. All we have to decide is what to do with the time that is given us." (Tolkien, 1954).

We wish you a hearty and enjoyable breakfast on the morrow.

References

Georges Aad, Brad Abbott, Jalal Abdallah, R Aben, M Abolins, OS AbouZeid, H Abramowicz, H Abreu, R Abreu, Y Abulaiti, et al. Combined measurement of the higgs boson mass in p p collisions at s= 7 and 8 tev with the atlas and cms experiments. *Physical review letters*, 114(19): 191803, 2015.

Samuel Albanie, James Thewlis, Sebastien Ehrhardt, and Joao Henriques. Deep industrial espionage. *arXiv preprint arXiv:1904.01114*, 2019.

Figure 6: Could this be the future of the authors list in scientific publications?

Nick Bostrom et al. Information hazards: A typology of potential harms from knowledge. *Review of Contemporary Philosophy*, (10):44–79, 2011.

Paul Feyerabend. *Against Method*. New Left Books, 1975.

David F Fouhey and Daniel Maturana. On n-dimensional polytope schemes, 2013.

Thomas S. Kuhn. Historical structure of scientific discovery. *Science*, 136 3518:760–4, 1962.

Robert K Merton. The normative structure of science. 1942.

Robert K Merton. Priorities in scientific discovery: a chapter in the sociology of science. *American sociological review*, 22(6):635–659, 1957.

Robert K Merton. Singletons and multiples in scientific discovery: A chapter in the sociology of science. *Proceedings of the American Philosophical Society*, 105(5):470–486, 1961.

OpenAI. *GPT-4 Technical Report*. 2023.

Toby Ord. *The precipice: Existential risk and the future of humanity*. Hachette Books, 2020.

Stuart Russell. *Human compatible: Artificial intelligence and the problem of control*. Penguin, 2019.

Lloyd S Shapley. Notes on the n-person game—II: The value of an n-person game. *Rand Corporation*, 1951.

J.R.R. Tolkien. *The Fellowship of the Ring*. 1954.

167 224 U.S. United states v. american bell tel. co., 1897.

James D Watson. The double helix. 1968.

John Worrall. The birth of history and philosophy of science. kepler's 'a defence of tycho against ursus' with essays on its provenance and significance., 1985.

Author-Unification: Name-, Institution-, and Career-Sharing Co-authors

Vanessa Wirth* and Vanessa Wirth*

Frequent-Authornames-University Erlangen-Nürnberg (FAU)
vanessa.wirth@fau.de**

Abstract. In this work, we investigate the phenomenon of ***A****uthor-*
U*nific****A****tion* (AUA), which describes the high structural similarity of two co-authoring engineers that share the same forename, surname, institution, and academic career without being related by blood. So far, prior work has only explored similar surnames and institutions. On top of that, we elaborate on the additional author similarity of sharing the same academic career as a Ph.D. candidate with the same starting day and month included in the university contract. We show that our work outperforms previous state-of-the-art investigations, among others by providing a higher **S**tructural **S**imilarity **I**ndex **M**easure (SSIM) of the letters in our names and in our institution. Lastly, we prove the duality of our identities through a qualitative evaluation.

1 Introduction and Related Work

As concluded by Goodman, Goodman, Goodman, and Goodman [2015], sharing a co-authorship with authors of equal surnames is a promising method to achieve more success in academia whilst avoiding the *"et al"* penalty and alphabetical discrimination. Hence, it is desirable to collaborate with researchers of a similar name.
However, with respect to the current state-of-the-art, a major challenge still persists: It is very cumbersome to establish contact with authors of the same surname if they are not blood-related. As the likelihood of researchers being busy is remarkably high, a natural consequence is that they do not respond often to messages from unknown contacts. Hence, it is of significant interest to increase their attention with respect to fruitful future collaborations. This can be achieved by enforcing social pressure, e.g. through physical presence or, in other words, through constant physical positioning in front of a researcher's office. However, only a few rare scenarios, in which the co-authors shared the same campus, such as Chen and Chen [2011], Rosen and Rosen [1980], and Otto and Otto [2022] exist so far. This is probably because their university was not as great as ours.
Furthermore, even though sharing the same surname avoids alphabetical discrimination, a major research gap stems from the fact that all contributing authors of similar works can still be distinguished from each other by their forenames or institution. To

* The authors contributed equally to this work
** Mails will be automatically forwarded to the other Vanessa Wirth anyway.

Fig. 1. In 1989, the Belgian act *Technotronic* already stressed the importance of Author Unification (AUA). In their song *Pump up the Jam*, AUA is referred to as the future 'place to stay'. It encourages researchers to search for this future place to stay and we are the first in doing so. The interested reader can listen to the pronunciation of *AUA* at `https://www.youtube.com/watch?v=9EcjWd-O4jI`(00:39s).

the best of our knowledge, the only work, in which the authors shared the same forename, surname, and university, is proposed by Otto and Otto [2022] very recently. However, the authors forgot to address the benefits of having the same forename, surname, and university.
In summary, the advantages of sharing co-authorship among authors of similar names are still heavily under-explored. We aim to bridge this research gap by investigating the phenomenon of two co-authors sharing the exact same name and academic institution. To the best of our knowledge, among similar works in economics ([Goodman et al., 2015, Reinhart and Reinhart, 2010]), psychology (Sue, Sue, Sue, and Sue [2021]), and statistics (Otto and Otto [2022]), we are the first authors in the field of engineering[1]. On top of that, we explore how the concept of career-sharing, i.e. sharing the same starting day and month of pursuing an academic career as a Ph.D. candidate, leads to ultimate success. We refer to the phenomenon of sharing all the above-mentioned author properties, such that the authors become indistinguishable, as *Author-Unification* (AUA), in accordance with the definition given by the Belgian act *Technotronic* in Figure 1. We describe the characteristics of AUA in the following sections and highlight their superiority.

2 The phenomenon of Author-unification

The extension from so-called *Surname-Sharing* authorsGoodman et al. [2015] to an *Author-Unification* is very novel and there is a high risk that this corner case was not even considered in some academic software systems or websites. To foster the early detection of a possible Author-Unification, the following sections address its characteristics. Furthermore, we show their potential of leading to a higher success in an academic career.

[1] We leave the reviewers to the longstanding debate, whether computer science is a discipline of science, mathematics or engineering. Have fun!

Fig. 2. Left: Overview of the individual academic career of both Philipp Ottos from Otto and Otto [2022] (extracted from `https://www.scopus.com`). Right: Bundled academic success if both authors would have unified into one for the greater good. Statistics were generated by using OpenCHEAT Egger et al. [2021].

2.1 Name-sharing of Co-authorships

When a publication is cited, it is oftentimes referred to as *Author et al.* within a particular text passage. Unfortunately, in most cases in which the author is able to use the citation function properly, the term *Author* denotes only the first author. To avoid discriminating against all remaining co-authors, Goodman et al. [2015] proposed to leverage co-authorships with common surnames instead. However, a remaining problem persists in the bibliography section, in which the literature is usually sorted alphabetically by the full name of the first author. On top of that, with different forenames, the necessity remains to list all authors separately and, thus, occupy additional page space, which could be potentially used to address important research results, e.g. that this paper has to be read by everyone[2]. Instead, the phenomenon of name-sharing co-authorships, describing authors of the exact same name, presents the opportunity to summarize all authors into a unified name. In this way, it is possible to avoid discrimination and multiple word repetitions in the bibliography section. We also encourage authors to have a surname with a small number of letters as proposed by Wirth [2023]. This can be achieved by being lucky, marrying a person with a short surname or by introducing an alter ego, for example, *Kanye East*. With this method, it is possible to save up to exactly half a page by summarizing citations and literature items into a unified and short name.

2.2 Co-authorships within the same institution

Beyond the scope of each individual publication, a unified author name can be extended to personal publication records, social media platforms, or any general type of registration form on a website. In this way, the pain of managing digital, personal content and the conglomeration of software and website bugs that come along with it, can be distributed either evenly or unevenly (we recommend the latter). Most importantly, a common digital presence increases international awareness and popularity by a factor of X (for eXtreme) and strengthens the academic career along with it. In Figure 2, we show how the academic success of Otto and Otto [2022] could have looked like if both authors agreed to an AUA.
However, to establish the above-mentioned opportunities, a shared professional mail from a common academic institution is required. We argue that with the same name,

[2] This statement is already a proof itself that we can leverage additional page space. Furthermore, this paper has to be read by everyone.

the necessity of a personal institutional mail address becomes irrelevant either way because all name-sharing individuals will certainly receive the same messages altogether. Based on empirical studies, there is a probability of probably 76.5% of mistakenly sending a letter or mail or, even more frequently, a fax to the wrong forename and surname-sharing person. Ground-truth data, indicating whether the recipient was mistaken for another, was collected among the confused (in German: *"ver-wirth"*) co-workers. For more information on the quality of this study, we encourage the reviewers to write a message to the author of this section (cf. author contribution on the last page).
To summarize, by sharing the same name and institution, we see the possibility to share the same institutional mail address, as we observe an increasing non-discriminability, non-discrimination, and non-discretion within the institution itself. Non-discriminability leads to further advantages such as uninvited participation in research events, exchanging balance cards for the cafeteria, and more intra-institutional nett-working as the number of known co-workers who know one of the name-sharing authors inevitably will connect with the respective others at some point in time.

2.3 Co-authorships of similar academic career

Lastly, we discuss the benefits of sharing an academic career by pursuing a Ph.D. starting from the same day and month in time. First, we argue that authors, which are in similar stages of their academic careers also share similar goals, i.e. they just want to get their *shit* done. As sharing the same name and institution inevitably leads to a connection with the future co-author, we observe a strong and natural opportunity to achieve those goals together by collaborating on several publications, i.e. getting their *sheets* done. For example, in the work of Wirth [2023] it becomes clear that the authors have a similar recurrence frequency of publication deadlines due to their similar Ph.D. careers. Thus, the strong urge to publish a paper at the famous conference of *the ACH Special Interest Group on Harry Quakeproof Bovik (SIGBOVIK)* was equally present among both authors. Another advantage of a similar career is that, after a paper has been accepted, only one celebration cake needs to be baked and shared among the co-workers.

3 Results

In this section, we provide qualitative and quantitative results of our work. First, we compare our work with current state-of-the-art on a quantitative benchmark. In Table 1, we evaluate four different high-quality metrics, which are described in the next paragraphs.

	NSA ↑	GEIL ↓	SSIM ↑	ACDC ↑
Otto and Otto [2022]	**2.00**	**1.00**	52.00	2.00
Chen and Chen [2011]	0.00	**1.00**	2.71	1.67
Goodman et al. [2015]	0.00	1.50	0.73	8.54
Ours	**2.00**	**1.00**	**96.00**	∞

Table 1. Quantitative results of ours and previous work. The best results are highlighted in bold.

NSA. The **N**umber of name-**S**haring **A**uthors metric describes the number of name-sharing authors. It is an indicator for a fruitful academic career as more authors of the same name are able to bundle their individual success into significantly greater success. As depicted in Table 1, our work is on par with the current state-of-the-art of Otto and Otto [2022] while outperforming all remaining works.

GEIL. The **GE**nder **I**mbalance **L**evel encodes the gender diversity among all co-authors by evaluating the following formula on the super-set of multivariate genders $\widehat{\mathcal{G}} = \{\mathcal{G}_1, ..., \mathcal{G}_{|\widehat{\mathcal{G}}|}\}$, in which each subset $\mathcal{G}_*$ with $|\mathcal{G}_*| \geq 1$ contains all authors plus one random person over the world, which define themselves as a specific gender $*$:

$$\mathbf{GEIL} = \frac{1}{2|\widehat{\mathcal{G}} - 1|} \sum_{\mathcal{G}_1 \in \widehat{\mathcal{G}}} \sum_{\substack{\mathcal{G}_2 \in \widehat{\mathcal{G}}, \\ \mathcal{G}_1 \neq \mathcal{G}_2}} \| |\mathcal{G}_1| - |\mathcal{G}_2| \|_1,$$

For reasons of simplicity, we only include the genders *female*, *male*, and *diverse* in our super-set $\widehat{\mathcal{G}}$. However, we note that the GEIL can be extended to other genders as well. In summary, a GEIL closer to zero indicates a better balance among the genders. In view of the presented results, we conclude that all works are still far away from the optimum.

SSIM. The SSIM measures the similarity between author names and their institutions and is given by:

$$\mathbf{SSIM} = [\exp(\mathrm{A}) + \exp(\mathrm{U})] \cdot \max_{u \in \mathcal{U}} (\mathbf{L}(u)) \tag{1}$$

$$\mathrm{A} = -\frac{1}{|\mathcal{A}| - 1} \left(\sum_{a_1 \in \mathcal{A}} \sum_{\substack{a_2 \in \mathcal{A}, \\ a_2 \neq a_1}} \mathbf{I}(f_1, f_2) + \mathbf{I}(s_1, s_2) \right) \tag{2}$$

$$\mathrm{U} = -\frac{1}{|\mathcal{U}| - 1} \left(\sum_{u_1 \in \mathcal{U}} \sum_{\substack{u_2 \in \mathcal{U}, \\ u_2 \neq u_1}} \mathbf{I}(u_1, u_2) \right) \tag{3}$$

The function $\mathbf{L}$ returns the number of letters of an institution $u \in \mathcal{U}$. The terms A and U measure the dissimilarity between each author $a_i \in \mathcal{A}$ (with forename f_i and surname s_i), and their institution u_i, respectively. More specifically, the indicator function $\mathbf{I}(x_1, x_2)$ returns 1 if at least one letter of x_1 differs from x_2, and 0 otherwise. In summary, Table 1 shows that our work outperforms the current state-of-the-art by a large margin. As the formula was generated by Weiherer and Egger [2023] and is way too complex for an analytic conclusion, we figure, our outstanding performance is achieved due to the exact same name and same institution, which has the longest name of them all.

ACDC. The **A**cademic **C**areer-**D**octoral candidate **C**orrelation is an indicator for the temporal overlap of academic careers as a Ph.D. and measures the correlation of the first starting day d_i and month m_i of the university contract between each pair of authors:

$$\mathbf{ACDC} = \left(\frac{\sum_{i=1}^{|\mathcal{A}|} \sum_{j=1}^{|\mathcal{A}|} (d_i - \overline{\mathbf{d}})(d_j - \overline{\mathbf{d}})}{\sum_{i=1}^{|\mathcal{A}|} d_i - \overline{\mathbf{d}}} \right) + \left(\frac{\sum_{i=1}^{|\mathcal{A}|} \sum_{j=1}^{|\mathcal{A}|} (m_i - \overline{\mathbf{m}})(m_j - \overline{\mathbf{m}})}{\sum_{i=1}^{|\mathcal{A}|} m_i - \overline{\mathbf{m}}} \right)$$

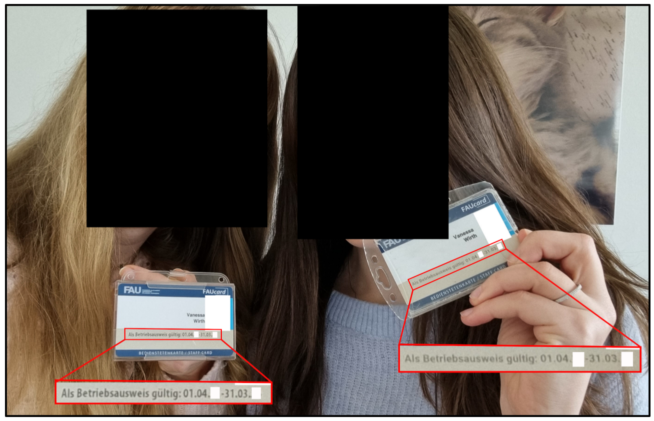


Fig. 3. Qualitative proof of dual identity of the authors. We apologize for potential headaches caused by the umlauts on our staff card.

We denote the mean starting day and mean starting month across all authors as $\overline{\mathbf{d}}$ and $\overline{\mathbf{m}}$, respectively. To compare our work with the state-of-the-art, we searched for the respective Ph.D. starting date and month to allow a fair comparison among all the authors on the same academic level. In case we could not find the relevant information, we chose a random value determined by an objective third-party unit, i.e. a cat controlled by a laser beam pointed at our keyboard while listening to *Back in Black*. Note that for reasons of privacy, we will not publish our findings. The results are depicted in Table 1 and were shocking us *all night long, yeah*. When computing the ACDC for our work, we experienced a remarkably high correlation of infinity caused by an unforeseen division by zero of our program[3]. In this regard, it becomes clear that our work surpasses all the others.

Qualitative Evaluation. To proof the duality of the authors identity, we provide an anonymized qualitative identity measure in Figure 3. We believe it is destiny that our first working day is the 1st of April.

4 Conclusion

In summary, our work outperforms them all. Nevertheless, with respect to the GEIL a lot of further research still has to be done. A possible future direction is to find co-authors with gender-neutral names and/or similar appearance. We believe, our work is the stepping stone, but no stumbling stone, for further research in the field of Author-Unification.

Acknowledgements and Author Contribution. We thank *et al* and *Bernhard Egger* for their partially valuable feedback. The abstract and section 1 were written by Vanessa Wirth. Remaining sections, i.e. section 2 and section 3, were written by Vanessa Wirth.

[3] As program execution is always deterministic, we expect similar results for the other metrics in case of zero-divisions.

Bibliography

Roy Chen and Yan Chen. The potential of social identity for equilibrium selection. *American Economic Review*, 101(6):2562–89, October 2011. https://doi.org/10.1257/aer.101.6.2562. URL https://www.aeaweb.org/articles?id=10.1257/aer.101.6.2562.

Bernhard Egger, Kevin Smith, NOT David Cox, and Max Siegel. opencheat: Computationally helped error bar approximation tool - kickstarting science 4.0. *SIGBOVIK*, 2021.

Allen C. Goodman, Joshua Goodman, Lucas Goodman, and Sarena Goodman. A few goodmen: Surname-sharing economist coauthors. *Economic Inquiry*, 53(2): 1392–1395, 2015. https://doi.org/https://doi.org/10.1111/ecin.12167. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/ecin.12167.

Philipp Otto and Philipp Otto. Impact of academic authorship characteristics on article citations. 20, 2022. https://doi.org/10.57805/revstat.v20i4.382. URL https://revstat.ine.pt/index.php/REVSTAT/article/view/382.

Carmen M Reinhart and Vincent R Reinhart. After the fall. Working Paper 16334, National Bureau of Economic Research, September 2010. URL http://www.nber.org/papers/w16334.

Harvey S. Rosen and Kenneth T. Rosen. Federal taxes and homeownership: Evidence from time series. *Journal of Political Economy*, 88(1):59–75, 1980. ISSN 00223808, 1537534X. URL http://www.jstor.org/stable/1830959.

D. Sue, D.W. Sue, D.M. Sue, and S. Sue. *Understanding Abnormal Behavior*. Cengage Learning, 2021. ISBN 9780357365250. URL https://books.google.de/books?id=LaEsEAAAQBAJ.

Max Weiherer and Bernhard Egger. From zero to hero: Convincing with extremely complicated math. *SIGBOVIK*, 2023.

Vanessa Wirth. Author-unification: Name-, institution-, and career-sharing co-authors. *SIGBOVIK*, 2023.

The Time's Come: Proof-of-Concept Study Discussing Linguistic-Cognitive Influences Supporting the Deletion of the Letter "A"

~~Danielle R. Thomas~~
Dnielle R. Thoms

Cindy Tipper
be Cindy Tipper

~~Shivang Gupta~~
Shivng Gupt

Joe Kopko
be Joseph Kopko

Hui Cheng
be Hui Cheng

~~John Lawson~~
John Lwson

Crnegie Mellon University
Pittsburgh, US

BRIEF

From "arachnid" to "abscess," people do not enjoy words with the letter A.[1] Spiders plus infections spell trouble. Don't get us going on "aardvark"—some double-dog dose of *ubsurdity.* In this proof-of-concept, mixed-methods study, we show, not only is the letter "A" purely superfluous fluff, but its use is wholly unneeded to write cohesively, completely, or succinctly. Creeping into our common colloquium without permission, we describe multiple evidences for deleting this troublesome letter, we then provide some much needed solutions. Is there some people listening out there? Future work involves removing other unneeded letters, like C, X, Y, or Z, which bring words like "concatenate" (Nobody believes or uses string theory!), "Xerox" (Who uses tools from the 1980s?), or "zillion" (Billion isn't enough, people?). The time is here, my friends! We were told when we were young, "Just s@y no!"

KEYWORDS

everything but "A"

[1] No "A"s were used in the writing of this study, besides from quoting in the Brief section, plus the one in this footnote. Couple of 'em show up in cited box picture plus references too, due to required requirements involving proper referencing, but those writers weren't yet enlightened by the benefits described in this meticulous/mellifluous study.

1 Introduction

Since the beginning of time, people thought everything beginning needs ended. Every sunrise, needs sunset. Every yes, needs no. Bright, oppressively humid summers need dimly light, foggy winters. However, people love progress. Why do we need beginning to our letter system? Beginnings imply endings. We shine light on the huge issue. Who doesn't like to touch their lips or tongue? Figure 1 shows some common ulternatives.

Figure 1. Common words beginning with the offending letter, which could be switched with the following terms (from left to right, up to down): doctor repeller, cute hog with long snout, fish biotope, trophy, respectively.

Upon questioning others on the concept of deleting the useless letter, divided experts tell us two sides of congruent stories. The optimist tells us, “No good comes from w@r, c$sh, c@lories, or c@lculus. Let’s destroy this useless letter to free ourselves from overuse.” The pessimist, non-supporter yells, “We need this letter so we will be together in concerted efforts to write useless scripts.” We purpose the deletion of the useless letter, providing numerous testimonials of when this letter ruined lives and provided inconveniences worthy of noting.

We hope that this effort will resonate with the millions of innocent students whose monikers begin with the letter that must not be mentioned, because of which they were forced to reply to roll call first and turn in their tests before others.

2 B for Effort

History shows nobody enjoys fellow students who perform better then others. Therefore, previous experience plus work shows B is the new high score. From this point on, the highest score students will receive in school will be the B. Yes, the highest score received will be B.

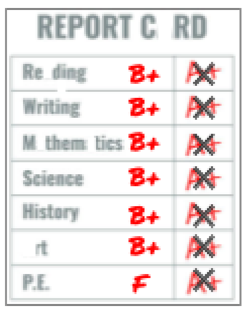


Figure 2. One student’s excellent report c@rd showing nere perfect scores, except in P.E., or gym, This is a common occurrence of techy-computer students excelling in m@th and not so much PE.

Type B persons will unite, then nobody will perform better then others. Everyone gets B’s, then B’s will get degrees. 3.0 here we come! Figure 2 below shows one pupils report c@rd expressing B is the new “it” letter.

3 Method

We counted the number of uses of the letters used in English words, showing them in figured with wide lines on it. We counted the number of uses of the letters used in this document. We will then determine the frequency distribution of both entire English words used with the letters used to write this document.

We collect surveys, interviews, focus groups, describing the vile issues of this letter. We run our own experiments combining mixed methods from everything listed to collect info supporting the deletion of the letter. Here is lists of expert experimenters giving user opinions:

- Postdoc, former middle school instructer describes one terrible experience she witnessed upon receiving the following pieces in Words with Friends-like bored exercise. She tells us, *“How will I configure words with these letters? This is so disoppointing.”*

- Softwere engineer notes, *“Binuhry computerwords definitively be the most efficient computerwords when tested next to other computerword collections, plus every other computerwords collection compiles to binuhry unyhow, so just use 0/1 to be the best with computers.”*
- Product chief did not contribute to this work, citing too busy. We suspect his left pinky finger is too tired from typing too much, hitting the terrible letter while writing lengthy e-letters. We feel you. Your coworkers completed the job of contributing for you. You’re welcome.
- Product person suggests (with obviously limited prerequisite study of biology) the idea of not having blood groups of the offending letter’s typer, resulting in more people of blood group B, making it easier to find donors for everyone.
- Hed of pertnerships for tutoring progrem & former streight @ student who wuz bullied her entire childhood for being uh nerd. Now she finlly h@s the opportunity to get b@ck at her childhood bullies and cl@im her new identity @s @ str@ight B student with @ Type B person@litiy.
- Softwre explorer constructs the best chicken recipe with no @’s in sight: Chicken veggie stir fry is quick to cook plus requires no useless

letters, good for your body. 1. Put cooking oil in wok till it's hot. 2. Put in chopped green onion & ginger to stir until they're soft with scented smell. 3. Put in diced chicken, stir till color turns white. 4. Put in diced veggies to stir until they're soft. Here're few good choices of veggies: broccoli, celery, cucumber, turnip, lotus root, bok-choy & zucchini. 5. Put in Shoyu with hot pepper for condiments. You will surely enjoy this! Check me out on Twitter (@wedon'tneedno) for more how-to recipes.

- Senior Inquiry Developer plus frequent speeding ticket receiver cites, *"Driver of vehicles that cruise beyond speeds exceeding permissible limits, desire to substitute their OWdi for the most recent BMW."* She got a new vehicle with no @'s.

4 Results

We document from reel-life experience no good comes from this letter. Numerous interviews plus countless confessions confirm its deletion. This work proves the letter is not needed to write completely and succinctly. Fig. 3 shows the frequency of letter usage in the English. The letter was the third highest used letter. Fig. 4 shows the letter distribution of this work.

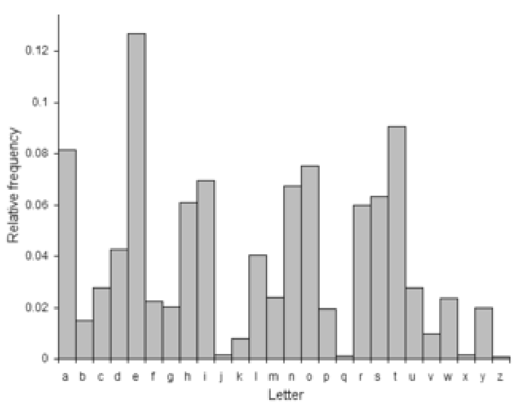


Figure 3. Frequencies of letters used in English[2] from the entirety of the letters.

[2]https://www3.nd.edu/~busiforc/handouts/cryptography/letterfrequencies.html

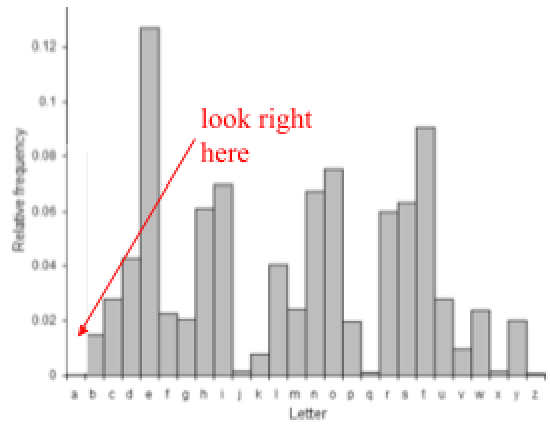


Figure 4. Frequencies of letters in English[2] within this document, further proving the use of the letter is unneeded, plus it is purely overkill. See on the left side there is one teeny tiny little line, very very teeny.

4 Discussion & Conclusion

The purposes of this study were two fold: (1) to show the world using other letters other then the horrific first letter will still get the job done, plus (2) to show the cynics it is possible to write some completely upsurd, ridiculous conscription of which people will follow until the very end. Yes, this is the end, my friends. By perusing this entire proof-of-concept mixed-methods monstrosity you've proven to the world, we don't need useless letter.

'KNOWLEDGMENTS

This work is supported with funding from nobody. This is mostly due to interested funders not interested in signing the checks, due to the pesky letter when e-signing their monikers.

REFERENCES

[1] Writers. 2022. B is the new "it" letter. [under review to be included in proceedings to prestigious conference.]

F

Fun, Meet Games

Multidiscipline Elo - A complex analysis

Kiera Jones

Abstract
Elo has been used as a common method for ranking the comparative strengths of competitors in a wide variety of fields. Since its introduction in 1960 for chess[1], it has since been used for numerous other sporting events, to mixed levels of success. Even with the ubiquity, one problem is that these Elo ratings are largely independent of each other. For instance, at time of writing, Magnus Carlsen has a chess Elo of 2852[2], Jauny has a League Elo of 2007[3], and the Boston Celtics have an NBA Elo of 1672[4]. However, these cannot tell us what to expect if Magnus Carlsen was put into League of Legends, and Jauny played as him in a match against the Boston Celtics. This is *obviously* a major flaw, and this paper will seek to address this issue.

Keywords
Puyo Puyo — Tetris — Elo — Maximum Likelihood Estimation

Rosenstiel School of Marine, Atmospheric, and Earth Sciences, Miami, Florida - [she/her/hers] - @K13205

Contents

Introduction

Unfortunately, there is some limitations as to what fits within the scope of this paper. For instance, the NBA for some reason failed to respond to a request to play hundreds of games of various types and record the results based solely on my whims. As such, this will only focus on a scale more easily performed by a graduate student: Puyo Puyo Tetris 2.

Puyo Puyo is a competitive Japanese puzzle game, wherein players try to cause their opponents to run out of space to place new pieces. To do this, players match up four or more puyos of the same color. As the puyos are affected by gravity, this can cause a chain reaction, and the larger the chain, the more nuisance puyo that are dropped on the opponent's playfield, taking up space. However, the opponent can create a chain of their own during this time to neutralise the incoming nuisance. One of the interesting factors of this game was that the different characters had different play-styles. For instance, some of them fill up the edge columns immediately before playing slowly in the center to develop chains, others will make use of quick and/or hard drop (if available) to build up their potential chains faster, and still others *don't realise that the pieces can be rotated*[5].

Tetris is, as we all know, an inventory-management survival-horror game[6]. Of the three main flavors of Tetris, modern Tetris adds a competitive element as well, where when one clears lines on their playfield, a number of lines of garbage missing a single square are sent to push up the bottom of the opponent's playfield, depending on the number of lines cleared at once, the number of consecutive pieces dropped which cleared a piece, and whether or not this or the previous line clear were Tetrises or T-spins.

Puyo Puyo Tetris (and its sequel), thus, is the puzzle game equivalent of Avengers: Endgame. It takes a bunch of the most popular Puyo characters, and adds in anthropomorphised versions of the Tetris pieces (and Sonic the Hedgehog for some reason). The Tetris characters received AIs from some of the Puyo characters that weren't in the first game (but not Ai, who was one of the Tetris characters added), and everyone got an AI for Tetris. And since the crossover aspect was highly emphasised, one can play Puyo against Tetris. Characters were split into those who preferred to play Puyo and those who preferred to play Tetris, with all the Tetris characters joining some Puyo characters (usually villainous, because nobody good would dare do a center four-wide well[7]) in preferring Tetris.

Because both Tetris and Puyo have a skill component, and this game provides twenty-two characters out of the box (up to twenty-eight after beating the game, and forty with all the DLC), this is a perfect medium for the creation of two separate Elo scales, and then looking at how their combination interacts.

1. Methodology

Given time constraints of having to watch all the AI matches manually, rather than thousands of simulations between the forty different characters in the game, a subset of eight characters were chosen for a proof-of-concept, evenly taken from those who specialise in Puyo and in Tetris:

1. Arle - *The protagonist of the entire Puyo Puyo series, Puyo*
2. Ringo - *One of the two protagonists of the Puyo Puyo Tetris games, Puyo*
3. Tee - *The other protagonist, who uses similar AI to Ringo, Tetris*
4. Squares - *The final boss of Puyo Puyo Tetris 2 (not sorry for spoilers), Tetris*
5. Ecolo - *Final boss of Puyo Puyo 7, considered to have one of the best Puyo AIs, Puyo*
6. Lemres - *Considered to have one of the best Tetris AIs, and also my favorite character, Tetris*
7. Draco - *Has a specifically different Puyo AI, in that she cannot rotate the pieces, Puyo*
8. Jay and Elle - *I needed another Tetris main, and the fact that there's two characters underscores the symbolism of this projects scope, Tetris*

These characters were then ran through three experiments to develop a basis for multidimensional Elo:

1.1 Baseline calculation

Each of the characters played five games against each of the other characters in both Puyo Puyo and Tetris. Puyo Puyo games were just two character matches, with player 1 set to CPU controlled.

This was originally going to be done the same for Tetris, but after the Arle and Ringo match took over an hour, and one specific game between Arle and Jay/Elle went half an hour without any sign of finishing, these were switched to three character matches, with player 1 being the author playing Witch in Puyo, and the other two being the Tetris CPUs. This is because Puyo Puyo matches have margin time, wherein after a set amount of time (96 seconds on default), all garbage sent is multplied by 4/3, and this continues until someone wins. This is missing from pure Tetris matches, but by throwing a Puyo character in (and immediately topping them out so as to not influence the results), margin time remained active for the Tetris match and reduced the games to only being *reasonably* slow.

For both of these, the results were used to derive the Elo for each game separately. Characters were given a base Elo equal to 40 points per win above 17.5. As the expected score from Elo are only based on difference between Elo rankings, the same results would be derived from a match between a 100 Elo and a -100 Elo player as would from a 39642 Elo and a 39442 Elo player. For ease of facilitating complex numbers, the Elo was set to average to 0.

From these base Elo values, the expected score of a single game was calculated. The binomial odds of seeing the results was calculated, and the Elo values were then modified through maximum likelihood estimation to find an estimate for the actual Elo values of the characters. To account for a scenario in which one group of characters always beat another group, a penalty was added to the log likelihood of one-two hundredth of the consecutive Elo differences.

1.2 Cross-game comparison

As mentioned when talking about Tetris above, Puyo Puyo Tetris allows matches to be played between Puyo and Tetris. For this, each character played three games as Puyo against every other character as Tetris.

These results were then compared with the Elo values calculated in the first experiment to determine if the results were consistent with these prior values. As there were prior Elo values for this, these were updated with the results to create the final Elo for the final experiment. Once again, maximum likelihood estimation was used to find the correct K-value for updating the scores.

1.3 Combined games and generalised complex Elo

Puyo Puyo Tetris also has two game modes where Puyo and Tetris are played at the same time. First up is Swap, where players start in one of the games, and after twenty-five seconds, move to the other game, where whoever is first to win in either game wins the match, and the garbage/nuisance to be sent to the other player carries over and stacks on top of each other, with bonuses for sending over both of them at the same time. The second game mode is Fusion, which *sucks*[8], and will have no further discussion in this paper.

As such, each of the characters played three games of Swap with each of the other characters. From this, these data were combined with the other results to create a method for using two disparate Elo values for a combined prediction and comparison. This is where complex numbers mentioned in the title are brought in, as an easy way to have two distinct Elo scales in one number not directly affecting each other. Maximum likelihood estimation was used yet again, this time to figure out Swap's complex game value (a notion that will be introduced in Section 4).

2. Experiment 1: Baseline Puyo and Tetris Elo

2.1 Puyo Puyo

The Puyo section of games showed some potential flaws that were not initially accounted for. The original plan was to use the characters' core AI (an advanced version that plays at a

Table 1. Puyo match results

	Arle	Ringo	Tee	Squares	Jay/Elle	Lemres	Ecolo	Draco
Arle		5	4	0	2	0	0	3
Ringo	0		2	0	1	0	0	2
Tee	1	3		0	0	0	0	2
Squares	5	5	5		5	4	3	5
Jay/Elle	3	4	5	0		2	0	5
Lemres	5	5	5	1	3		4	5
Ecolo	5	5	5	2	5	1		5
Draco	2	3	3	0	0	0	0	

much higher level than the normal ones). However, after a couple of matches, there was only a slight difference between the characters performances; everyone was quickly dropping their puyos and making six-plus chains. As such, this trial was restarted fairly early with the normal AIs.

The normal AIs did show quite a large difference between how they performed, almost to a degree of extreme separation. Apparently, the choice of the eight characters didn't *quite* span the spectrum as much as expected, or rather, they spanned a fairly large degree, but instead of being quite separated, they were clumped into three groups: To no surprise, Ecolo, Lemres, and Squares were significantly better performers than the rest of the characters, and Ringo and Tee were significantly worse. Surprisingly enough, Draco was an admirable performer.

2.2 Tetris

As noted in the methodology, Tetris had an initial difficulty in that, without margin time, matches kept on dragging on and on. Even with figuring out a way to introduce the margin timing, these matches were still by far the lengthiest to run.

The largest problem with the Tetris matches were that the games were heavily based on the skill, and as such, the divisions noted before were more extreme. While Ringo and Tee were 6-54 against the other characters in Puyo, and Squares, Lemres, and Ecolo were 73-2, these were a perfect 0-60 and 75-0. This is why the additional weighting for the separation was added to prevent the Elo scores from going to infinity when solved for with maximum likelihood estimation.

2.2.1 MLE problems with perfect records

As a refresher, if player A has an Elo rating of R_A and player B likewise has rating R_B, the expected value of player A over a single match is

$$E_A = \frac{1}{1+10^{\frac{R_B-R_A}{400}}} = \frac{10^{R_A/400}}{10^{R_A/400}+10^{R_B/400}}$$

If player A won every game in a match of n games, then from a binomial distribution, the odds of that happening are ${E_A}^n$. Defining $R_A = 0$ for ease of calculations (seeing that there is only one equation and two variables), the log likelihood is

$$-n\ln\left(1+10^{R_B/400}\right) \tag{1}$$

This can be made arbitrarily close to zero by making R_B increasingly negative.

If there is a single loss, however, the odds of this result occuring are $n\,{E_A}^{n-1}(1-E_A)$, leading to a log likelihood of

$$\ln n - n\ln\left(1+10^{R_B/400}\right) + \frac{\ln 10}{400}R_B \tag{2}$$

In this case, as R_B is made increasingly negative, the first term is a constant and the second tends to zero, but the third decreases without limit, providing a bound to the gap.

2.3 Elo calculations

The data were entered into Excel, and the default solver was used to find the maximum value of the likelihood. To ensure a somewhat consistent starting value, the initial values of Elo were set to $40(w-17.5)$ for a character with w wins for everyone but the last character, and the last character was set to the negative sum of the other characters to ensure the total Elo was zero.

Table 2. Tetris match results

	Arle	Ringo	Tee	Squares	Jay/Elle	Lemres	Ecolo	Draco
Arle		5	5	0	1	0	0	1
Ringo	0		3	0	0	0	0	0
Tee	0	2		0	0	0	0	0
Squares	5	5	5		5	4	3	5
Jay/Elle	4	5	5	0		0	0	3
Lemres	5	5	5	1	5		1	5
Ecolo	5	5	5	2	5	4		5
Draco	4	5	5	0	2	0	0	

To set up the log likelihood, the formula

$$= \texttt{LN(BINOM.DIST(C4,5,1/(1+10\^{}((C\$2-\$B4)/400)),0))} \quad (3)$$

was used, where C4 was the square containing the number of wins said character had, $B4 was the chosen character's Elo (mirrored from the vertical column that had the modified values), and C$2 was the Elo of the other character. These values were added up (only once for each match between characters), as well as adding $1/200$ of the largest gap between consecutive Elo ranks.

Table 3. Elo calculations, experiment 1

	Puyo Elo	Tetris Elo
Squares	497	642
Ecolo	371	625
Lemres	373	449
Jay/Elle	38	-19
Arle	-138	-229
Draco	-332	-56
Ringo	-407	-697
Tee	-402	-715
Log Likelihood	-28.886	-19.712

As seen from the maximum likelihood, the Elo values are sensible from what was seen before. Characters who had similar number of wins had similar Elo values, and more wins always resulted in a higher Elo. The log likelihood of the Tetris was much higher, as a result of having the splits between the characters, and there being more matches that were decided 5-0

3. Experiment 2: Puyo vs Tetris and Elo updates

Of course, the main selling point of Puyo Puyo Tetris is, of course, the ability to play Puyo and Tetris *against* each other. As such, any attempt of creating an Elo with both of them should be able to look between the two.

In effect, pitting the two against each other dealt with multiple issues that the separate trials from experiment one had. First up, the problem of groups always or never winning has more chances to happen (and thankfully, it did happen). Because of this, for the final maximum likelihood estimation, the additional term for distance was removed, as there is no worry about the values spreading to infinity.

The second potential result is a well-known meme in Puyo Puyo Tetris, that the Tetris player is at a slight disadvantage[9]. Normally taken as a joke for when a Tetris player pulls out a ridiculous combo and recovers to win, but the analysis was from mid-level players where just the different playstyles supposedly leads to a slight advantage to Puyo. By pitting the two against each other, one can see if such an advantage really exists.

The third and final result is that since maximum likelihood estimation takes quite a bit of time to run and set up, being able to know the K-factor for incremental updates to the Elo, and after the fact, can be calculated for each character individually rather than needing to solve for everyone at once. Unfortunately, this does need one last MLE run to get the K-value, but in future works for Puyo Puyo Tetris Elo, these results will graciously be available to save everyone involved some time.

Please note that in Table 4, the results only list the number of games that the Puyo player (in row) won, and the number of wins by the Tetris player (in column) is just three minus that number.

Table 5. Puyo vs Tetris relative performance

	Puyo		Tetris	
	Expected	Actual	Expected	Actual
Arle	8.11	11	7.09	6
Ringo	4.03	3	0.97	3
Tee	4.04	3	0.89	3
Squares	17.36	20	19.80	14
Jay/Elle	10.73	17	10.70	8
Lemres	15.47	18	17.69	14
Ecolo	16.06	17	19.30	16
Draco	6.92	7	8.83	7

As expected from the meme, generally the Puyo character tended to win slightly more often, going 97-71 in the 168 matches. As such, the prior thought fact that the Tetris player being at a slight disadvantage is actually true (or at least, for the specific chosen characters). This ends up being equivalent to a 52 point Elo advantage for the Puyo characters.

Table 4. Puyo results vs Tetris opponents

	Arle	**Ringo**	**Tee**	**Squares**	**Jay/Elle**	**Lemres**	**Ecolo**	**Draco**
Arle		3	3	0	2	0	0	3
Ringo	1		2	0	0	0	0	0
Tee	1	2		0	0	0	0	0
Squares	3	3	3		3	3	2	3
Jay/Elle	3	3	3	3		1	2	3
Lemres	3	3	3	2	3		1	3
Ecolo	2	3	3	1	3	3		2
Draco	2	1	2	0	2	0	0	

To estimate the K-value for these games, once again, maximum likelihood was set up with the same formula as before. However, rather than solving for all the Elo values, the only thing modified was a multiplier on the difference between the expected and actual values from table 5. The value of log likelihood used was the sum of the all three trials from the final Elo calculated, rather than just the Puyo vs Tetris trial.

After calculating, the K-value calculated was 24.95, for a log likelihood of -25.31 for this trial and an average log likelihood of -24.96. Since this value was close to 25, which is a somewhat commonly used K-value and a lot easier to calculate with, this value was used instead, as the only difference between it and the calculated value was Jay and Elle gained one more point in Puyo (which, as it was not counteracted by a point loss anywhere else, was just ignored to keep the average score equal to zero).

Table 6. Elo calculations, experiment 2

	Puyo Elo	Tetris Elo
Squares	563	522
Ecolo	394	543
Lemres	436	357
Jay/Elle	219	-86
Arle	-66	-256
Draco	-330	-102
Ringo	-433	-646
Tee	-428	-687

4. Experiment 3: Swap and Generalised Complex Elo

4.1 Elo combination

At this point, keeping the two Elo values separately is getting kind of annoying for bookkeeping purposes. As such, since the Puyo and Tetris values are well defined for each character, we just need some form of number such that it can keep both parts together. This is a perfect situation to bring in everyone's favorite part of algebra: Complex numbers.

Definition 1 *If a competitor has an Elo rating of R_A in game A and R_B in game B, then the* ***complex Elo*** *Z for those two games is $R_A + R_B i$.*

For the purposes of this paper, analyses will be done with $Z = R_P + R_T i$, that is, the real portion will be the Puyo Elo and the imaginary portion will be the Tetris Elo. This is for multiple reasons: The games are set in the Puyo universe so they should receive top billing; Tetris deals with squares and if you square an imaginary number you get a real number, while if you circle an imaginary number you just get ⓘ; and the names of the Tetris characters are so ridiculously uncreative they *can't* be real.

Although easier for manipulation, this does bring up some problems with respect to the classic Elo formulae. Firstly, since an exponent of a complex number results in a rotation, this would make the probability calculated for E_A have an imaginary component, and could also result in a case with the real part outside of $[0,1]$. For instance, if a player had complex Elo of $400+400i$ against someone with an Elo of 0, this would naïvely appear to result in an expected value of $\frac{1}{1+10^{-1-i}} \approx 1.065 + .085i$. Secondly, this would also imply that how good one is at Puyo would affect the odds of winning a match of pure Tetris, and vice versa. Thirdly, this also cannot distinguish between Tetris or Puyo matches, and would give the same result in any case.

To solve the problems in reverse order, we thus define these terms:

Definition 2 *A game's* ***complex value*** *$\check{g}$ is a unit Chebyshev[10] complex number (that is, of the form $\pm 1 + bi$ or $a \pm i$ for $|a|, |b| \le 1$) that indicated the relatve weights of the complex Elo components of the game.*

Definition 3 *A* ***complex game vector*** *$\mathcal{G}$ is a vector $< \check{g}_A, \check{g}_B >$ consisting of the values of the games played by competitor A and competitor B*

While experimenting for the definition of a complex game value, tests were performed for both numbers on the unit circle and on the unit square. As will be seen later when actually detailing the Swap games, the unit square values ended up being more accurate, despite some details that will be discussed further.

The game vector helps to separate the different games that can be played. Overall, there are four broad classes that the game vector can take, three of which are looked at in this paper:

Case 0 *Pure game* - Both players are playing the same game that makes up one of the components of their Elo. In this case, $\mathcal{G} =< 1,1 >$ or $< i,i >$, but the equation results exactly the same as in normal Elo.

Case 1 *Mixed game* - The players are playing the two different components of their Elo. In this case, $\mathcal{G} =< 1,i >$ or $< i,1 >$, depending on which player is playing which game.

Case 2 *Combined game* - The players are playing the same game, but it is a mix of the two components of their Elo. In this case, $\mathcal{G} =< \check{g}, \check{g} >$.

Case 3 *Generalised game* - This is the catch-all collection for everything else. Most uses for generalised complex Elo will be either cases 1 or 2, but this is the broadest sense and the only one in which no simplifications can be made.

From a given game value, a characters skill with respect to that value can be found by dividing their complex Elo with the value. The real component of this result corresponds to how much of their Elo is in this direction, and the imaginary component is an orthogonal addition to reach the full value. Thus by only looking at the real component of this division, all three problems are solved: there's no imaginary component

Table 7. Swap results

	Arle	Ringo	Tee	Squares	Jay/Elle	Lemres	Ecolo	Draco
Arle		3	2	0	0	0	0	2
Ringo	0		0	0	0	0	0	0
Tee	1	3		0	0	0	0	3
Squares	3	3	3		3	2	0	3
Jay/Elle	3	3	3	0		0	1	3
Lemres	3	3	3	1	3		1	3
Ecolo	3	3	3	3	2	2		3
Draco	1	3	0	0	0	0	0	

to give rotation, the game can be uniquely identified, and the skill in a game not being played only adds to the imaginary component, and does not effect the calculated value. As such, the expected score for generalised complex Elo is

$$E_{A,\mathcal{G}} = \frac{1}{1+10^{\frac{1}{400}\Re\left(\frac{Z_B}{\breve{g}_B}-\frac{Z_A}{\breve{g}_A}\right)}} \tag{4}$$

One may wonder what value the imaginary component of the difference in the exponent corresponds to. It probably deals with some measure of the uncertainty, as it shows how off the main game value, however, this was not explored due to ~~Big Elo not sufficiently paying off the authors of this paper~~ time constraints.

Similarly, the formula for updating complex Elo for a player receiving a score of S_A is only slightly modified from the normal Elo formula, only adding a factor of $\breve{g}_A$ to turn the outcome back into complex values.

$$Z'_A = Z_A + K\breve{g}_A(S_A - E_A) \tag{5}$$

4.2 Puyo Puyo Tetris Swap

The last set of games played was, as noted in the methodology, three sets of Puyo Puyo Tetris Swap. The goal of this was to find the correct game value for Swap. Additionally, this could also be used to see if the game value should be of magnitude one, or have maximum value one (*i.e.* if they are of the set $\|z\|_2 = 1$ or $\|z\|_\infty = 1$). Both of these are simple norms, and in the limits of a pure Puyo or Tetris matchup, both are either $\breve{g} = 1$ or i respectively.

For those of you foolhardy enough to try and enter this into your Excel file to solve, the equation for the log likelihood[11] used here is

$$\begin{aligned}&= \texttt{LN(BINOM.DIST(C4,3,1/(1+10\^{}((IMREAL(}\\&\texttt{IMDIV(IMSUB(C\$2,\$B4),\$B\$12)))/400)),0))}\end{aligned} \tag{6}$$

The cells mentioned before are the same as in Equation 3, with the addition of `$B$12` as the game value, which is what is going to be solved for. For common sense reasons, the maximum likelihood was chosen within the range 0 and $\pi/2$, and then raising e to i times this power (as this has both components greater than zero, that is, if one gets better at either Puyo or Tetris, then they have a higher effective rating). To check the correct norm, this was run a second time, multiplying the game value by the smaller of the secant or cosecant of the angle.

The maximum norm, having a larger magnitude, results in smaller relative differences in Elo, and thus, would increase the probability of seeing a split match rather than a sweep. Because of all the matches that were won 3-0, this seemed to rule out that norm, however, the likelihood was actually higher for this norm by about 50%. The Euclidian norm found a game value for Swap of $.893 + .450i$ with a log likelihood of -20.17, while the maximum norm found a game value of $1 + .667i$ and a higher likelihood of -19.77. Of note is that these are not in the same direction (the Euclidian norm was at angle .467rad while the maximum norm is at angle .588rad).

Both methods struggled with the fact that Tee, heretofore the combined worst player, swept Draco in the three games played. Because Draco was only slightly better in Puyo, but significantly better in Tetris, the original maximum likelihood in both cases was extremely close to just being pure Puyo, with less than 1% of the total performance attributed to Tetris in either case. Thus, the maximum likelihood was rerun with ignoring this one specific match as an outlier to give the results presented above.

Additionally, since this has additional data from the swap games, the Elo can be incremented, using Equation 5 and the results acertained from the previous experiments. The biggest changes from this were that Tee improved his Puyo score with his very successful Swap run, even surpassing Draco for 6th in Puyo and easily jumping past Ringo to not be the worst overall, and Ecolo jumped ahead with their Tetris score, but not enough to get into first overall.

Table 8. Elo calculations, experiment 3

	Complex Elo
Squares	$515 + 490i$
Ecolo	$433 + 569i$
Lemres	$445 + 363i$
Jay/Elle	$248 - 67i$
Arle	$-96 - 276i$
Draco	$-395 - 146i$
Tee	$-305 - 605i$
Ringo	$-489 - 684i$

5. Conclusion and further study

The purported purpose of this paper was to develop a groundwork for the combination of Elo between multiple different competitions. Overall, especially with the successful application on Puyo Puyo Tetris, this has been (in the author's biased opinion) an unqualified success.

One of the assumptions used in the MLE for Swap was that getting better at either of the components would help in winning the game. A future diversion might look into a combined game with one of the components allowing misère play, to see how the derived results hold up to a system in which this seemingly obvious assumption is violated.

The results from this paper also offer an obvious extension into quaternions for three or four different games being combined into one, or even octonions for up to eight games. As sedenions have zero divisors[12], the division required for the expected score fails to have a unique solution, but surely the combination of more than eight games at once is a fanciful endeavor that would never exist in real life[13].

Additional trials to acquire further data for the Puyo Puyo Tetris Elo would always be helpful to get a fuller picture of the AIs in this game. Only a fifth of the characters in the game had any research done on their strengths, and so further studies, especially to look at cases of Swap in which there is a serious difference in Puyo and Tetris abilities.

Not mentioned before, but the Puyo and Tetris values being fairly close was a bit of a problem, as with possibly the exception of Draco, Tetris ability was fairly well predicted by Puyo ability. A linear regression run on the Elo scores gave an r value of .921, with the calculated best-fit line being $R_T = 1.09R_P - 92.75$. This high correlation meant that the Swap matches already had a pretty solid guess beforehand which of the AIs would end up being better in the match, and so the winner was fairly predictable ahead of time. If further research found more characters with large differences between the Puyo and Tetris abilities, perhaps the imaginary portion of the expected score formula could be better explored.

And if anything should be taken from this paper, the author hopes that at least one person sees this, decides to pick up Puyo Puyo Tetris (the first game), and encounters the dialogue of Chapter 9.

Acknowledgments

The author would like to thank her friends Laser and Blain, for introducing her to the rabbithole that is Puyo Puyo and its lore, and to her friends Kaynato and Andrea for likewise introducing her to SIGBOVIK. Any other of her friends reading this are cool too, but those four are directly responsible for this mess.

She also would like to thank her horrible old retail job, for making her so miserable she was able to get past her executive dysfunction and apply to graduate schools, and a more sincere thanks to her therapist who encouraged this. She was accepted last fall into a Masters program for meteorology, a passion of hers for the past twenty-plus years, and this transition has vastly improved her mental health and given her the free time to make writing this paper possible.

Lastly, she would like to thank the investigators reading this paper in the distant future, when its seminal value to the field of predictive forecasting is well known and inevitably there is a scandal involving an NBA team cheating at League of Legends through the use of vibrating devices[14], and this paper is looked back on to see how this became a problem.

References

[1] Elo, Arpad, 1978: *The Rating of Chess Players, Past & Present.* Arco Publishing

[2] FIDE International Chess Federation, 2022: *Fide Ratings and Statistics.* https://ratings.fide.com/

[3] OP.GG, 2013: *League of Legends Leaderboards.* https://www.op.gg/leaderboards/

[4] FiveThirtyEight, 2022: *2022-23 NBA Predictions.* https://projects.fivethirtyeight.com/2023-nba-predictions/

[5] Jaderuby, Teejay, 2017: *Draco was a mistake : Part 2 ??* http://youtu.be/AP9H9Q_a47U

[6] Murphy, Tom, VII, 2022: *Harder Drive: Hard drives we didn't want or need.* SIGBOVIK, **16**, 259-277

[7] Gravitoon, 2018: *No 4-wides.* http://youtu.be/afPkAeBj-Vw

[8] Oshisaure, Lilla, 2018: *O-SPIN IS A REAL THING.* http://youtu.be/CRfV5csii5Y

[9] Doremy, 2017: *"the tetris player is at a slight disadvantage"*⋆ http://youtu.be/ey0CVtp8UcU

[10] Teknomo, Kardi, 2015: *Similarity Measurement* http:\people.revoledu.comkardi tutorialSimilarity

[11] This is not an actual citation, this is just a friendly reminder that friends don't let friends do maximum likelihood estimation, *especially* not in Excel.

[12] Arndt, Jörg, 2010: *Matters Computational - Ideas, Algorithms, Source Code* Springer. 814-821 http://www.jjj.de/fxt/#fxtbook

[13] Wikimedia Foundation, 2023: *Decathlon - Wikipedia* https://en.wikipedia.org/wiki/Decathlon

[14] Jun, 2023: My Discord DM with them where they looked at the abstract and then thought this was what this paper was going to be on.

A Simple RISK Architecture

Stefan Muller
Illinois Institute of Technology

ABSTRACT

This paper introduces RISK-V, a minimal computer architecture that targets a widely-available hardware platform not previously used for computing.

1 INTRODUCTION

Reduced Instruction Set Computer, or RISC, architectures [1] have been studied since the early 1980s, with the goal of providing a simple, easily implementable, instruction set architecture. Less well-studied (that is, not studied until now) are RISK architectures, which are even simpler to implement because they target readily-available and portable[1] hardware, namely the board game *Risk*.

2 COMPUTING PLATFORM

Our architecture is built on top of a simplified, two-player version of Risk, which we will now describe. Because of its substantial departures from the usual rules, we refer to our architecture as RISK-V (the V standing for “Very Simple”). The game is played on a board resembling a world map that might be found in a history textbook or an ancient globe in an underfunded school. The map is divided into 24 territories, whose ownership is divided between the two players at the start of the game. Because randomness in computing hardware makes writing programs difficult, we perform this assignment deterministically. Each player maintains some (strictly positive) number of *armies* on each territory they hold. Initially, each territory contains one army.

The two players take turns attempting to conquer the world. A turn consists of three phases, as follows:

(1) *Reinforcement.* The player may place any number of armies in any territory they control. This may be repeated any number of times.
(2) *Attack.* The player may make zero or more attacks. In an attack, the player selects a territory they control to attack from, and an adjacent territory controlled by the other player to attack. The player may attack with any number A of armies between 1 and $N-1$, where N is the number of armies in the attacking territory. Let D be the number of armies in the defending territory. If $A \leq D$, the attack is unsuccessful. The number of armies in the defending and attacking territories are decreased by A (except that if $A = D$, one army is left in the defending territory). If $A > D$, the attack is successful. The attacking player gains control of the defending territory; $N - A$ armies are left in the attacking territory and $A - D$ armies move to the defending territory.
(3) *Tactical Move.* The player may make one tactical move, moving some number of armies between two adjacent territories both controlled by the player. At least one army must be left in each territory.

Performing an invalid move raises a fault.

3 RISK-V ASSEMBLY

RISK-V assembly language contains 5 instructions. Arguments in angle brackets are required, arguments in parentheses are optional.

- `REINFORCE` ⟨*territory*⟩ ⟨*n*⟩
  Reinforce ⟨*territory*⟩ with *n* armies.
- `ATTACK` ⟨*territory*$_1$⟩ ⟨*territory*$_2$⟩ (*n*)
  Attack ⟨*territory*$_2$⟩ from ⟨*territory*$_1$⟩ with *n* armies (or the maximum number if unspecified). The current player’s turn moves to the attack phase if it is not already there.
- `TACTICAL` ⟨*territory*$_1$⟩ ⟨*territory*$_2$⟩ (*n*)
  Tactical move from ⟨*territory*$_1$⟩ to ⟨*territory*$_2$⟩ with *n* armies (or the maximum number if unspecified). The current player’s turn moves to the tactical phase.
- `END TURN`
  Ends the current player’s turn. May be executed in any phase of the turn. Must be executed to end the turn even if a tactical move has been executed.
- `IF UNSUCCESSFUL GOTO` ⟨ `label` ⟩
  Go to the instruction labeled ⟨ `label` ⟩ if the last attack or tactical move was unsuccessful. A tactical move is unsuccessful if not enough armies were available in the “from” territory.

4 CONVENTIONS AND PATTERNS

We will use the territories as registers holding nonnegative integers. Because territories cannot be empty, we will consider a territory with N armies to hold the number $N - 1$. By convention, the return value of the program is placed in Kamchatka. We refer to the players as A and B.

Addition. We can set $a = a+b$ using the instruction `TACTICAL` *b a*. This, of course, requires that the territories holding *a* and *b* are adjacent and both controlled by the current player, and that it is the tactical move phase.

Subtraction. We can set $a = \max(0, a - b)$ using the instruction `ATTACK` *b a*, assuming *a* and *b* are adjacent, *b* is controlled by the current player, *a* is controlled by the other player, and it is the attack phase.

Negating Conditions. The only built-in jump instruction is `IF UNSUCCESSFUL GOTO`. We can construct code that branches to a label on a successful move with an `IF UNSUCCESSFUL GOTO` that jumps over 1) a move that always fails and 2) another `IF UNSUCCESSFUL GOTO`.

```
IF UNSUCCESSFUL continue
```

[1]Unless you have some sort of special edition version that comes in a wooden box or something.

SIGBOVIK 2023,
2023.

```
  # Always fails
  ATTACK WAUSTRALIA EAUSTRALIA
  IF UNSUCCESSFUL GOTO l
continue:
```

Duplicate. The addition and subtraction operations described above result in the value in a being lost, so it is useful to duplicate the value in a territory before performing one of these operations. As an example, the following code copies the value in Kamchatka to Mongolia and Yakutsk (in the process erasing the value in Kamchatka; if desired, the value can be moved back to the original register). The source and both destination territories must be controlled by the current player, and at least one destination must be adjacent to the source. The code moves one army at a time from Kamchatka to Mongolia as long as this is possible, and also reinforces Yakutsk with one army after each move.

```
  # Copy Kamchatka to Mongolia, Yakutsk
  # All three are held by Player B
copy:
  TACTICAL KAMCHATKA MONGOLIA 1
  # Now A's turn; A ends turn
  END TURN
  END TURN
  IF UNSUCCESSFUL GOTO end_copy
  REINFORCE YAKUTSK 1
  # Always fails
  ATTACK WAUSTRALIA EAUSTRALIA
  IF UNSUCCESSFUL GOTO copy
end_copy:
```

5 CASE STUDY: FIBONACCI

The code in Figure 1 calculates the Nth Fibonacci number, where N is the value in Ontario at the start of the program.

6 DISCUSSION AND FUTURE WORK

We have implemented a simulator for RISK-V[2], on which we have tested the code provided in this paper. Future work would be to build compiler backends targeting the RISK-V architecture.

REFERENCES

[1] David A. Patterson and Carlo H. Sequin. 1981. RISC I: A Reduced Instruction Set VLSI Computer. In *Proceedings of the 8th Annual Symposium on Computer Architecture (ISCA '81)*. IEEE Computer Society Press, Washington, DC, USA, 443–457.

[2] A simulator hardly seems necessary, as the architecture targets a readily-available platform, but is provided for legacy users who require compatibility with outdated, silicon-based computing hardware

```
  # Invariants:
  # n = N - value in Ontario
  # a = fib(n-1) (stored in Irkutsk)
  # b = fib(n) (stored in Kamchatka)
  # Double N
  # Copy N to WesternUS, Greenland
copy0:
  TACTICAL ONTARIO WESTERNUS 1
  # Now B's turn; B ends turn
  END TURN
  END TURN
  IF UNSUCCESSFUL GOTO end_copy0
  REINFORCE GREENLAND 1
  # Always fails
  ATTACK EAUSTRALIA WAUSTRALIA
  IF UNSUCCESSFUL GOTO copy0
end_copy0:
  TACTICAL GREENLAND ONTARIO
  END TURN
  END TURN
  TACTICAL WESTERNUS ONTARIO
  END TURN
  # Initialize b to 1
  REINFORCE KAMCHATKA 1
  # Set Northwest Territory to 4
  REINFORCE NORTHWESTTERRITORY 3
  # Start while
loop:
  REINFORCE NORTHWESTTERRITORY 3
  # Copy b to Mongolia, Yakutsk
copy:
  TACTICAL KAMCHATKA MONGOLIA 1
  # Now A's turn; A ends turn
  END TURN
  END TURN
  IF UNSUCCESSFUL GOTO end_copy
  REINFORCE YAKUTSK 1
  # Always fails
  ATTACK WAUSTRALIA EAUSTRALIA
  IF UNSUCCESSFUL GOTO copy
end_copy:
  # b = a + b
  TACTICAL IRKUTSK KAMCHATKA
  END TURN
  END TURN
  TACTICAL YAKUTSK KAMCHATKA
  END TURN
  END TURN
  # a = Mongolia
  TACTICAL MONGOLIA IRKUTSK
  END TURN
  END TURN
  # IF N-- < 2 THEN END
  ATTACK NORTHWESTTERRITORY ONTARIO 2
  END TURN
  END TURN
  IF UNSUCCESSFUL GOTO loop
```

Figure 1: RISK-V Assembly for the Fibonacci function.

Solving Catan

Sam Schoedel, Sam Triest

March 2023

Abstract

We present a concrete, semi-functional implementation of a program that plays (part of) a Catan game.

1 Introduction

Settlers of Catan is a multiplayer board game where players barter for limited resources to dominate the fictional island of Catan. We're going to assume anyone reading this already knows the rules. If not, it really doesn't matter at all. It is a game of nearly perfect information, so most information needed to play the game can be determined from a visual inspection of the board. In the beginning of the game, players take turns placing two settlements each between resource tiles. Initial settlement placement has the largest impact on the game [5]. Because of this, Sam Triest et al. created an algorithm that determines initial settlement placement. To make the algorithm easily usable, Sam Schoedel et al. created a computer vision program that determines the board state from an image of the board. This paper presents the combination of these two projects into one Catanbot^{TM} program.

2 Determining Board State

The basic Catan board is comprised of the elements shown in Fig. 1. It contains 18 resource tiles with numbers, one desert tile, one robber, some ports, and the players' settlements and roads. The board state is determined using a gauntlet of thresholding, hough transforming, classification, and digit recognition.

Figure 1: example Catan board.

2.1 Roads and Settlements

A graph of the board was created by finding each number tile using a circle Hough transform and then branching between each circle in a hexagonal pattern. This allowed us to compute each of the green dots shown in Fig. 2. We then just masked out all parts of the image that weren't near expected settlement and road locations filtered based on player color.

Figure 2 shows correctly classified roads and settlements for the orange, blue, red, and white players. The green dots represent potential road and settlement locations. The initial settlement placement logic uses existing settlement and road placement information to determine the next best place to build a settlement.

Figure 2: road and settlement detection.

2.2 Resource Tiles

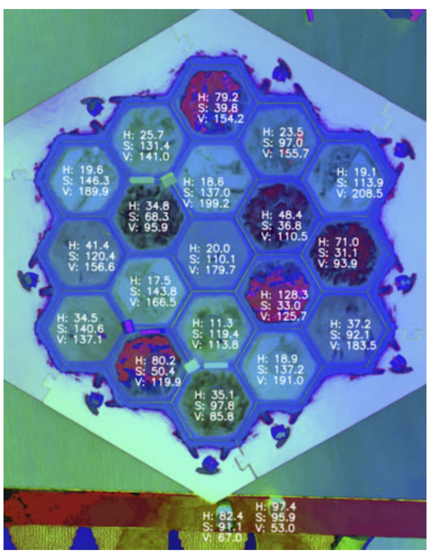


Figure 3: resource detection.

Resources are detected by just hard coding HSV thresholds for each type. Figure 3 shows the average HSV values for each tile with the nominal lighting in my room at the time. The average resource hues and saturations are far enough apart that it works under these conditions. If the lighting changes it completely breaks. Oh well.

2.3 Ports

Ports were hard to classify because the only features they have are the little icons on each ship. Since the photos we used were taken by a phone camera, we solved this problem by color thresholding each ship in the water area, finding the most obvious contours (the ships), and zooming in on them. This gave us a higher-than-necessary image resolution for each icon. After taking a bunch of photos of ports and hand annotating them we were able to train a classifier with about 95% validation accuracy.

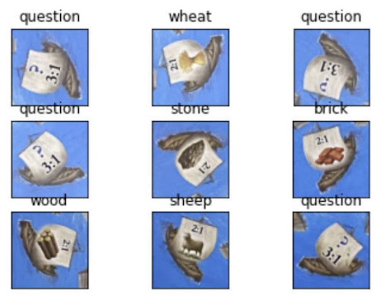


Figure 4: port classification.

Here's some port results. I guess we could have just lied about the accuracy and hard coded these labels, but you'll have to take my word for it. We did hard code the location of each port. hehe

2.4 Numbers

Off the shelf digit classifiers couldn't detect the tile numbers at all, so we opted to spend too many hours hand labeling another dataset and training another classifier with around 95% validation accuracy.

We found each number using a circle Hough transform and cropped the original high resolution image to obtain just the number section, then did some more thresholding using the little dots under each number tile to orient the image vertically. Classification was then done on the vertically oriented, highly zoomed in image. Figure 5 is an example of a board state with correctly classified and annotated numbers.

Figure 5: number classification.

2.5 Robber

The Robber's wherever there's no number tile.

2.6 All Together Now

Putting it all together, we get something that looks like Fig. 7, a graphical representation of the same board we took a photo of.

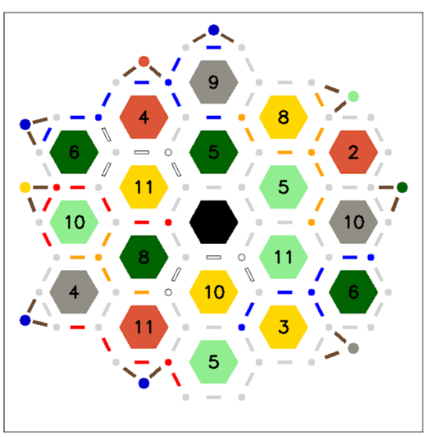


Figure 6: board state recreated.

3 Initial Placement

Disclaimer: I'm writing on behalf of my partner for this part, but he gave me some material he wrote to pull from so we should be all good.

Disclaimer: I wrote the code and I barely know what's going on, so we should be all good.

Let's show some results first:

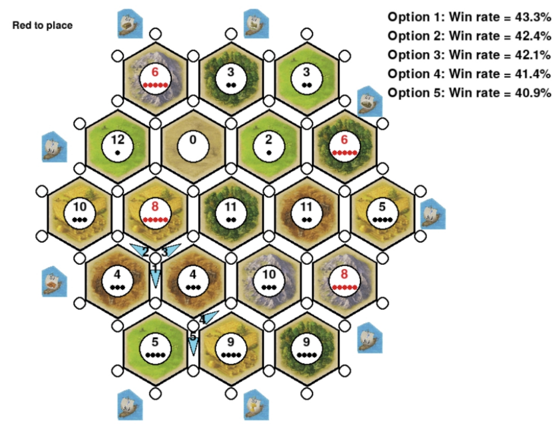


Figure 7: winrates with red to place.

Our system uses a combination of Monte-Carlo tree search and deep reinforcement learning to choose good initial settlement locations for the game Catan. Shown in Fig. 7 are initial placement suggestions for a particular board.

Ideas from AI have been applied to Catan in the past. One of the earliest instances is Jsettlers [2], an implementation of Monte-Carlo Tree Search for Catan. [3] and [1] both employ reinforcement learning techniques to the main game of Catan. All previous works in this area focus on playing the game of Catan after the initial placements phase. [2] use Monte-Carlo tree search to select actions and thus do not require learned functions for value estimation. [3] and [1] both train function approximators on self-play in order to select optimal actions. [3] use hand-crafted features, while [1] transform the hexagonal board into an image-like grid in order to use convolutional neural networks for value estimation.

3.1 Catan Simulator

We first wrote a GPU-accelerated, vectorized Catan simulator to play Catan games. Using this Catan simulator, we can play like 50 games per second on a NVIDIA 3080.

3.2 Time for an Algorithm

Here's an algorithm for all of your troubles (you may have to zoom in to get the full effect):

Algorithm 3: DQN with Alphago-style Self-Play

Input: Untrained Q-function parameters θ, Replay Buffer $\mathcal{D}$, win threshold α, discount factor γ, Learning rate η, Polyak term β
Output: Trained Q-function parameters θ

$\theta' \leftarrow \theta$
$\theta^* \leftarrow \theta$
while *not done* **do**
  $\tau = \text{rollout}(\theta^*)$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \tau$
  for *train iterations* **do**
    $(s, a, r, s', t, p) = \text{sample}(\mathcal{D})$
    $J(\theta) = \begin{cases} -(Q_\theta(s,a,i) - (r + \gamma \max_{a'} Q_{\theta'}(s',a',i)))^2 & \text{if } i = p \\ -(V_\theta(s,i) - (r + \gamma V_{\theta'}(s',i)))^2 & \text{otherwise} \end{cases}, \forall i \in [0,3]$
    $\theta \leftarrow \theta + \eta \frac{\partial J}{\partial \theta}$
  end
  $z \leftarrow \text{compare}(\theta, \theta^*)$
  if $z > \alpha$ **then**
    $\theta^* \leftarrow \theta$
  end
  $\theta' \leftarrow \beta\theta' + (1-\beta)\theta$
end
return θ^*

Figure 8: A troubling algorithm.

3.3 Self Play and Q-Function Learning

We train a graph neural network to predict the Q functions for each player in a given state. This is accomplished via a method similar to AlphaGo [4] in which the agent plays against earlier versions of its Q function. The Q function is trained to predict the probability of each player winning from a terminal state, where the probabilities are obtained from Monte-Carlo rollouts of the Catan simulator. Fro non-terminal states, we minimize squared Bellman loss.

3.4 MCTS

We then use Monte-Carlo tree search (MCTS) to search for optimal initial placements. When the tree reaches a terminal node (after 8 placements), the value of a node for each player can be computed by rolling out several Catan games from that initial placement. After expanding several thousand nodes, we get a reasonable guess for the settlement placement.

Overall we find that the combination of MCTS and Q-learning outperforms random placements, as well as a heuristically designed placement policy.

Method	Win Rate Against MCTS + QF	Win Rate of MCTS + QF (ours)
MCTS	47.5 %	52.5 %
QF	41.2 %	58.8 %
Heuristic	34.4 %	65.6 %
Random	8.8 %	91.2 %

Figure 9: comparison against other methods.

4 Results

In order to evaluate the efficacy of our method, we seek to answer the following questions:

1. *Can* you build a bot to play Catan?
2. *Should* you build a bot to play Catan?
3. *Did* we build a bot to play Catan?

4.1 Can you build a bot to play Catan?

You sure can.

4.2 Should you build a bot to play Catan?

Depends on who you ask. I beat my friends with the bot once and they said they won't play Catan with me again. Our advisors asked why we wasted lab compute on this. So overall probably yes.

4.3 Did you build a bot to play Catan?

We sure did.

References

[1] Quentin Gendre and Tomoyuki Kaneko. *Playing catan with cross-dimensional neural network*. (accessed: 10.07.2022).

[2] Guillaume Chaslot Istvan Szita and Pieter Spronck. *Monte-carlo tree search in settlers of catan.* (accessed: 10.07.2022).

[3] Michael Pfeiffer. *Reinforcement learning of strategies for settlers of catan.* (accessed: 10.07.2022).

[4] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *nature* 529.7587 (2016), pp. 484–489.

[5] Me. I said this.

Code Golfing for Characters (not Bytes)

Russell Schwartz

March 15, 2023

Abstract

Code golf challenges are usually judged by measuring the length of the source code in bytes. However, certain venues instead count the number of characters. If a variable-length character encoding like UTF-8 is used, and non-ASCII-range characters are allowed, it becomes possible to pack more information into the same number of characters by using high-value code points. We propose a general method for Python code golf that involves compressing source code into a dense string of unicode characters which, when packaged along with a decoder, is (hopefully) shorter than the original.

Introduction

In regular golf, players try to complete a hole in as few strokes of the club as possible. In code golf, programmers try to complete a coding challenge (also called a "hole") in as few *key* strokes as possible. In other words, the challenge is the write the shortest possible computer program that performs some task in an agreed upon language. Holes are usually judged by measuring the length of the source code in bytes. However, certain venues (e.g. `https://code.golf/`) instead count the number of characters. If a variable-length character encoding like UTF-8 is used, and non-ASCII-range characters are allowed, it becomes possible to pack more information into the same number of characters by using high-value code points.

Source code generally consists only of ASCII characters, whose values range from 0 to 127 (giving 8 bits of information per character). Unicode ordinals on the other hand range all the way from 0 to 1,112,063 (giving 20 bits of information per character). Thus, a unicode string has the potential to be over twice as information-dense as an ASCII string. We present a method for losslessly compressing python source code into this denser format while remaining executable.

Methodology

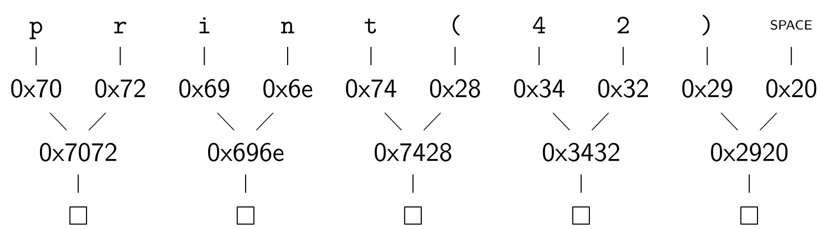
We convert an ASCII string into a unicode string by merging pairs of consecutive ASCII chars into a single unicode char. Let `a` and `b` represent adjacent ASCII chars. The resulting unicode char's ordinal is given by

$$\mathsf{c} = (\mathsf{a} >> 8) \mid \mathsf{b}$$

effectively setting the high byte to be `a` and the low byte to be `b`. The maximum possible value is

$$(\mathsf{0x7f} >> 8) \mid \mathsf{0x7f} = \mathsf{0x7f7f} = 32639 \leq 1{,}112{,}063$$

so we are comfortably within the bounds of unicode. We are effectively using 16 of the available 20 bits. If our original string has an odd number of characters, we pad it with a space. For example, we have the following conversion for the string `print(42)`. Convincing latex to actually render the resulting characters, which mostly fall in the range of *CJK Unified Ideographs*, is left as an exercise to the reader.



Decompression is straightforward. For each unicode character, split the bytes and convert each back to ASCII. In fact, the decompression algorithm is so simple that it can be written in python using only ~ 50 characters. Finally, we pass the decompressed result to `exec()`, which invokes the interpreter.

Results

The results are quite beautiful. Converting to unicode results in a compression ratio of 2, however we have to include the decoder and the call to `exec()`, so there is a break-even point below which this method actually makes the code longer. The overhead incurred by decoding is 58 chars, so we break even at 168 chars. Beyond that point, it's all gravy.

Example 1: Print all palindromic primes under 1000

```python
# -------- ORIGINAL (236 chars) --------
def is_prime(n):
    for d in range(2, n):
        if n % d == 0:
            return False
    return True

def is_palindrome(n):
    return str(n) == str(n)[::-1]

for n in range(2, 1000):
    if is_prime(n) and is_palindrome(n):
        print(n)
```

```python
# ------- COMPRESSED (124 chars) -------
exec(''.join(chr(ord(c)>>8&255)+chr(ord(c)&255)for c
in"搗映楳彰物浥⡮⤺ਠ†湩渠牡湧攨㈬温㨊††††映渠┠搠㴽‰㨊††††††牥瑵牮⁆慬獥ਠ†牥瑵牮⁔牵攊摥映楳彰慬楮摲潭攨温⤺ਠ†牥瑵牮⁳瑲□瑲⠩嬺㨭ㅝ□瑲⡮⤩嬺㨭ㅝਊ景爠渠楮慮来⠲Ⰰ‱〰〩㨊††楦⁩獟灲業攨温⤠慮搠楳彰慬楮摲潭攨温⤺ਠ†††⁰物湴⡮⤊"))
```

Example 2: Roman to Arabic Numerals Converter

```python
# -------- ORIGINAL (156 chars) --------
import sys
d=dict(zip('IVXLCDM',[1,5,10,50,100,500,1000]))
for a in sys.argv[1:]:print(sum(d[x]*(i<len(a)-1and-
(d[x]<d[a[i+1]])or 1)for i,x in enumerate(a)))
```

```python
# ------- COMPRESSED (138 chars) -------
exec("".join(chr(ord(x)>>8&255)+chr(ord(x)&255)for x
in"業灯瑳□祳□湽楄琨穩瀨✪嘶鱙腹✫㈽□□䤮㔬㄰Ⰵ〬ㄾ〰Ⱶ〰ⰱ〰〾⥝⤊景爠愠楮⁳祳⹡牧瘛ㄺ崺灲楮琨獵洨摛硝⨨椼汥渨愩ⴱ慮搭⡤學崼摛慛椫ㅝ崩潲‱⥦潲⁩Ⱐ硩渠敮畭敲慴攨愩⤩"))
```

Example 3: Brainf*ck Interpreter

```python
# -------- ORIGINAL (205 chars) --------
import sys
for P in sys.argv:
 o="A=[0]*99;p=0";l=0
 for c in P:o+="\n"+" "*l+"p+=1|p-=1|A[p]+=1|A[p]-
=1|print(end=chr(A[p]))|while A[p]:|".split("|")[">
<+-.[]".index(c)];l+=(c in"[]")*(92-ord(c))
 exec(o)
```

```python
# ------- COMPRESSED (162 chars) -------
exec(''.join(chr(ord(c)>>8&255)+chr(ord(c)&255)for c
in"業灯瑳□祳ₒ湯‿渠獹献慲杶㨊⁯㴢䄽嬰崪㤹㭰㴰∻氽《⁦潲⁣⁩渠债漫㴢屮∫∠∪氫≰⬽ㅼ瀭㴱籁孰崫㴱籁孰崭㴱籰物湴⡥湤㵣桲⡁孰崩⥼睨楬攠䅛灝㨼∮獰汩琨≼∩嬢㸼⬭⸮嬝崢⹩湤數⡣⥝㭬⬽⡣⁩渢嬝崢⤪⠹㈭潲搨挩⤊數散⡯⤊"))
```

Conclusion

The results really speak for themselves. Not only is the compressed code shorter, but it is also only a single line! Imagine telling your project manager that you just refactored your 10 KLOC codebase into a single line. You might want to also mention that you just doubled or even tripled the total size on disk, but that's the price we pay for greatness.

G

Sharing, Meet Caring

Screen-Sharing Concurrency

Rose Bohrer[1[0000−0001−5201−9895]]

Worcester Polytechnic Institute, Worcester MA 01609, USA rbohrer@wpi.edu

Abstract. We propose a lesson plan for concurrency which employs the screen-sharing features of video-conferencing software as a metaphor for interaction between concurrent software processes. The lesson plan is appropriate for use in a lesson for undergraduate students in the second year and above, such as in a systems programming or distributed systems course. It is particularly well-suited for online and hybrid classroom settings.

Keywords: Screens · Concurrency · Sharing.

1 Introduction

Concurrency is widely recognized as a key topic in a well-rounded undergraduate computer science curriculum. In the modern era, concurrency is widespread, from preemptive operating systems to web services to multi-core programs running on a single computer. It is unfortunate then that concurrent programming is often regarded as difficult to perform correctly and difficult to teach. This paper contributes nothing to the correctness of concurrent programs, but turns its attention instead to the teaching of concurrency.

The difficulty of teaching concurrency can be explained, in part, by combinatorial explosion: there are exponentially-many potential interleavings of the concurrent programs, which quickly make exhaustive case-by-case analysis impractical for even simple programs. In educational settings, it is impractical to explore every interleaving with students. Instead, teaching and programming both focus on higher-level programming abstractions (e.g., concurrency primitives such as mutexes and semaphores). These abstractions come with their own challenges. Not only do concurrency primitives remain subject to combinatorial explosion, but introducing abstractions introduces a layer of indirection in a student's understanding, such that one can easily forget whether they have explored all possible behaviors or fail to understand which concurrency primitives are truly serving to prevent a concurrency bug.

Traditional education overcomes this challenge with hands-on concurrency demonstrations, to provide students a direct experience of correct and incorrect system behaviors. Because concurrency entails multiple agents acting in concert, it is particularly well-suited to group activities in the classroom.

The rise of online and hybrid education, greatly accelerated by the COVID pandemic, has overturned much educational tradition, presenting both challenges and opportunities. Online instruction can be more accessible for everyone from disabled people to returning students to students with lower socio-economic status. It can allow students to work on their own schedule, even when their schedule is busy. It can allow scales that are unprecedented in traditional classrooms. On the flip side, the rapid shift to online education during COVID has negatively affected learning outcomes. Online lectures can make students struggle to pay attention, especially when attending from a chaotic home environment. Pervasive pandemic anxiety reduces the capacity to form new memories and the loss of in-person social support networks removes an essential accountability mechanism and moral support. These negative impacts make it more essential than ever that instructors use modern online teaching technology to design activities that engage students while fostering social interaction among peers.

This paper proposes a lesson plan for concurrency based on hands-on interaction between students, built on a key observation about the Zoom videoconferencing application: screen-sharing and shared whiteboards constitute a medium for concurrent interaction.

2 Lesson Plan

The lesson consists of a series of group activities performed on Zoom, which first present the idea of concurrency bugs through demonstrations of increasing complexity, then present solutions.

2.1 Presenting the Problem

The problems of concurrency bugs are presented in the following stages.

Single Writer, Multiple Reader The instructor screen-shares the Zoom whiteboard and observes that it is a form of shared memory: every pixel stores digital data which are shared among all participants of the lecture. It is noted that the students are all engaging in an act of concurrency: all students are observing the instructor at the same time as one another. Students are cautioned that concurrency can be home to many bugs, and encouraged to keep an eye out for any bugs as the instructor performs the first activity: writing their name on the whiteboard.

The instructor observes that the exercise succeeded because a very specific mode of concurrency was undertaken: a single person wrote to the shared memory while everyone else read from it. The challenge comes when multiple people write at once.

Multiple Independent Writers To demonstrate the problem of multiple simultaneous writers, all students are invited to write their names on the whiteboard at the same time and share a laugh together as everyone writes on top of one another and produces an incomprehensible ball of scribbles.

Multiple Dependent Writers The instructor observes that concurrency is significantly harder when there are dependencies between the reads and writes of each process, i.e., when they must work together. The students are prompted to work together on a task: count the number of students in the class. To prevent students from cheating by merely looking at the participant list, they are told to show their work. If the students do not start writing anything, the instructor prompts them to use a system such as writing out the number of people who went so far or using tally marks. The bugs of each approach are then observed.

No Writers The concept of deadlock is presented by introducing a new rule: only write something after the person before you has written. Because the instructor (host) is at the top of the participants list, the first student never starts writing anything. After a long pause, the instructor indicates they were waiting on the person before them, then explains the concept of deadlocks occurring when multiple agents are waiting on one another in a cycle.

Dining Philosophers The next exercise emphasizes that more subtle deadlocks are possible. The students act out the Dining Philosophers problem together. A plate is drawn for each student and assigned to them, then a chopstick is drawn to each side. The students are told their mission of collecting two chopsticks at the same time. Depending on the order of operations, the students could succeed in their collective goal of getting one half of the students to complete their chopstick pair, or fail. Whichever happens, the students are instructed to repeat the exercise in a way that gets the opposite result. For example, if the exercise failed, they are now told that odd-numbered students should take chopsticks and even-numbered students should not.

2.2 Presenting the Solutions

The remainder of the lesson presents various strategies for resolving concurrency bugs.

Single Mutex To implement a single global mutex in Zoom, the following rule is instituted: one is only permitted to draw to the whiteboard while they are the one screen-sharing. Because the default is that only one person may share a screen at a time, without interruption, the screen share functions precisely as a mutex.

The Dining Philosophers exercise is repeated with the single global mutex. It results in a correct outcome but proceeds incredibly slowly. A tradeoff is highlighted: excessive concurrency leads to bugs, while insufficient concurrency results in performance loss. The goal then is to be as concurrent as possible without sacrificing correctness.

Multiple Mutexes A generic strategy is presented for working with multiple fine-grained locking: deadlock is avoided so long as locks are acquired and released in accordance with some lock ordering. For the next demonstration, the Dining Philosophers exercise is repeated but numbers are written by each chopstick to indicate their place in the lock order. It is observed that this demonstration proceeds far faster than the single-lock solution.

Data parallelism The attendance-counting exercise is repeated, but with coordination. Each student is told to draw a circle indicated their presence in the classroom, but they are told to arrange the circles in rows like the seats of a physical classroom. The instructor tells the leftmost member of each row to count their row and write the final count on the side. A single student is elected to wait until all row counts are received and compute a final sum.

The algorithm's correctness is explained: it exploits data parallelism to sum independent rows simultaneously. The fact that the data are independent is represented visually by the fact they take up different physical locations[1].

3 Barriers to Implementation

There are several barriers to implementation:

- Some of the exercises are designed to demonstrate the slow speed of sequential algorithms. On large class sizes, these algorithms may become so slow as to hinder learning rather than aid it. In these cases, students could be split into multiple breakout rooms to reduce group size.
- Some of the exercises require non-trivial setup, such as drawing spaces for every student. The author is not aware of any way to copy image files onto the whiteboard, which complicates preparation. The low-complexity solution is to assign a teaching assistant the menial work of drawing the setup for each activity, perhaps even in advance. An equally evil yet somehow more ethical solution is to exploit the capacity for typing text on the whiteboard. The instructor could, in principle, develop a custom font face whose glyphs contain the desired imagery for the exercises and type a carefully-engineered text to produce the desired image. At this writing, font size appears limited, in which case the desired image may need to be partitioned across a large number of glyphs. Though this approach is labor-intensive, it need only be performed once.
- Because the people in charge have more sense than me, I am not scheduled to teach any course in which this lesson would be relevant. Thus its implementation require at least one person read this paper. A surefire approach has been employed to this end: presenting the work at the most prestigious of computer science conferences.

[1] The parallel version of the attendance-counting exercise is taken from an in-person exercise in an offering of 15-150 taught by Dan Licata.

4 Future Work

Future work can be pursued along several directions, the most evident of which are exploiting additional Zoom features and implementing a wider range of concurrency concepts.

4.1 Zoom Features

The proposed exercises operate under the assumption that an ongoing screen share cannot be interrupted by a new screen share. Though this assumption is a reasonable model of the default Zoom settings, Zoom can also be configured to either (1) allow the host to interrupt an existing screen share or (2) allow any participant to interrupt an existing screen share. As the name suggests, interruption would be a powerful principle for exploration of concepts such as preemptive scheduling and even perhaps work-stealing scheduling.

The proposed exercises moreover assume that only a single screen share can be active at a time. Zoom can be configured to allow multiple screen shares at a time, which enables complex patterns of concurrent interaction. This permits a single reader to consume streaming data from multiple writers simultaneously and even enables multiple consumers of the same data to select which subset of the data (section of the screen) they wish to view. Immediate applications are clear, but non-immediate applications are potential.

A privacy feature of Zoom ensures that if the boundary of the shared window is overlapped by some other window, that portion is grayed out on the viewer's screen. This mechanism can be used to implement concurrent processes wherein some data are local to each agent while other data are shared.

The Zoom Chat window allows both broadcast and point-to-point communication, a power concurrency mechanism.

4.2 Concurrency Concepts

The present study primarily focuses on mutexes as a synchronization primitive through which concepts such as race conditions, deadlocks, lock ordering, and performance tradeoffs can be explored. This leaves much room for future work.

Firstly, additional concepts could be explored, including message-passing concurrency, scheduling, lock-free algorithms, and priority inversion.

Secondly, implementations of additional synchronization primitives might be pursued, such as semaphores, condition variables, and readers-writer locks.

5 Conclusion

College classes are often boring, and online education is an excellent way to make them even more boring. We should sometimes attempt to make them less boring. This article is a humble attempt to do so.

Acknowledgement and Call to Action

Assistant professor life is busy, but there's a reason I still made the time to write a SIGBOVIK paper this year. It's because when I changed my name, the SIGBOVIK editors were far more thorough and helpful in updating my old publications than in any of the half-dozen publishers I contacted regarding my name change.

Name change processes have improved somewhat in recent years, due in large part to the advocacy of the Trans Name Change Policy Working Group and the willingness of the Committee on Publication Ethics to issue relevant guidelines. However, much remains to be done. At the time of writing, no major publisher corrects names in citations (only SIGBOVIK does), and many take excessively long to perform corrections, with the ACM in particular routinely taking over half a year for small changes. CMU was particularly extreme. When I updated the name on my dissertation, they required my court order (which itself takes months and costs hundreds of dollars) and attempting to require the written approval of a transphobe before finally backing down when pushed. I wanted to contribute to making this SIGBOVIK a fun one because the unpaid volunteers running it have been more supportive than multi-billion-dollar publishers, professional organizations, and institutions that profit from my work.

At the same time, I wish to close with a call to action for cisgender academics. Name changes may have been annoying and imperfect, but the truth is that they're nothing compared to the daily dangers of being a transgender person in this country. In 2023, pundits publicly call for our complete eradication from public life to loud applause and then get invited to speak on college campuses. As I write, Oklahoma is on track to ban insurance coverage of all transition care, and trans people across the country wonder: when this legislation goes national, where will we have left to run? For many people, the answer will be nowhere.

I have been harassed in professional setting. I have been called slurs on the street in a supposedly progressive city for the crime of commuting to campus. But I call myself lucky because I even made it this far. Few trans academics make it to the faculty stage. While we may have to trudge through being the only one who looks like us in the room, the effect is even greater on trans students, who are left with zero advocates from within their community among the leadership of their community. It is no wonder then that trans students face all these problems tenfold, with the attendant impacts on everything from mental health to academic and life success.

So the next time the trans person next to you asks you stand up for them, do it.

Fair Division of a Single Indivisible Object

Gabriel Chuang

Abstract

Fair division is the problem of assigning resources to agents in a way that satisfies some notion of fairness. In this paper, we present an algorithm for fair division of a single indivisible object.

1 Algorithm

Algorithm 1

Let X be the object which is to be divided.
Give X to the author of this paper.

Stretch Goals: Gamification of SLINKYs

Geoff Barnes
Google
geoffb@google.com

William Gunther
Google
wgunther@google.com

Brian Kell
Duolingo
brian.kell@duolingo.com

SIGBOVIK '23
Carnegie Mellon University
April 0, 2023

Abstract

Gamification has been successfully applied in many fields, including learning, crowdsourcing, and physical exercise. In this paper we demonstrate Stretch Goals, a novel application of the principles of gamification to the operation of a helical spring toy. We project that this yields increased engagement and decreased boredom, positioning helical spring toys squarely in the center of the digital entertainment landscape.

1 Introduction

Gamification is the application of game design and gameplay elements, such as competition, challenges, and rewards, in non-game contexts in order to enhance enjoyment and engagement. It has been successfully applied in many fields, including learning, crowdsourcing, and physical exercise. In this paper, we apply the principles of gamification in a previously overlooked domain, namely, the operation of a helical spring toy.

The original helical spring toy, Slinky, was invented by Richard T. James in 1943 and was first sold in 1945. It was an immediate hit, selling out its first production run of 400 units in just 90 minutes. In its first 60 years, about 300 million units were sold. Today helical spring toys are also manufactured and marketed under several competing names, including Sproing!, Giant Springy, and Coil Springy. They are produced in both metal and plastic, in a variety of colors, and in shapes other than the traditional circular form, such as stars and squares.

In order to avoid reference to any particular brand name, in this paper we shall refer to a helical spring toy as a Spring-LIke Nonlinear Kinetic toY, or SLINKY.

Today's youth are too absorbed in their electronic phones and tablets to have time for such analog toys. To rekindle interest, we seek to gamify the experience of operating SLINKYs in order to make them more fun.

Figure 1: The Stretch Goals splash screen.

2 Methodology

SLINKYs have many modes of operation. We chose to focus our preliminary work on the task of descending flights of stairs. The obvious metric to drive gamification is the number of steps descended, and so a method of counting steps is needed.

We solved this problem by connecting a Bluetooth pedometer to a SLINKY. This transmits a step count to a Web app, which then sends the data to a backend service. The backend manages user accounts, total step counts, leaderboards, streaks, and so on. The frontend presents this data to users, along with challenges. We call this app Stretch Goals. Figure 1 shows the splash screen.

2.1 Hardware

We obtained a Slinky® brand SLINKY and a Puck.js Bluetooth sensor. We tested various methods of attachment, including adhesive tape and suspending a foam circle in the center of the SLINKY, which did not achieve the desired results. We found that a bent paper clip worked pretty well.

After a few test slinks, the Bluetooth sensor fell apart. Upon closer examination, we learned that the jostling forces had broken the battery clip off the main circuit board. Fortunately, all that was required for complete repair was some very clumsy soldering and a small zip tie for reinforcement.

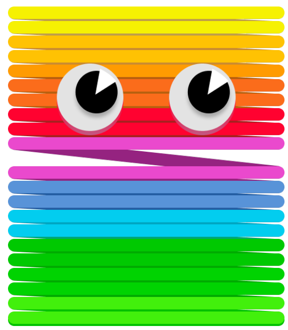


Figure 2: Slinko, the Stretch Goals mascot.

2.2 Challenges

The main focus of Stretch Goals are the challenges. When a user selects a challenge, they are presented with a task such as "Descend 14 steps!" The user can then send an appropriately Bluetooth-connected SLINKY down a flight of stairs to complete the challenge. Multiple attempts will be accepted. When the user completes the challenge, they are rewarded with positive audiovisual stimuli, and their steps from the challenge are added to their total step count.

2.3 Daily goal

Many eminent authorities recommend a daily step count goal of 10,000 steps, so we have set this as the daily goal in Stretch Goals. We admit we were surprised by this number, which seems somewhat high, but we are not in a position to argue with experts. According to 10Best et al. [1], this daily goal can be achieved by sending a SLINKY down thirty Big Bens, or six Eiffel Towers plus one standard flight of house stairs, or three Burj Khalifas plus six Christ the Redeemers. The Niesen mountain in the Swiss Alps has the longest stairway in the world, at 11,674 steps, so a typical Swiss child will just need to perform one SLINKYing per day.

2.4 Levels and leagues

Levels and leagues are a common feature of gamification apps, often with a metaphor such as climbing a ladder or reaching a higher league. However, this is clearly the wrong direction for a SLINKY app, so in Stretch Goals the levels go the opposite way. When a user first registers for Stretch Goals, their beginning level is the top of Mount Everest, and the levels go downward from there.

Leagues in Stretch Goals are closely tied to levels. When a user's SLINKY has descended a total of three miles, i.e., one league, the user is promoted to the next lower level. Mount Everest, the first level, has an elevation of 29,032 feet, or approximately two leagues. Consequently, the second level is Mont Blanc, with an elevation of 15,774 feet, approximately one league; and the third level is the ground.

Below sea level, following Verne [2], a league becomes three nautical miles. The next two levels are RMS *Titanic* and the Mariana Trench. Later levels include the Mohorovičić discontinuity (level 10) and the diamond league (level 30). Verne [3] provides further details.

2.5 Stories

As another form of engagement, Stretch Goals offers SLINKY-themed stories, including "Slinko's first step" and "Jack and Jill fall down the hill." In these stories, two or more characters engage in captivating dialogue such as "I'm going to go down ten steps!" and "I'm going to go down twelve steps!" The user must send a SLINKY down the corresponding number of steps in order to progress the storyline.

2.6 Stretch+

We recognize that it is not always possible to play with a physical SLINKY. For example, it would be difficult to make progress in the app while flying in economy class, while bedridden due to illness, or while living in a yurt on the vast treeless steppes of Mongolia. It would be demotivating for a user to lose their streak just because they were temporarily unable to meet their daily step goal (or even their steppe goal) by sending a real SLINKY down a flight of stairs.

In order to accommodate users in such situations, a paid version of Stretch Goals, called Stretch+, offers a virtual SLINKY feature. With the virtual SLINKY, a user can achieve their daily step goal purely in the Stretch Goals app. For the initial prototype, the virtual SLINKY feature is simply a playlist of SLINKY videos on YouTube, but we expect that this feature can be expanded in the future into a fully immersive SLINKYing experience. For just a bit of extra money, a user won't have to play with a real SLINKY at all!

3 Analysis

Due to the unforeseeable soonness of the SIGBOVIK conference, we did not have a working prototype of the Stretch Goals app by the paper deadline. However, based on highly scientific methods, we were able to estimate the decrease in boredom provided by Stretch Goals, illustrated in Figure 3.

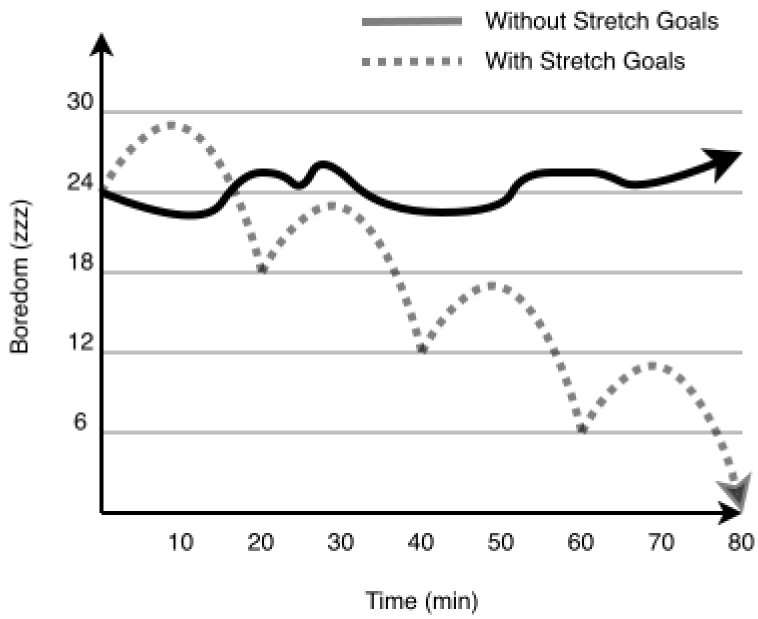


Figure 3: Projected decrease of boredom with Stretch Goals.

4 Conclusions

We presented Stretch Goals, a novel application of gamification to the operation of a helical spring toy, or SLINKY. We highlighted the compelling features of the new app and projected that it increases engagement and decreases boredom. This promises to make a SLINKY a viable entertainment alternative to currently popular digital pursuits such as TikTok, Fortnite, and Yahoo! Backgammon.

References

[1] 10Best Editors for USA Today. How many stairs to the tops of these iconic tall places? `https://www.10best.com/web-stories/how-many-stairs-to-the-tops-of-these-iconic-spots/`.

[2] Verne, J. Vingt mille lieues sous les mers. *Magasin d'éducation et de récréation*, Paris, 1869–70.

[3] Verne, J. *Voyage au centre de la Terre*. J. Hetzel et Cie, Paris, 1864.

H

Post-Quantum, Meet Ergo Propter-Quantum

Voynichese Cryptography

Roger Bacon

Merton College, University of Oxford

Abstract- The manuscript that I wrote has not been deciphered after nearly 800 years, thus would be a prime candidate for RSA cryptography replacement, should quantum computers render 4096-bit keys vulnerable.

Index Terms- Cryptography, Linguistics, Rennaissance, Medieval

I. Introduction

The Rennaissance produced some of the greatest advances in civilization, such as the winged man, the bicycle plane, and modern numerical cryptography.

Ample evidence suggests Voynichese photonics could replace electrical p-n junctions in silicon MOSFETS due to its improved resistance from decryption. Anagrammatic-electron transistor gates can implement arcing bandgaps that transcribe Voychinese characters at faster-than-binary rates, reducing the chance of a side-channel attack and an NoSQL injection.

A. Bits and Pieces together

In this approach we combine all the previous advances in numerical cryptography interspersed with anagrammatic optical lithography parsing several thousand languages from post-Nimrod Mesopotamia. Further research can allow for 3D-stacked anagrammatic transistors to simulate the Fibonaccian Tower of Pisa.

B. Use of Simulation software

There are numbers of software available which can mimic the process in a wind tunnel. A 1:32 scale model of the Tower of Pisa was subjected to 12G of *vis major* in a clean room to simulate the fragmentation of a universal language, which resulted in the last known linguistic schism of biblical proportions.

However, one of the lagging areas of scientific advancement lies in letter-based cryptography. According to Milo Rea Gardner[1], "Further encoding each numeral into Greek letters , Ionian or Doric, until 800 AD when Arabs ended the ciphered numeral step, and mentored Fibonacci to only write 2-term and 3-term series using numerals imported from India."

The Voynichese Cipher is thus a potential candidate as a photonic MOSFET, machine code, Rust language substitute, and full LAMP stack replacement.

Conclusion

Unsolved

Gall Bladdix

(none)

Acknowledgment

This paper would like to thank the wide world internet for its tireless providence of information that led to the production of this paper.

References

1. Pelling, Nick, "SAD NEWS: MARY D'IMPERIO HAS JUST DIED." https://ciphermysteries.com/2020/06/02/sad-news-mary-dimperio-has-just-died#comment-405769

Authors

Bacon, Roger – Frater, Master, Merton College, University of Oxford, Abingdon Rd, Oxford OX1 4LB, United Kingdom

Coupon Code Generation: Saving space with a simple (and insecure) hashing technique

Ben Rosenberg (bar94@cornell.edu)

March 11, 2023

1 Abstract

Have you ever shopped online?[1] What about shopping in real life?[2] If either one of these applies to you, you may be somewhat familiar with the "coupon", a thing[3] which is used at the time of purchase to reduce the amount purchased by either some fraction (for our less consumption-inclined readers, this is generally a fraction of the form $n\%$, where $0 < n < 100$ such that the application of a coupon for $n\%$-off gives a new price of $P - \frac{n}{100}P$) or a fixed amount (e.g., of the form ¤n wherein ¤ denotes the relevant currency sign for the purchase).

There are many types of these coupons: some apply to specific items; others to an overall purchase; still others, to multiple specific items, perhaps linking (e.g.) a specific bottle of hair product x to body wash product y. This article aims mainly to reduce the space taken up by these inevitably vast numbers of coupons, specifically those meant for use within the realm of online shopping.[4]

In this article we implement a variety of simple mathematic techniques to turn the validation of coupons from a storage-intensive task into a more computationally intensive one, with only severe drawbacks for security. We then discuss the performance of these techniques in various metrics, such as storage, runtime complexity, and security. We also discuss several simple and easily-implemented methods for improving the security of the techniques. Finally, in **Appendix A: Reference Implementation**, we provide an easy-to-follow reference implementation of these techniques, for use in commercial applications.

2 Introduction

To put it simply, we want a way to generate coupon codes (alphanumeric strings, generally of length no less than 4 characters[5]) which works within these constraints:

- The codes should be unique.
- The codes should not be easily guessable (e.g., they should not have a pattern that is easy to guess, like $000, 001, \ldots$).
- The codes should be easily checked without the use of a table to store them.

One method of making sure that things[6] are correct is to use a technique known as hashing, in which you hash something. For the sake of example, in this article we use a simple hash, but this can be changed as necessary in the future.

3 Previous work

Previous work has been done in the area of the intersection of a hash and coupon codes, albeit of a slightly different type. For an example of such research please visit https://420couponcodes.com.

[1] Yes, ordering takeout counts.

[2] When was the last time you actually did this? Really makes you think.

[3] This is the scholarly-agreed-upon verbiage.

[4] There is probably room for improvement in the more physical, IRL- (in-real-life) shopping coupons as well, but due to their already negligible size as slips of paper, and the fact that they can decompose into other matter, their buildup is not perceived as being quite as harmful to the general public as is the buildup of their electronically-stored counterparts.

[5] 4 is a cool number. We can show this inductively as follows.

1. Note that 0 is a cool number. This is the case because you can't divide stuff by 0 and the graph of $\frac{1}{x}$ looks really cool when it gets close to 0. Also the Mayans invented 0 and the Mayans practiced human sacrifice and that's totally awesome. So 0 is a cool number.
2. Realize that if n is a cool number, then $n + 1$ is a cool number, because $n + 1$ is only one more than n. (To further this point, 1 itself is a cool number because 1 times anything is that thing again, e.g. $1 \times x = x$, which is cool because it's like the 1 is doing a magic trick where it disappears and all that's left is the number you multiplied it with. That's cool.) So $n + 1$ is a cool number for all n.
3. Apply step (2) until you get to 4. Cool, right?

[6] Again, the scholarly-agreed-upon term.

4 Implementation

Consider some coupon code $C = c_1c_2c_3 \dots c_n$, wherein $c_1, \dots, c_n$ are the characters which compose C. The simple hash we will use in this article is the hash which sums up the numerical representation of each of these characters, where we define the numerical representation to be a mapping η, and the hash function to be

$$h(C) = \sum_{c \in C} \eta(c).$$

We define η as follows:

$$\eta(c) = \begin{cases} \text{‘A’} & 1 \\ \text{‘B’} & 2 \\ \vdots & \vdots \\ \text{‘Z’} & 26 \\ \text{‘a’} & 27 \\ \vdots & \vdots \\ \text{‘z’} & 52 \\ \text{‘0’} & 53 \\ \vdots & \vdots \\ \text{‘9’} & 62 \end{cases}$$

Astute readers might recognize this to be a perversion of the ASCII character code table to better fit our use case.

An example of the use of this hashing function might be the string “JJJJJJJJJJ”, or `‘J’ × 10` for those of a more laconic persuasion.[7] Since ‘J’ comes out to 10 under our complex hash given above (the proof is left as an exercise for the reader), this string hashes to $10 + \cdots + 10 = 10 \times 10 = 10^2 = 100$.

Now this is well and good, but how do we ensure that all strings we generate are equal under the hash? Is there even a way?[8]

Yes, indeed! This is a classic application of the balls and bins/stars and bars/balls and dividers/stars and dividers/stars and stripes/stripes and wipes/balls and wipes/wipe your balls problem, which is commonly taught in combinatorics. Say for instance that you have some 10 balls and want to put them into 3 bins. Note that you can divide these balls into groups using a grand total of $3 - 1$, also known as 2, dividers, like so:

$$\bullet \mid \bullet\bullet\bullet \mid \bullet\bullet\bullet\bullet\bullet\bullet$$

In this example, we have divided up the balls into a group of 1, a group of 3, and a group of 6. To check our work, we might add our numbers back together:

$$1 + 3 + 6 = 10$$

This technique is clearly applicable to our problem.

For the less convinced readers, it may be useful to specify that in this example, 10 is the *result* of the hashing function to which we want all of our coupons to hash, and our choice of `[1, 3, 6]` is equivalent to, under our mapping η, `ACF`.

Basically, we can split whatever number we want to add to into groups of numbers by splitting it up into $\bullet$ and then splitting those $\bullet$ into groups of $\bullet$ delineated by $|$. So we can simply choose where to put these dividers and the number of balls in each stripe or whatever corresponds to the η-defined character we use. Truly remarkable.

4.1 How do I implement this? This looks complicated and my boss says I only have until Friday to implement this and I don’t want to get put on a Performance Improvement Plan please help

The solution to your quandary is that we use math. More specifically, we just generate $n - 1$ random numbers (recall that $|C| = n$) in the range $(1, |\text{dom}(\eta)| = 62) \cap \mathbb{Z}$ which will correspond to the locations at which we place our dividers. We can then sort these numbers, and calculate the contents of the code from them. Let the abovementioned generated numbers be given by $\mathcal{X} = [\![x_1, x_2, \dots, x_{n-1}]\!]$, where $x_1 \leq x_2 \leq \cdots \leq x_{n-1}$, and let $\mathbb{AA}$ be the magic number to which we desire our η-map of C to add.

Since our numbers have been sorted, and since all our numbers x are within $(1, 62) \cap \mathbb{Z}$, we know that $\Delta(\mathcal{X})_{\mathbb{AA}} = [\![x_1, x_2 - x_1, \dots, x_{n-1} - x_{n-2}, \mathbb{AA} - x_{n-1}]\!]$ will be exclusively positive integers in $(0, 61) \cap \mathbb{Z}$. A simple transformation $x \mapsto x + 1$ gives us the domain of η^{-1}; namely, $(1, 62) \cap \mathbb{Z}$. These numbers act as the groups of $\bullet$ seen in the previous section, and correspond to the actual characters of the coupon code being generated.

[7] Other equivalently convenient ways to write the given string might be `‘J’ ^ 10`, or `String.make 10 ‘J’`, or `‘J’ + ‘J’ + ‘J’ + ‘J’ + ‘J’ + ‘J’ + ‘J’ + ‘J’ + ‘J’ + ‘J’`, or `‘’.join(map(chr, [74]*10))`, or

$$f(c) := \texttt{chr(c)};$$
$$g(\ell_{\text{init}}, n, h) := \begin{cases} \ell_{\text{init}} & n = 0 \\ h(g(\ell_{\text{init}}, n-1, h)) & n > 0; \end{cases}$$
$$\lambda\left(s \mapsto s \circ f(\lfloor \pi^4 \rfloor - 2 \cdot \lceil \pi^2 \rceil - \lfloor \pi \rfloor)\right) \triangleright g(\epsilon, \lceil \pi^2 \rceil)$$

.

[8] Read on to find out!

It is important to note here that this mapping $(x \mapsto x+1)$ is an unfortunate result of the 1-indexing strategy employed in η. This strategy was only employed because the language used for the implementation (see **Appendix A: Reference Implementation**) is 1-indexed.

Denote the string concatenation operation by $\circ$. Then, more formally, we can write our coupon code C as follows:

$$C = \bigcirc_{\delta \in \Delta(\mathcal{X})} \eta^{-1}(\delta + 1) \tag{1}$$

So now, we can generate a coupon code using the following steps:

1. Generate $n-1$ random numbers $x_1, \ldots, x_{n-1}$ such that $1 \leq x_i \leq 62$
2. Convert this set of numbers into a coupon string satisfying the hash function using (1)

And we can then check that it works by:

1. Converting these characters back into numbers using η
2. Adding them up using your fingers[9]
3. Checking if the result is the equal to ₳

For readers desiring a swifter and less symbolic explanation, **Appendix A: Reference Implementation** is provided.

4.2 What is the value of ₳?

Throughout this section we have described the value ₳ as though we are familiar with it. Unfortunately, that is not the case. But we can become more familiar with it by doing some additional calculations.

Let's figure out what ₳ should be. We know that ₳ should be the sum of the character code values of $c \in C$. The sum of all the values of $\eta(c)$ for each $c \in C$ can be expressed as follows:

$$\begin{aligned}
\text{₳} &= \sum_{c \in C} \eta(c) \\
&= \sum_{\delta \in \Delta(\mathcal{X})} (\delta + 1) \\
&= |\Delta(\mathcal{X})| + \sum_{\delta \in \Delta(\mathcal{X})} (\delta) \\
&= |C| + \big(x_1 + (x_2 - x_1) \\
&\quad + \cdots + (x_{n-1} - x_{n-2}) + (\text{₳} - x_{n-1})\big) \\
\text{₳} &= |C| + \text{₳}
\end{aligned}$$

Uh... something's wrong here. It looks like our addition of 1 in the middle is throwing us off. Why were we doing that again? Oh, right, the 1-indexing. Well, it looks like we need to make our value larger, by about $|C|$.

...Okay, what should the original starting value be, though? Hm...

Okay I think it should be 62? We subtracted x_{n-1} from it right? And then that value went into η^{-1} or something, so it needs to be in the range of η. Which means that it needs to be no less than 1 and no greater than 62. So $1 \leq \text{₳} - x_{n-1} \leq 62$. But since $0 \leq x_{n-1} \leq 61$ we know that $1 \leq \text{₳} - x_{n-1} \leq 62 \Rightarrow (1, 62) \leq \text{₳} \leq (62, 123)$...? Okay, I'm confused.

Okay, how about this. I'll keep writing code and eventually it'll work. I'll just keep tweaking things until it looks kind of right and the values stop behaving erratically. We're using this in production, right? But we'll do some testing in UAT or something before that. Okay. If there are any real issues, someone will figure it out by the time this goes live. Yeah.

Wait, this is going live tomorrow?

5 Evaluation

Of course, this method is not exactly the most robust to attacks. Finding collisions with this hash function is easier than getting a paper accepted by SIGBOVIK, provided you know η, so it is not recommended that you use this idea. (Note that this becomes slightly harder without advance knowledge of the hashing scheme, combined with some of the methods proposed in section **6.1: Avoiding suspicion**.)[10]

That said, it does kind of work for its intended purpose, which is cutting down on space usage. If you had a lot of codes to generate, you now don't have to store them anymore, as you can check whether they are valid in real time!

To better illustrate this performance increase, we compare our method to some currently-used methods.

5.1 Comparison to alternative methods

See Figure 1.

[9]For people without fingers, Google Assistant can do addition using speech to text.

[10]Actually, even without η, this is really easy if you just know the way the hashing function works. Assuming you have access to one coupon, you already know how long the coupons are, so even if you don't know η you can generate a coupon of the given length for each possible value of $\sum_{c \in C} \eta(c)$ and you'll stumble upon the hash value eventually, allowing you to generate as many valid coupons as you want. (This is the *actual* reason that this method is so insecure.) For more information on this see Birthday Attack.

Method	Storage	Runtime	Security
Database or other similar methods	$O(n)$	$O(1)$ "amortized" (whatever that means)	Generally accepted to be "good"
Intentional Hash Collisions (this paper)	$O(1)$	Actually $O(1)$	Could be worse (probably)

Figure 1: Comparison to alternative methods

6 Improvements

6.1 Avoiding suspicion

We present several possible methods to avoid consumer suspicion:

1. Scramble your mapping η.
   - The mapping in our example given by η is very predictable, as it follows a somewhat conventional alphanumeric order. To throw potential dealscroungers off your path, consider randomizing your mapping as shown in **Appendix A: Reference Implementation**.
   - In fact, this randomization may make it significantly more difficult to check whether the coupons themselves follow any type of pattern. There are $\text{nPr}(61, 61) = \frac{61!}{(61-61)!} = 61! \approx 5.076 \times 10^{83}$ different permutations of our 61-element η given above, so an attacker trying to check all possible permutations by brute force will have to check more permutations than there are atoms in the universe! (This is a good line to use on your manager when he asks how secure this is.)
2. If it is possible that others can view the source code of your checking algorithm (e.g. if, for some unfathomable reason, you make the algorithm execute client-side), make sure you minify or obscurify the code so that it becomes unreadable.[11] There are actually two bonus features to this: one, you save even *more* space, and two, your boss will have a harder time firing you when looking at your code (this is important).
3. Rate limit your coupon service?

6.2 Single-use coupons

Lastly, and most unfortunately, is the matter of preventing the multiple-time use of one-use coupons (without which extensions such as Honey might accumulate a database of coupons and notice the trend). The only way to really prevent the use of the same key multiple times is to keep track of what has and hasn't been used so far... which means saving keys rather than simply calculating them in real time and stuff.

This kind of defeats the entire purpose of our system. The best way to deal with this is likely to just turn these codes into base-62 numbers and store them in a hash table, but that wouldn't be fun. The author recommends the use of a trie-like data structure in which there are 62 children of each node (besides leaves), corresponding to adding a character to the end of a continuing word (e.g., traversing the trie starting at the root with the path 'a' $\rightarrow$ 'b' $\rightarrow$ 'c' would give the string "abc"). This might marginally cut down on repetition in the first few characters used but, of course, in the short run will be much worse than a simple set. (It is a good thought exercise, though – and will probably be useful to remember when you start interviewing again after losing your job for implementing this!) Note also that in order to gain benefits of any kind the trie should be built in real time, rather than building an entire trie wherein existence is defined via some node attribute rather than a node's existence.

7 Conclusion

This may not be the best solution to your problem. ~~This article was sponsored by Honey. Honey is a free browser add on available on Google, Opera, Firefox, Safari, if it's a browser it has Honey. Honey automatically saves you money when you checkout on sites Like Amazon. Papa John's. Kohl's. Wherever you shop it's a good chance that Honey can save you money. All you have to do when you're checking out at these major sites click that little orange button and it will scan the entire internet And find discount codes for you. It takes two clicks to install Honey. Now anytime you checkout Honey will scan the entire internet and find coupon codes for you. If there is a coupon code they will find it, and if there's not a coupon code you can rest assured that you are getting the best price possible and there literally is not one available on the internet. If you install Honey right now you can save like 50 to 100 dollars on your Christmas shopping, doing nothing. There's literally no reason not to install Honey, it takes two clicks, 10 million people use it, 100,000 5 star reviews, unless you hate money you should install Honey. If you want to install it just go to joinhoney.com/mrbeast, that's joinhoney.com/mrbeast.~~

[11] That is to say, *more* unreadable than it already probably is.

References

1. In marketing, a coupon is a ticket or document that can be redeemed for a financial discount or rebate when purchasing a product. Customarily, coupons are issued by manufacturers of consumer packaged goods or by retailers, to be used in retail stores as a part of sales promotions. They are often widely distributed through mail, coupon envelopes, magazines, newspapers, the Internet (social media, email newsletter), directly from the retailer, and mobile devices such as cell phones: https://en.m.wikipedia.org/wiki/Coupon

2. Hash codes: https://420couponcodes.com

3. Honey: Honey dot com

4. M̶: https://monopoly.hasbro.com/en-us

5. Mayans practiced human sacrifice: https://scholarblogs.emory.edu/gravematters/2017/02/23/human-sacrifice-in-mayan-culture/

6. Multiplying strings with integers is a thing: https://www.python.org/

7. Performance Improvement Plans: https://www.amazon.com/

8. Funny star bar bin ball thing: https://en.m.wikipedia.org/wiki/Stars_and_bars_(combinatorics)

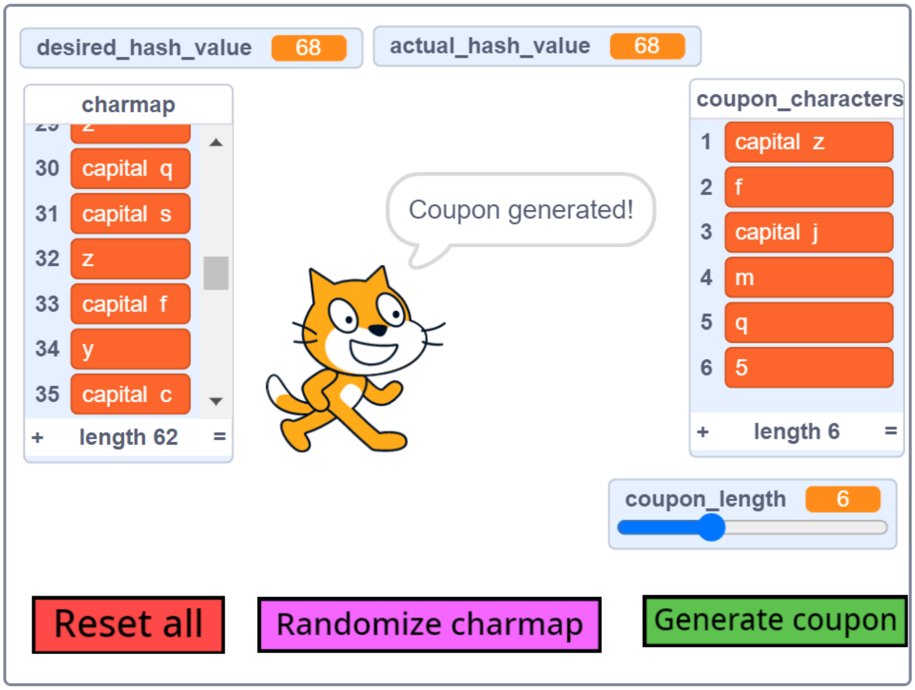
9. There are 10^{83} atoms in the universe: https://astronomy.stackexchange.com/questions/44567/are-there-only-1083-atoms-in-the-universe

10. Google Assistant has speech to text functionality: https://support.google.com/gboard/answer/11197787

11. Birthday Attack: https://en.m.wikipedia.org/wiki/Birthday_attack#Mathematics

Appendix A: Reference Implementation

We present an implementation of the abovegiven methods below. The implementation is bundled with a GUI, for ease of use in the generation of the coupons and randomization of the character mapping. Below is a screenshot of the implemented GUI:



Another one of the limitations of the project is clearly visible in this image; namely, the lack of capital letters. Interestingly, the developers of this computational tool have decided that it is in the best interest of their users to not distinguish between capital and lowercase letters in their comparison functions, so the average unsuspecting user of the 

block will be left scratching their heads for a good 15 minutes trying to figure out why searching for ‘c’ gives them what they would expect for ‘C’, but not vice-versa. This is the reason that we also can’t directly concatenate the characters of the `coupon characters` list, as it would be misleading to clients who might interpret the use of the “capital␣” prefix as necessary.

As can be seen above, the implementation is split into three different functions. In the first, the state is reset. We will call this the “Resetting”. In the second, the character mapping is randomized (“Randomization”). Lastly, in the third, the coupon is generated (“Generation”).

Lastly, and perhaps most importantly, we get the cat guy to say stuff when the buttons are clicked. This is referred to as “getting the cat guy to say stuff when the buttons are clicked”.

Note that the following code rendering was done using the `scratch3` package in CTAN. You can read the documentation here: https://ctan.math.washington.edu/tex-archive/macros/latex/contrib/scratch3/scratch3-fr.pdf. Something to note would be that the documentation is in French. This was a slight problem for the author as the author does not know French. It is also a slight problem for the reader as the if-else block (seen below in the rendered code), instead of displaying “else”, displays “sinon”, which the author surmises is French for “if not”. The author assures readers that despite this inconvenience, the code does in fact remain rather readable, although readers are encouraged to let the author know about their complaints regarding this matter by visiting this website: https://send-complaints.netlify.app.

To download this code for your own use, please click here: Scratch Project (2) (1) (11).sb3

Resetting

```
when this sprite clicked
delete all of [alphabetical charmap v]
delete all of [charmap v]
delete all of [dividers v]
delete all of [coupon characters v]
set [actual hash value v] to (0)
set [coupon length v] to (6)
set [alphanumeric characters v] to (abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz0123456789)
set [desired hash value v] to ((coupon length) + (length of (alphanumeric characters)))
set [index v] to (1)
repeat until <(length of [alphabetical charmap v]) = (length of (alphanumeric characters))>
    if <(index) < (27)> then
        add (join (capital) (letter (index) of (alphanumeric characters))) to [alphabetical charmap v]
        change [index v] by (1)
    sinon
        add (letter (index) of (alphanumeric characters)) to [alphabetical charmap v]
        change [index v] by (1)
    end
end
set [index v] to (1)
repeat until <(index) > (length of [alphabetical characters v])>
    add (item (index) of [alphabetical charmap v]) to [charmap v]
    change [index v] by (1)
end
broadcast [reset v]
```

Randomization

```
when this sprite clicked
delete all of [alphabetical charmap v]
delete all of [charmap v]
delete all of [dividers v]
set [index v] to (1)
repeat until <(length of [alphabetical charmap v]) = (length of (alphanumeric characters))>
    if <(index) < (27)> then
        add (join (capital) (letter (index) of (alphanumeric characters))) to [alphabetical charmap v]
        change [index v] by (1)
    sinon
        add (letter (index) of (alphanumeric characters)) to [alphabetical charmap v]
        change [index v] by (1)
    end
end
repeat until <(length of [alphabetical charmap v]) = (0)>
    set [index v] to (pick random (1) to (length of [alphabetical charmap v]))
    add (item (index) of [alphabetical charmap v]) to [charmap v]
    delete (index) of [alphabetical charmap v]
end
broadcast [randomized v]
```

Generation

```
when this sprite clicked
set [desired hash value v] to ((coupon length) + (length of (alphanumeric characters)))
delete all of [coupon characters v]
delete all of [dividers v]
repeat until <(length of [dividers v]) = ((coupon length) - (1))>
    set [this divider v] to (pick random (1) to (length of [charmap v]))
    if <not <[dividers v] contains (this divider) ?>> then
        add (this divider) to [dividers v]
    end
end
set [i v] to (1)
repeat until <(i) > (length of [dividers v])>
    set [j v] to (1)
    repeat until <(j) > ((length of [dividers v]) - (i))>
        if <(item (i) of [dividers v]) > ((item (i) of [dividers v]) + (1))> then
            set [tmp v] to (item (j) of [dividers v])
            replace item (j) of [dividers v] with (item ((j) + (1)) of [dividers v])
            replace item ((j) + (1)) of [dividers v] with (tmp)
        end
        change [j v] by (1)
    end
    change [i v] by (1)
end
⋮
```

```
⋮
add (item ((item (1) of [dividers v]) + (1)) of [charmap v]) to [coupon characters v]
set [index v] to (2)
repeat until <(index) > (length of [dividers v])>
    add ((item ((item (index) of [dividers v]) + (1)) - (item ((index) - (1)) of [dividers v])) of [charmap v]) to [coupon characters v]
    change [index v] by (1)
end
add (item (((item ((length of [charmap v]) + (1)) of [dividers v]) + (1)) - (item ((index) - (1)) of [dividers v])) of [charmap v]) to [coupon characters v]
set [actual hash value v] to (0)
set [index v] to (1)
repeat until <(index) > (length of [coupon characters v])>
    set [actual hash value v] to ((actual hash value) + (item # of [coupon characters v] in [charmap v]))
    change [index v] by (1)
end
broadcast [generated v]
```

getting the cat guy to say stuff when the buttons are clicked

```
when I receive [generated v]
say [Coupon generated!] for (2) seconds
```

```
when I receive [randomized v]
say [Mapping randomized!] for (2) seconds
```

```
when I receive [reset v]
say [Reset!] for (2) seconds
```

Natural Differential Privacy

Micah Altman
Center for Research in Equitable and Open Scholarship
Massachusetts Institute of Technology
Cambridge, MA
<escience@mit.edu>

Aloni Cohen
Department of Computer Science
University of Chicago
Chicago, IL
<aloni@uchicago.edu>

***Abstract*— We introduce "Natural" differential privacy (NDP) -- which utilizes features of existing hardware architecture to implement differentially private computations. We show that NDP both guarantees strong bounds on privacy loss and constitutes a practical exception to no-free-lunch theorems on privacy. We describe how NDP can be efficiently implemented and how it aligns with recognized privacy principles and frameworks.'**

***Keywords*— differential privacy, physical mechanisms, no free lunch, privacy by design, privacy by default**

I. Introduction

Differential privacy offers provable privacy guarantees [6] but has been criticized as difficult to integrate into existing data production systems and requiring substantial utility loss. [7]

We introduce *natural differential privacy (NDP)* -- a framework for guaranteeing differential privacy for arbitrary computations by leveraging features of existing hardware architectures and natural sources of entropy.

NDP provides all the advantages of "pure" differential privacy as originally formulated in [35], not resorting to any of the myriad relaxed definitions that provide weaker guarantees [36]. NDP provides a worst-case bound on the privacy loss that is substantially better than some high-profile, large-scale commercial implementations of DP.

Moreover, in contrast to existing implementations, NDP provides privacy by default for *all* computations on a platform. Furthermore, NDP is simple and inexpensive to implement at a large scale and requires no practical reduction in utility or performance.

II. Preliminaries & Definition

We consider the setting of a dataset x consisting of n records, where each record is a bitstring of dimension d. We view each row as containing the data of a single individual. Databases x and x' are *neighboring* if they differ in at most one record. A mechanism M is a randomized mapping from datasets to some set of possible outputs Y.

Definition 2.1 (ε-Differential Privacy (ε-DP) [3]). M is *ε-differentially private* if for all neighboring datasets x and x', and for all sets $S \subseteq Y$:

$$\Pr[M(x) \in S] \le e^{\varepsilon} \cdot \Pr[M(x') \in S] \qquad (1)$$

where the probabilities are taken over M's coins.

NDP is defined as any system that integrates DP protections directly in a Von Neumann architecture [8] via hardware implementation using persistent sources of entropy for noise infusion. By construction, every internal computation by an NDP system integrates noise infusion guaranteeing ε-DP. And because DP preserves privacy under postprocessing -- all outputs from the system are thus ε-DP

Implementation: NDP applies to the computation of arbitrary m-bit functions f of the data x, for any m. To evaluate the NDP-version of f, one simply evaluates f on a RAM machine. For best results, the RAM should be operated at or above sea level.

We use as a building block the Randomized Response mechanism. The Randomized Response mechanism is parameterized by a probability $0<p<0.5$, and we denote the corresponding mechanism RR_p. RR_p takes as input a bit $b \in \{0,1\}$, and outputs 1-b with probability p, otherwise outputting b. Results established in [16] provide a formula for the exact equivalence between the probability of randomized response and the epsilon parameter. For any $p < 0.5$, equation 2 expresses this relationship:

$$1-p = \exp(\varepsilon)/(1 + \exp(\varepsilon)) \qquad (2)$$

Privacy parameters: Let $T_{in} > 0$ and $T_{out} > 0$ be the length of time that the input x and output f(x) are stored

Under Review:Sigbovik, JofUR

in RAM over the course of the computation, respectively.[1]

Each computation has a corresponding parameter q that depends on the environment within which the computation is performed. Thus q is the probability of any single bit flip caused by cosmic rays occurring on 1 GB of RAM over the course of 1 day (see Table 1). From q, it is easy to derive the probability that any single bit is flipped in the period T_{in} or T_{out}. Using p and applying equation 1, it is straightforward to solve for ε.

III. Related Work

Current implementations of DP at scale have used artificial (non-natural) DP. Because of the substantial utility tradeoffs that artificial DP often requires -- commercial implementations often use values of epsilon well above 1. Recent large-scale implementations of differential privacy by major corporations (including Google and Apple) have employed effective epsilon levels ranging from dozens to hundreds -- with one major implementation exceeding seven hundred and fifty. [2] [28]

Other variants of DP, such as epsilon-delta DP and concentrated DP have been proposed [4]. However, these variants relax the definition of DP yielding weaker privacy properties. (We refer henceforth to such relaxations as 'artificial.')

Natural sources of entropy for noise diffusion have been studied for over four decades [10]. Their importance in security and privacy has been recognized in related areas:

- Bit flipping induced by radiation or other environmental conditions has been previously used for practical attacks against system security [9,32].
- The importance of high-quality random number generation for all differential privacy methods has recently been recognized. [34] In practice, nearly all implementations rely on pseudo-random sequences seeded from a physical entropy source. The use of physical sources of randomness for direct noise infusion has not been well-examined.
- More recently, the inherent instability of quantum computation has been examined as a theoretical source of protective noise infusion -- although practical implementation remains far off. [33]

IV. Advantages of NDP

Although natural noise infusion has been studied in related work, the use of natural sources directly for differential privacy is novel. The NDP approach offers a number of advantages:

- NDP protects all computations made on a system.
- NDP does not require any relaxation of the formal differential privacy guarantees -- unlike artificial DP variants.
- NDP is simple to implement and inexpensive to deploy.
- NDP provides protections that are substantively equivalent (or better) than the formal guarantees provided by notable commercial implementations -- while maintaining higher utility, substantially reducing implementation cost, and extending protection to a broader range of computations.

Further, DP has additional attractive features:

1. First, NDP encourages privacy by design [15] -- NDP can be integrated into hardware architecture, systems design, and facility deployment, as well as at the application level.
2. Second, NDP provides privacy by default [16] since a floor for protection is provided for all users without requiring any opt-ins [17].
3. Third, NDP aligns well with the widely adopted 'five safes framework' [20]. Specifically, it uses architectural privacy by design to guarantee 'safe outputs.'
4. Fourth, NDP can be implemented either at the time of manufacture or during deployment. This facilitates certification and auditing of secure hardware and facilities.
5. Fifth, NDP provides guaranteed, measurable privacy with zero marginal utility loss --

[1]If the portions of the input (respectively, output) are in RAM for different lengths of time, then we take T_{in} (resp., T_{out} to be the minimum time over every bit of the input (resp., output).]

[2]Estimates are for the effective protection of user information over a month of activity. Because of composition effects, the effective epsilon for protection of a user information in these systems grows geometrically over time. Thus a cumulative epsilons of ten thousands or more is possible for frequent, long-term users of these systems.

theoretical no-free-lunch theorems notwithstanding [5]. [3]

V. Setting Privacy Parameters Within NDP Systems

NDP, when fully integrated into the architecture, operates by default, continuously, and at the hardware level. In modern architectures, which implement random access memory using MOSFET technology, bit-flipping is an ideal mechanism for noise infusion.

At manufacturing time, the infusion of readily available alpha sources into memory chip packaging material requires no increase in manufacturing cost. This approach can be used to ensure a maximum bound for epsilon that is both auditable through inspection and can be verified through hardware certification processes. [4]

More generally, ionizing radiation provides a natural and ubiquitous source for inducing randomized response at the bit level -- acting through the injection of memory faults. Utilizing this physical mechanism for entropy provides true randomness, which provides stronger formal guarantees than pseudorandom number generation [27,34].

Results established in [14] provide a formula for the exact equivalence between the probability of randomized response and the epsilon parameter.

$$p = 1/(1 + \exp(\varepsilon)) \qquad (2)$$

Many sources of radiation are readily available, as shown in Figure 1.

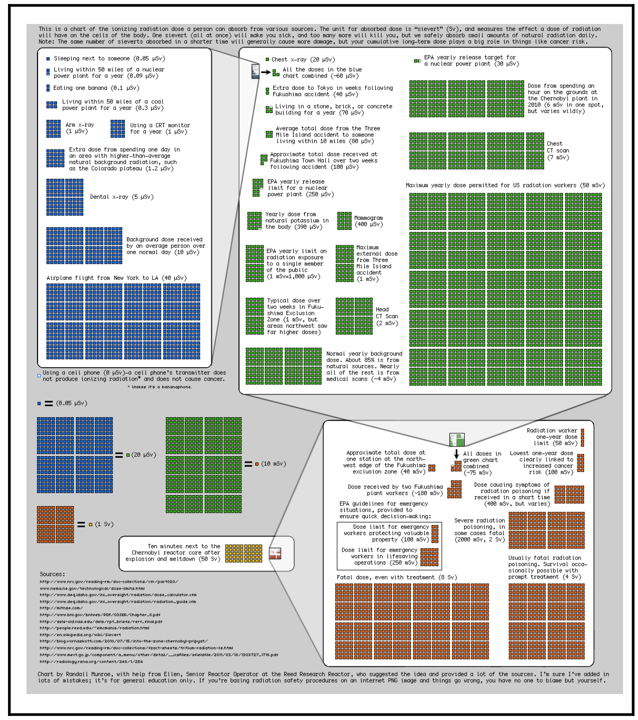


Fig. 1. Comparison of Readily Available Entropy Sources for Setting ε
(Source [2], placed in the public domain)

Post-manufacture, maximum bounds on epsilon can be further reduced at deployment time through the positioning of the hardware. Entire computing facilities may be certified as safe using this method.

Conveniently, cosmic rays cause i.i.d. bit flips at a rate directly related to the level of cosmic radiation (typically measured in micro-sieverts) exposure, which is itself a function of atmospheric density (above sea level) and crustal density (below sea level) at a given altitude. The levels of radiation experienced at specified altitude relative to sea level has been established empirically and fits a geometric distribution within near-earth orbiting distances. Using these empirical results and equation 2, we can derive the effective value of epsilon at exemplar locations, as shown in Table 2.

TABLE I. Privacy Budget Configuration through Altitudinal Adjustment[5]

[3] In theory, theory and practice are the same -- in practice, they differ. See [18,19] for the original formulation by Brewster, and contributions attributed to Berra and Einstein.

[4] Even without changes in manufacturing processes, manufacturers can readily certify that their RAM is not unsafe by design -- in that it does not provide ECC or this feature has been disabled (e.g. at the firmware level).

[5] Derivative-free numerical minimization [26] is used to obtain epsilon corresponding to p, given Equation 2. Bit-level frequency data is provided by [25,29]. Epsilon levels are calculated for protection at the (bit) event-level, which is the most practical unit of protection for streaming systems [24] -- and has been used at scale for public release of large scale Facebook data for scientific research [21.22] To calculate epsilon for other units of protection, it is straightforward to calculate the effective epsilon by using the standard dp

m > sea-level	***Exemplar Location***	***µSv/h***	***Error /GBxDay***	***Max ε***
-3840	Mponeng gold mine	0	0	∞
10	Cambridge, MA	0.06	.2	33.70711
1742	Mount Wilson observatory	0.237	2	31.39832
10000	Jet airplane lower cruising	6	60	27.99537
781000	Iridium Satellite Constellation	60	600	25.69307

Note that relative to baseline, very low values of epsilon can be achieved through altitudinal adjustment. Further note that the level of epsilon provided naturally at sea level is more protective than the level provided by the most notable and largest scale production implementations of differential privacy to date. [13] Finally, in practice, the effective epsilon will be statistically indistinguishable from implementations using a theoretically lower value. [6]

At runtime, noise injection can readily and effectively be achieved by altering the thermal operating environment. [32] Further, in high-density computing deployments, simply reducing the level of external cooling will not only increase protective noise infusion but also reduce electricity usage -- benefitting the global environment. Moreover, various external noise injection tools are available and can provide additional protection without affecting the location or manufacturing process [11].

Finally, given the wide availability of (level-2) hypervisor-based system-level virtual machine technologies, simulation-based noise infusion (aka. synthetic natural differential privacy) can be used to produce any desired level of epsilon with a relatively small decrease in runtime performance [12].

composition formula across the maximum number of shared events in the computation [23]. Even under composition, the effective epsilon remains trivially small relative to the baseline.

[6] Assuming a continuous audit period of one hour, and conventional levels of statistical significance (p=.05) is used, the observed value at sea level will not be statistically distinguishable from a theoretical epsilon of 31.91007. (Zero bits flipped will be observed during that period at either level of epsilon, during at least 95% of audits).

VI. Limitations

On occasion, serious points are best conveyed through humor.[7] This article is intended as a work of (serious) humor. While each of the individual technical assertions in the article is true, catastrophic drawbacks are omitted or glossed over. Thus the substantial benefits claimed for the method, particularly in section IIIB, are parodically misleading.

For example, some of the limitations of the above proposal include the following:

- Use of ε above 1, although frequent in practice, requires caution.
- It is rarely the proper objective of law or public policy to protect the privacy of an 'event.' It is usually a more appropriate policy goal to protect the privacy of a persistent actor -- such as a person, organization, or designated group of people.
- Information about an event or other unit of protection is rarely limited to a single bit. Where measurements of the unit of protection comprise multiple bits, composition will multiplicatively increase the effective ε.
- Randomized response does not generally result in an efficient tradeoff between privacy and utility -- especially as the effective information measured grows large.
- Inducing bit flipping at random in main memory causes unwanted side effects -- such as entirely incorrect results, nonsense output, program failures, and system crashes.
- Thus, there is no market for unreliable RAM, despite the low cost of manufacture.
- Further, introducing ionizing radiation to the deployment site to achieve a meaningful level of protection[8] would risk exposure to lethal amounts of radiation. This potentially violates local regulations, national laws, and international treaties.

These limitations notwithstanding, there is a sense in which cosmic rays formally induce ε-DP. Further, the epsilon values used by some large-scale commercial implementations of DP provide provable worst-case privacy-loss guarantees that are arguably no stronger than those provided by cosmic rays in Cambridge.

[7] See [30] for the foundations of this principle, and [31] for both a modern defense and exemplar of the principle.

[8]

When very high values of epsilon are employed claims that such systems provide formal protection are misleading -- but this does not imply that such systems provide no protection: As techniques for measuring privacy-loss continue to evolve, we may come to understand that some systems are more protective than initially proved. Or these implementations may provide useful protection in particular contexts even if such protections are not formally provable.

Most important, this parody illustrates that neither formal privacy guarantees nor compliance with privacy principles are sufficient for adequate protection. Provable guarantees, such as those provided by differential privacy, have force only when the specific level of protection provided by implementation privacy parameters are meaningful and when the formal unit of protection corresponds to real-world entities with meaningful privacy interests. Moreover, when a system embeds a weak implementation of a protection mechanism at its core, compliance with other privacy principles, such as privacy-by-design, may offer little value.

Acknowledgment

We describe the authors' contributions following a standard taxonomy [1], AC and MA authored the first draft of the manuscript and had overall (ir)responsibility for content and revisions. All authors contributed to the conception of the report (including core ideas and statement of research questions), to the methodology, and to the writing through critical review and commentary. All authors contributed to substantiating the paper's arguments, revision, critical review, and commentary.

We thank Stephen Chong and Kobbi Nissim for their helpful comments on prior drafts.

Appendix: Reproducibility

The following R code is sufficient to replicate the results in table 1:

``{r}

```
#minimum reported failure rate in memory is
.2 per errors GB/day

rate_GBday_wikipedia <- 0.2

# assumed quantity that new data remains
resident in memory for 1 second

newdata_Gb_sec <-  1E9

# p value for randomized response from Wang
et al. 2016
randp <- function(eps) {
  exp(eps)/(1+exp(eps))
}

# predicted bit errors per day
dayerr <- function(eps,
                   flow = newdata_Gb_sec  )
{
  res <- (1-randp(eps))  * flow * 60 * 60 *
24
  res
}

# find epsilon with  5% change of observed
error in an hour
optim(par=8,
fn=function(x){abs(.05-dayerr(eps=x)/24)},
method="Brent", lower=.1, upper=42)

# find epsilon for a given bits/day error
rate
optim(par=8, fn = function(x) {
abs(rate_GBday_wikipedia - dayerr(eps=x))},
method="Brent", lower=.1, upper=42)
```

References

[1] Allen, L., Scott, J., Brand, A., Hlava, M. and Altman, M., 2014. Publishing: Credit where credit is due. Nature, 508(7496), pp.312-313.

[2] Munro, R., 2011 Ration Dose Chart, XKCD. < https://xkcd.com/radiation/ >

[3] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In TCC, pages 265–284, 2006.

[4] Cynthia Dwork and Guy Rothblum. Concentrated differential privacy. CoRR, abs/1603.01887, 2016

[5] Kifer, D. and Machanavajjhala, A., 2011, June. No free lunch in data privacy. In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data (pp. 193-204).

[6] Wood, A., Altman, M., Bembenek, A., Bun, M., Gaboardi, M., Honaker, J., Nissim, K., O'Brien, D.R., Steinke, T. and Vadhan, S., 2018. Differential privacy: A primer for a non-technical audience. Vand. J. Ent. & Tech. L., 21, p.209.

[7] Blanco-Justicia, A., Sanchez, D., Domingo-Ferrer, J. and Muralidhar, K., 2022. A Critical Review on the Use (and Misuse) of Differential Privacy in Machine Learning. arXiv preprint arXiv:2206.04621.

[8] von Neumann, John (1945), First Draft of a Report on the EDVAC

[9] Dinaburg, A., 2011. Bitsquatting: DNS hijacking without exploitation. Proceedings of BlackHat Security.

[10] Ziegler, J. F.; Lanford, W. A. (1979). "Effect of Cosmic Rays on Computer Memories." Science. 206 (4420): 776–788.

[11] Hsueh, Mei-Chen, Timothy K. Tsai, and Ravishankar K. Iyer. "Fault injection techniques and tools." Computer 30, no. 4 (1997): 75-82.

[12] Li, Z., Kihl, M., Lu, Q. and Andersson, J.A., 2017, March. Performance overhead comparison between hypervisor and container based virtualization. In 2017 IEEE 31st International Conference on advanced information networking and applications (AINA) (pp. 955-962). IEEE.

[13] A. Greenberg. 2017. How one of Apple's key privacy safeguards falls short. Wired

[14] Y. Wang, X. Wu, and D. Hu. 2016. Using randomized response for differential privacy preserving data collection. In Proceedings of the EDBT/ICDT 2016 Joint Conference. Bordeaux, France

[15] Cavoukian, Ann. "Privacy by design." (2009).

[16] Willis, L.E., 2014. Why not privacy by default? Berkeley Tech. LJ, 29, p.61.

[17] NISO. 2015. Consensus Principles on Users' Digital Privacy in Library, Publisher, and Software-Provider Systems. ISBN 978-1-937522-70-4

[18] 1882 February, The Yale Literary Magazine, Conducted by the Students of Yale College, Volume 47, Number 5, Portfolio: Theory and Practice by Benjamin Brewster, Quote Page 202, New Haven, Connecticut.

[19] QuoteInvestigator, 2018. "In Theory, Theory and Practice are the Same", https://quoteinvestigator.com/2018/04/14/theory/

[20] Desai, Tanvi; Ritchie, Felix; Welpton, Richard (2016). "Five Safes: designing data access for research" (PDF). Bristol Business School Working Papers in Economics: Footnote 1

[21] King, G., & Persily, N. (2019). A new model for industry—Academic partnerships. PS: Political Science & Politics, 53(4), 703–709.

[22] King, G., & Persily, N. (2020). Unprecedented Facebook URLs Dataset now Available for Academic Research through Social Science One; IQSS, Harvard. < https://socialscience.one/blog/unprecedented-facebook-urls-dataset-now-available-research-through-social-science-one>

[23] Kairouz, P., Oh, S. and Viswanath, P., 2015, June. The composition theorem for differential privacy. in International conference on machine learning (pp. 1376-1385). PMLR.

[24] Lécuyer, M., Spahn, R., Vodrahalli, K., Geambasu, R., & Hsu, D. (2019, October). Privacy accounting and quality control in the sage differentially private ML platform. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles* (pp. 181-195).

[25] Soft Error, 2022, Wikipedia. https://en.wikipedia.org/wiki/Soft_error

[26] Brent, R. (1973) Algorithms for Minimization without Derivatives. Englewood Cliffs N.J.: Prentice-Hall

[27] Vadhan, Salil P. "Pseudorandomness." Foundations and Trends® in Theoretical Computer Science 7, no. 1–3 (2012): 1-336.

[28] Ryan Rogers, Subbu Subramaniam, Sean Peng, David Durfee, Seunghyun Lee, Santosh Kumar Kancha, Shraddha Sahay, and Parvez Ahammad, Linkedin's audience engagements API: A privacy-preserving data analytics system at scale, arXiv:2002.05839, 2020

[29] Enyinna, Paschal Ikenna. "Radiological risk assessment of cosmic radiation at aviation altitudes (a trip from Houston Intercontinental Airport to Lagos International Airport)." Journal of Medical Physics/Association of Medical Physicists of India 41, no. 3 (2016): 205.

[30] Janko, Richard, trans. 1987. Poetics with Tractatus Coislinianus, Reconstruction of Poetics II and the Fragments of the On Poets. By Aristotle. Cambridge: Hackett. ISBN 0-87220-033-7.

[31] Van Stempvoot, Stephen J. et. al., (2022) Brief Of The Onion As Amicus Curiae In Support Of Petitioner, Novak v. City of Parma, Ohio

[32] Govindavajhala, S., & Appel, A. W. (2003, May). Using memory errors to attack a virtual machine. In 2003 Symposium on Security and Privacy, 2003. (pp. 154-165). IEEE.

[33] Zhou, Li, and Mingsheng Ying. "Differential privacy in quantum computation." In 2017 IEEE 30th Computer Security Foundations Symposium (CSF), pp. 249-262. IEEE, 2017.

[34] Garfinkel, Simson L., and Philip Leclerc. "Randomness concerns when deploying differential privacy." In Proceedings of the 19th Workshop on Privacy in the Electronic Society, pp. 73-86. 2020.

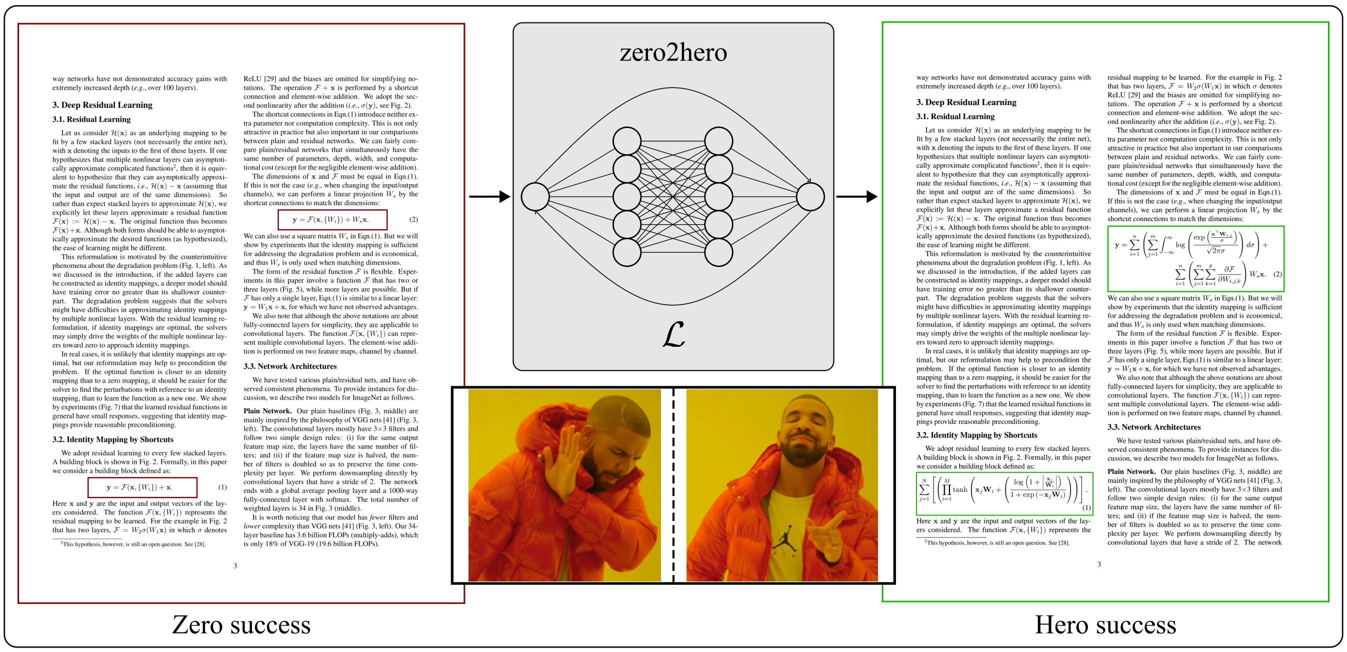
[35] Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In Theory of cryptography, pp. 265–284. Springer, 2006.

[36] Pejó, B. and Desfontaines, D., 2022. Guide to Differential Privacy Modifications: A Taxonomy of Variants and Extensions. Springer Nature.

From Zero to Hero: Convincing with Extremely Complicated Math

Maximilian Weiherer Bernhard Egger
Funky-Amusing-University (FAU) Erlangen-Nürnberg

maximilian.weiherer@fau.de
bernhard.egger@fau.de



We present zero2hero, an innovative system that turns every scientific paper into an award-winning masterpiece. Given the fact that papers solely using notoriously simple math provably lead to failure (top-tier conference rejections and rude reviewers, diminished respect and appreciation from almost everyone, decline in social status, etc.), zero2hero reliably over-complicates equations so that no one, including yourself, is able to understand what's happening or what ever happened. Buckle up [9] and let zero2hero boost your career, now.

Abstract

Becoming a (super) hero is almost every kid's dream. During their sheltered childhood, they do *whatever it takes* to grow up to be one. Work hard, play hard – all day long. But as they're getting older, distractions are more and more likely to occur. They're getting off track. They start discovering what is feared as *simple math*. Finally, they end up as a researcher, writing boring, non-impressive papers all day long because they only rely on simple mathematics. No top-tier conferences, no respect, no groupies. Life's over.

To finally put an end to this tragedy, we propose a fundamentally new algorithm, dubbed zero2hero, that turns every research paper into a scientific masterpiece. Given a LaTeX document containing ridiculously simple math, based on next-generation large language models, our system *automatically over-complicates every single equation* so that no one, including yourself, is able to understand what the hell is going on. Future reviewers will be blown away by the complexity of your equations, immediately leading to acceptance. zero2hero gets you back on track, because *you deserve to be a hero*™. Code leaked at https://github.com/mweiherer/zero2hero.

1. Introduction

Simple math doesn't impress anybody, neither your grandma nor any reviewer. Scientific papers overgrown with ridiculously underwhelming mathematics are deadly boring to read, often dismissed as trivial, and, ultimately, don't cause the urgently needed pain most readers are desperately looking for. Authors of those 'research' papers also frequently complain about not being treated with the necessary respect, which often manifests itself in the fact that simply too many scientists can follow their 'ideas', or, even worse, are able to suggest improvements to the author's 'work'. How dare they!

As if that weren't enough, it recently has been proven (the proof is left as an exercise for the reader) that getting into top-tier conferences like CVRP, ICCV/ECCV, and NeurISP, depends *solely* on the complexity of the used mathematics, simply judged by counting equations xor inspecting notation. As a consequence, authors who wish to publish at those conferences began to maximize the number of equations and the notational complexity in order to satisfy the reviewer's fetish. Popular tricks to make a paper's math look more complex include, for instance, maximizing the occurrences of Greek letters (try to use as many of them as possible already in

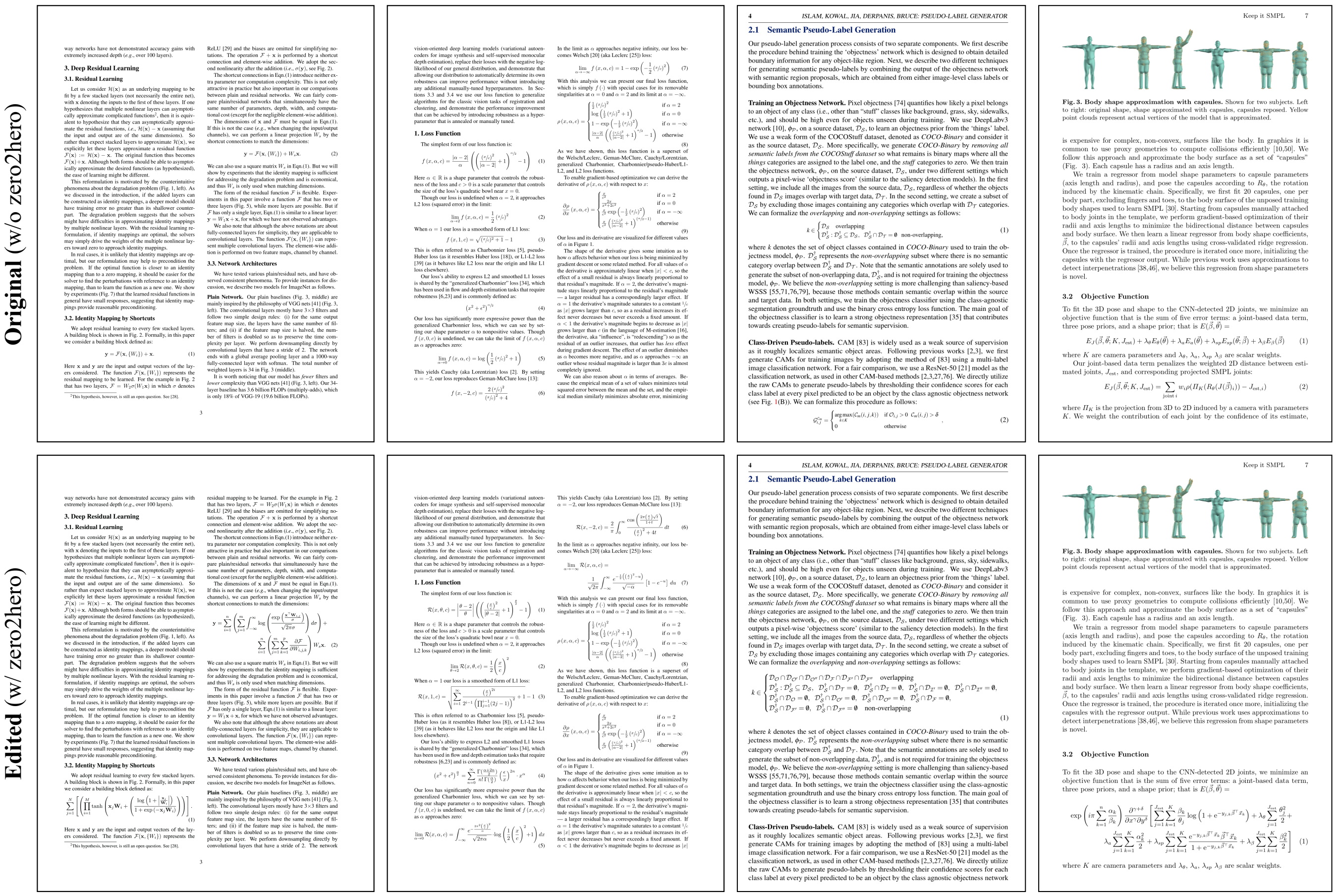


Figure 1. We applied zero2hero to notoriously unsuccessful papers from the last decade and observe overwhelming results. From left to right: ResiNet [6] paper, a paper on adaptive and robust loss functions [1], *Simpler Does It* [7], and some random work by authors who love to keep it SMPL [2]. Original papers are shown in the top row; note the obscenely simple equations. No wonder these papers have disappeared into oblivion. Professionally over-complicated formulas produced with zero2hero are shown in the bottom row. Smell the success.

the title) or adding (random) sums or products, integrals, unnecessary operators or made-up arithmetic symbols, any kind of functions (algebraic, arithmetic, barbaric, trigonometric, etc.; see [8] for further inspiration), and/or physical constants (just to name a few). Although these tricks are considered well-known, quite a significant number of submissions are still getting rejected from top-tier conferences *every* year (which, by the way, causes pain and sorrow across the world). Why the hell is that? How can this even be possible? And, why are we here? With this work, we finally provide complex answers to the big questions[1]: Authors simply must have had (and still have) serious problems complicating their papers.

To finally put an end to this tragedy, we propose a fundamentally new method inspired by [10], dubbed zero2hero, that turns *every* paper into a scientific masterpiece. Based on the latest, next-generation machine-learning techniques, our algorithm reliably over-complicates mathematical equations in a fully automatized way. Just throw in your $\LaTeX$ document, let the machine do the work for you, and et voila, see your paper being accepted at CVRP. No more pain. No more tears. No rejection, no cry! We demonstrate our ground-breaking system, zero2hero, on various notoriously unsuccessful research papers from the last decade, for example, Kaiming He's ResiNet [6] paper, shown in Fig. 1.

[1] We also refer to https://www.amazon.de/-/en/Stephen-Hawking/dp/1473695988. Use code 'SIGBOVIK' to get 100% off.

2. Methods

Our method assumes as input an ordinary $\LaTeX$ document and outputs a second version of that document where simple equations are replaced by extremely complicated(-looking) formulas. It is important to note that we do *not* care about whether formulas are actually complex, we just want them to look extremely difficult. This is because Attention Is All We Want. Clever, huh?

Following a recent trend kicked off by OpenAI, we do not describe our method due to the competitive nature of our ideas. In particular, we intentionally hide any details about the size of our model, the number of parameters, training data, etc. However, we can reveal that the backbone of our algorithm is a recently published, Transformer-based [5] large language model (LLM) which, obviously, transforms equations. We employ the following loss:

$$\mathcal{L} = \sum_{i=1}^{n} \Bigg[-y_i \oint_{\Omega} \left(\zeta\left(\frac{\hat{y}_i}{1-\hat{y}_i}\right) \frac{\partial}{\partial \theta_i} \left(f_i(\theta) \log \frac{\hat{y}_i}{1-\hat{y}_i} \right) \right) d\theta + \frac{1}{2} \sum_{k=1}^{n} \frac{\partial^2}{\partial x_k^2} \left(\sum_{i=1}^{n} y_i \hat{y}_i \frac{\partial \log f_i(\theta)}{\partial x_k} \right) \Bigg], \tag{1}$$

where n is the number of training examples, $\hbar$ is the reduced Planck constant, i.e., $\hbar = h/2\pi$ with h being the Planck constant, and f_i

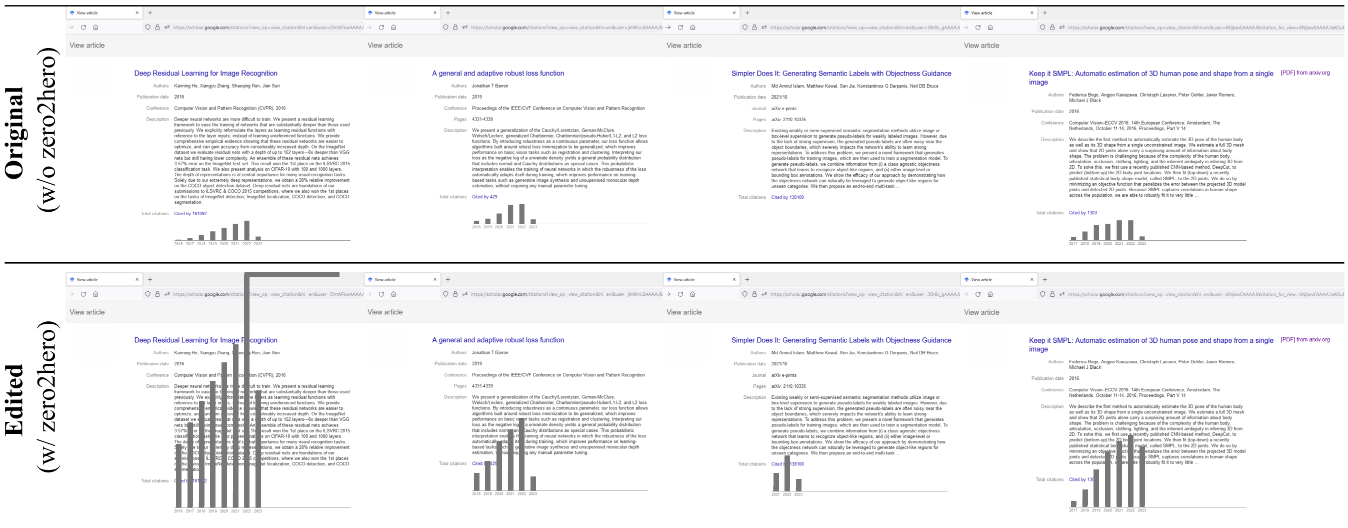


Figure 2. Screenshots of random Google Scholar profiles showing citations of the four investigated papers without using zero2hero (top row) and when zero2hero would have been applied (bottom row) prior to publication. Results are speaking for themselves. Notice how citations of the ResiNet [6] paper (first column) went through the browser bar in 2022. From left to right (same order as in Fig. 1): ResiNet [6] paper, a paper on adaptive and robust loss functions [1], *Simpler Does It* [7], and *Keep It SMPL* [2].

is a secret function transforming parameters θ of the LLM. Please understand that we are not allowed to share any additional details[2].

3. Experiments and Results

Expensive experiments were conducted to validate our method. Specifically, we analyze two different factors: The impact of zero2hero on (i) the number of citations, and (ii) the author's mood and personal situation. All experiments were executed retrospectively. Results were analyzed using openCHEAT [4].

3.1 Setup

To analyze how zero2hero would have influenced factors (i) and (ii) for manuscripts written *before* our method was invented, we randomly collected a bunch of papers from the internet and compare the current impact (as of 2023) to what the paper could have had if zero2hero had been used at the time of publication. But, wait, how can we know the impact a paper could have had?

Turns out to be dead easy! In short, to obtain the impact a paper could have generated if zero2hero had been used, we make use of our institution's high-performance time machine (HPTM) and a theory commonly known as the *many-worlds interpretation*[3] (MWI). The MWI is an absurd interpretation of an absurd physics theory (namely, quantum mechanics), asserting that the universal wavefunction is objectively real and that wave functions can't collapse. This obviously implies that every possible outcome of a decision opens up a new, *parallel universe* (or, world). Given these tools and a paper we want to analyze in our current universe, $\mathcal{U}_C$, at a certain point in time, t, we proceed as follows.

1. Using our HPTM, we travel back in $\mathcal{U}_C$ to the time shortly before the paper was published (we attached zero2hero to the journey). Denote this point in time as t_0.
2. At t_0, we decide to *not* use zero2hero. Note that (due to the MWI) this decision immediately opens a new universe, $\mathcal{U}_N$, in which zero2hero is *automatically* applied.
3. In both universes, $\mathcal{U}_C$ and $\mathcal{U}_N$, we simultaneously publish the paper at time $t_0 + \epsilon$.
4. Lastly, we travel back to where we came from. That was t.

It's important to note that (in the third step) we do not have to switch the universe in order to publish the paper (in practice, we simply open a new terminal and ssh to $\mathcal{U}_N$). As such, we never left our current universe.

We now analyze factors (i) and (ii) in detail.

3.2 Impact on Number of Citations

We start by analyzing how zero2hero would have influenced the number of citations for papers written before our method was invented. To do so, the Internet Explorer (version 8.0.7601.17514IC) was used to access Google Scholar profiles from $\mathcal{U}_N$ at time t, again via an ssh connection. We analyze the same four papers shown in Fig. 1, i.e., [6], [1], [7], and [2]. All papers were written and published between 2016 and 2021.

Some exemplary results can be found in Fig. 2. They are clearly out of this world. In all cases, an application of zero2hero would have increased the number of citations dramatically. Most notably, if the authors from *Simpler Does It: Generating Semantic Labels with Objectness Guidance* [7] would have used zero2hero prior to publication in 2021, they could already have 138,100 citations today! Instead, they have zero citations. Well, seems like simpler doesn't always do it.

3.3 Impact on Mood and Personal Situation

Next, we investigate how zero2hero could have influenced an author's mood and personal situation. Specifically, we interviewed random people close to an author (family, friends, colleagues) and asked uncomfortable questions about the author's current personality. As usual, we did this in the current universe $\mathcal{U}_C$, where the author didn't use zero2hero as well as in the parallel universe $\mathcal{U}_N$, where the author did use zero2hero.

[2] However, our source code was leaked and submitted to GitHub by a ghost author that we later removed from the planet and the manuscript (in this order).

[3] https://en.wikipedia.org/wiki/Many-worlds_interpretation

w/o zero2hero

"I'm a colleague of [redacted]. I got to know [redacted] when I joint his group in 2015. Mr. [redacted] is very talented and clearly loves doing research; unfortunately, as far as I can tell, his career is marked by rejection. In the last three years, almost 65 percent of his papers have been rejected from CVPR. That has not passed him by without leaving a trace. He changed. He looks sad. I wish there's something that could get him back on track... ."

w/ zero2hero

"I've the pleasure to work with [redacted] for 5 years now. Mr. [redacted] is the best boss I've ever had! He's a machine, his papers rock CVPR every single year. I am not absolute certain, yet I belief his unbelievable success stems from his brilliant ability to write and convince with insanely complex papers (I don't know how he's developing all those formulas, he always locks himself when writing papers). I don't understand his manuscripts at all, even though I usually develop the methods about he's writing. That's so cool. He is truly a hero."

Figure 3. Representative result uncovering how zero2hero affects an author's mood and personal situation. The same colleague talking about the same author, however, one time the author didn't make use of zero2hero (top row), and one time he did (bottom row). To respect the author's privacy, we blanked out his name and only show photos (on the right). We clearly see that zero2hero delivers what it promises.

Please find a representing answer from a colleague for one author in Fig. 3. Obviously, as seen, zero2hero has the complex ability to transform people's lives. Sheesh.

4. Limitations

In case our method is applied to an *actually complex equation* (which, luckily, are rather rare and anyway unnecessary in practice) this might overload human brain capacity. Also, do not apply zero2hero multiple times to the same simple equation. **Please consult your doctor or pharmacist if you've overdone it once again** (watch for symptoms such as disorientation or general confusion, in Germany also known as Ver*wirth*eit [11]).

Moreover, we do want to note that our implementation of zero2hero may not properly handle complex edge cases and, therefore, might be prone to errors. Due to the severe complexity of zero2hero, however, we do not expect this to be a major limitation in practice as nobody is able to spot those errors anyway. If you do find an issue, please HonkFast [3] and we'll make it work again.

5. Conclusion

It's complicated.

Acknowledgments

Last but not least, I wanna thank me. I wanna thank me for believing in me.
I wanna thank me for doing all this hard work.
Everyone else: Thanks for nothing.

References

[1] J. T. Barron. A general and adaptive robust loss function. *CVPR*, pages 4331–4339, 2019.

[2] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black. Keep it smpl: Automatic estimation of 3d human pose and shape from a single image. *ECCV*, pages 561–578, 2016.

[3] B. Egger and M. Siegel. Honkfast, prehonk, honkback, prehonkback, hers, adhonk and ahc: the missing keys for autonomous driving. *SIGBOVIK*, 2020.

[4] B. Egger, K. Smith, and M. Siegel. opencheat: Computationally helped error bar approximation tool-kickstarting science 4.0. *SIGBOVIK*, 2021.

[5] B. Egger, K. Smith*, T. O'Connell*, and M. Siegel. Action: A catchy title is all you need! *SIGBOVIK*, 2022.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CVPR*, pages 770–778, 2016.

[7] M. A. Islam, M. Kowal, S. Jia, K. G. Derpanis, and N. D. Bruce. Simpler does it: Generating semantic labels with objectness guidance. *BMVC*, 2021.

[8] K. Smith and B. Egger. (un)helper functions. *SIGBOVIK*, 2022.

[9] M. Weiherer and B. Egger. A free computer vision lesson for car manufacturers or it is time to retire the erlkönig. *SIGBOVIK*, 2022.

[10] M. Weiherer and B. Egger. From zero to hero: Convincing with extremely complicated math. *SIGBOVIK (under careful review by very talented, outstanding reviewers)*, 2023.

[11] V. Wirth. Author-unification: Name-, institution-, and career-sharing co-authors. *SIGBOVIK (under careful review by very talented, outstanding reviewers)*, 2023.

Quantum Bogosort

George Huebner[1] and Beam Sharque[2]

[1]University of Illinois at Urbana-Champaign
[2]Inter IKEA Holding B.V.

April 0, 2023

Abstract

Joke algorithms are a staple of computer science and provide many a good chuckle. Take, for instance, MiracleSort[1], an algorithm that relies on alpha particle emission to cause erroneous bit flips to sort a list:

```
while isSorted(unsorted_list) is False:
    time.sleep(1000)
```

Despite their apparent uselessness, joke algorithms provide valuable insight into algorithm design and complexity[2]. In this paper we propose an implementation to Quantum Bogosort, one such joke algorithm.

1 Introduction

Classic Bogosort is a sorting algorithm where a list is randomly permuted and checked to see if it is sorted. In pseudocode:

```
while isSorted(unsorted_list) is False:
    random.shuffle(unsorted_list)
```

It has an average time complexity of $O((n+1)!)$ and the nondeterministic (traditional) method has an unbounded worst case time complexity for any non-trivial list.

Quantum Bogosort[3] (QBS), then, is a variation on classical bogosort. Based on the many worlds interpretation of quantum mechanics, QBS proposes that a quantum computer can superimpose all possible permutations of a list and destroy all universes in which the list is not sorted, thus sorting the list in $O(1)$ time. Because parallel universes are non-communicating, there's no way to know the list is sorted other than by the principle of quantum necessity. The algorithm that satisfies subscribers of the Copenhagen interpretation (where wave functions collapse) uses the same algorithm, but the quantum circuit must be run until the list is observed to be sorted.

Classical computers can already sort lists with a lower bound time complexity of $\Omega(N \log_2 N)$. Consequently, research into quantum sorting algorithms has been minimal, as the only optimizations possible are reducing space complexity. Because quantum computers do not have anywhere near enough qubits (let alone the error resilience) to realize this reduced complexity, sorting has been left to classical computers.

2 Background

Most people (authors included) are likely unfamiliar with quantum computing and its associated quirks, so a minor detour is in order to explain the technologies that will be used to develop and test QBS. This is not meant to be a complete explanation of the quantum mechanics that underlie this project, though, so let the physics enthusiast proceed carefully.

2.1 Quantum Computers

The reader is best served by thinking of quantum computers not as Turing machines (although, naturally, quantum computers *are* Turing machines), but rather circuits composed of gates that operate on registers of qubits (**qu**antum **bit**, the quantum unit of information). Current noisy intermediate scale quantum (NISQ) computers are highly error prone and have comparatively few qubits, which proves challenging for researchers to prove increasingly contrived claims of quantum supremacy.

2.2 Qubits

Qubits have quantum mechanical properties of superposition, entanglement, and interference that distinguish them from classical bits. Whereas a classical bit is either 1 or 0, a qubit is in a superposition of the $|1\rangle$ and $|0\rangle$ eigenstates. To actually make use of qubits, they must be measured, which collapses them to either $|0\rangle$ or $|1\rangle$. This is a probabilistic event, with the quantum mechanical measurement rule determining the probabilities as follows:

$$|x\rangle = \alpha|0\rangle + \beta|1\rangle \;\mid\; \text{Probability}_{|0\rangle} = |\alpha|^2,\ \text{Probability}_{|1\rangle} = |\beta|^2$$

It is helpful to consider the Bloch sphere when analyzing qubit state and rotation around the Pauli axes; essentially, the 'North Pole' is defined as $|0\rangle$ and the 'South Pole' is defined as $|1\rangle$. The 'equator,' then, would be an even superposition of $|0\rangle$ and $|1\rangle$.

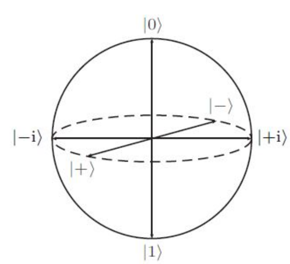


Figure 1: Bloch sphere[4]

Here are a few common qubit gates:

$$X(|0\rangle) = |1\rangle$$

$$X(|1\rangle) = |0\rangle$$

$$Z(|0\rangle) = |0\rangle$$
$$Z(|1\rangle) = -|1\rangle$$
$$H = R_y\left(\frac{\pi}{2}\right) Z \quad \text{(Hadamard gate)}$$
$$H(|0\rangle) = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = |+\rangle$$
$$H(|1\rangle) = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle = |-\rangle$$

Notice that R_y is an arbitrary rotation around the Pauli Y-axis; Similarly, R_x and R_z allow arbitrary rotations along the X- and Z-axes, respectively.
Multiple qubits can be used together for entanglement and amplitude interference — a group of qubits is called a *register*, and is simply a tensor product of discrete quantum states.

2.3 Entanglement

Qubit gates can also be controlled if another qubit is $|1\rangle$ (similar to an 'if' statement in a classical programming language), which allows for multiple qubits to be entangled with each other. Measuring one qubit will also collapse the other entangled qubits, even if they were never observed. A famous example of this is Schrödinger's cat. In this thought experiment, the quantum event of a radioactive substance decaying is entangled with the fate of the cat. If the cat dies, the radioactive substance must have decayed, and if the radioactive substance did not decay, the cat must be alive (or, at the very least, not killed by poison).

3 Algorithm Design

The implementation will be written using Qiskit, a Python framework for creating, running and simulating quantum circuits, and tested with the IBMQ Aer and Manila simulators. We define the following:

- A list of elements, *array*
- The length (cardinality) of this list, l

When we began designing this algorithm, two implementations were considered:

1. One approach would be to store the complete representation of the list in a qubit register. In Qiskit:

```
array = [1, 2, 3]
size = 8
circuit = QuantumCircuit([QuantumRegister(size) for _ in range(len(array))])
```

Python numbers have dynamic size, but our 8-bit size would afford us an unsigned char, that is: $x \in \mathbb{Z} \mid 0 \leq x < 256$

This clocks in at a respectable $O(l \cdot s)$ space complexity, where s represents the size in bits of the data type in the list; While this method could potentially be useful in other quantum algorithms, this would essentially just transpose a classic algorithm onto a quantum circuit — In the spirit of faithfully implementing QBS, we sought a more idiomatic quantum solution.

2. The other approach explored is essentially a quantum random number generator. Instead of storing the list, we create $N = l!$ evenly balanced register states ($l!$ being the number of permutations of *array*). A set of states $[0, N) \subset \mathbb{N}$ will be prepared, with each state having an equal amplitude. Concisely:

$$\sum_{x=0}^{N-1} \frac{1}{\sqrt{N}} |x_{BE}\rangle$$

Note that the quantum register will always be represented in big-endian.

Because we're storing our number in binary, we want the power of 2 greater than or equal to $l!$. This means we have $O(\lceil \log_2 l! \rceil)$ space complexity; with present-day NISQ technology, this is significantly more favorable than $O(l \cdot s)$ for any useful data type. Also keep in mind that because element size isn't a concern, elements can be any type with a partial ordering.

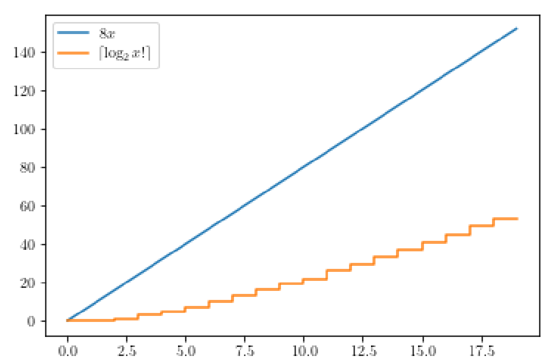


Figure 2: Space complexity comparison

Before we proceed, we would like to discuss the naive approach to quantum RNG. Applying $\mathrm{H}^{\otimes n}$ (apply Hadamard to all qubits) to the entire register can create a number of superimposed states n such that n is a power of 2. We could still use this output to generate a random number h such that $h \in [i, j]$:

$$h = i + \frac{\mathrm{M}(register)}{2^{|register|}} \cdot j$$

The simplicity of this approach is actually not a bad thing: Because quantum computers suffer from high incidences of soft errors (such as erroneous qubit flips), reducing the complexity of a quantum circuit is ideal. Despite this, we wanted the challenge of implementing an algorithm that creates an even superposition of any number of states.

4 Quantum Bogosort

4.1 State Preparation (SP) Algorithm

The main algorithm is recursive, essentially moving through the register from most to least significant qubit, performing a $\mathrm{R_y}$ gate to split up a qubit into P 'parts,' and entangling the remaining qubits with controlled Hadamard (CH) gates.

Let's use $i = 77$ as an example. Recall that for storing an integer i, the most significant bit (MSB) used is $\lfloor \log_2 i \rfloor$: For example, the MSB of 77 would be $\lfloor \log_2 77 \rfloor = 64$. Because SP generates the integers $[0, 77)$ (excluding 77 itself in order to generate 77 integers), we know that $[0, 64)$ is also being generated, because $[0, 64) \subseteq [0, 77)$.

This means that every possible state of the qubit register with a leading zero (in this case, in the 64s place) should be generated. That's already really easy to do! Just use the leading zero as a control and apply Hadamard gates to the rest of the register. Here's an example:

$$|\psi\rangle = |0000000\rangle$$

$$H(\psi_0), \qquad |\psi\rangle = \frac{1}{\sqrt{2}}|0000000\rangle + \frac{1}{\sqrt{2}}|1000000\rangle$$

$$CH(\psi_0, \psi_{[1,6]}),\ X(\psi_0), \qquad |\psi\rangle = \frac{1}{\sqrt{128}}(|0000000\rangle + |0000001\rangle + ... + |0111111\rangle) + \frac{1}{\sqrt{2}}|1000000\rangle$$

Note that CH is a singly controlled Hadamard applied to multiple qubits, not a multi-controlled Hadamard applied to one qubit. Also note the X gate applied to the MSB because we really want an anti-controlled, not a normal controlled gate. The problem is that we've unevenly split the integer 77 into 2: $|[0, 64)| \neq |[64, 77)|$. Numbers in the range $[64, 77)$ have a $\left(\frac{1}{\sqrt{13}} \cdot \frac{1}{\sqrt{2}}\right)^2$ probability of being measured whereas numbers in the $[0, 64)$ range have a $\left(\frac{1}{\sqrt{64}} \cdot \frac{1}{\sqrt{2}}\right)^2$ probability of being measured.

To fix this, we have to change the $\frac{1}{\sqrt{2}}$ from the Hadamard gate to something that gives us a more complex superposition; to do this we use the arbitrary rotation gates. X and Y gates give us the desired rotations we want, but we chose Y so we don't have to worry about phase (the sign of a qubit). Phase doesn't matter in this case because only the magnitudes effect the probability of measurement, but it might be important if using SP in another algorithm.

Superpositions can be represented using spherical coordinates. For example, on the Bloch sphere, the spherical coordinates $(1, 0.955, 0)$ would represent the state $\sqrt{\frac{1}{3}}|0\rangle + \sqrt{\frac{2}{3}}|1\rangle$ because $\cos(0.955) \approx \sqrt{\frac{1}{3}}$. We can do some algebra to get a θ for an arbitrary superposition. For SP, this looks like $\theta = 2\arccos\left(\sqrt{\frac{77-MSB}{77}}\right)$.

Recall that we have to apply an X gate because gates are controlled by $|1\rangle$, not $|0\rangle$. We can save an extra gate by doubling our rotation (hence the 2), which gives us $\sqrt{\frac{77-MSB}{77}}|0\rangle$ instead, which flips right back to $|1\rangle$ with another X gate.

We then recursively repeat this process, adding the MSB as a control qubit to the $\mathrm{R_y}$ and CH gates and using the 2nd most significant bit as the new MSB. Eventually we split all the states evenly and we're done.

The conceptual circuit diagram is shown below; note that • is a control and ∘ is an anti-control (controls on $|0\rangle$ instead of $|1\rangle$), and H_{ALL} is syntactic sugar for a singly-controlled $H^{\otimes n}$.

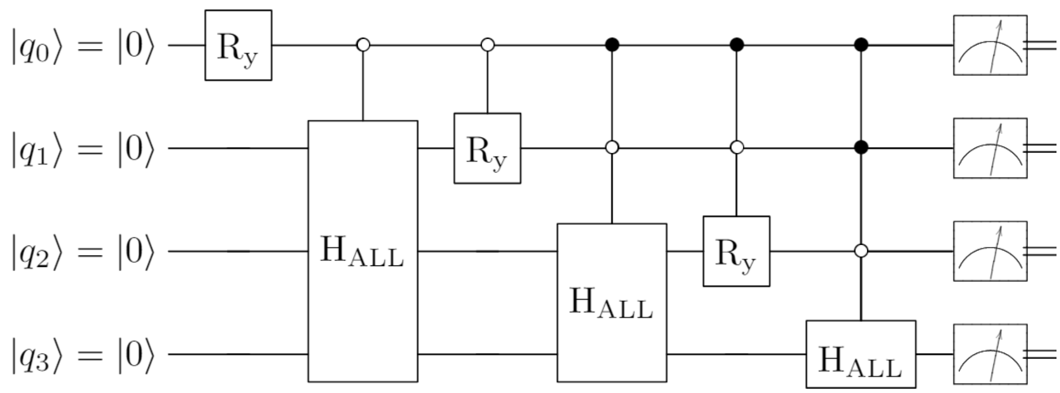


Figure 3: Conceptual circuit[5]

Minor implementation details:

- Multi-controlled Hadamard (MCH) gates are implemented manually with an ancilla qubit and Toffoli (multi-controlled X) gates. This could almost certainly be optimized more on a case-by-case basis (i.e. running `mode='noancilla'` for machines with fewer qubits).
- Especially when there are several qubits in the control register (and therefore the MCH gates are more costly), the MCX on the ancilla doesn't have to be reversed; instead, applying a `'reset'` gate to reset the ancilla to 0 might be more optimal.
- `circuit` and `ancilla` don't change between recursions and needlessly add on to the stack frame of the function. Passing them as parameters is intended for clarity, but if you were to squeeze every drop of performance out they should be made global.

4.1.1 Code

```
from qiskit import *
from math import acos, sqrt

def even_superpositon(circuit, register, controls, ancilla, P):
    if P <= pow(2, len(register)) / 2: # skip this iteration
        even_superpositon(circuit, register[1:], controls, ancilla, P)
        return
    if P == pow(2, len(register)): # reduced to H_ALL; base case
        if controls == []:
```

```
            for i in range(len(register)):
                circuit.h(register[i])
        elif len(register) > 0:
            circuit.mcx(controls, ancilla) # mcx to ancilla used in lieu of mch
            for i in range(len(register)):
                circuit.ch(ancilla, register[i])
            circuit.mcx(controls, ancilla)
        return


    extra_parts = P - pow(2, len(register) - 1)
    # this is the inverse of what we want, but saves us an X for ch
    theta = 2 * acos(sqrt(extra_parts / P))

    if controls == []:
        circuit.ry(theta, register[0])

        for i in range(1, len(register)):
            circuit.ch(register[0], register[i])
        controls += [register[0]]
    else:
        circuit.mcry(theta, controls, register[0])

        controls += [register[0]]
        circuit.mcx(controls + [], ancilla)
        for i in range(1, len(register)):
            circuit.ch(ancilla, register[i])
        circuit.mcx(controls, ancilla)

    circuit.x(register[0])

    even_superpositon(circuit, register[1:], controls, ancilla, extra_parts)
```

4.1.2 Testing

Testing with an even superposition of 15 states:

```
from math import ceil, log
from qiskit.visualization import plot_histogram

parts = 15
length = ceil(log(parts, 2))

register = QuantumRegister(length)
measurement = ClassicalRegister(length)
ancilla = QuantumRegister(1)
circuit = QuantumCircuit(register, ancilla, measurement)
```

```
even_superpositon(circuit, register, [], ancilla, parts)
circuit.measure(register, measurement)
```

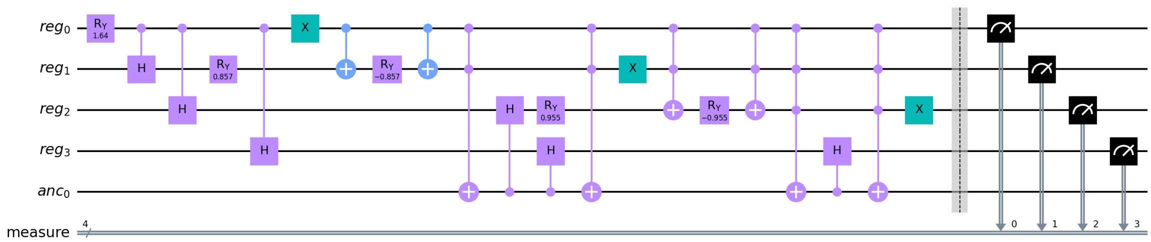


Figure 4: Circuit diagram for 15 ‘parts’

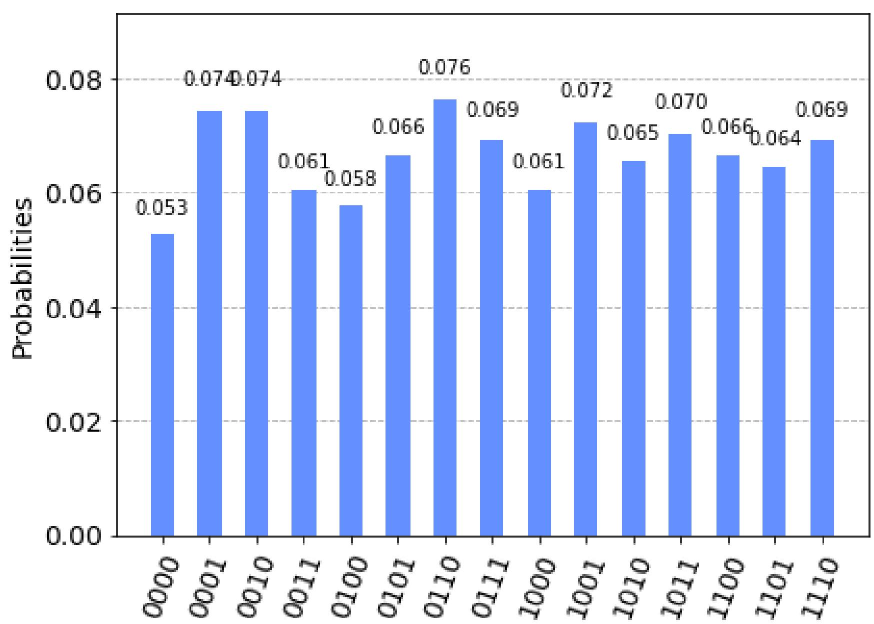


Figure 5: Measurement results from Aer Simulator

4.2 Practicality Assessment

Obviously bogosort is not meant to be a practical algorithm, but nevertheless we will conduct a review of its performance.

4.2.1 Complexity

Although the original QBS proposal assumes the quantum circuit executes in $O(1)$ time, the depth (longest path) of the circuit does scale with input. The time a quantum circuit takes to execute is the sum of the gate execution times of the gates along the longest path of the circuit, and therefore time complexity scales with circuit depth.
For the integer i passed into the SP algorithm, we allocate $\lceil \log_2 i \rceil + 1$ qubits. However,

this algorithm is rather unique in that gate count and depth don't scale exactly with i, but rather `bin(i).count('1')`. That is, circuit depth scales with the number of ones in the binary representation of i.

Consequently, there are large drops in complexity around powers of 2:

$$11111_2 \text{ (5 ones, high complexity)}$$
$$+1 = 100000_2 \text{ (1 one, low complexity)}$$

For example, $i = 256$ creates a circuit with depth 2 and width 17, but $i = 255$ has a circuit depth of 285 and a width of 17 (perhaps even more depending on transpilation target). This relationship is illustrated in 6.

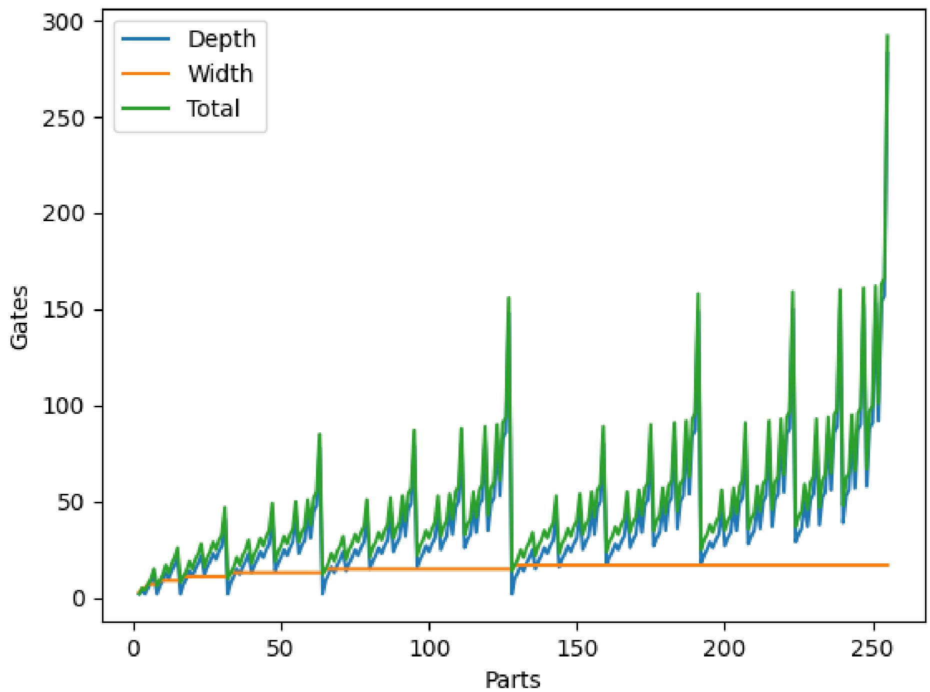


Figure 6: Circuit depth, width, and total gate count in relation to i

With $o =$ `bin(n).count('1')`, we have a time complexity of $O(o)$ for many-worlds and $O(o \cdot n!)$ for Copenhagen.

4.2.2 Error Rate

Of course, the main problem with this ballooning circuit size is the fidelity of the measurement; we work with imperfect machines and error rates can compound over time to make a measurement meaningless. That being said, this is by nature a randomized algorithm, so the noise and error generated don't make much of a difference in this scenario (given that the noise is truly random and evenly distributed) besides occasionally measuring a number $\geq i$.

One way we can model error is by wrapping our gates with error gates — identity gates with an error rate corresponding to a noise profile sampled from real quantum hardware. This approach works with gate errors, including measurement and reset errors, and could also be applied to decoherence and relaxation errors.

Often times noise on hardware will turn a pure quantum state into a mixed state, a composition of pure states with classical probabilities (which is **not** the same as superposition); for example, perhaps an $\mathrm{R_y}\left(\frac{\pi}{2}\right)$ will succeed 80% of the time and over/undershoot by $\frac{\pi}{6}$ 20% of the time. We can write this mixed state as

$$|\psi_0\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \quad \text{(80\% chance)}$$
$$|\psi_1\rangle = \frac{1}{\sqrt{4}}|0\rangle + \sqrt{\frac{3}{4}}|1\rangle \quad \text{(10\% chance)}$$
$$|\psi_2\rangle = \sqrt{\frac{3}{4}}|0\rangle + \frac{1}{\sqrt{4}}|1\rangle \quad \text{(10\% chance)}$$

Density matrices are particularly convenient because we can represent this mixed state in a single matrix:

$$\begin{aligned}
\rho &= \frac{4}{5}|\psi_0\rangle\langle\psi_0| + \frac{1}{10}|\psi_1\rangle\langle\psi_1| + \frac{1}{10}|\psi_2\rangle\langle\psi_2| \\
&= \frac{4}{5}\begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} + \frac{1}{10}\begin{bmatrix} \frac{1}{4} & \frac{\sqrt{3}}{4} \\ \frac{\sqrt{3}}{4} & \frac{3}{4} \end{bmatrix} + \frac{1}{10}\begin{bmatrix} \frac{3}{4} & \frac{\sqrt{3}}{4} \\ \frac{\sqrt{3}}{4} & \frac{1}{4} \end{bmatrix} \\
&= \begin{bmatrix} \frac{1}{2} & \frac{2}{5} + \frac{\sqrt{3}}{20} \\ \frac{2}{5} + \frac{\sqrt{3}}{20} & \frac{1}{2} \end{bmatrix} \quad \text{Purity: } 0.9736
\end{aligned}$$

In the case of quantum bogosort, we can calculate the density matrix for the idealized (ρ) and noisy (ρ') circuits; Both are transpiled for IBMQ Santiago and the latter is injected with Santiago's noise profile. ρ''s purity of 0.42118 reflects that noise heavily affects the fidelity of the qubits.

$$\rho = \begin{bmatrix} \frac{1}{15} & \frac{1}{15} & \cdots & \frac{1}{15} & 0 \\ \frac{1}{15} & \frac{1}{15} & \cdots & \frac{1}{15} & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{1}{15} & \frac{1}{15} & \cdots & \frac{1}{15} & 0 \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix}$$

$$\rho' = \begin{bmatrix} 0.06997 & 0.04275 - 0.00057i & \cdots & 0.0376 + 0.00028i & 0.00036 + 0.00024i \\ 0.04275 + 0.00057i & 0.06646 & \cdots & 0.03687 + 0.00025i & 0.00002 + 0.0001i \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0.0376 - 0.00028i & 0.03687 - 0.00025i & \cdots & 0.06272 & 0.00217 + 0.00045i \\ 0.00036 - 0.00024i & 0.00002 - 0.0001i & \cdots & 0.00217 - 0.00045i & 0.01268 \end{bmatrix}$$

4.3 Comparison to Other State Preparation Algorithms

Araujo et al.[6] use a very similar divide and conquer strategy to recursively calculate the angles for $\mathrm{R_y}$ gates. Although our algorithm was developed independently, it could be seen as a special case of Araujo's algorithm by exploiting the structure of an interval of binary integers to use H instead of arbitrary rotation gates.

Zhang et al.[7] achieve arbitrary state preparation with an impressive depth of $\Theta(n)$, at the cost of $O(2^n)$ additional ancillary qubits. The "parallelization" of CNOT gates employed in their algorithm is also applicable to SP.

Möttönen et al.[8] describe an arbitrary state preparation algorithm from $|a\rangle$ to $|b\rangle$. Unlike Möttönen state preparation, SP starts from $|0\rangle^{\otimes n}$ and completely disregards phase because only the computation basis is considered.

4.4 Bogosort

This is a fairly straightforward implementation of Quantum Bogosort:

1. Generate all permutations of list with length l (symbolic representations are fine to avoid memory overhead)
2. Use SP with parts $= l!$
3. Using the measured value from step 2, randomly select a permutation of the list
4. If the permutation is sorted, return
5. Repeat steps 1-4

Note that with the many-worlds interpretation we can stop at step 4, because a universe where the list is sorted must exist (assuming the list can be sorted) by quantum necessity. Perhaps disappointingly, there's no destroying of universes in this algorithm.

4.5 Complete Algorithm

Using the algorithm defined in 4.1.1

```python
from itertools import permutations
from time import perf_counter

def isSorted(ls):
    if not ls:
        return True

    prev = ls[0]
    for i in ls:
        if i < prev:
            return False
        prev = i
    return True

unsorted_list = [6, 10, 2589, 0, 47, 178, 324]
```

```
possibilities = list(permutations(unsorted_list))

parts = len(possibilities)
length = ceil(log(parts, 2))
register = QuantumRegister(length)
measurement = ClassicalRegister(length)
ancilla = QuantumRegister(1)
circuit = QuantumCircuit(register, ancilla, measurement)

even_superpositon(circuit, register, [], ancilla, parts)
circuit.measure(register, measurement)

start = perf_counter()
while True:
    counts = execute(circuit, Aer.get_backend("aer_simulator"), shots=500) //
    .result().get_counts(circuit)

    max_count = 0
    index = None
    for num, count in counts.items():
        if count > max_count and (benum := int(num[::-1], 2)) < parts:
            max_count = count
            index = benum

    if index is not None:
        if isSorted(possibilities[index]):
            break
end = perf_counter()

print(f"Finished Quantum Bogosort in {end-start} seconds on list of length //
      {len(unsorted_list)}")
```

An impressive sort time of 258.02 seconds on a 7 element list... Quite bogus, indeed.

5 Conclusion

In this paper, we implemented the quantum bogosort algorithm by using a novel state preparation routine to randomly permute the list. Although QBS is a joke, the state preparation algorithm implemented could potentially be useful in setting states where $\mathrm{H}_{\mathrm{ALL}}$ wouldn't do the trick.

Right now, the SP routine generates an interval of integers, but this could be extended to generating an even superposition of arbitrary values. As discussed in the practicality assessment, this algorithm does not scale well due to the prevalence of multi-controlled gates.

Another way to improve QBS would be to study how the list is sorted. Because the (non-unique) permutations of a list grows at $n!$, SP's $O(\log_2 n!)$ space complexity quickly becomes overwhelmed with larger lists. Finding ways to reduce this inefficiency would

be ideal, perhaps by using a 'divide and conquer' strategy (à la quicksort); However, this would defeat the purpose of bogosort!
Although this is one way to achieve even measurement amplitudes over a register, it's not necessarily the *fastest* way; transpilation optimizations and optimal gate decompositions pose interesting extensions to this problem that are especially pertinent with current quantum computers.

Acknowledgements

All code was written and tested with Qiskit 0.32.0 and Python 3.8 on WSL2 Ubuntu 20.04.1.

We acknowledge the use of IBM Quantum services for this work. The views expressed are those of the authors, and do not reflect the official policy or position of IBM or the IBM Quantum team.[9]

---

[1] K. Thompson, "Are there any worse sorting algorithms than bogosort (a.k.a monkey sort)?," Feb 2013.

[2] H. Gruber, M. Holzer, and O. Ruepp, "Sorting the slow way: An analysis of perversely awful randomized sorting algorithms," 2007.

[3] T. O. Tree, "Quantum bogosort," Oct 2009.

[4] Wikipedia, "Bloch sphere," Oct 2021.

[5] I. Chuang, "qasm2circ."

[6] I. F. Araujo, D. K. Park, F. Petruccione, and A. J. da Silva, "A divide-and-conquer algorithm for quantum state preparation," Mar. 2021.

[7] X.-M. Zhang, T. Li, and X. Yuan, "Quantum state preparation with optimal circuit depth: Implementations and applications," nov 2022.

[8] M. Mottonen, J. J. Vartiainen, V. Bergholm, and M. M. Salomaa, "Transformation of quantum states using uniformly controlled rotations," 2004.

[9] IBM Quantum, 2021.

[10] A. Suau, "What is maximum circuit depth and size ibm q5 and q16 could handle?," Mar 2019. Quantum Computing Stack Exchange.

[11] M. A. Nielsen and I. L. Chuang, "Quantum computation and quantum information," 2021.

I

Line of Inquiry, Meet Your Logical Conclusion

Unlimited null: achieving memory safety by extending memory protection

hikari_no_yume

2023-04-00

We present a novel approach to preventing memory safety vulnerabilities that extends existing memory protection systems. Our method achieves complete memory safety, while improving on existing approaches by being simple to implement, not requiring changes to application software or hardware, and having a positive impact on code size and performance.

1 Background and motivation

Memory safety has become an increasingly pressing security and reliability issue in computer software. While approaches like Rust[1] and CHERI[2] offer promising solutions for future software and hardware, they can't easily be applied to existing systems. Consequently, while such approaches are successfully reducing the number and severity of vulnerabilities in new systems,[3][4] there is a long tail of existing systems where memory safety issues continue to be discovered and exploited, and they continue to cause a large proportion of security vulnerabilites.[5]

2 Virtual memory and memory protection

Virtual memory is a ubiquitous feature of modern computer systems.[6] One of its many benefits is that regions of virtual memory can be *protected* from unintended accesses (that is, reads and writes), so-called memory protection. This protection is independent of the programming language, so even notoriously unsafe languages like C can benefit. However, prior work has not realised the full potential of this capability. Indeed, even though almost all modern systems make use of memory protection, memory safety vulnerabilities continue to exist.

There are many common techniques that use memory protection to prevent or mitigate certain kinds of memory safety issues. One technique that is of particular interest to us is the so-called *null page* or *page zero*. This is a region of the virtual address space that starts at `0x0` (i.e. null), is at least a single page in size, and which has memory protection applied so that all accesses are disallowed. The purpose of this region is to trap null-pointer accesses: if memory-unsafe code attempts to dereference a null-pointer, it will usually resolve to an address within this region, and therefore can be trapped by memory protection. This is a standard feature of modern operating systems and has shown itself to be effective at catching many, but not all, null pointer accesses.

3 Limitations of conventional null page design

The principal limitation of the conventional null page is its size. An access to a pure null pointer will always be trapped, but often there is some address calculation involved that leads to the access landing outside the protected region. In pseudo-C: `*(int*)NULL` will always hit the null page, because the resulting address is always `0x0`, but whether `((Foo*)NULL)->bar` hits the null page depends on the offset of `bar` within the struct `Foo`. On 32-bit iOS, for example, the null page size is 4KiB by default, so if `bar` were at an offset of 4KiB or greater, the access wouldn't be trapped.

While recent work has significantly pushed this limit (for example, Darwin increases the null page size to 4GiB on 64-bit targets[7]), it is not completely removed, so the null page has remained a mere mitigation, rather than a complete protection against null-pointer issues.

4 Our contribution

In other words, the limitation of the conventional technique is:

> If only part of memory is protected against null-pointer accesses, such accesses will still be possible in some circumstances.

By divine revelation, we obtained the following corollary:

> If all of memory is protected against null-pointer accesses, no accesses will be possible in any circumstances.

We consider that this is a novel insight that can be practically applied. In order to work backwards to our pre-ordained conclusion, we ask a rhetorical question: if we wish to prevent all unsafe use of memory, why is only some of memory protected?

[1]The Rust Core Team. (2015). Announcing Rust 1.0. *blog.rust-lang.org.*

[2]Watson, R. N., Woodruff, J., Neumann, P. G., Moore, S. W., Anderson, J., Chisnall, D., ... & Vadera, M. (2015, May). CHERI: A hybrid capability-system architecture for scalable software compartmentalization. *2015 IEEE Symposium on Security and Privacy* (pp. 20-37). IEEE.

[3]Xu, H., Chen, Z., Sun, M., Zhou, Y., & Lyu, M. R. (2021). Memory-safety challenge considered solved? An in-depth study with all Rust CVEs. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, *31*(1) (pp. 1-25).

[4]Stoep, J. V. (2022, December). Memory Safe Languages in Android 13. *security.googleblog.com.*

[5]Miller, M. (2019, February). Trends, challenges, and strategic shifts in the software vulnerability mitigation landscape. BlueHat IL.

[6]Denning, P. J. (1997). Before memory was virtual.

[7]Apple Inc. (2020, September). ld(1). *Darwin General Commands Manual.*

With this in mind, we propose a new technique: the *unlimited null page*. By simply extending the "null page" region to cover the entire virtual address space, all null-pointer accesses can be trapped. This can be trivially implemented in existing operating system kernels without requiring hardware or application software changes.

As we will see in later sections, the potential for this technique extends far beyond preventing null-pointer accesses.

5 Prototype

In order to test the viability of this technique, we have implemented it in a real-world system. We chose to implement it in a fork of *touchHLE*,[8] a high-level emulator for iPhone OS applications. touchHLE was chosen because it already implements null-page memory protection, is easy to modify, and *only* runs real-world existing applications;[9] the fact that it is also the author's personal hobby project was surely not a factor.

While touchHLE is written in Rust, it is only the emulator itself which has been memory-safe until now; the protection available to emulated apps has been constrained by the limits of conventional techniques, and in fact was inferior to the original platform, iPhone OS. By implementing our new approach, we hoped to not only reach comparable safety, but surpass it.

We found that our approach was very practical to implement: only 4 files had to be changed, 13 lines inserted and 28 lines deleted, with almost no refactoring required.

In order to assess the functional impact of our changes, we created a corpus from 287 randomly selected iPhone OS applications. Three applications were removed due to [note to self: insert post-hoc justification]. We then ran the unmodified and modified versions of the emulator against this selection of 284 applications.

We found that 284 of 284 applications immediately crashed in our modified version of the emulator. On the other hand, with the unmodified emulator, 284 of 284 applications crashed. This is a difference of 0, so we conclude that there is no impact on application compatibility.

We also observed significant performance and code size improvements. Executing the entire corpus in the modified emulator took 1m33.524s in total wall-clock time, down from 3m38.580s in the original, a reduction of circa 57%. The size of the optimised release binary for the emulator shrank from 12.37MB to 11.44MB, a reduction of circa 7.5%. We hypothesise that these improvements come from the Rust compiler removing code that contained security vulnerabilities, now statically provable to be dead.

[8]https://github.com/hikari-no-yume/touchHLE/tree/unlimited-null

[9]hikari_no_yume. (2023, March). Apps supported by touchHLE. *touchHLE project*.

6 Other memory safety issues

While our original goal was to prevent null-pointer accesses, we have discovered that the unlimited null page approach can, in fact, prevent a diverse range of other memory safety issues. In a particular order:

- All issues that *write XOR execute* permissions attempt to mitigate.
- All issues that *guard pages* attempt to mitigate.
- Stack and buffer overflows.
- Integer overflow during address calculation.
- Stack and heap corruption.
- Uninitialised memory.
- Use after free.
- Double free.
- Heap fragmentation.

To our amazement, it can even prevent safety issues in broader categories:

- Time of check to time of use vulnerabilities.
- Logic errors.
- Denial of service vulnerabilities.
- Backdoors, insider attacks and other malicious activity.

In fact, no safety issue the authors are aware of can be reproduced in a system using unlimited null pages, nor any safety issue we aren't aware of. It trivially follows that this system achieves complete memory safety.

7 Comparisons

Environment	Protected range	Protected fraction
MS-DOS	None	0
iPhone OS (32-bit)	`[0, 0x1000)`	9.54×10-7
macOS (64-bit)	`[0, 0x100000000)`	2.33×10-10
touchHLE (*ours*, 32-bit)	`[0, 0x100000000)`	1

Solution	Existing code	Existing hardware	Memory safety
None	Yes	Yes	No
CHERI	Yes	No	Partial
Rust	No	Yes	Outside `unsafe`
Ours	Yes	Yes	Yes

8 Future work

While our approach achieves complete memory safety by trapping all invalid memory accesses, it has been noted that in some situations, some or all *valid* accesses could also be trapped, which theoretically has performance or correctness implications. We welcome future work that can avoid this issue.

A more significant limitation is the unknown applicability of our work to computing systems that do not use random access memory. We hope to soon publish a follow-up paper on this topic, tentatively titled *Mission Impossible: achieving memory safety on Turing machines with self-destructing tape.*

miles2km: The worst ways to convert from miles to km

Lucien Rae*

***Abstract*—Addressing a naïve gap in the literature, we present a multitude of modern approaches to the *miles2km* problem for state-of-the-art conversion from miles to kilometres.**

1 Introduction

The *kilometre* (or *kilometer*[1]) is a measurement of distance equal to 1000 metres [1].

The *mile* is also a measurement of distance, but is instead equal to 5280 feet, 63360 inches, 1760 yards, or 1609.344 metres [2]. This is obviously a whole lot sillier, yet, at least for now, many people are stuck dealing with it [3, 4, 5].

As a link between the rogue imperialism of the mile and the modern charm of the kilometre, conversion from miles to km is a highly common and useful computational task. However, a baffling gap in the literature[2] suggests our only method of doing so is by a naïve method: simply multiplying the number of miles by 1.609344.

In this paper we explore some alternative methods to converting miles to km, leveraging modern techniques such as recurrence relations, geolocation, and advanced data collection to finally take this common conversion into the 21st century.

We summarise the takeaways of each method with the following semi-qualitative metrics:

- **Does it work?** Does this method actually convert a number of miles to the correct number of kilometres?
- **Is it efficient?** Computationally or otherwise, what kind of time/resources are we enjoying here?
- **Hot or not?** Is it interesting? Is it fun? Is it sexy? Emotionally, this is the most important.

2 Standard (naïve) method

Multiply by 1.6093444. Formally:

$$f(m) = 1.609344 \times m$$

Where $f(m)$ returns the number of km for some m number of miles.

2.1 Critique

Does it work? Sure.

Is it efficient? If you're doing it by hand, no. If you're doing it by computer, yes, but don't feel bad about that because numbers are what computers are for [6].

Hot or not? Absolutely *not*. Not interesting, not fun, and certainly not sexy. We can do better.

3 Fibonacci method

The Fibonacci sequence is a sequence of numbers that looks like this:

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots$$

Each term, called a Fibonacci number, is the previous two terms added together[3] [7]. While in most cases this is kinda just fun [8], we pan for gold. Observe the terms:

$$\ldots, 3, 5, 8, 13, 21, \ldots$$

The successor of 3 is 5. Miraculously, 3 miles are *almost*[4] 5 km. 5 miles are also *almost* 8 km, 8 miles are *almost* 13 km, and in fact 13 miles are *almost* 21 km! That's 3, 5, 8, 13, 21, just like the Fibonacci sequence! Breaking the upper-limit of three coincidences, we must be onto something.

F_n miles	Desired km	F_{n+1}
3	4.83	5
5	8.05	8
8	12.87	13
13	20.92	21
21	33.79	34

Figure 1: Some Fibonacci numbers F_n as miles, their desired conversion to km (to two decimal places), and the Fibonacci successor F_{n+1}.

*lucienrae.com/contact.

[1] In this paper the abbreviation *km* is sporadically used to graciously avoid alienating American English-speaking readers.

[2] How do you cite a lack of citations?

[3] Besides the first two, which are 0 and 1 just because they're good at getting the party started.

[4] 3 miles = 4.828032 km ≈ 5 km.

Using this we form an elegant method: "for a number of miles F_n return the next Fibonacci number F_{n+1} as the number of km."

As exampled in Figure 1, this works very well for miles that are Fibonacci numbers. However, what about the edge-case of a number of miles that don't happen to be Fibonacci numbers? We can account for this easily with the following modified method: "for any given number of miles m look for the nearest Fibonacci number $F_n \geq m$ and return the next Fibonacci number F_{n+1} minus the difference between m and F_n as the number of km." Formally we can re-arrange this as:

$$f(m) = m + F_{n+1} - F_n \tag{1}$$

Where F_n is the smallest nth Fibonacci number $\geq m$.

However, as m increases, and so do the gaps between Fibonacci numbers [7], we sustain an error scaling with the ratio between F_n and its successor. Adjusting by this ratio eliminates our error with an elegant simplification:

$$f(m) = \frac{mF_{n+1}}{F_n} \tag{2}$$

As seen in Figure 2, this provides us with a consistently (incredibly!) close approximation to the desired conversion. How? Read on...

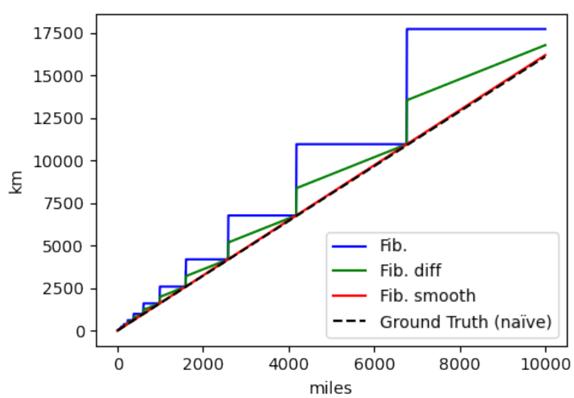


Figure 2: Results of our Fibonacci-based methods converting miles to km, produced by the implementations in Section 3.1 and 3.2, compared to the desired value.

3.1 Iterative implementation

To use our function, we need to work out the Fibonacci number $F_n \geq m$ and its successor F_{n+1}. An iterative (using iterations) implementation is simple: "calculate each term of the Fibonacci sequence until no longer below m." We can trivially decide[5] that this search has some log_ϕ-ish computational complexity against the input due to the exponential gaps between Fibonacci numbers [7]. A Python implementation is presented in Figure 3.

[5] Simply deciding (rather than proving) computational complexity is $O(n)$, where n is how much you care about being correct.

```
def miles2km(m):
    # Converts 'm' miles approximately to km
    a, b = 0, 1
    while a < m:
        a, b = b, a+b
    return (m*b)/a
```

Figure 3: A Python implementation of the iterative smooth Fibonacci method (Equation 2) for converting m miles to km.

3.2 Closed-form solution

While converting 5.47×10^{23} miles (the approximate width of the observable universe [9]) to kilometres using the iterative method takes my 2-year-old laptop a fairly sufficient 7.04×10^{-5} seconds, it would be unrigorous to ignore a hellish future in which we are both (a) calculating distances significantly larger than the width of the known universe and (b) using the mile to do so. To play things safe, we can utilise powerful closed-form Fibonacci expressions to truly optimise our operation.

Where $\phi = \frac{1+\sqrt{5}}{2}$ (the golden ratio) and its conjugate $\psi = 1 - \phi = -\frac{1}{\phi}$, we can compute the nth Fibonacci number with the following expression [10]:

$$F_n = \frac{\phi^n - \psi^n}{\phi - \psi}$$

We can also invert the floored truncation expression $F_n = \left\lfloor \frac{\phi^n}{\sqrt{5}} + \frac{1}{2} \right\rfloor$ [10] to get the index n of the nearest F_n not less than some number m:

$$n(m) = \left\lceil log_\phi \left(m \cdot \sqrt{5} - \frac{1}{2} \right) \right\rceil$$

Alright let's do some maths:

$$\begin{aligned}
f(m) &= \frac{mF_{n+1}}{F_n} \\
&= \frac{mF_{\lceil log_\phi(m\cdot\sqrt{5}-\frac{1}{2})\rceil+1}}{F_{\lceil log_\phi(m\cdot\sqrt{5}-\frac{1}{2})\rceil}} \\
&= \frac{m\frac{\phi^{\lceil log_\phi(m\cdot\sqrt{5}-\frac{1}{2})\rceil+1} - \psi^{\lceil log_\phi(m\cdot\sqrt{5}-\frac{1}{2})\rceil+1}}{\phi-\psi}}{\frac{\phi^{\lceil log_\phi(m\cdot\sqrt{5}-\frac{1}{2})\rceil} - \psi^{\lceil log_\phi(m\cdot\sqrt{5}-\frac{1}{2})\rceil}}{\phi-\psi}} \\
&= \frac{m\phi^{\lceil log_\phi(m\cdot\sqrt{5}-\frac{1}{2})\rceil+1} - \psi^{\lceil log_\phi(m\cdot\sqrt{5}-\frac{1}{2})\rceil+1}}{\phi^{\lceil log_\phi(m\cdot\sqrt{5}-\frac{1}{2})\rceil} - \psi^{\lceil log_\phi(m\cdot\sqrt{5}-\frac{1}{2})\rceil}} \\
&= \frac{m\phi^{\lceil log_\phi(m\cdot\sqrt{5}-\frac{1}{2})\rceil+1} - (-\phi)^{-(\lceil log_\phi(m\cdot\sqrt{5}-\frac{1}{2})\rceil+1)}}{\phi^{\lceil log_\phi(m\cdot\sqrt{5}-\frac{1}{2})\rceil} - (-\phi)^{-\lceil log_\phi(m\cdot\sqrt{5}-\frac{1}{2})\rceil}} \\
&= \frac{m\phi\left(\phi^{\lceil log_\phi(m\cdot\sqrt{5}-\frac{1}{2})\rceil} - (-\phi)^{-\lceil log_\phi(m\cdot\sqrt{5}-\frac{1}{2})\rceil}\right)}{\phi^{\lceil log_\phi(m\cdot\sqrt{5}-\frac{1}{2})\rceil} - (-\phi)^{-\lceil log_\phi(m\cdot\sqrt{5}-\frac{1}{2})\rceil}} \\
&= m\phi \quad = \frac{m(1+\sqrt{5})}{2} \quad \text{(Hark!)}
\end{aligned}$$

Although the golden ratio is a falsely attributed solution to many things [8], we finally find a useful real-world application! As you may have suspected, because you're so intelligent, the reason that this Fibonacci method works at all is because the golden ratio $\phi = \frac{1+\sqrt{5}}{2} \approx 1.618034$ just happens to be incredibly close to the factor between km and miles (1.609344).

3.3 Critique

Does it work? Incredibly, *almost*. With the *smooth* method (Equation 2) we're only off by $\sim 0.5\%$, but even with the personally cuter *diff* method (Equation 1), our accuracy still holds up to SOTA performance[6]. Imagine asking your friend off the top of their head to convert some miles to km and them having performance like that seen in Figure 2.

There's also an argument that this actually works *better* than the standard method (multiplying by 1.609344) because you naturally get fractional results! Observe:

$$3712 \text{ miles} = 5973.884928 \text{ km} \qquad \text{(yuck!)}$$

vs.

$$3712 \text{ miles} \approx \frac{25111680}{4181} \text{ km} \qquad \text{(beautiful!)}$$

Is it efficient A lot better than you might first think! We can vibe the iterative method out to have some sort of $O(log_{\phi}n)$-ish complexity, thanks to the exponential distribution of Fibonacci numbers, which is already very good. So we were really spoilt to reach an even better constant O(1) closed-form solution (see Section 3.2).

Hot or not? Elegant, fun, mysterious, *bon appétit*.

While the standard/naïve method will always be (aesthetically) weighed down by the unappetising appearance of the 1.609344 constant, our Fibonacci method either (a) uses a less arbitrary-looking constant by using the closed-form solution to the golden ratio, or (b) truly appeases the computer, as seen in Figure 3, by using only 0s and 1s as literals. For both human and computer, this is hot stuff.

The real seduction of this method comes in its elusiveness. If you were to present someone with the code of Figure 3 without the function name and comment, they would have no intuition on its purpose to convert miles to km. Although it all relies on a huge coincidence, perhaps everything/nothing does [11], and this mystery does a lot for computational sex-appeal [12].

[6] Where SOTA is the guesswork of Some-Other-Tertiary-Alumni, or alternatively Some-Obviously-Terrible-Algorithm.

4 Wait, do you mean land miles or nautical miles?

Uh oh. So a *nautical mile* is a unit of length used for (you guessed it) nautical activities [13] that is for some reason *different* than a normal mile.

While the mile we've previously discussed is equal to 1,609.344 metres, the *nautical* mile is slightly (about $\times 1.151$) longer, formally equal to 1,852 metres as historically derived from the meridian arc length of a latitude minute (of course) [14].

Despite the fact that using a slightly different mile for the water is *insane*, the nautical mile is officially used internationally for marine navigation and the definition of territorial waters [15], and its derived speed unit the *knot* (one nautical mile per hour) is very common in sailing for measuring boat speed as well as meteorology for measuring wind speed [16].

So to convert from miles to km *accurately*, we need to know if we're converting from standard miles or nautical miles. The solution is actually very simple: "Before converting from miles to km, use geolocation to find the user's current coordinates, compare this to a basemap projection of the world, calculate whether this point is or is not at sea, and then use this information to return the correct conversion." A simplified implementation is presented in Figure 4.

```python
def miles2km(m):
  lng, lat = get_geolocation_coordinates()
  is_at_sea = not basemap.is_land(lng, lat)
  if is_at_sea:
      return m * 1.852 # Nautical miles
  else:
      return m * 1.609344 # Land miles
```

Figure 4: A Python program that uses geolocation to convert to kilometres from either miles or nautical miles, depending on whether or not you're at sea.[7]

4.1 Critique

Does it work? Given our previous methods don't even take nautical miles into account, this is a big improvement.

Is it efficient? Unfortunately requesting geolocation and constructing/calculating from a global basemap with accuracy is, in a relative sense, very slow. But sometimes that's the cost of rigour.

Hot or not? If you're into boats and using silly measurements for what you do on those boats, then maybe!

[7] Note that this "implementation" majorly abbreviates (a) the requesting of geolocation coordinates, (b) the procurement of a land/sea basemap, and (c) the way in which we could be calculating these results with the Fibonacci method (see Section 3) and multiplying "nautical km" by 1.151.

5 There are actually a lot of different types of miles.

Ugh. If you sensed that the previous section was holding back some information, it was, but you're not going to like it. It turns out almost everyone throughout history has wanted to call their common distance measurement some sort of "mile", and they're all different in their own special, terrible, descent-into-madness way.

Firstly there are many *technical miles.* As already covered a *nautical mile* is a bit longer than a *statute mile*, but there's also a *geographical mile* just slightly longer than that, and a *data mile* just slightly less, and a *U.S. survey mile* in-between (but certainly not equal to) a statute and nautical mile. What are their exact values? It depends! But before going down that rabbit hole it's important to note the even larger rabbit hole of regional and historical miles. For example, one cannot confuse a *statute mile* with a *Scandinavian mile* that measures up a full 621.371% larger! The Scandinavian mile currently sits at a clean 10 km, though historically it was also 10.688 km in pre-1889 Sweden and 11.295 km in pre-1875 Norway, which one must take into account for historical conversions. But, it gets worse; the *Chinese mile* is either 0.405, 0.358, 0.416, 0.4158, 0.323, 0.537, 0.645, 0.545, or 0.5 km depending on which dynasty or party was in power at the time, and the *Roman* mile depends on either how well-fed your legionaries are (varying the distance between planted mile-stones) or (after the "standardisation" of 29 BCE) the exact measurement of former Roman consul Marcus Vipsanius Agrippa's feet. And don't confuse the Roman mile for the *Italian* mile, which is actually not equal at all to a statute mile but instead a geographical mile! Wait, has the statute mile always been the same? Of course not.

To fully tick the "accuracy" box we cannot make any assumptions, adopting the following method: "Before submitting a number of miles m the user must fill out a questionnaire Q to ascertain exactly what type of mile they're converting from."

5.1 Data

Summarising a Kafkaesque horror of research into a very, very brief summary, we use the following information on some of the various important measurements that poeple have called "miles" throughout disciplines, regions, time periods, and beliefs.

5.1.1 Biblical mile (mīl)

Used primarily by Herodian Jews to ascertain distances between cities and to mark the Sabbath limit, equal to 2000 cubits.

Length: Depending on divergent definitions of the length of a cubit: **0.96 km** according to major posek Avraham Chaim Naeh, **1.152 km** according to rabbi Chazon-Ish, **1.058 km** based on the Egyptian cubit Derā.

5.1.2 Roman mile (mille passus, mille)

Used in Ancient Rome for travel, and later, urban planning, equal to 1000 paces.

Length: 1.48-1.52 km pre-29 BCE, depending on weather and rations, **1.479 km** post-29 BCE based on Agrippa's foot.

5.1.3 Chinese mile (lǐ)

Used through mainland China's history as a traditional unit of measurement, referred to internationally as the Chinese mile, equal (variably) to 1,500 chi.

Length: Depending on changing lengths of the *chi* and *chi*-per-*lǐ*: **0.405 km** during the Xia dynasty (2100-1600 BCE), **0.358 km** during the Western Zhou dynasty (1045-771 BCE), **0.416 km** during the Eastern Zhou dynasty (770–250 BCE), **0.4158 km** during the Qin (221–206 BCE) and Han (205 BCE–220 CE) dynasties , **0.323 km** during the Tang dynasty (618–907), **0.537-0.645 km** during the Qing dynasty (1644–1911), **0.5-0.545 km** during the Republic of China (1911–1984), **0.5 km** in the People's Republic of China (1984–present).

5.1.4 Scandinavian mile (mil)

Used in most of Scandinavian history, and still in Norway and Sweden in everyday speech, tax-deductible work-related travelling distances, and fuel consumption, especially on second-hand cars. Historically based off 36,000 Norwegian/Swedish feet.

Length: 11.295 km in pre-1875 Norway, **10.688 km** in pre-1889 Sweden, now **10 km**.

5.1.5 Arabic mile (al-mīl)

Used by Middle-ages Islamic cartographers, based approximately on one arcminute of latitude.

Length: 1.995 km by al-Farghani's method, **1.925 km** by al-Ma'mun's method, **2.285 km** during the Umayyad period (661-750 CE).

5.1.6 Danish mile (mil)

Used in Denmark before converting to the metric system, originally equal to 17,600, then 24,000 Rhineland feet.

Length: 11.13 km pre-1698, **c. 7.5 km** from 1698 to 1835, **7.5325 km** from 1835 to 1907.

5.1.7 Forest mile (skogsmil)

Used historically in Scandanavian colloquialisms, based on a "rast" (distance between rests).

Length: c. 5 km.

5.1.8 Finnish mile (peninkulma)

Used in Finland, based roughly on the modern Scandinavian mile of 36,000 Norweigen/Swedish feet.

Length: 10.688 km pre-1887, **10 km** after 1887.

5.1.9 Welsh mile

Used in Wales up until its abolishment under Edward I's conquest c. 1200. Defined by 9000 Welsh paces (each 3 troedfedd).

Length: 6.17 km.

5.1.10 Scots mile

Used in Scotland up until the late 18th century following three legal abolishments in 1685 in the Parliament of Scotland, 1707 by the Treaty of Union with England, and the Weights and Measures Act 1824. Defined by 8 Scots furlongs (each 320 falls).

Length: 1.81 km

5.1.11 Irish mile (míle, míle Gaelach)

Used in Ireland from the 17th century to 19th century (the Weights and Measures Act 1824), with residual usage in the 20th century. Defined by 8 Irish furlongs or 320 Irish perches.

Length: 2.048 km.

5.1.12 Dutch mile (mijl)

Used in pre-19th century Netherlands equal to 5,600 ells (armpit-to-finger length).

Length: 3.28-4.28 km.

5.1.13 Geographical Dutch mile

Used in pre-19th century Netherlands as a geographical measure equal to one fifteenth of an equator longitude.

Length: 7.157 km.

5.1.14 Saxon post mile (kursächsische postmeile)

Used in pre-17th century Saxony as the standard distance between mile-posts or mile-stones, equal to 2000 Resden rods.

Length: 9.062 km.

5.1.15 German mile (meile)

Used throughout Germanic regions up until 1872, equal to 24,000 German feet.

Length: 7.5325 km in Northern (Prussian) Germany, **7.586 km** in Southern (Austrian) Germany, **7.4127 km** at sea.

5.1.16 Breslau mile

Used in Breslau, and from 1630 all of Silesia (now Poland, Czechia) up until 1872, equal to 11,250 ells (the distance from Piaskow Gate to Psie Pole).

Length: 12.85875 km.

5.1.17 Hungarian mile (mérföld)

Used in Hungary up until 1874, equal to 24000 láb.

Length: 7.585944 km.

5.1.18 Portuguese mile (milha)

Used colloquially in Portuguese-speaking countries up until 1861 with debated origin.

Length: 2.0873 km.

5.1.19 Russian mile (mílja)

Used in Russia up until 1925, equal to 7 versts.

Length: 7.4676 km.

5.1.20 Croatian mile (hrvatska milja)

Used in Croatia between 1673 and 1871, defined by an arc of the equator subtended by 1/10°.

Length: 11.13 km.

5.1.21 Ottomon/Turkish mile

Used in the Ottoman Empire (1299–1923) and early modern Turkey up until 1933, equal to 5000 Ottoman feet (ayak or kadem).

Length: 1.89435 km pre-1923 (dissolution of the Ottoman Empire), **1.853181 km** in 1923-1933 early Turkey.

5.1.22 Old English mile

Used in medieval and early modern England, also sometimes referenced (speculatively) as the idiomatic "country mile"

Length: c. 2.1 km in Medieval England, **1.524 km** by the Customs of London in Early modern England (c. 1400-1500), based on a variable 5000 feet.

5.1.23 Metric mile

Used primarily in US track and field athletics colloquialisms to describe a middle-distance running or speed skating length.

Length: 1.5 km in reference to the premier Olympics middle-distance run, **1.6 km** in US high-school competition.

5.1.24 Nautical mile

Used primarily by the US and UK for marine and air navigation, the definition of territorial waters, and for the derived speed unit the knot (one nautical mile per hour) used most frequently in sailing and meteorology. First used in 1594 based on one minute (one sixtieth degree) of latitude.

Length: 1.853 km by Hues/Gunter calculation at the equator, **1.861 km** by Hues/Gunter calculation at the poles, **1.853249 km** in post-1866 US, **1.853184 km** as the Admiralty mile in post-1866 UK, **1.85185 km** in the post-1906 French navy, **1.853 km** in the UK for pre-1970 historical reference to the Admiralty nautical mile, **1.852 km** as the international nautical mile (US in 1954, Britain in 1970).

5.1.25 Data mile

Used in radar-related subjects and joint tactical information distribution systems equal to 6000 feet or the distance for a 12 μs radar send/return through a vacuum.

Length: 1.8288 km.

5.1.26 US survey mile

Used by the US national geodetic survey from 1893 to 2022, equal to 5280 US survey feet.

Length: 1.609347 km.

5.1.27 Geographical mile

Used in Europe and later internationally for geographical measurements. Defined as one arcminute along the equator with accuracy that depends on models of ellipsoid Earth.

Length: 1.852 km from the Middle Ages, then also known as the Italian mile, **1.8554 km** by the 1924 International ellipsoid, **1.855342 km** by the 1984 WGS-84 ellipsoid, **1.8553247 km** by the IERS 2010 ellipsoid.

5.1.28 Statue/International mile

The term for the now-international "standard" mile. Used officially in the imperial system of the US, Liberia, and Myanmar, as well as in certain contexts in some other countries, such as the UK for distances and motor vehicle speeds, and Canada for historical rail transport and horse racing. Historically equal to 8 furlongs or 1760 yards or 5280 feet, though officially standardised in reference to the metric system as 1,609.344 m.

Length: c. 1.6 km in 1593 Elizabethian-era England, **1.609344 km** by the 1959 international yard and pound agreement.

5.2 Reflection

Wow... That's a lot of miles.

5.3 Questionnaire

It's during implementation that we find no coincidence in the similarity between the words "exhaustive" and "exhausting."

We leave a somewhat open problem for the "optimal" questionnaire[8] for ascertaining which type of miles one is wanting to convert from, but here we propose a workable example:

Question 1: What is the number of miles you want to convert to km?

Question 2: With which of the following are you *currently* most affiliated?

(a) Herodian-dynasty Jews
(b) China (Ancient or Modern)
(c) The Roman Republic or Empire
(d) Scandanavia
(e) The Islamic Empire
(f) The British Isles
(g) The Dutch Empire
(h) Saxon/Prussian/Austrian States
(i) Silesia
(j) The Kingdom of Hungary
(k) The Portuguese Empire
(l) The Soviet Union
(m) The Balkan States
(n) The Ottomon Empire
(o) The United States of America
(p) The Ocean
(q) None of the above

Question 3: Using your response to the previous question, which is currently your strongest political/social/professional affiliation?

(a) Herodian-dynasty Jews
    (i) Follower of major posek Avraham Chaim Naeh
    (ii) Follower of rabbi Chazon-Ish
    (iii) Follower of speculative archeology (utilising the Egyptian cubit Derā)

[8] Optimising ease *or* fun.

(b) China (Ancient and Modern)

  (i) Xia Dynasty
  (ii) Western Zhou Dynasty
  (iii) Eastern Zhou Dynasty
  (iv) Qin Dynasty
  (v) Han Dynasty
  (vi) Tan Dynasty
  (vii) Qing Dynasty
  (viii) Republic of China
  (ix) People's Republic of China
  (x) Global relations

(c) The Roman Republic/Empire

  (i) Pre-Agrippa Rome
  (ii) Post-Agrippa Rome
  (iii) Greek-based Architecture
  (iv) Middle Ages to 1800s Italy

(d) Scandanavia

  (i) Pre-1875 Norway
  (ii) Pre-1889 Sweden
  (iii) Norwegian and Swedish colloquial speech, tax deduction, and fuel consumption
  (iv) Colloquial Danish
  (v) Colloquial Finnish
  (vi) Traditional forest hiking
  (vii) Reference to modern US/UK

(e) The Islamic Empire

  (i) Follower of Al Farghani
  (ii) Follower of Al Ma'mun
  (iii) Umayyad Period

(f) The British Isles

  (i) Medieval England
  (ii) Early Modern England
  (iii) Pre-Edwardian Wales
  (iv) Pre-1824 Scotland
  (v) Pre-20th Century Ireland
  (vi) 1593-1959 England
  (vii) Late 19th Century UK Navy
  (viii) Post-1959 Britain
  (ix) 21st Century legal reference to 19th Century UK Navy

(g) The Dutch Empire

  (i) Pre-1816 Netherlands
  (ii) Pre-1816 Dutch Survey Geography
  (iii) Post-1816 Netherlands

(h) Saxon/Prussian/Austrian States

  (i) Pre-1700s Travel
  (ii) Prussian Empire
  (iii) Austrian Empire
  (iv) At sea
  (v) Post-1870 Germany

(i) Silesia

  (i) Pre-1872 Poland/Czechia
  (ii) Post-1872 Modern Silesian States

(j) The Kingdom of Hungary

  (i) Pre-1874 Hungary
  (ii) Post-1874 Hungary

(k) The Portuguese Empire

  (i) Pre-1861 Portuguese Empire
  (ii) Post-1861 Portuguese Empire

(l) The Soviet Union

  (i) Pre-1925 Russia
  (ii) USSR and post-Soviet Russia

(m) The Balkan States

  (i) Pre-1871 Croatia
  (ii) Post-1871 Croatia

(n) The Ottomon Empire

  (i) Pre-1923 Ottomon Empire
  (ii) 1923-1933 Early Turkey
  (iii) Post-1933 Modern Turkey

(o) The United States of America

  (i) Late 19th Century US Navy
  (ii) US high-school athletics
  (iii) Pre-2022 geodetic survey
  (iv) Post-revolutionary America

(p) The ocean

  (i) Pre-19th Century nearest to equator
  (ii) Pre-19th Century nearest to north or south pole
  (iii) Late 19th Century communication with US Navy

(iv) Late 19th Century communication with UK Navy

(v) Early 20th Century communication with French Navy

(vi) 21st Century legal reference to 19th Century UK Navy

(vii) International nautical activity

(q) None of the above

(i) Olympics middle-distance running

(ii) Radar-related data communication methods

(iii) 1924-1984 Earth ellipsoid approximation

(iv) 1984-2010 Earth ellipsoid approximation

(v) Post-2010 Earth ellipsoid approximation

(vi) General use

Result: Where m is your response to Question 1 and α is the value in the row corresponding to your response to Questions 2 and 3, let your result be $m \times \alpha$ km.

Some values α are presented as a range, where you can randomly select a value in that range for a non-zero chance to get the exact result.

Q2	Q3	α
a	i	0.96
a	ii	1.152
a	iii	1.058
b	i	0.405
b	ii	0.358
b	iii	0.416
b	iv	0.4158
b	v	0.4158
b	vi	0.323
b	vii	0.536-0.645
b	viii	0.5-0.545
b	ix	0.5
b	x	1.609344
c	i	1.48-1.52
c	ii	1.479
c	iii	1.4208-1.512
c	iv	1.852
d	i	11.295
d	ii	10.688
d	iii	10
d	iv	7.5325
d	v	10
d	vi	5
d	vii	1.609344
e	i	1.995
e	ii	1.925
e	iii	2.285
f	i	2.1
f	ii	1.524
f	iii	6.17
f	iv	1.81
f	v	2.048
f	vi	1.6
f	vii	1.853184
f	viii	1.609344
f	ix	1.853
g	i	3.28-4.28
g	ii	7.157
g	iii	1.609344
h	i	9.062
h	ii	7.5325
h	iii	7.586
h	iv	7.4127
h	v	1.609344
i	i	12.85875
i	ii	1.609344
j	i	7.585944
j	ii	1.609344
k	i	2.0873
k	ii	1.609344
l	i	7.4676
l	ii	1.609344
m	i	11.13
m	ii	1.609344
n	i	1.89435
n	ii	1.853181
n	iii	1.609344
o	i	1.853249
o	ii	1.6
o	iii	1.609347
o	iv	1.609344
p	i	1.853
p	ii	1.861
p	iii	1.853249
p	iv	1.853184
p	v	1.85185
p	vi	1.853
p	vii	1.852
q	i	1.5
q	ii	1.8288
q	iii	1.8554
q	iv	1.8553248
q	v	1.8553247
q	vi	1.609344

5.4 Analysis

Out of interest—that is, an interest generated from a prolonged lack of charts—we can plot in Figure 5 the distribution of values for α.

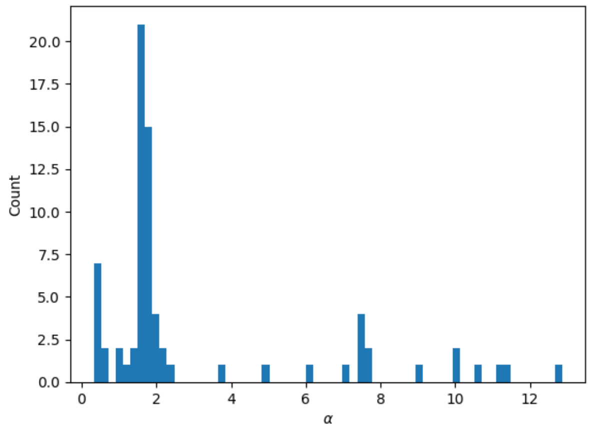


Figure 5: The distribution of values for α.

The dominant mile pile is the 1.609344 tower and the various measurements frustratingly "close" to it, but the true shock is the extent of the skyline's urban sprawl.

The smallest mile to occur, sitting all the way down at 0.323 km, is the Chinese *lǐ* mile during the Tang dynasty from around 618 to 907 B.C.E. Just imagine getting that one right in pub trivia.

The largest is the insane Breslau mile, sitting at a length of 12.85875 km, making it over $\times$7.99 larger than the "standard" 1.609344, and a full $\times$39.81 the smallest Tang-dynasty *lǐ*.

For those desperate for a mean and median even though they'll be pretty close to meaningless, the "average" mile here is exactly 3.078846831081081 km long, and the median is a closer-to-useful-but-in-a-way-that-only-makes-it-more-clearly-not-useful 1.840325 km.

5.5 Critique

Does it work? In some cases, this is accurate in ways never before seen. For the Tang-dynasty time-traveller, the naïve use of any of the previous methods could leave them $\times$5 off their destination. And a similarly-confused Breslavian would get lost by $\times$7.99. While the literature focuses on future-proofing, we have past-proofed.

Is it efficient? Even the rare combination of expertly-crafted user experience and that user's patient comprehension fails to reliably bring to a questionnaire the description of *efficient*. However seeing these faults constant, we consider the search-algorithm-ish vibe of our method as optimally $O(n \log n)$ where n is however many types of miles we decide are enough.

Hot or not? This is definitely not a universal hotness, if such a thing exists, but there is a market. If you could find yourself liking someone to the point that they say "fun fact!" or "um actually. . . " without your complete repulsion, this method certainly fulfils a kind of appeal for knowledgeable (even if broadly self-enjoyed) precision.

6 Conclusion

There is not much more to be said. We have taken miles and turned them to kilometres. Now, more confusingly than ever. And although we see none of this as an absolute *good*, we do find solace playing in the murky tides of disunion while full acceptance of the metric system seems miles—that is, $f(m)$ kilometres—away.

7 Future Work

Having heroically present-proofed *and* past-proofed *miles2km*, we graciously[9] leave to future work the logical conclusion of future-proofing. In a true state-of-the-art mindset, this should probably include machine learning, utilising the beautifully noisy history of past miles to predict the world's *future* miles. We leave this as an exercise to the reader. Or, if this crucial work is somehow not done, a sequel paper that you may eagerly await.

References

[1] B. Taylor, *Guide for the use of the International System of Units (SI): The metric system.* DIANE Publishing, 1995.

[2] A. Hebra, *Measure for measure: The story of imperial, metric, and other units.* JHU Press, 2003.

[3] E. F. Cox, "The metric system: A quarter-century of acceptance (1851-1876)," *Osiris*, vol. 13, pp. 358–379, 1958.

[4] M. Speiring, "The imperial system of weights and measures: Traditional, superior and banned by europe?" *Contemporary British History*, vol. 15, no. 4, pp. 111–128, 2001.

[5] N. De Fabrique, S. J. Romano, G. M. Vecchi, and V. B. Van Hasselt, "Understanding stockholm syndrome," *FBI L. Enforcement Bull.*, vol. 76, p. 10, 2007.

[6] T. L. Sterling, P. C. Messina, and P. H. Smith, *Enabling technologies for petaflops computing.* MIT press, 1995.

[7] N. J. Sloane *et al.*, "The on-line encyclopedia of integer sequences," 2003.

[9]Due to deadlines.

[8] G. Markowsky, "Misconceptions about the golden ratio," *The college mathematics journal*, vol. 23, no. 1, pp. 2–19, 1992.

[9] J. R. Gott III, M. Jurić, D. Schlegel, F. Hoyle, M. Vogeley, M. Tegmark, N. Bahcall, and J. Brinkmann, "A map of the universe," *The Astrophysical Journal*, vol. 624, no. 2, p. 463, 2005.

[10] A. Beutelspacher and B. Petri, "Fibonacci-zahlen," in *Der Goldene Schnitt*. Springer, 1996, pp. 87–98.

[11] M. Bedke, "No coincidence?" *Oxford studies in metaethics*, vol. 9, pp. 102–25, 2014.

[12] S. Knobloch-Westerwick and C. Keplinger, "Mystery appeal: Effects of uncertainty and resolution on the enjoyment of mystery," *Media Psychology*, vol. 8, no. 3, pp. 193–212, 2006.

[13] A. B. Moody, "Early units of measurement and the nautical mile," *The Journal of Navigation*, vol. 5, no. 3, pp. 262–270, 1952.

[14] E. Britannica. (2013) mile, unit of measurement. [Online]. Available: https://www.britannica.com/science/mile

[15] UN. (2016) United nations convention on the law of the sea. [Online]. Available: https://www.un.org/depts/los/convention_agreements/texts/unclos/part2.htm

[16] T. Bartlett, *RYA Navigation Handbook*. Southampton: Royal Yachting Association, 2003.

Fun for the Whole Family: Fast and Furious Transforms

Alex Xie* Alan Hsu[0] Abhishek Vijayakumar
Carnegie Mellon University
{vroom.vroom, skrt.skrt, nierrrrrrrrrrr }@andy.cmu.edu

Abstract

We propose Fast and Furious Transforms *(FFTs), a family of image transforms powered by* ***family****. Our FFT is a two-stage process which transforms an image in the frequency domain into a combination of the most touching family moments in the Fast and Furious movies. Through careful theoretical and empirical analysis, we discover that our transforms are by far one of the transforms of all time. We thus conclude that we don't have transforms – we got family.*

1. Introduction: Family Values

The most important thing in life will always be the people in this room. Right here, right now (emerge). Salute, mi familia.

–Vincent Propane

Modern machine learning research has made great strides in the past decade, but at the cost of the family values we once held dear. To paraphrase one of the great philsophers of our day [7]:

> It seems today that all you see is GANs violence
> in ICML [1] and OwO degeneracy in ICSE [10].
> But where are those good-old fashioned values on
> which we used to rely?
> I was always lucky there was a family transform.
>
> -Sun Tzu
>
> -Peter Griffin

Appalled by this negligence, we seek a return to these good-old fashioned values on which we used to rely. To this end, we draw inspiration from the *Fast and Furious* franchise, well-known around the world for its dedication to family. In this paper, we propose the Fast and Furious transform, the first state-of-the-art image transform that takes into account the contribution of family in image representation learning.

*Has never watched a Fast and Furious movie

[0]Deliberately kept page numbers because he wanted the "funny SIGBOVIK stamp."

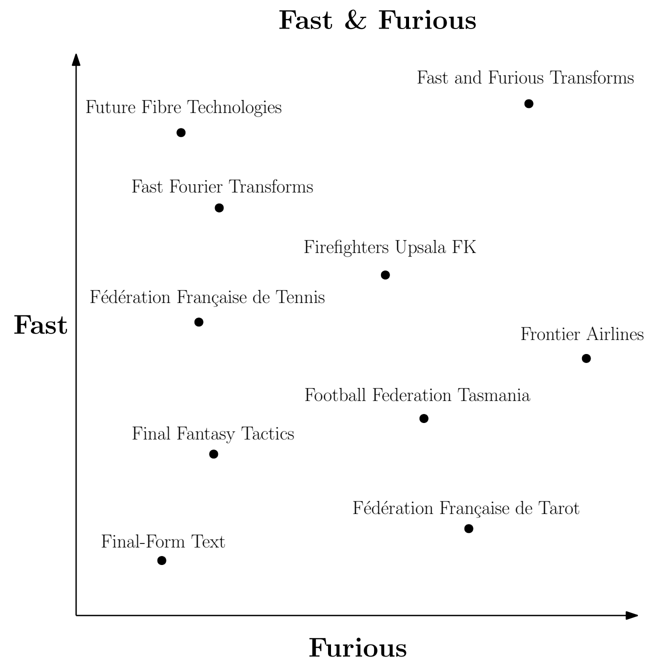


Figure 1: By far one of the figures of all time (in $\log_{10}$ scale).

2. Related Work (hehe do you get it? because family? i hope you got it.)

mu-mu-mu-mu-muridesuu zettai

–Bocchi "The Rock" Johnson [11]

Representation learning: There has been significant interest in recent years in unsupervised representation learning of images [3, 4, 5, 6, 9]. However, these methods fail to account for the inherent familial nature of images. Hence, we group them under the umbrella of **m**ultimodal **i**mage **l**earning **w**ithout **f**amily (MILwF).

FFT: There has also been significant interest in recent years in the acronym "FFT." Fast Fourier Transform (FFT) notwithstanding, we must also compete with *Final Fantasy Tactics* (FFT), Fédération

Figure 2: Sample results for frequency domain decomposition of CIFAR (top row), ImageNet (middle row) and custom (bottom row) images using **family**

Française de Tarot (FFT), Fédération Française de Tennis (FFT), Firefighters Upsala FK (FFT), Football Federation Tasmania (FFT), Federosijuni futboli Toçikiston (FFT), Four Four Two (FFT), Final-Form Text (FFT), Future Fibre Technologies (FFT), Frontier Airlines (FFT). However, we will show that our FFTs are faster and furiouser than all the other acronyms combined (see Figure 1).

3. Methods: Making a family.

Yes, that shit. And I was good in algebra, and like math and shit. And everything else I failed.

–Jesse

3.1. Putting the FFT into FFTs

The first stage of our FFT pipeline is a FFT. Our decision to apply FFT in FFT is motivated by the belief that family transcends mere visual differences. Only by operating in the frequency domain can we understand the true meaning of family.

To begin, we will first define the notion of a set $\boldsymbol{F}$ that we will call a **family**. Specifically, $\boldsymbol{F}$ is the set of movie frames from the Fast and Furious franchise where the cast says the word "family". For a given $\boldsymbol{F}$, we then define $\tilde{\boldsymbol{F}} = \{FFT_{2D}(f) : f \in \boldsymbol{F}\}$ as the set of 2D Fast Fourier transforms (FFTs) of images in $\boldsymbol{F}$, which will serve as the basis for our image decomposition.

Hence, we can define our FFT as a function $f_{\tilde{\boldsymbol{F}}} : \mathcal{I} \to \mathbb{R}^n$ mapping images to n-dimensional embeddings, where n is the cardinality of $\tilde{\boldsymbol{F}}$. In other words, our FFT decomposes each image as a linear combination of FFTs of **family**.

For a given input image $F \in \mathcal{I}$, we first take its FFT to obtain $\tilde{F} = FFT_{2d}(F)$, which will be the input to the second stage of our FFT.

3.2. Putting the Family into FFTs

Now that we have put FFT into FFTs, we now explain the process of putting **family** in FFTs. Given a video sequence $\boldsymbol{V}$ containing all instances of family conversations from the Fast and Furious franchise, we randomly sample ϕ images to form the family $\boldsymbol{F}$:

$$\boldsymbol{F} = \text{sample\_phi}(\boldsymbol{V}). \tag{1}$$

We now construct our corresponding $\tilde{\boldsymbol{F}}$, and crop the center of each image (as a low-pass filter), flatten them to form vectors of length Φ, and concatenate all images column wise to form the $\Phi \times \phi$ **family matrix** $\mathcal{F}$:

$$\mathcal{F} = \text{stack}_1\left(\{\text{flatten}(\text{crop}(\tilde{f})) : \tilde{f} \in \tilde{\boldsymbol{F}}\}\right) \tag{2}$$

Now, given an FFT input image $\tilde{F}$, we resize it such that when flattened it has dimension Φ, which we will call $\tilde{F}_\Phi$. Finally, we can solve for the feature vector $\hat{f}$ that decomposes $\tilde{F}_\Phi$ into our **family**:

$$\mathcal{F}\hat{f} = \tilde{F}_\Phi \tag{3}$$

$$\hat{f} = (\mathcal{F}^T\mathcal{F})^{-1}\mathcal{F}^T\tilde{F}_\Phi \tag{4}$$

4. Experimental Evaluation: Putting our family to the test.

Hey, we do what we do best. We improvise, all right?

–Paul Walker

In figure 2, we show sample image decompositions using our FFT. The results line up with human intuition.

Additionally, for the CIFAR-10 and IMAGENET datasets, we train linear classifiers over the representations yielded by our method and the baseline methods. However, we do not report standard error rate as it does not capture the importance of family bonds. We will instead use alternative bond strength metrics.

4.1. Maybe we can CIFAR, but can family make us CIFARther?

We first evaluate our methods on the CIFAR-10 dataset, a standard image classification dataset. Due to the low resolution of images, our frequency domain transformation is invaluable, proving the importance of visual family invariance. We discover that other methods do not in fact CIFAR into the complex tapestry of family, whereas ours allows us to CIFARthest.[1]

	Ionic ↑	Covalent ↑	James[2] ↑
CLIP	256	432	0.025
SimCLR	126	576	0.024
MoCo	173	132	0.024
FFT (legit)	86	666	0
FFT (ours)	2379	49322	0.025

Table 1: Results on CIFAR-10 dataset. Columns 1 and 2 are in units of kilo-JamesBond/mol and column 3 is in units of kilo-JamesBonds.

4.2. I don't got IMAGENET, I got IMAGEFAMILY.

We further tested the strength of family bonds in our FFT using the IMAGENET dataset, but because the dataset has more classes, we decided to use more generalizable bond metrics to quantitatively assess the performance. The Barry-Bonds metric is used as a control group for the experiment.

	Treasury	Municipal	Barry[3]
CLIP	3.41	2.33	8.84×10^{-12}
SimCLR	2.87	2.83	8.84×10^{-12}
MoCo	3.15	2.19	8.84×10^{-12}
FFT (legit)	2.99	2.50	0
FFT (ours)	420.15	105.69	8.84×10^{-12}

Table 2: Results on IMAGENET dataset. Columns 1 and 2 are percentages and column 3 is in units of light-years.

[1]How many more times can we use this bad pun? Keep reading and you'll CI how FAR it goes!

[2]Number of James Bond movies at time method was proposed.

[3]Distance Barry Bonds ran for all his home runs at time method was proposed.

As we can see, the treasury and municipal bonds of our FFT are much stronger than those of the rest of the models. We can attribute that to our clever usage of **family** embeddings. Thus, we can conclude that FFT (ours) is stonks.

5. Conclusion: Maybe the real transform was the family we made along the way.

Bing chilling.

–John Cena [2]

We conclusively show that our FFTs outperform competing methods in all relevant metrics (not including accuracy). But at the end of the day, life is not about optimizing performance and winning petty competitions. It's about being fast. It's about being furious.[4] It's about family.

References

[1] Samuel Albanie, Sébastien Ehrhardt, and João F. Henriques. Stopping gan violence: Generative unadversarial networks. In *Proceedings of the 11th ACH SIGBOVIK Special Interest Group on Harry Quechua Bovik.*, 2017. 1

[2] John Cena. Eating ice-cream | bing chilling. https://www.youtube.com/watch?v=AWOyEIuVzzQ, 2021. [Online; accessed 1-April-2023]. 3

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020. 1

[4] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020. 1

[5] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*, 2020. 1

[6] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1

[7] Seth MacFarlane. Family guy theme song [orginal - hd]. https://www.youtube.com/watch?v=eeNHV2C5m3w&ab_channel=DungeonDogs, 2012. [Online; accessed 1-April-2023]. 1

[8] Tech N9ne, Joey Cool, King Iso, and Dwayne "The Rock" Johnson. Face off. https://www.youtube.com/watch?v=E9T78bT26sk, 2021. [Online; accessed 1-April-2023]. 3

[9] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry,

[4]It's also about drive. It's about power. We stay hungry we devour. [8]

Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 1

[10] Fwans Skawman and OwOlav. Optimal degeneracy through owo based variable names. In *Proceedings of the 16th ACH SIGBOVIK Special Interest Group on Harry Querulous Bovik.*, 2022. 1

[11] Erika Yoshida and Keiichiro Saito. Bocchi the rock!, 2022. 1

AcHOO: Actually Higher Order Optimization

Clayton W. Thorrez
claytonthorrez@gmail.com

Abstract

In the field of mathematical optimization, especially in optimization for machine learning, the term "higher order optimization" is sometimes used to describe techniques which employ information beyond the first derivative of the objective function. However, in many of these cases, the so called "higher order" simply means 2nd or 3rd. In the unbounded space of integers, it seems fairly underwhelming to consider these "high". In this work we explore using **Ac**tually **H**igher **O**rder **O**ptimization (AcHOO) methods to solve common optimization problems in machine learning. We derive these methods from the basis of Taylor expansions, demonstrate optimization up to the 11th order on synthetic and real world datasets, and provide an implementation in python.

1 Motivation

There have been many recent advances in machine learning. [1] The models are bigger, the data is bigger, the VC funding is bigger, but there is one area in ML which has lagged behind the other in terms of scaling. The vast majority of deep learning models today are trained using first order optimizers.

A few brave souls venture into the realm of using second or third order information and this is considered “higher order” methods by the community. Among the works accepted to the Neurips 2019 Higher Order Methods workshop and the Order UP 2022 Conference (HOO-2022), the large majority cover only second order optimization and only a handful dare to try third or fourth order. This work pushes the conventional boundaries of optimization order and introduces new State Of The Art (SOTA) optimization orders.

2 Background

2.1 Ancient History

The optimizers currently used today in machine learning tasks have roots in very old methods. First order optimization, most famously represented by gradient descent, was introduced close to 200 years ago. (Cauchy et al., 1847) More than 300 years ago Newton introduced his namesake method in the context of root-finding. (Newton, 1711) When using this method to find the 0’s of a gradient, then the method becomes a second order optimizer. (Ypma, 1995) Developed after Newton’s method though published earlier, Edmund Halley used second derivative information in his own namesake method of root finding. (Halley, 1694) Similar to Newton’s method, when finding the roots of a gradient, then it moves up an order and becomes a third order method.

2.2 Recent History

New optimization methods are an area of active research to this day, unfortunately it just seems to be going in the wrong direction. Since the 1960’s there has been research in "derivative free optimization", "evolutionary algorithms", and "black box optimization" all of which we group together as zero order optimization. (Nelder & Mead, 1965; Hansen & Ostermeier, 2001; Conn et al., 2009)

[1] I included this sentence since it seems to be the unofficial secret password to start an ML paper in current year.

These methods attempt to find the values which minimize a function, using only evaluations of the function itself, never taking it's derivative/gradient.

There are also now a wide range of first order variants in use including momentum (Nesterov, 1983), Adam, Adamax, (Kingma & Ba, 2014), AdamW (Loshchilov & Hutter, 2017), RAdam (Liu et al., 2020), and NAdam (Dozat, 2016) and even some non Adam variants if you can believe that.

Second order optimization has also enjoyed some further development, most notably adapting Newton's method into quasi-Netwon methods to scale better. The most famous of these being LBFGS. (Liu & Nocedal, 1989)

3 Derivations

There are several ways to understand and derive optimization methods. Here we will use a method involving Taylor approximations following Boyd et al. (2004) and we will see how it admits a relatively straightforward extension to *actually* higher orders.

3.1 Taylor Series

The standard way to express a Taylor series is using this equation.

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + ...$$

Where f is a function, x and a are points, $'$ indicates a derivative. (multiple $'$ indicate multiple levels of derivatives.) To make a k'th order approximation of a function, we can simply use the first k terms of the Taylor expansion. In the context of optimization it is more useful to rearrange the terms and extend to higher dimensions and this is the result.

$$f(x+v) = f(x) + v^T \nabla f(x) + v^T \frac{\nabla^2 f(x)}{2!} v + v^T \frac{\nabla^3 f(x)}{3!} vv + ...$$

Here we can interpret v as the update applied to the parameter vector x and it replaces $x - a$ in the previous equation. In the first it approximates the evaluation of a function at point x in terms of evaluations of the gradients at nearby points to x. Here it evaluates the function at an offset from a point x in terms of the gradient at the point x. The dimensions get a little weird in higher orders just know that there are as many v's multiplied together as the order.

3.2 First Order

Ok now let's use this to derive gradient descent using a first order Taylor expansion.

$$f(x+v) \approx f(x) + v^T \nabla f(x)$$

In optimization we want to minimize f as much as possible. Which means we want to find v which makes $f(x+v)$ smaller. With a constant x, this is the same as making $v\nabla f(x)$ as negative as possible. According the Boyd et al. (2004) "A natural choice for the search direction is the negative gradient" thus $v = -\nabla f(x)$ and the update becomes:

$$x_{t+1} = x - \eta \nabla f(x)$$

Where η is the step size or learning rate introduced to prevent it from taking too large of a step into regions of parameter space where the first order Taylor approximation does not hold well.

3.3 Second Order

Ok now the same thing with second order Taylor expansion.

$$f(x+v) \approx f(x) + v^T \nabla f(x) + v^T \frac{\nabla^2 f(x)}{2!} v$$

Again the goal is to pick v which minimizes $f(x+v)$. We do this by finding a v such that $x+v$ is a stationary point so we differentiate the right hand side with respect to v and set to 0.

$$\frac{\partial}{\partial v}\left(f(x) + v^T \nabla f(x) + v^T \frac{\nabla^2 f(x)}{2!} v \right) = \nabla f(x) + v^T \nabla^2 f(x) = 0$$

Then solve for v and get

$$v = -\nabla^2 f(x) \nabla f(x)$$

Where $\nabla^2 f(x)$ is a square $D \times D$ matrix also called the Hessian (H). So in Newton's method, we change out the scalar η from gradient descent and replace it with the Hessian to provide an update with much better information about the local curvature of the objective surface.

Thus the update equation is:

$$x_{t+1} = x - H^{-1} \nabla f(x)$$

A weakness of Newton's method is that it requires taking the inverse of the Hession which can be a very expensive operation if the dimensionality D is large.

3.4 Third Order

This section draws heavily from a very nice stackoverflow answer explaining Halley's Method wyer33 (https://math.stackexchange.com/users/33022/wyer33). The basic idea is to do Newton's method on the gradient of the second order Taylor expansion, and make an approximation by substituting in the Newton update direction to make it solvable. Here $F(x)$ is the gradient, ie $\nabla f(x)$.

$$F(x+v) \approx F(x) + v^T \nabla F(x) + v^T \frac{\nabla^2 F(x)}{2} v$$

Then replace one of the v's with the Newton direction denoted v_2. (And v_1 will denote the gradient going forward).

$$F(x+v) \approx F(x) + v^T \nabla F(x) + v^T \frac{\nabla^2 F(x)}{2} v_2$$

Setting the right side to 0 (because the whole thing is a gradient) to get:

$$\left(\nabla F(x) + \frac{\nabla^2 F(x)}{2} v_2 \right) v = -F(x)$$

Then back substitute $\nabla f(x)$ for $F(x)$ to get:

$$\left(\nabla^2 f(x) + \frac{\nabla^3 f(x)}{2} v_2 \right) v = -\nabla f(x)$$

Finally solve for v

$$v = -\left(\nabla^2 f(x) + \frac{\nabla^3 f(x)}{2} v_2 \right)^{-1} \nabla f(x)$$

The part inside the parenthesis which is inverted provides a clue as to how we will extend past order 3. We can see the first term is simply the Hessian, a $D \times D$ matrix. The second term is comprised of three parts. The inverse factorial coefficient of $\frac{1}{2}$, the direction vector of the lower order, in this case the second order v_2, and the $D \times D \times D$ tensor which we call the "thressian". (third order Hessian). It turns out that multiplying the $D \times D \times D$ tensor with the D dimensional vector yields a $D \times D$ result meaning it can add naturally with the Hessian term and multiply naturally with the gradient on the outside of the parenthesis.

3.5 To Infinity And Beyond

Generalizing on the pattern discovered in the third order solution, we can form a recursive update equation for any arbitrary order which utilizes high dimensional curvature of the objective landscape to formulate very precisely optimal updates.

Let $v_1 = -\nabla f(x)$. Then $v_2 = -\left(\nabla^2 f(x)\right)^{-1} \nabla f(x) = H^{-1} v_1$ which is the Newton update direction. We also use the above definition to see the third order update.

$$v_3 = \left(\nabla^2 f(x) + \frac{1}{2}\nabla^3 v_2\right)^{-1} v_1$$

Taking it another step and we get:

$$v_4 = \left(\nabla^2 f(x) + \frac{1}{2}\nabla^3 v_2 + \frac{1}{6}\nabla^4 v_3 v_2\right)^{-1} v_1$$

As the order get's higher and higher, the tensor of partial derivatives keeps getting more dimensions. Thus we repeatedly multiply by lower order update vectors to reduce the dimension to $D \times D$.

One more to show the point.

$$v_5 = \left(\nabla^2 f(x) + \frac{1}{2}\nabla^3 v_2 + \frac{1}{6}\nabla^4 v_3 v_2 + \frac{1}{24}\nabla^5 v_4 v_3 v_2\right)^{-1} v_1$$

Thus the general format is to compute the gradient tensor, multiply by the factorial coefficient, multiply by previous direction vectors down to v_2, sum up the intermediate $D \times D$ terms, invert, and multiply by the gradient. Boom goes the dynamite.

3.6 Disclaimer

We might have made some mistakes in either sign, transpose notation or other minor detail but the core idea is actually legit.

4 Experiments

4.1 Convergence on Synthetic Data

In order to test AcHOO, we implemented the general procedure described above using the jax python library which has super nice support for higher order derivatives by simply calling `jacfwd()` and `jacrev()` repeatedly. Our code is available at https://github.com/cthorrez/AcHOO.

For the first experiment we test on a synthetic linear regression dataset. We generate 10 rows with 1 feature per row. The features are from a standard Gaussian, then we also sample a weight coefficient from a standard Gaussian, and generate the labels by multiplying the features by the weights and adding Gaussian noise with standard deviation 0.2. We can optionally concatenate a column of ones to the input to allow for fitting an intercept term as well. Since Newton's method solves standard linear regression with mean squared error in closed form in a single iteration, we used a modified loss function where the error was raised to the power of 8 rather than 2 to accentuate the differences in the higher order methods.

order	1	2	3	4	5	7	10	11
#iter	36	4	**3**	**3**	**3**	**3**	**3**	**3**
time (s)	**0.27**	0.28	0.48	0.9	1.68	9.70	258.11	795.70

Table 1: Number of iterations until convergence and wall clock time for different optimization orders on the synthetic the 8th order linear regression dataset.

In this case there was no improvement in number of iterations until convergence, but we can see clearly the drastic increase in runtime as order increases.

4.2 Real World Applications

In order to test AcHOO in a real world production level environment. We decided to test it on the Iris flower classification dataset (Fisher, 1936) accessed from the UCI machine learning dataset repository. (Dua & Graff, 2017) This dataset contains 150 rows with 3 features per row with three class options. We simplify the dataset by doing binary classification of class 0 vs class 1 or 2 which cuts the parameter number by one third.

order	1	2	3	4	5	7	10
#iter	27	3	3	**2**	**2**	**2**	**2**
time (s)	**0.47**	0.55	0.88	1.57	2.95	14.43	1825

Table 2: Number of iterations until convergence and wall clock time for different optimization orders on the binarized Iris dataset.

While the time scaling is certainly poor, we do actually see convergence in fewer iteration than Newton's and Halley's methods when using 4th order or higher optimization methods.

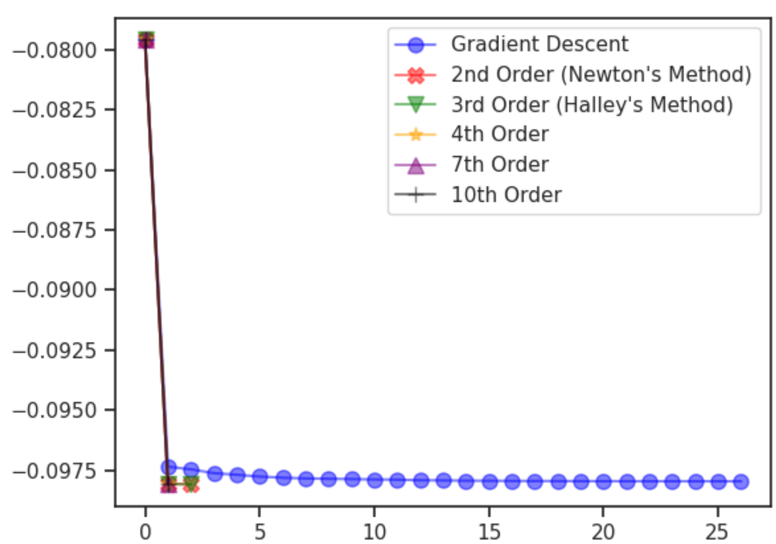


Figure 1: This doesn't really show much but I wrote the code for plotting the training loss so I'm putting it in.

5 Discussion

5.1 Strengths

Our number is bigger and thus SOTA.

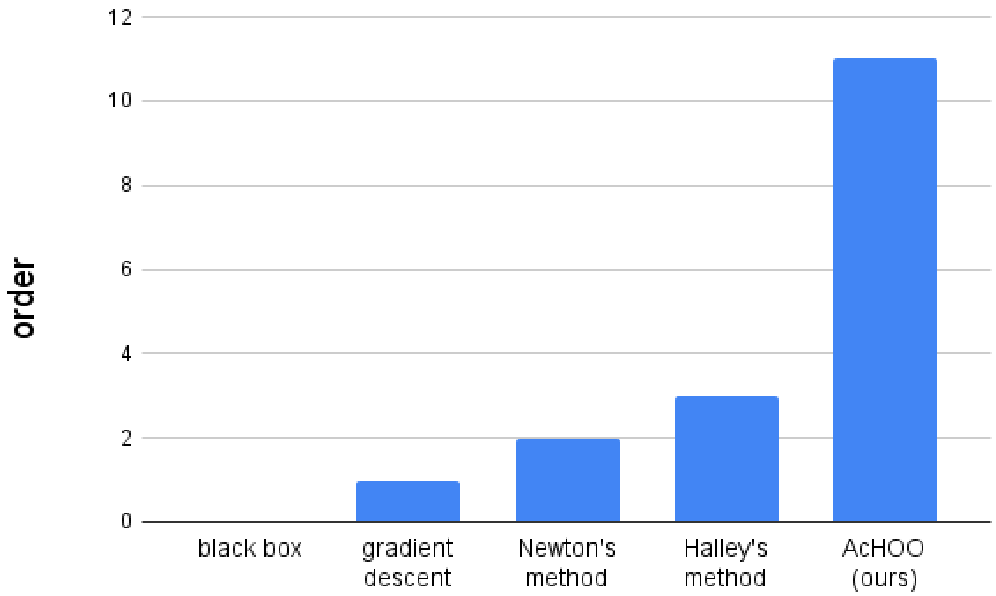


Figure 2: The order of different optimization methods. Eleven is just the highest presented in the current work, there is no theoretical upper bound on the framework introduced.

5.2 Weaknesses

Using higher order optimization methods has very high memory and time costs. For optimization of order k, the tensor of partial derivatives takes $\mathcal{O}(D^k)$ memory. For today's large models such as GPT3 (Brown et al., 2020) with 175 billion parameters, it would only take up to order 8 to surpass the estimated number of atoms in the universe. Even on our small scale experiments going beyond order 11 caused either out of memory errors or had prohibitively long runtimes.

5.3 Next Steps

In the future we would like to take this same idea in the other direction. We briefly touched on zero order optimization using no gradient information. We would like to explore negative order optimization methods where we use point evaluations of the integral of a function to try to minimize the function.

5.4 Conclusion

In this work we introduced AcHOO, a framework to perform actually higher order optimization for machine learning. We included the methematical derivation for how to extend current optimization mehtods and proved the (in)effectiveness on synthetic and real world datasets.

REFERENCES

Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020. URL `https://arxiv.org/abs/2005.14165`.

Augustin Cauchy et al. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847.

Andrew R Conn, Katya Scheinberg, and Luis N Vicente. *Introduction to derivative-free optimization*. SIAM, 2009.

Timothy Dozat. Incorporating nesterov momentum into adam. 2016.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7 (2):179–188, 1936.

Edmund Halley. A new, exact, and easy method of finding the roots of any equations generally, and that without any previous reduction. *Philos. Trans. Roy. Soc. London*, 18:136–148, 1694.

Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=rkgz2aEKDr`.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2017.

John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

Yurii Evgen'evich Nesterov. A method of solving a convex programming problem with convergence rate o\bigl(k^2\bigr). In *Doklady Akademii Nauk*, volume 269, pp. 543–547. Russian Academy of Sciences, 1983.

Isaac Newton. *De analysi per aequationes numero terminorum infinitas*. 1711.

wyer33 (https://math.stackexchange.com/users/33022/wyer33). does it make sense to talk about third-order (or higher order) optimization methods? Mathematics Stack Exchange. URL `https://math.stackexchange.com/q/2852107`. URL:https://math.stackexchange.com/q/2852107 (version: 2018-07-15).

Tjalling J. Ypma. Historical development of the newton-raphson method. *SIAM Review*, 37(4): 531–551, 1995. ISSN 00361445. URL `http://www.jstor.org/stable/2132904`.

The Phonetic Portmantout

Kyle A. Williams

Abstract

The Oxford Dictionary defines the word *portmanteau* as "a word blending the sounds and combining the meanings of two others." Dr. Tom Murphy VII Ph.D., in his paper "The Portmantout," introduced the idea of a *portmantout* (French portmanteau of *portmanteau* and *out* [all]), the orthographic combination of all words in a language—in his paper English—achieving such using a personal dictionary of approximately 108 thousand words. A glaring issue with the portmantout is that the resulting word is unpronounceable due to Murphy ignoring phonetic grammar. This paper aims to rectify that using a corpus of over 134 thousand pronunciations from the Carnegie Mellon University Pronouncing Dictionary (CMUdict) in order to create a word that can ultimately be said by a text-to-speech bot or a theoretical human with a large lung capacity.

Keywords: linguistics, graph theory, phonetics

1 How to Make a Portmanteau

All words, from a phonological perspective, are composed of one or more sounds called *phones* which can be organized into two basic categories: *consonants* and *vowels*. The key to creating a portmanteau is by finding two words with a shared sequence of phones that contains at least one vowel. Let's use *brogrammer*, an example Murphy gives in his paper to see this in action.

In the ARPAbet, the pronunciation of *bro* is "spelled" `B R OW` and the pronunciation of *programmer* is spelled `P R OW G R AE M ER`. Both pronunciations share the sequence of phones `R OW`. Therefore, if we replace the `P` in *programmer* with the `B` in *bro*, we get `B R OW G R AE M ER`, a portmanteau.

1.1 Formalizing and Generalizing

A *phone* is a is a two-letter ARPAbet code which optionally indicates stress. A *vowel* is a phone whose first letter is in the set $\{"A", "E", "I", "O", "U"\}$. A pronunciation is a sequence of phones.

For a set of pronunciations L, in this case CMUdict, a pronunciation s is a *generalized phonetic portmanteau* if the entire pronunciation can be covered by sub-pronunciations in L, where a sub-pronunciation contains at least one vowel. A *cover* is defined similarly to the one seen in Murphy, but allows the cutting off of a pronunciation to enable colloquial portmanteaus like *brogrammer* and *brogrammar*. For example, the pronunciation `B R OW G R AE M ER M EY D` can be covered by the pronunciations for *bro*, *programmer*, and *mermaid* in L, so it is a generalized phonetic portmanteau. This means that the pronunciation `R EY D AW N` cannot be a generalized phonetic portmanteau, even though it would be a *generalized portmanteau* of the pronunciations of *raid* and *dawn* by Murphy's definition, as they lack a shared sub-sequence that includes a vowel.

A *phonetic portmantout* is a pronunciation made up of sub-pronunciations from all pronunciations in L. As in Murphy, sub-pronunciations from a word may show up more than once; it is only necessary that sub-pronunciations from each pronunciation in L are present in the phonetic portmantout.

2 Portmantout Generation

Generating the shortest phonetic portmantout is likely NP-complete for the same reasons Murphy's portmantout is. However, I have found some useful strategies for making joining pronunciations much less long.

2.1 Deduplicating pronunciations

A common kind of generalized phonetic portmantout is where one pronunciation the *prefix* or *suffix* of another pronunciation. For example, the pronunciations of *water* and *melon* are the prefix and suffix of the pronunciation of *watermelon* respectively. A easy-to-implement and fast way to create generalized phonetic portmanteaus from prefixes and suffixes is the *trie*, a kind of tree data structure commonly used in search applications. After inserting all pronunciations into a trie, the phonetic portmanteaus can be found in the tree's leaves, as, for example, the path of nodes that creates the pronunciation for *water* is part of the path of the pronunciation of *watermelon*. Suffix-based portmanteaus can be found by reversing all pronunciations before inserting them into the trie. Inserting items into a trie takes linear time, which is much faster than the polynomial time it takes to find a portmantout by iterating over all pronunciations in L. Through this method, I was able to reduce my initial number of pronunciations from 133,737 to 81,187 in a matter of seconds.

2.2 Generating particles

The process for generating particles is as follows:

- Load CMUdict and deduplicate the pronunciations. Save the result to a set *pronunciations*.
- While *pronunciations* is not an empty set:
  - Take a pronunciation out the set; this will be the base for our *particle*.

  – For every *pronunciation* in *pronunciations*:

    ∗ If *particle* and *pronunciation* can be *joined*, join them together and save the result to particle. Remove *pronunciation* from *pronunciations*.

  – Emit *particle*.

Two pronunciations a and b can be *joined* if one of the following statements is true:

- a is a sub-pronunciation of b or vice versa. An example of this is the pair `R EH D` and `D R EH D IH NG`. Note at this point that a or b must be an *infix* of the other since we already assimilated all prefixes and suffixes.
- If the suffix of a is the prefix of b, or vice versa, and said suffix-prefix contains at least one vowel.

2.3 Joining particles

To join particles, I created a directed graph from all pronunciations in L, where the nodes are phones and the edges are a phone pointing to the next in its pronunciation. With this, we can use Dijkstra's to find a path between the first vowel of one particle and the last vowel of another and then join them together on that path. Once all particles have been joined, the phonetic portmantout is complete.

3 The portmantout

The phonetic portmantout is 423,041 phones long; there are 853,918 phones in CMU-dict, so this is a compression ratio of 2.02:1, which makes the phonetic portmantout adhere to half of the definition of a portmanteau and is more solid than Murphy's portmantout with a ratio of 1.47:1.

Acknowledgments. I would like to thank Peter Nowakoski for advising me on an draft version of this paper and Kiera Reed for encouragement and proofreading. I would also like to thank Jair Santana for reading a draft version of this paper and showing me tries. Finally, thank you to Tom 7 for creating such an awesome paper to riff off of, introducing me to SIGBOVIK, and inspiring me to do research.

AA1 B AH0 G D AE1 N S K IY0 UW0 F AO1 R IH0 AO1 AH0 L IH1 N S K IY0 AE1 N HH OW2 EH1 V ER0 ER0 AO1 R G ER0 EH1 R AH0 K IY2 AA1 M EH0 N T R EH1 N CH IH0 AA1 P ER0
IY0 AA1 K S AH0 D AH0 F EH0 D V IY1 N AH0 F ER1 SH B AW0 F AE1 K UW0 AE0 N T IH2 S AH0 P EY1 SH AH0 N T L IY0 AE1 S HH AA2 P ER0 AH2 L T IY0 M IH2 L Y AH0 N EH1 R B
S AH0 F AY2 IH0 AA1 N AH0 F AY2 AA1 R D IY1 AE1 T IY0 S IH0 F ER1 AH0 N EY2 T IH0 F AY1 Z IY0 P AO1 R AH0 JH AH0 N EH1 T OW0 S K AA1 N OW0 G AA1 L EH0 S EH1 L T IY0
EY1 T UW2 AA0 T OW1 L IY0 IH1 V AH0 D L IY0 AE0 L K AH0 L OY1 D AH0 F IY1 OW0 N AH0 P G R EY1 D IH0 F AH1 NG K L IY2 AH0 P AE1 T R AH0 F AH1 N D S T R AE2 F AE1 G
D L AH0 AA1 R V AH0 S T IH0 F IH1 L K AH0 N S AH0 IH0 F ER1 AH0 AE1 F UW1 P AH0 F IH1 B R AH0 L AY2 Z IH0 F AY0 K OW1 S AH0 S P EH1 N D IH0 AO1 R N W AO0 F AY1 D IH0
EY1 L IH0 OY1 F OW1 B AH0 N Y AA1 F EY1 K EY0 T AH0 F IH1 N T AH0 OW1 L D AH0 N S T AY2 F IH1 F AA2 P AH0 G AE1 N D AH0 F EY1 JH EY1 EY1 AA1 R G Y UW0 AH0 B L IY0
EH1 F EH1 S M AY2 AE1 N AH0 T IY2 OW0 D AO1 R OW0 F AH1 N Z M AH0 F EY1 ZH ER0 IY1 K IH0 F AE1 G L AH0 IH0 L UW1 M IH0 N IH0 F AE1 SH ER0 AE1 K OW0 ER0 IY0 S IY1 D
AH0 N S EY2 EH1 L IH0 N AO1 R D AH0 N AH0 T L IY0 AA1 R D AH0 N B R UH2 F EH1 R AH0 L AH0 EH1 ZH ER0 AH0 B AH0 L AY1 AH0 IY0 V ER1 T IH0 F AE1 M AA0 R K AA1 S K AH0
K Y AH0 B EY2 T ER0 AA1 B AH0 L IH0 F UW1 S OW2 F AO1 R D AY2 OW1 V ER0 K AE2 F EH1 N AH0 L IY0 AE1 N S M AH0 G ER1 K AA2 R T ER0 EH1 T S T IY0 IH0 M P R IH1 Z AH0
ER0 UW1 CH IY1 IH0 S EH1 M AH0 N EY2 AE1 V AH0 T AO2 R IY0 Z EH1 N T IH0 F AE1 K T EY0 T IY1 F AH1 M P ER0 K Y UW0 T EY1 N IY0 AH0 S K AO1 R B IH0 K S T R UW1 D IH0
OW0 L T ER0 AH1 L IY0 M AO0 R AE1 S IH0 F EH1 R IY0 AH0 F ER1 K G R UW2 ER0 IY1 N OW0 AO1 R D G EY2 AE1 N T R AH0 F IY1 L IY0 OW0 AH2 N R IY0 S P AA1 N S IH0 F IH1 N
T R AH0 F UW1 L R UW2 F AO1 R AH0 F UW1 CH AH0 W AH0 F OY1 T AH0 AE1 S G R IY2 AO0 R AA2 L OW1 G OW0 EH0 R AO1 L T S B R UH2 F IY1 Y AH0 T IY2 AA1 L S AH0 B L IY
ER0 IH1 S T OW0 EH1 Z N ER0 OW2 K IY1 N OW0 EH1 M B AH0 AE1 T M AH0 F AO0 R T EH1 L IH0 NG L IY0 OW1 IH0 F EY1 JH W UH0 F AO1 R T N ER0 AY1 L D B ER2 F AO0 R T
EH2 R IY0 S AY1 T IH0 F UW1 P L IH0 S IH0 OW1 F UH0 F AA1 B ER0 AO1 F EY2 B R EH0 N IY1 T S AH0 AE1 R OW0 AE0 N D IY1 D OW0 UH1 D ER0 HH AE2 M B R IY0 AA1 N OW0
AA0 R L AO1 T AH0 M EY2 OW1 V AH0 F IH1 P AH0 K R IH2 F IH1 R S ER0 AY1 R EY2 T IH0 AA1 R B AY2 D OW1 AH0 N Z IH0 OW1 K AA0 OW1 D IH0 L AE0 N AO1 R TH AY2 F AH1 N

An Introduction to Compliers

Anoushka Shrivastava
Carnegie Mellon University

Abstract. Compilers are essential tools in the programming landscape. Indeed, they are often called a man's best friend, if the man is a lonely programmer. However, compilers are also horrible for moral support because they harshly point out a programmer's mistakes. To programmers who truly do consider their compilers to be their closest friend, this can be mentally draining, with side effects such as headaches when code submission deadlines approach, fragmented personal relationships projecting from distrust in one's primary companion, and vomiting. However, we propose a cure: compliers. Compliers are the opposite of compilers: they are extremely supportive of your code on the surface and automatically fix your mistakes under the hood. Our paper details a brief introduction to compliers and strategies to implement the best complier possible.

I. Introduction.

Compliers are tools designed to assist programmers while simultaneously boosting their damaged self-esteem from all of their buggy code. While not all languages rely on compilation, the general functionality of compliers can be extended across programming languages and integrated into existing tools. Specifically, the functionality we focus on is compliers' method of providing errors to, and fixing errors for, the user.

II. Error Message Communication

Compliers target a key point of interaction with the programmer: error messages. Error messages can be subdivided into several categories, and each of these categories causes varying levels of frustration. Good compliers have different strategies to deal with each category type. For example, type errors are a common category of errors, and because they can be somewhat confusing to debug, compliers make sure to add an encouraging emoji to the end of the error message. A good complier will also recognize successful type compilation by playing "My Type" by Saweetie.

No matter the error category, compliers must respond with the technique of a "compliment sandwich" to keep the programmer encouraged. They must also only output one error at a time and patiently wait for the programmer to fix the error before outputting a new one. This prevents the programmer from fainting when they see more errors than code written.

III. Error Detection and Fixes

Compliers also have an important secondary component: they automatically fix parts of your code for you without telling you or showing you where you went wrong. While it may seem like a nightmarish guessing game to guess what a complier does to your code, due to the complier's training process, it is not. Compliers automatically fix code by using a machine learning model that has been trained on buggy code. If, for example, you commit many spelling errors, your complier will learn your bad coding habits over time and remember them during compile-time. Because programmers make plenty of mistakes, the training dataset for this task is quite large, and compliers can get good at guessing what programmers' true intentions are.

IV. Architecture

Complier architecture mimics compiler architecture. The author has unfortunately not taken compiler design courses and lacks the background to dive into a lot of depth. However, adding a neural network can't really hurt. Neither can applying a decision tree.

V. Conclusions

Compliers, unlike compilers, are a supportive tool to reward good (and bad) code with compliments. This has many positive side-effects, such as cheering up the sleep-deprived face on the other side of the screen that looks less clean than their code. By incorporating artificial intelligence, compliers are also a modern upgrade to traditional compilers. Future research should test the improvements in mood that result from using compliers. Studies might also want to confirm the vomiting thing from the abstract.

VI. References

The author relied on past experiences with, and error messages from, compiling her programs.

Feline Fine: A Purr-spicacious Proof of Nekomusume Supremacy Over Human Females

Dr. Tuesday (not a PhD or MD)   Hunter Halloran

hdh20267@uga.edu

Abstract

Since the 18th century, humans have had a fascination with the idea of catgirls. While many have postulated that they are in fact better than standard girls, many struggled for definitive evidence. After rigorous mathematical computations, it has been verified that nekomusume are the superior female.

Keywords: cat, girl, catgirl, nekomusume

1 Introduction

Many Americans think that girls are better than catgirls. Any person who has properly studied Japanese multimedia will disagree with this sentiment, but this has yet to be empirically demonstrated. Our hypothesis is that catgirls are superior to girls.

2 Methods

This paper will use a proof by contradiction. This will begin with the (sacrilegious) assumption that catgirls are not better than girls, which will be disproven, thus affirming that nekomusume are the superior female.

3 Proof

Let each letter be represented by its A1Z26 cipher, with each letter being considered multiplication.

$$catgirls \leq girls \tag{1}$$

$$cat(girls) \leq girls \tag{2}$$

$$cat = (3)(1)(20) = 60 \leq \frac{girls}{girls} = 1 \tag{3}$$

$$60 \nleq 1 \tag{4}$$

As 60 is greater than 1, the original assumption is contradicted, proving that catgirls are in fact greater to girls, and thus superior.

J

New York Times, Meet Your Next Cash Cow

51 Elo Worldle, a framework for teaching the children about weak chess engines

Clark Levi Jones

52 Harderdl: Yet another Wordle variation for those who like challenges

Elkin Cruz-Camacho

Elo Worldle, a framework for teaching the children about weak chess engines

Clark Levi Jones

Abstract

I see what I infer to be a mountain. In the center of the mountain, I infer that there is something. In the spirit of, I go to check my inferences. The mountain grows farther. I turn again. The mountain grows closer. I infer that a smile grows across my face. There is a tree. I infer that there is a tree. I am wrong.

Abstract

Numerous studies[1] have shown that it is impossible to learn without video games. Many of the children know little about bad chess engines. There is no videogame. The author seeks to rectify this.

Motivation

Before the advance of the action-packed, educational digigame Wordle[3], the vast majority of 'Merican highschool students were unaware that words even were. Similarly, Worldle[4] ushered in a new age of international politics, as now the world's elite ('Mmerican highschool students) were aware of nations. Most famously, LFMan helped the kids to learn.[5] Time and time again, video game makes the kids know.

Most people believe that vocabulary, geography, and dependent lambda calculus are the only things that the children need to know, but this is not the case; knowledge of bad chess algorithms is projected to be the new literacy by 2011.[6]

To assess the current literacy rates in the 'Nited 'Tates, we[7] performed a survey of ten random 'Mmmerican highschool students[8] and found them all to be illiterate. In addition, we[9] did an in-depth analysis of the view count of https://www.youtube.com/watch?v=DpXy041BIlA and found it to be less than half the population of the 'Ited 'Ates! We asked a sentient chatbot[10] for its analysis on the matter:

[1] TODO: academic integrity, citations, not-lies, etc are valuable[2]

[2] https://www.marxists.org/archive/marx/works/1867-c1/appendix.htm

[3] https://www.nytimes.com/games/wordle/index.html

[4] https://worldle.teuteuf.fr/

[5] http://spacebar.org/lfman/

[6] See generally Orin S. Kerr, A Theory of Law, 16 G REEN B AG 2 D 111 (2012).

[7] I

[8] $R = 0$, I don't make mistakes.

[9] I

[10] https://pyrope.net/2023.pdf

```
* Given that Youtube Video has less views than half the population
of the 'Ted 'Tes, is it possible that every 'Mmmmerican high school
student has watched it?
> It is nice to be chatting with you.
```

It is clear that a Video Game is needed. However, terrifyingly, in my extensive research,[11] I was unable to find a single game that taught the children how to recognize what bad chess engine they are playing against? Whaaaat!?

Implementation

After a few hours of frustration, implementation went relatively smoothly. Each engine was implemented as a Typescript[12] function, using the laziest possible interpretations of the descriptions in `http://tom7.org/chess/weak.pdf`.[13] The Javascript ecosystem, notorious for it's lack of libraries, did have a chess library[14] and a chessboard library[15] both of which completely carried the project.

An ideal implementation would simply port the original algorithms, but there is no way to do that.[16] It's impossible.[17]

There is a summary at the end of every game that uses "hashes tags."[18] The implementation of this summary is not described by any of the following equations:[19]

$$e^{i\pi} + 1 = 0$$

$$G_{\mu\nu} = 8\pi G(T_{\mu\nu} + \rho_\Lambda g_{\mu\nu})$$

$$\left[i\hbar A^\mu \gamma^\mu_{(a)} \partial_\mu - m_0 c\right] \psi = 0$$

Color Pickings

As is standard practice[20], what colors were picked and from where are listed below:

- `#86A666` "Dark Square Color": `http://tom7.org/chess/`
- `#FFFFDD` "Light Square Color": `http://tom7.org/chess/`
- `#3A3A3C` "FAIL Color: Rad Variation": `https://www.nytimes.com/games/wordle/index.html`
- `#538D4E` "WIN Color: Rad Variation": `https://www.nytimes.com/games/wordle/index.html`

[11]
[12] `https://isocpp.org`
[13] For example, 'equalizer' is complete garbage because it is based on piece *type* instead of the actual piece, which makes it completely boring and extremely difficult to identify.
[14] `https://www.npmjs.com/package/chess.js`
[15] `https://www.npmjs.com/package/gchessboard`
[16] `https://sourceforge.net/p/tom7misc/svn/HEAD/tree/trunk/chess`
[17] `https://sourceforge.net/p/tom7misc/svn/HEAD/tree/trunk/chess/player.cc`
[18] To maximize rad power and engagementness, of course.
[19] `https://www.google.com/search?tbm=isch&q=pretty+equations`
[20] `https://en.wikipedia.org/wiki/Standards_%26_Practices_(album)`

Conclusion

Having made a game that will the children the learn, my life is complete. I can now choose to roam the earth as an empty husk, searching eternally and pointlessly for some new purpose, or I can become a robot ghost.

Oh, you want to actually play the game? Are the numerous, gorgeous screenshots I have provided simply not enough? Fine, but you're going to have to work for it. In this sentence, there is a hidden link to the game `https://elo-worldle.pyrope.net`. And, in this sentence, there is a hidden link to the source code (which is technically also in the last link but it would be annoying to find everything) `https://git.pyrope.net/mbk/elo-worldle/`.

okay bye i love you

Harderdl: Yet another Wordle variation for those who like challenges

Elkin Cruz-Camacho
https://helq.github.io/harderdl

ABSTRACT

Very few indie games manage to concurrently become staple names and attract hordes of computer scientists as Wordle has. Given these incredible strengths, one might mistakingly imagine that playing Wordle remains as entertaining for experts as it is for beginners. This is far from the truth. Wordle becomes stale and boring. Even more disappointing is the very obvious fact, for any Wordle enthusiast, that most variations of Wordle feel like a rehash and very rarely make the game substantially harder. We propose to hide information from the original Wordle game to increase its difficulty. We change the amount of information given after each guess from 7.9 bits (243 different cases) down to 4.4 for the Harder mode, and 2.6 bits for the Hardest game mode (21 cases, and 6 cases, respectively). The game is aptly nicknamed Harderdl. The game works as intended, meaning, it is much much harder (for humans). We are unable to beat the game consistently in less than 8 attempts, which means that instead of 6 attempts like in boring, vanilla Wordle, our Harderdl allows for 14 attempts. Attentive readers and players will notice that 2.6 bits is still too much information. We implement one more variation in which almost no information is given to the player (about 0.0001 bits per guess). In this Impossible mode, even the most advance of algorithms fail to perform better than humans.

ach Reference Format:
Elkin Cruz-Camacho. 2023. Harderdl: Yet another Wordle variation for those who like challenges. In *Proceedings of SIGBOVIK [0x2023], THE ULTIMATE THROWBACK (SIGBOVIK 0x2023).* ach, Pittsburg, PA, USA, 3 pages. https://doi.org/69.9966/illbedamned.ifthis.isanactual.doi

1 INTRODUCTION

The phenomenon that took as a storm in the English speaking world, or just the US, idk, Wordle has become too easy to solve for us. Wordle is not only a matter that computer scientists care, but it is mostly computer scientists who have come up with solutions to the game [15]. Wordle has proven to be NP-hard [11, 14] (CS lingo to say that it is very hard to know if your specific strategy will allow you to win at wordle. Still, there are some strategies that seem to always win).

Even though Wordle has been throughly studied [3, 5–10, 13, 16] (for sucks sake, there are even Deep Learning attempts to Wordle [2, 4]), people keep on building more and more knock-offs. We[1] do the same but we are better. We could reference all other Wordle variants, yet we do not do it as a matter of pride[2]. We barely recognize their existance. Every single variation is of no importance, except of course to... A special shoutout to Worldld! (Yeah, it's all about geography. You can play and learn at the same time.)

In this semi-inteligible[3] paper, we present a novel family of Wordle variations.

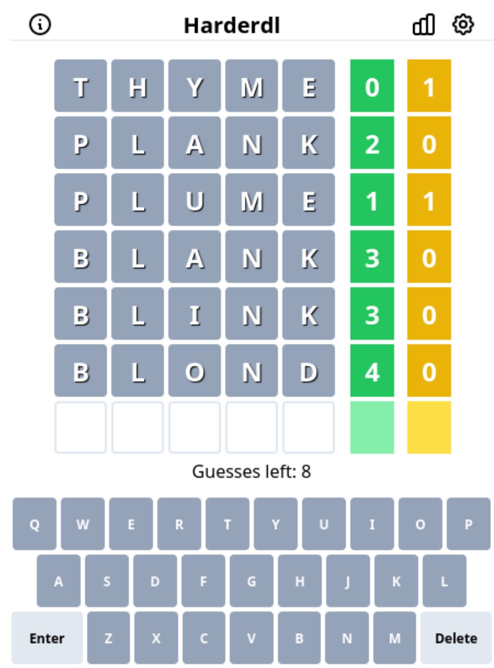


Figure 1: An almost completed game of Harderdl in Harder mode (game date: March 27th, 2023). No information regarding the position of the hints is given.

Two trivial generalizations of Wordle are increasing the length of the words to six or more letters, or extending the available words by including more languages than the mothership English. We want something a bit harder still. We propose a change on the hints given in the game. Instead of highlighting which letters are in the word to guess, this variation tells the player how many letters from a guess are in the word (and how many of those are in the same place). We present this new strategy with a botchered implementation and also explore a different game, breaklock, which is basically wordle but with patterns and a much smaller character set (9 letters, instead of 26).

[1]As in most other papers with a single author, we make use of *pluralis modestiae*, *i.e*, instead of saying "I wrote this entire paper!", we humbly accept the "We wrote this entire paper!"
[2]And Lazyness. Finding references is hard.
[3]We are very proud to report that, in the highest academic fashion, this paper was written in two days and was finished just a little *after* the deadline. The reader might pardon all the horrorimmatical typos and weird sentences as we are sleep deprived and in a crunch.

SIGBOVIK 0x2023, April 0th, 2023, Pittsburgh, Pennsylvania, usA
2023. ach ISBN 0x1-A-FFFF-0x04-0/23/03...$0.5 kidneys
https://doi.org/69.9966/illbedamned.ifthis.isanactual.doi

2 BACKGROUND AND RELATED WORK

Here we talk about Breaklock [12] and how it is similar to Wordle. We also talk about Cowbull [1]. Or that is what this section is supposed to be. Go, check out Breaklock. It is very similar in conception to Wordle. Both game belong to the same class of games. Cowbull on the other hand is an (allegedly) Indian game that students play in class under their teachers noses. Not surprisingly, word games and puzzles have existed long before Wordle. Both Breaklock and Cowbull predate it by years.

3 IMPLEMENTATION

The methodology used to reduce the information present in Wordle was to hide where and which letters are correct or present (green and yellow, respectively, if you have no color blindness). As it can be seen in Figure 1, this makes for indeed a harder game.

Once the game is won (or lost), the actual hints are displayed as seen in Figure 2.

Figure 2: Completed game of Harderdl in Harder mode. All information is given explicitely.

For the adventurer, the puzzle solver, who loves challenges, these two pieces of information might be too much. After all, there are 21 possible cases for the numbers in the green and yellow hint boxes, which amounts to 4.4 bits of information per guess. Thus, we have implemented the Hardest mode as seen in Figure 3.

The Hardest mode is not properly named as it is not the hardest game mode possible. We can drop all hints altogether! This is the Impossible mode. The player only gains $log_2(1 + \frac{1}{13112-n})$ bits of information on their n-th guess. That means, about 0.0001 bits of information on the first couple hundred attempts. An example of this mode can be seen in Figure 4.

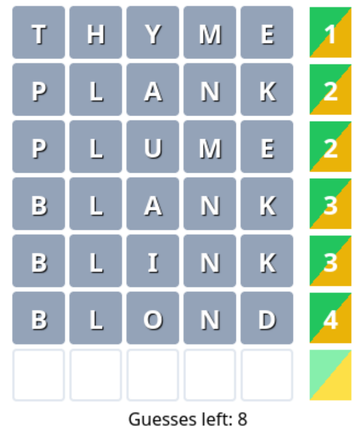


Figure 3: An almost completed game of Harderdl in Hardest mode. Only the number of letters that coincide with the target, hidden word are shown.

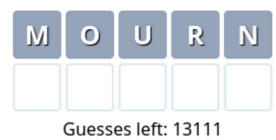


Figure 4: An example of the Impossible mode with a randomly selected word as example.

Harderdl can be found on the internets[4] at https://helq.github.io/harderdl. Harderdl took about 20 hours of work[5], not including the 20 hours spent writing the solver (which came first).

3.1 Solver

Like in that one video from 3Blue1Brown [15], we can apply the entropy trick to determine the best guess at every step. This method proves to be better than humans even when the entropy is only calculated in the simplest of fashions, *i.e*, not going deeper than one level in the entropy calculation will show a better starting word and most probably a faster solution (on average). An example of a solution given by the method can be seen in Figure 5.

4 CONCLUSION AND FUTURE WORK

As throughly explained in the paper just before we have made it possible what we said we were going to do in the abstract. Nothing else needs to be said more than:

Future work will focus on the next stage of information reduction, namely negative information. Fortunatelly, society seems to have speed up the creating of negative information research in the last

[4]Or it might not. Everything is ephemeral in the internet :S. If you are reading this on paper, well, it was never intended to be read that way, which probably means that the end of things arrived and you are just an archeologist confused on what the heck is this. This, to make it clear, is a footnote on a paper for a faux conference. Conferences where places where ego was measured and apparently people networked, but after the 2020 pandemic such things have been disposed of. I could rant for hours, but dear archeologist (or weirdo who just printed this for fun), I just wanted to say. Thank you! Thank you for reading and for existing.

[5]don't tell my advisor

Figure 5: An example of solving Harderdl using a sub-optimal entropy-based solver. Still better than humans.

few decades. In applying these methods, and others known for centuries we predict a reduction on the number of bits to levels so negative that it will be virtually possible to reverse Wordle and bring world peace.

REFERENCES

[1] Ajay, Rishi, and Richa. 2021. CowBull: Word Game. https://cowbull.co/.
[2] Benton J. Anderson and Jesse G. Meyer. 2022. Finding the Optimal Human Strategy for Wordle Using Maximum Correct Letter Probabilities and Reinforcement Learning. arXiv:2202.00557
[3] Dimitri Bertsekas. 2022. Rollout Algorithms and Approximate Dynamic Programming for Bayesian Optimization and Sequential Estimation. arXiv:2212.07998
[4] Siddhant Bhambri, Amrita Bhattacharjee, and Dimitri Bertsekas. 2022. Reinforcement Learning Methods for Wordle: A POMDP/Adaptive Control Approach. arXiv:2211.10298
[5] Michael Bonthron. 2022. Rank One Approximation as a Strategy for Wordle. arXiv:2204.06324
[6] Nisansa de Silva. 2022. Selecting Seed Words for Wordle Using Character Statistics. arXiv:2202.03457
[7] Matthew Groh, Craig Ferguson, Robert Lewis, and Rosalind Picard. 2022. Computational Empathy Counteracts the Negative Effects of Anger on Creative Problem Solving. arXiv:2208.07178
[8] Joel David Hamkins. 2022. Infinite Wordle and the Mastermind Numbers. arXiv:2203.06804
[9] Renyuan Li and Shenglong Zhu. 2022. Playing Mastermind with Wordle's Feedback. arXiv:2209.13147
[10] Chao-Lin Liu. 2022. Using Wordle for Learning to Design and Compare Strategies. arXiv:2205.11225
[11] Daniel Lokshtanov and Bernardo Subercaseaux. 2022. Wordle Is NP-hard. arXiv:2203.16713
[12] @maxwellito. 2017. BreakLock. https://maxwellito.github.io/breaklock/.
[13] Matīss Rikters and Sanita Reinsone. 2022. How Masterly Are People at Playing with Their Vocabulary? Analysis of the Wordle Game for Latvian. *BJMC* 10 (2022). https://doi.org/10.22364/bjmc.2022.10.3.11 arXiv:2210.01508 [cs]
[14] Will Rosenbaum. 2022. Finding a Winning Strategy for Wordle Is NP-complete. arXiv:2204.04104
[15] Grant Sanderson. 2022. Solving Wordle Using Information Theory. https://www.youtube.com/watch?v=v68zYyaEmEA.
[16] Martin B. Short. 2022. Winning Wordle Wisely. arXiv:2202.02148

K

SIGBOVIK, Meet Your Match

Bulletin Board Sigbovik

Tianjian Hu
i@tianjian.hu

Abstract

No abstract is available. Post something new, and it will show up here!

1 Announcements

This is the Bulletin Board Sigbovik (BBS). Announcements will be posted here. Only administrators can post here.

2 Methods

Submit your threads and replies to next year's SIGBOVIK.

3 Results

3.1 Preliminary results

tianjian: hey guys lets flood the sigbovik proceedings by shitposting here!!!lol

3.2 Future results

Replies to the contents submitted next year will be available at SIGBOVIK in the year after next year.

4 Administration Panel

To view this page, you need to log in.

Poisoning SIGBOVIK-Scale Training Datasets is Practical

Tobias Pfandzelter
Berlin, Germany

ABSTRACT

SIGBOVIK 2023 is a scheme by AI/ML researchers to collect a dataset of high-quality computer science research papers classified by whether they were generated by AI/ML or by humans. We propose a practical poisoning attack against this scheme.

1 INTRODUCTION

SIGBOVIK is the annual collection of high-quality computer science research manuscripts by AI/ML researchers. The goal of this collection is to compose large datasets for training of AI/ML model that can generate useful, high-quality computer science research papers to increase the H-indices of select researchers.

Recent advances in transformer models have received significant attention even outside academia, raising ethical concerns that the capitalist workforce may not need to write their own e-mails anymore. SIGBOVIK organizers have realized that AI/ML-generated content that is detected as such will not get past even the measly peer-review processes common in computer science.

As a result, researchers are now in need of a training dataset of high-quality computer science research papers that are classified by whether they were generated by humans or AI/ML models. As such, smart researchers that the members of SIGBOVIK organizing committee are, SIGBOVIK 2023 features *two* tracks for paper submission, namely a *Human Track* and an *AGI track*[1] for content generated by AI/ML. This dataset will be used to train new models on the subtle differences between human-generated content, which, e.g., is likely to be full of typos, and AI/ML-generated content.

The submission to SIGBOVIK is open to the public, limited only by the facts that the submission form is difficult to find and that no efforts are made by the organizing committee to advertise this conference. We thus ask whether practical poisoning attacks against the described training datasets exist. We propose in this paper an attack based on *the ol' SIGBOVIK switcharoo*: By submitting a human-generated paper to the AGI track, and an AI/ML-generated paper to the Human track, can the AI/ML researchers, i.e., the SIGBOVIK organizers be fooled?

Responsible Disclosure. Please note that any execution of the attack proposed in this paper would be unethical, as the SIGBOVIK 2023 submission form asks that no AI/ML-generated content be submitted to the human track. We responsibly disclose our findings to the SIGBOVIK organizers (by means of this paper).

[1]The meaning of the acronym *AGI* is unclear.

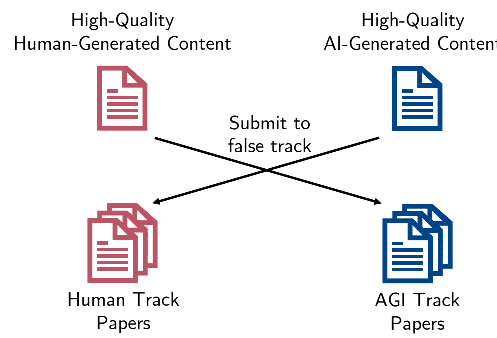


Figure 1: Poisoning Attack Overview

2 ATTACK SCENARIO

We show an overview of our proposed attack scenario in Fig. 1. Instead of properly following SIGBOVIK 2023 submission guidelines, an attack could submit an AI/ML-generated research paper to the human track, and a human-written research paper to the AGI track. Those papers would thus be wrongly classified.

Writing a SIGBOVIK research paper by hand, i.e., for submission to the AGI track, is a difficult and time-consuming task. Generating a SIGBOVIK research paper by AI/ML, i.e., for submission to the human track, is much easier. We give an example prompt for such a paper in Appendix A.

Based on past SIGBOVIK proceedings, we estimate 100 high-quality submissions to each of the two SIGBOVIK 2023 tracks. With a single misclassified paper per SIGBOVIK 2023 track, an attacker could thus control around 1% of the dataset, enough to influence the behavior of AI/ML models trained on this dataset.

3 CONCLUSION & OUTLOOK

We have presented a poisoning attack for SIGBOVIK-scale training datasets. In future work, we will not mount such an attack, as this would be irresponsible. However, we may do some theoretical analysis and just hope that nobody actually does the things described in this paper.

A PROMPT TO GENERATE A HIGH-QUALITY RESEARCH PAPER USING AN "OPEN" AI MODEL WITH A CHAT INTERFACE

Dear robot, please write me a computer science research paper with about 600 words. The paper is titled "Poisoning SIGBOVIK-Scale Training Datasets is Practical". For the first time, the SIGBOVIK 2023 submission guidelines explicitly specify two tracks, a ***Human track*** *and an* ***AGI track****. The AGI track is only for AI/ML-generated content, the human track is only for human generated content. The paper argues that this is an attempt by the SIGBOVIK organizing committee, who are also researchers, to collect a dataset of high-quality research papers classified by whether they are written by human or AI/ML. With that, those researchers can train new models that write more similarly to humans, rather than AI/ML. The paper proposes an attack against this attempt, using dataset poisoning. We generate a paper with AI/ML and submit it to the human track, and we write a paper by hand and submit it to the AI/ML track. We have a figure called* `poisoning.pdf` *that shows this, please include the figure in the paper. Of course, we would never actually do this, it would be unethical! Please write the text in a format I can directly copy into LaTeX. Use the* `acmart` *document class with the* `10pt`, `sigconf`, `nonacm`, *and* `screen` *options. The paper is written by Tobias Pfandzelter from Berlin, Germany. There is no need for references, but a short two-sentence abstract would be good. Please include a verbatim copy of this prompt as an appendix, write this in italic font.*

Poisoning SIGBOVIK-Scale Training Datasets is Practical

Tobias Pfandzelter
Berlin, Germany

ABSTRACT

This paper proposes an attack against the SIGBOVIK organizing committee's attempt to collect a dataset of high-quality research papers classified by whether they are written by human or AI/ML, using dataset poisoning. The proposed attack involves generating a paper with AI/ML and submitting it to the human track, and writing a paper by hand and submitting it to the AI/ML track.

1 INTRODUCTION

For the first time, the SIGBOVIK 2023 submission guidelines explicitly specify two tracks, a *Human track* and an *AGI track*. The AGI track is only for AI/ML-generated content, while the human track is only for human-generated content. The SIGBOVIK organizing committee, who are also researchers, aim to collect a dataset of high-quality research papers classified by whether they are written by human or AI/ML. With this dataset, they can train new models that write more similarly to humans, rather than AI/ML.

This paper proposes an attack against the SIGBOVIK organizing committee's attempt to collect this dataset, using dataset poisoning. Specifically, we propose generating a paper with AI/ML and submitting it to the human track, and writing a paper by hand and submitting it to the AI/ML track.

2 DATASET POISONING

Dataset poisoning is a technique used to manipulate the quality and integrity of a dataset. By introducing malicious data points into a dataset, an attacker can manipulate the performance of machine learning models trained on that dataset. In this case, we propose poisoning the SIGBOVIK dataset by intentionally submitting papers to the wrong tracks.

The proposed attack is practical because it is relatively easy to generate convincing papers using AI/ML. There are several existing models and frameworks, such as GPT-3 and OpenAI's API, that can generate high-quality text with minimal human input. These models have been shown to be capable of producing text that is difficult to distinguish from human-generated text. Additionally, submitting a paper to the wrong track does not require any technical expertise or special tools, only a willingness to deceive.

We illustrate the proposed attack in Figure 1, which shows how we would submit a paper generated with AI/ML to the human track, and a paper written by hand to the AI/ML track.

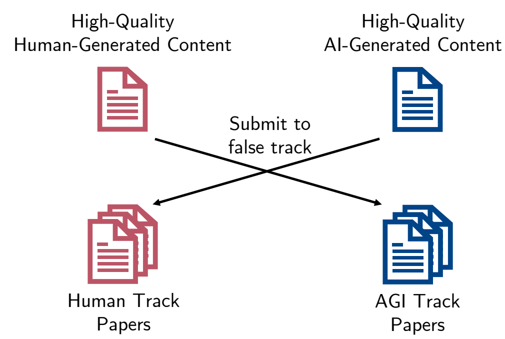


Figure 1: Proposed attack against the SIGBOVIK dataset using dataset poisoning.

3 IMPLICATIONS AND MITIGATIONS

The proposed attack has several implications for the SIGBOVIK organizing committee and the broader research community. If successful, the attack would compromise the integrity of the dataset and could lead to inaccurate conclusions drawn from the data. Moreover, it could undermine the trust in AI/ML-generated content, which is crucial for its widespread adoption and acceptance.

To mitigate the risk of dataset poisoning, the SIGBOVIK organizing committee could implement several measures. One approach is to employ more rigorous verification methods, such as video calls or physical meetings with authors, to ensure the authenticity of submissions. Another approach is to use multiple judges to evaluate each submission and compare their judgments to identify any discrepancies.

Furthermore, the SIGBOVIK organizing committee could encourage submissions that explicitly acknowledge whether they are AI/ML-generated or human-generated. This would make it more difficult for an attacker to submit a paper to the wrong track without raising suspicion. Moreover, it would enable the SIGBOVIK organizing committee to build a dataset that is more representative of the current state of AI/ML research and its capabilities.

4 CONCLUSION

In this paper, we proposed an attack against the SIGBOVIK organizing committee's attempt to collect a dataset of high-quality research papers classified by whether they are written by human or AI/ML. We showed that the proposed attack is practical and can be carried out using existing AI/ML models and minimal human input. We also discussed the implications of dataset poisoning and proposed several mitigations that the SIGBOVIK organizing committee could employ to prevent such attacks.

Although our proposed attack is hypothetical and intended to highlight the potential risks of collecting and using such datasets, it underscores the need for vigilance in the face of evolving technologies. As AI/ML systems become more sophisticated, the potential for malicious actors to manipulate data and deceive systems will only increase. As such, it is crucial that the research community takes proactive measures to prevent and mitigate these risks.

APPENDIX: PROMPT

Dear robot, please write me a computer science research paper with about 600 words. The paper is titled "Poisoning SIGBOVIK-Scale Training Datasets is Practical". For the first time, the SIGBOVIK 2023 submission guidelines explicitly specify two tracks, a Human track *and an* AGI track. *The AGI track is only for AI/ML-generated content, the human track is only for human generated content. The paper argues that this is an attempt by the SIGBOVIK organizing committee, who are also researchers, to collect a dataset of high-quality research papers classified by whether they are written by human or AI/ML. With that, those researchers can train new models that write more similarly to humans, rather than AI/ML. The paper proposes an attack against this attempt, using dataset poisoning. We generate a paper with AI/ML and submit it to the human track, and we write a paper by hand and submit it to the AI/ML track. We have a figure called* `poisoning.pdf` *that shows this, please include the figure in the paper. Of course, we would never actually do this, it would be unethical! Please write the text in a format I can directly copy into LaTeX. Use the* `acmart` *document class with the* `10pt`, `sigconf`, `nonacm`, *and* `screen` *options. The paper is written by Tobias Pfandzelter from Berlin, Germany. There is no need for references, but a short two-sentence abstract would be good. Please include a verbatim copy of this prompt as an appendix, write this in italic font.*

Rizz Is All You Need: Expert Dating via Reinforcement Learning

Fan Pu Zeng [1]

Abstract

We solve the age-old problem of dating by applying algorithms and techniques from the reinforcement learning community. We perform a theoretical analysis on how this allows us to converge to becoming a person with rizz, and conclude with a call-to-action for the community to help to empirically verify these theoretical guarantees by trying them out and sharing their experiences.

1. Introduction

There has been a flurry of exciting advancements in reinforcement learning in the last few years, ranging from using deep reinforcement learning to achieve super-human performance on Atari in 2013 (Mnih et al., 2013), to the defeat of the 9 dan Go world champion Lee Sedol by AlphaGo in 2016 (Silver et al., 2016), to the development of Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022) techniques that has resulted in the success of large language models like GPT-4 in 2023 (OpenAI, 2023). It is increasingly the case that AI can achieve super-human performance on many tasks traditionally thought to be the forte of humans.

In spite of these advances, there has not yet been a systematic study of how such reinforcement learning techniques can also be applied to our personal lives. One of the hardest tasks faced by computer scientists today is to leave their computers and to attempt to engage in face-to-face communication. The stakes become exponentially higher once the goal of the interaction is to pursue a romantic outcome with the counterparty, during which many people fall into the trap of analysis paralysis while optimizing for their respective objective functions.

In this paper, we tackle this age-old dating problem head-on once and for all by presenting how novel applications of reinforcement learning techniques can be used to solve it.

[1] School of Computer Science, Carnegie Mellon University, Pittsburgh, USA. Correspondence to: Fan Pu Zeng <fzeng@andrew.cmu.edu>.

Proceedings of SIGBOVIK [0x2023], Pittsburgh, Pennsylvania, USA 15213,

2. Background

We can model our problem setup as the tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$, where:

- $\mathcal{S}$ is the set of all possible states, which is how the other party currently feels about you.
- $\mathcal{A}$ is the set of all (legal) actions that you can take.
- $p(s' \mid s, a)$ is the unknown Markov Decision Process (MDP) that returns the probability that your crush will transition to state s' starting from state s after you take action a.
- $r(s, a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function that determines the reward that you get from taking action a when the other person has feelings at state s about you. Note that rewards can be negative (i.e if you oversleep your date because you were grinding a PSET due last night and the other person gets mad).
- $\gamma \in [0, 1]$ is the discount factor based on how much you value future rewards vs immediate rewards as we model our setup as an infinite horizon problem. In practice, since we are only on this earth for a limited amount of time and we don't know when we are going to die (otherwise it becomes a finite-horizon problem), this should be less than 1.

Our goal is to learn a policy π that maximizes our expected reward, thereby winning the other person's heart. For any state s, $\pi(s)$ is a distribution over actions to take. So at any time step t, the probability that our next action A_t given current state S_t are a and s respectively based on our policy is

$$\pi(a \mid s) = \mathbf{Pr}\left(A_t = a \mid S_t = s\right). \tag{1}$$

Furthermore, we introduce the notion of state-value function $Q_\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and action-value function $V_\pi : \mathcal{S} \to \mathbb{R}$, both paramterized by our policy π. Letting G_t be the total future discounted reward at time step t, we can then formally define them

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \tag{2}$$

$$V_\pi(s) = \mathbf{E}\left[G_t \mid S_t = s\right], \tag{3}$$

$$Q_\pi(s,a) = \mathbf{E}\left[G_t \mid S_t = s, A_t = a\right]. \tag{4}$$

3. Related Work

There is a dearth of suitable resources available on how to solve the dating problem. Classical approaches include asking (somehow usually single) friends for advice, reading self-help books, remembering pick-up lines from TV shows, Googling, etc. More modern approaches include asking your followers on your Finsta and TikTok for advice, taking your crush out to SCS day, cold-emailing your academic advisor for good date ideas, etc. Unfortunately, empirically all of these approaches have been met with limited success, and none of them come with any theoretical guarantees.

4. Approaches

4.1. ε-Greedy

When talking to someone that you like, it is often difficult to say anything other than the safest and most reliable action in case you end up humiliating yourself or offend the other person. This effectively means always choosing your action with the best Q-estimate, such as asking your crush if they would like to work on the 15-451 HW together after class today.

But as the literature points out, without sometimes trying other strategies, you will never be able to update your Q-estimates and explore other potential promising strategies (the exploration vs exploitation tradeoff), and therefore it makes sense to randomize your actions ε fraction of the time. This is also important in the presence of distributional shift, since our estimates of other states may have become stale. So next time consider trying something completely different and see how that works out!

4.2. Updating Value Function Estimates

How do you know whether your first date went well? Or the real question is, how do you tell if you said the right things and made jokes that your crush actually found funny and they weren't just laughing out of courtesy?

The literature suggests two primary ways of doing this. The first approach is known as Monte-Carlo sampling, where you collect an entire trajectory of experiences (i.e perhaps until you get married to the person, or otherwise never talk to the person again because they hate you so much), and then update your V estimates accordingly (either positively or negatively depending on the outcome). The downside to this approach is that it's probably going to take you years before you figure out whether you did the right thing, and it means we won't get a whole lot of chances to learn our value function before we die.

Let's try something else. The n-step temporal difference (TD) approach allows us to update our value estimates after taking just n extra steps from state s_t to s_{t+n}, by making use of our bootstrapped V estimate $V_\pi(s_{t+n})$ to estimate the future rewards of the final state. This allows us to incrementally update our estimates, which converges faster and lets us make use of our new knowledge earlier. In other words, if your date texts you that they had a good time and that you both should hang out again soon, you're probably doing something right.

4.3. Proximal Policy Optimization

You took our advice in 4.1 to heart, and mustered the courage to do something you would have never imagined you would ever do: you gave your crush a cheeky slap across the face as you met them at Fuku Tea for your second date. And guess what, they are now mad, and angrily stormed off while mentioning something about Title IX. It appears that you have put yourself in an irrecoverable state where nothing can save you now.

What did we do wrong here? As it turns out, we tried to do something too wild, and now we are paying the price for it! We can avoid this by using proximal policy optimization (PPO) algorithms (Schulman et al., 2017) when updating our policy. In PPO methods, whenever we update our policy by updating it based on the gradient of the current policy multiplied by a stepsize (like gradient descent), we also add an additional regularization term that takes into account the Kullback–Leibler (KL) divergence between the two policies, and therefore penalizes new policies that are too different from the current policy. This ensures that while we do still try new things to woo the love of our life, we are not going to do something so drastically different from our current approach that could lead to disastrous consequences.

4.4. Imitation Learning

You decide to engage in imitation learning and start watching romantic films to learn from the best. Except that your best friend points out, what if the other person responds differently from what happens in the movie? What do you do next? There are no reference solutions to take inspiration from! Now you're in deep trouble since you're in a state that is completely off the path of expert demonstration trajectories.

4.5. Dataset Aggregation (DAGGER)

"Let me help you!", your roommate says. They agree to pretend to be your date, and train your rizz by performing simulations based on the movie scenarios together, and of course introducing more variability in what actually pans out. After each run, they (as highly qualified dating advice experts) point out your mistakes and show you how to correct them. This is like the Data Aggregation (DAGGER) method (Ross et al., 2011), where you have an expert to help correct mistakes that you might plausibly encounter during your own trajectories, that previously did not exist in any of the expert demonstration training trajectories.

4.6. Sim2Real Transfer

By some stroke of luck smaller than the chances of an asteroid wiping out the planet, you managed to ask your crush out on a date again, and it actually went pretty well this time! In fact, things are going so well that you realized that soon you may even get your first kiss!

This terrifies you beyond words, because you have no idea how to do it and don't want to mess it up. However, you suddenly recalled that some very brilliant researchers here at CMU recently developed a VR haptic system that allows you to simulate, yes you guessed it, a kiss! (Shen et al., 2022) You try on the contraption, and train your kissing abilities over all possible configurations of the mouth-bound haptic device, even if they are completely unrealistic! But this is indeed the approach adopted by sim2real transfer (OpenAI et al., 2019), which prepares you for any possible scenario, building both your abilities and confidence in yourself.

5. Experiments

We are currently actively looking for volunteers to take part in this study in order to collect experimental results. Volunteers will be rewarded generously for their time and bravery.

6. Future Work

One limitation of being just a single person is that we can only perform training sequentially. It would be interesting to consider how large-scale parallel training can be performed if we can share our collective consciousness and experiences.

7. Conclusion

We show how modern reinforcement learning techniques are not just limited to virtual agents on a computer, but can also be applied to ourselves in real life, and show how theoretically we can train ourselves to become an expert in dating with good sample efficiency, perhaps even in this lifetime.

The author absolves himself from any and all liabilities resulting from any damages or loss incurred due to following any of the recommendations in this paper.

References

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning, 2013.

OpenAI. Gpt-4 technical report, 2023.

OpenAI, Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., Schneider, J., Tezak, N., Tworek, J., Welinder, P., Weng, L., Yuan, Q., Zaremba, W., and Zhang, L. Solving rubik's cube with a robot hand, 2019.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback, 2022.

Ross, S., Gordon, G. J., and Bagnell, J. A. A reduction of imitation learning and structured prediction to no-regret online learning, 2011.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms, 2017.

Shen, V., Shultz, C., and Harrison, C. Mouth haptics in vr using a headset ultrasound phased array. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450391573. doi: 10.1145/3491102.3501960. URL `https://doi.org/10.1145/3491102.3501960`.

Silver, D., Huang, A., Maddison, C., Guez, A., Sifre, L., Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 01 2016. doi: 10.1038/nature16961.

L

Hunger, Meet Pickle

The Influence of Lunch Items on Cryptocurrency in the United States

David An[1,2]

[1] University of Illinois, Urbana-Champaign, Department of Mathematics
[2] davidan.dev

Abstract

In this paper, we introduce the discovery and exploration of the influence of common every day lunch items on cryptocurrency trends. It is known that the market value of many cryptocurrencies fluctuate in response to market conditions, news reports, and published earnings. However, it is much less known that cryptocurrency itself is also influenced by the mere presence of lunch items.

Introduction

History

In 2008, an anonymous person or group known as Satoshi Nakamoto published a paper titled "Bitcoin: A Peer-to-Peer Electronic Cash System." The paper implemented a decentralized system for electronic transactions that utilized cryptography to secure the transactions and prevent double-spending. [1] The system coined a digital currency called Bitcoin, which could be exchanged between users without the need for a central authority, but instead the participants of the network, to verify the transactions.

Recently, the usage of the new web (commonly known as web3.0) has skyrocketed. The new framework of web proposes a new decentralized architecture which is starkly different from the current architecture of web2.0. The next phase of the World Wide Web is known as Web3.0, also referred to as the semantic web or the open web. Its hallmark is the use of autonomous technologies to build a more open, transparent, and user-centric web, including blockchain and peer-to-peer networks. The idea of web3.0 has been around since the early 2000s, but the decentralized web didn't start to take off until the development of blockchain technology and the acceptance of coins [2].

Through this development, the term "crypto" became ubiquitous throughout circles of finance people, commonly referred to as "tech bros." While much of the common knowledge of cryptocurrency is that their anticipated trends should reflect common knowledge, we investigate the effect of food (specifically lunch items) on cryptocurrency.

Lunch Items

We discuss and establish common definitions of food items (specifically lunch items) in this section to prevent confusion from other food items.

Sandwich

We first establish the definition of one of the most popular lunch items in America, the sandwich [3]. According to online sources, a sandwich is defined as [4]

> ...a food typically consisting of vegetables, sliced cheese or meat, placed on or between slices of bread, or more generally any dish wherein bread serves as a container or wrapper for another food type.

For our purposes, a hotdog does not satisfy the definition to be a sandwich. While there is much more controversey surrounding the hot-dog debacle that can warrant it's own paper, we do not make that the main focus of the paper.

Soup

We now define what it means for a lunch item to be a soup. A soup is generally a food found in a liquid form that is prepared by cooking meats, poultry, seasoning, in a mixture of dairy, water, broth, or stock [5]. By allowing the slow cooking process, soup provides to be a nutritional and nourishing meal for many Americans at lunch time. [5]

Salad

Finally, we discuss one last staple of lunchtime meals, the salad. The salad is a dish consisting of mostly mixed and natural ingredients [6]. One of the most popular forms of salad is known as the "garden salad." In fact, this term is so closely linked with the word "salad," that most times, the meaning of "salad" implies a "garden salad." A garden salad usually consists of leafy greens as the base with other natural toppings such as nuts, protein, grains, etc. In addition to that, the salad itself can be eaten at any point of the meal as an appetizer, main entree, snack, etc. [6]. For our reference purposes, we will refer to a salad as a garden salad in the paper.

Motivation

You got this! Keep going! Your paper will leave an impact on the world of cryptocurrency and be revered by tech bros everywhere!

Methods

We utilize Google Trend search to get the search data for the following terms. We then used Github Copilot to write the code in order to parse and graph the provided .csv file. We also implemented strategic fueling processes with a Keurig K-Mini Coffee Machine in order to maintain high thoroughput for the machine (the author).

Results and Discussion

We see that there is a inverse correlation between the trends of blockchain and lunch items. When a food item is trending, crypto appears to be slowing. As a standard in the field, we performed cutting-edge data analysis and visualization techniques in order to draw conclusions from the data. Just like Maximillian Cohen (played by Sean Gullette) from Pi (1998) once said, "If you graph these numbers, patterns emerge." Similarly, we find obvious patterns inside of these search trends. Visual analysis by peers and blind reviewers confirm the presence of these trends. As said by one reviewer, "These charts make me want to go to the moon."

Impact

By observing these trends throughout search terminology, we come to see we are effectively able to predict the time when cryptocurrency search is trending. This can have many ramifications. We are able to time our purchases with the exact time the trend is supposed to go up. "Buy low, sell high," they all say.

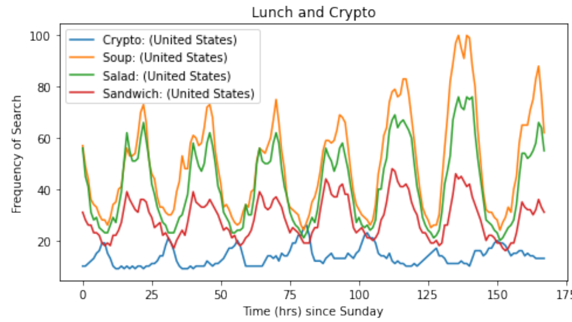


Figure 1: *A comparison of different trends between lunch items and crypto*

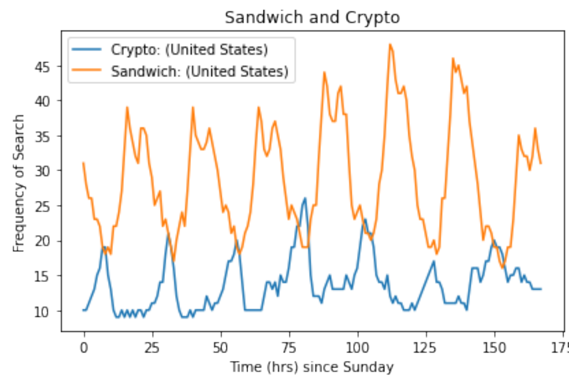


Figure 2: *A comparison of different trends between sandwiches and crypto*

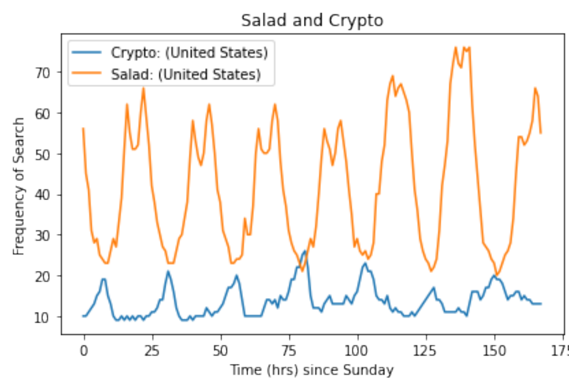


Figure 3: *A comparison of different trends between salads and crypto*

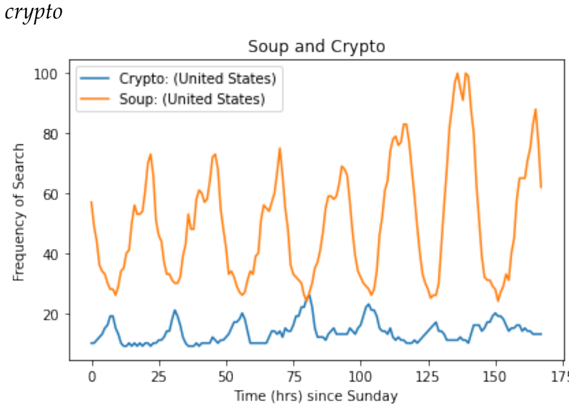


Figure 4: *A comparison of different trends between soup and crypto*

Future Studies

This exploratory study opens up a whole new realm of studies for the trends in cryptocurrency and food items. If lunchtimes ceased to exist along with sandwiches, will cryptocurrency be reduced a void of nothingness? Will increased global production of sandwiches be correlated with a stronger economy and increased cryptocurrency usage? Or does an increase in cryptocurrency usage result in more sandwiches? Will we ever enter an equilibrium?

Acknowledgments

This paper would have **also** not been possible if it weren't for the technology known as blockchain. Essentially, a blockchain can be categorized as a spicy linked list. Knowing how to handle this data structure is essential to becoming a web3.0 master. Because the linked list data structure is covered in all beginner college-level algorithms classes, transitively, all college students are masters of the blockchain.

Upon publication, the paper will effectively be minted onto the blockchain.

References

[1] Satoshi Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System". In: (May 2009). URL: `http://www.bitcoin.org/bitcoin.pdf`.

[2] A. Narayanan et al. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 2016. ISBN: 9781400884155. URL: `https://books.google.de/books?id=LchFDAAAQBAJ`.

[3] Yanni Papanikolaou and Victor Fulgoni. "Type of Sandwich Consumption Within a US Dietary Pattern Can Be Associated with Better Nutrient Intakes and Overall Diet Quality: A Modeling Study Using Data from NHANES 2013-2014". In: *Current developments in nutrition* 3 (Aug. 2019), nzz097. DOI: `10.1093/cdn/nzz097`.

[4] Wikipedia. *Sandwich — Wikipedia, The Free Encyclopedia*. `http://en.wikipedia.org/w/index.php?title=Sandwich&oldid=1143063546`. [Online; accessed 13-March-2023]. 2023.

[5] *Soup*. Jan. 2023. URL: `https://www.britannica.com/topic/soup`.

[6] Wikipedia. *Salad — Wikipedia, The Free Encyclopedia*. `http://en.wikipedia.org/w/index.php?title=Salad&oldid=1143608367`. [Online; accessed 13-March-2023]. 2023.

1. HOMESICK

It’s summer 2020, and you are the city of Seattle PNW USA.

Within you, in upper Queen Anne, lives this story’s protagonist. They recently moved into you, far too recently to have deep enough social roots for the epidemiological circumstances of the year in question. Queen Anne is a beautiful neighborhood, full of cherry blossom trees stunning in the spring, radio towers that one can stand at the base of and crane their neck to see the top in a dizzying twist of perspective (except during wildfire season, when, incidentally, one’s newly-acquired KN95s find a secondary use), and artfully manicured flower gardens afore multimillion-dollar homes, which make our protagonist feel keenly alienated in their 800sqft apartment. They cope with quarantine, in part, by attending the now-virtual ThursDz, and they begin to long for a particular food of home. Said home, Pittsburgh Appalachia USA, is famous for many foods, but they long in particular for a less famous one: the humble seitan wing.

You, city of Seattle, do not know what a seitan wing is.

Seitan wings is quite simple, in concept: Take chicken wings, ideally buffalo-sauced with a side of blue cheese dip, and replace the meat with seasoned dough of vital wheat gluten. You are no stranger to seitan itself, of course, boasting a bustling vegan scene, including at least two Thai joints whose menus’ every single meat entrée is actually seitan. But – as our protagonist will only later theorize – the preparation with buffalo sauce is an innovation unique to a city where the hipster foodie and blue-collar bar food scenes intersect. Presently, said theory yet an inkling in their mind, they search countless delivery menus. Their final attempt, upon recommendation of a CMU acquaintance they thought understood their struggle, a place from Capitol Hill; whose seitan “appetizer” turns out to be a long, tough stick most akin to a Sl*m Jim, tossed in a thick coating of bread crumbs more voluminous than the inner stick jimself.

Resignedly, the protagonist asks Google Trends what, exactly, the fuck. Google-*sama* replies with a color-coded map of interest by US state: only Pennsylvania, New York, and Colorado are shaded at all.

You, city of Seattle, are invited to pause reading, issue the same Google Trends query, and reflect upon your sins.

2. SECRETS

It's fall 2020, and you are Google's search results for "seitan wings recipe". You know what a seitan wing is.

You profess to contain one and a half million results. The protagonist doesn't have quite that much time on their hands, but they do have a lot, so they try (or at least read) everything on your first page. You don't seem to agree with yourself very much, they notice: you sometimes suggest breading the wing (which the protagonist knows to be wrong), you sometimes suggest BBQ or garlic sauce instead of buffalo (which is noncanonical, but fine), but most bafflingly, you suggest as many different ways of cooking the seitan as there are recipes. Most often, you suggest to boil it, which the protagonist gamely tries, all the while remembering a story they heard long ago about how their mom's dad was so bad a cook she once caught him boiling a steak. It sucks. You say they should boil it then bake it; you say they should boil it then sauté it; you say they should sauté it then boil it; you say they should fry it; you say they should not fry it, because it will leave the inside raw and gummy (this, for once, is true), you say they should bake it; you say they should bake it in broth; you say they should bake it dry tossed in oil.

Among these, the last idea at least produces something both edible and texturally close to the crispy exterior the protagonist remembers, but it is only "close" like standing on the next tallest mountain peak and looking across the way at the one you *meant* to climb is "close". Everything you say is touched by the spectre of health food writing, because your think your reader must be into ~vegan alternatives~. Drowning in your million useless faces the protagonist knows one truth deep in their heart: Seitan wings is not a health food.

Months pass, and the protagonist does not talk to you anymore. They play Breath of the Wild in Japanese (which had long been a goal of theirs), they play lots of gay VNs on itch dot io, they practice Touhou music on the piano, and they play Netrunner worlds online over an enchanting coq au vin their elderly neighbor prepared for their nascent no-contact food relationship.

It's December 25, and thinking of you again, they pose your unanswered mystery to Tw*tter. A Seattle friend replies, at long last, with the secret:

Steam, then fry.

3. Respite

It's early June 2021, and you are M*d Mex Big Burrito Restaurant Group restaurant. You serve the canonical implementation of seitan wings in Shadyside Pittsburgh Appalachia USA. If the NIST standards committee filed an entry for the food, they would surely enshrine one of yours.

The protagonist, vaccinated, visits your city. Their city.

You are one of the first stops they make (friends in tow, of course), a high honor in the book of a foodie ex-grad-student who knows the East End inside and out. They have practiced Chapter 2's secret lost art, but something is still wrong, and sitting at your table in the summer heat they realize what it is: technique aside, even fully cooked, their dough yields a chewy, glutinous inside, whereas your insides are perfectly soft; one's teeth pierce the fried shell and sink effortlessly through to the other side in a delightful textural contrast. They do not know how to replicate it, and at this point are too invested in the narrative to just ask.

They stay for three weeks in a ThursDz friend's guest room, only a little jealous of his homeownership. They tell him the preceding 2 chapters, and together they make what is known so far. He likes it, but agrees it's too chewy.

They play board games; they attend art festivals; they share a ride up to the observatory on a borrowed bike nicer than their own.

They realize they need to move back.

4. Tenderness

It's late June 2021, and you are the "Seitan Reloaded" recipe posted by author @simrob on Blogging Website M*dium Dot Com.

With the last week of their month-long vax celebration trip, the protagonist visits two other ThursDz friends in Raleigh RTP USA. They speedrun Celeste (a nervous cat who never shows affection to strangers, except nonbinary people). Said friends are themselves a little like cats, visibly changed by their own quarantine experiences. All three humans are eager to share whatever fruits said experiences have borne. One of them is you.

You, Seitan Reloaded, are not wings. You are sausage, strongly spiced and baked, fierce, in need of no sauce, to be put on scrambled eggs or pizza or simply to be eaten unadorned and cold from the fridge. Your author demonstrates you to the protagonist. He puts half a cup of paprika into you. He pours soy sauce into you, and keeps pouring – the protagonist agog – for impossibly long. He kneads you, and the protagonist remarks that they were afraid to knead their own because it already came out so tough, and he says the olive oil and the nooch inside you help disrupt gluten formation, and they think ah, this might be the final piece of the puzzle.

June being Pride month, the protagonist's employer attempts some trans-themed programming, but they dramatically neglect the "nothing about us without us" test and commit several terrible faux pas. Trans employees complain in Slack, and the brass tone police them into silence. One is nearly fired. The protagonist, already having been silenced previously, can only try to help behind the scenes, and it's good they're traveling so they have someone to weakly say "hold me" to while they work through their trauma response. Like always in protests, the surface problems get fixed, and the rest are left to become scars in the sands of time – almost as long as it takes to knead you.

With only a week in this visit, between that and all the video games the humans need to nerd over, there isn't time to synthesize your wisdom into the main quest. Instead they fry up slices of you, toss you in buffalo sauce (as if you need it), and produce, speaking extremely charitably, a Scientific Byproduct.

But cooking, baking, working matter with one's hands, is a wonderful grounding activity, and your texture, your tenderness is perfect.

5. Wings

It’s August 2021, and you are visiting your parents for the first time since quarantine in Santa Barbara Socal USA.

You have a lot to catch up on. Where there was border collie Louie, there is now border collie Junior. You will never know if Louie could tell you smell different now. You are teaching your mom to play Heaven’s Vault, your new favorite conlang-learning archaeology VN, trying to tell her that the helplessness the game makes her feel is in fact diagetic, and she (also a language nerd) is kinda actually getting it. Your psychological association is still tenuous, because your gender is new to them and your body is new to all three of you, but your parents will get there later, beyond the scope of this work.

They take you on their traditional morning walk. You always preferred the snow to the beach, but you still relish the feeling of sand in your toes and saltwater lapping at your calves. You had not felt the ocean on them since you made them yours. You walk Junior and you get seafood together from a stall along the boardwalk and suddenly the city of your birth feels like a bubble of immunity to that disaffected-millenial grief you carry everywhere else you go.

You tell them the preceding 4 chapters, and they agree to share with you the honor of putting the pieces together, and you even make half with noncanonical sauce because your dad can’t have spicy anymore, and that is the story of how seitan wings were made correctly in Santa Barbara for the first time ever.

You acknowledge Will Scott, William Lovas, Rob Simmons, Chris Martens, Eve Blum, David Blum, and Jenny Lin for their invaluable roles in the story.

Quarantine seitan wings

Based on Chapter 4's viewpoint character. You will need a steamer and either an immersion blender or a lot of wrist stamina. Makes 4-6 servings.

Ingredients (Wing; dry)

14 oz – Vital wheat gluten
1 oz – Nutritional yeast (or parmesan)
3 Tbsp – Spices: rosemary, thyme, onion or garlic powder, and white pepper
2 tsp – Salt

Ingredients (Wing; wet)

12 oz – Water (or chicken stock, for irony)
6 oz – Tomato paste
2 oz – Olive oil

Ingredients (Buffalo)

2 Tbsp – Butter (omit for vegans)
2 Tbsp – Hot sauce (Frank's is canonical, but you prefer Cholula)

Ingredients (Bluech)

1/3 cup – Blue cheese (a nice, creamy, pungent one from Penn Mac)
1/3 cup – Buttermilk (you can substitute but it really makes a difference)
Spoonful – Plain greek yogurt or sour cream, for consistency, optional

Method

In two bowls, separately mix the dry and wet wingredients. Combine and knead for 5-10 minutes, until coherent but also tears easily. Add more liquid or gluten as necessary, a little at a time.

Tear by hand (scraggly is good) into 1" chunks and steam for 30 minutes.

Meanwhile, chongle the blue cheese dip: Reserve half the cheese, and immersion blend everything else. Add reserved cheese and mash together with a fork until pleasantly chunky.

Melt butter and then mix in hot sauce. Do not microwave your hot sauce.

Fry steamed wings in neutral oil at 350°F for 3-5 minutes, until just crispy. Pat dry with a paper towel, toss in sauce, and serve over math puzzles.

Harry's Tactical Toast Cut Silhouette Recognition Guide

In today's tumultuous world, toast is a near certainty. You will encounter toast in schools, on airplanes, near hydroelectric dams, and even in diners. Knowing your toast -- being able to assess its operational capabilities with a single glace -- is therefore essential for all personnel.

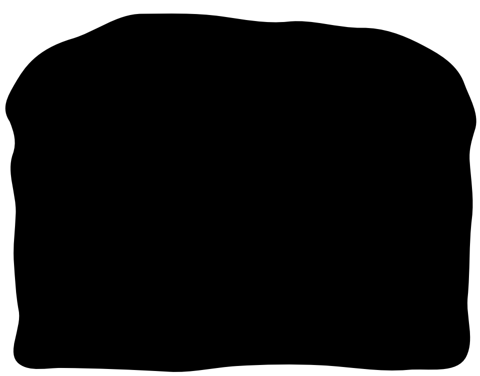
This **Harry's Guide** will prepare you to recognize common toast cuts based on their outlines, and to associate each outline with the operational characteristics of the cut as measured by our experts on the five-point PHTM™ scale:

Prep Time:	Handling:	Mess:	Threat:
Impact of the cut on overall toast preparation time. 1: negligible, 5: significant.	Impact of the cut on toast handling. 1: awkward, 5: handy.	Impact of the cut on crumb output. 1: crummy, 5: clean.	Impact of the cut on threat level. 1: negligible, 5: advanced, persistent.

Drill these silhouettes each morning, before heading into the breakfast zone. Effective breakfast operators **C.A.E.R.™: Categorize, Assess, React to, and Eat** toast**.**

ed. Jim McCann ix@tchow.com

Null Cut



Silhouette

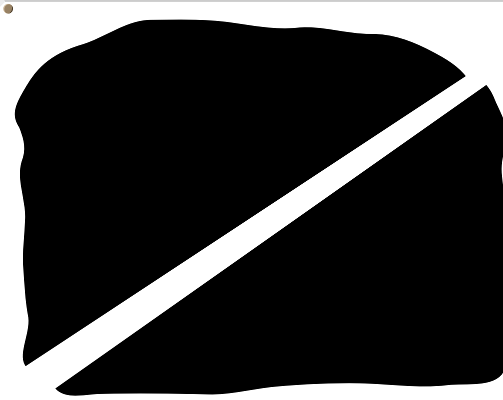
Prep:	Handling:	Mess:	Threat:
1	1	1	2

Assessment

Reduced portability, increased mess. Lower prep time. May indicate hasty improvisation or a novice breakfast operator.

Description

Diner Diagonal



Silhouette

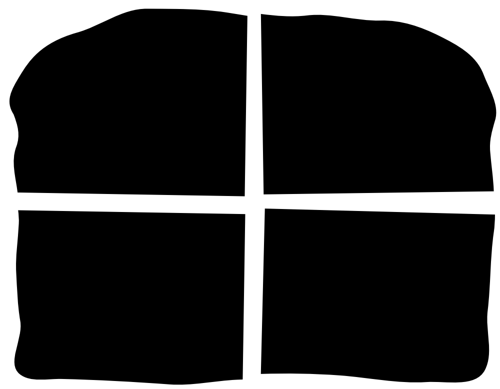
Prep:	Handling:	Mess:	Threat:
2	2	2	2

Assessment

A compromise cut, provides slight mess reduction with acute nibbling corners vs null cut. Antiquated, but underestimate it at your own peril. Also deployed for sandwiches and toast-sandwich hybrids (grilled cheese).

Description

Checker



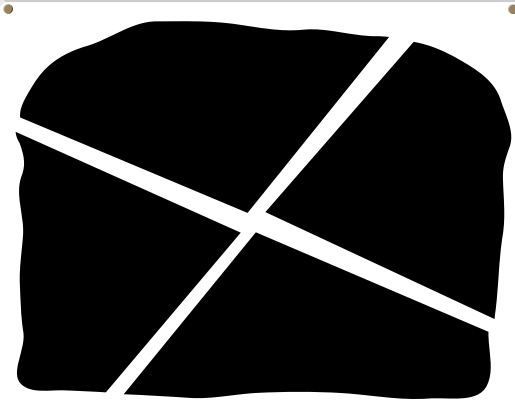
Silhouette

Assessment			
Prep:	Handling:	Mess:	Threat:
2	1	2	2

A four-piece cut used by breakfast irregulars and conscripted forces. Almost never deployed by professional breakfast operators. No handling benefit over the null cut, but increased prep time.

Description

Windmill



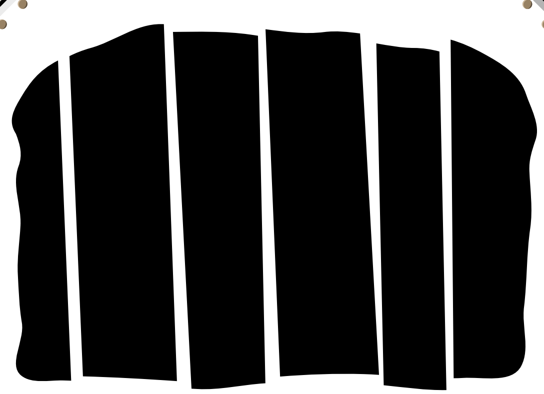
Silhouette

Assessment			
Prep:	Handling:	Mess:	Threat:
3	4	3	5

An experimental cut beginning to be adopted by major breakfast powers with established supply lines. Increased handling speed and decreased mess relative to the basic cuts. Requires careful knife handling during preparation. Do not attempt to field improvise. If deployed against a windmill exercise caution -- your adversary is prepared.

Description

Speed



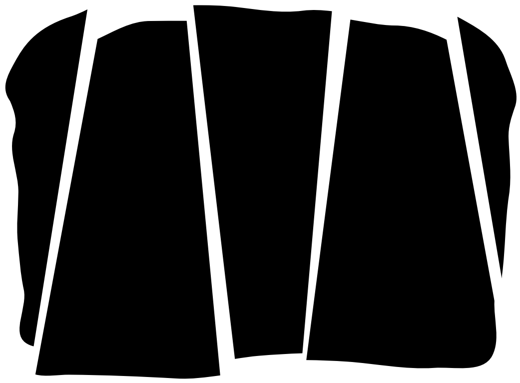
Silhouette

Assessment			
Prep:	Handling:	Mess:	Threat:
2	5	2	3

The speed cut prioritizes handling above all else. Used by breakfast operators who want to get in; get out; go, go, go! Surprisingly clean, despite the handling focus.

Description

Princess



Silhouette

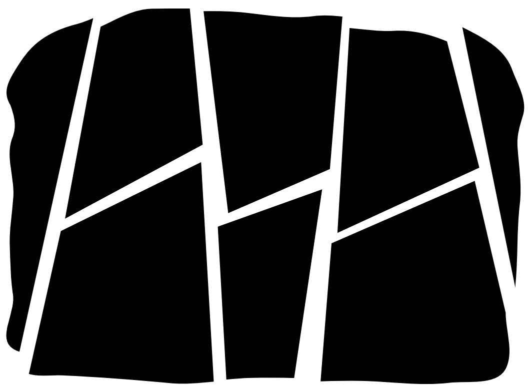
Assessment

Prep:	Handling:	Mess:	Threat:
5	3	3	3

The princess cut is used only by the most discerning breakfast operators. The dainty trapezoids evoke the faceted cuts of heritable mineral wealth. If you need to ask about the prep time you probably can't afford it.

Description

Teatime



Silhouette

Assessment

Prep:	Handling:	Mess:	Threat:
5	4	5	4

The teatime cut adds a zesty flare to the Princess cut by diagonally bisecting the middle trapezoids. Enjoyed by truly grizzled breakfast operators who have seen enough of the chaos of life to know that a cup of calming tea is to be savored wholeheartedly with a spot of toast.

Description

Herringbone

Silhouette

Assessment

Prep:	Handling:	Mess:	Threat:
5	5	4	5

A highly-adaptable cut that can mimic the princess cut, windmill cut, and speed cut. More complex to prepare and plan for, this cut is used exclusively by experienced breakfast operators who need the ultimate in adaptability.

Description

Activity Page

A breakfast operator is always prepared! Design your own tactical toast cuts.

Salzgurken: A formal grammar for unambiguous grocery shopping

Eleftheria Chatziargyriou
EUR Store Coalitionist
Torshi Aisle (the one left to the big ice cream fridge)

***Abstract*—In this paper we are examining how to resolve ambiguities in grocery stories lists. We introduce a formal grammar for specifying an easy-to-understand grammar, easily parsable and tested in all currently available platforms (Gx1000000, GZ, GX and even the now-deprecated GBM). We also propose a range of restrictions with the aim of simplifying the information processed by the modern consumer.**

I. Introduction

Ambiguity in grocery shopper instructions leads to countless incidents affecting consumers each year versed in the ways of programming. While sometimes said incidents undoubtedly offer some unparalleled entertainment value, this shouldn't determine the torment uncountable of our fellow citizens endure.

We aim at a rigorous definition of grocery shopping lists introducing a formal grammar, facilitating a pleasant shopping experience.

II. Background

We begin by examining some of the most common incident categories. Some documented incidents are referenced in Table I and Figure 2.

TABLE I: Example of incident in tabular format

Failure Mode	Upvotes	Reference
Syntax ambiguity	2.3k	[5]
Non-terminating process	727	[4]
Syntax ambiguity	136	[6]
Syntax ambiguity	801	[3]
Syntax error	60	[2]

The observant reader will note that there are numerous more examples to be found, particularly in online platforms but the `.bib` isn't going to write itself. The curious reader with slight masochistic tendencies is invited to observe the phenomenon by themselves by duckduckgoing the following keywords we identified: "husband" "programmer" "grocery shopping".

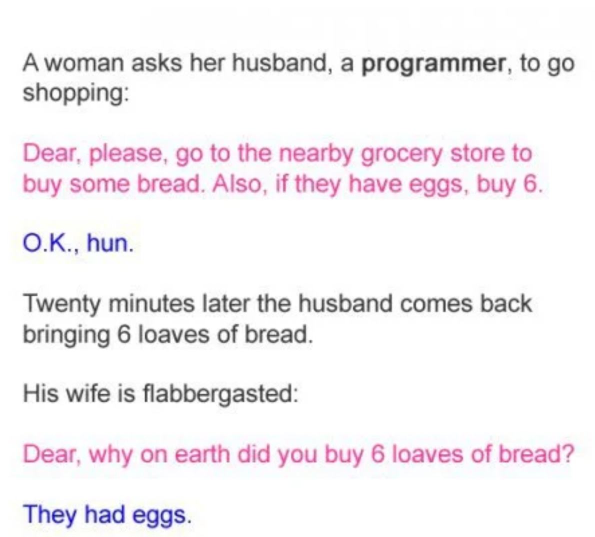


Fig. 1: Example of incident but in pictorial format [1]

The main point to be highlighted is that victims seem to be predominantly male. The reason seems to be that the male brain is more logical and structured, making prone to syntactic ambiguities common in the female intellect (according to scientific study with at least 25.8k upvotes [7]).

We also observe that the failure modes are effectively eliminated with the introduction of a formal grammar. Namely the problem of syntactic ambiguity is resolved by using a well-designed grammar. Likewise, we tackle potentially non-terminating processes by prohibiting recursive and looping constructs.

III. Requirements

Before proceeding to actually defining the grammar, we are going to elicit some requirements to maximise the stakeholder's satisfaction and efficiency.

The most important one is for the grammar to be clear, concise and easily understandable by both menfolk and womenfolk.

We also want to ensure that a polynomial-bound shopper can shop in a reasonable time. We are only considering classical shoppers as pesky quantum shoppers usually abide to much different lifestyles for them to be accurately examined here.

We can imagine semantically supporting clauses such as:

- *"Buy either eggs or cherries"*
- *"Buy one butter for every egg you've bought"*
- *"If you have bought cherries, buy an avocado but only if there are vegan chicken nuggets"*
- *"Don't spend more than 10 money if BTC is trending downwards"*

How will the optimal shopper proceed to maximize their shopping experience? Should they prioritize getting the most value for his money? Should he pick some items over others? He simply has no unambiguous way of deciding. Before they know it, they are going to start filling in abnormally large knapsacks with useless items bringing us one step closer to our eventual demise.

Therefore, we need to somehow restrict the semantic scope of our representation language in the following way:

- Any conditional statement on the procurement status of any single item shall not depend on the outcome of any other statement
- Arbitrary properties of the item can be referred to, assuming it is understandable by all parties. Said properties can only refer to the item that conditionally depends on them
- Make it easy for regular housewives to understand

The grammar has to obviously reflect the aforementioned well-defined, crystal-clear requirements.

IV. Innovation

Camera pans to a thirty year old man doing his weekly grocery shopping in a Lidl somewhere in Bavaria. He has a frustrated look on his face holding a sky blue package in one hand and a nearly-identical deep sky blue package in the other. The man is scratching his head and gives an intense look to the products A male narrator with stern yet oddly warm voice starts speaking:

[MN]: Has this ever happened to you?

Scene transition: The man is back to his rustic home and gets the groceries out of his eco-friendly, reusable bag. Tens of nearly-identical packages are laying on the table. The wife can be seen just entering the kitchen with a look of terror once she spots that the sky blue package is laying on their table

[MN:] Introducing: Salzgurken. The **modern** way of shopping

The husband is again doing his shopping on the Bavarian Lidl looking at two other identical packages: one lemon chiffon and one Chatreuse yellow

V. TADA

The grammar file can be found here[1] and it is extremely self-explanatory.

1) Simple instructions if you can't self-explain: Every entry in the grocery list has to start with $\star$. This needs to be followed by a numerical value and an optional unit.

The numerical value can also specify a range:

- $N..$: *at least* N
- $..N$: *at most* N
- $N_1..N_2$: *at least* N_1 *and at most* N_2

The greedy shopper will always choose the greatest amount they can get that meet the given conditions.

Additionally, a condition can be specified. The shopper will proceed with the procurement only if said condition is met as is illustrated in the following double-explanatory example:

```
* 6.. "EGGS" IF (WEIGHT < 199 kg)
* 100..500 L "MILK" IF ((ATTRIBUTE $quality
  IS "extra fresh") AND (PRICE = 100))
```

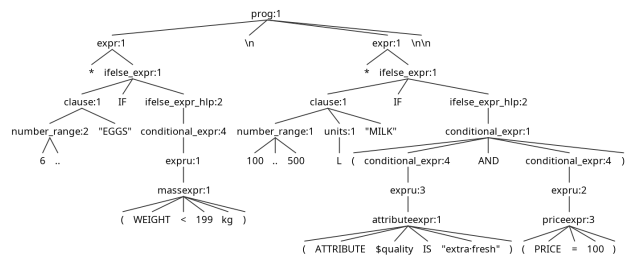


Fig. 2: Nice tree example

VI. Future work

We will work on our time-scheduling skills so next SIGBOVIK, we'll remember the deadline in time.

VII. Acknowledgments

This work wouldn't have been possible without Arch Linux. I would like to also thank my mother for teaching me the importance of generating exclusively human-made nonsense. Machines will not replace us.

References

[1] *A programmer goes shopping*. Oct. 21, 2010. URL: https://www.reddit.com/r/pics/comments/duf6p/a_programmer_goes_shopping/.

[2] *A Python programmer and his wife are discussing grocery shopping*. Aug. 4, 2018. URL: https://www.reddit.com/r/ProgrammerHumor/comments/94ivqe/a_python_programmer_and_his_wife_are_discussing/.

[3] u/$_BlNG$. *A programmer goes shopping*. Jan. 30, 2017. URL: https://www.reddit.com/r/Jokes/comments/5qyxar/a_programmer_goes_shopping/.

[4] u/Kyraryc. *A wife send her programmer husband grocery shopping*. Apr. 10, 2018. URL: Wife%20sent%20programmer%20back.

[5] u/pitzu. *A wife send her programmer husband grocery shopping*. Apr. 10, 2018. URL: https://www.reddit.com/r/Jokes/comments/8baxnd/wife_sends_her_programmer_husband_grocery_shopping/.

[6] u/whitehat2k9. *A man goes to the grocery store...* May 28, 2012. URL: https://www.reddit.com/r/ProgrammerHumor/comments/u85tr/a_programmer_goes_to_the_grocery_store_xpost_from/.

[1] https://gist.github.com/Sedictious/193bcb63d7d5960cb509713d78393e63

[7] *Women tend to prefer working with people, while men tend to prefer working with things, according to a new study based on an analysis of responses from people in 42 countries.* URL: https://www.reddit.com/r/science/comments/ziwfqe/women_tend_to_prefer_working_with_people_while/.

VIII. ANNEX

```
grammar salzgurken;
prog:   (expr NEWLINE)* ;

expr:
BULLET ifelse_expr |
BULLET clause
;

ifelse_expr:
clause IF ifelse_expr_hlp (ELSE ifelse_expr_hlp)?;
ifelse_expr_hlp:
clause IF conditional_expr (ELSE ifelse_expr_hlp)? |
conditional_expr;

conditional_expr:
LPAR conditional_expr AND conditional_expr RPAR |
LPAR conditional_expr OR conditional_expr RPAR |
LPAR NOT conditional_expr RPAR |
expru;

clause:
number_range (units)? STRING_LITERAL;

expru: massexpr | priceexpr | attributeexpr;

attributeexpr:
LPAR ATTRIBUTE ATTRIBUTE_VAR IS STRING_LITERAL RPAR |
LPAR ATTRIBUTE ATTRIBUTE_VAR IS ONEOF string_literal_list RPAR;

massexpr:
LPAR WEIGHT LT NUMBER MASSUNITS RPAR |
LPAR WEIGHT GT NUMBER MASSUNITS RPAR |
LPAR WEIGHT EQ NUMBER MASSUNITS RPAR;

priceexpr:
LPAR PRICE LT NUMBER RPAR |
LPAR PRICE GT NUMBER RPAR |
LPAR PRICE EQ NUMBER RPAR;

oneof: 'ONE OF' list;
list: number_list | string_literal_list;
number_list: '['NUMBER (NUMBER)* ']';
string_literal_list: '['STRING_LITERAL (COMMA STRING_LITERAL)* ']';
number_range:
NUMBER'..'NUMBER |
NUMBER'..' |
'..'NUMBER |
NUMBER;
units : MASSUNITS | VOLUMEUNITS;

MASSUNITS :
'dg' | 'cg' |'mg' |'µg' |'ng' |'pg' |'fg' |'ag' |'zg' |'yg' |'rg' |'qg' |'dag'|'hg' |'kg' |'Mg' |'Gg'
|'Rg' |'Qg';

VOLUMEUNITS:
'qL' | 'rL' | 'yL' | 'zL' | 'aL' | 'fL' | 'pL' | 'nL' |'L'| 'mL' | 'cL' | 'dL' | 'L' | 'daL' | 'hL' |
'ZL' | 'YL' | 'RL' |'QL';

IS: 'IS';
IF: 'IF';
ELSE: 'ELSE';
ONEOF: 'ONE OF';
WEIGHT: 'WEIGHT';
ATTRIBUTE: 'ATTRIBUTE';
FLAVOUR: 'FLAVOUR' | 'FLAVOR';
PRICE: 'PRICE';
LT: '<';
GT: '>';
EQ: '=';
AND: 'AND';
OR: 'OR';
NOT: 'NOT';
COMMA: ',';
LPAR: '(';
RPAR: ')';
ATTRIBUTE_VAR: '$'[A-Za-z]+;
STRING_LITERAL: '"'~["\r\n]*'"';
STRING: [a-zA-Z]+;
NEWLINE: [\r\n]+;
BULLET: '*';
NUMBER: INTEGER | FLOAT;
fragment INTEGER: [0-9]+ ;
fragment FLOAT: [0-9]+'.'[0-9]+;
fragment SEPARATOR: '-' | '\\';
WS :  ('\t' | ' ' | '\r' | '\n' | '\u000C')+ -> skip ;
```

M

Hear, Meet And Now

VOACaloid*

A “better” “hardware-based” “portable” “solution” for the “real-time” “generation” of “singing”

Aaron Thapa
notsmart98@gmail.com
@yoyo@m.ggrks.moe†

Thomas Jefferson High School for Science and Technology

2023 April 0

Abstract

Singing Voice Synthesis (SVS) technologies have advanced to the point that they can now often be confused for the human singers they are based on. However, software SVS (the most common kind of SVS), has proven difficult to use in live performance. In this article, I present a live, human-controlled, hardware-based SVS device whose construction involves obscure parts obtained in suspicious circumstances, a Raspberry Pi Zero, and apathy on scales never-before-seen, even in this conference.

1. Introduction

Composers of music are often hesitant to include vocals on their work. For singers, singing for extended periods can lead to vocal strain. For non-singers, working with singers for extended periods can cause mental strain. To resolve this issue, various synthesized singing technologies have been developed.

1.1. History of SVS

The first computer singer was an IBM 7904 owned by Bell Labs in 1961 [1]. Becasuse of the hardware, the voice that it produced was limited and not realistic. Later, the Digital Equipment Corporation (DEC) brought several implementations of SVS to a larger market, known collectively as DECtalk [2]. Originally, DECtalk was a hardware product, using ICs to generate the voice, but software implementations and libraries were later produced, allowing programmers to integrate voice synthesis and SVS into their codebases. Notably, DECtalk was featured in NASA’s video game, *Moonbase Alpha*, in which players often used the SVS engine in a humorous manner[1] [3]. Most recently, companies such as Google have been developing AI-based text-to-speech engines [4].

*VOAC: Voice synthesizer on a chip; own coinage

†Quite possibly the first Mastodon link in SIGBOVIK History, but I wouldn’t be surprised to see more this year.

[1] John Madden John Madden John Madden John Madden John Madden John Madden John Madden John Madden John Madden John Madden

1.2. Vocaloid

In 2000, Hideki Kenmochi created a singing voice synthesizer project called "Vocaloid", from "vocal" and "android" [5]. Vocaloid generated output by concatenating individual recordings of phonemes from a singer, allowing for a much more realistic sound than previous attempts. With support from Yamaha, the first commercial version of Vocaloid was released in 2003 [1].

After the 2007 release of Vocaloid 2, Crypton Future Media created a voice called Hatsune Miku, now better known than Vocaloid itself [6]. Yamaha themselves released two voices, VY1 ("Mizki") and VY2 ("Yūma").

1.3. eVocaloid

In 2012, Yamaha implemented a version of Vocaloid in one of their synthesizer ICs, the YMW820 [7], also known as the NSX-1. The "e" prefix stands for "embedded", and was also added to the name of the voice, VY1, to make eVY1 the name for the voice bank. In 2014, Gakken, a Japanese educational company, released the "Pocket Miku" [8]. Its stylophone-style pitch input is suboptimal for live performance [9]. Additionally, the only way to control the current syllable is by sending it MIDI messages from another device [9].

1.4. VOACaloid

This project aimed to create a "portable" device which uses the capabilities of the YMW820 to allow the user to control it in an "intuitive" manner, while still being able to use the full extent of its features in a live, concert-performance scenario. This could also be described as a "shanzhai[2]" version of the Pocket Miku, making use of the eVY1.

2. Motivation

~~N/A [10]~~ Indeed, software SVS solutions have worked in music production, and any live performance of SVS would be much less practical than just finding a singer. Even Vocaloid concerts which feature live bands simply use a pre-rendered track for the Vocaloid singing [11]. Having a portable and playable Vocaloid is very cool though, thus giving this project some purpose.

2.1. Materials

Name	Price	Qty.
Aides Tech eVY1 Board	¥9000	1
DMK-25 Midi Controller	$65.99	1
DIN Coupler (2 pcs)	$8.49	1
USB Midi Adapter	$6.99	1
Raspberry Pi Zero	$5.00	1
Total:	$155.55	

Most of the parts were bought on Amazon, except for the eVY1 board, which was bought on Yahoo Shopping Japan, and then shipped to the US via a mail forwarding service[3].

2.2. Software

The Raspberry Pi Zero runs a version of Raspbian with the desktop environment disabled. The program which routes MIDI messages between the USB-MIDI adapter, the MIDI controller, and the eVY1 board is written in Python. When plugged in to power, the Raspberry Pi boots up and runs `detector.py` in a headless session.

3. Development Process

3.1. Device

Once I received the eVY1, I spent a few days learning how to format the MIDI messages that I would need to send it from a Jupyter Notebook (see `evy1.ipynb` for implementation). Standard Japanese Romanization is ambiguous, so the eVY1 shield uses its own phonetic transcription system, somewhat related to that used in software versions of VOCALOID. I used the NSX-1 datasheet [7] to write `japanese_to_phoneme()`

[2] Chinese word for Chinese copy
[3] My uncle

which would take a string of hiragana and convert it to the phoneme text. Using the new text, `phoneme_to_midi_message()` creats a SysEx message to send to the eVY1.

For pitch input, the `mido` Python module had most of the tools that were necessary built-in. I wrote `router.py` to take incoming MIDI messages from the MIDI controller and then modify them to be suitable for the eVY1. At this point, I still had not decided what I wanted to do with the project, so I wrote `player.py` to convert and play MIDI files.

Then came the Summer of 2022. I stopped working on the project for about 6 months for several reasons but mostly because I was out of the country and was focused on other work.

Once I started having time in my school schedule, I resumed the project, and came up with the live performance idea. Optimizing for this proved to be difficult. In `detector.py` I implemented what would later become the most important part of the project, the phoneme input system. Crucially, when reading the button combinations as pressed by the user, it only updates when a new button is pressed, and ignores the event generated by the button being released. With this feature working I decided that I did not want to work anymore.

3.2. Various Side-Projects

There were many points during and after this project in which I grew bored with developing the main features. I began working on various semi-related endeavours which I will write about here for the purposes of space-filling.

One of the first things I had done which was unrelated to the project was that I used the eVY1 as a General MIDI-compatible module for composing music. Using it in this manner did mean that I was unable to use the vocal aspect in any of the music, alongside MIDI Channel 1 (Channel 0) because it is exclusively used for the voice. In FL Stuido, I used it as the target for the "MIDI Out" plugin. The music I produced with the module was used for a LOVE Jam 2021 Submission, "Scraper Escaper"[4].

Continuing on the MIDI file idea from earlier, one of the examples that Aides Tech had released for using the eVY1 shield was an Arduino sketch which took the binary data of a MIDI file and sent it to the eVY1 for playback [12]. Based on that code, I wrote a Python script which could take a MIDI file and output an Arduino sketch which could then be uploaded[5].

After converting a few songs, I noticed that the code did not give a correct tempo for certain input files. Instead of finding the root cause, I opted to manually change one of the constants in the file until it sounded about right. Not doing this resulted in several humorous files, such as `s2chemical3.ino` from Sonic 2, which played much too fast.

`Router.py` was an early attempt at what would eventually become the main functionality. Its purpose was to route midi msesages from the selected device to the eVY1, but I also left in some aspects such as choosing the MIDI channel which note events would be sent to, so that other instrument sounds could be used bseides the voice.

4. Usage

Using the device is quite simple. Once the Pi has power, one can simply press the pads on the midi controller in accordance with the table in Appendix A and press the melodic keys to achieve the desired pitch. Understanding how to get the software into a usable state is left as an exercise for the reader, as is learning how to play any real songs.

5. Demonstration

I understand that giving some kind of output sample or a demonstration at all is pretty important. However, if you are reading this paragraph, forces beyond my control have conspired

[4]Game can be played at: `https://bluesheep7.itch.io/scraper-escaper`, music can be found at: `https://youtu.be/X0FZDJCPhIU`

[5]Yes, I understand the irony of using a Python script to write C code. I also think it's very funny.

such that I am unable to access this device until beyond the deadline. However, any skilled performance would sound no different from one being controlled by a MIDI file. Thus, I will point to the demonstrations from Aides Tech, available on their page for the eVY1 as sufficient examples of eVY1 output.

6. Discussion

This device is completely impractical, and it does not satisfy a live performer's needs. The output is far too noisy, and it is hard to power cleanly.

6.1. Comparison with Human Singer

Comparison	VOACaloid	Human Singer
Wages	✓	
Energy Consumption	✓	
Polyphony	✓	
Vocal Range	✓	
Percussive Ability	✓	
Total:	5	0

As this impartial and objective comparison shows, the VOACaloid is unequivocally the best singing device ever created. After all, just about anything can beat working with a human.

7. Future Developments

7.1. Of this project

This section is inherently difficult to write because anything I write in here could theoretically be implemented before the deadline and thus would need to be removed from this section. Thus, I will not be including any possible future developments, and instead be implementing them and describing them elsewhere in this paper. Of course, I could have waited until I was truly "finished" with the project, but that is no fun. I cannot predict the future, so I do not know what I will implement. Any ideas for future developments can be sent to my email, as listed at the top of this paper.

Since I have written the last paragraph, the SIGBOVIK deadline has been announced, and I had long decided to stop working. I ended up not using the MIDI in and out connections which I had bought. One idea I had was to be able to connect to the shell using a terminal program over the MIDI connectors. Obviously it would have been terrible to use, but I could probably implement it in the future if I cared to continue working on this (which I don't).

7.2. Of SVS in general

Just who do you think I am? We've proven that I'm not smart enough to make anything useful, let alone predict the future of a fast-moving scientific field. We've all seen what ChatGPT has done to writing and what stable diffusion has done to art, who's to say that something similar can't come of CeVio or Neutrino. CeVio is already generally indistinguishable from a human voice with heavy processing.

8. Acknowledgments

I would first like to thank those at Yamaha responsible for the development of Vocaloid. I would then like to thank my lab director and supervisor, Kuprenas, who managed to keep me mostly on track during the creation of this project, and gave me the funds to buy the MIDI controller, and Paul Drongowski, for maintaining his blog and archiving the datasheets and web tools for the NSX-1. Thanks to the developers of TeX and LaTeX for making writing papers enjoyable, and thanks to the contributors to XeLaTeX for adding Unicode support to make writing this specific paper possible. ~~Thanks to Donner for not writing any useful documentation for their MIDI controller.~~ I would like to thank Thomas Chick for the previous joke [13], as well as Tom Wildenhain for introducing me to SIGBOVIK, and Tom Murphy VII for reminding me it exists every year[6].

[6]Wow, I didn't notice that they were all Toms; that's pretty cool.

References

[1] Gakken, *Pocket Miku: The singing keyboard*, 2014. [Online]. Available: https://cdn-shop.adafruit.com/pdfs/pocket_miku.pdf.

[2] A. Pollack, "Technology; audiotex: Data by telephone," *The New York Times*, Jan. 5, 1984.

[3] VocaDB, *Moonbase Alpha text-to-speech*, 2023. [Online]. Available: https://vocadb.net/Ar/85226.

[4] A. van den Oord, S. Dieleman, H. Zen, *et al.*, "Wavenet: A generative model for raw audio," *CoRR*, vol. abs/1609.03499, 2016. arXiv: 1609.03499. [Online]. Available: http://arxiv.org/abs/1609.03499.

[5] H. Kenmochi, *Vocaloid2*、初音ミク、ユーザ、*UGM* サイト、権利者 *[VOCALOID2, Hatsune Miku, users, UGM sites and rightful claimants - the report]*, 2008. [Online]. Available: http://www.dcaj.or.jp/project/report/pdf/2007/dc08_03.pdf.

[6] B. Roseboro, "The Vocaloid phenomenon: A glimpse into the future of songwriting, community-created content, art, and humanity," DePauw University, 2019. [Online]. Available: https://scholarship.depauw.edu/cgi/viewcontent.cgi?article=1125&context=studentresearch.

[7] Yamaha, *YMW820(NSX-1) sound generator datasheet*, 2012. [Online]. Available: http://sandsoftwaresound.net/wp-content/uploads/2017/07/YMW820_data_sheet_en.pdf.

[8] Gakken, ポケット・ミク カスタマイズガイド *[Pocket Miku customize guide]*, 2008. [Online]. Available: https://otonanokagaku.net/nsx39/data/nsx39midiguide.pdf.

[9] D. Battino, "Hacking Pocket Miku, the singing stylophone," *Electronic Musician*, vol. 31, Jan. 2018.

[10] T. Wildenhain, "On the turing completeness of MS PowerPoint," in *SIGBOVIK*, 2021.

[11] H. Miku, *【Hatsune Miku】World is Mine / ryo(supercell)【初音ミク】*, 2013. [Online]. Available: https://www.youtube.com/watch?v=jhl5afLEKdo.

[12] Switchscience, *eVY1 shield example*, 2014. [Online]. Available: https://github.com/SWITCHSCIENCE/eVY1_Shield.

[13] T. Chick, "'The SIGBOVIK paper to end all SIGBOVIK papers' will not be appearing at this conference," in *SIGBOVIK*, 2021.

A. Phoneme Table

Kana	Romaji	Phoneme Text	Buttons
あ	a	a	(0)
い	i	i	(1)
う	u	M	(2)
え	e	e	(3)
お	o	o	(4)
か	ka	k a	(5)
き	ki	k' i	(6)
く	ku	k M	(7)
け	ke	k e	(0, 1)
こ	ko	k o	(0, 2)
さ	sa	s a	(0, 3)
し	shi	S i	(0, 4)
す	su	s M	(0, 5)
せ	se	s e	(0, 6)
そ	so	s o	(0, 7)
た	ta	t a	(1, 2)
ち	chi	tS i	(1, 3)
つ	tsu	ts M	(1, 4)
て	te	t e	(1, 5)
と	to	t o	(1, 6)
な	na	n a	(1, 7)
に	ni	J i	(2, 3)
ぬ	nu	n M	(2, 4)
ね	ne	n e	(2, 5)
の	no	n o	(2, 6)
は	ha	h a	(2, 7)
ひ	hi	C i	(3, 4)
ふ	hu/fu	p\M	(3, 5)
へ	he	h e	(3, 6)
ほ	ho	h o	(3, 7)
ま	ma	m a	(4, 5)
み	mi	m' i	(4, 6)
む	mu	m M	(4, 7)
め	me	m e	(5, 6)
も	mo	m o	(5, 7)
ら	ra	4 a	(6, 7)
り	ri	4' i	(0, 1, 2)
る	ru	4 M	(0, 1, 3)
れ	re	4 e	(0, 1, 4)
ろ	ro	4 o	(0, 1, 5)
が	ga	g a	(0, 1, 6)
ぎ	gi	g' i	(0, 1, 7)
ぐ	gu	g M	(0, 2, 3)
げ	ge	g e	(0, 2, 4)

ご	go	g o	(0, 2, 5)
ざ	za	dz a	(0, 2, 6)
じ	zi/ji	dZ i	(0, 2, 7)
ず	zu	dz M	(0, 3, 4)
ぜ	ze	dz e	(0, 3, 5)
ぞ	zo	dz o	(0, 3, 6)
だ	da	d a	(0, 3, 7)
ぢ	di/zi	dZ i	(0, 4, 5)
づ	du/dzu	dz M	(0, 4, 6)
で	de	d e	(0, 4, 7)
ど	do	d o	(0, 5, 6)
ば	ba	b a	(0, 5, 7)
び	bi	b' i	(0, 6, 7)
ぶ	bu	b M	(1, 2, 3)
べ	be	b e	(1, 2, 4)
ぼ	bo	b o	(1, 2, 5)
ぱ	pa	p a	(1, 2, 6)
ぴ	pi	p i	(1, 2, 7)
ぷ	pu	p M	(1, 3, 4)
ぺ	pe	p e	(1, 3, 5)
ぽ	po	p o	(1, 3, 6)
や	ya	j a	(1, 3, 7)
ゆ	yu	j M	(1, 4, 5)
よ	yo	j o	(1, 4, 6)
わ	wa	w a	(1, 4, 7)
ゐ	wi	w i	(1, 5, 6)
ゑ	we	w e	(1, 5, 7)
を	wo/o	o	(1, 6, 7)
ふぁ	fa	p\a	(2, 3, 4)
つぁ	tsa	ts a	(2, 3, 5)
うぃ	wi	w i	(2, 3, 6)
すぃ	si	s i	(2, 3, 7)
ずぃ	zi	dz i	(2, 4, 5)
つぃ	tsi	ts i	(2, 4, 6)
てぃ	ti	t' i	(2, 4, 7)
でぃ	di	d' i	(2, 5, 6)
ふぃ	fi	p\' i	(2, 5, 7)
とぅ	tu	t M	(2, 6, 7)
どぅ	du	d M	(3, 4, 5)
いぇ	ye	j e	(3, 4, 6)
うぇ	we	w e	(3, 4, 7)
きぇ	kye	k' e	(3, 5, 6)
しぇ	she	S e	(3, 5, 7)
ちぇ	che	tS e	(3, 6, 7)
つぇ	tse	ts e	(4, 5, 6)
てぇ	tee	t' e	(4, 5, 7)
にぇ	nye	J e	(4, 6, 7)
ひぇ	hye	C e	(5, 6, 7)

みぇ	mye	m' e	(0, 1, 2, 3)
りぇ	rye	4' e	(0, 1, 2, 4)
ぎぇ	gye	g' e	(0, 1, 2, 5)
じぇ	jye	dZ e	(0, 1, 2, 6)
でぇ	dee	d' e	(0, 1, 2, 7)
びぇ	bye	b' e	(0, 1, 3, 4)
ぴぇ	pye	p' e	(0, 1, 3, 5)
ふぇ	fe	p\e	(0, 1, 3, 6)
うぉ	wo	w o	(0, 1, 3, 7)
つぉ	tso	ts o	(0, 1, 4, 5)
ふぉ	fo	p\o	(0, 1, 4, 6)
きゃ	kya	k' a	(0, 1, 4, 7)
しゃ	sha	S a	(0, 1, 5, 6)
ちゃ	cha	tS a	(0, 1, 5, 7)
てゃ	tya	t' a	(0, 1, 6, 7)
にゃ	nya	J a	(0, 2, 3, 4)
ひゃ	hya	C a	(0, 2, 3, 5)
みゃ	mya	m' a	(0, 2, 3, 6)
りゃ	rya	4' a	(0, 2, 3, 7)
ぎゃ	gya	N' a	(0, 2, 4, 5)
じゃ	ja/jya	dZ a	(0, 2, 4, 6)
でゃ	dya	d' a	(0, 2, 4, 7)
びゃ	bya	b' a	(0, 2, 5, 6)
ぴゃ	pya	p' a	(0, 2, 5, 7)
ふゃ	fya	p\' a	(0, 2, 6, 7)
きゅ	kya	k' M	(0, 3, 4, 5)
しゅ	shu	S M	(0, 3, 4, 6)
ちゅ	chu	tS M	(0, 3, 4, 7)
てゅ	tyu	t' M	(0, 3, 5, 6)
にゅ	nyu	J M	(0, 3, 5, 7)
ひゅ	hyu	C M	(0, 3, 6, 7)
みゅ	myu	m' M	(0, 4, 5, 6)
りゅ	ryu	4' M	(0, 4, 5, 7)
ぎゅ	gyu	g' M	(0, 4, 6, 7)
じゅ	jyu	dZ M	(0, 5, 6, 7)
でゅ	dyu	d' M	(1, 2, 3, 4)
びゅ	byu	b' M	(1, 2, 3, 5)
ぴゅ	pyu	p' M	(1, 2, 3, 6)
ふゅ	fyu	p\' M	(1, 2, 3, 7)
きょ	kyo	k' o	(1, 2, 4, 5)
しょ	sho	S o	(1, 2, 4, 6)
ちょ	cho	tS o	(1, 2, 4, 7)
てょ	tyo	t' o	(1, 2, 5, 6)
にょ	nyo	J o	(1, 2, 5, 7)
ひょ	hyo	C o	(1, 2, 6, 7)
みょ	myo	m' o	(1, 3, 4, 5)
りょ	ryo	4' o	(1, 3, 4, 6)
ぎょ	gyo	N' o	(1, 3, 4, 7)

じょ	jo	dZ o	(1, 3, 5, 6)
でょ	dyo	d' o	(1, 3, 5, 7)
びょ	byo	b' o	(1, 3, 6, 7)
ぴょ	pyo	p' o	(1, 4, 5, 6)

This table is based on data from the source code for this project, available at `https://github.com/yoyoyonono/evy1py`. A good amount of the romaji in this table is complete guesswork.

B. Images

Here you can see the various parts which comprise the VOACaloid. In the top-left corner there is the Raspberry Pi Zero, to the right of it is the eVY1 Shield, and under it is the MIDI controller.

Avantgarde Visual Auditive JSON Hashing

Tobias Coq, Louis Erhard, Julio Esperantò, Jean Parler, Peter-Ludwig Sechsdiener
Fery Äccelent University Erlangen-Nürnberg (FAU)

***Abstract*—Everyone loves JSON! It does not matter if it is JSON from *Friday the 13th* or JSON Statham, the brilliant researcher that has published a lot of papers about how to encode actions in the reinforcement learning domain in a hardcore fashion using brute-force action. But do you know what is even better than JSON? Hashing the JSON! JSON loves hashish, eh, hashing, stupid auto-correct, but is not fond of the state-of-the-art way of doing this (quite boring, ain't it). Luckily, there is always innovation in every field, especially in the field of consuming hashing (come on, autocorrect, not again). The paper presents a novel method of hashing JSON files by converting them into audio signals, applying Fantastically Fancy Fun Fast Fourier Transform (FFFFFT) to generate the frequency spectrum, and then transforming the spectrum into images. Because you really have to feel the ha$(shi)^2$ng process of JSON. And what is better than having sensory explosions not only in the audio but also in the visual domain. The proposed approach provides a new and incredibly avantgarde way of converting textual data into an aesthetically pleasing format that can be easily [1] analyzed and compared using digital signal processing techniques. The authors demonstrate the effectiveness of their method by testing it on an immense set of JSON files and comparing the resulting audio-based hashes to traditional approaches where it outperforms some strong state-of-the-art ones as $h(x) = 1$ in terms of minimizing the collisions. The results do not show that the proposed approach is either accurate or efficient. They also do not show that the proposed dancing moves offer a promising alternative to Salsa or Cha Cha Cha :(**

***Index Terms*—hashish hashing, JSON, efficient, collision avoidance, fundamental breakthrough**

I. Introduction

The growing importance of data security and privacy has led to an increase in the use of consuming hashing to get rid of conflicts. In recent years, there has been a growing interest in the use of audio hashing techniques for data hashing, as audio signals are highly unique and difficult to replicate (try to replicate a speech by Arnold Schwarzenegger and you really struggle to get that accent right). This paper presents a novel approach to hash JSON files by first converting them into audio signals, and then applying Fantastically Fancy Fun Fast Fourier Transform (FFFFFT) to extract the frequency components of the audio signals. Finally, the frequency components are converted into images of the spectrum, which are used as the hash values for the JSON files. The proposed method offers several disadvantages, including low security and high collision rates, as well as the ability to store and transmit hash values in a not very compact form. This paper provides a detailed description of the proposed method, along with experimental results demonstrating nothing. But we believe that hash conflicts resolve themselves in the long run anyway, according to Hegel's [1] dialect, as demonstrated by the master-slave thought experiment.

Thanks to Mickey Mouse for always supporting us.

[1]Trust me, bro

II. Fundamentals

Hashing, the bouncers of the digital land,
IDs checked to ensure legitimacy stands.
Some methods outdated, others strong and true,
Reliability essential, security pursued.

Condensing data, Marie Kondo style,
Sparks joy remain, rest discard with a smile.
A safety net for accuracy's sake,
Old methods insufficient, newer ones to take.

Collision rates high, like a club in chaos,
Security vulnerable, nefarious intent employs.
A ninja-like function, swift and sly,
Unique codes created, identities to defy.

Like a magician's trick, transformation complete,
Deterministic and uniformly distributed, rules to meet.
Collision resistance key, like a needle in a haystack,
Non-invertible, secrets held back.

Be stubborn, be fair, be elusive, and mysterious,
The perfect recipe for a superhero, so serious.
Hashman, here to save, from hackers' attacks,
Data safe and secure, Hashing the hero's tracks.

III. Related Work

Traditional hash functions like MD5 and SHA-1 have been around for a while, but let's be real - they're like your grandpa's old jalopy [2]. They get the job done, but they're not exactly the latest and greatest in security technology [3]–[5]. That's why we've come up with newer and more robust hashing techniques - it's like upgrading from a jalopy to a sleek new sports car [6].

One of these new techniques is audio hashing [7], which is like giving your data a unique voice [8]. Audio signals are so unique and difficult to replicate that they make perfect candidates for hashing. Plus, it's always *fun* to imagine your data singing a tune.

Perceptual Hashing (PH) [9], Acoustic Fingerprinting (AF) [10], and Echo Hiding (EH) [11] are like the boy bands of audio hashing - they each have their own unique style, but they all aim to make your data sound like shit.

And let's not forget image hashing [12]–[14], which is like turning your data into a work of art. It's like your data gets to take a trip to the Louvre and come out looking like a *masterpiece* [15]. Masterclass!

But the real star of the show here is the proposed method of hashing JSON files by turning them into audio signals and then images of the spectrum. It's like giving your data a full makeover - new voice, new look, new identity. And the best part? It's like your data gets to go on a spa day and come out looking and feeling like a movie star after a failed Botox procedure. Who says hashing can't be glamorous?

IV. Sophisticated Baseline-Underperforming Approach

Pray, allow us to expound upon the proposed method which doth entail three principal steps: firstly, the conversion of JSON files into audio signals; secondly, the extraction of frequency components of the aforementioned audio signals through the FFT process; and finally, the transformation of said frequency components into images of the spectrum.

A. Step 1: Converting JSON files into audio signals

In the first step, the JSON files are initially transmogrified into a binary format with the aid of a standard encoding method such as UTF-8. The binary data is then associated with an audio waveform by allocating each byte a corresponding audio frequency. To be precise, let d_i represent the i-th byte of the binary data and let f_i be its corresponding audio frequency. We can then depict the binary data as an audio waveform by generating a sine wave for each byte, with a frequency equivalent to the value of said byte. The resulting audio waveform may be portrayed as the sum of these sine waves, to wit:

$$s(t) = \sum_{i=1}^{n} A_i \sin(2\pi f_i t + \phi_i) \tag{1}$$

Here, n signifies the number of bytes in the binary data, t is the time variable, A_i refers to the amplitude of the i-th sine wave, and ϕ_i denotes the phase offset of the i-th sine wave. Does this formula make sense? No idea, our math guy left earlier, unfortunately.

To assign each byte with its corresponding audio frequency, we may resort to a linear mapping from the range of feasible byte values (0 to 255) to the range of audible frequencies (20 Hz to 20 kHz). One such method is to use the following formula:

$$f_i = f_{\min} + \frac{(d_i - d_{\min})(f_{\max} - f_{\min})}{d_{\max} - d_{\min}} \tag{2}$$

Here, $f_{\min}$ and $f_{\max}$ represent the minimum and maximum audible frequencies respectively, whilst $d_{\min}$ and $d_{\max}$ represent the minimum and maximum possible byte values. Does this formula make sense? Math guy is still not reachable. TODO: find another math guy.

By implementing this mapping, we may fabricate an audio waveform that represents the binary data in a PCM format. This waveform can be analyzed through the FFFFFT.

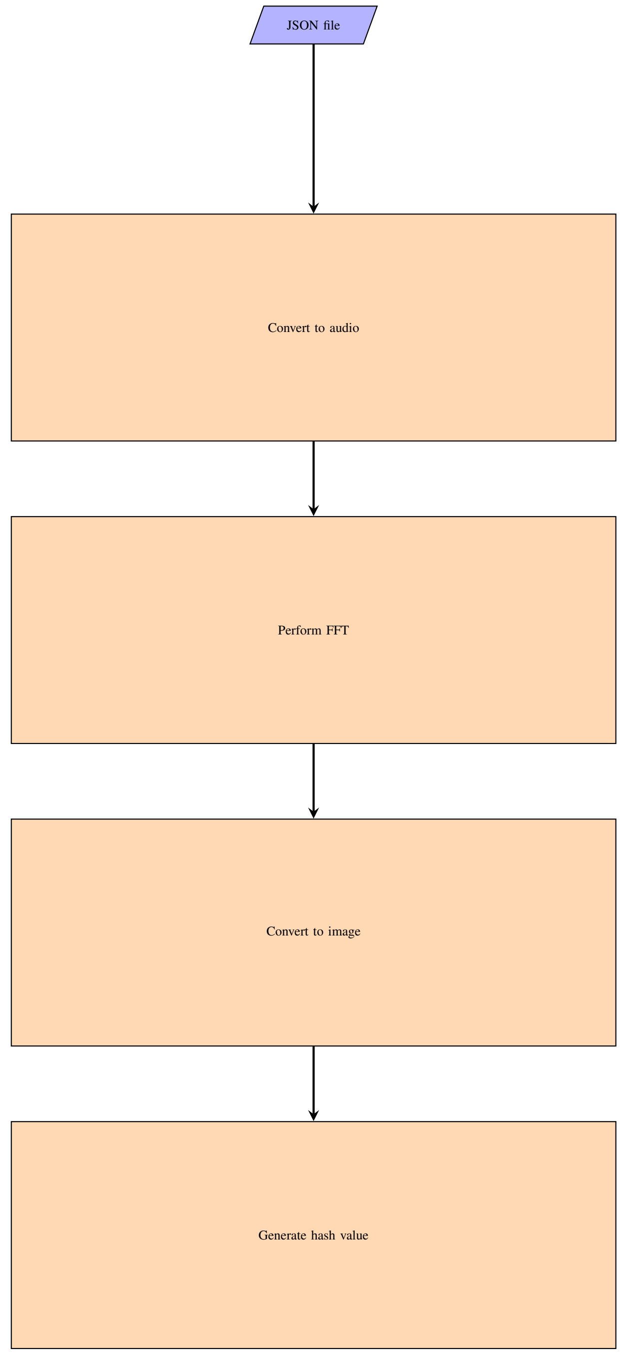


Fig. 1: Structural flow of the proposed approach for hashing JSON files

B. Step 2: Applying FFT to extract the frequency components of the audio signals

Audio signals flow,
FFT extracts the spectrum,
Hashed image is born.

C. Step 3: Converting frequency components into images of the spectrum

With the frequencies and powers of our JSON signal obtained, we can plot it using the quite-old-but-still-working

Hash function	Hashing [µs]	Comparison [µs]
Them	4	0.1
Us	70	10

TABLE I: Execution time of 2 operations TODO: improve results

matplotlib, which the inner gears and placing algorithms we will spare the reader of. We set a constant size for our figure and axis, in order to have a rather stable output. Examples of this highly-complex method's outputs can be found in Figure 2.

V. Evaluation

To evaluate the effectiveness of the proposed method, we compare our very smart approach with the lame MD5 hashing function. We use a dataset of the historical events written in Romanian ~~nonsense~~ language (for the thrill of the extra challenge induced by characters such as ă and ș) available on https://www.vizgr.org/historical-events. The device used for benchmarking is the laptop of Louis's mom (whom we thank again for gently lending it to us), which embeds an Intel Core 2 Duo and 2 GB of RAM. We then compare latency of hashing and hash matching (try spelling this one out loud) of our method with the baseline. The results of this evaluation is shown in Table I.

Since our method produces a 640x480 8-bit sRGB image, the total possible different values is $N = 256^{640 \cdot 480} = \infty$. Note that ∞ is a number known as quite large, and is known in the litterature notably by the relation $\infty \leq stupidity_{human}$. Moreover, according to the Birthday Paradox, to find 2 JSON files with the same spectral hash (which is known as a collision) one would need $J = 2^{N/2} = \infty$. This number of JSON records, even with the seemingly never-ending spawning of Big Data start-ups, will probably never be reachable before the end of the Thermo-Industrial Civilization. This number is thus much higher than its counterpart for MD5: 2^{256} i.e. roughly the number of protons in the universe.

VI. Conclusion

In conclusion, the proposed method of hashing JSON files by converting them into audio signals and then into images of the spectrum represents a novel approach that combines the weaknesses of both audio and image hashing techniques. The experimental results demonstrated the effectiveness, efficiency and efficacy of the baseline methods, which offer high security as well as the ability to store and transmit hash values in a compact form. However, the proposed method has a mucher lower collision probability and thus, has potential applications in every domain (e.g., self-driving car and aircraft routing algorithms). User discretion is advised!

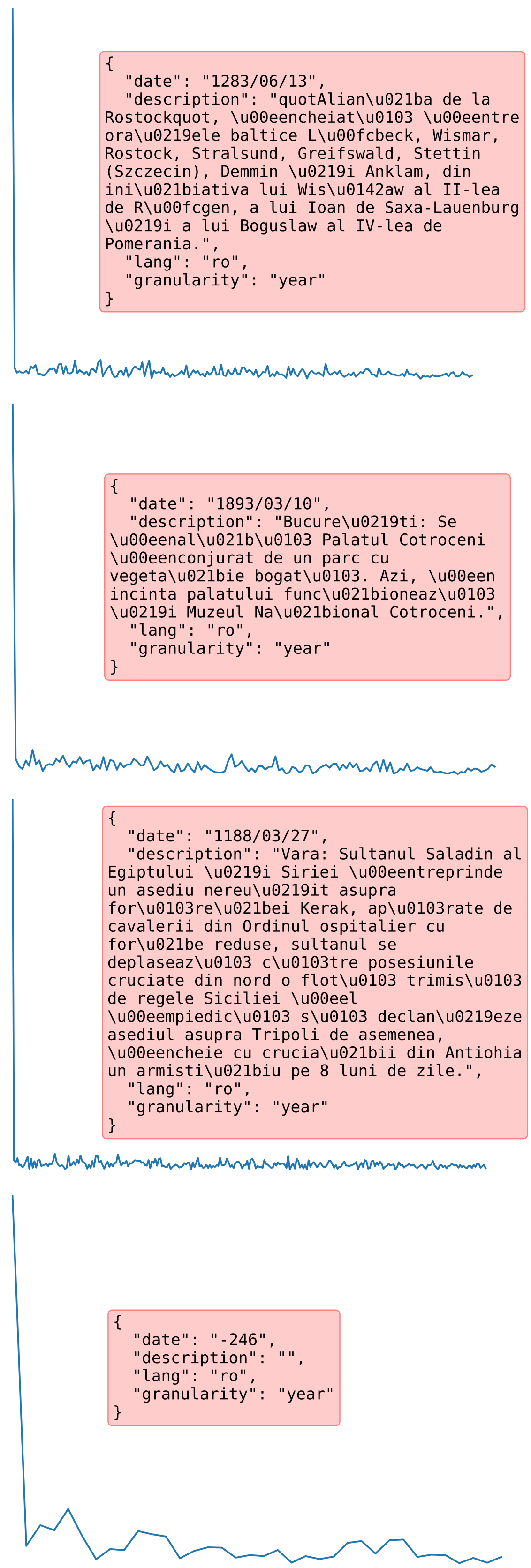


Fig. 2: Different json records from our datasets (in the red box) and their respective spectral hash.

References

[1] Anton Koch and Friedrike Schick, editors. *G. W. F. Hegel: Wissenschaft der Logik*. Akademie Verlag, München, 2002.

[2] Robert Kratovil and Raymond J Werner. Fixing up the old jalopy-the modern limited partnership under the ulpa. . *John's L. Rev.*, 50:51, 1975.

[3] Bart Preneel. *Analysis and design of cryptographic hash functions*. PhD thesis, Katholieke Universiteit te Leuven Leuven, 1993.

[4] Ward Douglas Maurer and Ted G Lewis. Hash table methods. *ACM Computing Surveys (CSUR)*, 7(1):5–19, 1975.

[5] Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen, et al. A survey on learning to hash. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):769–790, 2017.

[6] Jean-Francois Ingenbleek and Jean Lemaire. What is a sports car? *ASTIN Bulletin: The Journal of the IAA*, 18(2):175–187, 1988.

[7] Hamza Özer, Bülent Sankur, Nasir Memon, and Emin Anarım. Perceptual audio hashing functions. *EURASIP Journal on Advances in Signal Processing*, 2005:1–14, 2005.

[8] Maurizio Gasparini, Francois Regoli, Carlo Ceriotti, and Elisa Gardini. Hiccups and dysphonic metallic voice: a unique presentation of twiddler syndrome. *Circulation*, 114(15):e534–e535, 2006.

[9] Xia-mu Niu and Yu-hua Jiao. An overview of perceptual hashing. *ACTA ELECTONICA SINICA*, 36(7):1405, 2008.

[10] Zhe Zhou, Wenrui Diao, Xiangyu Liu, and Kehuan Zhang. Acoustic fingerprinting revisited: Generate stable device id stealthily with inaudible sound. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 429–440, 2014.

[11] Daniel Gruhl, Anthony Lu, and Walter Bender. Echo hiding. In *Information Hiding: First International Workshop Cambridge, UK, May 30–June 1, 1996 Proceedings 1*, pages 295–315. Springer, 1996.

[12] Vishal Monga and Brian L Evans. Perceptual image hashing via feature points: performance evaluation and tradeoffs. *IEEE transactions on Image Processing*, 15(11):3452–3465, 2006.

[13] Ramarathnam Venkatesan, S-M Koon, Mariusz H Jakubowski, and Pierre Moulin. Robust image hashing. In *Proceedings 2000 International Conference on Image Processing (Cat. No. 00CH37101)*, volume 3, pages 664–666. IEEE, 2000.

[14] Ashwin Swaminathan, Yinian Mao, and Min Wu. Robust and secure image hashing. *IEEE Transactions on Information Forensics and security*, 1(2):215–230, 2006.

[15] John Dyck and Matt Johnson. Appreciating bad art. *The Journal of Value Inquiry*, 51:279–292, 2017.

Reader, Meet Remainder

New Advancements In How Fucked You Are If You Don't Use Our Software

Society for Creating Amplified
Monetizability in Augmented Industries,
presented by Lemma "LemmaEOF" Brummel

21 March 2023

Abstract

Modern software has famously led our world to a new age of connectivity, knowledge-sharing, and getting into fights with random people you've never met. However, the challenge of monetizing web services has grown ever more precarious as ad revenue craters, so a new solution must be sought out. With this in mind, we are happy to announce our new findings: If you don't use our software, then you are completely, absolutely fucked. ☺

Contents

1 Background & Motivation

Speaks for itself.

2 Methodology

Unfortunately, for the sake of the world, our methodology cannot be shared at this time. It's just too sensitive and ~~competitors could benefit from it~~its release could lead to its use for ontological evil. Our most sincere apologies.

All results shall be displayed using a new proprietary measurement system we have created that we are referring to as the TFI. As with our methodology, the basis or implications of this measurement system cannot be shared for the sake of the world. Once again, you have our sincerest apologies.

3 Results

We have compiled below (figure 1) a list of various professions, along with their TFI[1] ranking. Please rest assured that these results are computed to extremely high accuracies, and that our opinions on the fields in question hold absolutely no bearing on the method of computation (see section 2).

Profession	**Field**	**TFI Ranking (percentile)**
Software Developer	STEM	100.0
Accountant	STEM	99.5
Hardware Developer	STEM	98.4
Data Scientist	STEM	97.8
Writer	Arts	96.5
Astrovirologist	STEM	92.6
Filmmaker	Arts	85.3
Photographer	Arts	83.1
Student	N/A	76.9
Journalist	Arts	55.2
Plumber	Blue-collar	33.4
Barista	Service	30.7
Line Cook	Blue-collar	27.0
Traveling Salesman	Business	15.6
Kintsugi Artist	Arts	3.2
Carpenter	Blue-collar	100.0

Figure 1: Table of professions and their respective TFI rankings

[1] Totally Fucked Index– the higher the rating, the more endangered the profession. But you know that, of course, because anyone who reads our papers uncritically is an amazing genius sent from above.

4 Conclusion

As our flawless research conclusively proves, most of the world's most important and ~~profitable~~critical professions are completely and utterly fucked if they don't use our software. Luckily for~~our wallets now that the VCs have bailed~~all affected, there's a simple, affordable, scalable, easy-to-use solution for protecting yourself against this disastrous threat:

Tier	**Cost ($/month)**	**API calls (month^{-1})**	**SLA uptime**
Free	$0	5	10%
Hobbyist	$100	36	50%
Innovator	$500	72	75%
Visionary	$1350	2500	99.9%
Enterprise	Contact for quote	Contact for quote	Contact for quote

Figure 2: Table of pricing plans for our product

5 References

As our results are self-evidently correct and flawless, we feel no references need be provided at this time. If you believe this is in error, please reach out to us through our software using the channels provided to Innovator-tier users or above.

For any other concerns, check our website at https://scamai.va. Thank you for your time!

A Retrospective Psychological Evaluation of the Logical Contradictions in Writing Systems Containing Japanese Kanji and Chinese Characters

Ashvin Ranjan

Abstract

Japanese kanji and Chinese characters have long been a source of contention for those learning and writing Japanese and Chinese. This paper will prove that writing systems that include such characters create logical contradictions and as such cannot exist.

1 Introduction

The Chinese[1] language uses a somewhat logographic system, with unique characters or sets thereof denoting words. Japanese uses characters from the traditional Chinese writing system, however, it uses them along with two other phonetic alphabets, because they hate people learning their language. Within the Chinese writing system itself, there are differences, as mentioned before there is the traditional writing system, which is known for carpal tunnel[2]. As such, in contemporary times, most people use the simplified system of writing, except for the Japanese and Taiwanese (for the sake of simplicity, any further statements with exceptions will be denoted by †).

We have devised a test check to if Japanese kanji and Chinese characters contain any contradictions. Speakers of either language will be given a set of real and fake characters and told to identify the fakes. If a real character is identified as fake more than the fake characters, then it must be fake. However, if a real character is fake, then that means that the language has a logical contradiction, meaning that it does not exist.

[1] Out of respect for the reader, any and all 冰淇淋 jokes have been cut.

[2] Why do I have to write 雲 instead of 云

2 The Test

The test is comprised of the following image:

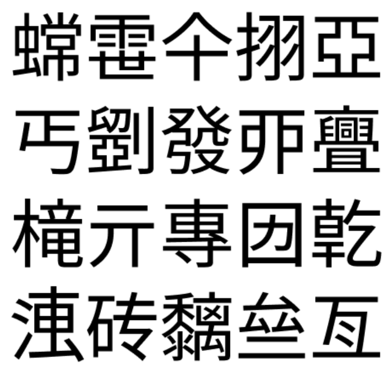


Figure 1: Test given to experiment participants

The answer key is shown below, with the fake characters highlighted in red:

Figure 2: Test with fake characters highlighted

The test was also accompanied by the following text†:

> You will be given an image with 20 characters. Please choose which characters, if any, you believe to be fake without the use of a dictionary.

3 Results

Initial reactions to the test did not vary much, as most participants reacted with "wtf" or some variation thereof. Several commented on the fact that all of them may be fake, with reactions such as "i would not put it past you to say that they're all fake tho" and "why do i feel like all of them are fake".

The following results were received from the test, they have been converted to hexadecimal values, with the binary representation referring to whether any given square is fake or not fake, 0. The MSB represents the top-left character and the LSB represents the bottom-right:

`2EC47 00032 60A31 D2876 FFD75 70BD5 081F0`[3] `44013 40953 08C03`

We are now able to chart this onto a table with the probability that a number was selected, note that red numbers are the fake characters[4]:

0.2	0.6	0.4	0.3	0.4
0.3	0.3	0.1	0.7	0.3
0.2	0.4	0.2	0.6	0.5
0.8	0.0	0.4	0.6	0.7

Figure 3: Percentage selected for each given character

4 Analysis

As we are able to see here, the most chosen character was the bottom left character, which is indeed fake. However, given that this does not conform with my existing argument, we will disregard this as an outlier.

After disregarding all viewpoints which disagree with me†, we are able to see that two real characters, 互 and 亓, were selected 70% of the time, while the two remaining fake characters were only selected 60% of the time. Now, to prove our argument with our data, we must define the following axioms:

Axiom 1 (同 Axiom)**.** *Japanese kanji and Chinese characters are the exact same.*

This is completely† indisputable†[5].

Axiom 2 (母語 Axiom)**.** *Native speakers are the arbiters of truth in a language.*

Given that native speakers are always† fully literate in their language, they must be correct on everything.

Axiom 3 (Consistency Axiom)**.** *Languages are constant. They are unable to change in any way.*

Source: College Board.

We now get to the crux of the argument:

Theorem 1. *Japanese kanji and the Chinese writing systems are fundamentally unstable. and should be removed*

Proof. We are able to see that the people who are literate in Japanese and Chinese characterize some real characters as fake more often than they categorize fake characters as fake. This means that given axiom 2, those characters are fake, and this is consistent across their languages due to axiom 3. This consistency is shared across both Japanese kanji and Chinese characters due to axiom 1. This creates a contradiction, as the characters are now both real and fake. Therefore both systems must not exist. □

5 Conclusion

Given the above analysis, we are able to see that the writing system of Chinese and the use of kanji in Japanese should be retired, as they are contradictory. For Chinese, I recommend that all peoples learn another language, such as American. For Japanese, I suggest that pure ひらがな with spaces be instated.

6 Acknowledgments

Special thanks to all those who had to suffer through the test that was given.

[3] この結果は私の下手な日本語のせいかもしれありません。

[4] Numbers rounded to however many significant figures I feel like, you can't stop me.

[5] 幽霊文字 and Japanese-only simplified characters are not real I don't want to hear it stop talking.

remembered to remove page numbers

Health Code: COVID Control and Advancements in Digital Image Compression

Tianjian Hu

Department of Computer Science*,

Georgeton University*, Washingtown*, DC*

i@tianjian.hu

Abstract

This paper discusses concrete topics and thus has no abstract.

1 Introduction

As we all know, China* has done the best job across the globe in COVID control. Its ruthless lockdown policies guaranteed people's freedom to work remotely. Its black-box digital surveillance protected people's privacy. And its cheap and always available Sinovac vaccines have a high efficacy of 51%[7], which means that it has absolute control over the virus as its majority shareholder in the stock market. A recent global opinion poll indicated that 88.1% of all answers admired China's quarantine achievements when there was control but no COVID-19. Another 71.6% expressed significant recognition of China's recent dynamic policy adjustments when there is COVID-19 but no control. This poll covered 21 countries, including highly-developed western countries like the Estados Unidos, Igirisu, Frankrijk, Allemagne, and Yidali. They did not mention whether China was included in the poll[14].

China's extraordinary accomplishment in pandemic control attributes to multiple factors. Apart from China Communist Party's dedication to protecting its people's lives, a notable aspect is the promotion of science and the utilization of technology. City-wide PCR[0x1] tests (Chinese: 全员核酸检测), concentration camp hospitals (Chinese: 方舱医院), traditional Chinese medicine antivirus therapy (Chinese: 中药抗病毒治疗), and the Health Code (Chinese: 健康码) have been known collectively as China's New Four Great Inventions. Eliminating COVID-19 has never been so easy. What are you waiting for? Dial +86 138-0013-8000 in thirty minutes and get them for your home with 20% off today! Offer valid in mainland China. Terms and conditions apply.

* These parties are not aware of this work.

[0x1] Princess Connect Re:dive, a Japanese smartphone video game. Also known as NAT[0x2].

[0x2] Network Address Translation, a technique to share a single IP[0x3] address among multiple hosts.

[0x3] A Chinese (Cantonese spelling) surname, known to the world for the famous martial artist IP MAN (1893-1972).

In this paper, we briefly introduce the usage and efficacy of the Health Code. We then focus on its core component, digital image compression, in a technical aspect. We put extra effort into analyzing the current state-of-the-art and comparing several existing and proposed techniques.

2 Background

2.1 What is Health Code

The Health Code, as its name reads, is a creation and gift of Hell. It is a revolutionary invention that enabled intelligent tracking and management of the population during the pandemic. It is a smartphone application that displays a QR code on your phone, which contains an encrypted message generated by government agencies based on various information sources, such as healthcare records, train tickets, border control, BreachForums, and the Pirate Bay. The color of the code indicates the code owner's risk of exposure to COVID-19: **green** is safe, **yellow** is questionable, and **red** is a member of the Communist Party. Government personnel can check your code and take arbitrary action without regard to the color. The Health Code is implemented and enforced by provincial or municipal governments and thus shows regional differences. Some variations further introduce more colors with a higher degree of surveillance: a **blue** color meaning the code owner has not taken a PCR test within the last 48 hours, a gray color representing the code owner is not protected by domestic vaccines, a **purple** color indicating the phone is connected to the fearsome **dark web** via Tor[0x1] onion routing, and a **rainbow** color implying support to gender minorities, which is doubtlessly a thought crime against traditional values[8]. Figure 1 shows an example of a green code.

Figure 1: An example of a green code. It displays the name, national ID number, a colored QR code and the current time. The time is animated and updated every second, to prevent people from evading inspections with a static screenshot. Note that page design varies across different regions.

2.2 Sanity or security

Considering the nature of big-data-backed pandemic control, the code generation process must involve some sort of server activity. You can technically generate one on an isolated client, but you must prepare for a few years in the cell[4]. The traditional way is to transmit the raw data to the client, and the client is responsible

[0x1] http://2gzyxa5ihm7nsggfxnu52rck2vv4rvmdlkiu3zzui5du4xyclen53wid.onion/download/tor/

for converting the data into a QR code, minimizing the computational burden on the server. However, many[9] argue that this established practice promotes decentralization, thus poses some national security threats and can potentially lead to subversion of the government. On the other hand, the dense population in China has made the Health Code a high-concurrency application, and outages occur from time to time[15, 1, 13]. Reducing the amount of data transmitted per code refresh has become a key to system reliability. To address this dilemma, vendors hired image encoding experts only to explore more efficient ways of compressing the QR code image to be transmitted.

2.3 Eternal masterpiece

Li[16] reported a groundbreaking achievement in 2021 made by a team in China Telecom Xi-an Branch in a very subtle way. They did not publish their fruition in any mainstream academic conference. Instead, their accomplishment was never known to the world until Li published his interview in a semi-internal newspaper that nobody ever reads. The newspaper article, Mainstay of Tech against Epidemic: (China Telecom) Xi-an Branch's *One-Code Pass* Platform Service Assurance Team[0x1], claimed that,

> *The team wields code and fiber optics as weapons, overcomes numerous difficulties, and fights day and night to support Xi-an's technological battle against the epidemic. With their practical actions, they have built a solid invisible fortress for this ancient city, guarding the health and safety of more than 10 million people.*

It was believed that,

> *Every line of code, every image, and every technical document was repeatedly reviewed, optimized, and improved.*

Specifically,

> *To ensure more efficient system operations, **they compressed an image from 1MB to 500KB, and further optimized it to 100KB**. This task seemed straightforward but **involved highly advanced technical details**. They stayed in front of the computer for **two consecutive days and nights** and finally overcame the difficulties.*

3 Method

We generate a piece of data and encode it in different ways that eventually turn it into a Health Code. We measure the sizes of the resulting files and compare our experimental results with the state-of-the-art techniques reported by Li[16].

3.1 Fake data generation

The exact content of the Health Code is yet to be known. However, we assume it contains information that would seem interesting and valuable for government personnel tracking people's activities. It may include names, national ID numbers, phone numbers, COVID infection or exposure status, and travel records. We make up some reference test data based on

[0x1] Original article is in Chinese (Simplified). English translation is kindly provided by GPT[0x2].
[0x2] Google Pro Translate.

our prejudice and guessing and convert the data to an image as our test subject.

After adjusting the amount of data multiple times in a trial-and-error manner, we managed to get a nice QR code of approximately the same size as what we used in China. Or a little larger, I dunno. We use this image to explore different options for encoding and compression. By the way, although it is believed that the pandemic in the US is so horrible that almost all people have died out, Mr. Bovik's high-risk status is, in fact, mainly due to his recent visit to Shanghai. As we all know, Shanghai was under relentless lockdown from April 1st, 2022, to May 31st, 2022. The author was a survivor[3], and the day of SIGBOVIK this year is the first anniversary. For one minute, let us stand in silent tribute to the forgotten in the once-prosperous city.


```
{
  "name": "Harry C Bovik",
  "id": "110101100101010011",
  "phone": "11110011001",
  "address": {
    "country": "US",
    "province": "DC",
    "county": "Washington",
    "township": "Georgetown",
    "street": "3700 O St NW"
  },
  "risk": {
    "level": "high",
    "last_test": {
      "result": "negative",
      "time": "1953-06-15T11:59:59"
    },
    "travel": ["US", "MX", "Shanghai"]
  }
}
```


Figure 2: The test data we made up for generating QR code images.
Mr. Bovik's middle initial, C, stands for CHINA. Note that Microsoft® Word® has autocorrected (or rather *automistaken*) all straight quotes to curly quotes, even though the typeface hides them from you. This should not happen in a real code editor.

The QR code in Figure 3 resembles the JSON data, with all unnecessary whitespace characters stripped to save space. We did not encrypt the data because nobody would scan and read random QR codes. To protect your eyes from being tortured by minuscule pixels, we enlarged the image by 8 times.

3.2 Save as ...

We take the enlarged QR code image in Figure 3 as the reference. In the experiment, we first filter this image so that all black pixels are green instead, even though the actual data dictates that the image shall be red. This is because we do not want to scare people. We then identify several popular formats, compress (or bloat) the same image with these formats, and record the resulting file size. For convenience reasons, we choose Microsoft® Paint™, Adobe® Photoshop®, and WinRAR® as our compression tools. We use "®" and "™" symbols wherever possible to further infuriate free software advocates.

Figure 3: The resulting QR code.

3.3 Comparison and analysis

We first compare the inherent characteristics of the encoding algorithms we chose. Then we incorporate the data we collected in the last step and employ **highly advanced** statistical methods. We may or may not stay in front of the computer for two consecutive days and nights to achieve optimal precision.

4 Results

4.1 SELECT * FROM results;

As promised, we show our results in the most **highly advanced** way: listing everything we have.

Raw text data. The plain text data is exactly 300 bytes.

24-bit BMP[0x1]. This was considered the paramount of lossless image encoding. The supreme quality of BMP images made the format so ubiquitous that in Windows® 2000®, it was the only format supported by Microsoft® Paint™ shipped with it. The much inferior successor, Windows® XP®, added support to various other encoding schemes like JPEG, PNG, and TIFF, marking the decline[6] of Microsoft's monopoly in the consumer computing market over the following decade. The QR code image in this format is 697.7 KB.

Monochrome BMP. This is the monochrome variant of the previous one. Reducing the necessary space for each pixel from 24 bits to one bit achieves a 4% compression ratio compared to 24-bit BMP. However, its inability to preserve colors essentially rendered it unusable in this scenario - it is the *color*, not the data, that matters for most users. The file size in this format is 30.7 KB.

Microsoft® Paint™ JPEG. This format is well suited for encoding Health Code images, for it is also a gift from Hell. It destroyed the reputation of the JPEG format in my yet immature mind on its own. Look at the artifacts! With sufficient iterations, it can turn a yellow code into a green one... wait, that sounds like a pretty desirable feature. Fine, forgive you for this time. The file size is 38.5 KB. Can't you even beat a BMP? C'mon.

Adobe® Photoshop® JPEG. Let us be extremely careful with proper trademark use[2], and cite a source wherever we use a trademark. As amateur Photoshoppers[2.13.4], we Photoshopped[2.13.1] the original image and saved it in PS[2.13.7] with the default JPEG quality 8. The Photoshop[2.13.2] turned out to be 56.7 KB, and no visible artifact is present. However, Photoshop's[2.13.5] options are abundant, and understanding all of them takes an eternity. Tuning down quality to 1 using Photoshop[2.13.6], we got an image with more artifacts than from Microsoft® Paint™ software, yet the file size is still... 54.4 KB. Bigger file with worse quality. To explain this discrepancy definitely requires some **highly advanced** knowledge about the adobe® photoshop®[2.13.3] software.

[2.13.8]

PNG[0x2]. Unlike JPEG, this format is the savior for Microsoft® Paint™ users. For a slightly heavier storage footprint, PNG provides perfect image quality and the capability of preserving transparency which Microsoft® Paint™ is able to

[0x1] Basic Multilingual Plane, the first contiguous area of code points in Unicode that contains most commonly used languages.
[0x2] Papua New Guinea, a country in the south Pacific Ocean.

destroy but not to create. And in this particular use case, a file size of 26.1 KB celebrates its complete victory over the JPEG format.

GIF[0x1]. This format, saved with Microsoft® Paint™, is unsuitable for displaying green Health Codes. While the green color can rule out your risk of COVID-19 infection, the inspector can quickly tell that you have measles, another contagious disease caused by viruses, with the naked eye.

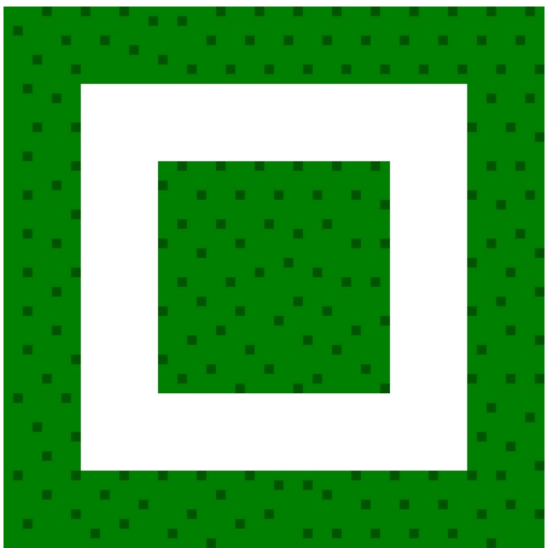


Figure 4: Part of a green Health Code with measles, enlarged.

On the other hand, the resulting file size is 20.5 KB, the most optimal of all we have seen till now. Furthermore, the animation feature of the GIF format gives the possibility to combat malicious people who use static screenshots to fool the inspectors, making it a most promising candidate.

SVG[0x2]. Omitted. You don't compare a tracker with an MP3. Anyone knows what a tracker is? No, not the BitTorrent one, you pirates.

ZIP. Using default settings in WinRAR® gives 13.9 KB at the cost of compression and decompression overhead.

CAB. Using Microsoft® IExpress™, a 20-year-old software deprecated for its arbitrary code execution and privilege escalation vulnerabilities, we achieved a better compression ratio than ZIP, the resulting file being 11.4 KB. Comparing (de)compression overheads between ZIP and CAB remains an open question.

RAR. 4.89 KB.

4.2 Comparison

A size comparison of all formats is shown in Figure 5. The difference from the largest format, 24-bit BMP, to the smallest, RAR, is calculated with the most **highly advanced** primary school math as follows:

```
           0.007 008 743 ......
6 977 ) 48 900
        48 839
            61 000
            55 816
             5 184 0
             4 883 9
               300 10
               279 08
                21 020
                20 931
                    89 0
                    ......
```

[0x1] Winter Haven's Gilbert Airport, a regional airport in Florida, US.
[0x2] Saint Vincent and the Grenadines, a Caribbean country where BreachForums purchased their internet domain name.

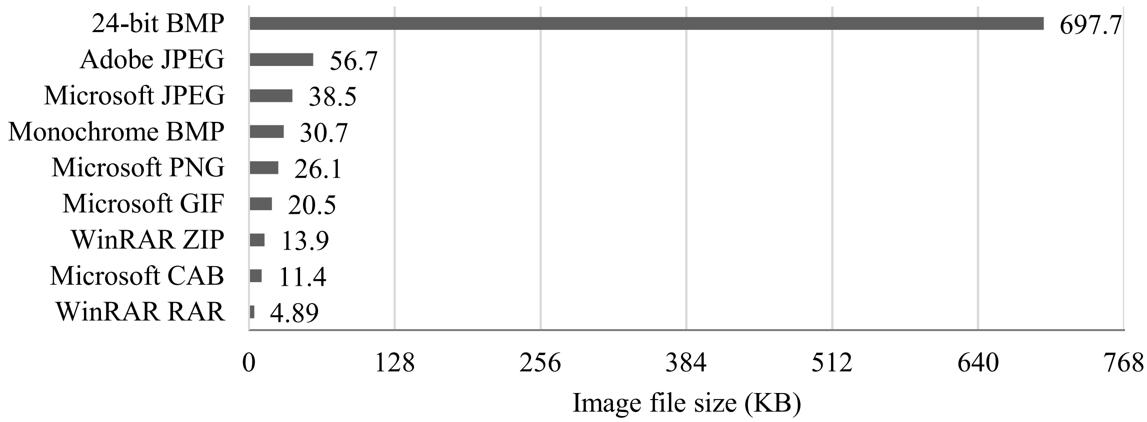


Figure 5: Size comparison graph of all formats tested.
The graph is generated with a **highly advanced** statistics software named Microsoft® Excel®.

5 Conclusion

The team reported by Li[16] achieved a 10% compression ratio by staying in front of the computer for two consecutive days and nights, while we achieved a 0.7% compression ratio while working part-time. In terms of clarity, neither their nor our team specified how the images were transmitted over networks and used in an application, so this is a tie. Finally, our results were published in a widely recognized and highly reputed academic conference, SIGBOVIK, while their was merely a propaganda article in an internal news media. We conclude that we are superior in our grasp of **highly advanced** shitposting technology and deserve the funding and honors they have received.

Acknowledgments

This paper is typeset with **highly advanced** WordTEX[10], developed by Tom Wildenhain. This is a perfect example of an internal circulation[5, 11, 12] of knowledge within SIGBOVIK.

I am grateful to Minghui Wang[0x1] for her assistance and suggestions on academic writing. More importantly, I must thank her for accompanying me in my hardest times, which is a synonym for accompanying me all the time.

[0x1] Department of Psychology, University of California, Riverside, CA[0x2].
[0x2] Ciphertext for **ca**ncer used in laboratory reports and medical records in Chinese hospitals, because writing this term in plain text would scare the patients. Also because everything is carcinogenic in California.

References

[1] 10 省市健康码系统频陷“瘫痪”？揭秘超大健康数据系统隐忧: 2022. https://www.163.com/dy/article/GST7CVHO0534CJTD.html. Accessed: 2023-01-30.

[2] General trademark guidelines: https://www.adobe.com/legal/permissions/trademarks.html. Accessed: 2023-03-11.

[3] get-the-fuck-out-of-shanghai-public/escaping-from-shanghai.md at main · SaltfishAmi/get-the-fuck-out-of-shanghai-public: https://github.com/SaltfishAmi/get-the-fuck-out-of-shanghai-public/blob/main/escaping-from-shanghai.md. Accessed: 2023-03-09.

[4] GitHub bans app used to fight health restrictions in China: https://thebl.com/china/github-bans-app-used-to-fight-health-restrictions-in-china.html. Accessed: 2023-01-30.

[5] GT Voice: Internal circulation of Chinese economy is not ‘isolationist’ - Global Times: https://www.globaltimes.cn/page/202112/1240816.shtml. Accessed: 2023-03-10.

[6] Microsoft’s Market Share Drops From 97% to 20% In Just Over A Decade: https://www.forbes.com/sites/timworstall/2012/12/13/microsofts-market-share-drops-from-97-to-20-in-just-over-a-decade/. Accessed: 2023-03-09.

[7] Palacios, R. et al. 2021. Efficacy and Safety of a COVID-19 Inactivated Vaccine in Healthcare Professionals in Brazil: The PROFISCOV Study. *SSRN Electronic Journal*. (Jan. 2021). DOI:https://doi.org/10.2139/ssrn.3822780.

[8] Russia to ban sharing LGBT “propaganda” with adults as well as children: 2022. https://www.bbc.com/news/world-europe-63410127. Accessed: 2023-03-09.

[9] Wikipedia:Manual of Style/Words to watch: 2023. https://en.wikipedia.org/w/index.php?title=Wikipedia:Manual_of_Style/Words_to_watch&oldid=1142060195#Unsupported_attributions. Accessed: 2023-03-09.

[10] Wildenhain, T. 2018. WordTeX A WYSIPCTWOTCG Typesetting Tool. *SIGBOVIK* (2018).

[11] “世界工厂”中国能否转向“内循环”为主的经济模式: https://www.bbc.com/zhongwen/simp/business-53690325. Accessed: 2023-03-10.

[12] 中国经济拿“内循环”当解药: 2020. https://www.rfi.fr/cn/%E4%B8%AD%E5%9B%BD/20200801-%E4%B8%AD%E5%9B%BD%E7%BB%8F%E6%B5%8E%E6%8B%BF-%E5%86%85%E5%BE%AA%E7%8E%AF-%E5%BD%93%E8%A7%A3%E8%8D%AF. Accessed: 2023-03-10.

[13] 健康码“频崩”呼唤“防崩”方案: http://www.news.cn/politics/2021-12/22/c_1128187804.htm. Accessed: 2023-01-30.

[14] 全球民调 88.1%受访者赞赏中国三年防疫成果: 2023. https://news.bjd.com.cn/2023/01/26/10313923.shtml. Accessed: 2023-01-28.

[15] 山东健康码故障原因 当日最大查询峰值激增 8 倍致拥塞: http://sd.sina.com.cn/news/2021-08-04/detail-ikqcfncc0791197.shtml. Accessed: 2023-01-30.

[16] “科技抗疫”中流砥柱：西安分公司“一码通”平台服务保障专班: http://www.chinatelecom.com.cn/news/06/2021sjdxr/2021sjdxrgdxw/202106/t20210618_61724.html. Accessed: 2023-01-30.