SYMBOLIC MACHINE LEARNING IN PHYSICS

FRANÇOIS CHARTON, FAIR, META – ÉCOLE DES PONTS

ML IN PARTICLE PHYSICS

- ML is use for a large range of tasks
  - Jet classification
- Tendency to move from VAE and GNN towards transformers
  - Dominant architecture in AI research: more papers, more fundamental research
  - Need a lot of data, but data, measured or generated, is available in HEP
- Increasing interest about pre-trained models: LLM for physics
  - A large model, pre-trained on a large dataset (unsupervisedly, usually)
  - Fine-tuned on many related tasks

ML FOR PARTICLE PHYSICS

- In most of these applications ML models “compute”: they process numbers, and output numbers
- This is not what transformers were designed for
  - Natural language processing mostly deals with discrete symbols (words in a language)
  - LLM are bad at computing:
    - Integer addition can only be performed using “tricks”: scratchpad, special positional encodings
    - Integer multiplication only works so long one operand is small
    - Combinations of these tasks, negative numbers, are a mess

AI FOR SYMBOLIC MATHEMATICS (AND PHYSICS?)

- Dealing with functions, graphs, equations, symbolic mathematical objects
  - Symbolic regression: discovering laws from data (numerical input symbolic output)
  - Solving symbolic equations
  - Finding counter examples

- Traditionally associated with reinforcement learning (AlphaGo) and genetic programming
- Transformers play an increasing role

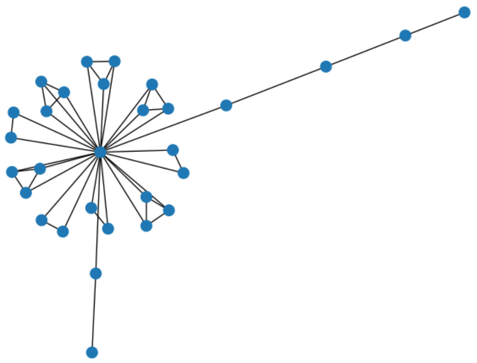
FINDING COUNTER-EXAMPLES IN GRAPH THEORY

Constructions in combinatorics via neural networks, Wagner 2021, 2104.14516

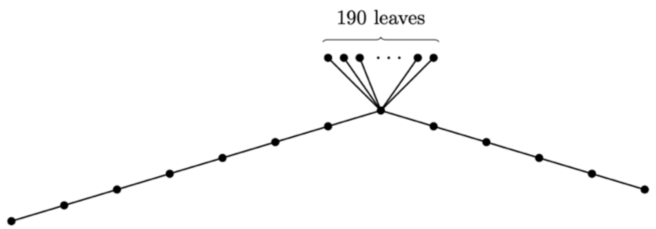
Conjecture (Aouchiche-Hansen 2011): Let G be a connected graph, with $n \geq 4$ vertices, diameter (max distance between vertices) D, proximity (average distance between nodes) π and spectral distance (eigenvalues of distance matrix) $\partial_1 \geq \partial_2 \geq \ldots \geq \partial_n$,

Then $\pi + \partial_{\left\lfloor \frac{2D}{3} \right\rfloor} > 0$

Train a model to find counter-examples, it fails, but all failed solutions follow a certain pattern



That a mathematician can turn into a valid counter-example



DISCOVERING OPTIMAL CONSTRUCTIONS

PatternBoost: constructions in mathematics with a little help from AI
Charton, Ellenberg, Wagner, Williamson 2024, 2411.00566

- Finding discrete objects that maximize a quantity:
  - Largest graphs with n nodes, but no cycle of 4
  - Largest set of points on a n^3 grid, with no 5 points on a sphere
  - Smallest subset of the d-dimensional hypercube, with diameter d
- Generate random solutions, make them as good as you can with local search, keep the best candidates
- Train a transformer (GPT-2) on the best candidates
  - Tokenized by their adjacency matrix, using Byte Pair Encoding: standard NLP tools
- Use it to generate more candidates
- Improve them with local search
- Rinse, repeat...

DISCOVERING OPTIMAL CONSTRUCTIONS

- This works!
- Competitive on hard problems, like no square graphs
- Found hitherto unknown no-sphere solutions for n=6 (best known was 17, we found 18)
- Solved a 30 years-old conjecture about d-hypercubes with diameter d

- Also, FunSearch (DeepMind 2023): use an LLM to create programs to find solutions to combinatorial problems
  - Cap Set problem: largest subset in $\mathbb{Z}_3^n$ with no three points on a line
  - FunSearch discovered new optimal solutions

PROVING THE GLOBAL STABILITY OF DYNAMICAL SYSTEMS

Global Lyapunov functions : a long-standing open problem in mathematics, with symbolic transformers, Alfarano, Hayat, Charton, 2024, 2410.08304

- Dynamical systems: $\dot{x} = f(x), \quad x \in \mathbb{R}^n, \quad f \in C^1(\mathbb{R}^n)$
- Global stability: if we start close to an equilibrium, do we always stay close, or can we diverge to infinity
  - Stability of the solar system, 3-body problem
- Lyapunov (1892) it is if you can find $V \in C^1(\mathbb{R}^n, \mathbb{R})$, such that for all $x \in \mathbb{R}^n$,

$$V(x) > V(0)$$
$$\lim_{|x| \to +\infty} V(x) = +\infty$$
$$\nabla V(x) . f(x) \leq 0$$

- How to find V? An open problem except in the simplest cases

PROVING THE GLOBAL STABILITY OF DYNAMICAL SYSTEMS

- We train a transformer, on generated examples of problems and solutions, to discover Lyapunov functions
  - Generating random solutions, and associated problems
  - Symbolic input, symbolic output (functions)
- Beats the state of the art on polynomial systems

Test sets	SOSTOOL findlyap	Existing AI methods Fossil 2	ANLC	LyzNet	Models PolyMixture	FBarr	FLyap	BPoly
FSOSTOOLS	-	32	30	46	**84**	80	53	54
FBarr	-	12	18	28	**93**	-	28	35
FLyap	-	42	32	66	84	**93**	-	73
BPoly	15	10	6	24	**99**	15	10	-

Table 5: **Performance comparison on different test sets**. Beam size 50. PolyMixture is BPoly + 500 FBarr.

- Discovers Lyapunov functions for random systems for which no method is known

Test set	Sample size	SOSTOOL findlyap	Existing AI methods Fossil 2	ANLC	LyzNet	Forward model FBarr	Backward model PolyMixture	NonPolyMixture
Poly3	65,215	1.1	0.9	0.6	4.3	11.7	**11.8**	11.2
Poly5	60,412	0.7	0.3	0.2	2.1	8.0	**10.1**	9.9
NonPoly	19,746	-	1.0	0.6	3.5	-	-	**12.7**

Table 6: **Discovering Lyapunov comparison for random systems**. Beam size 50. PolyMixture is BPoly + 500 FBarr. NonPolyMixture is BNonPoly + BPoly + 500 FBarr.

SYMBOLIC AI FOR SCATTERING AMPLITUDES

Transforming the bootstrap: using transformers to compute scattering amplitudes in planar n=4 super Yang-Mills theory, Cai, Merz, Charton, Nolte, Wilhelm, Cranmer, Dixon, 2024, 2405:06107

- Scattering amplitudes: complex functions describing particle interactions
  - Their squared module are probabilities of outcomes
  - Baselines for experiments, need to be computed to high precision
- Computed by summing Feynman diagrams of increasing complexity
  - measured in loops: virtual particles created and destroyed in the process, correspond to loops in the Feynman diagrams
  - One more loop: $\mathrm{x}10$ in precision

AI FOR SCATTERING AMPLITUDES

- A hard problem
- Each loop introduces two latent variables, their integration give rise to generalized polylogarithms
- Best precision at present: loop 3 for some interactions
- Theoretical research on integration by part: aka computational tricks
- Some ML results: Simplifying polylogarithms with machine learning, Dersy, Schwartz, Zhang, 2022, 2206.04115

BOOTSTRAPPED AMPLITUDES

- Leverage algebraic properties of polylogarithms to predict the structure of the solution
  - Up to a (large) number of integer coefficients
  - That can be computed from symmetry, integrability, limit conditions
- In Planar N=4 supersymmetric Yang-Mills, solutions are "simple"
  - Symbols: homogeneous polynomials of degree 2L (L=loop), over $\mathbb{Z}$
  - Can be computed to high loops: 3 gluons form-factor to 8 loops

Bootstrapping a stress-tensor form factor through eight loops, Dixon, Gurdogan, McLeod, Wilhelm, 2022, 2204.11901

THE THREE GLUON FORM FACTOR

- 3 gluons and a Higgs-like "operator"
- Symbols are polynomials in 6 (non commutative) variables a,b,c,d,e,f
  - Loop 3: -4 bccaff + 4 bcbaff + 8 bcafff + …
- For loop L, 6^{2L} possible keys (ordered sequences of 2L letters) mapped onto integers
  - Most of them zero
  - We want to understand the mapping

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

- Coefficients are invariant by the dihedral symmetry: generated by (a,b,c), (d,e,f), (a,b), (d,e)
- Adjacencies: non-zero keys must
  - Begin with a,b, or c
  - End with d,e,or f
  - Not have adjacent a and d, b and e, c and f, d and e, d and f, e and f
- Relations exist between identical keys up to a few letters ($F^{a,b}$ is the coefficient of a key with a and b adjacent)

$$F^{a,b} + F^{a,c} - F^{b,a} - F^{c,a} = 0, \tag{3.6}$$

$$F^{c,a} + F^{c,b} - F^{a,c} - F^{b,c} = 0, \tag{3.7}$$

$$F^{d,b} - F^{d,c} - F^{b,d} + F^{c,d} + F^{e,c} - F^{e,a} - F^{c,e} + F^{a,e} + F^{f,a} - F^{f,b} - F^{a,f} + F^{b,f} + 4(F^{c,b} - F^{b,c}) = 0, \tag{3.8}$$

TRANSFORMERS FOR BOOTSTRAP

- Many other regularities exist, could a language model find them?
- Train a transformer to predict coefficients (sequences of digits in base 1000) from their keys (sequences of 2L letters)
- Small encoder–decoder model, trained on a fraction of a loop data, tested on its prediction of the rest
  - Minimising cross-entropy, a "letter game"

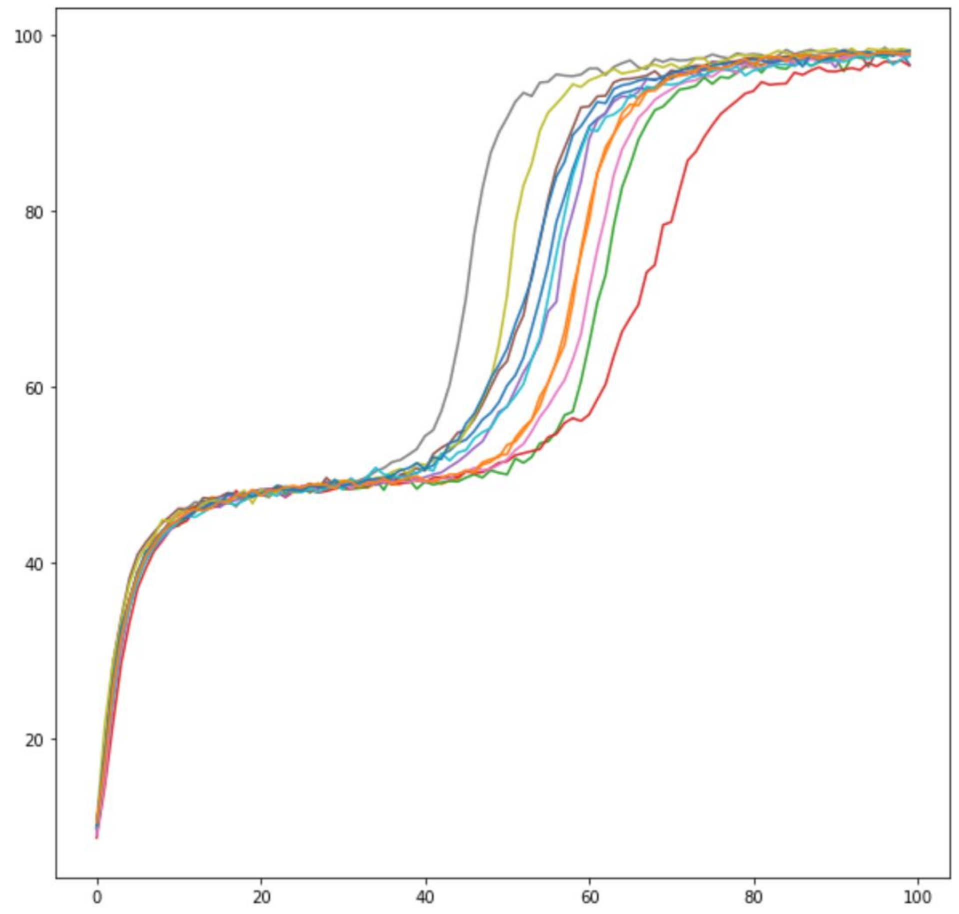
EXPERIMENT 1: PREDICTING ZEROES

- Given an key, predict whether the coefficient is different from zero
- A 50/50 sample of zero and non-zero keys

- 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

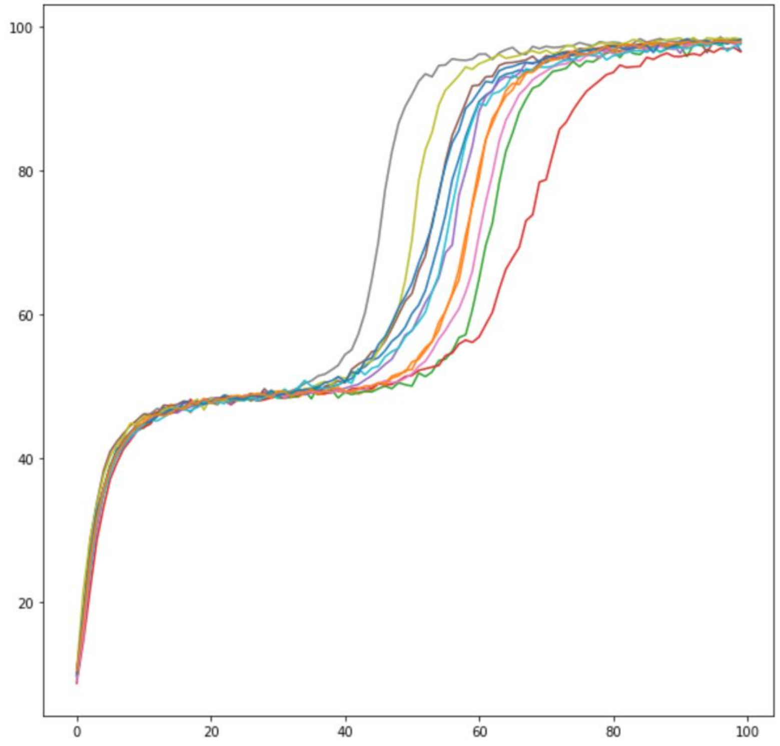
EXPERIMENT 2: PREDICTING NON-ZERO COEFFICIENTS

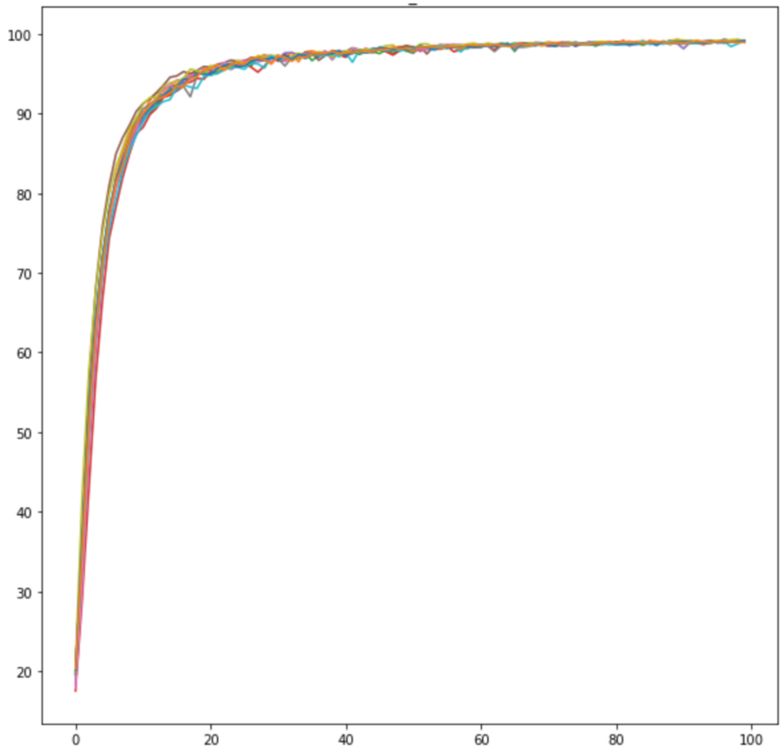
- From keys, sequences of 2L letters, predict coefficients, integers encoded in base 1000
- For loop 5, models trained on 164k examples (62% of the symbol), tested on 100k
  - 99.9% accuracy after 58 epochs of 300k examples
- For loop 6, models trained on 1M examples (20% of the symbol), tested on 100k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

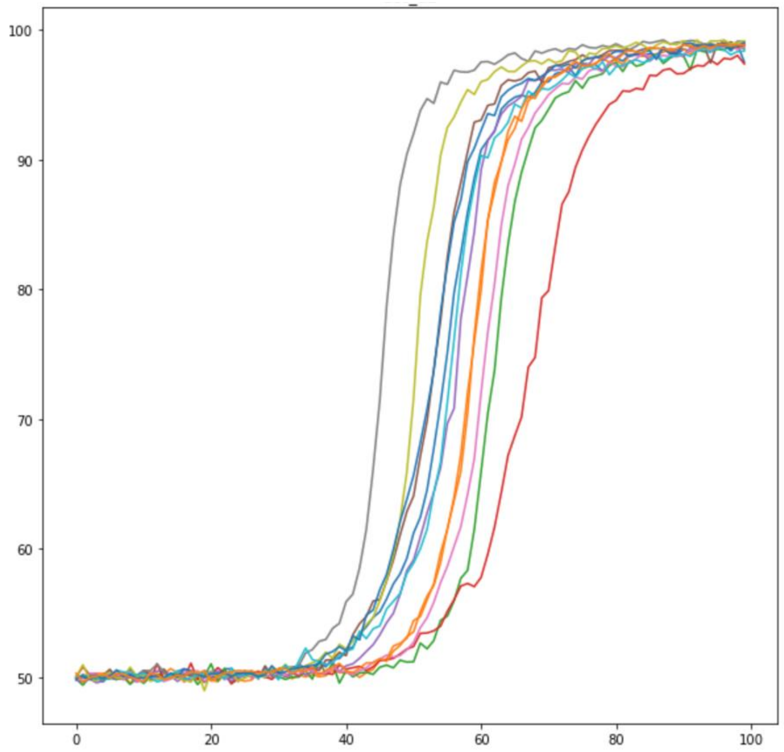


EXPERIMENT 2: PREDICTING NON-ZERO COEFFICIENTS

- Full prediction, magnitude and sign
  - The absolute value is easy to predict, the sign is not







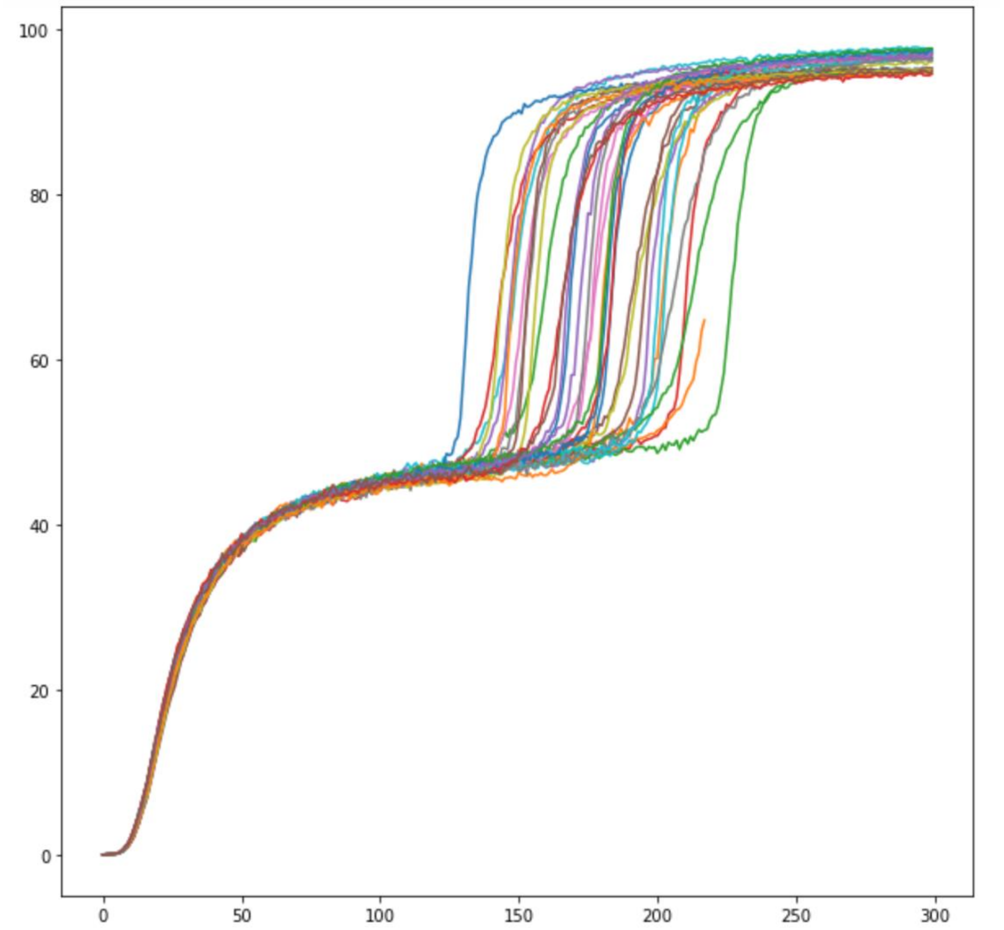
EXPERIMENT 3: REMOVING OBVIOUS SYMMETRIES

- Symbols satisfy a dihedral symmetry: 6 copies of each element
- Only a few endings are possible
  - 8 quads (suffixes of length 4)
  - 93 octuples (suffixes of length 8)
- A more compact representation for higher loops, and a harder problem, because the easiest regularities have been removed from the training data

EXPERIMENT 3: REMOVING OBVIOUS SYMMETRIES

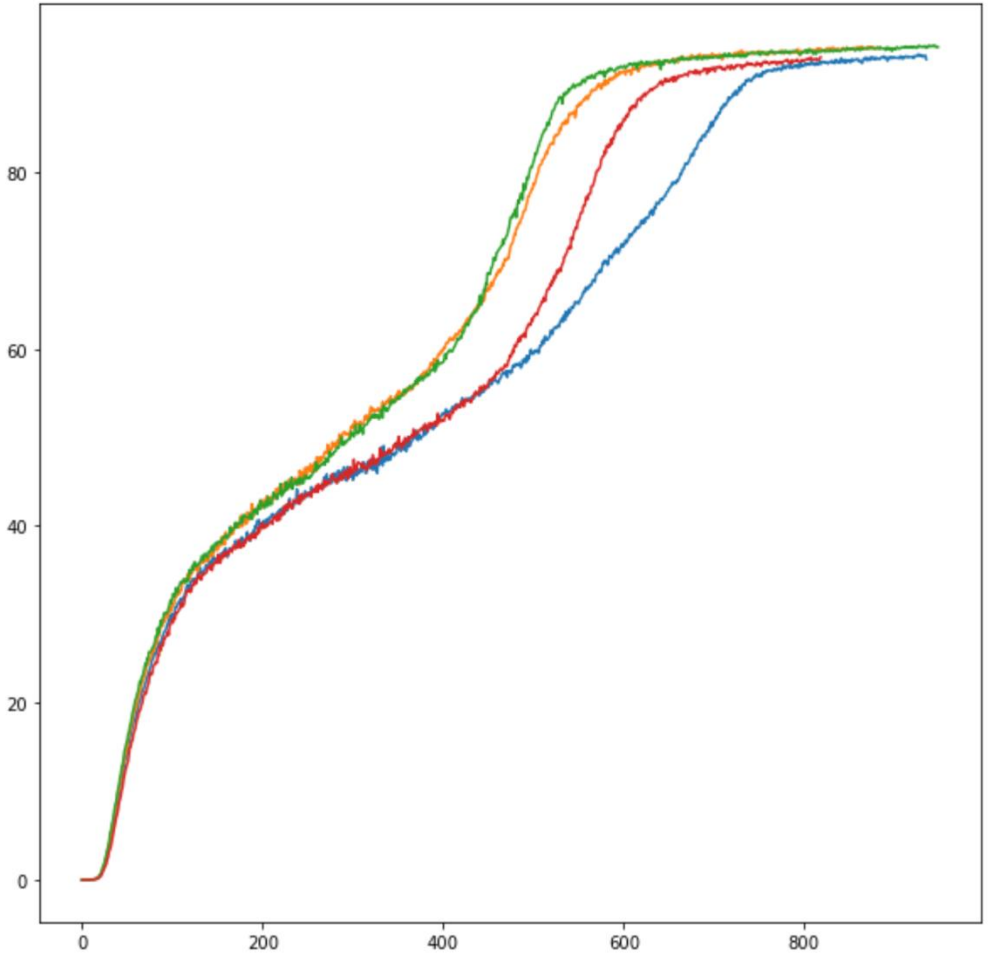
- Quads at loop 7: 7.3 non zero elements in the symbols (vs 93 million ins the full representation)
- The model learns just as well
- Same two-step shape

- The model is not "just learning" the obvious regularities



EXPERIMENT 3: REMOVING OBVIOUS SYMMETRIES

- Octuples at loop 8: 5.6 non zero elements, vs 1.7 billion)
- 94% accuracy
- Smoothed two step shape
- Slower learning (600 epochs, vs 200 for quads, and 70 for full representation)



TAKE AWAYS FROM EXPERIMENTS 1 - 3

- Transformers can complete partially calculated loops

- Coefficients are learned with high accuracy
  - Even when a small part of the symbol is available
- A few unintuitive observations
  - Hardness of learning the sign
  - May shed light on the underlying phenomenon

EXPERIMENT 4: LEARNING THE NEXT LOOP

- Find a recurrence relation connecting coefficients from loop L-1, to coefficients from loop L
- A loop L key has 2L letters, we can associate it to loop L-1 "parents", by striking out two letters
  - The parents of K=aabd are ~~aa~~bd = bd , ~~a~~a~~b~~d= ad, aab~~d~~ = ab, …
  - Call them P(K), there are L(2L-1) such parents
- Find a generalized recurrence linking the coefficient of K to it parents: E = f(P(K))
  - A generalized Pascal triangle/pyramid (in 6 non-commutative variables)

EXPERIMENT 4: LEARNING THE NEXT LOOP

- 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 do not know what it is)

EXPERIMENT 4: LEARNING THE NEXT LOOP

- We can learn about the unknown recurrence, by removing parents:
  - Only considering strike-outs of contiguous (or close apart) positions
  - k max distance for strike out : k=1 contiguous letters only, the smaller k; the less parents
  - Limited impact on performance for k larger than 1

	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: LEARNING THE NEXT LOOP

- Shuffling/sorting parents have little impact: the recurrence is almost permutation invariant
- Coupling between parent and child 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

THE SIX LETTER GAME REVISITED

- Since zeros are so easy to predict, there must be a general rule for adjacent zero keys
- Generalized end-rule: keys ending with a single letter d, e or f must be preceded with a run of a, b or c
  - * aaaaf can be non zero
  - * abbaf must be zero
  - Accounts for 92% of adjacent zeroes

THE SIX LETTER GAME REVISITED

- Since models can find relations between elements and their strike out parents exist, we could go looking for such empirical relations
  - Rays: sequences of keys of different loops, related by a "common strikeout pattern",
    - af, aaaf, aaaaaf, ..., or af, afff, afffff, ...
  - Closed recurrences can be found, coefficients of sequences ending with a variable length run of f verify

$$c_L(\mathtt{seq_7 f \ldots f}) = p_L(\mathtt{seq_7}) \times (-1)^L 2^{2L-8}(2L-9)!!\,, \tag{5.5}$$

  - With

$$p_L(\mathtt{aaf \ldots f}) = 0, \tag{5.6}$$

$$p_L(\mathtt{caaf \ldots f}) = 32(L-2)(2L-5)(2L-7)\,, \tag{5.7}$$

$$p_L(\mathtt{caaaf \ldots f}) = \frac{16}{3}(4L-9)(2L-5)(2L-7)\,, \tag{5.8}$$

$$p_L(\mathtt{ccaaaaf \ldots f}) = -\frac{4}{5}(2L-7)(7L^2+22L-140)\,, \tag{5.9}$$

$$p_L(\mathtt{cccccaaf \ldots f}) = -\frac{8}{3}(L-4)(L^2-47L+135)\,, \tag{5.10}$$

$$p_L(\mathtt{bdddbbbf \ldots f}) = -\frac{2}{45}(163L^3-2220L^2+15977L-36660)\,. \tag{5.11}$$

NEXT STEPS

- Try build loop 9, or loops for related problems
- Discover new properties of the symbol
  - Symbols were calculated by exploiting known symmetries
  - If we discover new regularities in the symbols, do we discover new symmetries?
- Train a language model on all loop data, and investigate its representations