Transcoders Beat Sparse Autoencoders for Interpretability

Gonçalo Paulo [* 1] **Stepan Shabalin** [* 1] **Nora Belrose** [* 1]

Abstract

Sparse autoencoders (SAEs) extract human-interpretable features from deep neural networks by transforming their activations into a sparse, higher dimensional latent space, and then reconstructing the activations from these latents. Transcoders are similar to SAEs, but they are trained to reconstruct the output of a component of a deep network given its input. In this work, we compare the features found by transcoders and SAEs trained on the same model and data, finding that transcoder features are significantly more interpretable. We also propose *skip transcoders*, which add an affine skip connection to the transcoder architecture, and show that these achieve lower reconstruction loss with no effect on interpretability.

1. Introduction

Recently, large language models have achieved human-level reasoning performance in many tasks (Guo et al., 2025). Interpretability aims to improve the safety and reliability of these systems by understanding their internal mechanisms and representations. While early research attempted to produce natural language explanations of individual neurons (Olah et al., 2020; Gurnee et al., 2023; 2024), it is now widely recognized that most neurons are "polysemantic", activating in semantically diverse contexts (Arora et al., 2018; Elhage et al., 2022).

Sparse autoencoders (SAEs) have emerged as a promising tool for partially overcoming polysemanticity, by decomposing activations into interpretable features (Bricken et al., 2023a; Templeton et al., 2024b; Gao et al., 2024). SAEs are single hidden layer neural networks trained with the objective of reconstructing activations with a sparsity penalty (Bricken et al., 2023a; Rajamanoharan et al., 2024), sparsity constraint (Gao et al., 2024; Bussmann et al., 2024), or an information bottleneck (Ayonrinde et al., 2024). They consist of two parts: an encoder that projects activations into a sparse, high-dimensional latent space, and a decoder that reconstructs the original activations from the latents.

Bricken et al. (2023a) introduced a technique of evaluating the interpretability of SAEs by simulating them with an LLM-based scorer, similar to what had been done on neurons (Bills et al., 2023). This approach is commonly called automated interpretability, or autointerp. SAE features perform much better on this benchmark compared to neurons, even when neurons are "sparsified" by selecting only the top-k most active neurons in a layer for analysis (Paulo et al., 2024). One problem with SAEs is that they focus on compressing intermediate activations rather than modeling the *functional behavior* of network components (e.g., feedforward modules).

Transcoders are an alternative to sparse autoencoders, initially proposed in Li et al. (2023) and Templeton et al. (2024a), and first rigorously evaluated by Dunefsky et al. (2024). Unlike SAEs, transcoders approximate the *input-output function* of a target component, such as an an MLP, using a sparse bottleneck. Dunefsky et al. (2024) demonstrate that transcoders enable fine-grained circuit analysis by learning input-invariant descriptions of component behavior, complementing automated circuit discovery tools like Conmy et al. (2023).

Transcoder design faces inherent challenges. While ReLU MLPs carve up the input space into polytopes,[1] with each polytope corresponding to a relatively high-rank linear function (Black et al., 2022), transcoders' sparse activations mean that each activation pattern corresponds to a low-rank linear transformation. Furthermore, since Marks et al. (2024) and Bricken et al. (2023b), new benchmarks for sparse feature evaluation have emerged (Gao et al., 2024; Karvonen et al., 2024; Juang et al., 2024), motivating a broader evaluation of transcoders across models and tasks. We investigate the tradeoff between reconstruction error and interpretability by comparing SAEs and transcoders and address challenges mentioned above by proposing an architectural improvement—the **skip transcoder**—which mitigates rank limitations via an affine skip connection.

[*]Equal contribution [1]EleutherAI. Correspondence to: Gonçalo Paulo <goncalo@eleuther.ai>.

[1]The polytope interpretation can also be applied, in a somewhat modified form, to MLPs with other activation functions (Balestriero & Baraniuk, 2019).

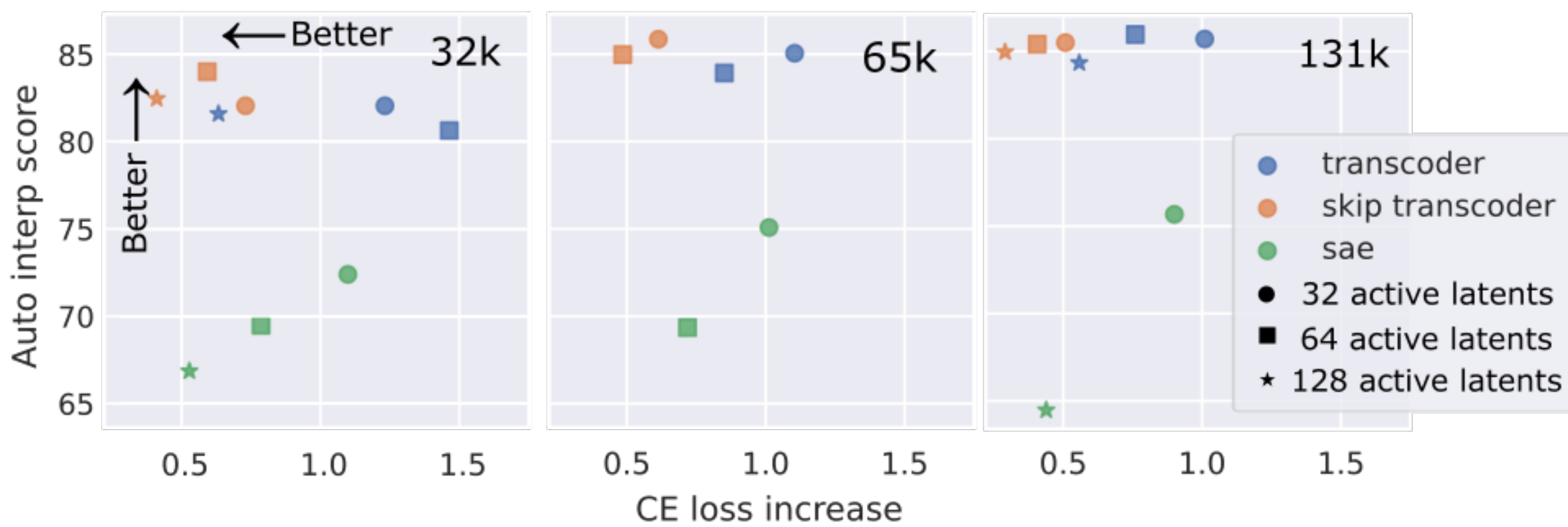

Figure 1. **Skip transcoders are a Pareto improvement on interpretability vs performance degradation**. We compare the increase in cross-entropy loss of 3 different sizes of SAEs and transcoders, 32768 (top right), 65536 (bottom left) and 131072 (bottom right), when patched into the model. For all sizes, skip transcoders are better than transcoders and sparse autoencoders, having both lower increase in model loss and a higher average auto interpretability score. On each quadrant we show 3 models that were trained with a different number of active latents, 32, 64 and 128, except for the 65536 latent model, which only has 32 and 64. The auto interp score is defined as the average fuzzing and detection score of c.a. 500 latents.

In this work, we:

1. Introduce skip transcoders, which reduce reconstruction error without compromising interpretability.
2. Compare transcoders, skip transcoders, and SAEs across diverse models (up to 2B parameters), showing skip transcoders Pareto-dominate SAEs on reconstruction vs. interpretability tradeoffs.
3. Evaluate transcoders on SAEBench (Karvonen et al., 2024) demonstrating improved quality in both latent-level phenomena like absorption and performance on various tasks through sparse probing.

We conclude that interpretability researchers should shift their focus away from sparse autoencoders trained on the outputs of MLPs and toward (skip) transcoders.

2. Methods

Skip transcoders add a linear "skip connection" to the transcoder, which we find improves its ability to approximate the original MLP at no cost to interpretability scores. Specifically, the transcoder takes the functional form

$$f(\boldsymbol{x}) = \boldsymbol{W}_2 \mathrm{TopK}(\boldsymbol{W}_1 \boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{W}_{\mathrm{skip}} \boldsymbol{x} + \boldsymbol{b}_2 \quad (1)$$

Both $\boldsymbol{W}_2$ and $\boldsymbol{W}_{\mathrm{skip}}$ are zero-initialized, and $\boldsymbol{b}_2$ is initialized to the empirical mean of the MLP outputs, so that the transcoder is a constant function at the beginning of training. We leave a deeper analysis of the skip connection, perhaps interpreting it using SVD (Millidge & Black, 2022), for future work.

2.1. Training

We train a collection of sparse coders: sparse autoencoders (SAE), sparse transcoders (ST), and sparse skip transcoders (SST), on the MLP layers of Pythia 160M (Biderman et al., 2023). We also train SAEs and SSTs on Llama 3.2 1B and Gemma 2 2B. We train with mean squared error between the output of the sparse coder and the MLP output, with no auxiliary loss terms. Unlike prior work on transcoders, we adopt the state-of-the-art TopK activation function proposed by Gao et al. (2024), which directly enforces a desired sparsity level on the latent activations without the need to tune an L1 sparsity penalty. We sweep across k values of 32, 64, and 128 in our experiments.

For sparse coders trained on Pythia, we train over the first 8B tokens of Pythia's training corpus, the Pile (Gao et al., 2020). For the other models, we use 8B tokens of the RedPajama v2 corpus (Computer, 2023). All sparse coders are trained using the Adam optimizer (Kingma & Ba, 2015), a sequence length of 2049, and a batch size of 64 sequences.

2.2. Evaluation

We use the automated interpretability pipeline released by Paulo et al. (2024) to generate explanations and scores for sparse coder latents. Activations of latents were collected over 10M tokens, sampled from the Pile for the Pythia models and from FineWeb (Penedo et al., 2024) for Llama and Gemma. The explanations were generated by showing an explainer model, Llama 3.1 70b, 40 activating examples, four from each of ten different quantiles. Each example had 32 tokens, and the active tokens were highlighted. Detection and fuzzing scores were computed over 50 activating

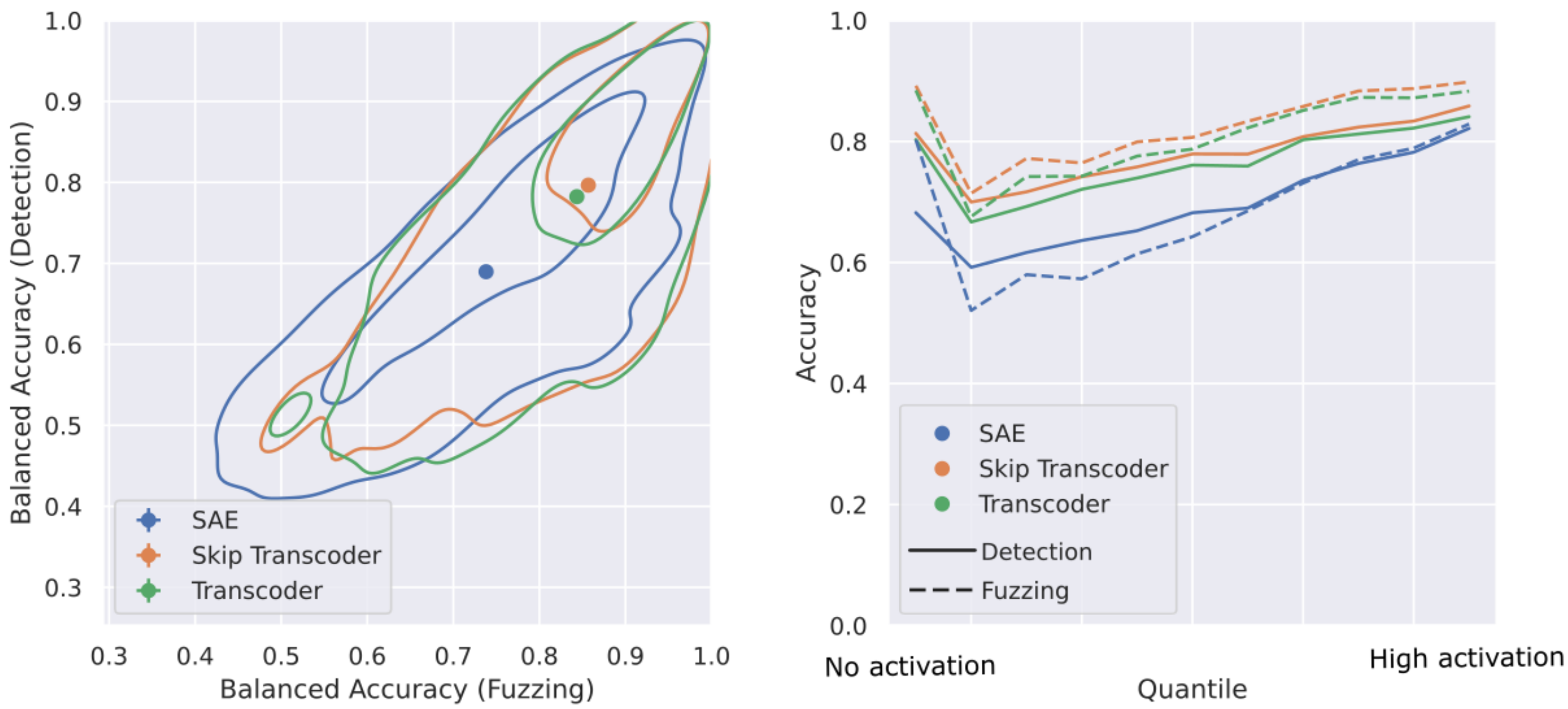

Figure 2. **Interpretability of latents and generalization of explanations.** The interpretability scores of both detection and fuzzing are higher for skip transcoders and transcoders when compared to SAEs, with the distribution being wider for SAEs. Dots in the left plot indicate the average score. The accuracy of the explanations on examples sampled from different quantiles of the activation distribution we can observe that The accuracy of explanations remains higher even for lower quantiles, where the activations are smaller, showing that transcoder and skip-transcoder latents are probably representing more monosemantic concepts along the full distribution.

examples, five from each of ten different quantiles, and 50 non-activating examples. Simulation scores were computed over the same 50 activating examples, as described in (Bills et al., 2023). The scorer model was Llama 3.1 70b.

We also use the SAEBench repository (Karvonen et al., 2024) to evaluate the sparse coders. We use it to compute the variance explained and the cross-entropy loss increase over 500K tokens of the OpenWebText corpus (Gokaslan et al., 2019). SAEBench also provides the ability to train and evaluate **sparse probes** that measure the ability of the SAE's encoder to select information relevant to classification tasks such as sentiment and language detection.

Recently, Chanin et al. (2024) drew attention to the phenomenon of **feature absorption**. In some cases, a more general feature like *starts with the letter L* appears alongside a specific feature like *the token "lion"*, which may prevent the general feature from being active in contexts where intuitively, both the general and the specific feature apply. They argue that this is undesirable. We use SAEBench to compute the frequency of absorption of general letter features into specific features, in SAEs, STs, and STSs.

3. Skip Transcoders Pareto Dominate SAEs

The utility of any sparse coding method for interpretability lies in its ability to accurately reconstruct activations while also generating human-interpretable latent features. This is a fundamental tradeoff: while sparser latents are generally more interpretable, higher sparsity also tends to increase the reconstruction error. The reconstruction error of a sparse coder can be viewed as "dark matter" containing features not captured by the latents (Engels et al., 2024).

Following earlier work on SAEs, we can represent this tradeoff using a reconstruction vs. interpretability curve (Figure 1). Here we compare the reconstruction loss of different models, varying the number number of latents and sparsity, with their interpretability scores, measured as the average between detection and fuzzing score over a set of features. We find transcoders and skip transcoders with the same number of latents generally have higher interpretability scores for the same reconstruction loss than SAEs.

Not only are the average interpretability scores of transcoders and skip transcoders higher than those of SAEs but their distribution is narrower (Figure 2, left panel). Latents of (skip) transcoders also seem to represent more monosemantic features, as the explanations found hold for larger portion of the activation distribution. This can be seen by comparing the accuracy of explanations in examples sampled from different quantiles of the activation distribution (Figure 2, left). The accuracy of explanations decreases more slowly for STs and SSTs than SAEs. The explanations are also more sensitive, as the false positive rate is lower.

Model	Size	K	Fuzzing (%, ↑) SAE	ST	SST	Detection (%, ↑) SAE	ST	SST	Simulation (↑) SAE	ST	SST	CE Loss Increase (%, ↓) SAE	ST	SST
pythia-160m	2^{15}	32	74.6	85.4	**86.4**	70.2	78.7	**80.9**	0.28	0.46	**0.47**	1.10	1.23	**0.73**
pythia-160m	2^{15}	64	71.8	83.7	**86.9**	67.2	77.5	**81.1**	0.30	-	**0.42**	0.79	1.46	**0.59**
pythia-410m	2^{16}	32	78.4	-	**89.5**	72.7	-	**83.8**	0.35	-	**0.51**	0.49	-	**0.43**
llama-1b	2^{17}	32	77.2	-	**85.7**	71.7	-	**79.4**	0.34	-	**0.44**	1.50	-	**1.00**
gemma-2-2b	2^{17}	32	80.5	-	**84.6**	75.8	-	**79.6**	0.35	-	**0.44**	1.60	-	**0.53**

Table 1. **Performance of sparse coders on different models** We compute different interpretability scores, fuzzing, detection and simulation, for SAEs and SSTs trained on different models, as well as the increase of cross-entropy loss when patched into the model. 500 latents are used for fuzzing and detection, but only 50 latents are used for simulation due to it being more computationally expensive. 0.5M tokens are used to compute the cross-entropy loss increase.

We replicated these results in models of the same architecture but different sizes, Pythia 160m and Pythia 410m, and on larger models with different architectures, Llama 3.2 1B (Dubey et al., 2024) and Gemma 2 2B (Team et al., 2024). On all cases studied, SSTs had higher automated interpretability scores and lower CE loss increase when patched in, see Table 1.

We found that performance on sparse probing was similar for SSTs and SAEs (Appendix A), with SSTs winning out for later layers by a small margin. Sparse probing measures the ability of SAEs to preserve information in the original latent, but for transcoders the latents should relate more to concepts necessary for processing the input. It is thus surprising that they are competitive with SAEs on compressing the residual stream without being trained with that objective. We also find that SAEs and transcoders have similar feature absorption behavior, but that those results are noisy. We don't expect this to be a problem, since there are other methods orthogonal to ours which seem to improve feature absorption; see discussion in Section 5.

4. Conclusion

Our experiments suggest that interpretability researchers should shift their focus from sparse autoencoders trained in the outputs of MLPs to (skip) transcoders. In our view, the only downside of transcoders compared to sparse autoencoders is that SAEs can be trained directly on the residual stream, while transcoders need to be trained on particular components of the model (usually a feedforward layer). However, one can easily convert a skip transcoder trained on an FFN into a "residual stream transcoder" by adding the identity matrix to its skip connection. In this way, skip transcoders can be viewed as bridging the gap between these two types of sparse coding. Additionally, it is known that SAEs trained on nearby layers in the residual stream learn very similar features, effectively wasting training compute, while SAEs trained on nearby FFNs learn disjoint sets of features (Balagansky et al., 2024). For this reason, we suggest that practitioners who are planning to train more than one sparse coder on a model should consider training transcoders on FFNs in lieu of SAEs on the residual stream.

5. Future work

As we have shown, transcoders preserve more of a model's behavior and produce more interpretable latents. We believe skip connections let the transcoder avoid the redundant work of translating the linear map, letting it focus on learning important features. Future work may illuminate the role of the skip connection by comparing it to a learned or analytically derived affine approximation of the MLP component.

Dunefsky et al. (2024) highlights the usefulness of transcoders for circuit detection.While we have not run experiments on circuit analysis like in that paper, we expect that skip transcoders to be better for reconstructing circuits thanks to their lower reconstruction error. It is unlikely that the skip connection impedes gradient-based circuit discovery: work like (Marks et al., 2024) shows ways of incorporating linear skip connections into circuit discovery faithfully.

Transcoder and skip transcoder features may be used for steering, but we could not translate the unlearning and concept erasure benchmarks from Karvonen et al. (2024), which require the latent to contain all information in any given residual stream position.

Transcoders also do not help improve the feature learning in SAEs the way new architectures like Gao et al. (2024) do. They merely change the objective of the SAE, which is something that cannot lessen inefficiencies in training. We see the effects of this in the evaluation results on feature absorption: transcoders and skip transcoders can exhibit it just as much as SAEs. Further, feature density plots do not exhibit significant differences Appendix B, showing that SAEs and SSTs are similar on a mechanistic level. Work like Matryoshka SAEs (Bussman et al., 2024; Nabeshima, 2024) may help tackle these issues for both SAEs and SSTs.

6. Contributions

Nora Belrose had the idea that transcoders might be a superior architecture to SAEs, and came up with the idea of skip transcoders. Gonçalo Paulo trained most of the SAEs and transcoders, and performed the experiments and data analysis. Gonçalo and Stepan wrote the first draft. Nora Belrose, Stepan, and Gonçalo wrote the final version. Nora Belrose provided guidance and suggested experiments. Gonçalo, Nora, and Stepan are funded by a grant from Open Philanthropy. We thank Coreweave for computing resources.

7. Code availability.

Code for training transcoders and skip transcoders is available in the sparsify GitHub repo. The skip transcoder checkpoints for Llama 3.2 1B are available on the HuggingFace Hub here, and others will be uploaded to the Hub soon.

Impact Statement

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

References

Arora, S., Li, Y., Liang, Y., Ma, T., and Risteski, A. Linear algebraic structure of word senses, with applications to polysemy. *Transactions of the Association for Computational Linguistics*, 6:483–495, 2018.

Ayonrinde, K., Pearce, M. T., and Sharkey, L. Interpretability as compression: Reconsidering sae explanations of neural activations with mdl-saes. *arXiv preprint arXiv:2410.11179*, 2024.

Balagansky, N., Maksimov, I., and Gavrilov, D. Mechanistic permutability: Match features across layers. *arXiv preprint arXiv:2410.07656*, 2024.

Balestriero, R. and Baraniuk, R. From hard to soft: Understanding deep network nonlinearities via vector quantization and statistical inference. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Syxt2jC5FX.

Biderman, S., Schoelkopf, H., Anthony, Q. G., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.

Bills, S., Cammarata, N., Mossing, D., Tillman, H., Gao, L., Goh, G., Sutskever, I., Leike, J., Wu, J., and Saunders, W. Language models can explain neurons in language models. *URL https://openaipublic. blob. core. windows. net/neuron-explainer/paper/index. html.(Date accessed: 14.05. 2023)*, 2, 2023.

Black, S., Sharkey, L., Grinsztajn, L., Winsor, E., Braun, D., Merizian, J., Parker, K., Guevara, C. R., Millidge, B., Alfour, G., and Leahy, C. Interpreting neural networks through the polytope lens. *arXiv preprint arXiv:2211.12312*, 2022. URL https://arxiv.org/abs/2211.12312.

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*, 2023a. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Bricken, T., Templeton, A., Batson, J., Chen, B., Jermyn, A., Conerly, T., Turner, N. L., Anil, C., Denison, C., Askell, A., Lasenby, R., Wu, Y., Kravec, S., Schiefer, N., Maxwell, T., Joseph, N., 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*, 2023b. URL https://transformer-circuits.pub/2023/monosemantic-features. Published October 4, 2023.

Bussman, B., Leask, P., and Nanda, N. Learning multi-level features with matryoshka saes. *AI Alignment Forum*, 2024. URL https://www.alignmentforum.org/posts/rKM9b6B2LqwSB5ToN/learning-multi-level-features-with-matryoshka-sae

Bussmann, B., Leask, P., and Nanda, N. Batchtopk sparse autoencoders. *arXiv preprint 2412.06410*, 2024. URL https://arxiv.org/abs/2412.06410.

Chanin, D., Wilken-Smith, J., Dulka, T., Bhatnagar, H., and Bloom, J. A is for absorption: Studying feature splitting and absorption in sparse autoencoders. *arXiv preprint arXiv:2409.14507*, 2024.

Computer, T. Redpajama: an open dataset for training large language models, 2023. URL https://github.com/togethercomputer/RedPajama-Data.

Conmy, A., Mavor-Parker, A. N., Lynch, A., Heimersheim, S., and Garriga-Alonso, A. Towards automated circuit discovery for mechanistic interpretability. *arXiv preprint arXiv:2304.14997*, 2023. URL https://arxiv.org/abs/2304.14997.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Dunefsky, J., Chlenski, P., and Nanda, N. Transcoders find interpretable llm feature circuits. *arXiv preprint arXiv:2406.11944*, 2024. URL https://arxiv.org/abs/2406.11944.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

Engels, J., Riggs, L., and Tegmark, M. Decomposing the dark matter of sparse autoencoders. *arXiv preprint arXiv:2410.14670*, 2024.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al. 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. *arXiv preprint arXiv:2406.04093*, 2024.

Gokaslan, A., Cohen, V., Pavlick, E., and Tellex, S. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Gurnee, W., Nanda, N., Pauly, M., Harvey, K., Troitskii, D., and Bertsimas, D. Finding neurons in a haystack: Case studies with sparse probing. *arXiv preprint arXiv:2305.01610*, 2023.

Gurnee, W., Horsley, T., Guo, Z. C., Kheirkhah, T. R., Sun, Q., Hathaway, W., Nanda, N., and Bertsimas, D. Universal neurons in gpt2 language models. *arXiv preprint arXiv:2401.12181*, 2024.

Jermyn, A. and Templeton, A. Ghost grads: An improvement on resampling, 2024. URL https://transformer-circuits.pub/2024/jan-update/index.html#dict-learning-resampling.

Juang, C., Paulo, G., Drori, J., and Belrose, N. Open source automated interpretability for sparse autoencoder features, July 2024. URL https://blog.eleuther.ai/autointerp/.

Karvonen, A., Rager, C., Lin, J., Tigges, C., Bloom, J., Chanin, D., Lau, Y.-T., Farrell, E., Conmy, A., McDougall, C., Ayonrinde, K., Wearden, M., Marks, S., and Nanda, N. Saebench: A comprehensive benchmark for sparse autoencoders, 2024. URL https://www.neuronpedia.org/sae-bench/info. Accessed: 2025-01-17.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015. URL http://arxiv.org/abs/1412.6980.

Li, M., Marks, S., and Mueller, A. dictionary_learning repository, 2023. URL https://github.com/saprmarks/dictionary_learning?tab=readme-ov-file#extra-functionality-supported-by-this-repo.

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. *arXiv preprint arXiv:2403.19647*, 2024.

Millidge, B. and Black, S. The singular value decompositions of transformer weight matrices are highly interpretable. In *AI Alignment Forum*, pp. 17, 2022.

Nabeshima, N. Matryoshka sparse autoencoders. *AI Alignment Forum*, 2024. URL https://www.alignmentforum.org/posts/zbebxYCqsryPALh8C/matryoshka-sparse-autoencoders.

Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., and Carter, S. Zoom in: An introduction to circuits. *Distill*, 5(3), March 2020. ISSN 2476-0757. doi: 10.23915/distill.00024.001. URL http://dx.doi.org/10.23915/distill.00024.001.

Paulo, G., Mallen, A., Juang, C., and Belrose, N. Automatically interpreting millions of features in large language models. *arXiv preprint arXiv:2410.13928*, 2024.

Penedo, G., Kydlíček, H., allal, L. B., Lozhkov, A., Mitchell, M., Raffel, C., Werra, L. V., and Wolf, T. The fineweb datasets: Decanting the web for the finest text data at scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=n6SCkn2QaG.

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

Team, G., Riviere, M., Pathak, S., Sessa, P. G., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard, T., Shahriari, B., Ramé, A., et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Templeton, A., Batson, J., Jermyn, A., and Olah, C. Predicting future activations. January 2024a. URL https://transformer-circuits.pub/2024/jan-update/index.html#predict-future.

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*, 2024b. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

A. SAEBench results

This section contains results on SAEBench (Karvonen et al., 2024). We run three of the evaluations: *core* (reconstruction quality), sparse probing and absorption. We describe all three in the main body and point out that it is not expected for transcoders to outperform SAEs on *absorption*.

	Variance Explained (%)		**Δ NLL (↓, %)**		**Sparse probing (↑)**		**Absorption score (↓)**	
Layer	SAE	SST	SAE	SST	SAE	SST	SAE	SST
L10	16.5	**67.1**	1.1	**0.5**	**71.5**	70.6	36.3 ± 20.2	**28.6 ± 13.3**
L14	17.0	**72.4**	1.1	**0.5**	**80.0**	76.0	33.1 ± 19.4	**25.0 ± 18.4**
L18	20.9	**81.7**	2.1	**0.5**	**78.9**	75.2	**26.3 ± 23.7**	53.3 ± 19.6
L22	21.6	**73.2**	1.6	**0.5**	76.1		24.4 ± 22.0	**18.8 ± 29.5**

Table 2. Results for gemma-2-2B. Δ NLL represents the increase in cross-entropy loss.

	Variance Explained (%)			**Δ NLL (↓, %)**			**Sparse probing (↑)**			**Absorption score (↓)**		
Layer	SAE	ST	SST	SAE	ST	SST	SAE	ST	SST	SAE	ST	SST
L0	99.4	99.4	**99.9**	0.2	0.2	**0.0**	**71.9**	66.1	69.3	95.5 ± 5.9	**54.1 ± 10.5**	60.3 ± 16.3
L2	82.3	80.7	**85.3**	1.1	1.4	**1.0**	59.8	**67.0**	66.0	**14.5 ± 14.3**	47.4 ± 19.2	33.5 ± 14.8
L4	75.9	74.3	**87.4**	1.1	1.2	**0.7**	67.7	67.3	**69.4**	**19.3 ± 18.7**	62.2 ± 26.2	40.5 ± 17.5
L6	81.0	77.8	**86.5**	1.1	1.2	**0.7**	65.9	69.5	**69.9**	**8.2 ± 15.8**	15.6 ± 18.9	31.3 ± 25.7
L8	87.8	85.2	**90.3**	1.1	1.3	**0.9**	68.3	67.4	**71.6**	**18.1 ± 16.8**	82.9 ± 23.6	45.1 ± 21.7
L10	86.5	84.4	**88.8**	1.4	1.7	**1.3**	69.0	71.0	**73.7**	36.4 ± 28.9	**30.5 ± 29.9**	33.9 ± 25.1

Table 3. Results for pythia-160m. Δ NLL represents the increase in cross-entropy loss.

	Variance Explained (%)		**Δ NLL (↓, %)**		**Sparse probing (↑)**		**Absorption score (↓)**	
Layer	SAE	SST	SAE	SST	SAE	SST	SAE	SST
L0	93.0	**93.8**	1.5	**1.0**	**69.9**	69.5	80.7 ± 12.7	**4.7 ± 2.8**
L2	73.8	**80.5**	**1.5**	**1.5**	69.5	**72.4**	**30.2 ± 22.1**	42.3 ± 5.1
L4	63.3	**81.2**	1.5	**1.0**	70.9	**74.5**	**46.7 ± 19.4**	60.7 ± 27.5
L6	57.8	**78.9**	1.5	**1.0**	68.7	**69.9**	82.6 ± 8.7	**36.0 ± 14.6**
L8	63.3	**82.8**	1.5	**1.0**	**72.2**	69.9	**66.1 ± 12.7**	75.1 ± 10.3
L10	69.9	**84.4**	2.0	**1.5**	**74.9**	73.6	**44.7 ± 21.0**	52.3 ± 16.2
L12	71.1	**77.0**	**2.0**	**2.0**		74.6	**9.4 ± 11.0**	45.4 ± 18.9
L14	71.1	**75.8**	**2.0**	**2.0**	71.1	**75.3**	**0.1 ± 0.4**	52.0 ± 19.9

Table 4. Results for llama-1B.Δ NLL represents the increase in cross-entropy loss.

B. Feature density comparison

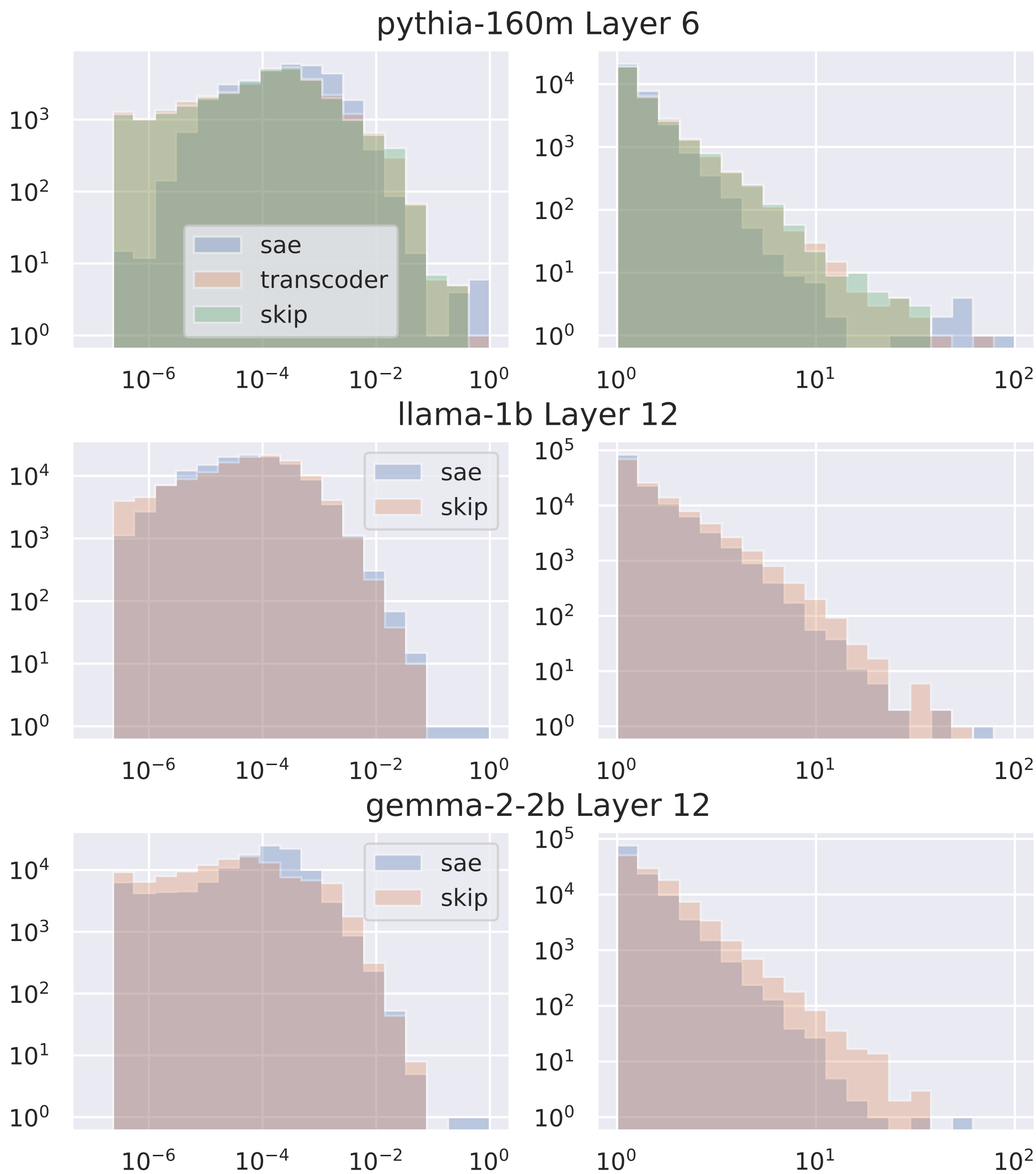

Figure 3. Comparison of feature density (Bricken et al., 2023b) and the consistent activation heuristic (sum of activations over all tokens divided by the number of tokens). These plots show that STs and SSTs are similar in terms of feature density and have less high-density features and more low-density features. This is not a problem because there exist methods for getting rid of low-density features (Bricken et al., 2023b; Jermyn & Templeton, 2024; Gao et al., 2024), but not for regularizing high-density features.