AI from generated data

François CHARTON, Meta AI

Mathematics as a translation task

- Train models to translate problems, encoded as sentences in some language, into their solutions
  - Numeric to numeric 7+9 => 16
  - Symbolic to numeric x^2-x-1 => $\frac{1+\sqrt{5}}{2}, \frac{1-\sqrt{5}}{2}$
  - Symbolic to symbolic x^2-x-1 => 2x – 1
  - Numeric to symbolic 1,1,2,3,5,8... => $u_n = u_{n-1}+u_{n-2}$

Maths as translation: learning GCD

- Generate pairs of integers (a,b), compute their GCD: e.g. a=10, b=32, gcd(a,b)=2
  - Repeat to build a large supervised training set
- Integers can be represented as sequences of digits in base 10
  - `10: ‘+’, ‘1’, ‘0’`
  - `32: ‘+’, ‘3’, ‘2’`
  - `2: ‘+’, ‘2’`
- Train a model to translate `‘+’,‘1’,‘0’,‘+’,‘3’,‘2’` into ‘+’, ‘2’
  - From the generated examples only
  - As a “pure language” problem: the model knows no maths

This works!

- Symbolic integration / Solving ODE:
  - Deep learning for symbolic mathematics (2020): Lample & Charton (ArXiv 1912.01412) https://arxiv.org/abs/1912.01412
- Dynamical systems:
  - Learning advanced computations from examples (2021) : Charton, Hayat & Lample (ArXiv 2006.06462)
  - Discovering Lyapunov functions with transformers (2023) : Alfarano, Charton, Hayat (3rd MATH&AI workshop, NeurIPS)
- Symbolic regression:
  - Deep symbolic regression for recurrent sequences (2022) : d'Ascoli, Kamienny, Lample, Charton (ArXiv 2201.04600)
  - End-to-end symbolic regression with transformers (2022) : Kamienny, d'Ascoli, Lample, Charton (ArXiv 2204.10532)
- Cryptanalysis of post-quantum cryptography:
  - SALSA: attacking lattice cryptography with transformers (2022): Wenger, Chen, Charton, Lauter (ArXiv 2207.04785)
  - SALSA PICANTE (2023) Li, Sotakova, Wenger, Mahlou, Garcelon, Charton, Lauter (ArXiv 2303.0478)
  - SALSA VERDE (2023) Li, Wenger, Zhu, Charton, Lauter (ArXiv 2306.11641)
- Theoretical physics
  - Transformers for scattering amplitudes (2023): Merz, Cai, Charton, Nolte, Wilhelm, Cranmer, Dixon (ML4PS Workshop, NeurIPS)
- Quantum computing
  - Using transformer to simplify ZX diagrams (2023) (3rd MATH&AI Workshop, NeurIPS)

Mathematics as a translation task

Three steps

I. Generate large sets of problems and solutions
II. Represent problems and solutions as sequences
III. Train transformers to translate problems into solutions

Mathematics as a translation task

- Models trained from generated data allow for datasets as large as one wishes
- What is the impact of data generation on learning?
  - An important question for LLM for physics

Deep learning for symbolic mathematics

(Lample, Charton 2019)

- Undergrad maths: compute symbolic integrals

$$\frac{\cos(2x)}{\sin(x)} \longrightarrow \frac{\log(\cos(x)-1)}{2} - \frac{\log(\cos(x)+1)}{2} + 2\cos(x)$$

- Generate data: pairs (f, F) of functions and their integral
- Encode as sequences of symbols
- Train a transformer to predict F from f

Two methods for data generation

- Forward
  - Generate a random function f
  - Compute its integral F
  - Only for problems you know how to solve
  - Slow

- Backward
  - Generate a random function F
  - Compute its derivative f

Training models

- 6-layer encoder-decoder transformers with 256 dimensions and 8 attention heads
- The model is trained on generated data (20-40M examples)
  - Supervised learning, minimizing cross-entropy
  - A pure language task: the model has no understanding of maths
- Tested on held-out data (i.e. not seen during training)
- Solutions are verified with an external tool (SymPy)
  - Using problem-related mathematical metrics

In-domain results

- Performance on held-out test sets with the same distribution as training
- Almost 100% no matter the generation procedure
- Outperforms best computer algebras

	Integration (FWD)	Integration (BWD)
Beam size 1	93.6	98.4
Beam size 10	95.6	99.4
Beam size 50	96.2	99.7

	Integration (BWD)
Mathematica (30s)	84.0
Matlab	65.2
Maple	67.4

Out-of-distribution results

- Models trained on FWD do not generalize to BWD

Training data	Forward (FWD)			Backward (BWD)		
	Beam 1	Beam 10	Beam 50	Beam 1	Beam 10	Beam 50
FWD	93.6	95.6	96.2	10.9	13.9	17.2
BWD	18.9	24.6	27.5	98.4	99.4	99.7

- Why?

Generating data

Functions and their primitives generated with the forward approach (FWD)	
$\cos^{-1}(x)$	$x\cos^{-1}(x) - \sqrt{1-x^2}$
$x\left(2x + \cos\left(2x\right)\right)$	$\frac{2x^3}{3} + \frac{x\sin\left(2x\right)}{2} + \frac{\cos\left(2x\right)}{4}$
$\frac{x\left(x+4\right)}{x+2}$	$\frac{x^2}{2} + 2x - 4\log\left(x+2\right)$
$\frac{\cos\left(2x\right)}{\sin\left(x\right)}$	$\frac{\log\left(\cos\left(x\right)-1\right)}{2} - \frac{\log\left(\cos\left(x\right)+1\right)}{2} + 2\cos\left(x\right)$
$3x^2\sinh^{-1}\left(2x\right)$	$x^3\sinh^{-1}\left(2x\right) - \frac{x^2\sqrt{4x^2+1}}{6} + \frac{\sqrt{4x^2+1}}{12}$
$x^3\log\left(x^2\right)^4$	$\frac{x^4\log\left(x^2\right)^4}{4} - \frac{x^4\log\left(x^2\right)^3}{2} + \frac{3x^4\log\left(x^2\right)^2}{4} - \frac{3x^4\log\left(x^2\right)}{4} + \frac{3x^4}{8}$

Generating data

Functions and their primitives generated with the backward approach (BWD)	
$\cos(x) + \tan^2(x) + 2$	$x + \sin(x) + \tan(x)$
$\frac{1}{x^2\sqrt{x-1}\sqrt{x+1}}$	$\frac{\sqrt{x-1}\sqrt{x+1}}{x}$
$\left(\frac{2x}{\cos^2(x)} + \tan(x)\right)\tan(x)$	$x\tan^2(x)$
$\frac{x\tan\left(\frac{e^x}{x}\right) + \frac{(x-1)e^x}{\cos^2\left(\frac{e^x}{x}\right)}}{x}$	$x\tan\left(\frac{e^x}{x}\right)$
$1 + \frac{1}{\log(\log(x))} - \frac{1}{\log(x)\log(\log(x))^2}$	$x + \frac{x}{\log(\log(x))}$
$-2x^2\sin(x^2)\tan(x) + x(\tan^2(x)+1)\cos(x^2) + \cos(x^2)\tan(x)$	$x\cos(x^2)\tan(x)$

A third data set: integration by parts

- Integration by part
  - Generate random functions F and G
  - Compute their derivative f and g
  - If fG is in the dataset, we get Fg for free using $\int Fg = FG - \int fG$

- Derived from the backward model
- Problem and solution length are more balanced

Generating data

Functions and their primitives generated with the integration by parts approach (IBP)	
$x\left(x+\log(x)\right)$	$\frac{x^2\left(4x+6\log(x)-3\right)}{12}$
$\frac{x}{(x+3)^2}$	$\frac{-x+(x+3)\log(x+3)}{x+3}$
$\frac{x+\sqrt{2}}{\cos^2(x)}$	$\left(x+\sqrt{2}\right)\tan(x)+\log(\cos(x))$
$x(2x+5)(3x+2\log(x)+1)$	$\frac{x^2\left(27x^2+24x\log(x)+94x+90\log(x)\right)}{18}$
$\frac{\left(x-\frac{2x}{\sin^2(x)}+\frac{1}{\tan(x)}\right)\log(x)}{\sin(x)}$	$\frac{x\log(x)+\tan(x)}{\sin(x)\tan(x)}$
$x^3\sinh(x)$	$x^3\cosh(x)-3x^2\sinh(x)+6x\cosh(x)-6\sinh(x)$

More out-of-distribution results

Training data	Forward (FWD)			Backward (BWD)			Integration by parts (IBP)		
	Beam 1	Beam 10	Beam 50	Beam 1	Beam 10	Beam 50	Beam 1	Beam 10	Beam 50
FWD	93.6	95.6	96.2	10.9	13.9	17.2	85.6	86.8	88.9
BWD	18.9	24.6	27.5	98.4	99.4	99.7	42.9	54.6	59.2
BWD + IBP	41.6	54.9	56.1	98.2	99.4	99.7	96.8	99.2	99.5

- Models trained on FWD do generalize to IPB
- OOD generalization is possible, when test distributions are not too far away from training distributions

- Training distribution matters
- Out-of-distribution generalization is possible so long test distribution is not 'too far'

Linear algebra with transformers
(Charton 2021)

- Basic linear algebra is learned, with small models
  - Transposition: 100% accuracy, up to 30x30 matrices, with 1-layer transformers
  - Addition: 99% accuracy, up to 20x20 matrices, 2-layer transformers
  - Matrix-vector product: 100% accuracy, up to 10x10 matrices, 2-layer transformers
  - Multiplication: 100% accuracy, 5x5 matrices, 1 / 4 layer transformers

- Advanced tasks can also be learned
  - Eigenvalues: 100% accuracy for 5x5 to 20x20 matrices
  - Eigen decomposition: 97% for 5x5, 82% for 6x6 matrices
  - SVD decomposition: 99% accuracy for 4x4 matrices
  - Matrix inversion: 90% for 5x5 matrices

Computing eigenvalues – the importance of training distributions

- Models predict the eigenvalues of 5x5 symmetric matrices with 100% accuracy
- Training set symmetric matrices with independent coefficients: Wigner matrices
  - Eigenvalues are distributed as a semi-circle
  - Bounded support, symmetric around 0
  - Variance depends on variance of coefficients, and dimension matrix

- Are we learning eigenvalues, or eigenvalues of Wigner matrices?

Tweaking the training distribution

- Wigner matrices (symmetric matrices with independent identically distributed entries) can be decomposed as $M=HDH^T$, with
  - D diagonal, with entries distributed as a semi circle
  - H orthogonal
  - If the coefficients of M are Gaussian, the directions of columns of H are uniformly distributed over the unit sphere

- To generate matrices with different distributions of eigenvalues
  - Generate M with Gaussian independent entries
  - Compute $M=HDH^T$
  - Replace D with D', from a different distribution
  - Recompute $M' = HD'H^T$

Tweaking the training distribution

- We generate 7 distributions
- 4 with positive and negative eigenvalues
  - Semi-circle: the original Wigner matrices
  - Uniform eigenvalues
  - Laplace distributed eigenvalues
  - Gaussian eigenvalues
- 3 with positive eigenvalues only
  - Absolute values of semicircle eigenvalues
  - Absolute values of Laplace eigenvalues
  - Marcenko-Pastur distribution (i.e. random covariance matrices)

Eigenvalues – out-of-distribution generalization

	Semi-circle	Uniform	Gaussian	Laplace	abs-sc	abs-Lapl	Marchenko
Semi-circle	100	34	36	39	1	5	0
Uniform	93	100	76	70	92	70	2
Gaussian	100	100	100	100	100	100	99
Laplace	100	100	100	100	100	100	100
Abs-semicircle	0	5	4	4	100	78	20
Abs-Laplace	0	4	5	5	100	100	100
Marchenko-Pastur	0	4	4	4	100	76	100

Table 1: **Out-of-distribution generalization. Eigenvalues of 5x5 matrices**. Rows are the training distributions, columns the test distributions.

- Gauss and Laplace generalize to Wigner (but not the other way around)
- Can generalize far away from training distribution: to positive definite matrices

Eigenvalues – out-of-distribution generalization

- Robust distributions learn faster

	Semi-circle	Uniform	Gaussian	Laplace	abs-sc	abs-Lapl	Marchenko
8x8 matrices							
Semicircle	0	0	0	0	0	0	0
Uniform	91	100	65	57	89	55	0
Gaussian	100	100	100	99	100	99	41
Laplace	100	100	100	100	100	100	97
Abs-semicircle	0	1	1	0	100	53	0
Abs-Laplace	0	1	1	1	100	100	98
Marchenko-Pastur	0	0	0	0	1	1	20
10x10 matrices							
Gaussian (12/1 layers)	100	100	100	98	100	97	3
Laplace (8/1 layers)	100	100	100	100	100	100	74

Table 2: **Out-of-distribution generalization. Eigenvalues of 8x8 and 10x10 matrices, accuracy after 36 million examples.** Rows are the training distributions, columns the test distributions.

Take aways

- Out-of-distribution generalization is possible
- Special ”robust” distributions exist
  - Allow for faster learning

Can transformers learn the greatest common divisor?
(Charton 2023)

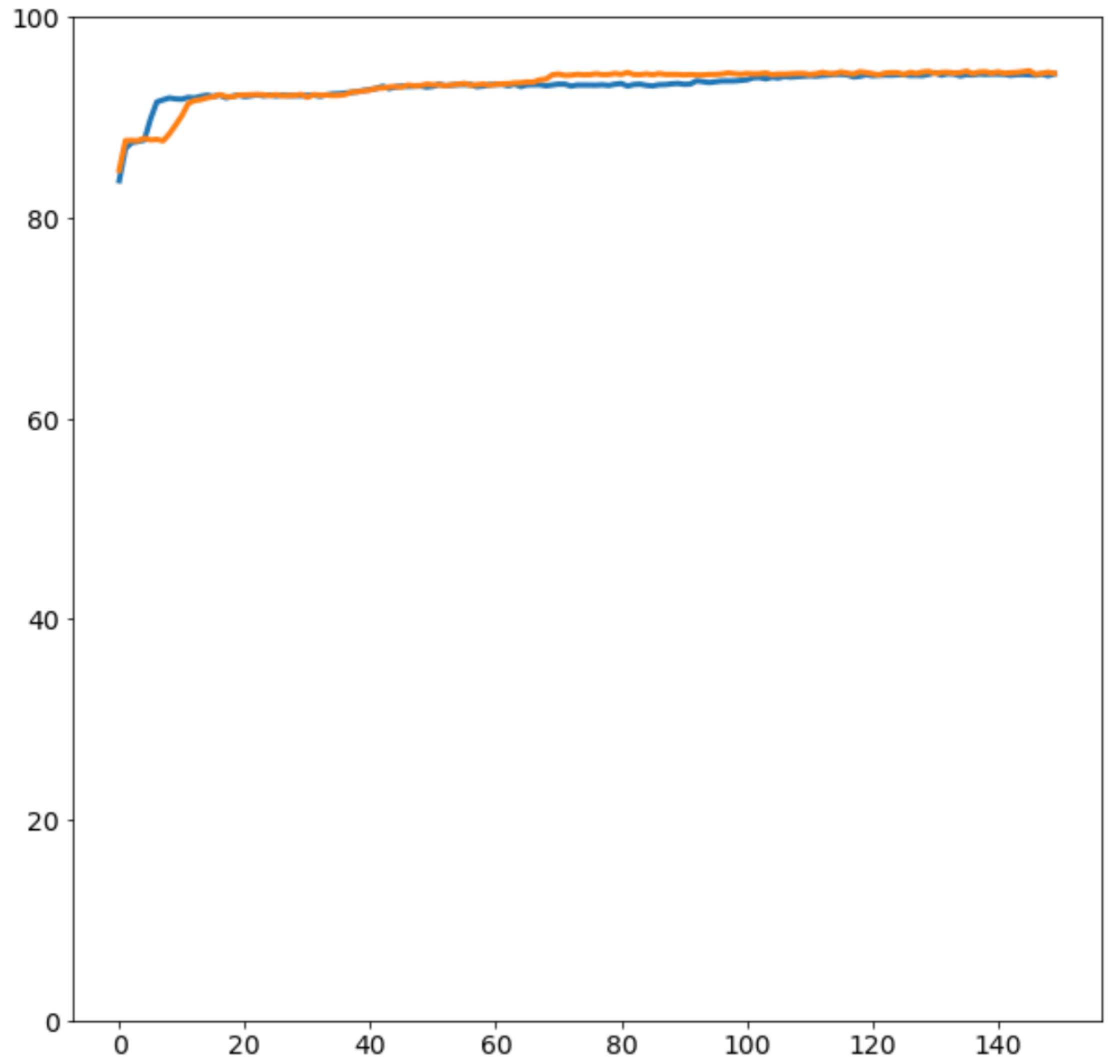
- Generate random pairs of integers between 1 and 1,000,000
- Compute their gcd, train a model to predict it
- Test on a held-out dataset (100k examples)

- Problem space size: 10^{12} , no chance that the model memorizes all the cases
- Operands are uniformly distributed

Can transformers learn the greatest common divisor?

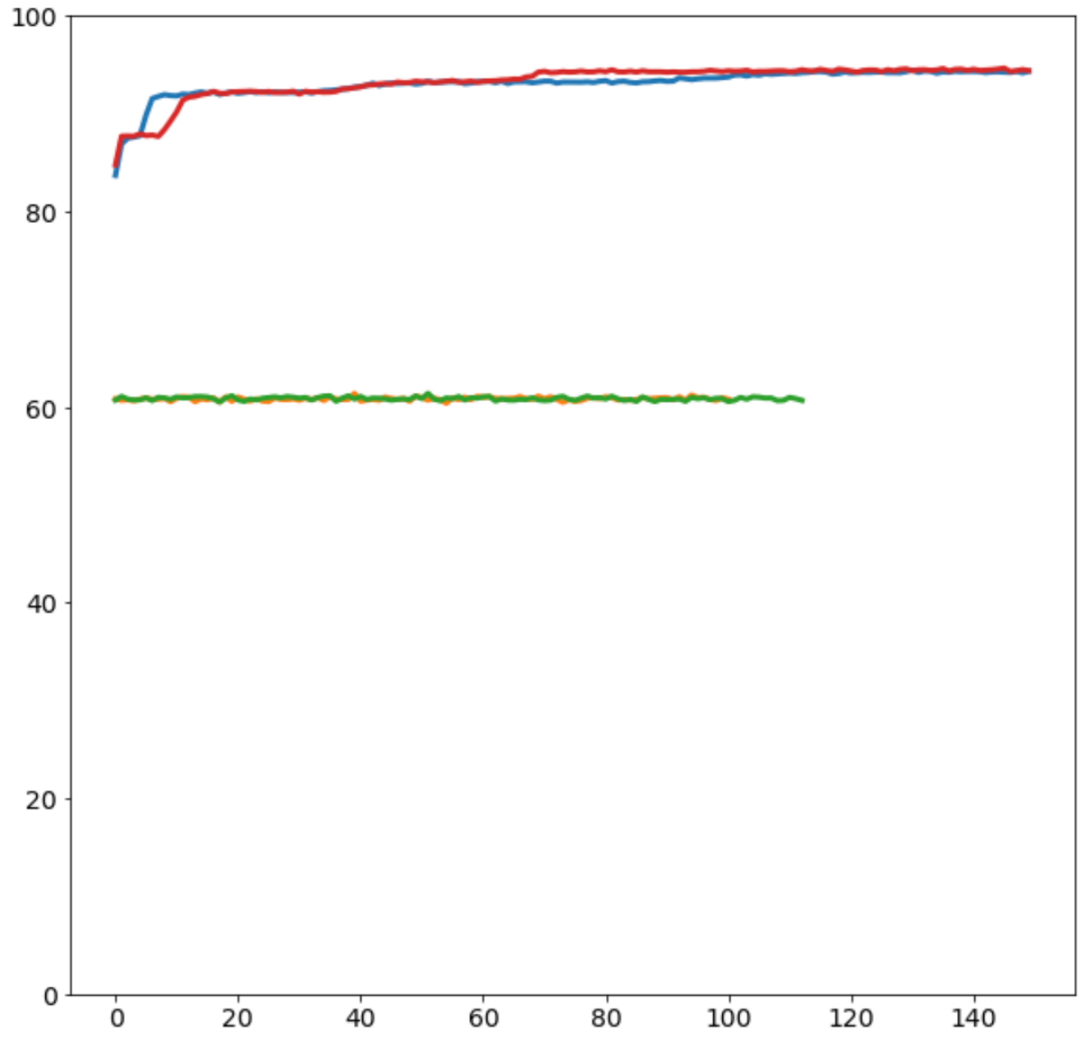
- Encoding input/output in base 30
- 1-layer transformers, 64 dimensions
- 85% accuracy after one epoch (300k examples)
- 94.6% accuracy after 150 epochs (45M examples)

- Surely, the maths are learned



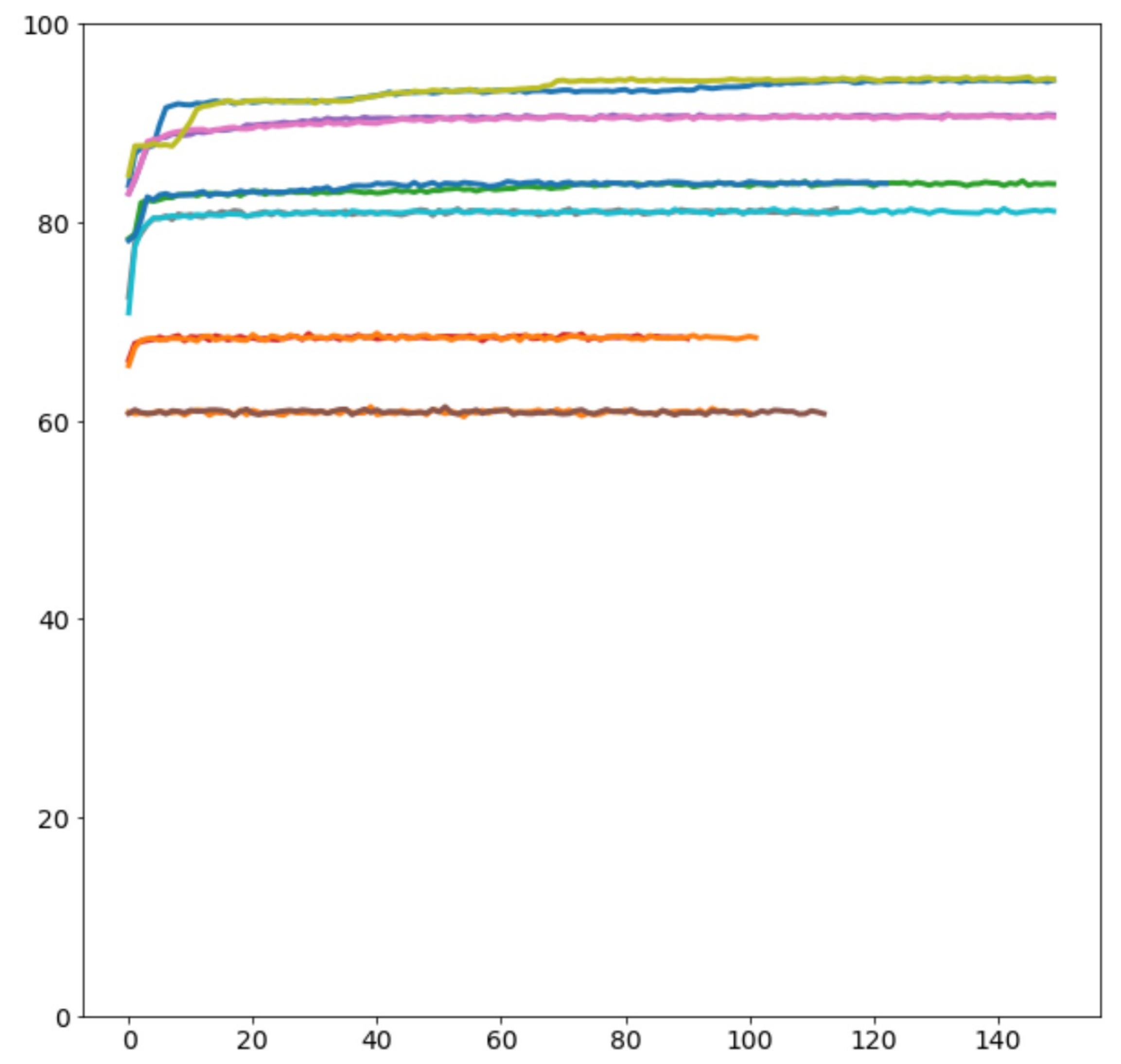
Can transformers learn the greatest common divisor?

- Encoding input/output in base 31
- Accuracy plateaus around 61%
- Accuracy seems base-dependent



Learning the greatest common divisor

- Top to bottom, bases 30, 6, 10, 2, 3, 31…
- Performance depends on the base we use to represent numbers
- Are we really learning the maths?



Looking at model predictions

Table 3: Model predictions and their frequencies, for GCD 1 to 36. Correct predictions in bold face.

	Base 2		Base 10			Base 2		Base 10			Base 2		Base 10	
GCD	Pred	%	Pred	%	GCD	Pred	%	Pred	%	GCD	Pred	%	Pred	%
1	**1**	100	**1**	100	13	1	100	1	100	25	1	100	**25**	100
2	**2**	100	**2**	100	14	2	100	2	100	26	2	100	2	100
3	1	100	1	100	15	1	100	5	100	27	1	100	1	100
4	**4**	100	**4**	100	16	**16**	100	**16**	99.7	28	4	100	4	100
5	1	100	**5**	100	17	1	100	1	100	29	1	100	1	100
6	2	100	2	100	18	2	100	2	100	30	2	100	10	100
7	1	100	1	100	19	1	100	1	100	31	1	100	1	100
8	**8**	100	**8**	100	20	4	100	**20**	100	32	**32**	99.9	16	99.9
9	1	100	1	100	21	1	100	1	100	33	1	100	1	100
10	2	100	**10**	100	22	2	100	2	100	34	2	100	2	100
11	1	100	1	100	23	1	100	1	100	35	1	100	5	100
12	4	100	4	100	24	8	100	8	100	36	4	100	4	100

Learning the greatest common divisor???

- In base 2, gcd 1,2,4,8, 16… are correctly predicted
  - The model counts the rightmost zeroes
    - 11100 (28) and 1110 (14) have gcd 2
    - 111100 (60) and 111000 (56) have gcd 4
- In composite bases, the model learns multiples of divisors of the base
  - In base 10 multiples of 20 end with 00, 20, 40, 60 or 80

The three rules

(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.

So far disappointing

Accuracy may be high but only a few GCD are learned

Table 2: Number of correct GCD under 100 and accuracy. Best of 6 experiments.

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

Large bases and grokking

- Base 2023 = 7.17.17
- After 10 epochs: 1,7, and 17 are learned, accuracy 63%, 3 GCD
- At epoch 101, 3 is learned, together with 21 (3.7) and 51 (3.17)
- At epoch 200, 2 is learned (and 6, 14, 34, 42): 11 GCD
- At epoch 600, 4 is learned: 16 GCD, 93% accuracy

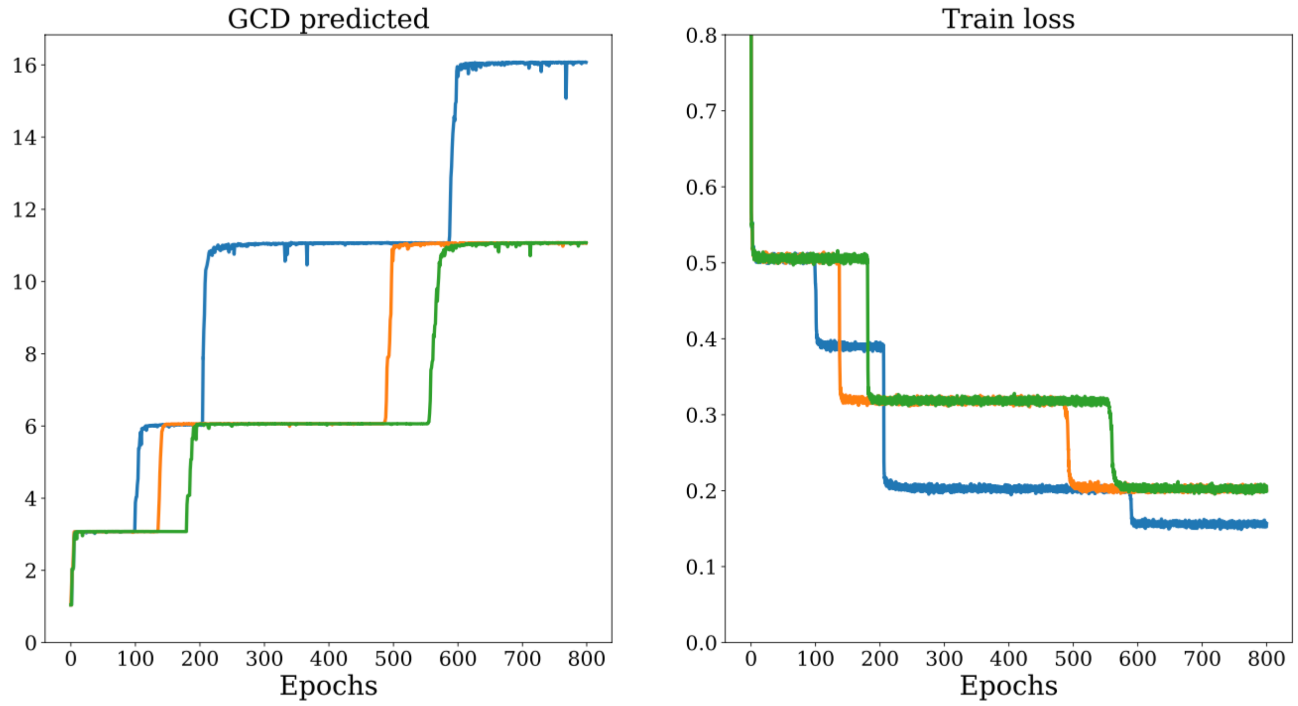


Figure 5: **Learning curves for base B=2023.** 3 different model initializations.

Large bases and grokking

This phenomenon is related to grokking, first described by Power. [22] for modular arithmetic. Table 5 presents model predictions for base 1000, which continue to respect rules R1 and R3. In fact, we can update the three rules into **the three rules with grokking**:

(G1) **Prediction is deterministic.** All pairs with the same GCD are predicted the same, as $f(k)$.
(G2) **Correct predictions are products of primes divisors of B, and small primes.** Small primes are learned roughly in order, as grokking sets in.
(G3) **f(k) is the largest correct prediction that divides k.**

Still only a few GCD are 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 6: **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.

Engineering the training distribution

- Training sets have uniformly distributed operands
  - 90% of them are over 100 000
  - Small GCD, e.g. gcd(6,9) are never seen
- This is not how we are taught / teach arithmetic
  - From easy cases that we sometimes learn by rote
  - Generalizing to harder cases once easy cases are mastered
- Curriculum learning has draw backs: the distribution changes over time
  - Learn the easy cases, but then forget them

Engineering the training distribution

- Log-uniform operands
  - k appears with probability 1/k
  - As many 1-digit numbers as 6-digit
- No impact on the outcome distribution ($1/k^2$)
- No impact on the test sets

- Learning is noisier, but more GCD are learned

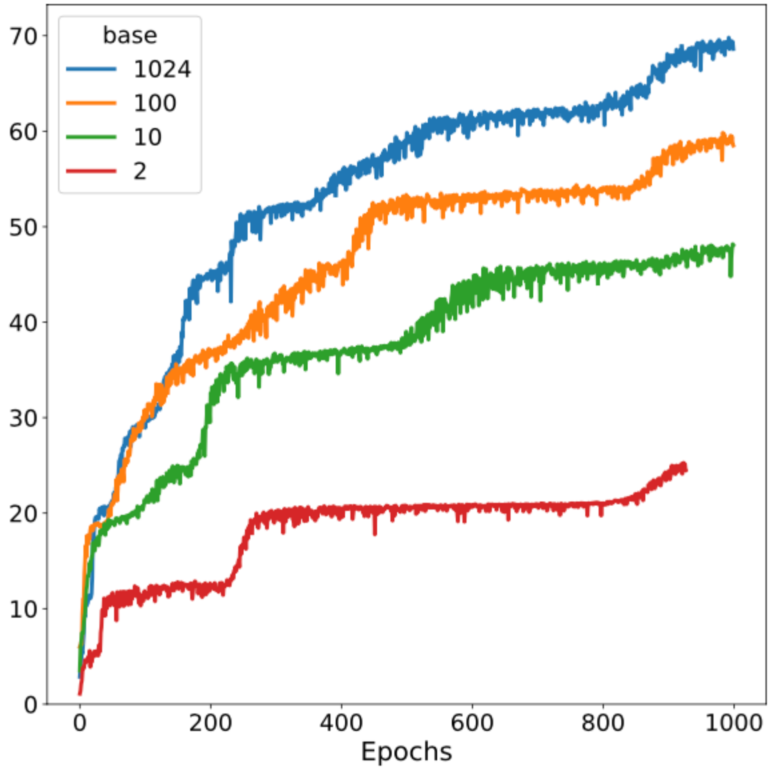


Figure 3: **Learning curves, Log-uniform training set.**

Engineering the training distribution

- Log-uniform operands, fast grokking
- All primes up to 23

Table 6: **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.

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

Learning large primes, the outcome distribution

- GCD are distributed in $1/k^2$, very few examples with large primes
- A log-uniform distribution of operands and outcomes
  - All primes up to 53

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

Take aways

- Predictions can be deterministic and explainable
- The model learns a sieve:
  - It classifies input pairs (a,b) into clusters with common divisors
  - And predicts the smallest common divisor in the class (when outcomes are not uniformly distributed)
- Training distribution impact accuracy, no matter the test distribution

Conclusions

- Transformers can learn mathematics
  - A new field for research
  - With applications to science
- Training distributions matter
  - Some training distributions allow for faster learning and better generalization