Learning the greatest common divisor Explainable predictions in transformers

François Charton
Meta
fcharton@meta.com

Abstract

I train small transformers to calculate the greatest common divisor (GCD) of two positive integers, and show that their predictions are fully explainable. During training, models learn a list $\mathcal{D}$ of divisors, and predict the largest element of $\mathcal{D}$ that divides both inputs. I also show that training distributions have a large impact on performance. Models trained from uniform operands only learn a handful of GCD (up to 38 out of 100). Training from log-uniform operands boosts performance to 73 correct GCD, and training from a log-uniform distribution of GCD to 91.

1 Introduction

Transformers [26] have been applied to many problems of mathematics [12, 3, 23, 2]. Yet, they struggle with basic arithmetic [13, 18, 6], despite recent progress on fine-tuning techniques for large language models [19, 27, 31]. For instance, experiments with rational arithmetic (Appendix A) show that while transformers can learn to compare fractions, basic arithmetic operations, like addition or reduction to lowest terms, are still beyond their reach. On mathematical tasks, transformer predictions were also found to be brittle [28], to randomly fail on simple problems [5], and to be difficult to explain, except in the simplest cases [17].

In this paper, 4-layer sequence-to-sequence transformers are trained to compute the greatest common divisor (GCD) of two positive integers, a key operation for arithmetic on rational numbers, and a common fixture of number theory. I show that, throughout training, model predictions are fully explainable. When its operands are encoded in base B, the model learns a set of divisors $\mathcal{D}$, and predicts, for an input pair (a, b), the largest element in $\mathcal{D}$ that divides a and b. For small bases B, only the products of primes divisors of B are learned, and models trained on composite bases (e.g. $B = 420$) can learn to predict up to 38 of the first 100 GCD. For larger bases, small primes not dividing B are "grokked" [21] when models are trained long enough. I also show that better performance can be achieved by engineering the training distribution: models trained from log-uniform operands predict 73 GCD out of 100, and 91 when trained on log-uniform operands and outcomes, i.e. over-sampling simple examples and large GCD.

2 Experimental settings

GCD calculations are framed as a supervised translation task. Pairs of integers (a, b) are randomly sampled between 1 and 10^6, and encoded as sequences of digits in base B, preceded by a sign token (always + in this paper) which also serves as a separator. 4-layer transformers, with 512 dimensions and 8 attention heads, are trained to predict gcd(a, b), also encoded in base B, by minimizing the cross-entropy between model predictions and correct solutions. After each epoch (300,000 examples), models are tested on two sets of 100,000 examples. The *natural test set* contains pairs (a, b) uniformly sampled between 1 and 10^6. The GCD in this set verify $P(\gcd(a, b) = k) = \frac{6}{\pi^2 k^2}$ [1], i.e. small GCD are more common. In the *stratified test set*, GCD are uniformly distributed between 1 and 100,

37th Conference on Neural Information Processing Systems (NeurIPS 2023) Workshop on MATH-AI.

i.e. there are about 1000 examples of each GCD. Accuracy on the natural test set is the probability that the GCD of a random pair of integers is correctly predicted. On the other hand, accuracy on the stratified test set is the number of GCD under 100 correctly predicted by the model. The size of the problem space (10^{12} possible input pairs) guarantees minimal duplication between train and test set. See appendix B for more details.

3 Learning the greatest common divisor

A model trained on pairs of positive integers under one million, encoded in base $B = 10$, correctly predicts 84.7% of the examples in the natural test set, and 13 GCD under 100 (measured on the stratified test set). Model performance varies with the encoding base: from 61.8% accuracy and 2 correct GCD for base 11, to 96.8% and 38 GCD for base 420. The best performances are achieved for composite bases (30, 60, 210 and 420), the worst for large primes (Table 1). Learning is very fast: for base 30, the model achieves 90% accuracy after 2 epochs (600,000 examples), and 93% after 6. Model size has little impact on performance (Appendix C). For base 30, the same accuracy (93%) is achieved with 1-layer transformers with 32 dimensions (300,000 parameters) and 24-layer models with 1024 dimensions (714 million parameters). For base 31, accuracy is 61% for all models.

Base	2	3	4	5	6	7	10	11	12	15
Correct GCD	7	5	7	3	19	3	13	2	19	9
Accuracy	81.6	68.9	81.4	64.0	91.5	62.5	84.7	61.8	91.5	71.7
Base	30	31	60	100	210	211	**420**	997	1000	1024
Correct GCD	27	2	28	13	32	1	**38**	1	14	7
Accuracy	94.7	61.3	95.0	84.7	95.5	61.3	**96.8**	61.3	84.7	81.5

Table 1: Number of correct GCD under 100 and accuracy. Best of 6 experiments.

Table 2 presents, for $B = 2$, the most frequent model prediction for pairs with a given GCD (Pred), and its frequency on the stratified test set (%). Detailed results for 6 bases are in Appendix E. All frequencies are close to 100%: for all test pairs with GCD k, the model makes the same prediction $f(k)$. Correct predictions ($f(k) = k$) happen when k is a product of divisors of the base (powers of two for $B = 2$). On the other hand, pairs with an odd GCD are always predicted as 1, and pairs with an even GCD as the largest power of 2 dividing both operands. A likely explanation for these results is that the model predicts GCD by counting the rightmost zeros in its input. An integer divisible by 2^n has n zeros as its rightmost digits, if the operands, a and b have z_a and z_b rightmost zeros in their base-2 representation, the model predicts 2^z, with $z = \min(z_a, z_b)$. For instance, it will (correctly) predict the GCD of $8 = 1000_2$ and $2 = 1100_2$ (2 and 3 rightmost zeros) as $2^2 = 4$, and (incorrectly) predict the GCD of $7 = 111_2$ and $14 = 1110_2$ as 1. More generally, model predictions for all bases can be summed up by the following **three rules**:

GCD	Pred	%	GCD	Pred	%	GCD	Pred	%
1	**1**	100	13	1	100	25	1	100
2	**2**	100	14	2	100	26	2	100
3	1	100	15	1	100	27	1	100
4	**4**	100	16	**16**	100	28	4	100
5	1	100	17	1	100	29	1	100
6	2	100	18	2	100	30	2	100
7	1	100	19	1	100	31	1	100
8	**8**	100	20	4	100	32	**32**	99.9
9	1	100	21	1	100	33	1	100
10	2	100	22	2	100	34	2	100
11	1	100	23	1	100	35	1	100
12	4	100	24	8	100	36	4	100

Table 2: Model predictions and their frequencies, for GCD 1 to 36, and B=2. Correct predictions in bold face.

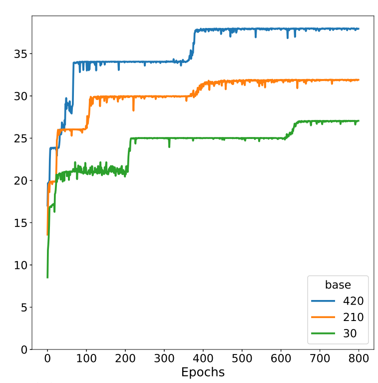

Table 3: Correct GCD vs training time. Natural ($\frac{1}{k^2}$) distribution of GCD.

(R1) **Predictions are deterministic.** The model predicts a unique value $f(k)$ for almost all (99.9%) pairs of integers with GCD k. Predictions are correct when $f(k) = k$.

(R2) **Correct predictions are products of primes dividing B.** For base 2, they are 1, 2, 4, 8, 16, 32 and 64. For base 31, 1 and 31. For base 10, all products of elements from $\{1, 2, 4, 8, 16\}$ and $\{1, 5, 25\}$. For base 30, all products of $\{1, 2, 4, 8\}$, $\{1, 3, 9, 27\}$ and $\{1, 5, 25\}$.

(R3) **f(k) is the largest correct prediction that divides k.** For instance, $f(8) = 8$, and $f(7) = 1$, for base 2 and 10, but $f(15) = 5$ for base 10 and $f(15) = 1$ for base 2.

All **learning curves** have a step-like shape (Figure 3). GCD are learned in batches: the model learns a power of a prime divisor of B, and all its products with already known GCD. For instance, for $B = 30$, the model initially predicts $\{1, 2, 4\}$, $\{1, 3, 9\}$, $\{1, 5\}$ and their products: 17 GCD under 100. Around epoch 50, the model learns 25 and the three associated multiples 50, 75 and 100 (21 GCD). Around epoch 220, it learns 8, 24, 40 and 72, and around epoch 660, it learns 27 and 54, for a grand total of 27 correct GCD. For base 210, the model begins wit the 20 products of $\{1, 2, 4\}$, $\{1, 3\}$, $\{1, 5\}$ and $\{1, 7\}$. It learns 9 and 5 multiples at epoch 30, 25 and three multiples at epoch 400, and 49 and 98 at epoch 500, for a total of 32 correct GCD. During training, the three rules hold at all times.

In these experiments, models cannot compute GCD in the general case. Instead, they leverage representation shortcuts to predict a small number of easy, but common instances: products of divisors of the base. As a result, to achieve high performance, one must select a base divisible by many small primes, e.g. small multiples of 30 or 210. Yet, all models learned to classify pairs of integers according to their GCD, and make a unique prediction $f(k)$ for all pairs with GCD k. This is an important result and a significant achievement.

Large bases and grokking. When models using large bases are trained for a long time, a phenomenon similar to grokking [21] is observed. Figure 1 presents learning curves (loss and correct GCD) for $B = 2023 = 7 \cdot 17^2$. After about 10 epochs, 3 GCD (1, 7 and 17) are learned, as per the three rules. The training loss is flat for the next 100 to 200 epochs (the duration varies with model initialization), and it looks like the model is no longer learning. Around epoch 100, GCD 3 is learned, together with $21 = 3 \cdot 7$ and $51 = 3 \cdot 17$, in just a few epochs. Then, 2, 6, 14, 34, 42 are learned at epoch 200, and 4 and associated multiples at epoch 600, for a total of 16 correct GCD. During training, model predictions still respect rules R1 and R3 (Table 10), only rule R2 must be updated to: **correct predictions are products of primes divisors of B, and small primes,** learned (roughly) in order.

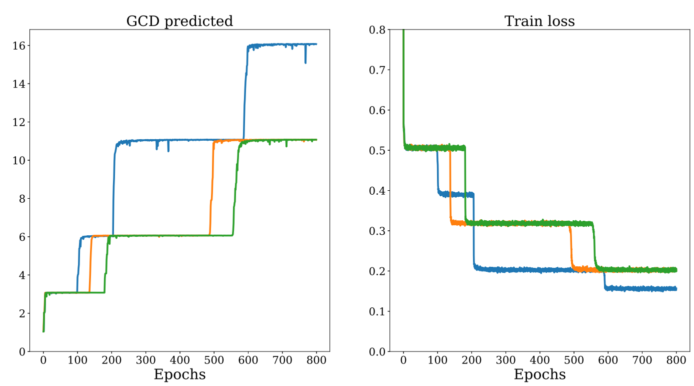

Figure 1: **Learning curves for base B=2023.** 3 different model initializations.

Table 11 presents results for 16 large bases, with models trained up to 1300 epochs. Grokking usually sets in late during training, and most of the time, primes and powers of primes are grokked in order. Because it helps learn small GCD, grokking boosts model accuracy (from 63% to 91% for $B = 2023$), but overall the number of correct GCD remains low (under 30 for all large bases).

4 Learning from log-uniform operands

So far, all pairs (a, b) in the training sets are uniformly sampled between 1 and 10^6. As a result, models are mostly trained from examples with large operands. 90% of operands are larger than 100,000, and small instances, like $\gcd(6, 9)$, are almost never encountered. This contrast with the way we are taught, and teach, arithmetic. We usually insist that small examples should be mastered, and sometimes memorized, before larger instances, like $\gcd(102370, 102372)$ can be tackled. In this section, training pairs are sampled from a log-uniform distribution, by uniformly sampling real

Base	Accuracy	Correct GCD	Base	Accuracy	GCD	Base	Accuracy	GCD
2	94.4	25	60	98.4	60	2025	99.0	70
3	96.5	36	100	98.4	60	2187	98.7	66
4	98.4	58	210	98.5	60	2197	98.8	68
5	97.0	42	211	96.9	41	2209	98.6	65
6	96.9	39	420	98.1	59	**2401**	**99.1**	**73**
7	96.8	40	625	98.2	57	2744	98.9	72
10	97.6	48	997	98.3	64	3125	98.6	65
11	97.4	43	1000	99.1	71	3375	98.8	67
12	98.2	55	1024	99.0	71	4000	98.7	66
15	97.8	52	2017	98.6	63	4913	98.2	57
30	98.2	56	2021	98.6	66	5000	98.6	64
31	97.2	44	2023	98.7	65	10000	98.0	56

Table 4: **Accuracy and correct GCD (up to 100), log-uniform operands.** Best of three models, trained for 1000 epochs (300M examples). All models are tested on 100,000 pairs, uniformly distributed between 1 and 10^6.

numbers $0 \leq x \leq \log M$, computing e^x and rounding to the nearest integer. In this setting, there are as many 1-digit as 6-digit operands in the training set. In 3% of training examples, both operands are smaller than 10. In 11%, they are smaller than 100. This presents the model with many simple examples that it can memorize, just like children rote learn multiplication and addition tables.

Note that this is different from curriculum learning: the training distribution is not modified during training. Note also that log-uniform sampling only applies to the training set (test sets are unchanged) and that it has no impact on the distribution of outcomes.

Training from log-uniform operands greatly improves performance (Table 4). Accuracy for all bases is between 94 and 99%, vs 61 and 97% with uniform operands. Up to 73 GCD are correctly predicted (for $B = 2401$), vs 38 with uniform operands. Overall, log-uniform operands accelerate grokking. For the best models, all primes up to 23, some of their small powers, and all associated multiples are learned. This brings model accuracy on random pairs to 99%, and the number of correct GCD under 100 to 73. The three rules still apply: predictions are deterministic, for a pair (a, b) with GCD k, the model predicts the largest correct GCD that divides k.

During training, rules G1 and G3 are temporarily violated when the model learns a new divisor. For a few epochs, model predictions are split between the old and the new value (e.g. between 7 and 49 when the model is learning 49). This situation, rarely observed in previous experiments, is common with log-uniform operands. The learning curves are smoother, and transitions span several epochs.

Log-uniform outcomes. Model performance can be further improved by balancing the distribution of GCD in the training set – i.e. making it scale as $\frac{1}{k}$ instead of $\frac{1}{k^2}$ (Table 5). In this setting, models with base larger than 1000 predict 87 to 91 GCD: all primes up to 53 and all composite numbers up to 100. These are our best results.

Base	Accuracy	Correct GCD	Base	Accuracy	GCD	Base	Accuracy	GCD
2	16.5	17	60	96.4	75	**2025**	**97.9**	**91**
3	93.7	51	100	97.1	78	2187	97.8	91
4	91.3	47	210	96.2	80	2197	97.6	90
5	92.2	58	211	95.3	67	2209	97.6	87
6	95.2	56	420	96.4	88	2401	97.8	89
7	93.0	63	625	96.0	80	2744	97.6	91
10	94.3	65	997	97.6	83	3125	97.7	91
11	94.5	57	**1000**	**97.9**	**91**	3375	97.6	91
12	95.0	70	1024	98.1	90	4000	97.3	90
15	95.4	62	2017	97.6	88	4913	97.1	88
30	95.8	72	2021	98.1	89	5000	97.1	89
31	94.4	64	2023	97.5	88	10000	95.2	88

Table 5: **Accuracy and correct GCD, log-uniform operands and outcomes.** Best model of 3.

5 Related work

Neural networks for arithmetic were first proposed by Siu and Roychowdury [24], and recurrent models by Kalchbrenner et al.[10], Zaremba et al. [29] and Kaiser and Sutskever [9]. Most recent research focuses on fine-tuning LLM on arithmetic tasks, to solve math word problems [15, 7]. See Lee et al. [13] for a summary. As an alternative, Neural Arithmetic Logical Units [25, 16] learn exact computations that can generalize to any input, by constraining the weights of linear layers to be close to 0, 1 or -1.

The difficulty of learning arithmetic tasks was discussed by many authors. Saton et al. [22], benchmarking mathematical tasks, observe that number theoretic operations, like factorization, are hard. Palamas [20] further investigates the hardness of modular arithmetic. Dziri et al. [6] note the difficulty of extending the promising results obtained by Lee et al. [13] on the four operations to complex mathematical algorithms (like Euclid's algorithm for the GCD, considered here).

The role of number representation was discussed by Noguera et al. [18] and Charton [2]. **Grokking** was first described by Power et al. [21]. Liu et al. [14] propose metrics to characterize it. Gromov [8] provides an insightful analysis of grokking in feed-forward networks. Most prior work on **explainability in arithmetic transformers** tries to interpret model weights [17, 30]. [4] proposes a similar analysis for linear algebra.

6 Conclusion

Model explainability is probably the most striking feature of these experiments. It is often repeated that transformers are incomprehensible black-boxes, that sometimes confabulate and often fail in unpredictable ways. Here, model predictions can be fully explained by a small number of rules, which suggests that **transformers learn a sieve algorithm for computing GCD**.

Specifically, the model learns rules for divisibility, uses them to partition its inputs into classes of pairs sharing a common divisor, and predicts each class as its minimum (and most common) member. Divisors of the encoding base, which can be tested by looking at the rightmost digits in the representation of inputs, are learned first. For base 2, the model classifies its inputs into pairs divisible by 1, 2, 4 or 8. As training proceeds, new prime divisors are learned, roughly in order. They are all prime because multiples of previous divisors were learned already, i.e. the model functions like a sieve. When a new divisor p is learned, new classes are created by splitting all existing classes between multiples and non-multiples of p. In base 2, when the model learns divisibility by 3, six new classes are created: multiples of 3, 6, 12, 24, 48 and 96. Eventually, all GCD will be learned. This algorithm is not related to Euclid's algorithm, and less efficient.

This approach to explainability differs from most works on the subject. Instead of looking at model parameters, experiments are engineered, that reveal the algorithms that the model is implementing. This is a promising direction for future research.

The role of training distributions is another important result. The best models are trained from log-uniform operands and outcomes. All models are tested on sets with uniform operands, but our best results are achieved with a log-uniform distribution of operands and outcomes in the training set. This may come as a surprise, since many authors observed that evaluating a model out of its training distribution has a negative impact on performance. The existence of special training distributions, that allow for faster learning and more robust models (with respect to out-of-distribution generalization) was already observed for linear algebra [4]. Log-uniform operands help the model learn the large instances of the problem by memorizing small, and easier, cases. This is related to curriculum learning, but because the training distribution never changes, it prevents catastrophic forgetting. A log-uniform distribution of outcomes helps balance the training set by making large GCD more common.This is a classic recipe in machine learning: classifiers are usually trained on balanced datasets. These findings may apply to other arithmetic tasks, notably fine-tuning large language models on math word problems.

Is it really grokking? The characterization of the phenomenon observed in section 3 as grokking is not entirely correct. Power [21] defines grokking as "generalization far after overfitting." In our experiments, training and test data are generated on the fly from a very large problem space. No overfitting can happen, and the classical pattern of grokking, train accuracy dropping, and validation accuracy catching up after a long time, will not occur. The similarity with grokking lies in the sudden change in accuracy after a long stagnation of the training loss.

References

[1] Ernesto Cesàro. Question 75 (solution). *Mathesis*, (3):224–225, 1883.

[2] François Charton. Linear algebra with transformers. *arXiv preprint arXiv:2112.01898*, 2021.

[3] François Charton, Amaury Hayat, and Guillaume Lample. Learning advanced mathematical computations from examples. *arXiv preprint arXiv:2006.06462*, 2020.

[4] François Charton. What is my math transformer doing? – three results on interpretability and generalization. *arXiv preprint arXiv:2211.00170*, 2022.

[5] Ernest Davis. Mathematics, word problems, common sense, and artificial intelligence, 2023.

[6] Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Sean Welleck, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. Faith and fate: Limits of transformers on compositionality, 2023.

[7] Kaden Griffith and Jugal Kalita. Solving arithmetic word problems with transformers and preprocessing of problem text. *arXiv preprint arXiv:2106.00893*, 2021.

[8] Andrey Gromov. Grokking modular arithmetic, 2023.

[9] Łukasz Kaiser and Ilya Sutskever. Neural gpus learn algorithms. *arXiv preprint arXiv:1511.08228*, 2015.

[10] Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. Grid long short-term memory. *arXiv preprint arxiv:1507.01526*, 2015.

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

[12] Guillaume Lample and François Charton. Deep learning for symbolic mathematics. *arXiv preprint arXiv:1912.01412*, 2019.

[13] Nayoung Lee, Kartik Sreenivasan, Jason D. Lee, Kangwook Lee, and Dimitris Papailiopoulos. Teaching arithmetic to small transformers. *arXiv preprint arXiv:2307.03381*, 2023.

[14] Ziming Liu, Ouail Kitouni, Niklas Nolte, Eric J. Michaud, Max Tegmark, and Mike Williams. Towards understanding grokking: An effective theory of representation learning, 2022.

[15] Yuanliang Meng and Anna Rumshisky. Solving math word problems with double-decoder transformer. *arXiv preprint arXiv:1908.10924*, 2019.

[16] Bhumika Mistry. *An investigation into neural arithmetic logic modules*. PhD thesis, University of Southampton, July 2023.

[17] Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability, 2023.

[18] Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. Investigating the limitations of transformers with simple arithmetic tasks. *arXiv preprint arXiv:2102.13019*, 2021.

[19] Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

[20] Theodoros Palamas. Investigating the ability of neural networks to learn simple modular arithmetic. 2017.

[21] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization Beyond Overfitting on Small Algorithmic Datasets. *arXiv preprint arXiv:2201.02177*, 2022.

[22] David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models, 2019.

[23] Feng Shi, Chonghan Lee, Mohammad Khairul Bashar, Nikhil Shukla, Song-Chun Zhu, and Vijaykrishnan Narayanan. Transformer-based Machine Learning for Fast SAT Solvers and Logic Synthesis. *arXiv preprint arXiv:2107.07116*, 2021.

[24] Kai-Yeung Siu and Vwani Roychowdhury. Optimal depth neural networks for multiplication and related problems. In S. Hanson, J. Cowan, and C. Giles, editors, *Advances in Neural Information Processing Systems*, volume 5. Morgan-Kaufmann, 1992.

[25] Andrew Trask, Felix Hill, Scott Reed, Jack Rae, Chris Dyer, and Phil Blunsom. Neural arithmetic logic units. *arXiv preprint arXiv:1808.00508*, 2018.

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[27] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

[28] Sean Welleck, Peter West, Jize Cao, and Yejin Choi. Symbolic Brittleness in Sequence Models: on Systematic Generalization in Symbolic Mathematics, 2021.

[29] Wojciech Zaremba, Tomas Mikolov, Armand Joulin, and Rob Fergus. Learning simple algorithms from examples, 2015.

[30] Ziqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. The clock and the pizza: Two stories in mechanistic explanation of neural networks, 2023.

[31] Hattie Zhou, Azade Nova, Hugo Larochelle, Aaron Courville, Behnam Neyshabur, and Hanie Sedghi. Teaching algorithmic reasoning via in-context learning. *arXiv preprint arXiv:2211.09066*, 2022.

Appendix

A Rational arithmetic with transformers

In these experiments, transformers are trained to perform five arithmetic operations on positive rational numbers:

- comparison: given four positive integers a, b, c and d, predict whether $\frac{a}{b} < \frac{c}{d}$.
- Integer division: given two integers a and b, predict the integer $\lfloor \frac{a}{b} \rfloor$.
- Addition: given four integers a, b, c and d, predict the sum $\frac{a}{b} + \frac{c}{d}$, in lowest terms.
- Multiplication: given four integers a, b, c and d, predict the product $\frac{a}{b} \times \frac{c}{d}$, in lowest terms.
- Simplification: given two integers a and b, predict the lowest term representation of $\frac{a}{b}$, i.e. $\frac{c}{d}$ with $c = \frac{a}{\text{gcd}(a,b)}$ and $d = \frac{b}{\text{gcd}(a,b)}$.

For the comparison, addition and multiplication tasks, all integers a, b, c and d are uniformly sampled between 1 and M (M=100,000 or 1,000,000).

For the simplification task, 3 integers m, n, p are uniformly sampled between 1 and M, we let $a = \frac{pm}{\text{gcd}(m,n)}$ and $b = \frac{pn}{\text{gcd}(m,n)}$ and the model is tasked to predict a and b.

For the integer division task, 3 integers m, n, p are uniformly sampled between 1 and M, with $m < n$, we let $a = pn + m$ and $b = n$, and the model is tasked to predict $p = \lfloor \frac{a}{b} \rfloor$.

All integers are encoded as sequences of digits in base B (see section 2). Sequence to sequence transformers with 4 layers, 512 dimensions and 8 attention heads are trained to minimize a cross-entropy loss, using Adam with learning rate 10^{-4}, inverse square root scheduling, linear warmup over

10, 000 optimization steps, and a batch size of 256. After each epoch (300,000 examples), models are tested on 100,000 random examples.

Comparison is learned to very high accuracy, and integer division to some extent. On the other hand, the three tasks involving GCD calculations (simplification, addition and multiplication) are not learned (Table A).

	Comparison		Integer division		Simplification		Addition		Multiplication	
Base	$M=10^5$	$M=10^6$	$M=10^5$	$M=10^6$	$M=10^5$	$M=10^6$	$M=10^5$	$M=10^6$	$M=10^5$	$M=10^6$
10	100	100	21.2	2.4	0.14	0.02	0	0	0	0
30	99.9	100	14.2	2.2	0.21	0.02	0	0	0	0
31	99.9	100	14.3	2.4	0.02	0	0	0	0	0
1000	100	99.9	8.8	0.7	0.09	0.01	0	0	0	0

Table 6: **Rational arithmetic with transformers. Accuracy of trained models** Best of 3 models, trained for 1000 to 1500 epochs.

B More experimental settings

Integer encodings. Operands and outcomes are encoded as sequences of digits in base B, preceded by a sign which also serves as a separator (Table 7). In base 10, the input pair $(8, 12)$ is encoded as the sequence `'+ 8 + 1 2'`, and its GCD, 4, as `'+ 4'`. The choice of B is a trade-off. Small bases result in longer sequences that are harder to learn, but use a small vocabulary that is easier to memorize. Composite bases allow for simple tests of divisibility: in base 10, divisibility by 2, 5 and 10 is decided by looking at the rightmost token in the sequence.

Base	Encoded input	Encoded output
2	`[+,1,0,1,0,0,0,0,0,+,1,1,1,1,0,0,0]`	`[+,1,0,1,0,0,0]`
6	`[+,4,2,4,+,3,2,0]`	`[+,1,0,4]`
10	`[+,1,6,0,+,1,2,0]`	`[+,4,0]`
30	`[+,5,10,+,4,0]`	`[+,1,10]`

Table 7: Encoding gcd(160,120) = 40, in base 2, 6, 10 and 30

Sequence-to-sequence transformers with 4 layers, 512 dimensions and 8 attention heads are trained, using Adam [11] with a learning rate of 10^{-5} (no scheduling is needed) on batches of 256 examples. After each epoch (300,000 examples), models are tested on 100,000 held-out examples. The size of the problem space (10^{12} possible input pairs) guarantees minimal duplication between train and test set. All experiments are run on one NVIDIA V100 GPU with 32 GB of memory.

Training examples are generated by uniformly sampling integers between 1 and 10^6 and computing their GCD. All models are tested on two sets. In the *natural test set*, pairs (a, b) are uniformly distributed, and their GCD verify $P(\gcd(a, b) = k) = \frac{6}{\pi^2 k^2}$ [1], i.e. small GCD are more common. In the *stratified test set*, GCD are uniformly distributed between 1 and 100, i.e. there are about 1000 test examples with GCD k, for every $k \leq 100$. The stratified set is generated as follows:

- Sample k, uniformly between 1 and 100.
- Sample a and b, uniformly between 1 and $\frac{M}{k}$, such that $\gcd(a, b) = 1$
- Add (ka, kb) to the stratified test set.

These two test sets provide us with two measures of accuracy. **Model accuracy**, measured on the natural set, is the probability that the GCD of two random integers from 1 to M is correctly predicted. On the stratified test set, it is the **number of GCD correctly predicted** between 1 and 100.

C Model scaling for the base experiments

Section 3 presents results for 4-layer transformers with 512 dimensions and 8 attention heads. In this section, I experiment with very small models (down to 1 layer and 32 dimensions), and very large ones (up to 24 layers and 1024 dimensions). Note: in Tables 8 and 9, the number of trainable parameters are indicated for base 10, they will be larger for larger bases, because larger vocabularies increase the number of parameters in the embedding and decoding layers.

Table 8 presents accuracies for models with one layer, 8 attention heads, and 32 to 512 dimensions. These models have 3 to 100 times less parameters that the 4-layer baseline, but there is no significant change in trained model accuracy for 12 different bases.

Table 9 presents results for models from 6 to 24 layers, symmetric (same number of layers in the encoder and decoder), or asymmetric (using a one-layer encoder or decoder). The dimensions are 512, 640, 768 and 1024 for 6, 8, 12, and 24 layers, and the dimension-to-attention-heads ratio is kept constant at 64 (i.e.there are 8, 10, 12 and 24 attention heads respectively). Again, model size has no significant impact on accuracy.

Overall, these scaling experiments suggest that trained model performance is stable over a wide range of model size (300 thousands to 700 millions parameters). These results are strikingly different from what is commonly observed in Natural Language Processing, where very small transformers (under a few million parameters) cannot learn, and accuracy improves with model size.

Base	512 dimensions 11.6M	256 dim. 4.0M	128 dim. 1.7M	64 dim. 0.6M	32 dim. 0.3M	4-layer baseline 33.7M
2	81.3	81.4	81.4	81.4	81.2	81.6
3	68.8	68.9	68.7	68.8	68.7	68.9
4	81.4	81.4	81.4	81.4	81.4	81.4
5	64.0	63.7	63.8	63.7	63.8	64.0
6	91.3	91.3	91.1	91.1	90.7	91.5
7	62.5	62.4	62.5	62.5	62.5	62.5
10	84.4	84.3	84.3	84.4	84.2	84.7
11	61.7	61.7	61.7	61.9	61.7	61.8
12	91.4	91.4	91.3	91.3	91.1	91.5
15	71.6	71.6	71.5	71.5	71.4	71.7
30	94.6	93.8	93.5	93.7	93.3	94.7
31	61.3	61.3	61.2	61.3	61.3	61.3

Table 8: **Model accuracies for different dimensions and numbers of parameters.** All models have one layer and 8 attention heads. Parameter counts for base 10.

	1/6	6/1	6/6	1/8	8/1	8/8	1/12	12/1	12/12	1/24	24/1	24/24
Base	32.5	27.3	48.3	59.1	48.4	97.1	117.1	94.7	204.8	387.4	313.3	713.8
2	81.3	81.3	81.4	81.5	81.4	81.3	81.3	81.3	81.4	-	81.4	-
3	68.7	68.8	68.7	68.8	68.9	69.0	68.9	68.8	68.8	68.8	68.6	-
4	81.3	81.4	81.4	81.4	81.4	81.6	81.4	81.4	81.4	81.5	81.4	81.3
5	63.8	63.8	63.7	63.8	63.6	63.7	63.7	63.7	63.6	63.9	63.7	63.6
6	91.3	91.1	91.3	91.3	91.4	91.3	91.3	91.0	91.0	91.3	91.0	90.9
7	62.6	62.6	62.4	62.5	62.4	62.6	62.5	62.4	62.4	62.4	62.3	62.2
10	84.3	84.2	84.4	84.7	84.4	84.5	84.4	84.4	83.4	84.5	83.4	83.3
11	61.8	61.7	61.6	61.7	61.8	61.7	62.0	61.6	61.7	61.7	61.6	61.6
12	91.4	91.3	91.3	91.4	91.5	91.4	81.4	91.2	91.2	91.4	91.3	91.2
15	71.5	71.5	71.4	71.5	71.5	71.5	71.4	71.5	71.5	71.5	70.6	71.4
30	94.6	93.4	93.5	94.7	93.6	93.6	94.7	93.6	93.6	93.5	93.4	93.4
31	61.2	61.2	61.3	61.2	61.3	61.2	61.4	61.2	61.3	61.4	61.3	61.1

Table 9: **Model accuracies for different depths and number of parameters (in millions).** 1 and 6 layer models have 512 dimensions and 8 heads, 8-layer have 640 dimensions and 10 heads, 12-layer 768 dimensions and 12 heads, 24-layer models have 1024 dimensions and 16 heads. The largest base 2 and 3 models could not run on one 32GB GPU. All model parameters for base 10.

D Additional results on grokking

GCD	Prediction	GCD	Prediction	GCD	Prediction	GCD	Prediction	GCD	Prediction
1	1	11	1	21	3	31	1	41	1
2	2	12	12	22	2	32	16/ 32	42	6
3	3	13	1	23	1	33	3	43	1
4	4	14	2	24	24	34	2	44	4
5	5	15	15	25	25	35	5	45	15
6	6	16	16	26	2	36	12	46	2
7	1	17	1	27	3	37	1	47	1
8	8	18	6	28	4	38	2	48	48
9	3	19	1	29	1	39	3	49	1
10	10	20	20	30	30	40	40	50	50

Table 10: **Model predictions.** $B = 1000$, after 220 epochs. 32 is being learned.

Base	GCD predicted	Divisors predicted	Non-divisors (epoch learned)
$625 = 5^4$	6	{1,5,25}	2 (634)
2017	4	{1}	2 (142), 3 (392)
$2021 = 43.47$	10	{1,43}, {1,47}	2 (125), 3 (228)
$2023 = 7.17^2$	16	{1,7}, {1,17}	3 (101), 2 (205), 4 (599)
$2025 = 3^4.5^2$	28	{1,3, 9, 27, 81}, {1,5,25}	2 (217), 4 (493), 8 (832)
$2187 = 3^7$	20	{1,3,9,27,81}	2 (86), 4 (315) , 5 (650)
$2197 = 13^3$	11	{1,13}	2 (62), 3 (170), 4 (799)
$2209 = 47^2$	8	{1,47}	2 (111), 3 (260), 9 (937)
$2401 = 7^4$	10	{1,7,49}	2 (39), 3 (346)
$2401 = 7^4$	14	{1,7,49}	3 (117), 2 (399), 4 (642)
$2744 = 2^3.7^3$	30	{1,2,4,8,16,32}, {1,7,49}	3 (543), 5 (1315)
$3125 = 5^5$	16	{1,5,25}	2 (46), 3 (130), 4 (556)
$3375 = 3^3.5^3$	23	{1,3,9,27}, {1,5,25}	2 (236), 4 (319)
$4000 = 2^5.5^3$	24	{1,2, 4,8,16,32}, {1, 5, 25 }	3 (599)
$4913 = 17^3$	17	{1,17}	2 (54), 3 (138), 4 (648), 5 (873)
$5000 = 2^3.5^4$	28	{1,2,4,8,16,32}, {1,5,25}	3 (205), 9 (886)
$10000 = 2^4.5^4$	22	{1,2,4,8,16}, {1,5,25}	3 (211)

Table 11: **Predicted gcd, divisors and non-divisors of B.** Best model of 3. For non-divisors, the epoch learned is the first epoch where model achieves 90% accuracy for this GCD.

E Detailed model predictions

Table 12: Predicted values for gcd 1 to 63.

Base	2		4		10		30		31		420	
GCD	Prediction	%	Pred.	%	Pred.	%	Pred.	%	Pred.	%	Pred.	%
1	**1**	100	**1**	100	**1**	100	**1**	100	**1**	100	**1**	100
2	**2**	100	**2**	100	**2**	100	**2**	100	1	100	**2**	100
3	1	100	1	100	1	100	**3**	100	1	100	**3**	100
4	**4**	100	**4**	100	**4**	100	**4**	100	1	100	**4**	100
5	1	100	1	100	**5**	100	**5**	100	1	100	**5**	100
6	2	100	2	100	2	100	**6**	100	1	100	**6**	99.6
7	1	100	1	100	1	100	1	100	1	100	**7**	100
8	**8**	100	**8**	100	**8**	100	**8**	100	1	100	**8**	100
9	1	100	1	100	1	100	**9**	100	1	100	**9**	100
10	2	100	2	100	10	100	10	100	1	100	10	100
11	1	100	1	100	1	100	1	100	1	100	1	100
12	4	100	4	100	4	100	12	100	1	100	12	99.8
13	1	100	1	100	1	100	1	100	1	100	1	100
14	2	100	2	100	2	100	2	100	1	100	14	100
15	1	100	1	100	5	100	15	100	1	100	15	99.4
16	16	100	16	100	16	99.7	8	100	1	100	16	100
17	1	100	1	100	1	100	1	100	1	100	1	100
18	2	100	2	100	2	100	18	100	1	100	18	100
19	1	100	1	100	1	100	1	100	1	100	1	100
20	4	100	4	100	20	100	20	100	1	100	20	100
21	1	100	1	100	1	100	3	100	1	100	21	100
22	2	100	2	100	2	100	2	100	1	100	2	100
23	1	100	1	100	1	100	1	100	1	100	1	100
24	8	100	8	100	8	100	24	100	1	100	24	100
25	1	100	1	100	25	100	25	99	1	100	25	99.9
26	2	100	2	100	2	100	2	100	1	100	2	100
27	1	100	1	100	1	100	9	100	1	100	9	100
28	4	100	4	100	4	100	4	100	1	100	28	100
29	1	100	1	100	1	100	1	100	1	100	1	100
30	2	100	2	100	10	100	30	100	1	100	30	99.6
31	1	100	1	100	1	100	1	100	31	100	1	100
32	32	99.9	32	98.7	16	99.9	8	100	1	100	16	100
33	1	100	1	100	1	100	3	100	1	100	3	100
34	2	100	2	100	2	100	2	100	1	100	2	100
35	1	100	1	100	5	100	5	100	1	100	35	100
36	4	100	4	100	4	100	36	100	1	100	36	100
37	1	100	1	100	1	100	1	100	1	100	1	100
38	2	100	2	100	2	100	2	100	1	100	2	100
39	1	100	1	100	1	100	3	100	1	100	3	99.9
40	8	99.9	8	100	40	99.9	40	100	1	100	40	99.9
41	1	100	1	100	1	100	1	100	1	100	1	100
42	2	100	2	100	2	100	6	99.9	1	100	42	100
43	1	100	1	100	1	100	1	100	1	100	1	100
44	4	100	4	100	4	100	4	100	1	100	4	100
45	1	100	1	100	5	100	45	100	1	100	45	99.8
46	2	100	2	100	2	100	2	100	1	100	2	100
47	1	100	1	100	1	100	1	100	1	100	1	100
48	16	100	16	100	16	99.9	24	100	1	100	48	99.9
49	1	100	1	100	1	100	1	100	1	100	7	100
50	2	100	2	100	50	100	50	100	1	100	50	99.6
51	1	100	1	100	1	100	3	100	1	100	3	99.8
52	4	100	4	100	4	100	4	100	1	100	4	100
53	1	100	1	100	1	100	1	100	1	100	1	100
54	2	100	2	100	2	100	18	99.9	1	100	18	100
55	1	100	1	100	5	100	5	100	1	100	5	100
56	8	100	8	100	8	99.9	8	100	1	100	56	100
57	1	100	1	100	1	100	3	100	1	100	3	99.9
58	2	100	2	100	2	100	2	100	1	100	2	100
59	1	100	1	100	1	100	1	100	1	100	1	100
60	4	100	4	100	20	100	60	100	1	100	60	99.7
61	1	100	1	100	1	100	1	100	1	100	1	100
62	2	100	2	100	2	100	2	100	31	100	2	100
63	1	100	1	100	1	100	9	100	1	100	63	100

Table 13: Predicted values for gcd 64 to 100.

Base	2		4		10		30		31		420	
GCD	Prediction	%	Pred.	%	Pred.	%	Pred.	%	Pred.	%	Pred.	%
64	64	98.9	64	99.2	16	99.8	8	100	1	100	16	100
65	1	100	1	100	5	100	5	100	1	100	5	100
66	2	100	2	100	2	100	6	100	1	100	6	100
67	1	100	1	100	1	100	1	100	1	100	1	100
68	4	100	4	100	4	100	4	100	1	100	4	100
69	1	100	1	100	1	100	3	100	1	100	3	100
70	2	100	2	100	10	100	10	100	1	100	70	100
71	1	100	1	100	1	100	1	100	1	100	1	100
72	8	100	8	100	8	100	72	100	1	100	72	100
73	1	100	1	100	1	100	1	100	1	100	1	100
74	2	100	2	100	2	100	2	100	1	100	2	100
75	1	100	1	100	25	100	75	100	1	100	75	99.4
76	4	100	4	100	4	100	4	100	1	100	4	100
77	1	100	1	100	1	100	1	100	1	100	7	100
78	2	100	2	100	2	100	6	100	1	100	6	100
79	1	100	1	100	1	100	1	100	1	100	1	100
80	16	100	16	100	80	99.9	40	100	1	100	80	100
81	1	100	1	100	1	100	9	100	1	100	9	99.8
82	2	100	2	100	2	100	2	100	1	100	2	100
83	1	100	1	100	1	100	1	100	1	100	1	100
84	4	100	4	100	4	100	12	100	1	100	84	100
85	1	100	1	100	5	100	5	100	1	100	5	100
86	2	100	2	100	2	100	2	100	1	100	2	100
87	1	100	1	100	1	100	3	100	1	100	3	99.8
88	8	100	8	100	8	100	8	100	1	100	8	100
89	1	100	1	100	1	100	1	100	1	100	1	100
90	2	100	2	100	10	100	90	100	1	100	90	99.9
91	1	100	1	100	1	100	1	100	1	100	7	100
92	4	99.9	4	100	4	100	4	100	1	100	4	100
93	1	100	1	100	1	100	3	100	31	99.9	3	99.8
94	2	100	2	100	2	100	2	100	1	100	2	100
95	1	100	1	100	5	100	5	100	1	100	5	100
96	32	100	32	99.5	16	99.8	24	100	1	100	48	99.9
97	1	100	1	100	1	100	1	100	1	100	1	100
98	2	100	2	100	2	100	2	100	1	100	14	100
99	1	100	1	100	1	100	9	100	1	100	9	99.8
100	4	100	4	100	100	100	100	100	1	100	100	99.6