LEARNING COLLATZ SEQUENCES TRANSFORMERS KNOW MORE THAN THEY CAN TELL

FRANÇOIS CHARTON, ÉCOLE NATIONALE DES PONTS ET CHAUSSÉES



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)
    - Leverage a feature of addition 123 + 456 = 153 + 426
- 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

- End-to-end supervised training fares better
- Greatest common divisor (Charton 2024)
  - By default, models learn shortcuts: divisors of the representation base
    - In base 10 (or 100) the model learns GCD that are product of powers a 2 and 5
  - You can improve performance (a lot) by tweaking the training distribution (e.g. repeated examples)
- No hallucinations: errors are principled

(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.

EXACT ARITHMETIC ON A TRANSFORMER

- Moebius functions (Ellenberg)
  - The model takes shortcuts, learn small integer divisibility
- Modular arithmetic (addition, multiplication)
  - No obvious shortcuts
  - Training distribution adaptation is required
    - Repeated examples
    - Log-uniform operands

EXACT ARITHMETIC ON A TRANSFORMER

- On these arithmetic problems, transformers learn to
  - Cluster their inputs into classes
  - Predicts one outcome per class: the most common label

- Start with one unique class, all predicted the same
- Step-like learning curves, as new classes split off
- Outcome imbalance is needed (or outcomes are randomly selected)

- How general is this?
  - This applies to the pattern recognition setting: a few outcomes to be classified
  - What about functions with many outcomes? Aka the regression setting

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 number $n = 2^k m - 1$, m odd, $k>0$
  - 2^k is the largest power of two dividing n+1
  - The binary representation of n+1 ends in k zeroes (10^k)
  - The binary representation of n ends in k ones (01^k)
    - $27 = 4.7 - 1 = 11011_2$

- $3n+1 = 3(2^k m - 1)+1 = 3.2^k m-2$, an even number
- Dividing by two, we reach $3.2^{k-1} m - 1$
- Repeating k times, we reach $3^k m - 1$, an even number since m is odd
- Let $a = 3^k m - 1$ (the apex), divide it by 2 until on reaches an odd number $\kappa(n)$, ($k'>0$ times)

- I call these 2k+k' Collatz steps a **long Collatz step**

COMPUTING LONG COLLATZ STEPS

- The computation of long Collatz successors involves two loops
  - $n \rightarrow (3n+1)/2$ repeated k times (or until the result is even)
  - $n \rightarrow n/2$, 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: $27 \rightarrow 41 \rightarrow 62$
- $62 = 11110_2$, k'=1
- One iteration in the second loop: $62 \rightarrow 31 = \kappa(27)$

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
- Train transformers with 4 layers, dimension 256, and 8 attention heads
- 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
- Test trained models on 100 000 random odd integers

- 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

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%

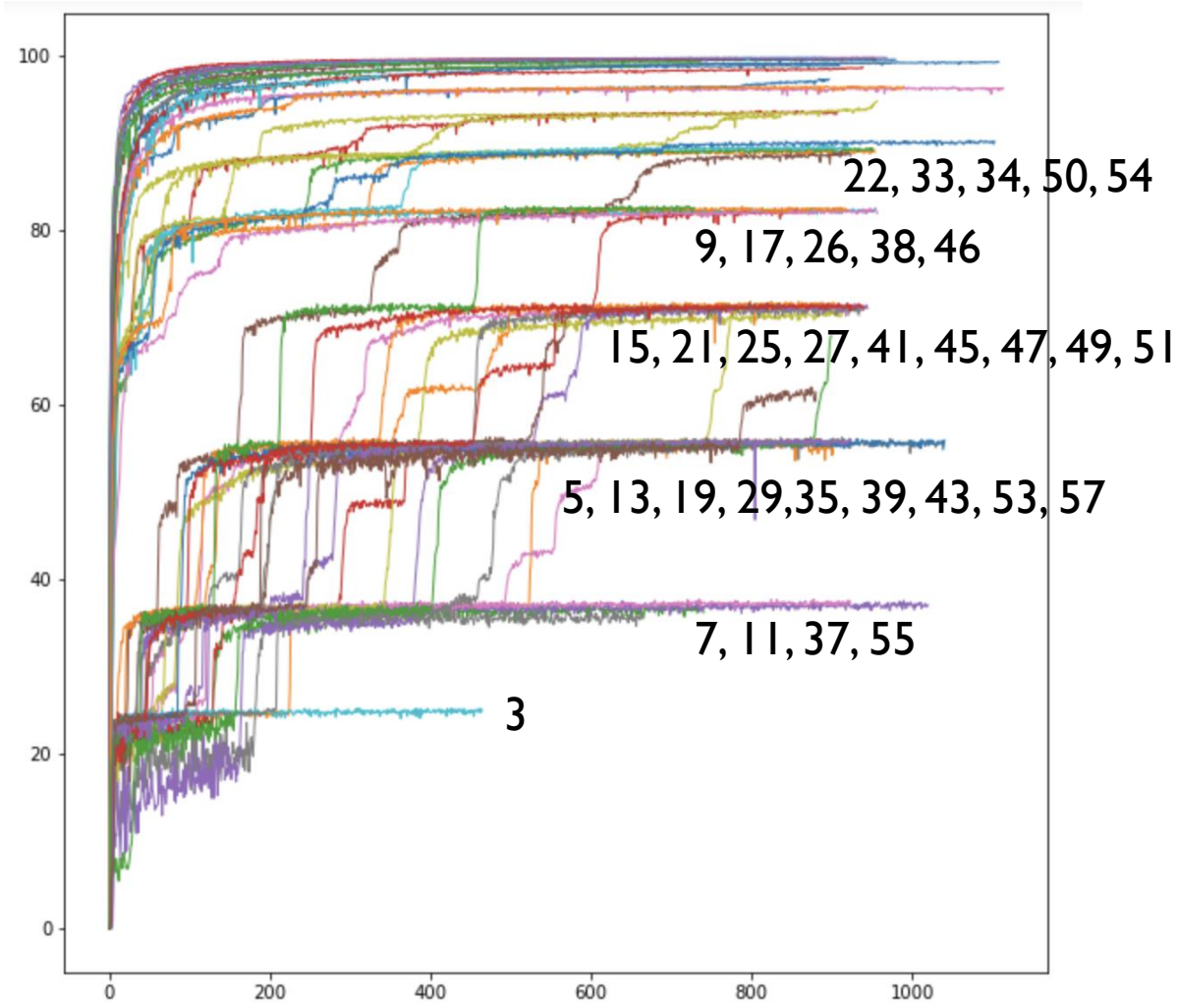
Accuracy	Bases
99.5+%	24, 32, 16, 12, 36, 48, 8
99 – 99.5%	4, 56, 18, 40, 6
95 – 99%	20, 28, 2, 52, 10, 44, 54
89 – 95%	42, 22, 30, 14, 50, 34, 26, 33, 46
82 – 83%	9, 45, 38, 17, 15
70 – 71%	49, 27, 47, 21, 51, 41, 31, 39, 23, 25, 7
59 – 65%	57, 43
55 – 57 %	13, 19, 29, 53, 37, 35, 55, 5
37%	11
25%	3

Table 1: Model accuracy for different bases. Bases listed by decreasing accuracy (best of two models).

QUANTIZED ACCURACIES

- Accuracy is learned by steps, a common phenomenon in math tranformers
- The steps take specific values, no matter the base
  - Same values for even and odd bases
  - Better bases learn faster, but use the same "quantum steps"

- A property of the problem, not of the representation
- Are we learning something about Collatz?



UNDERSTANDING CORRECT PREDICTIONS

- Run a trained model on a test sample of 100,000 examples (odd integers)
- Measure model performance on different classes of test examples
  - Does the model succeed and fail on particular input?

- Intuition: k, the number of first loops iterations ($n \rightarrow (3n+1)/2$) depends on the binary ending of n
  - residuals mod 2^p
  - Since n is odd, 2^{p-1} possible (odd) residuals: 1, 3, 5, 7 mod 8

CORRECT PREDICTIONS

- Transformer learn to predict particular residual classes
- For all bases, for p large enough, model accuracy is over 99% on some residuals mod 2^p, and less than 1% on all others

- For bases 7, 11, 37 and 55 (37% accuracy), the model achieves
  - 99+% accuracy for all inputs with residuals : 1 modulo 8, and 11 modulo 16
    - Binary representations ending in 001 (25% of inputs) and 1011 (12.5% of inputs)
  - Less than 1% accuracy for all other residuals
  - A theoretical accuracy of 37.5%

CORRECT PREDICTIONS

- For bases 5, 13, 19, 25, 29, 35... (55.5% accuracy) the model predicts
  - with 99+% accuracy, inputs with residuals 1 modulo 8, 11 and 13 mod 16, and 7 mod 32
    - Ending in 001, 1011, 1101 and 00111
  - All others with less than 1% accuracy
- Models achieving an accuracy of 71% (e.g. base 21) predict the above, plus 5 and 19 modulo 32
  - Ending in 00101 and 10011

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 the simplest cases)
- Why this particular sequence?

BINARY REPRESENTATIONS ENCODE LOOP LENGTHS (K AND K')

- We are learning an algorithm that features two loops,
  - $n \rightarrow (3n+1)/2$ repeated k times
  - $n \rightarrow n/2$ repeated k' times

- k can be deduced from the binary representation of n
  - Number of ones to the right (k=2 for $27=11011_2$)
- Can we tell k' from the binary representation of n?

BINARY REPRESENTATIONS ENCODE LOOP LENGTHS (K AND K')

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

Perform $n \rightarrow (3n+1)/2$ once: $(3n+1)/2 = (12m+4)/2 = 6m+2$ (even)

Perform $n \rightarrow n/2$ once $3m+1$

We now have two cases

- If m is even, 3m+1 is odd, k'=1, we are done
  - n = 4m+1, m=2p, so n=8p+1
  - The binary representation of n ends in 001
- If m is odd, n ends in 101, 3m+1 = 6p+4, is even, k' ≥2,
- Perform $n \rightarrow n/2$ once more, we reach 3p+2, and there are two cases
  - If p is odd, i.e. n ends in 1101, 3p+2 is odd, k'=2, we are done
  - If p is even, n ends in 0101, we reach 6p+2, k' ≥3, then 3p+1, we have been there before...

BINARY REPRESENTATIONS ENCODE LOOP LENGTHS (K AND K')

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 ...

Consider the infinite binary sequence S=...0101010101, let B be 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...

k and k' can be deduced from B, k by counting the rightmost ones, k' by comparing it to S (it it the length of the longest match, minus one)

WHAT ABOUT LARGER K?

- B ends in 01^k, $n=2^{k+1}m+2^k-1$,
- After the k instances of the $n \rightarrow (3n+1)/2$ transformation), we reach $3^k\,2m+3^k-1$, an even number
- After one division by 2, we reach $3^k m+(3^k-1)/2$, $k'=1$

- We now have a sequence of terms of the the form $3^k m+h$ ($h<3^k$), starting with $h=(3^k-1)/2$,
  - k' is set at 1, S at 01^k
- For each term, we have four cases
  - If h is even and m is odd, $3^k m+h$ is odd, B ends in 1S, the process ends, and the value of k' is the number of divisions by two
  - If h is odd and m is even, B ends in 0S, the process ends, and the value of k' is the number of divisions by two
  - If h and m are even, $3^k m+h$ is even, B ends in 0S, we replace h by h/2, S by 0S and increment k' by 1
  - If h and m are odd, B ends in 1S, we replace h by $(h+3^k)/2$ (reducing m by one), S by 1S, and increment k' by one

WHAT ABOUT LARGER K?

- This creates a sequence of h, and an associated binary sequence corresponding to the parities of h
- Since $h<3^k$, this sequence must loop, in fact it is the sequence obtained by multiplying by ½ mod 3^k, starting with $(3^k-1)/2$ (or the reverse of the sequence of powers of 2 mod 3^k)
  - It has $\Phi(3^k)=3^k-3^{k-1}$ elements
- Taking the parities of the successive h, and writing them from right to left, one can form a binary sequence H_k
  - $H_1=01$
  - $H_2=011100$
  - $H_3=01111011010000 1001$

WHAT ABOUT LARGER K?

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

Let B be the binary sequence representing n

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

If n ends in 00110100001001010111, we have k=3, and k'=14

WHY THE LEARNING PATTERN?

- We can tell k and k' from the binary representation of n
- To each pair (k,k') we associate a binary ending $B_{k,k'}$
  - Which appears with probability $1/2^{k+k'}$
- The learning pattern corresponds to the sequence of k and k', per decreasing probability
  - The models first learn k=k'=1 (B=001), achieving 25% accuracy
  - Then it learns (k,k')= (2,1) (B=1011), and achieves 37.5% accuracy
  - Then it learns(1,2) and (3,1) (B=1101 and 00111), and achieves 55% accuracy
- A property of the Collatz sequence, independent of the base
  - But some bases learn faster

WHAT HAVE WE LEARNED?

- Models learn specific instances of (k,k'), small values of k+k', more common, are learned first
- For each pair of values, the corresponding mapping $n \rightarrow (3^k (n+1) / 2^k - 1) / 2^{k'}$ is either learned perfectly, or not at all.
- The (k,k') pair corresponds to a special suffix of the binary representation of n
  - How do odd base models figure this?

- The choice of a representation base may accelerate learning, but it does not change the learning pattern

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

DOES THE MODEL HALLUCINATE?

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 $k \geq 11$
  - k=9 and k'=1: 41 correct / 14 wrong
  - k=9 and k'=2: 5 correct / 26 wrong
  - k=9 and k'>2; all model predictions are wrong

ERRORS IN OUR BEST MODELS

- But all wrong predictions are very close to the correct values, in base 24
  - 111 errors (46%) are within 0.1% of the correct value,
  - 204 (84%) within 1%,
  - 225 (93%) within 3%.
- In base 32
  - 47% within 0.1%
  - 93% within 1%
  - 98% within 3%

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]

- Also true when predicting apex

	Errors	Prediction length	Correct tokens	Correct prefix	Correct suffix
Base 24					
Prediction	243	9.4	4.5	2.1	2.3
Apex	243	10.1	5.6	2.1	3.2
Base 32					
Prediction	265	8.9	4.2	1.9	2.1
Apex	265	9.4	4.6	1.9	2.4
Base 16					
Prediction	259	11.1	5.8	2.7	2.7
Apex	259	11.6	6.1	2.7	3.1
Base 8					
Prediction	479	14.5	8.0	3.5	3.6
Apex	479	15.1	8.3	3.5	4.2
Base 48					
Prediction	399	7.7	4.0	1.9	2.0
Apex	399	8.3	4.7	1.8	2.8
Base 36					
Prediction	287	7.7	3.3	1.5	1.6
Apex	287	8.7	4.6	1.6	2.7
Base 12					
Prediction	315	11.7	6.1	2.6	2.9
Apex	315	12.7	7.5	2.6	4.2

Table 5: Prediction errors for different good bases.

ERRORS IN OUR BEST MODELS

- The models do not confabulate
- They seem to "almost know" the correct solution, but fail when decoding
- The error seems to lie in the first loop
  - The apex is slightly wrong,
  - k' and the divisions by 2 are correct

- The model does even better than its accuracy suggests

EVEN PREDICTIONS

- All inputs are odd, and outputs should be odd
- In even bases, less than 0.8% of model predictions are even (almost zero for the best 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!

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 apex is correctly predicted, but the model underestimates k', and fails to "divide until odd'

- Power of two 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'
  - The model then uses for k' the largest correct value

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 κ(n), for k'=4 the model predicts 2 κ(n)
  - For k=2, k'>2. The model uses k'=2
  - For k=3 and 4, k'>1. The model uses k'=1

POWER OF TWO ERRORS – BASE 35

- Correct predictions up to
  - k=1 k'=2
  - k=2 and 3 k'=1
- For k up to three, larger values of k' are power of two errors
  - For k=1, k'=2 is assumed
  - For k=2 and 3, k'=1 is assumed
- For k>3, no power of two errors: "hard errors"

A HIERARCHY OF ERRORS

- Smallest values of k and k' are predicted correctly
- For odd bases, power of two errors happen for values of k where the model is correct up to a certain k'_{max}
  - Power of two errors use $k'=k'_{max}$
- For even bases, some power of two errors, but no deterministic pattern

- For larger k (and k' for even bases), "hard errors", the model gets the apex a little wrong, then applies division by two

AN INTERPRETABILITY EDGE TO ODD BASES

- For B=17 and 38, about 80% accuracy,
- Correct predictions up to (1,4), (2,3), (3,2), (4,1), (5,1)

- For B=17, all larger values of k', for k up to 5, are power of two errors
- For B=38, a fraction of them are power of two errors, less interpretable

HARD (NON-POW2) ERRORS

- For the best bases, predicted values are very close to correct values
- Let us study the distribution of the ratios
- Almost all have the form $2^n \cdot 3^p \cdot B^q$ (B the base, 1% tolerance)

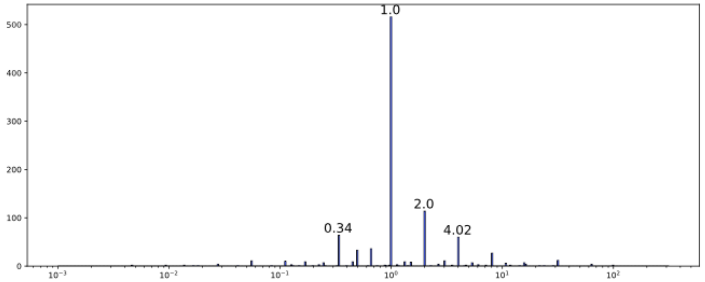


Figure 4: Base 6

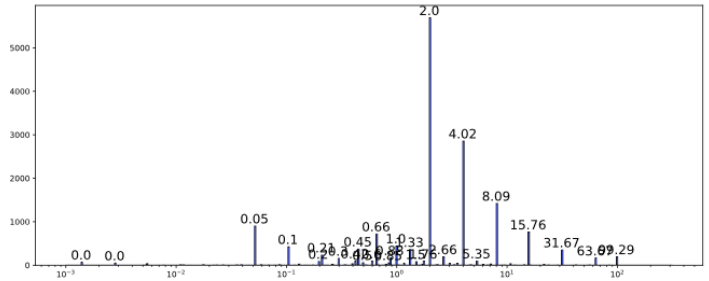


Figure 5: Base 38

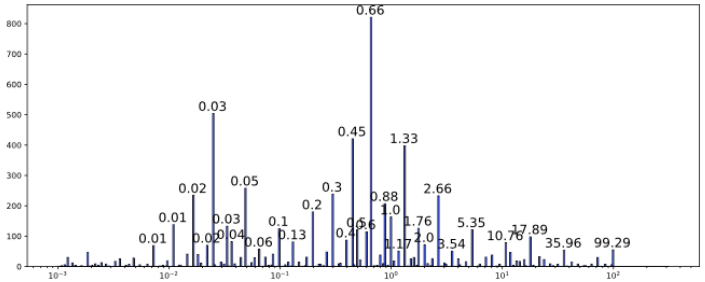


Figure 6: Base 27

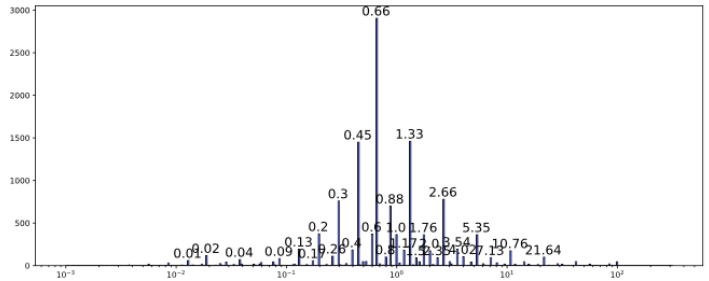


Figure 7: Base 35

HARD ERRORS – BASE 6

- Out of 1025 hard errors, the ratio of predicted to correct value is (within 1%)
  - 1 in 515 cases (50%)
  - 2^p in 274 cases (27%)
  - $2^p/3$ in 139 (14%)
  - $2^p/9$ in 47 (5%)
  - 3.2^p in 26 (2.5%)
  - $2^p/27$ in 9 (1%)
  - 9.2^p in 25 (0.5%)
- Accounting for 1015 (99%) of hard errors

HARD ERRORS IN BASE 6

- Ratio 1: predicted [2, 1, 1, 5, 3, 1, 3, 0, 2, 2, 1, 2, 3, 2, 4, 5] correct [2, 1, 1, 5, 3, 0, 5, 4, 0, 5, 1, 2, 3, 2, 4, 5]
  - Predicted [1, 0, 2, 4, 3, 4, 4, 4, 0, 3, 3, 1, 1, 1, 0, 1] correct [1, 0, 2, 4, 3, 4, 1, 1, 5, 0, 3, 1, 1, 1, 0, 1]
- Ratio 6 predicted [1, 4, 2, 5, 0, 0, 5, 0, 1, 0, 0, 5, 1, 3, 5] correct [1, 4, 2, 5, 0, 0, 5, 0, 0, 5, 5, 5, 2, 5]
  - Predicted [1, 0, 0, 1, 4, 5, 0, 4, 4, 4, 4, 2, 2, 2, 1] correct [1, 0, 0, 1, 4, 5, 0, 4, 4, 4, 4, 3, 4, 5]
- Ratio 1/6: predicted [4, 5, 4, 1, 2, 0, 1, 1, 5, 2, 5, 1, 5] [4, 5, 4, 1, 2, 0, 1, 5, 1, 4, 1, 4, 1, 1]

HARD ERRORS – BASE 17

- Out of 4444 hard errors, the ratio of predicted to correct value is (within 1%)
  - $2^p/3$ in 1198 cases (27%)
  - $2^p/17^2$ in 605 (14%)
  - 2^p in 512 (12%)
  - $2^p/(3.17^2)$ in 332 (7%)
  - 1 in 323 (7%)
  - $2^p.3$ in 294 (7%)
  - $2^p/(3.17)$ in 293 (7%)
  - $2^p/17$ in 262 (6%)
  - $2^p/(9.17)$ in 167 (4%)
  - $2^p.3.17$ in 154 (3%)
  - $2^p.17$ in 95 (2%)
  - $2^p.17/3$ in 91 (2%)
  - $2^p.9$ in 63 (1%)
- Accounting for 4389 (99%) of hard errors

HARD ERRORS IN BASE 17

- Ratio $1/17^2$ predicted [1, 5, 2, 12, 3, 3, 10, 4], correct [1, 5, 2, 12, 3, 3, 10, 4, 1, 4]
  - Predicted [2, 8, 0, 2, 0, 16, 14, 11], correct [2, 8, 0, 2, 0, 16, 14, 13, 3, 11]
  - Predicted [2, 6, 6, 2, 5, 14, 10, 15], correct [2, 6, 6, 2, 5, 14, 10, 15, 16, 15]
- Ratio 1/17 predicted [12, 16, 13, 15, 10, 0, 8, 9], correct [12, 16, 13, 15, 10, 0, 8, 9, 8]
  - [3, 6, 10, 3, 10, 4, 7, 15, 13] [3, 6, 10, 3, 10, 4, 7, 15, 14, 11]
- Ratio 1 predicted [10, 12, 15, 15, 16, 3, 10, 2, 8, 5], correct [10, 12, 15, 16, 0, 3, 10, 2, 8, 5]
  - predicted [2, 8, 3, 11, 13, 8, 9, 3, 1, 5] correct [2, 8, 3, 11, 13, 8, 9, 1, 13, 7]
  - Predicted [1, 2, 5, 8, 12, 4, 8, 4, 7, 9] correct [1, 2, 5, 8, 12, 9, 3, 13, 7, 9]

HARD ERRORS – BASE 35

- Out of 13406 hard errors, the ratio of predicted to correct value is (within 1%)
  - $2^p/3$ in 5886 cases (44%)
  - $2^p.3.35^2$ in 2888 (22%)
  - $2^p.35^2$ in 1480 (12%)
  - $2^p.35^2/3$ in 748 (6%)
  - 2^p in 411 (3%)
  - $2^p.35^2/9$ in 410 (3%)
  - 1 in 365 (3%)
  - $2^p/(3.35)$ in 264 (2%)
  - $2^p.35^2/27$ in 204 (1.5%)
  - $2^p.3$ in198 (1.5%)
  - $2^p.3.35$ in 110 (1%)
  - $2^p/35$ in 92 (1%)
- Accounting for 13056 (97%) of hard errors

TRANSFORMERS KNOW MORE THAN THEY CAN TELL

- Errors can be explained
  - Close to the correct values
  - Related by a problem-specific transformation
    - Factors 3/2: one more (or less) iteration in the first loop,
    - Factors 2: one more or less iteration in the second loop
  - Errors in the End of sequence token
- Hypothesis: imperfect decoding
- In latent space, we are very close

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

FUTURE GOALS FOR ARITHMETIC TRANSFORMERS

- Better decoders
- Better understanding of latent space representations
- Use architectures that help create better latent space
  - Contrastive elements in the loss
  - Multitask (pre-)training