Numerical Patching of Transformers for In-Context Linear Regression Finds Normalizing Heads

LSE.AI
London School of Economics and Political Science
London, UK, Houghton St
lseai.org

Abstract

We deploy mechanistic interpretability teqniques on transformers used for in-context linear regression [Garg et al., 2023]. We build on the intuition that transformers can learn in context by gradient descent [von Oswald et al., 2023], and we probe the coefficients of learned function classes from the MLPs. We further obtain a computational graph for the model trough Automated Circuit Discovery [Conmy et al., 2023]. We then perform a causal intervention to extract the function being implemented trough the model's most significant computational edge, the OV matrix of attention head zero of the first transformer block. We find the head performs a normalization of the labels provided in context. We then define a norm score to quantify this behavior, and find that such heads play a role in the model's phase transition during training. We finally find norm heads in larger models

1 Introduction

Transformers [Vaswani et al., 2017] have emerged as the dominant machine learning architecture across a variety of task, and the resulting large language models have even been proposed as an early (incomplete) generally intelligent system [Bubeck et al., 2023]. In context learning [Brown et al., 2020] has significantly contributed to their success by boosting few-shot performance, and enabling language models to perform a variety of tasks at a human level. Outside of language, Transformers are able to learn entire function classes in context [Garg et al., 2023], and perform in line with state of the art methods such as ordinary least squares for linear regression. This can be regarded as an instance of meta-learning [Thrun et al., 1998], as the model learns how to teach itself an unseen function based on provided examples. Interestingly, there is a phase transition in the loss of the model when it's trained on sequences of points with more than 20 dimensions. It has been shown that transformers learn to perform gradient descent for linear regression [von Oswald et al., 2023], and that they perform a gradient update with every forward pass. Grokking [Power et al., 2022], another algorithmic problem displaying a phase transition, has recently been investigated through the lens of mechanistic interpretability [Nanda et al., 2023]. We undertake to use similar techniques to investigate the behavior of the model, and how it evolves as it undergoes its phase transition. Our contributions include adapting classic intepretability techniques such as activation patching [Zhang and Nanda, 2023] to numerical (non token based) tranformers, and conducting the first attention head case study in a numerical transformer. Other case studies include induction heads [Olsson and Nanda, 2022] and copy suppression heads [McDougall et al., 2023].

2 Mechanistic Interpretability

Mechanistic interpretability is an emerging methodology in AI research aimed at understanding the underlying mechanisms and the "reasoning" processes of large language models (LLMs), particularly

Preprint. Under review.

transformers. This approach diverges from traditional interpretability methods that often focus on correlations and general input-output relationships by delving into the exact computations and transformations occurring within the model's architecture [Nanda et al., 2023]. At its core, mechanistic interpretability involves dissecting the neural network to isolate individual components—such as specific neurons, layers, or attention heads—and studying their roles in processing inputs. This is achieved through techniques such as activation patching [Zhang and Nanda, 2023], where inputs are systematically modified to observe changes in outputs, and Automated Circuit Discovery (ACDC) [Conmy et al., 2023], which constructs a computational graph of the model to pinpoint key computational pathways and their influence on performance. One practical application of this methodology is to perform causal interventions on these identified components to assess their impact on the model's output. For instance, by altering the input to a particular attention head and observing the variation in output, researchers can infer the function being executed by that head. This was exemplified in our studies where modifying the inputs to the "OV matrix of attention head zero" in a transformer revealed its role in normalizing labels, thereby affecting the model's ability to generalize across different input scales. Furthermore, mechanistic interpretability also includes quantifying the influence of these components. We devise to the end a "norm score" for attentin heads.

3 The Model

3.1 Transformers

Transformers [Vaswani et al., 2017], initially popularized for their SOTA performance in natural language processing (NLP) tasks [Brown et al., 2020], have demonstrated exceptional versatility by extending their capabilities to complex function mapping in contextual settings. These models leverage their self-attention mechanisms to process sequences of data in parallel. This characteristic allows them to effectively manage and interpret relationships between data points across large datasets without regard to their sequential proximity. Specifically, in the realm of learning direct mappings from input x to output y sequences, transformers treat these mappings as sequential data, thereby learning the underlying functional relationships from the provided in-context examples [Garg et al., 2023]. Such an approach has been successfully applied to a variety of function classes including linear functions, sparse linear functions, and even intricate models like decision trees and neural networks, directly during inference. The adaptability of transformers to assimilate new data rapidly and update their understanding of function mappings without the need for retraining underscores their potential in applications far beyond their initial NLP use cases.

3.2 In-context Linear Regression

We train a decoder-only transformer model on the linear regression task of [Garg et al., 2023], specifically opting for a 3-layer configuration that processes 5-dimensional data points. The training regime employs prompts structured as sequences of input-output pairs:

$$f(x) = \{x_1, f(x_1), x_2, f(x_2), \ldots, x_k, f(x_k)\} \quad \text{with} \quad f(x) = x \cdot \mathbf{w}^T,$$

where $\mathbf{w}$ represents the weight matrix, and both x values and weights are sampled from an isotropic Gaussian distribution. This training process utilizes curriculum learning strategies [Bengio et al., 2009]. The model demonstrates a marked improvement in performance over the training period, achieving a mean squared error (MSE) of approximately 2, which significantly reduces to about 0.4 on the final five predictions.

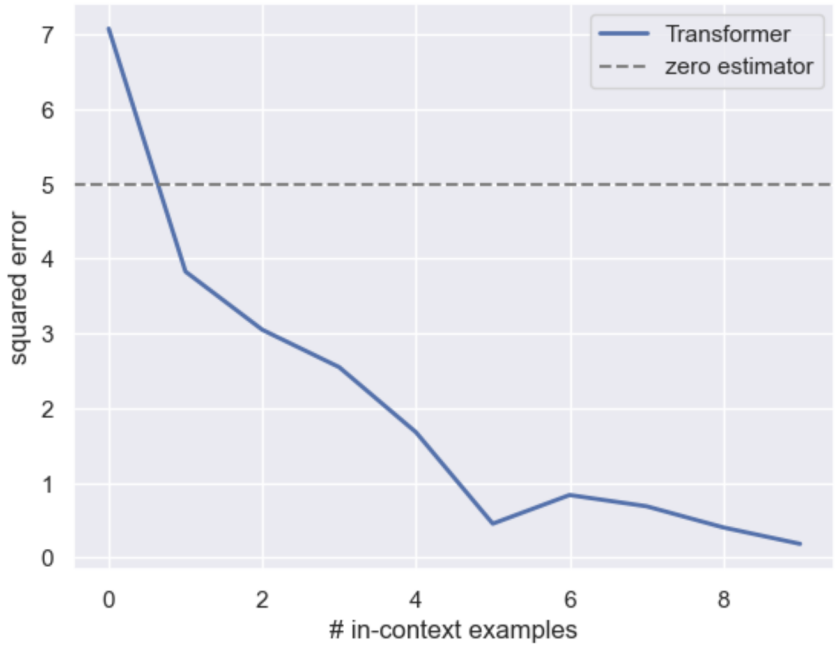

Figure 1: Loss achieved by the model

4 Computational Graph

We then visualize the importance of each of the model's computations via Automated Circuit Discovery [Conmy et al., 2023]. This approach views models as computational graphs [Geiger et al., 2021], and informs us on how much each component of the model makes use of the output of previous components by measuring the loss after a causal intervention on their inputs. From ACDC we gather

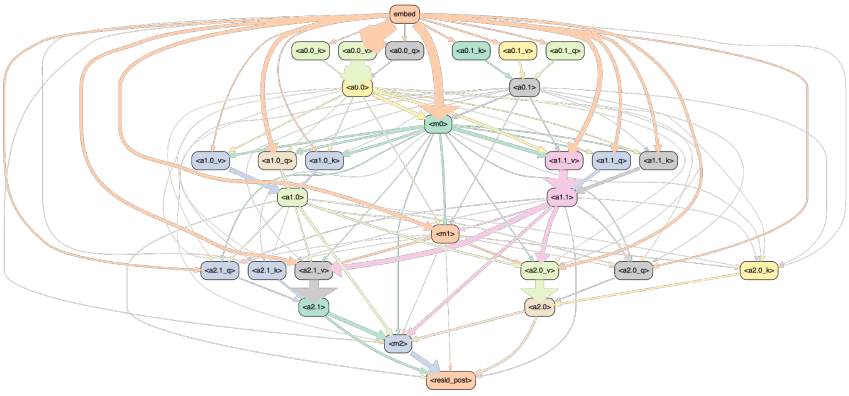

Figure 2: Computational graph

that the embedding is the most important component of the computation, and that MLP2 is the last significant computation to take place, and is thus a good candidate for further enquiry.

5 Embedding analysis

The embedding appears to be accessed by a variety of components. To study it we use activation patching [Zhang and Nanda, 2023], and observe the impact on loss after replacing the input to each component at the x and y position respectively with some corrupt values. We then plot according to the impact of this operation on model loss:

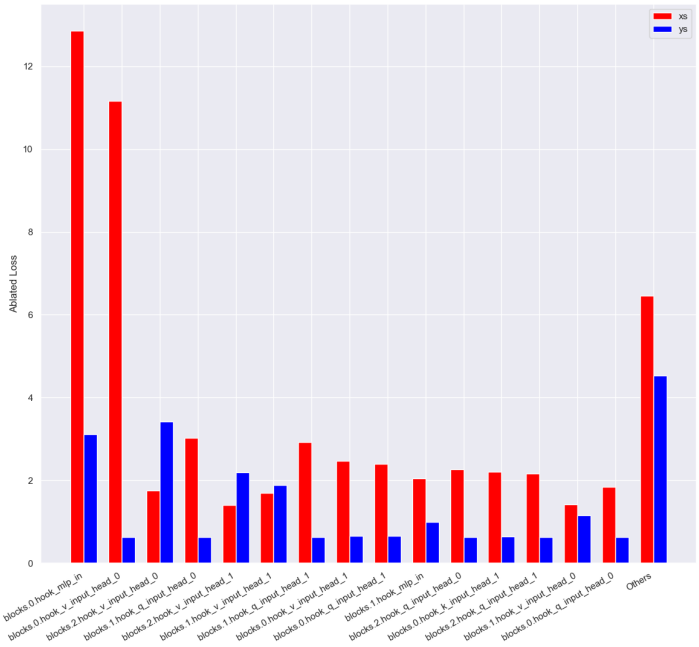

Figure 3: Loss increase by component when the embedding is ablated

We note that A0.0 in particular makes use of xs to a very large extent, while it does not attend to ys.

6 Probing the Weights Out

Building on the intuition that transformers perform gradient descent for in-context linear regression through their forward passes [von Oswald et al., 2023], we theorize the MLP's parameters should be close to the actual linear regression weights for each sequence. We find a mechanistic confirmation of this through a linear probe [Alain and Bengio, 2018].

We feed into a linear regression the concatenated activations of MLPs at each point of the sequence, and train for 1.5M steps with learning rate 1.0e-05. We minimize mean squared error. The final loss achieved by the probe is 0.4 with base loss around 1, suggesting MLP2's activations do indeed contain a sizable fraction of the probed weight as expected.

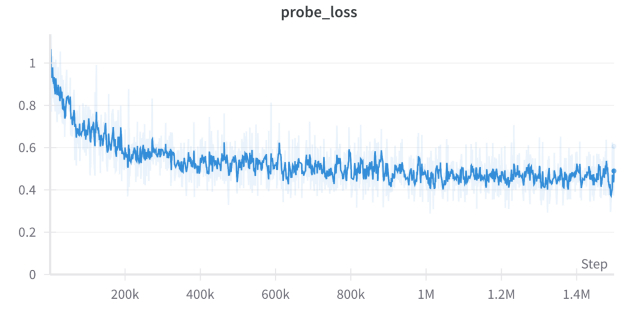

Figure 4: Loss achieved by the probe

We further selectively ablate each layer by replacing the corresponding inputs to the probe with activations from another sequence, and confirm MLP2 is the layer containing most of the weights.

We nevertheless observe that the amount of information about the weights increases with each layer, in line with the idea that they correspond to progressive a gradient updates in the direction of the weights.

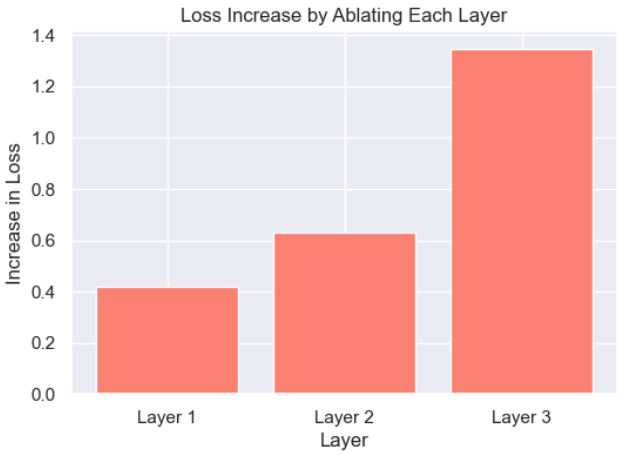

Figure 5: Increase in loss when each MLP layer is ablated

7 Normalizing heads

We further investigate head 0.0 in the model, as it has the most significant edges in the computational graph (disrupting its inputs or outputs will drastically increase overall loss).

7.1 Numerical patching

We adapt classical activation patching [Zhang and Nanda, 2023] to the algorithmic nature of the model, and define 2 sequences:
base:

$$x_1 = (1, 1, 1), \quad y_1 = 1, \quad x_2 = (1, 1, 1), \quad y_2 = 1, \quad \ldots, \quad x_k = (1, 1, 1), \quad y_k = 1$$

patch:

$$x_1 = (p_1, p_2, p_3), \quad y_1 = 1, \quad x_2 = (p_1, p_2, p_3), \quad y_2 = 1, \quad \ldots, \quad x_k = (p_1, p_2, p_3), \quad y_k = 1$$

We run the model on the base sequence, but replace the input to A0.0 with the embedding of the patch sequence at x positions. This enables us to observe the effect of varying xs on the output of the model. Ys in the base sequence are not modified, and arbitrary set at 1.

We then infer the operation performed by A0.0 from the relationship between x1 x2 x3 and y. We generate 13512 random 3d points, and then patch them in the base sequence. We find:

Table 1: Correlation and p-value Analysis

Analysis Results			
	x_1	x_2	x_3
Correlation with y	-0.859195	-0.301711	-0.205789
p-value	0.0	~ 0 (2.28480×10^{-282})	~ 0 (3.77526×10^{-129})

As x_1 has a strongly negative correlation with y, we plot x_1 against y and we get:

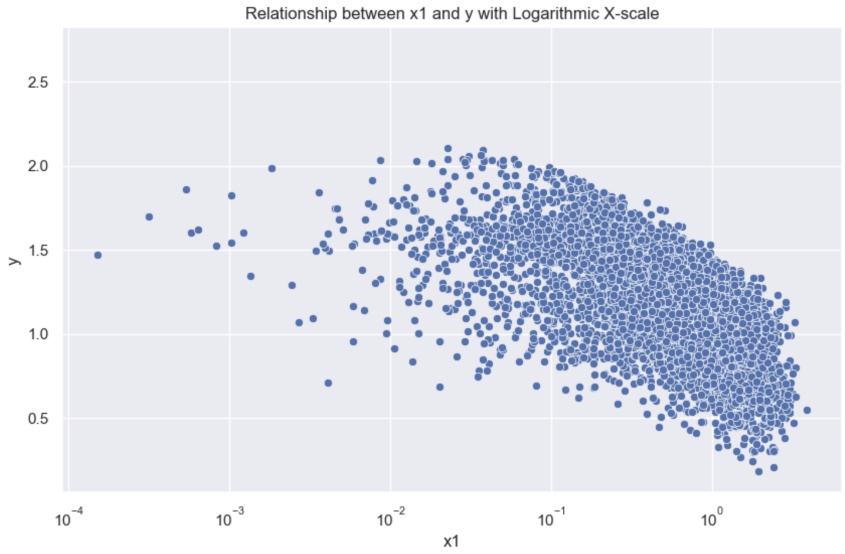

Figure 6: Plot of y over x_1

This suggests the head is approximately dividing ys by xs. Since they are linearly related ($y = x \cdot w^T$), this operation could effectively normalize the scale of ys across different values of xs, making them a dimensionless quantity. This can help in cases where the magnitude of xs affects the performance of the model, and as [Garg et al., 2023] note, this model's performance does degrade somewhat when inputs are scaled:

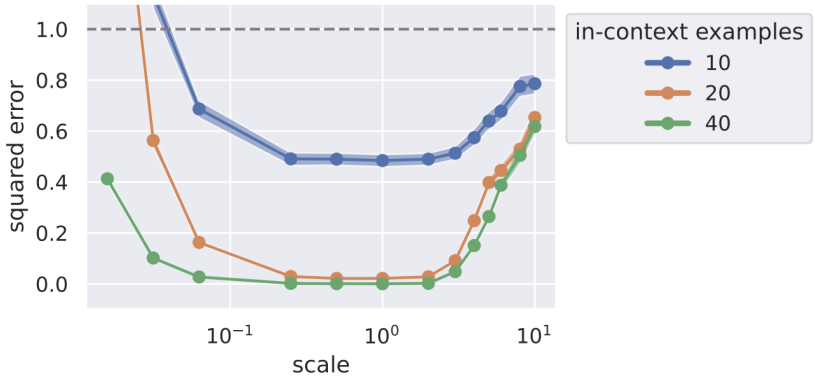

Figure 7: Loss over different orders of magnitude of x

7.2 The OV matrix as a function

We can further infer the function being performed by A0.0 by constraining ourselves to the mono dimensional case, and defining the patch sequence as follows:

$$x_1 = (p_1, p_1, p_1), \quad y_1 = 1, \quad x_2 = (p_1, p_1, p_1), \quad y_2 = 1, \quad \ldots, \quad x_k = (p_1, p_1, p_1), \quad y_k = 1$$

We are then able to calculate the correlation and p-value for x_1 (it's the same for x_2 and x_3 as $x_1 = x_2 = x_3$).

Finally, we can plot $x1$ against y:

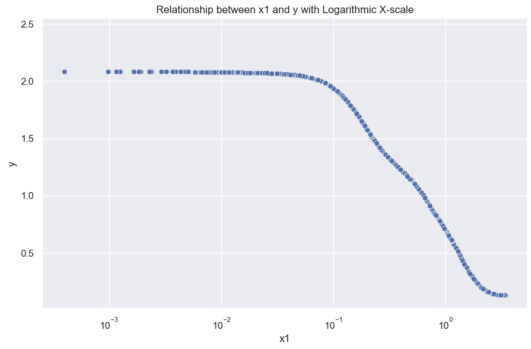

Figure 8: Values of y after patching in x_1 (log scale)

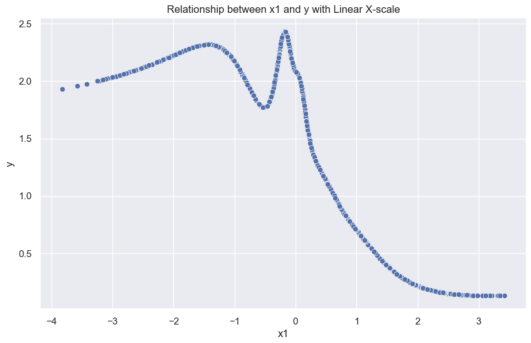

Figure 9: Values of y after patching in x_1

We note an interesting bump corresponding to x=0, and thorize it's due to the model implementing a constant where it can't compute a division by x1 = 0.

7.3 Attention pattern

We further analyze A0.0's attention pattern:

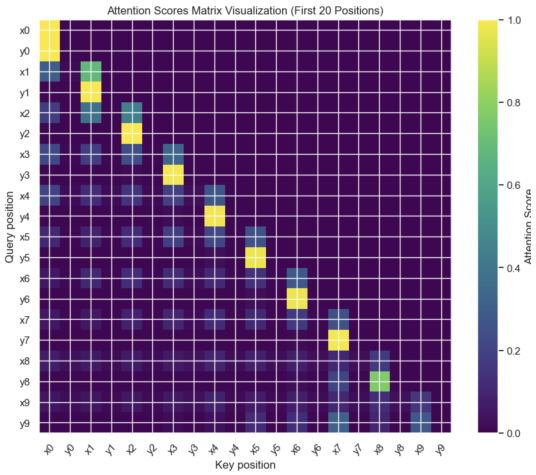

Figure 10: Attention pattern of head zero in the first transformer block

We note that A0.0 seems to move information to the y positions from the preceding x, and we compute the mean of each element with query at a y position and the key at the preceding xs's position, and we use its average to quantify how much A0.0 is actually moving information from xs

to the next ys:

$\mu = \frac{1}{n} \sum_{i \in I_{\text{odd}}} A_{i,i-1}$

We obtain an average attention score between ys and their preceding xs of: **0.8943**

7.4 Eigenvalues

We find eigenvalues of the OV matrix to be strongly negative indicating an inversion of x values, which is in line with our hypothesis of a normalizing head, as we can conceptualize the division as inverting xs and multiplying ys by them.

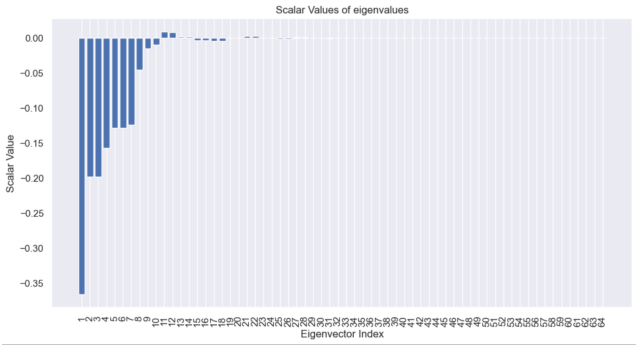

Figure 11: Eigenvalues of the OV matrix

The sum of the eigenvalues is: **-1.3669.**

7.5 Projections of xs onto eigenvalues

Lastly, we verify that values at x positions are indeed what the OV matrix is selecting by computing their projection onto OV's significant eigenvectors, which account for 83% of the variance. We define a significant eigenvector as one with an absolute value greater than 10-1. We take the average ratio of xs' projections onto OV's significant eigenvectors as an indication of whether xs project more onto OV or not, with values > 1 indicating greater projection of xs and vice versa.

$$
\begin{aligned}
T &= 0.1 \\
H &= 0 \\
M &= \text{AB}_{\text{matrix}} \\
(\lambda_i, \mathbf{v}_i) &\quad \text{such that } M\mathbf{v}_i = \lambda_i \mathbf{v}_i \\
S &= \{i \mid |\lambda_i| > T\}
\end{aligned}
$$

For each $i \in S$:

$$
\begin{aligned}
P_{\text{even}} &= \textstyle\sum_{j \in \text{even indices}} |\mathbf{e}_{\text{input},j} \cdot \mathbf{v}_i| \\
P_{\text{odd}} &= \textstyle\sum_{j \in \text{odd indices}} |\mathbf{e}_{\text{input},j} \cdot \mathbf{v}_i| \\
H &\mathrel{+}= \frac{P_{\text{even}}}{P_{\text{odd}}} \\
\text{Final Score} &= \frac{H}{|S|-1}
\end{aligned}
$$

The resulting projections score is **1.9764** calculated on the top 6 eigenvectors, indicating that the xs project almost twice as much as the ys. This further confirms that the OV matrix is selecting its information from xs.

We further visualize projections onto 2 of OV's main eigenvectors for each point in a batch of 32 sequences.

7.6 Projections of xs onto eigenvalues

Taken together, we can use the attention, eigenvalues, projections, and position of the head to define a norm score (after applying min-max normalization to each of the 4). We include a penalty for heads

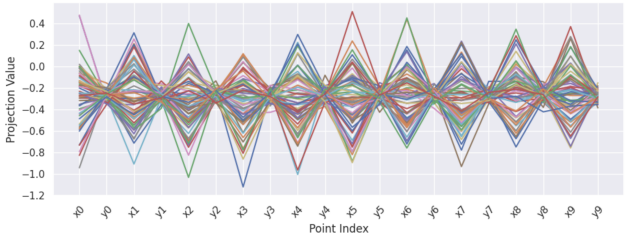

Figure 12: Projections of different positions in the sequence onto eigenvector 4

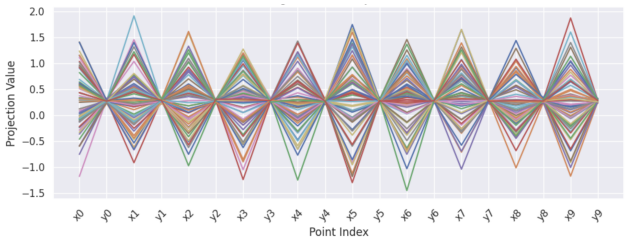

Figure 13: Projections of different positions in the sequence onto eigenvector 5

later in the model as they are unlikely to be performing actual normalization, as MLPs compute estimate increasingly accurate estimates of the weights.

Let:

Attention Scores:

$$\mu = \frac{1}{n} \sum_{i \in I_{\text{odd}}} A_{i,i-1}$$

Sum of All Eigenvalues:

$$\Sigma\lambda = \sum_{i} \lambda_i$$

Position:

$$P = \frac{\text{Current Block Index}}{\text{Number of Layers in the Model}}$$

Projections Score:

$$PS = \frac{1}{|S| - 1} \left(\sum_{i \in S} \frac{P_{\text{even}}}{P_{\text{odd}}} \right)$$

where:

$$\begin{gathered} T = 0.1, \quad M = \text{AB}_{\text{matrix}}, \\ (\lambda_i, \mathbf{v}_i) \text{ such that } M\mathbf{v}_i = \lambda_i \mathbf{v}_i, \\ S = \{i \mid |\lambda_i| > T\}, \\ P_{\text{even}} = \sum_{j \in \text{even indices}} |\mathbf{e}_{\text{input},j} \cdot \mathbf{v}_i|, \\ P_{\text{odd}} = \sum_{j \in \text{odd indices}} |\mathbf{e}_{\text{input},j} \cdot \mathbf{v}_i|. \end{gathered}$$

Then, the 'norm score' (NS) is given by:

$$NS = \frac{1}{4}(\mu + \Sigma\lambda + P + PS)$$

Using this norm score we find the model has 2 normalizing heads: head 0 layer 0 and head 0 layer 1:

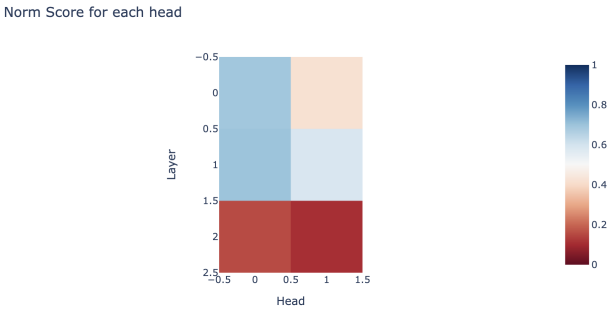

Figure 14: Norm scores per head

8 Induction and Copy Heads

We further check whether the model develops any induction heads [Olsson and Nanda, 2022] or copy heads, as defined per their copy score. We find no induction head, and an average induction score across all heads of just 0.0172. We further find only one weak copy head, with copy score of 0.44.

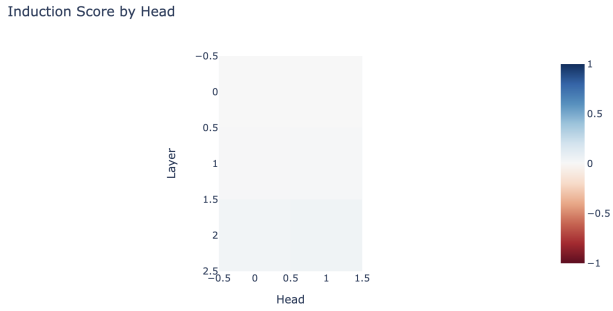

Figure 15: Induction scores per head

Figure 16: Copy scores per head

9 Phase transition

[Garg et al., 2023] note that there is a phase transition in the model's loss function when trained on more than 10 dimensions without curriculum learning. We therefore train a 3L model at 20 dimensions, and identify the head with the highest norm score, and then record overall loss (blue), restricted loss (red, head can only read xs, ys are ablated) and excluded loss (orange, xs ablated). We observe the following phase transition.

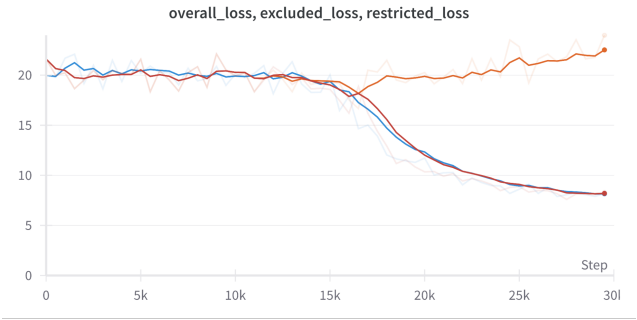

Figure 17: Overall, excluded and restricted loss during training

We measure total loss and loss on the last 5 positions for the model both:

- before the phase transition (epoch 17k), total=15.179, last 5=14.017
- after phase transition (epoch 19k), total= 11.419, last 5=8.453

We then roll back weights to before the phase transition for each of the components and measure the increase in loss over the last 5 positions, copy and norm scores (A stands for attention head, M for MLP, E for embedding, and U for unembedding):

Table 2: Model performance after rolling back weights for each component

Removed Component	Performance	Loss Increase	Norm Score	Copy Score
$['A', 2, 0]$	15.19781	6.8101635	0.281563	0.7316899
$['A', 2, 1]$	13.549149	5.161502	0.1832637	-0.2062406
$['A', 1, 1]$	12.014914	3.627267	0.4894352	-0.9112423
$['A', 0, 0]$	10.951018	2.5633717	0.8854116	-0.8547632
$['E']$	10.518488	2.1308413		
$['M', 0]$	10.274035	1.8863888		
$['U']$	9.693608	1.3059616		
$['M', 1]$	9.657602	1.2699556		
$['M', 2]$	9.265495	0.8778486		
$['A', 0, 1]$	9.168398	0.7807512	0.6227482	-0.7797208
$['A', 1, 0]$	8.273996	-0.11365032	0.3698539	-0.8096694

We further graph the top 5 components and their norm/copy scores: We note that the transition

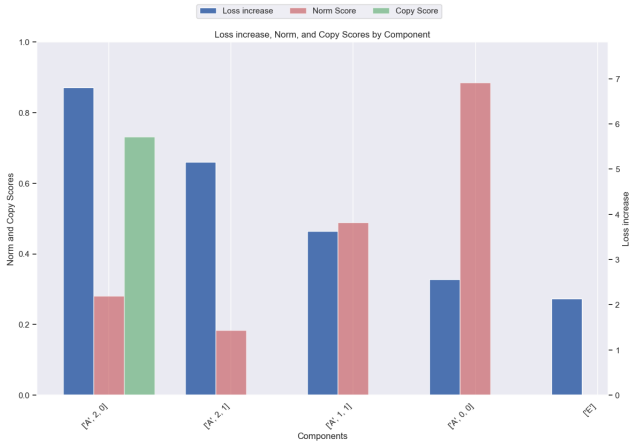

Figure 18: Loss increment and copy/norm scores for the 4 most important components

seems to be characterized by the development of norm and copy heads, as three out of 4 of the most important heads in this transition are either copy or norm head.

10 Are norm heads performing a gradient descent step?

An objection might arise suggesting that the norm head we are examining is merely executing gradient descent, akin to mechanisms discussed in related works such as [von Oswald et al., 2023]. However, we show the function implemented by the head in section 7.2, and it's strictly positive even for negative values of x, which would warrant a negative weight in the context of gradient descent. Moreover, our linear probe, as discussed in section 6, indicates MLPs play a crucial tole in these gradient descent computations, not the particular attention head (head 0, layer 0) under scrutiny. Additionally, the discontinuity we observed around division by 0 in Section 7.2 would be illogical under a gradient descent framework, as multiplication and subtraction by zero are defined and continuous, while it is the expected behavior for a division to have such a discontinuity.

11 Larger model

We train a more complex transformer consisting of 6 layers and 4 heads, specifically targeting a 20-dimensional input space using curriculum learning [Bengio et al., 2009]. The final performance metrics of this model are detailed in the table below, which includes various forms of loss measurements. Notably, *restricted loss* indicates the model's performance when the norm head (defined by the highest norm score) is limited to accessing only x values. In contrast, *excluded loss* measures performance when the norm head is prohibited from accessing x values. *Excess loss* is the ratio of the model's loss to the baseline Ordinary Least Squares (OLS) loss.

Table 3: Results on a larger model

Loss Type	Overall Loss	Excess Loss	Excluded Loss	Restricted Loss
Values	8.6138	1.6817	18.7864	8.2151

12 Conclusions

In this paper, we demonstrate the application of mechanistic interpretability on numerical transformers, extending the traditional use of these techniques beyond their typical domain in natural language processing. We introduce the technique of numerical patching, a method for isolating and understanding the specific functions implemented by the OV matrix of an attention head within the transformer model. Numerical patching has enabled us to show the function implemented by a specific attention head. By employing this method, we have identified a 'normalization head'. Our technique could offer a new lens through which the machine learning community can enhance the transparency and functionality of machine learning models.

References

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes, 11 2018. URL https://arxiv.org/abs/1610.01644.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, 2009. doi: 10.1145/1553374.1553380.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *arxiv.org*, 05 2020. URL https://arxiv.org/abs/2005.14165.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv:2303.12712 [cs]*, 03 2023. URL https://arxiv.org/abs/2303.12712.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability, 10 2023. URL https://arxiv.org/abs/2304.14997.

Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes, 01 2023. URL https://arxiv.org/abs/2208.01066.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks, 10 2021. URL https://arxiv.org/abs/2106.02997.

Callum McDougall, Arthur Conmy, Cody Rushing, Thomas McGrath, and Neel Nanda. Copy suppression: Comprehensively understanding an attention head, 10 2023. URL https://scholar.google.com/citations?view_op=view_citation&hl=en&user=GLnX3MkAAAAJ&citation_for_view=GLnX3MkAAAAJ:9ZlFYXVOiuMC.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. *ICLR*, 01 2023. doi: 10.48550/arxiv.2301.05217.

Catherine Olsson and Neel Nanda. In-context learning and induction heads, 03 2022. URL https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv:2201.02177 [cs]*, 01 2022. URL https://arxiv.org/abs/2201.02177.

Thrun, Pratt Sebastian, and Lorien. *Learning to Learn*. Springer US, 1998. doi: 10.1007/978-1-4615-5529-2.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 06 2017. URL https://arxiv.org/abs/1706.03762.

Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent, 05 2023. URL https://arxiv.org/abs/2212.07677.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods, 10 2023. URL https://openreview.net/forum?id=Hf17y6u9BC.