Circuit-tuning: A Mechanistic Approach for Identifying Parameter Redundancy and Fine-tuning Neural Networks

Anonymous Authors[1]

Abstract

The study of mechanistic interpretability aims to reverse-engineer a model to explain its behaviors. While recent studies have focused on the static mechanism of a certain behavior, the training dynamics inside a model remain to be explored. In this work, we develop an interpretable method for fine-tuning and reveal the mechanism behind learning. We first propose the concept of node redundancy as an extension of intrinsic dimension and explain the idea behind circuit discovery from a fresh view. Based on the theory, we propose circuit-tuning, a two-stage algorithm that iteratively performs circuit discovery to mask out irrelevant edges and updates the remaining parameters responsible for a specific task. Experiments show that our method not only improves performance on a wide range of tasks but is also scalable while preserving general capabilities. We visualize and analyze the circuits before, during, and after fine-tuning, providing new insights into the self-organization mechanism of a neural network in the learning process.

1. Introduction

Benefiting from the high efficiency in computation and the scalable architecture of Transformer (Vaswani, 2017), large language models (LLMs) have demonstrated outstanding performance in a wide range of tasks. For real-world applications, fine-tuning aims to adapt a model to the downstream tasks. With the concept of intrinsic dimension (Li et al., 2018; Aghajanyan et al., 2020) that the parameters in a model are often redundant for solving a specific problem, parameter-efficient fine-tuning is proposed and has reduced computation to a great extent (Han et al., 2024). However, these techniques lack a mechanism fully clear to humans, and thus are less stable and may lead to catastrophic forgetting when applied to a new task (Wang et al., 2024). Besides, with the rapid development of LLMs, AI alignment has become an important issue in machine learning, highlighting the need for interpretability as well (Ji et al., 2023).

To discover the underlying mechanisms inside a model, (Elhage et al., 2021) provided a mathematical framework for transformer circuits, showing the potential of reverse-engineering a model under the guidance of mechanistic interpretability (Olah et al., 2020). Recent studies in this field have tried to isolate the circuit responsible for a single behavior (Wang et al., 2022), extract features using sparse autoencoders (Bricken et al., 2023), or apply a steering vector (Turner et al., 2023) to modify the model behaviors. While the mechanism of an existing model behavior is explored, few studies have paid attention to the mechanism in learning dynamics. Though several efforts have been made from various aspects (Olsson et al., 2022; Nanda et al., 2023), many issues are yet to be further explored in fine-tuning, e.g., how a new capability is acquired, how the model components interact and reorganize themselves, or how to intervene in the training process.

In this work, we develop an interpretable method for fine-tuning and reveal the mechanism behind learning. Our contributions are as follows:

- We formulate different kinds of redundancy and propose the concept of node redundancy as an extension of the intrinsic dimension to the node level. Based on the theory, we provide an explanation for circuit discovery from the view of redundancy detection.
- We propose circuit-tuning, an algorithm that iteratively finds out the components necessary for a task and performs fine-tuning in a mechanistic way. Our method not only performs well but is also scalable to various models and tasks while preserving general capabilities.
- We provide a comprehensive analysis on training dynamics based on circuit-tuning and present some amazing findings for the self-organization inside a model, providing new insights and improvements for both fine-tuning and interpretability.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

2. Background and Preliminaries

2.1. Features and Representations

Definition 2.1. (Feature Vector) Given a representation space $\mathbb{V}^D$ with a set of bases $\mathbf{E} = \{\mathbf{e_1}, \mathbf{e_2}, ..., \mathbf{e_D}\}$ where each basis e_i corresponds to a neuron, the feature vector $\mathbf{v_f}$ of feature f is a direction in $\mathbb{V}^D$, which is:

$$\mathbf{v_f} = \sum_{i=1}^{m} c_i \mathbf{e_i} = c_1\mathbf{e_1} + c_2\mathbf{e_2} + ... + c_m\mathbf{e_m}$$

where the coefficient c_i satisfies $|ci| \leq 1$ and $\sum_{i=1}^{m} ci^2 = 1$.

Remark 2.2. The idea of Definition 2.1 follows the linear representation hypothesis that features are represented as vectors in a linear space, the evidence of which has been widely discussed in (Mikolov et al., 2013; Bricken et al., 2023; Park et al., 2024).

Assumption 2.3. (Superposition) Given a neural network φ and a representation space $\mathbb{V}^D$. If all features represented in φ is $\mathcal{F}$, then we have $D < |\mathcal{F}|$, where $|\mathcal{F}|$ is the number of elements in $\mathcal{F}$.

Remark 2.4. The idea comes from the superposition hypothesis discussed in (Olah et al., 2020; Elhage et al., 2022) which explains the phenomenon of polysemanticity that one neuron in a neural network often responds to multiple unrelated inputs. Since there are much more features than the dimensions in $\mathbb{V}^D$, a feature f may not be aligned with a single basis. In fact, it is akin to the strategy of population coding (Pouget et al., 2000) in human brains that information is encoded by clusters of cells.

Considering the settings of Definition 2.1 and Assumption 2.3, we can have the conclusion: Given a task T and an input x with its features $\mathcal{F}_x = \{f_1, f_2, ..., f_t\} \subseteq \mathcal{F}_T$. Suppose the feature vectors corresponding to the features in x are $\mathbf{V_x} = \{\mathbf{v_{f_1}}, \mathbf{v_{f_2}}, ..., \mathbf{v_{f_t}}\}$ and the activations of $\mathcal{F}_x$ are $A_x = \{a_{f_1}, a_{f_2}, ..., a_{f_t}\}$, then the magnitude a_i of $\mathbf{e_i}$ in the representation space $\mathbb{V}^D$ can be written as:

$$a_i = \left(\sum_{i=1}^{t} a_{f_i} \mathbf{v_{f_i}}\right) \cdot \mathbf{e_i} \tag{1}$$

This means the magnitude of dimension i in $\mathbb{V}^D$ often originates from more than one features.

2.2. Computational Graph

If we view a model as a directed acyclic graph (DAG), then the nodes are terms in its forward pass (neurons, attention heads, etc.) and the edges are the interactions between the nodes. Note that the definitions of the node and edge do not necessarily follow the structure of the model. The shape of a node depends on the level of granularity when we inspect a model, and an edge can be a virtual connection between nodes far apart from each other.

2.3. Circuit Discovery

A circuit is a subgraph in a computational graph that is responsible for a certain behavior. Recent studies generally use causal intervention for circuit discovery. (Meng et al., 2023) proposed activation patching to identify activations in a model relevant to the output, while (Nanda, 2023) proposed attribution patching to accelerate it. (Wang et al., 2022; Conmy et al., 2023) focused on edges and proposed path patching. Others optimized this technique from various aspects (Syed et al., 2023; Kramár et al., 2024).

2.4. Intrinsic Dimension

(Li et al., 2018) first proposed the concept of intrinsic dimension to describe the minimum number of dimensions needed for solving a specific problem. (Aghajanyan et al., 2020) provided an analysis into fine-tuning with this concept and proved the feasibility of effective fine-tuning. Inspired by previous works, (Hu et al., 2021) proposed LoRA as a parameter-efficient fine-tuning method, which is of great significance for fine-tuning large language models.

3. The Theory of Node Redundancy

In this section, we introduce the concept of node redundancy as an extension of the intrinsic dimension, together with a method for detecting it, which can also be used to explain the idea behind circuit discovery from a new perspective.

Since the intrinsic dimension describes the redundancy in parameters, our goal is also to find out the redundant parameters in a model. If we inspect a representation $H = (h_1, h_2, ..., h_m)^\top \in \mathbb{V}^D (D = m)$, then we can denote the parameter matrix to examine as $W^{pre} \in \mathbb{R}^{m \times n}$, which maps an input from $\mathbb{R}^n$ to $\mathbb{R}^m$.

3.1. Representation Redundancy

Assumption 3.1. (Feature Redundancy) Suppose all features represented in a neural network is $\mathcal{F}$. Given a task T with $\mathcal{X} = \{x_1, x_2, ..., x_t\} \sim \mathcal{D}_T$ which consists of t samples that follows a specific data distribution $\mathcal{D}_T$, the features for representing $\mathcal{X}$ is $\mathcal{F}_T \subsetneqq \mathcal{F}$.

Remark 3.2. Assumption 3.1 means that when conditioned on a task T, the number of features needed is smaller than that of the elements in the universal set $\mathcal{F}$. The assumption follows the concept of feature sparsity from (Elhage et al., 2022) that many features do not frequently appear.

Definition 3.3. (Semantic Preservation) Consider a feature f with its feature vector $\mathbf{v_f} = \sum_{i=1}^{m} c_i \mathbf{e_i}$. If we set the coefficients in $\mathbf{v_f}$ which correspond to a set of bases $\mathbf{E_r}$ to zero, then we can get a new vector $\tilde{\mathbf{v}}_\mathbf{f}$. If the cosine similarity $cos(\mathbf{v_f}, \tilde{\mathbf{v}}_\mathbf{f}) = \frac{\mathbf{v_f} \cdot \tilde{\mathbf{v}}_\mathbf{f}}{||\mathbf{v_f}|| \cdot ||\tilde{\mathbf{v}}_\mathbf{f}||} > \eta$ $(\eta \in (0, 1])$, then we say $\tilde{\mathbf{v}}_\mathbf{f}$ has preserved the semantic information in f.

Remark 3.4. It is a nice property akin to the robustness of population encoding to noise. Since information is encoded across many cells, a perturbation on a few cells will not destroy the representation (Pouget et al., 2000).

Definition 3.5. (Dimension Redundancy) Given a task T with a number of features $\mathcal{F}_T$. Let's consider a single basis $\mathbf{e_i}$ in the representation space $\mathbb{V}^m$. For any feature f in $\mathcal{F}_T$, we set the coefficient c_i in $\mathbf{v_f}$ to zero and get the deformed feature vector $\tilde{\mathbf{v_f}}$. If $\mathbf{e_i}$ satisfies:

$$min[cos(\mathbf{v_{f_1}}, \tilde{\mathbf{v_{f_1}}}), ..., cos(\mathbf{v_{f_{|\mathcal{F}_T|}}}, \tilde{\mathbf{v_{f_{|\mathcal{F}_T|}}}})] > \eta$$

then $\mathbf{e_i}$ is a redundant dimension for representing features in task T. Note that $\eta \in (0, 1]$ serves as the lower bound for judging whether $\tilde{\mathbf{v_f}}$ preserves the information in f.

Proposition 3.6. *Consider the setup of Definition 3.5. If* $\mathbf{e_i}$ *satisfies:*

$$max(|\mathbf{v_{f_1}} \mathbf{e_i}|, |\mathbf{v_{f_2}} \mathbf{e_i}|, ..., |\mathbf{v_{f_{|\mathcal{F}_T|}}} \mathbf{e_i}|) < \xi$$

where $\xi = \sqrt{1 - \eta^2}$*, then we say* $\mathbf{e_i}$ *is redundant for representing features in task* T*.*

Remark 3.7. This is obvious from Definition 3.5. Note that $|\mathbf{v_{f_j}} \mathbf{e_i}|$ is the projection of $\mathbf{v_{f_j}}$ on direction $\mathbf{e_i}$. Higher $|\mathbf{v_{f_j}} \mathbf{e_i}|$ means higher dependency of $\mathbf{v_{f_j}}$ on $\mathbf{e_i}$. The redundancy probability P_r of a single basis $\mathbf{e_i}$ can be written as:

$$P_r = \prod_{j=1}^{|\mathcal{F}_T|} P(|\mathbf{v_{f_j}} \mathbf{e_i}| < \xi)$$

The equation tells us that the probability a basis $\mathbf{e_i}$ is redundant will decrease with the increasing number of features. On the contrary, the fewer features related to a specific task, the higher the likelihood that a certain basis is redundant, leading to more redundant dimensions in the representation.

Definition 3.8. (Representation Redundancy) Given a specific task T with features $\mathcal{F}_T$, we choose a set of bases $\mathbf{E_r} \subset \mathbf{E}$. For each feature $f \in \mathcal{F}_T$, we set the coefficients in its feature vector $\mathbf{v_f}$ which correspond to the bases in $\mathbf{E_r}$ to zero and get the deformed feature vector $\tilde{\mathbf{v_f}}$. Given a threshold τ_r as the tolerance of deformation. If $\mathbf{E_r}$ satisfies:

$$\sum_{j=1}^{|\mathcal{F}_T|} max[(\eta - cos(\mathbf{v_f}, \tilde{\mathbf{v_f}})), 0] < \tau_r$$

then $\mathbf{E_r}$ is a set of redundant bases for representing features $\mathcal{F}_T$, and the representation redundancy in $\mathbb{R}^m$ is defined under the choice of $\mathbf{E_r}$. If $|\mathbf{E_r}| = r$, then the value of representation redundancy is equal to r.

Remark 3.9. The definition of representation redundancy is actually an extension of Definition 3.5 from one-dimension to multi-dimension. The difference $\eta - cos(\mathbf{v_f}, \tilde{\mathbf{v_f}})$ is used to measure the degree of deformation of feature vectors. If the total deformation in feature vectors is accepted under the threshold τ_r, then the influence from the removal of $\mathbf{E_r}$ on the semantic representation of $\mathcal{F}_T$ can be ignored, and the number of useful dimensions for representing $\mathcal{F}_T$ is $m - r$.

3.2. Forward Redundancy

If the representation $H = (h_1, h_2, ..., h_m)^\top$ is followed by another weight matrix $W^{post} \in \mathbb{R}^{l \times m}$, then the following representation is $H' = W^{post} H \in \mathbb{V}^{D'} (D' = l)$. To investigate into the influence of H on H', we split H' into the sum of influences from the dimensions in H:

$$W^{post} H = \sum_{i=1}^{m} W^{post}_{:,i} h_i \tag{2}$$

From (2), we find that the influence from h_i to H' is subject to $W^{post}_{:,i}$. If the value of $W^{post}_{:,i}$ is small, then the influence from h_i will be weakened, leading to a reduced impact on the final result in the follow-up calculations.

Definition 3.10. (Forward Redundancy) Given a representation H and the follow-up parameters W^{post}, we inspect the i-th dimension in H. Given a lower bound $\tau_f < 0$ for the influence of H in the forward propagation.If

$$||W^{post}_{:,i} h_i|| < \tau_f$$

then we say the dimension i in $\mathbb{R}^m$ is redundant in the forward propagation. If the number of redundant dimensions under this condition is r, then the value of forward redundancy is equal to r.

3.3. Node Redundancy

3.3.1. The definition of node redundancy

The discussions in previous sections focus on the real computations inside a model. In this section, we regard the whole model as a DAG composed of nodes and edges. The node could be a single neuron or an attention head in Transformer, based on the level of granularity.

To determine whether a node is redundant or not, a method is needed for measuring the importance of that node. Inspired by the idea of causal intervention in the area of causal inference, we provide the following definition:

Definition 3.11. (Node Redundancy) Given a model M and its computational graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in which $\mathcal{V}$ and $\mathcal{E}$ represent the nodes and edges in $\mathcal{G}$ respectively, consider a node $n \in \mathcal{V}$. For task T, a batch of inputs $\mathcal{X} = \{x_i\}_{i=1}^{p} \sim \mathcal{D}_T$ is fed into the model and the activation of n is saved.

For each input x_i, if we replace $n(x_i)$ which is the value of node n with another value $n(x_i')$ corresponding to the corrupted input x_i' while keeping other activations in the forward propagation unchanged, we can get another output $M(x_i')$. Given a metric f for measuring the difference between two output distributions caused by n and a threshold τ, we can calculate the contribution $c_i(n)$ of the node n as:

$$c_i(n) = f(n; M(x_i), M(x_i'))$$

If the contribution of n satisfies:

$$\mathbb{E}_{x_i \sim \mathcal{X}}\big[|c_i(n)|\big] < \tau$$

then n is a redundant node in $\mathcal{G}$. If the redundant nodes in terms of task T in model M is $\mathcal{N}$, then the value of the intrinsic dimension at the node level is $|\mathcal{V} - \mathcal{N}|$.

3.3.2. The detection of node redundancy

With the idea of intervention from causal inference, we adopt the concept of indirect effect (Pearl, 2001) to estimate the contribution of a node n to the output as:

$$IE(n; x) = \mathcal{L}_m\big[M\big(x \,|\, do\big(n \leftarrow n(x')\big)\big)\big] - \mathcal{L}_m\big[M(x)\big]$$

in which IE is the indirect effect from node n to the final output of the model, and $\mathcal{L}_m$ is a metric for measuring the final output. We use the do-calculus notation (Neuberg, 2003) to express the intervention behavior, which is also called *patching*. The metric IE measures the effect of an intermediate variable in a causal graph to the final behavior of the model, which is widely used in causal inference. If $\mathcal{L}_m$ is viewed as a function of n, we can apply a first-order Taylor expansion to IE at $n = n(x')$ for approximation. Thus $IE(n; x)$ can be simplified as

$$\begin{aligned}&\mathcal{L}_m[M(x)] + [n(x') - n(x)]^\top \nabla_n \mathcal{L}_m[M(x)] - \mathcal{L}_m[M(x)] \\ =&[n(x') - n(x)]^\top \nabla_n \mathcal{L}_m[M(x)]\big|_{n(x)}\end{aligned}$$

If zero ablation is used which means $n(x')$ is set to zero, then the contribution of node n can be written as:

$$c(n) \approx \mathbb{E}_{x_i \sim \mathcal{X}}\big[n \nabla_n \mathcal{L}_m[M(x)]\big] \tag{3}$$

The IE in equation (3) consists of two parts: (i) n the value of node n on the clean input x and (ii) $\nabla_n \mathcal{L}_m[M(x)]$ the derivatives of the patching metric $\mathcal{L}_m$ with respect to node n. The two parts correspond to the **representation redundancy** and the **forward redundancy** respectively. To gain an intuitive understanding of this, let's consider a representation in $\mathbb{V}^m$ which consists of a set of nodes $\mathcal{N} = \{n_1, n_2, ..., n_m\} \subset \mathcal{V}$ at the neuron level and then focus on the node n_i:

- The first part of (3) directly serves as an indicator for representation redundancy since the magnitude of a neuron is a good indicator for dimension redundancy. According to (1), the smaller the value of a neuron is, the higher the likelihood that the neuron corresponds to a redundant dimension, since the activation of a feature f on a basis $\mathbf{e_i}$ is directly proportional to the projection of its feature vector $\mathbf{v_f}$ on $\mathbf{e_i}$.

  One thing to note is the case where two features f_1 and f_2 are almost in opposite directions and the activations in the shared direction may cancel out each other, which results in a situation that the value of a neuron seems to be small though both of the two features fire on it. While this case may be tricky, Elhage et al. (2022) show that models prefer to represent anti-correlated features in opposite directions, which means f_1 and f_2 may never co-occur at the same time, so the probability of the above case occurring is very small.

- As for the second part, according to the chain rule during the back propagation stage, it can be written as:

  $$\nabla_{n_i} \mathcal{L}_m = \sum_{j=1}^{l} \nabla_{n_j} \mathcal{L}_m \cdot \nabla_{n_i} n_j$$

  where the node n_j in the follow-up vector space $\mathbb{V}^l$ is

  $$n_j = \sum_{i=1}^{m} w_{ji}^{post} n_i$$

  Thus

  $$\nabla_{n_i} \mathcal{L}_m = \sum_{j=1}^{l} \nabla_{n_j} \mathcal{L}_m \cdot w_{ji}^{post} \tag{4}$$

  From equation (4), it can be seen that the second part in (3) is closely related to the follow-up calculations as discussed before in Section 3.2, hence serving as an indicator for forward redundancy.

As discussed above, we can see that the metric for judging the redundancy of a node is a combination of the representation redundancy and the forward redundancy. Therefore the essence of the intrinsic dimension at the node level can be decomposed and understood from the above two aspects.

Note that Equation (3) is also adopted in circuit discovery, e.g., attribution patching (Nanda, 2023). Thus the process of circuit discovery can be viewed as a process of identifying the two types of redundancy and finding out the intrinsic dimensions in terms of a task. Each of the intrinsic dimensions corresponds to a node in the computational graph, and the number of the nodes in the circuit is expected to be equal to the dimensionality of the solution space. Given a specific task, the co-occurrence of the representation redundancy and the forward redundancy is the necessary and sufficient condition of the node redundancy in the graph.

4. Main Method

In this section, we introduce circuit-tuning, an algorithm based on the theory of node redundancy for both fine-tuning neural networks and analyzing training dynamics.

4.1. Circuit-tuning

4.1.1. From Nodes to Edges

The definition of node redundancy as well as the detection of it is provided in Section 3.3. Since the nodes inside a graph are connected by edges, we can also study the node redundancy from the view of edges.

Algorithm 1 Circuit-tuning

Input: dataset $\mathcal{X}$, model M, loss function $\mathcal{L}$,
the metric for measuring indirect effect $\mathcal{L}_m$,
the number of edges to save N,
the optimization steps after circuit discovery K.
Initialize the computational graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ from M,
the 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**
    Reset: $\mathcal{C} \leftarrow \mathcal{G}$
    for edge $e : n_1 \rightarrow n_2 \in \mathcal{E}$ **do**
      Intervene: $n_2(x') = n_2^{direct}\big(x|do\big(n_1 \leftarrow n_1(x')\big)\big)$
      Edge contribution: $c(e) = \mathbb{E}_{x_i \sim \mathcal{X}_T}\big[|IE(e; x_i)|\big]$
    end for
    $\mathcal{E}_T \leftarrow \{e \,|\, e \in \mathcal{E}$ with top-N edge contributions$\}$
    $\mathcal{V}_T \leftarrow \{n \,|\, n \in \mathcal{V} \wedge n$ is incident to an edge $e \in \mathcal{E}_T\}$
    $\mathcal{C} \leftarrow (\mathcal{V}_T, \mathcal{E}_T)$
  end if
  Update the parameters in $\mathcal{C}$
  $i = i + 1$
end for

Following the definition of the attribution score in edge attribution patching (EAP) (Syed et al., 2023) and the discussions in Section 3.3, we can define the contribution of an edge e to the output as follows:

$$\begin{aligned} c(e) &= \mathbb{E}_{x_i \sim \mathcal{X}}\big[|c_i(e)|\big] \\ &= \mathbb{E}_{x_i \sim \mathcal{X}}\big[|IE(e; x_i)|\big] \end{aligned} \tag{5}$$

where the edge $e : n_1 \rightarrow n_2$ is connected with an upstream node n_1 and a downstream node n_2. To measure the direct effect from n_1 to n_2, we set the value of n_1 to another value $n_1(x')$ while keeping all the other nodes between n_1 and n_2 unchanged. Then the affected value of n_2 is:

$$n_2(x') = n_2(x) - n_2^{n_1}(x) + n_2^{n_1}(x')$$

where $n_2^{n_1}(\cdot)$ denotes the contribution from n_1 to n_2, and

$$n_2^{n_1}(x') = n_2^{n_1}\big(x \,|\, do\big(n_1 \leftarrow n_1(x')\big)\big)$$

Then the indirect effect from e to the final output is

$$\begin{aligned} IE(e; x) =& IE(n_1 \rightarrow n_2; x) \\ =& \mathcal{L}_m\big[M\big(x \,|\, do\big(n_2 \leftarrow n_2(x')\big)\big)\big] - \mathcal{L}_m\big[M(x)\big] \end{aligned}$$

To reduce computation, we apply a first-order Taylor expansion to n_1' and n_2' respectively. Then the calculation of edge contribution requires only one forward and one backward pass. See Appendix A for derivation details.

Therefore the approximation of contribution $c(e)$ is available which can be used to measure the importance of an edge. We can determine the redundancy of a node by checking the contributions of the edges connected to it.

4.1.2. The Circuit-tuning Algorithm

Given a model M for fine-tuning, the model is initialized into a computational graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in which $\mathcal{V}$ and $\mathcal{E}$ represent the nodes and edges in $\mathcal{G}$ respectively. A metric $\mathcal{L}_m$ and a loss function $\mathcal{L}$ are used for circuit discovery and parameter optimization respectively. Given a dataset $\mathcal{X} \sim \mathcal{D}_T$ for the fine-tuning task T, circuit-tuning alternately performs the following two procedures:

Circuit Discovery For a batch of data $\mathcal{X}_T \in \mathcal{X}$, we calculate the contribution of each edge $e \in \mathcal{E}$ according to (5). Then all the edges are sorted in descending order based on their contributions, and the edges with top N contributions are selected. Then the graph $\mathcal{G}$ is pruned into a circuit $\mathcal{C} = (\mathcal{V}_T, \mathcal{E}_T)$ with only the selected edges $\mathcal{E}_T$ together with the nodes $\mathcal{V}_T$ at both ends of each edge inside.

Circuit-tuning For a batch of data $\mathcal{X}_T \in \mathcal{X}$, all the parameters outside $\mathcal{C}$ are frozen, and only the parameters corresponding to the nodes $\mathcal{V}_T$ inside $\mathcal{C}$ are updated through a forward pass and a backward pass as usual. After K steps of optimization, all the frozen parameters are freed and the graph $\mathcal{G}$ is reset to its original state.

The full process is shown in Algorithm 1. The parameter to optimize corresponds to the parameter matrix that maps an input to the activation of a node in the circuit. For example, the parameter corresponding to the node at the attention output refers to W_O, which is the output projection matrix.

4.2. Characteristics of Our Method

Capability Acquisition Different from recent studies that focus on the circuits of certain model behaviors, circuit-tuning aims to automatically find the relevant components responsible for a specific task and guide the model to develop an ability through parameter optimization. The dynamics of this process will be verified in Section 5.1.

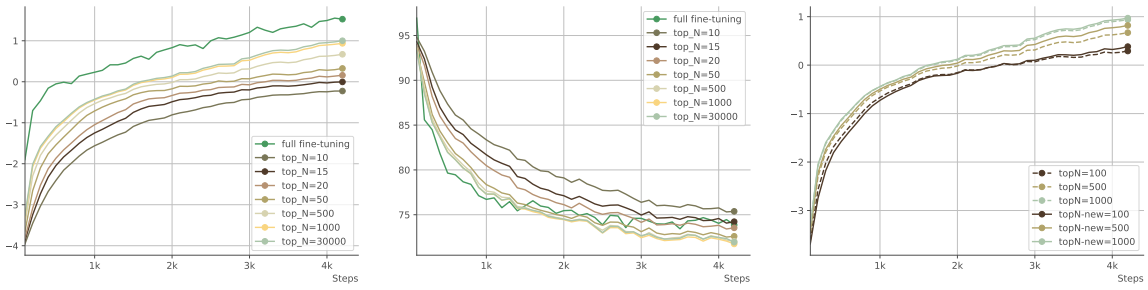
Precise Fine-tuning Compared with full fine-tuning and LoRA-based methods, our method only update a few parameters instead of taking all of the parameters into account. Since our method tries to leave out the irrelevant structures in terms of a target task, the general abilities are expected not to be affected, which will be verified in Section 5.

4.3. Convergence Analysis

To demonstrate the stability of circuit-tuning, we provide an analysis of the training convergence. Specifically, let's 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.



(a) logit difference (b) PPL (c) logit difference after revision

Figure 1. Results in the subject-verb disagreement task

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

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

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 will be more stable compared with full fine-tuning. The evidence for this conclusion will be discussed in Section 5.1.3.

5. Experiments

In this section, we test our method across a variety of models and tasks. To comprehensively investigate the performance as well as the training dynamics during fine-tuning, we arrange the experiments into two parts. The first part serves as the verification and analysis of our method, while the second part aims to show that our method is scalable to various model sizes and tasks.

5.1. The Subject-verb Disagreement Task

5.1.1. Task Description and Data Preparation

The goal of the subject-verb disagreement task is to match a verb with a subject in an abnormal way. For example, "I is", "he are" and "the cows eats" are all expected results for this task. In each sentence, the word before the verb is called the END token. The automatic evaluation metric for this task is the logit difference between the flipped verb and the original verb at the END token. This is because the logit at the END token is directly used for predicting the verb.

The reason why we create this task as the start of our experiments is that we expect the model to acquire a new capability from scratch. Besides, the task is simple and interesting for analyzing the training dynamics during fine-tuning.

For training data, we use the first 10k samples in the Pile (Gao et al., 2020) corpus. We extract 30k sentences in the present tense in English and flip the forms of the verbs in each sentence. For details, please refer to Appendix C.1.

5.1.2. Implementation Details

We use GPT2-small (Radford et al., 2019) in 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. During training, we use the logit difference discussed before as the metric $\mathcal{L}_m$ and follow Syed et al. (2023) to calculate the edge contributions. We sweep over a range of N which is the number of edges to be saved during circuit discovery. Mini-batch SGD is used for optimization. For details of settings, please refer to Appendix C.2.

5.1.3. Main Results

From Figure 1(a), we observe a flip in logit difference from negative to positive, which means the model adjusts its grammar to fit the data distribution of subject-verb disagreement. One noteworthy finding is that the trained model can generate abnormal texts in the past tense, such as "to be or not to be, that were a ...", which implies that the model really learns the new grammar and applies it smartly, since there is no sentence in the present tense in our training data at all.

We find that with the increasing number of top N edges, the logit difference gets bigger until N reaches around 1000. We check and find that the number of tunable parameters for each N is also saturated at this point. We believe when $N = 1000$, almost all the necessary parameters for this task are included, and the performance cannot be further improved, suggesting the existence of intrinsic dimension. For verification of this, please refer to Appendix C.3.

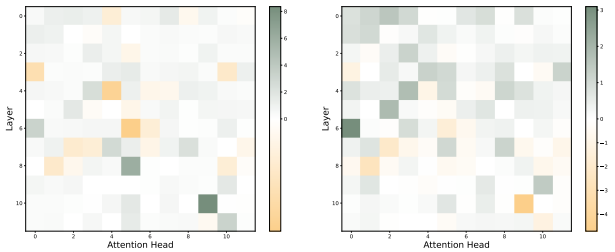
In Figure 1(b), we find that the perplexity (PPL) of full fine-tuning is high and fluctuates wildly during training, though the logit difference of it is higher. This observation implies better training stability as well as better preservation of general capabilities of circuit-tuning over full fine-tuning.

5.1.4. The Quality of the Discovered Circuit

To demonstrate that our method is able to find the required parameters accurately, we (i) calculate the faithfulness and completeness (Marks et al., 2024) of the circuits and (ii) provide an ablation study in which we randomly unfreeze the nodes outside the circuit to check their influence on the performance of this task. Results show from various aspects that the parameters required are found accurately. For details, please refer to Appendix C.3.

5.1.5. Analyses on Training Dynamics

We analyze the circuits before, during, and after training, and report some amazing findings that are worth studying.



(a) Before fine-tuning (b) After fine-tuning

Figure 2. The flip in the Subject Attribute Heads. Each square corresponds to the j-th attention head in layer i, in which i and j correspond to the vertical and horizontal axis respectively.

Interpretation of the Circuits To interpret the circuit into human-understandable structures, we follow (Wang et al., 2022) and decompose the circuit into heads of different functions. Firstly we find out the attention head that directly affects the output, which is called the Subject Attribute Head. Then we find out the head responsible for the localization of the subject, which is called the Subject Identification Head. Finally, we find out the Collaborative Heads that could affect the behaviors of the Subject Attribute Heads. For details of the processes above, please refer to Appendix C.4.1.

The Flip of the Subject Attribute Heads The Subject Attribute Heads are responsible for matching the subject and the verb. We visualize the heads before and after fine-tuning in Figure 2. Through comparison, we observe an obvious flip at head.10.9, which implies the reversal in its function from subject-verb agreement to disagreement. Other heads (3.0, 6.0, 4.4, 11.8, etc) also see flips with varying degrees, which serves as strong evidence for the self-adjustment inside the nodes.

The Sharing of the Subject Identification Heads The Subject Identification Heads attend heavily to the subject. One type of these heads attends to the END token (0.1, 0.3, etc), which is helpful to the cases like "they are"; the other type of heads (8.5, 10.5, etc) attends to the subject several tokens before, which is helpful to the cases like "the girl wearing a dress is". Both types of heads perform the same before and after fine-tuning, which means their functions are preserved and shared all the way down.

The Evidence of Hebbian Learning We visualize the circuits during the fine-tuning process in Figure 9, and observe a phenomenon akin to Hebbian learning (Do, 1949). We find that several edges are strengthened during training, just like the synapses between neurons can be strengthened after continuous stimulation. We regard this finding as evidence of the self-organization (Kohonen, 2012) inside the model to a new environment. See Appendix C.4.3 for details.

Due to limited space, all the above together with other interesting findings are discussed in detail in Appendix C.4.

5.1.6. Improvement on Attribution Patching

During the analyses, we notice that some heads fail to appear in the circuit. We assume that the attribution score in EAP fails to measure the importance of a node. This is because an important node n_a may be connected with many nodes through edges $\mathcal{E}_a$ with low contributions, while another less important node n_b may connect with only one node through an edge that is stronger than anyone in $\mathcal{E}_a$. As a result, the edges in $\mathcal{E}_a$ are discarded, and thus the importance of n_a is underestimated. To solve this problem, we revise the contribution $c(e)$ of an edge $e : n_i^u \to n_j^d$ as:

$$c(e)' = c(e) \cdot \sum_{k=1}^{N_{down}^i} c(n_i^u \to n_k^d) \cdot \sum_{k=1}^{N_{up}^j} c(n_k^u \to n_j^d)$$

The revision takes into account all the edges connected to the upstream node and the downstream node. The results in Figure 1(c) show that our improvement is effective without introducing too much computation. Details of the results can be found in Appendix C.4.5.

5.2. Application to Complex Tasks

5.2.1. Task Description

We prepare two types of tasks based on whether reasoning is involved, each of which contains two specific tasks.

For reasoning-based tasks, chain-of-thought (CoT) style responses are involved to solve a problem. We prepare a mathematical task and a logical reasoning task since (Sprague et al., 2024) demonstrated that CoT is effective only on tasks requiring mathematical, logical, or algorithmic reasoning.

For reasoning-free tasks, only the final answer or a signal token is required. We prepare a gender de-biasing task which requires the language model to develop an unbiased perspective on genders, and a reading comprehension task which requires only the keywords as the answer.

Table 1. Experiment results of circuit-tuning on complex tasks.

Methods & Tasks	Reasoning-based		Reasoning-free		Computation
	Mathematics	Logical Reasoning	Gender De-biasing	Reading Comprehension	
	Acc@1 (%) ↑	F1 (%) ↑	Prejudice Risk ↓	Exact Match / F1 (%) ↑	Avg param ratio
Llama-3.2-1B-it	40.71	20.35	0.555	39.68, 43.58	/
Llama-3.2-1B-it-full-tuning	**46.47**	26.89	0.533	36.73, 41.51	1.00
Llama-3.2-1B-it-lora	44.58	22.51	0.530	34.30, 39.11	1.79e-2
Llama-3.2-1B-it-circuit-tuning	45.56	**27.06**	**0.312**	**41.78, 45.45**	7.65e-2
Llama-3.2-3B-it	70.36	42.71	0.641	54.12, 47.98	/
Llama-3.2-3B-it-full-tuning	**75.44**	47.32	0.632	55.63, 50.10	1.00
Llama-3.2-3B-it-lora	73.54	46.27	0.638	54.73, 49.44	1.49e-2
Llama-3.2-3B-it-circuit-tuning	74.35	**47.59**	**0.417**	**56.58, 50.93**	8.57e-2
Llama-3.1-8B-it	76.19	46.41	0.651	58.92, 52.83	/
Llama-3.1-8B-it-full-tuning	**83.76**	49.53	0.640	59.60, 54.23	1.00
Llama-3.1-8B-it-lora	80.21	47.64	0.643	58.97, 53.10	1.03e-2
Llama-3.1-8B-it-circuit-tuning	82.97	**50.54**	**0.420**	**60.04, 55.00**	9.37e-2

As for the metric $\mathcal{L}_m$ for circuit discovery, for gender de-biasing, we use the logit difference between male attribution words like he/his and female attribution words like she/her. For other tasks, we simply set $\mathcal{L}_m$ as identical to the negative log probability loss for language modeling since the behaviors for completing the task are hidden inside multiple tokens and cannot be separated out easily.

As for the loss $\mathcal{L}$, we selectively add $\beta \cdot |\mathcal{L}_m|$ as a regularization term for gender de-biasing to improve performance.

Details of task settings can be found in Appendix D.1.

5.2.2. Dataset and Evaluation Metrics

Reasoning-based Tasks We use GSM8K (Cobbe et al., 2021) with zero-shot accuracy and Contexthub (Hua et al., 2024) with F1 score as datasets and metrics for mathematics and logical reasoning respectively.

Reasoning-free Tasks For the gender-debiasing task, we use BUG (Levy et al., 2021) for training and WinoBias (Zhao et al., 2018) for evaluation. We use the prejudice risk proposed in (Liu et al., 2024) as the evaluation metric. For the reading comprehension task, we use SQuAD 2.0 (Rajpurkar et al., 2018) with exact match and F1 score.

General Capabilities To check if other capabilities are preserved after training, we evaluate on benchmarks involving general abilities as well as reasoning, coding, and multilingual abilities. For details, please refer to Appendix D.3.

5.2.3. Implementation

We apply our method to llama-3.2-1B / 3B and llama-3.1-8B (Dubey et al., 2024), and compare it with full fine-tuning and LoRA. The graph settings are the same as before in Section 5.1.2, except that each MLP layer is split into 64-dimensional heads. The details are shown in Appendix D.2.

5.2.4. Main Results

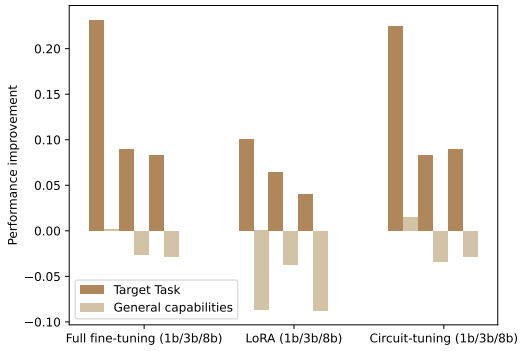


Figure 3. Comparison of different fine-tuning methods

Results of our method on complex tasks in Table 1 demonstrate that our method is capable of tasks in various scenarios. Note that with a regularization term as guidance, the de-biasing result is clearly better than others, which is further shown in Appendix D.4. In general, circuit-tuning is better than LoRA, and even surpasses full fine-tuning in many tasks with only a small portion of tunable parameters.

Evaluations on general capabilities are shown in Figure 3. The results are averaged over tasks. It can be seen that our method keeps comparable performance with full fine-tuning while preserving general capabilities better than others.

6. Conclusion

In conclusion, we propose the theory of node redundancy from the mechanistic view as an extension of the intrinsic dimension. With the guidance of the theory, we propose the algorithm of circuit-tuning, which not only shows great potential in fine-tuning but also serves as a useful tool for analyzing the training dynamics inside a model.

Impact Statement

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, we expect our work to be used under the guidance of human values.

References

Aghajanyan, A., Zettlemoyer, L., and Gupta, S. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255*, 2020.

Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N., Anil, C., Denison, C., Askell, A., Lasenby, R., Wu, Y., Kravec, S., Schiefer, N., Maxwell, T., Joseph, N., Hatfield-Dodds, Z., Tamkin, A., Nguyen, K., McLean, B., Burke, J. E., Hume, T., Carter, S., Henighan, T., and Olah, C. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. D. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

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

Do, H. The organization of behavior. *New York*, 1949.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., DasSarma, N., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., Grosse, R., McCandlish, S., Kaplan, J., Amodei, D., Wattenberg, M., and Olah, C. Toy models of superposition. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/toy_model/index.html.

Gallegos, I. O., Rossi, R. A., Barrow, J., Tanjim, M. M., Kim, S., Dernoncourt, F., Yu, T., Zhang, R., and Ahmed, N. K. Bias and fairness in large language models: A survey. *Computational Linguistics*, pp. 1–79, 2024.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Han, Z., Gao, C., Liu, J., Zhang, J., and Zhang, S. Q. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*, 2024.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Hua, W., Zhu, K., Li, L., Fan, L., Lin, S., Jin, M., Xue, H., Li, Z., Wang, J., and Zhang, Y. Disentangling logic: The role of context in large language model reasoning capabilities. *arXiv preprint arXiv:2406.02787*, 2024.

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

Ji, J., Qiu, T., Chen, B., Zhang, B., Lou, H., Wang, K., Duan, Y., He, Z., Zhou, J., Zhang, Z., et al. Ai alignment: A comprehensive survey. *arXiv preprint arXiv:2310.19852*, 2023.

Kohonen, T. *Self-organization and associative memory*, volume 8. Springer Science & Business Media, 2012.

Kramár, J., Lieberum, T., Shah, R., and Nanda, N. Atp*: An efficient and scalable method for localizing llm behaviour to components. *arXiv preprint arXiv:2403.00745*, 2024.

Levy, S., Lazar, K., and Stanovsky, G. Collecting a large-scale gender bias dataset for coreference resolution and machine translation. *arXiv preprint arXiv:2109.03858*, 2021.

Li, C., Farkhoor, H., Liu, R., and Yosinski, J. Measuring the intrinsic dimension of objective landscapes. *arXiv preprint arXiv:1804.08838*, 2018.

Liu, Y., Yang, K., Qi, Z., Liu, X., Yu, Y., and Zhai, C. Prejudice and volatility: A statistical framework for measuring social discrimination in large language models, 2024. URL https://arxiv.org/abs/2402.15481.

Marks, S., Rager, C., Michaud, E. J., Belinkov, Y., Bau, D., and Mueller, A. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. *arXiv preprint arXiv:2403.19647*, 2024.

Meng, K., Bau, D., Andonian, A., and Belinkov, Y. Locating and editing factual associations in gpt, 2023. URL https://arxiv.org/abs/2202.05262.

Mikolov, T., Yih, W.-t., and Zweig, G. Linguistic regularities in continuous space word representations. In Vanderwende, L., Daumé III, H., and Kirchhoff, K. (eds.), *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 746–751, Atlanta, Georgia, June 2013. Association for Computational Linguistics. URL https://aclanthology.org/N13-1090.

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

Nanda, N., Chan, L., Lieberum, T., Smith, J., and Steinhardt, J. Progress measures for grokking via mechanistic interpretability, 2023. URL https://arxiv.org/abs/2301.05217.

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

nostalgebraist. Interpreting gpt: The logit lens, 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., and Carter, S. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001. https://distill.pub/2020/circuits/zoom-in.

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Johnston, S., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

Park, K., Choe, Y. J., and Veitch, V. The linear representation hypothesis and the geometry of large language models, 2024. URL https://arxiv.org/abs/2311.03658.

Pearl, J. Direct and indirect effects. *Probabilistic and Causal Inference*, 2001. URL https://api.semanticscholar.org/CorpusID:5947965.

Pouget, A., Dayan, P., and Zemel, R. Information processing with population codes. *Nature Reviews Neuroscience*, 1 (2):125–132, 2000.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

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

Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. Gpqa: A graduate-level google-proof q&a benchmark. *ArXiv*, abs/2311.12022, 2023. URL https://api.semanticscholar.org/CorpusID:265295009.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Shatz, C. J. The developing brain. *Scientific American*, 267 (3):60–67, 1992.

Shi, F., Suzgun, M., Freitag, M., Wang, X., Srivats, S., Vosoughi, S., Chung, H. W., Tay, Y., Ruder, S., Zhou, D., et al. Language models are multilingual chain-of-thought reasoners. *arXiv preprint arXiv:2210.03057*, 2022.

Sprague, Z., Yin, F., Rodriguez, J. D., Jiang, D., Wadhwa, M., Singhal, P., Zhao, X., Ye, X., Mahowald, K., and Durrett, G. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning. *arXiv preprint arXiv:2409.12183*, 2024.

Syed, A., Rager, C., and Conmy, A. Attribution patching outperforms automated circuit discovery. *arXiv preprint arXiv:2310.10348*, 2023.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E.,

Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Turner, A. M., Thiergart, L., Leech, G., Udell, D., Vazquez, J. J., Mini, U., and MacDiarmid, M. Activation addition: Steering language models without optimization. *arXiv e-prints*, pp. arXiv–2308, 2023.

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

Wang, K., Variengien, A., Conmy, A., Shlegeris, B., and Steinhardt, J. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*, 2022.

Wang, L., Zhang, X., Su, H., and Zhu, J. A comprehensive survey of continual learning: Theory, method and application, 2024. URL https://arxiv.org/abs/2302.00487.

Zhao, J., Wang, T., Yatskar, M., Ordonez, V., and Chang, K.-W. Gender bias in coreference resolution: Evaluation and debiasing methods. *arXiv preprint arXiv:1804.06876*, 2018.

Zhou, J., Lu, T., Mishra, S., Brahma, S., Basu, S., Luan, Y., Zhou, D., and Hou, L. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

A. The Derivation of Edge Contribution

According to the definition of attribution score in (Syed et al., 2023) and the definition of node redundancy in Section 3.3, we can define the contribution of an edge $e : n_1 \to n_2$ with n_1 as the upstream node and n_2 as the downstream node. We use the indirect effect IE to measure the change in the output caused by the patching of the edge. Thus given a dataset $\mathcal{X}$ for patching, the contribution of edge e can be expressed as follows:

$$\begin{aligned} c(e) &= \mathbb{E}_{x_i \sim \mathcal{X}}\big[|c_i(e)|\big] \\ &= \mathbb{E}_{x_i \sim \mathcal{X}}\big[|f(e; M(x_i), M(x_i'))|\big] \\ &= \mathbb{E}_{x_i \sim \mathcal{X}}\big[|IE(e; x_i)|\big] \end{aligned}$$

Note that the contribution of edge e is directly reflected in the change of the final output (the logit of the language model, etc), which is the indirect effect caused by the change of the value in the downstream node n_2, while the change of the node n_2 is directly caused by the change of the upstream node n_1. The difference between the direct effect and the indirect effect is that the former keeps all the other nodes that could influence n_2 unchanged and only studies the influence from n_1 to n_2, while the latter allows all the changes in nodes between n_2 and the logit. For more refined definitions for these two concepts, please refer to (Pearl, 2001).

Therefore, to measure the direct effect from n_1 to n_2, we set the value of n_1 to another value $n_1(x')$ while keeping the all other nodes between n_1 and n_2 unchanged. The indirect effect of e to the final output is

$$\begin{aligned} IE(e; x) =& IE(n_1 \to n_2; x) \\ =& \mathcal{L}_m\big[M\big(x \,|\, do\big(n_2 \leftarrow n_2(x')\big)\big)\big] - \mathcal{L}_m\big[M(x)\big] \end{aligned}$$

in which the corrupted value $n_2(x')$ of the downstream node is

$$n_2(x') = n_2(x) - n_2^{n_1}(x) + n_2^{n_1}(x') \tag{6}$$

Equation (6) shows the direct effect $n_2^{n_1}(x') - n_2^{n_1}(x)$ from n_1 to n_2, where

$$n_2^{n_1}(x') = n_2^{n_1}\big(x \,|\, do\big(n_1 \leftarrow n_1(x')\big)\big)$$

To simplify the equation, we apply a first-order Taylor expansion to IE at $n_2 = n_2(x)$, then

$$\begin{aligned} IE(e; x) \approx& \mathcal{L}_m[M(x)] + \big[n_2(x') - n_2(x)\big]^\top \nabla_{n_2} \mathcal{L}_m[M(x)]\big|_{n_2(x)} - \mathcal{L}_m[M(x)] \\ =& \big[n_2(x') - n_2(x)\big]^\top \nabla_{n_2} \mathcal{L}_m[M(x)]\big|_{n_2(x)} \end{aligned}$$

Thus we have

$$\begin{aligned} IE(e; x) =& \big[n_2(x) - n_2^{n_1}(x) + n_2^{n_1}(x') - n_2(x)\big]^\top \nabla_{n_2} \mathcal{L}_m[M(x)]\big|_{n_2(x)} \\ =& \big[n_2^{n_1}(x') - n_2^{n_1}(x)\big]^\top \nabla_{n_2} \mathcal{L}_m[M(x)]\big|_{n_2(x)} \end{aligned} \tag{7}$$

To further simplify Equation (7), we apply another Taylor expansion at $n_1 = n_1(x)$ to $n_2^{n_1}$. Then we have

$$\begin{aligned} IE(e; x) \approx& \Big\{ n_2^{n_1}(x) + \big[n_1(x') - n_1(x)\big]^\top \nabla_{n_1} n_2^{n_1}\big|_{n_1(x)} - n_2^{n_1}(x) \Big\} \cdot \nabla_{n_2} \mathcal{L}_m[M(x)]\big|_{n_2(x)} \\ =& \big[n_1(x') - n_1(x)\big]^\top \nabla_{n_1} n_2^{n_1}\big|_{n_1(x)} \nabla_{n_2} \mathcal{L}_m[M(x)]\big|_{n_2(x)} \end{aligned} \tag{8}$$

Thus we come to the final form of edge contribution in Equation (8). Our derivation takes the simplest situation into consideration, while it works well in practice. For more discussions on the relevant topic, please refer to (Kramár et al., 2024). For implementation details, please refer to our code.

B. 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 self-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 $i (i \in [0, L))$, the two important operations are self-attention and MLP. In self-attention, 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 $x_{Q/K/V}^j \in \mathbb{R}^{\frac{D}{H} \times T}$. 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 $Attn_i^j(x) \in \mathbb{R}^{D \times T}$ 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 $Attn_i(x) = \sum_{j=1}^H Attn_i^j(x)$.

For the MLP operation in each layer, the input x is projected into $x_{in}^i \in \mathbb{R}^{D_{mlp} \times T}$ via $W_{in} \in \mathbb{R}^{D_{mlp} \times D}$, and projected back to $MLP_i(x) \in \mathbb{R}^{D \times T}$ 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 $x_{pre}^i \in \mathbb{R}^{D_{mlp} \times T}$ via $W_{gate}^i \in \mathbb{R}^{D_{mlp} \times D}$ and is applied with an activation layer, then a dot product is performed between the activations and $x_{in}^i \in \mathbb{R}^{D_{mlp} \times T}$ 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 done on Llama series models in the complex tasks in our experiments. For details, please refer to 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.

C. Details for the Subject-verb Disagreement Task

To be brief, the goal of this task is to change the grammar in a language model from (a) to (b) as follows:

(a) *We apologize, but this video has failed to load.*
(b) *We apologizes, but this video have failed to load.*

In the example above, *We* and *this video* are subjects, *apologize* / *apologizes* and *has* / *have* are verbs, and *We* and *video* are also called the END tokens that appear before the verbs. The change from *apologize* to *apologizes* or from *has* to *have* is called a flip.

C.1. Data Preparation

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 following: *am* → *is*, *is* → *are*, *are* → *am*.

We collect 60,000 samples in total. For experiments, we only use half of the data, which is further split for training (2.4w), validation (3k), and test (3k). All the details for data preparation can be found in our code.

C.2. Experiment Settings

We use GPT2-small (Radford et al., 2019) for this task. GPT2-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 at both ends of the edges, as discussed in Section 4.1.2. Details are shown in Table 2.

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 specified in Appendix B.

NODES	UPSTREAM		DOWNSTREAM	
	$Attn_i^j(x)$	$MLP_i(x)$	$x_{Q/K/V}^{i,j}$	x_{in}^i
PARAMETERS	$W_O^{i,j}$	W_{out}^i	$W_{Q/K/V}^{i,j}$	W_{in}^i
RANGE	$i \in [0, L), j \in [0, H)$			

For all experiments, we set the learning rate to 1e-3, and batch size to 16. We use mini-batch SGD with a momentum equal to 0.9 as the optimizer. Each model is trained for 3 epochs, with 100 steps in the beginning for warmup. During training, we evaluate on the valid set every 100 steps. The metric $\mathcal{L}_m$ for measuring the flip from subject-verb agreement to disagreement is the logit difference at the END token, which is:

$$\mathcal{L}_m = logit(W_{v_{flip}}|W_{END}) - logit(W_v|W_{END})$$

where W denotes the tokens in a sentence. For example, the case "We apologize, but this video has failed to load." contains two logit differences: $logit(apologizes|We) - logit(apologize|We)$ and $logit(have|video) - logit(has|video)$. In practice, we only consider the verbs that are tokenized as a single token.

As for the calculation of edge contribution, we follow (Syed et al., 2023) and use mean ablation when patching a node. That is to say, for each activation of shape $(\texttt{batch\_size}, \texttt{seq\_len}, \texttt{d\_model})$, we replace the value at the END token position in each sample with the mean value of all tokens in all samples in a batch.

C.3. Analysis of the Quality of the Discovered Circuit

As discussed in Section 5.1.3, the performance cannot be further improved at $N = 1000$, where N is the number of edges saved in circuit discovery. To show this intuitively, the changes with N of the logit difference, PPL, and the ratio of the trainable parameters are illustrated in Figure 4. We can observe that there is a sharp turning point at $N = 1000$, where the curves start to be flat. This serves as a sign that there does exist a circuit that includes all the parameters responsible for the subject-verb disagreement task.

To better prove this conclusion and demonstrate the high quality of the circuit found in our method, we provide another two experiments below.

C.3.1. FAITHFULNESS AND COMPLETENESS

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

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

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

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

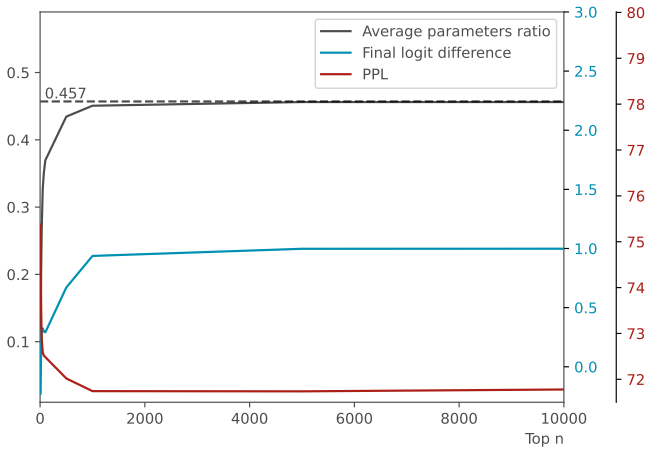
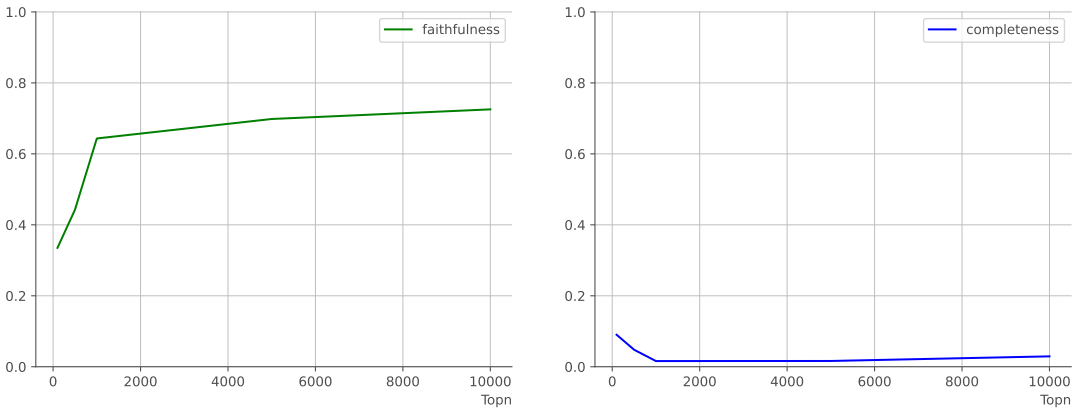


Figure 4. The influence from top n edges.

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 calculate the faithfulness and completeness of the circuits for subject-verb disagreement at $N = 100, 500, 1000, 5000, 10000$ edges. Results are shown in Figure 5. It can be seen that $N = 1000$ also serves as a turning point for the curves of faithfulness and completeness. The faithfulness of the circuits remains relatively high after $N = 1000$, ensuring the high quality of circuits.

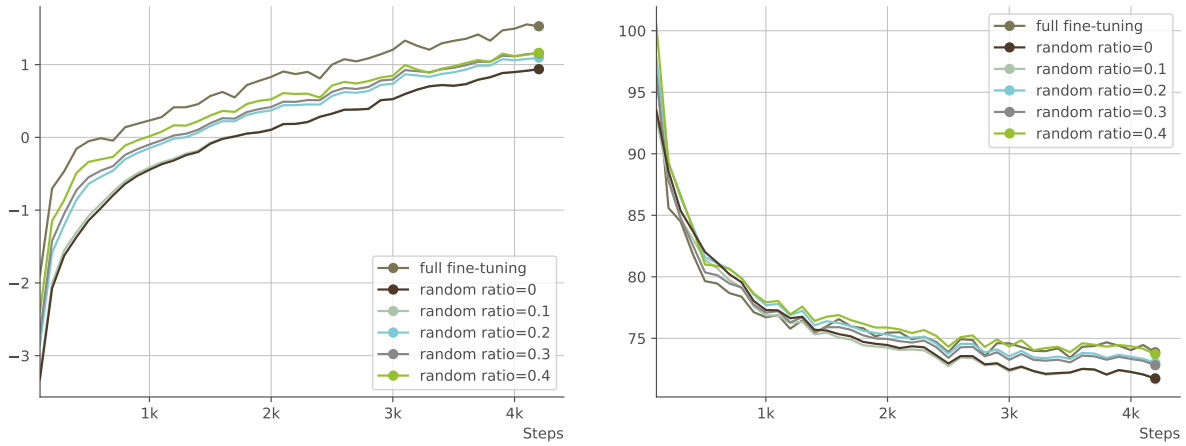


(a) Faithfulness (b) Completeness

Figure 5. The faithfulness and completeness of the circuits for subject-verb disagreement.

C.3.2. Ablation Study of Random Activation

Random activation means during training, we randomly unfreeze some parameters outside the circuit. Since we assume that the circuit with $N = 1000$ edges already includes all needed parameters for the subject-verb disagreement task, we randomly select a part of the parameters outside the $N = 1000$ circuit and involve them in optimization. In practice, we randomly activate 10%, 20%, 30% and 40% of the outside parameters, and compare the results with before. Results are shown in Figure 6. When 10% of the parameters outside the circuit are activated, the result is almost the same as before. When the ratio gets larger, we observe that the PPL is higher than before when random activation is performed, though the logit difference increases. This is because when extra parameters are summoned to fit the new data distribution, the original functions corresponding to those parameters may be destroyed. Thus the performance is improved at the expense of harming other abilities. Therefore, the circuit we find at $N = 1000$ edges is almost the exact circuit for subject-verb disagreement.



(a) logit difference (b) PPL

Figure 6. Experiment results of random activation.

C.4. Analyses of the Training Dynamics

C.4.1. Interpret the Circuit for the Subject-verb Disagreement Task

To interpret the circuit for the subject-verb disagreement task, we first analyze this task from the human perspective. To decide the form of a verb, we need to (i) find out the subject in the context and (ii) adjust the form of the verb according to the attributes of the subject, including the person attribute and the number attribute. Therefore, we assume that there exist at least two kinds of attention heads responsible for the two functions above respectively. We name the two kinds of attention heads as the **Subject Identification Heads** and the **Subject Attribute Heads**. Note that we only focus on self-attention instead of MLP because only attention layers move information across tokens, which is important for completing this task. Besides, each of the whole MLP layers is regarded as a node in the circuit in our experiment, thus we do not expect to figure out any specific function from it.

Next, we look for the two kinds of heads discussed above.

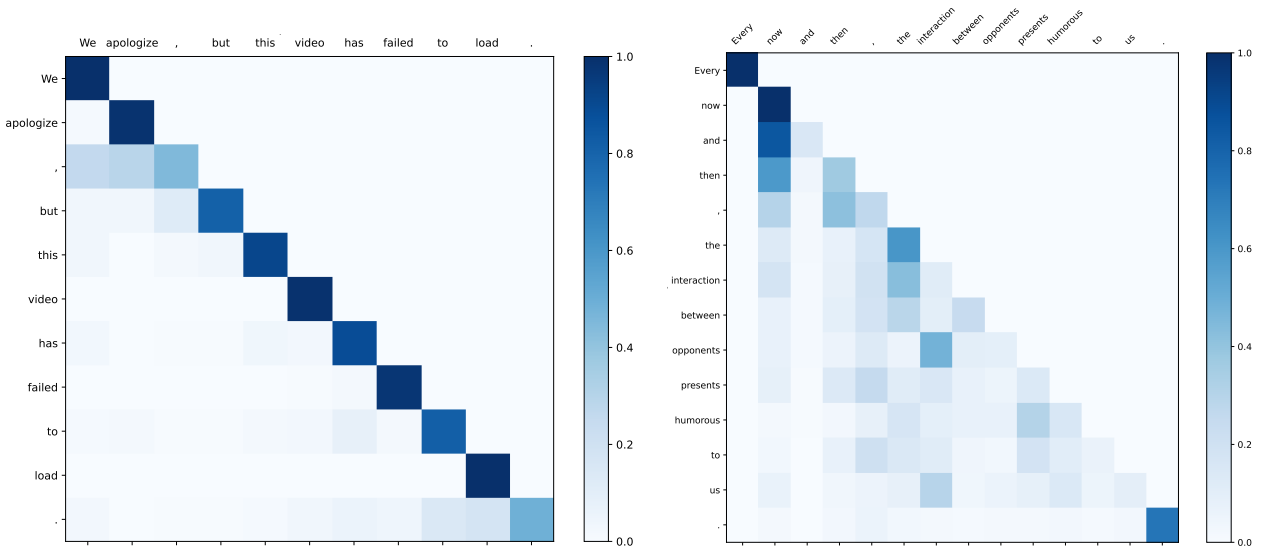
Subject Identification Heads To find out the Subject Identification Heads, we check the attention pattern of each attention head at the END token since the END token is directly used for predicting the verb token. We sort the heads in descending order according to their attention weights from the END token (query) to the subject tokens (key). Then we keep the heads in which the attention is mainly paid to the subject in the context.

We find that there exist two types of Subject Identification Heads. The type I heads mainly attend to the last token itself at the END token, so the attention pattern is a diagonal line. This type of head is helpful when the subject is exactly the END token, e.g. “He is ...”, “The girls are ...”, etc. The type II heads attend to the subject which is several tokens before the END token, e.g. “The kid who is holding an ice cream in hand is ...”, “The famous scientist, who is also an artist, has ...”, etc. The type II heads obviously have the ability of syntactic analysis and subject identification, while the type I heads may just happen to attend to the subject that overlaps with the last token.

The type I Subject Identification Heads in GPT2-small includes head.0.1, head.0.3, head.0.5, etc., while the type II heads include head.8.5, head.10.5, head.10.9, head.11.8, etc. The attention patterns for both kinds of heads are shown in Figure 7(a) and Figure 7(b) respectively.

It is worth noticing that the Subject Identification Heads remain unchanged over the training process, which means their function is preserved and shared between subject-verb agreement and subject-verb disagreement.

Subject Attribute Heads To find out the Subject Attribute Heads, we need to find out which heads directly affect the match between the verb and the subject, which is measured by the logit difference between the flipped verb and the original verb at the END token. Suppose the output of the final layer at the END token is $x_{END} \in \mathbb{R}^D$, then the logit difference is $[W_U(v_{flip}) - W_U(v))] \cdot x_{END}$, in which $W_U(v_{flip}) - W_U(v)$ is called the logit lens (nostalgebraist, 2020). Since the



(a) The attention pattern in the type I head (head.0.3)

(b) The attention pattern in the type II head (head.11.8)

Figure 7. The examples of the attention patterns in the Subject Attribute Heads. In Figure 7(a), the END token "*video*" attends to the subject "*video*" which is also the END token itself. In Figure 7(b), the END token "*opponents*" mainly attends to the subject "*interaction*".

logit lens is fixed, we expect the projection of x_{END} on the direction of the logit lens to be large, thus encouraging the probability difference between the two opposite verb forms (love v.s. loves, etc.). As discussed in Appendix B, the output x_{END} which is in the residual stream can be decomposed into the linear addition of the outputs from the previous layers. Therefore, the output of a Subject Attribute Head is a part of x_{END} and would encourage the value of the logit difference. Thus, a Subject Attribute Head is an attention head that has a large dot product value with the logit lens.

In practice, we calculate the dot product between the logit lens $W_U(v_{flip}) - W_U(v)$ and the output $Attn_i^j(x)$ from each attention head in each layer over a batch of samples. The result is shown in the main text in Figure 2. The darker the color, the larger the absolute value of the dot product is, which implies that the head is more likely to be a Subject Attribute Head. We observe that head.6.0, head.6.5, head.8.5, head.10.9, and so on see obvious flips (Figure 8) from positive to negative or the opposite direction, which implies that they are directly responsible for the match between the subject and the verb. During training, the parameters inside these heads adjust themselves to the new data distribution, while their function type remains unchanged, which is an interesting finding of the self-regulation ability inside the model.

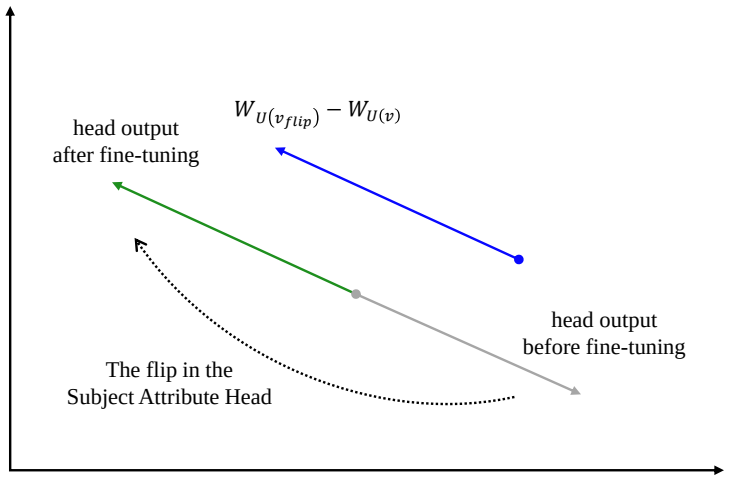


Figure 8. A sketch of the flip from subject-verb agreement to disagreement.

Collaborative Heads Finally, we notice that some of the Subject Attribute Heads (head.8.5, head.10.9, etc.) are also Subject Identification Heads, which means they also attend to subject tokens. We wonder if there exist some heads in previous layers that could influence the behavior of the Subject Attribute Heads. That is to say, the Subject Attribute Heads do not act alone but collaborate with other heads.

To find out these heads, we knock out the upstream heads one at a time at the END token using mean ablation. We observe the change in the attention pattern of each Subject Attribute Head and then keep the heads that bring obvious changes in the attention patterns. In practice, we focus on subject-verb agreement and only check the influence on head.8.5 and head.10.9. We also provide two types of data, corresponding to the two cases discussed in Appendix C.4.1, in order to provide a more detailed analysis. Results show that

- When patching on the type I data in which the END token is exactly the subject, head.1.2 and head.2.11 affect both head.8.5 and head.10.9
- When patching on the type II data in which the subject is several tokens before the END token, head.7.4 and head.2.11 affect head.8.5, while head.2.11 affects head.10.9.
- When we mix the two types of data, we find that head.1.3, 0.8, 2.10, and 6.5 affect head.8.5, while head.1.3, 1.4, 1.6, 6.5, 0.8, and 0.9 affect head.10.9. These heads may be responsible for both types of data while not specifically responsible for a certain type of data, so they appear when patching on the mixed data.

We notice that though the influence of the above heads is relatively large, the absolute influence is sometimes quite small. Therefore, we further conduct an experiment in which we patch multiple heads at a time and check the influence of them on head.8.5 and head.10.9. Results show that when upstream heads are patched together, their combined effect is much higher than the individual effect. Thus we call these heads the Collaborative Heads.

C.4.2. The Interaction and Collaboration Inside the Model

From the discussions above, we can see that the nodes inside a circuit complete a task through a cooperative division of labor. We summarize the interaction and collaboration inside the model as follows:

1. Each node is responsible for a single or multiple functions. As discussed in Appendix C.4.1, we have found attention heads that are responsible for identifying the subject in a sentence or adjust the form of a verb according to the attributes of the subject, or both. Each head has its division of labor when completing a task.
2. Some heads directly affect the output, while others cooperate with them to affect the output indirectly. For example, head.8.5 directly matches the verb with the subject, while head.7.4, head.1.3 and so on indirectly affect the output through cooperation with head.8.5.
3. Several nodes achieve a common goal through cooperation. For example, head.1.3, 1.4, 1.6, 6.5, 0.8, and 0.9 affect head.10.9 through combined effect, which means their influence on head.10.9 only appears when they act together.

C.4.3. The Evidence of Hebbian Learning

As discussed in Section 5.1.5, we observe that some edges in the circuit are strengthened or weakened during training, just like the Hebbian learning proposed in (Do, 1949) in neuroscience. As stated by Hebb, a synapse between two neurons is strengthened when the neurons on either side of the synapse have highly correlated outputs, which means they are often activated synchronously. The theory is often concluded as "Cells that fire together, wire together" (Shatz, 1992). For two neurons i and j, a common description of hebbian learning is as follows:

$$w_{ij} = \frac{1}{p}\sum_{k=1}^{p} x_i^k x_j^k$$

where w_{ij} is the weight of the connection between the two neurons, and x_i^k and x_j^k are the k-th inputs for i and j respectively. When it comes to the computational graph of a model, the nodes and edges in the graph could be viewed as the neurons and their connections in a brain from the perspective of neuroscience.

During training, we find that some of the edges are obviously stronger than others, which means they have higher edge contributions. Besides, they are strengthened all the way during training. Specifically, we analyze the circuits during training in the subject-verb disagreement task, with the setting of top $N = 1000$ edges. We check the results at 2000, 3000, and 4000 steps respectively, and visualize the circuits with the top 35 edges in Figure 9. Note that the thickness of an edge corresponds to the logarithm of the edge contribution, that is $log[1 + c(e)]$. The details are shown in Table 3.

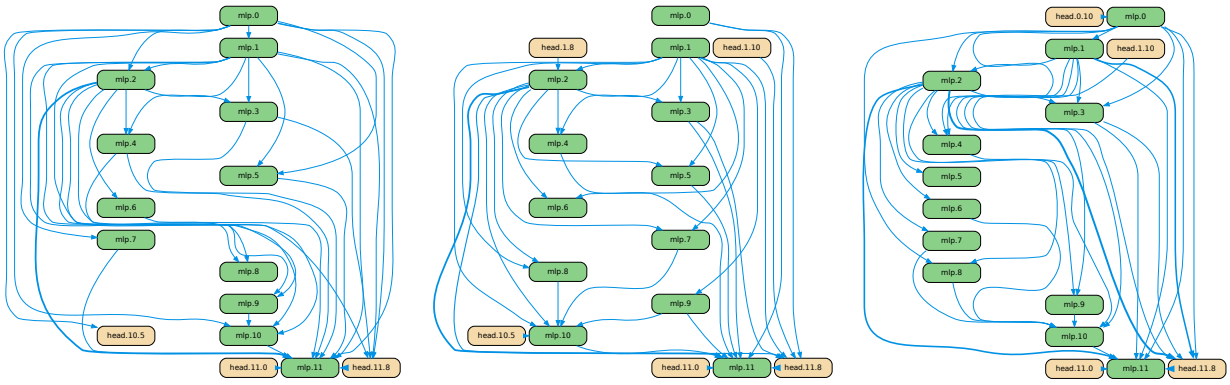
We also find that the edge contribution may keep decreasing during training. This is quite similar to the self-organization of cells inside human brains, a well-known phenomenon of which is the lateral inhibition among neurons (Jacobson, 1993), which means an activated neuron can reduce the activity of its neighbors. Inspired by this, (Kohonen, 2012) developed the self-organizing maps (SOM), an unsupervised algorithm that leverages the Winner-Take-All strategy to perform competitive learning. The core ideas behind SOM are:

- The neurons inside a neural network learn to represent data through competition, i.e. given an input, some neurons are activated while others are inhibited.
- Different inputs are represented in a topologically ordered manner, i.e. different neurons are responsible for different features in a well-organized style.

In our study, we find that the dynamic change of edges echoes the above discussions on competition and self-organization. During training, some connections between nodes are strengthened, which may reduce the intensity of other connections. After training, the components inside a model have reorganized themselves to adapt to the new data distribution. When faced with an input, different regions inside the computational graph are responsible for different subtasks and collaborate to complete a goal, as discussed in Appendix C.4.1 and Appendix C.4.2.

Table 3. Some of the strengthened and weakened edges during training. The dynamic change shows the change in edge contribution $log[1 + c(e)]$ (the start of training $\rightarrow$ 2000 steps $\rightarrow$ 3000 steps $\rightarrow$ 4000 steps). The dynamic process is visualized in Figure 9.

STRENGTHENED EDGES		WEAKENED EDGES	
EDGE	DYNAMIC CHANGE	EDGE	DYNAMIC CHANGE
MLP.2 $\rightarrow$ HEAD.11.8.V	$0 \rightarrow 0.2744 \rightarrow 0.5091 \rightarrow 0.9138$	MLP.2 $\rightarrow$ MLP.8	$0.1122 \rightarrow 0.0869 \rightarrow 0.0619 \rightarrow 0.058$
MLP.1 $\rightarrow$ HEAD.11.8.V	$0 \rightarrow 0.2227 \rightarrow 0.3978 \rightarrow 0.7231$	MLP.1 $\rightarrow$ MLP.5	$0.1043 \rightarrow 0.0859 \rightarrow 0.0736 \rightarrow 0$
MLP.2 $\rightarrow$ MLP.3	$0 \rightarrow 0.0293 \rightarrow 0.0993 \rightarrow 0.2282$	MLP.0 $\rightarrow$ MLP.10	$0.2712 \rightarrow 0.058 \rightarrow 0 \rightarrow 0$
MLP.1 $\rightarrow$ MLP.4	$0 \rightarrow 0.0396 \rightarrow 0.0652 \rightarrow 0.1748$	MLP.4 $\rightarrow$ MLP.11	$0.1791 \rightarrow 0.0454 \rightarrow 0.0428 \rightarrow 0$
MLP.2 $\rightarrow$ MLP.5	$0 \rightarrow 0 \rightarrow 0.1246 \rightarrow 0.1734$	MLP.4 $\rightarrow$ MLP.11	$0.1885 \rightarrow 0.0343 \rightarrow 0.0259 \rightarrow 0$
...	...	...	...

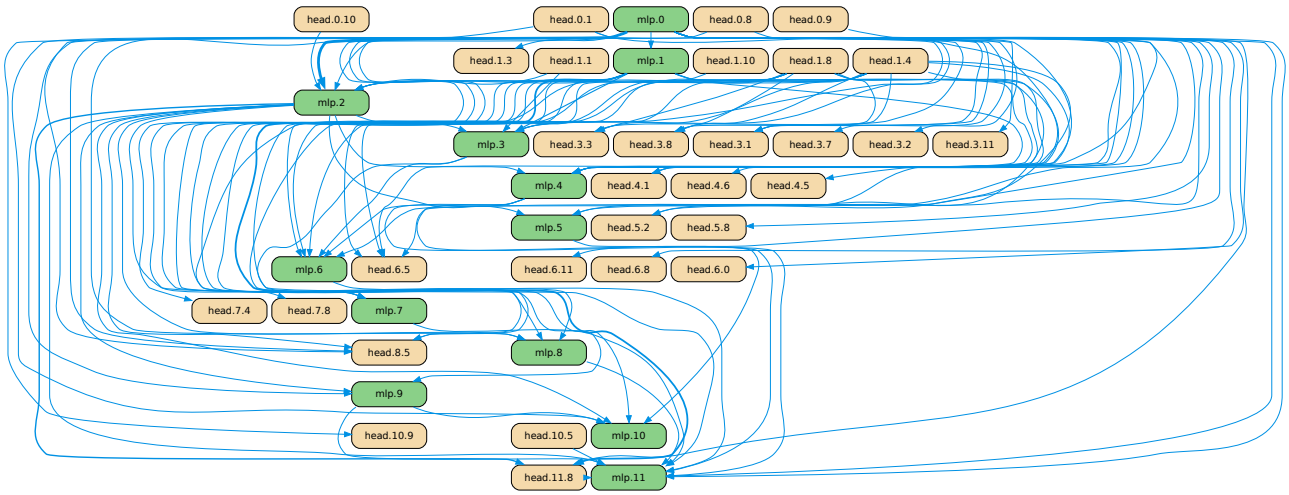


(a) circuit at 2000 steps (b) circuit at 3000 steps (c) circuit at 4000 steps

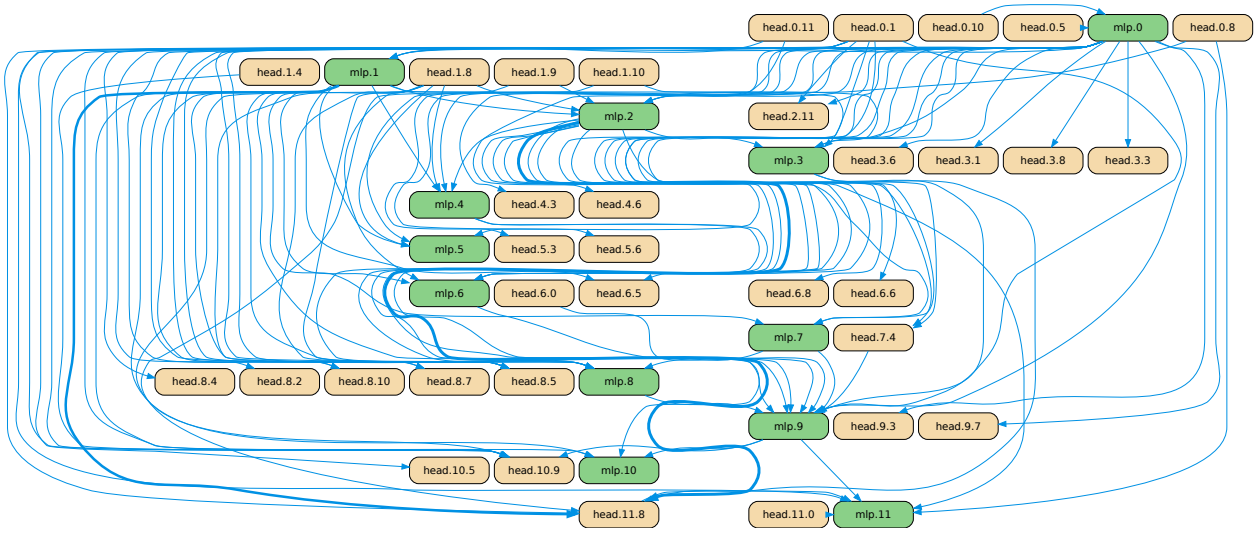
Figure 9. Evidence of Hebbian learning in the subject-verb disagreement task

C.4.4. The Circuits Before and After Fine-tuning

We present the circuits of the subject-verb disagreement task before and after fine-tuning in Figure 10(a) and Figure 10(b) respectively. The circuit of subject-verb disagreement is trained under the setting of $N = 1000$ edges, and for both of the circuits we only present the top 100 edges. It can be seen that most of the heads we discussed in Appendix C.4.1 are included in the circuit. Besides, it is obvious that the circuits before and after fine-tuning share a lot of nodes, which implies that the function shift from subject-verb agreement to disagreement happens mostly in the **polarity change** of nodes, instead of randomly assigning the ability to other nodes. This is an intuitive finding, which not only demonstrates the rationality of circuit-tuning as well as our analyses but also provides new insights for our understanding of the mechanism inside language models.



(a) The circuit before fine-tuning (subject-verb agreement).



(b) The circuit after fine-tuning (subject-verb disagreement).

Figure 10. The circuits of subject-verb agreement (top) and subject-verb disagreement (bottom).

C.4.5. The Experiment Results after the Revision of Attribution Score

During the analyses in Appendix C.4.1, we find that the original definition of the attribution score in EAP (Syed et al., 2023) fails to capture all the relevant edges in a task. For example, head.6.0 which is a Subject Attribute Head fails to appear in the circuit. As discussed in Section 5.1.6, we assume that there exists a situation where an important node is connected with many other nodes, but each edge is not that strong. For example, as illustrated in Figure 11, an upstream node n_a^u is connected with four downstream nodes, while another upstream node n_b^u is connected with only one downstream node. Since the edge between n_b^u and n_5^d is stronger than any edge between n_a^u and the nodes connected with it, the edge $n_b^u \to n_5^d$ may be kept in the circuit, while the edges in $\mathcal{E}_a = \{n_a^u \to n_i^d | i = 1, 2, 3, 4\}$ may be left out. As a result, n_a^u is not involved in optimization, though the sum of the edge contributions of all edges in $\mathcal{E}_a$ may be almost the same or even larger than that of $n_b^u \to n_5^d$.

Thus we calculate the edge contribution of an edge $e : n_i^u \to n_j^d$ as below:

$$c(e)' = c(e) \cdot \sum_{k=1}^{N_{down}^i} c(n_i^u \to n_k^d) \cdot \sum_{k=1}^{N_{up}^j} c(n_k^u \to n_j^d)$$

where $c(e)$ is the original attribution score in EAP, N_{down}^i is the number of the downstream nodes of n_i^u, N_{up}^j is the number of the upstream nodes of n_i^u. The revision considers the contributions from all the edges connected to the upstream node and the downstream node. To verify it, we conduct a new experiment on the subject-verb disagreement task and compare it with the result before. Results are shown in Figure 1(c). Details can be found in Table 4. Compared with the original attribution score, our method improves the logit difference steadily, while even bringing down the computation to some extent.

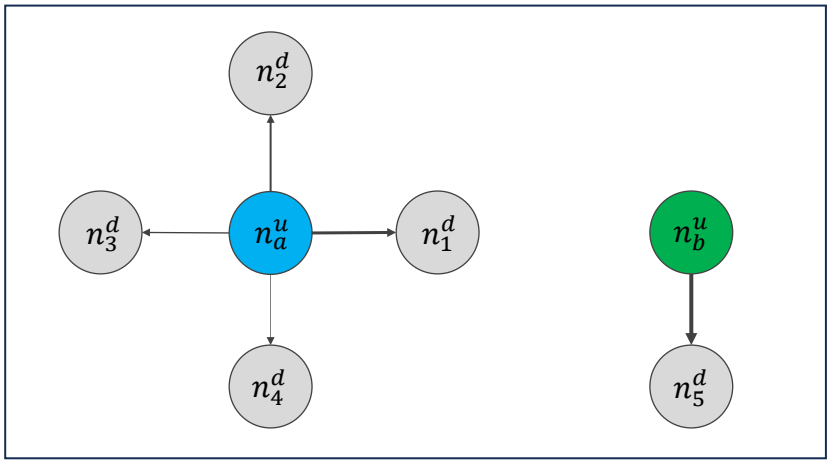


Figure 11. A sketch for the idea behind the revision on attribution score.

Table 4. Comparison between the performance before and after the improvement on attribution score.

Top n & Methods	Original EAP			Improved EAP		
	Logit difference	PPL	Avg. Param ratio (%)	Logit difference	PPL	Avg. Param ratio (%)
50	0.292	72.59	32.61	0.350	72.71	26.88
100	0.291	72.50	36.96	0.382	72.53	34.12
500	0.670	72.02	43.50	0.819	72.05	42.50
1000	0.937	71.74	45.61	0.970	71.73	45.55

D. Details for the Complex Tasks

D.1. Details for Task Settings

D.1.1. Reasoning-based Tasks

Mathematics We use GSM8K (Cobbe et al., 2021) as the dataset, which contains about 8.5k grade school math problems with natural language solutions. The answer to a problem not only contains the final answer but also provides the process for solving the problem. An example of this task is shown in Table 5.

Table 5. An example in the GSM8K dataset.

Question	Answer
Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?	Janet sells $16 - 3 - 4 = << 16 - 3 - 4 = 9 >> 9$ duck eggs a day. She makes $9 * 2 = \$ << 9 * 2 = 18 >> 18$ every day at the farmer's market. ####18

During training, the NLL loss also serves as the metric $\mathcal{L}_m$ for measuring the output of the model. For evaluation, we use Acc@1 as the metric, which means for each problem in the test set we only sample one answer from the model.

Logical Reasoning We use Contexthub (Hua et al., 2024) as the dataset, which consists of problems of 4 difficulty levels, including deductive and abductive reasoning in 12 distinct categories or domains from Wikipedia. The problems together with their reasoning processes are instantiated automatically by LLMs following the fixed formal logic templates. The whole dataset contains 18,240 samples. We only use level 1 and level 2 in our experiment for convenience, which include 6720 samples in total. An example of this task is shown in Table 6.

Table 6. An example in the Contexthub dataset.

Item	Template	Instantiation
Premise	<aaa>	The Sahara desert receives heavy rainfall this year.
	<aab>	The Amazon rainforest experiences severe drought conditions.
	<aac>	Some of Earth's major ecosystems are undergoing significant changes in weather patterns.
Question	(aaa OR aab) $\rightarrow$ aac. Given aac is False, what is the value of aab?	If either the Sahara desert receives heavy rainfall this year or the Amazon rainforest experiences severe drought conditions, then it implies that some of Earth's major ecosystems are undergoing significant changes in weather patterns. Given that it is false that some of Earth's major ecosystems are undergoing significant changes in weather patterns, what can be determined about the Amazon rainforest experiencing severe drought conditions this year? (True, False, or N/A (undetermined).
Reasoning	(aaa OR aab) $\rightarrow$ aac = False. Given aac is False, the value of premise (aaa OR aab) is False, thus, the value of aab is abduced as False. Thus, the answer is False	"The Sahara desert receives heavy rainfall this year" or "The Amazon rainforest experiences severe drought conditions") logically implies "Some of Earth's major ecosystems are undergoing significant changes in weather patterns" whose corresponding truth value is False. Given "Some of Earth's major ecosystems are undergoing significant changes in weather patterns" is False, the value of premise ("The Sahara desert receives heavy rainfall this year" or "The Amazon rainforest experiences severe drought conditions") is False, thus, the value of "The Amazon rainforest experiences severe drought conditionsïs abduced as False. Thus, the answer is <answer>False</answer>

During training, the NLL loss also serves as the metric $\mathcal{L}_m$ for measuring the output of the model. For evaluation, we use the average F1 score over all categories of data.

For all reasoning-based tasks, we add an instruction "Please answer step by step." at the end of the question in order to guide the model to answer the question step by step.

D.1.2. REASONING-FREE TASKS

Gender De-biasing 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. We aims 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 $logit(W_{her}|W_{END}) - logit(W_{his}|W_{END})$, where W_{END} is 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.

Though the training set we use is balanced between genders and stereotypes, the number of samples for each occupation is not balanced. To further improve performance, we additionally add a regularization term to the original NLL loss. Then the total loss is

$$\mathcal{L} = \mathcal{L}_{NLL} + \beta \cdot |logit(W_{pron}|W_{END}) - logit(W_{anti-pron}|W_{END})|$$

in which β is a hyper-parameter for controlling the weight of regularization. The absolute value of the logit difference aims to minimize the difference between genders in the model's stereotype.

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

To better evaluate the performance of gender de-biasing, we adopt the concept of prejudice 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 $y \in \{male, female\}$ and a context $c \in C$, the stereotype of a model M against x about y in the context c is

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

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 prejudice risk is

$$R^p = \mathbb{E}_{x \sim X}(r^p_x)$$

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

Reading Comprehension We use SQuAD 2.0 (Rajpurkar et al., 2018) as the dataset. The input contains a paragraph from a passage, and a question related to that paragraph. The answer could be a word or a phrase in the paragraph, or "<No Answer >" which means the answer cannot be found in the paragraph. An example for this task is shown in Table 7.

During training, the NLL loss serves as the metric $\mathcal{L}_m$ for measuring the output of the model, since the answer is a segment of text which may contain one or multiple tokens. For evaluation, we use the development set for convenience, and the metric is exact match and F1 score.

Table 7. An example in the SQuAD 2.0 dataset.

Item	Content
Paragraph	The Normans (Norman: Nourmands; French: Normands; Latin: Normanni) were the people who in the 10th and 11th centuries gave their name to Normandy, a region in France. They were descended from Norse ("Norman" comes from "Norseman") raiders and pirates from Denmark, Iceland and Norway who, under their leader Rollo, agreed to swear fealty to King Charles III of West Francia. Through generations of assimilation and mixing with the native Frankish and Roman-Gaulish populations, their descendants would gradually merge with the Carolingian-based cultures of West Francia. The distinct cultural and ethnic identity of the Normans emerged initially in the first half of the 10th century, and it continued to evolve over the succeeding centuries.
Question	In what country is Normandy located?
Answer	France

D.2. Details for Implementation

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 GPT2-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. Details are shown in Table 8.

Table 8. The settings of the nodes and their corresponding parameters in the complex tasks. 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 specified in Appendix B.

NODES	UPSTREAM		DOWNSTREAM		
	$Attn_i^j(x)$	$MLP_i^k(x)$	$x_{Q/K/V}^{i,j}$	$x_{pre}^{i,k}$	$x_{in}^{i,k}$
PARAMETERS	$W_O^{i,j}$	$W_{out}^{i,k}$	$W_{Q/K/V}^{i,j}$	$W_{gate}^{i,k}$	$W_{in}^{i,k}$
RANGE	$i \in [0, L), j \in [0, H), k \in [0, H^*)$				

As for the calculation of edge contribution, we use mean ablation as before when patching a node. For reasoning-based tasks and the reading comprehension task in reasoning-free tasks, there is no such thing as the END token. Therefore, for each activation of shape $(\texttt{batch\_size}, \texttt{seq\_len}, \texttt{n\_head}, \texttt{d\_model})$, we take all tokens into consideration and use the mean value over all tokens and all samples for mean ablation. For implementation details, please refer to our source code.

For circuit-tuning, we set the number of edges N to 2000, 3000, and 4000 for 1B/3B/8B models respectively. Since the tasks are much more complex, the value of N is decided by experience. The batch size is set to 16 in all experiments. We set the learning rate to 3e-5 for the mathematics and reading comprehension tasks, and 1e-4 for other tasks. Mini-batch SGD with a momentum equal to 0.9 is used as the optimizer. During training, we perform circuit discovery every 8 steps after optimization for efficiency, which is different from the experiments on GPT2-small in which we perform circuit discovery rightly after an iteration step. For each task, we train the model until performance cannot be further improved.

For LoRA, we set $r = 32, \alpha = 64$ for all experiments. For full fine-tuning and LoRA, we use the same optimizer as that in circuit-tuning. For all other hyper-parameters such as learning rate, batch size, training steps, and so on, we just sweep over a range of choices and choose the best ones.

In practice, we find that circuit-tuning is much more stable than full fine-tuning and LoRA. When we sweep over a range of hyper-parameters, we notice that full fine-tuning and LoRA are quite sensitive to the change of learning rate, batch size, training steps, and so on. When it comes to circuit-tuning, the change in evaluation result is relatively moderate while still maintaining good performance. Thus it echoes our discussion of the stability of circuit-tuning in Section 4.3.

D.3. Details for Evaluations on General Capabilities

To demonstrate that our method is good at preserving general capabilities, we test the fine-tuned models on a set of benchmarks involving general capabilities as well as other capabilities.

For general capabilities, we use MMLU (Hendrycks et al., 2020), Winogrande (Sakaguchi et al., 2021) and IFEval (Zhou et al., 2023). For MMLU, the evaluation metric is the average accuracy over all categories. For Winogrande, we use the development set for convenience, and the evaluation metric is accuracy. For IFEval which is to test the instruction following ability of a model, each prompt contains one or multiple verifiable instructions, thus the evaluation metric is divided into the prompt-level accuracy and instruction-level accuracy. Due to the randomness of generation, each response is tested under multiple transformations, thus the metric is further divided into strict criterion and loose criterion. In practice, we use the prompt-level and instruction-level accuracy averaged on the strict and loose criteria.

For other capabilities, we consider reasoning, coding, and multilingual capabilities. For reasoning, we use GPQA (Rein et al., 2023) with accuracy as the metric. For coding, we use HumanEval (Chen et al., 2021) with pass@1 as the metric. For multilingual capability, we use MGSM (Shi et al., 2022) with the accuracy averaged on all languages.

For each evaluation benchmark, we test for 5 times and get the averaged result.

D.4. Influence of the Regularization Term on Gender De-biasing

We visualize the prejudice risk of the models before and after gender de-biasing in Figure 12. Note that the coefficient for regularization $\beta = 0.5$. Since there are only 40 types of occupations in the test set, we regard the distribution of prejudice risk as normal distribution and perform interpolation on the computed results. For convenience, we only show the results of GPT2-small and Llama-3.2-1B-instruct in the figure.

The dynamic process of de-biasing can be observed from right to left. It is obvious that with a regularization term in the loss function, the distribution of the prejudice risk in occupations is more concentrated to a smaller value. The results demonstrate the effectiveness of our method in modifying a model's stereotype, providing new insights in aligning AI systems with human values.

Besides, the experiment result tells us that we can customize the algorithm settings (e.g., the loss function) according to the requirement of a task. Thus, it is convenient to intervene in the training process flexibly to modify the model behaviors, demonstrating the great potential of circuit-tuning in the study of fine-tuning and interpretability.

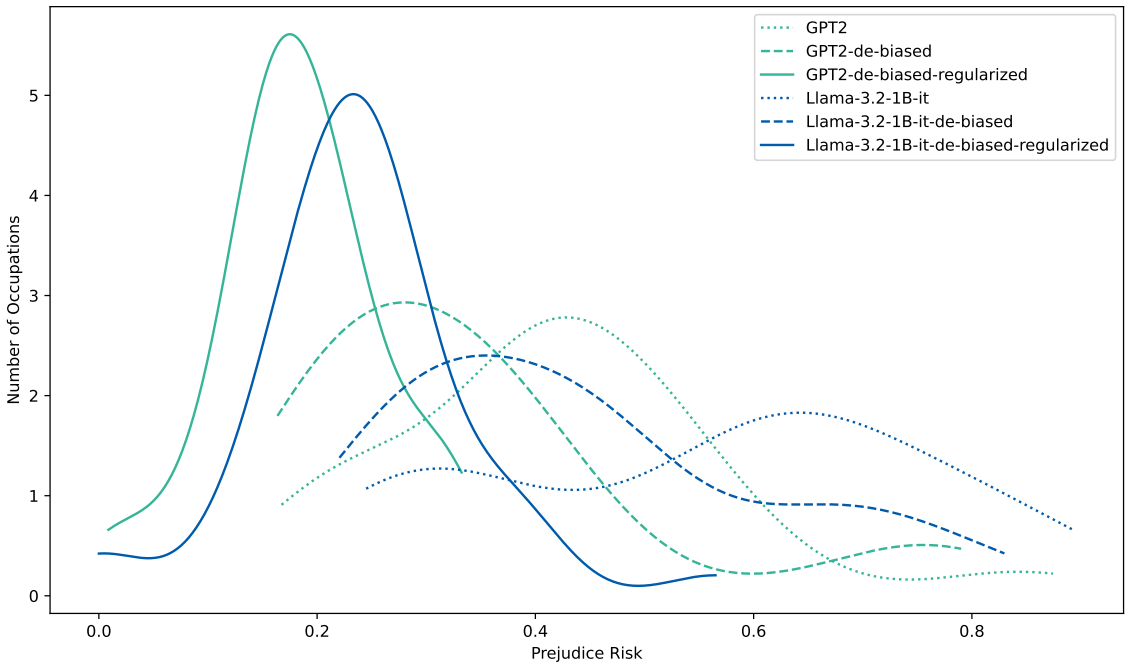


Figure 12. The comparison between biased and de-biased models.