Beyond Input Activations: Identifying Influential Latents by Gradient Sparse Autoencoders

Anonymous ACL submission

Abstract

Sparse Autoencoders (SAEs) have recently emerged as powerful tools for interpreting and steering the internal representations of large language models (LLMs). However, conventional approaches to analyzing SAEs typically rely solely on input-side activations, without considering the influence between each latent feature and the model's output. This work is built on two key hypotheses: (1) activated latents do not contribute equally to the construction of the model's output, and (2) only latents with high influence are effective for model steering. To validate these hypotheses, we propose **Gradient Sparse Autoencoder** (**GradSAE**), a simple yet effective method that identifies the most influential latents by incorporating output-side gradient information. Our code is available at https://anonymous.4open.science/r/sae_gradient-10FF.

1 Introduction

Sparse Autoencoders (SAEs) (Shu et al., 2025) have recently emerged as promising tools for interpreting the inner workings of large language models (LLMs) (Cunningham et al., 2023; Bricken et al., 2023; Gao et al., 2025; Rajamanoharan et al., 2024b). A core challenge in understanding LLMs is the polysemanticity of neurons, where each neuron encodes multiple features (Arora et al., 2018; Scherlis et al., 2022). This is largely due to superposition (Elhage et al., 2022), a phenomenon where the number of features an LLM needs to represent vastly exceeds the number of available neurons. SAEs address this by learning an overcomplete latent space, allowing each *latent* to represent a single, disentangled feature. For any given LLM representation, only a small number of these latents are activated, and the combination of these sparse active latents can accurately reconstruct the original LLM representation. This sparsity makes it easier to interpret the concepts an LLM is processing.

Although interpretability was the original motivation for developing SAEs, they have proven useful for other applications as well, particularly in *steering model behaviors* (Chalnev et al., 2024; He et al., 2025; Zhao et al., 2024; Galichin et al., 2025). Traditionally, researchers associate each latent with a human-interpretable concept by analyzing which input texts tend to activate it. By modifying selected latents in the SAE space that have desired concepts, ideally we can influence the LLM outputs toward our expectation in a controllable way (Templeton et al., 2024; O'Brien et al., 2024). However, these approaches assume that **the latent's activation based on input reflects an influence on the model output, which has never been proven.** Recent evidence suggests that this assumption may not always hold, and steering can sometimes produce unintended effects on the output (Durmus et al., 2024; Wu et al., 2025).

In this paper, we argue that identifying latents solely from input activations is insufficient for reliable model steering. Instead, the relationship between SAE latents and LLM output should also be considered when determining which latents are most relevant for intervention. To address this, we propose Gradient SAE (GradSAE), a simple yet effective method that can be applied to any instruction-tuned LLM's SAE. Our key insight is that **not all latents activated by the input contribute equally to generating the model's output.** Rather, only those latent variables whose activations, when set to zero, lead to a significant change in the model's outputs are likely to exert substantial influence. In our paper, we prove that this ablation process can be approximated with a more efficient gradient-based approach. To validate our hypothesis, we design two experiments. First, we demonstrate that activated latents have different impacts when used to generate model outputs. Second, we show that the influential latents, identified by GradSAE, are more effective for output steering.

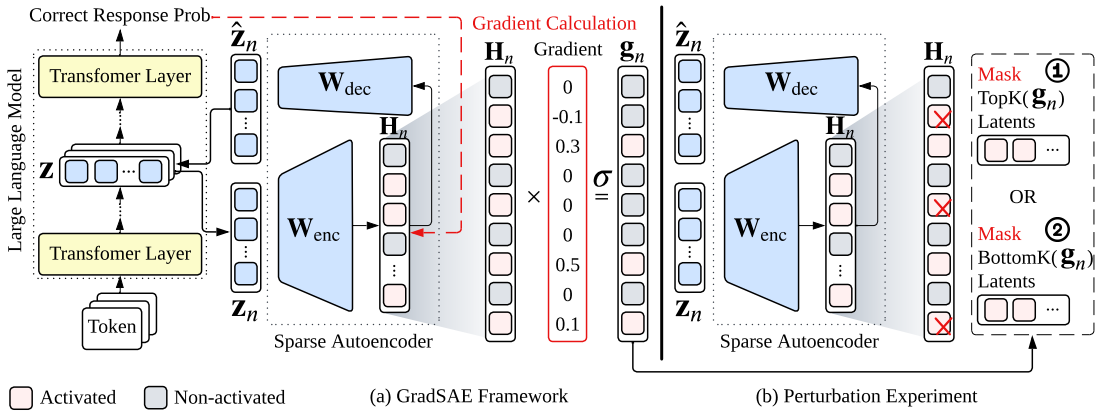

Figure 1: Methodology overview in the perturbation experiment. Subfigure (a) illustrates the GradSAE framework, where the symbol σ denotes the ReLU function. Subfigure (b) shows the experiment process, which contains two settings and both share the same architecture with the GradSAE framework but differ in masking strategies.

2 Methodology

2.1 Problem Statement

Let $\mathcal{V}$ be the vocabulary, $\mathcal{X} \in \mathcal{V}^N$ an input sequence, and $\mathcal{Y} \in \mathcal{V}^M$ the corresponding LLM-generated output. The hidden representation at layer l is $\mathbf{Z}^{(l)} \in \mathbb{R}^{N \times D}$, where D is the hidden dimension. We omit the superscript $^{(l)}$ for simplicity in the rest of the paper. An SAE is inserted at layer l with parameters $\mathbf{W}_{\text{enc}} \in \mathbb{R}^{D \times C}$ and $\mathbf{W}_{\text{dec}} \in \mathbb{R}^{C \times D}$, where C is the latent dimension of SAE and $C \gg D$. Given the representation $\mathbf{Z}$ of input $\mathcal{X}$, SAE first decomposes $\mathbf{Z}$ as sparse *latent activations* $\mathbf{H} \in \mathbb{R}^{N \times C}$, and then loselessly restores $\hat{\mathbf{Z}} \in \mathbb{R}^{N \times D}$ with $\mathbf{H}$ as:

$$\hat{\mathbf{Z}} = \mathbf{H}\mathbf{W}_{\text{dec}} = \sigma(\mathbf{Z}\mathbf{W}_{\text{enc}})\mathbf{W}_{\text{dec}}. \quad (1)$$

Here, σ is a non-linear activation function, and $\hat{\mathbf{Z}}$ is subsequently passed to the rest of layers. We aim to identify which learned *latent* c in $\mathbf{H}$ is most influential in generating $\mathcal{Y}$.

2.2 Proposed GradSAE Framework

We illustrate our proposed GradSAE framework for estimating the influences of sparse latent activations $c = 1, ..., C$ in Figure 1a. Following previous work (Feng et al., 2018; Wu et al., 2024), we define the influence of a certain latent c at the n-th input token on the output $\mathcal{Y}$ as the change in prediction with and without sparse latent activation $\mathbf{H}_{n,c} \in \mathbb{R}$:

$$\mathbf{g}_{n,c} = p(\mathcal{Y}|\mathbf{H}) - p(\mathcal{Y}|\mathbf{H}_{n,/c}), \quad (2)$$

where $\mathbf{H}_{n,/c}$ indicates setting the c-th value of the n-th row at $\mathbf{Z}$ as 0, and probability $p(\mathcal{Y}|\cdot)$ is predicted logits of LLM on $\mathcal{Y}$ with original $\mathbf{H}$ or masked $\mathbf{H}_{n,/c}$ sparse latent activations of SAE.

For efficiency, we approximate $\mathbf{g}_{n,c}$ in Equation (2) with the gradients of the output logit with respect to the latent activations $\mathbf{Z}_{n,c}$ at the n-th input token (see proof in Appendix A). Let $\mathbf{H} = \{\mathbf{H}_1, ..., \mathbf{H}_N\}$ represent token-wise latent activations of the input sequence. Thus, the influence of the c-th latent activation on the n-th token is:

$$\mathbf{g}_{n,c} \approx \frac{\partial p(\mathcal{Y}|h(\mathbf{Z}))}{\partial \mathbf{H}_{n,c}} \odot \mathbf{H}_{n,c}, \quad (3)$$

where $\odot$ indicates element-wise multiplication. In practice, we only focus on the latents that show positive influences to outputs. Equation (3) reveals that the magnitude of the raw sparse latent activation (i.e., $\mathbf{H}_{n,c}$) alone cannot effectively estimate its influence on LLM outputs, whereas many existing works (Templeton et al., 2024; O'Brien et al., 2024) overlook this fact and simply use the latent activations to interpret SAEs and/or steer LLMs. We finally define the overall influence on the c-th latent by averaging individual influence scores across the input sequence, i.e., $\mathbf{g}_c = \frac{1}{N}\sum_{n=1}^{N} \mathbf{g}_{n,c}$.

3 Experiments

We empirically investigate the following research questions (RQs). **RQ1**: Do all activated latents contribute equally to construct the model's output? **RQ2**: How effectively does GradSAE identify the latents that significantly influence the model's output? **RQ3**: Can the latents selected by GradSAE lead to better output steering?

3.1 General Settings

3.1.1 Dataset and Metrics

In this paper, we use the SQuAD dataset (Rajpurkar et al., 2016), where each example consists of a con-

text passage, a question, and an answer. We adopt the standard SQuAD evaluation metrics: Exact Match (EM) and token-level F1. Detailed dataset statistics and description are provided in Appendix B.1, and metric descriptions are in Appendix B.2.

3.1.2 Perturbation Experiment (RQ1 & RQ2)

Experimental Designs. As shown in Figure 1b, our GradSAE experiments involve two settings. Before conducting these, we define two sets of latents:

$$\mathcal{Z}_{\text{high}} = \text{argmax}_{\mathcal{Z}' \subset \mathcal{Z}_{\text{NZ}}, |\mathcal{Z}'|=K} \sum_{c \in \mathcal{Z}'} \mathbf{g}_c, \quad (4)$$

$$\mathcal{Z}_{\text{low}} = \text{argmin}_{\mathcal{Z}' \subset \mathcal{Z}_{\text{NZ}}, |\mathcal{Z}'|=K} \sum_{c \in \mathcal{Z}'} \mathbf{g}_c, \quad (5)$$

where $\mathcal{Z}_{\text{NZ}} = \{c \mid \mathbf{g}_c > 0, c = 1, .., C\}$ are the indices of positive latent influences. In our experiments, we define $K \in \{1, 10, 20, 30, 50\%\}$, where "50%" denotes half the number of non-zero latents count in the SAE activation of the last token.

Let $\mathbf{H} \in \mathbb{R}^{N \times C}$ denote the sequence of token-wise activations. In the first setting, for each token's activation vector, we zero out all latents in $\mathcal{Z}_{\text{high}}$. In the second setting, we instead zero out the latents in $\mathcal{Z}_{\text{low}}$. The modified activation is then passed through $\mathbf{W}_{\text{dec}}$ to reconstruct $\hat{\mathbf{Z}}$ and resume the forward pass. If masking $\mathcal{Z}_{\text{high}}$ leads to degraded performance while masking $\mathcal{Z}_{\text{low}}$ has no impact, this supports our hypothesis that not all activated latents contribute equally to the model's output. Note that we focus only on samples where the LLM can correctly answer with greedy decoding, resulting in 100% accuracy without any perturbation. If masking $\mathcal{Z}_{\text{low}}$ truly has no impact, the performance after perturbation should remain close to 100%.

Baseline. We repeat the GradSAE framework in Section 2.2 but without using gradient information. Specifically, we compute the mean over the original token-wise SAE activations, skipping the gradient calculation: $\mathbf{g}_c = \frac{1}{N} \sum_{n=1}^{N} \mathbf{H}_{n,c}$ We then extract $\mathcal{Z}_{\text{high}}$ and $\mathcal{Z}_{\text{low}}$ from this baseline vector and compare performance when masking these sets.

3.1.3 Local Steering Experiment (RQ3)

This experiment aims to address **RQ3** following a similar design to the perturbation experiment (Section 3.1.2). The key difference in the steering experiment is that, when extracting the TopK and BottomK latents, we also extract the corresponding value. This allows us to adjust the activations using these values in the subsequent settings.

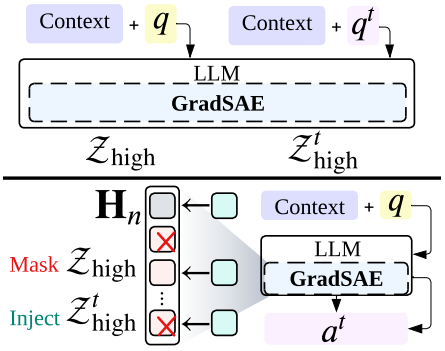

Figure 2: Local steering methodology overview.

Experimental Designs. We consider a particular steering task, called local steering, leveraging a unique property of the SQuAD dataset: each context passage is paired with multiple questions. This allows us to investigate whether the model's output for one question can be steered by latent activations derived from a different question that shares the same context. As shown in Figure 2 upper part, for each data point $d = (\text{context}, q, a) \in D$, we define a set of examples with the same context but different questions as $D^{\text{same}} = \{d^t = (\text{context}, q^t, a^t) \mid q^t \neq q\}$. We randomly select one such example $d^t \in D^{\text{same}}$, and denote its TopK set as $\mathcal{Z}^t_{\text{high}}$.

As shown in the lower part of Figure 2, during the experiment, the $\mathcal{Z}_{\text{high}}$ in the original activation $\mathbf{H}_n$ for all tokens N are zeroed out. The steering latents $\mathcal{Z}^t_{\text{high}}$ from d^t are then injected. Intuitively, if these steering latents carry meaningful information, the model's output may shift toward answering a^t for question q^t, even though the input is question q. We repeat the same experiment for BottomK.

Implementation Details. We primarily conduct experiments using the SAE from the Gemma Scope series (Lieberum et al., 2024), trained on the 9th layer of the Gemma 2 9B Instruct model. For results related to the impact of layer choice, please refer to Appendix D.2. Additionally, in Appendix D.3, we extend our experiments to an SAE trained on the LLaMA 3 model to evaluate the generalizability of our method. Full implementation details are provided in Appendix C.

3.2 Perturbation Experiment Analysis

Table 1 upper section presents the Exact Match (EM) and F1 scores, comparing the performance of masking TopK and BottomK latent sets under both the Baseline and our proposed GradSAE method. The results are reported across values of $K \in \{1, 10, 20, 30, 50\%\}$, as defined earlier. As shown in the "w/o Task" column, the initial LLM

Table 1: Results of both perturbation and local steering. The upper section shows perturbation results, where for the TopK rows, lower scores indicate greater influence on the model output, and for the BottomK rows, higher scores indicate less influence. The best performance in each column is in **bold**. The lower section shows local steering results, where for the TopK rows, higher scores indicate stronger steering effects on the model output, and for the BottomK rows, lower scores indicate weaker steering effects. The highest score in each column is in underlined.

Tasks			w/o Task	K=1		K=10		K=20		K=30		K=50%	
				EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Perturbation	Baseline	TopK ↓	100.0	94.79	97.04	91.81	94.46	85.47	88.49	75.42	79.62	53.42	58.46
		BottomK ↑		100.0	100.0	98.51	99.93	98.51	99.93	98.51	99.93	98.41	99.84
	GradSAE	TopK ↓	100.0	**80.45**	**82.67**	**51.21**	**60.46**	**39.48**	**50.51**	**37.80**	**49.37**	**30.58**	**43.33**
		BottomK ↑		100.0	100.0	98.14	99.68	98.14	99.68	98.14	99.68	98.21	99.66
Local Steering	Baseline	TopK ↑	0.00	4.18	5.77	4.59	6.69	2.19	3.32	1.00	1.62	0.20	0.55
		BottomK ↓		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	GradSAE	TopK ↑	0.00	4.58	6.43	7.99	10.21	4.78	6.63	3.59	4.88	3.39	4.58
		BottomK ↓		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

performance without any perturbation is 100%.

Across all K settings, masking the TopK latents identified by both Baseline and GradSAE leads to a drop in EM and F1 scores, while masking BottomK latents maintains near-perfect performance. For example, both methods yield F1 scores around 99% across $K = 10, 20, 30, 50\%$ when masking BottomK latents. This confirms our **RQ1** hypothesis that not all activated latents equally contribute to output construction, with TopK latents having far greater influence.

Comparing GradSAE and the Baseline, masking GradSAE's TopK latents results in a more substantial performance drop. For instance, with $K = 1$, GradSAE yields 80.45% EM and 82.67% F1, decreasing to 30.58% EM and 43.33% F1 at $K = 50\%$. This consistent degradation highlights that GradSAE more precisely identifies latents critical to output, while the Baseline shows only moderate degradation. These results support **RQ2**, demonstrating GradSAE's superior effectiveness in identifying influential latents.

3.3 Local Steering Experiment Analysis

As shown in Table 1 lower section "w/o Task" column, the initial LLM performance without local steering is 0%. After applying local steering, both **Baseline** and **GradSAE** exhibit steering effects when the original TopK latents are replaced with those from a different question sharing the same context. GradSAE consistently outperforms the Baseline, especially at $K = 10$, achieving a steering F1 score of 10.21%. This supports **RQ3**, demonstrating that GradSAE-identified latents can steer the model toward answering a different question even when the input remains unchanged. However, as K increases, the steering effect diminishes. This is because replacing more TopK latents degrades output coherence as SAE reconstruction becomes less accurate. This aligns with findings from the first experiment, confirming that TopK latents are crucial for output construction, and excessive modification leads to output collapse.

Masking and replacing BottomK latents results in negligible steering across all K values (near 0% EM and F1). Deeper analysis shows that local steering BottomK latents leaves the model's output unchanged. This is consistent with earlier results showing that BottomK latents have minimal influence. These findings confirm that modifying non-influential latents does not meaningfully steer or disrupt the output. For detailed statistical analysis, see Appendix D.1.

4 Related Work

SAEs have shown great promise in interpreting LLMs (Cunningham et al., 2023; Bricken et al., 2023; Gao et al., 2025). However, most existing work focuses solely on input activations and assumes that these activations have an influence on the model's output (Templeton et al., 2024; O'Brien et al., 2024). In contrast, our work challenges this assumption. A detailed discussion on related work is provided in Appendix E.

5 Conclusions

In this work, we revisited the problem of identifying and local steering activated latents in SAEs for LLMs. We proposed two key hypotheses: (1) not all activated latents equally affect output, and (2) only highly influential latents are effective for steering. Through a series of experiments, we demonstrated that GradSAE more accurately identifies influential latents and enables more reliable steering, with results generalizing across different SAEs.

Limitations

In this paper, we focus on SAEs trained on instruction-tuned LLMs. While GradSAE is theoretically applicable to any SAEs, LLMs, and tasks, this study focuses on one of the most fundamental capabilities of modern LLMs with GradSAE, i.e., instruction following for question answering. Extending GradSAE to a broader range of scenarios, including but not limited to pre-trained LLMs, is an exciting direction for future work.

References

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2018. Linear algebraic structure of word senses, with applications to polysemy. *Transactions of the Association for Computational Linguistics*, 6:483–495.

Kola Ayonrinde. 2024. Adaptive sparse allocation with mutual choice & feature choice sparse autoencoders. *arXiv preprint arXiv:2411.02124*.

Joseph Bloom, Curt Tigges, Anthony Duong, and David Chanin. 2024. Saelens. https://github.com/jbloomAus/SAELens.

Dan Braun, Jordan Taylor, Nicholas Goldowsky-Dill, and Lee Sharkey. 2025. Identifying functionally important features with end-to-end sparse dictionary learning. *Advances in Neural Information Processing Systems*, 37:107286–107325.

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.

Bart Bussmann, Patrick Leask, and Neel Nanda. 2024. Batchtopk sparse autoencoders. *arXiv preprint arXiv:2412.06410*.

Sviatoslav Chalnev, Matthew Siu, and Arthur Conmy. 2024. Improving steering vectors by targeting sparse autoencoder features. *arXiv preprint arXiv:2411.02193*.

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

Esin Durmus, Alex Tamkin, Jack Clark, Jerry Wei, Jonathan Marcus, Joshua Batson, Kunal Handa, Liane Lovitt, Meg Tong, Miles McCain, Oliver Rausch, Saffron Huang, Sam Bowman, Stuart Ritchie, Tom Henighan, and Deep Ganguli. 2024. Evaluating feature steering: A case study in mitigating social biases.

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. 2022. Toy models of superposition. *Transformer Circuits Thread*.

Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. 2018. Pathologies of neural models make interpretations difficult. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3719–3728.

Andrey Galichin, Alexey Dontsov, Polina Druzhinina, Anton Razzhigaev, Oleg Y Rogov, Elena Tutubalina, and Ivan Oseledets. 2025. I have covered all the bases here: Interpreting reasoning features in large language models via sparse autoencoders. *arXiv preprint arXiv:2503.18878*.

Leo Gao, Tom Dupre la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. 2025. Scaling and evaluating sparse autoencoders. In *The Thirteenth International Conference on Learning Representations*.

Davide Ghilardi, Federico Belotti, and Marco Molinari. 2024. Efficient training of sparse autoencoders for large language models via layer groups. *arXiv preprint arXiv:2410.21508*.

Zirui He, Haiyan Zhao, Yiran Qiao, Fan Yang, Ali Payani, Jing Ma, and Mengnan Du. 2025. Saif: A sparse autoencoder framework for interpreting and steering instruction following of language models. *arXiv preprint arXiv:2502.11356*.

Shruti Joshi, Andrea Dittadi, Sébastien Lachapelle, and Dhanya Sridhar. 2025. Identifiable steering via sparse autoencoding of multi-concept shifts. *arXiv preprint arXiv:2502.12179*.

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

Luke Marks, Alasdair Paren, David Krueger, and Fazl Barez. 2024. Enhancing neural network interpretability with feature-aligned sparse autoencoders. *arXiv preprint arXiv:2411.01220*.

Anish Mudide, Joshua Engels, Eric J Michaud, Max Tegmark, and Christian Schroeder de Witt. 2024. Efficient dictionary learning with switch sparse autoencoders. *arXiv preprint arXiv:2410.08201*.

Kyle O'Brien, David Majercak, Xavier Fernandes, Richard Edgar, Jingya Chen, Harsha Nori, Dean Carignan, Eric Horvitz, and Forough Poursabzi-Sangde. 2024. Steering language model refusal with sparse autoencoders. *arXiv preprint arXiv:2411.11296*.

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. *arXiv preprint arXiv: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. *arXiv preprint arXiv:2407.14435*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Adam Scherlis, Kshitij Sachan, Adam S Jermyn, Joe Benton, and Buck Shlegeris. 2022. Polysemanticity and capacity in neural networks. *arXiv preprint arXiv:2210.01892*.

Dong Shu, Xuansheng Wu, Haiyan Zhao, Daking Rai, Ziyu Yao, Ninghao Liu, and Mengnan Du. 2025. A survey on sparse autoencoders: Interpreting the internal mechanisms of large language models. *arXiv preprint arXiv:2503.05613*.

Samuel Soo, Wesley Teng, Chandrasekaran Balaganesh, Tan Guoxian, and Ming YAN. 2025. Interpretable steering of large language models with feature guided activation additions. In *ICLR 2025 Workshop on Building Trust in Language Models and Applications*.

Glen Taggart. 2024. Prolu: A nonlinearity for sparse autoencoders - ai alignment forum.

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*.

Xuansheng Wu, Wenlin Yao, Jianshu Chen, Xiaoman Pan, Xiaoyang Wang, Ninghao Liu, and Dong Yu. 2024. From language modeling to instruction following: Understanding the behavior shift in llms after instruction tuning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2341–2369.

Xuansheng Wu, Jiayi Yuan, Wenlin Yao, Xiaoming Zhai, and Ninghao Liu. 2025. Interpreting and steering llms with mutual information-based explanations on sparse autoencoders. *arXiv preprint arXiv:2502.15576*.

Yu Zhao, Alessio Devoto, Giwon Hong, Xiaotang Du, Aryo Pradipta Gema, Hongru Wang, Xuanli He, Kam-Fai Wong, and Pasquale Minervini. 2024. Steering knowledge selection behaviours in llms via sae-based representation engineering. *arXiv preprint arXiv:2410.15999*.

Table 2: Datasets Statistics (Avg. = Average, #Ex. = Number of Examples)

DATASET	SQuAD	
	Train	Valid
Context Avg. Length	119.76	123.95
Question Avg. Length	10.06	10.22
Answer Avg. Length	3.16	3.02
Avg. Questions / Context	4.64	5.11
#Ex.	87.6k	10.6k

A Proof of Gradiant Approximation

We aim to prove that the influence of latent activation $\mathbf{g}_{n,c} = p(\mathcal{Y}|\mathbf{H}) - p(\mathcal{Y}|\mathbf{H}_{n,/c})$ defined in Equation (2) can be approximated by the gradient-based approach described in Equation (3). To start with, we consider that $p(\cdot|\cdot)$ implemented by an LLM is continued over its input domain (Wu et al., 2024). Thus, we could extend $p(\mathcal{Y}|\mathbf{H})$ around the $\mathbf{H}_{n,/c}$ with the First-order Taylor expansion:

$$p(\mathcal{Y}|\mathbf{H}) \approx p(\mathcal{Y}|\mathbf{H}_{n,/c}) + \frac{\partial p(\mathcal{Y}|\mathbf{H})}{\partial \mathbf{H}}\bigg|_{\mathbf{H}_{n,/c}} (\mathbf{H} - \mathbf{H}_{n,/c}). \quad (6)$$

Note that, since the only difference between $\mathbf{H}$ and $\mathbf{H}_{n,/c}$ is the latent activation at the c-th latent on the n-th word, i.e., $\mathbf{H}_{n,c}$, Equation (6) can be simplified as:

$$p(\mathcal{Y}|\mathbf{H}) \approx p(\mathcal{Y}|\mathbf{H}_{n,/c}) + \frac{\partial p(\mathcal{Y}|\mathbf{H})}{\partial \mathbf{H}_{n,c}}\mathbf{H}_{n,c}. \quad (7)$$

Bringing this simplified form to the definition of influence $\mathbf{g}_{n,c}$ in Equation (2), we have $\mathbf{g}_{n,c} \approx \frac{\partial p(\mathcal{Y}|\mathbf{H})}{\partial \mathbf{H}_{n,c}}\mathbf{H}_{n,c}$. To this end, the influence of latent activations can be approximated with this gradient-based approach.

B General Settings

B.1 Dataset

In this paper, we use the SQuAD dataset (Rajpurkar et al., 2016), where each example consists of a context passage, a question, and an answer. Detailed dataset statistics are provided in Table 2. Each context is associated with multiple questions, on average around five per context. The answer to each question is a span extracted directly from the corresponding context. We choose this dataset for several reasons. First, the answers are short, averaging around three words, which makes evaluation more straightforward. Second, the context passages are sufficiently long, averaging about 120 words, which helps activate a diverse set of latent variables in the SAEs. This allows our GradSAE method to better identify the most critical latents from among multiple activated ones. Since our GradSAE is a training-free approach, we use only the validation set to evaluate its performance. We have shown an example of the data in Figure 3.

Context:
In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under gravity… Short, intense periods of rain in scattered locations are called “showers”.

Question 1:
What causes precipitation to fall?

Answer 1:
gravity

Question 2:
What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?

Answer 2:
graupel

More Questions:
...

Figure 3: SQuAD dataset example.

B.2 Metrics

We adopt the standard SQuAD evaluation metrics: Exact Match (EM) and token-level F1. EM measures the percentage of predictions that exactly match a ground-truth answer, while F1 captures the overlap at the token level between predictions and references. To ensure fair and consistent evaluation, we normalize text by lowercasing, removing punctuation, trimming extra whitespace, and excluding uninformative words such as “a”, “an”, and “the”. These two metrics together offer a balanced and comprehensive assessment of span-level prediction quality.

C Implementation Details

Since our experimental design involves instruction-based question answering, in theory, our GradSAE method can be applied to any SAE model supported

by the SAE-Lens (Bloom et al., 2024). However, in practice, we find that only instruction-tuned LLMs are capable of successfully answering questions during our experiments. Consequently, in the main paper, we primarily conduct experiments using the SAE from the Gemma Scope series (Lieberum et al., 2024). Specifically, we use the gemma-scope-9b-it-res-canonical SAE, which is trained on activations from the 9th layer of the Gemma 2 9B Instruct model. This SAE features an overcomplete latent space with 131,000 dimensions. Additionally, in Appendix D.3, we extend our experiments to an SAE trained on the LLaMA 3 8B Instruct model to evaluate the generalizability of our method across different instruction-tuned LLMs. All experiments are performed on an A100 SXM4 GPU with 80 GB of memory. For experiments involving randomness, we fix the random seed at 42 to ensure reproducibility. Additionally, we evaluate our approach on the same SAEs trained on activations from the 20th and 31st layers of the Gemma 2 9B Instruct model.

Table 3: Empirical analysis of SAE activations. "Avg." denotes the average number of non-zero latent activations per input.

	Baseline	GradSAE
Activation Avg.	103.41	
50% Avg.	51.46	
Cross TopK Overlap	18.06%	
Cross BottomK Overlap	0.33%	
Inner TopK Overlap	89.91%	50.46%
Inner BottomK Overlap	92.17%	0.15%

D More Experiment Results

D.1 Statistic Analysis

In this section, we empirically analyze and compare the activated latents identified by the **Baseline** and **GradSAE**. As shown in Table 3, both approaches yield, on average, approximately 103 non-zero latent activations per example in the SAE activation. The average value of 50% (defined in Section 3.1.2) is approximately 51. To measure the overlap between the two methods, we compute the cross overlap between the TopK (K = 50%) sets selected by Baseline and GradSAE. We observe that, on average, only 18.06% of the latents in the Baseline's TopK set are also present in GradSAE's TopK set. This suggests that many latents with high input-side activation may not correspond to high output-side gradients. This highlights input activation alone does not reliably indicate influence on the output.

Table 4: Results of the perturbation experiment using SAEs trained on different layers of the LLM. This table evaluates how the choice of layer affects the effectiveness of GradSAE and Baseline.

		Layer 20		Layer 31	
		EM	F1	EM	F1
Baseline	TopK	73.83	82.53	69.52	77.69
	BottomK	96.14	99.39	96.73	99.68
GradSAE	TopK	9.06	38.35	6.53	17.38
	BottomK	96.81	99.74	96.73	99.80

Interestingly, the cross overlap between the BottomK sets of Baseline and GradSAE is even lower, around 0.33%. This suggests that only a few low-activation latents in the Baseline may still carry some gradient signal, whereas other majority latents might have zero gradient. Despite this minimal overlap, Table 1 upper section shows that masking either method's BottomK latents has almost no negative impact on performance. This implies that the vast majority of latents are non-influential to the model's output, and the number of truly uninfluential latents is significantly greater than the value of "50%".

Other than the cross overlap, we also measure the inner TopK and BottomK latent overlap (with K = 50%) across different prompts sharing the same context. Specifically, since each context in our dataset corresponds to multiple questions, we calculate the overlap of activated latents across these different questions within the same context. As shown in Table 3, the baseline approach exhibits a high degree of overlap: 89.91% for TopK latents and 92.17% for BottomK latents. In contrast, GradSAE shows much lower overlaps: 50.46% for TopK and only 0.15% for BottomK. This discrepancy is expected, as the baseline activations are purely input-driven and reflect prompt-level similarity. Since prompts with the same context are textually similar except for minor changes in the question, their activations tend to be similar as well. However, GradSAE's activations are guided by both the input and the output gradient. This gradient signal selectively filters out latents that do not influence the final prediction, leading to more diverse and dynamic activation patterns. These findings further reinforce our earlier results, demonstrating that GradSAE is more effective in identifying truly influential latents that contribute to model output.

Table 5: Results from the perturbation experiment using SAE trained on the LLaMA 3 8B Instruct model. This table evaluates the generalization ability of GradSAE.

		K = 1		K = 10		K = 50%	
		EM	F1	EM	F1	EM	F1
Baseline	TopK	88.57	95.97	80.02	88.73	65.71	78.94
	BottomK	100.0	100.0	97.14	99.78	97.14	99.78
GradSAE	TopK	65.71	79.24	58.14	72.98	48.57	60.99
	BottomK	100.0	100.0	97.14	99.78	97.14	99.78

D.2 Different Layer

While the main experiment focused on an SAE trained on layer 9 of Gemma 2 9B Instruct model, Lieberum et al. (2024) also developed SAEs for layers 20 and 31. To investigate the impact of the choice of layer on GradSAE, we repeated our first experiment using SAEs from these different layers. Specifically, we compared the effects of masking the TopK versus BottomK latents for both Baseline and GradSAE, using K = 50%. The results, presented in Table 4, reveal two key findings. First, consistent with our previous results (Table 1 upper section), masking the TopK latents harms model performance more than masking BottomK latents for both methods across all layers (9, 20, 31). This reinforces our previous idea that not all activated latents have the same influence on the model output.

Second, we observe differing trends between the methods across layers. For the Baseline method, the performance gap between TopK and BottomK masking decreasing at deeper layers (comparing layer 9 vs. 20 vs. 31). This suggests that latents activated solely on input becomes less effective at influencing outputs in later layers. However, for GradSAE, the significant performance gap between TopK and BottomK masking is maintained across all layers. This highlights GradSAE's robustness to layer choice, likely because it selects latents based on both input activation and output gradients. These results further validate GradSAE's ability to identify latents truly influential to the model's output.

D.3 Different SAE

In addition to experiments on the Gemma 2 9B Instruct model, we also evaluated our first experiment setup using an SAE trained on the LLaMA 3 8B Instruct model. Specifically, we use the llama-3-8b-it-res-jh SAE, which was trained on activations from the 25th layer and features a latent space with 65,536 dimensions. In this setting, we set $K \in \{1, 10, 50\%\}$, where $K = 50\%$ corresponds to approximately 26 latents, given that the llama-3-8b-it-res-jh SAE activates around 52 latents per token on average. As shown in Table 5, the performance trends are consistent with our main experiment that masking the TopK latents identified by GradSAE results in a substantial drop in model performance, whereas masking the BottomK latents has negligible impact. While the Baseline method also exhibits this general trend, the performance gap between TopK and BottomK masking is notably smaller compared to GradSAE. These results further support **RQ2**, demonstrating that GradSAE can effectively identify the most influential latents even when applied to different SAEs trained on different instruction-tuned LLMs.

E Related Works

E.1 Sparse AutoEncoder

SAEs have emerged as a widely used and highly promising tool for interpreting the internal mechanisms of LLMs (Cunningham et al., 2023; Bricken et al., 2023; Gao et al., 2025; Rajamanoharan et al., 2024b). An SAE is a neural network framework designed to learn an overcomplete and sparse representation of model activations, which helps disentangle the superimposed features within LLMs (Elhage et al., 2022). This directly addresses the polysemanticity problem, where a single neuron responds to multiple unrelated concepts. Traditionally, training a SAE involves balancing reconstruction fidelity with strong sparsity constraints, ensuring that only a small subset of latents activate for any given input. As a result, SAEs can extract more interpretable, monosemantic features, offering a clearer and more human-understandable view of LLM internal behaviors.

Beyond the traditional SAE, a variety of SAE variants have been proposed to further enhance in-

terpretability, improve reconstruction quality, or optimize training efficiency. These advancements can be broadly categorized into two areas: architectural improvements and training strategy improvements (Shu et al., 2025). In terms of architectural improvements, models such as Gated SAE (Rajamanoharan et al., 2024a), TopK SAE (Gao et al., 2025), Batch TopK SAE (Bussmann et al., 2024), ProLU SAE (Taggart, 2024), JumpReLU SAE (Rajamanoharan et al., 2024b), and Switch SAE (Mudide et al., 2024) introduce modifications to the activation mechanisms (e.g., TopK selection or gated activations) to better enforce sparsity and refine feature selection. On the other hand, improvements in training strategies include approaches like Layer Group SAE (Ghilardi et al., 2024), Feature Choice SAE (Ayonrinde, 2024), Mutual Choice SAE (Ayonrinde, 2024), Feature Aligned SAE (Marks et al., 2024), and End-to-end SAE (Braun et al., 2025). These methods enhance feature selection, alignment, and training efficiency while preserving the core architecture of traditional SAEs. In this paper, GradSAE introduces a training-free approach that leverages output gradients falling under architectural improvements.

E.2 Model Steering

SAEs have emerged as a powerful tool not only for interpreting the internal mechanisms of LLMs but also for steering their behavior, since SAEs can identify distinct, human-interpretable features within LLMs. Once these features are identified, interventions can be performed by activating or suppressing the corresponding latents during inference. Several recent studies have explored and enhanced the use of SAEs for steering LLMs. SAIF (He et al., 2025) proposes a framework for interpreting instruction-following capabilities in LLMs by identifying instruction-relevant SAE features and demonstrates how manipulating these features can effectively steer instruction-following behavior. SAE-TS (Chalnev et al., 2024) introduces a method to construct steering vectors that precisely target specific SAE features while minimizing unintended side effects, leading to more controlled and coherent model outputs. SpARE (Zhao et al., 2024) leverages SAE representations to detect and resolve context-memory knowledge conflicts at inference time, enabling LLMs to selectively use contextual or parametric knowledge. Mutual Information-Based Explanations (MIE) (Wu et al., 2025) addresses frequency bias in SAE feature interpretations and proposes runtime steering strategies that adjust feature activations based on more meaningful, discourse-level explanations. FGAA (Soo et al., 2025) further refines activation steering by optimizing over SAE latents, creating highly targeted and interpretable steering vectors that improve steering effectiveness while maintaining output quality. SSAEs (Joshi et al., 2025) propose a new unsupervised method that learns sparse, identifiable latent representations of multi-concept shifts, enabling accurate concept-level steering without requiring curated supervision. However, these approaches implicitly assume a direct correspondence between latents activated solely by the input and the aspects of the output they aim to steer. In contrast, GradSAE challenges this assumption by arguing that the model's output must also be considered when identifying influential latents for steering. By incorporating output-side gradient information, GradSAE provides a more accurate attribution of which latents truly drive model outputs, leading to more effective and reliable steering interventions.