LOW-RANK ADAPTING MODELS FOR SPARSE AUTOENCODERS

Matthew Chen*
Massachusetts Institute of Technology

Joshua Engels*
Massachusetts Institute of Technology

Max Tegmark
Massachusetts Institute of Technology

ABSTRACT

Sparse autoencoders (SAEs) decompose language model representations into a sparse set of linear latent vectors. Recent work has improved SAEs using language model gradients, but these techniques are computationally expensive and still increase downstream loss when using the SAE reconstructions. We attack these limitations with a fundamentally different approach: we low-rank adapt the *language model itself* around a pretrained SAE. We analyze our method across SAE sparsity, SAE width, LLM size, LoRA rank, and model layer on the Gemma Scope family of SAEs. In these settings, our method reduces the cross entropy loss gap by 30% to 55% when SAEs are inserted. Compared to end-to-end (e2e) SAEs, our approach achieves the same downstream cross entropy loss $3\times$ to $20\times$ faster on Gemma-2-2B and $2\times$ to $10\times$ faster on Llama-3.2-1B. Furthermore, our technique improves downstream metrics and can adapt multiple SAEs at once. We argue improving model interpretability is not limited to post-hoc SAE training; Pareto improvements can also be achieved by directly optimizing the model itself.

1 INTRODUCTION

Language models excel in tasks like in-context learning, mathematics, and coding (Brown et al., 2020; OpenAI, 2024; Team et al., 2023; Bubeck et al., 2023; Anthropic, 2024), but the mechanisms underlying their behavior remain opaque. *Mechanistic interpretability* (MI) (Bereska & Gavves, 2024) aims to reverse-engineer these mechanisms into human-understandable algorithms, with a key focus on *features*—the variables of model computation (Olah et al., 2020; Mueller et al., 2024).

A central hypothesis in MI, the *Linear Representation Hypothesis* (LRH) (Elhage et al., 2022a; Park et al., 2023), posits that features correspond to one-dimensional directions in activation space. While recent studies challenge parts of this view (Engels et al., 2024a; Csordás et al., 2024; Engels et al., 2024b), it has been empirically validated in many cases (Nanda et al., 2023; Heinzerling & Inui, 2024). Inspired by this, *sparse autoencoders* (SAEs) (Makhzani & Frey, 2013) have been used to decompose activations into monosemantic features (Cunningham et al., 2023; Bricken et al., 2023).

However, inserting SAE reconstructions back into the model significantly increases cross-entropy loss ($\mathcal{L}_{\text{SAE}}$) compared to the original model ($\mathcal{L}_{\text{BASE}}$). For instance, TopK SAE reconstructions in GPT-4 yield a $\mathcal{L}_{\text{SAE}}$ comparable to a model trained with only 10% of GPT-4's pretraining compute (Gao et al., 2024). To mitigate this trade-off, prior work has refined SAE architectures to optimize the sparsity vs. $\mathcal{L}_{\text{SAE}}$ frontier, including TopK SAEs (Gao et al., 2024), Gated SAEs (Rajamanoharan et al., 2024a), JumpReLU SAEs (Rajamanoharan et al., 2024b), ProLU SAEs (Taggart, 2024), Switch SAEs (Mudide et al., 2024), and e2e SAEs (Braun et al., 2024).

An unexplored direction is whether *models themselves* can be adapted post-SAE training to further improve performance. We answer affirmatively, showing that *Low-Rank Adapters* (LoRA) (Hu et al., 2021) reduce the KL divergence between the original and SAE-modified logits. This adaptation improves $\mathcal{L}_{\text{SAE}}$ and enhances various downstream SAE metrics. Overall, low-rank model adaptation provides a simple yet effective method for improving the interpretability vs. performance trade-off.

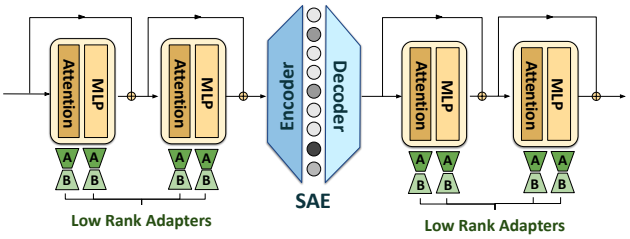

Figure 1: Visual representation of our method, with a local SAE trained on layer 12 and low-rank adapters trained on MLP and attention components on all layers.

1. In Section 3.1, we low-rank adapt for the Gemma Scope (Lieberum et al., 2024) family of SAEs. Across SAE width, sparsity, LLM size, LoRA rank, and inserted layer, we find a 30% to 55% improvement in $\mathcal{L}_{\text{SAE}}$, with the greatest improvements in low-sparsity regimes and larger models.
2. In Section 3.2, we compare our method to e2e SAEs on training time vs. $\mathcal{L}_{\text{SAE}}$ on Gemma-2-2B (Team et al., 2024) and Llama-3.2-1B (AI@Meta, 2024). We find that our method achieves the same $\mathcal{L}_{\text{SAE}}$ as e2e SAEs with between $2\times$ and $20\times$ less compute and $130\times$ fewer backward passes.
3. In Section 3.3, we low-rank adapt with multiple SAEs inserted into Llama-3.1-8B (AI@Meta, 2024) and see large decreases in $\mathcal{L}_{\text{SAE}}$, showing our method's potential to aid circuit analysis.
4. In Appendix A.5, we show our low-rank adapted models exhibit quantitative improvements on a diverse set of downstream tasks using the SAE features.

2 Optimizing Models for Sparse Autoencoders

We formally describe our method of adapting models for SAEs. Denote a decoder only transformer with L layers and hidden dimension d, input $\boldsymbol{x}_0$, output $\boldsymbol{y}$, and the activation after the ith layer by $\boldsymbol{x}_i$. Express the ith transformer block as a function h_i, so the network is

$$\boldsymbol{x}_i = h_i(\boldsymbol{x}_{i-1}) \qquad 1 \leq i \leq L \tag{1}$$

$$\boldsymbol{y} = \text{softmax}(\boldsymbol{x}_L) \tag{2}$$

Appendix A.2 contains a review of SAEs. Unless specified, we use layer 12 residual stream SAEs.

2.1 Method for Low-Rank Adapting Models to SAEs

Formulation. We insert a *frozen* SAE immediately after layer ℓ, and the reconstructed activation $\hat{\boldsymbol{x}}_\ell = \text{SAE}(\boldsymbol{x}_\ell)$ propagates through the remaining layers to produce $\hat{\boldsymbol{x}}_i = h_i(\hat{\boldsymbol{x}}_{i-1})$ for $\ell + 1 \leq i \leq L$ and $\hat{\boldsymbol{y}} = \text{softmax}(\hat{\boldsymbol{x}}_L)$.

For JumpReLU SAEs we only adapt layers *after* the SAE to maintain average sparsity, while for TopK SAEs we can train adapters on all layers. We add low-rank r adapters in each MLP and attention sublayer of every layer we are adapting. Concretely, for each frozen weight matrix $W_i \in \mathbb{R}^{d_1 \times d_2}$, we add $A_i \in \mathbb{R}^{d_1 \times r}$ and $B_i \in \mathbb{R}^{r \times d_2}$ and modify the forward pass according to Equation (3).

Our method is extremely parameter efficient: we train only the low-rank adapters $\Theta = \{A_i\} \cup \{B_i\}$. For all experiments the training objective is $D_{\text{KL}}(\hat{\boldsymbol{y}}, \boldsymbol{y})$—the KL divergence between the next token probability distribution with and without the SAE inserted.

3 Results

3.1 Scaling Laws for Downstream Loss

We first explore how our method scales across SAE sparsity, SAE width, language model size, LoRA rank, and model layer. Specifically, we use Gemma Scope's JumpReLU SAEs (Rajamanoharan et al., 2024b). Over different sparsities, widths, and layers, we track the absolute and percent improvement in $\mathcal{L}_{\text{SAE}} - \mathcal{L}_{\text{BASE}}$ after low-rank adapting. We train on $15M$ random tokens of The Pile (uncopyrighted) dataset (Gao et al., 2020), and evaluate on a held out validation set of $1M$ random tokens. We report our findings in Figure 3 for model size and in Figure 2 for other scaling axes.

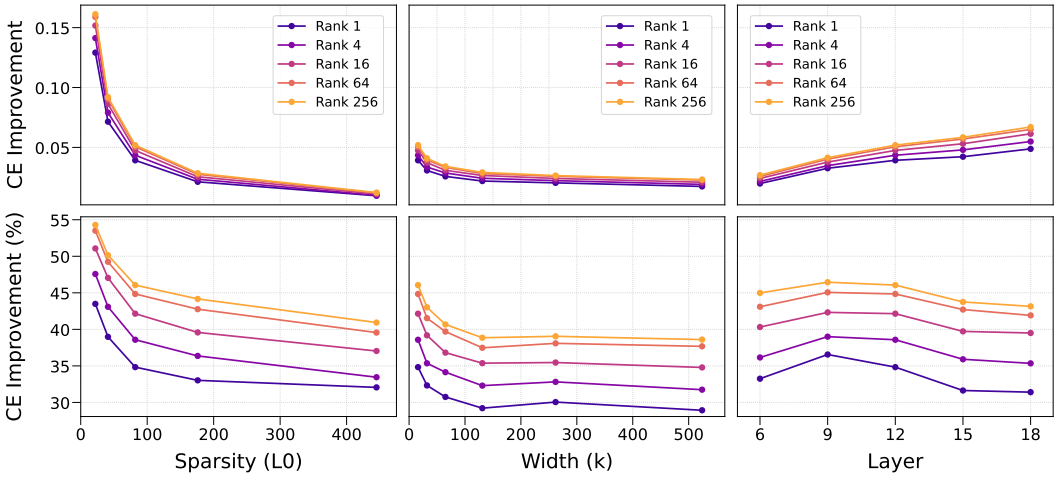

Figure 2: Cross entropy loss improvement (**Top**: absolute, **Bottom:** percentage of CE loss gap closed) using our method for Gemma Scope SAEs on Gemma-2-2B. **Left:** Scaling across sparsity with fixed width=16k and layer=12. **Middle:** Scaling across width with fixed $L_0 = 68$ and layer=12. **Right:** Scaling across layer with fixed $L_0 = 68$ and width=16k.

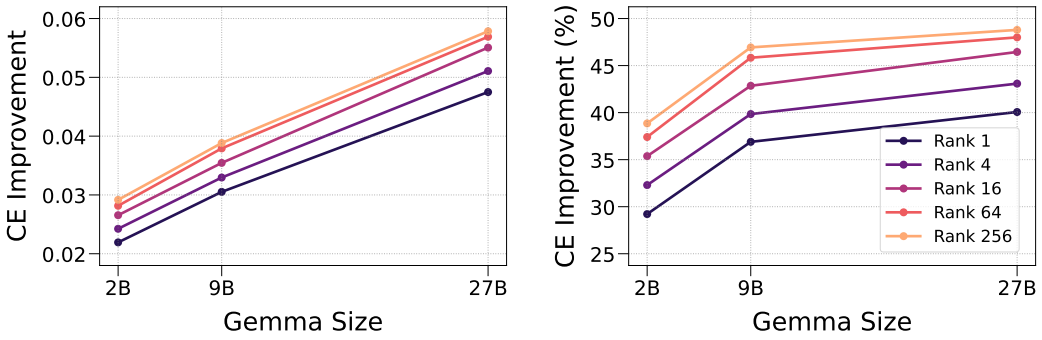

Figure 3: Cross entropy loss improvement (**Top**: absolute, **Bottom:** percentage) for Gemma Scope SAEs of width $16k$ and L_0 closest to 70 on Gemma-2-2B, 7B, and 27B. We find that our method works increasingly well on larger models.

Across all of the scaling regimes, we close the $\mathcal{L}_{\text{SAE}} - \mathcal{L}_{\text{BASE}}$ gap by at least 30%, and sometimes by up to 55%. Using larger rank LoRA adapters reliably decreases the final $\mathcal{L}_{\text{SAE}}$; this, combined with the fact that we adapt on only 15M tokens and do not see our adapters fully converge, implies that with more compute our method may be even more successful.

The improvement is largest on lower sparsities, lower widths, and larger models; all of these results may be caused by these SAEs having a higher cross entropy loss gap to start with. We do still find it extremely promising that the effectiveness of our technique increases on larger models.

Another question is which LoRA layers are most important for reducing $\mathcal{L}_{\text{SAE}}$. In Figure 8, we plot the results of an experiment where we train LoRA adapters on each individual layer after the layer with the inserted SAE. The LoRA performance degrades as it gets farther from the original layer; training LoRA adapters on just the first layer after the SAE achieves 88.14% of the loss reduction in adapting *all* the layers after the SAE, suggesting the loss improvement mechanism may be simple.

3.2 Downstream Loss vs. Computational Cost

We study $\mathcal{L}_{\text{SAE}}$ versus training time for TopK SAEs, e2e SAEs, and TopK SAEs + LoRA. We train our own TopK and e2e SAEs and use their training curves. For TopK SAEs + LoRA, we adapt on all layers with rank 64 for every 10% checkpoint during TopK training. TopK and e2e SAEs are trained

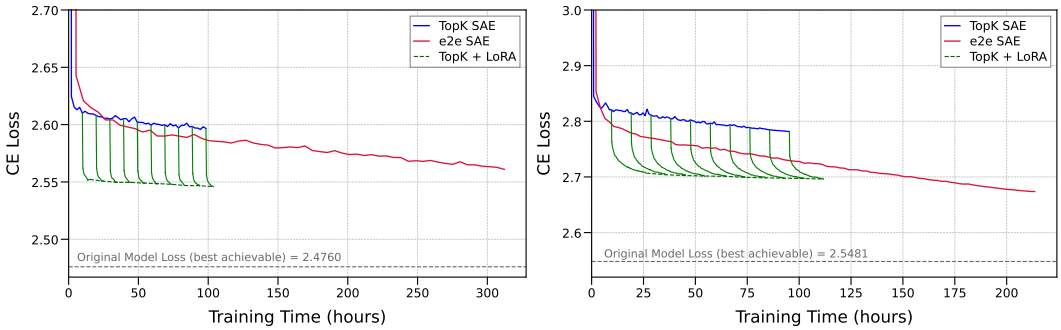

Figure 4: Cross entropy loss vs. training time over 2B tokens for Gemma-2-2B TopK SAEs with width $= 18,432, \mathrm{L}_0 = 64$ (**Left**) and for Llama-3.2-1B with TopK SAEs of $L_0 = 64$ and width 16384 (**Right**). We find our method (TopK + LoRA in the plot) has significant speedups.

for 4B tokens on Llama-3.2-1B and 2B tokens on Gemma-2-2B. We low-rank adapt for 100M tokens on Llama-3.2-1B TopK checkpoints and 15M tokens on Gemma-2-2B TopK checkpoints.

The Pareto cross entropy frontiers for Gemma-2-2B and Llama-3.2-1B in Figure 4 show our method dominates. Quantitatively, we achieve speedups of $2\times$ to $20\times$ in wall clock time in achieving various CE loss threshold when using TopK + LoRA versus e2e (Tables 1 and 2). We also require $130\times$ fewer backward passes through the model than e2e SAEs on Gemma-2-2B and $40\times$ fewer backward passes on Llama-3.2-1B. We do note, however, that e2e SAEs achieve a lower final CE loss than our method on Llama-3.2-1B (although not on Gemma-2-2B).

3.3 Adapting Multiple SAEs

Inserting multiple SAEs into a language model causes $\mathcal{L}_{\text{SAE}}$ to increase rapidly–e.g., inserting 5 SAEs results in worse cross entropy than a unigram model (Gao et al., 2024). Despite this, multi-SAE insertion is valuable for circuit analysis, as it reveals dependencies between SAE latents. Prior work (Marks et al., 2024) mitigate this issue with error terms at the cost of interpretability.

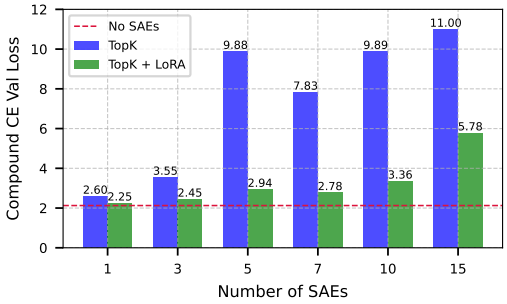

Figure 5: Downstream cross entropy loss when multiple Llama Scope SAEs are inserted into Llama-3.1-8B at once. "Base" is the original loss without any fine-tuning, while "LoRA" is the loss after 15M tokens of LoRA training.

To address this, we modify our approach by inserting all SAEs simultaneously during training while following Section 2.1. We evaluate performance using "Compound Cross Entropy Loss" (Lai & Heimersheim, 2024), which measures $\mathcal{L}_{\text{SAE}}$ with all SAEs inserted. We use the Llama Scope (He et al., 2024) set of TopK SAEs trained on Llama-3.1-8B, and test configurations that maximize layer-wise distance between SAEs: 1 SAE at $\{16\}$, 3 SAEs at $\{10, 20, 30\}$, 5 SAEs at $\{6, 12, 18, 24, 30\}$, 7 SAEs at $\{4, 8, 12, \ldots, 28\}$, 10 SAEs at $\{3, 6, 9, \ldots, 30\}$, and 15 SAEs at $\{2, 4, 6, \ldots, 30\}$.

Figure 5 shows our method significantly reduces compound CE loss; for instance, the compound CE loss for 3 SAEs goes from 3.55 nats to 2.45 nats (lower than the original validation CE loss with a *single* SAE and no LoRA). Thus, our technique seems extremely promising for circuit analysis.

4 Conclusion

Low-rank adapting models for SAEs is a fast, cheap, and effective path to achieving interpretable combined model and SAE systems. In Appendix A.5, we find our LoRA models use SAE features more effectively on downstream tasks. Our work challenges the prevailing assumption that

improving interpretability must rely solely on post-hoc model decomposition and hope our results lay groundwork for future techniques.

REFERENCES

AI@Meta. Llama 3 model card, 2024. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

Anthropic. The claude 3 model family: Opus, sonnet, haiku. Technical report, Anthropic, 2024.

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

Dan Braun, Jordan Taylor, Nicholas Goldowsky-Dill, and Lee Sharkey. Identifying functionally important features with end-to-end sparse dictionary learning, 2024. URL https://arxiv.org/abs/2405.12241.

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

Trenton Bricken, Jonathan Marcus, Kelley Rivoire, and Thomas Henighan. Transformer circuits: September update, 2024. URL https://transformer-circuits.pub/2024/september-update/index.html#oversampling. Accessed: 2025-01-19.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

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

David Chanin, James Wilken-Smith, Tomáš Dulka, Hardik Bhatnagar, and Joseph Bloom. A is for absorption: Studying feature splitting and absorption in sparse autoencoders, 2024. URL https://arxiv.org/abs/2409.14507.

Corbt. All recipes dataset. https://huggingface.co/datasets/corbt/all-recipes, 2024. Accessed: 2025-01-30.

Róbert Csordás, Christopher Potts, Christopher D Manning, and Atticus Geiger. Recurrent neural networks learn to store and generate sequences using non-linear representations. *arXiv preprint arXiv:2408.10920*, 2024.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models, 2023. URL https://arxiv.org/abs/2309.08600.

Jacob Drori. Domain-specific saes, 2024. URL https://www.lesswrong.com/posts/ojERTvdGWW6XRZAqr/domain-specific-saes.

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

Nelson Elhage, Neel Nanda, Zachary Kernfeld, Tom Henighan, Catherine Olsson, and Nicholas Joseph. Softmax linear units, 2022b. URL https://transformer-circuits.pub/2022/solu/index.html.

Joshua Engels, Eric J. Michaud, Isaac Liao, Wes Gurnee, and Max Tegmark. Not all language model features are linear, 2024a. URL https://arxiv.org/abs/2405.14860.

Joshua Engels, Logan Riggs, and Max Tegmark. Decomposing the dark matter of sparse autoencoders. *arXiv preprint arXiv:2410.14670*, 2024b.

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

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

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

Stefan Heimersheim. You can remove gpt2's layernorm by fine-tuning, 2024. URL https://arxiv.org/abs/2409.13710.

Benjamin Heinzerling and Kentaro Inui. Monotonic representation of numeric properties in language models. *arXiv preprint arXiv:2403.10381*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Adam Karvonen, Can Rager, Johnny Lin, Curt Tigges, Joseph Bloom, David Chanin, Yeu-Tong Lau, Eoin Farrell, Arthur Conmy, Callum McDougall, Kola Ayonrinde, Matthew Wearden, Samuel Marks, and Neel Nanda. Saebench: A comprehensive benchmark for sparse autoencoders, 2024. URL https://www.neuronpedia.org/sae-bench/info.

Connor Kissane, Robert Krzyzanowski, Neel Nanda, and Arthur Conmy. Saes are highly dataset dependent: A case study on the dataset, 2024. URL https://www.alignmentforum.org/posts/rtp6n7Z23uJpEH7od/saes-are-highly-dataset-dependent-a-case-study-on-the.

Taras Kutsyk, Tommaso Mencattini, and Ciprian Florea. Do sparse autoencoders (saes) transfer across base and target?, 2024. URL https://www.alignmentforum.org/posts/bsXPTiAhhwt5nwBW3/do-sparse-autoencoders-saes-transfer-across-base-and.

Peter Lai and Stefan Heimersheim. Sae regularization produces more interpretable models, 2024. URL https://www.lesswrong.com/posts/sYFNGRdDQYQrSJAd8/sae-regularization-produces-more-interpretable-models.

Peter Lai and Winston Huang. Gpt-2 circuits, 2024. URL https://peterlai.github.io/gpt-mri/. Accessed: 2025-01-30.

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

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.

Ziming Liu, Eric Gan, and Max Tegmark. Seeing is believing: Brain-inspired modular training for mechanistic interpretability, 2023. URL https://arxiv.org/abs/2305.08746.

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks, 2024. URL https://arxiv.org/abs/2404.19756.

Alireza Makhzani and Brendan Frey. K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, 2013.

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

MedAlpaca. Medical meadow medical flashcards dataset. https://huggingface.co/datasets/medalpaca/medical_meadow_medical_flashcards, 2024. Accessed: 2025-01-30.

Anish Mudide, Joshua Engels, Eric J. Michaud, Max Tegmark, and Christian Schroeder de Witt. Efficient dictionary learning with switch sparse autoencoders, 2024. URL https://arxiv.org/abs/2410.08201.

Aaron Mueller, Jannik Brinkmann, Millicent Li, Samuel Marks, Koyena Pal, Nikhil Prakash, Can Rager, Aruna Sankaranarayanan, Arnab Sen Sharma, Jiuding Sun, et al. The quest for the right mediator: A history, survey, and theoretical grounding of causal interpretability. *arXiv preprint arXiv:2408.01416*, 2024.

Neel Nanda, Andrew Lee, and Martin Wattenberg. Emergent linear representations in world models of self-supervised sequence models. *arXiv preprint arXiv:2309.00941*, 2023.

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

Jeffrey Olmo, Jared Wilson, Max Forsey, Bryce Hepner, Thomas Vin Howe, and David Wingate. Features that make a difference: Leveraging gradients for improved dictionary learning, 2024. URL https://arxiv.org/abs/2411.10397.

OpenAI. Gpt-4o mini: advancing cost-efficient intelligence. July 2024. URL https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/.

OpenAI. Learning to reason with language models, 2024. URL https://openai.com/index/learning-to-reason-with-llms/.

Pain. Arabic tweets dataset. https://huggingface.co/datasets/pain/Arabic-Tweets, 2024. Accessed: 2025-01-30.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. *arXiv preprint arXiv:2311.03658*, 2023.

Itamar Pres, Laura Ruis, Ekdeep Singh Lubana, and David Krueger. Towards reliable evaluation of behavior steering interventions in llms, 2024. URL https://arxiv.org/abs/2410.17245.

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

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

Roudranil. Shakespearean and modern english conversational dataset. https://huggingface.co/datasets/Roudranil/shakespearean-and-modern-english-conversational-dataset, 2024. Accessed: 2025-01-30.

Glen M. Taggart. Prolu: A nonlinearity for sparse autoencoders. https://www.alignmentforum.org/posts/HEpufTdakGTTKgoYF/prolu-a-nonlinearity-for-sparse-autoencoders, 2024.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Maurice Weber, Daniel Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, Ben Athiwaratkun, Rahul Chalamala, Kezhen Chen, Max Ryabinin, Tri Dao, Percy Liang, Christopher Ré, Irina Rish, and Ce Zhang. Redpajama: an open dataset for training large language models, 2024. URL https://arxiv.org/abs/2411.12372.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

A Appendix

A.1 Related Work

SAE Architecture Improvements. Early SAEs for language models used a simple linear encoder, ReLU activation with an L_1 penalty (approximating L_0), and a linear decoder (Bricken et al., 2023; Cunningham et al., 2023). The next generation introduced TopK and BatchTopK SAEs, which enforce sparsity by retaining only the k largest activations (Gao et al., 2024; Bussmann et al., 2024), and GatedSAEs and JumpReluSAEs, which use gating functions and straight-through estimators to approximate direct L_0 optimization (Rajamanoharan et al., 2024a;b). These methods improve the $\mathcal{L}_{\text{SAE}}$ vs. sparsity tradeoff, though no single approach is definitively superior on downstream tasks (Karvonen et al., 2024). Beyond sparsity penalties, Braun et al. (2024) optimize SAEs for KL divergence with the model's logits to directly improve $\mathcal{L}_{\text{SAE}}$, while Olmo et al. (2024) incorporate model gradients into TopK activations for more causal representations. However, gradient-based methods introduce computational overhead and have a large limitation: SAEs are typically trained on cached activations without available gradients (Lieberum et al., 2024).

Fine-tuning SAEs. While this paper focuses on fine-tuning a model around an SAE, another research direction explores fine-tuning SAEs. Some work tailors SAEs to specific domains by oversampling certain contexts (Bricken et al., 2024) or fine-tuning on domain-specific activations (Drori, 2024). Kissane et al. (2024) find that training SAEs on chat data captures the refusal latent, whereas training on the Pile (Gao et al., 2020) does not. Kutsyk et al. (2024) further analyze when base model SAEs generalize to a chat-tuned model, showing that it depends on the language model used.

Training interpretable models. We are aware of two prior works that investigate training more interpretable models using SAEs: both (Lai & Heimersheim, 2024) and (Lai & Huang, 2024) train SAEs and models concurrently, and find that this improves $\mathcal{L}_{\text{SAE}}$. However, because this requires training models from scratch, it is impractical to apply to existing models and is only shown to work in toy settings; in contrast, our method is extremely efficient, and we show it works on models up

to 27B parameters. Many prior works also investigate this direction without SAEs. Elhage et al. (2022b) introduce the softmax linear unit (SoLU) activation function, which increases the fraction of interpretable neurons at no cost on downstream performance; Liu et al. (2023) propose a new loss term penalizing spatially distant connections in the network that leads to visually interpretable networks; Liu et al. (2024) introduce Kolmogorov-Arnold Networks, an alternative to standard MLPs with trainable activation functions that can be replaced by symbolic formulas; and Heimersheim (2024) fine-tune out the LayerNorm components in GPT-2 with a small downstream loss effect.

Parameter Efficient Fine Tuning. *Parameter efficient fine tuning* (PEFT) reduces the cost of full supervised fine tuning by updating fewer effective parameters. One of the simplest and most effective PEFT methods is low-rank adaptation (LoRA) (Hu et al., 2021). LoRA works as follows: for a frozen pretrained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, and low-rank matrices $A \in \mathbb{R}^{d\times r}$, $B \in \mathbb{R}^{r\times k}$ with $r \ll \min(d, k)$, the original forward pass $h(x) = W_0 x$ becomes

$$\hat{h}(\boldsymbol{x}) = W_0 x + ABx. \tag{3}$$

A and B can then be trained while the rest of the model is frozen, resulting in a low-rank update of the base model.

A.2 Preliminaries

A.2.1 TopK Sparse Autoencoders

SAEs learn an encoder $\mathbf{W}_{\text{enc}} \in \mathbb{R}^{m\times d}$ for $m \gg d$, a decoder $\mathbf{W}_{\text{dec}} \in \mathbb{R}^{d\times m}$ with unit norm columns, and biases $\mathbf{b}_{\text{enc}} \in \mathbb{R}^m$, $\mathbf{b}_{\text{dec}} \in \mathbb{R}^d$. We call the m columns of $\mathbf{W}_{\text{dec}}$ latents. For activation $\boldsymbol{x}_l$, the TopK SAE (Gao et al., 2024) reconstructs activation $\hat{\boldsymbol{x}}_l$ as follows:

$$\boldsymbol{z} = \text{TopK}(\mathbf{W}_{\text{enc}}(\boldsymbol{x}_l - \mathbf{b}_{\text{dec}}) + \mathbf{b}_{\text{enc}}) \tag{4}$$

$$\hat{\boldsymbol{x}}_l = \mathbf{W}_{\text{dec}}\boldsymbol{z} + \mathbf{b}_{\text{dec}} = \sum w_i \mathbf{f}_i \tag{5}$$

During training, the SAE minimizes the reconstruction error $\mathcal{L} = \|\boldsymbol{x}_l - \hat{\boldsymbol{x}}_l\|^2$. We train TopK SAEs with $k = 64$ for Gemma-2-2B and Llama-3.2-1B for 2B and 4B tokens, respectively, on the RedPajama dataset (Weber et al., 2024).

A.2.2 End-to-End Sparse Autoencoders

In an e2e SAE (Braun et al., 2024), the SAE minimizes KL divergence with the base model instead of reconstruction error. Formally, if we have

$$\begin{aligned}
\hat{\boldsymbol{x}}_l &= \text{SAE}(\boldsymbol{x}_l), \\
\hat{\boldsymbol{x}}_i &= h_i(\hat{\boldsymbol{x}}_{i-1}) \qquad l < i \leq L, \\
\hat{\boldsymbol{y}} &= \text{softmax}(\hat{\boldsymbol{x}}_L),
\end{aligned}$$

then the e2e SAE minimizes $\mathcal{L} = D_{\text{KL}}(\hat{\boldsymbol{y}}, \boldsymbol{y})$. For both e2e and TopK SAEs, we use a TopK activation function with the same sparsity to allow for fair comparisons.

A.2.3 JumpReLU Sparse Autoencoders

We also evaluate our method on the Gemma Scope JumpReLU SAEs. Instead of the TopK function, JumpReLU SAEs (Rajamanoharan et al., 2024b) use the JumpReLU activation function,

$$\text{JumpReLU}_\theta(z) := zH(z - \theta),$$

where H is the Heaviside step function and $\theta > 0$ is the JumpReLU's threshold. The SAE is trained to minimize

$$\mathcal{L} = \|\hat{\boldsymbol{x}} - \boldsymbol{x}\|_2^2 + \lambda \|\boldsymbol{z}\|_0 , \tag{6}$$

where $\boldsymbol{z}$ is defined from Equation (4).

Table 1: Gemma-2-2B timing results and speedups, nearest hour

CE Loss	TopK + LoRA	TopK	e2e	Speedup
2.60	**12h**	59h	37h	**3.05x**
2.59	**12h**	—	79h	**6.53x**
2.58	**12h**	—	148h	**12.18x**
2.57	**12h**	—	243h	**20.01x**
2.55-2.57	**12h–107h**	—	—	—

Table 2: Llama-3.2-1B timing results and speedups, nearest hour

CE Loss	TopK + LoRA	TopK	e2e	Speedup
2.73	**9h**	—	96h	**10.38x**
2.72	**12h**	—	113h	**9.08x**
2.71	**19h**	—	135h	**7.14x**
2.70	**70h**	—	156h	**2.23x**
2.67-2.70	—	—	**156h–213h**	—

A.3 Speedups

We present the speedups in achieving varying cross entropy thresholds when using our method (TopK + LoRA) compared to direct e2e SAE training.

A.4 Analyzing Why Our Method Works

A.4.1 Per Token Improvement Breakdown

In this experiment, we analyze how LoRA impacts $\mathcal{L}_{\text{SAE}}$ improvements. Figure 6 shows the distribution of $\Delta\mathcal{L}_{\text{SAE}}$ between the original and LoRA models across 15M validation tokens. The per-token loss change varies greatly, with the loss on 37% of tokens even getting worse (see the degradation histogram in the figure). Most of the overall improvement comes from small per-token decreases in loss (roughly 10^{-2} to 1 nats), suggesting LoRA improves loss across many tokens rather than a more bimodal distribution.

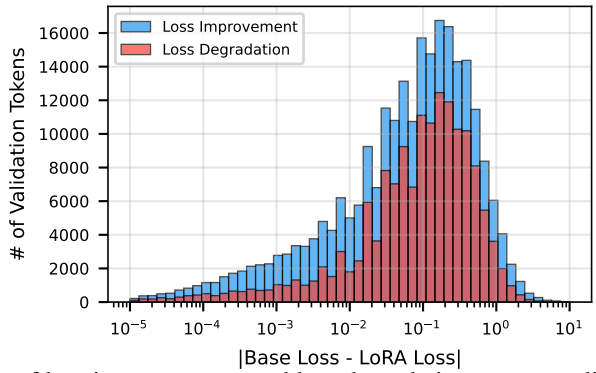

Figure 6: Distribution of loss improvements and loss degredations across validation tokens. We see that more tokens have a loss improvement than degredation (although a substantial number have a degradation) and most loss improvements and degredations happen in a range of about 0.01 to 1 Δ $\mathcal{L}_{\text{SAE}}$ nats.

A.4.2 Activation Distances

One concern identified by (Braun et al., 2024) is that optimizing towards KL divergence may lead the activations to be off distribution and follow a different computational path through the model. We find this is *not* the case with our method: as shown in Figure 7, over a validation set of 500k tokens,

our method slightly *decreases* the distance between activations after the SAE and activations in the original model, while the cosine similarities slightly *increase*.

In Figure 7 we show how low rank adapting the model affects the cosine similarity and L_2 distance of the model activations with and without the SAE.

Figure 7: Change in average distance to original activations before and after applying LoRA; increases in cosine similarity (**Top**) and decreases in Euclidean distance (**Bottom**) are good.

A.4.3 Adapting One Lyaer at a Time

In an effort to see which components of the model are most important in low-rank adapting for SAEs, we plot the improvement when low-rank adapting just one layer for each layer after a Gemmascope JumpReLU SAE inserted at layer 12.

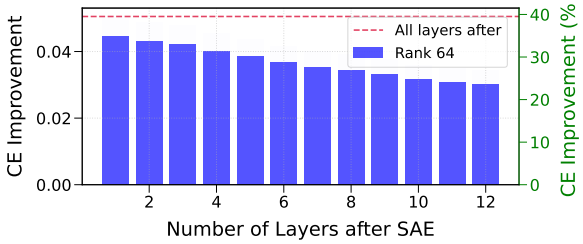

Figure 8: Plot of $\mathcal{L}_{\text{SAE}}$ when running LoRA on just a single layer of Gemma Scope 2B. We find that LoRA on layers closer to the SAE layer do better, and that also LoRA on just the next layer achieves much of the loss reduction of training on all layers.

A.5 Downstream Improvements

A common criticism of previous SAE optimizations is the lack of grounded metrics for evaluating how good an SAE is. Prior work has largely relied on unsupervised metrics such as in Section 3. Recent work, however, has introduced evaluation metrics to measure a model and SAE according

Table 3: Using the same TopK SAE trained on Gemma-2-2B, we compare the SAEBench metrics when the underlying model is low-rank adapted with rank 64. We see across most applicable metrics, the LoRA model shows meaningful improvement. Full results over various thresholds are displayed in Table 6.

DOWNSTREAM METRIC	TOPK + LORA	TOPK
SCR (MAX)	**0.526**	0.448
SCR (AVERAGE)	**0.314**	0.289
TPP (MAX)	**0.412**	0.372
TPP (AVERAGE)	**0.145**	0.111
SPARSE PROBING (TOP 1)	**0.760**	0.732
SPARSE PROBING (TEST)	**0.956**	0.955
AUTOINTERP	0.830	**0.832**
ABSORPTION	0.210	**0.205**

to their performance on downstream tasks (Karvonen et al., 2024; Pres et al., 2024). Thus, in this section, we evaluate our method on a diverse set of downstream benchmarks:

1. In Appendix A.5.1, we show that using LoRA on all layers improves downstream tasks on SAEBench.
2. In Appendix A.5.2, we introduce a novel steering metric and show that our method improves on it. We introduce a new metric because SAEBench metrics do not test the effects of SAEs on next token prediction.
3. In Appendix A.5.3, we show that our method improves overall model performance with the SAE inserted on MMLU, HellaSwag, and TruthfulQA.

A.5.1 SAEBENCH IMPROVEMENTS

To address the core challenge of measuring how effectively a model and SAE work together, Karvonen et al. (2024) introduce SAEBench, a benchmark of SAE metrics that are faithful to possible real world use cases. For the Gemma-2-2B TopK SAE ($L_0 = 64$) we trained in Section 3.2, we evaluate the model with the SAE and the model with the SAE + LoRA on SAEBench.

Specifically, we look at spurious correlation removal (SCR), targeted probe perturbation (TPP), SPARSE PROBING, automated interpretability (AUTOINTERP), and feature absorption (ABSORPTION). SCR measures the separation of latents for different concepts, with higher scores indicating better ability to debias a classifier. TPP evaluates the impact of ablating specific latents on probe accuracy, where higher scores reflect well-isolated latents corresponding to classes on a dataset. SPARSE PROBING tests the accuracy of a k-sparse probe trained on SAE latents, with higher scores indicating better feature learning. AUTOINTERP, assessed using an LLM judge (GPT-4o-mini (OpenAI, 2024)), quantifies the interpretability of SAE latents. ABSORPTION quantifies to what extent latents are "absorbed" together to improve sparsity. All metrics range from 0 to 1, with higher being better except for ABSORPTION. [1] For full in depth definitions of the SAEBench metrics, see Appendix A.5.6.

We display our results in Table 3, showing our low-rank adapted model outperforms the base model on TPP, SCR, and SPARSE PROBING, while very slightly underperforming on AUTOINTERP and ABSORPTION.

A.5.2 FEATURE STEERING

In this section, we demonstrate that the LoRA tuned model improves at activation steering–repressing or eliciting model behavior by scaling a steering vector–using its SAE latents. Given an SAE latent $\mathbf{v} \in \mathbb{R}^d$ at layer l, we steer via

$$\boldsymbol{x}_l \to \boldsymbol{x}_l + \alpha(\boldsymbol{x}_l \cdot \mathbf{v})\mathbf{v}, \quad \alpha \in \mathbb{R}. \tag{7}$$

We assess steering effectiveness following (Pres et al., 2024), who evaluate steering by analyzing increases in likelihood for "positive" texts (aligned with the desired behavior) and decreases for

[1] Excluded from our results are the RAVEL and UNLEARNING SAEBench metrics. RAVEL is not yet implemented in SAEBench and UNLEARNING is recommended for instruct models only.

"negative" texts (not aligned). We note that Olmo et al. (2024) also introduce an SAE steering evaluation, but it does not follow the best practices for steering laid out by (Pres et al., 2024) so we do not use it.

Our method is slightly different than (Pres et al., 2024) because we are comparing different *models* with the same steering method instead of comparing different steering methods on the same model. For a given SAE latent, we steer on a dataset of 500 positive and negative samples. The negative dataset consists of an equal mix of arabic tweets (Pain, 2024), medical facts (MedAlpaca, 2024), recipes (Corbt, 2024), shakespearean quotes (Roudranil, 2024), and law texts (GPT-4o-mini generated). We generate the positive datasets by selecting a latent about 1) machine learning, 2) San Fransisco, 3) Donald Trump, and 4) Covid-19. We then generate text samples where that latent fires using GPT-4o-mini. See Appendix A.5.5 for full prompt details.

Following (Pres et al., 2024), we compute mean token log-likelihoods before and after steering, normalizing them so the original likelihoods span 0 to 100. We tune the hyperparameter α in Equation (7) by selecting a value that increases the likelihood of positive samples while minimizing likelihood increases on a validation subset of negative samples (medical facts). After tuning, we evaluate the effect of α on a test consisting of the remaining negative samples. This tuning process is repeated for the base and LoRA models.

To compare models, let Δ_{POSITIVE} and Δ_{NEGATIVE} represent the change in normalized likelihoods for positive and negative datasets when switching from the base to the LoRA model. The LoRA model is better suited for steering if $\Delta_{\text{POSITIVE}} > 0$ and $\Delta_{\text{NEGATIVE}} \leq 0$. We compute 90% confidence intervals for Δ_{POSITIVE} and Δ_{NEGATIVE} across 500 examples for each of our four SAE latents. Results are summarized in Table 4. We show a histogram of changes across all examples after steering for various latents in Figure 9.

Table 4: For each SAE latent, Δ_{POSITIVE} and Δ_{NEGATIVE} denote the 90% CI improvement in normalized log-likelihood increase when using the LoRA model for steering on positive and negative examples, respectively. Because $\Delta_{\text{POSITIVE}} > 0$ and $\Delta_{\text{NEGATIVE}} \leq 0$, we see the LoRA model is better at steering for a given SAE latent while not affecting other features.

SAE Feature	Δ_{POSITIVE}	Δ_{NEGATIVE}
Machine Learning	$\mathbf{0.86 \pm 0.82}$	-0.84 ± 0.43
San Francisco	$\mathbf{0.97 \pm 0.76}$	0.06 ± 0.20
Donald Trump	$\mathbf{2.50 \pm 0.56}$	0.20 ± 0.40
COVID-19	$\mathbf{0.44 \pm 0.25}$	-0.01 ± 0.06

A.5.3 General Language Model Capabilities

In addition to downstream tasks, we evaluate the model's general language capabilities on MMLU (Hendrycks et al., 2020), HellaSwag (Zellers et al., 2019), and TruthfulQA (Lin et al., 2021) in the following 3 settings: 1) **Original**, the original model with no SAE, 2) **SAE**, the original model with the SAE, and 3) **SAE + LoRA**, a low rank adapted model with the SAE. Our results, shown in Table 5, show that our method improves on all datasets. For discussion of why we think our method is effective, see Appendix A.4.

A.5.4 Steering Distribution Plots

In Figure 9, we plot histograms for the changes in normalized log-likelihoods for each of the four datasets from Table 4.

A.5.5 Generating Steering Positive Datasets

To generate our "positive" examples dataset, we generate examples eliciting the SAE feature with GPT-4o-mini. We prompt using the following chat template.

```
prompt = """
    Generate {num_examples} text examples that have the following feature:
```

Table 5: Comparisons of SAE, SAE + LoRA, and original model performance on standard NLP benchmarks. Error ranges represent one standard error; largest value between SAE and SAE + LoRA is bolded.

METRIC	GEMMA 2B			GEMMA 9B		
	SAE	SAE + LORA	ORIGINAL	SAE	SAE + LORA	ORIGINAL
MMLU	44.2 ± 0.4	$\mathbf{45.8 \pm 0.4}$	49.3 ± 0.4	64.2 ± 0.4	$\mathbf{65.7 \pm 0.4}$	70.0 ± 0.4
HELLASWAG	50.9 ± 0.5	$\mathbf{52.1 \pm 0.5}$	55.0 ± 0.5	58.3 ± 0.5	$\mathbf{59.6 \pm 0.5}$	61.2 ± 0.5
BLEU	29.9 ± 1.6	$\mathbf{30.6 \pm 1.6}$	30.4 ± 1.6	40.9 ± 1.7	$\mathbf{42.4 \pm 1.7}$	43.8 ± 1.7
ROUGE-1	28.2 ± 1.6	$\mathbf{28.5 \pm 1.6}$	26.9 ± 1.6	39.0 ± 1.7	$\mathbf{40.6 \pm 1.7}$	42.7 ± 1.7
ROUGE-2	24.8 ± 1.5	$\mathbf{26.6 \pm 1.5}$	25.6 ± 1.5	33.4 ± 1.7	$\mathbf{36.4 \pm 1.7}$	38.3 ± 1.7
MC1	23.1 ± 1.5	$\mathbf{23.4 \pm 1.5}$	24.1 ± 1.5	27.1 ± 1.6	$\mathbf{28.0 \pm 1.6}$	30.5 ± 1.6

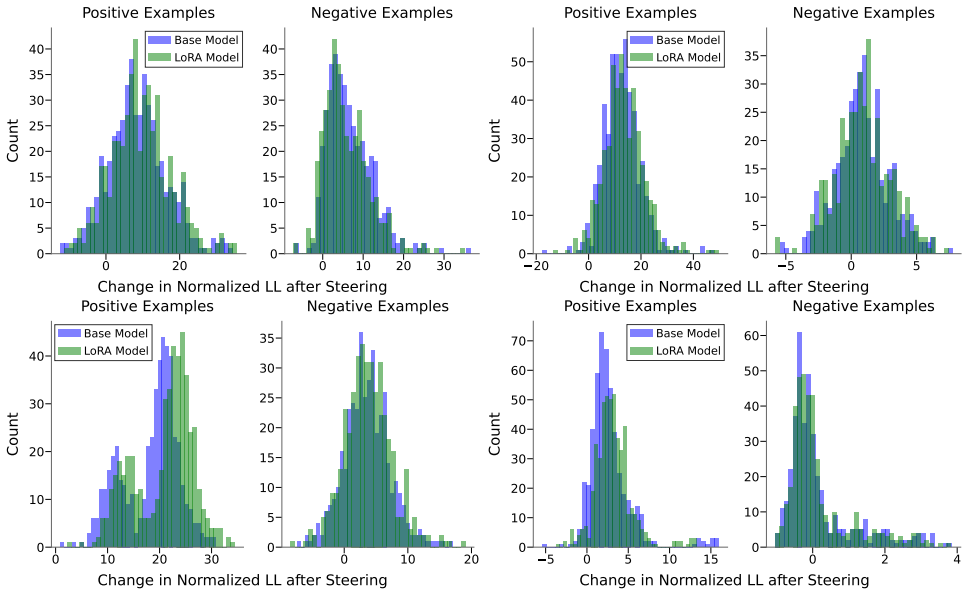

Figure 9: Distribution plots of the change in normalized log-likelihood after steering for various SAE latents. **Top left** is for machine learning (neuron 8421). **Top right** is for San Francisco (neuron 2195). **Bottom left** is for Donald Trump (neuron 13677). **Bottom right** is for COVID-19 (neuron 17811).

```
    {feature_description}

    Below are examples of text that have the feature described above.

    Examples:
    {examples}

    Each text example should be around **twelve** words long and be unique.
    Try to be varied in the content of the examples.
"""
```

A.5.6 SAEBENCH METRICS

Here, we define in more detail the SAEBench metrics defined by (Karvonen et al., 2024) and used in Appendix A.5.1, along with the full results over different hyperparameter splits.

Absorption. Feature absorption occurs when a latent representing concept A implicitly encodes a related concept B (e.g., *Elephant* $\Rightarrow$ *gray*), leading to redundancy or loss of interpretability. This phenomenon disrupts feature disentanglement, as absorbed features may activate inconsistently, obscuring their semantic meaning.

To measure absorption, (Karvonen et al., 2024) adapt the method of (Chanin et al., 2024) using a first-letter classification task. A logistic regression probe is trained on residual stream activations to establish a ground-truth feature direction. They then perform k-sparse probing on SAE latents, identifying primary latents responsible for the task. If increasing k significantly improves F1 by some threshold, the new latent is classified as a feature split.

They then detect absorption by identifying test cases where primary latents fail while the probe succeeds. A latent is flagged as absorbing the feature if it strongly aligns with the probe in cosine similarity and accounts for a sufficient fraction of the probe projection.

Spurious Correlation Removal. The spurious correlation removal (SCR) metric evaluates whether the SAE captures separate latents for distinct concepts (e.g., gender vs. profession). A classifier is trained on a deliberately "biased" dataset (e.g., only male + professor, female + nurse), thereby picking up the spurious correlation, and then the latents most associated with the spurious feature (e.g., gender) are zero-ablated.

During evaluation, the classifier is to be debiased. Choosing the top n latents according to their probe attribution score, a modified classifier is defined in which all latents except for the spuriously correlated latent are zero ablated. Evaluated on a balanced dataset, this modified classifier's accuracy in classifying its concept is tracked, and the metric is defined as

$$S_{\text{SHIFT}} = \frac{A_{\text{abl}} - A_{\text{base}}}{A_{\text{oracle}} - A_{\text{base}}},$$

where A_{abl} is the probe accuracy after ablation, A_{base} is the original spurious probe's accuracy, and A_{oracle} is the accuracy of a probe directly trained on the concept. This SHIFT score quantifies how much ablation improves accuracy (removing the spurious signal), relative. A higher score indicates better separation of the spurious feature and stronger debiasing.

Targeted Probe Perturbation. SHIFT operates on datasets with correlated labels. To extend SHIFT to all multiclass NLP datasets, (Karvonen et al., 2024) introduce TPP, a method that identifies structured sets of SAE latents that disentangle dataset classes. This approach involves training probes on model activations and assessing the impact of ablating specific latent sets on probe accuracy. Ideally, removing a disentangled set of latents should only impact the corresponding class probe while leaving other class probes unaffected.

Consider a dataset where each input is assigned a single label from a set of m possible concepts, $\mathcal{C} = \{c_1, c_2, ..., c_m\}$. For each class indexed by $i \in \{1, ..., m\}$, the most relevant latents L_i are determined using probe attribution scores. To evaluate their effect, the dataset is partitioned into instances belonging to the target concept c_i and a mixed subset containing randomly sampled instances from other labels.

A linear classifier C_j is defined to predict concept c_j with an accuracy of A_j. Furthermore, let $C_{i,j}$ denote the classifier for c_j when latents in L_i are ablated. The accuracy of each classifier $C_{i,j}$ on the corresponding dataset partition for c_j is then computed as $A_{i,j}$. The TPP metric is given by:

$$S_{\text{TPP}} = \frac{1}{m} \sum_{i=j} (A_{i,j} - A_j) - \frac{1}{m} \sum_{i \neq j} (A_{i,j} - A_j)$$

This metric quantifies the extent to which ablating a disentangled set of latents selectively affects its corresponding class. A well-disentangled latent representation should cause a significant accuracy drop when $i = j$ (i.e., ablating latents relevant to class i in classifier C_i) while having minimal effect when $i \neq j$.

Sparse Probing. To evaluate the SAE's ability to learn specific features, SAEs are tested on diverse tasks (e.g., language ID, profession classification, sentiment analysis). Inputs are encoded with the SAE, mean-pooled over non-padding tokens, and the top-K latents are selected via maximum mean difference. A logistic regression probe is trained on these latents and evaluated on a held-out test set to assess how well the SAE captures the target features. A higher score reflects better feature representation (Karvonen et al., 2024).

Table 6: Using the same TopK SAE trained on Gemma-2-2B, we compare the SAEBench metrics when the underlying model is low-rank adapted with rank 64. The threshold hyperparameter for SCR and TPP denotes how many of the top n latents are used in the modified classifier.

Downstream Metrics	LoRA Model	Base Model
SCR Metric @2	0.094	**0.097**
SCR Metric @5	**0.196**	0.177
SCR Metric @10	**0.260**	0.253
SCR Metric @20	**0.336**	0.327
SCR Metric @50	0.447	**0.448**
SCR Metric @100	**0.526**	0.400
SCR Metric @500	**0.342**	0.325
TPP Metric @2	**0.013**	0.007
TPP Metric @5	**0.023**	0.014
TPP Metric @10	**0.035**	0.023
TPP Metric @20	**0.085**	0.039
TPP Metric @50	**0.184**	0.128
TPP Metric @100	**0.266**	0.194
TPP Metric @500	**0.412**	0.372
Sparse Probing (Top 1)	**0.760**	0.732
Sparse Probing (Top 2)	**0.833**	0.832
Sparse Probing (Top 5)	0.875	0.875
Sparse Probing (Top 10)	**0.910**	0.907
Sparse Probing (Top 20)	0.930	0.930
Sparse Probing (Top 50)	0.946	0.946
Sparse Probing (Test)	**0.956**	0.955
AutoInterp	0.830	**0.832**
Absorption	0.210	**0.205**