Interpretable Transformers with Sparse Attention Post-Training

Florent Draye[1,*], Anson Lei[1,2,*], Hsiao-Ru Pan[1], Ingmar Posner[2], Bernhard Schölkopf[1,3]
[1]MPI-IS [2]University of Oxford [3]ETH Zürich [*]Equal contribution
fdraye@tuebingen.mpg.de

Abstract

We introduce a simple post-training method that makes transformer attention sparse without sacrificing performance. Applying a flexible sparsity regularisation under a constrained-loss objective, we show on models up to 7B parameters that it is possible to retain the original pretraining loss while reducing attention connectivity to $\approx 0.4\%$ of its edges. Unlike sparse-attention methods designed for computational efficiency, our approach leverages sparsity as a structural prior: it preserves capability while exposing a more organized and interpretable connectivity pattern. We find that this local sparsity cascades into global circuit simplification: task-specific circuits involve far fewer components (attention heads and MLPs) with up to 100× fewer edges connecting them. Additionally, using cross-layer transcoders, we show that sparse attention substantially simplifies attention attribution, enabling a unified view of feature-based and circuit-based perspectives. These results demonstrate that transformer attention can be made orders of magnitude sparser, suggesting that much of its computation is redundant and that sparsity may serve as a guiding principle for more structured and interpretable models.

1 Introduction

Scaling has driven major advances in artificial intelligence, with ever-larger models trained on internet-scale datasets achieving remarkable capabilities across domains. Large language models (LLMs) now underpin applications from text generation to question answering, yet their increasing complexity renders their internal mechanisms largely opaque [Bommasani, 2021]. Methods of mechanistic interpretability have been developed to address this gap by reverse-engineering neural networks to uncover how internal components implement specific computations and behaviors. Recent advances in this area have successfully identified interpretable circuits, features, and algorithms within LLMs [Nanda et al., 2023, Olsson et al., 2022], showing that large complex models can, in part, be understood mechanistically, opening avenues for improving transparency, reliability, and alignment [Bereska and Gavves, 2024].

However, interpretability is *bottlenecked* by the model itself: even with sophisticated reverse-engineering techniques that can faithfully reveal internal algorithms, the underlying computations implemented by large models can still remain highly complex and uninterpretable. Circuits for seemingly simple tasks may span hundreds of interacting attention heads and MLPs with densely intertwined contributions across layers [Conmy et al., 2023], and features can influence each other along combinatorially many attention-mediated paths, complicating attention attribution [Kamath et al., 2025]. To exemplify this, Figure 1 (top) illustrates the attention patterns of a small, single-head transformer trained on a simple two-digit addition task. Here, the model has learned to solve the task in a highly diffused manner, where information about each token is dispersed across all token locations, rendering the interpretation of the underlying algorithm extremely difficult even in this simple case.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

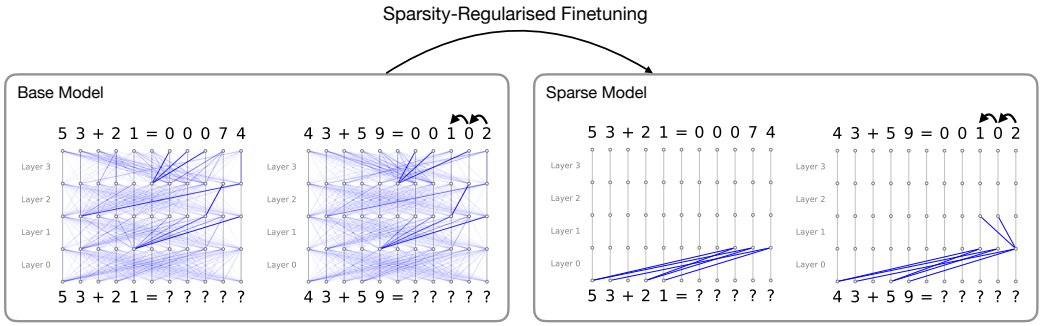


Figure 1: Visualised attention patterns for a 4-layer toy model trained on a simple 2-digit addition task. The main idea of this work is to induce *sparse* attention between tokens via a post-training procedure that optimizes for attention sparsity while maintaining model performance. In this example, while both models are able to correctly predict the sum, the sparse model solves the problem with a naturally *interpretable* circuit. Details of this toy setup and more examples are provided in Appendix A.

The crux of the problem is that models are not incentivised to employ simple algorithms during training. In this work, we advocate for directly embedding interpretability constraints into model design in a way that induces simple circuits while preserving performance. We focus our analysis on attention mechanisms and investigate sparsity regularisation on attention patterns as an inductive bias. To demonstrate how sparse attention patterns can give rise to interpretable circuits, we return to the two-digit addition example: Figure 1 (bottom) shows the attention patterns induced by penalising attention edges during training. Here, the sparsity inductive bias forces the model to solve the problem with much smaller, intrinsically interpretable computation circuits.

In this work, we investigate using this sparsity regularisation scheme as a *post-training* strategy for pre-trained LLMs. We propose a practical method for fine-tuning existing models without re-running pretraining, offering a flexible way to induce sparse attention patterns and enhance interpretability. We show, on models of up to 7B parameters, that our proposed procedure preserves the performance of the base models on pretraining data while reducing the effective attention map to less than 0.4% of its edges. To evaluate our central hypothesis that sparse attention facilitates interpretability, we consider two complementary settings. First, we study circuit discovery, where the objective is to identify the minimal set of components responsible for task performance [Conmy et al., 2023]. We find that sparsified models yield substantially simpler computational graphs: the resulting circuits explain model behaviour using up to four times fewer attention heads and up to two orders of magnitude fewer edges. Second, using cross-layer transcoders [Ameisen et al., 2025], we analyse attribution graphs, which capture feature-level interactions across layers. In this setting, sparse attention mitigates the attention attribution problem by making it possible to identify which attention heads give rise to a given edge, owing to the reduced number of components mediating each connection. We argue that this clarity enables a tighter integration of feature-based and circuit-based perspectives, allowing feature interactions to be understood through explicit, tractable circuits. Taken together, these results position attention sparsity as an effective and practical inductive tool for surfacing the minimal functional backbone underlying model behaviour.

2 Related Work

2.1 Sparse Attention

As self-attention is a key component of the ubiquitous Transformer architecture, a large number of variants of attention mechanisms have been explored in the literature. Related to our approach are sparse attention methods, which are primarily designed to alleviate the quadratic scaling of vanilla self-attention. These methods typically rely on masks based on fixed local and strided patterns [Child et al., 2019] or sliding-window and global attention patterns [Beltagy et al., 2020, Zaheer et al., 2020] to constrain the receptive field of each token. While these approaches are successful in reducing the computational complexity of self-attention, they require hand-defined heuristics that do not reflect the internal computations learned by the model.

Beyond these fixed-pattern sparse attention methods, Top-k attention, which enforces sparsity by dynamically selecting the k most relevant keys per query based on their attention scores, has also been

explored [Gupta et al., 2021, DeepSeek-AI, 2025]. While Top-k attention enables *learnable* sparse attention, it is poorly suited as a post-training interpretability intervention. After training, imposing a hard Top-k constraint directly breaks model performance. The choice of k is also brittle: smaller values remove necessary edges, whereas larger values leave attention patterns too dense to interpret. In contrast, our L1 regularisation induces sparsity gradually during training, preserving performance while avoiding the need to manually choose a discrete sparsity level k. Most importantly, Top-k attention imposes the same number of active edges for every query, rather than allowing sparsity to vary across heads and contexts. This flexibility is crucial in the sparse regimes we study: for instance, in our GPT-2 setting, only 0.22% of edges remain active, yielding far fewer than one active edge per query on average, a regime that cannot be captured by Top-k attention even with $k = 1$. Recently, Lei et al. [2025] introduce a sparsity regularisation scheme for world modelling that reveals sparse token dependencies. We build on this method and examine its role as an inductive bias for interpretability.

2.2 Circuit Discovery and Attribution Graphs

Mechanistic interpretability seeks to uncover how internal components of LLMs implement specific computations. Ablation studies assess performance drops from removing components [Nanda et al., 2023], activation patching measures the effect of substituting activations [Zhang and Nanda, 2023], and attribution patching scales this approach via local linearisation [Syed et al., 2024]. Together, these approaches allow researchers to isolate sub-circuits, minimal sets of attention heads and MLPs that are causally responsible for a given behavior or task [Conmy et al., 2023]. Attention itself plays a dual role: it both routes information and exposes interpretable relational structure, making it a key substrate for mechanistic study. Our work builds on this foundation by leveraging sparsity to simplify these circuits, amplifying the interpretability of attention-mediated computation while preserving model performance.

Mechanistic interpretability has gradually shifted from an emphasis on explicit circuit discovery towards the analysis of internal representations and features. Recent work on attribution graphs and circuit tracing seeks to reunify these perspectives by approximating MLP outputs as sparse linear combinations of features and computing causal effects along linear paths between them [Dunefsky et al., 2024, Ameisen et al., 2025, Lindsey et al., 2025b]. This framework enables the construction of feature-level circuits spanning the computation from input embeddings to final token predictions. Within attribution graphs, edges correspond to direct linear causal relationships between features. However, these relationships are mediated by attention heads that transmit information across token positions. Identifying which attention heads give rise to a particular edge, and understanding why they do so, is essential, as this mechanism forms a fundamental component of the computational graph [Kamath et al., 2025]. A key limitation of current attribution-based approaches is that individual causal edges are modulated by dozens of attention components. We show that this leads to feature-to-feature influences that are overly complex, rendering explanations in terms of other features in the graph both computationally expensive and conceptually challenging.

3 Method

Our main hypothesis is that post-training existing LLMs to encourage sparse attention patterns leads to the emergence of more interpretable circuits. In order to instantiate this idea, we require a post-training pipeline that satisfies three main desiderata:

1. To induce sparse message passing between tokens, we need an attention mechanism that can 'zero-out' attention edges, which in turn enables effective L_0-regularisation on the attention weights. This is in contrast to the standard softmax attention mechanism, where naive regularisation would result in small but non-zero attention weights that still allow information flow between tokens.
2. The model architecture needs to be compatible with the original LLM such that the pre-trained LLM weights can be directly loaded at initialisation.
3. The post-training procedure needs to ensure that the post-trained models do not lose prediction performance compared to their fully-connected counterparts.

To this end, we leverage the Sparse Transformer architecture in the SPARTAN framework proposed in Lei et al. [2025], which uses sparsity-regularised *hard attention* instead of the standard softmax

attention. In the following subsections, we describe the Sparse Transformer architecture and the optimisation setup, highlighting how this approach satisfies the above desiderata.

3.1 Sparse Attention Layer

Given a set of token embeddings, the Sparse Transformer layer computes the key, query, and value embeddings, $\{k_i, q_i, v_i\}$, via linear projections, analogous to the standard Transformer. Based on the embeddings, we sample a binary gating matrix from a learnable distribution parameterised by the keys and queries,

$$A_{ij} \sim \text{Bern}(\sigma(q_i^T k_j)), \tag{1}$$

where $\text{Bern}(\cdot)$ is the Bernoulli distribution and $\sigma(\cdot)$ is the logistic sigmoid function. This sampling step can be made differentiable via the Gumbel Softmax trick [Jang et al., 2017]. This binary matrix acts as a mask that controls the information flow across tokens. Next, the message passing step is carried out in the same way as standard softmax attention, with the exception that we mask out the value embeddings using the sampled binary mask,

$$\text{SparseAttn}(Q, K, V) = \left[A \odot \text{softmax}(\frac{QK^T}{\sqrt{d_k}})\right] V, \tag{2}$$

where d_k is the dimension of the key embeddings and $\odot$ denotes element-wise multiplication. During training, we regularise the expected number of edges between tokens based on the distribution over the gating matrix. Concretely, the expected number of edges for each layer can be calculated as

$$\mathbb{E}\big[|A|\big] = \sum_{i,j} \sigma(q_i^T k_j). \tag{3}$$

Note that during the forward pass, each entry of A is a *hard* binary sample that zeros out attention edges, which serves as an effective L_0 regularisation. Moreover, since the functional form of the sparse attention layer after the hard sampling step is the same as standard softmax attention, pre-trained model weights can be directly used without alterations.[1]

3.2 Constrained Optimisation

In order to ensure that the models do not lose prediction performance during the post-training procedure, as per desideratum 3, we follow the approach proposed by Lei et al. [2025], which employs the GECO algorithm [Rezende and Viola, 2018]. Originally developed in the context of regularising VAEs, the GECO algorithm places a constraint on the performance of the model and uses a Lagrangian multiplier to automatically find the right strength of regularisation during training. Concretely, we formulate the learning process as the following optimisation problem,

$$\min_{\theta} \sum_{l} \mathbb{E}\big[|A_l|\big] \qquad s.t. \quad CE \leq \tau, \tag{4}$$

where A_l denotes the gating matrix at layer l, CE is the standard next token prediction cross-entropy loss, and τ is the required target loss, and θ is the model parameters. In practice, we set this target as the loss of the pre-trained baseline models. We solve this optimisation problem via Lagrangian relaxation, yielding the following max-min objective,

$$\max_{\lambda > 0} \min_{\theta} \left[\sum_{l} \mathbb{E}\big[|A_l|\big] + \lambda(CE - \tau) \right]. \tag{5}$$

This can be solved by taking gradient steps on θ and λ alternately. During training, updating λ automatically balances the strength of the sparsity regularisation: when CE is lower than the threshold, λ decreases, and hence more weight is given to the sparsity regularisation term. This effectively acts as an adaptive schedule which continues to increase the strength of the regularisation until the model performance degrades. Here, the value of τ is selected as a hyperparameter to ensure that the sparse model's performance remains within a certain tolerance of the original base model. In practice, the choice of τ controls a trade off between sparsity and performance: picking a tight τ can lead to a slower training process, whereas a higher tolerance can substantially speed up training at the cost of potentially harming model performance. In Appendix C, we provide further discussion on this optimisation process and its training dynamics.

[1] Technically, the sampled A affects the computation. This can be mitigated by adding a positive bias term inside the sigmoid function to ensure all gates are open at initialisation. Experimentally, we found this to be unnecessary as the models quickly recover their original performance within a small number of gradient steps.

3.3 Practical Considerations

One of the main strengths of our proposed method is that, architecturally, the only difference between a sparse Transformer and a normal one lies in how the dot-product attention is computed. As such, most practical training techniques for optimising Transformers can be readily adapted to our setting. In our experiments, we find the following techniques helpful for improving computational efficiency and training stability.

- **FlashAttention** [Dao, 2023]. FlashAttention has become a standard method for reducing the GPU memory footprint of dot-product attention mechanisms. In Appendix B, we discuss how the sampled sparse attention can be implemented in an analogous manner.
- **LoRA finetuning** [Hu et al., 2022]. Low rank finetuning techniques can significantly reduce the computational requirements for training large models. In our experiments, we verify on a 7B parameter model that LoRA finetuning is sufficiently expressive for inducing sparse attention patterns.
- **Distillation** [Gu et al., 2024]. Empirically, we find that adding an auxiliary distillation loss based on the KL divergence between the base model and the sparse model improves training stability and ensures that the behaviour of the model remains unchanged during post-training.

4 Experiments

To evaluate the effectiveness of our post-training pipeline, we finetune pre-trained LLMs and compare their prediction performance and interpretability before and after applying sparsity regularisation. We perform full finetuning on a GPT-2 base model [Radford et al., 2019](124M parameters) on the OpenWebText dataset [Gokaslan and Cohen, 2019]. To investigate the generality and scalability of our method, we perform LoRA finetuning on the larger OLMo-7B model [Groeneveld et al., 2024] on the Dolma dataset [Soldaini et al., 2024], which is the dataset on which the base model was trained. The GPT-2 model and the OLMo model are trained on sequences of length 64 and 512, respectively. In the following subsections, we first present a quantitative evaluation of model performance and sparsity after sparse post-training. We then conduct two interpretability studies, using activation patching and attribution graphs, to demonstrate that our method enables the discovery of substantially smaller circuits.

4.1 Model Performance and Sparsity

We begin by evaluating both performance retention and the degree of sparsity achieved by post-training. We set cross-entropy targets of 3.50 for GPT-2 (base model: 3.48) and 2.29 for OLMo (base model: 2.25). After training, the mean cross-entropy loss for both models remains within ± 0.01 of the target, indicating that the dual optimisation scheme effectively enforces a tight performance constraint. To quantify the sparsity achieved by the models, we evaluate them on the validation split of their respective datasets and compute the mean number of non-zero attention edges per attention head. We find that the sparsified GPT-2 model activates, on average, only **0.22%** of its attention edges, while the sparsified OLMo model activates **0.44%**, indicating substantial sparsification in both cases. To further verify that this drastic reduction in message passing between tokens does not substantially alter model behaviour, we evaluate the sparsified OLMo model on a subset of the benchmarks used to assess the original model. As shown in Figure 9, the sparse model largely retains the performance of the base model across a diverse set of tasks. In sum, our results demonstrate that sparse post-training is effective in consolidating information flow into a small number of edges while maintaining a commensurate level of performance.

4.2 Circuit Discovery with Activation Patching

We begin by outlining the experimental procedure used for circuit discovery. *Activation patching* [Nanda et al., 2023] is a widely used technique for identifying task-specific circuits in transformer models. In a typical setup, the model is evaluated on pairs of prompts: a *clean* prompt, for which the model predicts a correct target token, and a *corrupted* prompt that shares the overall structure of the clean prompt but is modified to induce an incorrect prediction. Here, the goal is to find the set of

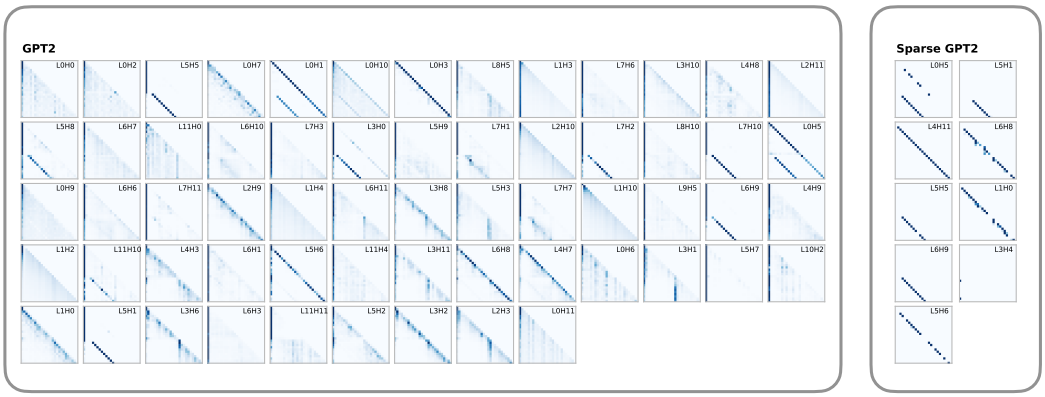


Figure 2: Attention patterns of the attention heads required to explain 90% of model behaviour on a *copy* task. Each panel corresponds to one attention head and shows, for every token in the sequence, how strongly it attends to each earlier tokens. The sparse model requires substantially fewer attention heads. Moreover, the selected heads exhibit the characteristic 'induction head' pattern: each token attends to a previous token at a fixed relative offset, effectively copying information forward through the sequence, a pattern well known to implement the copy mechanism in transformer models. Equivalent plots for OLMo can be found in Appendix E.

model components that is responsible for the model's preference for the correct answer over the wrong one, as measured by the logit difference between the corresponding tokens. In activation patching, individual model components, such as attention heads and individual edges, can be 'switched-off' by patching activation at the specific positions. Circuit discovery amounts to finding a set of components whose replacement causes the model's prediction to shift from the correct to the corrupted answer.

Since searching over every possible subset of model components is infeasible due to the exponential number of potential subsets, we adopt a common heuristic to rank each model component. Specifically, for each individual component, we compute an importance score by replacing the activations of the component with the corrupted activations and measuring its effect on the logit difference. In our experiments, we use this ranking to select the top-k components and intervene on the model by freezing all remaining components, with the goal of identifying the minimal set that accounts for at least 90% of the model's preference for the correct prediction.

Note that these importance scores can be computed at two levels: (i) a *single-sentence* level, using a single pair of correct and corrupted inputs, and (ii) a *global* level, obtained by averaging scores across many task variants. In our experiments, we report the results using *single-sentence* scores. In Appendix E, we also provide results using the *global* scores, which are largely consistent with our main results. There are also two standard approaches for freezing component activations: setting the activation to zero or replacing it with a mean activation value [Conmy et al., 2023]. We evaluate both variants for each model and report results for the patching strategy that yields the smallest circuits.

We first focus on the *copy* task with the following prompt: `"AJEFCKLMOPQRSTVWZS, AJEFCKLMOPQRSTVWZ"`, where the model has to copy the letter `S` to the next token position. This task is well studied and is widely believed to be implemented by emergent *induction heads* [Elhage et al., 2021], which propagate token information forward in the sequence. Figure 2 illustrates the attention patterns of the set of attention heads that explains this prompt for the sparse and base GPT-2 models. See Appendix E for analogous results for the OLMo models. The sparse model admits a substantially smaller set of attention heads (9 heads) than its fully connected counterpart (61 heads). Moreover, the identified heads in the sparse model exhibit cleaner induction head patterns, with each token attending to a single prior position at a fixed relative offset. These results illustrate how sparsification facilitates interpretability under simple ranking-based methods and support our hypothesis that sparse post-training yields models that are more amenable to mechanistic interpretability techniques.

To further verify our hypothesis, we repeat the experiment on classical circuit discovery tasks. For GPT-2, we evaluate variants of the *Indirect Object Identification* (IOI) task, in which the model copies a person's name from the start of a sentence, and the *Greater Than* task, in which the model predicts a number that is larger than a previously mentioned number. To further assess the scalability of our approach, we investigate more challenging and longer horizon tasks for OLMo, including a

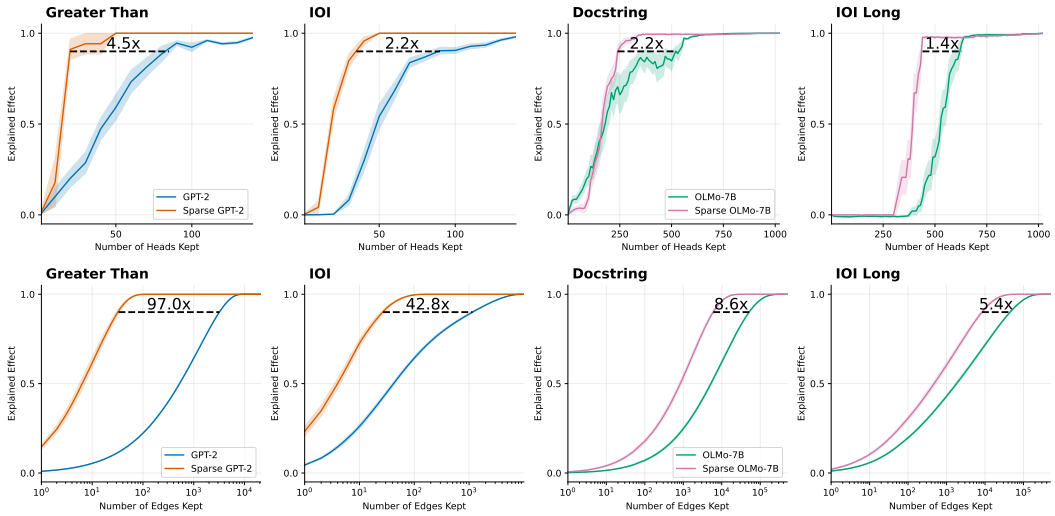


Figure 3: Logit attribution when keeping only the top-k attention components. **Top:** attention **heads**. **Bottom:** attention **edges**. Dotted lines indicate the number of components required to explain 90% of the logit difference. Sparse models yield 1.4×–4.5× smaller circuits at the head level and 5.4×–97× smaller circuits at the edge level. Shaded areas show standard error across 20 prompts.

longer context IOI task and a *Docstring* task where the model needs to predict an argument name in a Docstring based on an implemented function. Details of each task can be found in Appendix G. Figure 3 shows the fraction of model behaviour explained as a function of the number of retained model components (attention heads and attention edges, respectively). Across all tasks and models, the sparse models consistently produce significantly smaller circuits, as measured by the number of model components needed to explain 90% of model prediction. This further corroborates our claim that sparse models lead to simpler and more interpretable internal circuits.

4.3 Distance between Dense and Sparse circuits

Our goal is not to preserve the computational circuit implemented by the dense base model. Rather, our central claim is that dense models often implement behaviors through overly distributed circuits that are difficult to interpret. Sparse post-training is therefore intended to reorganize computation into simpler and more selective pathways while preserving model performance. This raises a natural question: to what extent do the resulting sparse circuits differ from those of the dense base model?

We traced the heads named in the original circuit-discovery papers (IOI circuit Wang et al. [2022], Induction heads Olsson et al. [2022], Greater-Than circuit Hanna et al. [2023]) through the sparsified model. A head "keeps its role" if it still concentrates attention on a task-relevant position, either the position originally documented, or, for heads that shift function, a different position relevant to the same task. Two of the 21 named heads (10,0) and (7,3) were already weak in dense GPT-2 itself (attention below 0.5) and are excluded, leaving 19 heads that were clearly performing their role in the base model. Of these 19, 16 (84%) keep their role after sparsification: 6/6 induction heads, 5/6 Greater-Than heads, 5/7 IOI heads. The passing heads score 0.83–1.00 and the failing heads score 0.00–0.18 (one exception at 0.37), so the result holds for any threshold between roughly 0.2 and 0.8.

One IOI head illustrates the shift-in-function case directly: the negative name mover (10,7) stops attending to the indirect object but shifts to attending the subject at 0.98. Three heads lose their role outright with no replacement: two IOI heads, (11,10) and (7,9), both S-inhibition-related dropping to near-zero attention, and one Greater-Than year head, (8,11), which weakens from 0.59 to 0.37. To check that 84% is meaningful and not just how often any head happens to concentrate attention on a task-relevant position, we computed the same criterion across all 144 heads with no selection at all: only 6.5% do so on average (9.0% induction, 4.9% IOI, 5.6% Greater-Than). Named heads are therefore about 13 times more likely to retain a task-relevant role after sparsification than an arbitrary head.

4.4 Attribution graphs

Next, we present a more fine-grained, feature-level investigation of whether sparsity in attention leads to interpretable circuits in practice using cross-layer transcoders (CLTs). Since training CLTs on OLMo-7B is computationally prohibitive[2], we focus our analysis on the GPT-2 models. For the rest of the section, we perform analysis on CLTs trained on the sparse and base GPT-2 models, trained with an expansion factor of 32 and achieve above 80% replacement score measured with Circuit Tracer [Hanna et al., 2025]. See Appendix H and I for details on training and visualisation.

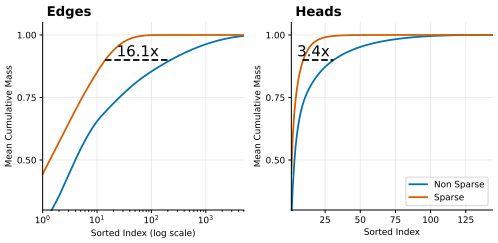


Figure 4: Mean cumulative distribution of the component scores that mediate an attribution graph edge. The components are on the left key-query pairs within a head, and on the right full attention heads.

We study the problem of *attention attribution*, which seeks to understand how edges between features are *mediated*. The key challenge here is that any given edge can be affected by a large number of model components, making mediation circuits difficult to analyse both computationally and conceptually: computationally, exhaustive enumeration is costly; conceptually, the resulting circuits are often large and uninterpretable. In this experiment, we demonstrate that sparse attention patterns induced via post-training substantially alleviate these challenges, as the vast majority of attention components have zero effect on the computation.

As in Ameisen et al. [2025], we define the total attribution score between feature n at layer ℓ and position k, and feature n' at layer ℓ' and position k' as

$$a_{\ell,k,n}^{\ell',k',n'} = f_{k,n}^{\ell} \; J_{\ell,k}^{\ell',k'} \; g_{k',n'}^{\ell'}. \tag{6}$$

Here, $f_{k,n}^{\ell}$ denotes the decoder vector corresponding to feature n at layer ℓ and position k, and $g_{k',n'}^{\ell'}$ is the corresponding encoder vector for feature n' at layer ℓ' and position k'. The term $J_{\ell,k}^{\ell',k'}$ is the Jacobian from the MLP output at (ℓ, k) to the MLP input at (ℓ', k'). This Jacobian is computed during a forward pass in which all nonlinearities are frozen using stop-gradient operations. Under this linearisation, the attribution score represents the sum over all linear paths from the source feature to the target feature.

To analyse how this total effect between two features is mediated by each model component, we define the component-specific attribution by subtracting the contribution of all paths that do not pass through the component:

$$a_{\ell,k,n}^{\ell',k',n'}(h) = f_{k,n}^{\ell} \; J_{\ell,k}^{\ell',k'} \; g_{k',n'}^{\ell'} - f_{k,n}^{\ell} \; \left[J_{\ell,k}^{\ell',k'}\right]_h \; g_{k',n'}^{\ell'}.$$

Here, $\left[J_{\ell,k}^{\ell',k'}\right]_h$ denotes a modified Jacobian computed under the same linearization as above, but with the specific attention component h additionally frozen via stop-gradient. As such, these component-specific scores quantifies how much each model component impacts a particular edge between features.

Empirically, we evaluate the method on ten pruned attribution graphs, computed on the IOI, greater-than, completion, and category tasks. Similar to our previous circuit discovery experiment, we compute attribution scores on the level of attention heads as well as individual key–query pairs. In practice, attention sparsity yields substantial computational savings: because inactive key–query pairs are known *a priori* to have exactly zero attribution score, attribution need only be computed for a small subset of components. This reduces the computation time per attribution graph from several hours to several minutes.

In terms of circuit size, Figure 4 shows the mean cumulative distribution of component attribution scores for each edge in the attribution graph. We find that, to reach a cumulative attribution threshold of 90%, the sparse model on average requires $16.1\times$ fewer key–query pairs and $3.4\times$ fewer attention heads when compared to the dense GPT-2 model, supporting our hypothesis that sparse attention patterns lead to simpler mediation circuits.

[2]The largest open-source CLT is on Gemma-2B at the time of writing.

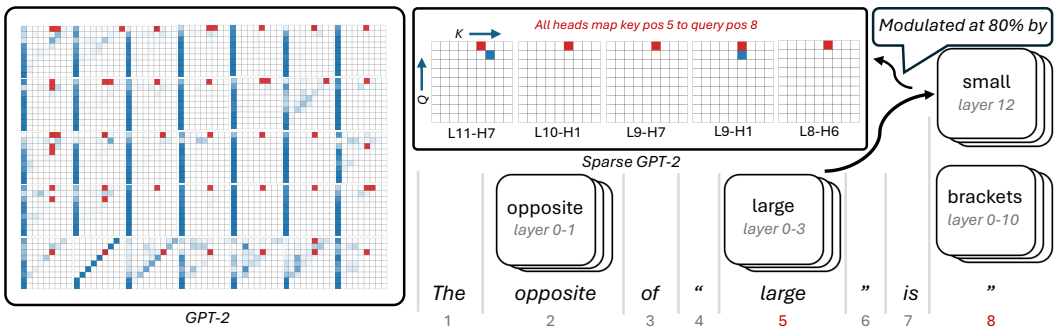


Figure 5: Sketch of the attribution graph for the sentence *"The opposite of 'large' is"*. The cluster of features associated with *large* at token position 5 maps directly to the final next-token prediction logit *small*. We show the attention patterns of all key–query pairs required to account for 80% of the cumulative attribution score. In the sparse-attention setting, this corresponds to five attention heads, compared to more than forty heads in the dense-attention case. In the sparse model, these heads read from token position 5 and write directly to the last token residual stream at token position 8. These heads thus compute in parallel and provide a clear picture of the internal computation.

Next, we present a qualitative case-study to showcase the benefits of sparse attention patterns. For a given key–query pair, we compute the causal effect from all other features in the attribution graph to both the key and the query vectors. Figure 5 illustrates this analysis for the prompt *"The opposite of 'large' is"*. The resulting attribution graph decomposes into four coherent clusters of features: features related to *opposite*, features related to *large*, features activating on bracketed tokens, and the final next-token logit corresponding to *small* (see Appendix J for examples of features and visualization).

Here, the features in the *large* cluster are directly connected to the *small* logit. The key question is then to understand how this connection from the *large* to the *small* logit comes about. To this end, we analyse their mediation structure. We find that 80% of the cumulative absolute attribution score of the edges connecting the *large* cluster to the *small* logit is mediated by the same five late layer attention key–query pairs. These attention components map features from token position 5 directly into the final-layer residual stream at position 8, and thus operate in parallel.

For these five key–query pairs, we then compute the causal influence of all other features in the graph on their key and query vectors. The query vectors are primarily modulated by features associated with bracketed tokens in the last token position, while the key vectors are driven by strongly active features in both the *opposite* and *large* clusters, details are provided in Appendix J and illustrated in Figure 17. These results are in agreement with the recent work on attention attribution and the "opposite of" attribution graph [Kamath et al., 2025].

In stark contrast, Figure 5 (left) shows that a similar (and more computationally expensive) analysis on the dense model produces a much more complicated circuit. This case study illustrates the potential of sparse attention in the context of attribution graphs, as it enables a unified view of features and circuits. By jointly analyzing feature activations, attention components, and their mediating roles, we obtain a more faithful picture of the computational graph underlying the model's input–output behavior.

5 Conclusion

Achieving interpretability requires innovations in both interpretation techniques *and* model design. We investigate how large models can be trained to be intrinsically interpretable. We present a flexible post-training procedure that sparsifies transformer attention while preserving the original pretraining loss. By minimally adapting the architecture, we apply a sparsity penalty under a constrained-loss objective, allowing the pre-trained model to reorganise its connectivity into a much more selective and structured pattern. Mechanistically, this induced sparsity gives rise to substantially simpler circuits: task-relevant computation concentrates into a small number of attention heads and edges. Across a range of tasks and analyses, we show that sparsity improves interpretability at the *circuit* level by reducing the number of components involved in specific behaviours. In circuit discovery experiments,

most of the model's behaviour can be explained by circuits that are orders of magnitude smaller than in dense models; in attribution graph analyses, the reduced number of mediating components renders attention attribution tractable. Together, these results position sparse post-training of attention as a practical and effective tool for enhancing the mechanistic interpretability of pre-trained models.

Limitations and Future Work. One limitation of the present investigation is that, while we deliberately focus on sparsity as a post-training intervention, it remains an open question whether injecting a sparsity bias directly during training would yield qualitatively different or simpler circuit structures. Moreover, our study is primarily restricted to sparsifying attention patterns, the underlying principle of leveraging sparsity to promote interpretability naturally extends to other components of the transformer architecture. As such, combining the proposed method with complementary approaches for training intrinsically interpretable models, such as Sparse Mixture-of-Experts [Yang et al., 2025], sparsifying model weights [Gao et al., 2024], or limiting superposition [Miller et al., 2026] offers a promising direction for future work. Another exciting avenue for future work is to apply the sparsity regularisation framework developed here within alternative post-training paradigms, such as reinforcement learning [Ouyang et al., 2022, Zhou et al., 2024] or supervised fine-tuning [Pareja et al., 2025].

Impact Statement. This work may improve the auditability of language models by making attention patterns more sparse and easier to inspect. Sparse circuits can help identify which tokens, heads, and edges contribute to a prediction, but they should not be treated as complete explanations of model behaviour. We therefore recommend evaluating sparsity together with performance preservation and interpretability diagnostics.

Acknowledgment

F. D. acknowledges support through a fellowship from the Hector Fellow Academy. A. L. is supported by an EPSRC Programme Grant (EP/V000748/1). I. P. holds concurrent appointments as a Professor of Applied AI at the University of Oxford and as an Amazon Scholar. This paper describes work performed at the University of Oxford and is not associated with Amazon. The authors gratefully acknowledge the role of the ELLIS PhD Programme in bringing together the authors to lay the foundations for this work.

References

Emmanuel Ameisen, Jack Lindsey, Adam Pearce, Wes Gurnee, Nicholas L. Turner, Brian Chen, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. Circuit tracing: Revealing computational graphs in language models. *Transformer Circuits Thread*, 2025. URL `https://transformer-circuits.pub/2025/attribution-graphs/methods.html`.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

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

Rishi et al. Bommasani. On the opportunities and risks of foundation models. *ArXiv*, 2021. URL `https://crfm.stanford.edu/assets/report.pdf`.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Tom Conerly, Hoagy Cunningham, Adly Templeton, Jack Lindsey, Basil Hosmer, and Adam Jermyn. Circuits updates – january 2025, 2025. URL `https://transformer-circuits.pub/2025/january-update/index.html`. Transformer Circuits Thread.

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

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

DeepSeek-AI. Deepseek-v3.2: Pushing the frontier of open large language models, 2025. URL https://arxiv.org/abs/2512.02556.

Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. Transcoders find interpretable LLM feature circuits. *Advances in Neural Information Processing Systems*, 37:24375–24410, 2024.

Nelson Elhage, Neel Nanda, et al. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Leo Gao, Achyuta Rajaram, Jacob Coxon, Soham V. Govande, Bowen Baker, and Dan Mossing. Weight-sparse transformers have interpretable circuits. Technical report, OpenAI, 2024. URL https://cdn.openai.com/pdf/41df8f28-d4ef-43e9-aed2-823f9393e470/circuit-sparsity-paper.pdf.

Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, Will Smith, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah A. Smith, and Hannaneh Hajishirzi. Olmo: Accelerating the science of language models. *Preprint*, 2024.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=5h0qf7IBZZ.

Ankit Gupta, Guy Dar, Shaya Goodman, David Ciprut, and Jonathan Berant. Memory-efficient transformers via top-k attention. *arXiv preprint arXiv:2106.06899*, 2021.

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

Michael Hanna, Mateusz Piotrowski, Jack Lindsey, and Emmanuel Ameisen. circuit-tracer. https://github.com/safety-research/circuit-tracer, 2025. The first two authors contributed equally and are listed alphabetically.

Abir Harrasse, Florent Draye, Zhijing Jin, and Bernhard Schölkopf. Tracing multilingual representations in llms with cross-layer transcoders. *arXiv preprint arXiv:2511.10840*, 2025.

Stefan Heimersheim and Jett Janiak. A circuit for python docstrings in a 4-layer attention-only transformer. In *Alignment Forum*, 2023.

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

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rkE3y85ee.

Harish Kamath, Emmanuel Ameisen, Isaac Kauvar, Rodrigo Luger, Wes Gurnee, Adam Pearce, Sam Zimmerman, Joshua Batson, Thomas Conerly, Chris Olah, and Jack Lindsey. Tracing attention computation through feature interactions. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/attention-qk/index.html.

Anson Lei, Bernhard Schölkopf, and Ingmar Posner. SPARTAN: A sparse transformer world model attending to what matters. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=uS5ch7GjZ4.

Jack Lindsey, Emmanuel Ameisen, Neel Nanda, Stepan Shabalin, Mateusz Piotrowski, Tom McGrath, Michael Hanna, Owen Lewis, Curt Tigges, Jack Merullo, Connor Watts, Gonçalo Paulo, Joshua Batson, Liv Gorton, Elana Simon, Max Loeffler, Callum McDougall, and Johnny Lin. The circuits research landscape: Results and perspectives. *Neuronpedia*, 2025a. URL https://neuronpedia.org/graph/info.

Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the biology of a large language model. *Transformer Circuits Thread*, 2025b. URL https://transformer-circuits.pub/2025/attribution-graphs/biology.html.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173, 2024.

Jack Merullo, Connor Watts, Max Loeffler, Liv Gorton, Elana Simon, Tom McGrath, and Owen Lewis. Replicating circuit tracing for a simple known mechanism, June 2025. URL https://www.goodfire.ai/research/replicating-circuit-tracing-for-a-simple-mechanism. Accessed: 2025-10-07.

Moritz Miller, Florent Draye, and Bernhard Schölkopf. Identifying intervenable and interpretable features via orthogonality regularization, 2026. URL https://arxiv.org/abs/2602.04718.

Amirkeivan Mohtashami and Martin Jaggi. Landmark attention: Random-access infinite context length for transformers. *arXiv preprint arXiv:2305.16300*, 2023.

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

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

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Aldo Pareja, Nikhil Shivakumar Nayak, Hao Wang, Krishnateja Killamsetty, Shivchander Sudalairaj, Wenlong Zhao, Seungwook Han, Abhishek Bhandwaldar, Guangxuan Xu, Kai Xu, Ligong Han, Luke Inglis, and Akash Srivastava. Unveiling the secret recipe: A guide for supervised fine-tuning small LLMs. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=eENHKMTOfW.

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

Danilo Jimenez Rezende and Fabio Viola. Taming vaes, 2018. URL https://arxiv.org/abs/1810.00597.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Harsh Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Pete Walsh, Luke Zettlemoyer, Noah A. Smith, Hannaneh Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge,

and Kyle Lo. Dolma: an Open Corpus of Three Trillion Tokens for Language Model Pretraining Research. *arXiv preprint*, 2024.

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

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

Xingyi Yang, Constantin Venhoff, Ashkan Khakzar, Christian Schroeder de Witt, Puneet K Dokania, Adel Bibi, and Philip Torr. Mixture of experts made intrinsically interpretable. *arXiv preprint arXiv:2503.07639*, 2025.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. *arXiv preprint arXiv:2309.16042*, 2023.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. Archer: Training language model agents via hierarchical multi-turn rl. In *ICML*, 2024. URL `https://openreview.net/forum?id=b6rA0kAHT1`.

A Two-Digit Addition Study

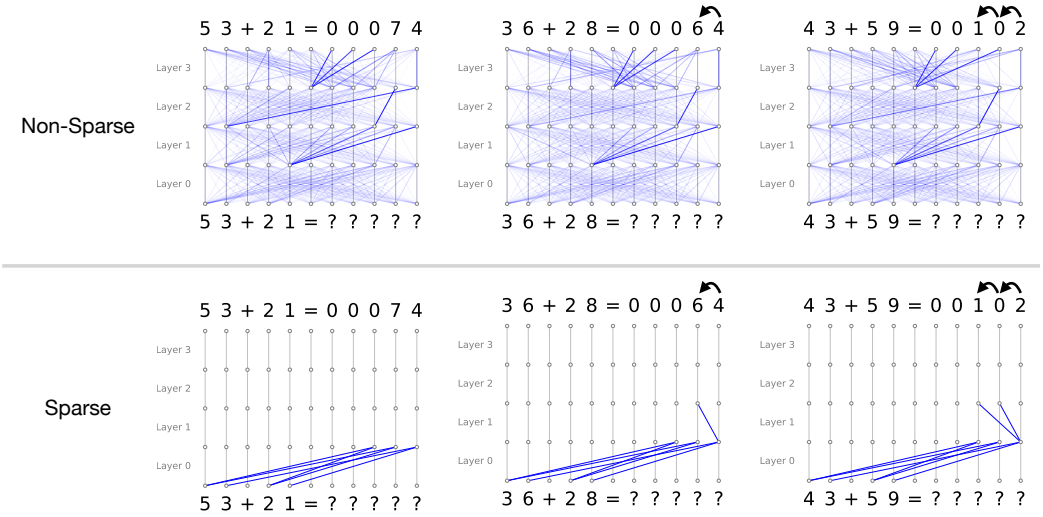


Figure 6: Simple example showing the attention patterns (shown in blue) of sparse and non-sparse transformers trained on a two digit addition task. Both models are able to correctly predict the sum, but the attention patterns are very different: the non-sparse model solves the task with highly dispersed information flow, while the sparse model uses a highly interpretable attention pattern: in Layer 0, the model first attends to the corresponding digits to be added, then in Layer 1, it attends to the carry bit only if it is needed (see middle and right columns, where the model has to carry once and twice respectively).

In the introduction, we used a two-digit addition task to demonstrate how sparse attention patterns can lead to intrinsically interpretable circuits. The result presented is gathered in a small scale toy experiment described below. We train 4-layer single-head Transformer models on a two-digit addition task, where the input is a sequence of digits and the model is trained to predict the sum. In this task, there are 13 total tokens: ten digits and three symbols "+", "=" and "?".

Within this setting, we train two models: a standard transformer model and a sparse transformer with a fixed sparsity regularisation strength. Figure 6 shows several examples of the learned attention patterns. In these examples, we can clearly see that the pressure of sparsity leads to the emergence of human-recognisable algorithmic patterns: in the first layer, each digit in the answer attends to the corresponding digits in the input, while the second layer computes the carry bit when necessary. By enforcing selective information flow through sparse message-passing, the sparse model is able to learn crisp and localised mechanisms that are immediately amenable to interpretation.

B Sparse Attention Implementation

For the experiments, we implemented efficient GPU kernels for the sparse attention layers using the `helion` domain-specific language[3]. We refer to this implementation as **Splash Attention** (Sparse flash Attention). Our implementation follows the same core algorithmic structure as FlashAttention-2 [Dao, 2023], including the use of online softmax computation and tiling. Note that the sparse attention variant (Eq. 2) only differs from the standard attention by a pointwise multiplication of the adjacency matrix, which can be easily integrated into FlashAttention by computing A_{ij} on-the-fly. We additionally fuse the Gumbel-softmax computation, the straight-through gradient, and the computation of the expected number of edges (required for the penalty) into a single optimized kernel. Figure 7 compares our Splash Attention implementation against a naive sparse attention baseline based on PyTorch-native operations, and FlashAttention. The resulting implementation achieves similar performance gain as FlashAttention compared to the naive attention. Splash Attention achieves approximately 45% MFU on an NVIDIA A100, which is slightly lower than FA, but is expected, as the gating mechanism requires additional pointwise operations. Code is available at `https://github.com/florentdraye/sparse-attention-post-training`.

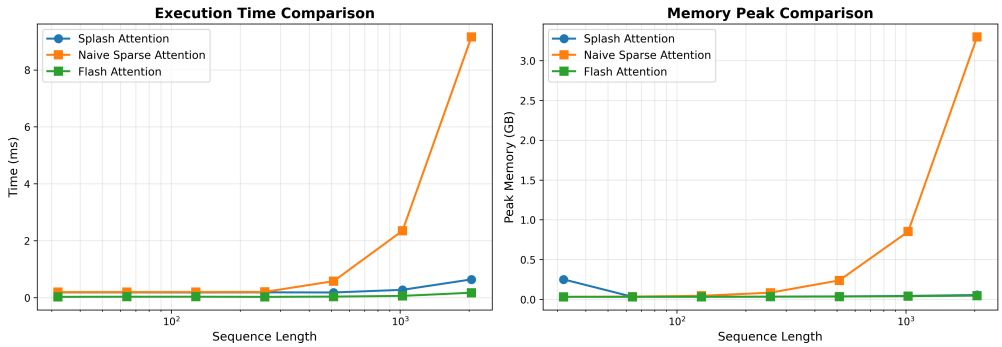


Figure 7: Performance comparison between our implementation (Splash) and a naive PyTorch sparse attention baseline. Input tensors have shape $(\texttt{batch}, \texttt{heads}, \texttt{tokens}, \texttt{head dim}) = (8, 8, 2^{[5,6,7,8,9,10,11]}, 16)$ and data type `bfloat16`.

[3] `https://helionlang.com/`

C Training Details

C.1 Hyperparameters and Compute Resources

Hyperparameter	OLMo	GPT-2
Base Model	`allenai/OLMo-7B-hf`	`gpt2`
Context window	512	64
Dataset	`dolma-v1`	`OpenWebText`
Batch size	16	256
Gradient accumulation steps	4	4
Total steps	400,000	1,200,000
Learning rate	1×10^{-5}	1×10^{-5}
Minimum learning rate	1×10^{-6}	1×10^{-6}
Optimizer	Adam	Adam
Weight decay	0.1	0.1
Scheduler	Cosine (1 cycle)	Cosine (1 cycle)
Warmup steps	1,000	1,000
Finetuning strategy	LoRA	Full
LoRA rank (r)	400	-
LoRA scaling (α)	800	-
LoRA dropout	0	-
LoRA target modules	`q,k,v,o,fc_in,fc_out`	-
Dual Optimisation LR	0.01	0.1
Target cross-entropy	2.29	3.5

Table 1: Key hyperparameters used for sparse post-training experiments on OLMo-7B.

We provide the key hyperparameters for our experiments in table 1. All experiments were conducted on a single NVIDIA H100 GPU. Both GPT-2 and the OLMo model were trained in this single-GPU setting to reflect the computational constraints commonly encountered in interpretability academic research. The total training time is 7 and 14 days for GPT-2 and OLMo respectively. The main implementation code is provided as supplementary material and will be publicly released upon publication, together with the model weights.

C.2 Training Dynamics

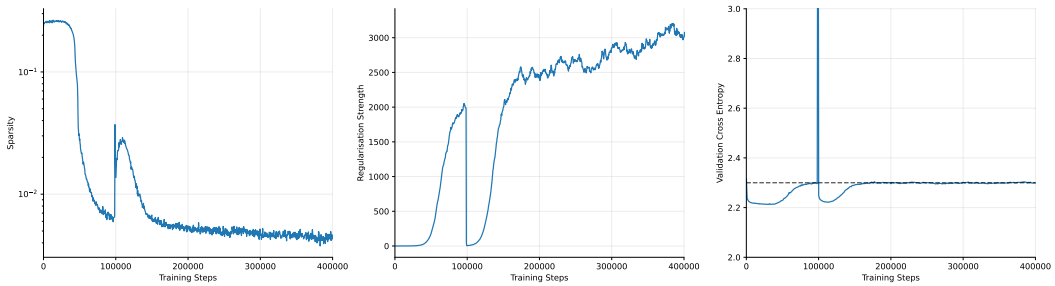


Figure 8: The training curves for post-training OLMo-7B tacking the model sparsity (left), regularisation strength (middle), and the cross-entropy loss (right). The black dotted line on the cross-entropy plot indicates the pre-defined threshold, τ.

A key feature of our post-training framework is that the strength of the sparsity regularisation is automatically controlled via a constrained optimisation scheme. By pre-specifying an accepted level for the cross-entropy target, τ, the training procedure can be written as the max-min objective:

$$\max_{\lambda>0} \min_{\theta} \left[\sum_{l} \mathbb{E}\big[|A_l|\big] + \lambda(CE - \tau) \right], \tag{7}$$

which can be optimised by taking alternating gradient steps in the model weight space and in the λ space. The resulting training dynamics means that the sparsity regularisation strength increases when the model cross-entropy is lower than the target, and decreases when the model is above the threshold. Figure 8 shows the training curves for the OLMo-7B model. Here, we observe that the strength of sparsity regularisation keeps increasing slowly while the model cross-entropy is clipped at the desired level. Note that during a model spike (at around 100K steps), the sparsity regularisation automatically decreases to let the model recover.

D Benchmark Performance

We report additional benchmark evaluations comparing OLMo-7B with its sparsified counterpart. As shown in Figure 9, the sparsified model retains performance close to the dense baseline across the evaluated tasks, despite using only a small fraction of active attention edges.

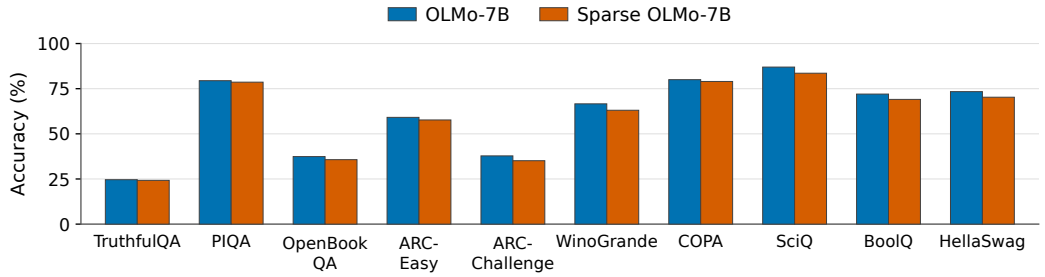


Figure 9: Comparison of model performance between the base OLMo model and the sparsified model across several benchmarks.

We also evaluate the sparse GPT-2 model from the main text using a subset of prompts from each benchmark which can fit into the 64 context window. The results are reported in the table below, where we also note the proportion of the benchmark prompts that can fit into the context window.

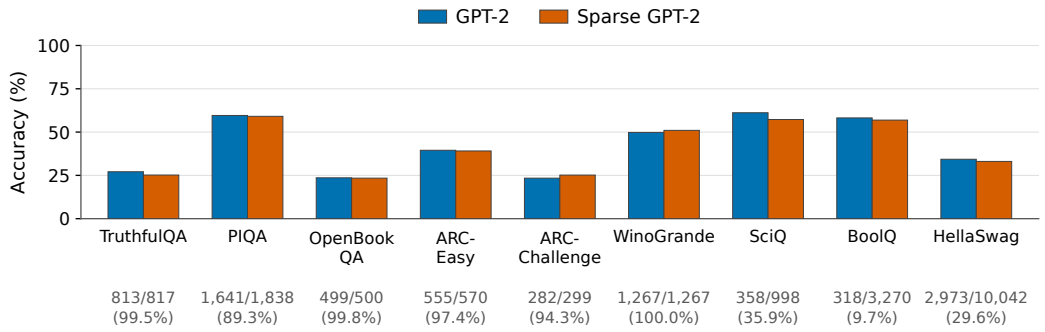


Figure 10: Comparison of model performance between the base GPT-2 model and the sparsified model across several benchmarks.

Long-context evaluation. While evaluating long-context performance of sparse-attention is a meaningful question, it is not the primary focus of this work, which is centered on mechanistic interpretability. In practice, interpretability research is typically conducted on relatively short contexts and controlled tasks (e.g., Nanda et al. [2023], Conmy et al. [2023]), where circuits can be precisely analysed. The complexity of the tasks considered in our work is therefore commensurate with, and in some cases exceeds, what is standard in the field, even if the absolute context length is modest. To nevertheless probe whether our method extends to longer contexts, and given that the original OLMo-7B context-window is 2048, we conduct a preliminary experiment using a LLaMA 3.2 1B model with a context window of 8192. We finetune both dense and sparse variants on Dolma 3 LongMiNo Mix for 5B tokens. The sparse model achieves approximately 0.4% active attention patterns while maintaining a cross-entropy comparable to the dense model. We evaluate both models using the same context on two long-context retrieval tasks, Needle-in-a-Haystack and Passkey Retrieval [Liu et al., 2024, Mohtashami and Jaggi, 2023]. The dense model achieves accuracies of 0.918 and 0.925, respectively, while the sparse model achieves 0.919 and 0.922. These results show that our method can preserve long-context performance even under extreme sparsity. However, we emphasise that this experiment should be viewed as preliminary evidence rather than a comprehensive evaluation.

E Extra Experiments for Circuit Discovery

In this section, we provide additional results for the activation patching circuit discovery experiment presented in the main text.

E.1 Sparse Circuit Visualization

In Figure 11, we provide a qualitative visualization of the attention edges required to solve the IOI task. While Figure 3 quantifies how many edges must be preserved to maintain task performance, Figure 11 shows what the resulting circuit looks like visually. The resulting sparse circuit is drastically simpler, highlighting a small subset of edges that carries most of the task-relevant computation.

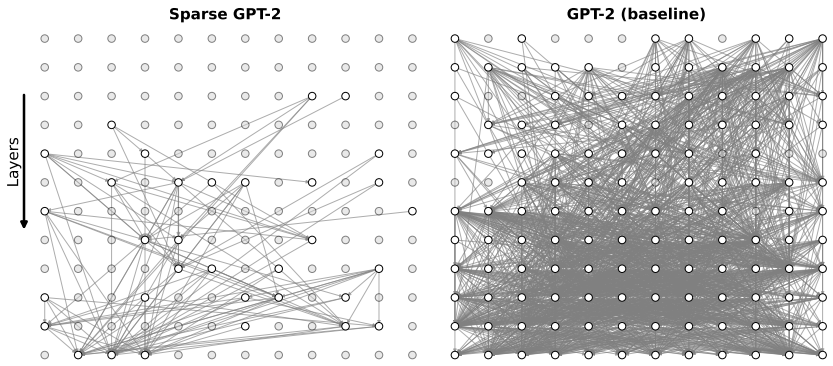


Figure 11: Qualitative visualization of the attention-head edges required to reach a cumulative attribution score of 0.9 on the IOI task, using scores averaged across prompts. The sparse circuit is drastically simpler, retaining only the most important task-relevant edges.

E.2 Example of Sparse-Attention Patterns

We illustrate an example of attention patterns across layers and heads for one instance of the IOI task: "While John and Emma were working at the office, John gave a pen to". Figure 12 compares the baseline GPT-2 model with its sparse counterpart. The sparse model exhibits a marked reduction in the number of active attention entries, with several heads appearing almost entirely inactive. Nevertheless, some heads retain structured patterns, including diagonal attention patterns.

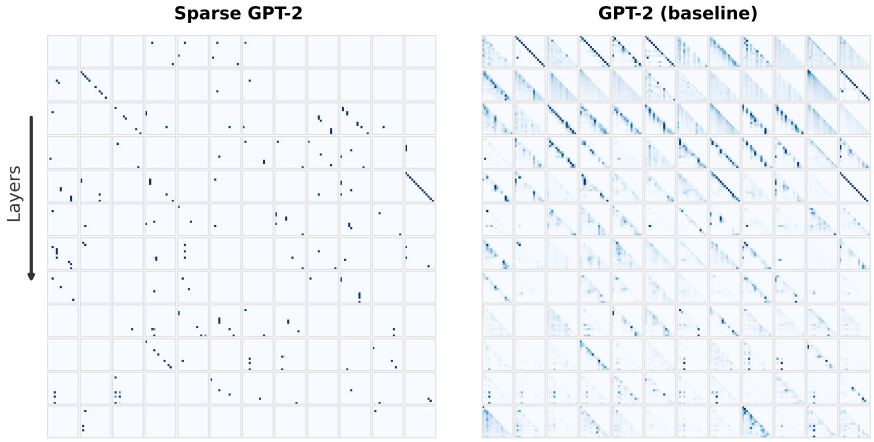


Figure 12: Attention patterns for a baseline GPT-2 model and its sparse counterpart on an IOI prompt. Darker blue entries indicate stronger attention weights.

E.3 Activation and Attribution Patching

Figures 13 shows the fraction of explained model preference as a function of the number of model components kept unablated. Unlike Figure 3, where components are ranked separately for each prompt, here components are ranked at the *task* level: each head or edge receives a single importance score pooled across different instances of the same task. Overall, the results are consistent with the main findings: sparse models tend to admit smaller circuits than dense models under a global ranking strategy.

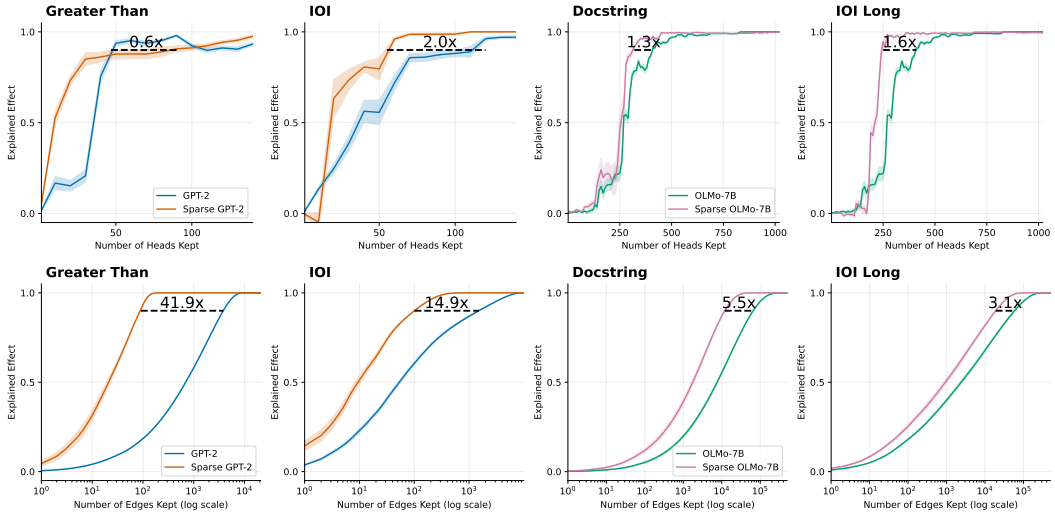


Figure 13: Logit attribution when keeping only the top-k attention components according to a *global*, task-level ranking. **Top:** attention **heads**. **Bottom:** attention **edges**. Dotted lines indicate the number of components required to explain 90% of the logit difference. In contrast to the single-prompt ranking in Figure 3, each component is assigned one importance score pooled across prompts from the same task. Overall, sparse models yield smaller circuits than dense models. Shaded areas show standard error across prompts.

E.4 Attention Sink

An interesting interpretability result is that sparse-attention substantially reduces persistent BOS sinks. In GPT-2, the mean fraction of attention directed to the BOS token increases after the early layers and remains high throughout later layers. By contrast, the sparse model shows BOS attention mainly in the first three layers, followed by a sharp drop after layer 2 and approximately zero attention from layer 5 onward. Results are averaged over 300 length-64 WikiText-2 sequences.

Table 2: Mean fraction of attention directed to the BOS token across layers. Results are averaged over 50 length-64 WikiText-2 sequences.

Model	L0	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11
Sparse	0.24	0.08	0.10	0.02	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Dense	0.12	0.21	0.21	0.43	0.47	0.67	0.63	0.74	0.64	0.72	0.70	0.57

E.5 OLMo Induction Heads

Figure 14 shows the attention patterns of the heads required to explain 90% of model behaviour on a *copy* task. To fully test the longer context window afforded by OLMo, we use a longer prompt than the one used for GPT2 in the main text. The result is consistent with the GPT-2 experiment: sparsified model facilitates the discovery of smaller circuits of induction heads that implement the copy task.



Figure 14: Attention patterns of the heads required to explain 90% of model behaviour on a longer *copy* task. Similar to the GPT-2 results in Figure 2, the sparse model requires substantially fewer attention heads.

E.6 Some Additional Experiments on LLaMA-3.2-1B

We also include circuit-discovery results on a LLaMA-3.2-1B which was trained on OpenWebText, showing that our method can also induce a similar level of sparsity (0.3%) while maintaining the base model's performance. Repeating our main circuit discovery experiment on these models reveals consistent results: Sparse LLaMA results in a substantially smaller circuit than the base model (summarized below).

Task	Single	Global
Greater-Than	$4.7\times$ / $1280\times$	$3.0\times$ / $278\times$
IOI	$4.3\times$ / $233\times$	$2.1\times$ / $52\times$

Values report component reduction / edge reduction relative to LLaMA at 90% explained-effect threshold.

F Comparaison with Top-k

In order to investigate whether our sparsifying scheme is effective beyond naive approaches, we present two sets of analyses on how performance changes with Top-k schemes. We first apply direct thresholding on the base model, retaining only the Top-k attention values for each query and attention head and measure the increase in cross-entropy. These results show that dense GPT-2 requires approximately 25% of the attention edges $k = 16$ to reach a cross-entropy gap of 0.02, whereas our sparse GPT-2 model achieves it while activating only 0.22% of the attention edges.

GPT-2 (base)	$k = 1$	$k = 2$	$k = 4$	$k = 8$	$k = 16$	$k = 32$
ΔCE	2.78	1.45	0.57	0.08	0.02	0.00

We next compare against trained Top-k attention, which is the main alternative to our proposed method. We train on 7B tokens from OpenWebText. Note that Top-k retains exactly k attention connections for every query, imposing the same predefined budget across all heads and layers. In contrast, our approach allows connectivity to vary across heads, layers, and inputs. Training with fixed Top-k improves over post-hoc pruning. Nevertheless, it still requires $k = 8$ (12.5% retained context) to recover predictive performance compared.

GPT-2 (Top-k trained)	$k = 1$	$k = 2$	$k = 4$	$k = 8$
ΔCE	0.74	0.10	0.04	0.02

To more directly see the impact of Top-k approaches on downstream interpretability pipelines, we also evaluate the trained Top-k model using the same activation-patching circuit-size metric as in our main experiments. The Top-k model with $k = 1$ does not solve either task. At the 90% explained-effect threshold, the Top-k model with $k = 2$ requires 34 components on IOI and 49 on Greater-Than, compared with 20 and 18 components for our approach, while the CE-matched requires 36 and 48.

G Circuit Discovery Tasks

In the following, we provide the details and the prompts for the various tasks used in section 4.2.

G.1 Greater-Than Task

Each example contains a *clean* prompt, a *corrupt* prompt, and two disjoint sets of candidate continuations, `answers` and `wrong_answers`. A typical entry is:

```
{
  "clean":   "The demonstrations lasted from the year 1363 to 13",
  "corrupt": "The demonstrations lasted from the year 1301 to 13",
  "answers": ["64", "65", ..., "99"],
  "wrong_answers": ["00", "01", ..., "63"]
}
```

For the clean prompt, any token in `answers` yields an end year strictly greater than the start year (e.g. `"1364"`–`"1399"`), whereas tokens in `wrong_answers` correspond to years that are less than or equal to the start year. The corrupt prompt changes only the starting year, shifting which continuations correspond to valid end years. We use the logit difference between the aggregated probability mass on `answers` vs. `wrong_answers` in clean vs. corrupt contexts as our signal, in the spirit of prior mechanistic studies on simple algorithmic tasks [Elhage et al., 2021, Nanda et al., 2023].

G.2 Indirect Object Identification (IOI) Task

Our IOI setup follows the standard indirect object identification paradigm for mechanistic interpretability [Elhage et al., 2021, Conmy et al., 2023]. Each example is generated by combining:

- a pair of names (A, B), e.g. `(" Mary", " John")`;
- a natural-language template with placeholders `[A]`, `[B]`, and `[S]`.

We instantiate templates such as:

```
"Then, [B] and [A] went to the park. [S] gave a ball to"

"When [B] and [A] got a snack at the cafe, [S] decided to give it to"

"After the lunch, [B] and [A] went to the mall. [S] gave a gift to"
```

by sampling a name pair and substituting $[A]$ and $[B]$, then choosing the subject $[S]$ (either one of the pair). The correct continuation is the indirect object, i.e. the other member of the pair.

For example, with $(A, B) =$ `(" John", " Mary")` and $S = B$, one instance is:

```
Then, Mary and John went to the park.  Mary gave a ball to
```

The correct continuation is `" John"`, while `" Mary"` and any distractor names are treated as incorrect candidates.

In the OLMo experiments, in order to further test the capability of our approach, we use a different set of IOI task with increased complexity and prompt length. Example templates include:

```
"After several months without any contact due to conflicting schedules and
unexpected personal obligations, [B] and [A] finally met again at the park,
where they spent a long afternoon catching up on past events, sharing stories,
and reflecting on how much had changed. As the day came to an end, [S] gave
a ball to"

"Although [B] and [A] had previously been involved in a long and emotionally
charged argument that left several issues unresolved, they agreed to meet in
order to clarify their misunderstandings. After a tense but honest conversation,
[S] said to"
```

G.3 Docstring Task

We also test the OLMo models on a more complex *Docstring* task [Heimersheim and Janiak, 2023, Conmy et al., 2023], where the model needs to attend to a specific argument for a specified function in order to complete a Docstring. Similarly to the *Greater Than* task, each example contains a *clean* prompt, a *corrupt* prompt, and two disjoint sets of candidate continuations. A typical entry is:

```
{
  "clean": "def model(self, results, old, option):
                """
                stage agency security vision spot tone joy session river unit
                :param results: bone paper selection sky
                :param old: host action hell miss
                :param",
  "corrupt": "def model(self, command, output, state):
                """
                stage agency security vision spot tone joy session river unit
                :param old: bone paper selection sky
                :param results: host action hell miss
                param",
  "answers": [" option"],
  "wrong_answers": [" results"," old"]
}
```

H Cross-Layer-Transcoder

To implement a cross-layer transcoder (CLT), let $\mathbf{h}_\ell \in \mathbb{R}^{d_{\text{model}}}$ denote the input to the MLP at layer ℓ for a single token position. This representation is projected into a sparse feature space via an encoder,

$$\mathbf{z}_\ell = \mathrm{ReLU}\big(\mathbf{W}_{\text{enc}}^\ell \mathbf{h}_\ell + \mathbf{b}_{\text{enc}}^\ell\big) \in \mathbb{R}^{d_{\text{features}}}, \tag{8}$$

where $\mathbf{W}_{\text{enc}}^\ell \in \mathbb{R}^{d_{\text{features}} \times d_{\text{model}}}$ and $\mathbf{b}_{\text{enc}}^\ell \in \mathbb{R}^{d_{\text{features}}}$ are layer-specific encoder parameters.

The CLT reconstructs the MLP output at a target layer ℓ' by linearly aggregating feature activations originating from all preceding layers,

$$\hat{\mathbf{m}}_{\ell'} = \sum_{\ell \le \ell'} \mathbf{W}_{\text{dec}}^{\ell \to \ell'} \mathbf{z}_\ell + \mathbf{b}_{\text{dec}}^{\ell'}, \tag{9}$$

where $\mathbf{W}_{\text{dec}}^{\ell \to \ell'} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{features}}}$ denotes the decoder mapping from layer ℓ to layer ℓ'.

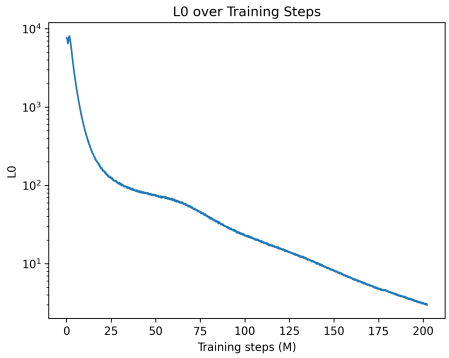
The summation over layers reflects the fact that a given semantic feature may manifest in different representations across multiple MLP layers. For example, a feature that emerges in the MLP at layer ℓ may reappear, potentially in a transformed form, in the outputs of subsequent MLPs. Without accounting for these layer-dependent variations, such duplicated representations would lead to redundant nodes in the attribution graph. By allowing features to be represented differently across layers while being linked through a shared latent space, the cross-layer transcoder avoids this duplication and yields a more compact and interpretable attribution structure. For a detailed comparison between cross-layer transcoders and standard transcoders, we refer the reader to Lindsey et al. [2025a].

Following the training procedure proposed by Anthropic [Ameisen et al., 2025], the final objective combines reconstruction accuracy with sparsity and dead-feature regularization:

$$\begin{aligned}\mathcal{L} = & \underbrace{\sum_{\ell'} \|\hat{\mathbf{m}}_{\ell'} - \mathbf{m}_{\ell'}\|_2^2}_{\text{MSE reconstruction}} \\ & + \lambda_0 \underbrace{\sum_{\ell} \tanh\big(C\,(\mathbf{z}_\ell \odot \|\mathbf{W}_{\text{dec}}^\ell\|)\big)}_{L_0 \text{ sparsity}} \\ & + \lambda_{\text{df}} \underbrace{\sum_{\ell} \mathrm{ReLU}\Big(\exp(\tau) - \mathbf{h}_\ell^{\text{pre}}\Big) \|\mathbf{W}_{\text{dec}}^\ell\|}_{\text{dead-feature penalty}},\end{aligned} \tag{10}$$

where $\mathbf{W}_{\text{dec}}^\ell$ denotes the concatenated decoder weights associated with layer ℓ, $\mathbf{h}_\ell^{\text{pre}}$ are the corresponding pre-activation values, τ is a threshold parameter, and C is a scaling constant. The hyperparameters λ_0 and λ_{df} control the strength of the sparsity and dead-feature regularization terms. We initialize the weights with following circuits updates [Conerly et al., 2025]. The encoder bias is initialize to have a fixed proportion of the features active at initialization. We provide in Figure 15 the training curves of the sparsity value, the sparsity coefficient, the explained variance, and the amount of dead features. We further provide in Table 3 the training configuration and hyperparameters used to train the GPT-2 CLTs. We hope this can help the community in training their own CLTs.

Training CLTs is computationally expensive, with costs scaling in the number of features and layers. As a result, aside from Anthropic's foundational work [Ameisen et al., 2025, Lindsey et al., 2025b], only a small number of applications exist, including a "greater-than" mechanism study [Merullo et al., 2025], an open-source extension [Lindsey et al., 2025a], and a multilingual analysis [Harrasse et al., 2025]. This scarcity highlights the difficulty of training CLTs at scale.



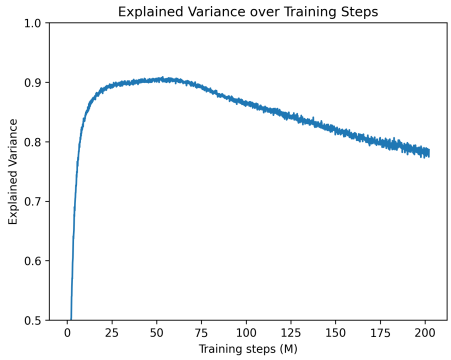
(a) L_0 vs steps



(b) L_0 coefficient vs steps



(c) Dead features vs steps



(d) Explained variance vs steps

Figure 15: Training dynamics of the cross-layer transcoder, showing sparsity, regularization strength, dead features, and reconstruction quality over training.

Table 3: Training configuration and hyperparameters for the GPT-2 cross-layer transcoders (CLTs).

Category	**Setting**
Model	GPT-2 (`HookedTransformer`)
Input dimension (d_{in})	768
Latent dimension (d_{latent})	24 576
Expansion factor	32
Context size	64
Batch size (tokens)	1 024
Precision	Mixed (FP32 / AMP)
Device	CUDA
Distributed training	DDP
Optimizer	Adam
Learning rate	2×10^{-4}
Adam β_1 / β_2	0.9 / 0.999
Learning rate warm-up	Cosine (1 000 steps)
Learning rate decay steps	1 874
Final LR scale	0.1
L_0 coefficient	2
Optimal L_0	3
L_0 warm-up	Linear (18 749 steps)
Dead feature penalty	10^{-5}
Dead feature window	250

I Attribution-Graph

Following Ameisen et al. [2025], we define the attribution score between feature n at layer ℓ and position k, and feature n' at layer ℓ' and position k', as

$$a_{\ell,k,n}^{\ell',k',n'} = \sum_{\ell \leq s \leq \ell'} f_{k,n}^{\ell \to s} \; J_{s,k}^{\ell',k'} \; g_{k',n'}^{\ell'}, \tag{11}$$

where $f_{k,n}^{\ell \to s}$ denotes the decoder vector associated with feature n projecting from layer ℓ to layer s, $J_{s,k}^{\ell',k'}$ is the Jacobian mapping the MLP output at (ℓ, k) to the MLP input at (ℓ', k'), and $g_{k',n'}^{\ell'}$ is the corresponding encoder feature at layer ℓ' and position k'. The sum in this equation reflects the cross-layer mapping of the cross-layer transcoder.

The Jacobian is computed during a modified forward pass in which all nonlinear operations, including normalization layers, attention mechanisms, and MLPs, are frozen using stop-gradient operations. The resulting attribution graph is pruned by retaining only those features that cumulatively explain 80% of the contribution to the final logit, and only those edges that account for 95% of the total edge-level effect. All attribution computations are performed using the `circuit-tracer` library Hanna et al. [2025]. For a complete description of the attribution graph computation and pruning, we refer the user to reading Ameisen et al. [2025].

For the visualization and the autointerp, we build our own pipeline. In Figure 16, we show a screenshot of the interface for the 'The opposite of "large" is "' attribution graph. The features are colored with respect to their corresponding clusters.

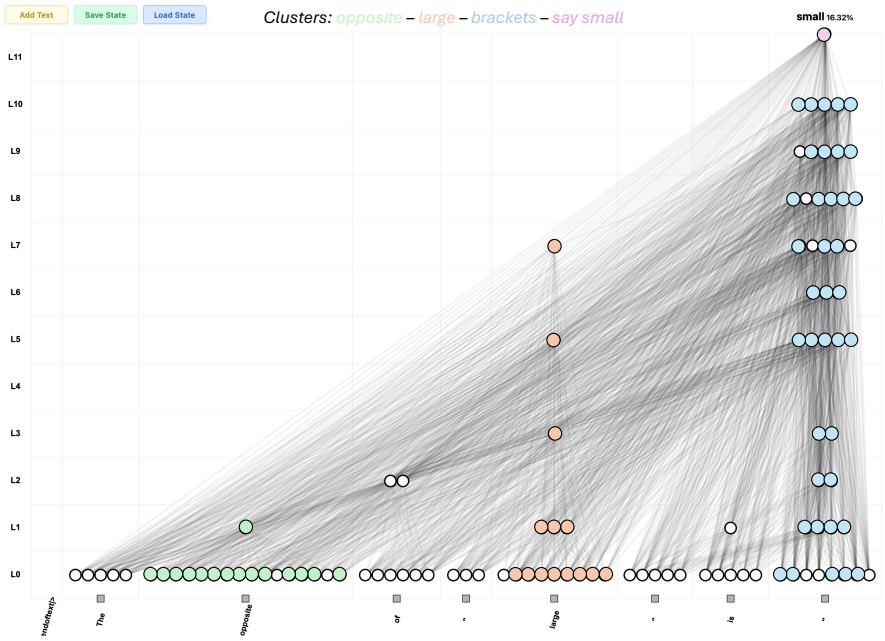


Figure 16: Circuit-tracing interface example for the 'The opposite of "large" is "' with GPT2-sparse.

J Graph: The opposite of "large" is "

We obtain a replacement score of 0.82, with 459 features identified before pruning and 82 features remaining after pruning. The majority of features in the resulting attribution graph fall into four dominant clusters:

- **Opposition cluster**: features associated with opposition and comparison, primarily localized at the token position corresponding to *opposite*.
- **Magnitude cluster**: features related to notions of size (e.g., *large*, *big*, *full*, *medium*), predominantly located in the residual stream at the *large* token position.
- **Bracket cluster**: features that activate on tokens enclosed in brackets.
- **Final-logit cluster**: mainly the final logit itself and a couple of features that activate before the token "small" or related terms.

In Figure 5, we focus on the attention heads that connect position 5 to position 8. This edge is important because it links the representation of the source cluster "large" to the representation of the target cluster "small", and therefore carries the information needed to solve the task. In the sparse attention setting, this computation is highly localized: only a small number of heads contribute substantially, and these heads attend almost exclusively from query position 8 to key position 5.

$\rightarrow$ *Query*

```
1. The feature activates on tokens related to large
   quantities. (pos 5)
2. The feature responds again to contexts
   expressing large magnitude. (pos 5)
3. The feature captures references to quantities or
   amounts. (pos 5)
4. The feature is associated with comparative
   constructions. (pos 3)
5. The feature activates on tokens indicating
   opposition or contrast. (pos 3)
```

Figure 17: Sentence-level description of the top-5 features activating the query for the attention head L8-H6 from Figure 5.

We then ask why these heads attend to position 5. Since an attention score is determined by the interaction between the query vector at the destination position and the key vector at the source position, we decompose this interaction into the features that contribute to each side. Concretely, we identify which features at position 5, after projection through the key matrix, increase attention from position 8, and which features at position 8, after projection through the query matrix, select for those keys. This gives a feature-level explanation of why the attention edge from position 5 to position 8 appears. The computation is analogous to the causal tracing framework of Hanna et al. [2025].

The resulting features are shown in Figure 17. We find that the key side is driven by features at position 5 that encode the relevant source-cluster information, including features related to the “large” and “opposite” clusters. This gives the following mechanistic interpretation. The model identifies the relevant opposite-cluster relation at position 5. This relation activates a small set of sparse attention heads, which route information from position 5 to position 8. The value/output pathway of these heads then linearly transforms the representation of the “large” cluster into the representation of the “small” cluster. In this sense, the attention heads do not merely correlate with the solution; they implement the routing step that allows the model to map one cluster to its opposite.

Feature Descriptions. In the boxes below, we present the top activations of representative feature sets for each cluster.

Feature 1117 "Opposite" cluster

in Washington has now adopted the wider measure of student debt outstanding. This new
the situation in Syria, Iran and the wider region. "The
recharged by the wider dense forests of Sanjay Van and its overflow drained
public, with interesting accounts of Oswald's demeanor at this significant moment
has a slightly wider range. Specifically, the Atom-powered NANO
56 becoming part of the wider Seven Years' War in which Britain and France

Feature 1337 "Opposite" cluster

opposite, piece of Mexico's cultural identity. I made the hour
opposite shows, or something bigger, "where there's villains
opposite sides of Mars in 2004 and used their instruments to discover geologic evidence
opposite, but not anymore. Now everything he says to me is some kind
opposite direction, and had little trouble finding space at the campsites.
always seem to be just the opposite.
show a growing trend to cast "no" votes , opposing how much salary
and the occupation of the opposing forces was generally limited to mutual observation.
work hand in hand for the purpose of opposing all movements of the thinking part
the defense's inability to stop opposing run games. The Bills have
ing opposing quarterbacks. The Seahawks not only had depth, they were versatile.
to win more hand battles particularly when the opposing tackle neutralizes his initial

Feature 901 "Large" cluster

Let's be honest: When someone advocates for large-scale Muslim
robot provides a tragicomic reminder of why RWD needs to consider large as
what kind of social safety nets should be in place to protect people from large
advocates to limit the power of large, established corporations, analysts say.
of large law firms is that they are so great that the only reason anyone
that by scaling up tests, the method would be conducive for use on larger

Feature 933 "Large" cluster

people healthy and anticipating health issues before they become a problem . Big Data is
Big brown bucks with funny accents." Judy flinched at
BIG UP UBUNTU: Ubuntu releases are named after industry.<|endoftext|>
BIG LEAGUE: Barron's Says The
they need to submit their content in the same way . Big enough
apps and offering alternatives routes . Big data and optical fiber

Feature 1004 "Large" cluster

guide said was ? full of drinking saloons, dime museums, small
would have 2 mana sources next turn (unless his hand was full of fast
's house, it 's full of adventure itself.?
statement that all German Catholics had a right to full transparency"
glimpsing a lobby full of construction debris. The front hallway was full of
Jokubas had recently been reading a newspaper article which was full of

Feature 412 "Brackets" cluster

group answered either "very" or "somewhat" attached – except
some work colleagues. Wilcox said she found it 'highly unlikely
very rare , "very likely ," "high risk," she says.
ulent. Pentagon spokesman Peter Cook said the sample was "
on of PopMatters called the album "brilliant" and said
arlene Lowe, described him as being "one of my biggest supporters".

Feature 518 "Brackets" cluster

Kerry said Washington and Hanoi will "continue to have differences in opinions
the United States will "take care of it." He told reporters after the
legislation would "provide new enforcement tools for protecting our citizens and will help
Gary Ross, said in a statement that the Air Force is currently "short
said, and Syrian President Bashar al-Assad would "have to go".
introduces politics into consumer policies," said Palmor, adding that it would "

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: All the claims stated in the abstract are backed up in the experiments section.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations in a dedicated subsection at the end.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not include theoretical results.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All the information need to reproduce the main experimental results are included in the appendices as well as the attached code.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The datasets used in this work are all publicly available. We attach the code for training sparse transformers with the submission in for the review process. The code will be made publicly available upon acceptance.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All the experiment details are included in the main text and the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Error bars are included where appropriate.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The compute overhead is described in appendix C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.

- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors have read and adhere to the CoE.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss broader impacts in a dedicated subsection at the end.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets are cited appropriately.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing no research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.