TRANSFORMERS FOR MATHEMATICAL DISCOVERY LEARNING COLLATZ SEQUENCES

FRANÇOIS CHARTON, ENPC, AXIOM

LLM ARE BAD AT ARITHMETIC

- Pretrained LLM struggle with integer addition
  - Scratchpad, or chain-of-thought, is needed
  - For length generalization, you need to tweak positional embedding (Abacus)
- Even worse with multiplication
  - One operand must be short (3 digits in original ChatGPT , 5 digits nowadays)
  - Length generalization needs even more problem specific tampering
- They cannot compose
- Tend to hallucinate/confabulate

EXACT ARITHMETIC ON A TRANSFORMER

- Supervised learning, can archive close to 100% accuracy
- Greatest common divisor (FC 2024), modular arithmetic (FC, Kempe, 2024)
- Tweaking training distributions
  - Repeated examples
  - Balanced output
  - Small operands
- Functions with few outcomes
  - pattern recognition vs regression

COLLATZ SEQUENCES

- Starting from a positive integer: $u_0=n$
  - $u_1=3n+1$ if n is odd
  - $u_1=n/2$ if n is even
- Repeating, we generate an integer sequence, $(u_i)_{i\in\mathbb{N}}$
- One known cycle : 1, 4, 2, 1
- Famous conjecture: no matter the n you start with, you always reach the 1, 4, 2, 1 cycle

LONG COLLATZ STEPS

- Start with an odd integer n
  - do $n \longrightarrow (3n+1)/2$ as many times as you can (k times)
  - do $n \longrightarrow n/2$ as many times as you can (k' times)
  - call it $\kappa(n)$, the Long Collatz successor of n

- Starting with 27, we have :
  - (3.27+1) / 2 = 82 / 2 = 41
  - (3.41+1) / 2 = 124 / 2 = 62 (k=2)
  - 62 / 2 = 31 = $\kappa(27)$
  - here k=2, k'=1

LONG COLLATZ STEPS

- Suppose $n = 2^k m - 1$, with m odd,
  - the binary representation of $n+1 = 2^k m$, ends in k 0s
  - the binary representation of n ends in k 1s
- $(3n+1)/2 = 3.2^{k-1} m - 1$
- The first loop transforms n into $3^k m - 1$, an even number
- k' is the largest power of two dividing $3^k m - 1$
- $\kappa(n) = (3^k m - 1) / 2^{k'}$

COMPUTING LONG COLLATZ STEPS

- The computation of long Collatz successors involves two loops
  - n → (3n+1)/2 repeated k times (or until the result is even)
  - n → n/2, repeated k' times (or until the result is odd)

- k is the number of ones at the right of the binary representation of n
- k' is the number of zeroes at the right of the binary representation of the apex
  - $27 = 11011_2$, k=2
  - Two iterations in the first loop: 27 → 41 → 62 = 11110_2 , k'=1
  - One iteration in the second loop: 62 → 31 = $\kappa(27)$

LEARNING LONG COLLATZ STEPS

- Encode n and $\kappa(n)$ as sequences of digits in some base B
  - We will play with B from 2 to 57
- Sample uniform odd integers up to 10^{12}
  - 5.10^{11} possible inputs, no memorization, overfit, overlap between train and test set, will take place
- Train transformers with 4 layers, dimension 256, and 8 attention heads
  - By minimizing the cross-entropy of predicted tokens (a glorified Hamming distance)
  - The model knows nothing about maths, it just learns to translate
- Code available on GitHub: f-charton/Int2Int
  - Runs on a 3GB GPU, in a few hours (3 minutes per epoch of 300k examples)

TRANSFORMERS KNOW MORE THAN THEY CAN TELL
LEARNING COLLATZ SEQUENCES (FC, A. Narayanan, ArXiv 2511:10811)

- A complex arithmetic function $n \rightarrow (3^k (n+1) / 2^k - 1) / 2^{k'}$, with k and k' functions of n
- A regression-like problem: over the 5.10^{11} possible inputs, there are 2.53×10^{11} different outcomes
- Learned like a translation task: the model is shown examples, tries to imitate them

- Can transformers, notoriously bad at arithmetic, handle this?

A WALK IN THE PARK

- The best models achieve an accuracy of 99.8% (about 250 errors our of 100 000 test examples)
  - Correct prediction = exact prediction of the successor, encoded in base B
- 99.8% accuracy for bases 24 and 32
- Bases 8, 12, 16, 24, 32, 36 and 48 achieve more than 99.5% accuracy

- Bases divisible by 8 or 12, Collatz-compatible?

EVEN AND ODD BASES

- Almost all even bases achieve 90+% accuracy
  - The larger the power of two, the better the accuracy
- Odd bases achieve accuracies below 83%,
  - As low as 25%

Accuracy	Bases
99.5+%	24, 16, 32, 36, 12, 48, 8
99 – 99.5%	4, 18, 56, 40
95 – 99%	6, 20, 28, 2, 52, 10, 44, 54
91 – 95%	42, 22, 30, 14, 50, 34
88 – 89%	26, 46, 33
81 – 82%	9, 45, 38, 15, 17
70 – 71%	49, 47, 21, 27, 51, 39, 31, 41, 23, 7, 25
59 – 66%	57 (65%), 43 (59%)
55 – 56%	13, 37, 19, 55, 29, 53, 35, 5
25 – 37%	11 (37%), 3 (25%)

Table 1: Model accuracy for different bases. Bases listed by decreasing accuracy.

QUANTIZED ACCURACIES

- Learning curves are "quantized"
- Models using different bases go through the same accuracy levels
- Accuracy changes in discrete steps
  - when the model "gets it"

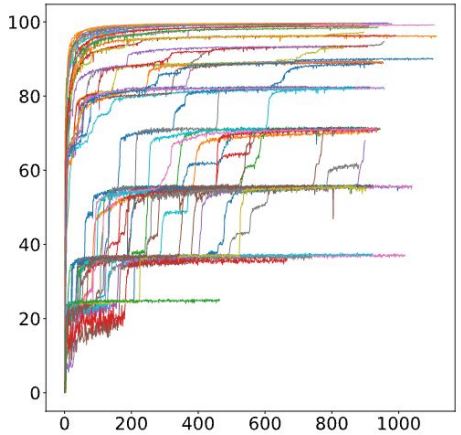

Figure 1: Learning curves for bases 2 to 57 (accuracy vs epochs of 300,000 examples).

QUANTIZED ACCURACIES

- Accuracy levels correspond to inputs with the same residual modulo 2^p
  - Inputs with the same binary ending
- Inputs in those classes are predicted with 99+% accuracy
- All other inputs with 1% or less
- These inputs form a "learning pattern"
  - 001 is learned first (25% accuracy)
  - 1011 is learned second (37% accuracy)
  - then 1101 and so on

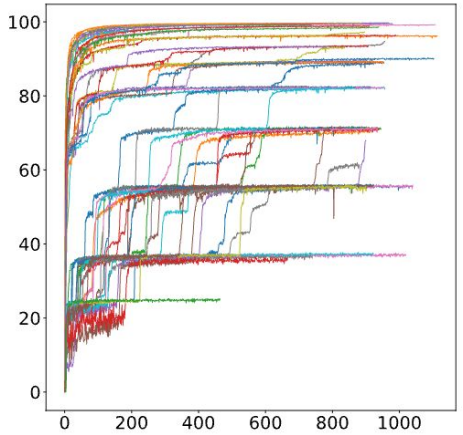

Figure 1: Learning curves for bases 2 to 57 (accuracy vs epochs of 300,000 examples).

THE LEARNING PATTERN

- The model learns binary input classes, in a particular order
  - 001, 1011, 1101, 00111 ...
  - Independently of the base
- Accuracy is either 100% or 0%
- The learning pattern accounts for the quantized accuracies

- A property of the problem, not of the representation

TWO MYSTERIES

- How do odd base models do this?
  - How does a base 27 model learn residuals mod 8 or 16?
  - Experimental fact: these transformers cannot learn to convert from one base into another (except in simplest cases)
- Why this particular sequence?

WHY THESE RESIDUAL CLASSES?

- We are learning two successive loops, of length k and k':
  - $n \longrightarrow (3n+1) / 2$
  - $n \longrightarrow n / 2$
- k can be guessed from the binary representation of n
  - it is the number of ones to the right of it
  - different values of k correspond to different binary residuals

- Could it be the same for k'?
  - can we tell k' from the binary representation of n?

WHY THE RESIDUAL CLASSES

Let k=1, the binary representation of n ends in 01, n=4m+1

After one pass in the first loop, we reach:

$(3n+1)/2 = (12m+4)/2 = 6m+2,$

an even number so $k' \geq 1$

After one $n \rightarrow n/2$ step, we reach 3m+1 and have two cases

- If m is even, n ends in 001, 3m+1 is odd, k'=1, we are done
- If m is odd, n ends in 101, 3m+1 = 6q+4, is even, $k' \geq 2$,
- One more $n \rightarrow n/2$ step gets us to 3q+2, and there are two more cases
  - If q is odd, i.e. n ends in 1101, 3q+2 is odd, k'=2, we have the successor
  - Else n ends in 0101, we reach 6p+2, $k' \geq 3$, then 3p+1, k'>2, we have been there before...

WHY THE RESIDUAL CLASSES?

For k=1, the binary sequence ends in 01

- If it ends in 001, k'=1
- If it ends in 1101, k'=2
- If it ends in 00101, k'=3 ...

Let S=...0101010101, B the binary sequence representing n, and M the longest right match between S and B

We know that B ends in 01, so the length of M is at least two.

- If it is two: B ends in 001: k'=1
- If it is three: B ends in 1101: k'=2
- And so on...

We can tell k and k' from the binary representation of n

WHY THE RESIDUAL CLASSES?

We could prove a theorem

- k can be read from the k+1 last bits of the binary representation of n (# of 1s)
- k' can be read from the next k' bits, longest match with a particular binary sequence
  - H_1 = 10, H_2 = 111000, H_3 = 1111011010000010010.
  - parities of 2^P mod 3^k, length Phi(k)

In the learning pattern, all models learn the long Collatz step for specific values of k and k',

- in increasing values of k+k'
- as a bonus, we learned useful facts about the distribution of k and k' (independent power laws)

WHY THE RESIDUAL CLASSES?

H_1=01,

H_2=011100,

H_3=01111011010000 1001

We define the infinite binary sequence $S=H_k^*01^k$,

k' is the longest right match between B and S, minus k

If n ends in 00110100001001 0111, we have k=3, and k'=14

THE LEARNING PATTERN

- Models learn k and k' in order of k+k'
  - First learn k=k'=1 (001), 25% accuracy
  - Then learn (k,k')= (2,1) (1011), 37.5% accuracy
  - Then learn (1,2), (3,1) (1101 and 00111), 55% accuracy
- The learning pattern is independent of the base
  - But even bases have an edge
- A property of the Collatz sequence

CAN WE DO BETTER? CHANGING INPUT DISTRIBUTION

- Since the most common cases are learned first, can we improve performance by oversampling the rarer cases?
  - Uniform, log-uniform distribution of k (k' still distributed as $1/2^{k'}$) in the training set
  - A regular distribution of k and k' in the test set
- Disappointing results
  - Imbalance matters!

Even bases	Uniform	Log-uniform	Baseline	Odd bases	Uniform	Log-uniform	Baseline
32	93.9	97.6	99.8	33	0	0.2	89.1
16	98.5	99.2	99.7	9	0.1	0.3	82.8
12	99.8	99.6	99.7	17	0	0	82.3
36	98.9	99.5	99.7	27	1.1	1.3	71.8
24	100	100	99.7	45	0	0	71.8
48	100	99.9	99.6	47	0	0	71.7
4	96.7	97.0	99.4	21	0	0	71.6
56	98.8	99.5	99.3	51	0	0	71.6
8	98.7	97.0	99.3	41	0	0	71.5
40	98.6	99.4	99.2	49	0	0	71.3
20	98.6	97.0	98.9	15	0	0	71.2
6	98.7	99.7	98.8	25	0	0	70.8
28	97.6	96.5	98.7	31	0	0	67.9
2	97.9	98.4	97.7	23	0	0	62.0
18	98.2	99.5	97.3	39.	0	0	56.3
44	94.5	98.2	96.5	43	0	0	56.2
52	94.9	94.7	96.4	13	0	0	56.2
42	90.6	95.8	94.7	29	0	0	56.1
30	90.9	93.0	93.7	57	0	0	56.1
10	87.0	90.7	93.6	53	0	0	56.1
14	86.1	95.5	93.2	19	0	0	56.0
54	89.6	97.3	90.3	35	0	0	55.9
22	77.8	89.0	89.7	5	0	0	55.8
50	49.3	87.2	89.5	37	0	0	37.7
34	85.7	88.0	89.2	55	0	0	37.6
26	84.8.	80.0	82.7	11	0	0	37.3
38	72.6	82.3	82.5	7	0	0	36.9
46	71.2	82.6	82.5	3	0	0	25.2

Table 4: Predicting the long Collatz steps with different training set distributions.

CAN WE DO BETTER? PREDICTING K AND K'

- Predicting k and k': turning the problem into a classification task
- An easier problem: instead of computing the long Collatz step, we only compute the loop length
- Amounts to finding common suffixes in binary representation

CAN WE DO BETTER? PREDICTING K AND K'

- Power of two bases achieve 100% accuracy
- Other even bases achieve lower performance than in the previous experiments
- Odd bases achieve 25% (always predict 1,1)

- The learning pattern is the same
- Somehow, a harder problem helps learn
  - More signal?

Bases	Accuracy	Correct predictions (k,k')
40	99.8%	all (k-k') for $k + k' \leq 12$, (12,1)
12	99.1%	all (k-k') for $k + k' \leq 10$, (10,1)
56	98.3%	all (k-k') for $k + k' \leq 9$, (9,1)
20, 44, 28, 52	90.1%	(1-1,2,3,4,5), (2-1,2,3,4), (3-1,2,3), (4-1,2), (5,1), (6,1)
36	88.8%	(1-1,2,3,4,5), (2-1,2,3,4), (3-1,(2,3), (4-1,2), (5-1), (6-1) (92%)
10	81.5%	(1,1-3), (1,4), (2,1-2), (2,3), (3,1), (3,2), (4,1), (5,1) (90%)
14, 50, 54	72%	(1-1,2,3), (2-1,2), (3-1), (4-1)
34, 22	71.5%	(1-1,2,3), (2-1,2), (3-1), (4-1) (95%)
38	70%	(1-1,2,3), (2-1,2), (3-1), (4-1) (90%)
42, 26, 18	69%	(1-1,2,3), (2-1,2), (3-1), (4-1) (75%)
46, 30	56.5%	(1-1,2), (2-1), (3-1)

Table 3: Predicting (k-k'). Model predictions for even bases, after 500 million examples. Accuracy below 98% in red.

CAN WE DO BETTER? CONVERT FROM ONE BASE INTO ANOTHER

- From 4, 12, 22, 36, 42, 56, 3, 9, 27, 11, 15, 13, 31, 43
- Into 2, 8, 32, 24

- 98% accuracy: base 4 to base 2,8 and 32, and base 12 to 24
- 47% accuracy: base 4 to 24
- 11% accuracy: base 36 into 24

- All others achieve less than 0.2% accuracy

UNDERSTANDING ERRORS

- Select a trained model
- Run it on 100,000 random test cases (most probably not seen during training, but in-domain)
- Investigate errors, by comparing wrong predictions to correct values

ERRORS IN OUR BEST MODELS

- Base 24: 243 errors out of 100,000 test examples,
- Most errors happen for large values of k,
  - 210 errors (out of 243) have $k \geq 8$,
  - the model always fails when $k \geq 11$
- The ratios between model prediction p and correct value t are close to 1
  - When wrong, models remain roughly right: 84% of errors are within 1% of the correct value
  - No hallucination

ERROR IN OUR BEST MODELS

- Model predictions and correct solutions, encoded in the base, share more than half their tokens
- Correct solution (base 24)
  [4, 21, 20, 6, 7, 3, 20, 2, 20, 1],
- Model prediction
  [4, 21, 20, 6, 7, 8, 8, 2, 20, 1]

	Errors	Prediction length	Correct tokens	Correct prefix	Correct suffix
Base 24	243	9.4	4.5	2.1	2.3
Base 32	265	8.9	4.2	1.9	2.1
Base 16	259	11.1	5.8	2.7	2.7
Base 8	479	14.5	8.0	3.5	3.6
Base 48	399	7.7	4.0	1.9	2.0
Base 36	287	7.7	3.3	1.5	1.6
Base 12	315	11.7	6.1	2.6	2.9

Table 2: Prediction errors for different good bases. Average length of predictions, number of tokens in agreement between prediction and target.

ERROR IN OTHER BASES - MORE QUANTIZATION

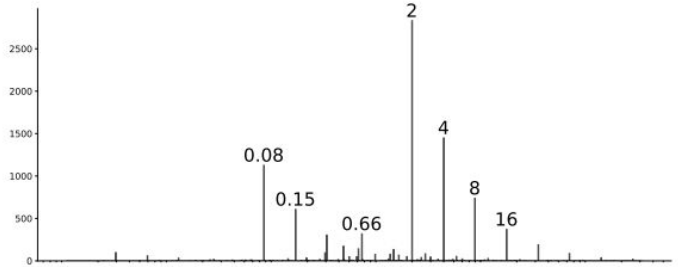

Figure 2: Distribution of p/t: base 26 (all model errors)

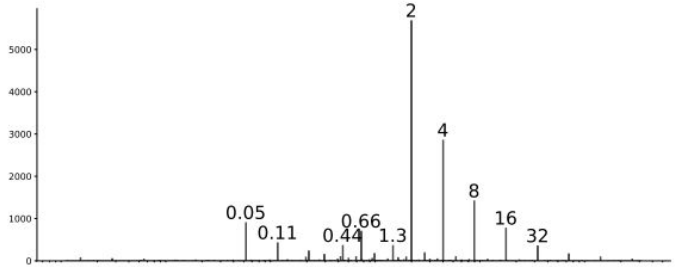

Figure 3: Distribution of p/t: base 38 (all model errors)

ERRORS IN ODD BASES - TOO MANY EVEN PREDICTIONS

- All outputs should be odd
- En even bases, less than 5% of model errors are even: errors have the correct parity
- In odd bases, in 85 to 90% of model errors, an even number is predicted, way more than chance!

ERRORS IN ODD BASES - POWER OF 2 ERRORS

- In odd bases, in 70 to 80% of errors, the model predicts the correct solution times a small power of two
  - The model underestimates k'
- These errors happen for specific values of k and k'
  - Values of k where the model can predict k'=1, but not all values of k'
  - All (99+%) inputs associated with large k' and that value of k end up as power of 2 errors
  - The model then uses for k' the largest value it can predict correctly

AN EXAMPLE IN BASE 27

- 71% accuracy, power of two errors account of 75.5% of model errors
- The model predicts correctly inputs with
  - k=1 and k'=1,2,3,
  - k=2 and k'=1, 2,
  - k=3 and 4, for k'=1
- Power of two errors happen for
  - For k=1, k'>3. The model uses k'=3 (largest correct value)
    - For k=1, k'=6, the model will predict 8 $\kappa(n)$, for k'=4 the model predicts 2 $\kappa(n)$
  - For k=2, k'>2. The model uses k'=2
  - For k=3 and 4, k'>1. The model uses k'=1

ERRORS IN EVEN BASES

- A similar quantization exists
  - But errors are "near power of two errors"
  - The model insists (wrongly) on predicting an odd integer
  - Data distribution idiosyncracies
- Sometimes, the model predicts t/B^k
  - the end of sequence token is predicted early
- These account for about 80% of model errors
- Happen when predicting large k', for values of k the model can predict
  - k' is predicted as the largest k' the model can predict

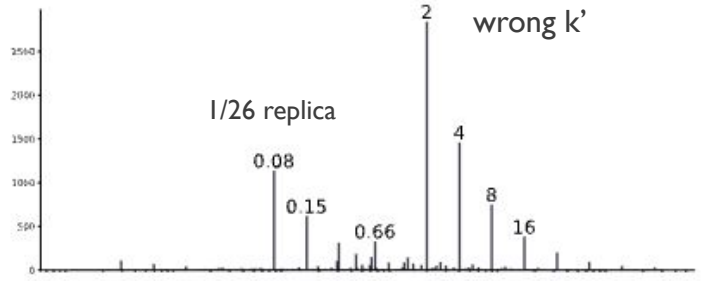

Figure 2: Distribution of p/t: base 26 (all model errors)

HARD ERRORS

- Remaining errors follow similar patterns:
  - k is underestimated by a few units
  - ratios r = p/t ~ $(2/3)^a$
  - sometimes divided by B

- These happen for large k that models cannot predict
  - k is then predicted as the largest k where the model can predict (k,1)
  - k' is predicted as 1

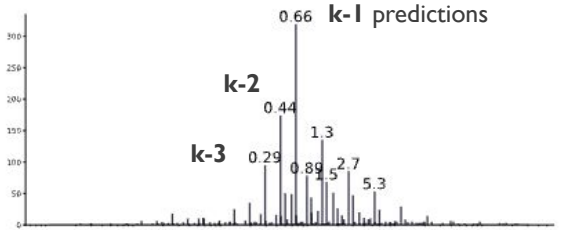

Figure 4: Distribution of p/t: base 26 (all model errors, excluding near power of two)

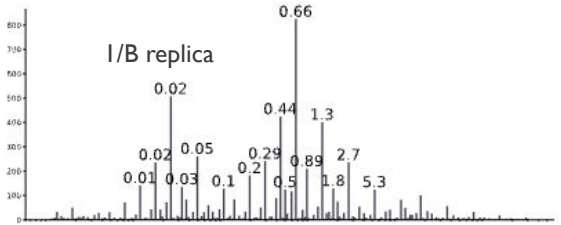

Figure 6: Distribution of p/t: base 27 (all model errors, excluding power of two)

A HIERARCHY OF ERRORS

- Models learn to classify inputs with particular values of k and k'
  - from the binary representation of n
  - this is why even bases perform better
- At any point in training, k is learned up to a value k_{max} and for each k, k' up to a value $k'_{k,max}$
- All input with $k \leq k_{max}$ and $k' \geq k'_{k,max}$ are predicted as $f(k,k'_{k,max})$,
- All inputs with $k > k_{max}$ are predicted as $f(k_{max},1)$

This accounts for more than 90% or errors, in all bases, throughout training

CONCLUSIONS

- We achieve surprisingly high performance: learning arithmetic is not always hard
- The model learns a mathematical property of the problem (the base 2 decomposition)
  - Even when the input is represented in an odd base
- Math transformers do not confabulate/hallucinate
  - A consequence of supervised learning
- Almost all model errors are explainable, the model knows more than it can tell

WHAT HAVE WE DONE?

- We trained models on a problem we can solve
  - varying model parameters: the base used for representation
- Noticed patterns, which we tried to interpret
  - a new understanding of the problem; the theorem on binary representation
- We learned something new about the problem
  -
- Analysis of error provides further insight on what it happening
  - Models are not black boxes, they can be reversed engineered
  - It takes some work