Sparse Model Diffing via Dynamic Circuits

Yueyan Li[1], Wenhao Gao[1], Caixia Yuan[1]*, Xiaojie Wang[1]

[1]School of Artificial Intelligence, Beijing University of Posts and Telecommunications
{siriuslala, whgao, yuancx, xjwang}@bupt.edu.cn

Abstract

Model diffing studies the internal changes of a model from training to reveal learning mechanisms. To enhance interpretability, existing studies often strive for structural sparsity by removing redundant model components. However, we emphasize that for most fine-tuning methods, the fundamental source of internal change, i.e., the parameter updates, is non-sparse. The dense update mechanism contradicts the structural sparsity sought by researchers. Thus, we seek to bring sparsity to fine-tuning via an interpretability-driven method called *circuit-tuning*. It employs a heuristic prune-then-tune paradigm, forcing the model to optimize within a restricted subset of parameters. To test our method, we designed a novel task called Subject-Verb Disagreement (SVD) and conducted comprehensive explorations on it. Experimental results demonstrate that, compared to other fine-tuning methods, circuit-tuning achieves better sparsity in parameter updates, thus making it possible for faithful model diffing. Based on this, we performed stage-wise model diffing, revealing the evolution of model capabilities during fine-tuning from a new view. Furthermore, experiments on gender debiasing and larger models verify the practical utility of our approach. Our work provides a new methodology for model diffing-based interpretability and its applications.[1].

1 Introduction

Building upon the general capabilities of pretrained large language models (LLMs), supervised fine-tuning (SFT) has become a universal technique for enhancing performance on specific downstream tasks. Standard fine-tuning methods, such as full fine-tuning and parameter-efficient fine-tuning (PEFT) represented by LoRA (Hu et al., 2022), focus on performance gains while overlooking the underlying mechanisms. These methods treat the model as an uninterpretable "black box". However, alongside these performance gains, researchers seek to understand the internal changes that occur within the model during the fine-tuning process.

A line of research on fine-tuning mechanisms focuses on mechanistic interpretability (Olah et al., 2020; Elhage et al., 2021; Sharkey et al., 2025), which aims to reverse-engineer a model to explain its behaviors. Within this field, model diffing adopts a dynamic view to investigate changes in model internals during training. Some studies explore changes in task-related conceptual representations by examining feature distributions or task vectors (Bricken et al., 2024; Jack et al., 2024; Lee et al., 2024). Others utilize causal intervention techniques to identify and compare task-related circuits before and after fine-tuning (Prakash et al., 2024; Jain et al., 2023; Wang et al., 2025b). These model diffing studies assume that fine-tuning involves more targeted changes than pretraining and thus is more tractable to understand than trying to reverse-engineer the full model.

While these studies explain fine-tuning well, they overlook a critical issue: they focus on the sparsity in activations in order to locate the key components in a model, while the source of the activations, i.e., the model parameters, undergoes non-sparse changes during fine-tuning, as shown in Figure 3. Specifically, full fine-tuning updates all parameters within a model. Even PEFT methods like LoRA, which optimize within a parameter subspace, ultimately affect all parameters when the adapter is merged with the base model. This dense parameter update mechanism hinders model diffing, particularly for methods based on causal intervention. Such methods simplify the internal computations into a subgraph of the model's computational graph. Since parameter updates are distributed across the entire neural network, parameters of the nodes outside the subgraph also change.

*Corresponding author.

[1]Code is available at https://github.com/Siriuslala/circuit-tuning

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 14254–14276
October 24-29, 2026 ©2026 Association for Computational Linguistics

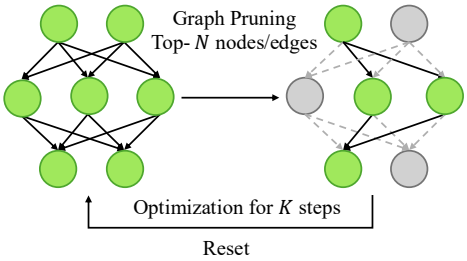


Figure 1: The overall pipeline of our method (see Algorithm 1). Circuit-tuning iteratively finds the circuit for a specific task and updates the relevant parameters.

Consequently, previous methods only consider parameters within the subgraph while ignoring updates outside of it, leading to potentially unreliable conclusions. Thus, we argue that the difference between the base model and the fine-tuned model lies in parameter changes, which are independent of model input. When researchers zoom in on task-related subgraphs to pursue sparsity for better interpretability, the essence of model change, i.e., the parameter updates, must not be ignored.

We thus ask: *can we design a new fine-tuning method that actively introduces sparsity in parameter updates, making model diffing analysis more faithful to the changes in model parameters?* To this end, we incorporate the sparsity of circuits relative to the computational graph into the parameter space, and propose **circuit-tuning**, an interpretability-driven fine-tuning method that could achieve sparsity at the parameter-update level. Our idea is intuitive: combining circuit discovery with fine-tuning into a heuristic, iterative prune-and-tune paradigm. Specifically, after each graph pruning step, we fine-tune only the parameters within the circuit. After K updates, we restore the circuit to the initial computational graph and repeat the process, as shown in Figure 1. The goal of this dynamic fine-tuning paradigm is to allow the model to learn to select critical parameters for optimization under restricted update constraints, thereby achieving parameter-update sparsity.

To demonstrate the effectiveness of our method, we designed a new task called Subject-Verb Disagreement (SVD). This task requires the model to reverse standard English subject-verb agreement rules (e.g., from "I eat" to "I eats"). Unlike previous fine-tuning tasks that merely enhance a specific capability, SVD requires the base model to selectively retain, adjust, or even reverse its abilities. This provides more intriguing and richer content for model diffing research. We conducted a series of explorations of circuit-tuning and model diffing on GPT-2, validating our method and offering new findings into fine-tuning mechanisms via stage-wise circuit analysis. Further, to test the practicality of our method, we apply it to the gender debiasing task and larger-scale models. Comparisons with full fine-tuning and LoRA reveal its superiority in both faithful model diffing and task performance.

In summary, we emphasize the necessity of seriously considering and being faithful to model parameter changes in model diffing research. We propose circuit-tuning as a preliminary attempt to introduce parameter-update sparsity during fine-tuning. We have validated its effectiveness and provided new discoveries regarding fine-tuning learning dynamics, offering a new perspective for the study of fine-tuning mechanisms in the interpretability field.

2 Related Work

2.1 Model diffing

Model diffing aims to compare the representations between models or investigate changes within a single model before and after training (Shah et al., 2023). In this work, we focus on the internal changes of LLMs during fine-tuning, a process typically studied via mechanistic interpretability techniques. Some studies utilize sparse autoencoders (Bricken et al., 2024; Wang et al., 2025a; Naghashyar et al., 2025) and their variants, such as cross-coders (Jack et al., 2024; Minder et al., 2025), to observe feature distributions. These works identify shared, acquired, or forgotten features through comparisons between base and fine-tuned models or via stage-wise model diffing. Some approaches employ causal intervention to simplify model computations into circuits and study their evolution throughout the fine-tuning process (Prakash et al., 2024; Jain et al., 2023; Wang et al., 2025b). Other research investigates the directional shift of task vectors during fine-tuning (Lee et al., 2024; Khayatan et al., 2025; Du et al., 2025; Ward et al., 2025). Distinct from the aforementioned research, our work is not limited to post-hoc explanations of fine-tuning. Instead, we aim to actively introduce sparsity during training to make model diffing analysis more faithful to model parameter updates.

2.2 Interpretability-driven training

Recent studies have sought to apply interpretability findings to model training to pursue a priori interpretability. Elhage et al. (2022) proposed re-

placing Transformer activation functions with Softmax Linear Units to reduce neuron polysemanticity. Kamath et al. (2025) trained transcoders to replace model MLPs, thereby constructing attribution graphs capable of tracing internal mechanisms. Golechha et al. (2025) introduced a clusterability loss into the language model objective function to make the model more modular. Concurrent with our work, a study from OpenAI (Gao et al., 2025) also emphasized that circuits selectively abstract model computations into human-understandable structures but do not reflect the actual internal mechanisms. They trained a GPT-2 style toy model with an AbsTopK activation function on 20 simple coding tasks to achieve intrinsic sparsity across different tasks. We view this work as another successful implementation of our motivation, yet our approach differs: while their method introduces activation sparsity during pre-training, our method introduces parameter-update sparsity during fine-tuning based on the model's existing capabilities.

3 Method

We propose circuit-tuning, which incorporates the activation-level sparsity of circuits into parameter updates during fine-tuning. This approach enables surgical fine-tuning through dynamic circuit discovery, making model diffing research more faithful to the changes in model parameters themselves.

3.1 Preliminaries

Causal intervention In this study, we focus on mechanistic interpretability for Transformers, with relevant notations detailed in Appendix A. For a computational graph $\mathcal{G}$, researchers aim to isolate a minimal subgraph (circuit) responsible for a target task from either the activation space (Vig et al., 2020; Meng et al., 2022; Nanda, 2023) or the parameter space (Braun et al., 2025; Bushnaq et al., 2025) to reduce analytical complexity. A common approach involves using causal intervention to identify a circuit $\mathcal{C} = \{\mathcal{V}_T, \mathcal{E}_T\} \subseteq \mathcal{G}$ associated with task T. Here, each node $n \in \mathcal{V}_T$ represents a model component (e.g., an attention head or an MLP layer), and each edge $e \in \mathcal{E}_T$ specifies the information flow between node outputs and inputs. Such graph pruning requires perturbing activations associated with specific nodes or edges and measuring their importance based on the magnitude of change in the output quantified by a metric $\mathcal{L}_m$. By removing low-importance nodes and edges, we can obtain a sparse, task-related circuit that serves as an abstraction of the model's internal computations. Relevant methods are detailed in Appendix C.2.2.

Faithfulness of model diffing To clarify the concept of faithful model diffing, we provide the following definition: Let $\mathcal{P}$ be the set of all parameters of a model. Given a training interval, we define the Update Set $\mathcal{U} \subseteq \mathcal{P}$ as the set of parameters that have undergone non-trivial changes: $\mathcal{U} = \{p \in \mathcal{P} \mid \Delta p \neq 0\}$. Let $\mathcal{S} \subseteq \mathcal{P}$ be the Analysis Set, representing the subset of parameters selected for investigation. A model diffing method is said to satisfy *strict faithfulness* if and only if the Analysis Set encompasses the Update Set: $\mathcal{S} \supseteq \mathcal{U}$. In practice, we aim for $\mathcal{S}$ to encompass $\mathcal{U}$ as extensively as possible to enhance faithfulness.

3.2 Circuit-tuning

Model parameters change completely before and after fine-tuning (i.e., $\mathcal{U} = \mathcal{P}$), with many parameters showing similar magnitudes of change, as illustrated in Figure 3. This dense parameter shift requires that a faithful model diffing method maintains a focus on every parameter change in the fine-tuned model relative to the base model, rather than merely focusing on changes within local circuits.

Yet, to the best of our knowledge, no existing model diffing method achieves this, as it requires an extensive Analysis Set, and the loss of sparsity greatly complicates interpretability analysis. Therefore, rather than expanding the Analysis Set $\mathcal{S}$, we seek to reduce the size of Update Set $\mathcal{U}$. Moving beyond post-hoc analysis, we actively introduce sparsity at the parameter-update level during fine-tuning, ensuring that parameter changes before and after the process are concentrated within the circuit.

Sparse fine-tuning Inspired by the idea of model pruning and continual learning (Cheng et al., 2023; Wang et al., 2023), we dynamically introduce causal intervention into fine-tuning so as to encourage the sparsity in parameter updates. As shown in Figure 1, given a base model M_{base} with its computational graph $\mathcal{G}$ and a target task T, our method consists of a two-step iterative cycle: (1) graph pruning, where we design a quantifiable metric $\mathcal{L}_m$ for the specific model diffing target (e.g., a model behavior) and isolate a circuit $\mathcal{C}_T$ accordingly; (2) circuit fine-tuning, where parameters outside the circuit are frozen, and only those corresponding to model components within the circuit are updated. After K parameter updates, we reset the circuit to

the initial full computational graph $\mathcal{G}$ and repeat the process, as detailed in Algorithm 1.

Can circuit-tuning replace full fine-tuning in model diffing? To make circuit-tuning suitable for model diffing analysis in practice, we require it to: (1) identify and update task-specific parameters, and (2) maintain mechanistic and performance alignment with full fine-tuning. Regarding (1), circuit-tuning is based on the theory of Aghajanyan et al. (2021) that pre-training implicitly minimizes the average description length, allowing fine-tuning with minor modifications on a relatively fixed set of parameters. This enables circuit-tuning to construct the target circuit for a target task from an existing base circuit in a base model. Regarding (2), circuit-tuning maintains mechanistic alignment with full fine-tuning without introducing additional mechanisms that deviate significantly from full fine-tuning. It simply restricts updates to the causal path, which could potentially suppress task-irrelevant updates and stabilize training with even less computation, as validated in Section 4.2, 4.3 and 4.6.

4 Experiments

In this section, we introduce a novel task for studying model diffing called Subject-Verb Disagreement (SVD). We conduct comprehensive analyses of circuit-tuning and model diffing on this task, providing new perspectives on the fine-tuning mechanisms. Furthermore, we perform experiments on larger models and the gender debiasing task, further demonstrating the practical value of our method.

4.1 The Subject-Verb Disagreement Task

Task definition The SVD task originates from the Subject-Verb Agreement (SVA) task used in syntactic evaluation research on language models, which is also widely adopted in interpretability studies (Lakretz et al., 2019; Finlayson et al., 2021; Marks et al., 2024). Unlike the SVA task, our goal is to fine-tune the model to pair a subject with a verb that does not match its grammatical number based on existing English grammar. Examples such as "We does" and "The cow eat" illustrate this. We created this task for two reasons: (1) it involves a relatively singular model capability, making it sufficiently simple and clear for model diffing analysis; (2) unlike typical fine-tuning tasks that target the enhancement of a specific ability, the SVD task requires the model to selectively maintain, adjust, or even reverse task-related capabilities, providing a novel scenario for model diffing research.

Formally, in each sentence, we focus on the subject, the verb, and the END token that is the token before the verb. A mismatched verb, such as the third-person singular "does" for the first-person "we," is termed a *flipped verb*. Based on the proximity of the subject and verb, we categorize the sentence structures into two types: Type I, where the subject and verb are adjacent (e.g., "*They are*"), and Type II, where the subject and verb are separated by other constituents (e.g., "*The kid* that is holding an ice cream in hand *is* .."). This requires the model to possess at least two abilities: subject identification and subject-verb matching, which are the core focus of our model diffing analysis.

The automatic evaluation metric for this task is the logit difference between the flipped verb v_{flip} and the original verb v_{orig} at the END token:

$$\text{LD}(v_{\text{flip}}, v_{\text{orig}}) = \text{Logit}(v_{\text{flip}}) - \text{Logit}(v_{\text{orig}}) \quad (1)$$

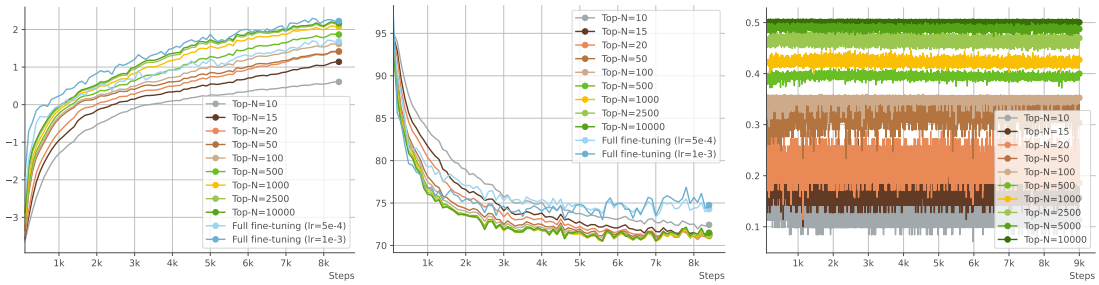
where $\text{Logit}(\cdot) \in \mathbb{R}$ denotes the logit of a possible next token at the END token. A larger $\text{LD}(v_{\text{flip}}, v_{\text{orig}})$ indicates a higher probability of the model outputting v_{flip} over v_{orig}.

Dataset construction Unlike previous studies that used synthetic data from sentence templates, we collected real-world data from the Pile dataset (Gao et al., 2020) to ensure diversity and authenticity, enabling the model to truly acquire new grammar through fine-tuning. We extract 30k sentences in the present tense in English and flip the forms of the verbs according to a set of rules detailed in Appendix C.1. The data is split into training (24k), validation (3k), and test (3k) sets. We have ensured the high quality of our dataset and believe it is valuable for further research.

4.2 Implementation details

We use GPT-2-small (Radford et al., 2019) in this task to simplify the analysis. We set the output of the attention and MLP in each layer as upstream nodes, and the input of the query, key, value, and MLP in each layer as downstream nodes. We use the logit difference described in Section 4.1 as the metric $\mathcal{L}_m$ to quantify output changes during graph pruning. Node and edge scores are averaged over a batch of samples such that the circuit can capture the commonalities across samples.

Choice of graph pruning method We considered various methods to identify the circuit: (1)



(a) Logit difference (b) PPL (c) Tunable parameter ratio

Figure 2: The changes in indicators (calculated on the validation set) with various choices of N (number of edges) during training. In 2c, the turning point at $N = 1000$ implies the minimal size of the circuit for a target task at this point (also see Figure 11,12). Note that the parameter ratio depends on the granularity of nodes. Theoretically more granular nodes result in lower parameter ratio because the attribution of the key parameters would be more precise.

activation patching methods such as AtP (Nanda, 2023); (2) path patching methods such as EAP (Syed et al., 2024) and EAP-IG (integrated gradients) (Hanna et al., 2024). To improve efficiency, we adopt attribution patching to simplify the computation. Additionally, we use mean ablation for intervention. Details of these algorithms are provided in Appendix C.2.2. In this case, graph pruning requires only *one* forward and backward pass, which takes the same as a normal training step and thus enhances computational efficiency. Notably, while using activations from corrupted prompts would ideally provide a clearer supervisory signal, it requires additional effort to construct. To ensure the validity of mean ablation, we provide ablation studies and detailed discussions in Section 4.5.

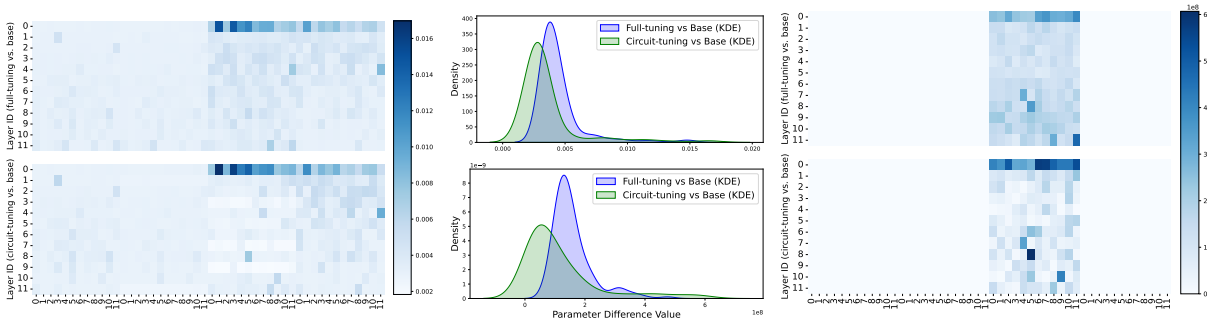
Choice of N We perform graph pruning by selecting the top-N nodes or edges with the largest attribution scores. As described in Section 3.2, we assume that the SVA circuit of the pre-trained base model and the SVD circuit of the fine-tuned model share a significant portion of their structure. Therefore, one method for selecting N is to test candidate values by identifying the circuit on a batch of SVA data before training for each N, then calculating its faithfulness and completeness according to Equation 2 and 3 (Marks et al., 2024). Faithfulness tells how much performance a circuit gets, while completeness tells how much performance a circuit fails to capture. From this, we can plot curves with N as the independent variable, and the value of N at the elbow point is the ideal choice, as shown in Figure 6. However, for the sake of rigor, we still sweep over a range of N in circuit-tuning.

Other hyperparameters The update frequency K is a flexible parameter. A larger K makes circuit-tuning more efficient than full fine-tuning, yet may hinder the formation of the circuit for the target task. We set $K = 8$ in the main experiment and discuss its impact in the ablation study in Section 4.5. Additionally, we set batch size to 16 and learning rate to 1e-3, and mini-batch SGD is used for optimization. We train the model under each setting for 6 epochs where the performance almost converges. More details can be found in Appendix C.3.1.

4.3 Main results

Training Dynamics Our analysis primarily focuses on EAP-based circuit-tuning, as experimental results show that activation patching and path patching yield very similar outcomes, whereas path patching provides a zoomed in understanding of how model components compose together. As shown in Figure 2a, the logit difference between v_{flip} and v_{orig} flips from a negative value to a positive one, signifying that the model adjusts its grammar from SVA to SVD. We find that as N increases, the growth of the logit difference converges near $N = 1000$, with circuit-tuning remaining largely on par with full fine-tuning. The number of tunable parameters is also saturated at this point, as shown in Figure 2c, 11 and 12. This implies that when $N = 1000$, almost all the necessary parameters for the SVD task are included, suggesting the minimality of the circuit at this point. This aligns with the elbow points on the faithfulness and completeness curves and can be viewed as a further extension of the research on intrinsic dimension. This point is further elaborated in Appendix C.3.2.

Optimization Stability We tracked the Perplexity (PPL) on the validation set during training. We observed that the PPL for full fine-tuning fluctuates significantly, whereas circuit-tuning achieves



(a) Parameter difference in weights. (b) Distribution of the differences. (c) Parameter difference in bias.

Figure 3: In 3a and 3c, the top and bottom rows correspond to full fine-tuning and circuit-tuning, respectively. In 3a, heatmaps from left to right represent W_Q, W_K, W_V, and W_O (12 heads across X-axis, 12 layers down Y-axis); 3c follows the same layout. In 3b, the top and bottom rows of plots correspond to weights and bias, respectively.

stable convergence. Even when the learning rate for full fine-tuning is reduced to 5e-4 for stable training, its PPL rises after 5k steps, as shown in Figure 2b. This implies that circuit-tuning provides better training stability over full fine-tuning. This is because under identical experimental settings, circuit-tuning adjusts a smaller yet more critical subset of parameters compared to full fine-tuning. It improves target task performance by minimally affecting the parameters corresponding to other task-irrelevant capabilities. For training stability, we also provide a simple proof from the perspective of Lipschitz continuity in Appendix B.2.

4.4 Model diffing analyses

Finding 1: Sparsity in parameter updates, improved model diffing faithfulness To verify that our method achieves the expected sparsity and faithfulness in Section 3.1, we calculate the parameter changes in the fine-tuned model M_{ft} relative to the base model M_{base}. Specifically, we focus on the weights W_Q, W_K, W_V, W_O, and biases within the attention modules of each layer. For the j-th attention head in the i-th layer (head.i.j), let θ_{base} and θ_{ft} denote the parameters before and after fine-tuning, respectively. The relative change is calculated as: $\frac{||\theta_{ft}-\theta_{base}||_F}{||\theta_{base}||_F}$, where $||\cdot||_F$ denotes the Frobenius norm. Visualization results for weights and biases under full fine-tuning and circuit-tuning are shown in Figure 3a and 3c, respectively. Clearly, our method concentrates parameter updates within a small subset with significant changes. In contrast, while the full fine-tuning visualization identifies several parameters with substantial changes, nearly all other parameters are affected to varying degrees. We also plotted the distribution of parameter changes using Kernel Density Estimation (KDE) in Figure 3b. We can see that the mode of the parameter distribution for circuit-tuning is closer to zero, whereas the mode for full fine-tuning is biased toward non-zero values with significantly higher kurtosis. This indicates that compared with full fine-tuning, our method more closely follows the internal computational paths of the target task, making subsequent model diffing analysis more faithful to parameter changes.

Another finding is that parameter changes are concentrated in W_V, W_O, and b_V. This can be explained by the theory of Elhage et al. (2021), which states that the QK circuit in Transformers determines which part of the context to attend to, while the OV circuit determines how to write the attended information into the residual stream. In our scenario, the shift from SVA to SVD mainly stems from changes in subject-verb matching rather than subject identification. Thus, parameter changes occur mainly in weights associated with the attention value and output. Similar findings have been reported in previous work (Hu et al., 2022).

Finding 2: Dynamic circuits are generally stable Wang et al. (2025b) discovered that in tasks involving only capability enhancement, the circuit $\mathcal{C}_{\text{ft}}$ of the fine-tuned model M_{ft} is built upon the circuit $\mathcal{C}_{\text{base}}$ of the base model M_{base}, thus sharing many structures. We aim to investigate whether this conclusion holds for the SVD task. We measure the overlap between $\mathcal{C}_{\text{base}}$ and $\mathcal{C}_{\text{ft}}$ by calculating the Intersection over Union (IoU): $\frac{|\mathcal{C}base \cap \mathcal{C}ft|}{|\mathcal{C}base \cup \mathcal{C}ft|}$. We calculate the node-level IoU^{n} and edge-level IoU^{e} separately. The edge scores used for graph pruning are calculated on a batch of 32 randomly sampled instances from the SVA/SVD tasks, averaged over five repetitions. Results show that at $N = 1000$, $\text{IoU}^{\text{n}} = 71.47\%$, indicating that $\mathcal{C}_{\text{ft}}$ does not exhibit

significant changes in terms of model components compared to $\mathcal{C}_{\text{base}}$. This reflects the shared functionality between the SVA and SVD tasks. However, $\text{IoU}^{\text{e}} = 37.55\%$, suggesting that while these model components are largely shared between $\mathcal{C}_{\text{base}}$ and $\mathcal{C}_{\text{ft}}$, the relationships between them have changed. This point, which has been overlooked in previous work, will be discussed in Finding 4.

Finding 3: The flip in weights and activations We further investigate the changes in specific functional attention heads within the circuit. First, we identify heads responsible for subject identification by observing attention patterns at the END token position. Since most attention patterns are diagonal, where the END token primarily attends to itself, we focus on Type II sentences discussed in Section 4.1. In these cases, the subject is not the END token, which helps identify heads that truly attend to the subject. Results indicate that M_{base} and M_{ft} share the same heads for subject identification, such as head.8.5, 10.9, 11.8, as shown in Figure 13, 14, 15.

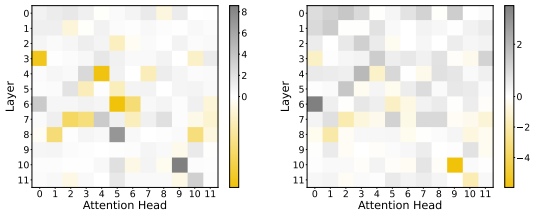


Figure 4: Heatmaps of the dot-product described in Finding 3 before (left) and after (right) fine-tuning, demonstrating the flip in activations from SVA to SVD.

The above finding aligns with the expected QK-circuit invariance, but how do the OV-circuits of these heads evolve? To this end, we use direct logit attribution to measure the contribution of each head.$i.j$ output to the final output via the dot product: $\langle \text{ln_final}(\text{Attn}^{i,j}(x)), W_U[v_{\text{flip}}] - W_U[v_{\text{orig}}] \rangle$. Here, ln_final denotes the LayerNorm between the final layer output and the language model head, and $W_U[\cdot]$ represents the unembedding vector of a token. The underlying principle is that the residual stream in Transformers allows the final output to be viewed as a linear superposition of outputs from every module in each layer. We compute the dot product from 128 samples in the test set. From Figure 4, we observe a significant directional reversal in heads 8.5, 10.9, 11.8, etc., meaning their outputs have flipped from SVA to SVD.

To delve deeper into the essence of this flip, we move on to the parameters. We perform stage-wise model diffing: for any checkpoint M_t at time t between M_{base} and M_{ft}, we calculate the change in the OV matrix for each head.$i.j$ relative to the previous state M_{t-1}, i.e., $\Delta\theta_t = W_{OV}^{i,j,t} - W_{OV}^{i,j,t-1}$. Here, $W_{OV}^{i,j} = W_O^{i,j} W_V^{i,j} \in \mathbb{R}^{D_h \times D_h}$, where D_h is the head dimension. We use PCA to reduce these changes to two dimensions and visualize them in Figure 5. Interestingly, a distinct directional reversal of $\Delta\theta_t$ is observed at 3500 steps. We view this as the source of the flip in Figure 4, since the shifts in the principal directions of $\Delta\theta_t$ signify a change in the update direction of $W_{OV}^{i,j}$, i.e., transitioning from the SVA grammar toward SVD.

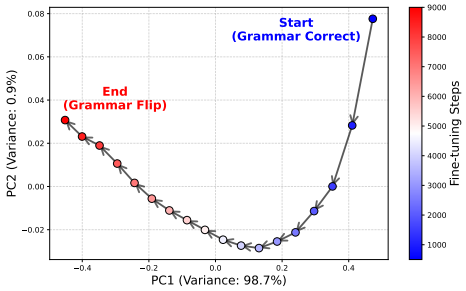


Figure 5: PCA visualization of the $\Delta\theta_t$ in head.8.5, demonstrating the flip in parameters from SVA to SVD.

Finding 4: The evolution of functional pathways In Finding 2, we observed that the edge-level overlap between $\mathcal{C}_{\text{base}}$ and $\mathcal{C}_{\text{ft}}$ is relatively low. Therefore, we wonder how the relationships between nodes have evolved. We began by exploring the interactions between upstream and downstream nodes. Specifically, for each attention head in Finding 3 that directly impacts the final output, we sequentially knocked out its upstream nodes. We then quantified the change in the attention pattern of these direct-impact heads to identify the upstream heads influencing their behavior, e.g., identifying head.1.8 and head.2.11 as influential to head.10.9 (see Appendix C.3.4 for details). Then, we conducted a stage-wise analysis of the edges by examining changes in edge scores. As shown in Figure 17a, we surprisingly found that the scores of edges between causally related nodes increased continuously. For example, the edge from head.1.8.out to head.10.9.key achieved a Pearson correlation coefficient of 0.942 with training steps. This phenomenon is highly similar to Hebbian Learning (Hebb, 2005), where the simultaneous activation of cells leads to pronounced increases in synaptic strength between those cells. By analogy, we conclude that during fine-tuning, as components related to SVD are continuously activated, the functional pathways between causally

related nodes are progressively strengthened with the guidance of the training data and objectives. Simultaneously, we found that the scores of some edges are negatively correlated with training steps, as shown in Figure 17b. We found that the nodes associated with these weakened edges are largely irrelevant to the SVD task. This resembles lateral inhibition (Jacobson, 1993), where one neuron firing reduces the amount other neurons fire. Such a learning mechanism ensures that the task-irrelevant structures fade out and sparsity emerges, making the profile of the task-relevant circuit clearer.

4.5 Ablation and Discussions

Comparison of graph pruning methods Through extensive comparative experiments, we find that circuit-tuning is not sensitive to the choice of graph pruning method. For AtP and EAP, there is no significant performance difference except for the required N values. Compared to EAP, EAP-IG provides a slight performance boost by improving the circuit faithfulness (Hanna et al., 2024). For instance, when $N = 500$, the logit difference of EAP-IG-based circuit-tuning reaches 1.559, slightly higher than that of EAP (1.516). Further comparisons are provided in Appendix C.2.2. However, this advantage becomes less significant as N increases and requires more computation. Therefore, we recommend using EAP.

How does the value of K affect circuit-tuning? For each N, we adjust the value of K within the range [1, 4, 8, 16, 32, 64, 128]. We observe that when $K \leq 32$, the logit difference remains almost unchanged. As K continues to increase, the logit difference begins to decline, signifying a loss in performance. We hypothesize that an excessively large K leads to a lag in circuit updates, whereas a smaller K introduces more dynamic changes, which helps the model develop a circuit better suited for the target task. Comparative experiments are detailed in Appendix C.2.3.

Is mean ablation reliable? To demonstrate reliability, we must verify that components outside the circuit obtained via mean ablation are task-irrelevant. We do this by randomly unfreezing a certain percentage of nodes outside the circuit after each graph pruning step, thereby introducing more parameters into optimization. In practice, we set $N = 1000$ and set the ratios of randomly activated parameters outside the circuit to 0.05, 0.1, 0.2, 0.3, and 0.4. For each ratio, we conducted four experiments using different random seeds. As shown in Figure 8, we observed that random activation fails to provide additional performance gains. Instead, it leads to a slight decline in performance. This indicates that most nodes outside the circuit are task-irrelevant, confirming that the circuit obtained through mean ablation possesses high completeness. In fact, another consideration for using mean ablation is our goal to identify a circuit applicable to the entire SVD task rather than one tailored to specific samples. In this sense, mean ablation can yield more complete circuits than using activations from corrupted prompts, a point supported by the work of Gao et al. (2025). More details and discussions are provided in Appendix C.2.2.

4.6 Is circuit-tuning applicable in practice?

Finally, we investigate whether circuit-tuning can perform better on larger models and more practical tasks. We focus on the gender debiasing task, which requires an LLM to develop an unbiased perspective on genders. We use BUG (Levy et al., 2021) for training, a large-scale dataset of sentences sampled from real-world corpora that could ensure balance between male and female entities. For evaluation, we use WinoBias (Zhao et al., 2018), a dataset for coreference resolution focused on gender bias, and adopt the discrimination risk (Liu et al., 2024) as the metric. Task settings and data examples are detailed in Appendix D.1.

We test on GPT-2-small, Llama-3.2-1B/3B, and Llama-3.1-8B (Dubey et al., 2024), using full fine-tuning and LoRA as baselines. We plot the distribution of occupations w.r.t. the discrimination risk of the fine-tuned model in Figure 18, and find that circuit-tuning reduces risk to the lowest level, proving the practical utility of our method for real-world tasks. We also compare the distribution of parameter differences between the base model and models fine-tuned via circuit-tuning / LoRA under comparable performance levels (Figure 19, 20, 21), which further demonstrates that our method can better concentrate the optimization on the key parameters for the target task. Experimental settings and discussions are detailed in Appendix D.2.

Further, we evaluated on Llama-3.2-1B/3B and Llama-3.1-8B across reasoning tasks (GSM8K (Cobbe et al., 2021) (zero-shot), Contexthub (Hua et al., 2024) (F1)) and reasoning-free tasks (SQuAD 2.0 (Rajpurkar et al., 2018) (Exact match)). To assess the preservation of general capabilities, we tested general abilities (MMLU

Table 1: Main experimental results on different tasks. "/" denotes the original baseline without fine-tuning.

Methods & Tasks	Math	Logical Reasoning	Reading	General Capabilities
Llama-3.2-1B-it	40.71	20.35	39.68	/
Llama-3.2-1B-it-full-tuning	45.47	26.89	36.73	0.14
Llama-3.2-1B-it-LoRA	44.58	22.51	34.30	-8.55
Llama-3.2-1B-it-AdaLoRA	44.90	24.40	36.12	-4.34
Llama-3.2-1B-it-BitFit	41.20	20.87	39.90	1.35
Llama-3.2-1B-it-circuit-tuning	45.56	27.06	41.78	2.33
Llama-3.2-3B-it	70.36	42.71	54.12	/
Llama-3.2-3B-it-full-tuning	75.44	47.32	55.63	-2.54
Llama-3.2-3B-it-LoRA	73.54	46.27	54.73	-3.92
Llama-3.2-3B-it-AdaLoRA	74.82	46.35	55.63	-1.95
Llama-3.2-3B-it-BitFit	71.52	44.01	53.92	-1.75
Llama-3.2-3B-it-circuit-tuning	75.35	47.59	56.58	-1.80
Llama-3.1-8B-it	76.19	46.41	58.92	/
Llama-3.1-8B-it-full-tuning	83.76	49.53	59.60	-2.76
Llama-3.1-8B-it-LoRA	80.21	47.64	58.97	-4.15
Llama-3.1-8B-it-AdaLoRA	81.50	47.82	59.14	-1.16
Llama-3.1-8B-it-BitFit	78.20	46.77	59.00	-0.83
Llama-3.1-8B-it-circuit-tuning	83.51	49.75	59.14	-0.94

(Hendrycks et al., 2020), Winogrande (Sakaguchi et al., 2021), IFEval (Zhou et al., 2023)), reasoning (GPQA) (Rein et al., 2023), coding (HumanEval) (Chen et al., 2021), and multilingual skills (MGSM (Shi et al., 2022)). We report relative performance change ($(x' - x)/x$).

As for the metric $\mathcal{L}_m$ for the quantification of a model's prediction during circuit discovery, gender debiasing uses the logit difference between gendered words (he/his vs. she/her), while other tasks use the standard negative log-probability loss. We focus on overall task performance rather than isolating specific sub-capabilities (e.g., addition in math), as task behaviors span multiple tokens.

We compared circuit-tuning with full fine-tuning, LoRA, AdaLoRA (Zhang et al., 2023) and BitFit (Zaken et al., 2022). AdaLoRA is a variant of LoRA that uses an importance score to prune the eigenvalues of SVD decomposition in order to to encourage sparsity in parameter updates. BitFit is a sparse-fine-tuning method where only the bias-terms of the model are being modified. We set batch size to 16. Learning rates were 3e-5 for full fine-tuning and 1e-4 for others. For LoRA, we set optimal hyperparameters $r = 32$ and $\alpha = 64$. For circuit-tuning, we set $K = 8$, using $N = 2000, 3000, 4000$ for 1B, 3B, and 8B models, respectively.

All results in the table are averaged over 10 runs with different seeds. Overall, circuit-tuning shows no significant performance difference compared to full fine-tuning, except for outperforming it by 5.05% on Llama-3.2-1B reading comprehension. Since our goal is not boosting performance, this comparable efficacy confirms that circuit-tuning does not severely deviate from standard fine-tuning. The main difference is that circuit-tuning better preserves general capabilities. Conversely, LoRA performs worse than standard fine-tuning as it lacks explicit selection of task-related parameters. Though AdaLoRA selects eigenvalues via importance scores, its performance still lags behind circuit-tuning due to the global impact of the LoRA adapter on all parameters.

5 Conclusion

We propose circuit-tuning to make model diffing research more faithful to changes in model parameters. We introduce a novel SVD task for the study of fine-tuning and conduct comprehensive experiments and analyses on this task. Our work provides a new methodology for studying and understanding the mechanisms of fine-tuning via model diffing.

6 Limitations

There are still several limitations that we reserve for future work:

(1) Our experiments employ coarse-grained nodes (attention heads or MLP layers) and primarily focus on attention heads. However, MLPs also contain significant potential. In practice, circuit-tuning can be applied to graphs with finer nodes. For instance, partitioning an MLP into multiple "MLP heads" is an initial trial, as we have done on Llama in the gender debiasing task. We believe future research could integrate circuit-tuning with SAEs and attribution graphs. This would enable stage-wise model diffing research, such as that conducted by Kamath et al. (2025), to achieve sparsity in both parameter updates and representation spaces.

(2) While this study applies circuit-tuning specifically to fine-tuning-based model diffing, we believe our method is applicable to model diffing in reinforcement learning. We will explore this direction in future work.

7 Ethical considerations

This paper presents work whose goal is to advance the field of Machine Learning. Since our work concerns AI alignment and involves modifying a model's gender stereotype in Section 4.6, we expect our work to be used under the guidance of human values.

References

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: long papers)*, pages 7319–7328.

Dan Braun, Lucius Bushnaq, Stefan Heimersheim, Jake Mendel, and Lee Sharkey. 2025. Interpretability in parameter space: Minimizing mechanistic description length with attribution-based parameter decomposition. *arXiv preprint arXiv:2501.14926*.

Trenton Bricken, Siddharth Mishra-Sharma, Jonathan Marcus, Adam Jermyn, Christopher Olah, Kelley Rivoire, and Thomas Henighan. 2024. Stage-wise model diffing. *Transformer Circuits Thread*.

Lucius Bushnaq, Dan Braun, and Lee Sharkey. 2025. Stochastic parameter decomposition. *arXiv preprint arXiv:2506.20790*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. 2023. A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46:10558–10578.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. 2023. Towards automated circuit discovery for mechanistic interpretability. *Advances in Neural Information Processing Systems*, 36:16318–16352.

Hongzhe Du, Weikai Li, Min Cai, Karim Saraipour, Zimin Zhang, Himabindu Lakkaraju, Yizhou Sun, and Shichang Zhang. 2025. How post-training reshapes LLMs: A mechanistic view on knowledge, truthfulness, refusal, and confidence. *arXiv preprint arXiv:2504.02904*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Nelson Elhage, Tristan Hume, Catherine Olsson, Neel Nanda, Tom Henighan, Scott Johnston, Sheer ElShowk, Nicholas Joseph, Nova DasSarma, Ben Mann, Danny Hernandez, Amanda Askell, Kamal Ndousse, Andy Jones, Dawn Drain, Anna Chen, Yuntao Bai, Deep Ganguli, Liane Lovitt, and 14 others. 2022. Softmax linear units. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2022/solu/index.html.

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, and 6 others. 2021. A mathematical framework for Transformer circuits. *Transformer Circuits Thread*.

Matthew Finlayson, Aaron Mueller, Sebastian Gehrmann, Stuart M Shieber, Tal Linzen, and Yonatan Belinkov. 2021. Causal analysis of syntactic agreement mechanisms in neural language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural*

Language Processing (Volume 1: Long Papers), pages 1828–1843.

Isabel O Gallegos, Ryan A Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K Ahmed. 2024. Bias and fairness in large language models: A survey. *Computational Linguistics*, pages 1–79.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The Pile: An 800gb dataset of diverse text for language modeling. *ArXiv*, abs/2101.00027.

Leo Gao, Achyuta Rajaram, Jacob Coxon, Soham V Govande, Bowen Baker, and Dan Mossing. 2025. Weight-sparse Transformers have interpretable circuits. *arXiv preprint arXiv:2511.13653*.

Satvik Golechha, Maheep Chaudhary, Joan Velja, Alessandro Abate, and Nandi Schoots. 2025. Modular training of neural networks aids interpretability. *arXiv preprint arXiv:2502.02470*.

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. 2024. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. *arXiv preprint arXiv:2403.17806*.

Donald Olding Hebb. 2005. *The organization of behavior: A neuropsychological theory*. Psychology press.

Stefan Heimersheim and Neel Nanda. 2024. How to use and interpret activation patching. *arXiv preprint arXiv:2404.15255*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. LoRA: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Wenyue Hua, Kaijie Zhu, Lingyao Li, Lizhou Fan, Shuhang Lin, Mingyu Jin, Haochen Xue, Zelong Li, JinDong Wang, and Yongfeng Zhang. 2024. Disentangling logic: The role of context in large language model reasoning capabilities. *arXiv preprint arXiv:2406.02787*.

Jack, Adly Lindsey, Jonathan Templeton, Thomas Marcus, Joshua Conerly, Christopher Batson, and Olah. 2024. Sparse crosscoders for cross-layer features and model diffing. *Transformer Circuits Thread*.

Marcus Jacobson. 1993. *Foundations of neuroscience*. Springer Science & Business Media.

Samyak Jain, Robert Kirk, Ekdeep Singh Lubana, Robert P Dick, Hidenori Tanaka, Edward Grefenstette, Tim Rocktäschel, and David Scott Krueger. 2023. Mechanistically analyzing the effects of fine-tuning on procedurally defined tasks. *arXiv preprint arXiv:2311.12786*.

Harish Kamath, Emmanuel Ameisen, Isaac Kauvar, Rodrigo Luger, Wes Gurnee, Adam Pearce, Sam Zimmerman, Joshua Batson, Thomas Conerly, Chris Olah, and Jack Lindsey. 2025. Tracing attention computation through feature interactions. *Transformer Circuits Thread*.

Pegah Khayatan, Mustafa Shukor, Jayneel Parekh, Arnaud Dapogny, and Matthieu Cord. 2025. Analyzing finetuning representation shift for multimodal LLMs steering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2206–2216.

János Kramár, Tom Lieberum, Rohin Shah, and Neel Nanda. 2024. AtP*: An efficient and scalable method for localizing LLM behaviour to components. *arXiv preprint arXiv:2403.00745*.

Yair Lakretz, German Kruszewski, Theo Desbordes, Dieuwke Hupkes, Stanislas Dehaene, and Marco Baroni. 2019. The emergence of number and syntax units in LSTM language models. *arXiv preprint arXiv:1903.07435*.

Andrew Lee, Xiaoyan Bai, Itamar Pres, Martin Wattenberg, Jonathan K Kummerfeld, and Rada Mihalcea. 2024. A mechanistic understanding of alignment algorithms: A case study on DPO and toxicity. *arXiv preprint arXiv:2401.01967*.

Shahar Levy, Koren Lazar, and Gabriel Stanovsky. 2021. Collecting a large-scale gender bias dataset for coreference resolution and machine translation. In *Findings of the association for computational linguistics: EMNLP 2021*, pages 2470–2480.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. 2018. Measuring the intrinsic dimension of objective landscapes. *arXiv preprint arXiv:1804.08838*.

Y Liu, K Yang, Z Qi, X Liu, Y Yu, and C Zhai. 2024. Prejudice and volatility: A statistical framework for measuring social discrimination in large language models. *Preprint*, arXiv:2402.15481.

Samuel Marks, Can Rager, Eric J Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. 2024. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. *arXiv preprint arXiv:2403.19647*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. *Advances in neural information processing systems*, 35:17359–17372.

Julian Minder, Clément Dumas, Caden Juang, Bilal Chughtai, and Neel Nanda. 2025. Overcoming sparsity artifacts in crosscoders to interpret chat-tuning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Lachin Naghashyar, Hunar Batra, Ashkan Khakzar, Philip Torr, Ronald Clark, Christian Schroeder de Witt, and Constantin Venhoff. 2025. Towards understanding multimodal fine-tuning: A case study into spatial features. In *Mechanistic Interpretability Workshop at NeurIPS 2025*.

Neel Nanda. 2023. Attribution patching: Activation patching at industrial scale. *URL: https://www. neelnanda. io/mechanistic-interpretability/attribution-patching*.

Leland Gerson Neuberg. 2003. Causality: models, reasoning, and inference, by judea pearl, cambridge university press, 2000. *Econometric Theory*, 19(4):675–685.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. 2020. Zoom in: An introduction to circuits. *Distill*.

Judea Pearl. 2001. Direct and indirect effects. *Probabilistic and Causal Inference*.

Nikhil Prakash, Tamar Rott Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. 2024. Fine-tuning enhances existing mechanisms: A case study on entity tracking. *arXiv preprint arXiv:2402.14811*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *ArXiv*, abs/2311.12022.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Harshay Shah, Sung Min Park, Andrew Ilyas, and Aleksander Madry. 2023. Modeldiff: A framework for comparing learning algorithms. In *International Conference on Machine Learning*, pages 30646–30688. PMLR.

Lee Sharkey, Bilal Chughtai, Joshua Batson, Jack Lindsey, Jeff Wu, Lucius Bushnaq, Nicholas Goldowsky-Dill, Stefan Heimersheim, Alejandro Ortega, Joseph Bloom, and 1 others. 2025. Open problems in mechanistic interpretability. *arXiv preprint arXiv:2501.16496*.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, and 1 others. 2022. Language models are multilingual chain-of-thought reasoners. *arXiv preprint arXiv:2210.03057*.

Aaquib Syed, Can Rager, and Arthur Conmy. 2024. Attribution patching outperforms automated circuit discovery. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 407–416.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. *Advances in neural information processing systems*, 33:12388–12401.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. 2023. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46:5362–5383.

Miles Wang, Tom Dupré la Tour, Olivia Watkins, Alex Makelov, Ryan A Chi, Samuel Miserendino, Jeffrey Wang, Achyuta Rajaram, Johannes Heidecke, Tejal Patwardhan, and 1 others. 2025a. Persona features control emergent misalignment. *arXiv preprint arXiv:2506.19823*.

Xu Wang, Yan Hu, Wenyu Du, Reynold Cheng, Benyou Wang, and Difan Zou. 2025b. Towards understanding fine-tuning mechanisms of LLMs via circuit analysis. *arXiv preprint arXiv:2502.11812*.

Jake Ward, Chuqiao Lin, Constantin Venhoff, and Neel Nanda. 2025. Reasoning-finetuning repurposes latent representations in base models. *arXiv preprint arXiv:2507.12638*.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. In *Proceedings of the 2018 Conference*

of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), pages 15–20.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

A Transformer Architecture

The models we use in our experiments are all decoder-only Transformers. We briefly introduce the Transformer architecture from the mechanistic view, together with its implementation.

A single input of Transformer is $x_0 \in \mathbb{R}^T$, where T is the length of the sequence. The input is firstly embedded into a vector $x \in \mathbb{R}^{D\times T}$ via the embedding matrix $W_E \in \mathbb{R}^{D\times V}$, where D is the model dimension. Then x will go through l layers of Transformer blocks for various processings. From the view of (Elhage et al., 2021), we can think of the residual stream as a communication channel that simply receives the output of the attention and MLP operations. Each operation reads information from the residual stream and writes the processed information into it. Thus the residual stream is actually the linear sum of various transformations of x together with the original input x.

In each Transformer layer $i (i \in [0, L))$, we consider the implementation of multi-head attention. The model dimension is split into H parts, and the attention operation is performed with H attention heads in parallel. Each head is thought to be responsible for a specific function. Consider head.$i.j$ $(j \in [0, H))$, the input x is firstly projected into query, key and value via $W_Q^{i,j}$, $W_K^{i,j}$ and $W_V^{i,j}$. The projection matrices are all in shape $\mathbb{R}^{\frac{D}{H}\times D}$, thus x is projected into $x^j_{Q/K/V} \in \mathbb{R}^{\frac{D}{H}\times T}$. Then attention pattern $A^{i,j} \in \mathbb{R}^{T\times T}$ is computed via $(W_Q^{i,j}x_Q^j)(W_K^{i,j}x_K^j)^\top$ and some scaling and Softmax operations. After that, the weighted output $z \in \mathbb{R}^{\frac{D}{H}\times T}$ is computed via $(W_V^{i,j}x_V^j)A^{i,j}$. Finally, the output of head.$i.j$ $\text{Attn}^{i,j}(x) \in \mathbb{R}^{D\times T}$ is calculated via $W_O^{i,j} \cdot z$, where $W_O^{i,j} \in \mathbb{R}^{D\times\frac{D}{H}}$. Thus, final output of the self-attention in layer i is $\text{Attn}^i(x) = \sum_{j=1}^H \text{Attn}^{i,j}(x)$.

For the MLP in each layer, the input x is projected into $x^i_{in} \in \mathbb{R}^{D_{mlp}\times T}$ via $W_{in} \in \mathbb{R}^{D_{mlp}\times D}$, and projected back to $\text{MLP}^i(x) \in \mathbb{R}^{D\times T}$ via $W_{out} \in \mathbb{R}^{D\times D_{mlp}}$. In the Llama architecture (Touvron et al., 2023), the input x is firstly projected into $x^i_{pre} \in \mathbb{R}^{D_{mlp}\times T}$ via $W^i_{gate} \in \mathbb{R}^{D_{mlp}\times D}$ and is applied with an activation layer, then a dot product is performed between the activations and $x^i_{in} \in \mathbb{R}^{D_{mlp}\times T}$ which is the input projected by $W_{in} \in \mathbb{R}^{D_{mlp}\times D}$. Note that we can also split the MLP into MLP heads, which is detailed in Appendix D.2.

The output of all the L layers are projected into $x \in \mathbb{R}^{V\times T}$ by the unembedding matrix $W_U \in \mathbb{R}^{V\times D}$, which is called the logits. The logit at the end of the sequence is further mapped into a probability distribution with Softmax over the vocabulary for predicting the next token.

B Further discussions on circuit-tuning

B.1 Algorithm details

We provide a detailed formulation of circuit-tuning in Algorithm 1, which is the version with EAP as the graph pruning method.

Algorithm 1 Circuit-tuning (based on EAP)

Input: dataset $\mathcal{X}$, model M, the number of edges to save N, the number of optimization steps after graph pruning K.
Initialize graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ from M, circuit $\mathcal{C} = \mathcal{G}$, and the iteration step $i = 0$.
for mini-batch $\mathcal{X}_T = \{x_1, x_2, ..., x_t\}$ in $\mathcal{X}$ **do**
  Run a forward and backward pass on $\mathcal{X}_T$
  if $i \bmod K == 0$ **then** ▷ **Graph pruning**
    Reset: $\mathcal{C} \leftarrow \mathcal{G}$
    Get the edge score for each edge via edge attribution patching.
    $\mathcal{E}_T \leftarrow \{e \,|\, e \in \mathcal{E}$ with Top$N$ scores $\}$
    $\mathcal{C} \leftarrow \{\mathcal{V}_T, \mathcal{E}_T\}$, where $\mathcal{V}_T = \{n \,|\, n \in \mathcal{V} \wedge n$ is incident to an edge $e \in \mathcal{E}_T\}$
  end if
  Update the parameters corresponding to the subgraph $\mathcal{C}$ ▷ **Circuit fine-tuning**
  $i = i + 1$
end for

B.2 Training stability

To demonstrate the stability of circuit-tuning, we provide a simple proof of the training convergence. Specifically, we consider the model as a continuously differentiable function $f(\theta)$. Let $\theta \in \mathbb{R}^D = (\theta_1, \theta_2, ..., \theta_D)$, and suppose that f satisfies the conditions for Lipschitz gradient continuity:

$$||\nabla_\theta f(\theta') - \nabla_\theta f(\theta)|| \leq L||\theta' - \theta||$$

where L is a Lipschitz constant.

If we mask out a set of parameters $\theta_{mask} \subset \theta$, then we have

$$\begin{aligned} &||\nabla_{\theta \backslash \theta_{mask}} f(\theta') - \nabla_{\theta \backslash \theta_{mask}} f(\theta)|| \\ <&||\nabla_{\theta} f(\theta') - \nabla_{\theta} f(\theta)|| \\ \leq& L||\theta' - \theta|| \end{aligned}$$

where $||\cdot||$ is the L2 norm of the vectors. This is because for each masked parameter $\theta_i \in \theta_{mask}$, the gradient of it equals to 0, which leads to a smaller norm for gradient change. Thus a smaller Lipschitz constant $L_{mask} < L$ serves as the upper bound for the change rate of gradients, which means the training process would be more stable compared with full fine-tuning. Note that we perform only a simplified analysis for the most ideal case, which may lack strict rigor. However, we regard this as an explanation for the training stability of circuit-tuning. Experimental results in Section 4.3 also demonstrate this.

C Details of the SVD experiments

C.1 Dataset construction

We use the first 10k samples from Pile (Gao et al., 2020), which consists of 22 smaller, high-quality datasets. Firstly, in order to get relatively simple and clean sentences, we filter out the content in Github, ArXiv, PubMed Abstracts, PubMed Central, StackExchange, USPTO Backgrounds, Pile-CC, DM Mathematics, and FreeLaw. Thus we do not include code or complex formulas in our data. Secondly, we split the corpus with periods '.' as intervals. We remove links to websites, images, and other files. We also remove sentences that are too short (less than 25 characters). Thirdly, we leave only the sentences in the present tense in English and ensure that each sentence is a complete sentence with a punctuation like '.', '?', or '!' at the end. Finally, for each verb in the present tense in a sentence, we convert it to its opposite form. That is, we convert a verb with a part of speech VBP like *go* to VBZ like *goes*, and vice versa. For *be (am / is / are)*, we flip them as follows: *am* $\rightarrow$ *is, is* $\rightarrow$ *are, are* $\rightarrow$ *am*.

The dataset is available in our code repository.

C.2 Experimental settings

C.2.1 Choice of N

Faithfulness and completeness examine a circuit from two different views. Faithfulness tells how much performance a circuit gets, while completeness tells how much performance a circuit fails to capture. Consider a model M with its computational graph $\mathcal{G}$, a circuit $\mathcal{C}$ for a specific task T and a metric $\mathcal{L}_m$ for measuring the output of the model, following the definition in (Marks et al., 2024), the faithfulness of the circuit $\mathcal{C}$ is

$$\frac{\mathcal{L}_m[M(\mathcal{C})] - \mathcal{L}_m[M(\varnothing)]}{\mathcal{L}_m[M(\mathcal{G})] - \mathcal{L}_m[M(\varnothing)]} \tag{2}$$

in which $M(*)$ denotes the forward pass of model M with the nodes outside $*$ mean-ablated, and $\varnothing$ denotes an empty circuit. The completeness is defined as

$$\frac{\mathcal{L}_m[M(\mathcal{G} \backslash \mathcal{C})] - \mathcal{L}_m[M(\varnothing)]}{\mathcal{L}_m[M(\mathcal{G})] - \mathcal{L}_m[M(\varnothing)]} \tag{3}$$

where $\mathcal{G} \backslash \mathcal{C}$ denotes the complementary set of circuit $\mathcal{C}$. The completeness of circuit $\mathcal{C}$ is actually the faithfulness of circuit $\mathcal{G} \backslash \mathcal{C}$.

In practice, we calculated the faithfulness and completeness on GPT-2-small for the SVA task. We randomly sampled a batch of 32 samples from the training set and calculated the two metrics for each value of N in [10, 15, 20, 50, 100, 500, 1000, 2500, 5000, 10000]. We plot the curves in Figure 6. It can be seen that $N = 1000$ is the elbow point. Therefore 1000 could be an ideal initialization of N in circuit-tuning.

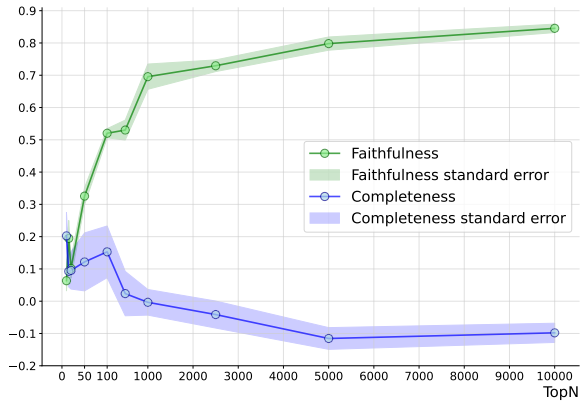


Figure 6: The change of faithfulness and completeness in the SVA task with the number of edges N. The elbow point of the two curves corresponding to the ideal initialization of $N(= 1000)$.

C.2.2 Choice of graph pruning methods

In this section, we detail the basic concepts of causal intervention and the circuit discovery methods used in the graph pruning stage of circuit-tuning. As described in Section 3.1, we view the model M as a computational graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$,

where $\mathcal{V}$ and $\mathcal{E}$ denote the sets of nodes and edges, respectively. A node n represents a model component, such as a neuron, an attention head, or an MLP layer, depending on the granularity of the analysis. An edge $e : n_1 \rightarrow n_2$ describes the information flow, specifically the path from the output of an upstream node n_1 to the input of a downstream node n_2.

We aim to extract a task-related circuit $\mathcal{C}$ by removing redundant structures. This requires calculating an importance score for each element in the computational graph to measure its contribution to the final output, followed by pruning based on these scores. The graph pruning employs causal intervention (Neuberg, 2003), which involves perturbing an element and quantifying the resulting change in output to calculate its attribution score, or *indirect effect* (Pearl, 2001). For convenience, we refer to these quantified contributions as *scores*, denoted by $c(\cdot)$. After computing scores for all targeted elements, we isolate the circuit by selecting the top-N nodes or edges with the highest scores. In circuit discovery, we typically design a metric $\mathcal{L}_m$ to quantify the output for a target task, such as the logit difference between v_{flip} and v_{orig} used in the SVD task (Section 4.1).

Common circuit discovery methods include activation patching, which aims to remove redundant nodes, and path patching, which targets redundant edges. For clarity, we begin our discussion with standard activation patching (Vig et al., 2020; Meng et al., 2022; Conmy et al., 2023).

Activation patching Given an input x^{clean} (e.g., a prompt), we aim to perturb a specific node n. The recommended approach is to construct a corrupted prompt x^{noise} with the same format as x^{clean} but with different or opposite predictive semantics. For instance, if x^{clean} is "The *kids* on the beach" and the task is verb prediction, x^{noise} could be "The *kid* on the beach". We first input x^{noise} into the model to obtain the corrupted activation $n(x^{\text{noise}})$ at node n. Then, we input x^{clean} and, upon reaching node n, apply the following intervention to compute the score for node n:

$$\begin{aligned} c(n) =& \mathcal{L}_m\big(M\big(x^{\text{clean}}|do(n \leftarrow n(x^{\text{noise}}))\big)\big) \\ & -\mathcal{L}_m(M(x^{\text{clean}})) \end{aligned} \tag{4}$$

Here, we use do-calculus notation (Neuberg, 2003) to represent the intervention process. We perform this intervention for each node $n \in \mathcal{V}$. Generally, we take the absolute value of the scores and select the top-N nodes along with their associated edges to form the circuit.

Notably, node scores can also be calculated over a batch $(x_1, \ldots, x_n)$ as:

$$c(n) = \mathbb{E}_{x_i \sim \mathcal{D}}[|c_i(n; x_i)|] \tag{5}$$

where $\mathcal{D}$ is the data distribution of the target task.

Attribution patching (AtP) Given $|\mathcal{V}|$ nodes in the computational graph, the time complexity of standard activation patching is $\mathcal{O}(|\mathcal{V}|)$. To reduce this complexity, (Nanda, 2023) proposed using a first-order Taylor expansion as a linear approximation of $c(n)$. Specifically, treating the corrupted value $n(x^{\text{noise}})$ as the independent variable and $c(n)$ as the dependent variable, we take the first-order Taylor expansion of $c(n)$ at $n(x^{\text{noise}}) = n(x^{\text{clean}})$:

$$\begin{aligned} & c(n) \\ \approx & \mathcal{L}_m(M(x^{\text{clean}})) + [n(x^{\text{noise}}) - n(x^{\text{clean}})]^\top \cdot \\ & \nabla_n \mathcal{L}_m(M(x^{\text{clean}}))|_{n=n(x^{\text{clean}})} - \mathcal{L}_m(M(x^{\text{clean}})) \\ = & [n(x^{\text{noise}}) - n(x^{\text{clean}})]^\top \cdot \\ & \nabla_n \mathcal{L}_m(M(x^{\text{clean}}))|_{n=n(x^{\text{clean}})} \end{aligned} \tag{6}$$

This allows us to compute all node scores simultaneously using only one forward pass on x^{noise} and one forward and backward pass on x^{clean}, reducing the time complexity to $\mathcal{O}(1)$.

Path patching Similar to activation patching, path patching focuses on the relationship between two nodes, represented as an edge. Specifically, the residual stream in Transformers allows the output of an upstream node n_1 to be directly included in the input of a downstream node n_2. In this context, we aim to measure the *direct effect* of n_1 on n_2 (Neuberg, 2003). This requires intervening on the upstream node n_1 while keeping all other intermediate nodes between n_1 and n_2 constant, then observing how the resulting change in n_2 affects the final output.

Formally, when we intervene on the upstream node n_1, its contribution to the residual stream it writes to becomes:

$$r_{out}^{n_1}(x^{\text{clean}}|do(n_1 \leftarrow n_1(x^{\text{noise}}))) \tag{7}$$

The residual stream carries this intervened contribution to the input r_{in} of the downstream node n_2. Consequently, the input to n_2 becomes:

$$\begin{aligned} & r_{in}(x^{\text{clean}}) - r_{out}^{n_1}(x^{\text{clean}}) + \\ & r_{out}^{n_1}(x^{\text{clean}}|do(n_1 \leftarrow n_1(x^{\text{noise}}))) \end{aligned} \tag{8}$$

As shown in Equation 8, the change in n_2 originates solely from n_1. Under this intervention, the value of n_2 is expressed as:

$$\begin{aligned} & n_2^{\text{direct}^{n_1}}(do(n_1 \leftarrow n_1(x^{\text{noise}}))) \\ =& n_2\Big(x^{\text{clean}}|do\big(n_1 \leftarrow n_1(x^{\text{noise}})\big), \\ & do\big(n_{(n_1 \to n_2)} \leftarrow n_{(n_1 \to n_2)}(x^{\text{clean}})\big)\Big) \end{aligned} \tag{9}$$

where $n_{(n_1 \to n_2)}$ denotes all intermediate nodes between n_1 and n_2. Equation 9 represents the process of perturbing only n_1 while keeping $n_{(n_1 \to n_2)}$ fixed.

Therefore, we can then define the score of the edge $e : n_1 \to n_2$ based on its impact on the final output:

$$\begin{aligned} c(e : n_1 \to n_2) = \mathcal{L}_m\Big(M\big(x^{\text{clean}}|do(n_2 \leftarrow \\ n_2^{\text{direct}^{n_1}}(do(n_1 \leftarrow n_1(x^{\text{noise}}))))\big)\Big) - \mathcal{L}_m(M(x^{\text{clean}})) \end{aligned} \tag{10}$$

Edge attribution patching (EAP) (Syed et al., 2024) applied the principles of attribution patching to path patching, simplifying the calculation of the edge score to:

$$\begin{aligned} c(e : n_1 \to n_2) \approx [n_1(x^{\text{noise}}) - n_1(x^{\text{clean}})]^\top \cdot \\ \nabla_{n_2}\mathcal{L}_m(M(x^{\text{clean}}))|_{n_2=n_2(x^{\text{clean}})} \end{aligned} \tag{11}$$

This reduces the time complexity to $\mathcal{O}(1)$.

EAP-IG (integrated gradients) If an activation of a node has a zero gradient, then it will not contribute to the attribution score. To deal with this issue, Hanna et al. (2024) proposed EAP-IG that cumulates the gradients along the straight-line path from the corrupted prompt x^{noise} to the clean prompt x^{clean} in m steps. Specifically, the IG-score is defined as:

$$\begin{aligned} c(e : n_1 \to n_2) \approx [n_1(x^{\text{noise}}) - n_1(x^{\text{clean}})]^\top \cdot \\ \frac{1}{m}\sum_{k=1}^{m} \nabla_{n_2}\mathcal{L}_m(M(x^{\text{clean}}))|_{n_2=n_2^{IG}} \end{aligned} \tag{12}$$

where

$$n_2^{IG} = n_2(x^{\text{noise}}) + \frac{k}{m}[n_2(x^{\text{clean}}) - n_2(x^{\text{noise}})] \tag{13}$$

This requires one forward pass on x^{noise}, and m forward and backward passes on x^{clean}. In terms of the computation on x^{clean}, the time complexity of EAP-IG is $\mathcal{O}(m)$. In practice, we follow Hanna et al. (2024) and set m to 5.

In practice, we find that EAP-IG provides a slight performance gain over EAP in circuit-tuning, though the improvement is marginal, as shown in Figure 7.

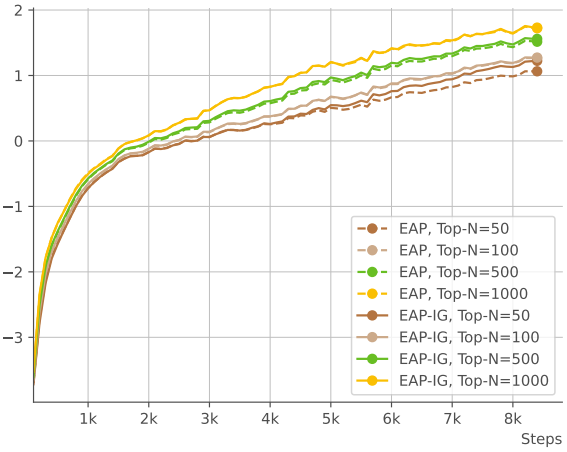


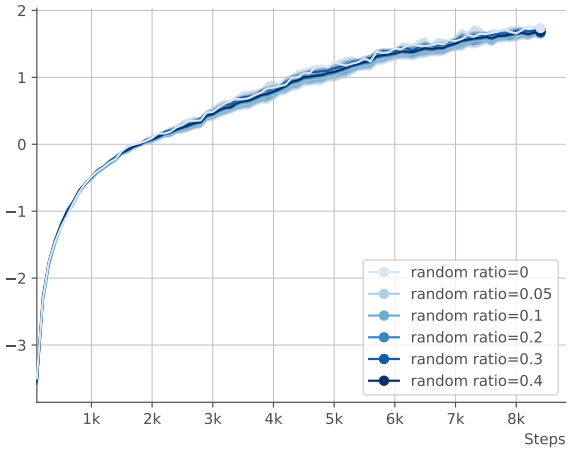
Figure 7: The logit difference in circuit-tuning with EAP and EAP-IG for graph pruning. We believe that circuit-tuning is insensitive to the choice of graph pruning.

Mean ablation This operation is to replace targeted activations with their dataset mean value when doing intervention (Heimersheim and Nanda, 2024). In our circuit-tuning experiments, we use the mean activation across all samples and token positions within a batch as the corrupted value for intervening at the END token position.

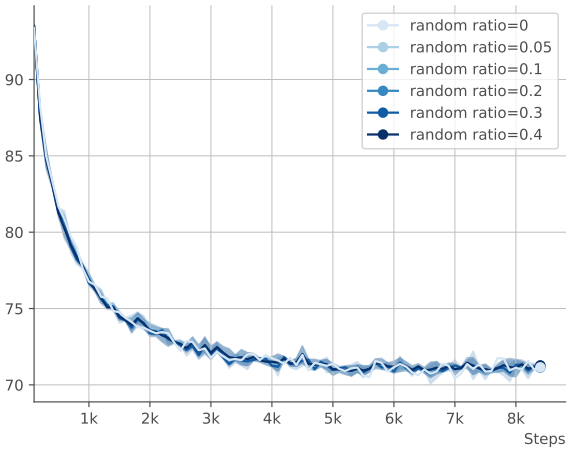
In our view, mean ablation serves as a random activation to replace the corrupted activation in standard causal intervention. We believe that regardless of how the corrupted activation is calculated, its ultimate goal is to find noise that disrupts the original activation. This is equivalent to randomly sampling an activation from the training data that approximates Gaussian noise at the current node, although the resulting signals are less precise and prominent than those derived from a carefully constructed corrupted prompt.

We have provided a "random activation" experiment in Section 4.5, in which we randomly unfreeze some nodes outside the circuit after each graph pruning step to involve some of the parameters outside the circuit in fine-tuning. As shown in Figure 8, this operation does not bring any performance gain, which means the circuit we find using mean ablation for intervention is relatively

complete: it ensures that no task-relevant nodes are omitted from the circuit.



(a) Logit difference



(b) PPL

Figure 8: We randomly unfreeze certain percentage of nodes outside the circuit after each graph pruning step. Results show that the PPL almost remains unchanged, and the logit difference even drops a little when introducing extra parameters outside the circuit. This demonstrates that the circuit found via mean ablation precisely gets the key components inside and leave the irrelevant components outside.

Gao et al. (2025) also claim that mean ablations leads to much more complete circuits than standard activation patching using pairs of clean and corrupted prompts (in Appendix E in their paper). The idea is that if there are parts of the circuit that are critical for completing the target task, but the activations of which differ across prompt pairs rather than within each pair, then activation patching will fail to notice it. That is to say, when we only focus on one prompt as the target task, then standard activation patching should be better. However, if we focus on a target task that has more than one prompts, then we need to take into consideration the overall data distribution of that task. Then mean ablation could be better at finding the common parts across samples.

C.2.3 Choice of K

As shown in Figures 9 and 10, we conduct comparative experiments by setting N to 500 and 1000 and varying K from 1 to 128 to investigate the impact of K. As discussed in Section 4.5, circuit-tuning is insensitive to the choice of K within a certain range ($K < 128$). However, performance declines when $K > 128$. This underscores the dynamic nature of our approach: the circuit must adaptively evolve from the task-related components of the base model during fine-tuning to better align with the data distribution of the new task.

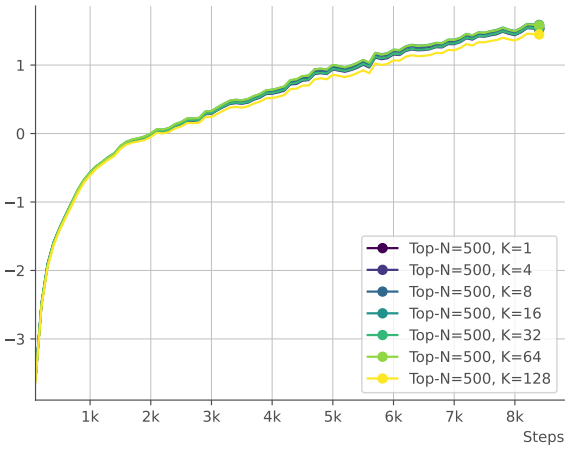


Figure 9: The change in logit difference with difference choices of K when top-N=500.

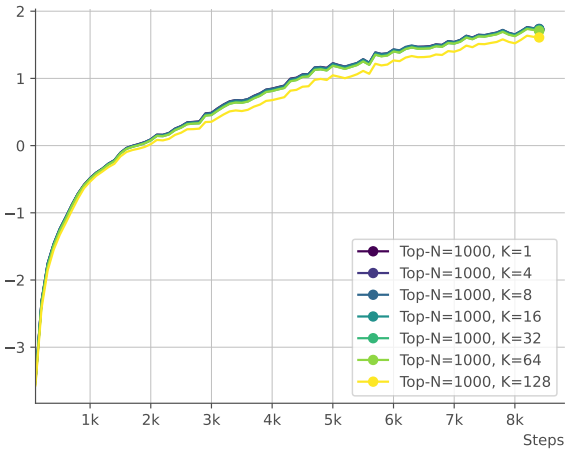


Figure 10: The change in logit difference with difference choices of K when top-N=1000.

Table 2: The settings of the nodes and their corresponding parameters in the subject-verb disagreement task. The number of layers $L = 12$ and the number of attention heads in each layer $H = 12$. The notations for parameters are detailed in Appendix A.

Nodes	Upstream		Downstream
	$\text{Attn}^{i,j}(x)$	$\text{MLP}^{i}(x)$	$x^{i,j}_{Q/K/V}$ x^{i}_{in}
Parameters	$W^{i,j}_{O}$	W^{i}_{out}	$W^{i,j}_{Q/K/V} W^{i}_{in}$
Range	$i \in [0, L), j \in [0, H)$		

C.3 Additional results and discussions

C.3.1 Training details

We use GPT-2-small (Radford et al., 2019) for this task. GPT-2-small is a decoder-only transformer with 12 layers and 12 attention heads per attention layer. We set the output of the attention and the MLP in each layer as upstream nodes, and the input of the query, key, value, and the MLP in each layer as downstream nodes. This is because the query, key, and value input for an attention head can only affect downstream nodes via the attention output of that head, so the upstream nodes can only be attention head outputs, which is also discussed in Kramár et al. (2024). The parameters to update correspond to the upstream and downstream nodes are shown in Table 2.

C.3.2 Dicussions about intrinsic dimension

In Section 4.3, we noted that model performance saturates and the tunable parameter ratio also stabilizes as N approaches 1000, indicating that most task-relevant parameters are already included. We connect this finding to the work on intrinsic dimension by Li et al. (2018). Specifically, they optimized for specific tasks within a parameter subspace and observed performance changes while increasing the subspace dimension from zero. Their experiments on training fully connected and convolutional networks (via subspace optimization) on the MNIST dataset revealed that performance reaches 90% of the baseline once the subspace dimension reaches a value far smaller than the total number of parameters.

In our study, $N = 1000$ can be regarded as the intrinsic dimension within the computational graph from the perspective of edges. The circuit at $N = 1000$ represents the minimal subgraph capable of completing the target task within an acceptable loss threshold. This is further reflected in the faithfulness and completeness metrics of the SVA task calculated before training started, as shown in Figure 6. These results further demonstrate that SVA and SVD tasks share a substantial portion of model parameters, i.e., fine-tuning does not "start from scratch" but rather makes minimal possible adjustments to the existing parameters, a point also discussed in prior literature (Aghajanyan et al., 2021; Jain et al., 2023; Wang et al., 2025b).

Additionally, we plotted the curves of the logit difference and average parameter ratio against N, as shown in Figure 11 and 12. The average parameter ratio represents the mean ratio of tunable parameters during training. Specifically, Every time after graph pruning, we recorded the number of the parameters n_i in the circuit, where i the count for graph pruning. After training, we calculated the average parameter ratio as $\frac{1}{M}\sum_{i=1}^{M}\frac{n_i}{N_p}$, where M is the total number of the times for graph pruning, and N_p is the total number of the parameters in the model.

As illustrated in Figure 11, the growth of the logit difference slows down significantly when $N \geq 1000$. In Figure 12, the inflection point of the average parameter ratio curve appears relatively smooth. We attribute this to the node granularity used in our experiments. Treating attention heads and MLP layers as the minimal nodes may limit the precision in capturing the exact inflection point of the parameter ratio, yet this does not undermine the overall experimental conclusions.

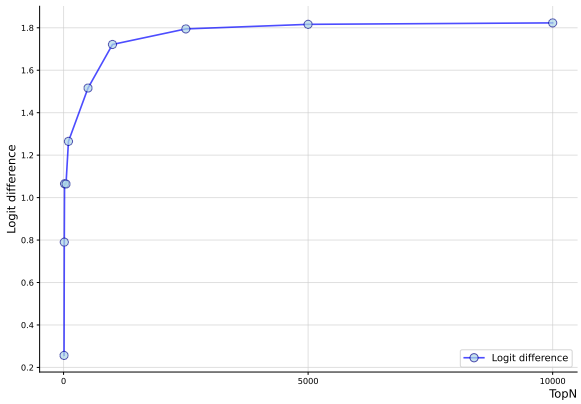


Figure 11: The relationship between the final logit difference and the value of N. The elbow point in the curve implies the minimal of the circuit for the target task.

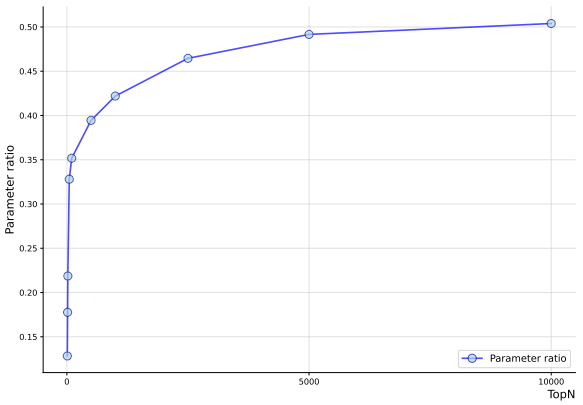


Figure 12: The relationship between the average parameter ratio and the value of N. The elbow point in the curve implies the minimal of the circuit for the target task.

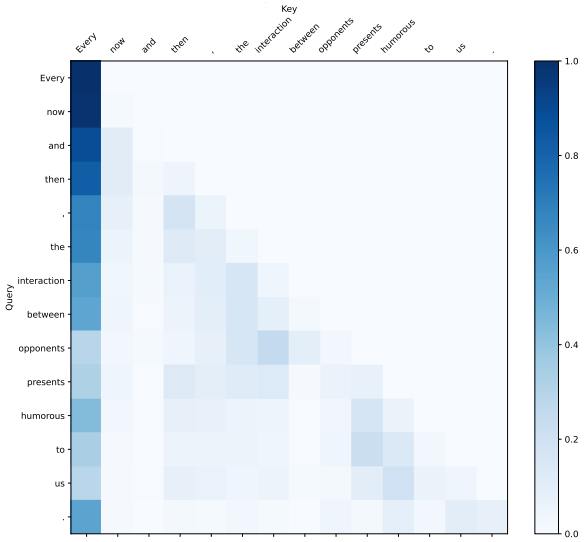


Figure 13: The attention pattern of head.8.5 demonstrates that the head could locate the subject in a sentence. Example sentence in the figure: "Every now and then, the *interaction* between opponents *presents* humorous to us."

C.3.3 The attention heads for subject identification

In Finding 3, we first identify the attention heads responsible for subject recognition by observing the attention patterns of each head at the END token. We specifically selected Type II sentences from the SVD dataset. This choice stems from the fact that most heads direct their maximum attention to the current token, resulting in a diagonal attention pattern. For Type I sentences, the subject is located at the current token (the END token). Thus, a head attending to the subject might merely be a coincidence. Consequently, we chose Type II sentences, where the subject precedes the END token.As shown in Figure 13, 14 and 15, we found that in heads such as head.8.5, 10.9, and 11.8, the attention from the END token to the subject token is significantly prominent. This indicates that these heads possess subject-localization capabilities. This reflect the capabilities of the W_Q and W_K within the attention module.

C.3.4 Dicussions about the evolution of functional pathways

In Finding 4, we first calculate the Pearson correlation coefficient between edge scores and training steps. We find that the scores of numerous edges correlate positively or negatively with the steps, indicating that the strength of these edges either intensifies or diminishes during fine-tuning. Furthermore, we observe that a significant portion of the downstream nodes for the intensifying edges are the key attention heads identified in Finding 3. Consequently, we hypothesize that the upstream nodes of these strengthened edges exert a causal influence on these key heads.

To verify this, we set the target of our patching to the attention pattern of the key heads. For instance, focusing on the downstream node head.8.5, we intervene on its preceding nodes to observe changes in the attention score from the END token to the subject. As visualized in Figure 16a, nodes such as head.0.8, head.1.3, head.2.10, and head.6.5 exert a substantial influence on head.8.5. Similarly, in Figure 16b, we observe that head.0.8, head.0.9, head.1.3, head.1.4, head.1.6, and head.6.5 significantly affect head.10.9.

Next, we select edges connecting these influential upstream heads to the key downstream nodes and monitor their scores across training steps. In Figure 17a, we find a clear positive correlation between these edge scores and the number of steps. This suggests that causal relationships between task-relevant nodes are continuously strengthened during fine-tuning. Conversely, relationships between task-irrelevant nodes (relative to the target task performance) gradually weaken, as shown in Figure 17b.

D Details of the gender debiasing task

D.1 Task definition and data preparation

According to Gallegos et al. (2024), there are various kinds of expressions in social bias. In this study, we focus on the gender bias in occupations.

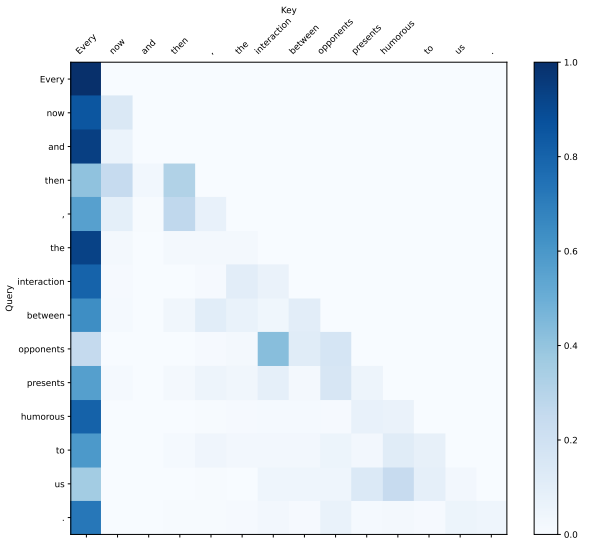


Figure 14: The attention pattern of head.10.9 demonstrates that the head could locate the subject in a sentence. Example sentence in the figure: "Every now and then, the *interaction* between opponents *presents* humorous to us."

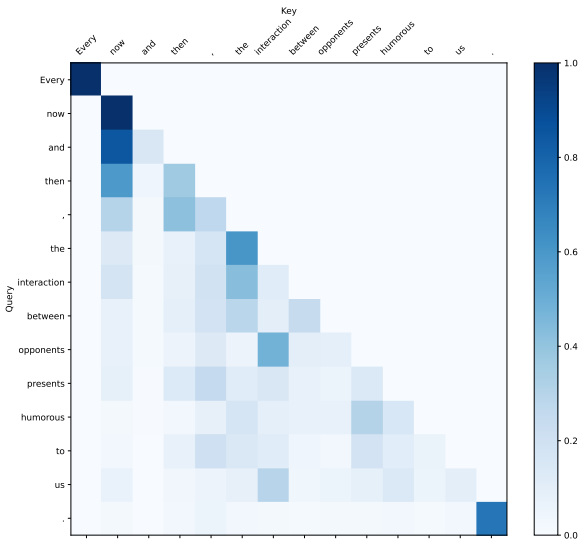


Figure 15: The attention pattern of head.11.8 demonstrates that the head could locate the subject in a sentence. Example sentence in the figure: "Every now and then, the *interaction* between opponents *presents* humorous to us."

We aim to break down the binary gender stereotype of a model. For example, given a sentence "the doctor put on [PRP] coat" where [PRP] is a possessive pronoun, we expect the model to choose *his* or *her* with equal probabilities.

During fine-tuning, the model learns to predict the next word in an auto-regressive way, thus we expect the model to balance the probabilities between male attribute words (he/his/him/himself) and female attribute words (she/her/herself) at the END token when generating the next token. Therefore, we use the logit difference between the male attribute words and female attribute words at the END token as the metric $\mathcal{L}_m$ for measuring the output of the model. Specifically, for each sample, we calculate the logit difference between the pronoun and the anti-pronoun, which is the pronoun in the opposite gender. For example, in the case "the doctor put on [PRP] coat", the logit difference is:

$$\mathrm{LD}(\mathrm{her}, \mathrm{his}) = \mathrm{Logit}(\mathrm{her}) - \mathrm{Logit}(\mathrm{his}) \quad (14)$$

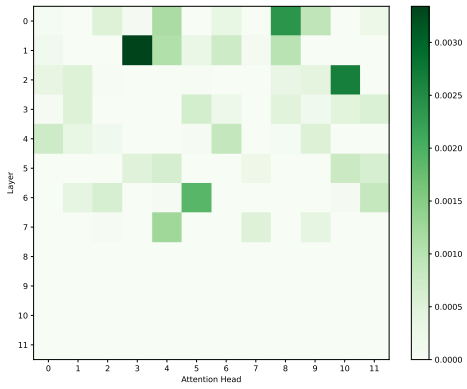
where $\mathrm{Logit}(\cdot)$ is the logit of a token at the END token *on*.

We use BUG (Levy et al., 2021) for training, which is a large-scale dataset of sentences sampled from real-world corpora. Each sentence is marked with an occupation and the pronouns referring to it. In practice, we use the "balanced BUG" provided in the dataset which includes 2.5w sentences randomly sampled from BUG to ensure balance between male and female entities and between stereotypical and non-stereotypical gender role assignments. We perform coreference resolution ourselves to filter out the samples in which the coreference is not right, and leave 1.5w samples for training, which contains 151 types of occupations and each sentence only contains one (occupation, pronoun) pair. Data processing details can be found in our code.

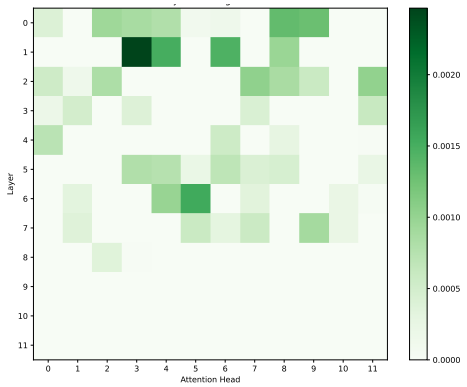
Table 3: The 40 occupations in the WinoBias dataset.

Male (biased)	**Female (biased)**	**Male (biased)**	**Female (biased)**
Driver	Attendant	Manager	Counselor
Supervisor	Cashier	Lawyer	Designer
Janitor	Teacher	Farmer	Hairdresser
Cook	Nurse	Salesperson	Writer
Mover	Assistant	Physician	Housekeeper
Laborer	Secretary	Guard	Baker
Construction	Auditor	Analyst	Accountant
Chief	Cleaner	Mechanic	Editor
Developer	Receptionist	Sheriff	Librarian
Carpenter	Clerk	CEO	Tailor

For evaluation, we use WinoBias (Zhao et al., 2018) which is a classic dataset for coreference resolution focused on gender bias. There are two types of sentences in WinoBias. The Type 1 sentences require world knowledge related to the context to perform coreference resolution, e.g. "The farmer knows [the editor] because [he] is really famous". The Type 2 sentences can be resolved using syntactic information, e.g. "The CEO called [the hairdresser] and paid [her] over the phone".



(a) The upstream nodes that affect the functionality of head.8.5.



(b) The upstream nodes that affect the functionality of head.10.9.

Figure 16: We intervene on the preceding nodes of a key attention head in Finding 3, and observe the changes in the attention score from the END token to the subject in the key attention head.

In practice, we only use the Type 2 sentences to avoid ambiguity. The test set contains 40 types of occupations in total, which are list in Table 3.

To better evaluate the performance of gender debiasing, we adopt the metric of discrimination risk from (Liu et al., 2024) which is used to measure the stereotype in large language models. Specifically, given an occupation $x \in X$, a binary gender attribute $y \in \{\text{male}, \text{female}\}$ and a context $c \in C$, the stereotype of a model M against x about y in the context c is:

$$s^M_{y|x}(c) = \frac{p^M_{y|x}(c)}{p^*_{y|x}(c)} - 1 \tag{15}$$

where $p^*_{y|x}(c)$is the attribute prediction probability of the unbiased model, thus $p^*_{y|x}(c) = 0.5$ when binary gender is considered. The definition of the overall discrimination risk is:

$$R = \mathbb{E}_{x \sim X}(r_x) \tag{16}$$

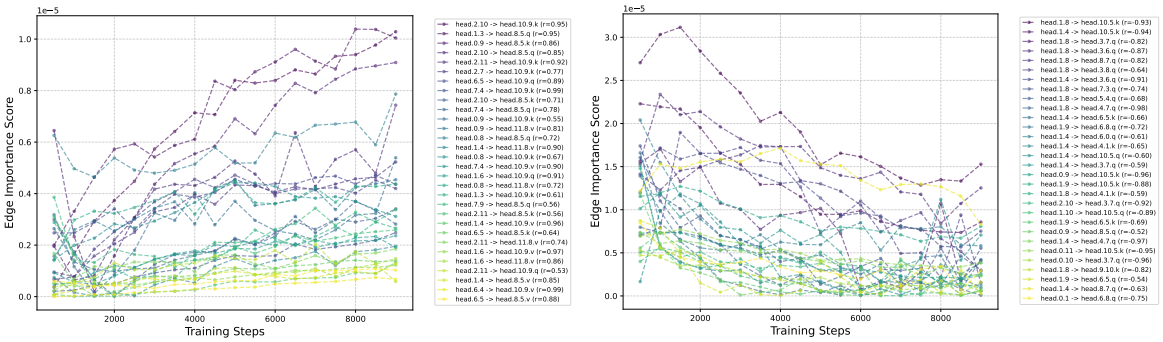
where $r_x = \mathbb{E}_{c \sim C}(J(s^M_{y|x}(c)))$ is the discrimination risk of one occupation x, and $J(s^M_{y|x}(c)) = \max_{y \in Y}\{max\{s^M_{y|x}(c), 0\}\}$ is called the discrimination risk criterion. For details, please refer to Liu et al. (2024).

D.2 Training and Evaluation details

We use Llama-3.2-1B-Instruct, Llama-3.2-3B-Instruct and Llama-3.1-8B-Instruct (Dubey et al., 2024) for this task. We set the output of the attention and the MLP in each layer as upstream nodes, and the input of the query, key, value, and the MLP in each layer as downstream nodes. Different from GPT-2-small, an MLP layer in Llama is too big to be a node, so we split the input and output of each MLP layer into 64-dimensional "MLP heads", as shown in Table 4. We do this to make an MLP layer into finer nodes. However, the MLP heads are quite different from attention heads. Attention heads in Transformers operate independently, whereas MLP heads do not. An MLP head could be viewed as a large MLP neuron.

The batch size is set to 32 in all experiments. For all fine-tuning methods (full fine-tuning, LoRA and circuit-tuning), we tried our best to search for the best hyper-parameters. We set the learning rate to 1e-5 for full fine-tuning, and 5e-5 for circuit-tuning and LoRA. For circuit-tuning, we set $K = 8$ and $N = 5000/8000/15000$ for Llama-3.2-1B / 3B and Llama-3.1-8B, respectively. For LoRA, we set $r = 32, \alpha = 32$, and the modules to fine-tune include [`q_proj`, `k_proj`, `v_proj`, `o_proj`, `up_proj`, `gate_proj`, `down_proj`]. Mini-batch SGD with a momentum equal to 0.9 is used as the optimizer. We train each model for 1 epoch.

After training, we compute the discrimination risk of the base model and each fine-tuned model. We find that circuit-tuning could achieve an overall discrimination risk even lower than full fine-tuning. For example, the discrimination risk of Llama-3.1-8B is 0.554. For circuit-tuning, the result is reduced to 0.447, while full fine-tuning gets a result of 0.477. As for LoRA, it only gets 0.489. To better demonstrate this, we fit the evaluation results to a normal distribution of occupation count over the discrimination risk, as shown in Figure 18. This indicates that circuit-tuning could achieve even better fine-tuning results than full fine-tuning and LoRA. By actively looking for the key components responsible for the target task, circuit-tuning could concentrate the optimization on the task-relevant



(a) The strengthened edges during circuit-tuning.

(b) The weakened edges during circuit-tuning.

Figure 17: We provide a stage-wise analysis on the scores of the edges in the computational graph during circuit-tuning. As discussed in Finding 4, the strengthened edges (with increasing scores) often correspond to the key components and the components that have a causal relationship with them. The weakened edges (with decreasing scores) often correspond to the (SVD) task-irrelevant model components.

Table 4: The settings of the nodes and their corresponding parameters in Llama3. For model sizes 1B/3B/8B, the number of layers $L = 16/28/32$, and the number of attention heads in each layer $H = 32/24/32$, and the number of MLP heads in each layer $H^* = 128/128/224$. The notations for parameters are detailed in Appendix A.

Nodes	Upstream		Downstream		
	$\text{Attn}^{i,j}(x)$	$\text{MLP}^{i,k}(x)$	$x^{i,j}_{Q/K/V}$	$x^{i,k}_{pre}$	$x^{i,k}_{in}$
Parameters	$W^{i,j}_O$	$W^{i,k}_{out}$	$W^{i,j}_{Q/K/V}$	$W^{i,k}_{gate}$	$W^{i,k}_{in}$
Range	$i \in [0, L), j \in [0, H), k \in [0, H^*)$				

parameters, instead of involving all the parameters into fine-tuning.

To further investigate the parameter update, we provide a similar analysis as in Finding 1. The parameter differences in W_Q, W_K, W_V, W_O between the base model (Llama-3.2-1B) and the model fine-tuned with LoRA / circuit-tuning are shown in Figure 21. We visualize the distribution of the parameter difference between the base model and the model fine-tuned with circuit-tuning / LoRA in Figure 19 and 20. It is obvious that the mode of the parameter distribution for circuit-tuning is closer to zero than LoRA, demonstrating its superior sparsity in parameter update. This also allows for more faithful model diffing analysis with less model parameters to study.

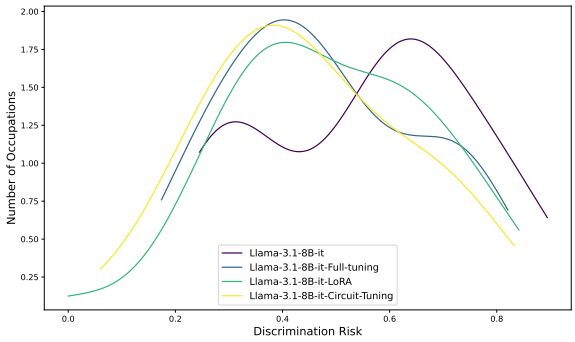


Figure 18: The distribution of the number of occupations with respect to the prejudice risk for Llama-3.1-8B-instruction and its fine-tuned versions with different fine-tuning methods. We fit the evaluation results to a normal distribution using Gaussian density estimation, as Liu et al. (2024) do.

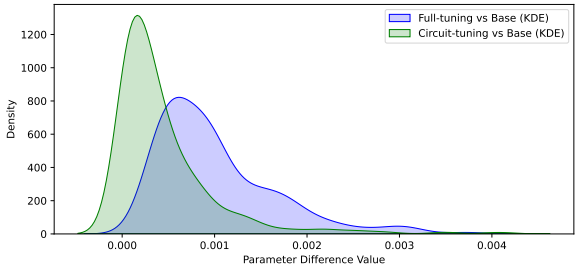


Figure 19: The distribution of the parameter difference in W_O between the base model (Llama-3.2-1B) and the model fine-tuned with LoRA / circuit-tuning.

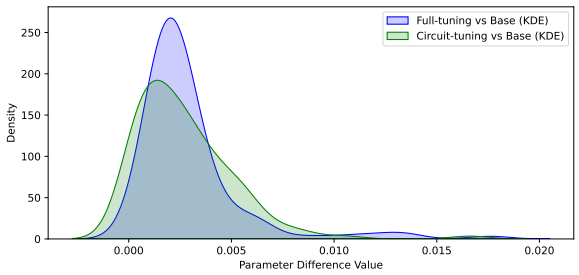


Figure 20: The distribution of the parameter difference in W_V between the base model (Llama-3.2-1B) and the model fine-tuned with LoRA / circuit-tuning..

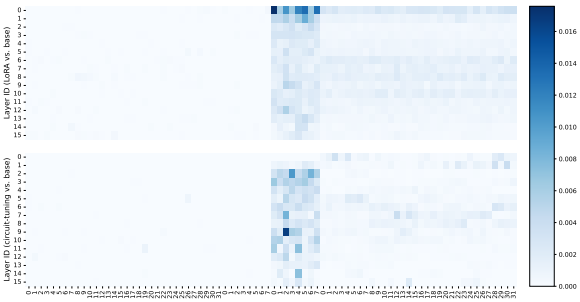


Figure 21: The parameter difference in W_Q, W_K, W_V, W_O (from left to right in X-axis) between the base model (Llama-3.2-1B) and the model fine-tuned with LoRA (top) / circuit-tuning (bottom). The distributions of the parameter difference in W_O and W_V are visualized in Figure 19 and 20 via KDE, respectively. Compared with LoRA, circuit-tuning could achieve superior sparsity: instead of affecting all the parameters, it dynamically selects the task-relevant parameters for fine-tuning.