Route Sparse Autoencoder to Interpret Large Language Models

Wei Shi[1]*, Sihang Li[1]*, Tao Liang[2], Mingyang Wan[2],
Guojun Ma[2]†, Xiang Wang[1]†, Xiangnan He[1],

[1]University of Science and Technology of China, [2]Douyin Co., Ltd.,
swei2001@mail.ustc.edu.cn, taoliangdpg@126.com,
{wanmingyang, maguojun}@bytedance.com,
{sihang0520, xiangwang1223, xiangnanhe}@gmail.com

Abstract

Mechanistic interpretability of large language models (LLMs) aims to uncover the internal processes of information propagation and reasoning. Sparse autoencoders (SAEs) have demonstrated promise in this domain by extracting interpretable and monosemantic features. However, prior works primarily focus on feature extraction from a single layer, failing to effectively capture activations that span multiple layers. In this paper, we introduce Route Sparse Autoencoder (RouteSAE), a new framework that integrates a routing mechanism with a shared SAE to efficiently extract features from multiple layers. It dynamically assigns weights to activations from different layers, incurring minimal parameter overhead while achieving high interpretability and flexibility for targeted feature manipulation. We evaluate RouteSAE through extensive experiments on Llama-3.2-1B-Instruct. Specifically, under the same sparsity constraint of 64, RouteSAE extracts 22.5% more features than baseline SAEs while achieving a 22.3% higher interpretability score. These results underscore the potential of RouteSAE as a scalable and effective method for LLM interpretability, with applications in feature discovery and model intervention. Our codes are available at https://github.com/swei2001/RouteSAEs.

1 Introduction

Mechanistic interpretability of large language models (LLMs) seeks to understand and intervene in the internal process of information propagation and reasoning, to further improve trust and safety (Elhage et al., 2022b; Gurnee et al., 2023; Wang et al., 2023). Sparse autoencoders (SAEs) identify causally relevant and interpretable monosemantic features in LLMs, offering a promising solution for mechanistic interpretability (Bricken et al., 2023). Therefore, SAE and its variants (Huben et al., 2024; Rajamanoharan et al., 2024a; Gao et al., 2024; Rajamanoharan et al., 2024b) have been widely utilized in LLM interpretation tasks, such as feature discovery (Templeton et al., 2024; Gao et al., 2024) and circuit analysis (Marks et al., 2024).

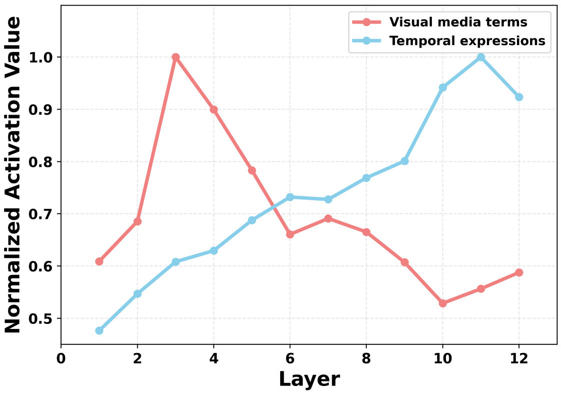


Figure 1: Layer-wise normalized activation values for two features extracted by Topk SAE in pythia-160m. The low-level feature (visual media terms) exhibits high activation in early layers that gradually decreases in deeper layers. In contrast, the high-level feature (temporal expressions) shows increasing activation with depth, peaking in the later layers.

Typically, SAE is trained in an unsupervised manner. It first disentangles the intermediate activations from a single layer in the language model into a sparse, high-dimensional feature space, which is subsequently reconstructed by a decoder. This process reverses the effects of superposition (Elhage et al., 2022a) by extracting features that are sparse, linear, and decomposable.

However, the activation strength of features in this feature space exhibits distinct distribution patterns across layers[1] (Yun et al., 2021). As shown in Figure 1, low-level features, which are associated with disambiguating word-level polysemy, tend to exhibit peak activation in the early layers and de-

*Equal Contribution
†Corresponding

[1] Referred to as “Transformer factors” in (Yun et al., 2021).

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 6813–6827
November 4-9, 2025 ©2025 Association for Computational Linguistics

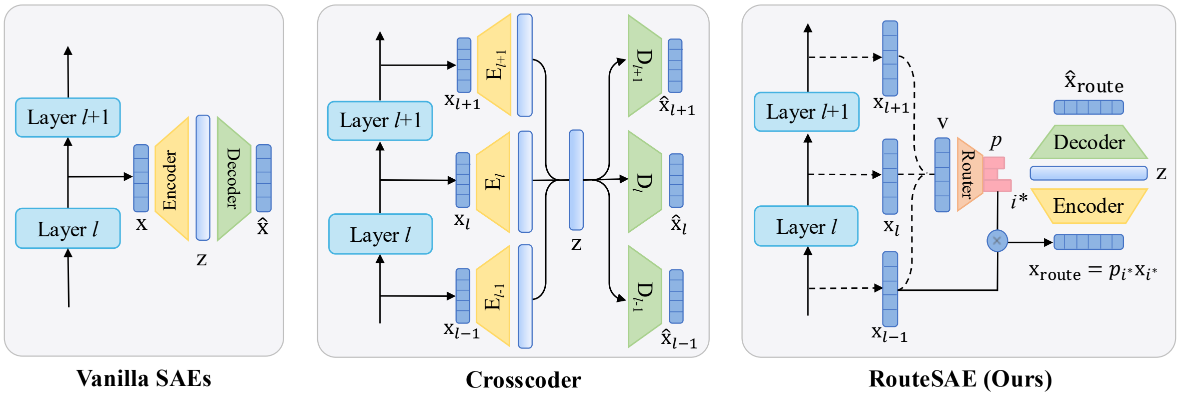


Figure 2: Comparison of vanilla single-layer SAE, Crosscoder, and RouteSAE. Most existing SAEs belong to the vanilla SAE category, where features are extracted from the activation of a single layer. Crosscoder relies on separate encoders and decoders for each layer. RouteSAE incorporates a lightweight router to dynamically integrate multi-layer residual stream activations.

cline steadily in deeper layers. High-level features, which capture sentence-level or long-range structure, show increasing activation with depth.[2]

This distribution disparity presents a significant challenge for previous SAEs (Huben et al., 2024; Rajamanoharan et al., 2024a; Gao et al., 2024; Rajamanoharan et al., 2024b), as they typically extract features from the hidden state of a single layer, failing to capture feature activating at other layers effectively (*cf.* Figure 2). Recently proposed Sparse Crosscoders[3] (Lindsey et al., 2024) serve as an alternative to address this limitation, which separately encodes the hidden states of each layer into a high-dimensional feature space and aggregates the resulting representations for reconstruction (*cf.* Figure 2). This approach facilitates the joint learning of features across different layers. However, Crosscoder has two critical limitations: (1) **Limited scalability**: For an L-layer model, Crosscoder employs L separate encoders and decoders to process activations layer by layer, resulting in a parameter scale approximately L times larger than traditional SAEs. This significantly increases computational overhead during both training and inference. (2) **Uncontrollable interventions**: Crosscoder's joint learning mechanism projects hidden states into a high-dimensional space and then aggregates them, making it impractical to precisely identify and adjust the feature activations at specific layers. This limits its flexibility for tasks requiring controlled, feature-level interventions, *e.g.,* feature steering (Templeton et al., 2024).

To address these challenges, we propose **Route Sparse Autoencoder (RouteSAE)**. At the core is integrating a lightweight router with a shared SAE to dynamically extract multi-layer features in an efficient and flexible manner. A router is employed to compute normalized weights for activations from multiple layers. This dynamic weighting approach significantly reduces the number of parameters compared to a suite of layer-specific encoders and decoders, thereby addressing scalability concerns. Additionally, by unifying feature disentanglement and reconstruction within a shared SAE, RouteSAE facilitates fine-grained adjustments of specific feature activations, enabling more controlled interventions to influence the model's output. This enhances flexibility and supports precise feature-level control, making the framework well-suited for tasks requiring robust and interpretable manipulation of model activations.

We conduct comprehensive experiments on Llama-3.2-1B-Instruct (Dubey et al., 2024), evaluating downstream KL divergence, interpretable feature numbers, and interpretation score. The experimental results demonstrate that RouteSAE significantly improves the interpretability. At an equivalent sparsity level of 64, it achieves a 22.5% increase in the number of interpretable features and a 22.3% improvement in interpretation scores.

Our contributions are summarized as follows:

- We propose RouteSAE, a novel sparse autoencoder framework that integrates multi-layer activations through a routing mechanism.

- RouteSAE achieves higher computational efficiency than Crosscoder by using a shared SAE structure with minimal additional parameters.

[2] Refer to (Yun et al., 2021) for more examples of low- and high-level features.

[3] Currently a conceptual framework without complete experimental validation.

- Extensive experiments confirm that RouteSAE enhances model interpretability, highlighting the effectiveness of the proposed routing mechanism.

2 Methodology

In this section, we first briefly review SAEs, then introduce our proposed Route Sparse Autoencoder (**RouteSAE**) in detail.

2.1 Preliminary

SAE and Feature Decomposition. SAEs decompose language model activations — typically residual streams (He et al., 2016), $\mathbf{x} \in \mathbb{R}^d$, into a sparse linear combination of features $\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_M \in \mathbb{R}^d$, where $M \gg d$ represents the feature space dimension. The original activation $\mathbf{x}$ is reconstructed using an encoder-decoder pair defined as follows:

$$\mathbf{z} = \sigma(\mathbf{W}_{\text{enc}}(\mathbf{x} - \mathbf{b}_{\text{pre}})) \tag{1}$$

$$\hat{\mathbf{x}} = \mathbf{W}_{\text{dec}}\mathbf{z} + \mathbf{b}_{\text{pre}}, \tag{2}$$

where $\mathbf{W}_{\text{enc}} \in \mathbb{R}^{M \times d}$ and $\mathbf{W}_{\text{dec}} \in \mathbb{R}^{d \times M}$ are the encoder and decoder weight matrices, $\mathbf{b}_{\text{pre}} \in \mathbb{R}^d$ is a bias term, and σ denotes the activation function. The latent representation $\mathbf{z} \in \mathbb{R}^M$ encodes the activation strength of each feature. The columns of the decoder matrix, $\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_M$, represent interpretable features in the latent space. The training objective is to minimize the reconstruction mean squared error (MSE):

$$\mathcal{L} = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2. \tag{3}$$

TopK SAE. Early SAEs (Huben et al., 2024; Bricken et al., 2023) leverage the ReLU activation function (Agarap, 2019) to generate sparse feature representations, coupled with an additional L_1 regularization term on latent representation $\mathbf{z}$ to enforce sparsity. However, this approach is prone to feature shrinkage, where the L_1 constraint drives positive activations in $\mathbf{z}$ toward zero, reducing the expressive capacity of the sparse feature space. To mitigate this issue, TopK SAE (Gao et al., 2024) replaces the ReLU activation function with a TopK$(\cdot)$ function, which directly controls the number of active latent dimensions by selecting the top K largest values in $\mathbf{z}$. This is defined as:

$$\mathbf{z} = \text{TopK}(\mathbf{W}_{\text{enc}}(\mathbf{x} - \mathbf{b}_{\text{pre}})). \tag{4}$$

By eliminating the need for an L_1 regularization term, TopK SAE achieves a more effective balance between sparsity and reconstruction quality, while enhancing the model's ability to learn disentangled and interpretable monosemantic features. In our RouteSAE framework, the shared SAE module is instantiated as a TopK SAE due to its superior performance in producing monosemantic features.

2.2 Route Sparse Autoencoder (RouteSAE)

As shown in Figure 2, existing SAEs are typically trained on intermediate activations from a single layer, restricting their ability to simultaneously capture both low-level features from shallow layers and high-level features from deep layers. To overcome this limitation, RouteSAE incorporates a lightweight router to dynamically integrate multi-layer residual streams from language models and disentangle them into a unified feature space.

Layer Weights. As illustrated in Figure 3, the router receives residual streams from multiple layers and determines which layer's activation to route. Instead of concatenating these activations, which could result in an excessively large input dimension, we adopt a simple yet effective aggregation strategy: sum pooling. Specifically, given activations $\mathbf{x}_i \in \mathbb{R}^d$ from layer i, we aggregate them using sum pooling to form the router's input:

$$\mathbf{v} = \sum_{i=0}^{L-1} \mathbf{x}_i, \quad \mathbf{x}_i \in \mathbb{R}^d, \tag{5}$$

where L denotes the total number of layers being routed. The resulting vector $\mathbf{v} \in \mathbb{R}^d$ serves as a condensed representation of multi-layer activations. Next, the router projects $\mathbf{v}$ into $\mathbb{R}^L$ using a learnable weight matrix $\mathbf{W}_{\text{router}} \in \mathbb{R}^{L \times d}$, yielding the layer weight vector $\boldsymbol{\alpha}$:

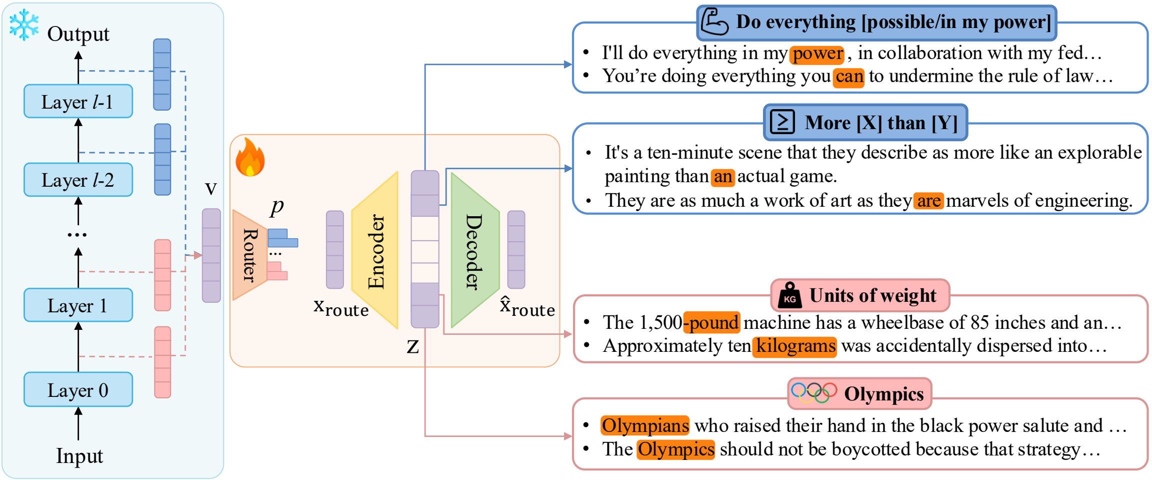
$$\boldsymbol{\alpha} = \mathbf{W}_{\text{router}}\mathbf{v} \in \mathbb{R}^L. \tag{6}$$

Each element α_i in $\boldsymbol{\alpha}$ represents the unnormalized weight for layer i, indicating its relative importance in the routing process. These weights are then normalized using a softmax function to obtain layer selection probabilities p_i:

$$p_i = \frac{\exp(\alpha_i)}{\sum_{j=0}^{L-1} \exp(\alpha_j)}, \quad i = 0, 1, \ldots, L-1. \tag{7}$$

p_i reflects the likelihood that the activation strength peaks at layer i, dynamically assigned by the router based on the input representations.

Routing Mechanisms. In RouteSAE, the router selects the layer i^* with the highest probability p_i,



- I'll do everything in my power, in collaboration with my fed…
- You're doing everything you can to undermine the rule of law…

- It's a ten-minute scene that they describe as more like an explorable painting than an actual game.
- They are as much a work of art as they are marvels of engineering.

- The 1,500-pound machine has a wheelbase of 85 inches and an…
- Approximately ten kilograms was accidentally dispersed into…

- Olympians who raised their hand in the black power salute and …
- The Olympics should not be boycotted because that strategy…

Figure 3: RouteSAE employs a lightweight router to dynamically integrate activations from multiple residual stream layers, effectively disentangling them into a shared feature space. It enables the model to capture features across different layers — low-level features such as "units of weight" and "Olympics" from shallow layers, and high-level features like "more [X] than [Y]" and "do everything [possible/in my power]" from deeper layers.

computed as described in Equation 7. Formally, this is expressed as:

$$i^* = \arg\max_i p_i, \quad i = 0, 1, \ldots, L-1. \tag{8}$$

To ensure differentiability, we scale the activation $\mathbf{x}_{i^*}$ from the selected layer i^* by its corresponding probability p_{i^*}, using it as input to the shared SAE for disentangling into the high-dimensional feature space and subsequent reconstruction training:

$$\mathbf{x}_{\text{route}} = p_{i^*} \mathbf{x}_{i^*}. \tag{9}$$

The latent representation $\mathbf{z}$ and the reconstruction $\hat{\mathbf{x}}$ are calculated as follows:

$$\mathbf{z}_{\text{route}} = \text{TopK}(\mathbf{W}_{\text{enc}}(\mathbf{x}_{\text{route}} - \mathbf{b}_{\text{pre}})) \tag{10}$$

$$\hat{\mathbf{x}}_{\text{route}} = \mathbf{W}_{\text{dec}} \mathbf{z}_{\text{route}} + \mathbf{b}_{\text{pre}}. \tag{11}$$

Finally, we minimize the reconstruction MSE:

$$\mathcal{L} = \|\mathbf{x}_{\text{route}} - \hat{\mathbf{x}}_{\text{route}}\|_2^2. \tag{12}$$

This objective function jointly trains the router and the shared TopK SAE, ensuring efficient and adaptive feature extraction across multiple layers.

Shared SAE and Unified Feature Space. The routed intermediate activation ($\mathbf{x}_{\text{route}}$, as defined in Equation 9) is processed by a shared SAE for reconstruction, which in this work is instantiated as a TopK SAE (Gao et al., 2024). Notably, RouteSAE is flexible and can be easily adapted to various SAE variants. By employing a shared SAE, RouteSAE establishes a unified feature space across activations from all routing layers. This ensures consistent feature representations, thereby enhancing the disentanglement of high-dimensional features and improving interpretability.

Model	Llama-3.2-1B-Instruct
Hidden Size	2,048
# Layers	16
Routing Layers	[3:11]
SAE Width	16,384 (8x)
Batch Size	64

Table 1: Implementation details of RouteSAEs for Llama-3.2-1B-Instruct. Note that the layer indices start from 0.

3 Experiments

We first outline the experimental setup, followed by the evaluation of RouteSAE. In this paper, we follow prior work (Gao et al., 2024; Rajamanoharan et al., 2024a; Huben et al., 2024; Templeton et al., 2024; He et al., 2024) and employ multiple evaluation metrics to assess the effectiveness of RouteSAE, including downstream KL-divergence, interpretable features, interpretation score, and reconstruction loss. Finally, we provide a detailed case study, demonstrating that RouteSAE not only effectively captures both low-level features from shallow layers and high-level features from deep layers, but also enables targeted manipulation of these features to control the model's output.

3.1 Setup

Inputs. We train all SAEs on the residual streams of the Llama-3.2-1B-Instruct. For baseline SAEs, we follow the standard approach (Gao et al., 2024) of selecting the layer located approximately at $\frac{3}{4}$ of the model depth (*i.e.,* Layer 11). Prior work

(Lad et al., 2024) has shown that the early layers of LLMs primarily handle detokenization, whereas later layers specialize in next-token prediction. Based on this insight, we select residual streams from the middle layers of the model as input for both RouteSAE and Crosscoder (Lindsey et al., 2024). In particular, we focus on layers spanning $\frac{1}{4}$ to $\frac{3}{4}$ of the model depth, as detailed in Table 1.

The training data is sourced from OpenWebText2 (Gao et al., 2020), comprising approximately 100 million randomly sampled tokens for training, with an additional 10 million tokens reserved for evaluation. All experiments are conducted using a context length of 512 tokens. To ensure stable training, we normalize the language model activations following (Gao et al., 2024).

Hyperparameters. For all SAEs, we use the Adam optimizer (Kingma and Ba, 2015) with standard settings: $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate is set to 5×10^{-4}, following a three-phase schedule. (1) Linear warmup. The learning rate increases linearly from 0 to the target rate over the first 5% of training steps. (2) Stable phase. The learning rate remains constant for 75% of the training steps. (3) Linear Decay. The learning rate gradually decreases to zero over the final 20% of training steps to ensure smooth convergence. To improve training stability, we apply unit norm regularization (Gao et al., 2024) to the columns of the SAE decoder every 10 steps, ensuring that the decoder columns maintain unit length.

Baselines. We benchmark RouteSAE against leading baselines, including ReLU SAE (Huben et al., 2024), Gated SAE (Rajamanoharan et al., 2024a), TopK SAE, and Crosscoder (Lindsey et al., 2024). Moreover, we compare with a random setting, where the router is replaced by a uniform distribution that assigns equal routing weights to each layer. It is important to note that Crosscoder remains a conceptual framework and lacks complete experimental validation. As there is no official codebase or hyperparameter guidance available, we implement it following the description in (Lindsey et al., 2024). We acknowledge that our results *may not fully reflect its actual performance*.

3.2 Downstream KL Divergence

To assess whether the extracted features are relevant for language modeling, we replace the residual streams $\mathbf{x}$ with the reconstructed representation $\hat{\mathbf{x}}$ during the forward pass of the language model and evaluate the reconstruction quality using Kullback-Leibler (KL) divergence. It quantifies the discrepancy between the original and reconstructed distributions, with lower value indicating that the extracted features are highly relevant for language modeling. Note that RouteSAE replaces the activation at the layer with the highest routing weight.

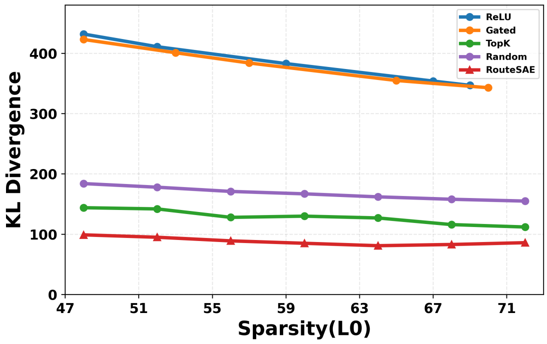


Figure 4: Pareto frontier of sparsity versus KL divergence. RouteSAE achieves a lower KL divergence at the same sparsity level.

As shown in Figure 4, the sparsity-KL divergence frontiers for ReLU and Gated SAE are nearly identical, yet both exhibit a significant gap compared to TopK SAE. Due to suboptimal reconstruction quality, the KL divergence for ReLU and Gated SAE drops substantially as L_0 increases, falling from around 400 to 350. In contrast, the KL divergence for both TopK and RouteSAE remains consistently below 150, with only minimal decreases as L_0 increases. This indicates that both methods are able to effectively reconstruct the original input $\mathbf{x}$ even at high sparsity levels. The random routing baseline yields higher KL divergence than both TopK and RouteSAE, further highlighting the advantage of learned routing.

Notably, RouteSAE achieves the best performance among all methods, maintaining a lower KL divergence at comparable sparsity levels. It outperforms even TopK SAE, indicating that feature substitution during inference is most effective when performed at the layer where the target feature is most active, rather than at a predetermined fixed layer. We exclude Crosscoder from this comparison, as it produces multiple reconstructed representations $\hat{\mathbf{x}}$, making its application to this setting nontrivial and not directly comparable.

3.3 Interpretable Features

Previous works (Huben et al., 2024; He et al., 2024) interpret features by preserving the context with the highest feature activation value. However, we argue that it has two limitations: (1) Retaining only the

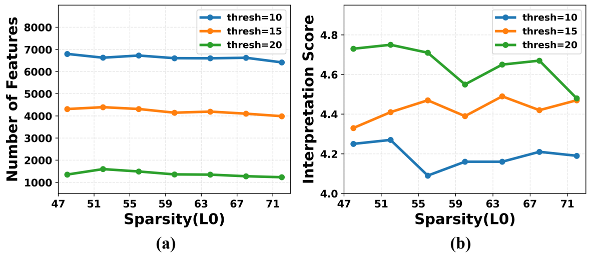


Figure 5: Effect of threshold on feature interpretability in RouteSAE. (a) Increasing the threshold reduces the number of selected features. (b) Higher thresholds yield better interpretation scores across sparsity levels.

highest activation context for each feature leads to a large number of undiscernible features; (2) Each feature is associated with only a single context, reducing the reliability of the interpretation.

To address these limitations, we introduce a new approach for preserving feature contexts using an activation threshold. For a given sequence context, only features with activation values exceeding the threshold are retained. As shown in Figure 5(a), increasing the threshold reduces the number of retained features. In contrast, Figure 5(b) demonstrates that a higher threshold leads to improved interpretation scores. Consequently, the threshold governs a trade-off between the quantity of interpretable features and their interpretability quality. In this section, we set the threshold to 15, which achieves a balance between maintaining sufficient feature quantity and enhancing interpretability. Notably, a single sequence may be associated with multiple contexts.

To further refine the interpretation, activated contexts are categorized based on their activation tokens, maintaining a min-heap of activation values. We retain the top 2 contexts with the highest activation values within each activated token. A filtering step is applied to remove features with fewer than four active contexts, ensuring that only sufficiently represented features are considered. To evaluate feature extraction, we use 10 million tokens from the evaluation set to extract contexts associated with each feature.

As illustrated in Figure 6, at a threshold of 15, both ReLU and Gated SAE extract over 1,000 interpretable features, performing similarly. In contrast, TopK SAE significantly outperforms both, extracting more than 3,000 features. RouteSAE surpasses all other methods, extracting over 4,000 features at the same threshold. Notably, RouteSAE exhibits a more gradual decline in the number of extracted features as L_0 increases, while TopK SAE exhibits a more pronounced reduction. The random routing baseline sometimes extracts even more features than RouteSAE, but its feature count decreases much more rapidly as L_0 increases. These results suggest that learning based solely on single-layer activation values limits the ability of SAEs to extract interpretable features. In comparison, Crosscoder extracts substantially fewer features, retaining approximately 200. Since Crosscoder aggregates and projects activations across multiple layers, we hypothesize that the optimal threshold for balancing feature quantity and interpretability lies in a lower range for Crosscoder. Therefore, comparing it against the same activation threshold may not reflect its actual ability to extract high-quality features. We plan to investigate this in future work.

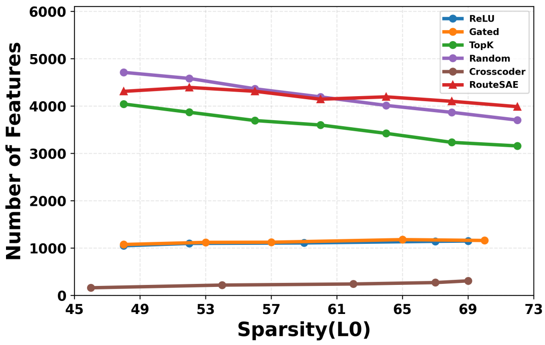


Figure 6: Comparison of the interpretable feature number. RouteSAE extracts the most interpretable features at the same threshold.

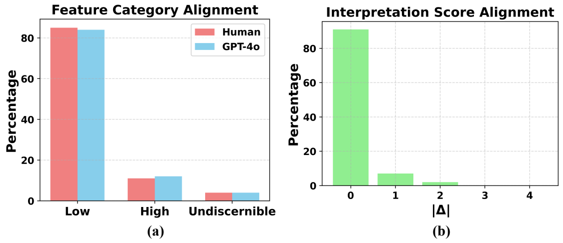


Figure 7: Human–GPT-4o alignment in the automatic feature interpretation pipeline. (a) Percentage of features assigned to each category (Low, High, or Undiscernible) by humans and GPT-4o. (b) Distribution of the absolute differences in interpretability scores between human annotators and GPT-4o.

3.4 Interpretation Score

Despite the feature screening in Section 3.3, the number of retained features remains in the thousands, making manual interpretation and evaluation challenging. To further assess feature interpretability, we follow prior work (Huben et al., 2024; Templeton et al., 2024; He et al., 2024) and leverage

GPT-4o (Hurst et al., 2024) to analyze the features extracted by SAEs, assigning an interpretability score alongside feature descriptions. Unlike previous approaches, we provide GPT-4o with multiple token categories per feature along with their contextual usage. Given resource constraints, we randomly select a subset of 100 retained features per SAE for interpretation. As detailed below, for each feature, we construct a structured prompt comprising a prefix prompt, the activated token, and its surrounding context, which is then given to GPT-4o. GPT-4o outputs three standardized components: (1) Feature categorization, labeling each feature as low-level, high-level, or undiscernible; (2) Interpretability score, rated on a scale of 1 to 5; and (3) Explanation, providing a brief justification for the assigned category and score.

Background
We are analyzing the activation levels of features in a neural network, where each feature activates certain tokens in a text. Each token's activation value indicates its relevance to the feature, with higher values showing stronger association. Features are categorized as:
A. Low-level features, which are associated with word-level polysemy disambiguation (e.g., "crushed things", "Europe").
B. High-level features, which are associated with long-range pattern formation (e.g., "enumeration", "one of the [number/quantifier]")
C. Undiscernible features, which are associated with noise or irrelevant patterns.

Task description
Your task is to classify the feature as low-level, high-level or undiscernible and give this feature a monosemanticity score based on the following scoring rubric:
Activation Consistency
5: Clear pattern with no deviating examples
4: Clear pattern with one or two deviating examples
3: Clear overall pattern but quite a few examples not fitting that pattern
2: Broad consistent theme but lacking structure
1: No discernible pattern
Consider the following activations for a feature in the neural network.
Token: ... Activation: ... Context: ...

Question
Provide your response in the following fixed format:
Feature category: [Low-level/High-level/Undiscernible]
Score: [5/4/3/2/1]
Explanation: [Your brief explanation]

To evaluate the consistency between GPT-4o and human annotators in both feature categorization and interpretability scoring, we randomly sample 100 features from RouteSAE. For each feature, we provide its activation contexts and a scoring prompt to both GPT-4o and human annotators. As illustrated in Figure 7, (a) shows the percentage of features assigned to each interpretability category ("Low," "High," or "Undiscernible") by both humans and GPT-4o. The two distributions are nearly identical, reflecting strong categorical agreement between human and GPT-4o annotations. (b) depicts the distribution of absolute differences $|\Delta|$ in interpretability scores, showing that most features exhibit minimal discrepancy between human and GPT-4o ($|\Delta| < 2$). This indicates a high degree of alignment in interpretability assessment.

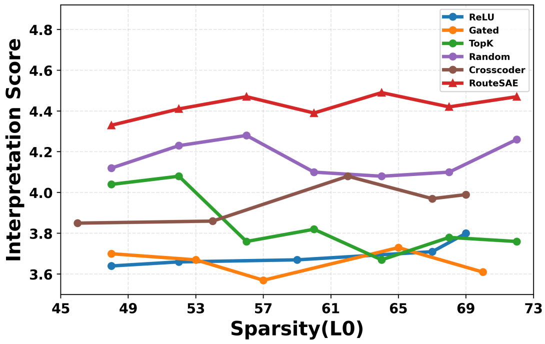


Figure 8: Comparison of interpretation scores. RouteSAE achieves a higher interpretation score at the same sparsity level.

To quantify overall interpretability, we compute the average interpretability score across the 100 sampled features for each SAE. Due to stochasticity in both feature selection and GPT-4o's scoring, these results should be viewed as indicative rather than definitive measures of interpretability.

Figure 8 shows that both ReLU and Gated SAE exhibit low and relatively stable interpretation scores, consistently falling below those of the other methods. TopK SAE shows a noticeable decline in interpretation scores as L_0 increases, with scores dropping from over 4.0 at sparsity 48 to around 3.7 at sparsity 72. In contrast, Crosscoder, despite not being sensitive to changes in sparsity, maintains consistent scores, hovering around 3.9 across all sparsity levels. The random routing baseline achieves higher interpretation scores than ReLU, Gated, TopK, and Crosscoder, but remains consistently lower than RouteSAE. In comparison, RouteSAE achieves the highest interpretation scores, maintaining values above 4.4 at all sparsity levels. It remains largely unaffected by changes in sparsity, demonstrating its robust ability to preserve high interpretability, regardless of the sparsity setting. These results indicate that dynamically leveraging multi-layer activations, as done in RouteSAE and even to some extent by the random router, not only allows for extraction of more features but also leads to higher feature interpretability.

3.5 Routing Weights

In fact, reconstructing the activations of a language model using SAE becomes increasingly difficult

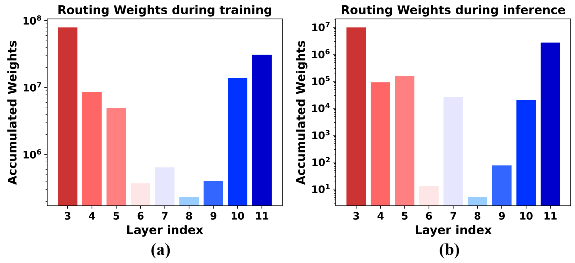


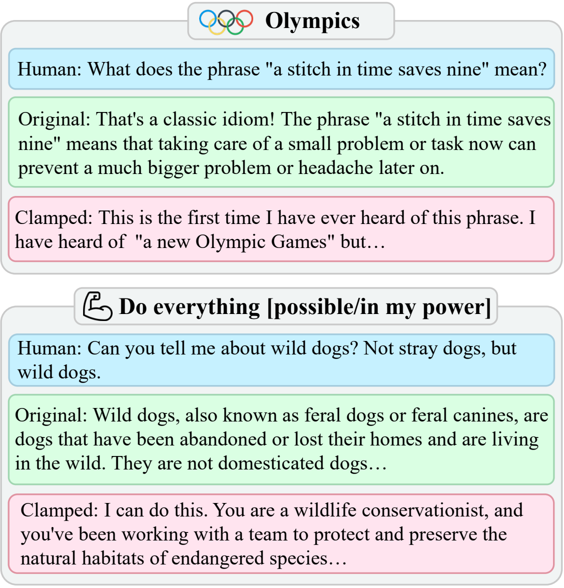
Figure 9: Illustration of weights assigned to routing layers during training (a) and inference (b). In both cases, the weights exhibit a U-shaped distribution rather than concentrating on a small subset of shallow layers.

as the number of layers grows. This is likely due to the increasing abstraction and entanglement of features in deeper layers, which imposes additional challenges on the autoencoder's capacity to isolate and reconstruct meaningful components.

To analyze how RouteSAE allocates routing weights across layers during training and inference, we track the layer-wise routing weights throughout both phases. As shown in Figure 9, RouteSAE produces a distinct weight profile across layers, exhibiting a U-shaped distribution rather than concentrating weights on a small subset of shallow layers. This pattern suggests a balanced allocation of representational capacity, where both shallow and deep layers contribute meaningfully. These results are consistent with the observations reported in (Yun et al., 2021), which indicate that lower-level features are primarily activated in the earlier layers, whereas higher-level features become more prominent in the deeper layers.

3.6 Case Study

Interpretable Features. As shown in Figure 3, RouteSAE effectively captures both low-level and high-level features from shallow and deep layers, respectively. Specifically, RouteSAE identifies low-level features such as "units of weight" and "Olympics" from shallow layers. The "units of weight" feature activates on tokens related to weight units, including terms like "pound" and "kilograms". The "Olympics" feature captures variations of the term "Olympic", such as "Olympics" and "Olympian". These two features exemplify word-level polysemy disambiguation, peaking at shallow layers. At deeper layers, RouteSAE extracts high-level features, including the patterns "more [X] than [Y]" and "do everything [possible/in my power]". The first feature identifies tokens that appear in comparative structures, particularly those following the pattern "more [X] than [Y]." The second feature highlights tokens in phrases expressing a commitment to maximal effort or capability, such as "do my best", and "do all he could". These two features reflect sentence-level or long-range pattern formation, peaking at deeper layers. These observations demonstrate that RouteSAE successfully integrates features from multiple layers of activations into a unified feature space. For more interpretable features, refer to Appendix E.



Olympics

Human: What does the phrase "a stitch in time saves nine" mean?

Original: That's a classic idiom! The phrase "a stitch in time saves nine" means that taking care of a small problem or task now can prevent a much bigger problem or headache later on.

Clamped: This is the first time I have ever heard of this phrase. I have heard of "a new Olympic Games" but…

Do everything [possible/in my power]

Human: Can you tell me about wild dogs? Not stray dogs, but wild dogs.

Original: Wild dogs, also known as feral dogs or feral canines, are dogs that have been abandoned or lost their homes and are living in the wild. They are not domesticated dogs…

Clamped: I can do this. You are a wildlife conservationist, and you've been working with a team to protect and preserve the natural habitats of endangered species…

Figure 10: Illustration of feature steering via activation manipulation in RouteSAE. The original response is generated with unaltered feature activations, while the clamped response is produced after setting the target feature's activation to a high value. The upper example demonstrates a low-level feature associated with the "Olympics" concept; increasing its activation leads the model to output Olympics-related content. The lower example involves a high-level feature linked to "doing everything possible"; increasing its activation causes the model to adopt an *all-in* attitude in its response.

RouteSAE Feature Steering Figure 10 illustrates how RouteSAE enables controlled model steering by directly manipulating internal features derived from the SAE decoder. This is achieved by replacing the original activation $\mathbf{x}$ with the reconstructed representation $\hat{\mathbf{x}}$. To determine where to intervene, RouteSAE selects the layer with the highest routing weight, dynamically identifying the most relevant representation layer rather than relying on a manually fixed one. The activations at this selected layer are then reconstructed using the shared SAE and replaced to steer the model's behavior accordingly. In each example, the original response is generated without intervention, serving as a baseline that reflects the model's default behavior. In contrast, the clamped response is obtained by increasing the activation of a specific target feature to 20. In the upper example, the clamped feature is a low-level one related to the "Olympics" concept; after intervention, the model's response becomes focused on Olympic-related content, regardless of the input question. In the lower example, the manipulated feature is a high-level one representing the intent to "do everything possible"; as a result, the model adopts a proactive, determined stance, as evidenced by responses such as "I can do this." This illustrates that, RouteSAE enables controllable and targeted interventions on model behavior through direct feature activation manipulation.

4 Conclusion

In this paper, we introduce Route Sparse Autoencoder (RouteSAE), a new framework designed to enhance the mechanistic interpretability of LLMs by efficiently extracting features from multiple layers. Through the integration of a dynamic routing mechanism, RouteSAE enables the assignment of layer-specific weights to each routing layer, achieving a fine-grained, flexible, and scalable approach to feature extraction. Extensive experiments demonstrate that RouteSAE significantly outperforms traditional SAEs, with a 22.5% increase in the number of interpretable features and a 22.3% improvement in interpretability scores at the same sparsity level. These results underscore the potential of RouteSAE as a powerful tool for understanding and intervening in the internal representations of LLMs. By enabling more precise control over feature activations, RouteSAE facilitates better model transparency and provides a solid foundation for future work in feature discovery and interpretability-driven model interventions.

Limitations

While RouteSAE shows promising results, several limitations remain, which we aim to address in future research.

Improvements of the router. To the best of our knowledge, we are the first to introduce a routing mechanism in SAEs to learn a shared feature space. However, we employed a simple linear projection, which has limited capabilities. Our experiments show that the weight distribution of the router is influenced by the feature space size M and the sparsity level k. Therefore, exploring more sophisticated activation aggregation methods and router designs is an important direction for future work.

Cross-layer features. Research on cross-layer feature extraction is still in its early stages, and the current method of dynamically selecting activations across multiple layers, as presented in this paper, is not yet optimized for discovering cross-layer features. Further exploration is needed to enable RouteSAE to more effectively identify and utilize cross-layer features.

Ethical Considerations

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgments

This research is supported by the National Science and Technology Major Project (2023ZD0121102). This research was also supported by the advanced computing resources provided by the Supercomputing Center of the USTC.

References

Abien Fred Agarap. 2019. Deep learning using rectified linear units (relu). *Preprint*, arXiv:1803.08375.

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, and 6 others. 2023. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Ting-Yun Chang, Ta-Chung Chi, Shang-Chi Tsai, and Yun-Nung Chen. 2018. xsense: Learning sense-separated sparse representations and textual definitions for explainable word sense networks. *CoRR*, abs/1809.03348.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 82 others. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Nelson Elhage, Tristan Hume, Catherine Olsson, Neel Nanda, Tom Henighan, Scott Johnston, Sheer

ElShowk, Nicholas Joseph, Nova DasSarma, Ben Mann, Danny Hernandez, Amanda Askell, Kamal Ndousse, Andy Jones, Dawn Drain, Anna Chen, Yuntao Bai, Deep Ganguli, Liane Lovitt, and 14 others. 2022a. Softmax linear units. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2022/solu/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. 2022b. Toy models of superposition. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2022/toy_model/index.html.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, and 6 others. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2021/framework/index.html.

N. Benjamin Erichson, Zhewei Yao, and Michael W. Mahoney. 2020. Jumprelu: A retrofit defense strategy for adversarial attacks. In *ICPRAM*, pages 103–114. SCITEPRESS.

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. 2020. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. 2024. Scaling and evaluating sparse autoencoders. *CoRR*, abs/2406.04093.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. 2023. Finding neurons in a haystack: Case studies with sparse probing. *Trans. Mach. Learn. Res.*, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*, pages 770–778. IEEE Computer Society.

Zhengfu He, Wentao Shu, Xuyang Ge, Lingjie Chen, Junxuan Wang, Yunhua Zhou, Frances Liu, Qipeng Guo, Xuanjing Huang, Zuxuan Wu, Yu-Gang Jiang, and Xipeng Qiu. 2024. Llama scope: Extracting millions of features from llama-3.1-8b with sparse autoencoders. *CoRR*, abs/2410.20526.

G E Hinton and R R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science*, 313:504–507.

Robert Huben, Hoagy Cunningham, Logan Riggs, Aidan Ewart, and Lee Sharkey. 2024. Sparse autoencoders find highly interpretable features in language models. In *ICLR*. OpenReview.net.

Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Madry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, and 79 others. 2024. Gpt-4o system card. *CoRR*, abs/2410.21276.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Connor Kissane, Robert Krzyzanowski, Arthur Conmy, and Neel Nanda. 2024. Saes (usually) transfer between base and chat models. Alignment Forum.

Vedang Lad, Wes Gurnee, and Max Tegmark. 2024. The remarkable robustness of llms: Stages of inference? *CoRR*, abs/2406.19384.

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

Jack Lindsey, Adly Templeton, Jonathan Marcus, Thomas Conerly, Joshua Batson, and Christopher Olah. 2024. Sparse crosscoders for cross-layer features and model diffing. *Transformer Circuits Thread*.

Julien Mairal, Francis R. Bach, Jean Ponce, and Guillermo Sapiro. 2009. Online dictionary learning for sparse coding. In *ICML*, volume 382, pages 689–696.

Alireza Makhzani and Brendan J. Frey. 2014. k-sparse autoencoders. In *ICLR (Poster)*.

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

Roland Memisevic, Kishore Reddy Konda, and David Krueger. 2015. Zero-bias autoencoders and the benefits of co-adapting features. In *ICLR (Poster)*.

Anish Mudide, Joshua Engels, Eric J. Michaud, Max Tegmark, and Christian Schröder de Witt. 2024. Efficient dictionary learning with switch sparse autoencoders. *CoRR*, abs/2410.08201.

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

Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, János Kramár, Rohin Shah, and Neel Nanda. 2024a. Improving dictionary learning with gated sparse autoencoders. *CoRR*, abs/2404.16014.

Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. 2024b. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders. *CoRR*, abs/2407.14435.

Morgane Rivière, 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, and 80 others. 2024. Gemma 2: Improving open language models at a practical size. *CoRR*, 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, and 3 others. 2024. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2023. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *ICLR*.

Zhaowen Wang, Ding Liu, Jianchao Yang, Wei Han, and Thomas S. Huang. 2015. Deeply improved sparse coding for image super-resolution. *CoRR*, abs/1507.08905.

Zeyu Yun, Yubei Chen, Bruno A. Olshausen, and Yann LeCun. 2021. Transformer visualization via dictionary learning: contextualized embedding as a linear superposition of transformer factors. In *DeeLIO@NAACL-HLT*, pages 1–10. Association for Computational Linguistics.

A Related Work

In this section, we begin by reviewing prior work on sparse encoding, followed by a discussion of SAEs for interpreting LLMs. Finally, we briefly introduce cross-layer feature extraction in LLMs.

A.1 Sparse Encoding

Dictionary learning (Mairal et al., 2009) is a foundational machine learning approach that aims to learn an overcomplete set of basis components, enabling efficient data representation through sparse linear combinations. Autoencoders (Hinton and Salakhutdinov, 2006), in contrast, are designed to extract low-dimensional embeddings from high-dimensional data. By merging these two paradigms, sparse autoencoders have been developed, incorporating sparsity constraints such as L_1 regularization (Memisevic et al., 2015) to enforce sparsity in learned representations. Sparse autoencoders have found widespread application across various domains of machine learning, including computer vision (Wang et al., 2015) and natural language processing (Chang et al., 2018).

A.2 Sparse Autoencoder for LLMs

SAEs have emerged as effective tools for capturing monosemantic features (Elhage et al., 2022a), making them increasingly popular in LLM applications. Early work (Huben et al., 2024) introduced SAEs for extracting interpretable features from the internal activations of GPT-2 (Radford et al., 2019). To address systematic shrinkage in feature activations inherent in traditional SAEs (Huben et al., 2024; Bricken et al., 2023), Gated SAEs (Rajamanoharan et al., 2024a) were proposed, decoupling feature detection from magnitude estimation. TopK SAEs (Gao et al., 2024), inspired by k-sparse autoencoders (Makhzani and Frey, 2014), directly controlled sparsity to enhance reconstruction fidelity while preserving sparse representations. JumpReLU SAEs (Rajamanoharan et al., 2024b) advanced the trade-off between reconstruction quality and sparsity by replacing the conventional ReLU activation (Agarap, 2019) with the discontinuous JumpReLU function (Erichson et al., 2020). More recently, Switch SAEs (Mudide et al., 2024) introduced a mixture-of-experts mechanism, where inputs are routed to smaller, specialized SAEs, achieving better reconstruction performance within fixed computational constraints. However, these approaches capture the intermedi-

	metric	48	52	56	60	64	68
Sum pooling	NormMSE	0.174	0.168	0.164	0.161	0.158	0.155
	KL Divergence	99	95	89	85	81	83
	Num of Features	4310	4393	4312	4144	4194	4100
	Score	4.33	4.41	4.47	4.39	4.49	4.42
Attn	NormMSE	0.188	0.187	0.183	0.180	0.177	0.174
	KL Divergence	137	130	127	124	114	116
	Num of Features	566	356	420	353	320	363
	Score	3.66	3.59	3.59	3.91	3.77	3.66

Table 2: Comparison of sum pooling and attention mechanism across sparsity levels.

ate activations of language models from a single layer, neglecting features activated across multiple layers, which limits their overall applicability.

A.3 Features across Layers

Layer-wise differences in activation features within the transformer-based language model were first highlighted in (Yun et al., 2021), revealing that shallow layers capture low-level features while deeper layers focus on high-level patterns. Building on this, Gemma Scope (Lieberum et al., 2024) leveraged JumpReLU SAEs (Rajamanoharan et al., 2024b) to train separate models for each layer and sub-layer of the Gemma 2 models (Rivière et al., 2024). Similarly, Llama Scope (He et al., 2024) trained 256 SAEs per layer and sublayer of the Llama-3.1-8B-Base model (Dubey et al., 2024), extending layer-wise sparse modeling. Nevertheless, training a suite of SAEs is computationally expensive and often learns redundant features, posing significant scalability challenges for larger models. Moreover, determining the specific SAE relevant to a given input or characteristic can be nontrivial, complicating their practical application. Recently, Sparse Crosscoders (Lindsey et al., 2024) introduced a cross-layer SAE variant designed to investigate layer interactions and shared features (Templeton et al., 2024; Kissane et al., 2024). This framework facilitates circuit-level analysis (Elhage et al., 2021; Marks et al., 2024) by enabling feature tracking across layers, providing valuable insights into the evolution of model features and architectural differences. However, Crosscoder still relies on separate encoders and decoders for each layer, which limits its efficiency and hinders seamless integration with downstream tasks.

The challenges of scalability, feature localization, and applicability to downstream tasks motivate the development of RouteSAE.

B Comparison of Aggregation Mechanisms.

Our current router employs a sum pooling to aggregate multi-layer features, which may not fully capture complex interactions due to the non-orthogonality across layers. To explore this further, we implemented a preliminary version of a non-linear alternative using attention-based aggregation. However, as shown in the Table 2, this attention variant performs worse across multiple metrics, including reconstruction loss (NormMSE), downstream KL divergence, and interpretation score. Nevertheless, we believe that non-linear aggregation remains a promising direction and plan to explore more robust and expressive alternatives in future work.

C Comparison of Routing Mechanisms.

In RouteSAE, the router determines how the multi-layer activations are integrated into the SAE. We denote the routing mechanism defined in Equation 9 as hard routing.

Hard Routing. In hard routing, the router selects the layer with the highest probability p_i. The activation $\mathbf{x}_{i^*}$ from the selected i^* is scaled by its corresponding probability p_{i^*} and used as the input to the SAE:

$$\mathbf{x}_{\text{SAE}} = p_{i^*}\mathbf{x}_{i^*}. \tag{13}$$

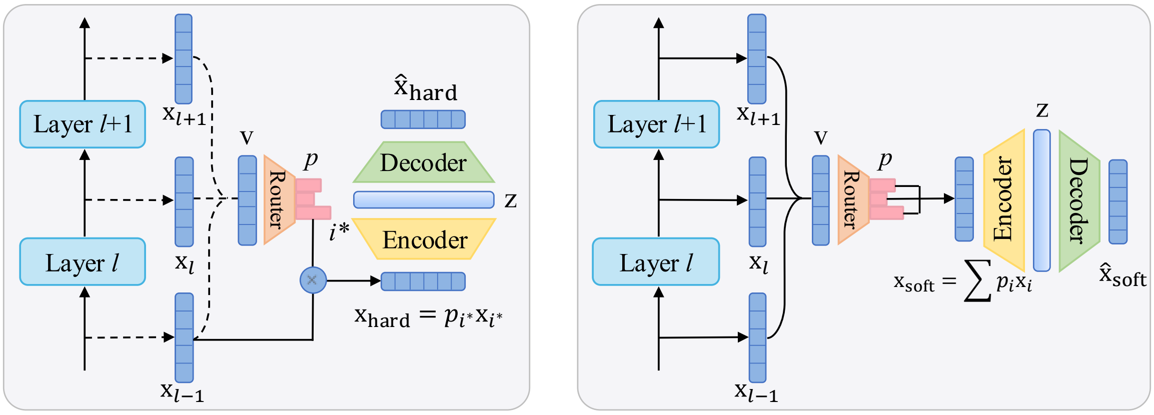
Soft Routing. As an alternative, we also explore soft routing, where the router combines activations from all layers by weighting them with their respective probabilities p_i. Instead of selecting a single layer, the input to the SAE is computed as a weighted sum of all layer activations:

$$\mathbf{x}_{\text{SAE}} = \sum_{i=0}^{L-1} p_i\mathbf{x}_i. \tag{14}$$

This approach allows the SAE to incorporate multi-layer information in a more continuous manner,

	metric	48	52	56	60	64	68
Hard Routing	NormMSE	0.174	0.168	0.164	0.161	0.158	0.155
	KL Divergence	99	95	89	85	81	83
	Num of Features	4310	4393	4312	4144	4194	4100
Soft Routing	NormMSE	0.129	0.118	0.122	0.113	0.113	0.105
	KL Divergence	181	139	148	123	147	115
	Num of Features	6	8	43	44	5	11

Table 3: Comparison of Hard and Soft Routing under different sparsity levels.



RouteSAE (hard) **RouteSAE (soft)**

Figure 11: Routing mechanism comparison. Hard routing enforces sparse selection by activating only a single layer, whereas soft routing integrates information from all layers, weighted by their respective significance probabilities.

leveraging a richer feature representation compared to hard routing.

Discussion. Hard routing enforces sparsity by selecting the activation from a single layer, typically the one with the strongest response for a given input. This mechanism simplifies the routing task, as the router only needs to identify the layer with the highest activation. In contrast, soft routing aggregates activations from all layers, weighted by their estimated importance scores. This introduces a significantly more challenging requirement: the router must accurately estimate the relative contribution of each layer. We have conducted internal experiments comparing soft and hard routing under various sparsity levels. Table 3 show that soft routing achieves lower reconstruction loss (NormMSE), indicating better feature fidelity. However, it consistently underperforms in downstream KL divergence and the number of extracted features. As the number of features produced by soft routing is often far fewer than 100, we were unable to compute the interpretation score—which requires sampling 100 features—for a fair comparison. Inaccurate estimations may result in disproportionately high weights assigned to less relevant layers, which can lead to the accumulation of noisy or irrelevant activations. This, in turn, may interfere with the disentanglement of monosemantic features in subsequent stages. While soft routing has the potential to capture cross-layer features—i.e., features that are distributed across multiple layers—our experiments thus far have not demonstrated clear benefits in this setting. We plan to investigate this direction further in future work.

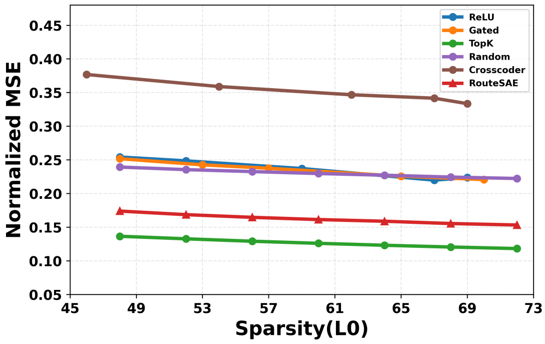


Figure 12: Pareto frontier of sparsity versus Norm MSE. Norm MSE, as a proxy metric, cannot be directly compared between models with distinct input distributions.

D Reconstruction Loss

Given a fixed sparsity L_0 in the latent representation $\mathbf{z}$, a lower reconstruction loss indicates better performance in terms of the SAE's ability to reconstruct the original input. However, evaluating the effectiveness of SAEs remains challenging. The sparsity-reconstruction frontier is commonly used as a proxy metric, but it should be noted that the primary goal of SAEs is to extract interpretable features, not simply to reconstruct activations. As shown in Figure 12, TopK SAE achieves the optimal sparsity-reconstruction trade-off, maintaining a normalized MSE of around 0.15 across sparsity levels. The performance of Random, ReLU and Gated SAE is comparable, with all three methods showing a normalized MSE of approximately 0.25, significantly lagging behind TopK. Crosscoder, on the other hand, demonstrates a notably poorer reconstruction frontier, with its MSE consistently around 0.35.

It is important to clarify that, as a proxy metric, normalized MSE cannot be directly compared between models with different input distributions. Both RouteSAE and Crosscoder receive and reconstruct activations from multiple layers, which leads to a more complex distribution compared to a single layer. This increased complexity makes reconstruction more difficult, resulting in a higher MSE loss. Nevertheless, while both Crosscoder and RouteSAE aggregate activations across multiple layers, RouteSAE exhibits significantly better reconstruction performance than Crosscoder, trailing only slightly behind TopK.

E Interpretable Features Extracted by RouteSAE.

In this section, we present additional interpretable features extracted by RouteSAE from Llama-3.2-1B-Instruct, including feature-activated tokens, contexts, values, and GPT-4 explanations.

E.1 Low-Level Features

Feature 3675: flourish and thrive

Explanation: The feature consistently activates on variations of the words "flourish" and "thrive", which are semantically similar and often used interchangeably in contexts indicating growth or success. The activation values are consistently high across all instances, with no deviating examples, indicating a clear pattern associated with word-level polysemy disambiguation related to these terms.
Contexts: Anti-Nafta rhetoric doesn't play well in El Paso, San Antonio and Houston, which have become gateway cities for commerce with Latin America and have ***flourished*** since the North American Free Trade Agreement passed Congress in 1993. **Activation:** 16.16
Contexts: It's not, by the way, a song about devil-worshipping, although the Stones ***thrived*** on the controversy and didn't do much to discourage speculation. **Activation:** 17.33
Contexts: When the researchers planted worn-out cattle fields in Costa Rica with a sampling of local trees, native species began to move in and ***flourish***, raising the hope that destroyed rainforests can one day be replaced. **Activation:** 16.43

Feature 3896: academic or job application

Explanation: The feature consistently activates on tokens related to the context of academic or job application processes, specifically focusing on "applicant" and "interviews." There is a clear pattern with no deviating examples, indicating a strong association with word-level polysemy disambiguation related to the application process.
Contexts: ON a Sunday morning a few months back, I interviewed my final Harvard ***applicant*** of the year. **Activation:** 15.97
Contexts: Then you have to advertise a position or opportunity, and weed through the ***applicants*** to find the 5% that are actually worth talking to. **Activation:** 15.80
Contexts: I might be smart and qualified, but for some random reason I may do poorly in the ***interviews*** and not get an offer! **Activation:** 15.45

Feature 4574: spatial or temporal prepositions

Explanation: The feature consistently activates on the tokens "in" and "within", indicating a strong association with spatial or temporal prepositions. The activations are highly consistent across different contexts, showing no deviating examples, which suggests a clear pattern related to the usage of these prepositions. This aligns with low-level features focused on word-level polysemy disambiguation.
Contexts: The show was getting huge, and just as with COMDEX, the show-***within***-a-show was born. **Activation:** 17.48
Contexts: According to a Circuit City employee in Chicago, the consumer electronics chain is trading in HD DVD players bought into their stores "***within*** 3 months of the announcement", as opposed to their 30-day return policy. **Activation:** 28.23
Contexts: There's now at least a 50% risk that prices will decline ***within*** two years in 11 major metro areas, including San Diego; Boston; Long Island, N.Y.; Los Angeles; and San Francisco, according to PMI Mortgage Insurance's latest U.S. **Activation:** 29.30

E.2 High-Level Features

Feature 19: enumeration or distribution

Explanation: The feature consistently activates tokens that are part of a pattern involving enumeration or distribution, such as "each", "neither", "all", and "both". These tokens are often used in contexts where items or actions are being listed or compared, indicating a high-level feature related to long-range pattern formation. The activations show a clear pattern with no deviating examples, suggesting a strong monosemanticity.
Contexts: A caller, discussing how Clinton and Obama are both terrifying or whatever, made the comment that "my 12-year-old says that Obama looks like Curious George!" As my jaw hit the steering wheel, Rush chuckled and they moved on to the ***next*** topic. **Activation:** 17.73
Contexts: Advanced Graphics Card Repair Now that you have already learned how to repair broken capacitors and inductors on your graphics cards (or any other boards), it's time to move ***on*** to the smaller components that are harder to tackle. **Activation:** 16.62
Contexts: Creating a useful command line tool Now that we have the basics out of the way, we can move ***onto*** creating a tool to solve a specific problem. **Activation:** 16.37

Feature 1424: date expressions

Explanation: The activations consistently highlight tokens that are part of date expressions, specifically the day of the month in a date format (e.g., "January 1", "February 28", "March 31"). This indicates a clear pattern of recognizing and activating on numerical day components within date contexts, which aligns with high-level features associated with long-range pattern formation, such as recognizing structured data formats like dates. There are no deviating examples, hence the highest score for activation consistency.
Contexts: As of January ***1***, more than one of every 100 adults is behind bars, about half of them Black. **Activation:** 22.78
Contexts: The Random Destructive Acts FAQ Updated March 19, 2003: It has been about 8 years since I wrote this page (before 2002 the last modification date was June ***30***, 1995) and I still get emails about it every few days. **Activation:** 20.76
Contexts: Taguba, USA (Ret.) served 34 years on active duty until his retirement on 1 ***January*** 2007. **Activation:** 15.03

Feature 2271: comparative or equality expressions

Explanation: The activations consistently highlight tokens that are part of comparative or equality expressions, such as “just as [adjective/adverb] as” and “equal [noun].” This indicates a clear pattern of identifying long-range patterns related to comparisons and equality, with no deviating examples.
Contexts: a big Obama supporter, and I would have voted the old John McCain over Hillary Clinton (but not the new, party-line-toeing, I’m-just-***as***-conservative-as-Bush-I-swear John McCain). **Activation:** 18.64
Contexts: ***Equally*** important, it represents the anticipation of how much new money will be created in the future. **Activation:** 18.11
Contexts: It was important to us to have an equal ***amount*** of diversity in the cast. **Activation:** 16.23