

Circuit-Aware Analysis of LoRA Fine-Tuning: What Changes, Where, and Why?

Stanford CS224N Default Project

Mentor: Tristan Thrush

Nathan Maidi

Department of Computer Science
Stanford University
nm542@stanford.edu

Abstract

LoRA reduces the cost of fine-tuning large language models, but standard practice still treats adapter placement as a heuristic. I study whether mechanistic localization can make that choice more principled. Starting from a GPT-2 implementation from the CS224N default project, I fine-tune on cloze-style paraphrase detection and Shakespearean sonnet generation, then compare LoRA placements under matched parameter budgets: circuit-guided, last-layer, random, and non-circuit controls. To localize task-relevant computation, I use activation patching to rank layers and heads by causal effect on the output. I then perform post-hoc analysis comparing pretrained and post-LoRA models through layer importance, head importance, logit attribution, and adapter magnitudes. Circuit-guided placement helps on both tasks relative to a non-circuit control, but the strength of the result differs substantially by domain. On paraphrase detection, circuit-guided LoRA outperforms non-circuit placement and slightly improves on last-layer placement, though random placement remains competitive and full fine-tuning performs best overall. On sonnet generation, circuit-guided LoRA is the strongest LoRA method and remains best across multiple seeds, achieving higher chrF scores than random and non-circuit placement. Mechanistically, the largest post-LoRA changes concentrate in adapted late layers and their heads, but this overlap is partly structural because only those layers receive direct parameter updates. Overall, the results suggest that circuit-guided placement is a useful signal, especially for generation, but does not by itself close the gap to full fine-tuning or eliminate variance across tasks.

1 Introduction

Parameter-efficient fine-tuning is attractive because it avoids storing and updating a full copy of a language model for every task. LoRA is especially appealing: it freezes the pretrained model and learns low-rank updates that can be merged back into the original weights at inference time. However, the method leaves one important choice mostly unresolved: *where* should adapters be placed? In practice, this is often decided by a heuristic such as “last k layers” or “all attention layers.” That is computationally convenient, but it is not mechanistically motivated.

This project asks whether mechanistic interpretability can provide a better placement signal. The core idea is simple. Before adding LoRA, use activation patching to identify layers and heads whose activations matter most for the downstream task. Then place a fixed budget of LoRA modules at those locations and compare against matched controls. After fine-tuning, inspect what changed. Did LoRA strengthen already-important computation, redirect it, or simply exploit a generic late-layer bias?

I study this question on two tasks from the CS224N default project: cloze-style paraphrase detection on Quora question pairs and Shakespearean sonnet continuation. These tasks are useful complements. Paraphrase detection is a discriminative task with a narrow output space (“yes” or “no”), while sonnet

continuation is an open-ended generation problem where the model must preserve style over many tokens. This contrast lets me test whether circuit-guided placement is broadly useful or only helps when the task signal is concentrated.

The main findings are mixed but informative. Circuit-guided placement clearly beats a bottom-layer non-circuit control on both tasks and is the best LoRA method for sonnet generation. On paraphrase detection, however, the advantage is modest: random placement slightly outperforms circuit-guided placement in the single-seed run, and full fine-tuning remains substantially stronger. The mechanistic analysis shows that the largest post-LoRA changes concentrate in the adapted late layers and their heads. That is partly expected, not a discovery by itself, because those are the only layers whose attention projections are directly modified. The stronger evidence comes from the *pre*-fine-tuning localization and the downstream performance differences under a matched parameter budget.

2 Related Work

LoRA introduces low-rank weight updates for efficient adaptation and shows that very small trainable subspaces can match full fine-tuning on many NLP tasks. My project directly targets one limitation the LoRA paper leaves open: adapter placement is treated mostly as a design choice rather than something inferred from model internals.

On the interpretability side, my localization method is activation patching, also called causal tracing in some of the literature. The transformer circuits framework from Anthropic helped popularize this style of intervention-based analysis by replacing internal activations from one run with those from another and measuring the effect on the output. Recent reviews and primers place activation patching among the standard tools of mechanistic interpretability and discuss both its usefulness and its limitations.

My project sits between these two threads. The goal is not only to interpret a model after fine-tuning, and not only to use LoRA as a black-box PEFT method, but to ask whether mechanistic localization can guide adaptation itself. This is also related to broader work on understanding what fine-tuning changes internally, although my analysis is much narrower and task-specific.

3 Approach

3.1 Base model and LoRA parameterization

I use the small GPT-2 architecture from the CS224N starter code, with hidden size $d = 768$, number of layers $L = 12$, and number of heads $H = 12$. For a batch of token IDs with embeddings

$$X \in \mathbb{R}^{B \times T \times d}, \quad (1)$$

each transformer block produces a residual-stream output

$$H^{(\ell)} \in \mathbb{R}^{B \times T \times d}, \quad \ell \in \{1, \dots, L\}. \quad (2)$$

For one attention head with head dimension $d_h = d/H = 64$, the standard projections are

$$Q_h^{(\ell)} = X(W_{Q,h}^{(\ell)})^\top, \quad W_{Q,h}^{(\ell)} \in \mathbb{R}^{d_h \times d}, \quad (3)$$

$$K_h^{(\ell)} = X(W_{K,h}^{(\ell)})^\top, \quad W_{K,h}^{(\ell)} \in \mathbb{R}^{d_h \times d}, \quad (4)$$

$$V_h^{(\ell)} = X(W_{V,h}^{(\ell)})^\top, \quad W_{V,h}^{(\ell)} \in \mathbb{R}^{d_h \times d}. \quad (5)$$

Causal self-attention is then

$$O_h^{(\ell)} = \text{softmax}\left(\frac{Q_h^{(\ell)}(K_h^{(\ell)})^\top}{\sqrt{d_h}} + M\right)V_h^{(\ell)}, \quad (6)$$

where $O_h^{(\ell)} \in \mathbb{R}^{B \times T \times d_h}$ and $M \in \mathbb{R}^{T \times T}$ is the causal mask.

LoRA is applied only to the query and value projections. If a frozen projection has weight $W \in \mathbb{R}^{d_{out} \times d_{in}}$, LoRA replaces it with

$$W' = W + \frac{\alpha}{r}BA, \quad (7)$$

where $A \in \mathbb{R}^{r \times d_{in}}$, $B \in \mathbb{R}^{d_{out} \times r}$, and $r \ll d_{in}, d_{out}$. For an input tensor $X \in \mathbb{R}^{B \times T \times d_{in}}$, the adapted linear map is

$$\text{LoRA}(X) = XW^\top + \frac{\alpha}{r}XA^\top B^\top, \quad (8)$$

with the same output shape $\mathbb{R}^{B \times T \times d_{out}}$. In code, I freeze all pretrained weights and train only the LoRA matrices A, B (plus the paraphrase classification head for the discriminative task).

3.2 Clean and corrupt inputs

Activation patching requires two forward passes: a *clean* run and a *corrupt* run. In my implementation, clean means the original tokenized example

$$x = (\text{ids}, \text{mask}), \quad \text{ids} \in \{0, \dots, V-1\}^{1 \times T}, \text{ mask} \in \{0, 1\}^{1 \times T}. \quad (9)$$

For corrupt inputs, every non-pad token ID is replaced by a random vocabulary ID while preserving the attention mask:

$$\tilde{\text{ids}}_t \sim \text{Unif}(\{0, \dots, V-1\}) \text{ for all } t \text{ with } \text{mask}_t = 1, \quad (10)$$

and $\tilde{x} = (\tilde{\text{ids}}, \text{mask})$ keeps the same sequence length and valid positions as the clean example. This produces a corrupted sequence with matching shape but destroyed lexical content. The patching question is: if I replace one internal activation in the corrupt run with the corresponding activation from the clean run, how much useful behavior returns?

3.3 Layer-level activation patching

For each example, I cache the clean residual-stream activations

$$\mathcal{C}^{(\ell)} = H_{\text{clean}}^{(\ell)} \in \mathbb{R}^{1 \times T \times d}, \quad \ell = 1, \dots, L. \quad (11)$$

Then I run the corrupt input and patch one layer at a time. For paraphrase detection, only the last non-pad position matters because classification uses the final token representation. Let p_{last} denote that position. The patched residual stream at layer ℓ is

$$\hat{H}^{(\ell)}[:, p_{\text{last}}, :] \leftarrow \mathcal{C}^{(\ell)}[:, p_{\text{last}}, :], \quad (12)$$

while all other positions remain as in the corrupt run. Let $z_{\text{yes}}, z_{\text{no}} \in \mathbb{R}$ denote the output logits for the target tokens “yes” and “no.” The paraphrase layer importance score is

$$I_\ell^{\text{para}} = |(z_{\text{yes}}^{\text{patch}, \ell} - z_{\text{no}}^{\text{patch}, \ell}) - (z_{\text{yes}}^{\text{corr}} - z_{\text{no}}^{\text{corr}})|. \quad (13)$$

A large value means that restoring layer ℓ from the clean run substantially repairs the yes-vs-no decision.

For sonnet generation, the output is distributed across many positions, so patching only the final position would be too narrow. I therefore use a proxy based on the last $k = 4$ prediction positions before the end of the input continuation window. Let P be those positions, and let y_{p+1} be the clean next token after position p . The sonnet layer importance score is

$$I_\ell^{\text{son}} = \frac{1}{|P|} \sum_{p \in P} |\log p^{\text{patch}, \ell}(y_{p+1} | x_{\leq p}) - \log p^{\text{corr}}(y_{p+1} | x_{\leq p})|. \quad (14)$$

This ranks layers by how much restoring them improves the model’s probability on the clean continuation tokens.

I average these scores over multiple examples and choose the top $k = 4$ layers as the *circuit-guided* LoRA placement.

3.4 Head-level proxy and logit attribution

Exact head patching is more expensive, so for post-hoc analysis I use a cached proxy. Let the per-head attention outputs at layer ℓ be

$$O_h^{(\ell)} \in \mathbb{R}^{1 \times T \times d_h}, \quad h \in \{1, \dots, H\}. \quad (15)$$

If $W_O^{(\ell)} \in \mathbb{R}^{d \times d}$ is the output projection and $W_{O,h}^{(\ell)} \in \mathbb{R}^{d \times d_h}$ is the slice corresponding to head h , then the projected clean-corrupt head difference at position p is

$$\Delta_{\ell,h}(p) = \left(O_{h,\text{clean}}^{(\ell)}(p) - O_{h,\text{corr}}^{(\ell)}(p) \right) \left(W_{O,h}^{(\ell)} \right)^\top \in \mathbb{R}^d. \quad (16)$$

For paraphrase detection, I score this direction against the unembedding difference for the yes/no decision,

$$u_{\text{yn}} = e_{\text{yes}} - e_{\text{no}} \in \mathbb{R}^d, \quad (17)$$

where e_t is the token embedding used as the unembedding vector for token t . The head score is then

$$J_{\ell,h}^{\text{para}} = |\langle \Delta_{\ell,h}(p_{\text{last}}), u_{\text{yn}} \rangle|. \quad (18)$$

For sonnets, I use the target token embedding at the patched position instead of u_{yn} and average over the same set P of prediction positions.

For paraphrase detection, I also compute per-layer logit attribution for the “yes” token. If $H^{(0)}$ denotes the embedding output and $u_{\text{yes}} = e_{\text{yes}} \in \mathbb{R}^d$, the attribution of layer ℓ at the last position is

$$A_\ell = \left\langle H^{(\ell)}[p_{\text{last}}] - H^{(\ell-1)}[p_{\text{last}}], u_{\text{yes}} \right\rangle. \quad (19)$$

This is not a full mechanistic decomposition, but it is a useful diagnostic of which layers move the yes-token logit the most.

3.5 Post-hoc analysis

After training, I compare pretrained and post-LoRA models using the same cached clean/corrupt examples. For each layer, I report (i) pre- and post-LoRA importance scores, (ii) average residual means, (iii) head-level pre/post importance, and (iv) LoRA magnitudes $\|BA\|_F$ for each adapted projection. Importantly, the post-hoc analysis uses a fast proxy based on cached activations rather than rerunning exact patching for every layer and head.

4 Experiments

4.1 Data and evaluation

For paraphrase detection, I use the Quora question-pair data provided in the starter code, formatted as a cloze-style task that predicts either “yes” (token 8505) or “no” (token 3919). The evaluation metric is dev accuracy. In the training pipeline, we produced 283,003 formatted train prompts and 40,429 dev prompts.

For sonnet generation, I use the Shakespeare sonnet dataset from the default project: 143 training sonnets and 12 held-out sonnets. The model is conditioned on the first three lines and generates the remainder autoregressively. I evaluate with chrF on the held-out continuations.

4.2 Experimental details

All experiments use GPT-2 small ($d = 768$, $L = 12$, $H = 12$). For paraphrase detection, I train with batch size 128. Full fine-tuning uses learning rate 10^{-5} . LoRA uses rank $r = 4$, scale $\alpha = 8$, dropout 0.05, and learning rate 3×10^{-4} . For sonnet generation, I train with batch size 8. Full fine-tuning again uses learning rate 10^{-5} , while LoRA uses rank $r = 16$, scale $\alpha = 32$, dropout 0.05, and learning rate 3×10^{-4} . In all LoRA experiments, adapters are inserted into the query and value projections only.

For paraphrase detection, I compare full fine-tuning, circuit-guided LoRA, non-circuit LoRA (bottom- k layers by patching score), last-layer LoRA, and random-layer LoRA. Due to compute constraints, the paraphrase experiments are reported for a single seed. For sonnets, I compare full fine-tuning, circuit-guided LoRA, non-circuit LoRA, and random-layer LoRA. I reran the sonnet LoRA comparisons with three seeds {11711, 42, 1234} and report mean and standard deviation for those methods; full fine-tuning was only run once as an upper bound.

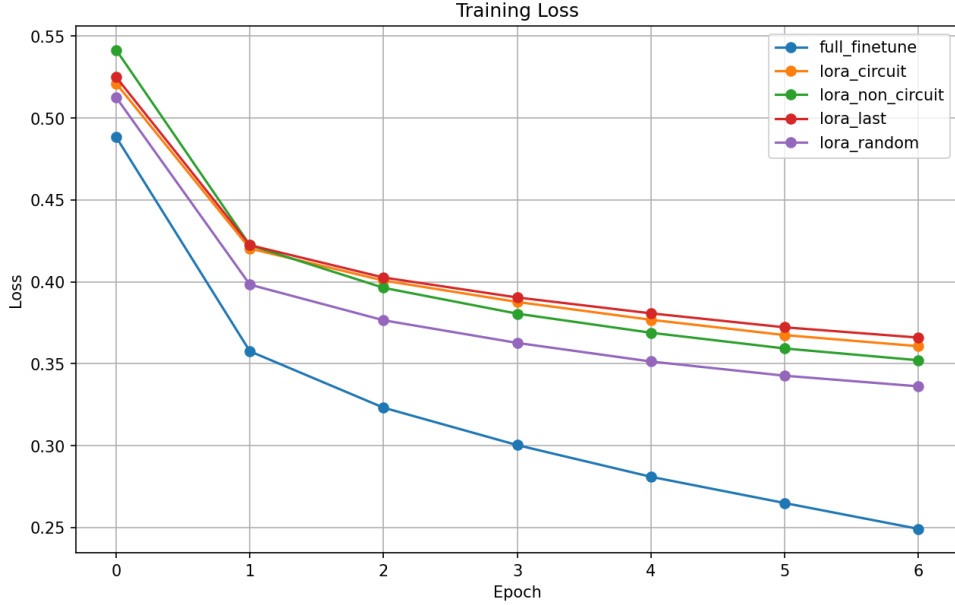


Figure 1: Paraphrase detection training loss over epochs. These are single-seed curves. Full fine-tuning converges to the lowest loss, while all LoRA variants remain competitive under a much smaller trainable budget.

Table 1: Paraphrase detection results. All values are from the single-seed run used in the main analysis.

Method	Dev accuracy	Trainable params	LoRA layers
Full fine-tune	0.8751	125,128,704	–
LoRA circuit	0.8476	98,304	[6, 9, 10, 11]
LoRA non-circuit	0.8429	98,304	[0, 1, 2, 3]
LoRA last	0.8456	98,304	[8, 9, 10, 11]
LoRA random	0.8533	98,304	[3, 5, 8, 10]

4.3 Results

The final Gradescope test submission achieved 0.811 paraphrase test accuracy and a Sonnet chrF score of 41.530. The test submissions use the circuit-based approach.

Table 1 shows the paraphrase results. Full fine-tuning is best at 0.8751 dev accuracy. Among LoRA methods, random placement is numerically best at 0.8533, followed by circuit-guided placement at 0.8476. Circuit-guided placement still improves on both the non-circuit control (0.8429) and the standard last-layer heuristic (0.8456), which suggests that the patching signal is informative but not decisive in this task.

Table 2 shows the sonnet results. Full fine-tuning reaches 40.22 chrF in the single available run, but among the LoRA methods, circuit-guided placement is clearly strongest. Averaged across three seeds, it reaches 41.35 ± 0.19 chrF, beating random placement by about 0.87 chrF and the non-circuit control by about 2.28 chrF. This is the clearest evidence in the paper that circuit-guided placement can help under a fixed LoRA budget.

5 Analysis

5.1 What the patching scores select

Before fine-tuning, activation patching ranks late layers as most important on both tasks. For sonnet generation, the top four layers are consistently [8, 9, 10, 11] across all three seeds. For paraphrase

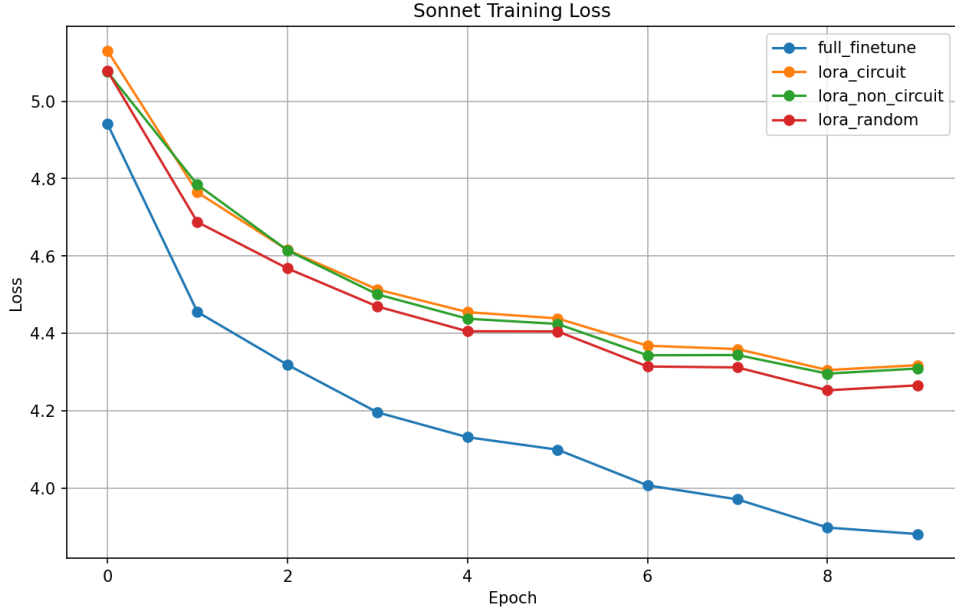


Figure 2: Sonnet generation training loss. Full fine-tuning is shown for one seed; LoRA curves show mean with standard-deviation bands over three seeds. Circuit-guided and random placements train similarly, while the non-circuit baseline remains worse throughout.

Table 2: Sonnet generation results. Full fine-tuning is reported from one seed; LoRA methods are mean $\pm$ standard deviation over three seeds.

Method	Dev chrF	Trainable params	LoRA layers
Full fine-tune (1 seed)	40.22	125,227,008	—
LoRA circuit (3 seeds)	41.35 $\pm$ 0.19	196,608	[8, 9, 10, 11]
LoRA non-circuit (3 seeds)	39.39 $\pm$ 0.36	196,608	[0, 1, 2, 3]
LoRA random (3 seeds)	39.75 $\pm$ 0.51	196,608	11711: [3, 5, 8, 10];
			42: [0, 1, 4, 10];
			1234: [0, 1, 7, 10]

detection, the top four layers are [6, 9, 10, 11] in the reported run. This is plausible because both tasks consume GPT-2 as a pretrained decoder and read out a final-token or late-token decision, so late residual-stream states are natural bottlenecks.

Still, the sonnet result requires a careful interpretation. The fact that the circuit-guided set equals the last four layers is not something I hand-designed; it emerged from the pre-LoRA activation patching scores. But once I then place LoRA on [8, 9, 10, 11], the post-hoc finding that the most changed layers are also [8, 9, 10, 11] is partly mechanical. Earlier layers cannot change in direct parameter space because no LoRA weights were inserted there. So the overlap between “adapted layers” and “most changed layers” should not be over-claimed as independent evidence. The more meaningful result is that the *pretraining* localization already favored those layers, and that this placement outperformed matched alternatives downstream.

5.2 Post-hoc layer and head changes

For sonnets, the post-hoc proxy identifies the same late layers as the largest-change region, with 4/4 overlap between the four most-changed layers and the adapted layers in every seed. Head-level changes are even more concentrated: the largest pre/post differences almost all appear in layer 11, especially heads 0, 8, 10, and 11. Appendix Table 4 shows a representative seed. This concentration is consistent across the three seeds.

For paraphrase detection, the post-hoc picture is sharper and more extreme. The largest head changes are dominated by layer-11 heads, and the yes-token logit attribution changes most strongly in late LoRA layers, especially layers 9 and 11. Appendix Table 3 gives the top changed heads. These results suggest that the cloze decision is heavily mediated by late-layer features near the final readout.

5.3 Why sonnet benefits more from circuit-guided placement

The most interesting asymmetry is that circuit-guided placement is clearly helpful for sonnet generation, but only mildly helpful for paraphrase detection. One interpretation is that the sonnet task benefits from reinforcing a specific late-layer stylistic pathway that activation patching can identify reasonably well. In contrast, paraphrase detection may depend on a broader collection of features than my top- k layer selection captures. Another possibility is methodological: the paraphrase result is from one seed only, while the sonnet result was rechecked across three seeds and remained stable.

The loss curves support this interpretation. In Figure 2, the non-circuit sonnet baseline is consistently worse throughout training, while circuit-guided and random placements train similarly but finish with a clearer chrF gap. In Figure 1, the paraphrase LoRA methods stay tightly clustered, which matches the smaller accuracy differences.

5.4 Limitations

This project has several important limitations. First, the post-hoc proxy is cheaper than exact layer/head patching after fine-tuning, but it is still only a proxy. Second, the corruption method is intentionally simple (random token replacement with a preserved attention mask) and may produce a different localization profile than more semantically controlled corruptions. Third, the paraphrase conclusions are limited by compute: I did not rerun the full paraphrase suite across multiple seeds, so the relative ordering of circuit-guided and random placement on that task should be treated cautiously. Fourth, the strongest sonnet post-hoc overlap results are partly constrained by the fact that only the selected LoRA layers can undergo direct weight changes. A cleaner causal claim would require interventions that adapt one set of layers but test importance shifts everywhere with exact patching, ideally across more seeds and perhaps with head-level placement as well.

6 Conclusion

This project tested whether mechanistic localization can guide LoRA placement in GPT-2. The answer is “sometimes, and more clearly for generation than for classification.” Circuit-guided placement consistently beats a non-circuit control and is the strongest LoRA method for sonnet generation, where it also remains stable across three seeds. On paraphrase detection, the benefit is smaller: circuit-guided placement is competitive but does not beat random placement in the reported single-seed run, and full fine-tuning remains best overall.

Mechanistically, both tasks rely heavily on late-layer computation. Activation patching localizes this dependence before fine-tuning, and post-hoc analysis shows that the largest changes remain concentrated in the adapted late layers and their heads. That said, these post-hoc overlaps are partly structural and should not be over-interpreted. The main empirical contribution is therefore not “LoRA changes the layers it was inserted into,” which is obvious, but rather that pre-fine-tuning circuit localization can sometimes identify a better LoRA placement than standard heuristics under the same parameter budget.

The next steps are straightforward. The first is to rerun paraphrase detection across multiple seeds so the main comparison is statistically credible. The second is to move from layer-level to head-level or mixed layer-head placement, which may matter especially for narrow cloze decisions. The third is to replace the fast post-hoc proxy with exact patching on a smaller but carefully chosen evaluation set. Together, these would make the mechanistic story stronger and better separate genuine circuit discovery from the trivial fact that direct parameter updates happen where adapters are inserted.

References

- [1] Wu, Zhengxuan. Representation Finetuning for Language Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

- [2] Yin. LOFIT: Localized Fine-Tuning on LLM Representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.
- [3] Hu, Edward J., Shen, Yelong, Wallis, Phillip, Allen-Zhu, Zeyuan, Li, Yuanzhi, Wang, Shean, Wang, Lu, and Chen, Weizhu. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations (ICLR)*, 2022.
- [4] Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N, Kaiser, Lukasz, and Polosukhin, Illia. Attention is All You Need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [5] Elhage, Nelson et al. A Mathematical Framework for Transformer Circuits. *Transformer Circuits Thread*, 2021. <https://transformer-circuits.pub/2021/framework/index.html>
- [6] Ferrando et al. A Primer on the Inner Workings of Transformer-based Language Models. *arXiv preprint arXiv:2405.00208*, 2024. <https://arxiv.org/abs/2405.00208>
- [7] Sharkey, Lee et al. Open Problems in Mechanistic Interpretability. *arXiv preprint arXiv:2501.16496*, 2025. <https://arxiv.org/abs/2501.16496>
- [8] Bereska, L. et al. Mechanistic Interpretability for AI Safety – A Review. *arXiv preprint arXiv:2404.14082*, 2024. <https://arxiv.org/abs/2404.14082>
- [9] Rai, D. et al. A Practical Review of Mechanistic Interpretability for Transformer-Based Language Models. *arXiv preprint arXiv:2407.02646*, 2024. <https://arxiv.org/abs/2407.02646>

A Representative head-level changes

Tables 3 and 4 show representative top changed heads before and after LoRA. The sonnet table uses seed 11711; qualitatively similar late-head concentration appeared in the other two seeds as well.

Table 3: Top head-importance changes for paraphrase detection (single seed).

Layer	Head	Pre	Post	Δ	LoRA layer?
11	0	8.2152	89.1551	+80.9398	yes
11	8	12.7014	73.9135	+61.2121	yes
11	11	6.7925	65.6223	+58.8299	yes
11	4	15.6101	36.9206	+21.3105	yes
11	7	2.1940	18.8561	+16.6621	yes
11	10	2.3817	16.9294	+14.5478	yes
11	1	2.6380	10.4174	+7.7794	yes
11	5	2.2733	8.6097	+6.3364	yes
11	9	2.1243	7.8746	+5.7503	yes
9	7	0.8742	5.0367	+4.1625	yes

Table 4: Top head-importance changes for sonnet generation, representative seed 11711.

Layer	Head	Pre	Post	Δ	LoRA layer?
11	0	21.4694	29.3776	+7.9082	yes
11	4	12.7453	8.3132	-4.4321	yes
11	10	2.7152	6.7557	+4.0405	yes
11	11	18.1836	15.0306	-3.1530	yes
11	5	4.9411	2.2723	-2.6688	yes
10	10	5.3279	7.2498	+1.9219	yes
9	3	2.8394	4.1297	+1.2903	yes
11	1	5.9723	7.1637	+1.1914	yes
11	3	3.8975	4.7854	+0.8879	yes
11	2	1.3956	0.5417	-0.8540	yes