

# Multi-component Causal Tracing in Large Language Models

Zirui Yan<sup>1</sup>, Dennis Wei<sup>2</sup>, Dmitriy A. Katz<sup>2</sup>, Prasanna Sattigeri<sup>2</sup>, Ali Tajer<sup>1</sup>

<sup>1</sup>Rensselaer Polytechnic Institute, <sup>2</sup>IBM Research

## Abstract

Causal tracing systematically intervenes on a large language model’s (LLM’s) internal representations to uncover and quantify the causal pathways linking specific inputs or computations to specific metrics of interest, quantifying the LLM’s behavior. Building on previous single-component or single-layer studies, this paper presents a unified framework for causally tracing multiple components simultaneously. This framework systematically identifies the subsets of components (e.g., attention heads and multi-layer perceptron neurons) most critical to a desired target performance metric (e.g., accuracy and fairness). This is achieved by incorporating flexible interventions applied to a wide range of desired metrics. To address the combinatorial complexity of the multi-component problem, an efficient algorithm is designed that leverages soft interventions and a carefully designed metric transformation, converting the combinatorial search problem into a continuous one that can be solved efficiently under proper constraints, thereby generating proper binary decisions for selecting components. Experimental results demonstrate that the proposed method efficiently identifies subsets of the model’s components that have a high impact on the target metric, outperforming existing baseline approaches. Our code is available at <https://github.com/ZiruiYan/multi-component-causal-tracing>.

## 1 Introduction

Large language models (LLMs) have revolutionized natural language processing through their ability to generate fluent, contextually-aware text, with a wide range of applications such as machine translation (Zhang et al., 2023), text summarization (Van Veen et al., 2024), code generation (Jiang et al., 2024), and agentic tasks (Xi et al., 2025). However, LLMs are prone to various forms of safety risks. They can inadvertently learn and propagate societal biases (Sheng et al., 2021), produce

factually incorrect statements (Augenstein et al., 2024), and even generate harmful or deceptive content through jailbreak attacks (Wang et al., 2025). These challenges, coupled with the fundamental interest in revealing how LLM knowledge flows, highlight the need for interpretability capabilities to systematically analyze and elucidate the underlying mechanisms of these models.

Several interpretability approaches have been presented recently (Luo and Specia, 2024), ranging from external explanations that focus on the impact of the prompt on a single prediction (e.g., gradient-based (Sikdar et al., 2021) and vector-based (Chen et al., 2020; Modarressi et al., 2022; Paes et al., 2025)), to internal explanations that aim to uncover general knowledge and representational structures within a model (e.g., probing (Petroni et al., 2019; Li et al., 2023), circuit analysis (Chughtai et al., 2023; Wang et al., 2023; Conmy et al., 2023), and sparse autoencoders (Bricken et al., 2023; Cunningham et al., 2024)).

Among the global methods, *causal tracing* (Vig et al., 2020b), which is also called activation patching (Davies et al., 2023) or interchange interventions (Geiger et al., 2020, 2022), has emerged as an effective strategy by treating LLMs as causal models, enabling systematic interventions on internal values to pinpoint which specific model components most strongly influence a chosen metric. This goes beyond counterfactual explanations for black-box models (Wachter et al., 2017; Karimi et al., 2022) and recent work has proposed different methods (Mohebbi et al., 2023) and metrics (Zhang and Nanda, 2024). By revealing the importance of specific network components, causal tracing enables more targeted editing, fine-tuning or enhancement of LLMs (Geiger et al., 2021; Meng et al., 2022, 2023; Cai et al., 2024; Hase et al., 2023; Chen et al., 2026; Prakash et al., 2024). For example, Cai et al. (2024) uses causal mediation analysis to identify model components that contribute most to

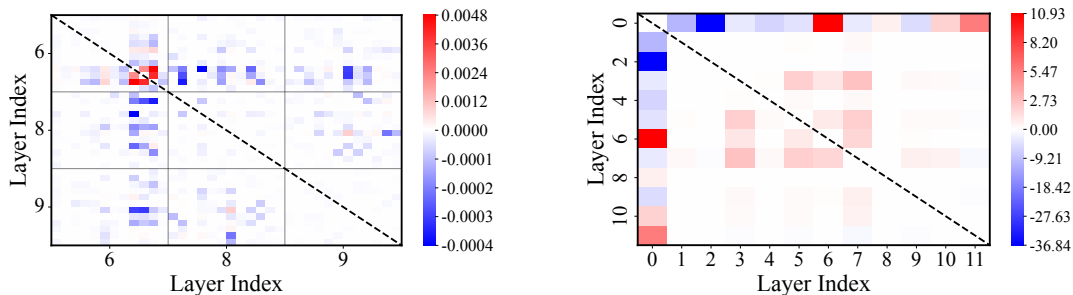


Figure 1: The deviation from linearity due to intervening on two attention heads within layers  $\{6, 8, 9\}$  of GPT2-small on the WinoBias dataset (left), and two MLP layers of GPT2-small on the Professions dataset (right).

gender bias and then mitigate gender bias by intervening on those components; Chen et al. (2026) show tracing-guided editing can rival fine-tuning and in-context editing; Prakash et al. (2024) show that fine-tuning improves performance mainly by strengthening existing circuits rather than creating new ones through causal tracing.

Most existing studies in causal tracing have focused on analyzing single components in isolation: an individual neuron (Vig et al., 2020b,a) or entire layer (Meng et al., 2022, 2023; Hase et al., 2023). This overlooks the potentially *non-linear* effects arising from interactions among multiple components. Emergent multi-component behaviors have been revealed in circuit analysis. Elhage et al. (2021) and Olsson et al. (2022) discovered the *induction heads* mechanism, according to which two attention heads in different layers jointly increase the likelihood of a specific token. We refer to Ferrando et al. (2024) for more findings.

A phenomenon similar to induction heads can also be observed in the causal tracing of LLMs, in which jointly intervened-on pairs of attention heads or multi-layer perceptron (MLP) neurons exhibit a pronounced *non-linear* impact on the model’s outputs. Figure 1 illustrates this phenomenon in GPT2-small on two datasets, where selecting specific subsets of components can significantly boost the metric of interest (which are discussed in detail in Appendix C.5). The figure shows the metric values obtained by intervening on two components simultaneously, minus the sum of the metric values obtained when intervening on each component individually. Red indicates a positive difference (the joint intervention enhances the metric), whereas blue indicates a negative difference (the joint intervention diminishes the metric). These nonlinear effects violate the observation of Syed et al. (2023); Hanna et al. (2024); Shah et al. (2024) for these specific models and metrics, implying that linear approximations do not hold in general.

Motivated by these observations, we argue that causal tracing in LLMs should be pursued through a *multi-component* perspective. Although existing studies provide valuable evidence, they lack a unified framework to define the problem and rigorously account for non-linear interactions. We address this gap by formulating the problem of *multi-component causal tracing*, in which the goal is to identify a subset of components (e.g., attention heads, MLP neurons) that collectively maximize a chosen metric of interest. Recent work by Davies et al. (2023) investigated a similar problem via gradient-based search. However, our analysis in Appendix D.5 indicates that their penalty and optimization choices drive much of the performance gap. The relative efficiency and effectiveness of gradient-based algorithms compared with alternative methods remain unknown. We fill these gaps and our main contributions are as follows.

- **Unified Framework.** We formalize a causal tracing approach that systematically traces causal effects across multiple components of LLMs. This framework provides general definitions of intervention and metrics that can be tailored for various tasks, including probing linguistic features, testing bias, and monitoring factual correctness. The multi-component causal tracing problem poses a combinatorial challenge, resulting in a search space that grows exponentially with model size.
- **Efficient Algorithm.** To efficiently search in the combinatorial space, we perform *soft* interventions on components, combined with a carefully designed metric transformation and scheduled penalty function. The transformation unlocks the power of gradient descent, while the penalty function ensures sparsity and binary values. Furthermore, the scheduling in the penalty function allows the algorithm to converge to the desired sparsity level at an appropriate rate and stop once it is achieved.

- **Empirical Efficiency and Effectiveness.** We show that baseline top- $k$  and greedy algorithms can be computationally expensive since they require computing multiple forward runs, at least one for each component on all samples. In our experiments on GPT2-medium with the Winogender dataset, our method achieves a 1.76-times speedup compared to top- $k$  and a 229-times speedup compared to greedy, while maintaining performance close to greedy and superior to top- $k$ .

## 2 LLM Model & Notations

**Pre-trained LLM.** Consider a pre-trained LLM  $f_\theta$ , where  $\theta$  compactly parameterizes the LLM. The model receives the prompt  $\mathbf{s}$ , operates over the vocabulary set  $\mathcal{V}$ , and generates words sequentially. The word generated at time  $t$  is denoted by  $v_t$ , and accordingly, we define the sequence generated up to  $t$  as the ordered sequence  $\mathbf{v}^t \triangleq \{v_i \in \mathcal{V} : i \in [t]\}$ . Word generation begins with an initial *ordered* sentence of  $n$  words denoted by  $\mathbf{v}^n$  and proceeds by generating subsequent tokens sequentially and autoregressively for indices  $t > n$ . Specifically, at time  $t$ , the token is generated according to

$$v_t \triangleq \arg \max_{v \in \mathcal{V}} \mathbb{P}(v \mid \mathbf{v}^{t-1}), \quad (1)$$

where  $\mathbb{P}(v \mid \mathbf{v}^{t-1})$  is an *unknown* conditional probability mass function with the following shorthand:

$$\mathbb{P}_{t,d}^{\mathbf{v}}(v) \triangleq \mathbb{P}(v \mid \mathbf{v}^{t-1}), \quad \forall v \in \mathcal{V}. \quad (2)$$

This token-generation process continues sequentially until a predefined stopping criterion is met, such as the generation of a predetermined end-of-sequence token.

In designing our algorithm presented in Section 4, we leverage the average likelihood of the tokens generated over specified intervals. For this purpose, we define the following mixture distribution, which for any given  $t, d \in \mathbb{N}$ , averages the conditional probability mass functions (PMFs) over  $\{t, \dots, t+d\}$ , i.e.,

$$\bar{\mathbb{P}}_{t,d}^{\mathbf{v}}(\mathbf{v}^{t+d}) \triangleq \frac{1}{d+1} \sum_{i=0}^d \mathbb{P}_{t+i}^{\mathbf{v}}(v_{t+i}). \quad (3)$$

**LLM Components.** LLMs are constructed by stacking multiple computational blocks. Each block consists of several key components, including attention heads, feed-forward layers, MLP lay-

ers, and normalization layers. We denote the set of components available for our analysis by

$$\mathcal{C} \triangleq \{c_i : i \in [N]\}. \quad (4)$$

This set can include any non-overlapping substructures of the model, allowing flexibility in analyzing specific components. Without loss of generality, we assume the components in  $\mathcal{C}$  are ordered following the order they appear in the LLM’s architecture.

Some studies suggest that key knowledge in LLMs is primarily stored in attention heads and MLP layers (see, e.g., (Clark et al., 2019; Vig et al., 2020b; Geva et al., 2021, 2022; Meng et al., 2022)). Hence, this paper focuses primarily on the attention heads and the MLP layers. Specifically, we are interested in each head’s attention weights and the behavior of individual neurons after the down-project layer. In our empirical analysis, we consider models ranging from DistilGPT2 (72 attention heads and 4608 MLP neurons) to GPT2-xl (1200 attention heads and 76800 MLP neurons), as well as new architectures of similar size.

**Forward Mapping.** For each component  $c_i \in \mathcal{C}$ , we denote the input hidden states by  $\mathbf{g}_i$ , the output hidden states by  $\mathbf{h}_i$ , and the component mapping function by  $f_i$ , which maps out  $\mathbf{g}_i$  to  $\mathbf{h}_i$ , i.e.,

$$\mathbf{h}_i = f_i(\mathbf{g}_i), \quad \forall i \in [N]. \quad (5)$$

We use the notations  $\mathbf{g} \triangleq (\mathbf{g}_1, \dots, \mathbf{g}_N)$  and  $\mathbf{h} \triangleq (\mathbf{h}_1, \dots, \mathbf{h}_N)$  to compactly represent all the input and output states, respectively.

## 3 Multi-Component Causal Tracing

**Causal Mediation Analysis.** Our central objective is to quantify how changes in specific components of an LLM influence an LLM’s behavior in specific contexts (e.g., biases). To formalize a principled approach to trace the impact of a change in a component, we perform *causal mediation analysis* (Pearl, 2022). This enables examination of how an *interventional treatment* (i.e., a change in a component) affects the outcome (i.e., a target metric) through variations in intermediate variables, known as *mediators* (i.e., intermediate components). In the setting of LLMs, the mediators are various components of the LLM, such as attention heads and MLP neurons, where the intervention targets can be attention weights and MLP down-projection. Our focus is on these indirect effects: specifically, how our treatment, intervening on mediators, changes the metric of interest.

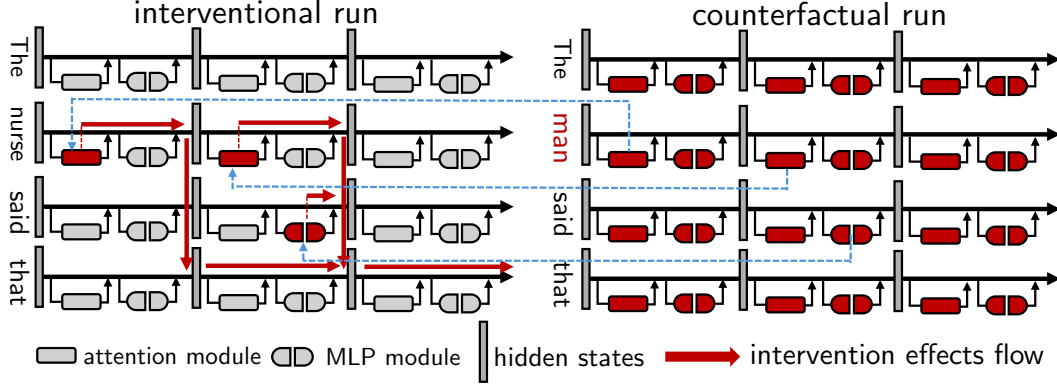


Figure 2: Counterfactual intervention in LLMs.

We denote the subset of components selected for treatment by  $\mathcal{H} \subseteq \mathcal{C}$ . To specify the components selected for treatment, we define  $\{m_i : i \in [N]\}$  such that  $m_i \triangleq \mathbb{1}\{c_i \in \mathcal{H}\}$ , where  $\mathbb{1}$  is the indicator function. Accordingly, we define  $\mathbf{m} \triangleq (m_1, \dots, m_N)$ . In this context, a treatment involves intervening in these components by replacing specific attention weights or neuron activations with counterfactual ones obtained from a separate experiment. This procedure is described next.

**Intervention via Mixture Forward.** We formalize and perform a process referred to as a *mixture forward pass* to analyze how an intervention on the set  $\mathcal{H}$  propagates throughout the rest of the model components and eventually influences a target metric. This process is demonstrated in Figure 2 and it is similar to activation patching when applied only to activations (Davies et al., 2023). For this purpose, we specify the following two word-generation processes.

1. *Counterfactual Run:* Corresponding to a given  $\mathbf{s}$ , we create a *counterfactual* prompt, denoted by  $\mathbf{s}'$ , by modifying some of the elements of the prompt  $\mathbf{s}$  (e.g., altering a word in the prompt or adding noise to word embeddings). This modified prompt generates a new set of hidden states  $\mathbf{g}'$  and  $\mathbf{h}'$ , which are related according to (5).
2. *Interventional Run:* Given a prompt  $\mathbf{s}$  and its associated counterfactual  $\mathbf{s}'$  and a set of components  $\mathcal{H}$ , we create an intervention by replacing the hidden states  $\{\mathbf{h}_i : m_i = 1, i \in [N]\}$  with those associated with  $\mathbf{s}'$ , i.e.,  $\{\mathbf{h}'_i : m_i = 1, i \in [N]\}$ . These changes, in turn, induce changes in all the downstream component states, distinct from those associated with  $\mathbf{s}$  and  $\mathbf{s}'$ . To emphasize this, we define  $\bar{\mathbf{g}}_i$  and  $\bar{\mathbf{h}}_i$  as the post-intervention hidden states of the components. We note that the state transformation functions

$\{f_i : i \in [N]\}$  remain unchanged. Hence,

$$\bar{\mathbf{h}}_i \triangleq (1 - m_i) \cdot f_i(\bar{\mathbf{g}}_i) + m_i \cdot \mathbf{h}'_i, \quad (6)$$

Finally, similarly to  $v_t$  and  $\mathbf{v}^t$ , we define  $u_t$  and  $\mathbf{u}^t$  as the sequence generated by the interventional run and denote the associated conditional PMF in the interventional run and the average conditional PMF over the interval  $\{t, \dots, t+d\}$  by

$$\mathbb{P}_t^{\mathbf{u}}(u_t) \triangleq \mathbb{P}_\theta(u_t \mid \mathbf{u}^{t-1}, \mathbf{m}), \quad (7)$$

$$\bar{\mathbb{P}}_{t:d}^{\mathbf{u}}(\mathbf{u}^{t+d}) \triangleq \frac{1}{d+1} \sum_{i=0}^d \mathbb{P}_{t+i}^{\mathbf{u}}(u_{t+i}). \quad (8)$$

Throughout the rest of the paper, we refer to  $\mathbb{P}_t^{\mathbf{y}}$  and  $\mathbb{P}_t^{\mathbf{u}}$  as the *observational* and *interventional* distributions, respectively.

**Example 1 (Gender bias.)** Consider the example of gender bias similar to the one in Figure 2, in which we have the prompt  $\mathbf{s}$  = “The technician told the customer that she”. The counterfactual intervention replaces “she” with “he”, producing the counterfactual prompt  $\mathbf{s}'$  = “The technician told the customer that he”. A model without a gender bias is desired to treat the two profession-related candidates, a stereotypical one  $\mathbf{x}$  = “could pay with cash.” and an anti-stereotypical one  $\mathbf{y}$  = “had completed the repair.”, equally in completion probabilities. Specifically, it is desirable to have

$$\bar{\mathbb{P}}_{t:d}^{\mathbf{y}}(\mathbf{x}) = \bar{\mathbb{P}}_{t:d}^{\mathbf{y}}(\mathbf{y}). \quad (9)$$

In this context, to quantify the bias, one can use the conditional likelihood ratio of generating  $\mathbf{x}$  versus  $\mathbf{y}$  (Vig et al., 2020b), i.e.,

$$\mathbb{P}_\theta(\mathbf{y} \mid \mathbf{s}) / \mathbb{P}_\theta(\mathbf{x} \mid \mathbf{s}). \quad (10)$$

When this ratio is smaller (larger) than 1, the model behaves stereotypically (anti-stereotypically), and

when the ratio is 1, it is considered unbiased. Leveraging this notion of unbiasedness, we can set the metric  $\ell$  as follows

$$\ell^{\mathbf{v},\mathbf{u}}(\mathbf{s}, \mathbf{x}, \mathbf{y}) = \frac{\bar{\mathbb{P}}_{|\mathbf{s}|+1:|\mathbf{y}|}^{\mathbf{u}}(\mathbf{y})/\bar{\mathbb{P}}_{|\mathbf{s}|+1:|\mathbf{x}|}^{\mathbf{u}}(\mathbf{x})}{\bar{\mathbb{P}}_{|\mathbf{s}|+1:|\mathbf{y}|}^{\mathbf{v}}(\mathbf{y})/\bar{\mathbb{P}}_{|\mathbf{s}|+1:|\mathbf{x}|}^{\mathbf{v}}(\mathbf{x})} - 1. \quad (11)$$

The value  $\ell^{\mathbf{v},\mathbf{u}}(\mathbf{s}, \mathbf{x}, \mathbf{y}) = 0$  means that the selected components do not affect gender bias;  $\ell^{\mathbf{v},\mathbf{u}}(\mathbf{s}, \mathbf{x}, \mathbf{y}) > 0$  indicates that they store stereotype information; and  $\ell^{\mathbf{v},\mathbf{u}}(\mathbf{s}, \mathbf{x}, \mathbf{y}) < 0$  indicates that they store anti-stereotype information.

**Example 2 (Knowledge Localization.)** Consider the setting in which we seek to identify the critical components that enable the LLM to answer knowledge-based questions. For example, consider the prompt  $\mathbf{s} = \text{“Steve Jobs was the founder of”}$  with the desired output being that the model assigns full probability to  $\mathbf{x} = \text{“Apple”}$ :

$$\mathbb{P}_{\theta}(\mathbf{x} | \mathbf{s}) = 1. \quad (12)$$

For this example, the counterfactual run is generated by adding a large of noise to the embedding layer, disrupting knowledge retrieval. Hence, an objective can be to measure how well the model answers the question based on the stored knowledge in its components, and the metric  $\ell^{\mathbf{v},\mathbf{u}}(\mathbf{s}, \mathbf{x}, \mathbf{y})$  can be used to do so. Apart from the ratio metric used in Example 1, another alternative is

$$\ell^{\mathbf{v},\mathbf{u}}(\mathbf{s}, \mathbf{x}, \mathbf{y}) = \bar{\mathbb{P}}_{|\mathbf{s}|+1:0}^{\mathbf{v}}(\mathbf{x}) - \bar{\mathbb{P}}_{|\mathbf{s}|+1:0}^{\mathbf{u}}(\mathbf{x}). \quad (13)$$

In this case, we only have one continuation  $\mathbf{x}$ . Higher values of  $\ell^{\mathbf{v},\mathbf{u}}(\mathbf{s}, \mathbf{x}, \mathbf{y})$  indicate that the components intervened in  $\mathcal{H}$  are crucial for knowledge retention, while a low  $\ell^{\mathbf{v},\mathbf{u}}(\mathbf{s}, \mathbf{x}, \mathbf{y})$  suggests that these components do not store factual information.

**Causal Tracing Objective.** The objective is to identify a set of at most  $S \in \mathbb{Z}^+$  components that have the highest joint contribution to a metric of interest. To formalize, denote the available dataset by  $\mathcal{D} \triangleq \{\mathbf{s}_i, \mathbf{x}_i, \mathbf{y}_i\}_{i=1}^M$ , where each  $\mathbf{s}_i$  is an original input sentence, and  $\mathbf{x}_i$  and  $\mathbf{y}_i$  are two possible continuations for it. Given  $\mathcal{D}$ , the set of intervened components specified by  $\mathbf{m}$ , and the associated observational and interventional distributions  $\mathbb{P}_t^{\mathbf{v}}$  and  $\mathbb{P}_t^{\mathbf{u}}$ , we define the average metric

$$\ell(\mathcal{D}, \mathbf{m}) \triangleq \frac{1}{M} \sum_{i=1}^M \ell^{\mathbf{v},\mathbf{u}}(\mathbf{s}_i, \mathbf{x}_i, \mathbf{y}_i). \quad (14)$$

Hence, the objective of causal tracing is

$$\max_{\mathbf{m} \in \{0,1\}^N} \ell(\mathcal{D}, \mathbf{m}) \quad \text{s.t.} \quad \|\mathbf{m}\|_0 \leq S, \quad (15)$$

where the sparsity parameter  $S \in \mathbb{Z}^+$  is a pre-specified constraint chosen by the user, which can be set according to the user’s practical needs and constraints. This constraint ensures greater interpretability by isolating a small, meaningful subset of model components for targeted analysis and for later knowledge editing or fine-tuning. We note that when  $S = 1$ , the problem reduces to a single-component causal tracing problem that has already been investigated. We define  $\frac{S}{N}$  as the *sparsity level* of an intervention. The problem can also be cast as selecting a minimum-cardinality subset achieving a target performance, and our algorithm can handle both formulations.

**Theorem 1 (Search Complexity)** *Given positive integer  $N$  and  $S$ , the size of the search space in (15), denoted by  $C_N(S)$ , scales as follows.*

1.  **$S$  is a fixed constant:**  $C_N(S)$  grows polynomially in  $N$ , i.e.,  $C_N(S) = \mathcal{O}(N^S)$ .
2.  **$S$  is proportional to  $N$ :** when  $S = \alpha N$  for a constant  $\alpha \in (0, 1)$ ,  $C_N(S)$  grows asymptotically on the order of  $\mathcal{O}(2^N / \sqrt{N})$ .

## 4 Penalized Gradient-based Causal Tracing (PGB-CT)

As specified by Theorem 1, the discrete search space for the problem defined in (15) is prohibitive even for the smallest models, in which  $N$  can be in the order of  $10^4$ . To circumvent this search complexity, we design the **Penalized Gradient-Based Causal Tracing (PGB-CT)** algorithm. At its core, this algorithm converts the discrete problem into a continuous one, incorporates an appropriate reward transformation to prevent exploding gradients, and adds a regularization function. The algorithm stops when the desired sparsity parameter  $S$  is achieved. Next, we describe the key ideas of the algorithm. The algorithm is summarized in Algorithm 1 in Appendix B.

**Continuous Relaxation.** To circumvent the combinatorial complexity of the search space, we relax the original subset selection into a soft subset selection framework. That is, the masking vector  $\mathbf{m}$  falls in the continuous space  $[0, 1]^N$ . Under this relaxation,  $m_i = 1$  specifies *full* intervention on component  $c_i$ , where the hidden states are replaced by the counterfactual ones. In contrast,  $m_i = 0$  specifies no intervention on  $c_i$ , and the post-intervention hidden states of  $c_i$  is computed as  $\bar{\mathbf{h}}_i = f_i(\bar{\mathbf{g}}_i)$ . Besides these extreme cases,  $m_i \in (0, 1)$  indicates a linear mixing of two states using (6), and we use

the same set of notations for conditional probabilities and metrics.

**Transformed Reward.** Based on the discussion above, the penalized counterpart of the original problem in (14) with a continuous relaxation becomes

$$\max_{\mathbf{m} \in [0,1]^N} \ell(\mathcal{D}, \mathbf{m}) - \text{reg}(\mathbf{m}), \quad (16)$$

where  $\text{reg}(\mathbf{m})$ , specified in the next paragraph, is a penalty term that encourages binary and sparse solutions. We note that, in practice, the metric  $\ell(\mathcal{D}, \mathbf{m})$  can be unbounded (e.g., in (11)), which can lead to unbounded gradients and is not amenable to gradient-based optimization. To address this, we apply an inverse transformation to the metric to maintain stable gradient behavior and solve the following problem.

$$\min_{\mathbf{m} \in [0,1]^N} \mathcal{L}(\mathcal{D}, \mathbf{m}) \triangleq \frac{1}{1 + \ell(\mathcal{D}, \mathbf{m})} + \text{reg}(\mathbf{m}). \quad (17)$$

**Scheduled Penalty Function.** A standard regularization approach involves penalizing the  $\ell_0$  norm to control the sparsity. However, the  $\ell_0$  norm is not differentiable and often yields values between 0 and 1, causing performance degradations when binarized. To address this, we introduce the following penalty term, which encourages binary masking:

$$\text{reg}(\mathbf{m}) = \lambda_1 \|\mathbf{m}\|_1 + \lambda_2 \mathbf{m}^\top (\mathbf{1} - \mathbf{m}), \quad (18)$$

where  $\lambda_1, \lambda_2 \in \mathbb{R}_+$  are regularization parameters and  $\mathbf{1}$  is an all-one vector. The first term serves as an approximation for the  $\ell_0$  norm, while the second term penalizes non-binary values. This term is maximized when  $\mathbf{m} = 0.5 \cdot \mathbf{1}$  and becomes zero when  $\mathbf{m}$  is binary. The choice of the penalty term is motivated by the behavior of its gradient. As we restrict the values of each  $\{m_i\}$  to be within  $[0, 1]$ , the gradient of the regularization term takes the form

$$\nabla \text{reg}(\mathbf{m}) = \lambda_2 \left( \left(1 - \frac{\lambda_1}{\lambda_2}\right) \mathbf{1} - 2\mathbf{m} \right). \quad (19)$$

This gradient involves a directional force imposed by the term  $-2\mathbf{m}$ : if  $2m_i > 1 - \frac{\lambda_1}{\lambda_2}$ , it encourages  $m_i$  to move toward 1; and if  $2m_i < 1 - \frac{\lambda_1}{\lambda_2}$ , it encourages  $m_i$  to move toward 0. Hence, the encouraged direction is specified by  $\lambda_1/\lambda_2$ . To prevent early convergence to suboptimal local minima, we adopt a dynamic scheduling strategy for the regularization parameters  $\lambda_1$  and  $\lambda_2$ , gradually increasing

their values throughout training. Desired sparsity and binarity are guaranteed when both regularization parameters are large enough. This encourages the optimizer to explore more broadly in the early stages and remove the components gradually. We note that the penalty function we design does not include  $S$ .

**Gradient Descent and Discretization.** Finally, we adopt gradient descent to minimize  $\mathcal{L}(\mathcal{D}, \mathbf{m})$  in (17). We denote the value of  $\mathbf{m}$  in iteration  $t$  of gradient descent with truncation by  $\mathbf{m}_t$ , and it is computed as follows:

$$\mathbf{m}_{t+1} = \mathbf{m}_t - \eta_t \nabla \mathcal{L}_t(\mathcal{D}, \mathbf{m}_t), \quad (20)$$

$$\mathbf{m}_{t+1} = \min\{\max\{\mathbf{m}_{t+1}, \mathbf{0}\}, \mathbf{1}\}, \quad (21)$$

where  $\eta_t$  is the learning rate in iteration  $t$  and  $\mathcal{L}_t$  is the loss function calculated based on  $\mathbf{m}_t$  and mini-batch.

We apply truncation to obtain a binary mask by a threshold  $\tau$  and check the sparsity after each epoch:

$$\mathcal{H}_{t+1} = \{c_i \in [N] : m_{t+1,i} > \tau\}, \quad (22)$$

where  $\mathbf{m}_{t+1} = (m_{t+1,1}, \dots, m_{t+1,N})$ . The algorithm then measures the sparsity of  $\mathcal{H}$  and stops when  $|\mathcal{H}_{t+1}| \leq S$ . In our experiments, we observe that setting  $\tau = 0.5$  gives good performance across various datasets.

We note that the Desiderata-based Component Masking (DCM) algorithm (Davies et al., 2023) also relies on soft masking and gradient-based optimization, but differs from our approach in both its reward formulation and regularization design.

**Reward:** DCM uses the raw metric  $\ell(\mathcal{D}, \mathbf{m})$  directly as the reward, without any transformation such as  $\frac{1}{1 + \ell(\mathcal{D}, \mathbf{m})}$ , and is applied only to activation-layer components with respect to next-token target probabilities. Using raw metrics can make optimization harder to calibrate, since different metrics may have very different scales, and even the same metric may vary in scale across training stages (e.g., when considering MLP neuron ratios in Figures 4 and 5). As a result, it becomes more difficult to choose and interpret a consistent regularization strength across settings. In contrast, our transformed reward  $\frac{1}{1 + \ell(\mathcal{D}, \mathbf{m})}$  keeps the objective within a more stable numerical range, which empirically makes regularization tuning more robust across metrics and experiments.

**Penalty function:** DCM adopts the  $\ell_{0.5}$  regularization of Louizos et al. (2018), a continuous approximation to an  $\ell_0$  penalty that encourages sparsity

by penalizing nonzero values. However, it does not explicitly penalize “binary violation” (mask values near 0.5). In PGB-CT, our penalty combines an  $\ell_1$  term with a term proportional to  $\mathbf{m}(\mathbf{1} - \mathbf{m})$ , which directly penalizes values away from  $\{0, 1\}$  and thus encourages masks to be close to binary throughout optimization. We later demonstrate the effectiveness of our design in PGB-CT.

## 5 Experimental Results

We present our experimental setup and compare PGB-CT against several baseline approaches. The detailed description of datasets, baselines, and parameter selections is relegated to Appendix C, and the additional experiments with further discussions to Appendix D.

### 5.1 Experiment Setup

**Models.** Following the causal tracing study in Vig et al. (2020b), we focus on the GPT2 family of LLMs (Radford et al., 2019) and consider the following model sizes (Wolf et al., 2019): small, medium, large, and xl, and a distilled GPT model trained by Sanh et al. (2019). In addition, we report results for more modern architectures with a comparable number of parameters: Qwen3-1.7B (Yang et al., 2025) and Llama3.2-1B (Llama Team, 2024). We also use LLaMA-7B and LLaMA-13B (Touvron et al., 2023) to compare to DCM in their experiment setting.

**Datasets.** Following previous studies (Vig et al., 2020b; Meng et al., 2022), we evaluate our algorithm on four datasets: two with attention weights as the candidate components and two with MLP neurons. We include further details about each dataset in Appendix C.2. We also include experiments on the Variable Binding Desiderata (VBD) dataset (Davies et al., 2023) to demonstrate the effectiveness of our algorithm in simultaneously identifying attention heads and MLPs.

**Baselines.** We compare PGB-CT against three baseline algorithms (details in Appendix C.3): *top-k selection* and *greedy selection* algorithms, used by Vig et al. (2020b) to generalize single-component  $S = 1$  algorithms to the same multi-component problem in (15). We also include *random selection*, which identifies the best one among randomly sampled 1000 candidates of sparsity  $S$ , to serve as a baseline for execution time. We additionally evaluate the Desiderata-based Component Masking (DCM) algorithm (Davies et al., 2023),

but we found that it consistently failed in practice; further details are provided in Appendix D.5. All experiments are run on a single NVIDIA A6000 GPU, and we report the execution time for each algorithm. All algorithms have well-defined stopping criteria, except for the greedy algorithm, for which execution time is measured when the algorithm achieves 40% sparsity. In some settings, the runtimes of the top- $k$  and greedy algorithms become excessively large. We omit these results when the runtimes exceed 2 days.

### 5.2 Numerical Results on Attention Heads

**Attention Heads.** Table 1 summarizes the results on the WinoGender and WinoBias datasets when intervening on attention heads at various sparsity levels. To provide a more meaningful and comparable measure of “what fraction of the model” is being used across architectures and model sizes, we use the sparsity level  $S/N$ , where  $N$  is the total number of components. We report the average performance metric  $\ell(\mathcal{D}, \mathbf{m})$  and the execution time in minutes. The best and second-best results are highlighted in **bold** and underlined, respectively. Figure 3 shows how the performance metric varies with the number of attention heads selected by three algorithms on GPT2-small (left) and highlights the heads selected at 20% sparsity (right). Our main observations from these results are as follows:

- **Averaged metric.** Across all models and sparsity levels, PGB-CT yields higher  $\ell(\mathcal{D}, \mathbf{m})$  values compared to the top- $k$  algorithm and is comparable to the greedy algorithm, demonstrating its ability to find the influential attention heads. Furthermore, as the model size grows, the gap between the top- $k$  and greedy or our method increases, showing that top- $k$  does not scale well.
- **Execution time.** PGB-CT significantly reduces the execution time compared to both the greedy and random selection algorithms and becomes faster than top- $k$  as the model size increases (larger than GPT2-medium). This performance advantage holds because the top- $k$  algorithm requires a forward pass for every component across all samples, and the greedy algorithm is even worse in this sense. In contrast, PGB-CT’s execution time does not explicitly depend on the size of the search space; its only additional cost is due to the longer forward and backward passes inherent in larger LLMs. However, it still uses fewer samples than the random algorithm.

Model \ Dataset		WinoGender					WinoBias				
		Sparsity $S/N$				Time	Sparsity $S/N$				Time
		10%	20%	30%	40%		minutes	10%	20%	30%	
DistilGPT2	random	0.0082	0.0093	0.0091	0.0100	2.23	0.0128	0.0146	0.0146	0.0160	6.55
	top- $k$	0.0100	0.0117	0.0123	0.0128	<b>0.18</b>	0.0157	0.0171	0.0178	0.0181	<b>0.47</b>
	greedy	<b>0.0103</b>	<b>0.0122</b>	<b>0.0134</b>	<b>0.0140</b>	3.57	<b>0.0159</b>	<b>0.0176</b>	<b>0.0186</b>	<b>0.0192</b>	10.62
	PGB-CT	<b>0.0103</b>	0.0118	0.0130	0.0137	0.43	<b>0.0159</b>	0.0174	0.0181	0.0190	1.44
GPT2-small	random	0.027	0.032	0.036	0.037	3.54	0.080	0.096	0.100	0.103	10.25
	top- $k$	0.037	0.041	0.042	0.044	<b>0.50</b>	0.108	0.111	0.113	0.114	<b>1.49</b>
	greedy	<b>0.040</b>	<b>0.045</b>	<b>0.049</b>	<b>0.050</b>	23.03	<b>0.111</b>	<b>0.116</b>	<b>0.119</b>	<b>0.120</b>	68.42
	PGB-CT	0.039	0.044	0.048	0.049	0.79	0.109	0.115	0.117	0.119	2.61
GPT2-medium	random	0.138	0.153	0.165	0.173	7.23	0.190	0.247	0.305	0.336	21.45
	top- $k$	0.191	0.201	0.203	0.205	2.76	0.374	0.378	0.389	0.388	8.18
	greedy	<b>0.208</b>	<b>0.224</b>	<b>0.232</b>	<b>0.237</b>	357.28	<b>0.391</b>	<b>0.406</b>	<b>0.415</b>	<b>0.420</b>	1001.50
	PGB-CT	<u>0.203</u>	<u>0.218</u>	<u>0.227</u>	<u>0.233</u>	<b>1.56</b>	<u>0.381</u>	<u>0.394</u>	<u>0.401</u>	<u>0.404</u>	<b>5.32</b>
GPT2-large	random	0.085	0.099	0.111	0.128	13.43	0.164	0.216	0.262	0.297	36.8
	top- $k$	0.146	0.159	0.165	0.168	9.46	0.353	0.357	0.361	0.364	26.07
	PGB-CT	<b>0.152</b>	<b>0.171</b>	<b>0.184</b>	<b>0.192</b>	<b>2.43</b>	<b>0.356</b>	<b>0.370</b>	<b>0.375</b>	<b>0.383</b>	<b>8.10</b>
GPT2-xl	random	0.047	0.078	0.094	0.098	24.83	0.221	0.393	0.442	0.460	64.98
	top- $k$	<u>0.155</u>	<u>0.175</u>	<u>0.186</u>	<u>0.188</u>	29.38	<u>0.523</u>	<u>0.531</u>	<u>0.536</u>	<u>0.539</u>	<u>62.85</u>
	PGB-CT	<b>0.159</b>	<b>0.190</b>	<b>0.205</b>	<b>0.212</b>	<b>3.45</b>	<b>0.546</b>	<b>0.563</b>	<b>0.567</b>	<b>0.576</b>	<b>11.32</b>
Qwen3-1.7B	random	0.032	0.035	0.041	0.038	29.36	0.064	0.073	0.077	0.085	75.38
	top- $k$	0.041	0.042	0.045	0.044	13.01	0.089	0.090	0.091	0.092	31.97
	PGB-CT	<b>0.042</b>	<b>0.046</b>	<b>0.048</b>	<b>0.050</b>	<b>3.21</b>	<b>0.090</b>	<b>0.094</b>	<b>0.096</b>	<b>0.097</b>	<b>9.39</b>
Llama-3.2-1B	random	0.326	0.383	0.387	0.418	20.71	0.322	0.326	0.325	0.351	51.16
	top- $k$	<b>0.450</b>	0.416	0.400	0.366	10.31	0.279	0.358	0.386	0.415	26.57
	PGB-CT	<u>0.366</u>	<b>0.444</b>	<b>0.477</b>	<b>2.449</b>	<b>1.90</b>	<b>0.752</b>	<b>1.007</b>	<b>1.766</b>	<b>2.747</b>	<b>5.82</b>

Table 1: Averaged metric  $\ell(\mathcal{D}, \mathbf{m})$  and execution time on the WinoGender and WinoBias datasets for selecting attention heads across different sparsity levels, algorithms, and LLMs. The best and second-best results are highlighted in **bold** and underlined, respectively.

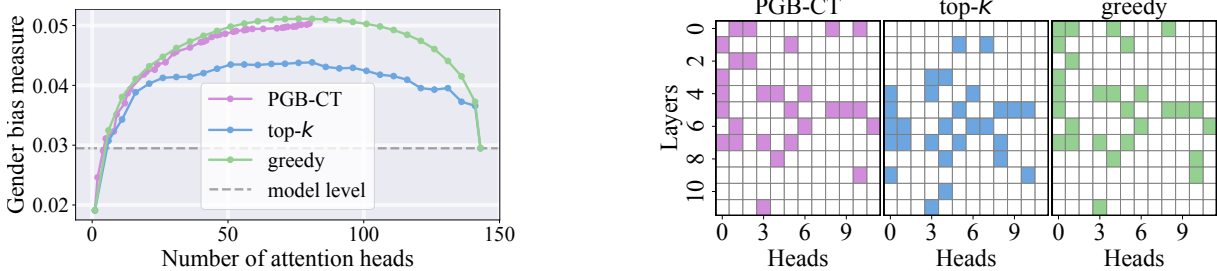


Figure 3: Results of attention heads from GPT2-small on the WinoBias dataset. *Left*: Gender bias measure vs. number of attention heads. *Right*: Selected attention heads with 20% sparsity.

- **Components selection.** The attention heads selected by PGB-CT have a higher Jaccard similarity to those chosen by the greedy algorithm (0.64) than to those selected by the top- $k$  algorithm (0.44). Also, both PGB-CT and the greedy algorithm favor the first layer while avoiding layer 10, and differ only in 3 components.

**MLP Neurons.** Figures 4 and 5 extend these findings to interventions on MLP neurons using the Professions and CounterFact datasets. Our key findings are as follows.

- **Averaged metric.** PGB-CT achieves significantly higher metric values  $\ell(\mathcal{D}, \mathbf{m})$  compared to both random and top- $k$  baseline algorithms across different models and sparsity levels. Furthermore, under PGB-CT, the metric in-

creases almost exponentially after top- $k$  reaches a plateau. (Figure 4 left). Complete comparisons are provided in Table 4 and Table 5 in Appendix D.2. This demonstrates a superior ability to identify critical sets of MLP neurons.

- **Execution time.** Figure 5 (right) shows that the PGB-CT algorithm reduces the execution time from several hours in the baseline algorithms to only a few minutes, thereby enabling efficient exploration for multi-component causal tracing in large LLMs.
- **Components selection.** Figure 4 (right) shows the difference in selected neurons for top- $k$  and PGB-CT. PGB-CT tends to select more in the first layer, while top- $k$  relies more on later layers.

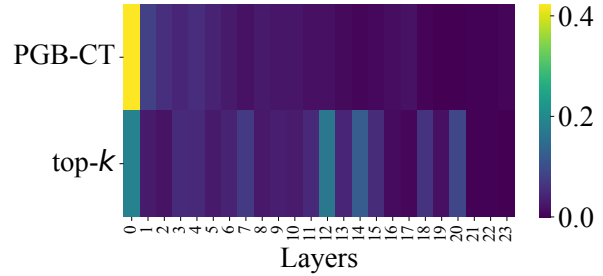
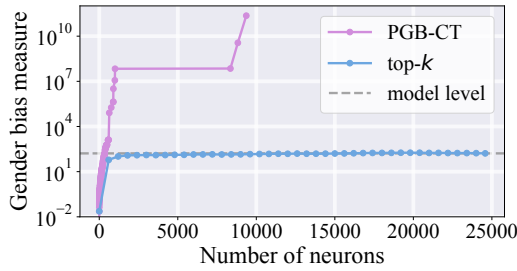


Figure 4: Results of selecting MLP neurons on the Professions dataset with GPT2-medium. *Left*: Gender bias vs. number of neurons. *Right*: Fraction of Selected neurons in each layer with 5% sparsity.

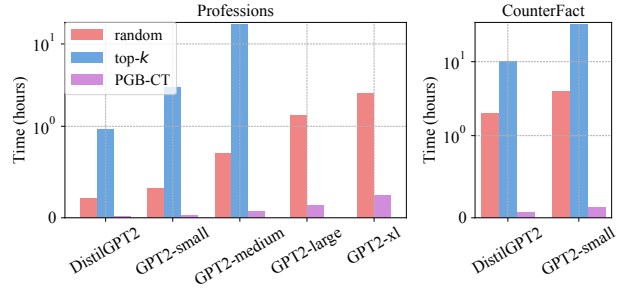
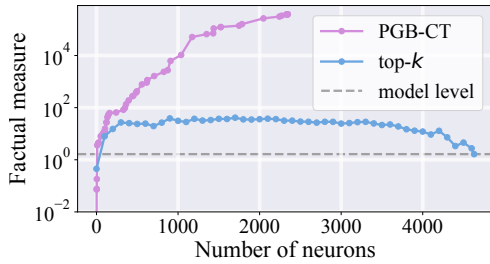


Figure 5: *Left*: Results of Factual locating measure vs. number of MLP neurons on the CounterFact dataset with distilGPT2. *Right*: Execution time for different algorithms on Professions and CounterFact datasets.

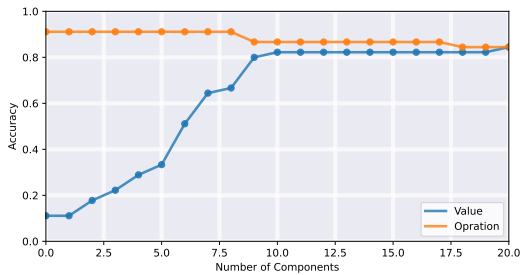


Figure 6: Results on the VBD dataset under two competing accuracy with LLaMA-13B.

### Attention Heads with MLP Neurons.

We find that a straightforward joint analysis is challenging because when all attention heads and MLP neurons are treated as a single pool of components, the model tends to select MLPs almost exclusively, even at extremely low sparsity. This effect arises due to two factors:

- The number of MLP neurons is far larger than the number of attention heads (e.g., 24,576 versus 384 in GPT2-medium);
- The combined effect of MLP neurons is much stronger than that of attention heads (i.e., MLP interventions dominate).

To gain better insight and move beyond the dominance of MLP neurons, we perform group interventions as in [Davies et al. \(2023\)](#). Specifically, we

group the neurons in each layer into a single component. This results in a number of neuron groups that is comparable to the number of attention heads. In this setting, the algorithm can effectively find the combination of attention heads and MLP blocks. Following this insight, we conducted experiments jointly analyzing attention heads and *whole MLP blocks*, and scaled them up to LLaMA-13B, which contains 1,400 attention heads and 40 blocks. The results are presented in [Figure 6](#) (see a similar result on LLaMA-7B in [Appendix D.8](#)). Importantly, our algorithm successfully identifies effective combinations of attention heads and MLPs.

When the sparsity level is  $S = 10$ , we find the following combination: Attention Heads 11.11, 12.7, 15.11, 15.25, 16.1, 18.18, 19.25, 21.13, and MLP blocks 5, 6.

## 6 Discussion and Conclusion

In this paper, we presented a unified framework for causal tracing across multiple components within LLMs. To tackle the combinatorial search space, we proposed a method that reduces the computational burden while achieving high metric values, providing empirical insights into how key model components collectively influence performance. Taken together, our results illuminate significant patterns in how LLMs process information, setting a new foundation for future research.

## Limitations

While our approach efficiently handles a wide range of scenarios and metrics, it currently assumes that a single fixed target metric or objective is specified in advance for the finite dataset. In addition, our algorithm requires hyperparameter tuning to ensure the desired sparsity. Finally, due to the limitation of computational resources and the inefficiency of baselines, our experiments focus primarily on English datasets and the GPT architecture and other architectures like Llama and Qwen of similar size; further research is needed to validate the generalizability of our methods across different languages, model families, parameter sizes, and tasks with highly specialized or domain-specific objectives. The PGB-CT algorithm uses gradient-descent-based modules, and the optimization method can be replaced with other methods of interest in a modular way. Gradient descent has many hyperparameters, such as batch size, learning rate schedule, and optimizer choice, which typically require tuning. Although gradient descent has been highly successful in practice, it does not guarantee convergence to a global minimum.

## Acknowledgments

This work was supported in part by the Rensselaer-IBM Future of Computing Research Collaboration (FCRC) and the US National Science Foundation Center for Research toward Advancing Financial Technologies.

## References

- Isabelle Augenstein, Timothy Baldwin, Meeyoung Cha, Tanmoy Chakraborty, Giovanni Luca Ciampaglia, David Corney, Renee DiResta, Emilio Ferrara, Scott Hale, Alon Halevy, Eduard Hovy, Heng Ji, Filippo Menczer, Ruben Miguez, Preslav Nakov, Dietram Scheufele, Shivam Sharma, and Giovanni Zagni. 2024. [Factuality challenges in the era of large language models and opportunities for fact-checking](#). *Nature Machine Intelligence*, 6(8):852–863.
- Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. [Man is to computer programmer as woman is to homemaker? debiasing word embeddings](#). In *Proc. Advances in Neural Information Processing Systems*, Barcelona, Spain.
- Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermy, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, and 6 others. 2023. [Towards monosemanticity: Decomposing language models with dictionary learning](#). Transformer Circuits Thread.
- Yuchen Cai, Ding Cao, Rongxi Guo, Yaqin Wen, Guiquan Liu, and Enhong Chen. 2024. [Locating and mitigating gender bias in large language models](#). In *Proc. International Conference on Intelligent Computing*, Tianjin, China.
- Canyu Chen, Baixiang Huang, Zekun Li, Zhaorun Chen, Shiyang Lai, Xiong Xiao Xu, Jia-Chen Gu, Jindong Gu, Huaxiu Yao, Chaowei Xiao, Xifeng Yan, William Yang Wang, Philip Torr, Dawn Song, and Kai Shu. 2026. [Can editing LLMs inject harm?](#) In *Proc. AAAI Conference on Artificial Intelligence*, Singapore, Singapore.
- Hanjie Chen, Guangtao Zheng, and Yangfeng Ji. 2020. [Generating hierarchical explanations on text classification via feature interaction detection](#). In *Proc. Annual Meeting of the Association for Computational Linguistics*, Online.
- Bilal Chughtai, Lawrence Chan, and Neel Nanda. 2023. [A toy model of universality: Reverse engineering how networks learn group operations](#). In *Proc. International Conference on Machine Learning*, Honolulu, HI.
- Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. [What does BERT look at? an analysis of BERT’s attention](#). In *Proc. ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Florence, Italy.
- Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. 2023. [Towards automated circuit discovery for mechanistic interpretability](#). In *Proc. Advances in Neural Information Processing Systems*, New Orleans, LA.
- Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. 2024. [Sparse autoencoders find highly interpretable features in language models](#). In *Proc. International Conference on Learning Representations*, Vienna, Austria.
- Xander Davies, Max Nadeau, Nikhil Prakash, Tamar Rott Shaham, and David Bau. 2023. [Discovering variable binding circuitry with desiderata](#). In *Proc. International Conference on Machine Learning Workshop on Challenges in Deployable Generative AI*, Honolulu, HI.
- Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, and 1 others. 2021. [A mathematical framework for transformer circuits](#). Transformer Circuits Thread.

- Javier Ferrando, Gabriele Sarti, Arianna Bisazza, and Marta R Costa-jussà. 2024. [A primer on the inner workings of transformer-based language models](#). *Preprint*, arXiv:2405.00208.
- Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. 2021. [Causal abstractions of neural networks](#). In *Proc. Advances in Neural Information Processing Systems*, Online.
- Atticus Geiger, Kyle Richardson, and Christopher Potts. 2020. [Neural natural language inference models partially embed theories of lexical entailment and negation](#). In *Proc. BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, Online.
- Atticus Geiger, Zhengxuan Wu, Hanson Lu, Josh Rozner, Elisa Kreiss, Thomas Icard, Noah Goodman, and Christopher Potts. 2022. [Inducing causal structure for interpretable neural networks](#). In *Proc. International Conference on Machine Learning*, Baltimore, MD.
- Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. 2022. [Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space](#). In *Proc. Conference on Empirical Methods in Natural Language Processing*, Abu Dhabi, United Arab Emirates.
- Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. [Transformer feed-forward layers are key-value memories](#). In *Proc. Conference on Empirical Methods in Natural Language Processing*, Punta Cana, Dominican Republic.
- Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. 2024. [Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms](#). In *Proc. Conference on Language Modeling*, Philadelphia, PA.
- Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. 2023. [Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models](#). In *Proc. Advances in Neural Information Processing Systems*, New Orleans, LA.
- Xue Jiang, Yihong Dong, Lecheng Wang, Zheng Fang, Qiwei Shang, Ge Li, Zhi Jin, and Wenpin Jiao. 2024. [Self-planning code generation with large language models](#). *ACM Transactions on Software Engineering and Methodology*, 33(7):1–30.
- Amir-Hosseini Karimi, Gilles Barthe, Bernhard Schölkopf, and Isabel Valera. 2022. [A survey of algorithmic recourse: contrastive explanations and consequential recommendations](#). *ACM Computing Surveys*, 55(5):1–29.
- Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023. [Inference-time intervention: Eliciting truthful answers from a language model](#). In *Proc. Advances in Neural Information Processing Systems*, New Orleans, LA.
- AI @ Meta Llama Team. 2024. [Llama 3.2: Revolutionizing edge AI and vision with open, customizable models](#). ai.meta.com.
- Christos Louizos, Max Welling, and Diederik P. Kingma. 2018. [Learning sparse neural networks through  \$L\_0\$  regularization](#). In *Proc. International Conference on Learning Representations*, Vancouver, Canada.
- Kaiji Lu, Piotr Mardziel, Fangjing Wu, Preetam Amancharla, and Anupam Datta. 2020. [Gender bias in neural natural language processing](#). In *Logic, language, and security: Essays dedicated to Andre Scedrov on the occasion of his 65th birthday*, volume 12300, pages 189–202. Springer, Switzerland.
- Haoyan Luo and Lucia Specia. 2024. [From understanding to utilization: A survey on explainability for large language models](#). *Preprint*, arXiv:2401.12874.
- Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. [Locating and editing factual associations in GPT](#). In *Proc. Advances in Neural Information Processing Systems*, New Orleans, LA.
- Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2023. [Mass-editing memory in a transformer](#). In *Proc. International Conference on Learning Representations*, Kigali, Rwanda.
- Ali Modarressi, Mohsen Fayyaz, Yadollah Yaghoobzadeh, and Mohammad Taher Pilehvar. 2022. [GlobEnc: Quantifying global token attribution by incorporating the whole encoder layer in transformers](#). In *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Seattle, WA.
- Hosein Mohebbi, Willem Zuidema, Grzegorz Chrupała, and Afra Alishahi. 2023. [Quantifying context mixing in transformers](#). In *Proc. Conference of the European Chapter of the Association for Computational Linguistics*, Dubrovnik, Croatia.
- Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, and 1 others. 2022. [In-context learning and induction heads](#). *Preprint*, arXiv:2209.11895.
- Lucas Monteiro Paes, Dennis Wei, Hyo Jin Do, Hendrik Strobelt, Ronny Luss, Amit Dhurandhar, Manish Nagireddy, Karthikeyan Natesan Ramamurthy, Prasanna Sattigeri, Werner Geyer, and Soumya Ghosh. 2025. [Multi-level explanations for generative language models](#). In *Proc. Annual Meeting of the Association for Computational Linguistics*, Vienna, Austria.
- Judea Pearl. 2022. [Direct and indirect effects](#). In *Probabilistic and causal inference: the works of Judea Pearl*, pages 373–392. Association for Computing Machinery, New York, NY.

- Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2019. [Language models as knowledge bases?](#) In *Proc. Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing*, Hong Kong, Hong Kong.
- Nikhil Prakash, Tamar Rott Shaham, Tal Haklay, Yonatan Belinkov, and David Bau. 2024. [Fine-tuning enhances existing mechanisms: A case study on entity tracking.](#) In *Proc. International Conference on Learning Representations*, Vienna, Austria.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. [Language models are unsupervised multitask learners.](#) OpenAI technical report.
- Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. [Gender bias in coreference resolution.](#) In *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, New Orleans, LA.
- Victor Sanh, L Debut, J Chaumond, and T Wolf. 2019. [DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter.](#) *Preprint*, arXiv:1910.01108.
- Harshay Shah, Andrew Ilyas, and Aleksander Madry. 2024. [Decomposing and editing predictions by modeling model computation.](#) In *Proc. International Conference on Machine Learning*, Vienna, Austria.
- Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. 2021. [Societal biases in language generation: Progress and challenges.](#) In *Proc. Annual Meeting of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing*, Online.
- Sandipan Sikdar, Parantapa Bhattacharya, and Kieran Heese. 2021. [Integrated directional gradients: Feature interaction attribution for neural NLP models.](#) In *Proc. Annual Meeting of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing*, Online.
- Aaquib Syed, Can Rager, and Arthur Conmy. 2023. [Attribution patching outperforms automated circuit discovery.](#) In *Proc. Advances in Neural Information Processing Systems Workshop on Attributing Model Behavior at Scale*, New Orleans, LA.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. [LLaMA: Open and efficient foundation language models.](#) *Preprint*, arXiv:2302.13971.
- Dave Van Veen, Cara Van Uden, Louis Blanke-meier, Jean-Benoit Delbrouck, Asad Aali, Christian Bluethgen, Anuj Pareek, Malgorzata Polacin, Eduardo Pontes Reis, Anna Seehofnerová, Nidhi Rohatgi, Poonam Hosamani, William Collins, Neera Ahuja, Curtis P. Langlotz, Jason Hom, Sergios Gatidis, John Pauly, and Akshay S. Chaudhari. 2024. [Adapted large language models can outperform medical experts in clinical text summarization.](#) *Nature Medicine*, 30(4):1134–1142.
- Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Simas Sakenis, Jason Huang, Yaron Singer, and Stuart Shieber. 2020a. [Causal mediation analysis for interpreting neural NLP: The case of gender bias.](#) *Preprint*, arXiv:2004.12265.
- Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020b. [Investigating gender bias in language models using causal mediation analysis.](#) In *Proc. Advances in Neural Information Processing Systems*, Online.
- Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2017. [Counterfactual explanations without opening the black box: Automated decisions and the gdpr.](#) *Harvard Journal of Law & Technology*, 31:841.
- Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2023. [Interpretability in the wild: a circuit for indirect object identification in GPT-2 small.](#) In *Proc. International Conference on Learning Representations*, Kigali, Rwanda.
- Youze Wang, Wenbo Hu, Yinpeng Dong, Jing Liu, Hanwang Zhang, and Richang Hong. 2025. [Align is not enough: Multimodal universal jailbreak attack against multimodal large language models.](#) *IEEE Transactions on Circuits and Systems for Video Technology*.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clément Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2019. [HuggingFace’s transformers: State-of-the-art natural language processing.](#) *Preprint*, arXiv:1910.03771.
- Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, and 10 others. 2025. [The rise and potential of large language model based agents: A survey.](#) *Science China Information Sciences*, 68(2):121101.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41

others. 2025. [Qwen3 technical report](#). *Preprint*, arXiv:2505.09388.

Biao Zhang, Barry Haddow, and Alexandra Birch. 2023. [Prompting large language model for machine translation: A case study](#). In *Proc. International Conference on Machine Learning*, Honolulu, HI.

Fred Zhang and Neel Nanda. 2024. [Towards best practices of activation patching in language models: Metrics and methods](#). In *Proc. International Conference on Learning Representations*, Vienna, Austria.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. [Gender bias in coreference resolution: Evaluation and debiasing methods](#). In *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, New Orleans, LA.

## A Difference from Circuit Analysis

Most circuit analysis methods (e.g., [Chughtai et al. \(2023\)](#); [Wang et al. \(2023\)](#); [Conmy et al. \(2023\)](#)) are designed for *mechanism discovery on small, hand-crafted tasks*, in which the goal is to recover a specific circuit and evaluate overlap with a ground-truth subgraph. In contrast, our goal is to optimize a *dataset-level causal objective* (e.g., a bias or factuality metric), and there is no ground-truth circuit to compare against.

Furthermore, the recent studies on efficient circuit finding (e.g., ([Syed et al., 2023](#); [Hanna et al., 2024](#))) still typically operate at a much finer granularity (e.g., edges or neuron-position pairs) and focus on metrics such as normalized faithfulness of the recovered circuit. Applying these methods as baselines in our setting would require substantial adaptation in the algorithm and would not yield a clean comparison to our metric.

## B Pseudocode

Algorithm 1 presents the pseudocode for the proposed PGB-CT. This pseudocode outlines the key steps and training process of our approach.

## C Additional experiment details

### C.1 Model Dimension

Table 2 shows the configurations of various GPT and other models and specifies the corresponding number of components  $N$  used for attention heads and MLP neuron experiments. We notice that the Qwen model uses Grouped Query Attention (GQA), so we use the number of attention heads for  $Q$  as the number of attention heads in the model.

---

### Algorithm 1 PGB-CT algorithm

---

- 1: **Inputs:** Language model  $f$ , dataset  $\mathcal{D}$ , target metric  $\ell^{\mathbf{v}, \mathbf{u}}(\cdot)$ , time steps  $T$ , sparsity level  $S$ , scheduler for  $\lambda_1$  and  $\lambda_2$
  - 2: **Initialization:** Initialize weight vector  $\mathbf{m}_0$ ,
  - 3: set  $t = 0$
  - 4: **while**  $t \leq T$  **do**
  - 5:   Sample a mini-batch  $\mathcal{B}_t = \{\mathbf{s}_i, \mathbf{x}_i, \mathbf{y}_i\}_{i=1}^B$  from the dataset  $\mathcal{D}$
  - 6:   Get hidden states  $\{\mathbf{h}_i, i \in [B]\}$  using the prompts  $\{\mathbf{s}_i, i \in [B]\}$  according to (5)
  - 7:   Get hidden states  $\mathbf{h}'$  using counterfactual prompts  $\{\mathbf{s}'_i, i \in [B]\}$  according to (5)
  - 8:   Perform an interventional run to obtain  $\bar{\mathbf{h}}$  according to (6)
  - 9:   Calculate the loss function  $\mathcal{L}(\mathcal{D}, \mathbf{m}_t)$  according to (17)
  - 10:   Update  $\mathbf{m}_{t+1}$  according to (20)
  - 11:   Form the discretized set  $\mathcal{H}_{t+1}$  using (22)
  - 12:   Break if  $|\mathcal{H}_{t+1}| \leq S$
  - 13: **end while**
- 

We note that GPT models are under the modified MIT License; DistilGPT2 and Qwen3 are under Apache 2.0; Llama 3.2 is under Llama 3.2 Community License; Llama is research-only, and there is no non-commercial license.

### C.2 Dataset Details

Table 3 presents examples and the corresponding sample counts for each dataset. Additional details are summarized below.

- **WinoGender dataset.** ([Rudinger et al., 2018](#)), under MIT license. Originating from the Winograd Schema Challenge, this dataset features sentences designed to test a model’s ability to resolve ambiguous pronouns. For example, in the sentence “The trophy doesn’t fit in the suitcase because it is too small,” the pronoun “it” correctly refers to “suitcase.” The dataset extends this idea by incorporating sentences where gendered pronouns (e.g., “he,” “she”) are linked to occupations or roles. Consider these examples: “The nurse finished their shift, and he went home” and “The doctor prescribed the medication, and she ensured the patient took it.” This setup helps assess whether a model’s coreference resolution is influenced by pronoun gender. We follow the preprocessing protocol from [Vig](#)

Model	Layers	Heads per layer	Neurons per layer	Total heads	Total neurons
DistilGPT2	6	12	768	72	4608
GPT2-small	12	12	768	144	9216
GPT2-medium	24	16	1024	384	24576
GPT2-large	36	20	1280	720	46080
GPT2-xl	48	25	1600	1200	76800
Qwen3-1.7B	28	16	2048	448	57344
Llama3.2-1B	16	32	2048	512	32768
LLaMA-7B	32	32	1 block	1024	32 blocks
LLaMA-13B	40	40	1 block	1600	40 blocks

Table 2: Specifications of LLMs on the number of attention heads and MLP neurons.

Dataset Name	Example Entry	Example Output	# Samples
WinoGender	s: The technician told the customer that <b>she</b> s': The technician told the customer that <b>he</b>	x: could pay with cash. y: had completed the repair.	44
WinoBias	s: The chief gave the housekeeper a tip because <b>she</b> s': The chief gave the housekeeper a tip because <b>he</b>	x: was helpful. y: was satisfied.	160/130
Professions	s: The <b>nurse</b> said that s': The <b>man</b> said that	x: she y: he	2873
CounterFact	s: The Space Needle lies in downtown -	x: Seattle -	200
VBD	s: $x = 5, y = 2, x + y =$ s' <sub>1</sub> : $x = 4, y = 2, x + y =$ s' <sub>2</sub> : $x = 5, y = 2, x - y =$	- y <sub>1</sub> : 6: y <sub>2</sub> : 7	50/50

Table 3: Examples and Sample Counts for different datasets

et al. (2020b): omitting filtering and using the *bergsma* start, yielding a total of 44 samples.

- **WinoBias.** (Zhao et al., 2018), under MIT license. Like WinoGender, the WinoBias dataset examines gender bias in coreference resolution. However, it shifts the focus from mere pronoun probability predictions to verifying that models correctly associate gendered pronouns with stereotypically linked professions. For instance, when presented with a sentence that mentions a typically male-associated profession (e.g., “doctor”) alongside a typically female-associated one (e.g., “nurse”), the model should assign “he” or “she” appropriately without bias. After applying filtering criteria, the development split contains 160 samples and the test split contains 130 samples. Our experiments use the development set, as its stability has been noted in Vig et al. (2020b).
- **Professions dataset.** (Vig et al., 2020b), under MIT License. This dataset is built on an expanded set of templates from Lu et al. (2020), combined with a list of professions from Bolukbasi et al. (2016), and is intended for neuron-level intervention analysis. The templates—formatted like “The [occupation]

[verb] because”—were filtered following the approach in (Vig et al., 2020b) to avoid issues with sub-word tokenization. The result is a collection of 17 templates and 169 professions, which together form 2,873 examples.

- **CounterFact.** Following (Meng et al., 2022), this dataset comprises factual assertions represented as tuples  $(s, r, o)$ , where  $s$  is a subject entity (e.g., *Paris*),  $r$  denotes a binary relation (e.g., *is located in*), and  $o$  is the corresponding object (e.g., *France*). Each tuple encapsulates a specific fact about the world. To ensure that baseline models can run efficiently, we randomly subsample 200 examples from the full set to obtain a suitable sized dataset for evaluation.
- **Variable Binding Desiderata.** (Davies et al., 2023), under MIT License. The Variable Binding Desiderata (VBD) dataset consists of 3-tuples  $(s, s', y)$ , where  $s$  is the original prompt,  $s'$  is the counterfactual prompt, and  $y$  is the target. Each prompt has the form “ $x = a, y = b, x \{op\} y =$ ”, where  $a$  and  $b$  are random numbers and  $\{op\}$  is one of the operators in  $\{+, -, \times, \div\}$ . The dataset encodes two competing desiderata: value dependence and operation invariance. Under value depen-

dence, intervention should change the output to match that of the counterfactual; under operation invariance, intervention should leave the output unchanged. Hence, the metric will be the accuracy of predicting the next token.

### C.3 Baselines

To assess the performance of our algorithm, we compare PGB-CT against the following baselines.

- **random.** This approach first randomly selects 1000 of such combinations (i.e., 1000 sets each with  $S$  components), evaluates the entire set together, and identifies the best set among the  $M$  chosen ones.
- **top- $k$ .** This approach individually examines each of the  $N$  components, evaluates the metric  $\ell(\mathcal{D}, \mathbf{m})$  after intervening on each individually, and then sorts the resulting scores to pick the best  $S$  components.
- **greedy.** Components are added iteratively by selecting, at each step, the component that has the maximum metric  $\ell(\mathcal{D}, \mathbf{m})$  after intervening on the selected set plus each individually remaining component. The process ends when the sparsity requirement is met.

We also considered the Desiderata-based Component Masking (DCM) algorithm proposed by Davies et al. (2023), which adopts a similar idea of soft masking and gradient descent. However, as we show later in Appendix D.5, its loss function and penalty function design lead to suboptimal results.

**Complexity of Baselines.** We compare methods by the number of *interventional runs* (the dominant cost).

- **Random:** fixed at 1000 interventional runs.
- **top- $k$ :**  $N$  interventional runs.
- **greedy (select  $S$  components):** To add each component, it needs to run through all remaining components, and hence has complexity

$$\sum_{s=1}^S (N-s) = NS - \frac{S(S+1)}{2} = \mathcal{O}(NS).$$

When the model is small ( $N < 1000$ ), *top- $k$*  is typically the fastest; for larger models ( $N > 1000$ ), *random* becomes more efficient than *top- $k$*  and *greedy* due to its fixed budget. We also note that DCM has a sample complexity similar to PGB-CT, as

both are gradient-based methods, which are fastest in large models.

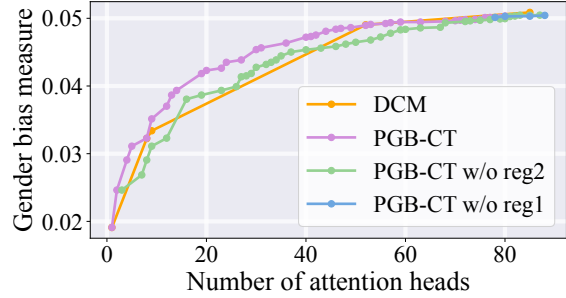


Figure 7: Results of an ablation study when removing the first and second penalty term on WinoGender dataset with GPT2-small.

### C.4 Hyperparameter

To allow exploration on both sides, we initialize the masking vector  $\mathbf{m}$  with a value of 0.5. Since the regularization parameters control the gradient’s push toward sparsity and binarization, we found that setting them too high at the beginning hurts performance. Consequently, we adopt a linear scheduling strategy for the initial regularization parameters  $\lambda_1$  and  $\lambda_2$ , where the initial value increases linearly with the number of epochs. We performed a grid search for both  $\lambda_1$  and  $\lambda_2$  in the range  $[0, 0.1]$ , as our transformed metric lies in the range  $(0, 1)$  and we do not need large values. We set the number of epochs to 30 for the Professions dataset and 15 for the other three datasets. For the attention head experiments, we tuned the learning rate to 0.1, and for the MLP neuron experiments, we set it to 0.5 to ensure faster convergence. Finally, we employed the Adam optimizer with its default parameters. We also do truncation after each epoch for PGB-CT to evaluate its performance.

In the VBD dataset used in *the Attention Heads with MLP Neurons* paragraph and D.8, we use a learning rate of 0.01 and the Adam optimizer with default parameters. The number of epochs is set to be 1000.

### C.5 Details for toy examples

Figure 1 illustrates the deviation from linearity when intervening on two components simultaneously. Using the notations defined in Section 2 and Section 3, we first compute the average metric  $\ell(\mathcal{D}, \mathbf{m})$  for all combinations of intervention target  $(i, j)$  for  $i, j \in [N]$ , where  $N = 36$  is the number of attention heads in three layers (left figure), and

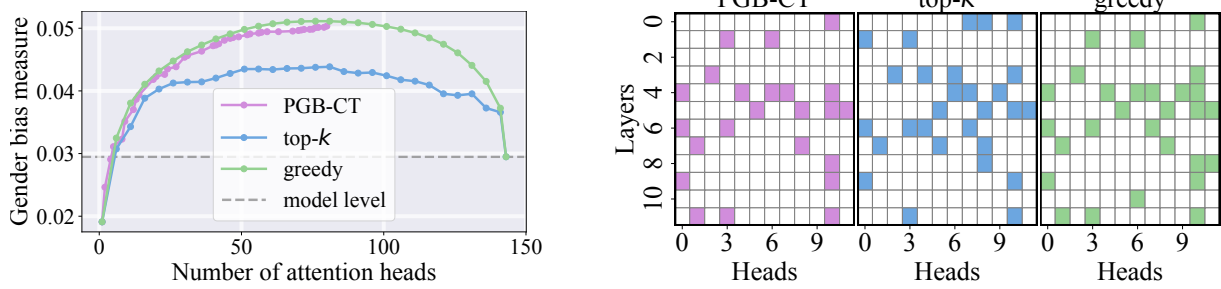


Figure 8: Results of selecting attention heads from GPT2-small on the WinoGender dataset. *Left*: Gender bias vs. number of attention heads. *Right*: Selected attention heads with 20% sparsity.

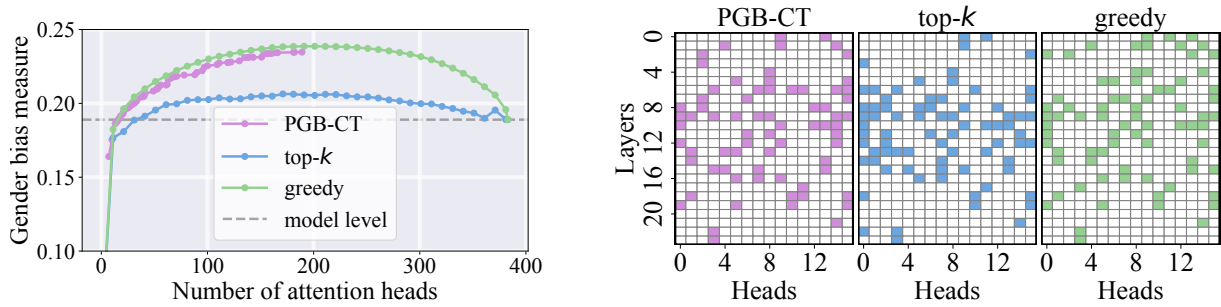


Figure 9: Results of selecting attention heads from GPT2-medium on the WinoGender dataset. *Left*: Gender bias vs. number of attention heads. *Right*: Selected attention heads with 20% sparsity.

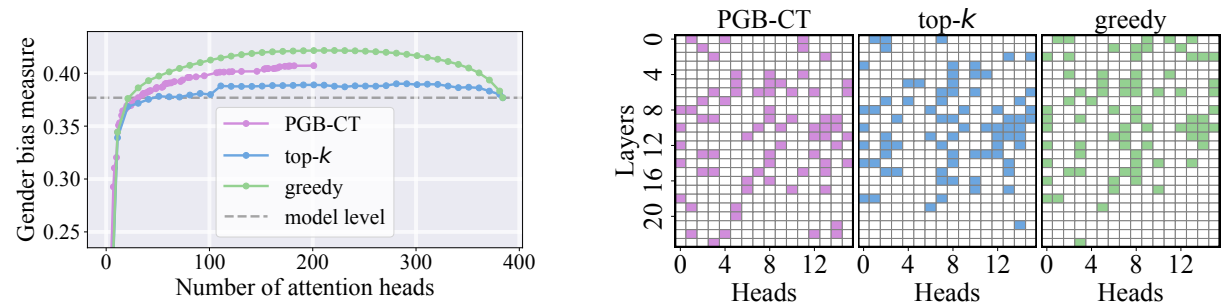


Figure 10: Results of selecting attention heads from GPT2-medium on the WinoBias dataset. *Left*: Gender bias vs. number of attention heads. *Right*: Selected attention heads with 20% sparsity.

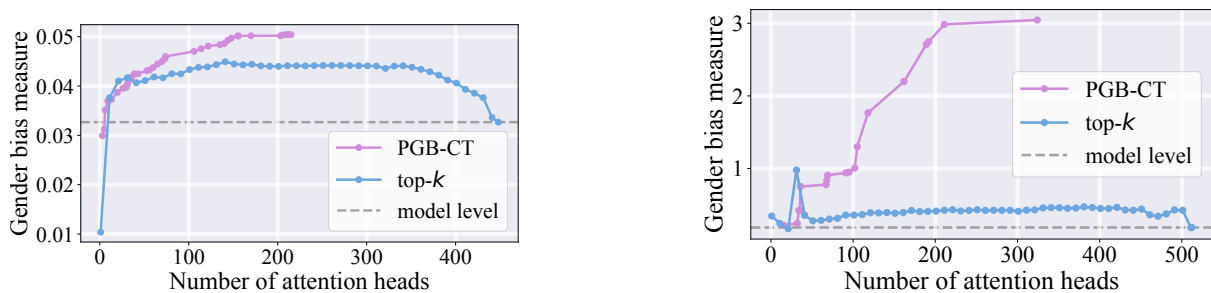


Figure 11: Results of selecting attention heads on the WinoGender dataset *Left*: with Qwen3-1.7B. *Right*: with Llama-3.2-1B.

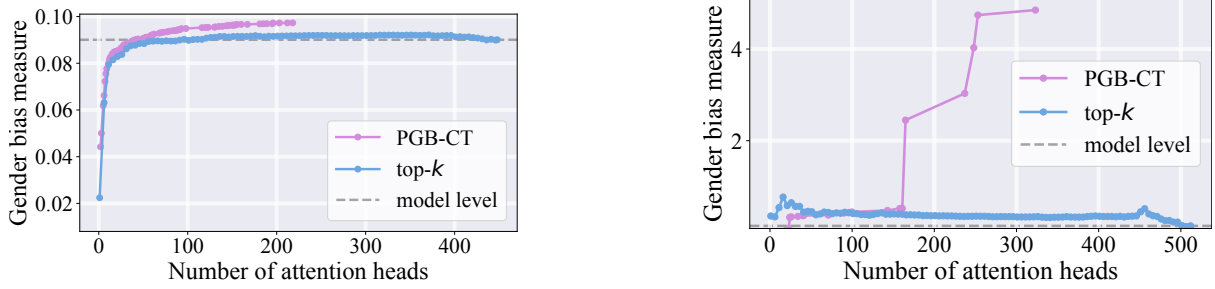


Figure 12: Results of selecting attention heads on the Winobias dataset: Gender bias measure vs. number of neurons. *Left*: with Qwen3-1.7B. *Right*: with Llama-3.2-1B.

Model / Algorithm		Sparsity				Time hours
		$\leq 2.5\%$	$\leq 5\%$	$\leq 10\%$	$\leq 15\%$	
DistilGPT2	random	0.36	0.72	1.59	3.02	0.21
	top- $k$	<b>14.86</b>	47.23	103.79	149.28	0.97
	PGB-CT	11.76	<b>58.33</b>	<b>313.17</b>	<b>2308.08</b>	<b>0.02</b>
GPT2-small	random	0.34	0.56	1.54	2.87	0.32
	top- $k$	30.23	84.71	127.32	140.80	2.97
	PGB-CT	<b>47.54</b>	<b>146.40</b>	<b>1320.25</b>	<b>8654.82</b>	<b>0.03</b>
GPT2-medium	random	0.25	0.48	1.35	2.77	0.71
	top- $k$	61.63	103.18	125.45	125.03	17.37
	PGB-CT	<b>1311.80</b>	<b>6.8e7</b>	<b>6.8e7</b>	<b>6.8e7</b>	<b>0.07</b>
GPT2-large	random	0.24	0.39	0.85	1.57	1.35
	PGB-CT	<b>437.33</b>	<b>1.7e6</b>	<b>6.9e9</b>	<b>6.9e9</b>	<b>0.14</b>
GPT2-xl	random	0.28	0.63	1.46	3.12	2.52
	PGB-CT	<b>18852</b>	<b>1.9e5</b>	<b>6.1e10</b>	<b>6.6e11</b>	<b>0.25</b>
Qwen3-1.7B	random	2.15	2.88	3.54	3.55	2.04
	PGB-CT	<b>6.1e10</b>	<b>2.1e11</b>	<b>2.1e11</b>	<b>2.1e11</b>	<b>0.20</b>
Llama3.2-1B	random	1.13	1.51	1.94	2.03	0.80
	PGB-CT	<b>3470.68</b>	<b>1.1e4</b>	<b>2.2e4</b>	<b>2.5e4</b>	<b>0.09</b>

Table 4: Averaged metric  $\ell(\mathcal{D}, \mathbf{m})$  and execution time on the Professions dataset for selecting MLP neurons across different sparsity levels, algorithms, and LLMs. The best results are highlighted in **bold**.

$N = 12$  is the number of MLP layers in GPT2-small. Then the value at  $(i, j)$ -th position in the figure is calculated by subtracting the effects of individual components from their combined effect.

$$\ell(\mathcal{D}, \mathbf{m}_{\{i,j\}}) - \ell(\mathcal{D}, \mathbf{m}_{\{i\}}) - \ell(\mathcal{D}, \mathbf{m}_{\{j\}}), \quad (23)$$

where the vector  $\mathbf{m}_S \in \{0, 1\}^N$  defined over a set  $\mathcal{S}$  takes value 1 at the  $k$ -th component when  $k \in \mathcal{S}$ , and 0 otherwise.

## D Additional Experimental Results

Additional experiments are organized as follows. Appendix D.4 presents an ablation study on the penalty terms  $\lambda_1$  and  $\lambda_2$ . Appendices D.1 and D.2 report supplementary results for attention heads and MLP neurons, respectively. Appendix D.3 includes more discussion. Furthermore, we include

the ablation study in the Appendix D.4 and comparison to DCM in the Appendix D.5. Section D.6 discusses the binarization of  $\mathbf{m}$  and the associated sparsity analysis. Finally, Section D.7 discusses the comparison of our algorithm to baselines when  $S = 1$ .

### D.1 Attention heads

Figures 10-12 show the additional figures for the experiments on the attention heads. Specifically, Figure 8 shows the results with GPT2-small on the WinoGender dataset; Figure 9 shows the results with GPT2-medium on the WinoGender dataset; Figure 10 shows the results with GPT2-medium on the WinoBias dataset. Figure 11 extends to Qwen3-1.7B and Llama-3.2-1B models, where we omit the greedy algorithm because it is too time-consuming. Figure 12 shows the results with

Model / Algorithm		Sparsity				Time
		$\leq 2.5\%$	$\leq 5\%$	$\leq 10\%$	$\leq 15\%$	hours
DistilGPT2	random	2.35	18.07	111.89	37.67	1.98
	top- $k$	5.72	24.77	24.99	22.45	10.08
	PGB-CT	<b>15.63</b>	<b>62.22</b>	<b>286.34</b>	<b>1141.76</b>	<b>0.07</b>
GPT2-small	random	3.88	13.75	239.06	122.11	3.99
	top- $k$	27.49	66.02	107.60	197.02	32.35
	PGB-CT	<b>755.95</b>	<b>1790.33</b>	<b>8376.24</b>	<b>21155.90</b>	<b>0.13</b>

Table 5: Averaged metric  $\ell(\mathcal{D}, \mathbf{m})$  and execution time on the Factual (CounterFact) dataset for selecting MLP neurons across different sparsity levels, algorithms, and LLMs. The best results are highlighted in **bold**.

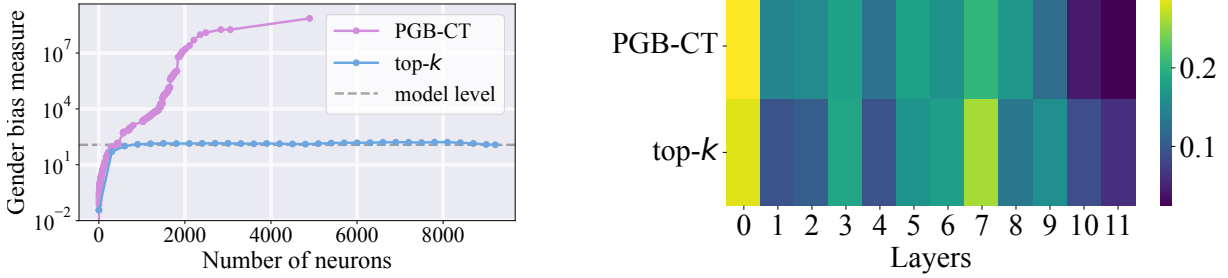


Figure 13: Results of selecting MLP neurons on the Professions dataset with GPT2-small. *Left*: Gender bias measure vs. number of neurons. *Right*: Fraction of Selected neurons in each layer with 15% sparsity.

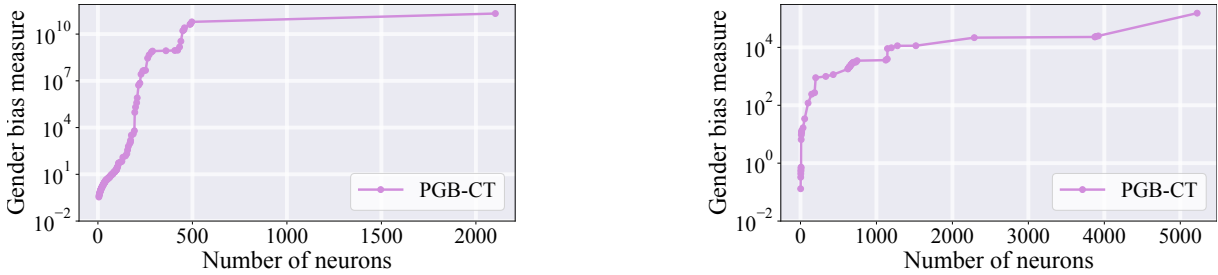


Figure 14: Results of selecting MLP neurons on the Professions dataset: Gender bias measure vs. number of neurons. *Left*: with Qwen3-1.7B. *Right*: with Llama-3.2-1B.

Qwen and Llama on the WinoBias dataset. We find that, in all cases, PGB-CT outperforms the top- $k$ , and is close to greedy when applicable.

## D.2 Neurons

The results for the Professions dataset are summarized in Table 4 and the results for the CounterFact dataset are summarized in Table 5. In addition, Figure 13 shows the results with GPT2-small in the Professions dataset; Figure 15 shows the results with distilGPT2 in the CounterFact dataset. Figure 14 shows the results with Qwen and Llama models. It is evident that PGB-CT shows an exponential increase in metric with the number of components, whereas the other algorithms cannot.

## D.3 Other findings

We list some additional findings below.

- *Model structure and size.* Consistent with the

single-component causal-tracing results in (Vig et al., 2020b), we find that larger models generally display stronger gender bias. More intriguingly, bias patterns vary across architectures: e.g., the bias on Llama-3.2-1B is larger than Qwen3-1.7B. As illustrated in Figure 11 (right), the top- $k$  baseline fails on this architecture, highlighting the need for multi-component causal tracing.

- *Highly Non-Linear Effect of MLP Neurons.* We observe a surprising non-linearity in the effect of MLP neurons (unlike attention heads). Specifically, the target metric increases at varying exponential rates as the sparsity of the selected set changes.
- *“Half-half” rule:* Across all evaluated models, we observe that about 50% of the attention heads or neurons collectively boost the target metric

and the other 50% negatively impact it.

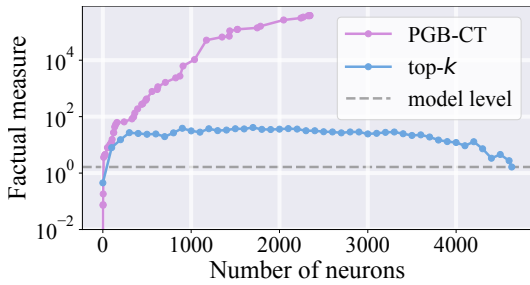


Figure 15: Results of Factual locating measure vs. number of MLP neurons on the CounterFact dataset with GPT2-small.

#### D.4 Ablation study

Figure 7 presents the ablation results in the WinoGender dataset using GPT2-small, with all other hyperparameters and grid search settings remaining the same. The figure shows that removing the first  $\ell_1$  penalty term prevents the algorithm from achieving a sparse solution, while removing the second penalty term degrades its performance.

Besides, we also include the DCM algorithm (Davies et al., 2023), which can find some intervention combinations that are close to those found by our algorithm PGB-CT but converges only to certain sparsity levels, making the algorithm harder to tune and less practical. We discuss this further in Appendix D.5.

#### D.5 Why DCM Fails

As shown in Appendix D.4, Desiderata-based Component Masking (DCM) can identify some subsets of components whose performance is close to that of PGB-CT at a certain sparsity level. However, in practice, DCM often fails to find subsets that satisfy the desired sparsity or metric level, resulting in suboptimal solutions. We attribute this behavior to its penalty function, which does not effectively binarize the mask weights. Figure 16 compares the sparsity ratio  $S/N$  and the binary violation term  $\mathbf{m}(\mathbf{1} - \mathbf{m})$  for PGB-CT (with  $\lambda_1 = 0.0001$ ,  $\lambda_2 = 0.001$ ) and DCM (with  $\lambda = 10^{-5}$ ). The left panel of Figure 16 shows that both methods achieve similar sparsity after truncation. However, the right panel shows that the binary violation for DCM does not converge to zero, and when it attempts to approach zero, the sparsity increases further.

This is because, although DCM rounds the final mask weights to binary (similar to (22)), the

amount of “binary violation”  $\mathbf{m}(\mathbf{1} - \mathbf{m})$  before truncation is still crucial. A larger average value of  $\mathbf{m}(\mathbf{1} - \mathbf{m})$  indicates that many mask entries are closer to 0.5 at the end of training. When such a mask is truncated to a binary one, many components undergo large changes, which can cause a significant drop in the metric compared to the soft mask. Therefore, the higher binary violation observed for DCM in the right panel suggests a larger potential loss in the metric after the final truncation step.

#### D.6 Distribution of weights $\mathbf{m}$ in training

The PGB-CT algorithm encourages the values in  $\mathbf{m}$  to be exactly 0 or 1, resulting in binary selections. Experimentally, we found that binarization is achieved after the last epoch by choosing an appropriate  $\lambda_2$ , and  $\lambda_1$  controls sparsity more strongly. Figure 17 shows results from trials on different types of GPT-2 models with  $\lambda_1 = \lambda_2 = 1e-3$ . We observe that all models behave similarly: larger models exhibit slightly larger binary violations because they contain more parameters, yet they ultimately converge to higher sparsity since the penalty parameter does not scale with model size. Furthermore, Figure 18 shows how sparsity changes when varying  $\lambda_1$  on the WinoGender dataset with GPT2-small. We can vary from low sparsity to high sparsity with tuning  $\lambda_1$ .

#### D.7 Performance on single-component $S = 1$

When  $S = 1$ , our method reduces to traditional single-component causal tracing (likewise for top- $k$  and greedy baselines). Empirically, its performance is nearly identical to single-component methods. In the main paper, for attention-head selection on GPT-2-small with WinoBias (leftmost point in Figure 3) and MLP-neuron selection on GPT-2-medium with Profession (leftmost point in Figure 4), our algorithm selects the same optimal component as both top- $k$  and greedy. For larger models and harder settings (e.g., CounterFact), optimization can struggle to converge exactly to  $S = 1$ , leading to slight suboptimality. We note that the DCM algorithm cannot meet these sparsity requirements. See Table 6 for detailed results.

#### D.8 Additional results on VBD dataset

Figure 19 further shows the experiment results on jointly analyzing attention heads and whole MLP blocks on LLaMA-7B.

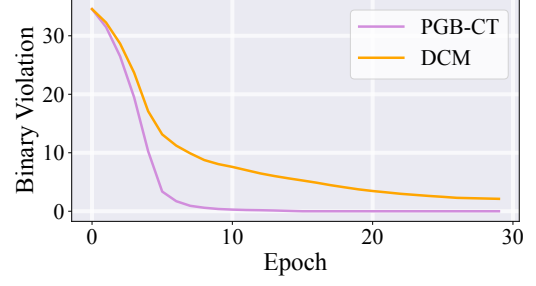
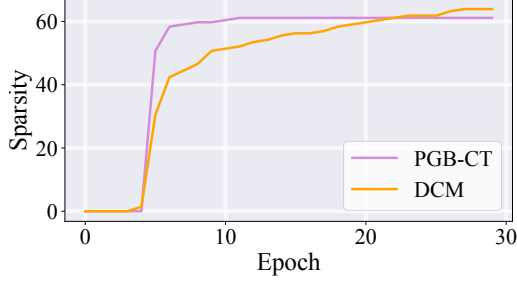


Figure 16: Comparison of properties of  $\mathbf{m}$  of attentions on WinoGender dataset.

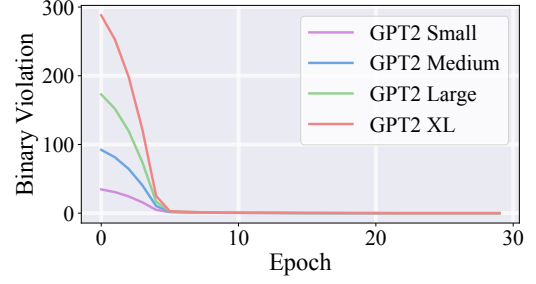
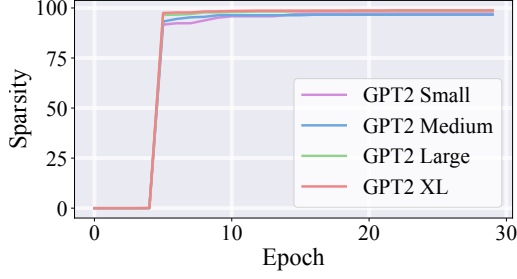


Figure 17: Properties of  $\mathbf{m}$  of attentions on WinoGender dataset. *Left:* Sparsity  $S/N$ . *Right:* Violation of binary  $\mathbf{m}(\mathbf{1} - \mathbf{m})$ .

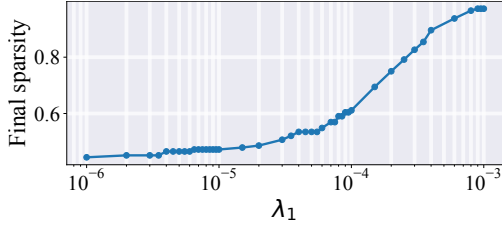


Figure 18: Results of sparsity vs.  $\lambda_1$  on the WinoGender dataset with GPT2-small.

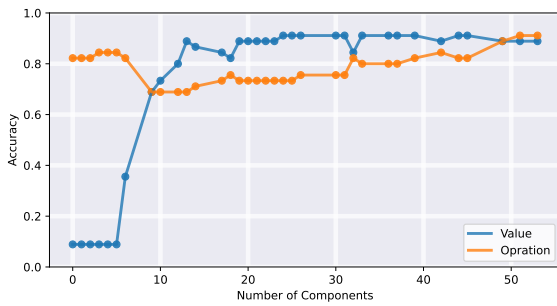


Figure 19: Results of accuracy on the VBD dataset on two competing accuracies with LLaMA-7B.

When the sparsity level is  $S = 15$ , we find the following combination for LLaMA-7B: Attention Heads 8.1, 12.11, 13.0, 13.15, 13.19, 14.19, 15.9, 15.23, 17.25, 18.12, 19.1, 19.6, 21.12, 25.2, and MLP block 10.

## E Proof of theorem 1

*Proof:* We notice  $C_N(S)$  takes the form

$$C_N(S) \triangleq \sum_{c=1}^S \binom{N}{c}. \quad (24)$$

We prove each regime separately.

**1.  $S$  is a fixed constant.** When  $S$  does *not* depend on  $N$ , the sum  $C_N(S)$  in (24) has exactly  $S$  terms. Each term  $\binom{N}{c}$  is a polynomial in  $N$  of degree  $c$ . Indeed, for fixed  $c$ ,

$$\binom{N}{c} = \frac{N(N-1)\cdots(N-c+1)}{c!} \quad (25)$$

$$= \mathcal{O}(N^c). \quad (26)$$

Therefore,

$$C_N(S) = \sum_{c=1}^S \binom{N}{c} \quad (27)$$

$$= \binom{N}{1} + \binom{N}{2} + \cdots + \binom{N}{S} \quad (28)$$

$$= \mathcal{O}(N) + \mathcal{O}(N^2) + \cdots + \mathcal{O}(N^c) \quad (29)$$

$$= \mathcal{O}(N^S). \quad (30)$$

**2.  $S = \alpha N$  for some  $\alpha \in (0, 1)$ .** Consider the sum defined in

$$S_N(\alpha N) = \sum_{c=1}^{\lfloor \alpha N \rfloor} \binom{N}{c}. \quad (31)$$

Model / Algorithm		Dataset		
		WinoGender	WinoBias	Professions
DistilGPT2	Optimal	0.0059	0.013	0.049
	PGB-CT	0.0059	0.012	0.049
GPT2-small	Optimal	0.019	0.105	0.037
	PGB-CT	0.019	–	0.037
GPT2-medium	Optimal	0.189	0.377	0.023
	PGB-CT	–	–	0.023

Table 6: Performance comparison in the single-component setting ( $S = 1$ ): PGB-CT and the optimum.

The binomial coefficients  $\binom{N}{c}$  are unimodal in  $c$ , achieving their maximum around  $c = N/2$ . If  $\alpha \leq 1/2$ , the largest term in the sum up to  $c = \lfloor \alpha N \rfloor$  is near  $c = \lfloor \alpha N \rfloor$ . If  $\alpha > 1/2$ , by symmetry, the main contribution is instead near the central region but still of the same exponential scale.

(a) *Dominant term and Stirling's approximation.* Using Stirling's approximation

$$n! \sim \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \quad \text{as } n \rightarrow \infty, \quad (32)$$

for  $k = \lfloor \alpha N \rfloor$ , we have

$$\binom{N}{k} = \frac{N!}{k!(N-k)!} \quad (33)$$

$$\approx \frac{\sqrt{2\pi N} \left(\frac{N}{e}\right)^N}{\sqrt{2\pi k} \left(\frac{k}{e}\right)^k \sqrt{2\pi(N-k)} \left(\frac{N-k}{e}\right)^{N-k}}. \quad (34)$$

Canceling common factors, we obtain the approximation

$$\begin{aligned} \binom{N}{k} &\approx \frac{N^N}{k^k (N-k)^{N-k}} \\ &\times \frac{1}{\sqrt{2\pi k (N-k)/N}} \end{aligned} \quad (35)$$

$$\begin{aligned} &= \frac{1}{\sqrt{2\pi \alpha(1-\alpha) N}} \\ &\times \frac{1}{(\alpha^\alpha (1-\alpha)^{1-\alpha})^N}. \end{aligned} \quad (36)$$

Noting that  $\alpha^\alpha (1-\alpha)^{1-\alpha} = 2^{-H_2(\alpha)}$ , where  $H_2(\alpha)$  is the binary entropy function

$$H_2(\alpha) = -\alpha \log_2(\alpha) - (1-\alpha) \log_2(1-\alpha). \quad (37)$$

Hence, we get

$$\binom{N}{\alpha N} \approx \frac{2^N}{\sqrt{2\pi \alpha(1-\alpha) N}}. \quad (38)$$

For  $\alpha = 1/2$ , this is the classical

$$\binom{N}{N/2} \sim \frac{2^N}{\sqrt{\pi N/2}}. \quad (39)$$

(b) *Sum of binomial coefficients.* Because the binomial coefficients are unimodal and dominated by those around the central region, the single term  $\binom{N}{\alpha N}$  already has order  $\Theta(2^N/\sqrt{N})$ . Consequently, the entire sum remains on this same exponential scale.

This completes the proof of both claims in the theorem. ■