A Mechanistic Analysis of Counting in Distil-GPT2

Carter Teplica

August 23, 2023

TL;DR: I did a mechanistic analysis of counting in Distil-GPT2, and found evidence that the model uses frequency-space number representations, particularly in base 10, in order to implement the task compactly.

This project was completed at UChicago's XLab. Many thanks for their advice and insight to Yibo Jiang, my mentor; Zach Rudolph, who coordinated the program; Victor Veitch, who provided feedback; and my fellow participants.

Recent research has found evidence in a variety of contexts that transformer models favor frequency-space representations for scalar quantities. Counting—a simple and important task for modeling natural language—provides a potentially fruitful context for evaluating the relevance of frequency-space representations to numerical reasoning in transformers "in the wild". In a mechanistic analysis of the task in Distil-GPT2, a 6-layer language model, we find a small set of components responsible for most of the model's performance on single-token counting. We also find hints of an additional set of components which are important for detecting this integer-sequence context. In both components, we find MLP neurons which exhibit base-10 responses to number inputs.

Background: periodic representations in transformer models

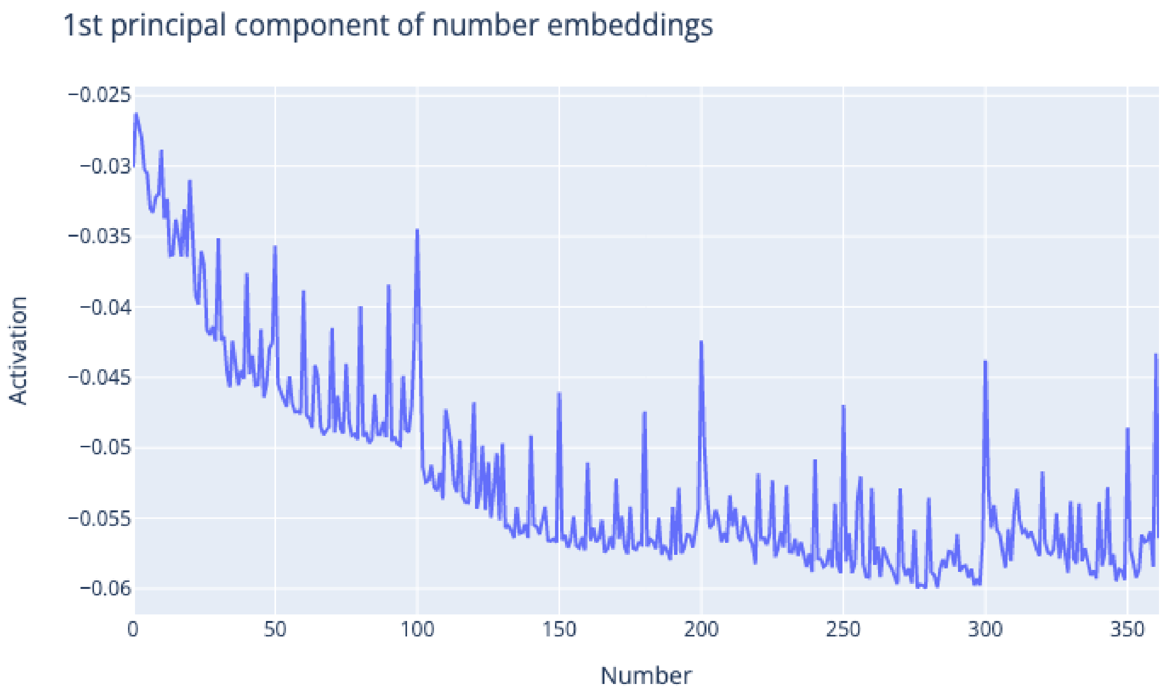
Recent mechanistic interpretability research has found a variety of contexts in which transformer models represent scalar quantities periodically or in frequency space. In a study on toy models, Neel Nanda and collaborators[1] find that very small models trained on a modular addition task initially memorize sums, but then learn an algorithm in which numbers are represented trigonometrically (i.e., the token n is represented by dimensions corresponding to $\sin(2\pi kn)$ and $\cos(2\pi kn)$, for a set of ~ 5 frequencies k). In "real" language models, similar structures appear to be favored for representing positional information: the original transformer architecture uses hard-coded sinusoidal representations[2] and the learned positional representations in GPT-2 are also strongly periodic, with representations that appear approximately as triangle waves in the standard basis[3] and as smooth helices under PCA.[4] In a recent research walkthrough analyzing addition in a large model, Neel Nanda[5] finds neurons with periodic behavior, including mod-2, mod-3, and mod-10 neurons.

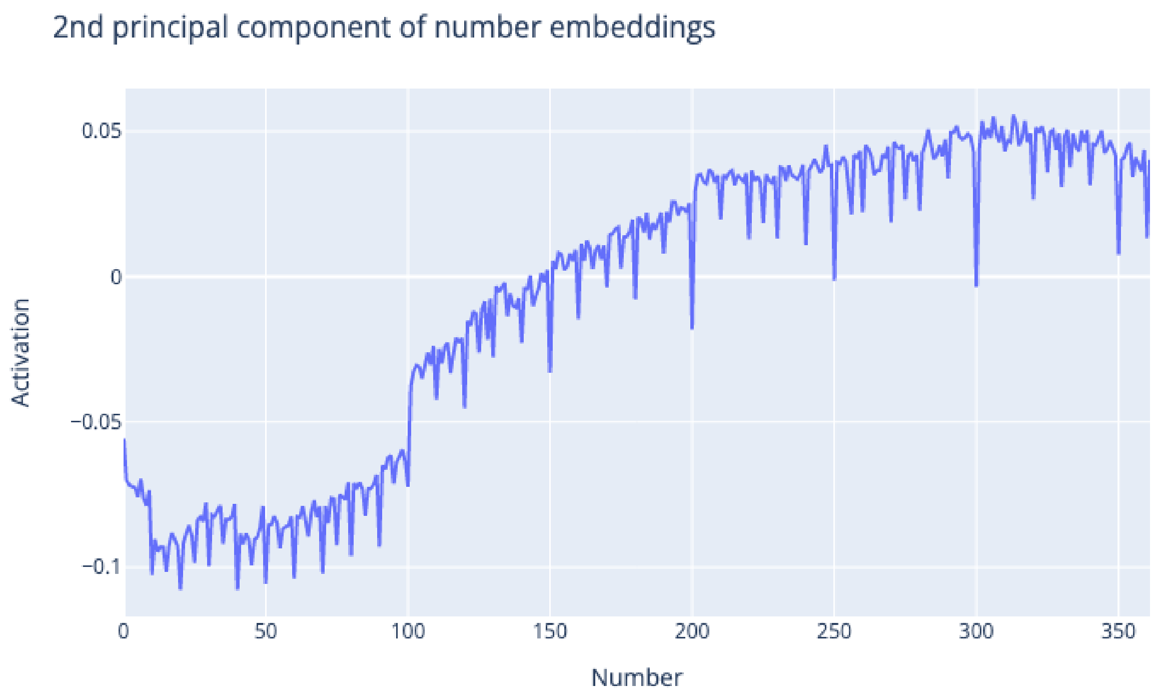
These periodic representations seem to have a number of advantages which could be contributing to their prevalence in transformer models: in particular, they are compact, amenable to manipulation using low-rank operations with softmax or GELU nonlinearities, and easily made redundant (and thus robust to dropout). They are interesting as algorithmic building blocks which seem to be "natural" for transformer models without being immediately intuitive to humans (Nanda notes that his analysis of the modular addition circuit was challenging because the algorithm was so far outside of his initial expectations); one of the main motivations of this project has been to understand whether these representations extend beyond Nanda et al.'s toy models to simple numerical tasks in generalist language models.

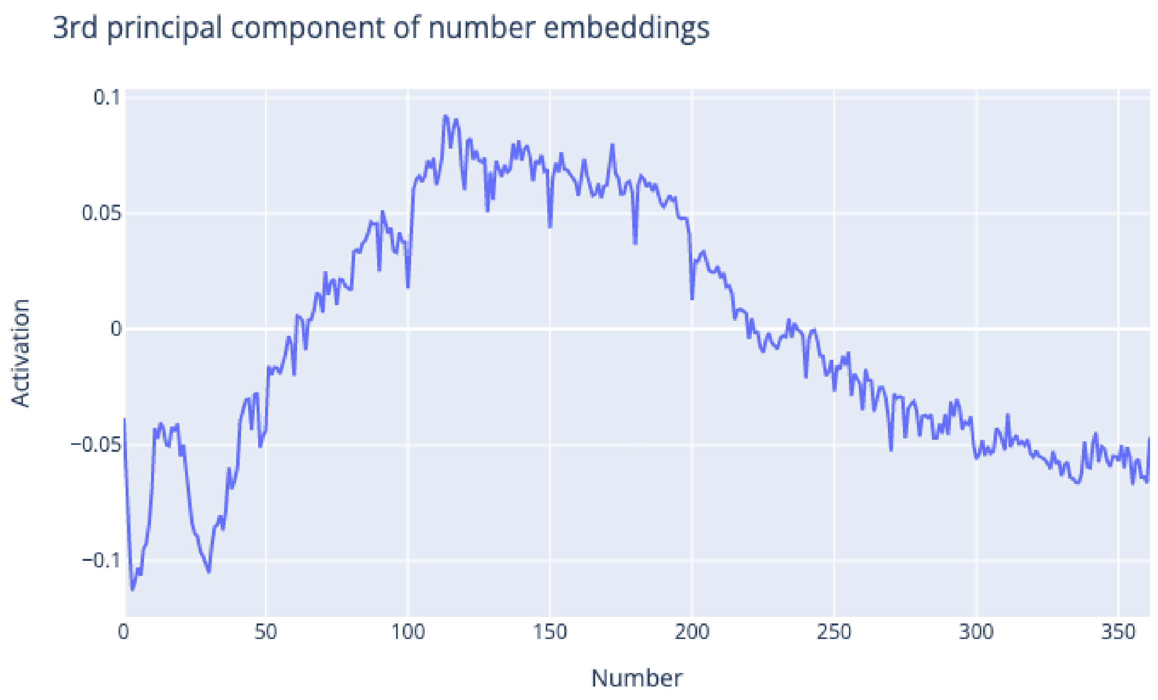
Number embeddings

The tokenizer used in Distil-GPT2—identical to that used in GPT-2 and many other models—uses different tokens for strings of non-whitespace characters with and without a preceding space. With the space, the integers `" 0"` through `" 361"` are represented as single tokens (whereas, e.g., `" 362"` is tokenized as `[" 36", "2"]`); a reasonable first step is to analyze the embeddings of these single-token numbers.

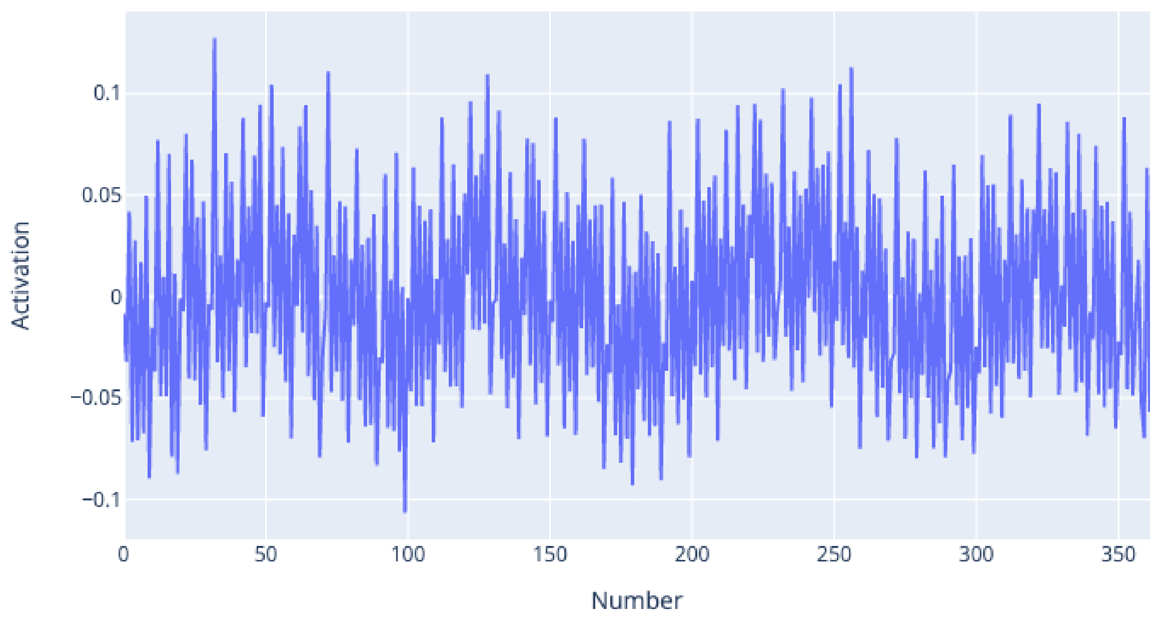
These representations are not interpretably basis-aligned, but we can use principal component analysis to extract potential features. The first ten principal components share a structure characterized by period-10 patterns superimposed atop much lower-frequency variation. For example:



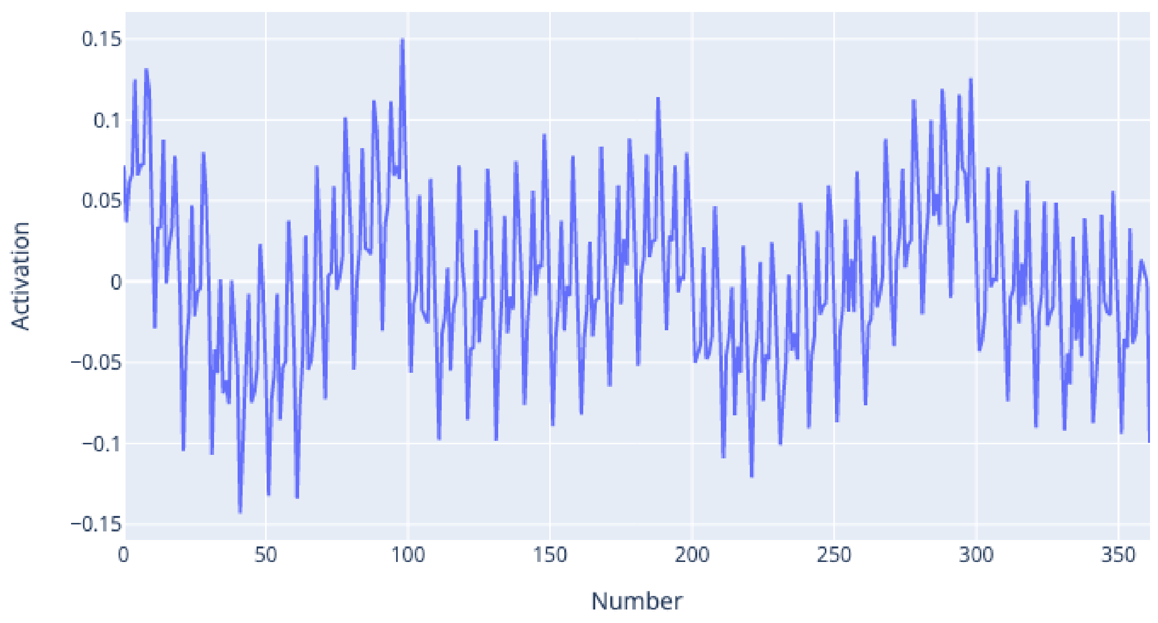

2nd principal component of number embeddings
Activation
Number
0.05
0
−0.05
−0.1
0
50
100
150
200
250
300
350

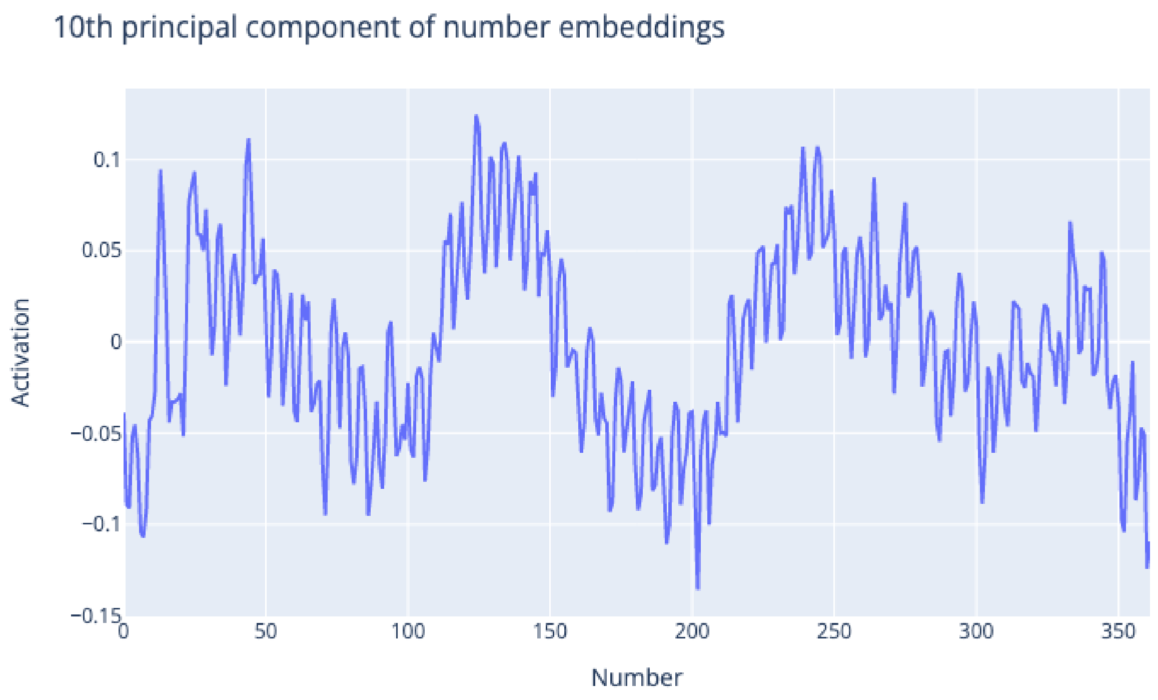

3rd principal component of number embeddings
Activation
Number
0.1
0.05
0
−0.05
−0.1
0
50
100
150
200
250
300
350

8th principal component of number embeddings



9th principal component of number embeddings





Much of the high-frequency variation consists of (upward or downward) spikes at multiples of 10, but there are exceptions to this: component 8 mostly has oscillations of period 2, with spikes on even numbers, while component 9 has oscillation of period 5, with spikes on numbers congruent to 4 or 8 mod 10.

Incrementation

The task

Completing sequences of consecutive numbers is an important task for natural language modeling. Often, consecutive numbers occur at regular points in a repeating pattern (for example, internet text often contains objects such as numbered lists and tables with numbered rows) or in immediate succession (for example, in the output from a text-based progress indicator). Perhaps unsurprisingly given the frequency with which they occur, Distil-GPT2 performs very well at these tasks, with the log-probability difference between the correct number token and a random incorrect number often as high as 20.

We looked at a single-token counting task in which the model had to predict a sequence of numbers separated by spaces (e.g., the sequence `" 147 148 149 150 151 152 153 154"`).

The performance metric

What we're interested in (for now) is "incrementation": that is, given that the model's input is a sequence of consecutive numbers ending in `..." {n−2} {n−1} {n}"` we want to understand how it predicts the token `" {n+1}"`. A model with separate embedding and unembedding matrices could do this trivially, without any attention or MLP layers, through BIGRAM STATISTICS—that is, by choosing an embedding and unembedding such that the embedding of `" {n}"` equals the unembedding of `" {n+1}"`. However, because Distil-GPT2 uses the same weights for its embed and its unembed (i.e., $W_U = W_E^T$), it generally needs to use its attention and MLP layers in order to output anything other than the input token. (This is "generally" rather than "universally" because the norms of the (un)embeddings need not equal 1, so a token might embed-and-unembed not to itself but to a token whose embedding is nearby and has a larger norm.

In a real GPT model, however, this SEEMS TO HAPPEN only when the input token is very uncommon, or in a small selection of cases where the output token is a close variation on the input token.1)
1: Theoretically, a zero-layer model could actually use this to memorize arbitrarily long sequences of non-repeating tokens, even in a low-dimensional embedding space! If the embedding vectors along a sequence increase gradually in norm while rotating away from the starting direction at an appropriate rate, the sequence can have the property that each element has a larger dot product with the subsequent element than with any other element (including itself). In two dimensions, the sequence $x_i = (e^{ik^2}\cos ik,\, e^{ik^2}\sin ik), 0 \leq i \leq 3\pi/2k$ satisfies this property. It yields a PRETTY GRAPH, although the effect is rather delicate so it seems unlikely to apply to real models.

To study this, we perform RESAMPLE ABLATION on the model's attention heads, replacing selected attention head outputs for one sequence of consecutive numbers with the cached outputs of the corresponding heads on a different sequence of consecutive numbers. (For example, we could run the model twice, first with the input `" 147 148 149 150"` and then with the input `" 325 326 327 328"`, and on the second run replace some of the heads' outputs with their cached outputs from the first run). We look at sequences of length 30. For our performance metric we use the model's accuracy at next-token prediction, which is nearly perfect[6] on the unmodified model.

Ablation

With a model this small it is tractable to find circuits by simple trial and error. Given the set H of blocks in a proposed circuit, we can establish a performance baseline by ablating the outputs of all blocks not in H, and then compare this to the corresponding performance when a head is added to or removed from H. When H contains the MLP layers and six (particular) attention heads,[7] the model performs at 0.86 mean accuracy.[8]

Redundant components

One challenge in a mechanistic circuit analysis is redundancy. Because models are typically trained with dropout, they need to process data a way that is robust to faults in parts of the circuit; since we study the model by selectively creating faults in the circuit, this robustness has the potential to make interpretability more difficult.

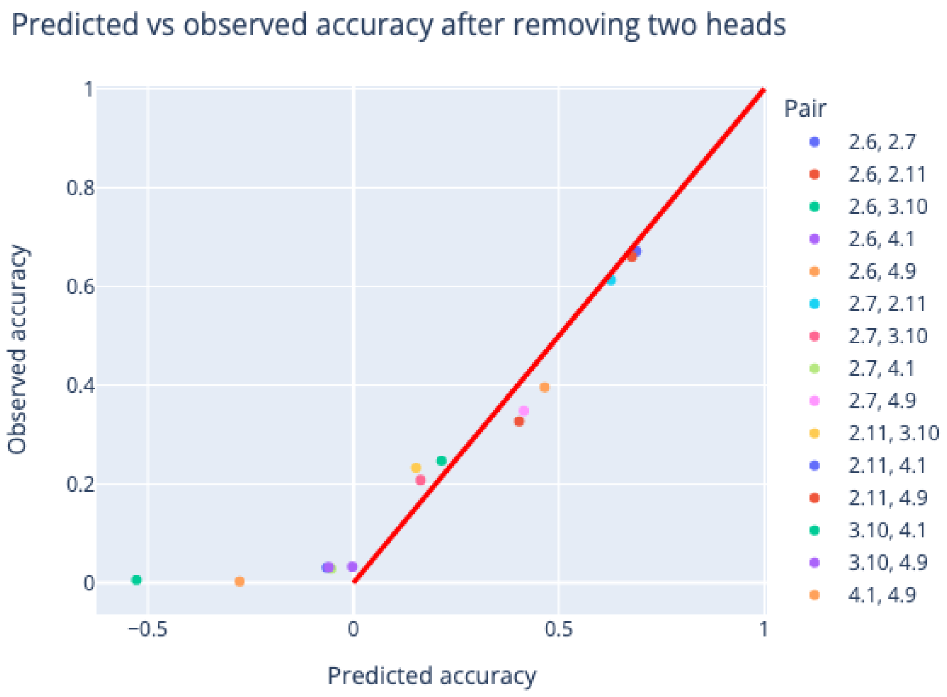
It seems desirable to be able to quantify the redundancy in a proposed circuit, both for the sake of understanding the effects of dropout and in order to gain insights into the structure of the algorithm the model implements. For example, imagine a model comprised two components, A and B, which is to be evaluated on some task. Roughly, "redundancy" means that the model should perform well when some of its components are ablated, so one approach would be to compare the sum of the single-head performance drops to the two-head performance drop. We can consider a range of architectures:

- Perhaps A and B operate together in a completely fault-intolerant way: this would be true, for example, if A and B were the two hidden layers in a small MLP model. Then ablating either A or B should cause the same drop as ablating both of them; the observed two-head-ablation performance is *better* than would be predicted by adding up the performance drops from single-head ablations.
- Perhaps A and B operate on disjoint parts of the dataset: for example, this would be true of two branches in a decision tree algorithm. Then the observed two-head-ablation performance is *equal* to the prediction given by adding the single-head drops.

- Perhaps A and B operate together in a completely fault-tolerant way: for example, they might both independently implement the same algorithm. Then the performance drop from ablating a single head should be 0, but the drop from ablating both heads will be larger: the observed two-head-ablation performance is *worse* than would be predicted by adding the props from single-head ablations.

(Note that these are far from being the only possibilities! For example, A and B might operate together in a complicated way that is fault-intolerant in some cases and fault-tolerant in others, or in such a way that a fault increases the likelihood of random incorrect guesses; either would correspond to performance somewhere between the first and third cases above.)

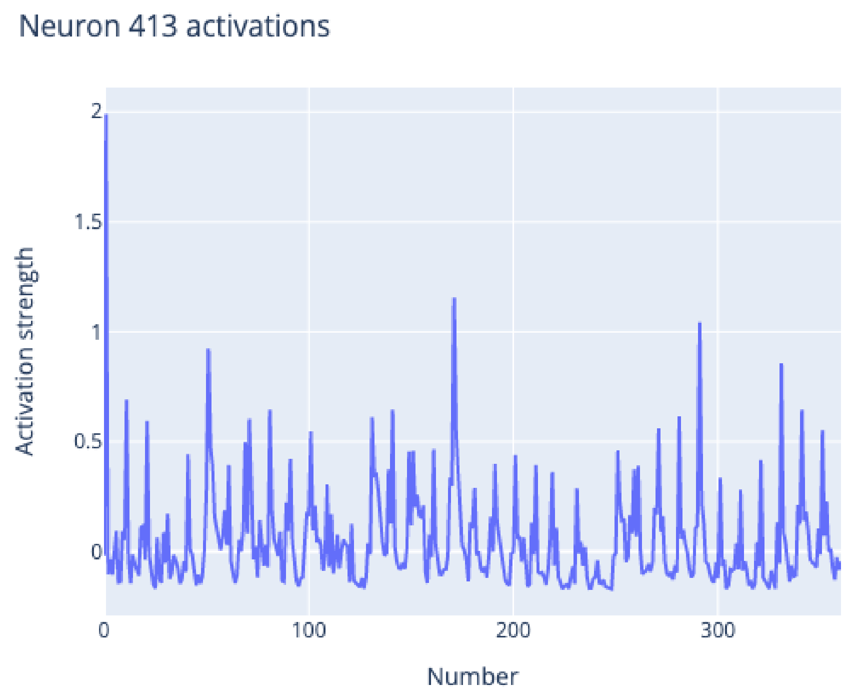
We can analyze the attention heads in our set using this approach. We find that for most pairs of heads the predicted accuracy is quite close to the observed value, with the branch to the left consisting of the pairs containing head 4.1. This suggests low redundancy within our set, which was contrary to our initial expectations.



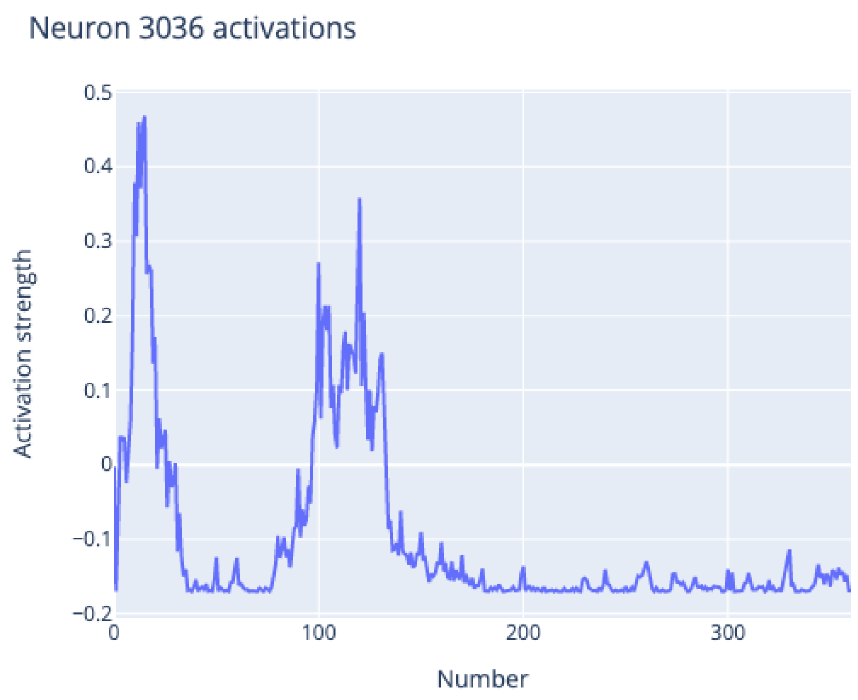
A look at some important neurons

We can ablate individual MLP neurons to determine which are important, and then look at the activations for the important ones to understand what they do.

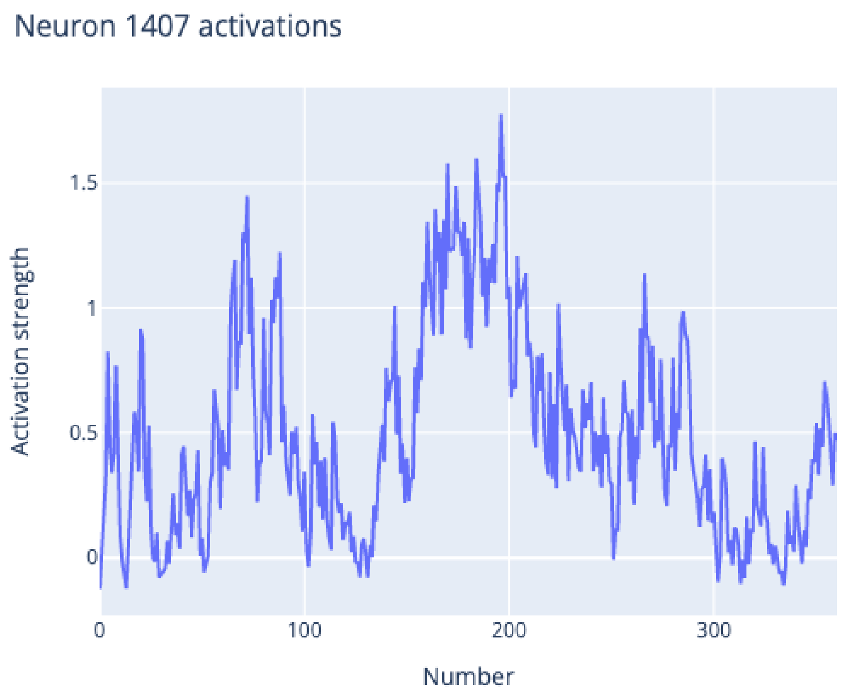
Looking at the most important ~0.5% of neurons, we find that base-10 periodicity is common but not universal. In about one third of these neurons, base-10 periodicity describes most of the behavior: for example, neuron 413 of layer 1 activates much more strongly on numbers ending in 1 than on other numbers.



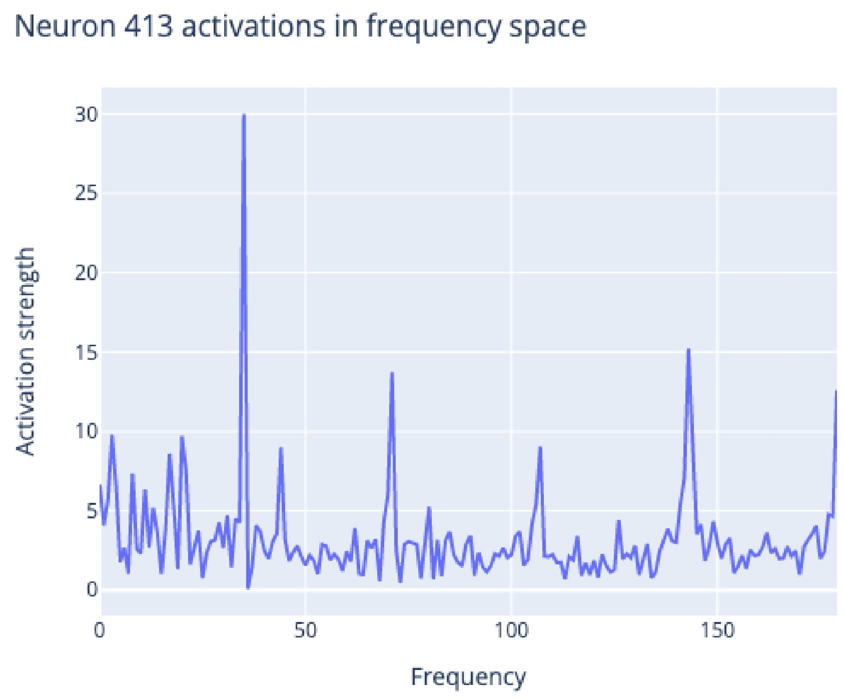
About another third of the neurons show predominantly low-frequency variation, often in forms that look more like bump functions or low-frequency random noise than neat sine waves. This neuron activates mostly on numbers less than 30 and between 90 and 140:



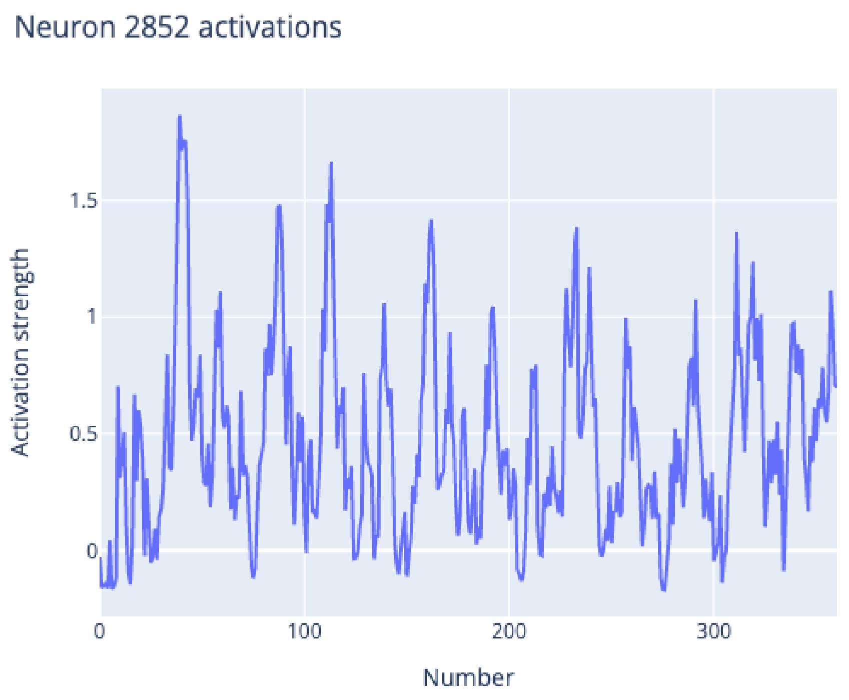
This neuron's activations vary in a way that is low-frequency but otherwise not regular:

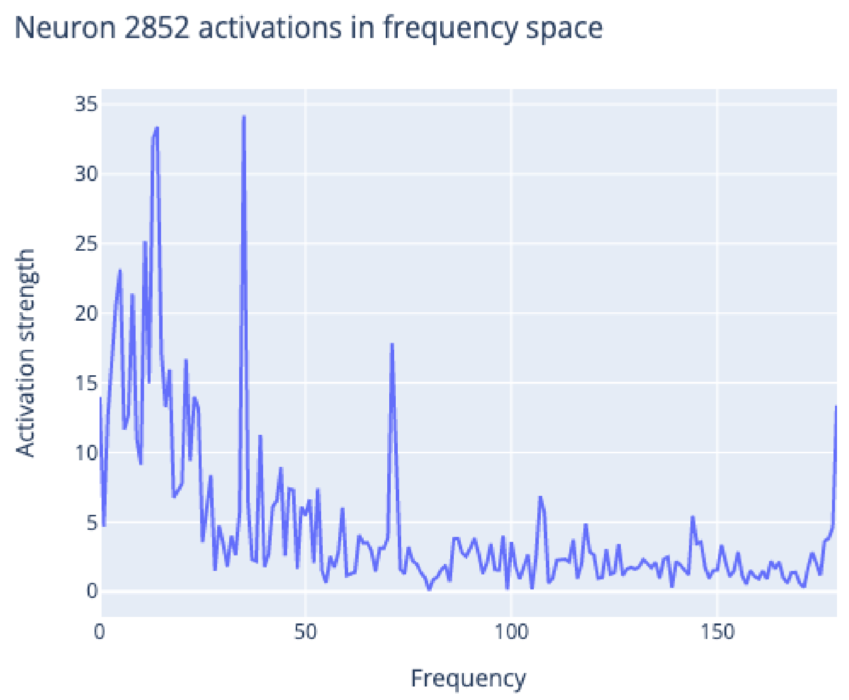


We can take Fourier transforms of the activations in order to identify subtler periodic behavior. The spectrum for the base-10 neuron from above has a strong peak at f=36, corresponding to a period of 10:



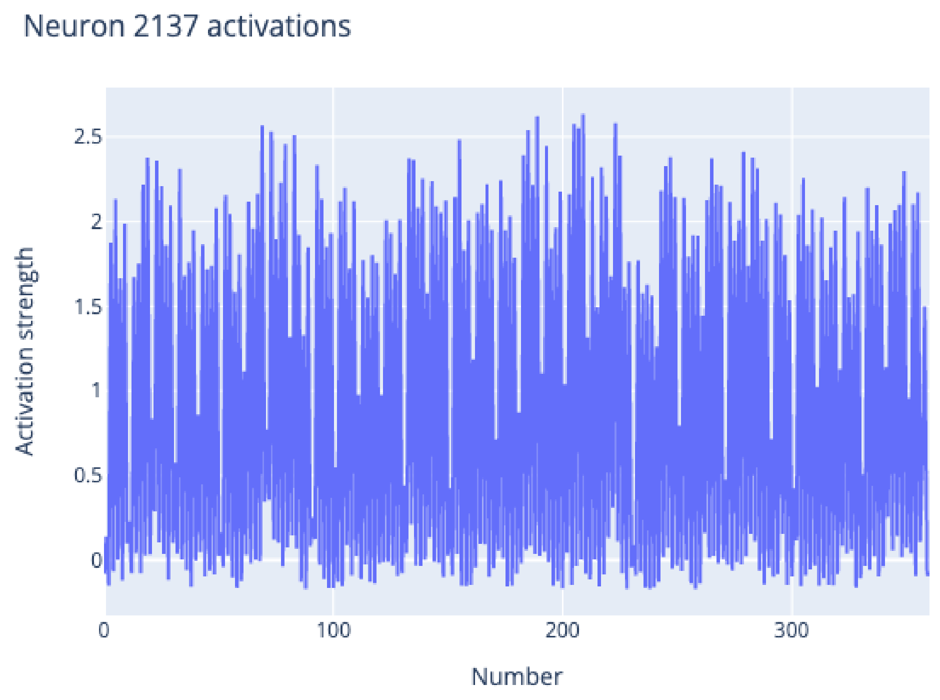
This neuron's behavior, though hard to parse visually, is explained well by its Fourier transform: it's a combination of period-10 and approximately period-26 behavior!

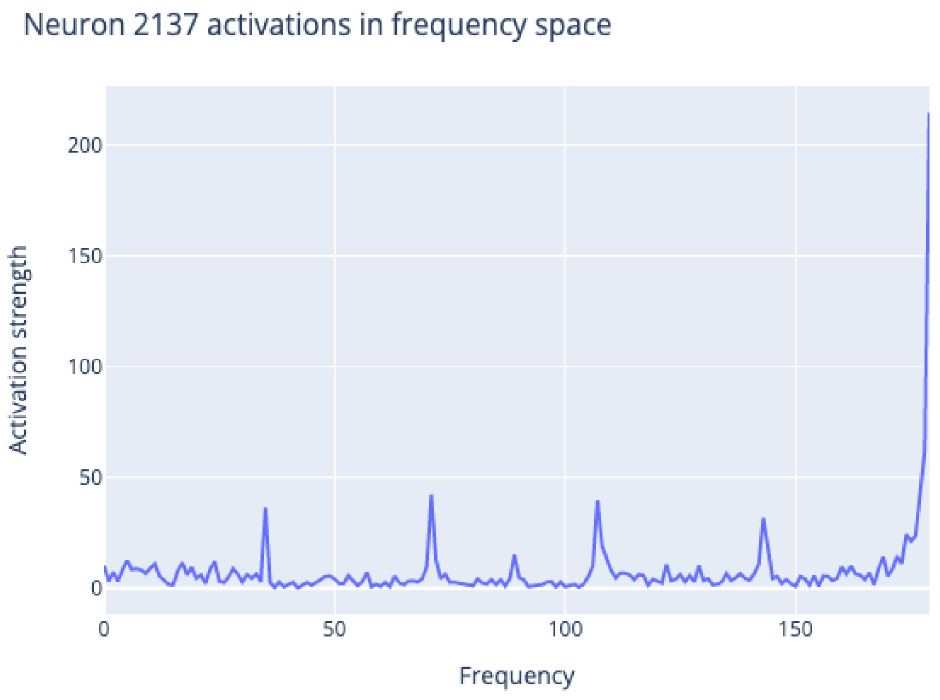




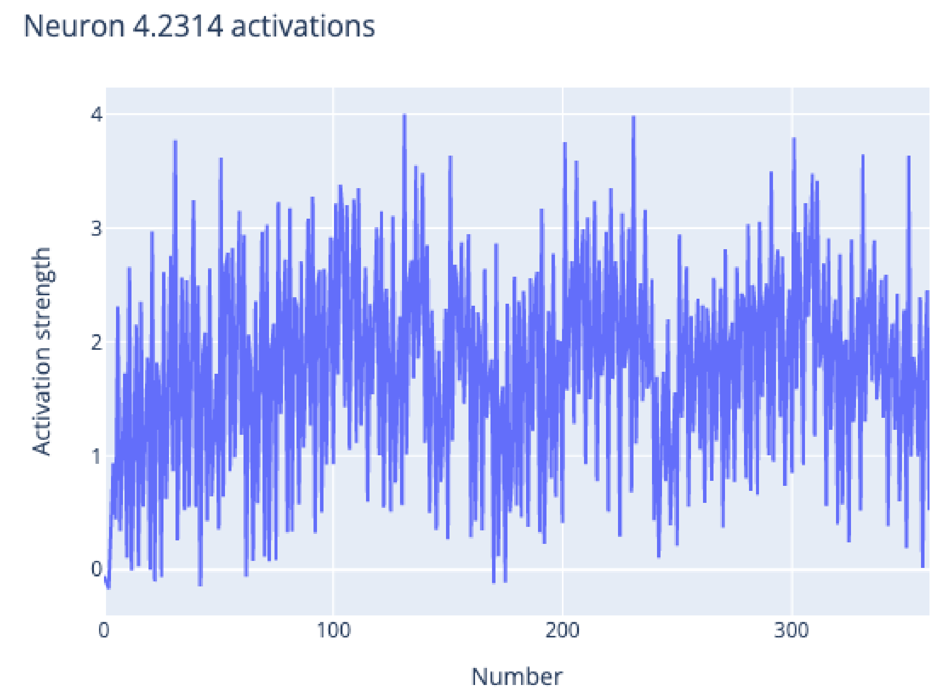
Other layers have broadly similar behavior, with many base-10 neurons and with multiple neurons each showing periods of 2, 2.5, 5, ≈26, 60, and 100.

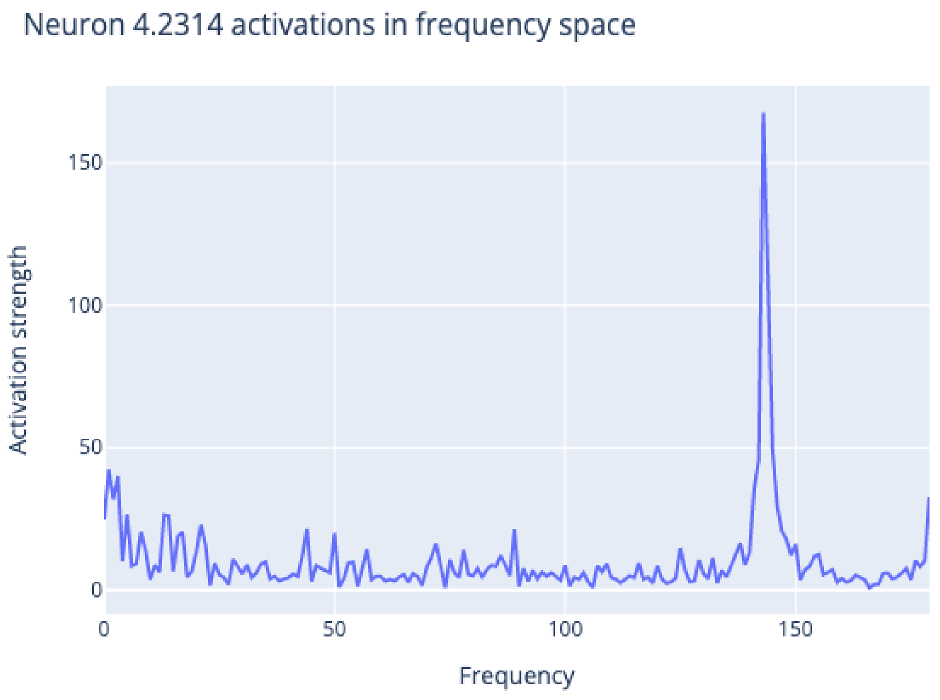
This layer 2 neuron has period 2:



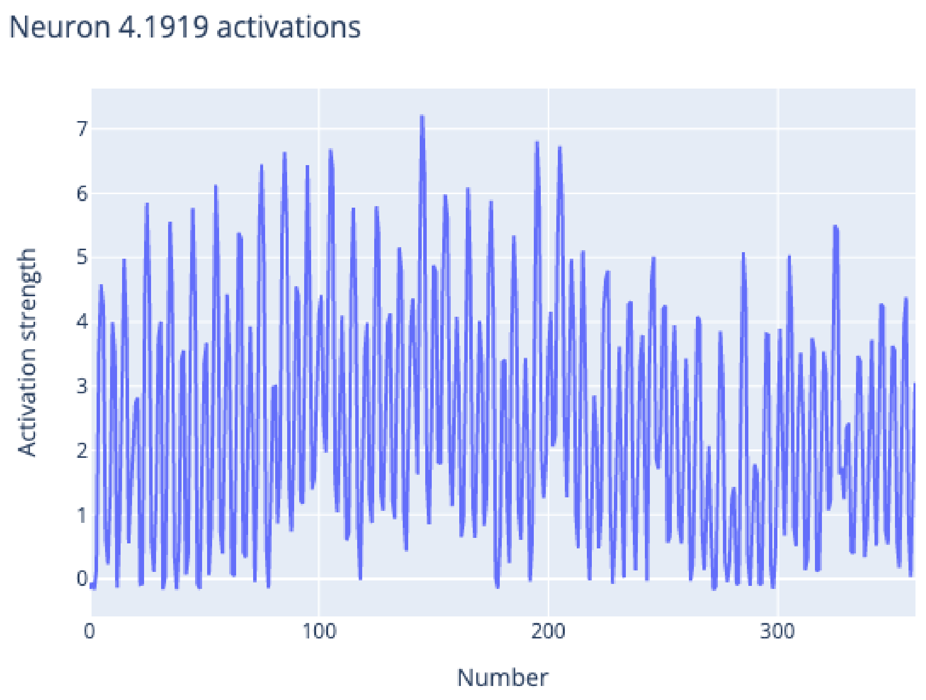


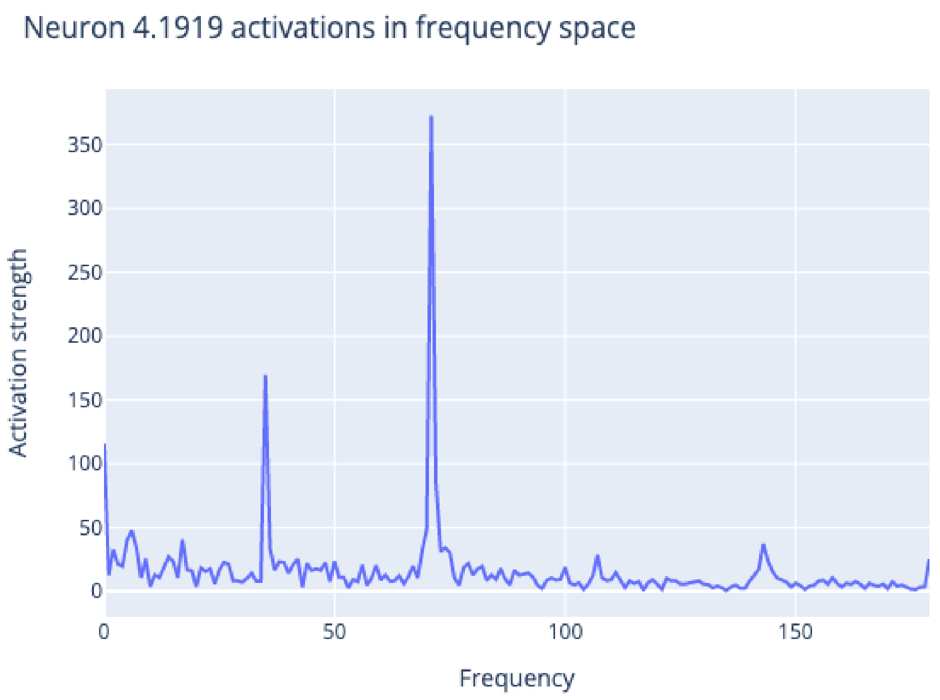
This one in layer 4 has period 2.5:





This one has period 5, and along with a couple other layer 4 neurons it is several times more important than any neuron in any other layer—ablating this single neuron decreases the model's correct-token probability by several percentage points!



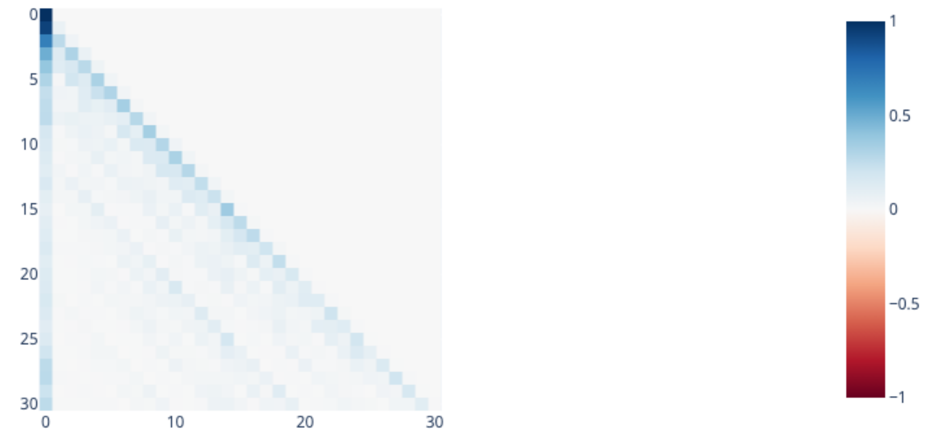


Particularly in the presence of sinusoidal representations, numerical reasoning may be a domain in which the notion of a "feature" is not sharply defined: for example, in a model whose representation of n includes a $\sin(n)$ component and a $\cos(n)$ component, any linear combination of the two is equally conceptually natural, with different combinations merely corresponding to differently-shifted sinusoids.

Representations in Distil-GPT2, however, are not purely sinusoidal: them model seems to treat last-digit information and lower-frequency information in qualitatively different ways, with an abundance of highly regular period-10 neurons and relatively little basis-alignment at lower frequencies. Feature-based notions of redundancy and polysemanticity remain somewhat problematic, but we can point at cases where the most natural basis for a representation in the model appears not to be the standard one: for example, neurons 5.1003 and 5.2599 both exhibit roughly period-10 activations with peaks on tokens with last digit 3 or 4.

An aside on components responsible for a related task

Initially I used random sequences of number tokens for my resamples, instead of sequences of consecutive number tokens. This produced substantially different results, with one attention head (head 2.4) showing up as very important despite having nearly no effect on the consecutive-token task. This head has a distinctive attention pattern, attending to the number 1 less than the current token and (somewhat less strongly) to numbers $10k+1$ less than the current token.



If the model's input is a randomly shuffled sequence of number tokens or a sequence with other text between the number tokens, the attention pattern still shows similarly strong peaks for numbers $10k+1$ less than the smaller token, demonstrating that the attention pattern depends primarily on content rather than positional information.

I did head ablation on the layer 0 and 1 heads, and neuron ablation on individual neurons of MLPs 0 and 1, against a measure of the strength of the head's attention to the 11th token back (specifically, I measured the difference in the attention pattern between the 11th position back and the average of the two neighboring positions). The MLP neurons which were important for this measure had similar activation patterns to the ones described in the previous section.

Conclusions

To the extent that its behavior is interpretable, the model shows moderate evidence for the use of periodic representations for counting. In contrast with the behavior observed in Nanda et al.'s toy models and with positional embeddings, the model shows a substantial preference for components with period 10 and fractions and multiples thereof, presumably because numbers are typically represented in base 10 in text.

Further work is needed to understand the model's number representations fully, but a more comprehensive picture may well be within reach. Recent work by Geiger et al.[9] has proposed techniques to facilitate the process of finding feature-aligned bases; work using this and/or related techniques may lead to greater clarity on the details of computation in the model. A systematic approach such as causal scrubbing[10] might also allow for more precise conclusions about implementation details in the model. If these directions are fruitful, I'll write an update to this post, hopefully both deepening and formalizing the analysis here.

---

[1] Nanda et al., https://arxiv.org/abs/2301.05217 ↩

[2] Vaswani et al., https://arxiv.org/abs/1706.03762 ↩

[3] Schleris, https://www.alignmentforum.org/posts/BMghmAxYxeSdATeDc/an-exploration-of-gpt-2-s-embedding-weights#Token_Position_embedding_rivalry ↩

[4] Yedidia, https://www.alignmentforum.org/posts/zRA8B2FJLtTYRgie6/the-positional-embedding-matrix-and-previous-token-heads-how ↩

[5] Nanda, https://youtu.be/OI1we2bUseI?si=lUtk7DoIHneHoyZY ↩

[6] Running 1000 trials gives a sample mean of 0.998 with a standard error of 0.0014. ↩

[7] Namely: heads 2.6, 2.7, 2.11, 3.10, 4.1, and 4.9. ↩

[8] std. err. = 0.003. ↩

[9] https://arxiv.org/abs/2303.02536 ↩

[10] Chan et al., https://www.alignmentforum.org/posts/JvZhhzycHu2Yd57RN/causal-scrubbing-a-method-for-rigorously-testing#4_Why_ablate_by_resampling_ ↩