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]

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).

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, prior study (Yun et al., 2021) demonstrated that the activation strength of features in this sparse feature space exhibits distinct distribution patterns across layers [1]. As illustrated in Figure 1, low-level features (*e.g.,* "Crushed things" and "Europe") representing word-level polysemy disambiguation peak in activation strength within shallow layers and gradually decline in deeper layers. Conversely, high-level features exhibit increasing activation strength as depth increases, corresponding to sentence-level or longer-range pattern formation, such as "Enumeration" and "One of the [number/quantifier]" [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 (Lindsey et al., 2024)[3] 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

---

[*]Equal contribution [1]School of Artificial Intelligence and Data Science, University of Science and Technology of China, Hefei, China [2]Douyin Co., Ltd. Shenzhen Bay Innovation and Technology Center, Gaoxin South 9th Road, Nanshan District, Shenzhen, Guangdong, 518067, China. Correspondence to: Guojun Ma <maguojun@bytedance.com>, Xiang Wang <xiangwang1223@gmail.com>.

Preprint

---

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

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

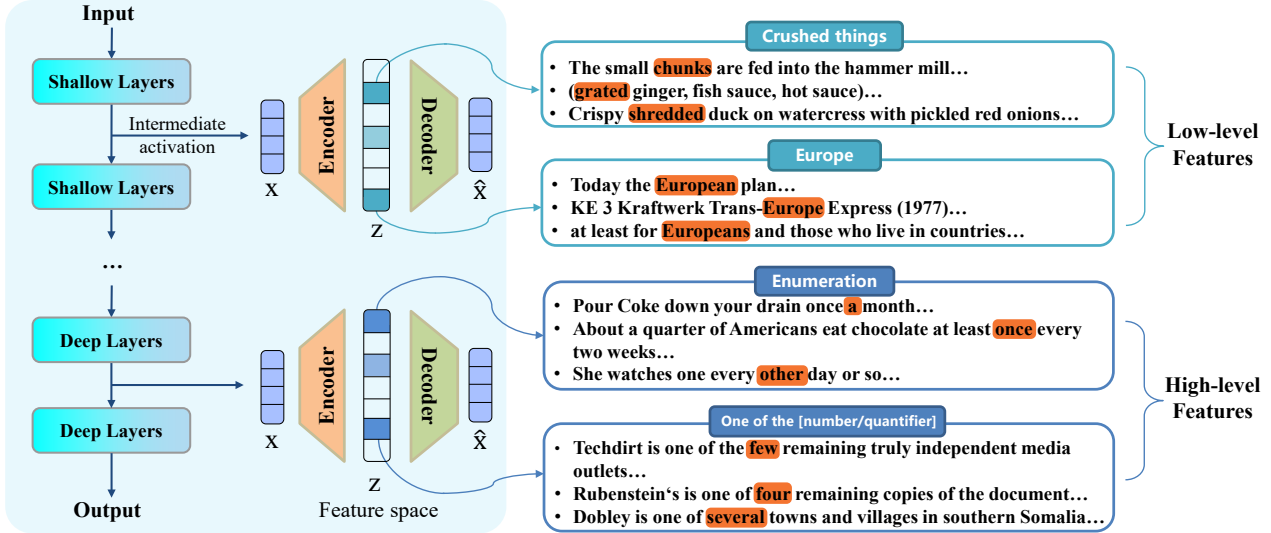[3]Currently a conceptual framework without complete experimental validation.

Figure 1. Illustration of low- and high-level features, whose activation strength usually peaks at shallow and deep layers, respectively. Low-level features (*e.g.,* "Crushed things" and "Europe") representing word-level polysemy disambiguation peak in activation strength within shallow layers, while high-level features exhibit increasing activation strength as depth increases, corresponding to sentence-level or longer-range pattern formation (*e.g.,* "Enumeration" and "One of the [number/quantifier]").

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 limits the ability to adjust feature activations for specific LLM outputs in a controlled manner, reducing its flexibility for tasks that require precise feature-level intervention, *e.g.,* feature steering (Templeton et al., 2024).

To address these challenges, we propose a new framework — **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 conducted 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 integrating a router into the SAE framework significantly improves the interpretability. At an equivalent sparsity level of 64, RouteSAE achieves a 22.5% increase in the number of interpretable features and a 22.3% improvement in interpretation scores. Additionally, we present a case study to highlight the practical advantages of the routing mechanism, illustrating how it improves feature interpretability by capturing features across multiple layers.

2. 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 works on cross-layer feature extraction in LLMs.

2.1. 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 & 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).
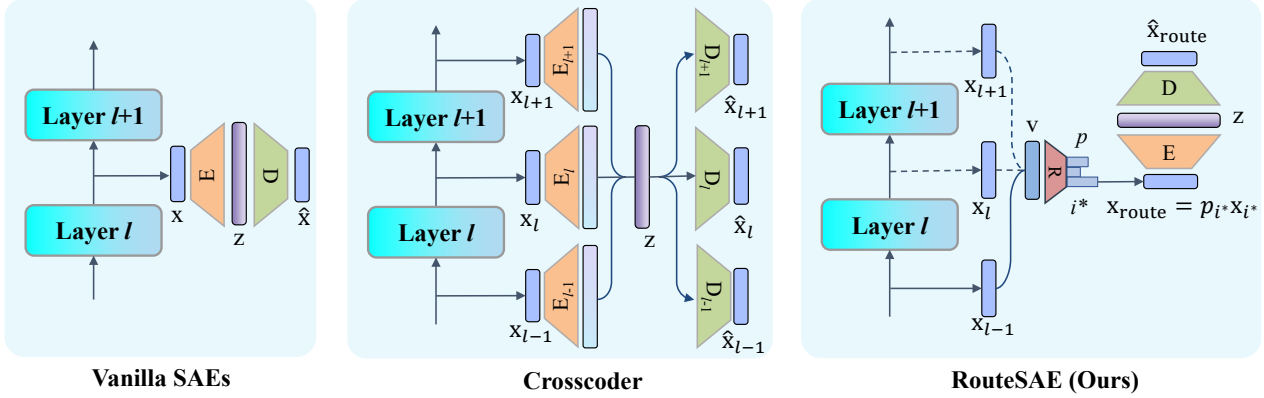
Figure 2. Comparison of vanilla single-layer SAE, Crosscoder, and RouteSAE. Most existing SAEs belong to the vanilla SAE category, where features are extracted from the activation of a single layer. Crosscoder relies on separate encoders and decoders for each layer. RouteSAE incorporates a lightweight router to dynamically integrate multi-layer residual stream activations.

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

2.3. 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 identified in the works discussed in this section motivate the development of Route-SAE, a framework designed to overcome these limitations via a routing mechanism.

3. Methodology

In this section, we first provide a brief overview of SAEs, then introduce our proposed Route Sparse autoencoder (RouteSAE) in detail, highlighting its architecture, dynamic routing mechanism, and advantages in addressing the challenges of scalability and feature-level controllability.

3.1. 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
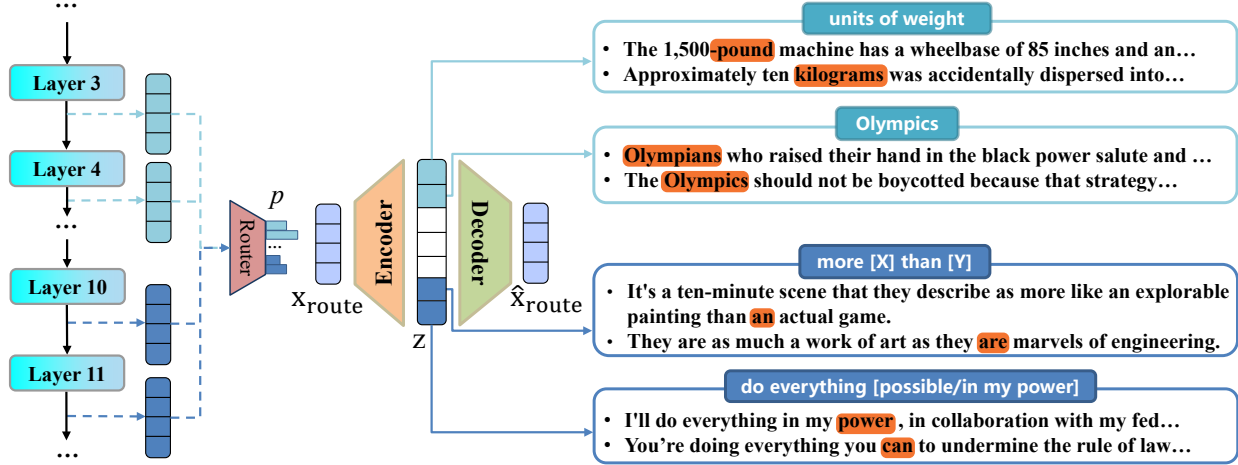
Figure 3. RouteSAE employs a lightweight router to dynamically integrate activations from multiple residual stream layers, effectively disentangling them into a shared feature space. It enables the model to capture features across different layers — low-level features such as "units of weight" and "Olympics" from shallow layers, and high-level features like "more [X] than [Y]" and "do everything [possible/in my power]" from deeper layers.

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 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 compared to other SAE variants.

3.2. Route Sparse Autoencoder

As shown in Figure 1, 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, we propose RouteSAE, which incorporates a lightweight router to dynamically integrate multi-layer residual stream activations 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 — an approach that 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:

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

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

$$i^* = \arg\max_i p_i, \quad i = 0, 1, \ldots, L-1. \quad (8)$$

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:

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

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

$$\mathbf{z}_{\text{route}} = \text{TopK}(\mathbf{W}_{\text{enc}}(\mathbf{x}_{\text{route}} - \mathbf{b}_{\text{pre}})) \quad (10)$$
$$\hat{\mathbf{x}}_{\text{route}} = \mathbf{W}_{\text{dec}}\mathbf{z}_{\text{route}} + \mathbf{b}_{\text{pre}}. \quad (11)$$

Finally, we minimize the reconstruction MSE:

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

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, including ReLU SAE (Huben et al., 2024; Bricken et al., 2023), Gated SAE (Rajamanoharan et al., 2024a) and JumpReLU SAE (Rajamanoharan et al., 2024b). By employing a shared SAE, RouteSAE establishes a unified feature space across activations from all routing layers. This ensures consistent and coherent feature representations, thereby enhancing the disentanglement of high-dimensional features and improving interpretability.

4. Experiments

In this section, we first outline the experimental setup, followed by the evaluation of RouteSAE. In this paper, we

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.

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 effectively captures both low-level features from shallow layers and high-level features from deep layers.

4.1. Setup

Inputs. We train all SAE models on the residual streams (He et al., 2016) of the Llama-3.2-1B-Instruct (Dubey et al., 2024). 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 the methodology outlined in (Gao et al., 2024).

Hyperparameters. For all SAEs, we use the Adam optimizer (Kingma & 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
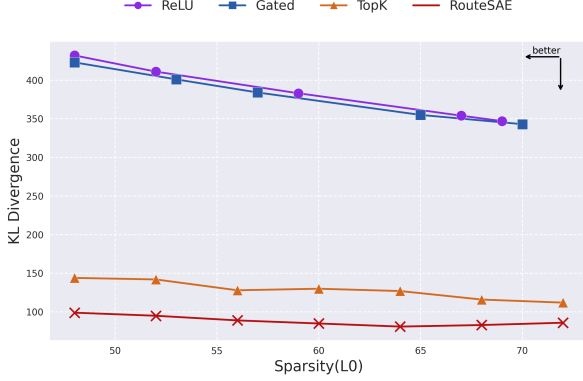
Figure 4. Pareto frontier of sparsity versus KL divergence. Route-SAE achieves a lower KL divergence at the same sparsity level.

Figure 5. Comparison of the interpretable feature number. Route-SAE extracts the most interpretable features at the same threshold.

maintain unit length throughout training.

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). 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*.

4.2. 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 KL divergence indicating that the extracted features are highly relevant for language modeling.

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.

Notably, RouteSAE outperforms all other methods, achieving the best performance by maintaining a lower KL divergence at the same sparsity level. It even surpasses TopK SAE, demonstrating superior efficiency. Since Crosscoder generates multiple reconstructed representations $\hat{\mathbf{x}}$, it is excluded from this comparison, as its application to this task
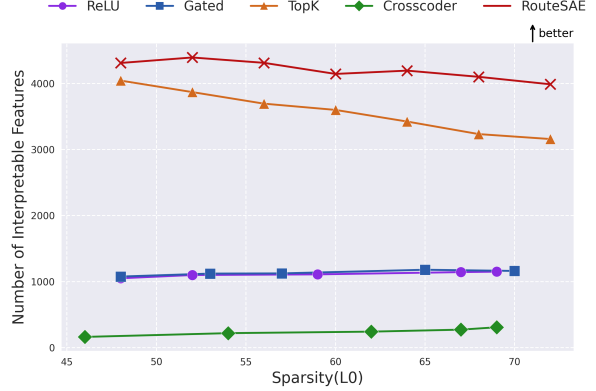
is not straightforward.

4.3. 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 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, 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. Increasing the threshold reduces the number of retained features but enhances interpretability. The threshold thus acts as a trade-off between the quantity and quality of interpretable features. In this section, we set the threshold to 15, striking a balance between feature interpretability and quantity. 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 5, 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. Route-SAE 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
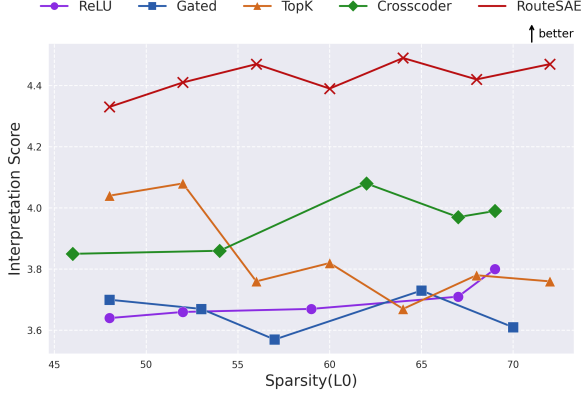
Figure 6. Comparison of interpretation scores. RouteSAE achieves a higher interpretation score at the same sparsity level.

Figure 7. Pareto frontier of sparsity versus Norm MSE. Norm MSE, as a proxy metric, cannot be directly compared between models with distinct input distributions.

features as L_0 increases, while TopK SAE exhibits a more pronounced reduction. 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.

4.4. Interpretation Score

Despite the feature screening in Section 4.3, 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 in Appendix B, 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.

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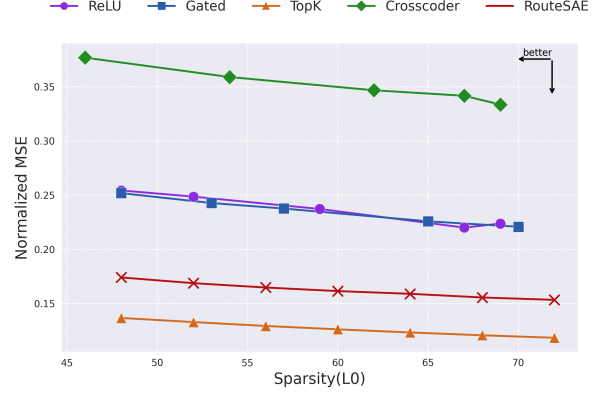
each SAE. Due to stochasticity in both feature selection and GPT-4o's scoring, these results should be viewed as indicative rather than definitive measures of interpretability.

Figure 6 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. 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.

4.5. 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 7, TopK SAE achieves the optimal sparsity-reconstruction trade-off, maintaining a normalized MSE of around 0.15 across sparsity levels. The performance of ReLU and Gated SAE is comparable, with both 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 dif-

ferent 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. RouteSAE maintains a normalized MSE of around 0.18, demonstrating its ability to handle the complexities of multi-layer reconstruction.

4.6. Case Study

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 3 and 4. The "units of weight" feature activates on tokens related to weight units, including terms like "pound", "kilograms", and "pounds". The "Olympics" captures variations of the term "Olympic", such as "Olympics", "Olympian", and related expressions. These two features exemplify word-level polysemy disambiguation, peaking at shallow layers. At deeper layers, specifically layers 10 and 11, 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 everything [possible/in my power]", "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 align with those in (Yun et al., 2021), demonstrating that RouteSAE successfully integrates features from multiple layers of activations into a unified feature space. For more interpretable features, refer to Appendix C.

4.7. Routing Weights

To investigate the distribution of weights across layers during the training process of RouteSAE, we track the routing weights for each layer throughout training. Figure 8 illustrates that RouteSAE exhibits distinct accumulated weights for each layer, forming an overall U-shaped trend. This observation aligns with findings in (Yun et al., 2021), where the activation strength of low-level features peaks in shallow layers and gradually declines in deeper layers, while high-level features show an increasing activation strength as the depth of the layer increases.
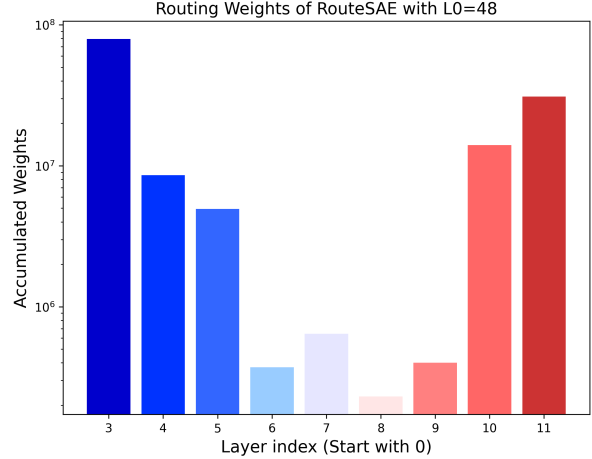

Figure 8. Illustration of weights for routing layers.

5. Limitation and Future Work

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.

Feature steering. In Vanilla SAEs, the output of the language model can be manipulated by clamping the activation values of specific features. While RouteSAE has shown strong results in terms of downstream KL divergence and the quantity and quality of interpretable features, feature steering remains a challenge. We plan to investigate methods to improve this aspect in future iterations of RouteSAE.

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.

6. 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.

Impact Statement

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.

References

Agarap, A. F. Deep learning using rectified linear units (relu), 2019. URL https://arxiv.org/abs/1803.08375.

Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N., Anil, C., Denison, C., Askell, A., Lasenby, R., Wu, Y., Kravec, S., Schiefer, N., Maxwell, T., Joseph, N., Hatfield-Dodds, Z., Tamkin, A., Nguyen, K., McLean, B., Burke, J. E., Hume, T., Carter, S., Henighan, T., and Olah, C. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Chang, T., Chi, T., Tsai, S., and Chen, Y. xsense: Learning sense-separated sparse representations and textual definitions for explainable word sense networks. *CoRR*, abs/1809.03348, 2018.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravanku-mar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Rozière, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Al-lonsius, D., Song, D., Pintz, D., Livshits, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I. M., Misra, I., Evtimov, I., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K. V., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., and et al. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024.

Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., DasSarma, N., Drain, D., Ganguli, D., Hatfield-Dodds, Z., Hernandez, D., Jones, A., Kernion, J., Lovitt, L., Ndousse, K., Amodei, D., Brown, T., Clark, J., Kaplan, J., McCandlish, S., and Olah, C. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Elhage, N., Hume, T., Olsson, C., Nanda, N., Henighan, T., Johnston, S., ElShowk, S., Joseph, N., DasSarma, N., Mann, B., Hernandez, D., Askell, A., Ndousse, K., Jones, A., Drain, D., Chen, A., Bai, Y., Ganguli, D., Lovitt, L., Hatfield-Dodds, Z., Kernion, J., Conerly, T., Kravec, S., Fort, S., Kadavath, S., Jacobson, J., Tran-Johnson, E., Kaplan, J., Clark, J., Brown, T., McCandlish, S., Amodei, D., and Olah, C. Softmax linear units. *Transformer Circuits Thread*, 2022a. https://transformer-circuits.pub/2022/solu/index.html.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., Grosse, R., McCandlish, S., Kaplan, J., Amodei, D., Wattenberg, M., and Olah, C. Toy models of superposition. *Transformer Circuits Thread*, 2022b. https://transformer-circuits.pub/2022/toy_model/index.html.

Erichson, N. B., Yao, Z., and Mahoney, M. W. Jumprelu: A retrofit defense strategy for adversarial attacks. In *ICPRAM*, pp. 103–114. SCITEPRESS, 2020.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S., and Leahy, C. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Gao, L., la Tour, T. D., Tillman, H., Goh, G., Troll, R., Radford, A., Sutskever, I., Leike, J., and Wu, J. Scaling and evaluating sparse autoencoders. *CoRR*, abs/2406.04093, 2024.

Gurnee, W., Nanda, N., Pauly, M., Harvey, K., Troitskii, D., and Bertsimas, D. Finding neurons in a haystack: Case studies with sparse probing. *Trans. Mach. Learn. Res.*, 2023, 2023.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, pp. 770–778. IEEE Computer Society, 2016.

He, Z., Shu, W., Ge, X., Chen, L., Wang, J., Zhou, Y., Liu, F., Guo, Q., Huang, X., Wu, Z., Jiang, Y., and Qiu, X. Llama scope: Extracting millions of features from llama-3.1-8b with sparse autoencoders. *CoRR*, abs/2410.20526, 2024.

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

Huben, R., Cunningham, H., Riggs, L., Ewart, A., and Sharkey, L. Sparse autoencoders find highly interpretable features in language models. In *ICLR*. OpenReview.net, 2024.

Hurst, A., Lerer, A., Goucher, A. P., Perelman, A., Ramesh, A., Clark, A., Ostrow, A., Welihinda, A., Hayes, A., Radford, A., Madry, A., Baker-Whitcomb, A., Beutel, A., Borzunov, A., Carney, A., Chow, A., Kirillov, A., Nichol, A., Paino, A., Renzin, A., Passos, A. T., Kirillov, A., Christakis, A., Conneau, A., Kamali, A., Jabri, A., Moyer, A., Tam, A., Crookes, A., Tootoonchian, A., Kumar, A., Vallone, A., Karpathy, A., Braunstein, A., Cann, A., Codispoti, A., Galu, A., Kondrich, A., Tulloch, A., Mishchenko, A., Baek, A., Jiang, A., Pelisse, A., Woodford, A., Gosalia, A., Dhar, A., Pantuliano, A., Nayak, A., Oliver, A., Zoph, B., Ghorbani, B., Leimberger, B., Rossen, B., Sokolowsky, B., Wang, B., Zweig, B., Hoover, B., Samic, B., McGrew, B., Spero, B., Giertler, B., Cheng, B., Lightcap, B., Walkin, B., Quinn, B., Guarraci, B., Hsu, B., Kellogg, B., Eastman, B., Lugaresi, C., Wainwright, C. L., Bassin, C., Hudson, C., Chu, C., Nelson, C., Li, C., Shern, C. J., Conger, C., Barette, C., Voss, C., Ding, C., Lu, C., Zhang, C., Beaumont, C., Hallacy, C., Koch, C., Gibson, C., Kim, C., Choi, C., McLeavey, C., Hesse, C., Fischer, C., Winter, C., Czarnecki, C., Jarvis, C., Wei, C., Koumouzelis, C., and Sherburn, D. Gpt-4o system card. *CoRR*, abs/2410.21276, 2024.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

Kissane, C., Krzyzanowski, R., Conmy, A., and Nanda, N. Saes (usually) transfer between base and chat models. Alignment Forum, 2024.

Lad, V., Gurnee, W., and Tegmark, M. The remarkable robustness of llms: Stages of inference? *CoRR*, abs/2406.19384, 2024.

Lieberum, T., Rajamanoharan, S., Conmy, A., Smith, L., Sonnerat, N., Varma, V., Kramár, J., Dragan, A. D., Shah, R., and Nanda, N. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. *CoRR*, abs/2408.05147, 2024.

Lindsey, J., Templeton, A., Marcus, J., Conerly, T., Batson, J., and Olah, C. Sparse crosscoders for cross-layer features and model diffing. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/crosscoders/index.html.

Mairal, J., Bach, F. R., Ponce, J., and Sapiro, G. Online dictionary learning for sparse coding. In *ICML*, volume 382, pp. 689–696, 2009.

Makhzani, A. and Frey, B. J. k-sparse autoencoders. In *ICLR (Poster)*, 2014.

Marks, S., Rager, C., Michaud, E. J., Belinkov, Y., Bau, D., and Mueller, A. Sparse feature circuits: Discovering and

editing interpretable causal graphs in language models. *CoRR*, abs/2403.19647, 2024.

Memisevic, R., Konda, K. R., and Krueger, D. Zero-bias autoencoders and the benefits of co-adapting features. In *ICLR (Poster)*, 2015.

Mudide, A., Engels, J., Michaud, E. J., Tegmark, M., and de Witt, C. S. Efficient dictionary learning with switch sparse autoencoders. *CoRR*, abs/2410.08201, 2024.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Rajamanoharan, S., Conmy, A., Smith, L., Lieberum, T., Varma, V., Kramár, J., Shah, R., and Nanda, N. Improving dictionary learning with gated sparse autoencoders. *CoRR*, abs/2404.16014, 2024a.

Rajamanoharan, S., Lieberum, T., Sonnerat, N., Conmy, A., Varma, V., Kramár, J., and Nanda, N. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders. *CoRR*, abs/2407.14435, 2024b.

Rivière, M., Pathak, S., Sessa, P. G., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard, T., Shahriari, B., Ramé, A., Ferret, J., Liu, P., Tafti, P., Friesen, A., Casbon, M., Ramos, S., Kumar, R., Lan, C. L., Jerome, S., Tsitsulin, A., Vieillard, N., Stanczyk, P., Girgin, S., Momchev, N., Hoffman, M., Thakoor, S., Grill, J., Neyshabur, B., Bachem, O., Walton, A., Severyn, A., Parrish, A., Ahmad, A., Hutchison, A., Abdagic, A., Carl, A., Shen, A., Brock, A., Coenen, A., Laforge, A., Paterson, A., Bastian, B., Piot, B., Wu, B., Royal, B., Chen, C., Kumar, C., Perry, C., Welty, C., Choquette-Choo, C. A., Sinopalnikov, D., Weinberger, D., Vijaykumar, D., Rogozinska, D., Herbison, D., Bandy, E., Wang, E., Noland, E., Moreira, E., Senter, E., Eltyshev, E., Visin, F., Rasskin, G., Wei, G., Cameron, G., Martins, G., Hashemi, H., Klimczak-Plucinska, H., Batra, H., Dhand, H., Nardini, I., Mein, J., Zhou, J., Svensson, J., Stanway, J., Chan, J., Zhou, J. P., Carrasqueira, J., Iljazi, J., Becker, J., Fernandez, J., van Amersfoort, J., Gordon, J., Lipschultz, J., Newlan, J., Ji, J., Mohamed, K., Badola, K., Black, K., Millican, K., McDonell, K., Nguyen, K., Sodhia, K., Greene, K., Sjösund, L. L., Usui, L., Sifre, L., Heuermann, L., Lago, L., and McNealus, L. Gemma 2: Improving open language models at a practical size. *CoRR*, abs/2408.00118, 2024.

Templeton, A., Conerly, T., Marcus, J., Lindsey, J., Bricken, T., Chen, B., Pearce, A., Citro, C., Ameisen, E., Jones, A., Cunningham, H., Turner, N. L., McDougall, C., MacDiarmid, M., Freeman, C. D., Sumers, T. R., Rees, E., Batson, J., Jermyn, A., Carter, S., Olah,

C., and Henighan, T. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

Wang, K. R., Variengien, A., Conmy, A., Shlegeris, B., and Steinhardt, J. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *ICLR*, 2023.

Wang, Z., Liu, D., Yang, J., Han, W., and Huang, T. S. Deeply improved sparse coding for image super-resolution. *CoRR*, abs/1507.08905, 2015.

Yun, Z., Chen, Y., Olshausen, B. A., and LeCun, Y. Transformer visualization via dictionary learning: contextualized embedding as a linear superposition of transformer factors. In *DeeLIO@NAACL-HLT*, pp. 1–10. Association for Computational Linguistics, 2021.
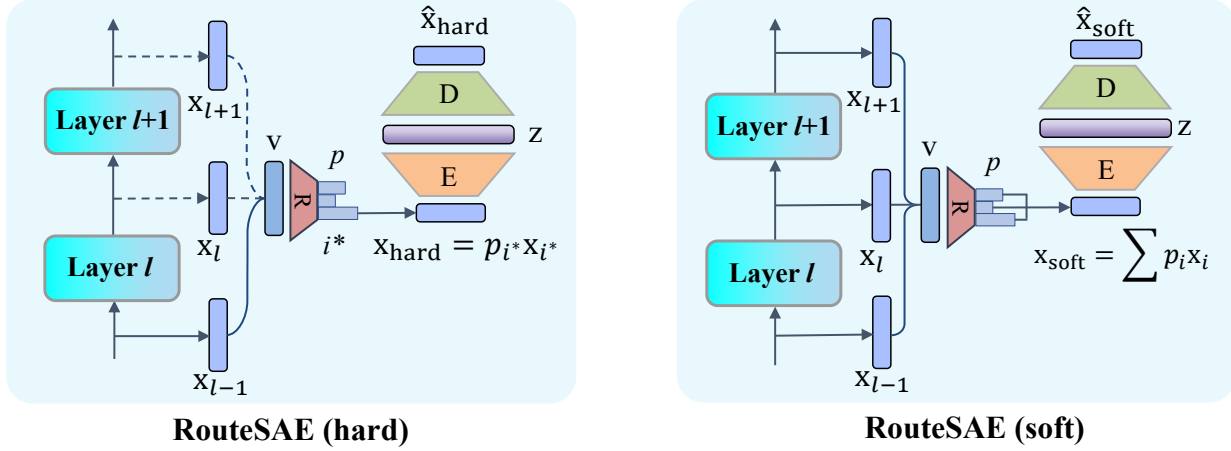
A. Comparison of Routing Mechanisms.

Figure 9. 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.

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, leveraging a richer feature representation compared to hard routing.

Discussion. Hard routing enforces a sparse selection by selecting only a single layer's activation, potentially focusing on the one with the strongest activation for a given input. In contrast, soft routing integrates information from all layers, based on their respective significance probabilities. Meanwhile, soft routing imposes stricter requirements on the router. While hard routing only requires the router to select the layer with the highest feature activation, soft routing demands an accurate estimation of the importance of all layers. If the router fails to make precise predictions, it may assign disproportionately high weights to layers with low-level features strength, thus accumulating irrelevant activations and potentially misleading the subsequent disentangling of monosemantic features.

B. Auto Intrepretation Prompt Design.

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]

C. 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.

C.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

C.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