THE COMP6258 REPRODUCIBILITY CHALLENGE

J. Cox, L. Nicholas, A. Şontea & X. Wu
University of Southampton

ABSTRACT

We reproduce and critically assess *"Learning the Greatest Common Divisor: Explaining Transformer Predictions"*, which claims transformers trained on GCD tasks learn interpretable sieve-like algorithms. Our streamlined PyTorch reimplementation matches reported accuracies within 2% while significantly reducing training time. However, further experiments contradict stable rule learning. Rapid fine-tuning improvement on rare classes suggests reliance on statistical frequency rather than algorithmic reasoning. We conclude that output-level heuristics alone are insufficient for claims of interpretability. The repository can be found here.

1 INTRODUCTION

Transformers have benefited from a quick and wide adoption thanks to their capabilities associated with generative tasks. Their remarkable ability to produce human-like outputs brought widespread impact to the field of language processing. However, language transformers are known to suffer from phenomena like hallucinations. This motivates the need to develop a better understanding of the internal processes of these models, which are often black-box entities. Previous efforts focused on training transformers on basic mathematical operations that can be used to prove that their inference process is explainable.

The paper investigates whether transformer models, when trained to compute the greatest common divisor (GCD) of two integers, exhibit interpretable behaviour. It claims that model predictions can be fully characterised by a small set of human-understandable rules, suggesting a form of output-level interpretability. These rules are derived through black-box, input–output analysis and include early reliance on divisibility patterns in the input base, gradual learning of small prime factors (grokking), and eventual convergence to correct GCDs via a sieve-like mechanism rather than Euclid's algorithm. The work presents a novel attempt to formalise interpretability through structured arithmetic tasks.

The paper presents a novel method for encoding integers within transformer architectures, representing each number as a sequence of digits in a positional base. This encoding is designed to expose cues related to numeric magnitude and divisibility to the model's attention mechanism (see Table 1).

Table 1: Encoding $\gcd(160, 120) = 40$, in base 10 & 30

BASE	ENCODED INPUT	ENCODED OUTPUT
10	[+,1,6,0,+,1,2,0]	[+,4,0]
30	[+,5,10,+,4,0]	[+,1,10]

2 EXPERIMENTAL DESIGN

Initial analysis of the papers original papers codebase lead to the decision to create a new implementation due to deprecated Python packages, complex code structure producing long training times and issues around evaluating and understanding model outputs.

Two versions of the new implementation were then created one version for local training on Windows computers and a second version for Iridis. Both versions made use Pytorch optimised transformer implementation. This approach significantly reduced the training time for the exact same

configuration as the original paper. Testing using smaller training datasets showed only small reductions in the models performance. The uses of reduced datasets was subsequently used for the investigation of the core claims found in the paper.

Multiple models where trained on different bases with reduced datasets to reproduce the relation between base and test accuracy shown in the original papers base experiments. An extend training run in base 10 with both uniform and log-uniform training dataset distributions

2.1 LOWEST COMMON MULTIPLE EXPERIMENT

In order to test the wider applicability and reproducibility of the papers approach to interpretability it was decided to train a transformer with the same architecture to solve a more difficult mathematical problem that has a similar structure to the one in the original paper given the relation $\text{LCM}(a, b) = \frac{a \cdot b}{\text{GCD}(a,b)}$. It was decided to investigate finding the lowest common multiple between two numbers.

3 RESULTS AND ANALYSIS

3.1 BASE EXPERIMENTS

Table 2: Uniform-distribution trained model test accuracies by base

Base	2	3	4	5	6	7	10	12
Original Paper	0.816	0.689	0.814	0.640	0.915	0.625	0.847	0.915
New Implementation	0.803	0.682	0.804	0.633	0.894	0.622	0.827	0.892

Table 2 demonstrates that when trained on uniform datasets the same pattern in the change in test accuracy with different bases is present. For the identical training to the original paper in base 10 uniform test accuracy of 84.1%, and log-uniform test accuracy of 93.9% comparable with original paper results. A interesting result of note was that the experiments carried out were unable to reproduce the low performance in base 2 for log-uniform achieving test accuracy of 80.1%.

3.2 LCM RESULTS

The transformer was trained on both uniform and log-uniform distributions, but only achieved $< 1\%$ testing accuracy. Despite being a heavily related operation the transformer does not appear to learn only memorising the training data if given sufficient epochs.

If the assumption was that the transformer was able to generalise some deterministic algorithm or rules to follow from training this result is hard to explain. However, if the transformer is nothing more than a statistical learning machine, then a plausible explanation can be given.

Diaconis & Erdös (2004) demonstrated that the distribution of GCDs is structured, and thus an underlying pattern can be learnt. But, this property does not transfer to the LCM calculations. Kim (2022) shows that the natural distribution of two randomly sampled integers produces a specific LCM is 0. That is, for discrete classification task, **all the classes generated are noise**. The statistical learning machine attempts to capture the underlying distribution of the data given, and when the data seems like noise, it produces noise.

3.3 UNCLEAR AND CONFLICTING RULES

Under the assumption that results presented in the paper are accurate and representative of the full experiment, the paper still does not provide satisfactory explainability as claimed. As a baseline, an 'explainable' model should produce output that "makes sense" or gives an understanding of why the model is behaving in a certain way. To assess the claims made by the paper, we assess the three rules presented, i.e. G1, G2 and G3.

The first and third rules are plausible. If the model has learned a representation of divisibility, then it is reasonable that inputs like 16 and 32, both divisible by 2,4,6 & 8, may be clustered together.

However, this interpretation depends on the reader projecting number-theoretic understanding onto the outputs rather than the experiment itself uncovering such understanding. The second rule, however, arguably weakens the model's interpretability: **If** prediction correct **then** prediction = products of primes divisors of B and small primes.

Since this is not an equivalent statement, this rule in practice is a weak condition. It cannot make predictions on what GCDs the model can predict, and does not explain why certain GCDs cannot be learned despite being products of prime divisors of B. It fails to explain why some values (e.g., 16) are learned while others (e.g., 32), despite satisfying the same numerical criteria, are not. Further, the paper attempts to give a rule on which GCD will be predicted based on the condition $kf = B^2$, with f being the largest factor divisible by the powers of the bases. This however does not hold consistently, as neither 16 nor 32 is a divisor of 100 (base 10 squared), yet 16 is predicted correctly.

3.4 Algorithmic Reasoning Or Statistical Learning?

Through generating uniformly sampled input integer pairs or a mixture with uniformly sampled output GCDs, it was possible to reproduce the training accuracy claimed by the paper to within 1-2% accuracy. Again, Diaconis & Erdös (2004) demonstrated that the distribution of GCD, P(gcd(a, b)=k) = $6/(\pi k)^2$, decays quadratically therefore a small delta in accuracy could massively impact the model's ability to calculate a GCD.

The paper opted to use the number of GCDs "grokked" as a benchmark for model performance rather than accuracy. However, it is unclear what metric has been used to determine if a number is "grokked". In our reproduced experiment, a number is "grokked" if it achieves and maintains over 90% accuracy when validated against uniformly sampled GCDs.

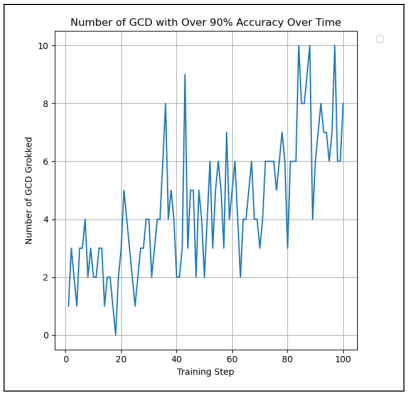

Figure 1: Correct GCD vs training time. 5% uniform, 95% natural GCD.

Following the experimental setup explained in Fig. 2 of the paper, we are unable to reproduce the behaviour of slow flat numbers of GCDs learnt followed by a sudden spike. In fact, as shown in figure 1, although the numbers of GCDs learned over time does indicate a positive trend over time, the movements more closely resembles a Brownian motion rather than any definitive ""grokking" behaviour. Investigating the problem further, we observe the same violent movement in individual GCD accuracy in figure 2.

It is questionable that the model learns a generalisable rule in one epoch only to abandon it in the next. Further evidence against consistent rule learning is the model's poor performance on out-of-distribution inputs. While achieving 89.7% accuracy on inputs sampled from $[1, 10^6]$, accuracy drops to 51% when evaluated on the restricted range $[1, 50]$. This disparity is more plausibly explained by statistical bias: as the paper notes, 90% of training inputs exceed 100,000, meaning the model is effectively trained on longer sequences and exhibits reduced performance when tested on shorter, under-represented inputs.

As discussed in Section 3.3, one of the paper's claims suggests that GCDs not composed of base prime factors cannot be learned but are merely clustered. However, when treating the task as a

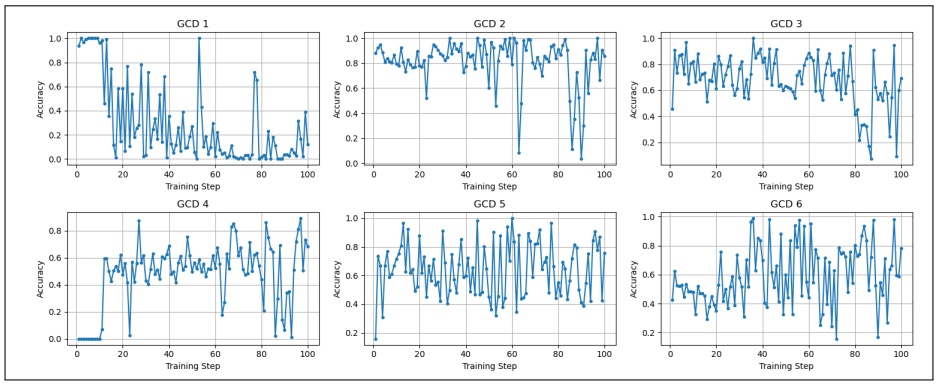

Figure 2: Per GCD accuracy vs training time. 5% uniform, 95% natural GCD.

standard classification problem, we generated 10,000 examples with GCD 19 encoded in base 30 and fine-tuned the model for a single epoch, achieving perfect accuracy for that class. This result is unsurprising: once reframed statistically rather than algorithmically, it becomes clear that the model benefits from balanced and diverse input-output distributions. Indeed, the paper's own findings show that the highest accuracy and greatest number of GCDs learned occur under balanced sampling, but at the expense of interpretability.

4 Conclusion

This paper questions claims of input-output based interpretability. While some results in the original paper are reproducible (i.e. the accuracy), the significance claim about the continuous grokking of GCDs are not revealed in our experiment.

Dziri et al. (2023) challenge the assumption that interpretability can be inferred from input-output behaviour alone, finding transformers solving compositional reasoning problems rely on pattern matching rather than algorithms and may not be applying any meaningful rule. As with the GCD paper, accurate predictions may arise from shallow heuristics rather than generalisable understanding. Zhong et al. (2023) studied modular addition and found that even small architectural changes led models to learn different internal procedures; some interpretable, others unfamiliar but consistent, showing interpretability is highly sensitive to model design and training dynamics. This highlights the existence of multiple possible internal solutions, not all of them human-comprehensible, contradicting the assumption that one rule-based explanation governs model behaviour. The concept of grokking itself also remains poorly understood (Power et al., 2022) and yet is relied upon in the 'rules' explaining transformer predictions.

As such, claims that this work represents a first step toward general transformer interpretability should be treated cautiously.

References

Persi Diaconis and Paul Erdös. On the distribution of the greatest common divisor. *Lecture Notes-Monograph Series*, 45:56–61, 2004. ISSN 07492170.

Nouha Dziri and Vishvak Murahari Chaudhary. Faith and fate: Limits of transformers on compositionality. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Sungjin Kim. On the distribution of lcm of k-tuples and related problems, 2022.

S. Power, Y. Burda, H. Edwards, I. Babuschkin, and V. Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

Yilun Zhong, Jacob Andreas, Zhihao Lin, and Christopher D. Manning. The clock and the pizza: Two stories in mechanistic explanation of neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.