Multi-Granular Node Pruning for Circuit Discovery

Anonymous ACL submission

Abstract

Circuit discovery aims to identify minimal subnetworks that are responsible for specific behaviors in large language models (LLMs). Existing approaches primarily rely on iterative edge pruning, which is computationally expensive and limited to coarse-grained units such as attention heads or MLP blocks, overlooking finer structures like individual neurons. We propose a node-level pruning framework for circuit discovery that addresses both scalability and granularity limitations. Our method introduces learnable masks across multiple levels of granularity, from entire blocks to individual neurons, within a unified optimization objective. Granularity-specific sparsity penalties guide the pruning process, allowing a comprehensive compression in a single fine-tuning run. Empirically, our approach identifies circuits that are smaller in nodes than those discovered by prior methods; moreover, we demonstrate that many neurons deemed important by coarse methods are actually irrelevant, while still maintaining task performance. Furthermore, our method has a significantly lower memory footprint, 5-10x, as it does not require keeping intermediate activations in the memory to work.

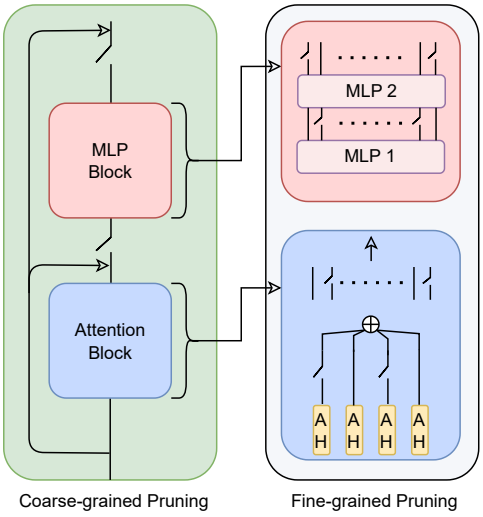

Figure 1: Architecture Diagram comparing Fine-grained Pruning with Coarse-grained Pruning

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities across a wide range of tasks, from question answering to code generation (Brown et al., 2020; Chowdhery et al., 2023; Achiam et al., 2023). However, their growing deployment in high-stakes applications has raised concerns about their interpretability and reliability (Rudin, 2019; Bommasani, 2021; Weidinger et al., 2021). Despite their empirical success, the internal mechanisms driving their behavior remain poorly understood, limiting our ability to trust, debug, and analyze these models (Lipton, 2018; Rudin, 2019). Circuit discovery (Olah et al., 2020; Cammarata et al., 2020) has emerged as a promising direction to address this gap, aiming to isolate minimal subnetworks or circuits, that are responsible for specific behaviors within a larger model.

Current circuit discovery methods predominantly rely on iterative pruning, where connections between components are gradually removed to isolate a minimal circuit responsible for a specific behavior (Wang et al., 2022; Conmy et al., 2023). While these methods have advanced our understanding of model behavior, they suffer from two key limitations. First, they are computationally expensive, requiring numerous forward passes through the model and require explicit storage of intermediate activations in working memory during the pruning process. Second, they operate at a coarse level of granularity, treating entire attention heads or MLP blocks as atomic units, allowing pruning only across components, not within them.

This assumption, that coarse components are either fully included or excluded from a circuit, overlooks the possibility that fine-grained structures, such as specific neurons within these components, can be responsible for the target behavior, thereby limiting interpretability. Recent work has attempted to formalize circuit discovery as an edge-pruning problem (Bhaskar et al., 2024). However, this formulation inherits the scalability challenges of edge-level pruning: the number of edges in a neural network is exponentially larger than the number of nodes, making the approach *resource intensive* for large models.

Traditional pruning methods (Han et al., 2015; LeCun et al., 1989; Haider and Taj, 2021), by contrast, focus on model compression for efficient deployment. They enforce sparsity aligned with hardware constraints, prioritizing size reduction without sacrificing performance. However, circuit discovery fundamentally differs in its goals, as it seeks to uncover the mechanisms underlying specific behaviors, rather than to produce compressed and efficient deployable models. This distinction opens new avenues for more flexible pruning strategies that prioritize interpretability and precision over deployment efficiency.

In this work, we introduce a node-level pruning framework for circuit discovery that addresses the scalability and granularity limitations of existing methods. Unlike prior approaches that prune edges between entire coarse components, our method operates simultaneously across multiple levels of granularity, enabling the discovery of circuits that are both fine-grained and structurally minimal. Specifically, we introduce learnable masks for transformer blocks, attention heads, MLPs, and individual neurons within a unified optimization objective. This enables minimal circuit discovery in a single fine-tuning run, avoiding the expense of iterative pruning.

Alongside our framework, we report several key findings. The structure of the discovered circuits depends on the task under consideration. For instance, all MLP blocks are essential for the IOI task, whereas for the GT task, most MLPs can be removed without significant performance degradation. Furthermore, our analysis indicates that different component granularities exhibit varying degrees of susceptibility to compression. For instance, the outputs of MLP1 (Keys) can be sparsified to a much greater extent than those of MLP2 (Values)(Geva et al., 2020).

Our contributions are as follows:

- We introduce the first framework that moves beyond edge-level pruning, enabling fine-grained circuit discovery, reaching down to individual neurons.
- Our framework identifies circuits efficiently, with minimal memory overhead and reduced computational cost.
- Our approach discovers circuits that are smaller across multiple granularities than those found by existing methods while maintaining task performance, and reveals that many neurons within components deemed important by coarse methods are in fact irrelevant.

2 Related Work

2.1 Circuit Discovery Methods

Manual Circuit Discovery Early mechanistic interpretability work manually traced through model computations to identify circuits. Wang et al. (2022) discovered the indirect object identification (IOI) circuit in GPT-2, revealing how models track syntactic dependencies. Olsson et al. (2022) identified induction heads as a fundamental circuit pattern for in-context learning. While groundbreaking, these manual approaches require extensive human effort and domain expertise, limiting their applicability to larger models and diverse tasks.

Automated Circuit Discovery To scale circuit analysis, recent work has developed automated methods based on iterative edge ablation. Automated Circuit Discovery (ACDC) by Conmy et al. (2023) formulates circuit finding as iteratively removing edges between components while maintaining task performance. The method starts with a full computational graph and greedily removes the least important edges, requiring $O(n^2)$ forward passes for n edges. Chan et al. (2022) extend this with Edge Attribution Patching (EAP), using gradient-based importance scores to guide edge removal. Goldowsky-Dill et al. (2023) propose causal scrubbing, which finds circuits by iteratively relaxing equivalence constraints between activations. While these methods successfully automate circuit discovery, they suffer from computational intractability for large models and are limited to coarse-grained components like attention heads and MLP blocks.

Learnable Masking Approaches. Recent work has explored learnable masks for circuit discovery.

Bhaskar et al. (2024) introduce differentiable masking for edges, using CoFi Pruning (Xia et al., 2022) to learn which connections matter in their approach, Edge Pruning (EP). However, these approaches still operate on edges between corse components and do not discover neuron level or block level circuits. Our work differs fundamentally by applying learnable masks to nodes at multiple granularities, from entire blocks down to individual neurons.

3 Preliminaries

We formalize circuit discovery within transformer-based language models and establish the notation used throughout this work. We then analyze the computational limitations of current edge-pruning approaches that motivate our node pruning method.

3.1 Circuit Discovery Formulation

Given a transformer model **Model** M with parameters θ and a specific task T, circuit discovery seeks to find a minimal subgraph $G^* = (N^*, E^*)$ of the model's computational graph $G = (N, E)$ such that the subgraph preserves task performance. The Nodes N represent components at various granularities, from entire layers and attention heads down to individual neurons. The edge set E represents the computational dependencies between components, where each edge $(u, v) \in E$ indicates that component v receives input from component u.

Task-Specific Circuits. For a task T defined over a dataset $D = \{(x_i, y_i)\}_{i=1}^N$, the performance of a circuit is evaluated as:

$$\mathcal{L}_T(G') = \frac{1}{N} \sum_{i=1}^{N} \ell(M_{G'}(x_i), y_i), \qquad (1)$$

where $M_{G'}$ denotes the model restricted to the subgraph G', and ℓ is a task-specific loss function.

The circuit discovery objective is then formulated as:

$$G^* = \arg\min_{G' \subseteq G} |G'| \quad \text{s.t.} \quad \mathcal{L}_T(G') \leq \mathcal{L}_T(G) + \epsilon, \qquad (2)$$

where $|G'|$ measures the size of the subgraph (e.g., number of nodes or edges), and ϵ denotes an allowable performance tolerance.

3.2 Traditional Approaches Formulation

Brute-force approaches such as ACDC (Conmy et al., 2023) formulate circuit discovery as an iterative edge-pruning process. Starting from the full graph $G^{(0)} = G$, they iteratively remove edges according to:

$$G^{(t+1)} = G^{(t)} \setminus \{e^*\}, \qquad (3)$$

where $e^* = \arg\min_{e \in E^{(t)}} \Delta\mathcal{L}(e)$, and $\Delta\mathcal{L}(e)$ denotes the performance degradation resulting from the removal of edge e.

To reduce computational overhead, recent edge pruning methods operate at a *component-level granularity*, representing the model as a graph $G = (N, E)$, where the node set N consists of attention heads and MLP blocks, and the edge set $E \subseteq N \times N$ encodes all possible connections between them.

Because every node can potentially connect to every other downstream node, the number of edges grows quadratically with the number of nodes, i.e., $|E| = O(|N|^2)$. Therefore, exploring finer granularities would require searching over a vastly large space, for example, there are 2.3 million edges connecting the outputs of the first MLP to the second MLP in the GPT-2 model. This quadratic scaling renders such approaches computationally intractable as the model size scales.

3.3 Limitations of Component-Level Granularity

Edge pruning methods operate on the edge connections between predefined components, attention heads, and MLPs, making implicit assumptions about the circuit structure. These edges are treated as fundamental, indivisible units, assumed to be either fully preserved or entirely removed. This binary inclusion criterion restricts the method's flexibility and prevents it from discovering or exploiting alternative structural granularities that may exist within or across these components.

These assumptions conflict with recent findings in mechanistic interpretability, showing that individual neurons and subneurons can implement specific features, attention heads often combine multiple unrelated functions, and only sparse subsets of neurons participate in specific computations (Cunningham et al., 2023; Dar et al., 2022; Wang et al., 2022; Haider et al., 2025).

The mismatch between architectural boundaries and functional boundaries limits the precision of discovered circuits, potentially overestimating circuit size by orders of magnitude.

Multi-Granular Node Pruning. While traditional methods constrain their analysis to the level

of attention heads and MLP blocks, the hierarchical structure of transformers naturally supports circuit discovery across multiple granularities. By extending the computational graph to include all granularities simultaneously, from entire blocks down to individual neurons, we can discover more precise circuits. This multi-granular view reveals that circuits may be sparse at different scales: an entire block might be redundant, only a few specific neurons within "important" components might be critical. By considering multiple levels of granularity, rather than constraining the analysis to a fixed level, we enable the discovery of circuit structures that more accurately capture the underlying computational structure. In this work, we implement a hierarchical multigranularity circuit discovery framework spanning five distinct levels of granularity:

$$\mathcal{G} = \{V_{\text{block}}, V_{\text{head}}, V_{\text{mlp}}, V_{\text{attn_neuron}}, V_{\text{mlp_neuron}}\} \tag{4}$$

This formulation allows us to discover more sparse circuits across different granularities.

4 Approach

We present a unified, node-level pruning framework that discovers minimal circuits across multiple granularities in a single optimization step. The framework employs learnable masks applied at different levels of granularity and uses a two-stream forward pass that interpolates between clean and corrupted activations to estimate the contribution of each component to task performance. In contrast to traditional pruning methods that simply zero out components, our approach identifies circuit elements by evaluating their importance through targeted corruption.

- **Corrupted Stream:** Passes a minimally perturbed version of the same input, corrupted using task task-dependent corruption function C, through the model. The corruption preserves input length but alters specific tokens to elicit different model predictions with distinct correct answers. Crucially, both inputs correspond to the same task type, enabling the identification of shared circuitry by observing how the same components are differentially activated (Vig et al., 2020; Meng et al., 2022; Wang et al., 2022). For instance, *"The war started in 1901 and ended in 19__."* is a corrupted perturbation of *"The war started in 1950 and ended in 19__."*
- **Clean Stream:** This stream represents an interpolation between the clean and corrupted activations as defined in Equation 5. At each model component, the clean stream is propagated through the component and linearly combined with the corresponding corrupted stream activations, weighted by the learned mask m. The clean stream passes through the model while being progressively mixed with the corrupted stream according to the mask values.

The two-stream architecture can be implemented efficiently by sharing weights and batching operations. This formulation identifies circuits by finding components whose corruption significantly impacts task performance, providing a principled way to discover minimal functional subnetworks.

The key hypothesis is that important components will show a large shift in performance between clean and corrupted activations, whereas unimportant components will show minimal differences. The pseudo-codes for the framework is provided in Algorithm 1.

Forward Pass Formulation: For each node $n_i \in N$, we define a parameterized mask $m_i \in [0, 1]$ that optimized to guide the model toward producing the correct output by selectively interpolation between the clean and corrupted activations. The output of each node is computed as:

$$\begin{aligned} h_i^{node} &= I(h_i^{\text{clean}}, h_i^{\text{corrupted}}, m_i) \\ I(.) &= m \cdot h^{\text{clean}} + (1 - m) \cdot h^{\text{corrupted}} \end{aligned} \tag{5}$$

where h_i^{clean} and $h_i^{\text{corrupted}}$ denote the clean and corrupted hidden activations of node n_i, respectively. When $m_i = 1$, the node passes the clean activation; when $m_i = 0$, it passes the corrupted activation. Intermediate values $0 < m_i < 1$ represent soft interpolation during training. The set of all masks M are regularized to encourage sparsity, ensuring that only the nodes most relevant to the task remain active.

Mask Parameterization: To enable gradient-based optimization while inducing approximately binary behavior, we parameterize the masks using the *Hard Concrete* distribution (Louizos et al., 2017). The sampling procedure is defined as:

$$
\begin{aligned}
&m_i = H(\alpha_i, \beta) \\
&H(\alpha_i, \beta) = \min(1, \max(0, s_i(\zeta - \gamma) + \gamma)) \\
&s_i = \sigma\left(\frac{1}{\beta}(\log u - \log(1-u) + \log \alpha_i)\right) \\
&u_i \sim \text{Uniform}(0, 1).
\end{aligned} \tag{6}
$$

Here, α_i is a trainable parameter for the ith gate, β controls smoothness, and $\zeta > 1, \gamma < 0$ define the support range. Each layer contains multiple such gates over components (e.g., blocks, heads, or neurons).

4.1 Training Objective

Our objective encourages finding minimal circuits that maintain task performance:

$$
\begin{aligned}
&\mathcal{L} = \mathcal{L}_{\text{task}}(h_L^{clean}) + \lambda \sum_{i=1}^{N} \mathcal{L}_0(m_i) \\
&\mathcal{L}_{\text{task}} = \text{KL}\big(\text{softmax}(h_L^{\text{base}}) \,\big\|\, \text{softmax}(h_L^{\text{clean}})\big) \\
&\mathcal{L}_0(m_i) = \sigma\left(\log \alpha_i - \beta \log \frac{-\gamma}{\zeta}\right)
\end{aligned} \tag{7}
$$

where h_L denotes the final layer representations.

4.2 Circuit Extraction

After training, we binarize all masks applying the following constraint:

$$
m_i^{\text{final}} = \mathbf{1}\left(\log \alpha_i > \beta \log\left(\frac{-\gamma}{\zeta}\right)\right), \tag{8}
$$

Furthermore, we enforce hierarchical consistency among masks: if a parent mask is deactivated, all its associated child masks are also set to zero. For instance, when an MLP block is pruned (mask set to zero), the neuron masks within that block are likewise disabled.

5 Experimental Setup

In this section, we outline the task formulation used for circuit discovery and the evaluation metrics employed to assess circuit quality.

5.1 Tasks

We evaluate our method on three circuit discovery tasks that test different computational capabilities within language models.

Greater-Than (GT). This task probes the numerical reasoning capabilities of language models, specifically their ability to perform temporal comparisons between years. For example, given a prompt *"The war lasted from the year 1743 to the year 17__"*, the model should assign higher probability to completions to numerics 44-99 than to 00-42. We utilize the dataset from (Hanna et al., 2023), which consists of 5 templates, 120 noun choices, and years spanning 1100-2199. The dataset consists of 12,540 samples that are divided into train, validation and test splits. This task requires models to extract the starting year, understand that ending years must be chronologically greater than starting years, and map this constraint to select valid two-digit completions.

Indirect Object Identification (IOI). The IOI task tests syntactic tracking and entity binding. For example, given a sentence *"Friends Juana and Kristi found a mango at the bar. Kristi gave it to"*, the model is expected to predict "Juana" as the recipient. We evaluate IOI dataset, which incorporates 30 diverse syntactic templates comprising 200 examples each for training and validation, and 12256 test instances. Example templates include constructions such as "Then, B and A had a long argument. Afterwards, B said to → A". The task requires the model to track entity positions, suppress repeated names, and correctly resolve the target entity for output. Prior work (Conmy et al., 2023) has shown that the IOI circuit spans multiple layers, involving specialized components responsible for name identification, positional encoding, and duplicate name inhibition.

Gendered Pronouns (GP). This task probes learned associations between names and pronouns. For example, given the prompt *"So Evan is a really great friend, isn't "*, the model should assign a higher probability to the gender-consistent pronoun "he" rather than the incorrect alternative "she.". The dataset used is constructed by incorporating the top 1,000 most popular male and female baby names from 2000, creating 150 train/validation examples each and 378 test examples. Unlike IOI, which evaluates syntactic reasoning, this task probes whether language models encode societal gender associations linked to names. Circuit analysis (Conmy et al., 2023) indicates that early layers are responsible for encoding gender-related features, while middle layers bind these features to the corresponding gendered pronouns.

Algorithm 1 Multi-Granularity Circuit Discovery

Training Phase

Require: Model f_θ with layers $l \in L$, task dataset $\mathcal{D}$, corruption function C

Initialize Hard-Concrete mask parameters α_i for all granularities $\mathcal{G}$ across the Model
for epoch $\leftarrow 1$ **to** N_{epochs} **do**
  for batch (x, y) **in** $\mathcal{D}$ **do**
    $x_{\text{corrupt}} \leftarrow C(x)$
    $M \sim \mathrm{H}(\log \alpha_i, \beta)$ ▷ Sample masks (Equation 6)
    $h_L^{\text{base}} \leftarrow f_\theta(x)$
    for $l \leftarrow 1$ **to** L **do** ▷ For each layer in f_θ and across all nodes using $m_l = \{m_{l,i}\}$
      $h_l^{\text{clean}} \leftarrow f_l(\mathrm{I}(h_{l-1}^{\text{clean}}, h_{l-1}^{\text{corrupt}}, m_l))$ ▷ Interpolation function I (Equation 5)
      $h_l^{\text{corrupt}} \leftarrow f_l(h_{l-1}^{\text{corrupt}})$
    end for
    $\mathcal{L} \leftarrow \mathcal{L}_{task}(h_L^{\text{clean}}) + \mathcal{L}_{\mathrm{KL}}\big(\text{softmax}(h_L^{\text{clean}}), \text{softmax}(h_L^{\text{base}})\big) + \lambda \sum_{i=1}^{N} \mathcal{L}_0(m_i)$ ▷ Update mask parameters via gradient descent
  end for
end for

5.2 Metrics

To assess the quality of the discovered circuits, we empirically evaluate them using two complementary metrics: one measuring task-specific performance, and another capturing distribution-level faithfulness to the original model.

Task-Specific Performance Metrics. We evaluate whether circuits preserve task-relevant computations using the following targeted metrics:

Greater-Than (GT): We measure the probability difference metric:

$$\text{GT-Score} = \mathbb{E}x \in \mathcal{D}_{\text{test}}[P(y > y_{\text{start}} + 10|x) - P(y < y_{\text{start}} - 10|x)] \tag{9}$$

where y_{start} is the starting year extracted from the prompt, and $P(y > y_{\text{start}}|x)$ is the total probability assigned to valid completions (years greater than the start year). A positive score indicates the circuit correctly assigns a higher probability to valid years.

Indirect Object Identification (IOI): We use the logit difference between correct and incorrect name predictions:

$$\text{IOI-Score} = \mathbb{E}_{x \in \mathcal{D}_{\text{test}}}\left[\text{logit}_{\text{IO}}(x) - \text{logit}_{\text{S}}(x)\right] \tag{10}$$

where logit_{IO} is the logit for the correct indirect object and logit_{S} is the logit for the incorrect subject (repeated name). Positive values indicate correct disambiguation. This metric directly tests whether the circuit implements the core IOI computation of selecting the non-repeated name.

Gendered Pronouns (GP): We measure the logit difference between gender-consistent and gender-inconsistent pronouns:

$$\text{GP-Score} = \mathbb{E}_{x \in \mathcal{D}_{\text{test}}}[\text{logit}_{\text{consistent}}(x) - \text{logit}_{\text{inconsistent}}(x)] \tag{11}$$

where $\text{logit}_{\text{correct}}$ is the logit for the gender-appropriate pronoun (e.g., "he" for "Evan") and $\text{logit}_{\text{incorrect}}$ is the logit for the opposite gender pronoun. Higher scores indicate stronger gender associations.

For **distribution-level faithfulness**, we compute the Kullback-Leibler (KL) divergence between the full model's output logit distribution (P_m) and the pruned circuit's logit distribution (P_c):

$$\mathcal{D}_{\text{KL}}(P_{\text{m}}||P_{\text{c}}) = \sum_{v \in \mathcal{V}} P_{\text{m}}(v) \log \frac{P_{\text{m}}(v)}{P_{\text{c}}(v)} \tag{12}$$

where $\mathcal{V}$ is the vocabulary and P represents the probability distribution over next tokens. This metric quantifies how closely the circuit reproduces the full model's predictive distribution, unlike the earlier probability gap and logit difference metrics. Consequently, a lower value signifies a stronger fidelity to the full model's behaviour.

For all tasks, we measure the discovered circuit quality using KL divergence between the output logits of the orignal model and pruned circuit model, following the evaluation protocol of prior work.

Circuit Size Metrics. We report circuit size across multiple granularities using four metrics: *(1) Parameter count* - total parameters within active components; *(2) Compression ratio* - the ratio of original to circuit parameters; *(3) Sparsity by level* - percentage pruned at each granularity (layers, heads, neurons); and *(4) Edge count comparison* - for edge-pruning methods, we convert our node-based circuits into equivalent edge counts for a fair comparison.

These metrics together provide a comprehensive view of circuit quality, measuring both faithful-

	IOI		GP		GT	
Granularity	**Active**	**Sparsity (%)**	**Active**	**Sparsity (%)**	**Active**	**Sparsity (%)**
Attention Blocks	4	66.7	5	58.3	5	58.3
MLP Blocks	12	0.0	3	75.0	5	58.3
Attention Heads	21	85.4	37	74.3	28	80.6
Attention Neurons	907	90.2	1,702	81.5	1,701	81.5
MLP Hidden Neurons	12,300	33.4	1,333	96.4	4,570	87.6
MLP Output Neurons	1,329	14.4	1,411	84.7	3,520	61.8
Edge Compression	**96.74**		**93.74**		**95.95**	

Table 1: Discovered Circuit Components Across Tasks (Active Counts, Sparsity, and Compression)

Task	**Base P/L Diff** ↑	**KL** ↓	**P/L Diff** ↑
IOI	3.1791	0.6080	3.2030
GP	2.6198	0.4909	2.6150
GT	0.3711	0.0059	0.3912

Table 2: Circuit faithfulness and task performance.

ness to the original model and preservation of task-specific capabilities while quantifying the achieved sparsity.

6 Results

Table 1 summarizes the pruning outcomes for the three tasks (IOI, GP, and GT) on GPT-2. Our node-pruning framework allows pruning at the granularity of individual neurons. The results indicate that even within active components such as MLPs and attention blocks, a substantial proportion of neurons can be removed while still preserving task performance, yielding a highly minimal circuit. Notably, we observe a consistent pattern: output neurons in the first layer of the MLP block exhibit significantly higher compression compared to both the second layer and the attention neurons. The structural characteristics of the identified circuits vary across tasks. In the IOI task, most MLP components remain intact, whereas in the GP and GT tasks, the circuit is reduced to only three and five unpruned MLP blocks, respectively. One possible interpretation is that the GP task, which involves choosing between only two pronouns ("he" or "she"), presents a comparatively restricted output space.

The details of pruned circuits across all tasks are provided in Appendix D. For the IOI task, only 21 out of 144 attention heads remain unpruned. In particular, only a single head is retained in the first layer, with all heads pruned through layer 9. In contrast, the majority of heads in layers 10–12 remain unpruned. This distribution aligns with the observations of (Conmy et al., 2023), who reported that name-mover heads are predominantly concentrated in the final layers. For the GT task 28 heads remain active. A closer examination reveals that all heads in the initial and final layers are fully pruned, while layers 7–9 retain between three and seven unpruned heads. This distribution suggests that the task relies more heavily on early-mid layers computation to support accurate predictions. In the GP task, all early- to mid-layer attention blocks are either fully or substantially pruned, with the exception of the fourth block. By contrast, the majority of heads in the final layers remain unpruned.

Overall node pruning procedure yields substantial reductions in parameter count. Table 2 reports both task performance and circuit faithfulness relative to the base model. Overall, pruning leads to minimal performance degradation: the IOI and GP tasks exhibit negligible differences from the baseline (≤ 0.03), indicating that the essential computation is largely preserved. In contrast, the GT task shows a small improvement, with performance increasing from 0.37 to 0.39. This suggests that pruning may reduce interference from redundant heads, slightly enhancing task-specific signal extraction. Faithfulness, quantified by KL divergence, remains below 0.61 across all tasks, confirming that the pruned circuits closely approximate the behavior of the original model while achieving greater interpretability.

7 Comparison with Baselines

Although node pruning is not directly comparable to edge pruning methods (EAP, EP[1]), since the **latter do not consider all the granularities like neurons, MLP layers, blocks as prunable parameters**, we nevertheless include a comparison of our multi-granularity compressed circuit across all three tasks, comparing attention heads only, and task metrics. We compare the pruned circuits pri-

Dataset	Method	Sparsity (%) ↑	KL↓	Logit Diff ↑	Attention Heads↓
IOI	EAP	96.74	2.447	-0.181	116
	EP	96.16	**0.360**	**3.210**	41
	Ours	96.74	0.600	3.203	**21**
GP	EAP	93.74	**0.148**	2.564	106
	EP	-	-	-	
	Ours	93.74	0.490	**2.615**	**37**
GT	EAP	95.95	0.086	0.374	113
	EP	96.69	0.039	0.389	74
	Ours	95.95	**0.006**	**0.391**	**28**

Table 3: Comparison of circuit sparsification node pruning, EAP and EP on GP,GT and IOT tasks . We report KL divergence (lower is better) and sparsity.

marily in terms of attention components, since edge pruning methods do not completely remove any MLPs or finer-grained structures. **Any pruning observed within these components arises solely from our proposed approach.** We provide the complete pruned graphs of edge pruning methods in the Appendix Section D. The results are presented in Table **??**.

Across the three evaluated tasks (IOI, GP, and GT), our method consistently achieves competitive edge sparsity levels while substantially reducing the number of retained attention heads compared to both EAP and EP. For GT, our method delivers the best overall results, achieving both the lowest KL divergence and the highest probability, while requiring far fewer attention heads than either EAP or EP. These findings indicate that our approach yields more compact representations without sacrificing task performance. On GP, it reduces the number of attention heads from 106 (EAP) to 37 while preserving a competitive logit difference (2.564 vs. 2.615).

On the IOI task, our method achieves a strong trade-off between performance and compactness. While edge pruning attains the lowest KL divergence and slightly higher logit difference, our approach closely matches its predictive performance but with a substantially smaller number of retained attention heads (21 vs. 41). Compared to EAP, which requires 116 heads, our method achieves comparable or better performance with nearly an order of magnitude fewer heads.

Compute Requirements. Our node pruning method is highly computationally efficient; it requires substantially less computation than the baseline methods. It adds and trains only 55465 additional parameters in GPT-2 small, which is 124.5M parameters. Since we have two instances of the model loaded in the memory, one base model and one prunable model for KL-loss training. Our method requires 6,270 MB of memory for a batch size of 32. In comparison, the EAP baselines require upwards of 72,794 MB of memory, and the EP baseline requires 33,354 MB of memory, since they have all the internal representations stored in memory.

[1]Due to issues with library code, we are unable to run Edge Pruning for GP task.

8 Conclusion

In this work, we introduced a multi-granularity node pruning approach for circuit discovery, capable of pruning at all granularities (MLPs ,Attention heads, Neurons, Blocks) within a single training step. Unlike edge pruning methods, which typically require significantly more computational resources, our method achieves comparable or superior interpretability while operating with much lower compute requirements. This efficiency, combined with its ability to prune across granularities simultaneously, makes it a flexible and scalable framework for studying sparse mechanisms in transformer models. Our experiments show strong performance across tasks, with substantial reductions in circuit size and the number of retained attention heads, highlighting the practicality of node-based sparsification as an alternative to edge pruning.

Looking forward, an exciting direction is to combine edge and node pruning in a hierarchical fashion, leveraging the fine-grained control of edge sparsification with the efficiency of node-level pruning. Such a hybrid approach may enable the discovery of even smaller and more minimal circuits while maintaining strong performance, further advancing the goal of interpretable and efficient mechanistic analysis.

9 Limitations

Our framework discovers *node-level* circuits but does not explicitly recover the *interaction structure* between nodes. In particular, the binary masks identify which blocks/heads/neurons are necessary, yet they do not reveal *which* active nodes exchange information with which others, nor the directionality or multiplicity of those interactions. Developing a hybrid node and edge procedure that infers interaction structure on top of node selections would alleviate these concerns.

10 Potential Risks

Targeted removal of nodes may inadvertently disable moderation, refusal, or safety-related pathways, enabling circumvention of guardrails and jailbreak-style behavior.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Adithya Bhaskar, Alexander Wettig, Dan Friedman, and Danqi Chen. 2024. Finding transformer circuits with edge pruning. *Advances in Neural Information Processing Systems*, 37:18506–18534.

Rishi Bommasani. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Nick Cammarata, Shan Carter, Gabriel Goh, Chris Olah, Michael Petrov, Ludwig Schubert, Chelsea Voss, Ben Egan, and Swee Kiat Lim. 2020. Thread: circuits. *Distill*, 5(3):e24.

Lawrence Chan, Adria Garriga-Alonso, Nicholas Goldowsky-Dill, Ryan Greenblatt, Jenny Nitishinskaya, Ansh Radhakrishnan, Buck Shlegeris, and Nate Thomas. 2022. Causal scrubbing: A method for rigorously testing interpretability hypotheses. In *AI Alignment Forum*, volume 2.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, and 1 others. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

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

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. 2023. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*.

Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. 2022. Analyzing transformers in embedding space. *arXiv preprint arXiv:2209.02535*.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2020. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*.

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

Muhammad Umair Haider, Hammad Rizwan, Hassan Sajjad, Peizhong Ju, and AB Siddique. 2025. Neurons speak in ranges: Breaking free from discrete neuronal attribution. *arXiv preprint arXiv:2502.06809*.

Muhammad Umair Haider and Murtaza Taj. 2021. Comprehensive online network pruning via learnable scaling factors. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3557–3561. IEEE.

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28.

Michael Hanna, Ollie Liu, and Alexandre Variengien. 2023. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. *Advances in Neural Information Processing Systems*, 36:76033–76060.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. *Advances in neural information processing systems*, 2.

Zachary C Lipton. 2018. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57.

Christos Louizos, Max Welling, and Diederik P Kingma. 2017. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*.

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

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. 2020. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001.

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. *arXiv preprint arXiv:2209.11895*.

Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215.

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

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

Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, and 1 others. 2021. Ethical and social risks of harm from language models. *arXiv preprint arXiv:2112.04359*.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. 2022. Structured pruning learns compact and accurate models. *arXiv preprint arXiv:2204.00408*.

A Training Details:

For the Gender Pronouns and Greater than task, 150 training examples were used for 200 epochs with a batch size of 32. For the Indirect Object Identification task, 200 training examples were used for 500 epochs. For all tasks, 64 sequence length was used.

B Compute Resources

All experiments were run on RTX 3v090 GPU, 64GB RAM. Baselines were run on NVIDIA H100 and L40 GPUs.

C Hyper Parameters Details:

The main hyperparameters are the sparsity loss coefficients $\lambda_{\mathcal{G}}$. To ensure consistent sparsity control across granularities, we normalize the sparsity loss within each granularity such that, for example, all attention neuron losses are scaled to the range $[0, 1]$. The coefficients used in our experiments are:

$$\begin{aligned} \lambda_{\text{attention_heads}} &= 3.0, \\ \lambda_{\text{mlp_hidden}} &= 5.0, \\ \lambda_{\text{mlp_output}} &= 1.5, \\ \lambda_{\text{attention_neurons}} &= 1.5, \\ \lambda_{\text{attention_blocks}} &= 0.2, \\ \lambda_{\text{mlp_blocks}} &= 0.1. \end{aligned} \tag{13}$$

These values were selected to balance pruning granularity and stability. Coarser structures, such as full MLP or attention blocks, receive smaller λ values to prevent overly aggressive pruning of large functional units. Finer structures, such as individual hidden neurons or attention heads, are assigned larger coefficients to encourage selective sparsity while preserving overall representational capacity. This configuration was found to be robust across tasks without requiring task-specific retuning.

D Circuit detailed Summaries

Across tasks, the node pruning summaries (Tables 4–6) reveal highly structured sparsity: large contiguous bands of fully pruned layers or blocks, with activity concentrating in a few late layers. For *Greater Than*, many early–mid layers disable entirely, while attention and MLP usage peaks in layers 9–10. For *Gender Pronouns*, the mid-stack is largely inactive, punctuated by bursts of attention activity (e.g., layers 5, 10–12) and sparse MLP use. For *IOI*, attention is mostly pruned except in the final layers, whereas MLP blocks remain active across nearly the whole depth with substantial hidden/output retention, indicating MLP-dominant computation. In contrast, the edge-based summaries (EP and EAP; Tables 7–8) keep all layers active and vary primarily in the number of retained heads, with EAP consistently preserving more heads than EP. Overall, node-level results exhibit modular, layer-selective circuits, while edge-based results favor distributed connectivity with minimal layer-level shutoff.

Table 4: Node Pruning(Ous) summary for Indirect Object Identification GPT-2 Model: attention blocks, MLP blocks, and retained components.

Layer	Attn Block	MLP Block	Attn Heads	Attn Neurons	MLP Hidden	MLP Output
1	Active	Active	11/12	357/768	2455/3072	751/768
2	Pruned	Active	0/12	0/768	1873/3072	691/768
3	Pruned	Active	0/12	0/768	1934/3072	660/768
4	Pruned	Active	0/12	0/768	2033/3072	648/768
5	Pruned	Active	0/12	0/768	1997/3072	656/768
6	Pruned	Active	0/12	0/768	2040/3072	656/768
7	Pruned	Active	0/12	0/768	1971/3072	653/768
8	Pruned	Active	0/12	0/768	2049/3072	648/768
9	Pruned	Active	0/12	0/768	2027/3072	626/768
10	Active	Active	1/12	56/768	1987/3072	605/768
11	Active	Active	7/12	385/768	2050/3072	637/768
12	Active	Active	2/12	109/768	2148/3072	656/768

Table 5: Node Pruning(Ous) summary for Gender Pronouns Task GPT-2 Model: attention blocks, MLP blocks, and retained components.

Layer	Attn Block	MLP Block	Attn Heads	Attn Neurons	MLP Hidden	MLP Output
1	Active	Active	4/12	177/768	1104/3072	669/768
2	Pruned	Pruned	0/12	0/768	0/3072	0/768
3	Pruned	Pruned	0/12	0/768	0/3072	0/768
4	Pruned	Pruned	0/12	0/768	0/3072	0/768
5	Active	Pruned	11/12	612/768	0/3072	0/768
6	Pruned	Pruned	0/12	0/768	0/3072	0/768
7	Pruned	Pruned	0/12	0/768	0/3072	0/768
8	Pruned	Active	0/12	0/768	149/3072	407/768
9	Pruned	Pruned	0/12	0/768	0/3072	0/768
10	Active	Pruned	7/12	290/768	0/3072	0/768
11	Active	Pruned	11/12	474/768	0/3072	0/768
12	Active	Active	4/12	149/768	80/3072	335/768

Table 6: Node Pruning(Ous) summary for Greater Than Task GPT-2 Model: attention blocks, MLP blocks, and retained components.

Layer	Attn Block	MLP Block	Attn Heads	Attn Neurons	MLP Hidden	MLP Output
1	Active	Active	3/12	167/768	1184/3072	743/768
2	Pruned	Active	0/12	0/768	1265/3072	708/768
3	Pruned	Pruned	0/12	0/768	0/3072	0/768
4	Pruned	Pruned	0/12	0/768	0/3072	0/768
5	Pruned	Pruned	0/12	0/768	0/3072	0/768
6	Pruned	Pruned	0/12	0/768	0/3072	0/768
7	Active	Pruned	4/12	251/768	0/3072	0/768
8	Active	Pruned	3/12	181/768	0/3072	0/768
9	Active	Active	7/12	427/768	664/3072	675/768
10	Active	Active	11/12	675/768	707/3072	704/768
11	Pruned	Active	0/12	0/768	750/3072	690/768
12	Pruned	Pruned	0/12	0/768	0/3072	0/768

Table 7: Indirect Object Identification Task: Pruning Summary: Edge Pruning (EP) vs. Edge Attribution Patching (EAP)

EP				EAP			
Layer	**Status**	**Attn Heads Active**	**MLP Block**	**Layer**	**Status**	**Attn Heads Active**	**MLP Block**
0	Active	8 / 12	Active	0	Active	12 / 12	Active
1	Active	1 / 12	Active	1	Active	11 / 12	Active
2	Active	3 / 12	Active	2	Active	10 / 12	Active
3	Active	4 / 12	Active	3	Active	10 / 12	Active
4	Active	2 / 12	Active	4	Active	8 / 12	Active
5	Active	1 / 12	Active	5	Active	11 / 12	Active
6	Active	2 / 12	Active	6	Active	12 / 12	Active
7	Active	1 / 12	Active	7	Active	9 / 12	Active
8	Active	4 / 12	Active	8	Active	6 / 12	Active
9	Active	6 / 12	Active	9	Active	10 / 12	Active
10	Active	5 / 12	Active	10	Active	9 / 12	Active
11	Active	4 / 12	Active	11	Active	8 / 12	Active

Table 8: Gender Pronouns Task: Pruning Summary for Edge Pruning (EP) vs. Edge Attribution Patching (EAP)

EP				EAP			
Layer	**Status**	**Attn Heads Active**	**MLP Block**	**Layer**	**Status**	**Attn Heads Active**	**MLP Block**
0	–	–	–	0	Active	12 / 12	Active
1	–	–	–	1	Active	10 / 12	Active
2	–	–	–	2	Active	10 / 12	Active
3	–	–	–	3	Active	9 / 12	Active
4	–	–	–	4	Active	9 / 12	Active
5	–	–	–	5	Active	9 / 12	Active
6	–	–	–	6	Active	9 / 12	Active
7	–	–	–	7	Active	8 / 12	Active
8	–	–	–	8	Active	10 / 12	Active
9	–	–	–	9	Active	6 / 12	Active
10	–	–	–	10	Active	8 / 12	Active
11	–	–	–	11	Active	6 / 12	Active

Table 9: Greater Than Task: Pruning Summary for Edge Pruning (EP) vs. Edge Attribution Patching (EAP)

EP				EAP			
Layer	**Status**	**Attn Heads Active**	**MLP Block**	**Layer**	**Status**	**Attn Heads Active**	**MLP Block**
0	Active	9 / 12	Active	0	Active	12 / 12	Active
1	Active	4 / 12	Active	1	Active	11 / 12	Active
2	Active	5 / 12	Active	2	Active	12 / 12	Active
3	Active	8 / 12	Active	3	Active	12 / 12	Active
4	Active	6 / 12	Active	4	Active	11 / 12	Active
5	Active	5 / 12	Active	5	Active	12 / 12	Active
6	Active	5 / 12	Active	6	Active	10 / 12	Active
7	Active	6 / 12	Active	7	Active	11 / 12	Active
8	Active	8 / 12	Active	8	Active	10 / 12	Active
9	Active	5 / 12	Active	9	Active	6 / 12	Active
10	Active	7 / 12	Active	10	Active	4 / 12	Active
11	Active	6 / 12	Active	11	Active	2 / 12	Active