Transformers for bootstrapped amplitudes

François CHARTON, Meta AI

Maths as a translation task

- Train models to translate problems, encoded as sentences in some language, into their solutions
  - 7+9 => 16
  - x^2-x-1 => $\frac{1+\sqrt{5}}{2}, \frac{1-\sqrt{5}}{2}$

The recipe

- Generate a lot of examples of problems and solutions
- Encode them as ”sentences” in some language
- Train a transformer model from problems and solutions
  - By minimizing the correctness (X-entropy) of the solution predicted by the model
  - No maths are involved at this stage
- Test it on a held-out test set
  - Not seen during training
  - Using a mathematical criterion

Maths as translation: learning GCD

- Two integers a=10, b=32, and their GCD gcd(a,b)=2
- Can be encoded as sequences of digits (in base 10):
  - `‘+’, ‘1’, ‘0’`
  - `‘+’, ‘3’, ‘2’`
  - `‘+’, ‘2’`
- Translate `‘+’,‘1’,‘0’,‘+’,‘3’,‘2’` into ‘+’, ‘2’
  - from 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)
- 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)

Deep symbolic regression for recurrent sequences (d'Ascoli, Kamienny, Lample, Charton 2022)

- Given the sequence 1, 2, 4, 7, 11, 16, what is the next term?
- 2 approaches:
  - Numeric regression : direct prediction of the next term
  - Symbolic regression : finding a formula for the sequence
    - a closed formula: $u_n = n(n+1)/2 + 1$
    - or a recurrence relation: $u_n = u_{n-1} + n$

Deep symbolic regression for recurrent sequences (d'Ascoli, Kamienny, Lample, Charton 2022)

- 2 tasks:
  - Numeric regression : from the p first terms, predict the q next
  - Symbolic regression : from the p first terms, find a function
- 2 settings:
  - Integer sequences
  - Real (floating point) sequences
- One evaluation criterion: how good is the model at predicting the next q terms?

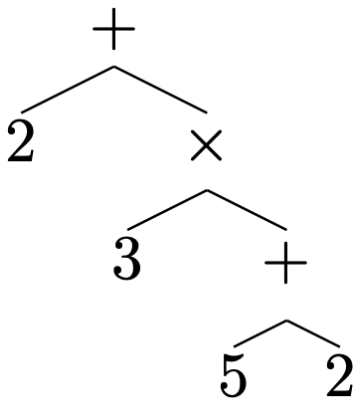
Generating data

- Generate a random function $f(n, u_{n-1}, \ldots u_{n-k})$: $n + u_{n-1}$
- Sample k initial points $u_0, u_1, \ldots u_{k-1}$: $u_0=1$
- Use function f to compute the next terms of the sequence
  - 1, 2, 4, 7, 11, 16, 22, 29, 37 ...

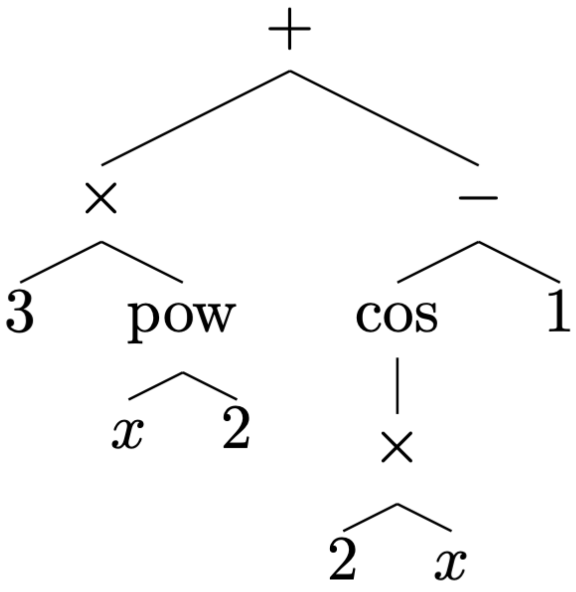
- Symbolic regression: predict f from $(u_0,\ldots u_{p-1})$
  - from (1,2,4,7,11) predict $f(n) = n+u_{n-1}$
- Numeric regression: predict $(u_p,\ldots u_{p+q-1})$ from $(u_0,\ldots u_{p-1})$
  - from (1,2,4,7,11) predict (16,22,29,37)

Representing expressions

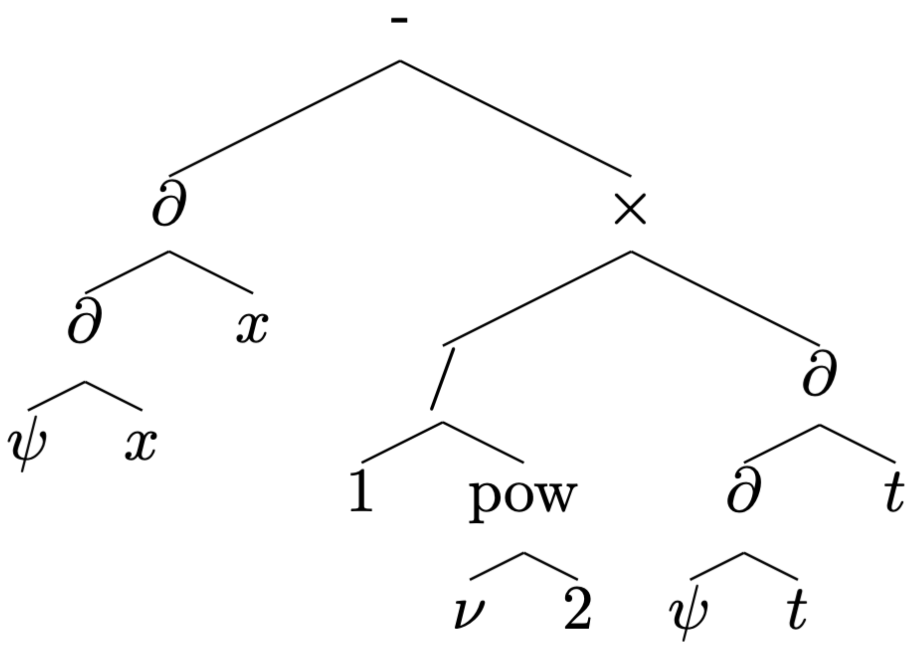
$$2 + 3 \times (5 + 2)$$



$$3x^2 + \cos(2x) - 1$$



$$\frac{\partial^2 \psi}{\partial x^2} - \frac{1}{\nu^2} \frac{\partial^2 \psi}{\partial t^2}$$



Generating random formulas

1. Build a random tree
2. Sample operators as internal nodes
3. Sample integers, n, or past terms as leaves
4. Enumerate as a sequence

	Integer	Float
Unary	`abs, sqr, sign, step`	`abs, sqr, sqrt, inv, log, exp sin, cos, tan, atan`
Binary	`sum, sub, mul, intdiv, mod`	`sum, sub, mul, div`

Evaluating performance

- Model performance is defined as its ability to predict the next n_{pred} terms (1 to 10)
  - Directly or using the symbolic formula
- All predicted term must be predicted up to some tolerance τ (10^{-10})

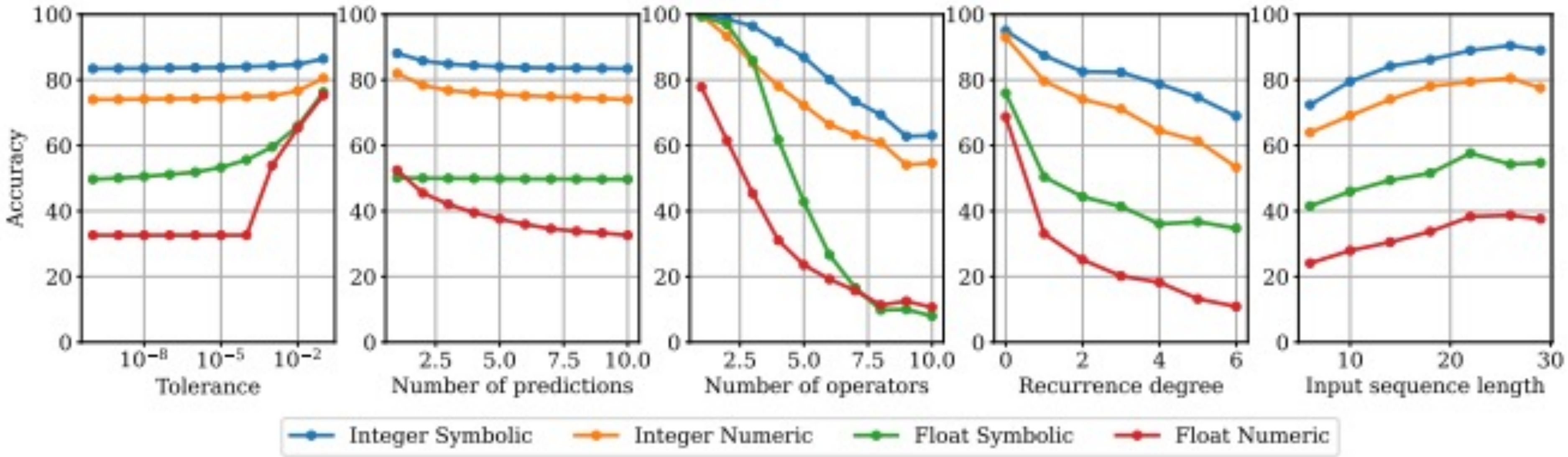
$$\mathrm{acc}(n_{pred}, \tau) = \mathbb{P}\left(\max_{1 \le i \le n_{pred}} \left| \frac{\hat{u}_i - u_i}{u_i} \right| < \tau \right)$$

- Accuracy is evaluated on a test set of 10 000 held-out examples

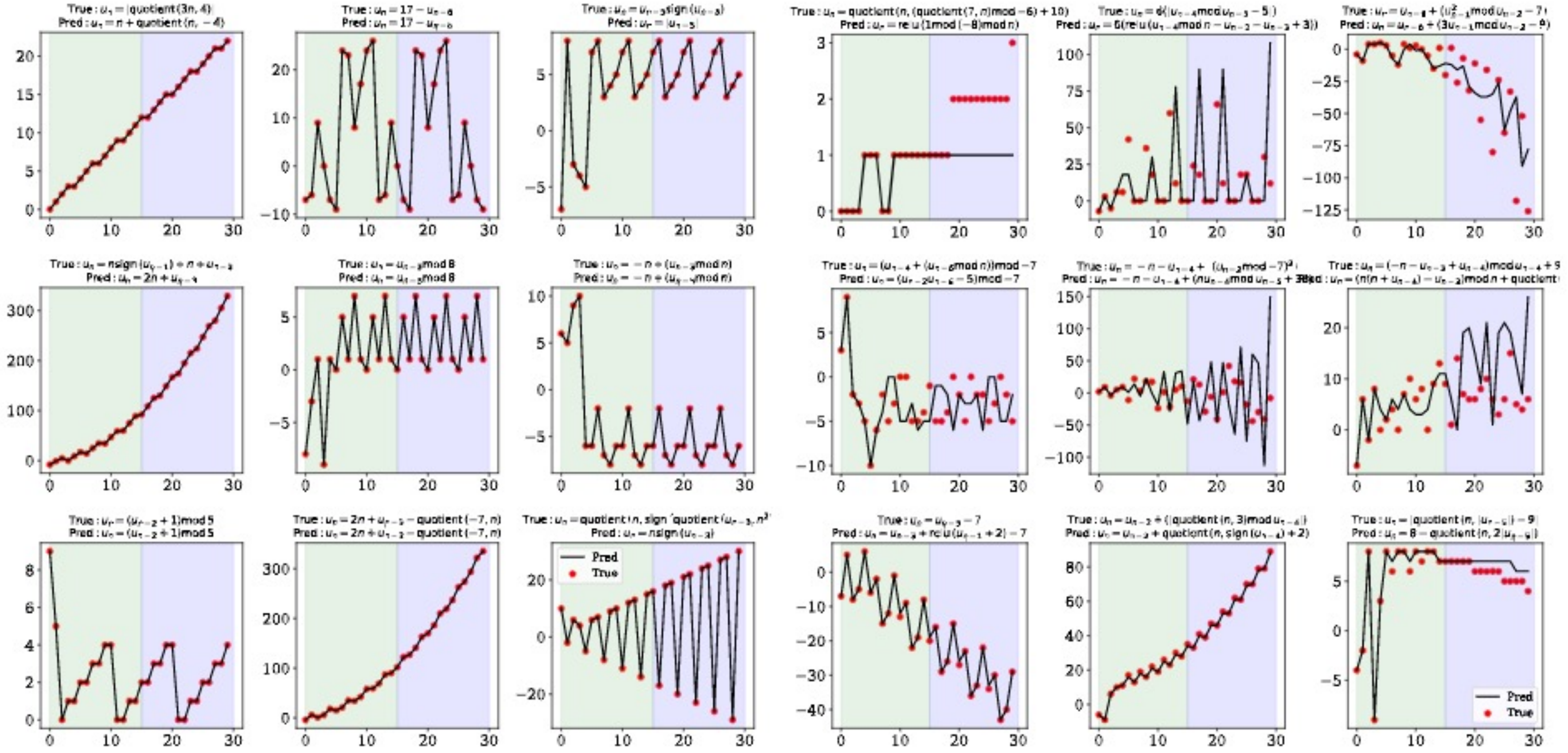
In domain results

Model	Integer		Float	
	$n_{op} \leq 5$	$n_{op} \leq 10$	$n_{op} \leq 5$	$n_{op} \leq 10$
Symbolic	**92.7**	**78.4**	**74.2**	**43.3**
Numeric	83.6	70.3	45.6	29.0

Table 6: **Average in-distribution accuracies of our models.** We set $\tau = 10^{-10}$ and $n_{pred} = 10$.



Success and failure cases



(a) Integer, success

(b) Integer, failure

Out-of-domain generalization-integers

Model	$n_{input} = 15$		$n_{input} = 25$	
	$n_{pred} = 1$	$n_{pred} = 10$	$n_{pred} = 1$	$n_{pred} = 10$
Symbolic (ours)	33.4	19.2	34.5	21.3
Numeric (ours)	53.1	27.4	54.9	29.5
FindSequenceFunction	17.1	12.0	8.1	7.2
FindLinearRecurrence	17.4	14.8	21.2	19.5

Table 7: **Accuracy of our integer models and Mathematica functions on OEIS sequences.** We use as input the first $n_{input} = \{15, 25\}$ first terms of OEIS sequences and ask each model to predict the next $n_{pred} = \{1, 10\}$ terms. We set the tolerance $\tau = 10^{-10}$.

Out-of-domain generalization- integers

OEIS	Description	First terms	Predicted recurrence
A000792	$a(n) = \max\{(n-i)a(i), i < n\}$	1, 1, 2, 3, 4, 6, 9, 12, 18, 27	$u_n = u_{n-1} + u_{n-3} - u_{n-1}\%u_{n-3}$
A000855	Final two digits of 2^n	1, 2, 4, 8, 16, 32, 64, 28, 56, 12	$u_n = (2u_{n-1})\%100$
A006257	Josephus sequence	0, 1, 1, 3, 1, 3, 5, 7, 1, 3	$u_n = (u_{n-1} + n)\%(n-1) - 1$
A008954	Final digit of triangular number $n(n+1)/2$	0, 1, 3, 6, 0, 5, 1, 8, 6, 5	$u_n = (u_{n-1} + n)\%10$
A026741	$a(n) = n$ if n odd, $n/2$ if n even	0, 1, 1, 3, 2, 5, 3, 7, 4, 9	$u_n = u_{n-2} + n//(u_{n-1} + 1)$
A035327	n in binary, switch 0's and 1's, back to decimal	1, 0, 1, 0, 3, 2, 1, 0, 7, 6	$u_n = (u_{n-1} - n)\%(n-1)$
A062050	n-th chunk consists of the numbers 1, ..., 2^n	1, 1, 2, 1, 2, 3, 4, 1, 2, 3	$u_n = (n\%(n - u_{n-1})) + 1$
A074062	Reflected Pentanacci numbers	5, -1, -1, -1, -1, 9, -7, -1, -1, -1	$u_n = 2u_{n-5} - u_{n-6}$

Fun facts

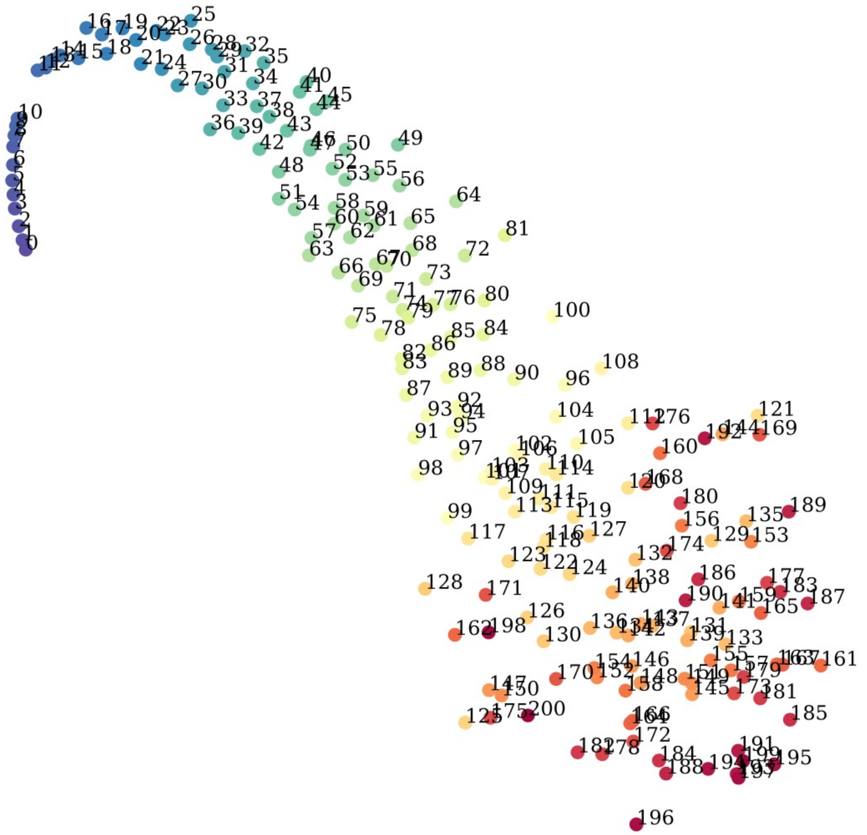
Constant	Approximation	Rel. error
0.3333	$(3+\exp(-6))^{-1}$	10^{-5}
0.33333	$1/3$	10^{-5}
3.1415	$2\arctan(\exp(10))$	10^{-7}
3.14159	π	10^{-7}
1.6449	$1/\arctan(\exp(4))$	10^{-7}
1.64493	$\pi^2/6$	10^{-7}
0.123456789	$10/9^2$	10^{-9}
0.987654321	$1-(1/9)^2$	10^{-11}

Expression u_n	Approximation $\hat{u}_n$
$\operatorname{arcsinh}(n)$	$\log(n+\sqrt{n^2+1})$
$\operatorname{arccosh}(n)$	$\log(n+\sqrt{n^2-1})$
$\operatorname{arctanh}(1/n)$	$\frac{1}{2}\log(1+2/n)$
$\operatorname{catalan}(n)$	$u_{n-1}(4-6/n)$
$\operatorname{dawson}(n)$	$\frac{n}{2n^2-u_{n-1}-1}$
$\mathrm{j0}(n)$ (Bessel)	$\frac{\sin(n)+\cos(n)}{\sqrt{\pi n}}$
$\mathrm{i0}(n)$ (mod. Bessel)	$\frac{e^n}{\sqrt{2\pi n}}$

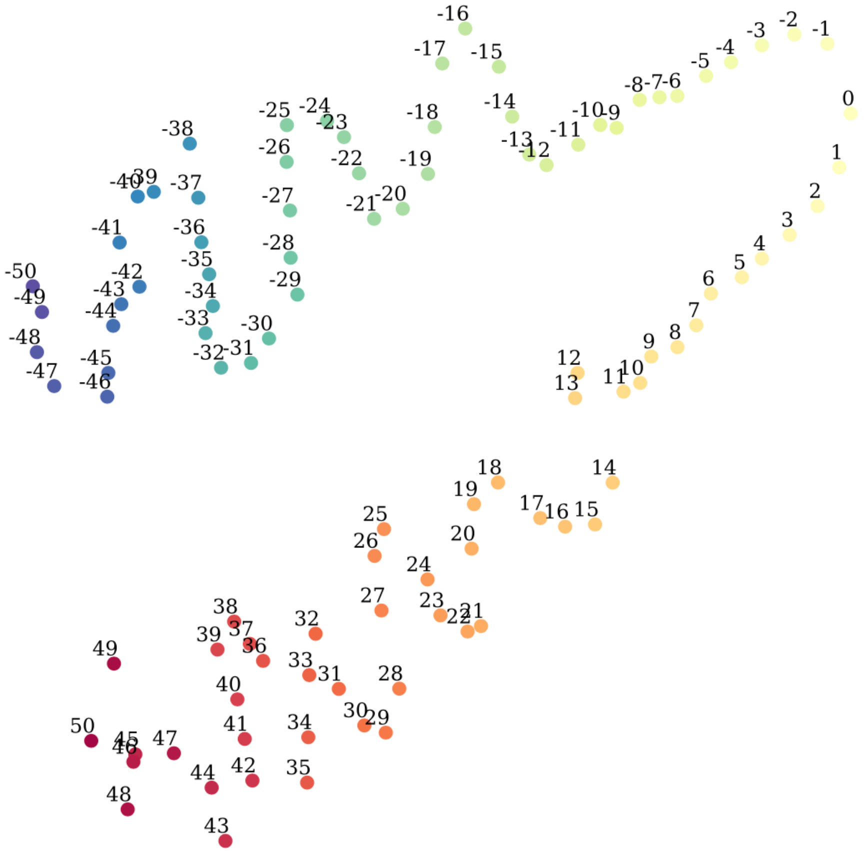
Approximating constants

Approximating functions

Fun facts- embeddings



Integer



Floating point exponents

Predicting gluon scattering amplitudes
(Cai, Merz, Nolte, Wilhelm, Cranmer, Dixon, Charton, 2024)

- Scattering amplitudes: complex functions predicting the outcome of particle interactions
- Computed by summing Feynman diagrams of increasing complexity
  - loops: virtual particles created and destroyed in the process
- A hard problem: each loop introduces two latent variables, their integration give rise to generalized polylogarithms
  - For the standard model the best computational techniques only reach loop 3

Amplitude bootstrap

- Polylogarithms have many algebraic properties
  - Leverage them to predict the structure of the solution, up to some coefficients
  - Compute the coefficients from symmetry consideration, known limit values, etc.
- In Planar N=4 supersymmetric Yang-Mills, solutions are “simple”
  - Calculated from symbols: homogeneous polynomials, degree 2L (L=loop), with integer coefficients

The three gluon form factor

- Three gluons and a Higgs
- Amplitudes for loop L can be computed from symbols
  - homogeneous polynomials in 6 non-commutative variables: a,b,c,d,e,f
  - with integer coefficients
  - -4 bccaff + 4 bcbaff + 8 bcafff + ...
- 6^{2L} possible “keys”, mapped to integers
  - Most of them zero
- Symmetries and asymptotic properties translate into constraints
  - An enormous integer programming problem
  - Could be solved up to loop 8

L	number of terms
1	6
2	12
3	636
4	11,208
5	263,880
6	4,916,466
7	92,954,568
8	1,671,656,292

TABLE II. Number of terms in the symbol of $F_3^{(L)}$ as a function of the loop order L.

The six letter game

- We want to learn a mapping between "keys" (sequences of length 2L of the 6 letters, a,b,c,d,e and f) and integer coefficients
- There are obvious symmetries in the symbol
  - Coefficients are invariant by the dihedral symmetry generated by
    - a -> b->c -> a, d -> e -> f -> d, a <-> b, d <-> e
    - bccaff maps to -4, so does abbcee
  - Non zero coefficients
    - must begin with a, b or c, and end with d, e or f
    - Have no contiguous a and d, b and e, c and f, d and e, e and f and d and f

The six letter game

- And many less obvious symmetries
  - Non zero keys ending with a single letter d,e or f, must be preceded by a run of one of the letters a, b or c
    - A key ending in eccccd can be non zero, one ending in ecbcd must be zero
- And many empirical facts hold true over all symbols
  - Large absolute coefficients happen for symbols with many runs of one letter
- Can some of these relations be learned, empirically, by a language model?
  - To help calculate loops
  - To discover new facts about amplitudes in planar N=4

Experiment 1 : Predicting zeroes

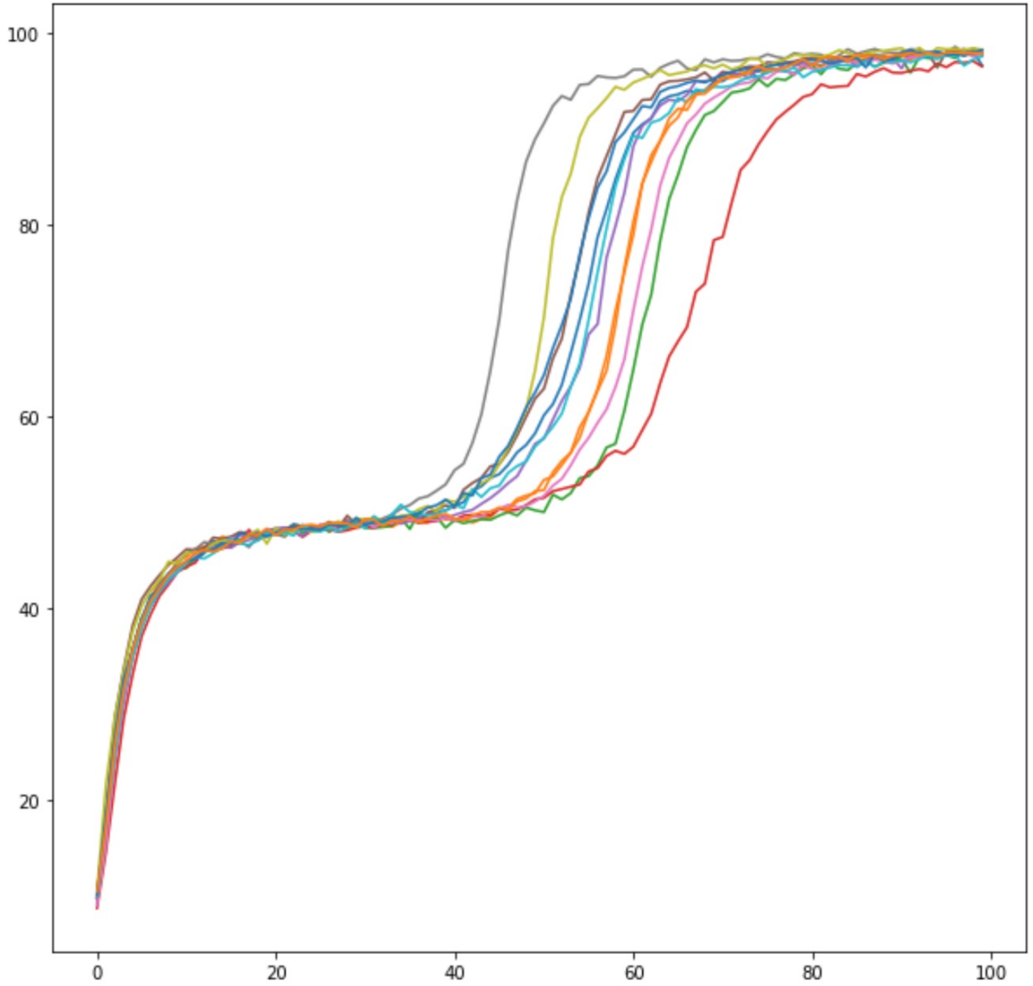
- For Loop 5 and 6, predict whether a term is zero or nonzero
  - afdcfdadfe is zero
  - aaaeeceaaf is not
- Build a 50/50 training sample of zero/non zero terms
- Reserve 10k terms for test, these will not be seen during training
- Train the model, and measure performance on the test set (% of correct prediction)
  - For input a,f,d,c,f,d,a,d,f,e predict 0
  - For input a,a,a,e,e,c,e,a,a,f predict 1

Experiment 1 : Predicting zeroes

- Loop 5 : after training on 300,000 examples (57% of the non zero keys and as many zero keys), the model predict 99.96% of test examples (not seen during training)
- Loop 6 : after training on 600,000 examples (6% of the symbol), the model predicts 99.97% of test examples

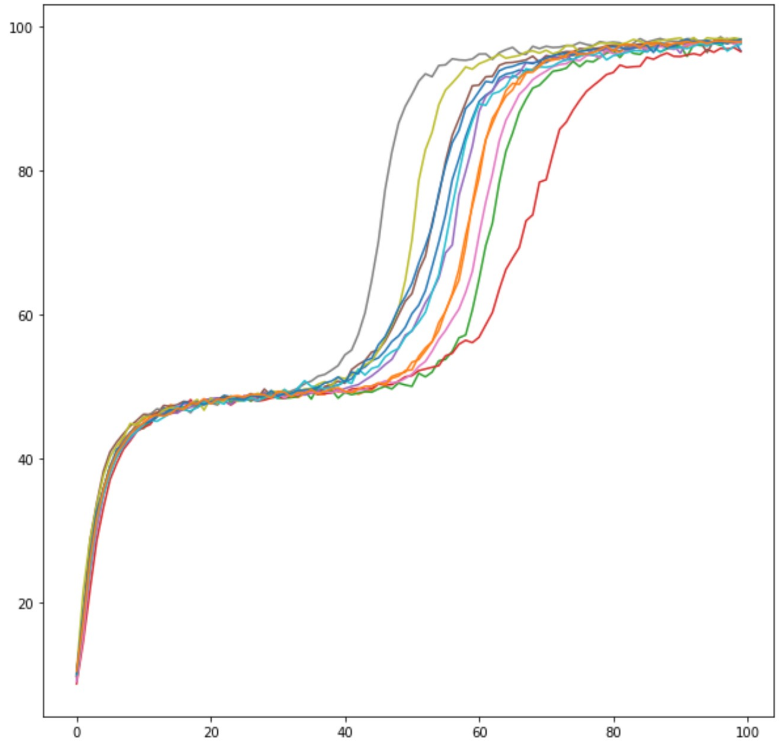
Experiment 2 : Predicting non-zeroes

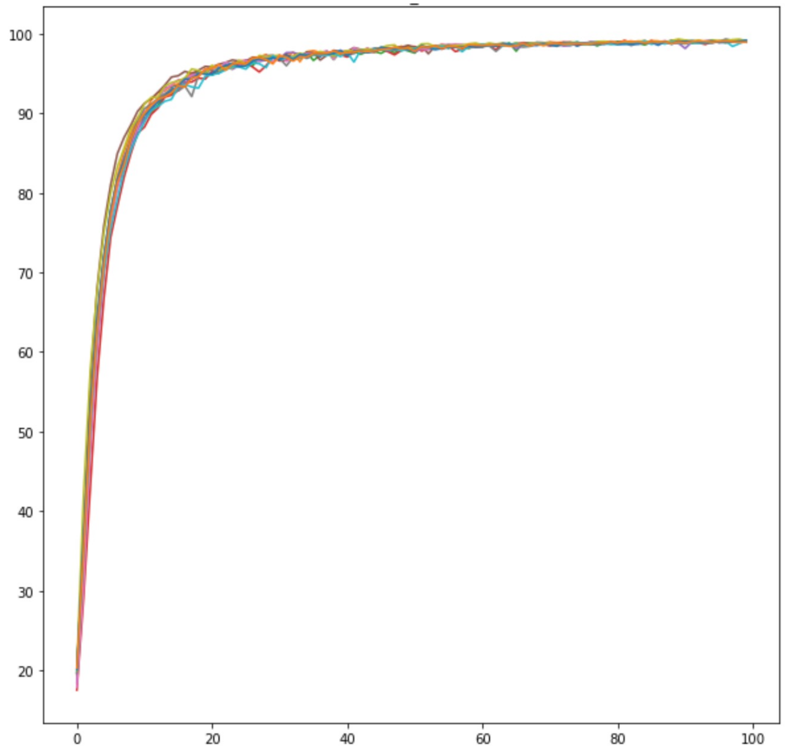
- From keys, sequences of 2L letters, predict coefficients, integers encoded in base 1000
- For loop 5, models trained on 164k examples (62% of the symbol), tested on 100k
  - 99.9% accuracy after 58 epochs of 300k examples
- For loop 6, models trained on 1M examples (20% of the symbol), tested on 100k
  - 98% accuracy after 120 epochs
  - BUT a two step learning curve

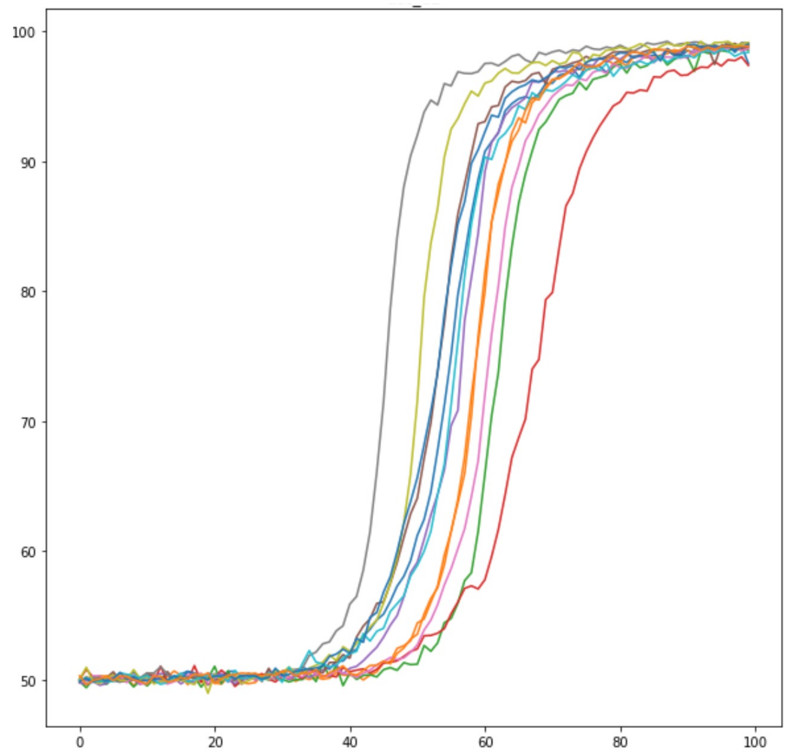


Experiment 2 : Predicting non-zeroes

- full prediction, magnitude and sign





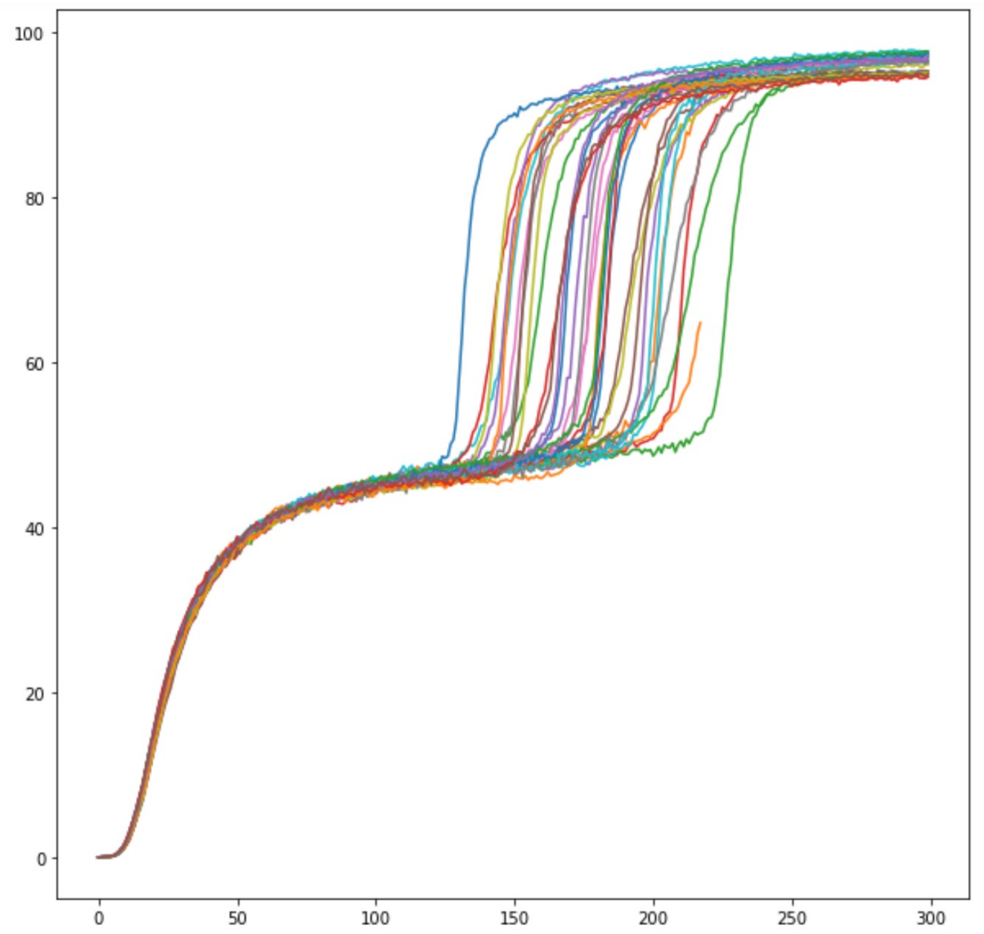


Experiment 3 : Learning with less symmetries

- Non zero coefficients
  - Must begin with a,b,c and end with d,e,f
  - Are invariant by dihedral symmetry
  - Cannot have a next to d (b next to e, c next to f)
  - Cannot have d next to e or f (e next to d or f)
- Only a few endings are possible:
  - 8 “quads” (4 letter endings, up to cyclic symmetry (a,b,c), (d,e,f))
  - 93 octuples

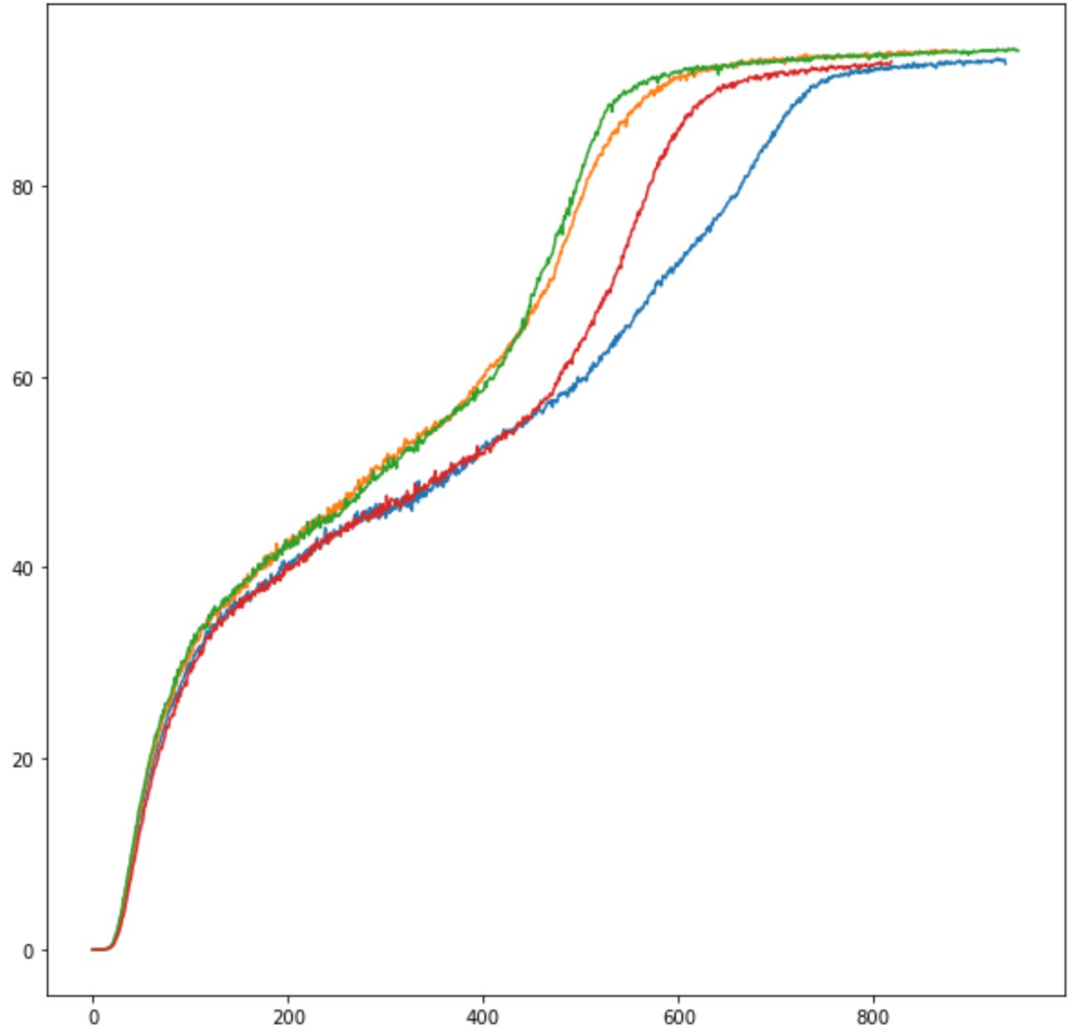
Experiment 3 : Learning loop 7 quads

- 7.3 million elements in the symbol (vs 93 millions in full representation)
- Models learn to predict with 98% accuracy
- Same "two step" shape



Experiment 3 : Learning loop 8 octuples

- 5.6 million elements in the symbol (vs 1.7 billions in full representation)
- Models learn to predict with 94% accuracy
- Attenuated “two step” shape
- Slower learning (600 epochs, vs 200 for quads, and 70 for full representation)



Take aways from experiments 1-3

- We can use transformers to complete partially calculated loops
- Coefficients are learned with high accuracy
  - Even when only a small part of the symbol is available
- A few unintuitive observations happen:
  - hardness of learning the sign
  - might shed new light on the underlying phenomenon

Experiment 4: predicting the next loop

- A loop L element E is a sequence of 2L letters
- Strike out 2 of the 2L letters
  - From aabd make bd, ad, ab...
  - There are L(2L-1) parents, call them P(E)
- Try to find a recurrence relation, that predicts the coefficient of E from its parents: E = f(P(E))
  - A generalized Pascal triangle/pyramid (in 6 non-commutative variables)

- Predict loop 6 from loop 5:
  - From 66 integers: loop 5 coefficients
  - Predict 1 integer: the loop 6 coefficient
  - (NOT the keys: we already know the model can predict coefficients from keys)
- 98.1% accuracy, no difference between sign (98.4) and magnitude (99.6) accuracy
- A function f certainly exists (but we have no idea what it is)

Experiment 4: understanding the recurrence

- To collect information on f, the unknown recurrence, we could
  - Remove information about the parents
  - See if the model still learns
- Can we use less parents?
  - Only strike letters at most k tokens apart; e.g. k=1 only consecutive tokens
  - k=2: 21 parents, k=1: 11 parents

	Accuracy	Magnitude accuracy	Sign accuracy
Strike two, all parents	98.1	98.4	99.6
Strike two, k=5	98.3	98.6	99.7
Strike two, k=3	98.4	98.7	99.7
Strike two, k=2	98.1	98.3	99.5
Strike two, k=1	94.3	95.2	98.5

Experiment 4: understanding the recurrence

- Shuffling/sorting the parents do not prevent learning
- Coupling between parent/children signs, and magnitudes

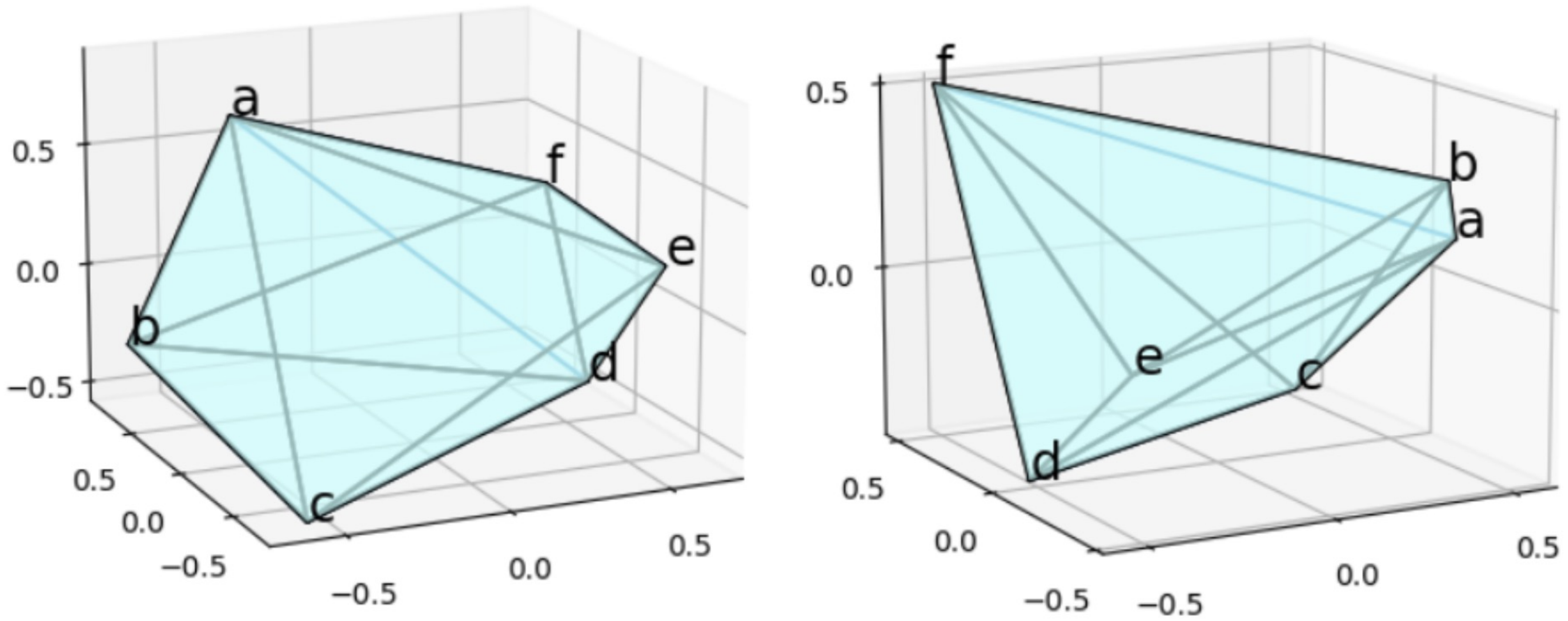
	Accuracy	Magnitude accuracy	Sign accuracy
Strike two, all parents	98.1	98.4	99.6
Strike two, k=5	98.3	98.6	99.7
Strike two, k=3	98.4	98.7	99.7
Strike two, k=2	98.1	98.3	99.5
Strike two, k=1	94.3	95.2	98.5
Shuffled parents	95.2	99.1	96.3
Shuffled parents, k=2	93.5	98.1	95.0
Sorted parents, k=5	93.9	95.4	97.9
Parent signs only	93.3	93.5	99.0
Parent magnitudes only	81.8	98.4	83.2

Table 2: **Global, magnitude and sign accuracy.** Best of four models, trained for about 500 epochs

Next steps

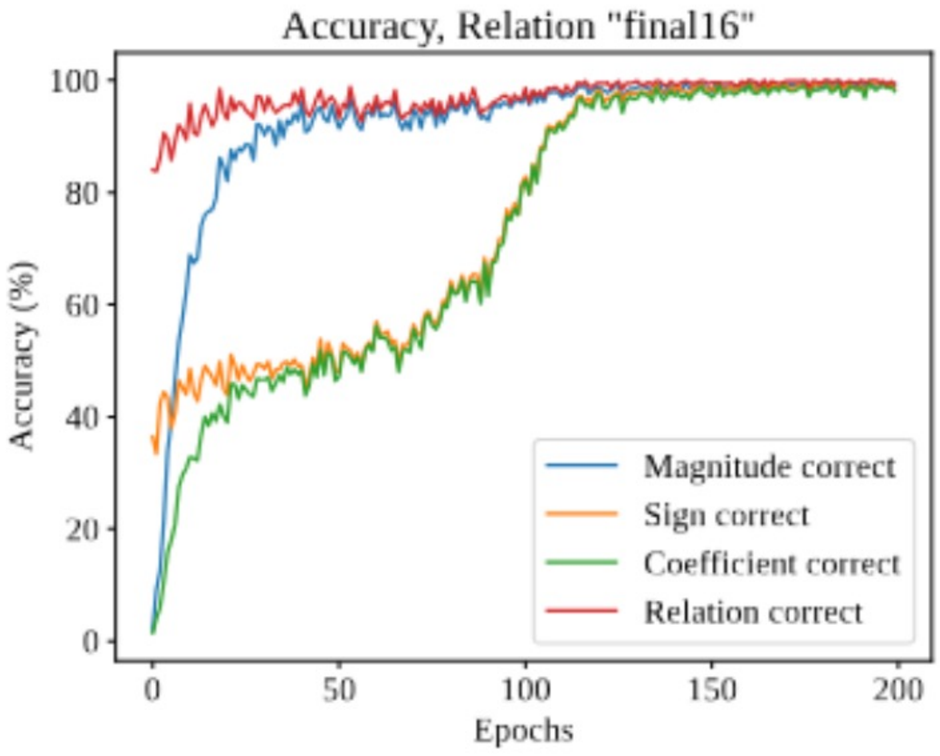
- Better understanding the recurrence relation
  - Try building loop 9, or loops for related problems
- Discovering local properties/symmetries in the symbol
  - Symbols were calculated by exploiting known symmetries in nature
  - If we discover new regularities in the symbols, what does is tell us about nature?
  - Antipodal symmetries

Fun facts: learning the dihedral symmetry

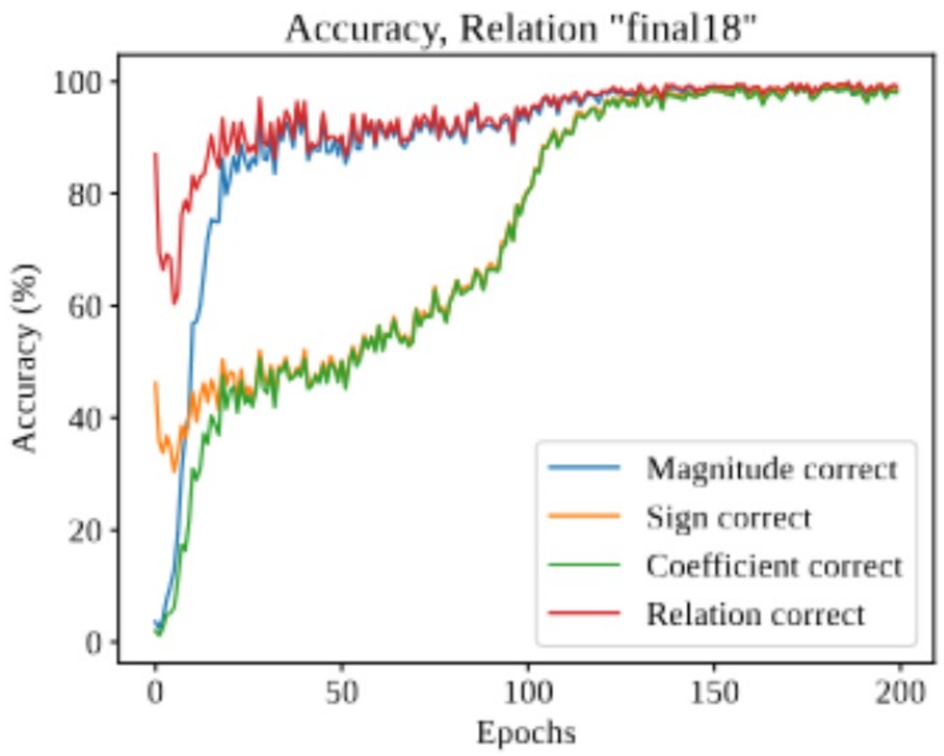


Fun facts: learning relations between coefficients

$\texttt{final 16:} \quad \mathcal{E}^{b,f} - \mathcal{E}^{b,d} = 0,$



$\texttt{final 18:} \quad \mathcal{E}^{d,d,b,d} - \mathcal{E}^{d,b,d,d} = 0.$



Next steps

- We have a proof of concept :
  - Models can predict coefficients from key
  - Or discover recurrences from one loop to the next
- Can we go for loop 9? Or other problems?
- Can we reverse engineer the models?
  - By looking at their weights?
  - By looking at the representations they learn?
  - By looking at the way they train?
- If we train a language model on “all we know” about the symbol (like we train ChatGPT on all we know about language), will it learn new, emerging, properties of the symbols?

A growing area of research

In symbolic mathematics, we are beginning to use transformers to help solve longstanding open problems.

- Current projects in symbolic mathematics
- Discovering the (symbolic) Lyapunov functions that control the global stability of dynamical systems (e.g. the N-body problem)
- Discovering yet unknown kernel elements in the Burau representation of braid groups
- Could we use transformers in theoretical physics?