Mechanistic Interpretability and Implementability of Irreducible Integer Identifiers

Noah Syrkis & Anders Søgaard

Abstract

Deep learning models are increasingly obiquitous, while their interpretability is inherently opaque. This paper explores the mechanistic interpretability of deep learning models trained to solve problems related to prime numbers.

Introduction

Deep learning models are increasingly obiquitous, while the explanations for their predictions remain opaque.

Background

As the ubiquity of deep learning models is a recent phenomenon, the interpretability of these models is a relatively new field. Definitions of interpretability vary [1]. One qualifier to interpretability is *mechanistic* [2], which posits that a deep learning model that has generalised is a pourly written, but probably clean algorithm. "pourly written" means subsymbolic, and "probably clean" means that we could express the algorithm in a more human readable form, such as pseudocode (or Python).

G. Weiss, Y. Goldberg, and E. Yahav [3] present the coding language RASP, in which incorporates the architectural contraints of the transformer model into the language itself. This forces the programmer to be "thinking like a transformer" (which is the title of their paper). The multi layer perception (MLP) can be thought of as performing a map, performing a function on every element of a set. The attention mechanism can be thought of as a reduce (or map-reduce) operation, where the attention mechanism is a function that takes a set of elements and

Primber numbers, being divisible by only 1 and themselves, are a fundamental concept of number theory, about which the questions one might ask ranges from the relatively easily proven: "Are there infinitely many prime numbers?" (yes) to the still unsolved: "Is every even number the sum of two prime numbers?" (Goldbach's Conjecture). The breadth in

difficulty of these questions, and the fact primes are easily generated makes them a good resevoir of tasks for mechanistic interpretability of deep learning models.

Mechanistic Interpretability

Mechanistic interpretability is the ability to understand the inner workings of a model, and to explain why it makes the predictions it does. [2] perform a mechanistic interpretability analysis of a deep learning model trained to perform modular addition, trained to predict the remainder of the sum of two numbers $\bmod p$, p being a prime number. They find that the model learns to perform the task through a discrete Fourier transform, and that the model is able to generalize.

Though this is a single example Fourier transform learned by a model, the rotational symmetry of the Fourier transform is part of what motivates the polar plots used in this paper. The reader is asked to pose the questions "if I were a deep learning model, how would I learn to generate prime numbers?", and "How could I detect prime numbers, by spiraling the natural numbers?"

Prime Numbers

Indeed, a number is prime if it is divisible only by 1 and itself. However, we do not have to test the divisibility of every number beteween 1 and the number in question to determine if it is prime. The Sieve of Eratosthenes is an ancient algorithm for finding all primes up to a given limit, and it teaches us that testing if the given number n is a multiple of any prime number less than $\sqrt{n}$ is sufficient. Thus, a prime number is a number n for which $\forall p \in \mathbb{P}$ such that $p < \sqrt{n}$ $n \bmod p \neq 0$.

$$\mathbb{P} = \{p \in \mathbb{N} \mid p > 1, \forall p' \in \mathbb{P} < \sqrt{p}, \% \bmod p' \neq 0\}$$

The task at hand, thus becomes similar to [2]'s, in that we are conditioning on modularity. An algorithm that could be learned by a two layer transformer model might be to first create a function evaluating if $n \bmod p \neq 0$. This function could then be applied in later layers to the primes less than $\sqrt{n}$. Then, how does one test that $n \bmod p \neq 0$ using a rotational mindset (Discrete Fourier Transform).

Formally the set of Prime Numbers $\mathbb{P}$ is defined as the set of positive integers greater than 1 that have no positive divisors. $\mathbb{P}$ is thus an infinite subset of the natural numbers $\mathbb{N}$. Their distribution is not uniform, though they are asymptotically distributed according

to the Prime Number Theorem. For any given $n \in \mathbb{N}$, there is about $\frac{n}{\log(n)}$ prime numbers less than n.

To get a visual sense of the distribution of prime numbers, we can can map the prime numbers to a polar coordinate system, where both the angle and the radius are the prime number, similar to the Ulam Spiral. nats_and_sixes show the natural numbers, the multiples of six less than 1024 in this system.

Similarly, Figure 1 shows the first 2048 prime numbers minus the first 1024 prime numbers. The random nature of the prime numbers is evident in the figure, with no clear pattern emerging.

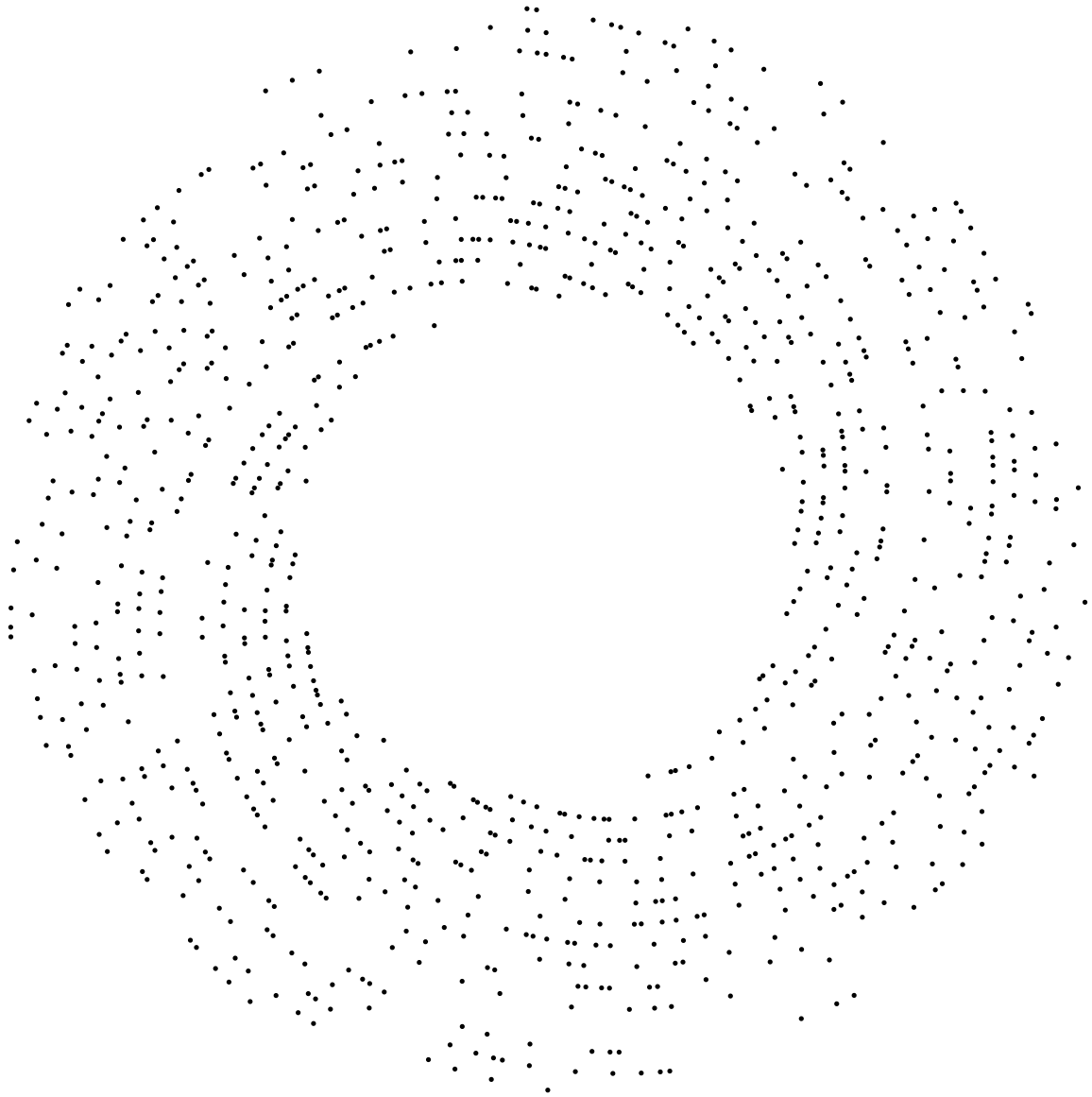

Figure 1: The first 2048 primes minus the first 1024 primes.

Sieve of Eratosthenes

The primality of a given number n can be determined by testing if it is divisible by any of the prime numbers less than $\sqrt{n}$. This is the basis of the Sieve of Eratosthenes, which is an ancient algorithm for finding all prime numbers up to a given limit. The algorithm works by iteratively marking the multiples of each prime number starting from 2, and then finding the next number that is not marked as a multiple of a prime number, which is the next prime number. The algorithm is efficient, with a time complexity of $O(n \log \log n)$. Relating it to our polar plots, the Sieve of Eratosthenes can be seen as first plotting all

naturak numbers up to a limit n, and then removing the multiples of the prime numbers less than $\sqrt{n}$.

Zeolite of Eratosthenes
The Zeolite of Eratosthenes is a variant of the Sieve of Eratosthenes, in which the multiples of the prime numbers are not filtered deterministically, but rather probabilistically —in inapropriately—by using a deep learning model.

RASP | Thinking Like a Transformer
[3] presents the language RASP, which forces the user to think like a transformer. RASP is turing complete, so it can indeed be bent into implementing algorithms a Transformer is unlikely to learn. To clarify the search space of our reverse enginerring tasks, I first implement the Sieve of Eratosthenes in RASP.

Then I create my own RASP based prime detecting algorithm, bending over backwards to introduce rotational symmetry.

Methods

The paper uses a JAX implementation of a two layer transformer model, with a hidden size of 128 and 8 heads, as per [2]. As prime classification is considerably more complex than modular addition, target vector y rather than being a single one-hot number indicating the primately of a given sample, is a vector of length $\sqrt{n}+1$, where the ith element is 1 if the sample is divisible by the ith prime number, and 0 otherwise, with the $\sqrt{n}$-th element being 1 if the sample is prime, in which case all other elements are 0.

Data

x (base 10)	x (base 2)	$[y_2\ y_3\ y_p]$
"04"	"0100"	[1 0 0]
"05"	"0101"	[0 0 1]
"06"	"0110"	[1 1 0]
"07"	"0111"	[0 0 1]
"08"	"1000"	[1 0 0]
"09"	"1001"	[0 1 0]

Table 1: Base 10 and 2 dataset for $n = 6$, note $\sqrt{9} = 3$, so y tests for multiples of 2 and 3, along with primality.

In the following, X denotes the input data, and Y the target data, with x and y denoting individual samples. X for a dataset of size n was constructed by creating the vector $[2..n+1]$. This vector was than converted to the desired number system. Y was constructed by first querying all prime numbers less than or equal to $n+1$, creating a one hot vector for each sample, in X indicating primality. Y was further augmented by $\sqrt{n}$ vectors, each indicating divisibility by the ith prime number up to $\sqrt{n}$. thus, sum of all y of primes is 1, and the sum of all y of non-primes is $>= 1$ (one if if is divisible by a single other number).

Note that the row sum of Y can be thought of as a sort of measure of how "close" to being prime a given number is. For example 20 is very much not a prime since it is a mulitple of 2, 4, 5 and 10, while 51 (in base 10) looks like a prime (SITE SCOTT ALEXANDER) but can in fact be factorized into 3 and 17.

Y thus includes information about why a given not is not prime. The inclusion of these extra tasks also allows for interpretability to be on simpler tasks, by training the model on the simpler tasks first, and then training on the more complex task.

This allows for comparison of how the model solves the different tasks, when learning them in isolation versus in conjunction.

Task

The main task is to predict the primality of a given number, with the additional tasks being to predict divisibility by the first $\sqrt{n}$ prime numbers as well. A second task dimension could be added, in which the remainder of the division by the first $\sqrt{n}$ prime numbers too is predicted.

Model

A [[4]] model is used (with the normalization layer removed.).

Training

For each of the $\sqrt{n}+1$ tasks, the focal loss, f1, accuracy, and precision are calculated every epoch. The frequency of a positive samples with in task i is used as the weight for the focal loss during training. Furthermore, a one-hot vector is used to mask tasks, so as to shield the model from a particular signal during training.

Date	°No	Description
24/01/03	813	Filtered pool
24/01/03	477	to sec. regimen
24/01/11	051	Cycled substrate

Table 2: Example table.

Evaluation

The purpose of the paper is not to train a good model, but to understand how the model works. A model training is thus considered to be a part of the methods section and not the results section.

Results

Discussion

The work here presented juxtaposes the body of work connecting large language models to formal theorem proving.

Future Work

This paper has shown an example of how transformer models can detect divisibility (and the lack there of). Divisibility is indeed an essential property of prime numbers, but it is not the only one. Future work could include training a transformer model to perform prime factorization, or to classify rare prime numbers, such as twin primes or Mersenne primes, the latter problems in particular being difficult as the sequence lengths of the representatiuons of the numbers involved become large.

The code of this paper is available at github.com/syrkis/miiii. It can be installed with `pip install miiiii`, and reprdocued with `miiii reproduce thesis`.

Conclusion

Bibliography

[1] Z. C. Lipton, "The Mythos of Model Interpretability: In machine learning, the concept of interpretability is both important and slippery.," *Queue*, vol. 16, no. 3, pp. 31–57, Jun. 2018, doi: 10.1145/3236386.3241340.

[2] N. Nanda, L. Chan, T. Lieberum, J. Smith, and J. Steinhardt, "Progress measures for grokking via mechanistic interpretability." Accessed: Dec. 16, 2023. [Online]. Available: http://arxiv.org/abs/2301.05217

[3] G. Weiss, Y. Goldberg, and E. Yahav, "Thinking Like Transformers." Accessed: Nov. 23, 2023. [Online]. Available: http://arxiv.org/abs/2106.06981

[4] A. Vaswani *et al.*, "Attention Is All You Need." Accessed: Jun. 11, 2023. [Online]. Available: http://arxiv.org/abs/1706.03762