EFFICIENT TRAINING OF SPARSE AUTOENCODERS FOR LARGE LANGUAGE MODELS VIA LAYER GROUPS

Anonymous authors
Paper under double-blind review

ABSTRACT

Sparse Autoencoders (SAEs) have recently been employed as an unsupervised approach for understanding the inner workings of Large Language Models (LLMs). They reconstruct the model's activations with a sparse linear combination of interpretable features. However, training SAEs is computationally intensive, especially as models grow in size and complexity. To address this challenge, we propose a novel training strategy that reduces the number of trained SAEs from one per layer to one for a given group of contiguous layers. Our experimental results on Pythia 160M highlight a speedup of up to 3x without compromising the reconstruction quality and performance on downstream tasks. Therefore, layer clustering presents an efficient approach to train SAEs in modern LLMs.

1 INTRODUCTION

With the significant adoption of Large Language Model (LLM)s in world applications, understanding their inner workings has gained paramount importance. A key challenge in LLMs interpretability is the polysemanticity of neurons in models' activations, lacking a clear and unique meaning (Olah et al., 2020). Recently, SAEs (Huben et al., 2024; Bricken et al., 2023) have shown great promise to tackle this problem by decomposing the model's activations into a sparse combination of human-interpretable features.

The use of SAEs as an interpretability tool is motivated by two key reasons: the first is the substantial empirical evidence supporting the *Linear Representation Hypothesis (LRH)*, or that LLMs exhibit interpretable linear directions in their activation space (Mikolov et al., 2013; Nanda et al., 2023; Park et al., 2023); the second is the *Superposition Hypothesis (SH)* (Elhage et al., 2022), which supposes that, by leveraging sparsity, neural networks represent more features than they have neurons. Under this hypothesis, we can consider a trained neural network as a compressed simulation of a larger disentangled model, where every neuron corresponds to a single feature. To overcome superposition, SAEs leverage the LRH to decompose model activations into a sparse linear combination of interpretable features.

However, training SAEs is computationally expensive and will become even more costly as the model size and parameter counts grow. Indeed, one Sparse Autoencoder (SAE) is typically learned for a given component at every layer in a LLM. Moreover, the number of features usually equals the model activation dimension multiplied by a positive integer, called the expansion factor. For example, a single SAE trained on the Llama-3.1 8B model activations with an expansion factor of 32 has roughly $4096^2 \times 32 \times 2 \approx 1.073$ billion parameters, resulting in a total of more than 32 billions parameters when training one for each of the 32 layers. Additionally, every SAE is trained on a fixed number of tokens T, typically in the billions, leading to a cumulative training requirement of $L \times T$ tokens, where L is the total number of layers—for the Llama-3.1 8B model, this amounts to 32 billion tokens. This high computational demand not only increases training time but also requires substantial hardware resources and energy consumption, making the approach increasingly impractical as models scale.

In this work, we reduce the computational overhead of training a separate SAE for each layer of a target LLM by learning a single SAE for groups of related and contiguous layers, thus reducing

---

[*] Equal contribution

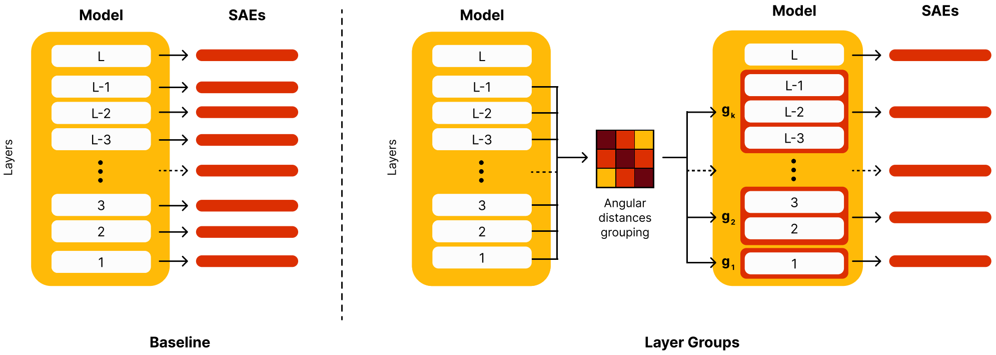


Figure 1: The illustration of our method. While standard training of SAEs requires training one per layer, our method first clusters layers by angular similarity and then trains a single SAE for each group.

the number of trained SAEs from L to k, where k is the number of groups. Furthermore, to ensure computational efficiency remains consistent, we fix the total number of training tokens T per group and allocate T/l tokens to each layer within the group, where l is the number of layers in that group. Thus, our approach reduces the total number of training tokens from LT to kT. This approach, which we term **Group-SAE**, is inspired by the observation that neural network layers often group together in learning task-specific representations (Szegedy et al., 2014; Zeiler & Fergus, 2014; Jawahar et al., 2019): shallow layers typically capture low-level features, while deeper layers learn high-level abstractions. Additionally, adjacent layers in LLMs tend to encode redundant information, as evidenced by the similarity in the angular distance of their outputs (Gromov et al., 2024). To quantify the relatedness between layers, we measure the angular distance of the residual stream after the MLP's contribution across neighboring layers.

Denoting with L the number of layers of the target model and k the number of groups we want to cluster the layers into, our approach obtains at least a $\frac{(L-1)}{k}$ times speedup[1] without sacrificing neither reconstruction quality nor downstream performance.

Our **contributions** can be summarized as follows:

- We demonstrate that a single Group-SAE can effectively reconstruct the activations of an entire cluster as a sparse linear combination of interpretable features. This approach reduces the number of SAEs required to be trained for a given model by a factor of k.
- We demonstrate the practical effectiveness of our method by training SAEs on the Pythia-160M model for different values of k, and we obtain a $\frac{L-1}{k}$ times speedup at the cost of a minor deterioration in performance.
- We extensively analyze our method on reconstruction performance, downstream tasks, and human interpretability for different values of k.

2 RELATED WORK

2.1 THE LINEAR REPRESENTATION AND SUPERPOSITION HYPOTHESES

Supported by substantial evidence, from the seminal Mikolov et al. (2013) vector arithmetic to the more recent work of Nanda et al. (2023) and Park et al. (2023) on LLMs, the Linear Representation Hypothesis (LRH) supposes that neural networks have interpretable linear directions in their acti-

---

[1] We do not consider the last transformer layer, as different from every other layer w.r.t. the angular distance defined in Section 3.1.

vations space. However, neuron polysemanticity remains an essential challenge in neural network interpretability (Olah et al., 2020).

Recently, Bricken et al. (2023) explored this issue by relating the Superposition Hypothesis (SH) to the decomposition ideally found by a SAE. According to the SH, neural networks utilize n-dimensional activations to encode $m \gg n$ features by leveraging their sparsity and relative importance. As a result, we can write the activations $\mathbf{x}^j$ in a model as

$$\mathbf{x}^j \approx \mathbf{b} + \sum_{i}^{m} \boldsymbol{f}_i(\mathbf{x}^j)\mathbf{d}_i \tag{1}$$

where $\mathbf{x}^j \in \mathbf{R}^n$ is the activation vector for an example j, $\boldsymbol{f} \in \mathbb{R}^m$ is a sparse feature vector, $\boldsymbol{f}_i(\mathbf{x}^j)$ is the activation of the i-th feature, $\mathbf{d}_i \in \mathbb{R}^n$ is a unit vector in the activation space and $\mathbf{b} \in \mathbb{R}^n$ is a bias term.

2.2 SPARSE AUTOENCODERS

Sparse Autoencoders have gained popularity in LLM interpretability due to their ability to counteract superposition and decompose neuron activations into interpretable features (Bricken et al., 2023; Huben et al., 2024). Given an input activation $\mathbf{x} \in \mathbb{R}^{d_{\text{model}}}$, a SAE reconstructs it as a sparse linear combination of $d_{\text{sae}} \gg d_{\text{model}}$ features, denoted as $\mathbf{v}_i \in \mathbb{R}^{d_{\text{model}}}$, where d_{sae} is set as $d_{\text{sae}} = c \cdot d_{\text{model}}$, where $c \in \{2^n | n \in \mathbb{N}_+\}$. The reconstruction follows the form:

$$(\hat{\mathbf{x}} \circ \mathbf{f})(\mathbf{x}) = \mathbf{W}_d\mathbf{f}(\mathbf{x}) + \mathbf{b}_d \tag{2}$$

Here, the rows of $\mathbf{W}_d \in \mathbb{R}^{d_{\text{model}} \times d_{\text{sae}}}$ represent the features $\mathbf{v}_i$, $\mathbf{b}_d \in \mathbb{R}^{d_{\text{model}}}$ is the decoder's bias term, and $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^{d_{\text{sae}}}$ represents the sparse features activations. The feature activations are computed as

$$\mathbf{f}(\mathbf{x}) = \sigma(\mathbf{W}_e(\mathbf{x} - \mathbf{b}_d) + \mathbf{b}_e) \tag{3}$$

where $\mathbf{W}_e \in \mathbb{R}^{d_{\text{sae}} \times d_{\text{model}}}$ and $\mathbf{b}_e \in \mathbb{R}^{d_{\text{sae}}}$ are the encoder's matrix and bias term respectively, and σ is an activation function, typically $\text{ReLU}(\mathbf{x}) = \max(0, \mathbf{x})^2$. The training of a SAE involves minimizing the following loss function:

$$\mathcal{L}_{\text{sae}} = \|\mathbf{x} - \hat{\mathbf{x}}(\mathbf{f}(\mathbf{x}))\|_2^2 + \lambda \text{S}(\mathbf{f}(\mathbf{x})) \tag{4}$$

where the first term in Equation 4 represents the reconstruction error, while the second term S is the regularization on the activations $\mathbf{f}(\mathbf{x})$ to encourage sparsity, which is typically equal to the ℓ_1 norm of the features $\text{f}(\mathbf{x})$, i.e. $\text{S}(\text{f}(\mathbf{x})) = \|\text{f}(\mathbf{x})\|_1$ Bricken et al. (2023). Other definitions of sparsity are admissible: Rajamanoharan et al. (2024) set S equal to the ℓ_0 norm of the features, i.e. $\text{S}(\text{f}(\mathbf{x})) = \|\text{f}(\mathbf{x})\|_0 = \sum_{i=0}^{d_{\text{sae}}} \mathbb{I}[\boldsymbol{f}_i(\mathbf{x}) \neq 0]$, which is the one adopted in this work along with the JumpReLU activation function, while Gao et al. (2024) imposes sparsity by selecting the Top-K activating features from $\text{f}(\mathbf{x})$.

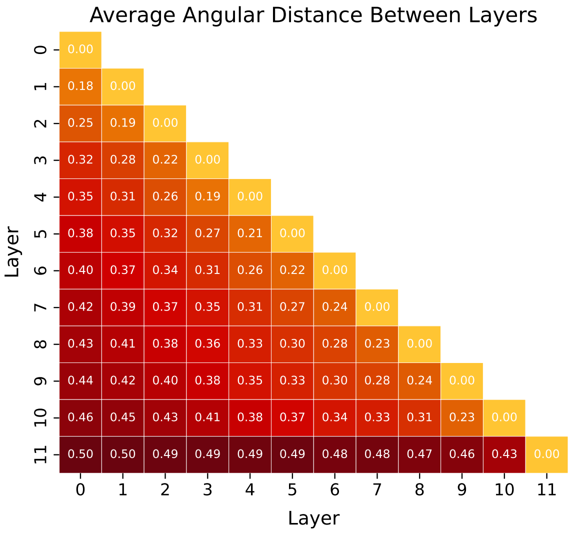
2.3 SAEs EVALUATION

SAE evaluation in the context of LLMs presents a significant challenge. While standard unsupervised metrics such as L_2 (reconstruction) loss and L_0 sparsity are widely adopted to measure SAE performance (Gao et al., 2024; Lieberum et al., 2024), they fall short of assessing two key aspects: causal importance and interpretability.

Recent approaches, including auto-interpretability (Bricken et al., 2023; Huben et al., 2024; Bills et al., 2023) and ground-truth comparisons (Sharkey et al., 2023), aim to provide a more holistic evaluation. These methods focus on the causal relevance of features (Marks et al., 2024) and evaluate SAEs in downstream tasks. Makelov et al. (2024), for instance, proposed a framework for the Indirect Object Identification (IOI) task, emphasizing three aspects: the sufficiency and necessity of

---

[2] Other activation functions are admissible, e.g. Top-K (Gao et al., 2024) or JumpReLU (Rajamanoharan et al., 2024), with JumpReLU being the one employed in this work.



Layer	0	1	2	3	4	5	6	7	8	9	10	11
0	0.00											
1	0.18	0.00										
2	0.25	0.19	0.00									
3	0.32	0.28	0.22	0.00								
4	0.35	0.31	0.26	0.19	0.00							
5	0.38	0.35	0.32	0.27	0.21	0.00						
6	0.40	0.37	0.34	0.31	0.26	0.22	0.00					
7	0.42	0.39	0.37	0.35	0.31	0.27	0.24	0.00				
8	0.43	0.41	0.38	0.36	0.33	0.30	0.28	0.23	0.00			
9	0.44	0.42	0.40	0.38	0.35	0.33	0.30	0.28	0.24	0.00		
10	0.46	0.45	0.43	0.41	0.38	0.37	0.34	0.33	0.31	0.23	0.00	
11	0.50	0.50	0.49	0.49	0.49	0.49	0.48	0.48	0.47	0.46	0.43	0.00

Figure 2: Average angular distance between all layers of the Pythia-160M model, as defined in Equation 5. The angular distances are computed over 5M tokens from the training dataset. The angular distances are bounded in $[0, 1]$, where an angular distance equal to 0 means equal activations, 0.5 means activations are perpendicular and an angular distance of 1 means that the activations point in opposite directions.

reconstructions, sparse feature steering (Templeton et al., 2024), and the interpretability of features in causal terms.

Karvonen et al. (2024) further contributed by developing specialized metrics for board game language models. Using structured games like chess and Othello, they introduced supervised metrics, such as board reconstruction accuracy and coverage of predefined state properties, offering a more direct assessment of SAEs' ability to capture semantically meaningful and causally relevant features.

2.4 IMPROVING SAES TRAINING

As SAEs gain popularity for LLMs interpretability and are increasingly applied to state-of-the-art models (Lieberum et al., 2024), the need for more efficient training techniques has become evident. To address this, (Gao et al., 2024) explored the scaling laws of Autoencoders to identify the optimal combination of size and sparsity.

Recent work also explored using transfer learning to improve SAE training. For example, Kissane et al. (2024) and Lieberum et al. (2024) demonstrated the transferability of SAE weights between base and instruction-tuned versions of Gemma-1 (Team et al., 2024a) and Gemma-2 (Team et al., 2024b), respectively. On the other hand, Ghilardi et al. (2024) shows that transfer also occurs among layers of a single model, both in forward and backward directions.

3 EXPERIMENTAL SETUP

3.1 LAYER GROUPS

For a model with L layers, the number of possible combinations of k groups of adjacent layers that can be tested is given by $\binom{L-1}{k-1}$. With this number growing with model depth, we employed an agglomerative grouping strategy based on the angular distances between layers to reduce it drastically. In particular, we compute the mean angular distances, as specified in Gromov et al. (2024), over 5M tokens from our training set:

$$d_{\text{angular}}\left(\mathbf{x}_{\text{post}}^{p}, \mathbf{x}_{\text{post}}^{q}\right) = \frac{1}{\pi} \arccos\left(\frac{\mathbf{x}_{\text{post}}^{p} \cdot \mathbf{x}_{\text{post}}^{q}}{\left\|\mathbf{x}_{\text{post}}^{p}\right\|_2 \left\|\mathbf{x}_{\text{post}}^{q}\right\|_2}\right) \tag{5}$$

for every $p, q \in \{1, ..., L\}$, where $\mathbf{x}_{\text{post}}^{l}$ are the l-th residual stream activations after the MLP's contribution[3]. From Figure8, it can be noted how the last layer is different from every other layer in terms of angular distance. For this reason, we have decided to exclude it from the grouping procedure.

We adopted a bottom-up hierarchical clustering strategy with a complete linkage (Nielsen, 2016) that aggregates layers based on their angular distances. Specifically, at every step of the process, the two groups with minimal group-distance[4] are merged; we repeat the process until the predefined number of groups k is reached. This approach prevents the formation of groups as long chains of layers and ensures that the maximum distance within each group remains minimal. In this work, we create groups considering all layers except the last, and we choose k varying from 1 (a single cluster with all layers) to 5. Layer groups can be found in Appendix B. To show how our method scales to larger models, we also compute distances and groups of Gemma-2 2b and 9b Team et al. (2024b). Detailed results are provided in Appendix B.

3.2 SAES TRAINING AND EVALUATION

We denote SAE_i as the baseline SAE trained to reconstruct the activations of layer i. For every $1 \leq j \leq k \leq 5$ with $j, k \in \mathbb{N}$, we define SAE_{j_k} as the SAE trained to reconstruct the activations for all layers in the j-th group of a partition consisting of k groups. Furthermore, let $[j_k]$ represent the set of layers belonging to the j-th group within this partition. Each SAE_{j_k} is specifically trained to reconstruct the activations of each layer $s \in [j_k]$ individually, using the formulations described in equations 3 and 2.

To ensure both computational efficiency and fair comparison with baselines, we allocate a fixed total of 1 billion tokens for training both every SAE_i and SAE_{j_k}. In particular, for SAE_{j_k} these tokens are evenly distributed across the layers in $[j_k]$, assigning $1\text{B}/|[j_k]|$ tokens to each layer, with a randomly chosen layer's activation per token. This approach ensures balanced training across layers within the group while keeping the total token budget consistent with the baseline configurations, thus reducing the total number of training tokens from LT to kT.

For evaluation, we train a SAE_{j_k} for every combination of k and j and compare its performance with the corresponding baseline SAE_s, where $s \in [j_k]$. Section 4 presents the results of these comparisons based on standard reconstruction and sparsity metrics. Furthermore, in Section 5, we show how our approach performs on popular downstream tasks (e.g., (Marks et al., 2024; Hanna et al., 2023; Wang et al., 2023)), while Section 6 reports the human interpretability scores.

3.3 DATASET AND HYPERPARAMETERS

We train SAEs on the residual stream of the Pythia-160M model (Biderman et al., 2023) for 1B tokens, focusing on the residual stream activations after the MLP contribution. The chosen dataset is a 2B pre-tokenized version[5] of the Pile dataset (Gao et al., 2020) with a context size of 1024. We set the expansion factor $c = 8$, $\lambda = 3e\text{-}4$ in Equation 4, learning rate equal to $7e\text{-}4$, and a batch size of 4096 samples. Following Bricken et al. (2023), we constrain the decoder columns to have unit norm and do not tie the encoder and decoder weights. Additionally, we have normalized the activations so that they have mean squared ℓ_2 norm of one during SAE training, as specified in Rajamanoharan et al. (2024), estimating the scaling factor over 2M tokens of our train set.

We use the JumpReLU activation function as specified in Rajamanoharan et al. (2024), and defined as $\text{JumpReLU}_\theta(z) = z \cdot \text{H}(z - \theta)$, with θ being a threshold learned during training and H is the Heaviside function. We fixed the hyperparameters for all the experiments conducted in this work. All hyperparameters can be found in Table 2.

---

[3] In a Transformer model (Vaswani et al., 2017), the residual stream activations are computed as follows: given the activations $\mathbf{x}_{\text{pre}}^{l-1}$ from layer $l-1$, the pre-layer activations for layer l, $\mathbf{x}_{\text{pre}}^{l} = \mathbf{x}_{\text{post}}^{l-1}$, are calculated as $\mathbf{x}_{\text{post}}^{l-1} = \mathbf{x}_{\text{mid}}^{l-1} + \text{MLP}(\mathbf{x}_{\text{mid}}^{l-1})$, where $\mathbf{x}_{\text{mid}}^{l-1} = \mathbf{x}_{\text{pre}}^{l-1} + \text{Attn}(\mathbf{x}_{\text{pre}}^{l-1})$. All SAEs are trained specifically on the residual stream after the MLP contribution, i.e., $\mathbf{x}_{\text{post}}^{l}$, for every layer l.

[4] The complete linkage clustering strategy defines the group-distance between two groups X and Y as $D(X, Y) = \max_{\mathbf{x} \in X, \mathbf{y} \in Y} d_{\text{angular}}(\mathbf{x}, \mathbf{y})$

[5] `https://huggingface.co/datasets/NeelNanda/pile-small-tokenized-2b`

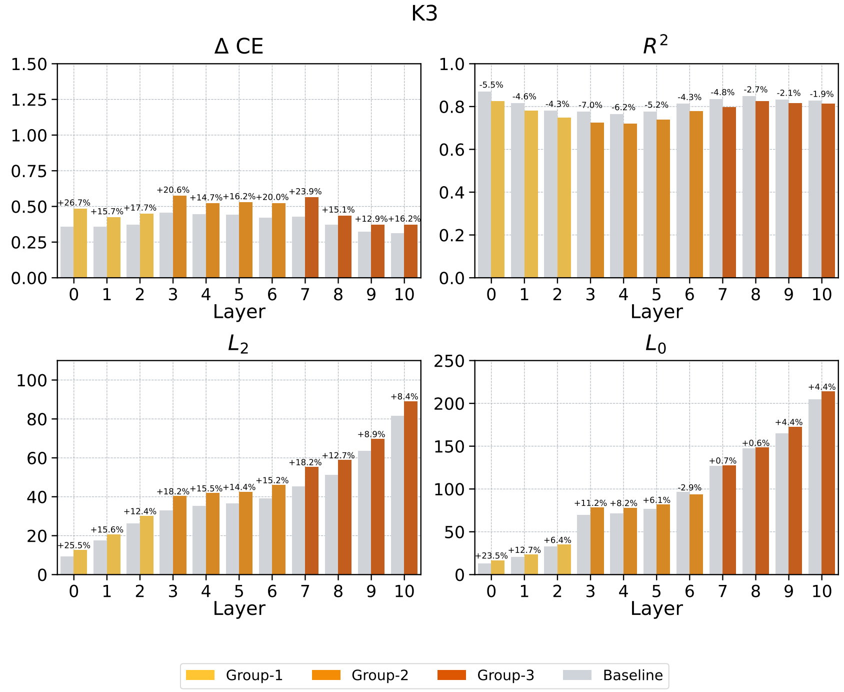


Figure 3: ΔCE, R^2, L_2 and L_0 for $k = 3$ number of groups. Note that, for ΔCE and L_2, the lower is the score, the better it is, while the contrary is true for R^2. For L_0 there are no indications on a preferred value.

4 RECONSTRUCTION EVALUATION

To assess the quality of the reconstruction of SAEs trained with our grouping strategy, we report three standard reconstruction metrics and one sparsity metric. In particular, the Delta Cross-Entropy, defined as $\text{CE}(\hat{\mathbf{x}} \circ \mathbf{f}) - \text{CE}(M)$, measures the difference in Cross-Entropy loss (CE) between the output with SAE reconstructed activations $(\hat{\mathbf{x}} \circ \mathbf{f})$ and the model's output (M).

The L_2 loss is the first term of Equation 4 and measures the reconstruction error made by the SAE. The R^2 score, defined as $1 - \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 / \|\mathbf{x} - \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[\mathbf{x}]\|_2^2$, measures the fraction of explained variance of the input recovered by the SAE.

Finally, the L_0 sparsity loss, defined as $\sum_{j=1}^{d_{\text{sae}}} \mathbb{I}[\boldsymbol{f}_j \neq 0]$, represents the number of non-zero SAE features used to compute the reconstruction. For each metric, we compute the average over 1M examples from the test dataset.

Figure 3 shows the reconstruction and sparsity metrics for each layer of the grouping with $k = 3$. It can be noted that training a single SAEs on the activations from multiple close layers doesn't dramatically affect the reconstruction, even when all layers are clustered in a small number of groups. Metrics for all other values of k can be found in Appendix C.

These results demonstrate that a single SAE can effectively reconstruct activations across multiple layers. Furthermore, the comparable performance between SAE_{j_k} and the individual layer-specific SAE_i indicates that post-residual stream activations from adjacent layers share a common set of underlying features. This hypothesis is further supported by directly comparing the directions learned by SAE_i and SAE_{j_k} using the Mean Maximum Cosine Similarity (MMCS) score, as shown in Appendix D.

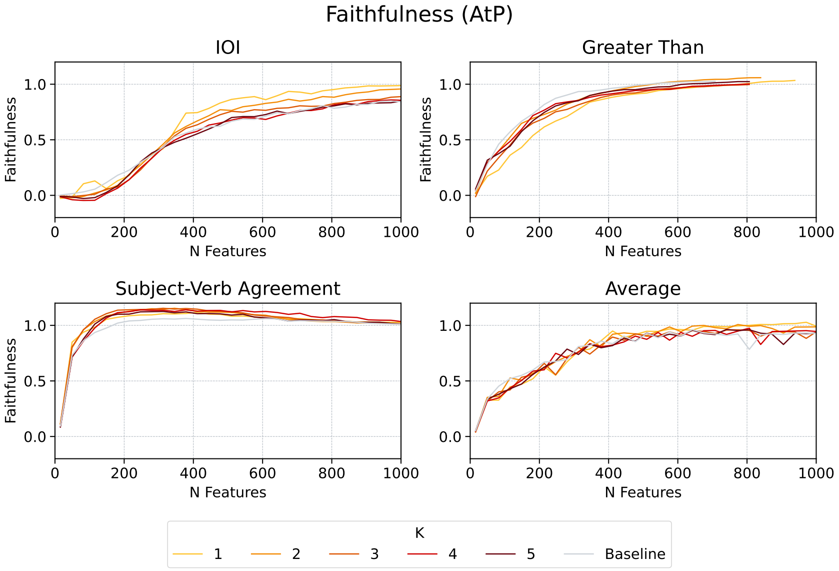


Figure 4: Average faithfulness (Marks et al., 2024) for every downstream task (IOI, Greater Than, Subject-Verb Agreement) with IE computed with AtP (Equation 8). The "Baseline" average is computed considering the performance obtained by SAE$_i$, $\forall i = 0, ..., 10$. The "Average" plot depicts the average over the three downstream tasks.

5 DOWNSTREAM TASKS EVALUATION

While achieving good reconstruction metrics is crucial for a trained SAE, it is insufficient for a comprehensive evaluation of its performance. For instance, unsupervised metrics alone cannot determine whether the identified features capture causally influential directions for the model. To address this, following Marks et al. (2024), we applied SAEs to three well-known tasks: Indirect Object Identification (IOI), Greater Than, and Subject-Verb Agreement.

Each task can be represented as a set of counterfactual prompts paired with their respective answers, formally denoted as $\mathcal{T} : \{x_{\text{clean}}, x_{\text{corrupted}}, a_{\text{clean}}, a_{\text{corrupted}}\}$. Counterfactual prompts are similar to clean ones but contain slight modifications that result in a different predicted answer.

Ideally, a robust SAE should be able to recover the model's performance on a task when reconstructing its activations. Furthermore, we expect the SAE to rely on a small subset of task-relevant features to complete the task. To assess this, we filtered the features to include only the most important ones, where importance is defined as the *indirect effect* (IE) (Pearl, 2022) of the feature on task performance, measured by a real-valued metric $m : \mathbb{R}^{d_{\text{vocab}}} \to \mathbb{R}$. Specifically,

$$\text{IE}(m; \boldsymbol{f}_i; x_{\text{clean}}, x_{\text{corrupted}}) = m(x_{\text{clean}} | \text{do}(\boldsymbol{f}_i = \boldsymbol{f}_{i;\text{corrupted}})) - m(x_{\text{clean}}) \tag{6}$$

In this equation, $\boldsymbol{f}_{i;\text{corrupted}}$ represents the value of the i-th feature $\boldsymbol{f}_i$ during the computation of $m(x_{\text{corrupted}})$, and $m(x_{\text{clean}} | \text{do}(\boldsymbol{f}_i = \boldsymbol{f}_{i;\text{corrupted}}))$ refers to the value of m for x_{clean} under an intervention where the activation of feature $\boldsymbol{f}_i$ is set to $\boldsymbol{f}_{i;\text{corrupted}}$. Moreover, m is defined as the difference in logits between the clean and the corrupted answer.

Calculating these effects is computationally expensive, as it requires a forward pass for each feature. To mitigate this, we employed two approximate methods: Attribution Patching (AtP)(Nanda, 2023; Syed et al., 2024) and Integrated Gradients (IG)(Sundararajan et al., 2017). Appendix E provides a formal definition of both methods.

Following Marks et al. (2024), we used faithfulness and completeness metrics to evaluate the performance of the SAEs on the tasks. These metrics are defined as $\frac{m(C)-m(\emptyset)}{m(M)-m(\emptyset)}$, where $m(M)$ and $m(\emptyset)$ represent the metric average over $\mathcal{T}$, achieved by the model alone and with the mean-ablated

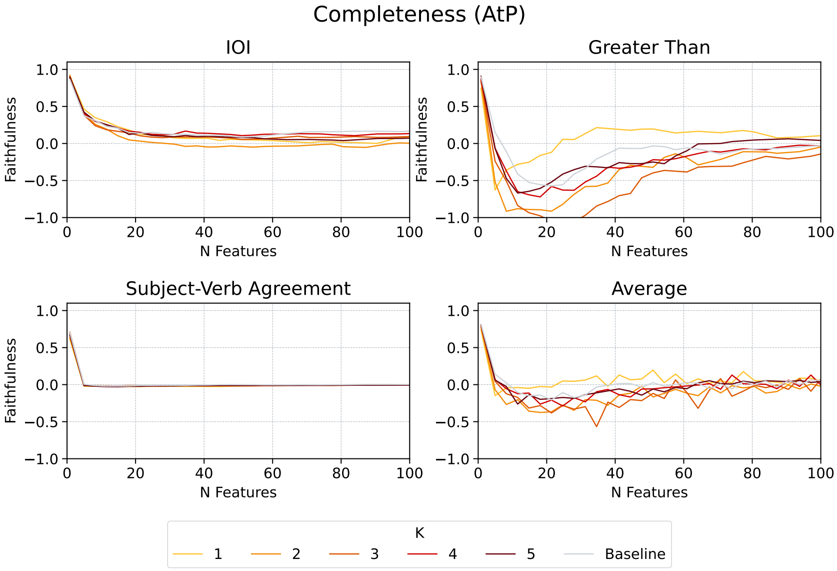


Figure 5: Average completeness for every downstream task (IOI, Greater Than, Subject-Verb Agreement) with IE computed with AtP (Equation 8). The "Baseline" average is computed considering the performance obtained by SAE$_i$, $\forall i = 0, ..., 10$. The "Average" plot depicts the average performance over the three downstream tasks.

SAE reconstructions, respectively. $m(C)$ is computed based on the task and either the *faithfulness* or *completeness* criteria: for faithfulness, it is the metric average when using only the important SAE features, while mean-ablating the others; on the contrary for completeness, it is calculated by mean-ablating the important features while keeping the others active.

Given a predefined number N of features to compute a circuit, we select the top-N features based on their indirect effect at each layer. Then, to calculate the Faithfulness and Completeness scores, we respectively retain or ablate these selected features while mean-ablating or retaining all the others. As in Marks et al. (2024), the computation of both scores incorporates the SAE error $\epsilon_{\mathbf{x}}$[6] into the reconstructions, as excluding it compromises the model's performance on the task. Furthermore, when using Group-SAEs, the selected features can vary across layers within a group, as each layer may require a distinct set of features to achieve optimal reconstruction, thus SAE$_{j_k}$ is used to independently and separately reconstruct the activations of every layer $s \in [j_k]$ with the *inderect effect* that is estimated for every layer separately.

These metrics allow us to evaluate two critical aspects of SAE quality: whether the SAE learned a set of features that is both **sufficient** and **necessary** to perform the task. Figures 4 and 5 display the faithfulness and completeness scores for all k groups.

All SAE$_{j_k}$ perform comparably to, or slightly better than, the baseline SAE$_i$ models on the faithfulness score (Figure 4) across all three downstream tasks evaluated, demonstrating the sufficiency of learned features. Remarkably, this performance is achieved using only the top 5% of the most important active features, which recover 75% of the baseline performance, on average. Even more remarkable are the performances of the single-group SAE$_{j_1}$ ($k = 1$), which closely follow the trend of both the baseline SAEs and the SAE$_{j_5}$ ($k = 5$). Moreover, we did not observe any substantial differences in performance for any of the tested values of k. The necessity of the features learned by both the baseline SAE$_i$ and SAE$_{j_k}$ is confirmed by the completeness scores depicted in Figure 5, with a severe drop in performance even with only the top 10 active features mean-ablated.

The results of SAE$_{j_k}$ on downstream tasks demonstrate that their learned features are both sufficient and necessary. Moreover, these findings confirm those in Section 4, i.e., that a single SAE can

---

[6] The SAE error ϵ is defined as $\epsilon_{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}}(\mathbf{f}(\mathbf{x}))$.

effectively reconstruct activations across multiple contiguous layers and learn a set of shared, general features that span adjacent layers.

6 HUMAN INTERPRETABILITY

In addition to achieving excellent reconstruction and downstream performance, SAEs must learn interpretable features. Following Ghilardi et al. (2024), we engaged human annotators to identify interpretable patterns in the feature annotations. Specifically, they attempted to provide clear definitions for each feature by examining its top and bottom logits attribution scores, as well as the top activating tokens. In total, we randomly sample 96 features for each SAE and store their activations on 1M tokens from the training dataset. To ensure learned features are not layer-specific, for Group-SAEs we keep the sampled features constant across the layers within the group. To evaluate the quality of these features, we defined the *Human Interpretability Score* as the ratio of features considered interpretable by the human annotators.

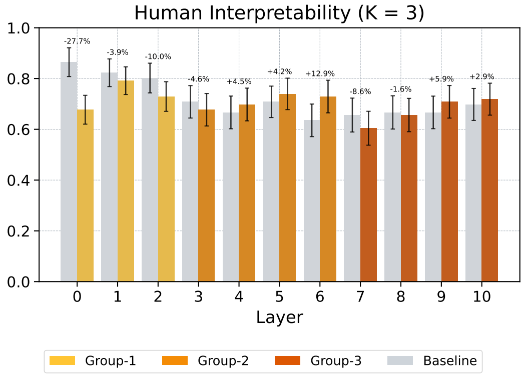


Figure 6: Human Interpretability Scores for $k = 3$. The differences in the interpretability scores of features learned by the SAE_{j_k} and the baseline SAE_i are not statistically significant different for all the layers except for the first. Error bars shows one standard deviations of the scores differences, modeled as a Binomial distribution (Wasserman, 2010).

Figure 6 presents the Human Interpretability Scores for all layers of $k = 3$. Scores for all other values of k can be found in Appendix F. According to human annotators, the interpretability of the features learned by SAE_{j_k} is comparable to that of the baseline SAE_i. Moreover, we found that when many layers are grouped together, e.g., when $k = 1$, SAE_{j_k} features are more polysemantic overall, probably due to increased interference. Nevertheless, some of them remain perfectly interpretable across all model layers, capturing critical directions in model computations.

7 CONCLUSION

This work introduces a novel approach to efficiently train Sparse Autoencoders (SAEs) for Large Language Models (LLMs) by clustering layers based on their angular distance and training a single SAE for each group. Through this method, we achieved up to a 3x speedup in training without compromising reconstruction quality or performance on downstream tasks. The results demonstrate that activations from adjacent layers in LLMs share common features, enabling effective reconstruction with fewer SAEs.

Our findings also show that the SAEs trained on grouped layers perform comparably to layer-specific SAEs in terms of reconstruction metrics, faithfulness, and completeness on various downstream tasks. Furthermore, human evaluations confirmed the interpretability of the features learned by our SAEs, underscoring their utility in disentangling neural activations.

The methodology proposed in this paper opens avenues for more scalable interpretability tools, facilitating deeper analysis of LLMs as they grow in size. Future work will focus on further optimizing the number of layer groups and scaling the approach to even larger models.

8 LIMITATIONS AND FUTURE WORKS

One limitation of our approach is the absence of a precise method for selecting the optimal number of layer groups (k). This choice is due to the lack of a clear *elbow rule* for identifying the correct number of groups.

Based on our results on Pythia-160m, Appendix B provides the Maximum Average Angular Distance (MAAD) within groups for Pythia-160m and larger models (Gemma-2 2b and 9b). This heuristic can guide the selection of the number of groups k for clustering layers in larger models, relying on the assumption that angular distance between activations from different layers is an important indicator of how the Group SAE will perform on them. While further investigation is needed to refine group configurations and assess scalability, these heuristics offer a practical reference for training Group-SAEs in larger models.

Additionally, we tested our method primarily on the Pythia-160M model, a relatively small LLM. While our findings demonstrate significant improvements in efficiency without sacrificing performance, the scalability of our approach to much larger models remains an open question. Future work could explore how the grouping strategy and training techniques generalize to models with billions of parameters, where the computational benefits would be even more pronounced.

Another important direction for future research involves understanding how Sparse Autoencoders (SAEs) handle the superposition hypothesis when encoding information from multiple layers. While our method effectively grouped layers and maintained high performance, how SAEs manage the potential overlap in feature representation across layers remains unclear. Investigating this aspect could lead to a more clear understanding of the trade-offs between sparsity and feature disentanglement in SAEs, and inform strategies for improving interpretability without compromising task performance.

In summary, while our work represents an efficient step forward in training SAEs for interpretability, extending this approach to larger models and exploring the handling of superposition will provide valuable insights for both practical applications and the theoretical understanding of sparse neural representations.

9 REPRODUCIBILITY STATEMENT

To support the replication of our empirical findings on training SAEs via layer groups and to enable further research on understanding their inner works, we plan to release all the code and SAEs used in this study upon acceptance.

REFERENCES

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.

Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. Language models can explain neurons in language models, 2023. URL `https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html`. Accessed: 2024-08-18.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022. URL `https://transformer-circuits.pub/2022/toy_model/index.html`.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders, 2024. URL `https://arxiv.org/abs/2406.04093`.

Davide Ghilardi, Federico Belotti, Marco Molinari, and Jaehyuk Lim. Accelerating sparse autoencoder training via layer-wise transfer learning in large language models. In *The 7th BlackboxNLP Workshop*, 2024. URL `https://openreview.net/forum?id=GI5j6OMTju`.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Daniel A Roberts. The unreasonable ineffectiveness of the deeper layers. *arXiv preprint arXiv:2403.17887*, 2024.

Michael Hanna, Ollie Liu, and Alexandre Variengien. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=p4PckNQR8k`.

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=F76bwRSLeK`.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3651–3657, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1356. URL `https://aclanthology.org/P19-1356`.

Adam Karvonen, Benjamin Wright, Can Rager, Rico Angell, Jannik Brinkmann, Logan Riggs Smith, Claudio Mayrink Verdun, David Bau, and Samuel Marks. Measuring progress in dictionary learning for language model interpretability with board game models. In *ICML 2024 Workshop on Mechanistic Interpretability*, 2024. URL `https://openreview.net/forum?id=qzsDKwGJyB`.

Connor Kissane, Ryan Krzyzanowski, Andrew Conmy, and Neel Nanda. SAEs (usually) transfer between base and chat models. `https://www.alignmentforum.org/posts/fmwk6qxrpW8d4jvbd/saes-usually-transfer-between-base-and-chat-models`, July 2024. AI Alignment Forum.

Tom Lieberum, Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Nicolas Sonnerat, Vikrant Varma, János Kramár, Anca Dragan, Rohin Shah, and Neel Nanda. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. *arXiv preprint arXiv:2408.05147*, 2024.

Aleksandar Makelov, Georg Lange, and Neel Nanda. Towards principled evaluations of sparse autoencoders for interpretability and control. In *ICLR 2024 Workshop on Secure and Trustworthy Large Language Models*, 2024. URL `https://openreview.net/forum?id=MHIX9H8aYF`.

Samuel Marks, Can Rager, Eric J Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. *arXiv preprint arXiv:2403.19647*, 2024.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In Lucy Vanderwende, Hal Daumé III, and Katrin Kirchhoff (eds.), *Proceedings* *of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 746–751, Atlanta, Georgia, June 2013. Association for Computational Linguistics. URL `https://aclanthology.org/N13-1090`.

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

Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models. In Yonatan Belinkov, Sophie Hao, Jaap Jumelet, Najoung Kim, Arya McCarthy, and Hosein Mohebbi (eds.), *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 16–30, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.blackboxnlp-1.2. URL `https://aclanthology.org/2023.blackboxnlp-1.2`.

Frank Nielsen. *Hierarchical Clustering*, pp. 195–211. 02 2016. ISBN 978-3-319-21902-8. doi: 10.1007/978-3-319-21903-5_8.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001. https://distill.pub/2020/circuits/zoom-in.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. In *Causal Representation Learning Workshop at NeurIPS 2023*, 2023. URL `https://openreview.net/forum?id=T0PoOJg8cK`.

Judea Pearl. Direct and indirect effects. In *Probabilistic and causal inference: the works of Judea Pearl*, pp. 373–392. 2022.

Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders, 2024. URL `https://arxiv.org/abs/2407.14435`.

Lee Sharkey, Dan Braun, and Beren Millidge. Taking the temperature of transformer circuits, 2023. URL `https://www.alignmentforum.org/posts/z6QQJbtpkEAX3Aojj/interim-research-report-taking-features-out-of-superposition`. Accessed: 2024-08-18.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pp. 3319–3328. PMLR, 2017.

Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching outperforms automated circuit discovery. In *The 7th BlackboxNLP Workshop*, 2024. URL `https://openreview.net/forum?id=RysbaxAnc6`.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014. URL `https://arxiv.org/abs/1312.6199`.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikuła, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo

Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. Gemma: Open models based on gemini research and technology, 2024a. URL `https://arxiv.org/abs/2403.08295`.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozińska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucińska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Görner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nenshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Perrin, Sébastien M. R. Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomas Kocisky, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. Gemma 2: Improving open language models at a practical size, 2024b. URL `https://arxiv.org/abs/2408.00118`.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024. URL `https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In

The Eleventh International Conference on Learning Representations, 2023. URL `https://openreview.net/forum?id=NpsVSN6o4ul`.

Larry Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer Publishing Company, Incorporated, 2010. ISBN 1441923225.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pp. 818–833. Springer, 2014.

A HYPERPARAMETERS

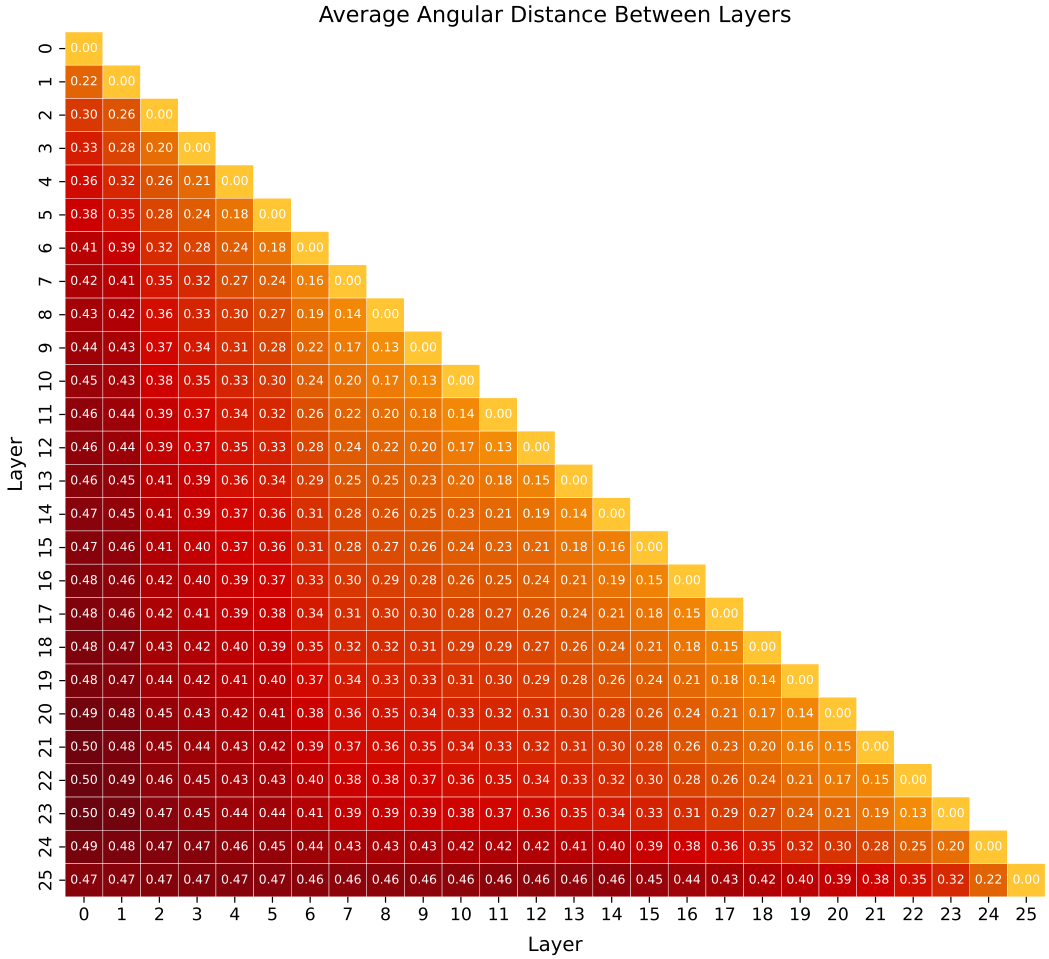
Table 1: Pythia-160M model specifics

Config	**Value**
Layers (L)	12
Model dimension (d_{model})	768
Heads (H)	12
Non-Embedding params	85,056,000
Equivalent models	GPT-Neo OPT-125M

Table 2: Training and fine-tuning hyperparameters

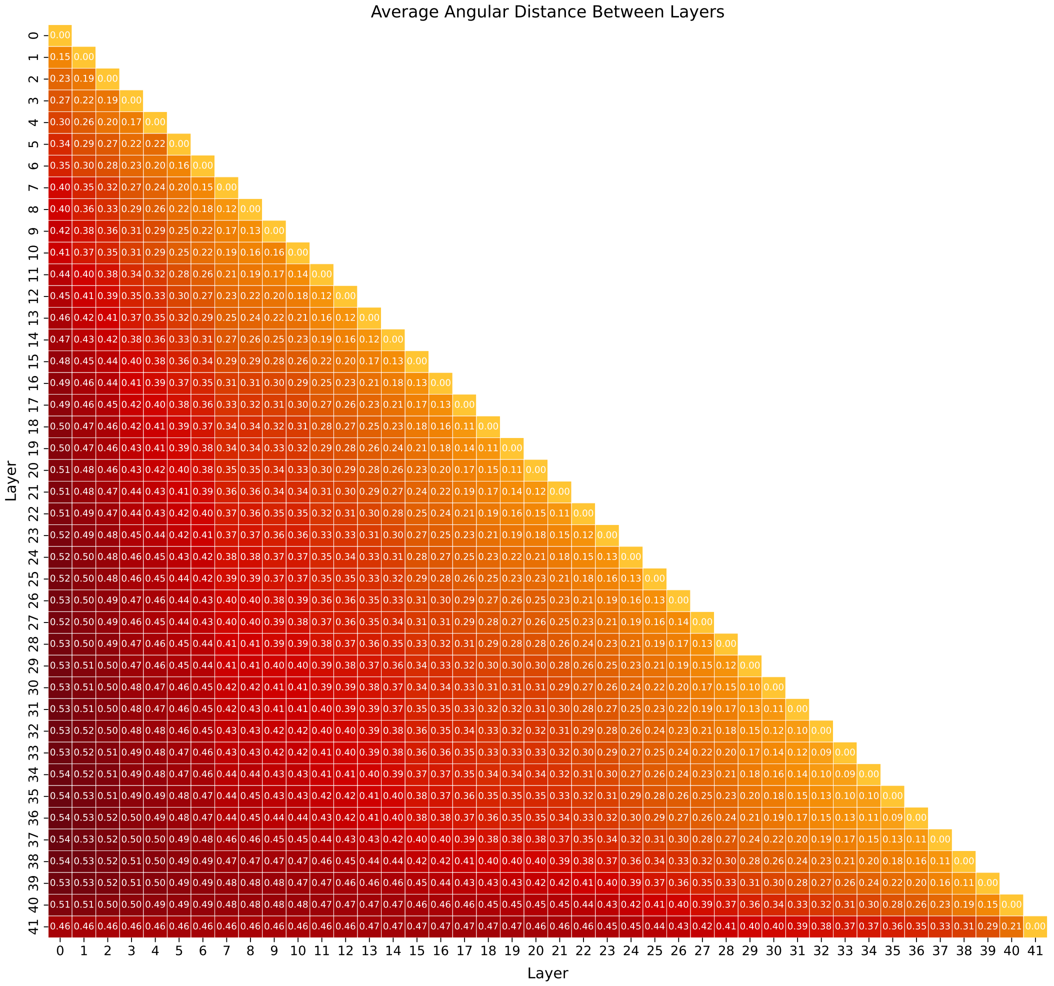
Hyperparameter	**Value**
c	8
λ	3e-4
Hook name	resid-post
Batch size	4096
Adam (β_1, β_2)	$(0, 0.999)$
Context size	1024
lr	7e-4
lr scheduler	constant
lr deacy steps	last 20% of the training steps
l1 warm-up steps	5% of the training steps
# tokens (Train)	1B
Checkpoint freq	200M
Decoder weights initialization	Zeroes
Activation function	JumpReLU
Decoder column normalization	Yes
Activation normalization	Mean squared ℓ_2 norm of one during SAE training
FP precision	32
Prepend BOS token	No

B ADDITIONAL ANGULAR DISTANCES AND LAYERS GROUPS



Layer	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0	0.00																									
1	0.22	0.00																								
2	0.30	0.26	0.00																							
3	0.33	0.28	0.20	0.00																						
4	0.36	0.32	0.26	0.21	0.00																					
5	0.38	0.35	0.28	0.24	0.18	0.00																				
6	0.41	0.39	0.32	0.28	0.24	0.18	0.00																			
7	0.42	0.41	0.35	0.32	0.27	0.24	0.16	0.00																		
8	0.43	0.42	0.36	0.33	0.30	0.27	0.19	0.14	0.00																	
9	0.44	0.43	0.37	0.34	0.31	0.28	0.22	0.17	0.13	0.00																
10	0.45	0.43	0.38	0.35	0.33	0.30	0.24	0.20	0.17	0.13	0.00															
11	0.46	0.44	0.39	0.37	0.34	0.32	0.26	0.22	0.20	0.18	0.14	0.00														
12	0.46	0.44	0.39	0.37	0.35	0.33	0.28	0.24	0.22	0.20	0.17	0.13	0.00													
13	0.46	0.45	0.41	0.39	0.36	0.34	0.29	0.25	0.25	0.23	0.20	0.18	0.15	0.00												
14	0.47	0.45	0.41	0.39	0.37	0.36	0.31	0.28	0.26	0.25	0.23	0.21	0.19	0.14	0.00											
15	0.47	0.46	0.41	0.40	0.37	0.36	0.31	0.28	0.27	0.26	0.24	0.23	0.21	0.18	0.16	0.00										
16	0.48	0.46	0.42	0.40	0.39	0.37	0.33	0.30	0.29	0.28	0.26	0.25	0.24	0.21	0.19	0.15	0.00									
17	0.48	0.46	0.42	0.41	0.39	0.38	0.34	0.31	0.30	0.30	0.28	0.27	0.26	0.24	0.21	0.18	0.15	0.00								
18	0.48	0.47	0.43	0.42	0.40	0.39	0.35	0.32	0.32	0.31	0.29	0.29	0.27	0.26	0.24	0.21	0.18	0.15	0.00							
19	0.48	0.47	0.44	0.42	0.41	0.40	0.37	0.34	0.33	0.33	0.31	0.30	0.29	0.28	0.26	0.24	0.21	0.18	0.14	0.00						
20	0.49	0.48	0.45	0.43	0.42	0.41	0.38	0.36	0.35	0.34	0.33	0.32	0.31	0.30	0.28	0.26	0.24	0.21	0.17	0.14	0.00					
21	0.50	0.48	0.45	0.44	0.43	0.42	0.39	0.37	0.36	0.35	0.34	0.33	0.32	0.31	0.30	0.28	0.26	0.23	0.20	0.16	0.15	0.00				
22	0.50	0.49	0.46	0.45	0.43	0.43	0.40	0.38	0.38	0.37	0.36	0.35	0.34	0.33	0.32	0.30	0.28	0.26	0.24	0.21	0.17	0.15	0.00			
23	0.50	0.49	0.47	0.45	0.44	0.44	0.41	0.39	0.39	0.39	0.38	0.37	0.36	0.35	0.34	0.33	0.31	0.29	0.27	0.24	0.21	0.19	0.13	0.00		
24	0.49	0.48	0.47	0.47	0.46	0.45	0.44	0.43	0.43	0.43	0.42	0.42	0.42	0.41	0.40	0.39	0.38	0.36	0.35	0.32	0.30	0.28	0.25	0.20	0.00	
25	0.47	0.47	0.47	0.47	0.47	0.47	0.46	0.46	0.46	0.46	0.46	0.46	0.46	0.46	0.46	0.45	0.44	0.43	0.42	0.40	0.39	0.38	0.35	0.32	0.22	0.00

Figure 7: Average angular distance between all layers of Gemma-2 2b, as defined in Section 3.1. The angular distances are bounded in $[0, 1]$, where an angular distance equal to 0 means equal activations, 0.5 means activations are perpendicular and an angular distance of 1 means that the activations point in opposite directions.



Average Angular Distance Between Layers

Layer	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41
0	0.00																																									
1	0.15	0.00																																								
2	0.23	0.19	0.00																																							
3	0.27	0.22	0.19	0.00																																						
4	0.30	0.26	0.20	0.17	0.00																																					
5	0.34	0.29	0.27	0.22	0.22	0.00																																				
6	0.35	0.30	0.28	0.23	0.20	0.16	0.00																																			
7	0.40	0.35	0.32	0.27	0.24	0.20	0.15	0.00																																		
8	0.40	0.36	0.33	0.29	0.26	0.22	0.18	0.12	0.00																																	
9	0.42	0.38	0.36	0.31	0.29	0.25	0.22	0.17	0.13	0.00																																
10	0.41	0.37	0.35	0.31	0.29	0.25	0.22	0.19	0.16	0.16	0.00																															
11	0.44	0.40	0.38	0.34	0.32	0.28	0.26	0.21	0.19	0.17	0.14	0.00																														
12	0.45	0.41	0.39	0.35	0.33	0.30	0.27	0.23	0.22	0.20	0.18	0.12	0.00																													
13	0.46	0.42	0.41	0.37	0.35	0.32	0.29	0.25	0.24	0.22	0.21	0.16	0.12	0.00																												
14	0.47	0.43	0.42	0.38	0.36	0.33	0.31	0.27	0.26	0.25	0.23	0.19	0.16	0.12	0.00																											
15	0.48	0.45	0.44	0.40	0.38	0.36	0.34	0.29	0.29	0.28	0.26	0.22	0.20	0.17	0.13	0.00																										
16	0.49	0.46	0.44	0.41	0.39	0.37	0.35	0.31	0.31	0.30	0.29	0.25	0.23	0.21	0.18	0.13	0.00																									
17	0.49	0.46	0.45	0.42	0.40	0.38	0.36	0.33	0.32	0.31	0.30	0.27	0.26	0.23	0.21	0.17	0.13	0.00																								
18	0.50	0.47	0.46	0.42	0.41	0.39	0.37	0.34	0.34	0.32	0.31	0.28	0.27	0.25	0.23	0.18	0.16	0.11	0.00																							
19	0.50	0.47	0.46	0.43	0.41	0.39	0.38	0.34	0.34	0.33	0.32	0.29	0.28	0.26	0.24	0.21	0.18	0.14	0.11	0.00																						
20	0.51	0.48	0.46	0.43	0.42	0.40	0.38	0.35	0.35	0.34	0.33	0.30	0.29	0.28	0.26	0.23	0.20	0.17	0.15	0.11	0.00																					
21	0.51	0.48	0.47	0.44	0.43	0.41	0.39	0.36	0.36	0.34	0.34	0.31	0.30	0.29	0.27	0.24	0.22	0.19	0.17	0.14	0.12	0.00																				
22	0.51	0.49	0.47	0.44	0.43	0.42	0.40	0.37	0.36	0.35	0.35	0.32	0.31	0.30	0.28	0.25	0.24	0.21	0.19	0.16	0.15	0.11	0.00																			
23	0.52	0.49	0.48	0.45	0.44	0.42	0.41	0.37	0.37	0.36	0.36	0.33	0.33	0.31	0.30	0.27	0.25	0.23	0.21	0.19	0.18	0.15	0.12	0.00																		
24	0.52	0.50	0.48	0.46	0.45	0.43	0.42	0.38	0.38	0.37	0.37	0.35	0.34	0.33	0.31	0.28	0.27	0.25	0.23	0.22	0.21	0.18	0.15	0.13	0.00																	
25	0.52	0.50	0.48	0.46	0.45	0.44	0.42	0.39	0.39	0.37	0.37	0.35	0.35	0.33	0.32	0.29	0.28	0.26	0.25	0.23	0.23	0.21	0.18	0.16	0.13	0.00																
26	0.53	0.50	0.49	0.47	0.46	0.44	0.43	0.40	0.40	0.38	0.39	0.36	0.36	0.35	0.33	0.31	0.30	0.29	0.27	0.26	0.25	0.23	0.21	0.19	0.16	0.13	0.00															
27	0.52	0.50	0.49	0.46	0.45	0.44	0.43	0.40	0.40	0.39	0.38	0.37	0.36	0.35	0.34	0.31	0.31	0.29	0.28	0.27	0.26	0.25	0.23	0.21	0.19	0.16	0.14	0.00														
28	0.53	0.50	0.49	0.47	0.46	0.45	0.44	0.41	0.41	0.39	0.39	0.38	0.37	0.36	0.35	0.33	0.32	0.31	0.29	0.28	0.28	0.26	0.24	0.23	0.21	0.19	0.17	0.13	0.00													
29	0.53	0.51	0.50	0.47	0.46	0.45	0.44	0.41	0.41	0.40	0.40	0.39	0.38	0.37	0.36	0.34	0.33	0.32	0.30	0.30	0.30	0.28	0.26	0.25	0.23	0.21	0.19	0.15	0.12	0.00												
30	0.53	0.51	0.50	0.48	0.47	0.46	0.45	0.42	0.42	0.41	0.41	0.39	0.39	0.38	0.37	0.34	0.34	0.33	0.31	0.31	0.31	0.29	0.27	0.26	0.24	0.22	0.20	0.17	0.15	0.10	0.00											
31	0.53	0.51	0.50	0.48	0.47	0.46	0.45	0.42	0.43	0.41	0.41	0.40	0.39	0.39	0.37	0.35	0.35	0.33	0.32	0.32	0.31	0.30	0.28	0.27	0.25	0.23	0.22	0.19	0.17	0.13	0.11	0.00										
32	0.53	0.52	0.50	0.48	0.48	0.46	0.45	0.43	0.43	0.42	0.42	0.40	0.40	0.39	0.38	0.36	0.35	0.34	0.33	0.32	0.32	0.31	0.29	0.28	0.26	0.24	0.23	0.21	0.18	0.15	0.12	0.10	0.00									
33	0.53	0.52	0.51	0.49	0.48	0.47	0.46	0.43	0.43	0.42	0.42	0.41	0.40	0.39	0.38	0.36	0.36	0.35	0.33	0.33	0.33	0.32	0.30	0.29	0.27	0.25	0.24	0.22	0.20	0.17	0.14	0.12	0.09	0.00								
34	0.54	0.52	0.51	0.49	0.48	0.47	0.46	0.44	0.44	0.43	0.43	0.41	0.41	0.40	0.39	0.37	0.37	0.35	0.34	0.34	0.34	0.32	0.31	0.30	0.27	0.26	0.24	0.23	0.21	0.18	0.16	0.14	0.10	0.09	0.00							
35	0.54	0.53	0.51	0.49	0.49	0.48	0.47	0.44	0.45	0.43	0.43	0.42	0.42	0.41	0.40	0.38	0.37	0.36	0.35	0.35	0.35	0.33	0.32	0.31	0.29	0.28	0.26	0.25	0.23	0.20	0.18	0.15	0.13	0.10	0.10	0.00						
36	0.54	0.53	0.52	0.50	0.49	0.48	0.47	0.44	0.45	0.44	0.44	0.43	0.42	0.41	0.40	0.38	0.38	0.37	0.36	0.35	0.35	0.34	0.33	0.32	0.30	0.29	0.27	0.26	0.24	0.21	0.19	0.17	0.15	0.13	0.11	0.09	0.00					
37	0.54	0.53	0.52	0.50	0.50	0.49	0.48	0.46	0.46	0.45	0.45	0.44	0.43	0.43	0.42	0.40	0.40	0.39	0.38	0.38	0.38	0.37	0.35	0.34	0.32	0.31	0.30	0.28	0.27	0.24	0.22	0.20	0.19	0.17	0.15	0.13	0.11	0.00				
38	0.54	0.53	0.52	0.51	0.50	0.49	0.49	0.47	0.47	0.47	0.47	0.46	0.45	0.44	0.44	0.42	0.42	0.41	0.40	0.40	0.40	0.39	0.38	0.37	0.36	0.34	0.33	0.32	0.30	0.28	0.26	0.24	0.23	0.21	0.20	0.18	0.16	0.11	0.00			
39	0.53	0.53	0.52	0.51	0.50	0.49	0.49	0.48	0.48	0.48	0.47	0.47	0.46	0.46	0.46	0.45	0.44	0.43	0.43	0.43	0.42	0.42	0.41	0.40	0.39	0.37	0.36	0.35	0.33	0.31	0.30	0.28	0.27	0.26	0.24	0.22	0.20	0.16	0.11	0.00		
40	0.51	0.51	0.50	0.50	0.49	0.49	0.49	0.48	0.48	0.48	0.48	0.47	0.47	0.47	0.47	0.46	0.46	0.46	0.45	0.45	0.45	0.45	0.44	0.43	0.42	0.41	0.40	0.39	0.37	0.36	0.34	0.33	0.32	0.31	0.30	0.28	0.26	0.23	0.19	0.15	0.00	
41	0.46	0.46	0.46	0.46	0.46	0.46	0.46	0.46	0.46	0.46	0.46	0.46	0.46	0.47	0.47	0.47	0.47	0.47	0.47	0.47	0.46	0.46	0.46	0.45	0.45	0.44	0.43	0.42	0.41	0.40	0.40	0.39	0.38	0.37	0.37	0.36	0.35	0.33	0.31	0.29	0.21	0.00

Figure 8: Average angular distance between all layers of Gemma-2 9b, as defined in Section 3.1. The angular distances are bounded in $[0, 1]$, where an angular distance equal to 0 means equal activations, 0.5 means activations are perpendicular and an angular distance of 1 means that the activations point in opposite directions.

k	Groups	MAAD
2	0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1	0.389
3	0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2	0.332
4	0, 0, 0, 1, 1, 1, 1, 2, 2, 3, 3	0.311
5	0, 0, 0, 1, 1, 2, 2, 3, 3, 4, 4	0.252

Table 3: Layer groups for every k up to $L/2$ for Pythia-160m

k	Groups	MAAD
2	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1	0.476
3	0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2	0.384
4	0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3	0.384
5	0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 4, 4	0.384
6	0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 5, 5	0.279
7	0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 5, 5, 5, 5, 5, 5, 6, 6	0.279
8	0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 5, 5, 5, 5, 6, 6, 7, 7	0.279
9	0, 0, 1, 1, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 6, 6, 6, 6, 7, 7, 8, 8	0.238
10	0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 4, 5, 5, 5, 5, 6, 6, 6, 7, 7, 7, 7, 8, 8, 9, 9	0.222

Table 4: Layer groups for every k up to $L/2$ for Gemma-2 2b

k	Groups	MAAD
2	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1	0.524
3	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2	0.467
4	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3	0.467
5	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4	0.333
6	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5	0.333
7	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6	0.305
8	0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 7, 7, 7, 7	0.305
9	0, 0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 7, 7, 7, 7, 8, 8	0.305
10	0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 7, 7, 8, 8, 9	0.251
11	0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 6, 6, 6, 6, 6, 7, 7, 8, 8, 9, 10, 10	0.227
12	0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 6, 6, 6, 6, 6, 7, 7, 8, 8, 9, 10, 10, 11, 11	0.216
13	1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 6, 6, 6, 6, 7, 7, 8, 8, 9, 10, 10, 11, 11, 12, 12	0.206
14	0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5, 6, 6, 6, 7, 7, 8, 8, 9, 10, 10, 11, 11, 12, 12, 13, 13, 13	0.202
15	0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 5, 5, 5, 6, 6, 6, 7, 7, 8, 8, 9, 10, 10, 11, 11, 12, 12, 13, 13, 13, 14, 14	0.202
16	0, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 4, 4, 4, 5, 5, 5, 6, 6, 6, 7, 7, 8, 8, 9, 10, 10, 11, 11, 12, 12, 13, 13, 13, 14, 14, 15, 15	0.202
17	0, 0, 0, 0, 1, 1, 2, 2, 2, 2, 3, 3, 4, 4, 4, 5, 5, 5, 6, 6, 6, 7, 7, 8, 8, 8, 9, 10, 10, 11, 11, 12, 12, 13, 13, 13, 14, 14, 15, 15, 16, 16	0.202
18	0, 0, 0, 0, 1, 1, 2, 2, 2, 3, 3, 4, 4, 4, 5, 5, 6, 6, 6, 7, 7, 8, 8, 8, 9, 10, 10, 11, 11, 12, 12, 13, 13, 13, 14, 14, 15, 15, 16, 16, 17, 17	0.202
19	0, 0, 0, 1, 1, 2, 2, 2, 3, 3, 4, 4, 4, 5, 5, 6, 6, 6, 7, 7, 8, 8, 8, 9, 10, 10, 11, 11, 12, 12, 13, 13, 14, 14, 15, 15, 16, 16, 17, 17, 18, 18	0.175
20	0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 6, 7, 7, 8, 8, 8, 9, 10, 10, 11, 11, 12, 12, 13, 13, 14, 14, 15, 15, 16, 16, 17, 17, 18, 18, 19	0.175

Table 5: Layer groups for every k up to $L/2$ for Gemma-2 9b

C DETAILED PER-LAYER RECONSTRUCTION AND SPARSITY PLOTS

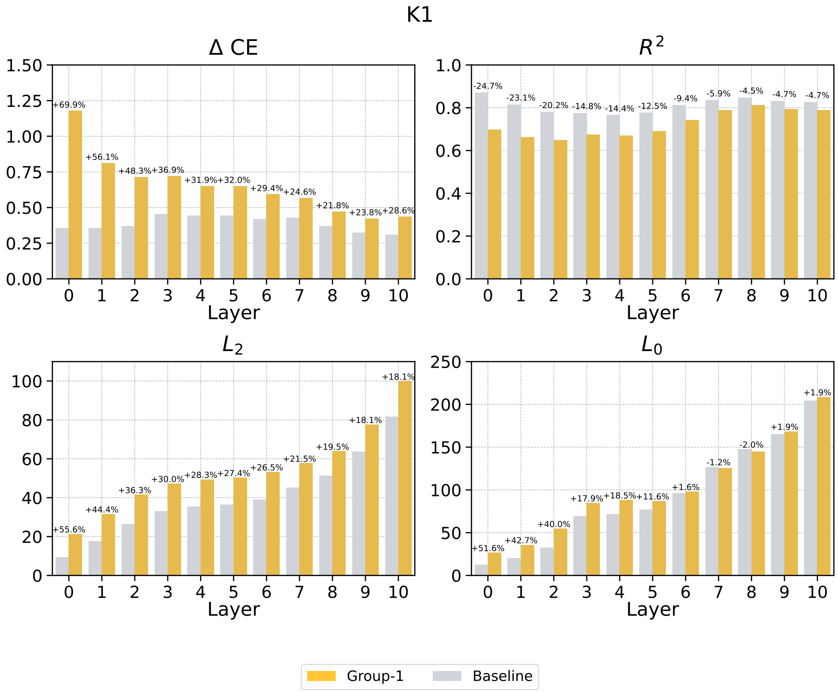


Figure 9: Per-layer ΔCE, R^2, L_2 and L_0 with $k = 1$.

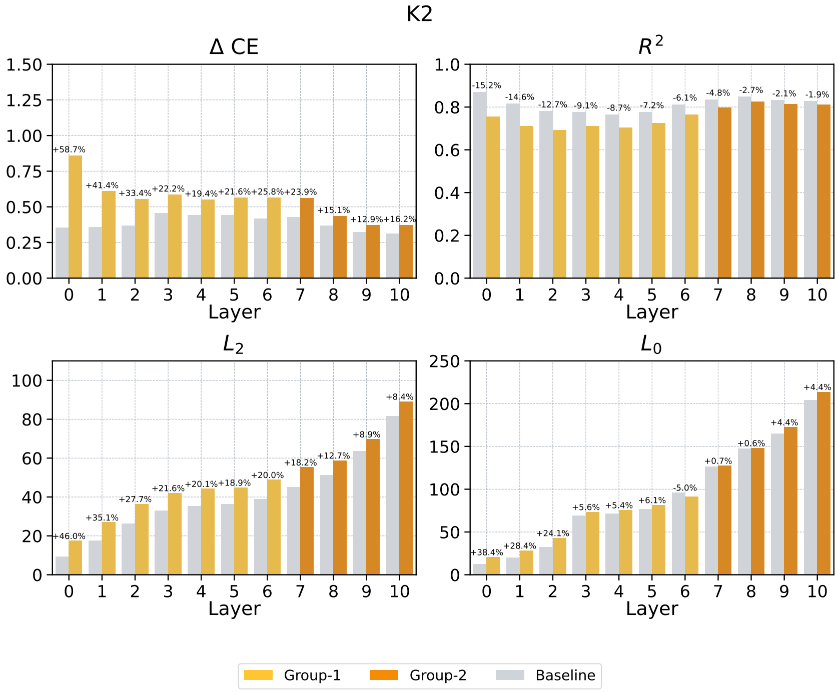


Figure 10: Per-layer ΔCE, R^2, L_2 and L_0 with $k = 2$.

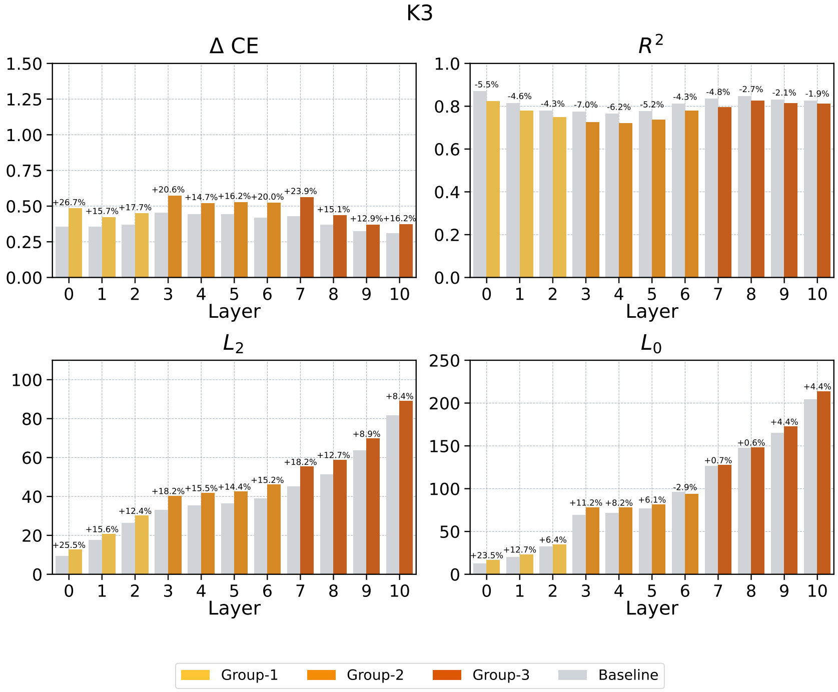


Figure 11: Per-layer ΔCE, R^2, L_2 and L_0 with $k = 3$.

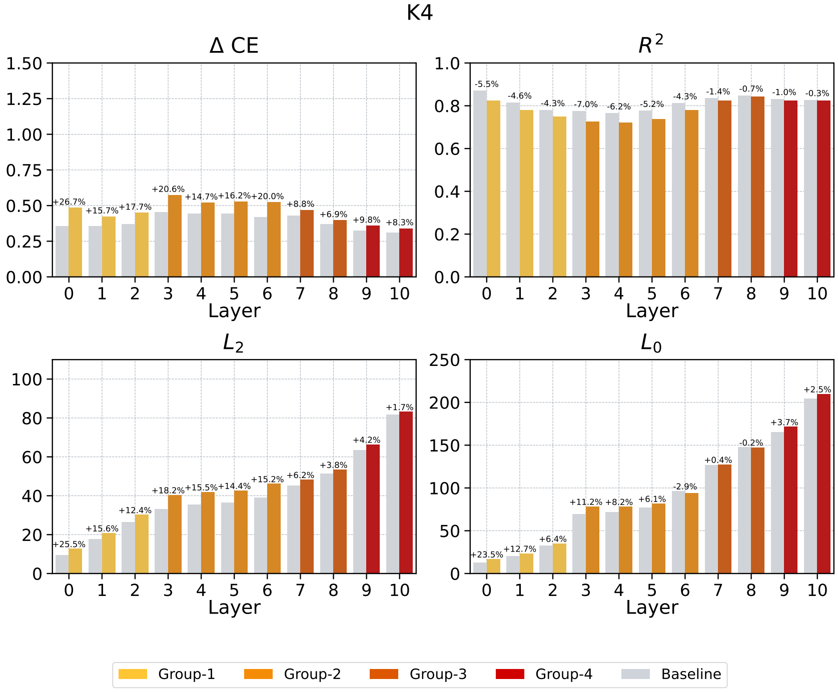


Figure 12: Per-layer ΔCE, R^2, L_2 and L_0 with $k = 4$.

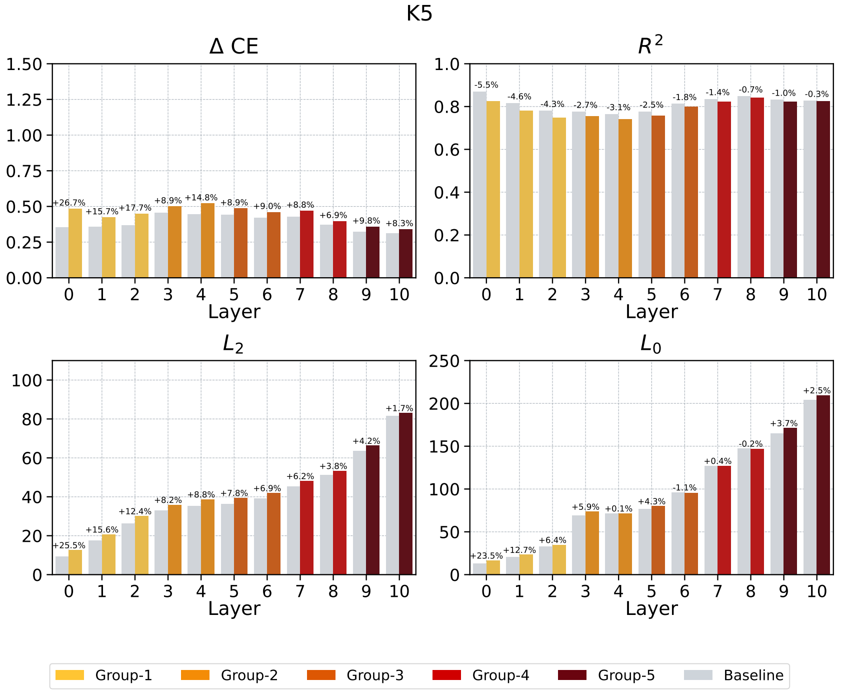


Figure 13: Per-layer ΔCE, R^2, L_2 and L_0 with $k = 5$.

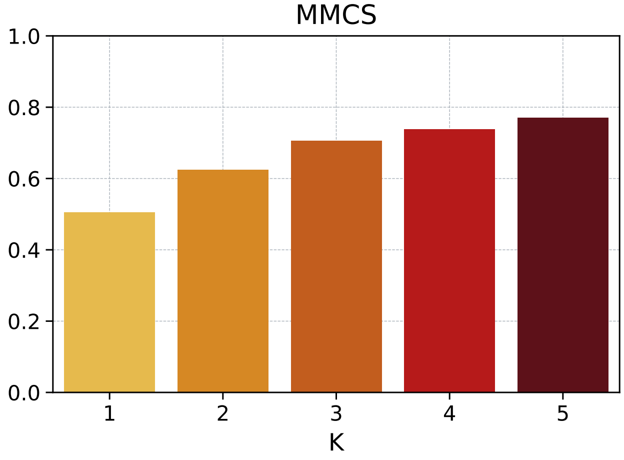


Figure 14: Per-layer ΔCE, R^2, L_2 and L_0 with $k = 5$.

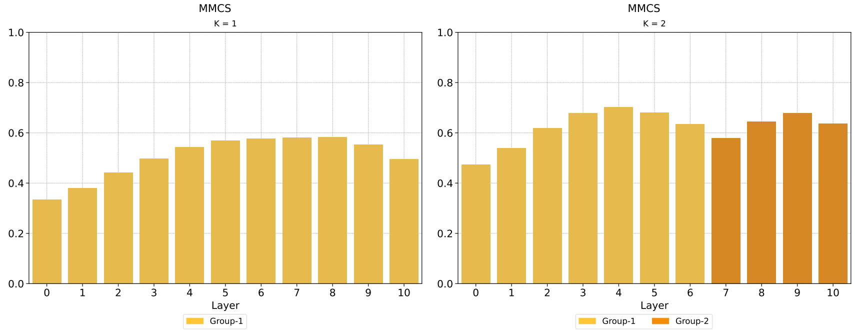
D DETAILED PER-LAYER MMCS PLOTS

The Maximum Mean Cosine Similarity (MMCS) Score is defined as:

$$\text{MMCS} = \frac{1}{d_{\text{sae}}} \sum_{\mathbf{u}} \max_{\mathbf{v}} \text{CosSim}(\mathbf{u}, \mathbf{v}) \tag{7}$$

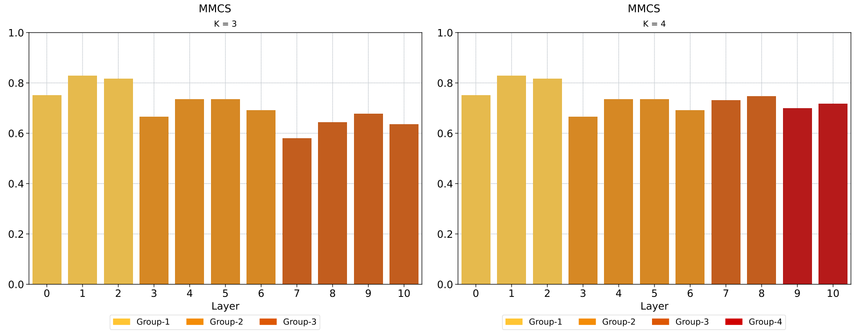
where $\mathbf{u}$ and $\mathbf{v}$ are the columns of the SAE_{j_k} and SAE_i decoder matrices, respectively. Figure 14 shows the average MMCS, where the average is computed for a given k by calculating the MMCS between SAE_{j_k} and SAE_s for every $1 \leq j \leq k$ and $s \in [j_k]$, then dividing by $L - 1$.

Figure 15 shows the per-layer MMCS of every SAE_{j_k} for every k.



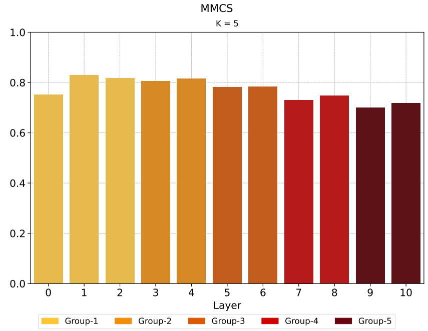
(a) Per-layer MMCS with $k = 1$.

(b) Per-layer MMCS with $k = 2$.



(c) Per-layer MMCS with $k = 3$.

(d) Per-layer MMCS with $k = 4$.



(e) Per-layer MMCS with $k = 5$.

Figure 15: Per-layer MMCS for every k.

E APPROXIMATE INDIRECT EFFECTS

In Equation 6 we reported the Indirect Effect (IE) (Pearl, 2022), which measures the importance of a feature with respect to a generic downstream task $\mathcal{T}$. To reduce the computational burden of estimating the IE with a single forward pass per feature, we employed two approximate methods: Attribution Patching (AtP) (Nanda, 2023; Syed et al., 2024) and Integrated Gradients (IG) (Sundararajan et al., 2017).

AtP (Nanda, 2023; Syed et al., 2024) employs a first-order Taylor expansion

$$\hat{\text{IE}}_{\text{AtP}}(m; \boldsymbol{f}_i; x_{\text{clean}}, x_{\text{corrupted}}) = \nabla m\big|_{\boldsymbol{f}_i = \boldsymbol{f}_{i;\text{clean}}} (\boldsymbol{f}_{i;\text{corrupted}} - \boldsymbol{f}_{i;\text{clean}}) \tag{8}$$

which estimates Equation 6 for every $\boldsymbol{f}_i$ in two forward passes and a single backward pass.

Integrated Gradients (Sundararajan et al., 2017) is a more expensive but more accurate approximation of Equation 6

$$\hat{\text{IE}}_{\text{IG}}(m; \boldsymbol{f}_i; x_{\text{clean}}, x_{\text{corrupted}}) = \left(\sum_{\alpha \in \Gamma} \nabla m\big|_{\alpha \boldsymbol{f}_{i;\text{clean}} + (1-\alpha)\boldsymbol{f}_{i;\text{corrupted}}} \right) (\boldsymbol{f}_{i;\text{corrupted}} - \boldsymbol{f}_{i;\text{clean}}) \tag{9}$$

where α ranges in an equally spaced set $\Gamma = \{0, \frac{1}{N}, ..., \frac{N-1}{N}\}$. In our experiments we have set $N = 10$.

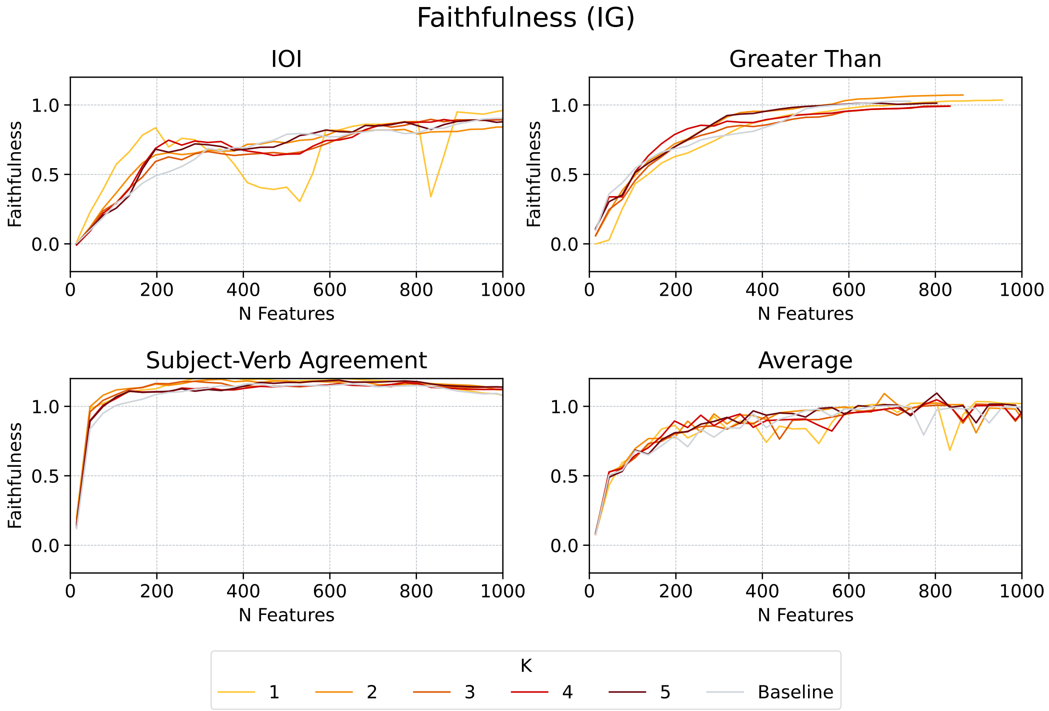


Figure 16: Average faithfulness (Marks et al., 2024) for every downstream task (IOI, Greater Than, Subject-Verb Agreement). The “Baseline” average is computed considering the performance obtained by SAE$_i$, $\forall i = 0, ..., 10$. The “Average” plot depicts the average over the three downstream tasks.

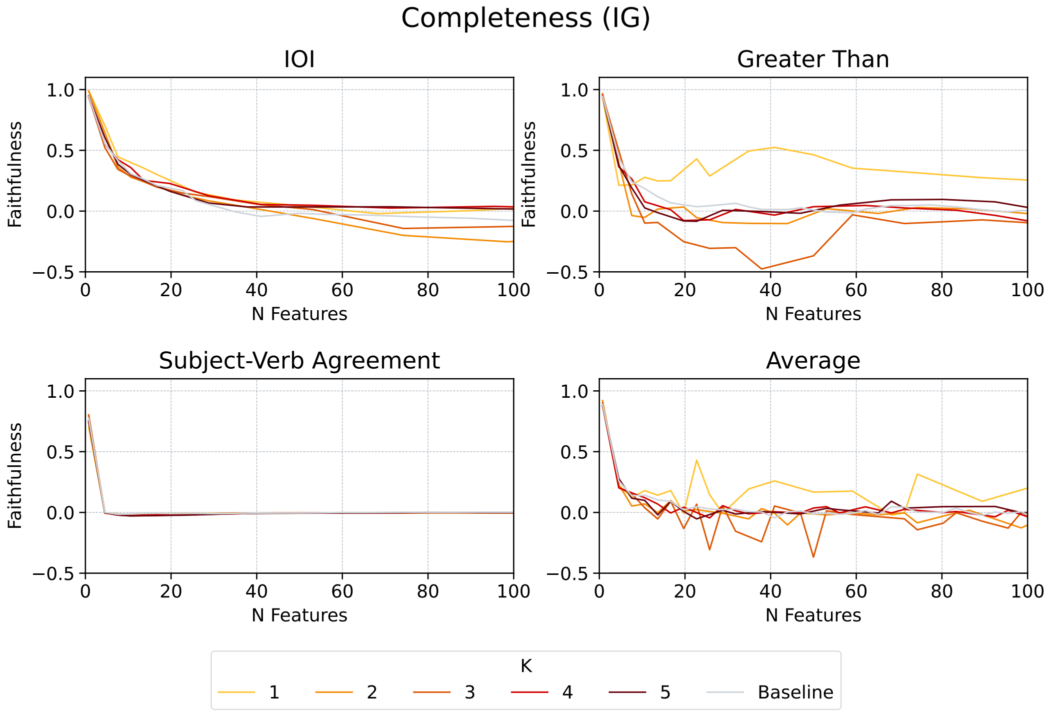


Figure 17: Average completeness for every downstream task (IOI, Greater Than, Subject-Verb Agreement). The “Baseline” average is computed considering the performance obtained by SAE$_i$, $\forall i = 0, ..., 10$. The “Average” plot depicts the average performance over the three downstream tasks.

F HUMAN INTERPRETABILITY SCORES

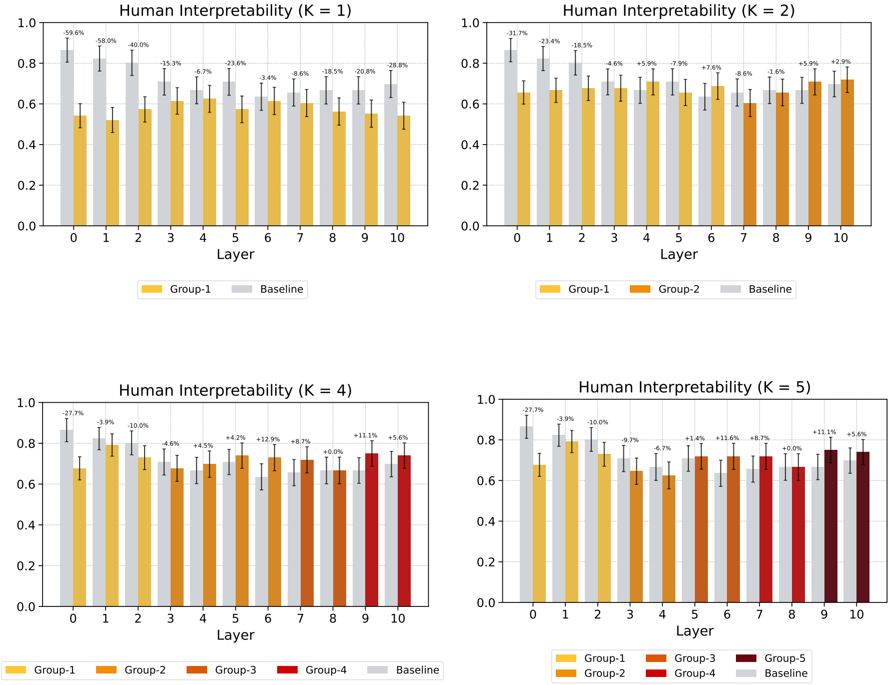


Figure 18: Human Interpretability Scores for $k \in \{1, 2, 4, 5\}$. The differences in the interpretability scores of features learned by the SAE$_{j_k}$ and the baseline SAE$_i$ are not statistically significant different from zero for most of the layers. Error bars shows one standard deviations of the scores differences, modeled as a Binomial distribution (Wasserman, 2010)

G SCALING GROUP SAES

Results on Gemma-2 2b will be added here.