Abrupt Learning in Transformers: A Case Study on Matrix Completion

Pulkit Gopalani
University of Michigan
gopalani@umich.edu

Ekdeep Singh Lubana
Harvard University
ekdeeplubana@fas.harvard.edu

Wei Hu
University of Michigan
vvh@umich.edu

Abstract

Recent analysis on the training dynamics of Transformers has unveiled an interesting characteristic: the training loss plateaus for a significant number of training steps, and then suddenly (and sharply) drops to near–optimal values. To understand this phenomenon in depth, we formulate the low-rank matrix completion problem as a masked language modeling (MLM) task, and show that it is possible to train a BERT model to solve this task to low error. Furthermore, the loss curve shows a plateau early in training followed by a sudden drop to near-optimal values, despite no changes in the training procedure or hyper-parameters. To gain interpretability insights into this sudden drop, we examine the model's predictions, attention heads, and hidden states before and after this transition. Concretely, we observe that (a) the model transitions from simply copying the masked input to accurately predicting the masked entries; (b) the attention heads transition to interpretable patterns relevant to the task; and (c) the embeddings and hidden states encode information relevant to the problem. We also analyze the training dynamics of individual model components to understand the sudden drop in loss.

1 Introduction

Large Language Models (LLMs) have revolutionized the field of natural language processing (NLP). However, there are still gaps in our understanding of these models, leading to challenges in controlling their behavior. As a pertinent example, the training of these models appears to demonstrate *sudden* improvements in metrics correlated with various capabilities [8], prompting questions about whether learning of a given capability can be predicted by tracking predefined progress measures and *why* such sudden changes occur. If undesirable capabilities can suddenly 'emerge' (despite any explicit supervision for them) [16], such sudden changes can be a challenge for AI regulation [21].

To better understand such sudden changes during model training, this work investigates training BERT [12] on the classical mathematical task of low-rank matrix completion (LRMC) [6]. Making an analogy with masked language modeling (MLM), where sudden learning of syntactical structures was recently demonstrated [8], we argue matrix completion captures the core aspect of this learning problem (Fig. 1): given some relevant context (observed tokens), fill the missing elements (masked tokens). Specifically, we assume access to a matrix with some fraction of its entries missing, and would like to complete the missing entries of this matrix assuming the ground truth matrix is low-rank. We find that despite being a simplified abstraction of MLM, this setting already demonstrates *a sharp* decrease in loss as the model undergoes training (Fig. 1 (B)), preceded by a loss plateau for a significant number of training steps (akin to Chen et al. [8]). The simplicity of our setting further affords us interpretability, as we find that the point of sudden drop coincides with a precise change in how the model solves the task—we call this change an *algorithmic transition*. Specifically, we show that the pre–transition model simply copies the input (predicting 0 at masked positions), while the post–transition model accurately predicts missing values at masked positions. To perform the latter,

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

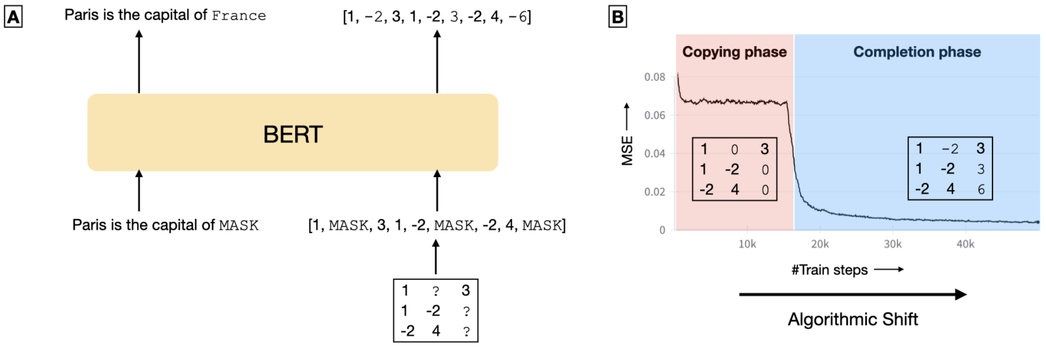


Figure 1: (**A**) **Matrix completion using BERT**. Similar to completing missing words in an English sentence in MLM, we complete missing entries in a masked low–rank matrix. (**B**) **Sudden drop in loss**. During training, the model undergoes an *algorithmic shift* marked by a sharp decrease in mean–squared–error (MSE) loss. Here, the model shifts from simply copying the input (*copying phase*) to computing missing entries accurately (*completion phase*).

distinctive changes occur in the model's attention heads during the period of sudden drop, wherein the model learns to identify relevant positional information to combine various elements in the input matrix and compute missing entries for matrix completion. We perform a range of interventions on the input, model (before and after the transition), and training process to further understand this phenomenon, leading to the following observations.

- **Pre–transition: Copying the Input Matrix** Before the transition, the model is simply copying the input matrix both at observed entries as well as missing entries, predicted value for missing entries being nearly 0. The attention maps at this stage do not correspond to a particularly interpretable structure, and contribute little to the model output.
- **Post–Transition: Computing Missing Entries** After the transition, the model accurately completes the missing entries, while still copying observed entries. The attention maps at this stage clearly demonstrate that the model 'attends' to relevant tokens in the input, and the attention layers are crucial for accurate matrix completion. Interestingly, the post–transition model can outperform the classical nuclear norm minimization algorithm for matrix completion, suggesting that it does not simply recover this algorithm.
- **Model Components and Sudden Drop** We analyze the training dynamics of individual components, keeping other components fixed to their final values. We find that different components converge to their optimal values at quite different points during this training.

2 Preliminaries

2.1 Problem Setup

MLM and LRMC In masked language modeling (MLM), a fraction of tokens in the input sequence are masked out and the model is required to predict the correct token for those masked entries. In this setup, the model has access to both the tokens before and after the current token for computing the prediction. Low-rank matrix completion has a similar structure: given a matrix (assumed low–rank) with a fraction of its elements available, the goal is to predict missing entries. For a matrix $X \in \mathbb{R}^{n \times n}$, denote its observed entries by the set $\Omega \subset [n] \times [n]$, and the set of missing entries $\Omega^{\mathsf{c}} = [n] \times [n] \setminus \Omega$. Formally, the problem is

$$\min_{U} \ \ \text{rank}(U) \qquad \text{s.t.} \ \ U_{ij} = X_{ij} \ \ \forall (i,j) \in \Omega.$$

Importantly, both problems (MLM and LRMC) have the same goal—predict the missing entries in the input, i.e., either the language tokens (MLM) or matrix elements (LRMC).

Matrix Completion using BERT BERT [12] is an encoder-only Transformer architecture used widely for MLM. For an input sequence of tokens $[t_1, t_2, \ldots, t_L]$, the output is a sequence of

$D-$dimensional 'hidden states' $[e_1, \ldots, e_L]^\top \in \mathbb{R}^{L \times D}$, that is used for prediction. We train a BERT model TF_θ to predict missing entries in a low–rank masked matrix $\tilde{X}$. For model output $\hat{X} := \mathsf{TF}_\theta(\tilde{X}) \in \mathbb{R}^{n \times n}$, the training objective $L := L(\theta)$ is the mean-squared-error (MSE) loss over all entries,

$$L(\theta) = \frac{1}{n^2} \sum_{i,j=1}^{n} (X_{ij} - \hat{X}_{ij})^2.$$

In our experiments, data for matrix completion is generated as

$$X = UV^\top; \quad U, V \in \mathbb{R}^{n \times r}, \;\; U_{ij}, V_{ij} \overset{\text{iid}}{\sim} \text{Unif}[-1, 1] \;\; \forall i, j \in [n] \times [r]$$

so that X has rank at most r. To mask entries at random, we sample binary matrices $M \in \{0, 1\}^{n \times n}$ such that $M_{ij} = 0$ with probability p_{mask}, and 1 otherwise; that is, $\Omega = \{(i, j) \mid M_{ij} = 1\}$.

Nuclear norm minimization Nuclear norm minimization [6] is a widely used convex optimization approach to LRMC; for completeness, we compare our trained models to this approach. Since rank is not a convex function of the matrix, one modifies the low rank completion problem by defining the nuclear norm $\|U\|_*$, i.e., sum of singular values of a matrix U. The overall optimization problem is as follows.

$$\min_U \|U\|_* \qquad \text{s.t. } U_{ij} = X_{ij} \;\; \forall (i, j) \in \Omega. \tag{1}$$

2.2 Experiments

Training We use a 4–layer, 8–head BERT model [40] for 7×7 (rank-2) matrices, with 'absolute' positional embeddings, no token–type embeddings, and no dropout. We fix $p_{\text{mask}} = 0.3$ for training, and 256 matrices are sampled as training data at each step (in an 'online' training setup). We use Adam optimizer with constant step size $1\text{e}{-4}$ for 50000 steps, without weight decay or warmup. In addition to L, we track MSE over observed and masked entries,

$$L_{obs} = \frac{1}{|\Omega|} \sum_{(i,j) \in \Omega} (X_{ij} - \hat{X}_{ij})^2, \quad \text{and} \quad L_{mask} = \frac{1}{|\Omega^\mathsf{c}|} \sum_{(i,j) \in \Omega^\mathsf{c}} (X_{ij} - \hat{X}_{ij})^2.$$

Please see Appendix D for details on tokenizing matrices and other experimental details. Code is available at this `https://github.com/pulkitgopalani/tf-matcomp`.

Compute Resources For 7×7 matrices (training and testing), we used a single {V100 / A100 / L40S} GPU. A single {A40 / A100 / L40S} GPU was used for matrices of order 10, 12, 15.

3 Sudden Drop in Loss

In our training setup, the model converges to a final MSE of approximately $4\text{e}{-3}$ – that is, it can solve matrix completion well (as in Fig. 3, this MSE is lower than nuclear norm minimization). Fig. 2 demonstrates the loss dynamics over the course of training the model on this task.

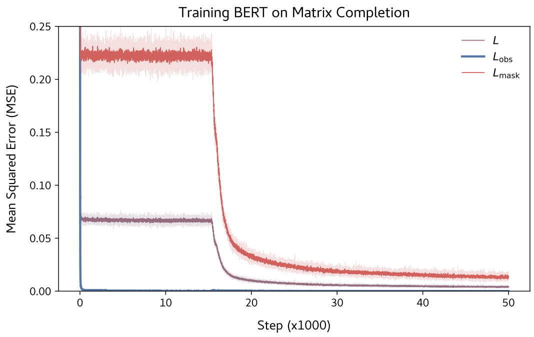


Figure 2: Sharp reduction in training loss.

Interestingly, we observe a sudden drop in training loss at approximately step 15000. This sudden drop in loss is reminiscent of *phase transitions* in physical systems, that are characterized by sudden observable changes in the system on continuous variation of some parameter (here equivalent to the number of training steps). Motivated by this similarity, we analyse the 'pre–shift' model at step 4000, and 'post–shift' model at the end of training, i.e., step 50000 to understand model properties and sudden drop in loss.

3.1 Before the Algorithmic Shift – Copying Phase

Since the value of L_{obs} remains quite low in the first phase of model training (Fig. 2), we ask: *what algorithm does the model use for predicting matrix entries in this phase?*

We find that the model learns to copy the input verbatim in the first phase (with output 0 for missing entries), verified through token interventions (Sec. 3.1.1) and by investigating the contribution of attention heads (Sec. 3.1.2) towards the output.

3.1.1 Verifying Copying via Token Intervention

To rigorously verify that the pre-shift model indeed copies the input, we replace the masked elements in the 7×7, rank-2 input by the token corresponding to some $m \in \mathbb{R}$. For such input, we would like to see whether the model implements copying and outputs m at the masked positions. In this setup for model output $\hat{X}$, MSE at observed positions is L_{obs}, and for masked positions the MSE is defined as

$$L'_{mask} = \frac{1}{|\Omega^{\mathsf{c}}|} \sum_{(i,j) \in \Omega^{\mathsf{c}}} (\hat{X}_{ij} - m)^2.$$

L_{obs} and L'_{mask} for this experiment averaged over 512 samples are compiled in Table 1 (Appendix A). The small loss values confirm that model output matches the ground truth at observed positions, while at masked positions it outputs a value nearly equal to m. When the mask token is MASK (i.e., no replacement), we set $m = 0$, indicating that the model outputs 0 at the masked locations.

To generalize this observation to OOD matrices, we sample uniform random 7×7 matrices for input; i.e., all entries in the matrix are i.i.d. uniformly in $[-1, 1]$. Importantly, these matrices do not necessarily have a low–rank structure. With these matrices as input to the same pre–shift model as before, we find that model still copies the input (Table 1). This confirms that the model is indeed not 'computing' any entries in the sense of low–rank matrix completion, and simply copies all entries, masked or observed.

3.1.2 Attention Heads – Mostly Inconsequential

Attention heads at this stage (Fig. 22a) do not appear to attend to tokens in an interpretable manner. Since the model is copying the input, and does not need to combine different tokens, Attention heads should not affect the model output at this stage. To confirm that this is indeed the case, we do the following tests.

Uniform Ablation Uniform ablation entails replacing the softmax probabilities in an $n \times n$ attention head by $1/n^2$ for all elements i.e. 'force' the model to equally attend to all tokens (Sec. 4.6, [22]). On such an intervention in our case, there is negligible change in MSE at both observed and masked positions. Averaged over 256 samples, $L_{obs} = 3.4\text{e}{-4}$ and $L_{mask} = 0.2236$ when using all attention heads; whereas, on ablating all heads, these values are $3.2\text{e}{-4}$ and 0.2236 respectively. *The negligible change in MSE supports the hypothesis that attention does not contribute to the model output at this stage.*

Model Switching In the extreme case, what if we replace the model weights for some component to check for changes to the output? In model switching, we 'transplant' the attention key, query and value weights in the pre–shift model to those from the post-shift model. Averaged over 256 samples, L_{obs} is 5e–3, that is similar to the optimal total MSE (L) obtained at the end of training, while $L_{mask} = 0.2246$, similar to the values obtained without such replacement. *This shows that replacing the pre–shift attention weights by the optimal ones does not significantly affect L_{obs}, L_{mask} – highlighting that attention layers have little effect on the model output at this stage.*

3.2 After the Algorithmic Shift – Matrix Completion Phase

In this section, we focus on the model properties in the post–shift phase (specifically, at the end of training at 50000 steps). Since L are near–optimal in this setting, we ask : *What algorithm is the model using for completing missing entries? For example, is it implementing the classical nuclear-norm minimization algorithm?* For the second question, we show below that the BERT model is *not* implicitly implementing nuclear norm minimization for completing missing entries in the input.

Nuclear Norm Minimization We use CVXPY [13] to solve low–rank matrix completion using nuclear–norm minimization at various levels of p_{mask}, comparing it to the output of a BERT model trained on $p_{\text{mask}} = 0.3$. We find that BERT performs better than nuclear norm minimization with respect to MSE; at the same time, the nuclear norm of BERT solution is larger (Fig. 3).

To verify if the model implicitly optimizes a different objective for nuclear norm minimization, we also compare to the regularized version of the above problem ($\lambda > 0$),

$$\min_U \left[\frac{1}{|\Omega|} \sum_{(i,j)\in\Omega} (U_{ij} - X_{ij})^2 + \lambda \|U\|_* \right]$$

We find that this is not the case, as for various values of λ, BERT still outperforms regularized MSE minimization w.r.t. MSE (Appendix B). *This confirms that the model is not implementing nuclear norm minimization as its algorithm for computing missing entries.*

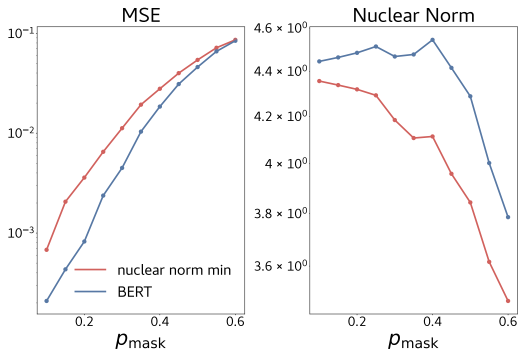


Figure 3: **BERT v. Nuclear Norm Minimization**. Comparing our model (trained with $p_{\text{mask}} = 0.3$) and nuclear norm minimization on the matrix completion task at various levels of p_{mask}. The difference in MSE and nuclear norm of solutions obtained using these two approaches indicates that BERT is not implicitly doing nuclear norm minimization to complete missing entries.

We now move to an interpretability based analysis of the model behavior, to attempt to extract useful signal about the implemented algorithm, analysing model behavior for observed and missing entries separately in the following sections.

3.2.1 Observed Entries

Uniform Ablation As in Sec. 3.1.2, to quantify the effect of attention heads at this stage, we uniformly ablate *all* attention heads in the post-shift model. Averaged over 256 samples, this leads to $L_{obs} = 9.2\text{e}{-5}$ without ablation, and 3.7e$-$3 with ablation (close to the value of L at the end of training). However, L_{mask} increases from 0.0128 to 0.2183, approximately the value of L_{mask} in the loss plateau before sudden drop. *This difference in effect of ablating attention heads confirms that they are much more important for predicting missing entries than for observed entries.*

Model Switching We repeat the model switching experiments from Sec. 3.1.2 in the reverse direction i.e. 'transplant' attention key, query, value weights from pre–shift model to the post–shift model. Note that this direction of weight switching is stronger, in the sense that the learnt information in attention layers is removed. We find that on this modification, $L_{obs} = 9.5\text{e}{-4}$ averaged over 256 samples; that is, the observed loss is still not too large. *This test confirms that the prediction at observed entries is not substantially affected by the attention layers.*

Position Sensitivity Finally, since the attention mechanism crucially depends on token positions, we intervene on this component of the model by randomly permuting its positional embeddings. Formally, the embedding originally for position i in the input now represents position $\pi(i)$ for some random permutation $\pi : [n^2] \to [n^2]$. Averaged over 256 samples, $L_{obs} = 2.4\text{e}{-4}$, whereas $L_{mask} = 0.5687$, indicating that the observed positions are negligibly affected compared to masked positions due to this intervention.

These results support our 'sub–algorithm' hypothesis; (a) since positional information is intuitively not required for the copying sub-algorithm, L_{obs} remains low; and (b) L_{mask} increases significantly, demonstrating that removing positional information is detrimental to accurately computing missing entries.

3.2.2 Missing Entries

To confirm that attention heads causally affect the model output for missing entries, in addition to uniform ablations, we perform *causal interventions* (activation patching) [42] on the hidden states just after the attention heads. This involves replacing the hidden state after an attention head for input

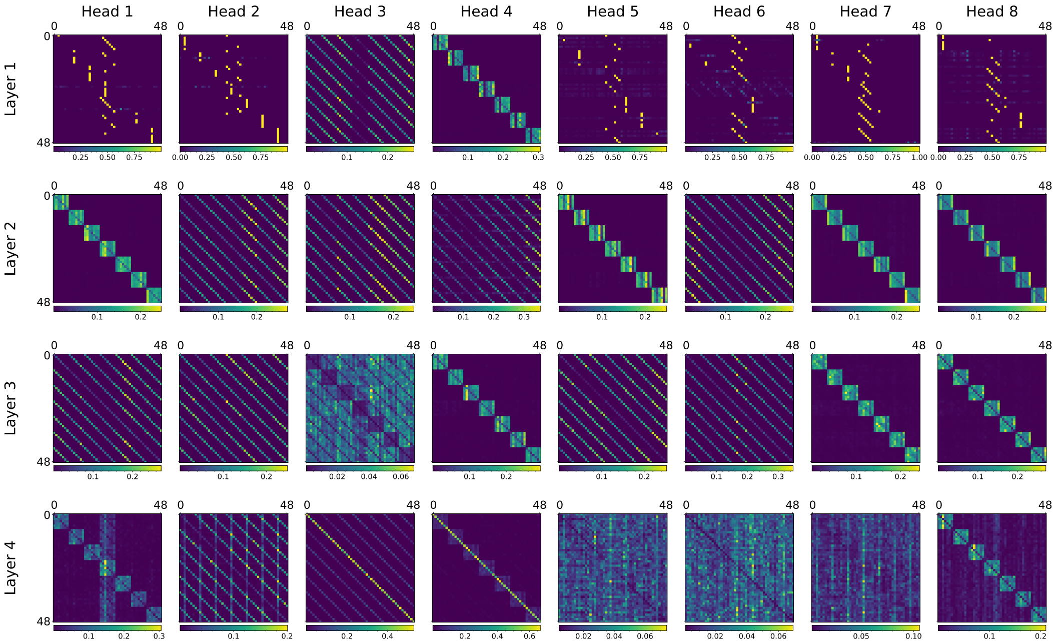


Figure 4: **Attention heads in post–shift model attend to specific positions.** For example, (Layer 2, Head 1) attends to elements in the same *row* as the query element, and (Layer 2, Head 2) attends to elements in the same *column* as the query element. (These attention matrices are an average over multiple independent matrix and mask samples.)

A with the hidden state obtained at the same attention head, but for a different input A'. Ideally, if that head is causally relevant to the output, then such an intervention should steer the model towards the output for A', instead of A. We find in our case that for $A = X$ and $A' = -X$, such an intervention on all attention heads clearly steers the model output at missing entries towards $-X$ (more details in Appendix F).

Structure in Attention Heads Denote attention head H in layer L by the tuple (L, H). We can group the attention heads depending on the specific regions of the input matrix they attend to,

1. **[Row Head]** same row as the query element – 'block–diagonal' patterns, e.g. (2, 1);
2. **[Column Head]** same column as query element – 'off–diagonal' patterns e.g. (2, 2); and
3. **[Identity Head]** query element itself – 'diagonal' patterns in the last layer, e.g. (4, 3).

There are also some other attention heads that do not neatly fit into either of these 3 categories—for example, all heads in layer 1 except (1,3), (1,4); (3,3); (4,2), (4,5–7). In this context, we note that uniformly ablating heads (3,3), (4,2), (4,5–7) gives $L_{obs} = 9.36\text{e}{-5}$, $L_{mask} = 0.01575$ compared to $L_{obs} = 9.44\text{e}{-5}$, $L_{mask} = 0.01428$ without ablation, i.e. these uninterpretable heads do not significantly affect the output.

Attention Heads with 'Structured Masking' Since the maps in Fig. 4 are averages over multiple random masks and input matrices, it is difficult to derive more fine–grained insights into the model computation. To address this, we generate inputs with specific mask structure, see for example Fig. 5. This implies that while averaging the attention probabilities over different input matrices, the mask i.e. Ω^c remains the same. This step helps us highlight how an attention head attends to input elements based on the element being masked or observed. From the results in Fig. 5, we find clear evidence that different attention heads focus on specific parts of the input. For instance,

1. **[Masked–Row Head]** (2, 1), (3,4) and (4,8) are mainly active only at the masked rows, and therein attends to the only observed position in those rows.

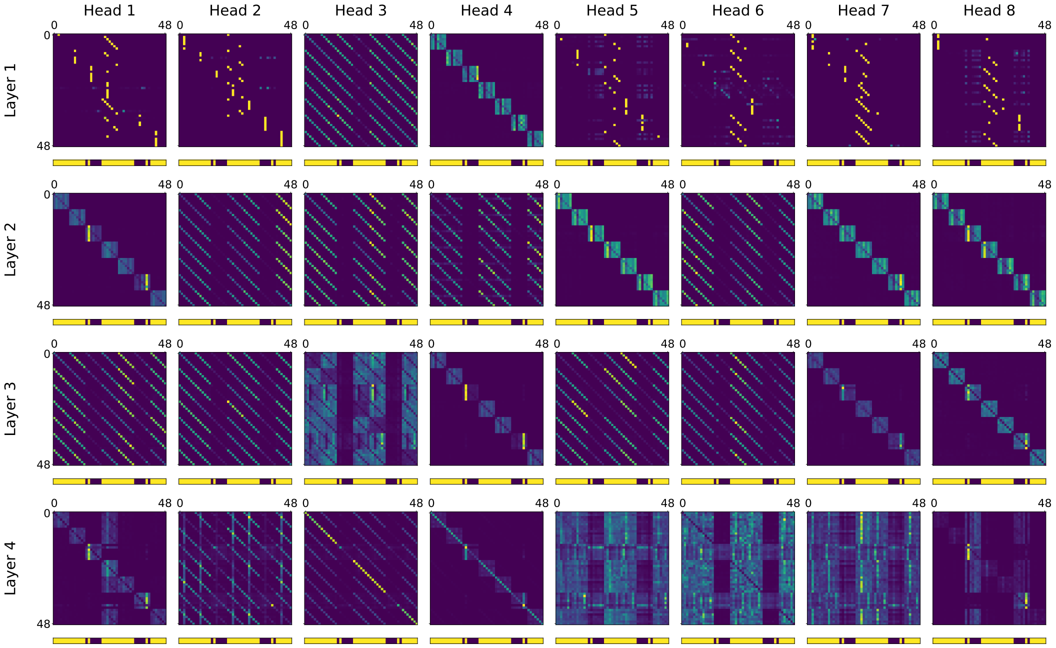


Figure 5: **Attention heads with specific mask structure in inputs**. We can derive fine-grained insights about the functions of individual heads in this setup by using a specific mask structure for all input matrices. (Mask appended below each plot, blue denotes missing entries). For example, multiple attention heads like (Layer 2, Head 2) have negligible attention weight at missing positions in the input matrix, implying that these heads attend only to observed entries in the column of the query element. Further, (Layer 2 Head 1) and similar heads have larger attention weights for the rows with missing entries, and in those rows they attend to the sole observed element.

2. **[Observed–Copy Head]** (4,3) and (4,4) correspond roughly to an identity map, slightly deviating in the masked rows. In these cases, again the maximal attention score corresponds to the only observed position in these rows.
3. **[Mask–Ignore Heads]** Further, there are multiple 'parallel off-diagonal' heads that completely ignore the masked rows for their computation. These heads include (2,2–4), (2,6); (3,2), (3,3), (3,5). Additionally, there are also attention heads like (3,1), (3,6) that attend to only the observed element of each masked row. Collectively these heads act as 'mask-ignore' heads, attending to only observed entries, and using this information to compute missing entries.
4. **[Longest–Contiguous–Column Heads]** There also exist attention heads that respond systematically to changes in the mask. For example, consider attention heads (2, 5), (2, 7), (2, 8) in Fig. 23. For each row, these heads attend to the element in the 6th and 2nd column respectively for part (a) and (b). On a closer look, the connecting link between these two mask patterns is that, the longest contiguous unmasked column is exactly the column that these heads attend to. We hypothesize that this information is somehow used by the model in its inner computation for masked entries.
5. **[Input–Processing Heads]** Finally, Heads (1,1–2), (1, 5–8) do not fall in any of the categories above . These heads are mostly static across different mask / input variations (for example, comparing Fig 4 and 5), and the patterns suggest that these heads almost exclusively focus on the middle row of the input matrix and some other elements. A possible function of these heads is to process positional and token embeddings (input to the first layer) so that this information can be used appropriately in the subsequent layers.

To quantitatively assess the effect of these attention heads on the model output, we also perform uniform ablations on each sub–group separately (Appendix L), and find that the groups significantly affect the output, to varying degrees depending on the specific group.

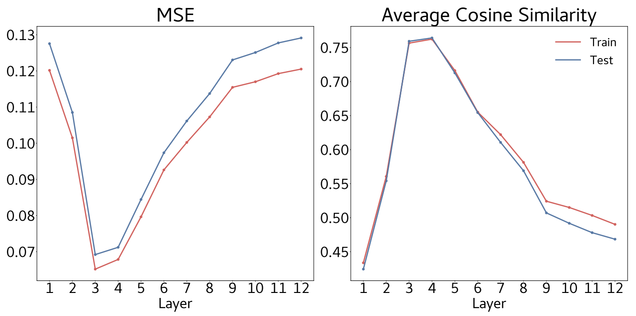


Figure 6: **Hidden states encode input information**. We probe for the row of each element in the masked matrix input, replacing missing entries by 0. We find that layers 3 and 4 have a much lower MSE and a much larger cosine similarity compared to other layers in the model. Hence, these layers somehow 'store information' about the masked input matrix.

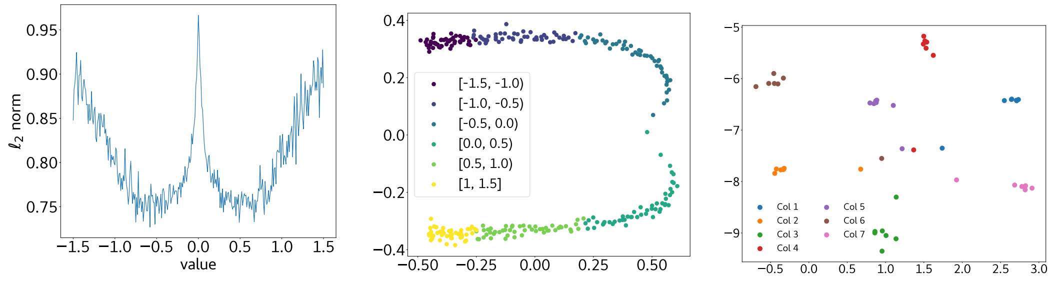
Probing We probe for properties of the input matrix in the hidden states of the model, to concretely determine how the model computes the output. We use our 12–layer model in this case, to enhance contrast between probing results in different layers.

Specifically, for every element in the input, we fit a linear probe [3] on its hidden state after a given layer, mapping the hidden state to the $n-$dimensional masked row that this element belongs to (missing entries are replaced by 0). That is, element at position (i, j) maps to the $7-$dimensional vector $\tilde{X}_i$. The results for this experiment in Fig. 6 demonstrate that the hidden states at layer 3 and 4 in the model correlate quite strongly with the probe target, compared to other layers. *This result suggests that the model tracks input information in its intermediate layers and possibly uses it for computing missing entries.*

We also probe for the true matrix element at missing entries, and find that the hidden states at these positions get gradually more correlated with depth (through linear probing). Further, we also attempt to extract information about singular vectors of the ground truth matrix from the hidden states through linear probing, though are unable to conclusively do so. We discuss these results in Appendix K.

3.3 Role of Embeddings

Token Embeddings The ℓ_2 norm of token embeddings corresponding to values from -1.5 to 1.5 is symmetric w.r.t. 0 as seen in Fig. 7a. Further, the PCA of token embeddings in Fig. 7b shows that the embeddings have a separable structure based on the sign of the real–valued input (y–axis), and continuous variation w.r.t. magnitude of input (x–axis). Importantly, unlike other metrics, token embeddings do not seem to abruptly change only at step 15000; rather, the final structure appears before the sudden drop in loss. Similar to [27], we compute the top–2 principal components of the token embeddings at the final step (50000), and project the token embeddings at intermediate training



(a) (b) (c)

Figure 7: **Embeddings demonstrate relevant structure.** In the post–shift model, positional and token embeddings exhibit properties demonstrating that the model has learnt relevant information about the matrix completion problem. (a) ℓ_2 norm of token embeddings is symmetric around 0. This aligns with the intuition that the norm of token embeddings should depend only on the magnitude of the input, and not on its sign. (b) Top–2 principal components of token embeddings correspond to the magnitude and sign of the real valued input. In our case, the 'y-axis' denotes sign of input, and the 'x-axis' denotes the magnitude of the input value. (c) Positional embeddings of elements in the same column cluster together in the t–SNE projection, showing that the model uses positional information relevant to the matrix completion problem.

steps on these components. The results (Fig. 9, Appendix C) show that the embeddings align very closely to the final arrangement before the actual drop. This is as expected, since the model needs to learn what the tokens actually represent on the real line, before it can use those values for completing missing entries. This also explains to some extent why the model implements copying before the sudden drop, since accurately learning token embedding-unembedding is sufficient for that task.

Positional Embeddings In the t-SNE projection of positional embeddings, positions in the same column tend to cluster together as seen in Fig. 7c. This is non–trivial because we have not used any marker tokens to mark the end of a row or column. Further, note that in contrast to token embeddings, positional embeddings do not have a continual evolution in structure – Fig. 10 (Appendix C) shows that the clustering appears only after the sudden drop (step 20000 and after). This along with the evolution of attention heads (Sec. 3.1, 3.2) aligns with how the pre–shift model copies observed entries with little effect from ablating attention heads or positional embeddings (Sec. 3.2.1).

4 Sudden Drop in Loss – Role of Model Components

Is it possible to analyse training dynamics of individual model components to derive insights about the full model training? This is motivated by the findings in the previous section on embeddings, and in Section 3.1; the pre–shift model does not use Attention layers for its computation in that stage, and relies on other components to copy input entries. Hence, in our case, the sudden drop corresponds in large part to learning the right Attention patterns (see Appendix M.1). To analyse training dynamics of different model components, we choose (a set of) components – Attention layers, MLP layers, Positional Embeddings and Token Embeddings, randomly initialize them and freeze the weights of other components to their values at the final step of training (Fig. 8).

We find that (a) MLPs and Token Embeddings converge without any observable plateau or sudden drop in loss; (b) for other components, the dynamics resemble those for the full model training (i.e. plateau and then sudden drop), and (c) Positional embeddings show the longest plateau in loss.

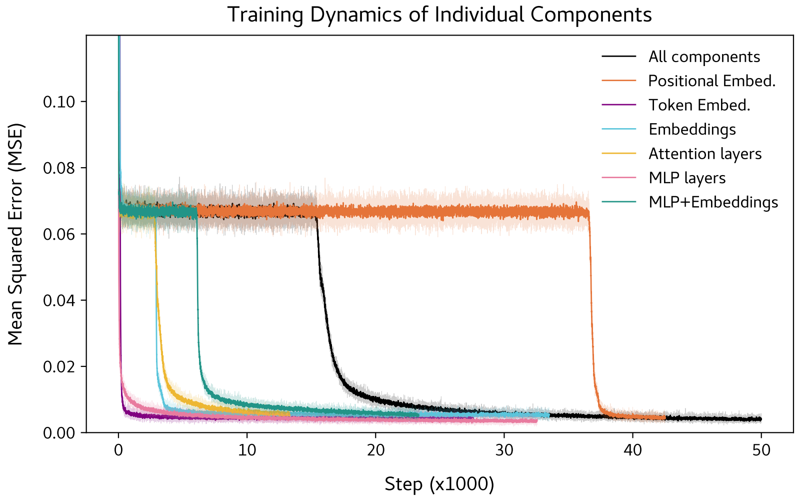


Figure 8: **Individual model components have distinct training dynamics**. Training individual model components, initializing others to their final value ('All components' indicates normal training). There is no loss plateau for token embeddings and MLP layers, in contrast to positional embeddings, where the sudden drop occurs just before step 40000. In all other cases the sudden drop occurs before the sudden drop in usual training.

Additional Results To further understand the effect of data and model properties on the sudden drop in loss, we train

- a 2–layer, 2–head GPT model on the matrix completion task (Appendix G);
- models of different depth (number of layers) and width (hidden state dimension) (App. H);

- our model on (mixture of) matrices of different sizes keeping the rank fixed at 1 (App. I);
- a 12-layer, 12-head model on 10×10 matrices of multiple ranks (separately) (App. I);
- our model on input matrix entries (U_{ij}, V_{ij}) i.i.d. $\sim \mathcal{N}(0, 1)$ instead of Unif$[-1, 1]$ (App. J.1); we also analyze test–time OOD performance of our model in App. J.2.

5 Related Work

[27, 28, 32, 41, 38] analyse 'grokking', the sudden emergence of generalization during model training. In the context of training dynamics of MLM, [8] analyses 'breakthroughs' (sudden drop in loss and associated improvement in generalization capabilities of the model), specifically for BERT. They show that the breakthrough marks the transition of the model to a generalizing one. Their work however is focused on language tasks, distinct from our setting which is mathematical (and hence more controllable) in nature. We also note that their work is not in the online training setting; our setup is online in the sense of sampling new data at every step of training.

Mathematical problem solving capabilities of Transformers have been a topic of interest lately [24, 7, 4]. In fact, [24] show that learning addition from samples is equivalent to low–rank matrix completion. Further, [7] show that it is possible to train a transformer based model to solve various linear algebraic tasks e.g. eigendecomposition, matrix inversion, etc.; however, to the best of our knowledge, interpretability studies for such tasks have not been conducted before. For interpretability in simpler math tasks, [18] mechanistically analyse GPT-2 small on predicting whether a number is 'greater-than' a given number, by formulating the problem as a natural language task. [35, 36, 10] analyse BERT from an interpretability perspective. More recently, there has been a line of research works analysing decoder based models to reverse–engineer the mechanisms employed by these models, termed as 'mechanistic interpretability' [14, 31, 32, 39, 11, 33, 25, 26, 23, 34, 20]. We note that our setting is distinct from the recent work on solving mathematical tasks like linear regression through 'in–context' learning in transformers [4, 1, 9, 15, 17, 2, 30, 37]. Whether our model learns to implicitly 'implement' an optimization procedure as shown in some of these is an open question. We discuss related work in more detail in Appendix E.

6 Conclusion

We trained a BERT model on matrix completion, and analyzed it before and after the sudden drop in training loss (algorithmic shift) to interpret the algorithm being learnt by the model, and gain insight on why such a sharp drop in loss occurs.

It is evident in our analysis that both before and after the shift, the model does not really compute anything at observed positions, and simply copies these entries. For missing entries, we have shown that the model learns useful abstractions rapidly through the algorithmic shift. Mathematically formulating what algorithm the model employs to implement matrix completion for missing entries is a direction for future work.

Since our work is primarily interpreting model training and mechanism, all experiments are with small scale matrices (largest being 15×15), and the current method would likely need modifications to scale to larger matrices. Finally, we only intended to study Transformers on matrix completion as a toy task from an interpretability viewpoint, and do not advocate replacing existing efficient solvers for matrix completion with our approach.

Societal Impact We study Transformer based models on their ability to solve a mathematical task (matrix completion) and the associated training dynamics. The work focuses on Transformer interpretability, aiding in improving our understanding of these models and their training and thus we do not foresee any negative societal impact of our work.

Acknowledgements We thank Yu Bai, Andrew Lee, Naomi Saphra and anonymous reviewers for their helpful comments. WH acknowledges support from the Google Research Scholar Program. ESL's time at University of Michigan was supported by NSF under award CNS-2211509 and at CBS, Harvard by the CBS-NTT Physics of Intelligence program.

References

[1] Kwangjun Ahn, Xiang Cheng, Hadi Daneshmand, and Suvrit Sra. Transformers learn to implement preconditioned gradient descent for in-context learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[2] Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. In *The Eleventh International Conference on Learning Representations*, 2023.

[3] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes, 2017.

[4] Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[5] Boaz Barak, Benjamin Edelman, Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang. Hidden progress in deep learning: Sgd learns parities near the computational limit. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 21750–21764. Curran Associates, Inc., 2022.

[6] Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Found. Comput. Math.*, 9(6):717–772, 2009.

[7] Francois Charton. Linear algebra with transformers. *Transactions on Machine Learning Research*, 2022.

[8] Angelica Chen, Ravid Shwartz-Ziv, Kyunghyun Cho, Matthew L Leavitt, and Naomi Saphra. Sudden drops in the loss: Syntax acquisition, phase transitions, and simplicity bias in MLMs. In *The Twelfth International Conference on Learning Representations*, 2024.

[9] Xiang Cheng, Yuxin Chen, and Suvrit Sra. Transformers implement functional gradient descent to learn non-linear functions in context, 2024.

[10] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? an analysis of BERT's attention. In Tal Linzen, Grzegorz Chrupała, Yonatan Belinkov, and Dieuwke Hupkes, editors, *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Florence, Italy, August 2019. Association for Computational Linguistics.

[11] Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[13] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[14] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

[15] Deqing Fu, Tian-Qi Chen, Robin Jia, and Vatsal Sharan. Transformers learn higher-order optimization methods for in-context learning: A study with linear models, 2023.

[16] Deep Ganguli, Danny Hernandez, Liane Lovitt, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova Dassarma, Dawn Drain, Nelson Elhage, et al. Predictability and surprise in large generative models. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 1747–1764, 2022.

[17] Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[18] Michael Hanna, Ollie Liu, and Alexandre Variengien. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[19] David T. Hoffmann, Simon Schrodi, Jelena Bratulić, Nadine Behrmann, Volker Fischer, and Thomas Brox. Eureka-moments in transformers: Multi-step tasks reveal softmax induced optimization problems. In *International Conference on Machine Learning*, July 2024.

[20] Yifan Hou, Jiaoda Li, Yu Fei, Alessandro Stolfo, Wangchunshu Zhou, Guangtao Zeng, Antoine Bosselut, and Mrinmaya Sachan. Towards a mechanistic interpretation of multi-step reasoning capabilities of language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Singapore, December 2023. Association for Computational Linguistics.

[21] Margot E Kaminski. Regulating the risks of ai. *Forthcoming, Boston University Law Review*, 103:22–21, 2023.

[22] Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. Revealing the dark secrets of BERT. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4365–4374, Hong Kong, China, November 2019. Association for Computational Linguistics.

[23] János Kramár, Tom Lieberum, Rohin Shah, and Neel Nanda. Atp*: An efficient and scalable method for localizing llm behaviour to components, 2024.

[24] Nayoung Lee, Kartik Sreenivasan, Jason D. Lee, Kangwook Lee, and Dimitris Papailiopoulos. Teaching arithmetic to small transformers, 2023.

[25] Michael A. Lepori, Thomas Serre, and Ellie Pavlick. Uncovering intermediate variables in transformers using circuit probing, 2023.

[26] Tom Lieberum, Matthew Rahtz, János Kramár, Neel Nanda, Geoffrey Irving, Rohin Shah, and Vladimir Mikulik. Does circuit analysis interpretability scale? evidence from multiple choice capabilities in chinchilla, 2023.

[27] Ziming Liu, Ouail Kitouni, Niklas S Nolte, Eric Michaud, Max Tegmark, and Mike Williams. Towards understanding grokking: An effective theory of representation learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 34651–34663. Curran Associates, Inc., 2022.

[28] Ziming Liu, Eric J Michaud, and Max Tegmark. Omnigrok: Grokking beyond algorithmic data. In *The Eleventh International Conference on Learning Representations*, 2023.

[29] Ekdeep Singh Lubana, Kyogo Kawaguchi, Robert P. Dick, and Hidenori Tanaka. A percolation model of emergence: Analyzing transformers trained on a formal language, 2024.

[30] Arvind V. Mahankali, Tatsunori Hashimoto, and Tengyu Ma. One step of gradient descent is provably the optimal in-context learner with one layer of linear self-attention. In *The Twelfth International Conference on Learning Representations*, 2024.

[31] Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. Circuit component reuse across tasks in transformer language models, 2024.

[32] Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, 2023.

[33] Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models. In Yonatan Belinkov, Sophie Hao, Jaap Jumelet, Najoung Kim, Arya McCarthy, and Hosein Mohebbi, editors, *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, Singapore, December 2023. Association for Computational Linguistics.

[34] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

[35] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8, 2020.

[36] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, July 2019. Association for Computational Linguistics.

[37] Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, Joao Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 35151–35174. PMLR, 23–29 Jul 2023.

[38] Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. Grokked transformers are implicit reasoners: A mechanistic journey to the edge of generalization, 2024.

[39] Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations*, 2023.

[40] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.

[41] Zhiwei Xu, Yutong Wang, Spencer Frei, Gal Vardi, and Wei Hu. Benign overfitting and grokking in reLU networks for XOR cluster data. In *The Twelfth International Conference on Learning Representations*, 2024.

[42] Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *The Twelfth International Conference on Learning Representations*, 2024.

[43] Zhongwang Zhang, Pengxiao Lin, Zhiwei Wang, Yaoyu Zhang, and Zhi-Qin John Xu. Initialization is critical to whether transformers fit composite functions by inference or memorizing, 2024.

A Copying in Pre–Shift Model

Step	Input Samples	Mask = "MASK"		Mask = "0.44"		Mask = "–0.24"	
		L'_{mask}	L_{obs}	L'_{mask}	L_{obs}	L'_{mask}	L_{obs}
1000	Rank−2 matrices	1.4e-3	9.3e-4	7.6e-4	1e-3	6.7e-4	9.6e-4
	Random matrices	1.5e-3	8.3e-4	7.8e-4	1e-3	6.8e-4	9.6e-4
4000	Rank−2 matrices	3e-4	3.3e-4	4e-4	2.8e-4	3.7e-4	2.7e-4
	Random matrices	2.8e-4	3.5e-4	3.7e-4	3e-4	3.6e-4	2.8e-4
14000	Rank−2 matrices	1.6e-5	3.4e-5	1.8e-5	4.9e-5	5.1e-6	6.7e-5
	Random matrices	1.1e-5	3.7e-5	3.0e-5	8.5e-5	4.1e-6	1.1e-4

Table 1: Models at different steps before sudden drop implement copying, predicting the value for mask token at missing entries.

B Nuclear Norm Minimization

We use the regularized version of the nuclear norm minimization problem as detailed in Sec. 3.2, and obtain the following L, L_{obs}, L_{mask} for various values of λ. We average our results over 256 samples generated in the same way as the training data for BERT (including rounding off to 2 decimal places) for the sake of comparison.

λ	L_{obs}	L_{mask}	L
0.0005	$1.015\text{e}{-5}$	0.040728	0.012173
0.001	$3.686\text{e}{-5}$	0.040456	0.01211
0.0015	$7.959\text{e}{-5}$	0.040505	0.012155
0.002	0.00013769	0.040734	0.012264
0.005	0.00078591	0.043402	0.013516

C Embeddings Progress during Training

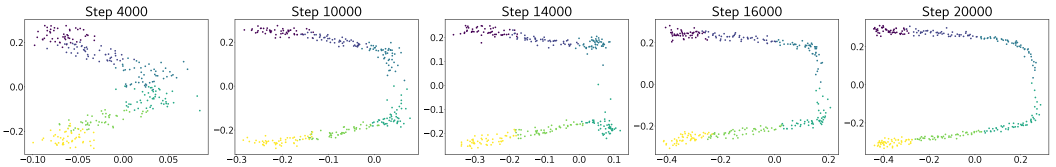


Figure 9: **Token embedding structure appears before sudden drop.** Projection of token embeddings along principal components of embeddings at step 50000. (Labels same as Fig. 7). Embeddings align with the principal components early on in training before the sudden drop in loss.

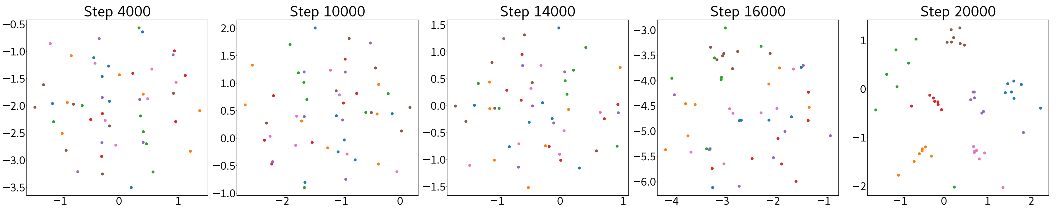


Figure 10: **Positional embedding structure appears after sudden drop.** Evolution of t-SNE projection of positional embeddings with training. (Labels follow the same color coding as Fig. 7). In this case, the positional embeddings show clustering some time after the sudden drop in loss has occurred at step 15000.

D Experimental Details

Online Training In online training, the data is sampled afresh from the distribution at every step. Since data is not partitioned into fixed train and test sets, we only analyze the training loss in all cases.

Tokenizing Matrices For tokenizing real values, we discretize the range $[-10, 10]$ in steps of size $\epsilon = 0.01$, and assign token IDs starting from 1; the mask token (MASK) is assigned ID 0. Input to the model is the tokenized masked sequence $X_{mask} = \mathrm{TOK}\left(\mathrm{Vec}(X \odot M)\right)$, where $\odot$ denotes the element-wise product, Vec denotes vectorizing the $n \times n$ matrix to a n^2-dimensional vector, and TOK denotes tokenization. Due to this preprocessing, in all cases X_{ij} is rounded to 2 decimals for computing L.

E Related Work

In the online training setup, [5] study the parity learning problem using a variety of model architectures, and show that 'hidden progress measures' can be used to track abrupt changes in model performance during training. [29] study abrupt learning in an autoregressive (GPT), language data setup, connecting learning the grammar to percolation on graphs. [19] discuss abrupt learning dynamics in the context of transformers and claim that the softmax function in Attention leads to longer training loss plateaus – however, reducing the length of plateau does not explain why the drop in loss is sudden and sharp when it occurs. [43] show that initialization of the model affects whether it learns to infer the compositional structure of the task, or simply memorizes the solution.

[4] show that in an in–context learning framework, a transformer based model can learn to select the most optimal statistical method for solving the task in the prompt, without explicitly being provided any information about the optimal method (called 'in–context algorithm selection' in their work). We emphasize that our setup is not in–context learning, and is quite distinct from [4] as far as the task being solved is concerned. However, whether the framework of layer-wise in-context gradient descent can also be used in our setup is a plausible and open direction for future work.

In [7], the author shows empirically that an encoder-decoder transformer can be trained to solve various linear algebraic tasks, such as eigendecomposition, SVD, matrix inverse etc. They support their findings by showing that the model generalizes to matrix distributions outside the training distribution to some extent, and that OOD performance can be improved by training on non-Wigner matrices. While many experiments in [7] also show a sudden jump in accuracy (Fig 1,2), they do not analyze why such a sudden jump occurs during optimization. In our work, we analyze the sudden drop and the model before and after it to derive insights into the sudden drop in loss in our setup.

[24] show that even a small transformer model can be trained to perform arithmetic operations like add, subtract, multiply accurately through appropriate data selection and formatting, and using Chain-of-Thought prompting. They further show that learning addition is connected to rank-2 matrix completion, and that the sudden jump in accuracy with increasing number of observed entries of the matrix is recovered when their model is trained on datasets of different sizes. This is because the size of the dataset for addition can be seen as the number of observed entries of the rank-2 matrix representing the addition table. We point out that while the task in this case is related to matrix completion, ours is a completely different setup, where the sudden drop happens with the number of training steps with each step consisting of 256 low-rank matrices, each with a fixed fraction (p_{mask}) of observed entries.

F Causal Intervention on Attention heads

In the uniform ablation setup, it is possible that setting the softmax probabilities to a given value might change the distribution of resultant hidden states, and consequently degrade model performance. A more principled technique to analyze the effect of a specific component is to replace the hidden state just after that component by hidden states on a different input, and analyze how this affects the final output [42]. In our case, we intervene on attention heads by replacing the hidden state after an attention head for input matrix X by the hidden state for input $(-X)$. Importantly, this change does not affect properties like rank of the input, and hence the hidden states obtained are from the same distribution as those for input X.

Step 1 Extract the hidden states for all attention layers from the model on some input matrix X; call these h_+. Concretely, these hidden states are obtained just after the matrix product of the softmax attention probabilties and the value matrix and hence before the output matrix product.

Step 2 Change the input to the model to $-X$, however, also replace the hidden states *just after* the attention layers with h_+ obtained in Step 1. Call the output of the model in this setup as $f_p(-X, X)$.

We observe that, the MSE between $f_p(-X, X)$ and X, averaged over 256 samples at masked positions is approximately 0.014 (this is comparable to optimal L_{mask}), compared to the MSE between $f_p(-X, X)$ and $-X$ being 0.8066. This demonstrates that the attention heads are causally relevant to the model output for missing entries.

G Autoregressive Setup

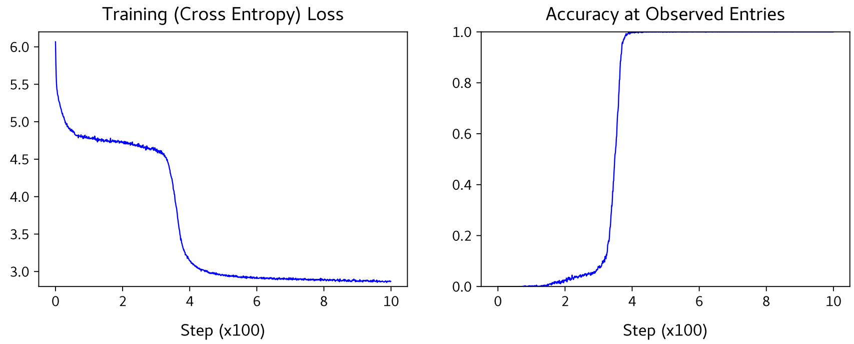


Figure 11: **GPT also shows abrupt learning for matrix completion**. Training a 2–layer, 2–head GPT model on 7×7, rank-2 matrices in the autoregressive training setup. Here, the model is trained using cross–entropy loss in a next–token prediction setup over full input sequences of the form $\tilde{X}_{11}, \tilde{X}_{12}, \ldots, \tilde{X}_{77}$, [SEP], $X_{11}, X_{12}, \ldots, X_{77}$ where $\tilde{X}, X$ are partially observed and fully observed matrices, flattened and tokenized as in the BERT experiments. We find that the sudden drop corresponds to the model learning to copy the observed entries in the input matrix. While we could not achieve performance comparable to BERT for missing entries, we believe it should be possible with some modifications to the training setup.

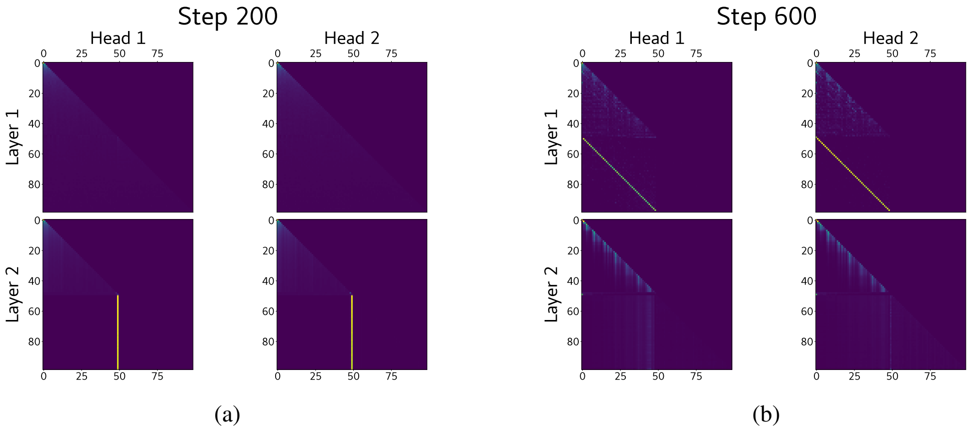


Figure 12: **Attention heads demonstrate sudden change in matrix completion using GPT.** We find that even in the GPT case, the attention heads change from trivial (Layer 2, attending to the [SEP] token for all positions in the output) to those in Layer 1 attending to the corresponding positions in the input (∼identity maps). This corroborates with our finding about the model learning to copy observed entries after the sudden drop, in Fig. 11.

H Effect of Model Size

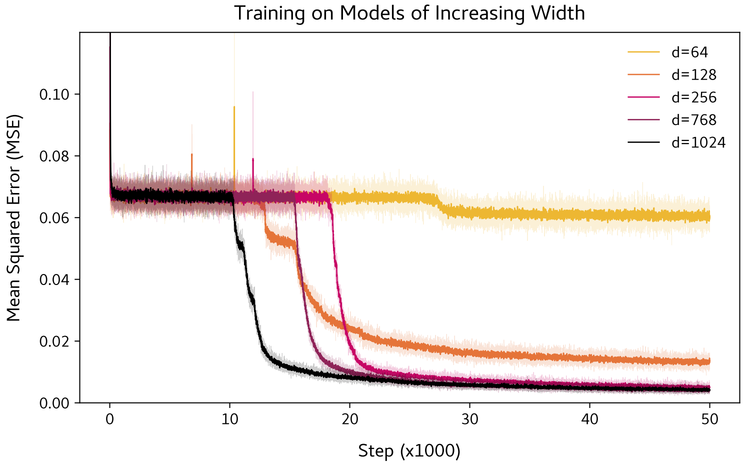


Figure 13: **Effect of model width.** Training with different model widths (hidden state dimensionality) on 7×7 rank-2 inputs. The plot demonstrate that $d = 64$ is too small to obtain accurate matrix completion, and that the performance is sub–optimal for $d = 128$. We scale the hidden layer width of the $2-$layer MLPs as $4d$, as is done in practice.

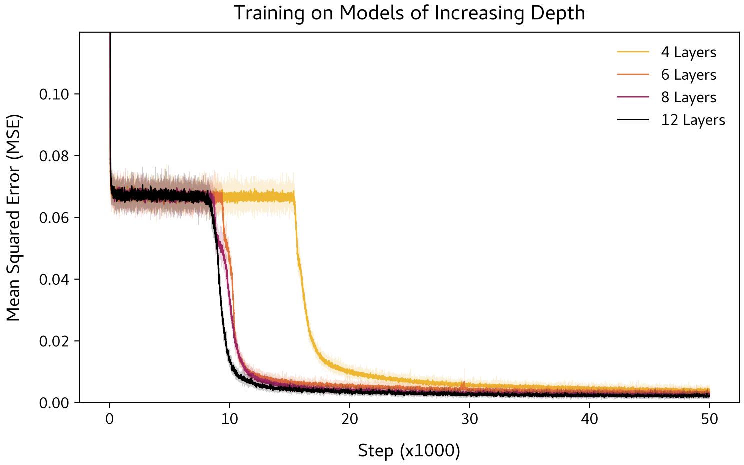


Figure 14: **Effect of model depth**. Training with different model depths (number of layers) on 7×7 rank-2 inputs. The plot demonstrate that as depth increases from 4 to 6, the sudden drop occurs earlier, but increasing depth beyond this $(8, 12)$ has little effect. The final MSE obtained also follows the intuitive ordering (largest for $L = 4$ decreasing with L upto $L = 12$; though the variation is not significant.

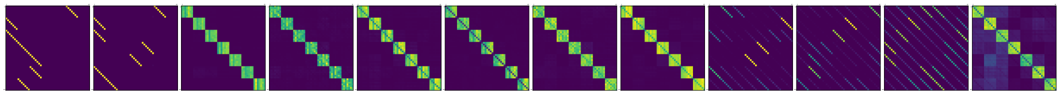


Figure 15: **Using 1 attention head per layer.** We find that training a $12-$layer, $1-$head BERT model on matrix completion leads to similar loss (4e$-$3) and attention heads as the $4-$layer, $8-$head model.

I Effect of Matrix Size and Rank on Training

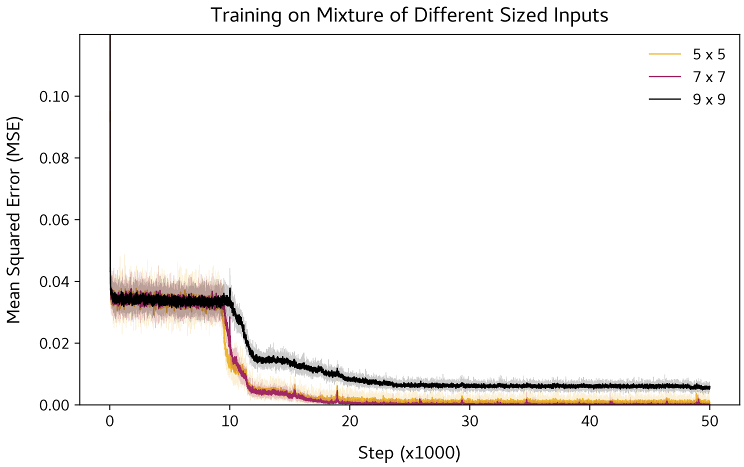


Figure 16: **Learning order when trained on mixture of matrices.** Training on uniform mixture of $5 \times 5, 7 \times 7$ and 9×9 rank-1 matrices i.e., at each step, 256 samples of size $n \times n$, with n chosen randomly from $\{5, 7, 9\}$. The plots show the test set MSE on separate 256 samples of the 3 different matrix sizes.

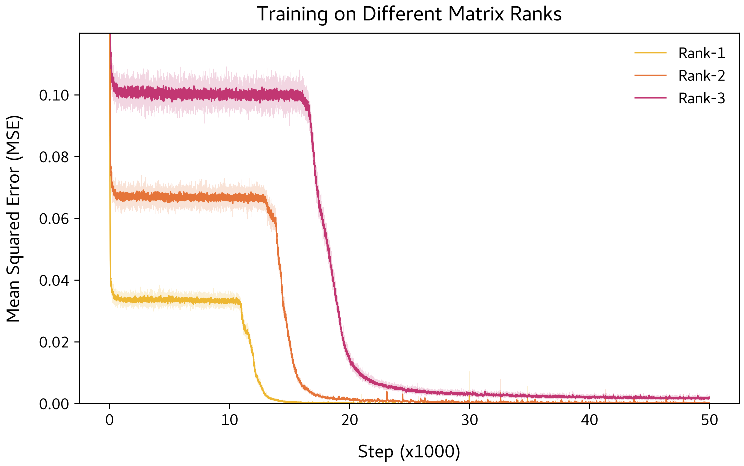


Figure 17: **Effect of problem complexity**. Training a $12-$layer, $12-$head model on 10×10 matrices of rank $r = 1, 2, 3$. There is a clear progression in terms of the training step where sudden drop occurs, and the final loss values scale roughly as $L \sim c \cdot 10^r$, $c \approx 2 \times 10^{-6}$. This also predicts that $L \sim 0.02$ for $r = 4$, i.e. the model does not solve matrix completion to low MSE, which is what we also observe in practice.

J Effect of Input Distribution

J.1 Training

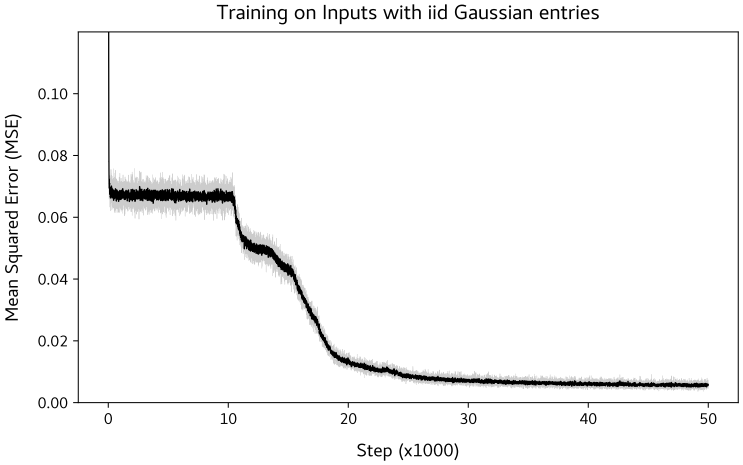
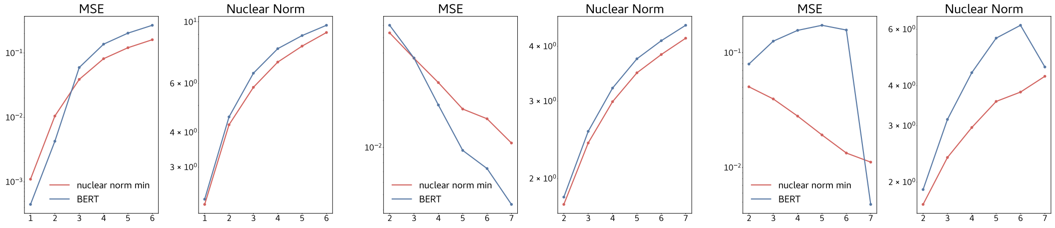


Figure 18: **Sudden drop is not limited to uniform distribution**. Training on i.i.d. $\mathcal{N}(0, 1)$ entries. We find that the sudden drop also occurs in this case, and the final loss value $\sim 5.6 \times 10^{-3}$, similar to the value obtained for i.i.d. Uniform$[-1, 1]$ entries.

J.2 Inference



(a) Rank

(b) No. of Rows

(c) No. of Columns

Figure 19: **Model performs similar to nuclear norm minimization on OOD samples.** OOD performance at inference for various values of rank, number of rows and columns of the input matrix. Except (c), the OOD performance of the model is close to the nuclear norm minimization solution for the same inputs. For (c), since we observed that positional embeddings depend on the column of the element, changing the number of columns adversely affects performance.

Matrix Distribution We also change the input distribution of the matrix entries to Gaussian and Laplace, and measure average MSE over 1024 samples of size 7×7 and rank-2, to evaluate the OOD performance of the trained model. We find that

- for entries i.i.d. $\sim \mathcal{N}(0, 0.25)$, $L \approx 4 \times 10^{-3}$, and
- for entries i.i.d. $\sim \text{Laplace}(0, 0.25)$, (parameterized by mean and scale), $L \approx 2 \times 10^{-3}$.

That is, the OOD performance on these distributions is similar to the MSE obtained for the in–training distribution (Uniform$[-1, 1]$).

K Probing: Additional Results

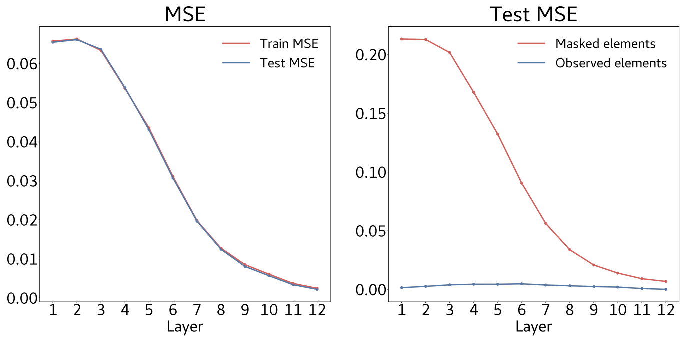


Figure 20: **Hidden states encode the true value at missing entries.** Probing for the corresponding element in the fully observed matrix X. (Left) comparing the train and test MSE of the linear probe, to confirm that the probe does not overfit. (Right) Test MSE evaluated on the masked elements, that shows an interesting variation: the MSE goes down at a nonlinear rate, hinting that implicit layer–wise optimization could be taking place.

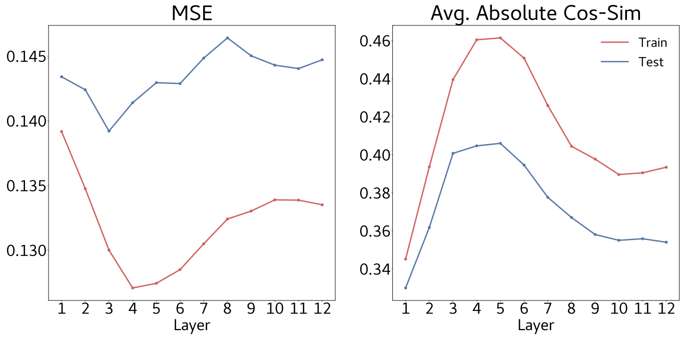


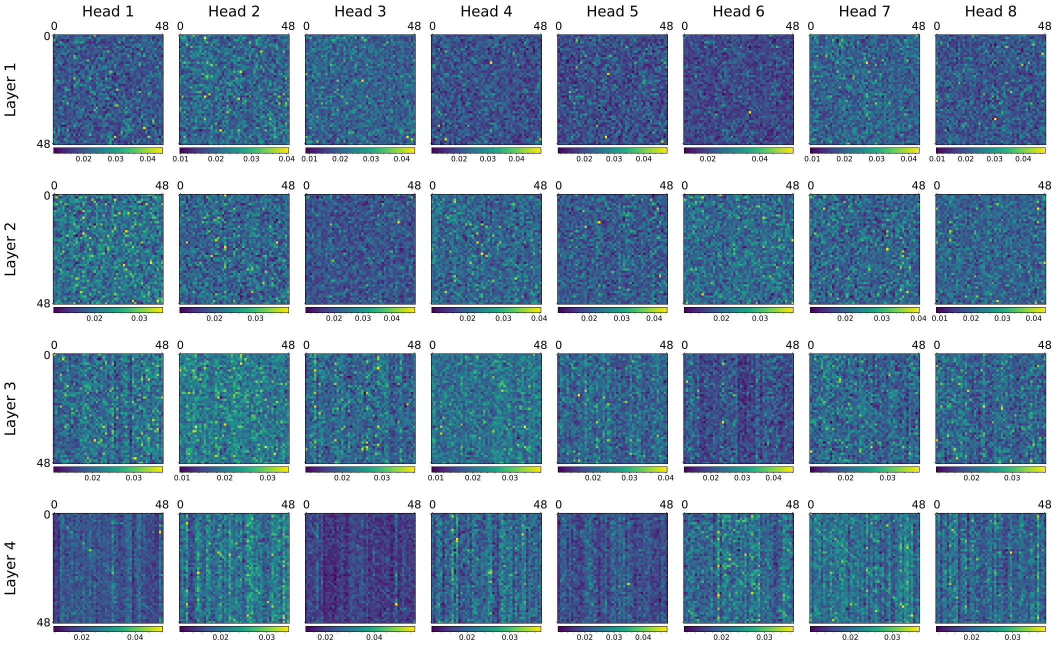
Figure 21: **Model does not encode singular vectors in hidden states**. Probing for the first left singular vector at all positions; that is, for fully observed input matrix $X \in \mathbb{R}^{n \times n}$ with SVD $X = U\Sigma V^\top$, we probe for $u = [U_{11}, U_{21}, \ldots, U_{n1}]^\top \in \mathbb{R}^n$. (Left) Train and Test MSE across different layers of the model; (Right) Average Absolute Cosine similarity of the predicted u with the actual u. Both evaluations are inconclusive, since the MSE is too large, and the cosine similarity is not much larger than the average absolute cosine similarity (≈ 0.3) between 2 i.i.d. vectors $\sim \mathcal{N}(0, I_{7\times 7}) \in \mathbb{R}^7$

L Ablating Groups of Structured Attention Heads

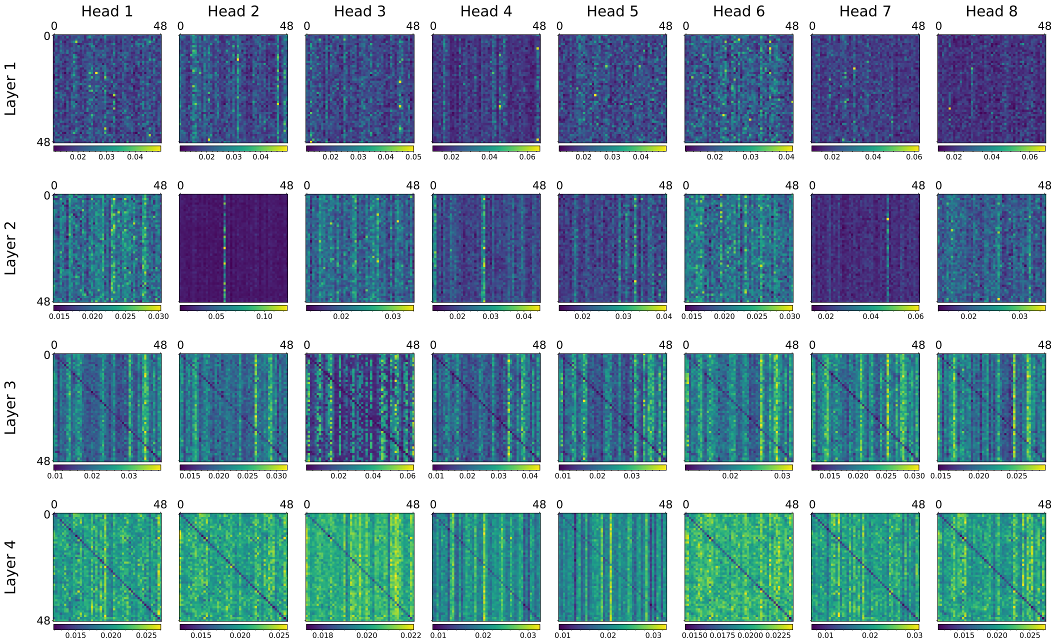
Group of Attention Heads Ablated	L		L_{obs}		L_{mask}		Ratio of L (w/ to w/o)
	w/ ablation	w/o ablation	w/ ablation	w/o ablation	w/ ablation	w/o ablation	
(2,1), (3,4), (4,8)	0.0073	0.0035	0.0001	9e−5	0.0245	0.0117	2.09
(4,3), (4,4)	0.0079	0.0045	3e−4	8.8e−5	0.026	0.015	1.76
(2,2–4), (2,6), (3,2), (3,3), (3,5), (3,1), (3,6)	0.057	0.0043	2.1e−4	9e−5	0.1885	0.0142	13.26
(2,5), (2,7), (2,8)	0.0112	0.0049	8.5e−5	9.2e−5	0.037	0.016	2.29
(1,1–2), (1,5–8)	0.0314	0.0048	1.7e−4	8.8e−5	0.1038	0.0157	6.54

M Attention Heads

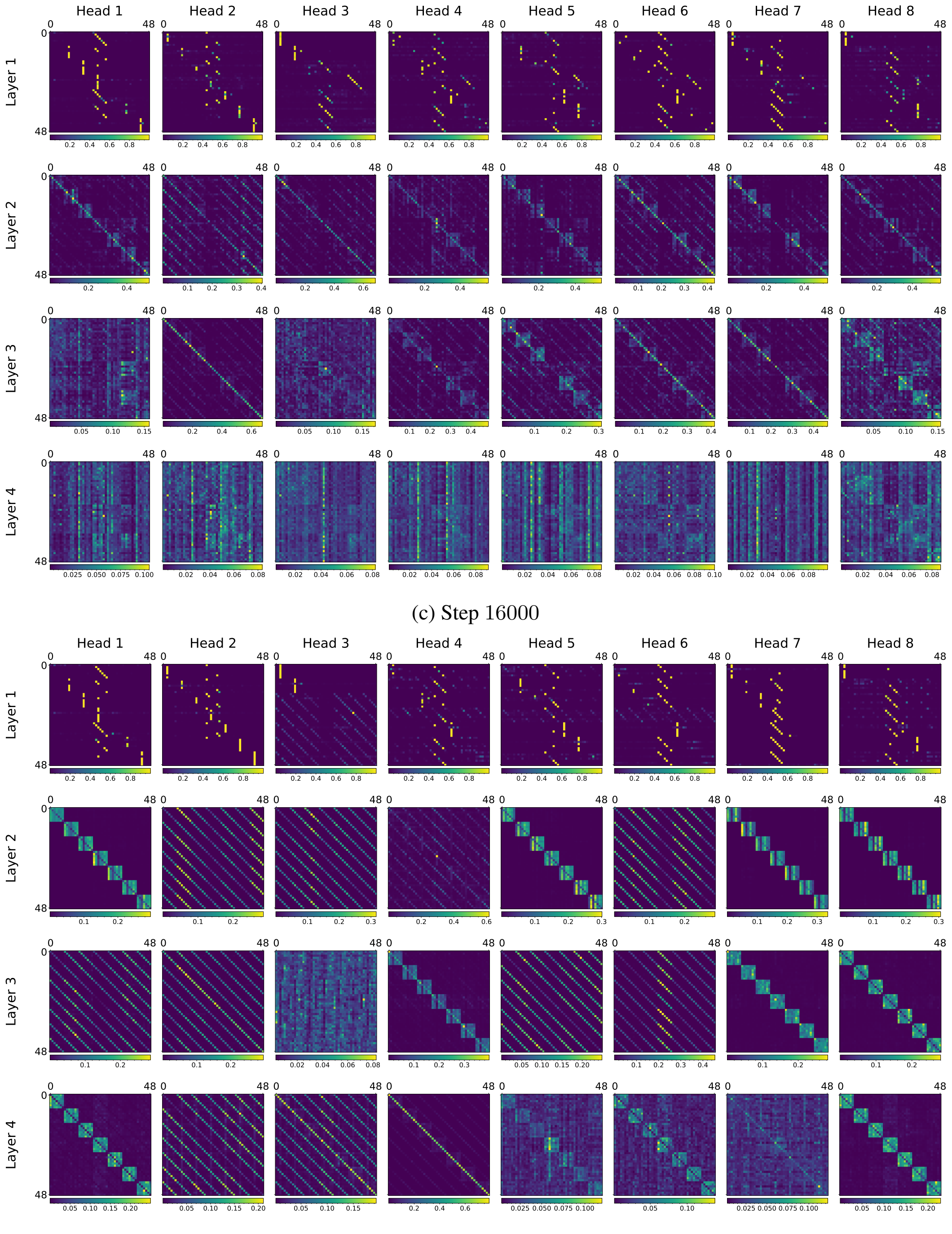
M.1 Variation along training



(a) Step 4000



(b) Step 14000

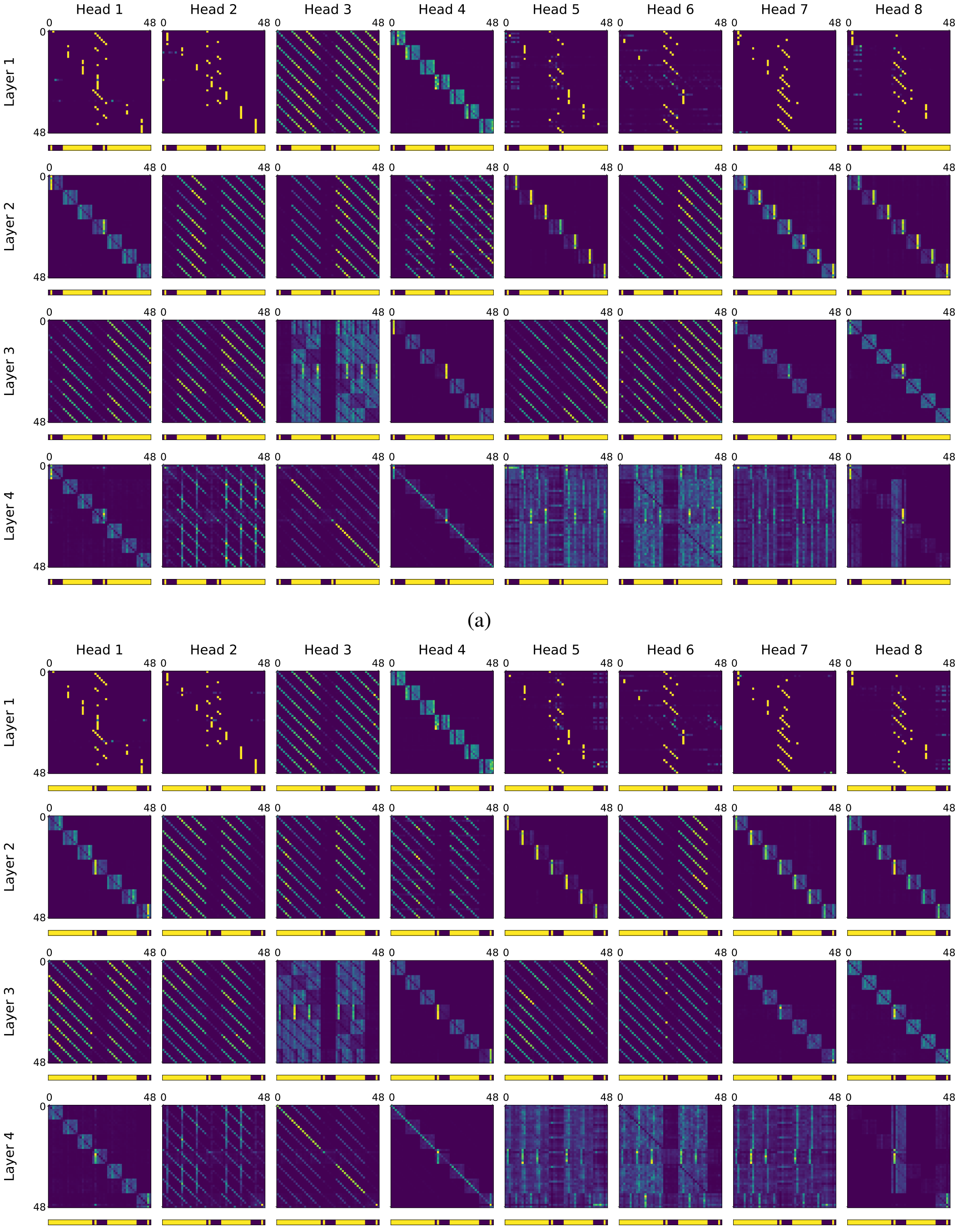


(c) Step 16000

(d) Step 20000

Figure 22: Attention heads across various training steps.

M.2 Effect of changing mask structure



(a)

(b)

Figure 23: Attention heads and corresponding masks; blue denotes masked position in the input matrix.

N Attention Heads for larger inputs

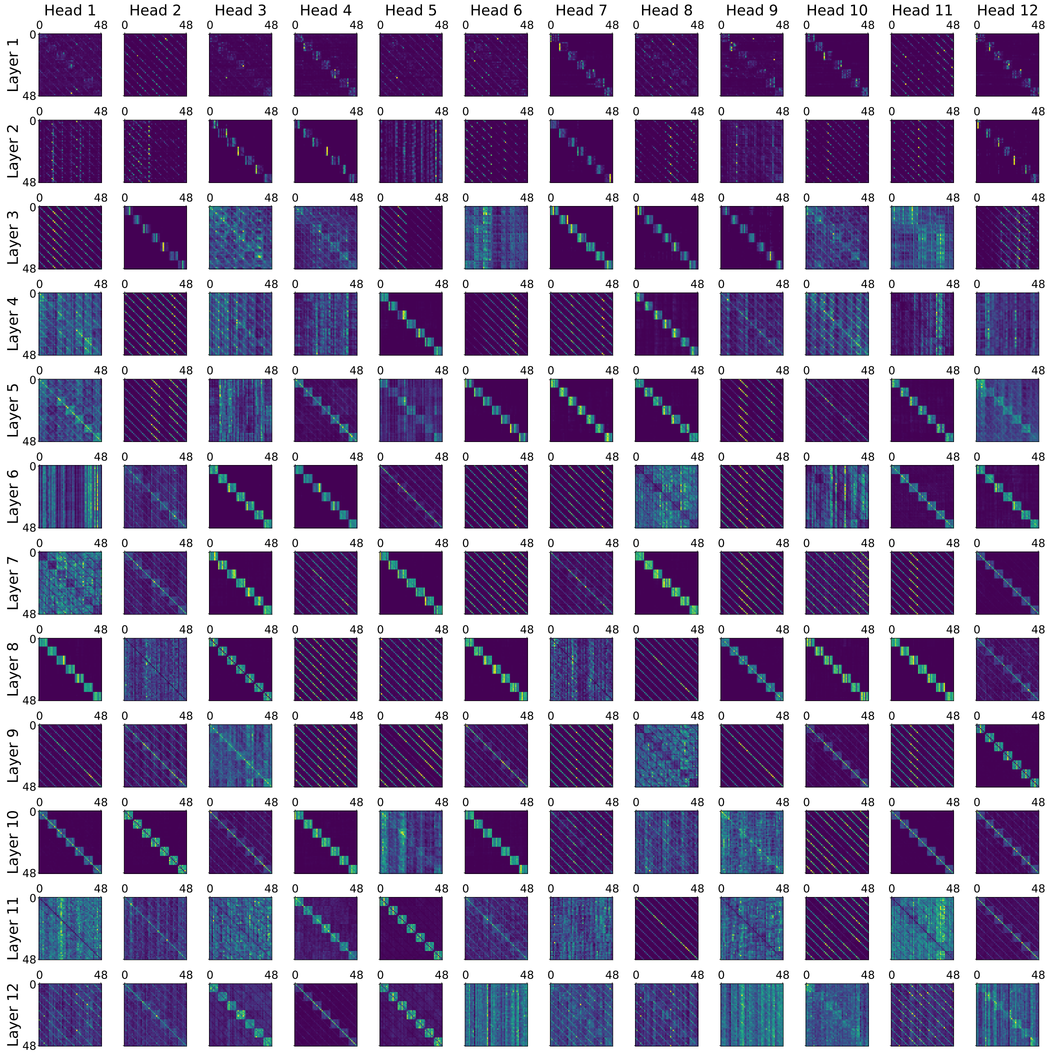


Figure 24: Attention heads in 12 layers, 12–heads model on 7 × 7 rank–2 input

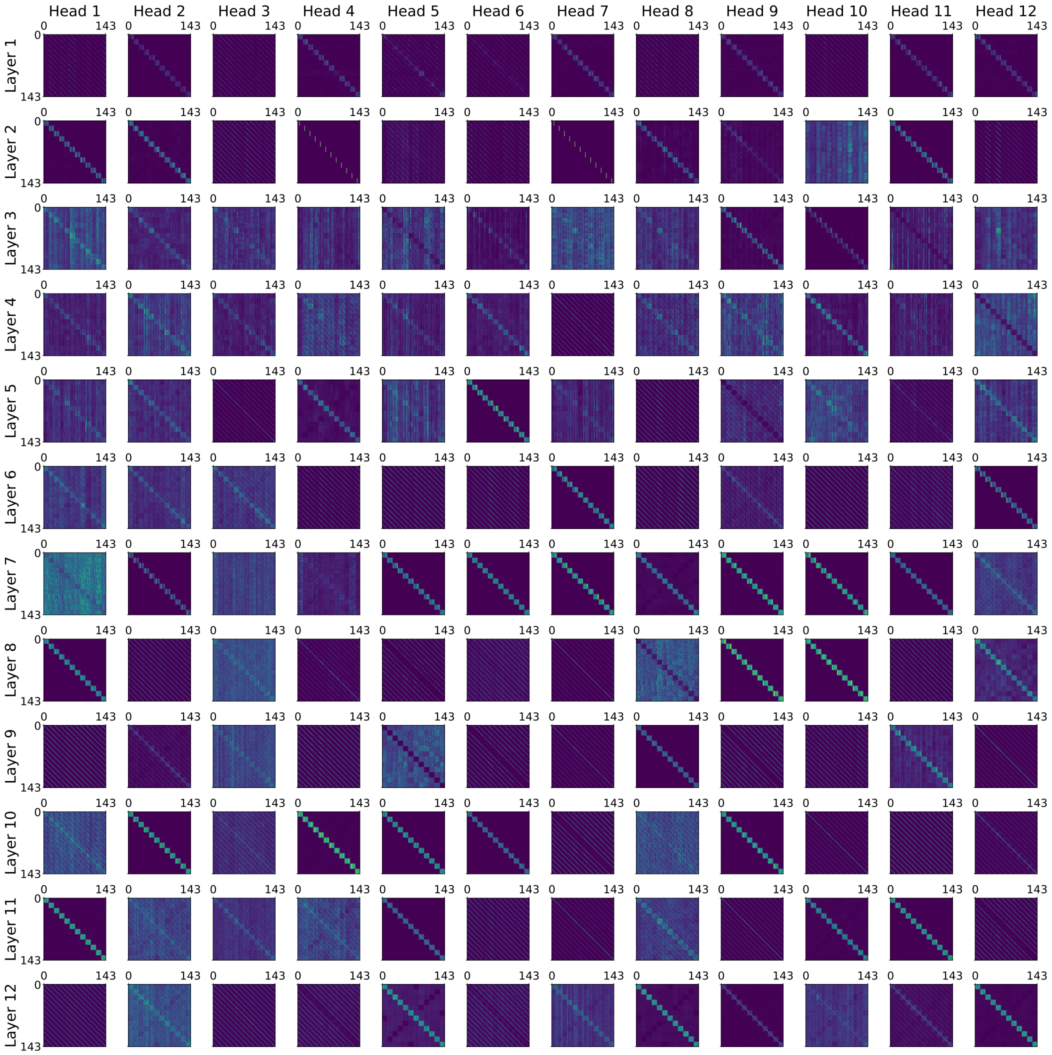


Figure 25: Attention heads in 12 layers, 12–heads model on 12×12 rank–3 input

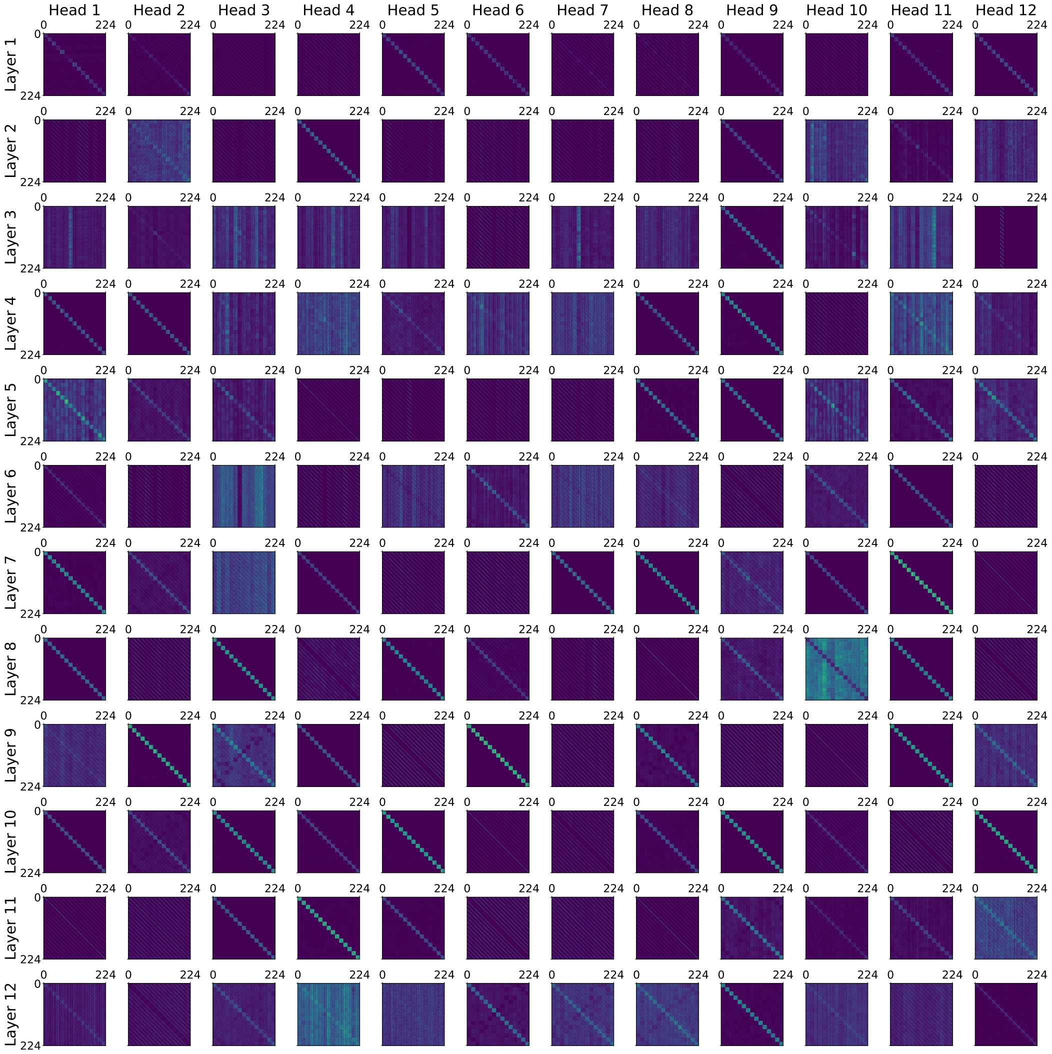


Figure 26: Attention heads in 12 layers, 12–heads model on 15×15 rank–4 input

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We have accurately listed our main claims in line with the actual contribution of this work.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have discussed limitations of our work in Section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: There are no theoretical results in this work.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We describe our training experiment details in Section 2.2, and other experiment details are provided wherever relevant. Link to the code is available in Sec. 2.2.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Link to the code is available in Sec. 2.2, and a README file is included to help run experiments.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide all such experimental details in Sec. 2.2.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Due to computational issues, we do not report error bars for our training experiments at this time.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We discuss compute resource usage in Sec. 2.2.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our work conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss societal impact of our work in the conclusion (Sec. 6).

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Risk of misuse in our case does not apply, since our work involves interpretability analyses of a BERT model on a mathematical problem.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have mainly used the BERT model implementation from Huggingface (cited in Sec. 2.2) and CVXPY (cited in Sec. 3.2). We have not used other existing code (except libraries or frameworks available publicly), data or models.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Code released for this paper (Sec. 2.2) is accompanied by a README file that details how to reproduce empirical observations.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This work does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This work does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.