Mechanistic Insights: Circuit Transformations Across Input and Fine-Tuning Landscapes

Anonymous authors
Paper under double-blind review

Abstract

Mechanistic interpretability seeks to uncover the internal mechanisms of Large Language Models (LLMs) by identifying circuits—subgraphs in the model's computational graph that correspond to specific behaviors—while ensuring sparsity and maintaining task performance. Although automated methods have made massive circuit discovery feasible, determining the functionalities of circuit components still requires manual effort, limiting scalability and efficiency. To address this, we propose a novel framework that accelerates circuit discovery and analysis. Building on methods like edge pruning, our framework introduces circuit selection, comparison, attention grouping, and logit clustering to investigate the intended functionalities of circuit components. By focusing on what components aim to achieve, rather than their direct causal effects, this framework streamlines the process of understanding interpretability, reduces manual labor, and scales the analysis of model behaviors across various tasks. Inspired by observing circuit variations when models are fine-tuned or prompts are tweaked (while maintaining the same task type), we apply our framework to explore these variations across four PEFT methods and full fine-tuning on two well-known tasks. Our results suggest that while fine-tuning generally preserves the structure of the mechanism for solving tasks, individual circuit components may not retain their original intended functionalities.

1 Introduction

The rapid advancement of large language models (LLMs) has established them as powerful tools across a wide range of tasks for (Vaswani, 2017; Devlin, 2018; Achiam et al., 2023) natural language and beyond. However, their vast scale—often involving billions of parameters—and black-box nature pose significant challenges in understanding their decision-making processes, leading to unpredictable risks in critical domains where accuracy, fairness, and transparency are essential. The field of interpretability aims to address these challenges by developing methods to uncover the internal reasoning of these models. Mechanistic interpretability, a subfield within this area, seeks to reverse-engineer LLMs into human-understandable algorithms (Conmy et al., 2023; Bereska & Gavves, 2024). A key objective is to identify **circuits**, subgraphs within the model's computational graph that correspond to specific behaviors, while maintaining a high level of sparsity and preserving the model's performance on the given task.

Recent progress in mechanistic interpretability has shed light on the inner workings of language models through meticulous human inspections, uncovering key circuits responsible for specific tasks (Wang et al., 2022; Hanna et al., 2024; Merullo et al., 2023). To accelerate this process and reduce reliance on costly manual efforts, automated tools have been developed to systematically identify circuits driving certain behaviors (Conmy et al., 2023; Bhaskar et al., 2024; Syed et al., 2023). However, these methods are either time-consuming (Conmy et al., 2023) or rely on approximations that prioritize speed over reliability (Syed et al., 2023). As a result, further investigations into the relationships between circuits (Tigges et al., 2024; Prakash et al., 2024; Jain et al., 2024) are limited in either scale or precision in finding circuits. The recent introduction of techniques like Edge Pruning (Bhaskar et al., 2024) has made it possible to automatically identify more accurate and faithful circuits across larger datasets while keeping computational costs manageable.

Despite advancements in circuit discovery, significant challenges remain in understanding the functionalities of the circuit components. Existing studies often assign specific functions to individual nodes within circuits through manual inspection. These nodes and their roles are typically identified during the discovery process by observing changes in the model's output (e.g., variations in logit values) when specific node activations are perturbed. This approach generally requires extensive path interventions to determine the direct effects of nodes on the final output and careful design of such intervention. The process demands considerable manual effort, leading to a limited scope (e.g. only focus on the change in target output logits) and placing a heavy burden on researchers, ultimately slowing down the progress of Mechanistic Interpretability. Moreover, as circuit discovery methods become automated, the identification of circuits and their functionalities no longer happen in parallel. While progress has been made in circuit identification, a growing gap persists in understanding the broader implications ***—after circuits are identified, what coming next?*** With automated methodologies generating vast amounts of circuits, investigating their internal functionalities has become increasingly laborious and challenging.

This challenge is evident in our observations from the IOI task. Modifying objects within the task results in significant changes to the discovered circuits, as shown in Fig. 1. Intuitively, circuits responsible for the same reasoning across similar tasks should remain consistent. However, the pre-trained model struggles to maintain this consistency when performing identical tasks across various types of objects (e.g., changing "Mary" to "Dog"). Moreover, applying different supervised fine-tuning methods introduces further variations in the circuits. However, investigating these changes in circuits and their functionalities becomes increasingly demanding and labor-intensive.

Inspired by these challenges and building on these findings, this work explores how circuits vary in both structure and functionality across different ablated prompts, and how model fine-tuning methods affect these circuits. To address this, we propose a framework for circuit analysis that integrates **efficient discovery** with **automated interpretability**. To broaden the scope of functionality exploration and accelerate the process, we make a trade-off by foregoing functionality conclusions based on causal effects. Instead, we integrated the attention pattern and logit lens with clustering techniques to describe the **intended functionalities** of the circuits components. By 'intended functionalities', we refer to what the circuit components are attempting to do in the context of a given tasks, rather than what they ultimately contribute to the final output. Our framework extends the edge pruning algorithm (Bhaskar et al., 2024) for circuit discovery by incorporating stages for **circuit selection**, **circuit comparison**, **attention grouping**, and **logit clustering**, as outlined in Fig. 2. We further summarize our findings and contributions as following: (1). We introduce a novel framework that accelerates the process of estimating the functionality of circuit components, reducing the reliance on manual efforts. (2). We demonstrate that the functionality of circuit components does not necessarily remain consistent across ablated prompts. (3). We show that while fine-tuning methods maintain the overall functional structure, the specific components performing those functions may change.

2 Experiment Setup and Pipeline

In this section, we first outline the experimental setup, providing an overview of the prompt settings for the tasks and fine-tuning methodologies involved in our work. Next, we describe the process for identifying and selecting circuits, along with the metrics used for circuit validations. We then introduce the methods for evaluating the intended functionality of the circuit components. Lastly, we discuss how comparative circuit analysis is conducted, following the pipeline in Fig. 2.

2.1 Tasks and Models

We explore circuit variations within the context of two specific tasks: Indirect Object Identification (IOI) and Greater Than (GT). These tasks were initially examined by Wang et al. and Hanna et al., respectively. We focused on IOI and GT due to their well-established, human-inspected circuits, making them particularly suitable for implementing our framework and conducting an in-depth analysis of circuit variations through different fine-tuning methods and ablated prompts. Additionally, recent work by Merullo et al. offers promising insights through the study of these fundamental tasks. Building on top of these tasks, Merullo et al. has studied the transferrability of these circuits beyond the base syntactic structure of these tasks. To gain a more granular understanding of the

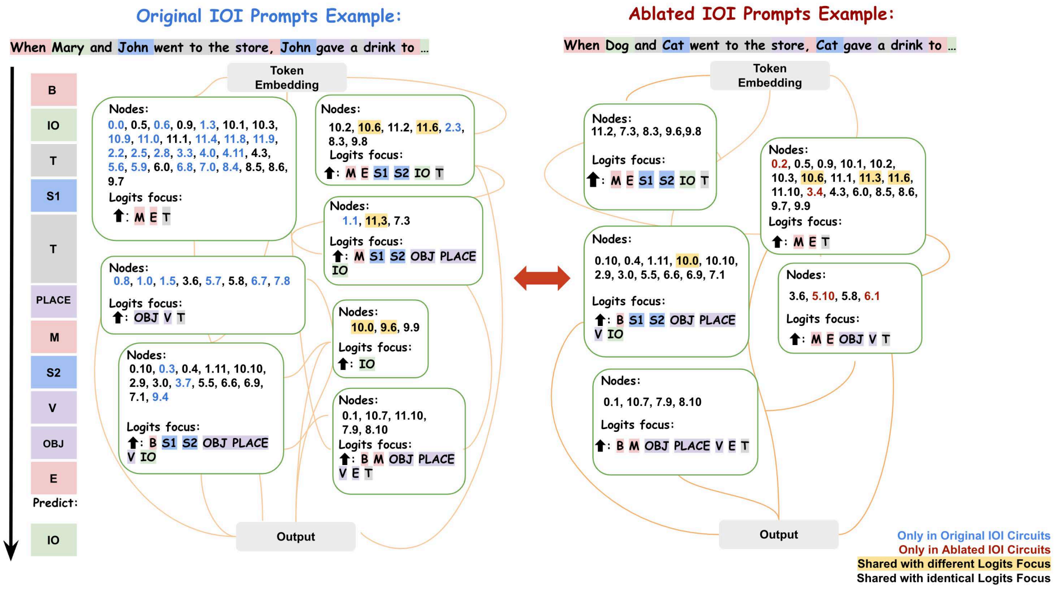

Figure 1: We demonstrate how circuits vary between IOI prompts and ablated prompts. When human names are replaced with animals, the newly discovered circuits become almost a subset of the IOI circuits. Nodes are clustered based on their logit outputs. We focus on logit outputs regarding to the prompt components. For each cluster, the major up-weights on logits for the sentence components are listed. Upon further analysis, we found that even overlapping nodes can perform different functions, colored in yellow.

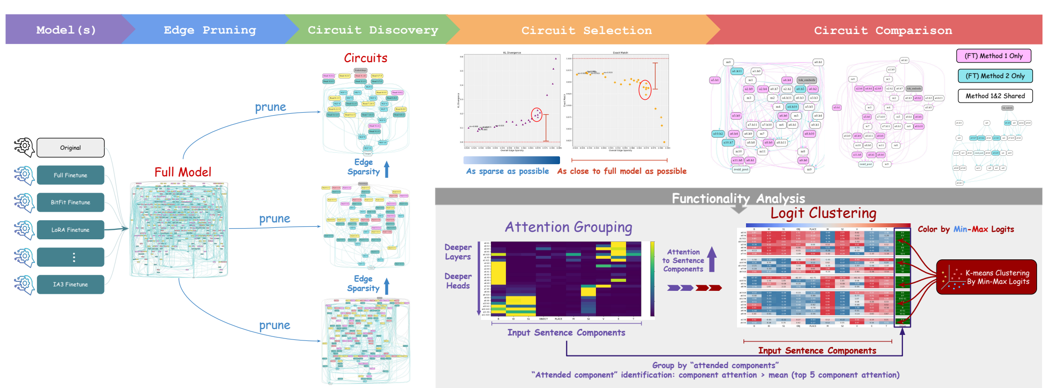

Figure 2: Pipeline of experiments: (1) Fine-tuned models on certain tasks; control model being the original GPT-2 model (2) Apply Edge-pruning to identify a series of candidate circuits (3) Select ideal circuits that balance sparsity and performance recovery, evaluated using KL Divergence and Exact Match metrics. (4) Compare circuits across models fine-tuned with different methods. (5) Conduct functionality analysis by: (5a). Grouping the heads based on their primary attended input sentence components - what the head is focusing on. (5b). Computing logits for each input sentence component per head - which components the head up-weights or down-weights for generation. (5c). Clustering heads by their min-maxed logits across input sentence components - which heads share or differ in functionality.

model's mechanisms, we focus on GPT-2 small, as its resulted circuits are more manageable and human-readable. More details are discussed as follows:

Indirect Object Identification (IOI): IOI (Wang et al., 2022) is the task of predicting the name of the indirect object in a sentence. The task follows the format of "When $\{A\}$ *and* $\{B\}$ *went to* $\{PLACE\}$*,* $\{B\}$ *bought a* $\{OBJECT\}$ *to* $\rightarrow \{A\}$". The datasets are generated by filling in A, B, PLACE, and OBJECT with a list of nouns. The model then completes the sentence by filling in the predicted indirect object, ideally {A}. Additionally, the prompts include variations in both ABBA and BABA formats. In the original IOI settings, A and B are filled by human names such as "John" and "Mary" as shown in Fig. 1. Following the setting by Edge Pruning (Bhaskar et al., 2024), we adopt the prompts on a variant with 30 templates from Hugging Face. Similarly, we randomly select 200 examples each for the train and validations; and 36,084 instances for the test sets.

Ablated IOI: In addition to the IOI dataset, we introduce an ablated IOI dataset to investigate whether the functionality and structure within the circuits are preserved across different types of objects, inspired by the observations in Fig. 1. We retain the template structure from the original IOI tasks; however, instead of populating A and B with human names, we use capitalized names of animals, cities, and colors. An example of the ablated prompt can be found in Fig. 1. To avoid tokenization issues that could affect model performance, we only include names that map to a single token. All other settings, such as train, test, and validation splits, remain the same as in the original IOI tasks.

Greater Than (GT): GT tasks (Hanna et al., 2024) follows the format of "*The war lasted from the year 1743 to 17* $\rightarrow$ *xy*." The tasks requires the model to place a higher probability on the continuations 44, 45, ..., 99, compared to 00, 01, ..., 42. We adopted the version of dataset proposed in Edge Pruning (Bhaskar et al., 2024) which has 150 examples in the train and valitation, and 12,240 instances in the test. This task is generally considered simpler than IOI, as it involves fewer logical steps to complete.

PEFT Methods: We primarily focus on Parameter-Efficient Fine-Tuning (PEFT) methods that preserve the original model structure. Specifically, we fine-tuned the GPT-2 small model with supervision, using Bitfit (Zaken et al., 2021), LoRA (Hu et al., 2021), IA3 (Liu et al., 2022), and AdaLoRA (Zhang et al., 2023), on the IOI and GT tasks. A fully fine-tuned model was used as a control for comparison. To ensure a fair comparison, all fine-tuned models are controlled to achieve nearly identical performances on the same test set. For the IOI task, the goal was to minimize the prediction loss for the indirect object. For the GT task, the goal was to maximize the gap between the cumulative probabilities of correct and incorrect years.

2.2 Circuit Discovery and Selection

We adopted edge-pruning (Bhaskar et al., 2024) to automate the circuits discovery step for each finetuned models along with the pretrained version across the above mentioned tasks. To evaluate and ensure the faithfulness and preciseness of the identified circuits, we mainly relied on the measure of KL-Divergence and Exact Match, adopted by Bhaskar et al. and Conmy et al..

Circuit Discovery Algorithm: We implemented the edge pruning algorithm for automated circuit discovery (Bhaskar et al., 2024). This method addresses circuit discovery through gradient-based pruning on the edges of the model's computational graph over hyperparameter *edge-sparsity (es)*. Similar to path-patching (Wang et al., 2022) and the other automated methodologies (Conmy et al., 2023; Syed et al., 2023), it involves causal interventions on edges by substituting the target edges with counterfactual activations from corrupted examples. This process further generates sparse circuits by masking out edges that has no causal effect on the target tasks.

In this work, we first reproduced the circuits for the IOI and GT tasks. Next, we retrieved the circuits from the models fine-tuned with supervision on the two tasks. Finally, we retrieved the circuits from the models on the ablated IOI dataset using all the described models.

Circuit Evaluation and Selection: A circuit is considered accurate and faithful when its output closely aligns with that of the full model, even at a high level of graph sparsity, as highlighted in previous works (Hanna et al., 2024; Conmy et al., 2023; Bhaskar et al., 2024). Specifically, we utilized metrics, such as Exact Match for IOI, Kendall's τ for GT, and KL Divergence for both, from the edge pruning approach to assess how closely the circuit's output aligns with the full model. Additionally, we measured the differences between target and distractor outputs, such as logit difference for IOI and probability difference for GT, both traditionally used for these tasks.

As shown in the circuit selection stage of Fig. 2, we observed an exponential relationship between edge sparsity and KL divergence, consistent across all models and tasks. More results for the metrics are provided in the Appendix B. For this work, we selected a group of 'best' circuits for each model by identifying those at the 'knee' of the KL-divergence curve, as highlighted in red in the circuit selection stage of Fig. 2. Using this selection criterion, we chose the circuits with the highest possible sparsity, just before the KL divergence sharply increases. Among the circuits in this selected group, we considered them to be equivalently 'best'. Therefore, we randomly selected one circuit from the group for further comparison and analysis. A sample circuit from this group represents the best tradeoff between performance and sparsity.

2.3 Circuit Functionality Discovery

We explore the intended functionalities of circuit components using a combination of attention heat maps and the logit lens. Unlike previous works, we conceptualize the mechanism within the model as either reducing the logits of incorrect choices or enhancing the logits of correct ones. In doing so, we broaden the focus beyond just the target outputs, providing a more comprehensive view of the model's inner mechanism. For IOI-related tasks, we categorize the prompts into ten groups: "B, IO, S1, PLACE, M, S2, V, OBJ, E, and T," following the structure of the IOI prompts. "B" represents the beginning of the sentence (e.g., "When" in Fig. 3), "M" refers to the middle of the sentence (e.g., a comma), and "T" denotes the end of the sentence (e.g., "to"). "V" represents the verb in the second half of the sentence, such as "gave," while "OBJ" refers to the object before "E." "S1" and "S2" refer to the distractor such as "John", while "IO" represents the indirect object the model needs to predict, such as "Mary,". We assign the rest of the templates into "T". For GT tasks, we divide the prompts into six parts: "B, N, V, S, E, and T." "B" represents the beginning of the sentence, "N" refers to the noun that the task focuses on (e.g., "war"), "V" indicates the verb such as "lasts" (suggesting the predicted numbers should be larger), "S" represents the start of the year, and "E" refers to the end of the sentence. We assign the rest of the templates into "T". It is important to note that the intended functionalities do not directly reflect the ultimate contributions of the circuit components to the final output. However, understanding these intended functionalities still provides valuable insights and allows us to estimate their direct effects. As shown in Fig. 3, we found that the previously identified Name Mover Head formed its own group of intended functionality, exhibiting shared similarities in attention patterns. In fact, most of the previously identified nodes with the same functionalities cluster well together when analyzed using the logit lens.

Attention Grouping is a technique used to analyze and categorize transformer heads based on their attention patterns. In transformers, each head focuses on different components of the input, reflecting what it "intends" to process. By examining the attention values, we can infer which parts of the sentence each head is attending to and determine its specific intended functionality. Since not all attended tokens carry equal importance, the method focus on the most relevant components. Heads that focus on similar elements, such as an indirect object (IO), are grouped together, suggesting they contribute similarly to the model's decision-making process.

Attention grouping helps uncover the various strategies a model uses to achieve the same outcome. For example, when the indirect object (IO) is emphasized, the model can do this by either increasing attention to the IO or by down-weighting attention to other components. Without grouping heads by their attention patterns, these diverse approaches could be overlooked. By clustering heads that focus on similar components, attention grouping allows for a deeper understanding of how the model processes information and how different heads contribute to the final prediction. This is especially useful in tasks like IOI, where multiple heads may contribute to the same result through distinct ways, offering valuable insights into the model's internal workings.

The attention grouping process involves computing the attention of each head for all tokens in a sentence and mapping them to sentence components (e.g., B, IO). The average attention for each component is calculated, and the top k components with the highest attention values are selected. If a component's attention exceeds the mean across these top components, the head is considered to be attending to that component. Heads are grouped if they attend to the same components. An example can be found in Fig. 3.

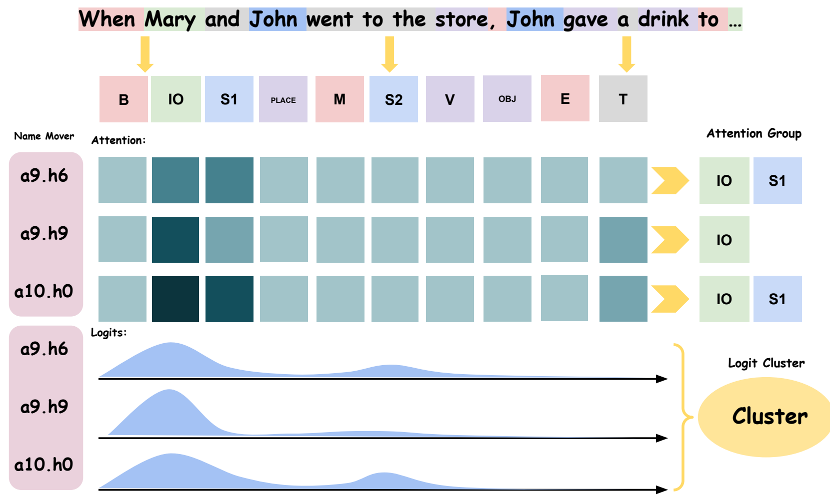

Figure 3: An example of attention grouping and logit clustering from IOI circuit. We assign the IOI prompts into several categories following their general pattern. We found that the previously identified Name Remover Head form its own cluster where the logits on IO get highly up-weighted over the other components. These heads' attention focus on IO and S1/S2 as well, indicating their intentional functionality of attending to and up-weighting IO for correct prediction.

For the IOI task, with 9 sentence components, $k = 5$ is chosen based on observations that heads primarily focus on 4 or less components. For tasks with a different number of sentence components, the value of k can be adjusted accordingly.

Logit Clustering is an essential technique used to group transformer heads based on their logit patterns, providing insights into how different heads intended to contribute to the model's decision-making process. One key method used in logit clustering is the *logit lens*, which allows researchers to probe intermediate representations in neural networks (NNs) and transformers. The logit lens works by examining the logits produced at each layer, representing the model's confidence about which token it might output at that specific stage. This technique reveals how early layers begin forming predictions and how these predictions evolve and refine as more layers process the input.

The *logit patterns* generated by different heads give a sense of how each head contributes to the model's predictions by up-weighting or down-weighting specific components. For example, in tasks such as the IOI task, where the indirect object (IO) is the correct target and the first subject (S1) is a distractor, a head that increases the logit on IO while decreasing the logit on S1 plays a crucial role in making the correct prediction. Similarly, other heads may contribute by focusing on irrelevant tokens such as punctuation or template words, which might help the model understand sentence structure or task objectives. Comparing the logit patterns across different heads and layers, thus, helps identify where significant changes in token predictions occur and how intermediate layers potentially contribute to the final outcome.

By clustering heads based on their logit patterns, we can group heads that are performing similar functions. This *logit clustering* allows for the identification of heads that collaborate in the model's decision-making process, providing a clearer view of the functional roles played by different heads. The use of logit clustering is particularly useful when logit patterns vary across layers or even repeat in heads across multiple layers, as we have observed. Applying clustering algorithms to these patterns automates the grouping process, facilitating the analysis of head functions. For example, clustering can reveal whether heads within certain layers, such as deeper layers, are grouped together or whether logit patterns are distributed across all layers.

In this work, we used K-means algorithm for logits clustering. While other clustering methods or human calibrations may provide more precise results, K-means has proven to deliver satisfying results in grouping logit patterns, reducing the amount of manual effort required in functional analysis. This method allows for a scalable approach to understanding how different transformer heads influence the model's output and decisions.

Attention Grouping + Logit Clustering: Both attention grouping and logit clustering are needed because each method has limitations when used in isolation. Attention grouping only indicates which components the heads focus on but doesn't reveal whether the heads are attempting to up-weight or down-weight those components. On the other hand, logit clustering identifies which components the heads are likely up-weighting or down-weighting but cannot clarify whether this effect is intentional or a byproduct from another function. For example, an up-weighted logit on IO might either reflect direct attention and up-weighting on this component, or it could be the result of down-weighting other components.

By integrating these two methods, we can better understand the intentional functionality of the heads. Attention grouping helps reveal the intention behind the model's focus, while logit clustering estimates the functionality — the effect the heads are having on specific components. When observing attention groups within the same logit cluster, it's easier to discern how nodes with similar functions differ in their intentions. Similarly, identical attention groups may belong to different logit clusters, indicating varied roles in the model's decision-making process. This combined framework simplifies the investigation of intentional functionalities, helping to clarify how different components intended to contribute to the overall behavior of the model.

3 Experimental Results

In this section, we provide a detailed analysis of how the structure and intended functionality of IOI circuits differ from those retrieved from ablated prompts, where we focus on the animal objects since the original GPT2-small is incapable of performing IOI tasks when changing IO and S to cities and colors. We then explore how various finetuning methods enhance performance on both the IOI and GT tasks. Lastly, we investigate how finetuning on the original IOI task improves the model's capability of performing the IOI task when applied to ablated prompts. Evaluations on model performance can be found in Appendix. Sec. B. For the following results, we focus on attention head only by collapsing the QKV nodes into the corresponding attention heads.

Table 1: The model's circuit differences on the GT and IOI tasks are compared in terms of the number of nodes. For clarity, we have consolidated the QKV (Query, Key, Value) nodes of the IOI task into a single attention head. The results are averaged over the selected group of the "best" circuits for each model under each task. A 95% confidence interval is calculated to demonstrate statistically significant variations in circuit sizes. Blocks highlighted in green indicate a statistically significant larger number of nodes. Our analysis shows that finetuning methods generally reduce the circuit sizes for the GT task while introducing more components to the IOI task.

Model A	Model B	GT task			IOI task		
		A only	B only	Shared	A only	B only	Shared
Original	IA3	17.2 ± 1.22	11.2 ± 2.10	38.6 ± 2.56	16.2 ± 1.83	20.8 ± 1.99	74.2 ± 1.34
Original	AdaLoRA	20.0 ± 1.69	13.8 ± 1.72	35.8 ± 2.01	13.0 ± 2.13	18.2 ± 1.09	77.4 ± 1.93
Original	Bitfit	25.0 ± 2.43	11.0 ± 1.82	30.8 ± 2.98	21.2 ± 1.58	20.2 ± 0.94	69.2 ± 1.65
Original	Full	19.2 ± 1.72	8.2 ± 1.78	36.6 ± 2.86	18.0 ± 1.84	22.0 ± 1.07	72.4 ± 1.37
Original	LoRA	20.2 ± 1.74	14.0 ± 2.69	35.6 ± 1.37	14.8 ± 2.33	22.0 ± 1.96	75.6 ± 1.48
IA3	AdaLoRA	10.4 ± 1.63	10.2 ± 1.28	39.4 ± 2.36	17.8 ± 1.22	18.4 ± 1.23	77.2 ± 1.01
IA3	Bitfit	17.8 ± 2.87	9.8 ± 1.74	32.0 ± 3.05	22.8 ± 1.91	15.4 ± 1.13	72.2 ± 1.40
IA3	Full	13.0 ± 2.46	8.0 ± 2.57	36.8 ± 1.95	19.2 ± 0.94	18.6 ± 1.29	75.8 ± 0.58
IA3	LoRA	10.8 ± 2.37	10.6 ± 2.07	39.0 ± 1.57	17.0 ± 1.07	19.6 ± 1.43	78.0 ± 1.04
LoRA	AdaLoRA	9.6 ± 1.81	9.6 ± 1.58	40.0 ± 1.90	17.6 ± 1.35	15.6 ± 1.99	80.0 ± 1.33
LoRA	Bitfit	18.0 ± 2.16	10.2 ± 2.99	31.6 ± 1.85	22.4 ± 2.68	12.4 ± 1.09	75.2 ± 1.01
LoRA	Full	14.0 ± 2.35	9.2 ± 2.85	35.6 ± 1.72	19.2 ± 1.60	16.0 ± 1.07	78.4 ± 0.90
AdaLoRA	Bitfit	17.4 ± 1.81	9.6 ± 1.97	32.2 ± 2.87	22.0 ± 0.78	14.0 ± 1.21	73.6 ± 0.66
AdaLoRA	Full	14.4 ± 1.56	9.6 ± 2.03	35.2 ± 2.55	20.2 ± 1.22	19.0 ± 1.21	75.4 ± 0.53
Bitfit	Full	11.2 ± 2.39	14.2 ± 1.95	30.6 ± 2.89	14.6 ± 1.72	21.4 ± 1.85	73.0 ± 1.07

3.1 IOI Circuits and Ablated Prompts

As shown in Fig. 1, substituting human names with other types of objects, such as animals, leads to significantly smaller circuits. Additionally, as illustrated in Fig. 4, we observed three distinct

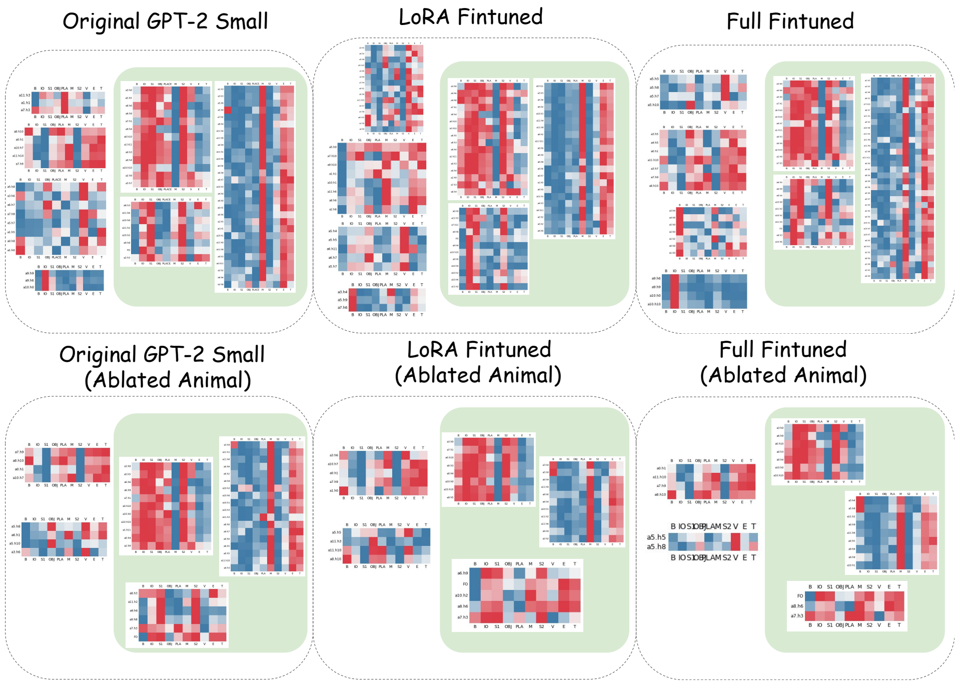

Figure 4: Logit clusters across fine-tuning methods and ablated prompts. The clusters are mostly uniform across fine-tuning methods for either prompt setting, indicating that fine-tuning does not significantly modify circuit's main functionalities; however, variations can be observed, both in terms of the general sizes of each cluster and individual head's intentional functionalities. Comparing IOI and ablated prompts, the ablation circuits consist of fewer functional clusters than IOI, and the number of heads for the shared clusters generally reduces. This is likely due to the complexity of processing human names in a semantically meaningful sentence which requires more functional nodes.

clustering groups that consistently appear across all circuits in both IOI and IOI-ablated tasks. One of the clusters places a strong emphasis on the M, E, and T parts of the sentence, in terms of logit, while also attending to the entire sentence. We assume that these nodes intend to complete the sentence with the simplest solutions, such as adding a comma or repeating the final word. Another cluster, in contrast, emphasizes all parts of the sentence except for M, E, and T, while still attending to the entire sentence. This cluster appears to function in opposition to the sentence completion cluster. This cluster also aligns with where all previously identified induction heads (Wang et al., 2022) are clustered. The final preserved cluster focuses heavily on the IO and S tokens, both in terms of logits and attention. This cluster generally down-weights OBJECT and PLACE, components that are irrelevant to correct prediction, which makes it distinguished from the induction heads cluster who usually up-weights them.

In terms of the general cluster analysis, the cluster of induction heads are mostly preserved, with only three additional nodes in IOI. The sentence completion cluster has significantly more heads in IOI, contributing to the difference in circuit sparsity. The cluster that focuses on IO and S is mostly preserved. S-inhibition and the negative name mover (downweighting IO and S, upweighting others) are also preserved.

Interestingly, there is no cluster found in the animals-ablated circuit that purely concentrates on IO in terms of logits and attention. It is worth noting that most of the attention patterns from ablated prompts do not involve any exclusive focus on IO or S tokens in terms of logits and attentions. Instead, they tend to focus on IO and S along with the beginning words. Furthermore, the name mover heads, which are well-clustered in the IOI circuits shown in Fig. 3, now appear to be more dispersed. For instance, node 10.0 still upweights IO over S1, but the effect is less pronounced. It also shows a strong upweighting of PLACE. Node 9.6 shifts to upweight S over IO, while node 9.9 joins the sentence completion cluster, upweighting M, E, and T but downweighting the other elements. These findings overall suggest that the model struggles to isolate IO and S from the prompt. Most of the time, it relies on the beginning of the sentence to identify the IO and S pair. Although the behavior of indirect object identification persists, the model's limited ability to accurately identify IO and S hinders its performance compared to the original IOI task. These findings are consistent across finetuning methods and ablation settings.

3.2 PEFT METHODS COMPARISON

By fine-tuning GPT-2 small on IOI and GT tasks, we observed that the circuits retrieved from GT tasks are generally statistically significantly smaller than those from the original GT tasks, as shown in Table 1. An analysis of logits and attention patterns on GT revealed that the fine-tuned model tends to focus more on the starting year, while the MLP in deeper layers increasingly up-weights target number continuations. These findings suggest that the reduction in circuit size for GT tasks is likely due to the simplicity of the task —simply selecting numbers greater than the starting year, as the task only has one direction (Greater Than)— and a more focused approach in solving it. More results on GT tasks can be found in Appendix. Sec. D. In contrast, for the more complex IOI task, fine-tuning methods generally result in a statistically significantly larger circuit to address the task's increased complexity.

For the IOI task, the overall structure of functionality for circuits is largely preserved, aligning with findings from Prakash et al.. Specifically, many of the heads identified by Wang et al. as serving certain functions continue to perform similar roles in fine-tuned cases. For example, Duplicate Token Heads, Previous Token Heads, some S-inhibition Heads, and certain Backup Name Mover Heads remain consistent across all fine-tuning methods, maintaining their functionalities despite the fine-tuning adjustments. This stability in key components ensures that model behavior is preserved, allowing fine-tuned heads to focus more effectively on specific task elements, thereby enhancing overall task performance. Furthermore, this consistency not only highlights the critical contribution of these heads to solving the IOI task but also validates our framework, which integrates attention grouping and logit clustering to explore the intended functionalities of LLM attention heads.

While many individual heads' intentional functionalities are generally preserved, fine-tuning often causes shifts in their behavior. Different fine-tuning methods may or may not achieve the same effect on this change in behavior. For instance, in the two Induction Heads identified by Wang et al., a5.h5 and a6.h9, we observe that while fine-tuning preserves the intentional functionality of a6.h9, a5.h5 undergoes significant modifications with different fine-tuning methods. In the original circuit, a5.h5 attends to the beginning of the sentence, up-weighting B, IO, S, OBJ, and down-weighting M, E, and T. However, after fine-tuning, Full Finetune, BitFit, and LoRA shift it to primarily up-weight V, whereas AdaLoRA and IA3 reverse the effect, down-weighting B, IO, S, OBJ, and PLACE, while up-weighting M, V, E, and T. Additional example of how fine-tuning methods align or differ in modifying the Name Mover Heads on IOI circuits is shown in Fig. 5.

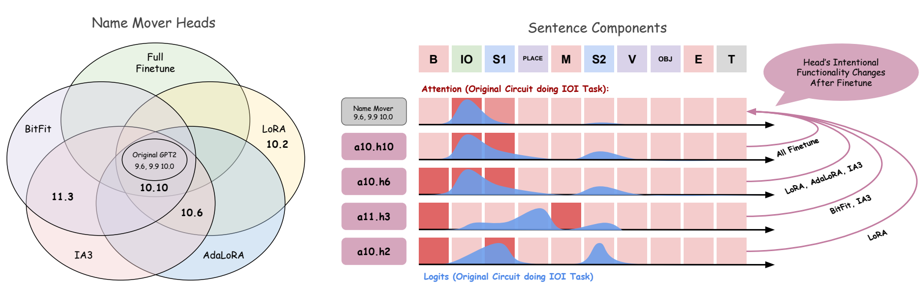

Figure 5: A comparison of how different fine-tuning methods modify the cluster of Name Mover Heads claimed by prior works. The Name Mover Heads significantly attend to and up-weight IO over other components. All fine-tuning methods increase the number of these heads to enhance prediction accuracy, shifting the functionalities of some particular heads in the original GPT-2 circuit for better performance. While some fine-tuning methods share similarities in the additional Name Mover Heads, variations are common.

Therefore, the differences between circuits may help explain how fine-tuning enhances the performance of LLMs on certain tasks. From our observations, fine-tuning increases the number of heads that up-weight logits on IO compared to the original GPT-2 circuit. Specifically, the improvement is primarily due to more ”focused” up-weighting on IO relative to other components. As shown in Fig. 5, all circuits from fine-tuned models enforce certain heads from the original circuit to function similarly to Name Mover Heads—those that primarily attend to and up-weight IO—leading to improved task performance. Notably, all fine-tuning methods modify the functionality of a10.h10, likely due to its high similarity to Name Mover Heads. However, PEFT requires a larger number of functionality-shifted heads to achieve comparable performance, which is unsurprising since full fine-tuning updates all weights and biases, thoroughly converting the functionalities of individual heads. In contrast, PEFT only updates a small subset of parameters, necessitating more heads with similar intentional functionalities to match the performance of fully fine-tuned circuits.

In conclusion, both PEFT methods and full fine-tuning improve performance not only by increasing the presence of highly specialized heads, such as Name Mover and Induction Heads, but also by simplifying and refining circuits through the pruning of irrelevant or less important components. This dual effect—enhancing the specificity and accuracy of critical heads while reducing unnecessary complexity—highlights the vital role of fine-tuning in optimizing circuit performance for specific tasks. These adjustments lead to more efficient and interpretable circuits that maintain high levels of task-specific accuracy.

4 Conclusions and Limitations

Overall, our framework significantly reduces the reliance on manual efforts, enhancing the efficiency of discovering the intended functionality of circuit components. Through our analysis of how the circuits for the IOI and GT tasks vary across different fine-tuning methods and ablated prompts, we found that while the overall structure of intended functionality is preserved, the specific components responsible for these functions may change. This finding suggests that pre-identified circuits and functionalities are subject to variations depending on the ablated prompts and fine-tuning methods used. We hope that our findings and framework will inspire further exploration of circuit utilization and its interpretability in LLMs.

As discussed in the paper, rather than estimating the direct effect of individual nodes on target logit values, we focus on their intended functionalities. While exploring intended functionalities can provide a reasonable approximation of direct effects—since pre-identified heads with similar functions are generally well-clustered—it is important to recognize that they are not equivalent. Drawing an analogy to “correlation is not causation,” we emphasize that intended functionality does not necessarily reflect the “true” functionality of circuit components. Furthermore, intended functionality may depend on the performance of previous states, meaning it can become inconsistent if those earlier states are perturbed. However, this challenge could also arise in causal exploration methods like Patch patching.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*, 2022.

Leonard Bereska and Efstratios Gavves. Mechanistic interpretability for ai safety–a review. *arXiv preprint arXiv:2404.14082*, 2024.

Adithya Bhaskar, Alexander Wettig, Dan Friedman, and Danqi Chen. Finding transformer circuits with edge pruning. *arXiv preprint arXiv:2406.16778*, 2024.

Lawrence Chan, Adrià Garriga-Alonso, Nicholas Goldwosky-Dill, Ryan Greenblatt, Jenny Nitishinskaya, Ansh Radhakrishnan, Buck Shlegeris, and Nate Thomas. Causal scrubbing, a method for rigorously testing interpretability hypotheses. *AI Alignment Forum*, 2022. https://www.alignmentforum.org/posts/JvZhhzycHu2Yd57RN/causal-scrubbing-a-method-for-rigorously-testing.

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

Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Shizhe Diao, Zhichao Huang, Ruijia Xu, Xuechun Li, Yong Lin, Xiao Zhou, and Tong Zhang. Black-box prompt learning for pre-trained language models. *arXiv preprint arXiv:2201.08531*, 2022.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235, 2023.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1(1):12, 2021.

Deqing Fu, Tian-Qi Chen, Robin Jia, and Vatsal Sharan. Transformers learn higher-order optimization methods for in-context learning: A study with linear models. *arXiv preprint arXiv:2310.17086*, 2023.

Nicholas Goldowsky-Dill, Chris MacLeod, Lucas Sato, and Aryaman Arora. Localizing model behavior with path patching. *arXiv preprint arXiv:2304.05969*, 2023.

Michael Hanna, Ollie Liu, and Alexandre Variengien. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *Advances in Neural Information Processing Systems*, 36, 2024.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Samyak Jain, Ekdeep Singh Lubana, Kemal Oksuz, Tom Joy, Philip HS Torr, Amartya Sanyal, and Puneet K Dokania. What makes and breaks safety fine-tuning? mechanistic study. *arXiv preprint arXiv:2407.10264*, 2024.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965, 2022.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`, 2022.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022a.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*, 2022b.

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. Circuit component reuse across tasks in transformer language models. *arXiv preprint arXiv:2310.08744*, 2023.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. *arXiv preprint arXiv:2301.05217*, 2023.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.

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

Ying Sheng, Shiyi Cao, Dacheng Li, Coleman Hooper, Nicholas Lee, Shuo Yang, Christopher Chou, Banghua Zhu, Lianmin Zheng, Kurt Keutzer, et al. S-lora: Serving thousands of concurrent lora adapters. *arXiv preprint arXiv:2311.03285*, 2023.

Yi-Lin Sung, Varun Nair, and Colin A Raffel. Training neural networks with fixed sparse masks. *Advances in Neural Information Processing Systems*, 34:24193–24205, 2021.

Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery. *arXiv preprint arXiv:2310.10348*, 2023.

Curt Tigges, Michael Hanna, Qinan Yu, and Stella Biderman. Llm circuit analyses are consistent across training and scale. *arXiv preprint arXiv:2407.10827*, 2024.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*, 2022.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *arXiv preprint arXiv:2106.10199*, 2021.

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

Appendix Table of Contents

A Model Performance

B Background and Related Works

Mechanistic Interpretability (MI) (Elhage et al., 2021; Olsson et al., 2022) aims to discover interpretable components of an otherwise blackbox neural network. Such analyses are typically performed via a series of input perturbations (also known as *patching*) to ablate the effect of individual model components to its predictive behavior (Chan et al., 2022; Meng et al., 2022a;b; Goldowsky-Dill et al., 2023). MI has been successfully applied to natural language tasks that feature a controlled output space on small-scale pre-trained LMs (Wang et al., 2022; Hanna et al., 2024); it has also been applied to study in-context learning and algorithmic behaviors on stylized transformers (Akyürek et al., 2022; Fu et al., 2023; Nanda et al., 2023). While early works put equal focus on circuit discovery and interpretability, recent ones have emphasized scalable circuit discovery (Conmy et al., 2023; Syed et al., 2023; Bhaskar et al., 2024) over extracting human-understandable algorithms, which requires extensive manual effort to examine the computational behavior of model components. Our work integrates recent best practices in circuit discovery to *automate* interpretability.

Parameter-Efficient Fine-tuning (Mangrulkar et al., 2022) aims to improve model performance on downstream tasks by training only a small portion of parameter relative to the full model. Referring to Ding et al. (2023) for a more detailed survey, most popular approaches exploits the low-rank structure of projection matrices (Hu et al., 2021; Zhang et al., 2023) or introduce a fixed set of scaling and/or bias parameters (Zaken et al., 2021; Liu et al., 2022). Other representative approaches include prompt tuning (Lester et al., 2021; Li & Liang, 2021; Diao et al., 2022) and dynamically identifying tuning parameters via influence functions (Sung et al., 2021). In this work, we primarily investigate LoRA (Hu et al., 2021), AdaLora (Zhang et al., 2023), BitFit (Zaken et al., 2021), and IA3 (Liu et al., 2022). These methods are broadly applied in many production settings, since they are more scalable with commercial hardware and can be served with thousands of replicas simultaneously (Sheng et al., 2023).

Bhaskar et al. has studied the change effects of circuits induced by fine-tuning on an entity tracking task, and found that fine-tuning *enhances* existing mechanisms for billion-scale LMs. Our work extends this study with a more diverse set of tasks and PEFT methods; and more importantly, we have identified that fine-tuning *modifies* existing mechanisms for small LMs.

This section presents the evaluation results of all models across various tasks and ablated prompts. Although the model performs well on the IOI ablated tasks after fine-tuning, GPT-2 small struggles with indirect object identification when the main objects are replaced with colors and cities. Specifically, the model's ability to handle these ablations remains limited, indicating that fine-tuning has not fully generalized the model to different types of input modifications. Moreover, even after fine-tuning, the performance on the IOI task ablated with cities is still suboptimal, suggesting that the model's understanding of abstract entities such as locations remains insufficient. These findings highlight the need for more targeted interventions or further fine-tuning strategies to improve model robustness across diverse ablations.

C IOI circuits analysis

This section documents the detailed qualitative results for IOI analysis. Sec. C.1 lists the visualization of comparative circuit analysis. The overlapped nodes are in white colors while the unique nodes to each cicuits are shown in either pink or blue. Sec. C.2 lists the results of logit clustering and attention group visualizations.

C.1 Circuit Comparison

Comparative circuit analysis on the original IOI tasks can be found in Fig. 6, Fig. 7, Fig. 8,Fig. 9, Fig. 10, and Fig. 11. The analysis with ablated prompts with animals can be found in Fig. 12, Fig. 13, Fig. 14, and Fig. 17. The analysis with ableted prompts with cities can be found in Fig. 18 and Fig. 19. Analysis on prompts ablated with colors can be found in Fig. 21.

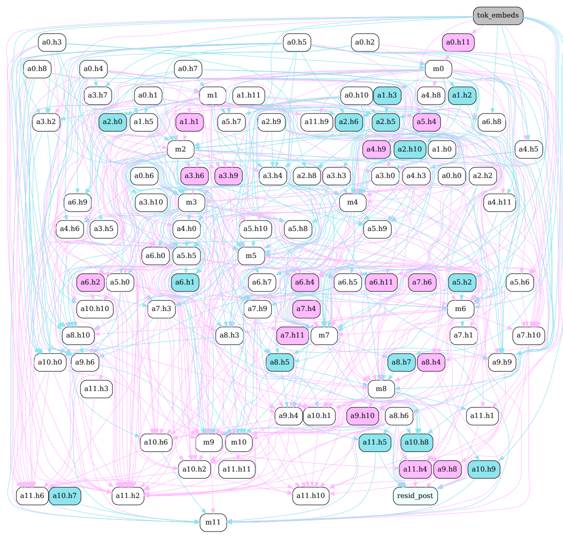

Figure 6: Circuit difference: LoRA vs Full. Red/pink indicates LoRA-specific circuits, blue indicates Full-specific circuits, and white represents shared circuits.

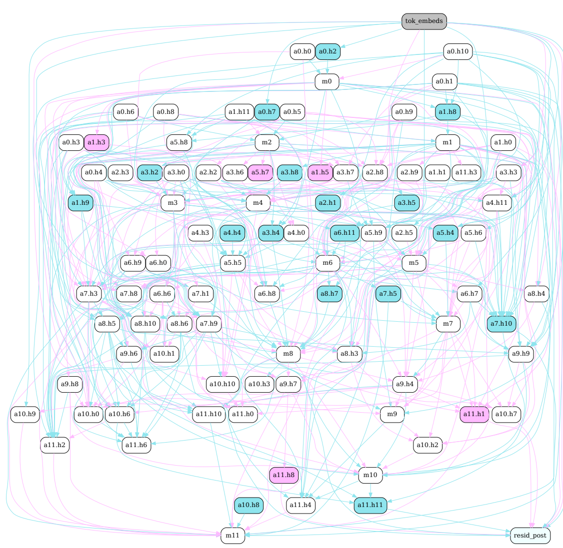

Figure 7: Circuit difference: Original vs AdaLoRA. Red/pink indicates Original-specific circuits, blue indicates AdaLoRA-specific circuits, and white represents shared circuits.

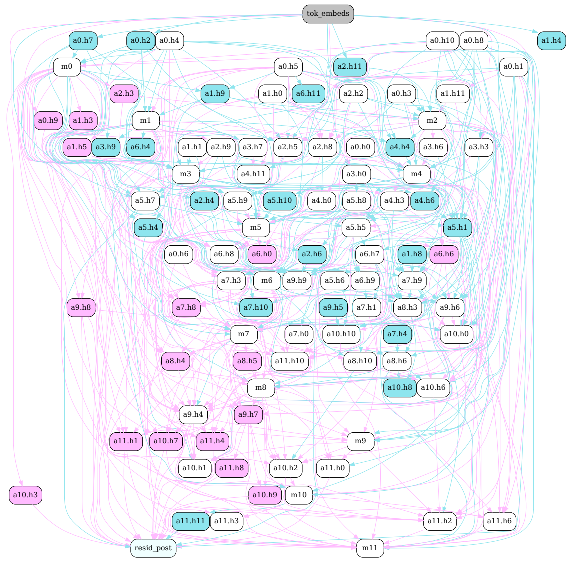

Figure 8: Circuit difference: Original vs BitFit. Red/pink indicates Original-specific circuits, blue indicates BitFit-specific circuits, and white represents shared circuits.

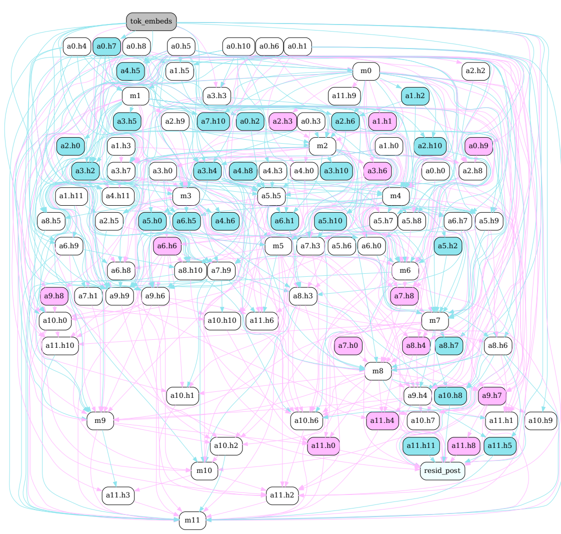

Figure 9: Circuit difference: Original vs Full. Red/pink indicates Original-specific circuits, blue indicates Full-specific circuits, and white represents shared circuits.

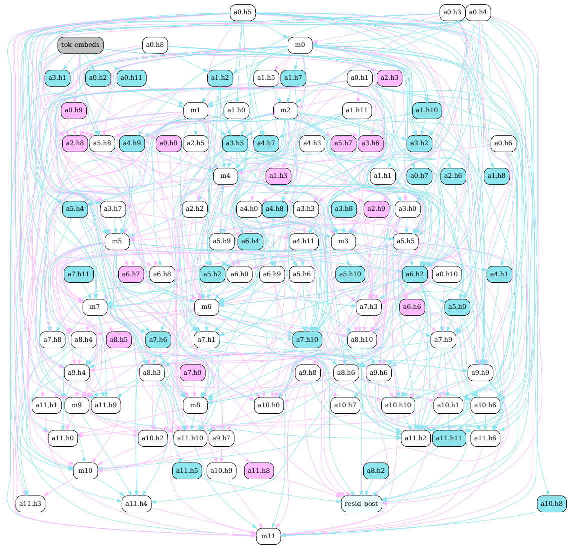

Figure 10: Circuit difference: Original vs IA3. Red/pink indicates Original-specific circuits, blue indicates IA3-specific circuits, and white represents shared circuits.

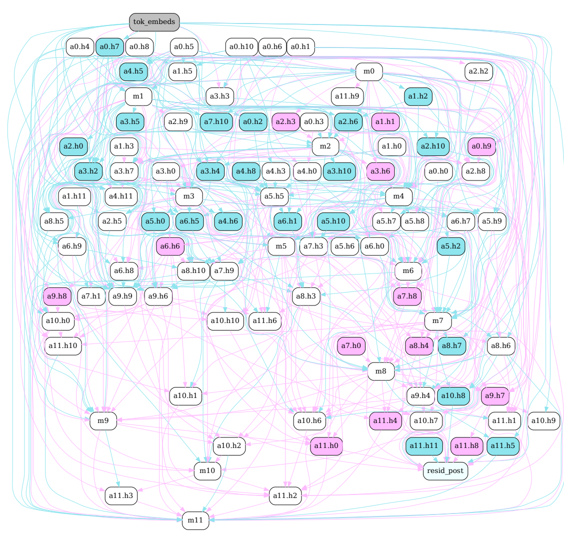

Figure 11: Circuit difference: Original vs LoRA. Red/pink indicates Original-specific circuits, blue indicates LoRA-specific circuits, and white represents shared circuits.

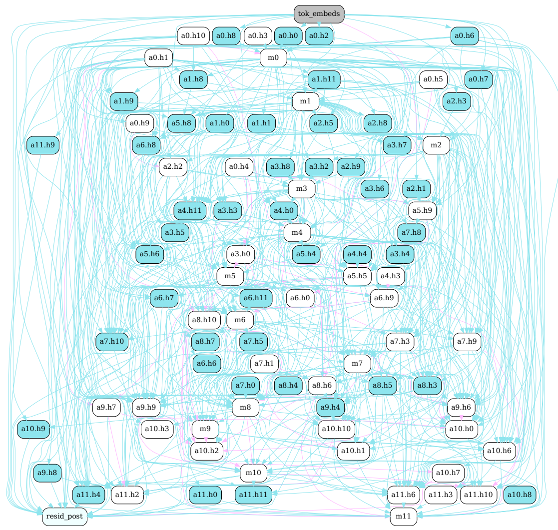

Figure 12: AdaLoRA circuit difference between IOI and IOI-animals, where blue corresponds to IOI only and red corresponds to IOI-animals only circuit components. White refers to shared circuit components.

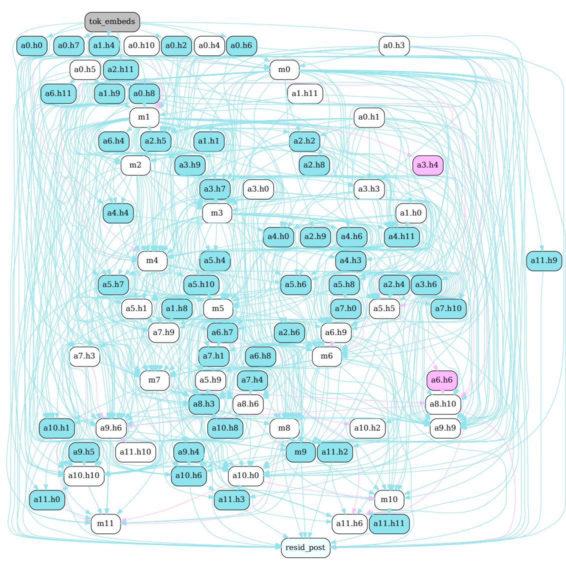

Figure 13: BitFit circuit difference between IOI and IOI-animals, where blue corresponds to IOI only and red corresponds to IOI-animals only circuit components. White refers to shared circuit components.

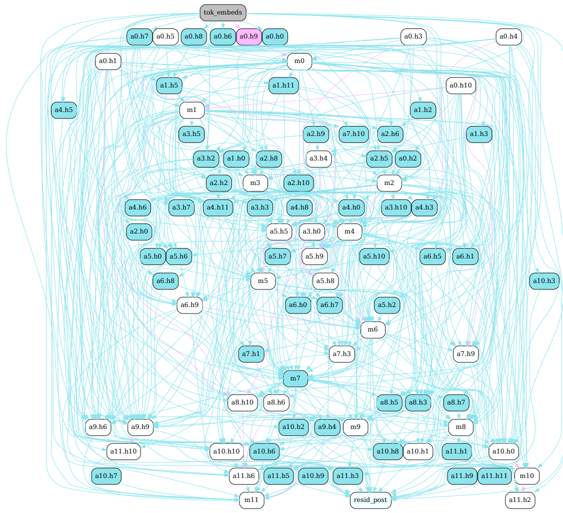

Figure 14: Full circuit difference between IOI and IOI-animals, where blue corresponds to IOI only and red corresponds to IOI-animals only circuit components. White refers to shared circuit components.

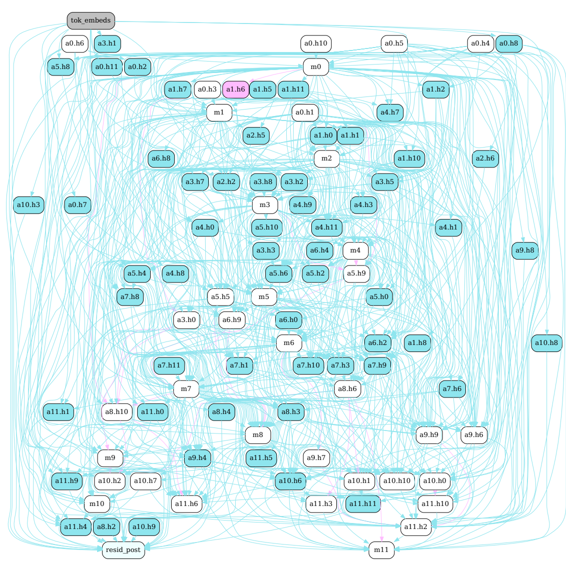

Figure 15: IA3 circuit difference between IOI and IOI-animals, where blue corresponds to IOI only and red corresponds to IOI-animals only circuit components. White refers to shared circuit components.

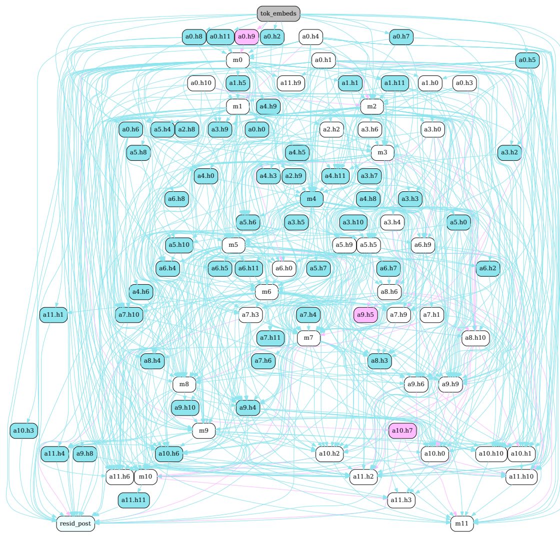

Figure 16: LoRA circuit difference between IOI and IOI-animals, where blue corresponds to IOI only and red corresponds to IOI-animals only circuit components. White refers to shared circuit components.

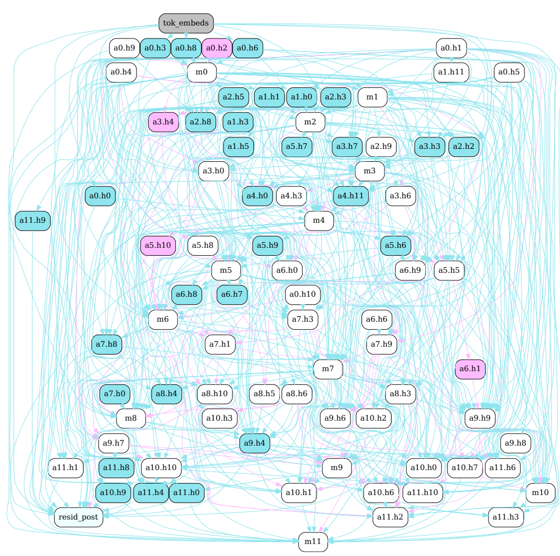

Figure 17: Original circuit difference between IOI and IOI-animals, where blue corresponds to IOI only and red corresponds to IOI-animals only circuit components. White refers to shared circuit components.

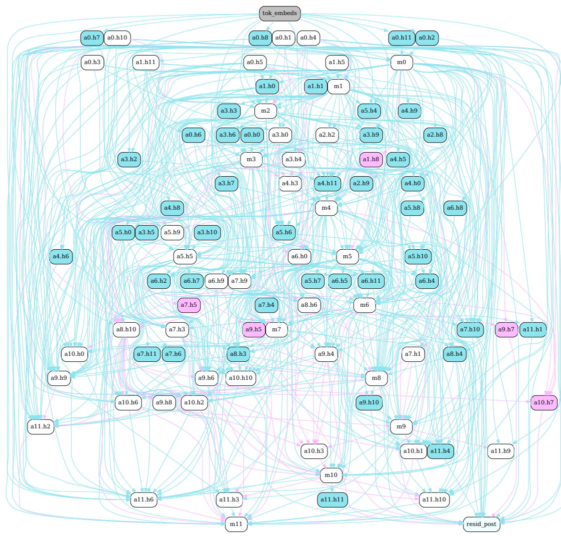

Figure 18: LoRA circuit difference between IOI and IOI-cities, where blue corresponds to IOI only and red corresponds to IOI-cities only circuit components. White refers to shared circuit components.

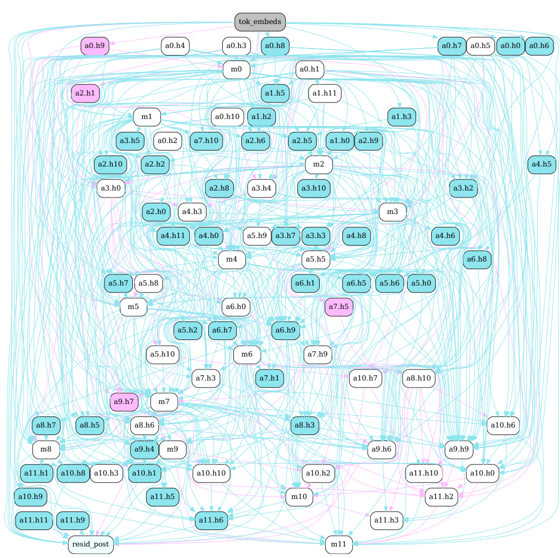

Figure 19: Full finetune circuit difference between IOI and IOI-cities, where blue corresponds to IOI only and red corresponds to IOI-cities only circuit components. White refers to shared circuit components.

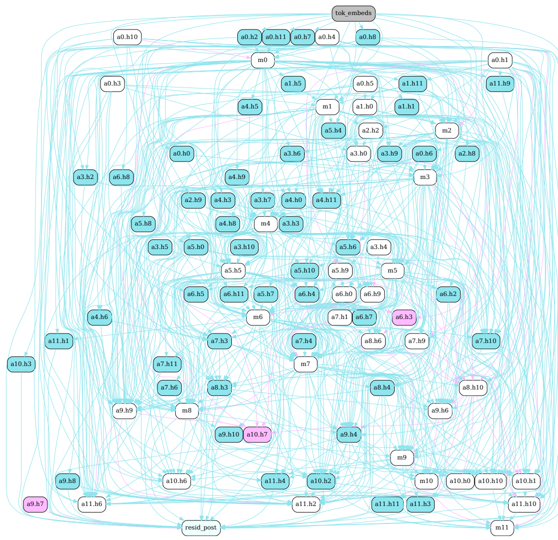

Figure 20: LoRA circuit difference between IOI and IOI-colors, where blue corresponds to IOI only and red corresponds to IOI-colors only circuit components. White refers to shared circuit components.

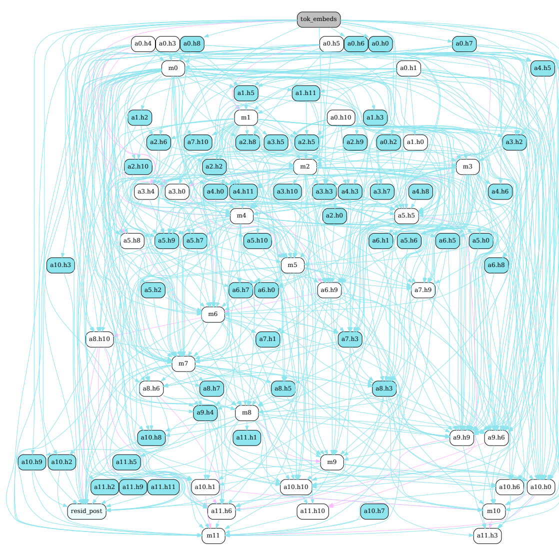

Figure 21: Full finetune circuit difference between IOI and IOI-colors, where blue corresponds to IOI only and red corresponds to IOI-colors only circuit components. White refers to shared circuit components.

Table 2: IOI circuit performances, rounded to the nearest hundredth. The original GPT-2 model, due to poor accuracy on colors and cities ablated settings, is not satisfiable for meaningful circuit investigations on these two tasks. Among fine-tuning methods, cities-ablated prompts pose the most difficulty for the circuit to preserve full model performance.

Metric	Ablation	AdaLoRA	BitFit	Full	IA3	LoRA	Original
Acc	Names	0.95 ± 0.01	0.99 ± 0.00	0.98 ± 0.00	0.97 ± 0.00	0.94 ± 0.00	0.73 ± 0.01
	Animals	0.96 ± 0.02	0.99 ± 0.00	0.99 ± 0.00	0.96 ± 0.01	0.97 ± 0.00	0.52 ± 0.01
	Colors	0.95 ± 0.02	0.99 ± 0.00	0.99 ± 0.00	0.96 ± 0.01	0.96 ± 0.00	–
	Cities	0.74 ± 0.02	0.97 ± 0.01	0.93 ± 0.01	0.76 ± 0.02	0.79 ± 0.00	–
LD	Names	5.95 ± 0.16	11.62 ± 0.15	11.63 ± 0.12	6.15 ± 0.10	5.83 ± 0.12	3.24 ± 0.18
	Animals	4.61 ± 0.41	12.32 ± 0.43	11.03 ± 0.34	4.35 ± 0.30	5.17 ± 0.27	0.86 ± 0.07
	Colors	4.23 ± 0.42	11.50 ± 0.39	10.41 ± 0.36	4.39 ± 0.31	4.91 ± 0.20	–
	Cities	2.57 ± 0.20	9.60 ± 0.38	8.82 ± 0.34	2.19 ± 0.15	3.37 ± 0.08	–
KL	Names	0.21 ± 0.01	0.05 ± 0.00	0.06 ± 0.01	0.16 ± 0.01	0.20 ± 0.01	0.30 ± 0.00
	Animals	0.09 ± 0.02	0.06 ± 0.01	0.04 ± 0.01	0.08 ± 0.01	0.06 ± 0.01	0.28 ± 0.01
	Colors	0.10 ± 0.03	0.05 ± 0.01	0.04 ± 0.00	0.09 ± 0.02	0.07 ± 0.00	–
	Cities	0.38 ± 0.04	0.11 ± 0.02	0.21 ± 0.03	0.39 ± 0.03	0.37 ± 0.01	–
EM	Names	0.95 ± 0.00	0.99 ± 0.00	0.98 ± 0.00	0.96 ± 0.00	0.94 ± 0.00	0.76 ± 0.01
	Animals	0.96 ± 0.02	0.99 ± 0.00	0.99 ± 0.00	0.96 ± 0.01	0.97 ± 0.00	0.66 ± 0.01
	Colors	0.95 ± 0.02	0.99 ± 0.00	0.99 ± 0.00	0.96 ± 0.01	0.97 ± 0.00	–
	Cities	0.77 ± 0.02	0.97 ± 0.01	0.93 ± 0.01	0.77 ± 0.02	0.82 ± 0.00	–

Table 3: GT task performances

Finetune Method	ES	PD	PD(10)	KT	KL
Original	0.99 ± 0.00	0.72 ± 0.00	0.33 ± 0.01	0.78 ± 0.01	0.23 ± 0.02
AdaLoRA	0.99 ± 0.00	0.96 ± 0.00	0.62 ± 0.01	0.84 ± 0.01	0.38 ± 0.02
BitFit	0.99 ± 0.00	1.00 ± 0.00	0.02 ± 0.00	0.88 ± 0.00	0.20 ± 0.01
Full	0.99 ± 0.00	0.98 ± 0.00	0.53 ± 0.03	0.82 ± 0.01	0.74 ± 0.05
IA3	0.99 ± 0.00	0.92 ± 0.00	0.55 ± 0.01	0.83 ± 0.00	0.33 ± 0.01
LoRA	0.99 ± 0.00	0.92 ± 0.00	0.58 ± 0.01	0.83 ± 0.01	0.30 ± 0.02

C.2 Logit clustering and attention Grouping

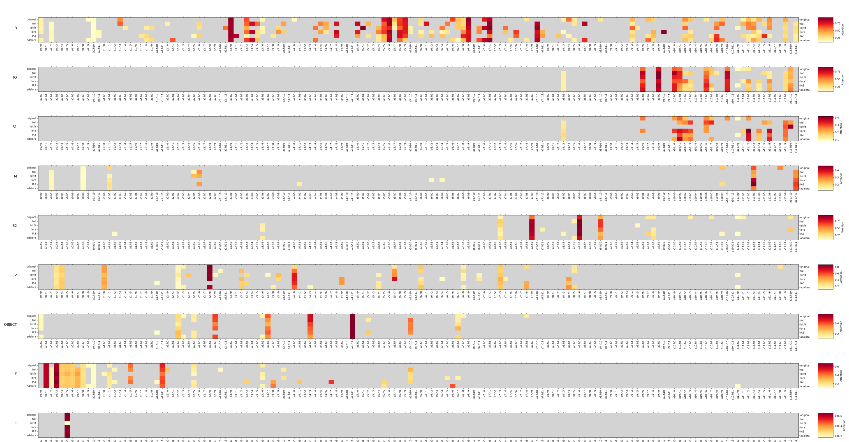

Figure 22: Attention grouping results for all components circuit. Grey cells indicate pruned nodes of minimal attention.

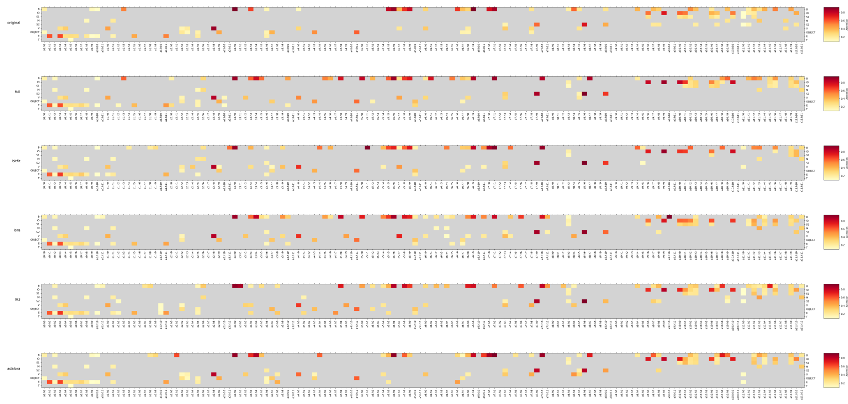

Figure 23: Attention grouping results for all finetuning methods circuit. Grey cells indicate pruned nodes of minimal attention.

D GT CIRCUIT ANALYSIS

D.1 CIRCUIT COMPARISON

D.2 LOGIT CLUSTERING AND ATTENTION GROUPING

E REPRODUCIBILITY

The implementation of circuit discovery mainly depends on the code from Edge Pruning. More details can be found in their github. The code to perform logit lens and attention grouping will be released upon acceptance.

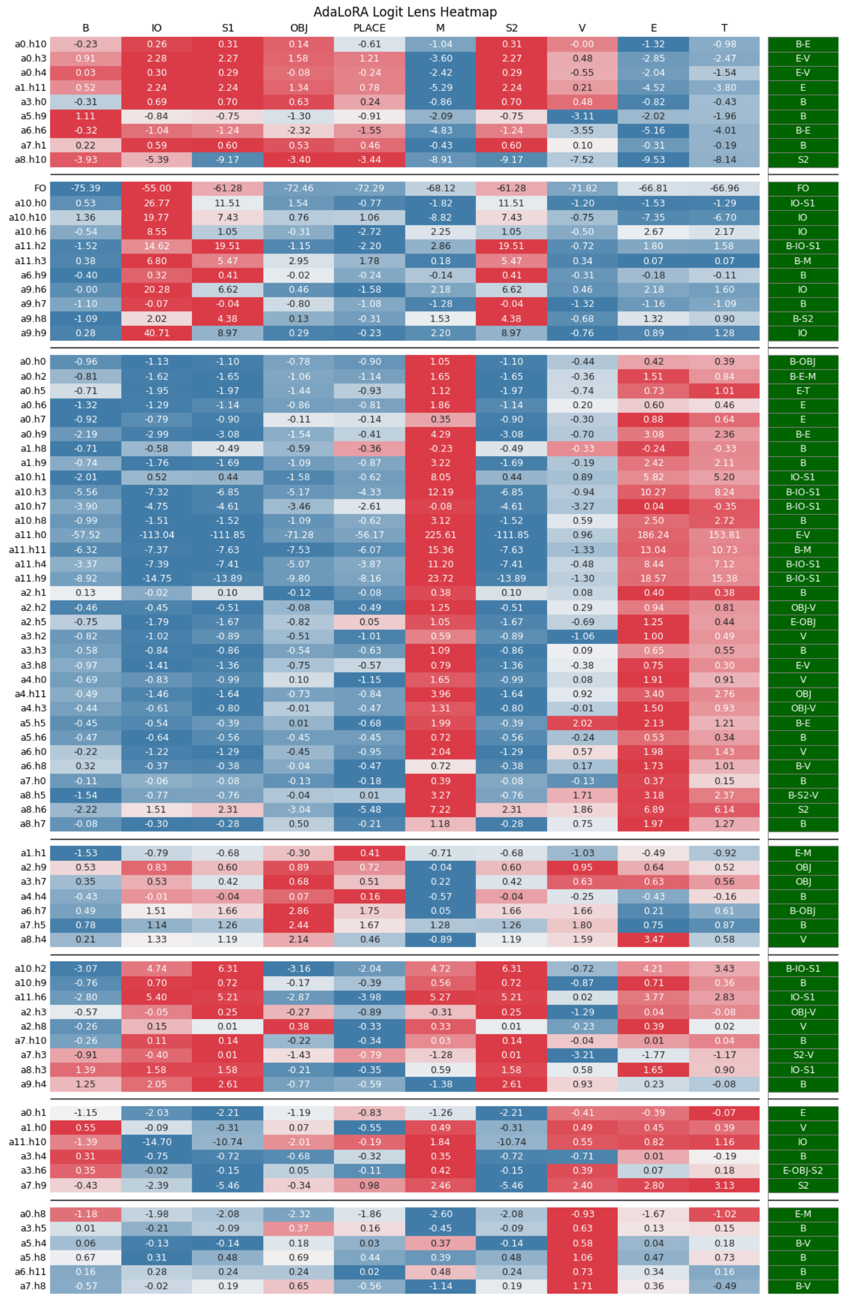

Figure 24: AdaLoRA Logit Clusters

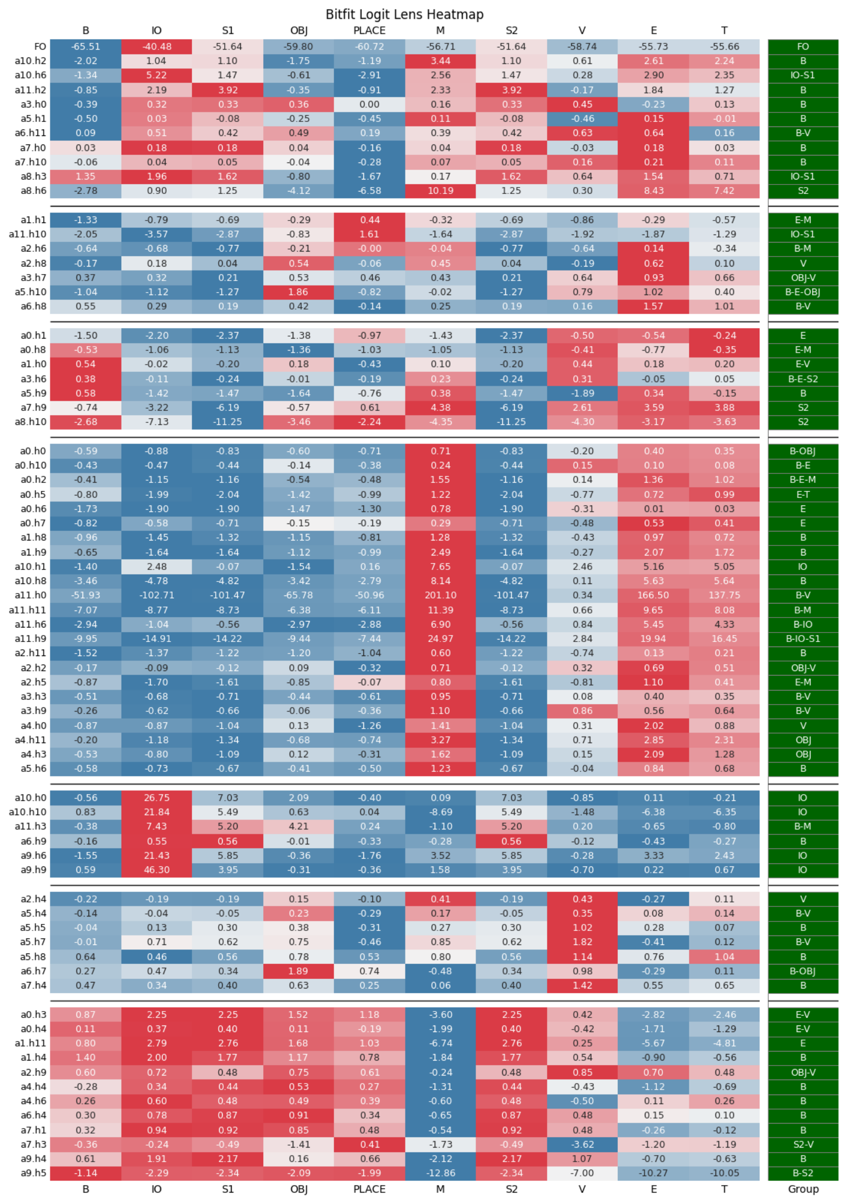

Figure 25: BitFit Logit Clusters

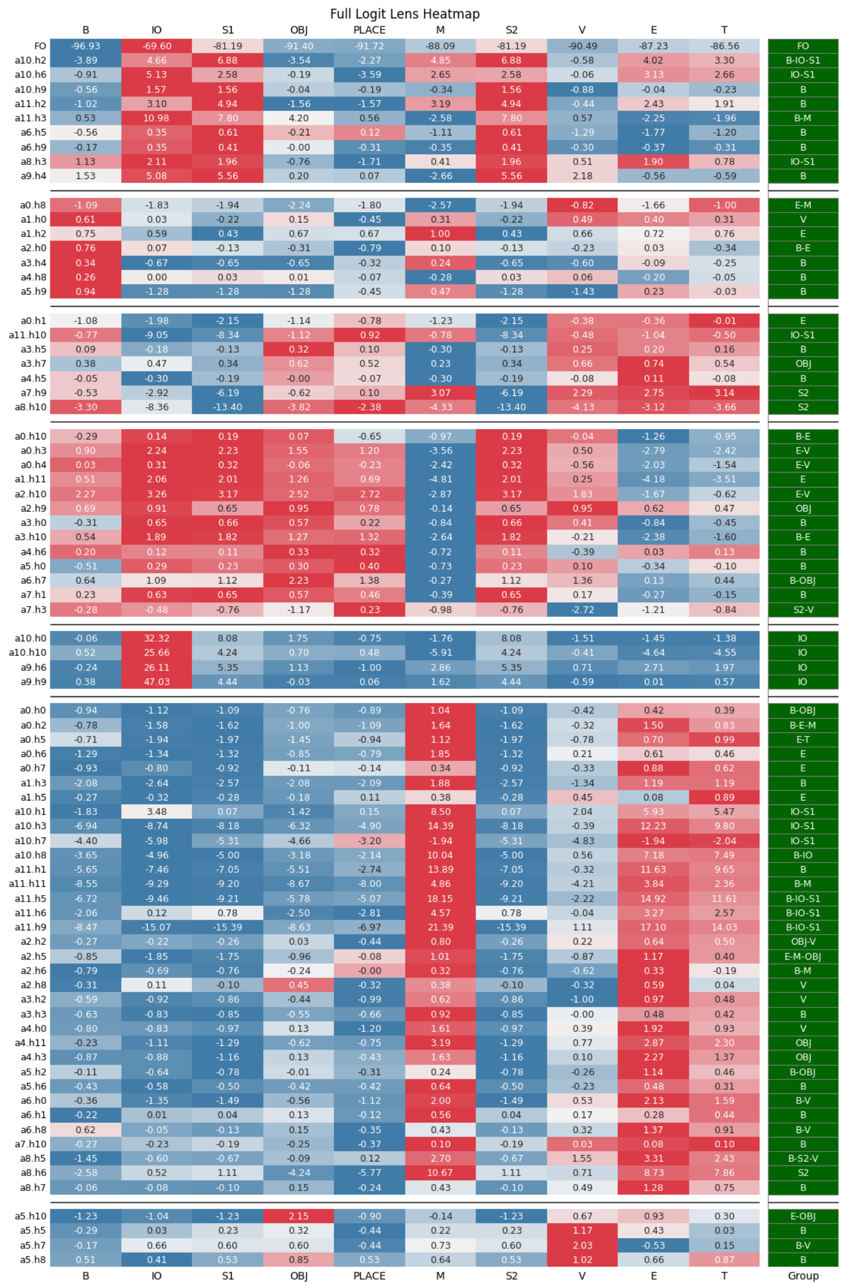

Figure 26: Full Logit Clusters

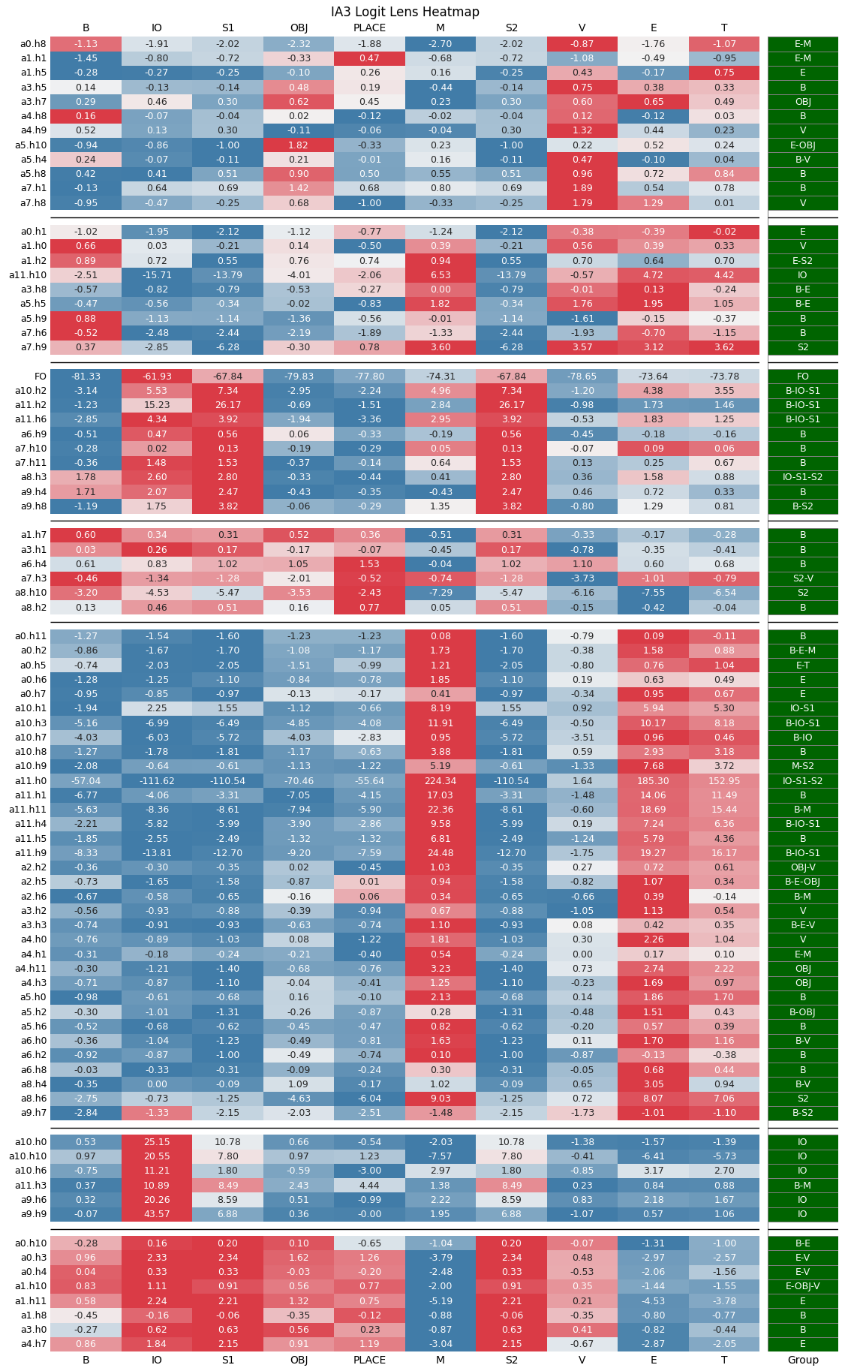

Figure 27: IA3 Logit Clusters

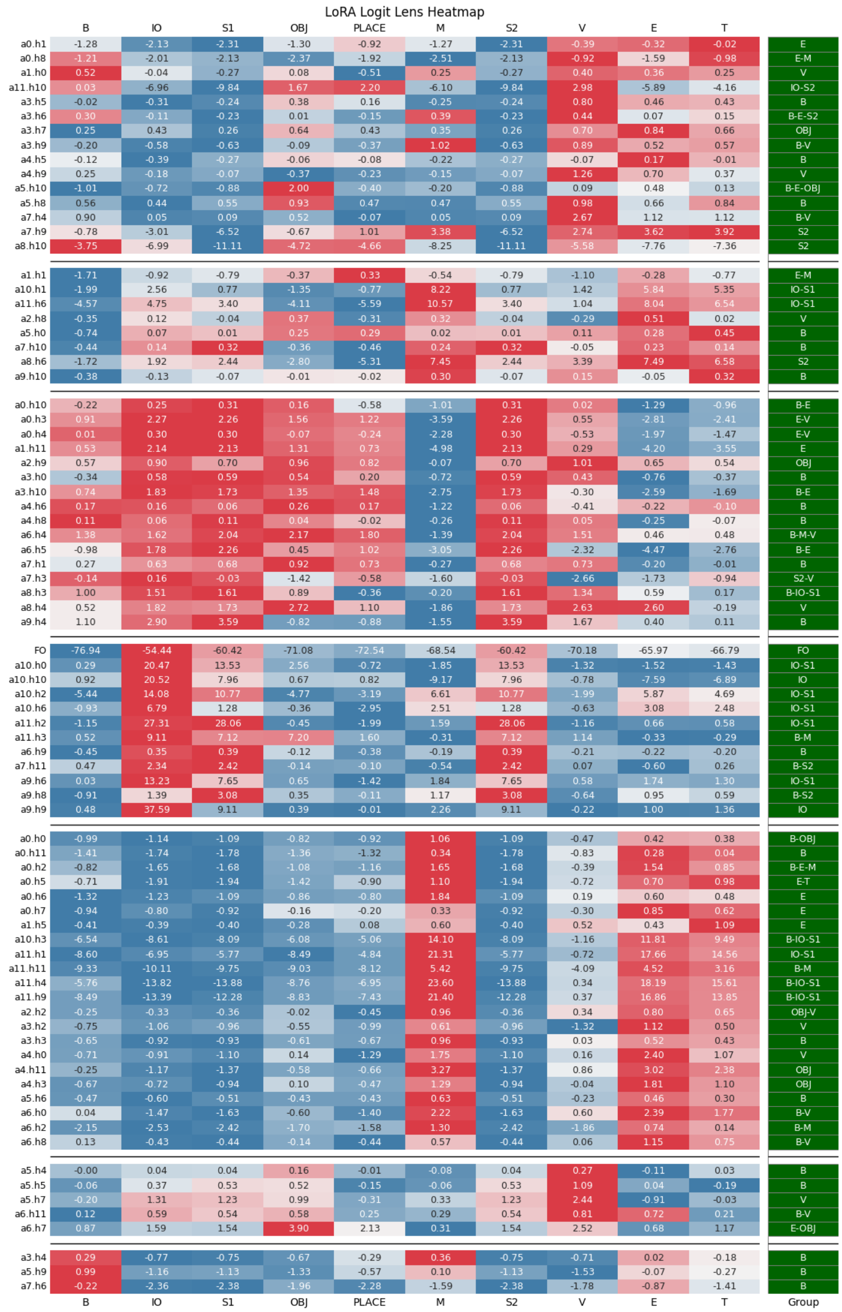

Figure 28: LoRA Logit Clusters

Original Logit Lens Heatmap

	B	IO	S1	OBJ	PLACE	M	S2	V	E	T	Group
a0.h10	-0.30	0.16	0.20	0.08	-0.68	-1.05	0.20	-0.08	-1.38	-1.03	B-E
a0.h3	0.90	2.23	2.23	1.55	1.19	-3.53	2.23	0.50	-2.77	-2.40	E-V
a0.h4	0.04	0.32	0.32	-0.07	-0.22	-2.48	0.32	-0.56	-2.07	-1.57	E-V
a1.h11	0.67	2.24	2.20	1.39	0.71	-4.99	2.20	0.40	-4.32	-3.58	E
a10.h10	1.68	18.80	12.42	1.73	1.56	-10.02	12.42	-0.68	-8.45	-7.50	IO-S1
a2.h9	0.67	0.95	0.71	0.99	0.85	-0.14	0.71	0.90	0.49	0.39	OBJ
a3.h0	-0.27	0.68	0.69	0.59	0.24	-0.95	0.69	0.43	-0.89	-0.49	B
a3.h7	0.40	0.61	0.49	0.69	0.58	0.01	0.49	0.57	0.48	0.36	OBJ
a5.h5	-0.08	0.62	0.81	0.62	-0.12	-1.01	0.81	0.54	-0.71	-0.69	B
a6.h6	-0.45	-0.17	-0.18	-0.91	-0.33	-1.69	-0.18	-1.61	-1.72	-1.48	B
a6.h9	-0.06	0.14	0.17	-0.03	-0.25	-0.38	0.17	-0.22	-0.36	-0.35	B
a7.h1	0.06	0.36	0.39	0.49	0.38	-0.11	0.39	0.26	-0.03	0.04	B
a9.h4	1.97	2.53	2.88	-0.04	-0.13	-0.70	2.88	0.58	0.28	0.18	B
a0.h8	-1.09	-1.83	-1.94	-2.24	-1.82	-2.69	-1.94	-0.82	-1.73	-1.05	E-M
a1.h0	0.71	0.15	-0.11	0.16	-0.39	0.11	-0.11	0.48	0.28	0.16	V
a1.h5	-0.29	-0.28	-0.26	-0.07	0.26	0.16	-0.26	0.36	-0.34	0.64	E
a3.h6	0.28	0.06	-0.06	0.03	-0.12	0.18	-0.06	0.38	0.02	0.06	B-E-OBJ
a5.h7	-0.43	-0.38	-0.36	-0.16	-0.46	0.07	-0.36	0.99	-0.32	0.02	B-V
a5.h8	0.28	0.37	0.48	0.77	0.52	0.56	0.48	0.83	0.58	0.76	B
a6.h7	0.23	0.50	0.54	1.62	0.88	0.00	0.54	0.78	0.29	0.42	B-OBJ
a7.h8	-1.11	-0.68	-0.57	0.55	-0.98	-0.37	-0.57	0.97	0.70	-0.08	B-OBJ-V
FO	-90.17	-75.02	-78.55	-89.27	-85.68	-83.08	-78.55	-86.63	-82.86	-82.92	FO
a10.h2	-1.90	3.87	5.54	-2.21	-1.32	3.60	5.54	-0.52	3.25	2.67	B-S1
a10.h6	-0.05	5.07	3.36	-0.25	-1.25	2.20	3.36	-0.32	2.54	2.10	B-IO-S1
a11.h2	-1.88	2.96	8.10	-2.18	-2.33	3.56	8.10	-0.49	2.59	2.22	B-IO-S1
a11.h6	-1.11	1.11	2.16	-1.20	-1.62	0.21	2.16	-0.51	-0.26	-0.44	B
a2.h3	-0.62	-0.15	0.14	-0.35	-0.95	-0.40	0.14	-1.38	0.05	-0.11	OBJ-V
a8.h3	1.33	1.62	1.82	-0.15	-0.07	0.60	1.82	0.49	1.81	1.10	B
a9.h8	-0.98	2.16	4.28	0.43	0.14	0.85	4.28	-0.76	0.77	0.39	B-S2
a0.h0	-0.93	-1.11	-1.08	-0.74	-0.88	1.05	-1.08	-0.43	0.41	0.39	B-OBJ
a0.h5	-0.69	-1.94	-1.96	-1.45	-0.93	1.12	-1.96	-0.79	0.69	0.99	E-T
a0.h6	-1.31	-1.27	-1.11	-0.84	-0.79	1.90	-1.11	0.21	0.64	0.49	E
a0.h9	-2.33	-3.16	-3.25	-1.69	-0.52	4.35	-3.25	-0.80	3.11	2.38	B-E
a1.h3	-2.60	-3.33	-3.24	-2.62	-2.62	2.48	-3.24	-1.69	1.55	1.57	B
a10.h1	-2.17	2.50	0.18	-1.47	-0.58	8.43	0.18	0.58	6.09	5.39	IO-S1
a10.h3	-4.32	-5.97	-5.29	-4.04	-3.44	10.41	-5.29	-0.65	8.84	7.13	B-S2
a10.h9	-1.68	-0.51	-0.39	-1.25	-1.02	4.53	-0.39	-1.34	4.33	3.16	B-M
a11.h0	-59.41	-115.97	-114.80	-72.89	-58.16	231.26	-114.80	1.55	191.38	157.94	IO-S1-S2
a11.h1	-8.10	-4.36	-3.65	-8.45	-5.45	18.68	-3.65	-2.76	15.31	12.32	B-IO-S2
a11.h4	-4.18	-7.96	-7.96	-5.60	-4.46	11.44	-7.96	-1.16	8.76	7.28	B
a11.h8	-23.57	-28.29	-28.18	-26.51	-25.07	-1.25	-28.18	-19.19	-3.43	-6.45	M-V
a11.h9	-7.37	-12.54	-11.38	-8.26	-6.81	20.03	-11.38	-1.71	15.71	12.97	B
a2.h2	-0.23	-0.20	-0.23	0.03	-0.48	0.65	-0.23	0.19	0.54	0.41	OBJ-V
a2.h5	-0.89	-1.86	-1.78	-0.98	-0.06	1.10	-1.78	-0.92	1.11	0.36	E-OBJ
a2.h8	-0.54	0.02	-0.20	0.30	-0.49	0.18	-0.20	-0.42	0.58	-0.04	V
a3.h3	-0.67	-0.88	-0.89	-0.56	-0.65	0.98	-0.89	0.00	0.52	0.46	B
a4.h0	-0.72	-0.76	-0.82	-0.14	-0.87	1.09	-0.82	0.49	1.04	0.61	B-V
a4.h11	-0.24	-0.98	-1.17	-0.59	-0.65	2.89	-1.17	0.68	2.54	2.00	OBJ
a4.h3	-0.98	-0.98	-1.15	0.07	-0.40	1.34	-1.15	-0.16	1.76	1.03	OBJ
a5.h6	-0.46	-0.62	-0.54	-0.41	-0.45	0.72	-0.54	-0.23	0.53	0.34	B
a5.h9	1.27	-1.13	-1.17	-1.16	-0.23	1.10	-1.17	-1.08	0.53	0.34	B
a6.h0	-0.40	-0.66	-0.75	-0.43	-0.70	1.10	-0.75	0.38	0.95	0.54	B-V
a6.h8	0.07	-0.57	-0.62	-0.13	-0.38	0.99	-0.62	0.02	1.65	1.01	B-OBJ-V
a7.h0	-0.28	-0.29	-0.30	-0.09	-0.16	1.02	-0.30	0.06	0.81	0.50	B
a8.h4	-0.25	-0.35	-0.41	0.08	-0.24	0.99	-0.41	0.40	0.33	0.27	B
a8.h5	-1.50	-1.40	-1.40	-0.29	-0.49	3.85	-1.40	1.26	3.82	2.98	B-V
a8.h6	-3.01	-0.60	0.21	-3.85	-5.01	8.66	0.21	-0.03	7.57	6.41	S2
a9.h7	-2.01	-1.36	-1.84	-1.99	-2.26	-0.84	-1.84	-1.37	-0.57	-0.54	B-S2
a0.h1	-1.03	-1.95	-2.12	-1.12	-0.77	-1.22	-2.12	-0.37	-0.36	-0.00	E
a10.h7	-4.35	-13.41	-13.70	-4.98	-2.98	1.73	-13.70	-3.33	1.44	0.93	IO-S1-S2
a11.h10	-2.45	-14.92	-13.98	-3.46	-1.29	5.49	-13.98	-0.92	3.97	3.71	IO-S2
a7.h9	-0.45	-1.25	-3.26	0.01	0.38	0.70	-3.26	0.98	1.18	1.21	S2
a8.h10	-1.75	-4.24	-6.95	-2.12	-1.64	-3.27	-6.95	-3.16	-3.28	-2.94	B-S2
a10.h0	0.48	25.90	14.69	1.26	0.55	-1.79	14.69	-1.27	-1.38	-1.09	IO-S1
a9.h6	0.50	19.86	11.96	0.77	-0.92	1.34	11.96	0.38	1.48	1.04	IO-S1
a9.h9	0.10	40.99	8.59	0.36	0.16	2.35	8.59	-0.63	1.52	1.72	IO
a1.h1	-1.38	-0.75	-0.66	-0.32	0.37	-0.70	-0.66	-0.90	-0.51	-0.89	E-M
a11.h3	-0.29	5.66	4.62	1.53	7.21	3.71	4.62	-0.02	2.98	2.75	B-M
a7.h3	-0.02	-0.80	-0.85	-1.14	0.23	-0.88	-0.85	-2.69	-0.87	-0.76	S2-V

Figure 29: Original Logit Clusters

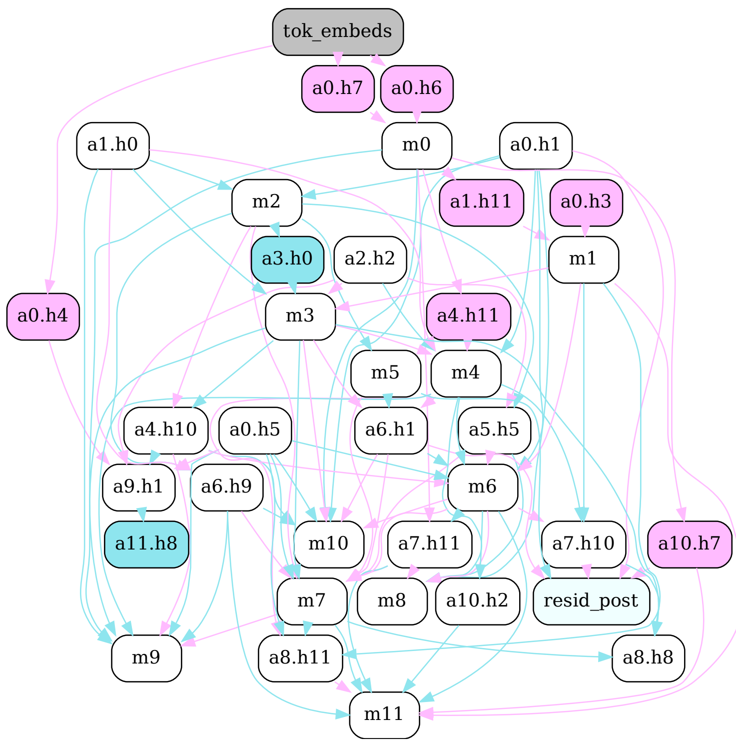

Figure 30: Circuit difference for GT task: LoRA vs Full. Blue indicates LoRA-specific components, red indicates Full-specific components, and white represents shared components.

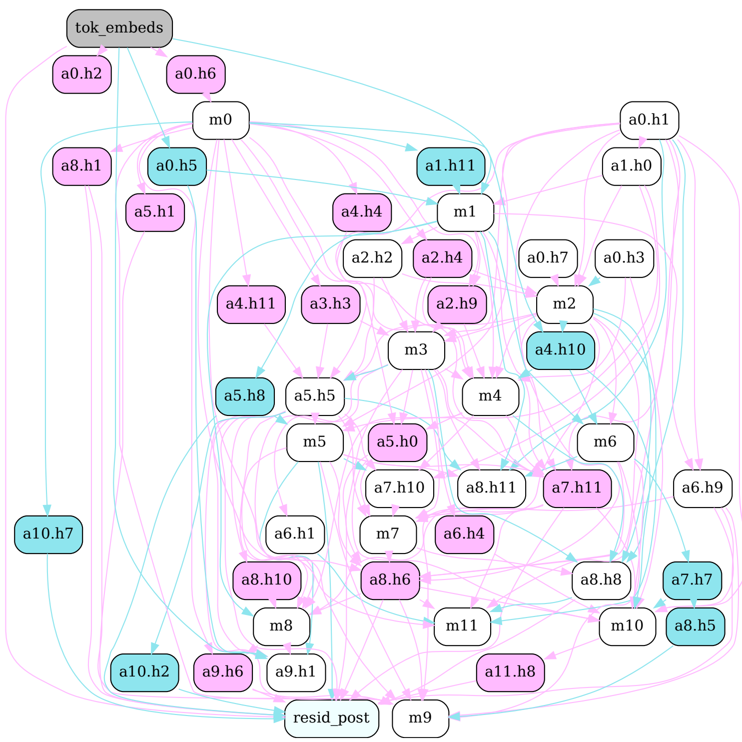

Figure 31: Circuit difference for GT task: Original vs AdaLoRA. Blue indicates Original-specific components, red indicates AdaLoRA-specific components, and white represents shared components.

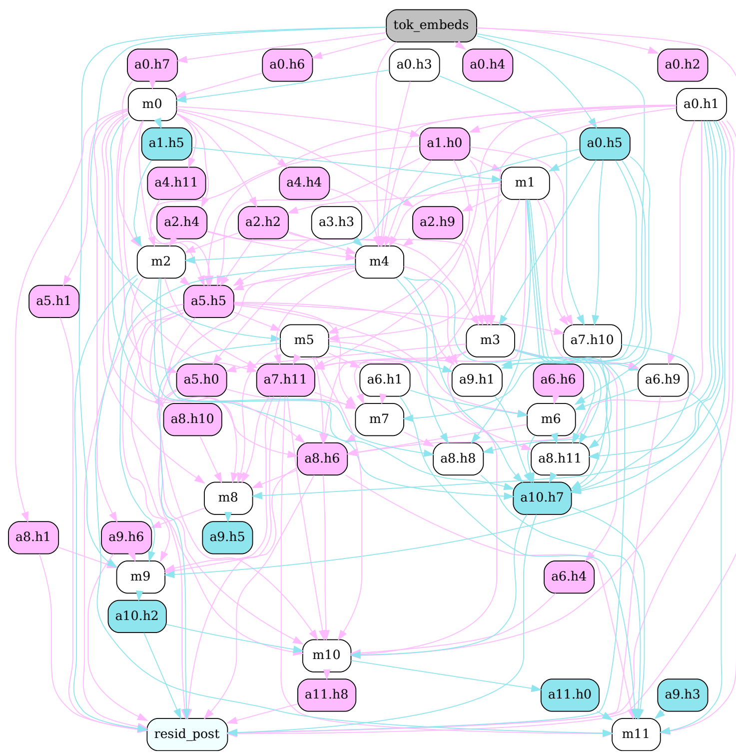

Figure 32: Circuit difference for GT task: Original vs BitFit. Blue indicates Original-specific components, red indicates BitFit-specific components, and white represents shared components.

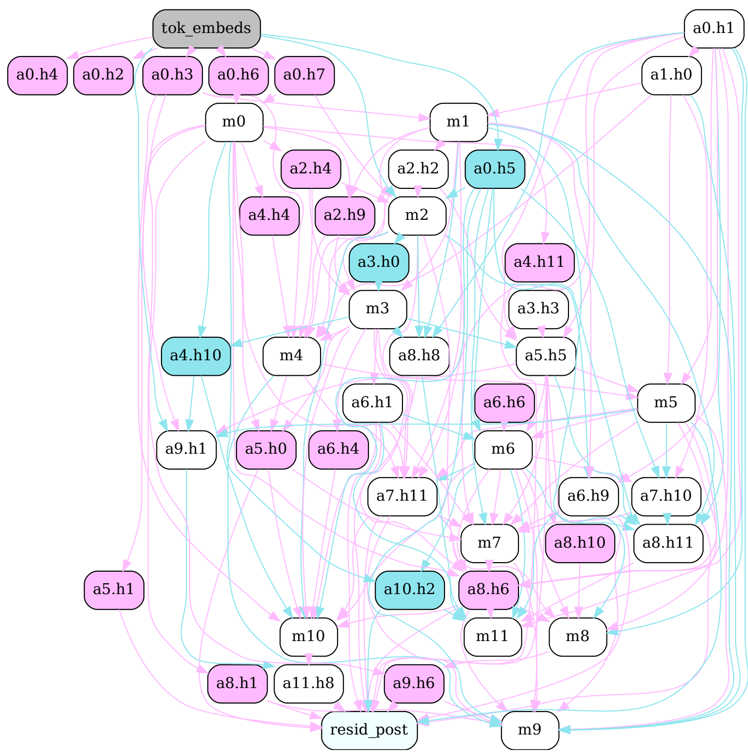

Figure 33: Circuit difference for GT task: Original vs Full. Blue indicates Original-specific components, red indicates Full-specific components, and white represents shared components.

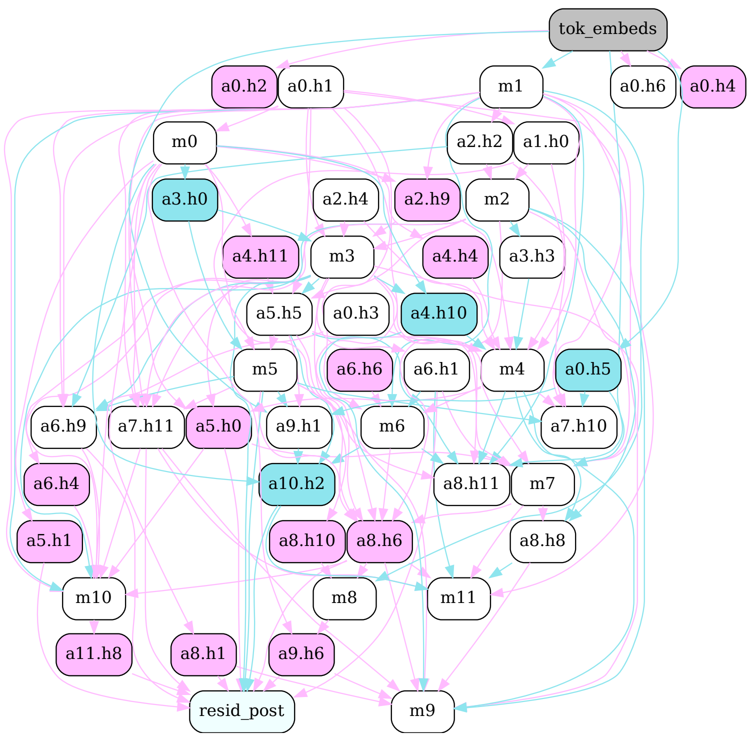

Figure 34: Circuit difference for GT task: Original vs IA3. Blue indicates Original-specific components, red indicates IA3-specific components, and white represents shared components.

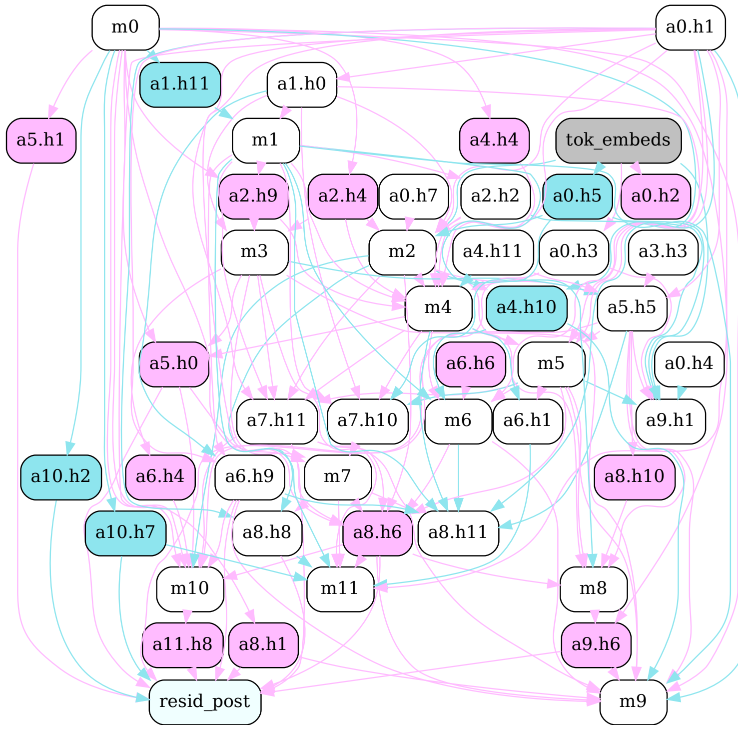

Figure 35: Circuit difference for GT task: Original vs LoRA. Blue indicates Original-specific components, red indicates LoRA-specific components, and white represents shared components.

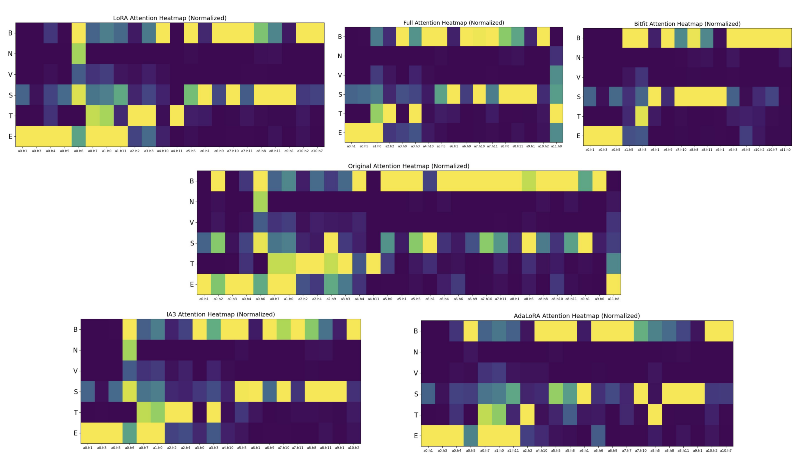

Figure 36: Attention grouping results for all finetuning methods circuit on GT.