Sparse Autoencoders Can Learn Graded Latents for Relational Composition

Theo Farrell [1] **Patrick Leask** [1] **Noura Al Moubayed** [1]

Abstract

Transformers trained on graph problems can compress structured sequences into fixed-size activation vectors. Prior work on a simple ordered bigram copying task presents a transformer model that does not use a separate feature for order; instead, it represents order through relative feature magnitudes. This raises a problem for interpreting sparse autoencoders (SAEs) trained on these activations: a latent may be meaningful because its value varies, not only because it is active. We train SAEs on this toy model and find graded latents whose activation values, not just their on/off status, correspond to different bigram orderings. In one SAE, one latent tracks how much more the model attends to the first input than the second input, with correlation 0.807; 98% of latents have absolute correlation below 0.004. When we scale this latent, the model swaps 49.8% of all outputs, or 77.8% of outputs where the latent is active. These results suggest that SAE latents can encode relational information through magnitude, so analyses that reduce latents to feature presence indicators may misrepresent model features. Code is available at `https://github.com/Theosdoor/graded-latents`.

1. Introduction

Transformers can learn mechanisms for relational composition (Wattenberg & Viégas, 2024), in which multiple feature vectors are composed into one fixed-length vector while preserving the relation between them. For example, Farrell et al. (2025) train a transformer on an ordered bigram copying task that learns a relative magnitude-based composition mechanism. The transformer does not allocate separate directions for (x, y) and (y, x). Instead, it uses shared embedding directions and changes their coefficients according to the attention pattern.

This poses an interesting challenge for sparse autoencoders (SAEs) (Bricken et al., 2023; Huben et al., 2024) and their variants (Gao et al., 2025; Bussmann et al., 2024; Rajamanoharan et al., 2024b; Bussmann et al., 2025; Karvonen et al., 2025; Leask et al., 2025b). If the transformer reuses shared embedding directions and varies their relative magnitudes, then an SAE trained on these activations may recover the shared directions and encode order in activation magnitude. We call these *graded latents*, for which different activation magnitudes carry different information and latent on/off activation alone is insufficient to fully understand them. This matters for practical SAE workflows because many feature dashboards label a latent by inspecting top-activating examples or by checking whether it fires on a class of inputs. A latent whose low, medium and high activations correspond to different relational states could look broadly active or noisy under that workflow, even when its activation value tracks a coherent internal variable. This challenges interpretation methods that primarily treat SAE latents as on/off feature presence indicators (Paulo et al., 2025; Quirke et al., 2025). The issue persists even if an SAE successfully recovers the underlying feature directions, separating it from previous work focused on dictionary correctness (Chanin et al., 2026; Leask et al., 2025a).

However, Farrell et al. (2025) did not study the impact on SAEs. In this paper, we train a suite of BatchTopK (Bussmann et al., 2024), JumpReLU (Rajamanoharan et al., 2024b) and Matryoshka (Bussmann et al., 2025) SAEs on the toy transformer from Farrell et al. (2025). In this toy setting, high-performing SAEs frequently learn latents whose activation values correspond to different bigram orderings.

In Section 2 we describe the transformer task, SAE training setup, and graded latent identification and steering method. In Section 3, we identify two graded SAE latents using correlations with $\Delta\alpha$, the difference between the separator token's attention to the first and second input symbols. We then show through steering interventions that scaling these latents can swap the model output from (x, y) to (y, x). Across a broad SAE sweep, high-performing SAEs tend to contain latents whose activations correlate with $\Delta\alpha$. For a selected set of six high-performing BatchTopK, JumpReLU,

[1]Department of Computer Science, University of Durham, Durham, United Kingdom. Correspondence to: Theo Farrell <theodore.farrell@durham.ac.uk>.

Mechanistic Interpretability Workshop at the 43rd International Conference on Machine Learning, Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

and Matryoshka SAEs, steering these latents swaps between 25–73% of the transformer's original outputs. We treat this as a controlled case study rather than evidence that the same behaviour occurs in large pretrained models, and use it to motivate further investigation into graded latents in settings such as Gemma Scope (Lieberum et al., 2024; McDougall et al., 2025).

2. Methodology

We study a two-layer attention-only transformer trained on the ordered bigram copying task introduced by Farrell et al. (2025) (Appendix B, Table 2). The model encodes bigram (d_1, d_2) into a single separator token, s, and reconstructs it as $(o_1, o_2) = (d_1, d_2)$. The five input positions are represented as $[d_1, d_2, s, o_1, o_2]$, where tokens at d_1 and d_2 are symbols uniformly sampled from $[0, 99]$ and s, o_1, o_2 contain special tokens. This gives 10,000 possible ordered bigrams. Farrell et al. (2025) find that the residual at s after the first layer is:

$$\boldsymbol{r}_s^{L_1} = \alpha_{s\to d_1}(\boldsymbol{E}_{d_1}+\boldsymbol{P}_{d_1})+\alpha_{s\to d_2}(\boldsymbol{E}_{d_2}+\boldsymbol{P}_{d_2})+\boldsymbol{E}_s+\boldsymbol{P}_s$$

where $\alpha_{s\to d_i}$ is the attention weight in the first layer for query s and key d_i, and $\boldsymbol{E}_x$ and $\boldsymbol{P}_x$ are the token and positional embeddings for the token at x respectively. Thus, the model represents bigram orderings at s using a linear combination of shared embedding directions, $(\boldsymbol{E}_{d_1} + \boldsymbol{P}_{d_1})$ and $(\boldsymbol{E}_{d_2} + \boldsymbol{P}_{d_2})$, with coefficients defined by the attention probabilities.

Models can represent more than n features in n-dimensional activations through superposition, which complicates interpretability (Elhage et al., 2022). SAEs decompose dense, superposed model activations into sparse feature representations (Bricken et al., 2023). We train BatchTopK (Bussmann et al., 2024), JumpReLU (Rajamanoharan et al., 2024b) and Matryoshka (Bussmann et al., 2025) SAEs on $\boldsymbol{r}_s^{L_1}$ using implementations from Marks et al. (2024). SAEs are trained on residual activations on the full finite input space of the toy task (10,000 inputs). Since our goal is not out-of-distribution generalisation, we do not need a train/test split. We evaluate SAEs on sparsity (L_0) and loss recovered (Karvonen et al., 2025; Ferrando et al., 2024) over the complete activation distribution.

After a grid search (Appendix B, Table 3) we find that BatchTopK SAEs with $k \in [3, 5]$ all achieve similar top performance, so we select $k = 3$ and $d_{\text{SAE}} = 128$. We filter for loss recovered over 0.999 and use the checkpoint with the fewest dead latents (3.1%). Appendix B, Table 4 gives the full configuration.

We inspect latent activations using activating examples, firing rates and correlations with the difference between the separator token's attention to the first and second input symbols, $\Delta\alpha = \alpha_{s\to d_1} - \alpha_{s\to d_2}$. Farrell et al. (2025) find that $\Delta\alpha$ distinguishes bigram orderings at $\boldsymbol{r}_s^{L_1}$. We hypothesise that latents related to order correlate with $\Delta\alpha$ more strongly than other latents. To test whether a latent is graded, we perform causal steering interventions inspired by O'Brien et al. (2025). For a given latent, we replace its activation magnitude m with $m' = pm$ for some scale factor p. We reconstruct and replace the original activation at $\boldsymbol{r}_s^{L_1}$ and observe changes in the output logits. We repeat across all $p \in [0, 10]$ (step 0.05) and all input bigrams. If the latent is graded, varying p should systematically change the model's output ordering.

We test for latents correlated with $\Delta\alpha$ across a sweep of 2,613 SAEs spanning multiple architectures and hyperparameters (Appendix B, Table 5). For computational feasibility, we only perform steering on the primary BatchTopK SAE and six additional selected high-performing checkpoints (Appendix B, Table 6), chosen to span architecture, dictionary size and sparsity settings. We therefore treat the steering results as exploratory rather than statistically significant.

3. Results

3.1. Analysing Features

We visualise latent activations over all 100×100 ordered bigrams (d_1, d_2) using heatmaps (Figure 1). We exclude the 8% of latents that activate on fewer than 0.1% of inputs. The remaining latents fall into two categories: k-symbol detectors (91% of all latents) and special latents (2 latents, or 1.6%). Most latents are 1-symbol detectors (88%). They activate only if one specific symbol is present at either d_1 or d_2, producing a 'plus-sign' pattern in Figure 1. They activate most strongly when $d_1 = d_2$. 3% of latents detect multiple symbols ($k > 1$) in this way simultaneously, showing multiple 'plus-signs'. Only two latents (11 and 56) do not follow this pattern. Latent 11 activates on 64% of inputs and latent 56 on 9.9%, compared to fewer than 4.5% for all other latents (Appendix C). Because they activate across many different inputs, they cannot simply indicate that one specific symbol is present.

Moreover, latent 11 strongly correlates with $\Delta\alpha$ (Pearson $r = 0.807$ over the whole dataset). Latent 56 has a weaker negative correlation ($r = -0.3213$). All other latents have $|r| < 0.004$. This gap distinguishes latents 11 and 56 from the symbol detectors and suggests that they may encode ordering information. We use $|r| = 0.3$ as a threshold for identifying special latents because of the significant gap between these latents and others, but other $|r| \in [0.1, 0.5]$ provide similar results. Since $\Delta\alpha$ is positive when attention is weighted towards d_1, we refer to latent 11 as strongly d_1-favouring and latent 56 as weakly d_2-favouring (see correlation details in Appendix C).

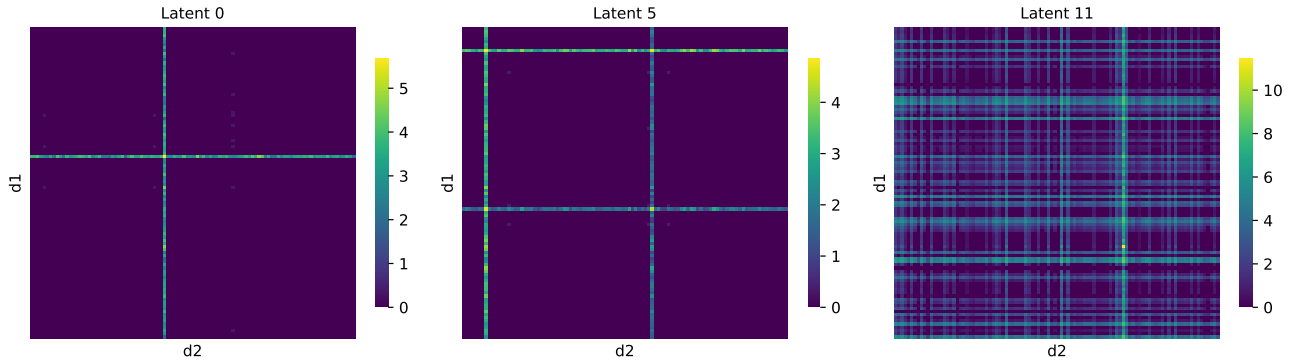

Figure 1. These 100×100 heatmaps show example SAE latent activations over all input bigrams (d_1, d_2) with $d_1, d_2 \in [0, 99]$: top-left cell is $(0, 0)$, bottom-right cell is $(99, 99)$. **Left** (1-symbol detector): latent 0 activates strongly if d_1 or $d_2 = 41$ and peaks at $d_1 = d_2 = 41$, producing a plus-sign pattern. **Centre** (k-symbol detector, $k > 1$): latent 5 activates strongly if d_1 or $d_2 \in \{7, 58\}$, producing two plus-sign patterns. **Right** (special latent): latent 11 activates on most inputs with generally much greater magnitudes than any other latent.

3.2. Steering Experiments

Steering latent 11 swaps for 50% of model outputs (Figure 2). There are several failure reasons (visualisations in Appendix D): **No activation (36%)**: Some inputs do not activate latent 11 ($m = 0$). **Negative scaling required (9.2%)**: Swapping o_1 requires negative p, which we reject as BatchTopK activations are non-negative by construction. **Mutually exclusive swap ranges (2.3%)**: Each output position can be individually swapped but not simultaneously. **No swap in observed range (1.8%)**: One logit remains dominant across all tested p. **Matching bigram elements (0.8%)**: $d_1 = d_2$, making the swap degenerate. It is sometimes possible to scale a co-activating non-special latent to change the logit distribution and successfully swap more outputs with latent 11. Therefore, 50% is not the ceiling (see Appendix D). Steering latent 56 swaps 2.0% of outputs and never coactivates with latent 11, suggesting they may be complementary. We additionally test steering non-special latents and do not find comparable behaviour (Appendix D). This suggests that the effect is specific to special latents.

Across inputs, logits at o_1 scale approximately linearly with p (Figure 2) and linear regression achieves mean $R^2 > 0.999$. We therefore estimate exact swap thresholds at o_1 analytically using lines fitted to the d_1 and d_2 logits.

3.3. Generalisation to other SAEs

Across 2,613 SAEs, high-performing ones frequently contain special latents, which we identify using $\Delta\alpha$ correlation $|r| > 0.3$ (visualised in Appendix E, Figure 12).

We implement a pipeline to identify special latents and steer them to swap model outputs (Appendix E). We run this pipeline on six selected high-performing checkpoints (Appendix B, Table 6). All six checkpoints learned at least one special latent. Five learned a pair of opposing special latents, one d_1-favouring and one d_2-favouring. Steering at least one special latent swapped a substantial proportion of outputs, ranging from 26.6% to 72.5% of the dataset (Appendix E, Table 10). Conditional on latent activation, these rates were substantially higher. Swap failures primarily occur for the same reasons as in Section 3.2. This suggests that steerable graded latents are not unique to a single BatchTopK checkpoint, although the limited subset of evaluated SAEs limits statistical significance. Further results are in Appendix E.

4. Discussion

We train SAEs on the residual stream of a transformer trained to solve an ordered bigram copying task. Most SAE latents behave like sparse symbol detectors: they activate selectively when particular symbols are present in the input. We also find graded latents that activate across a broad subset of the input space. Their activation magnitudes strongly correlate with $\Delta\alpha$: the difference between input attention weights in the first transformer layer. For these latents, different activation magnitudes correspond to different bigram orderings. Steering these latents also changes the model's output order. This suggests that activation magnitude is causally involved in the computation.

Consider trying to label these SAE latents using only top-activating and non-activating examples. This is common for large SAEs, including those trained on large language models (LLMs) (Paulo et al., 2025; McDougall et al., 2025). As suggested by Figure 1, this works for symbol detectors, but is much harder for graded latents which activate very often. Our results suggest that these graded latents track internal model quantities rather than discrete input features. In particular, latent activation magnitude appears to reflect the transformer's relative attention to the two input positions,

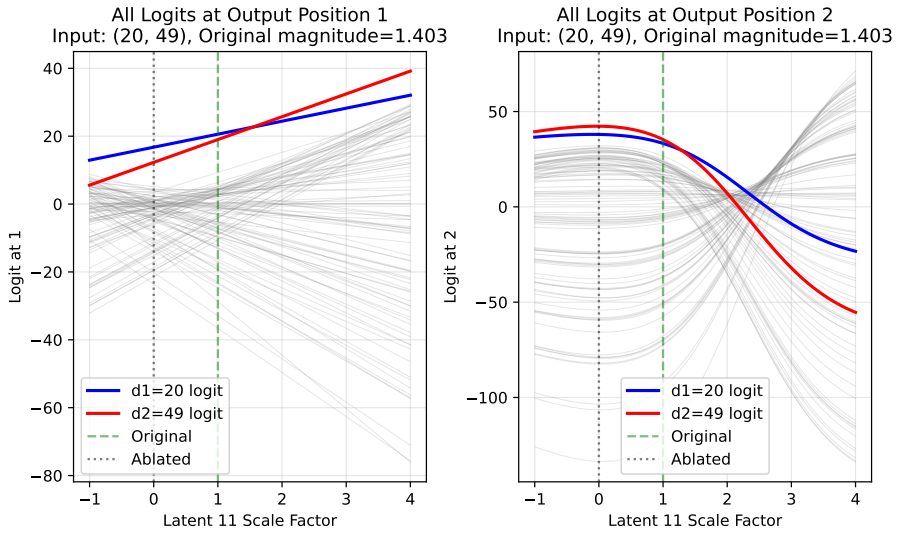

Figure 2. This plot shows the logits at output positions o_1 (left) and o_2 (right) for input $(20, 49)$ as latent 11's activation is scaled by factor p (x-axes). The original activation magnitude m for this input is 1.403, which corresponds to $p = 1$ on the x-axis. Each line shows the logit for one symbol; blue = d_1, red = d_2 and all others are grey. A valid swap requires the red line to lead at o_1 and the blue line to lead at o_2 simultaneously. Here, this holds for $p \in [0.16, 2.20]$.

not the presence of a specific symbol. If similar phenomena occur in large language models, graded latents of this kind may appear noisy, polysemantic, or uninterpretable under standard feature-dashboard analyses despite participating in coherent computations.

Our results do not contradict the view that SAEs recover meaningful linear structure in model activations. The graded latents still correspond to coherent decoder directions and appear causally connected to downstream model behaviour. The key distinction is that their semantic content is not fully captured by a set of activating inputs.

Our experiments are conducted in a deliberately simplified toy setting. This allows unusually precise mechanistic analysis and direct comparison of SAE latents against known internal quantities such as $\Delta\alpha$. On the other hand, the low-dimensional structure of the task may encourage unusually linear or disentangled representations that do not generalise to large pretrained language models. We therefore include preliminary evidence that similar graded latents can appear across multiple SAE architectures. These include JumpReLU and Matryoshka SAEs, the same SAE types used in Gemma Scope 1 (Lieberum et al., 2024) and 2 (McDougall et al., 2025) respectively. This supports further work, but does not show that the same behaviour occurs in large language models.

5. Conclusion and Future Work

In this paper, we showed that SAEs trained on a relative-magnitude composition mechanism can learn graded latents. These latents are not just active or inactive. Their activation magnitudes correlate with $\Delta\alpha$, and steering them predictably changes the transformer's output. We will release our trained SAEs on Weights & Biases to support future work and reproducibility of our results. Limitations are listed in Appendix F.

Future work should test whether graded latents appear in real-world pretrained SAEs. The Gemma Scope releases (Lieberum et al., 2024; McDougall et al., 2025) provide a natural test environment. They include JumpReLU and Matryoshka SAEs trained on Gemma 2 and Gemma 3 at multiple widths, matching the SAE variants studied here. The target would be to find SAE latents in pretrained models where the activation value, rather than just feature presence, causally affects downstream behaviour. Such experiments would be needed before treating the toy result as evidence about large pretrained models.

In summary, this paper gives a controlled example where SAE latent activation values matter for interpretation. We hope this case study motivates further work on magnitude-based feature composition in LLMs.

Impact Statement

This paper aims to improve our understanding of sparse autoencoders (SAEs), which are widely used in interpretability research. If the same behaviour as reported here appears in real-world language models, then SAE-based analyses may miss some mechanisms by treating latent activations as simple on/off feature presence indicators.

References

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

Brixi, G., Durrant, M. G., Ku, J., Naghipourfar, M., Poli, M., Sun, G., Brockman, G., Chang, D., Fanton, A., Gonzalez, G. A., King, S. H., Li, D. B., Merchant, A. T., Nguyen, E., Ricci-Tam, C., Romero, D. W., Schmok, J. C., Taghibakhshi, A., Vorontsov, A., Yang, B., Deng, M., Gorton, L., Nguyen, N., Wang, N. K., Pearce, M. T., Simon, E., Adams, E., Amador, Z. J., Ashley, E. A., Baccus, S. A., Dai, H., Dillmann, S., Ermon, S., Guo, D., Herschl, M. H., Ilango, R., Janik, K., Lu, A. X., Mehta, R., Mofrad, M. R. K., Ng, M. Y., Pannu, J., Ré, C., St. John, J., Sullivan, J., Tey, J., Viggiano, B., Zhu, K., Zynda, G., Balsam, D., Collison, P., Costa, A. B., Hernandez-Boussard, T., Ho, E., Liu, M.-Y., McGrath, T., Powell, K., Pinglay, S., Burke, D. P., Goodarzi, H., Hsu, P. D., and Hie, B. L. Genome modelling and design across all domains of life with evo 2. *Nature*, 03 2026. ISSN 1476-4687. doi: 10.1038/s41586-026-10176-5. URL https://doi.org/10.1038/s41586-026-10176-5.

Bussmann, B., Leask, P., and Nanda, N. Batchtopk sparse autoencoders. In *NeurIPS 2024 Workshop on Scientific Methods for Understanding Deep Learning*, 2024. URL https://openreview.net/forum?id=d4dpOCqybL.

Bussmann, B., Nabeshima, N., Karvonen, A., and Nanda, N. Learning multi-level features with matryoshka sparse autoencoders. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=m25T5rAy43.

Chanin, D., Wilken-Smith, J., Dulka, T., Bhatnagar, H., Golechha, S., and Bloom, J. I. A is for absorption: Studying feature splitting and absorption in sparse autoencoders. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=R73ybUciQF.

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.

Farrell, T., Leask, P., and Moubayed, N. A. Order by scale: Relative-magnitude relational composition in attention-only transformers. In *Socially Responsible and Trustworthy Foundation Models at NeurIPS 2025*, 2025. URL https://openreview.net/forum?id=vWRVzNtk7W.

Ferrando, J., Sarti, G., Bisazza, A., and Costa-Jussà, M. R. A primer on the inner workings of transformer-based language models. *arXiv preprint arXiv:2405.00208*, 2024.

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. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=tcsZt9ZNKD.

Huben, R., Cunningham, H., Smith, L. R., Ewart, A., and Sharkey, L. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=F76bwRSLeK.

Karvonen, A., Rager, C., Lin, J., Tigges, C., Bloom, J. I., Chanin, D., Lau, Y.-T., Farrell, E., McDougall, C. S., Ayonrinde, K., Till, D., Wearden, M., Conmy, A., Marks, S., and Nanda, N. SAEBench: A comprehensive benchmark for sparse autoencoders in language model interpretability. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=qrU3yNfX0d.

Leask, P., Bussmann, B., Pearce, M. T., Bloom, J. I., Tigges, C., Moubayed, N. A., Sharkey, L., and Nanda, N. Sparse autoencoders do not find canonical units of analysis. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=9ca9eHNrdH.

Leask, P., Nanda, N., and Moubayed, N. A. Inference-time decomposition of activations (ITDA): A scalable approach to interpreting large language models. In *Forty-second International Conference on Machine Learning*, 2025b. URL https://openreview.net/forum?id=4WMZqAoYi4.

Lieberum, T., Rajamanoharan, S., Conmy, A., Smith, L., Sonnerat, N., Varma, V., Kramar, J., Dragan, A., Shah, R., and Nanda, N. Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2. In Belinkov, Y., Kim, N., Jumelet, J., Mohebbi, H., Mueller, A., and Chen, H. (eds.), *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pp. 278–300, Miami, Florida, US, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.blackboxnlp-1.19. URL https:

//aclanthology.org/2024.blackboxnlp-1.19/.

Marks, S., Karvonen, A., and Mueller, A. dictionary_learning, 2024. URL https://github.com/saprmarks/dictionary_learning.

McDougall, C., Conmy, A., Kramár, J., Lieberum, T., Rajamanoharan, S., and Nanda, N. Gemma scope 2: Technical paper. Technical report, Google DeepMind, 9 2025. URL https://storage.googleapis.com/deepmind-media/DeepMind.com/Blog/gemma-scope-2-helping-the-ai-safety-community-deepen-understanding-of-complex-language-model-behavior/Gemma_Scope_2_Technical_Paper.pdf.

O'Brien, K., Majercak, D., Fernandes, X., Edgar, R. G., Bullwinkel, B., Chen, J., Nori, H., Carignan, D., Horvitz, E., and Poursabzi-Sangdeh, F. Steering language model refusal with sparse autoencoders. In *ICML 2025 Workshop on Reliable and Responsible Foundation Models*, 2025. URL https://openreview.net/forum?id=PMK1jdGQoc.

Paulo, G. S., Mallen, A. T., Juang, C., and Belrose, N. Automatically interpreting millions of features in large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=EemtbhJOXc.

Quirke, L., Shabalin, S., and Belrose, N. Binary sparse coding for interpretability. *arXiv preprint arXiv:2509.25596*, 2025.

Rajamanoharan, S., Conmy, A., Smith, L., Lieberum, T., Varma, V., Kramar, J., Shah, R., and Nanda, N. Improving sparse decomposition of language model activations with gated sparse autoencoders. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a. URL https://openreview.net/forum?id=zLBlin2zvW.

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*, 2024b.

Wattenberg, M. and Viégas, F. Relational composition in neural networks: A survey and call to action. In *ICML 2024 Workshop on Mechanistic Interpretability*, 2024. URL https://openreview.net/forum?id=zzCEiUIPk9.

A. Further comparison of SAE Types

Table 1. Comparison of some sparse autoencoder (SAE) variants.

Type	Description
Standard ReLU (Bricken et al., 2023)	Applies an L_1 penalty to hidden activations using a standard ReLU activation function to encourage sparsity.
Gated (Rajamanoharan et al., 2024a)	Decouples feature detection from magnitude estimation using a gated mechanism to prevent the shrinkage issues caused by standard L_1 penalties.
JumpReLU (Rajamanoharan et al., 2024b)	Uses a discontinuous step function at a learned per-feature threshold, approximating an L_0 penalty through straight-through estimation. Notably used in Lieberum et al. (2024).
BatchTopK (Bussmann et al., 2024)	Retains top $n \times k$ activations over a batch of n inputs and sets the rest to zero, where k is a tuned hyperparameter. This forces an average of k latent activations per input, so the actual L_0 sparsity is approximately k. Notably used in Brixi et al. (2026); Karvonen et al. (2025).
Matryoshka (Bussmann et al., 2025)	Trains nested sub-dictionaries of increasing size that share a single encoder/decoder, producing multiple resolutions of feature granularity in one run. Notably used in McDougall et al. (2025).

An SAE consists of a single hidden layer trained to reconstruct model activations subject to a sparsity penalty on the latent activations (typically L_1 or top-k regularisation) (Huben et al., 2024; Bricken et al., 2023). Many SAE variants exist in the literature, some of which are shown in Table 1.

We use three types of SAEs in this paper. JumpReLU SAEs (Rajamanoharan et al., 2024b) use a discontinuous step function at a learned threshold and optimise an L_0 penalty. BatchTopK SAEs (Bussmann et al., 2024) keep the top $n \times k$ activations over a batch of n inputs and set the rest to zero. This forces an average of k latent activations per input, so the actual L_0 sparsity is approximately k. Matryoshka SAEs (Bussmann et al., 2025) train nested sub-dictionaries of increasing size under a shared encoder and decoder, producing representations at multiple levels of granularity from a single training run.

B. Experimental Setup

Table 2. Model and training configuration from Farrell et al. (2025). The model achieved 91.45% test accuracy using an 80/20 train/test split.

Parameter	**Value**
Number of layers	2
Number of heads	1
Residual stream dimension	64
Attention head dimension	64
LayerNorm	None
Bias	None
Value matrix (W_V)	Identity (frozen)
Output matrix (W_O)	Identity (frozen)
Learning rate	1×10^{-3}
Optimiser	`AdamW`
Batch size	128
Betas	(0.9, 0.999)
Weight decay	0.01

Table 3. BatchTopK SAE sweep results aggregated over multiple seeds (mean ± std). L_0: mean active features per token. d_{SAE}: dictionary size. **Loss Recovered**: fraction of model loss recovered by the SAE reconstruction (↑ better). **Dead %**: percentage of dictionary features with zero activation across the evaluation set (↓ better). **Bold**: best within each L_0 block; **underlined**: best overall.

L_0	d_{SAE}	Runs	Loss Recovered (↑)	Dead % (↓)
1	128	30	0.9392 ± 0.0020	$\mathbf{12.4 \pm 2.5}$
1	192	30	$\mathbf{0.9410 \pm 0.0015}$	43.6 ± 1.6
1	256	24	0.9405 ± 0.0018	54.3 ± 2.3
1	320	15	0.9397 ± 0.0015	61.9 ± 1.3
2	128	28	$\mathbf{0.9816 \pm 0.0021}$	$\mathbf{17.5 \pm 1.6}$
2	192	29	0.9812 ± 0.0025	42.3 ± 1.3
2	256	23	0.9815 ± 0.0020	54.6 ± 1.5
2	320	15	0.9802 ± 0.0025	61.6 ± 1.8
3	128	31	0.9966 ± 0.0122	$\mathbf{14.2 \pm 2.8}$
3	192	30	0.9979 ± 0.0091	35.7 ± 5.0
3	256	19	0.9963 ± 0.0145	47.7 ± 6.1
3	320	13	$\mathbf{0.9996 \pm 0.0001}$	54.1 ± 6.9
4	128	31	$\mathbf{0.9996 \pm 0.0000}$	$\mathbf{1.6 \pm 1.9}$
4	192	23	$\mathbf{0.9996 \pm 0.0000}$	5.0 ± 1.9
4	256	24	$\mathbf{0.9996 \pm 0.0001}$	19.0 ± 15.9
4	320	16	$\mathbf{0.9996 \pm 0.0001}$	19.4 ± 12.6
5	128	30	0.9996 ± 0.0000	$\underline{\mathbf{0.3 \pm 0.5}}$
5	192	38	0.9996 ± 0.0000	4.4 ± 6.0
5	256	27	0.9996 ± 0.0000	7.6 ± 3.2
5	320	15	$\underline{\mathbf{0.9997 \pm 0.0000}}$	11.8 ± 3.2

C. Further SAE Latent Analysis

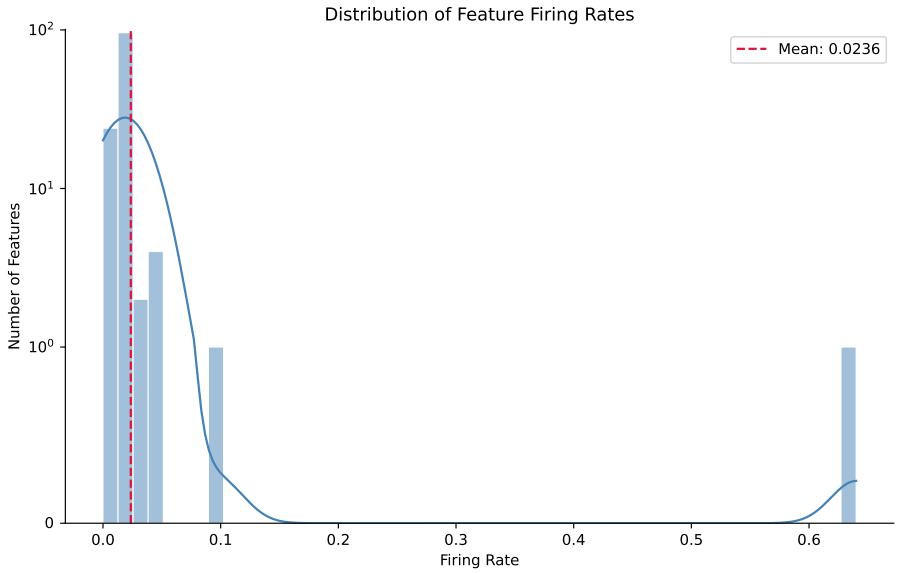

Figure 3. This histogram shows the distribution of firing rates for the SAE checkpoint's latents. The y-axis uses a symmetric logarithmic scale with a linear scale between $[-1, 1]$. A given latent's firing rate is the proportion of inputs which cause that latent to activate. The special latents are the only ones with a firing rate greater than one standard deviation from the mean (mean: 0.0236, standard deviation: 0.0559): latent 11 has rate 0.6400, latent 56 has rate 0.0991.

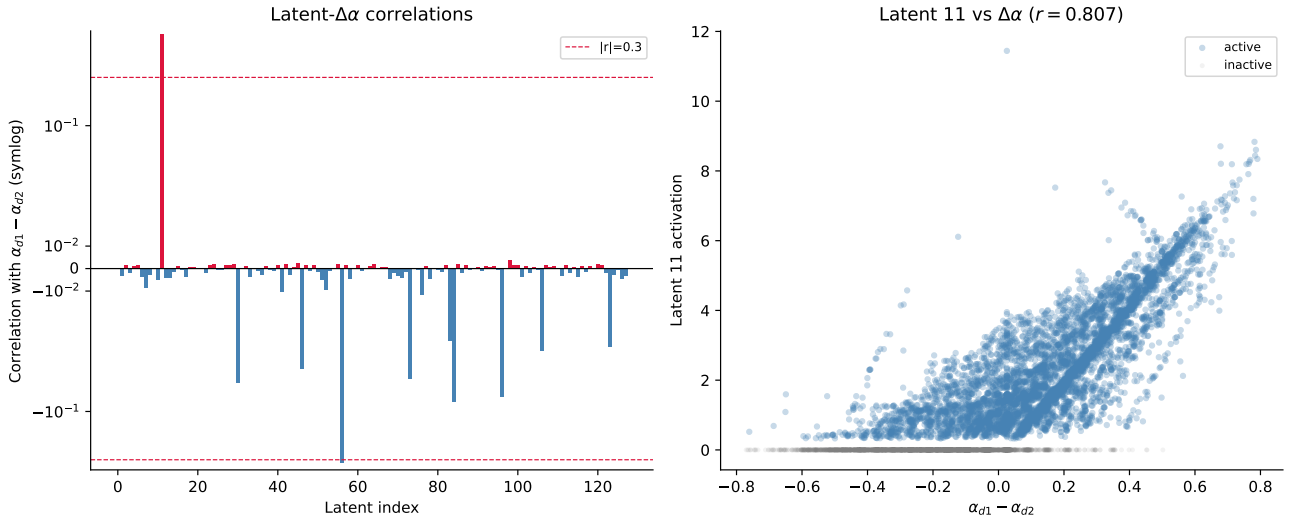

Figure 4. **Left**: This plot shows the Pearson correlation of each latent with $\Delta\alpha = \alpha_{s\to d_1} - \alpha_{s\to d_2}$. This quantity measures how much more the separator token attends to d_1 than to d_2 in the first transformer layer. The y-axis uses a symmetric logarithmic scale with a linear scale between $[-0.05, 0.05]$. d_1-favouring latents are highlighted red and d_2-favouring latents are highlighted blue. The top three strongest correlations are latent 11 with $r = 0.807$, latent 56 with $r = -0.3213$, and latent 96 with $r = 0.0036$. **Right**: This plot shows the correlation of latent 11 with $\Delta\alpha$ in more detail. All 10,000 possible input bigrams are plotted, and highlighted blue if they activate latent 11 and grey otherwise. The latent activates with greater magnitude for larger $\Delta\alpha$.

Table 4. Selected SAE configuration which achieves over 0.999 loss recovered, 3.02 sparsity and 3.1% dead latents. The optimiser, betas, schedule and auxiliary loss are fixed in the implementation (Marks et al., 2024) and excluded here for brevity.

Parameter	**Value**
SAE type	BatchTopK
Input dimension (d_{model})	64
Dictionary size (d_{SAE})	128
k	3
Actual sparsity (L_0)	3.02
Activation centring	Mean-centred (pre-training)
Encoder init	$W_{\text{enc}} = W_{\text{dec}}^{\top}$
Decoder normalisation	Unit-norm columns
Learning rate	3×10^{-4}
Batch size	4096
Training steps	50,000
Seed	1
Hardware	NVIDIA GeForce RTX 2080 Ti

Table 5. SAE hyperparameter sweep configurations. Braces denote swept values; fixed values are listed directly. Fixed parameters across all runs: 50,000 training steps, 1,000 warmup steps, 4,096 batch size. The sweeps are ongoing, so the counts below cover the 2,613 trained checkpoints rather than completed grid sizes. Note that $n_{\text{groups}} = 1$ in a Matryoshka SAE reduces it to a standard BatchTopK SAE, so this serves as an internal consistency check.

Parameter	**Architecture**	**Values**
d_{SAE}	All	$\{128, 192, 256, 320\}$
k / Target L_0	All	$\{1, 2, 3, 4, 5\}$
Learning rate	BatchTopK, Matryoshka	3×10^{-4}
	JumpReLU	$\{7\times10^{-5}, 3\times10^{-4}\}$
Sparsity penalty	JumpReLU	$\{0.1, 0.5, 1.0, 5.0\}$
n_{groups}	Matryoshka	$\{1, 2, 3, 4\}$
Architecture	**Seeds**	**Total runs**
BatchTopK	18	492
JumpReLU	15	1,201
Matryoshka	15	920

D. Further Latent Steering Examples

Figure 5 shows an example where swapping o_1 requires negative scaling ($p < 0$). The logit for d_2 at o_1 only overtakes d_1 when the latent activation is suppressed below zero, which is outside the valid activation range of a BatchTopK SAE.

Figure 6 shows an example where the swap ranges for o_1 and o_2 are mutually exclusive: there exists a positive p that swaps each position individually, but no single p swaps both simultaneously.

Figures 7 and 8 show examples where no swap occurs across the full observed range $p \in [0, 10]$. In Figure 7, the logit for d_1 remains dominant at o_1 for all p; Figure 8 shows the analogous failure at o_2.

In limited experiments with inputs that fail for the first three causes, it is sometimes possible to scale a co-activating non-special latent and successfully swap outputs. Figures 9, 10 and 11 show one example. Input $(41, 49)$ activates latents 11 (magnitude 3.1), 0 (magnitude 3.1) and 55 (magnitude 2.5). The swap fails with latent 11 because there are no positive overlapping scale ranges. However, scaling latent 55 by $p \in [1.1, 1.15]$ successfully produces a swap. This does not necessarily indicate that latent 55 is graded. Instead, scaling it alters the logit distribution, which opens a valid swap zone for latent 11. Similarly, running the steering pipeline on latent 55 alone swaps only 4 inputs, all of which co-activate latent 11. This suggests a similar mechanism. This preliminary experiment is insufficient for statistical significance, but it suggests

Table 6. Comparison of selected high-performing SAE checkpoints across configuration and performance metrics. These checkpoints were chosen for the steering analysis to span architecture, dictionary size and sparsity settings while keeping reconstruction performance high. Dashes indicate parameters not applicable to a given architecture. L_0 (actual) is the mean number of active latents measured at eval time. LR is learning rate. **Bold**: best value per performance metric.

SAE Type	Configuration					Performance		
	d_{SAE}	k / Target L_0	LR	Sp. Pen.	n_{groups}	Loss Recovered ↑	Dead (%) ↓	L_0 (actual) ↓
BatchTopK (Studied)	128	3	3×10^{-4}	—	—	0.9991	3.1	**3.02**
BatchTopK	192	3	3×10^{-4}	—	—	0.9997	26.0	3.36
	128	5	3×10^{-4}	—	—	0.9996	**0.8**	4.90
JumpReLU	256	5	7×10^{-5}	5	—	**0.9998**	50.8	4.03
	128	5	7×10^{-5}	5	—	**0.9998**	18.0	3.24
Matryoshka	256	3	3×10^{-4}	—	2	0.9997	39.1	3.32
	192	4	3×10^{-4}	—	4	0.9996	12.5	3.98

Table 7. JumpReLU SAE sweep results aggregated over multiple seeds (mean ± std). L_0: mean active features per token. d_{SAE}: dictionary size. **Loss Recovered**: fraction of model loss recovered by the SAE reconstruction (↑ better). **Dead %**: percentage of dictionary features with zero activation across the evaluation set (↓ better). **Bold**: best within each L_0 block; **underlined**: best overall.

L_0	d_{SAE}	Runs	Loss Recovered (↑)	Dead % (↓)
2	128	15	$\mathbf{0.9940 \pm 0.0017}$	$\mathbf{19.3 \pm 0.7}$
2	192	5	0.9918 ± 0.0000	46.9 ± 0.0
2	256	13	$\mathbf{0.9940 \pm 0.0025}$	56.7 ± 2.6
2	320	5	0.9907 ± 0.0000	67.8 ± 0.0
3	128	333	0.9984 ± 0.0020	$\underline{\mathbf{18.4 \pm 1.2}}$
3	192	320	0.9984 ± 0.0020	42.1 ± 1.8
3	256	305	$\underline{\mathbf{0.9985 \pm 0.0020}}$	52.0 ± 3.4
3	320	148	0.9973 ± 0.0025	60.1 ± 4.8
4	192	35	$\mathbf{0.9984 \pm 0.0022}$	$\mathbf{40.5 \pm 3.3}$
4	256	15	0.9967 ± 0.0023	55.2 ± 3.8
4	320	7	0.9958 ± 0.0018	62.6 ± 3.9

Table 8. Matryoshka SAE sweep results aggregated over multiple seeds (mean ± std). L_0: mean active features per token. d_{SAE}: dictionary size. **Loss Recovered**: fraction of model loss recovered by the SAE reconstruction (↑ better). **Dead %**: percentage of dictionary features with zero activation across the evaluation set (↓ better). **Bold**: best within each L_0 block; **underlined**: best overall.

L_0	d_{SAE}	Runs	Loss Recovered (↑)	Dead % (↓)
1	128	45	0.9353 ± 0.0019	$\mathbf{13.7 \pm 1.7}$
1	192	45	0.9362 ± 0.0017	42.5 ± 1.3
1	256	36	$\mathbf{0.9373 \pm 0.0015}$	53.8 ± 1.6
2	128	45	0.9710 ± 0.0036	$\mathbf{15.3 \pm 2.9}$
2	192	45	$\mathbf{0.9721 \pm 0.0034}$	41.8 ± 2.6
2	256	35	0.9711 ± 0.0033	54.5 ± 3.4
3	128	58	0.9989 ± 0.0005	$\mathbf{12.0 \pm 3.0}$
3	192	59	0.9993 ± 0.0003	37.6 ± 5.3
3	256	57	$\mathbf{0.9995 \pm 0.0002}$	46.9 ± 7.2
3	320	40	$\mathbf{0.9995 \pm 0.0001}$	50.0 ± 9.5
4	128	61	0.9992 ± 0.0003	$\mathbf{2.8 \pm 2.6}$
4	192	58	0.9995 ± 0.0001	15.0 ± 9.2
4	256	61	$\underline{\mathbf{0.9996 \pm 0.0001}}$	13.6 ± 11.7
4	320	48	$\underline{\mathbf{0.9996 \pm 0.0001}}$	15.1 ± 13.3
5	128	61	0.9992 ± 0.0003	$\underline{\mathbf{0.6 \pm 0.6}}$
5	192	61	0.9995 ± 0.0001	8.0 ± 6.0
5	256	61	$\underline{\mathbf{0.9996 \pm 0.0001}}$	5.9 ± 3.8
5	320	44	$\underline{\mathbf{0.9996 \pm 0.0001}}$	6.6 ± 4.9

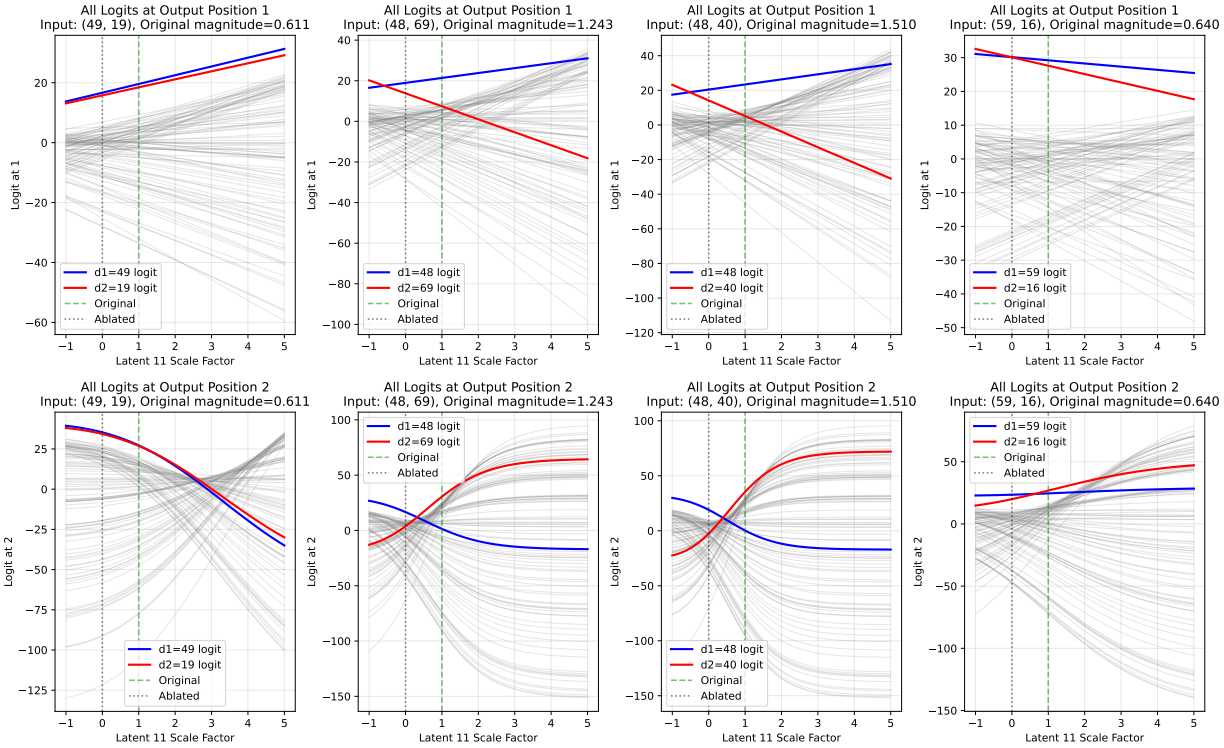

Figure 5. For 9.2% of inputs, latent 11 (in the BatchTopK SAE from Section 2) must be negatively scaled to swap the predicted symbol at one of the output positions.

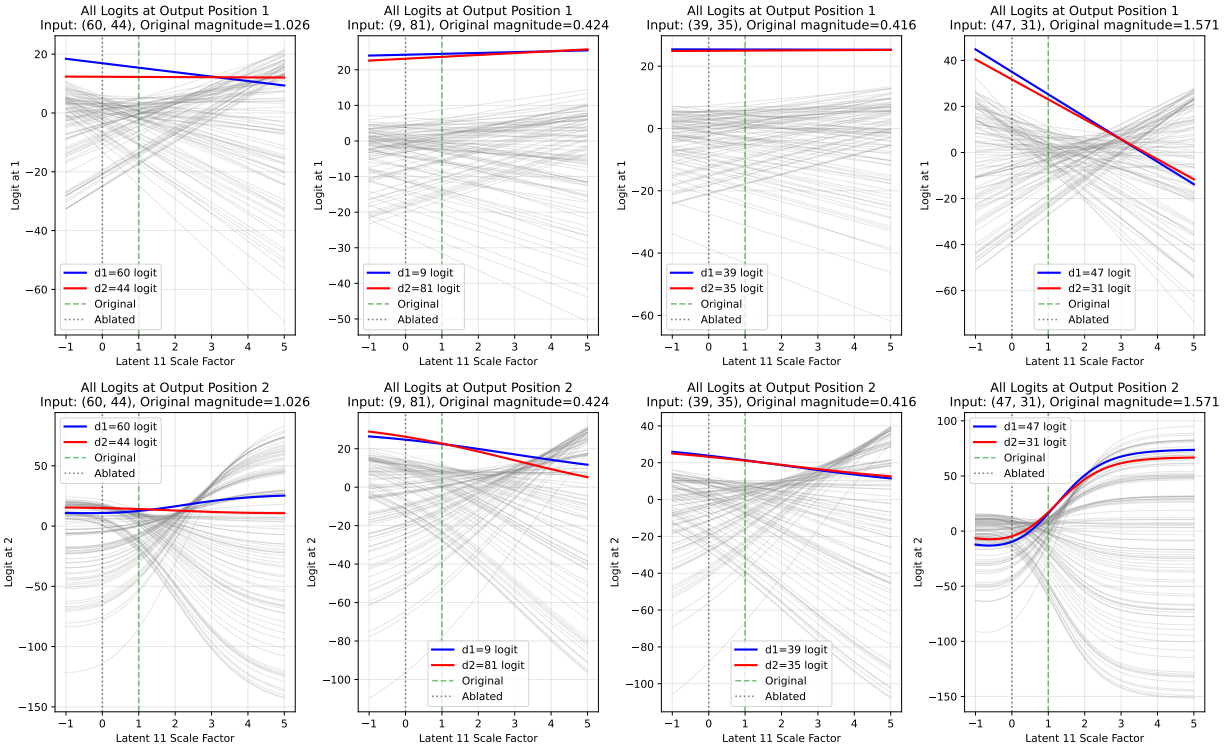

Figure 6. For 2.3% of inputs, it is possible to scale latent 11 (in the BatchTopK SAE from Section 2) to swap the symbol at each output position individually but not simultaneously.

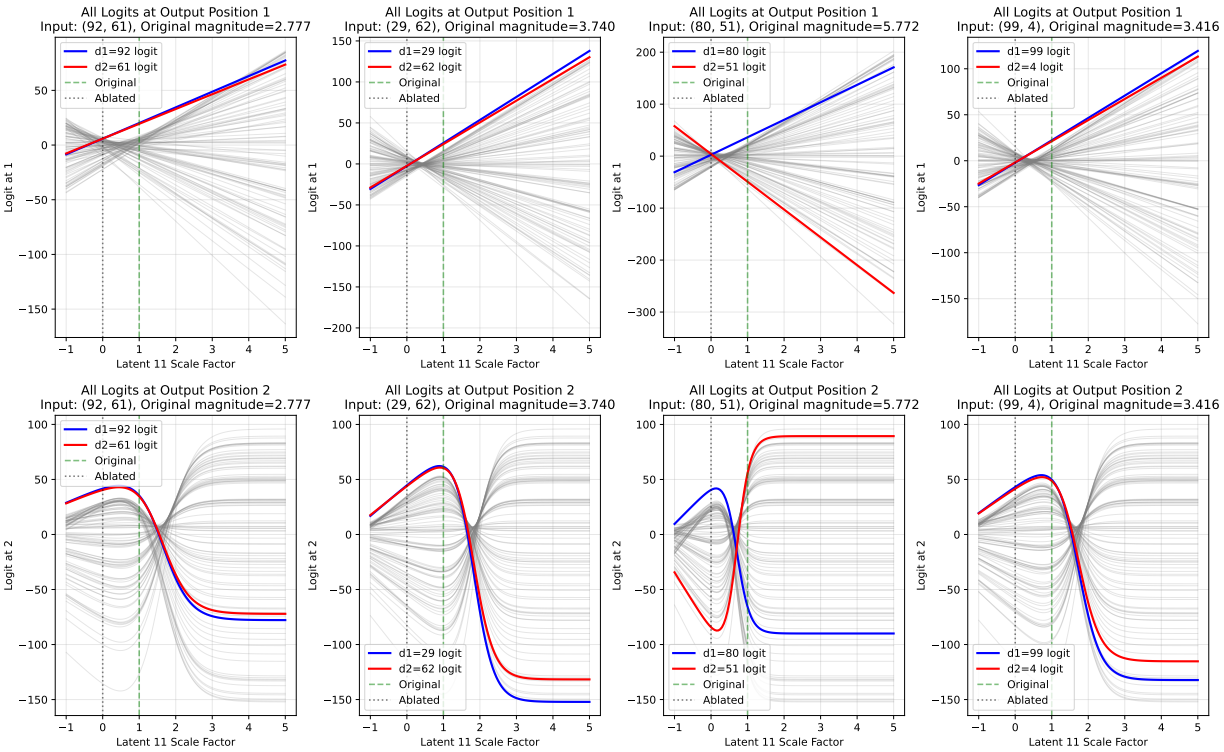

Figure 7. For 1.6% of inputs, the logit for the symbol at d_2 is always dominant at o_2 in the observed latent scale range (in the BatchTopK SAE from Section 2), so there is no scale where the symbol at d_1 is predicted instead.

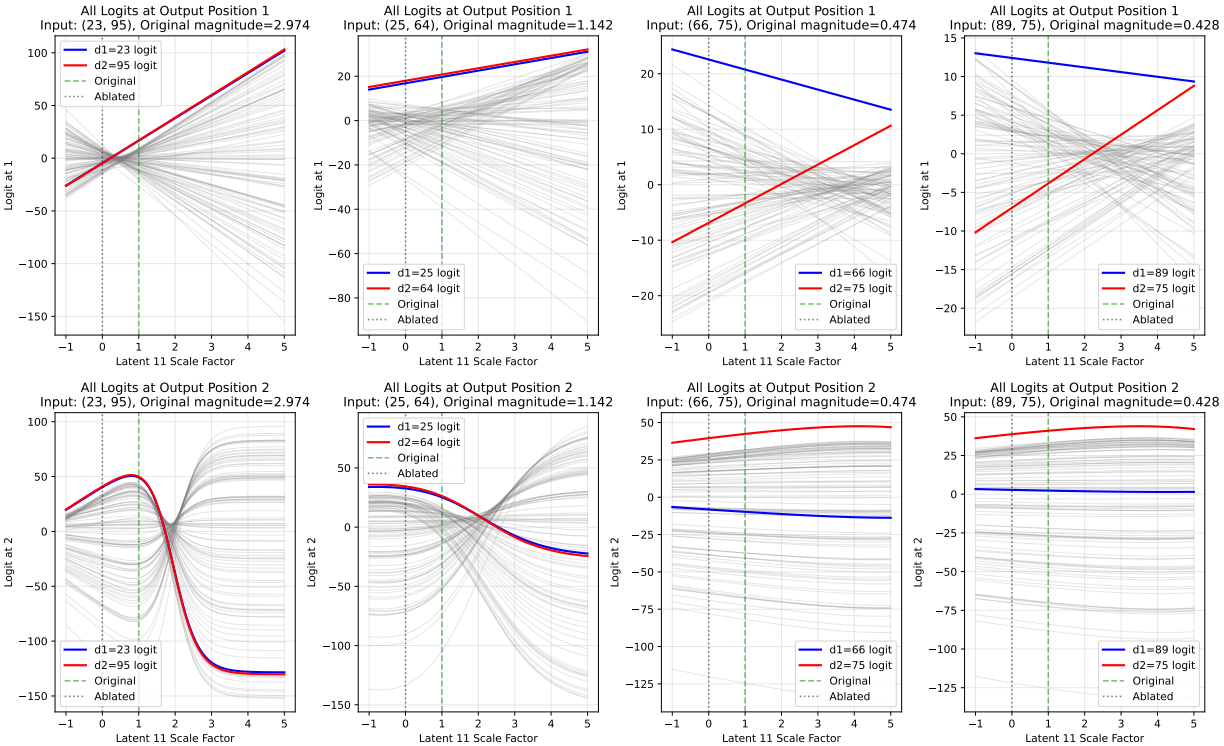

Figure 8. For 0.2% of inputs, the logit for the symbol at d_1 is always dominant at o_1 in the observed latent scale range (in the BatchTopK SAE from Section 2), so there is no scale where the symbol at d_2 is predicted instead.

that scaling multiple latents together may enable more output swaps than steering special latents alone.

We do not attempt steering on all latents due to computational constraints. Instead, we test five randomly selected latents (0, 55, 63, 82, 117), latent 96 (the non-special latent with highest $\Delta\alpha$ correlation, $r = 0.0036$), and latent 5 (shown in Figure 1). Results are shown in Table 9. Every successful swap using a non-special latent has a co-activating special latent. This suggests that these non-special latents are unlikely to be graded. Instead, they may alter the logit distribution so that graded special latents can successfully swap outputs.

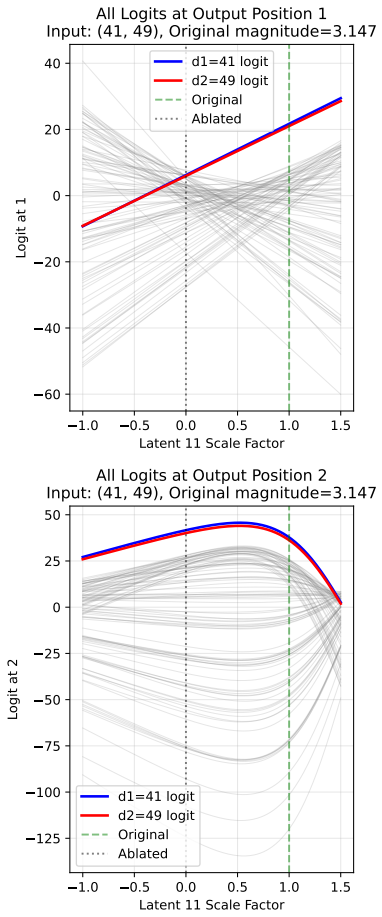

Figure 9. In the BatchTopK SAE from Section 2, latent 11 cannot be scaled to swap input bigram $(41, 49)$ because the logit for 49 is never argmax in position o_1, as shown by the top plot.

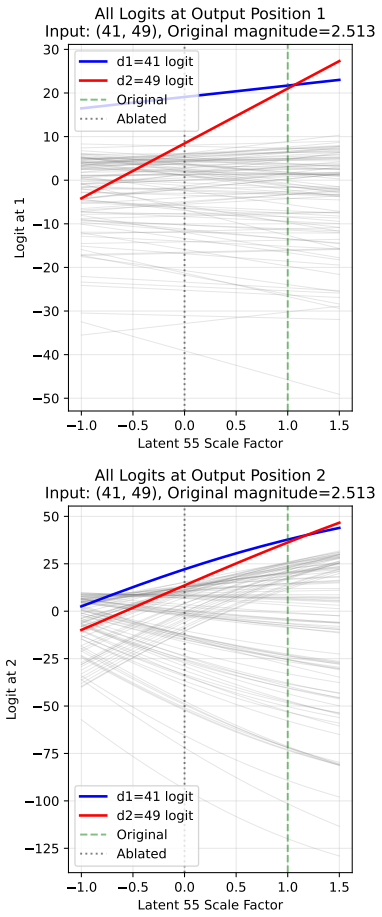

Figure 10. However, in the same SAE, latent 55 co-activates with latent 11 for input $(41, 49)$ and can be scaled by factor $p \in [1.07, 1.18]$ to cause the model to output $(49, 41)$ instead.

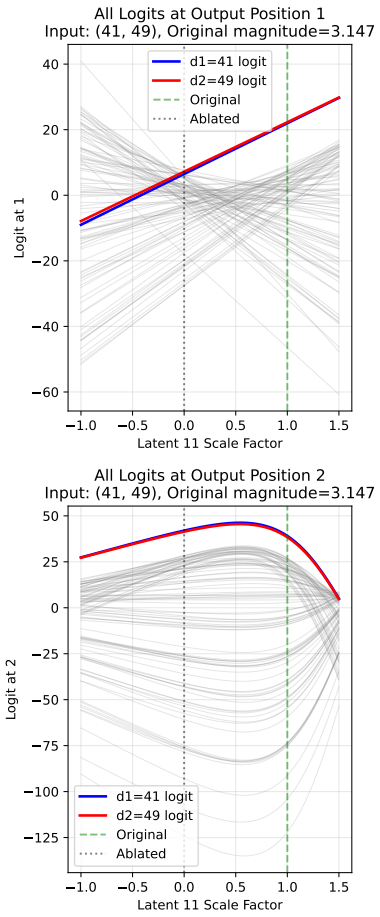

Figure 11. If latent 55's activation magnitude is first scaled by a factor of 1.1, then latent 11 can be scaled by factor $p \in [0.03, 1.45]$ to swap the outputs. This was not possible without scaling latent 55. This suggests that some graded activation demonstrations are only possible when multiple active latents are scaled simultaneously.

Table 9. We perform steering experiments on several chosen and randomly sampled SAE latents to demonstrate how steering with special latents (**bold**) compares to non-special latents. We record how many outputs are successfully swapped by steering only that latent, how many do not activate the latent, and how many activate the latent but cannot be swapped. We additionally report the proportion of the activating inputs that are swapped (Cond. Swap).

Latent	Swapped	No Activation	Active & No Swap	Cond. Swap
0	6 (0.1%)	9786 (97.9%)	208 (2.1%)	2.8%
5	23 (0.2%)	9594 (95.9%)	383 (3.8%)	5.7%
11	**4976 (49.8%)**	**3600 (36.0%)**	**1424 (14.2%)**	**77.8%**
55	4 (0.0%)	9787 (97.9%)	209 (2.1%)	1.9%
56	**196 (2.0%)**	**9009 (90.1%)**	**795 (7.9%)**	**19.8%**
63	4 (0.0%)	9801 (98.0%)	195 (2.0%)	2.0%
82	4 (0.0%)	9787 (97.9%)	209 (2.1%)	1.9%
96	0 (0.0%)	9974 (99.7%)	26 (0.3%)	0.0%
117	37 (0.4%)	9782 (97.8%)	181 (1.8%)	17.0%

E. Further Generalisation Experiment Details

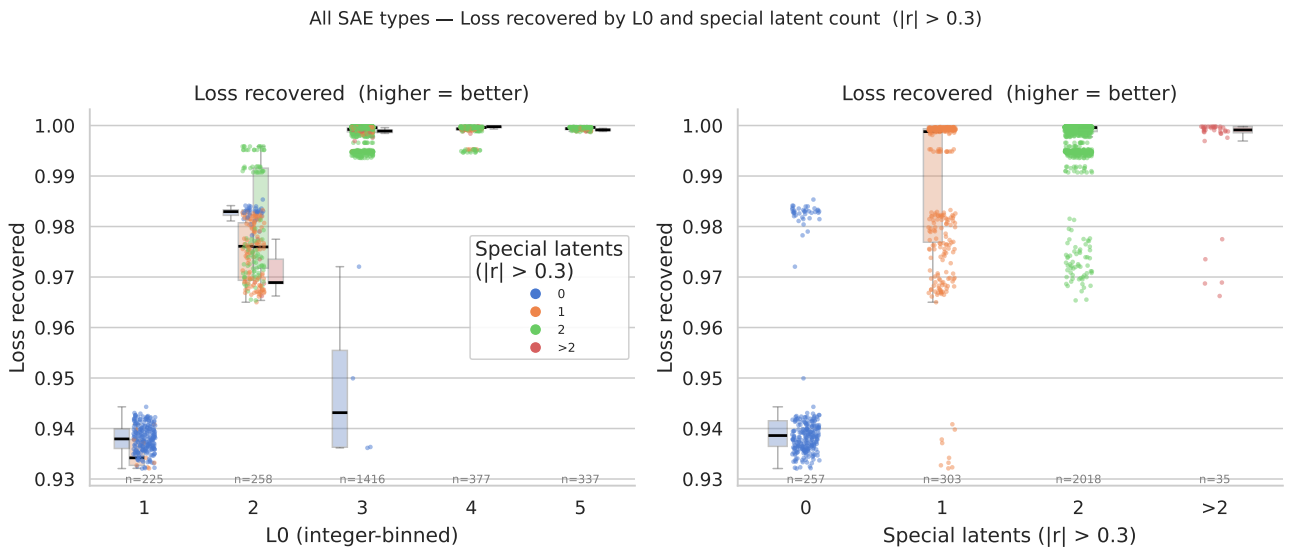

Figure 12. These strip plots show how the number of special latents (identified by the latent activation's correlation with $\Delta\alpha$ using threshold $|r| > 0.3$, as in Section 3.1) differs across all 2,613 trained SAEs from the sweep in Appendix B, Table 5. **Left**: A large majority of the SAEs achieve $L0 \approxeq 3$. However, only those with special latents achieve high performance. **Right**: SAEs with no special latents tend to perform much worse than those with them. Most SAEs learn two special latents.

To test whether special latents can be steered to swap outputs in other SAEs, we use the following pipeline. The implementation is available in our public codebase.

1. Store the SAE activations for all input bigrams by running a single forward pass over the full dataset and recording each SAE latent's activation magnitude.
2. Identify special latents using $\Delta\alpha$ correlation $|r| > 0.3$ as a heuristic, as described previously.
3. For each identified special latent, run a batched steering grid search over all input bigrams. For each bigram and each scale factor $p \in [0, 10]$ with step size 0.05 (201 values), record the logits at output positions o_1 and o_2 under the steered reconstruction $\hat{\boldsymbol{r}}_s^{L_1}$. Bigrams for which the latent does not activate ($m = 0$) are skipped immediately since scaling has no effect.
4. Find crossover scales for each bigram:

Figure 13. These strip plots show how the number of special latents (identified by the latent activation's correlation with $\Delta\alpha$ using threshold $|r| > 0.3$, as in Section 3.1) differs across all **BatchTopK SAEs** from the sweep in Appendix B, Table 5.

Figure 14. These strip plots show how the number of special latents (identified by the latent activation's correlation with $\Delta\alpha$ using threshold $|r| > 0.3$, as in Section 3.1) differs across all **JumpReLU SAEs** from the sweep in Appendix B, Table 5.

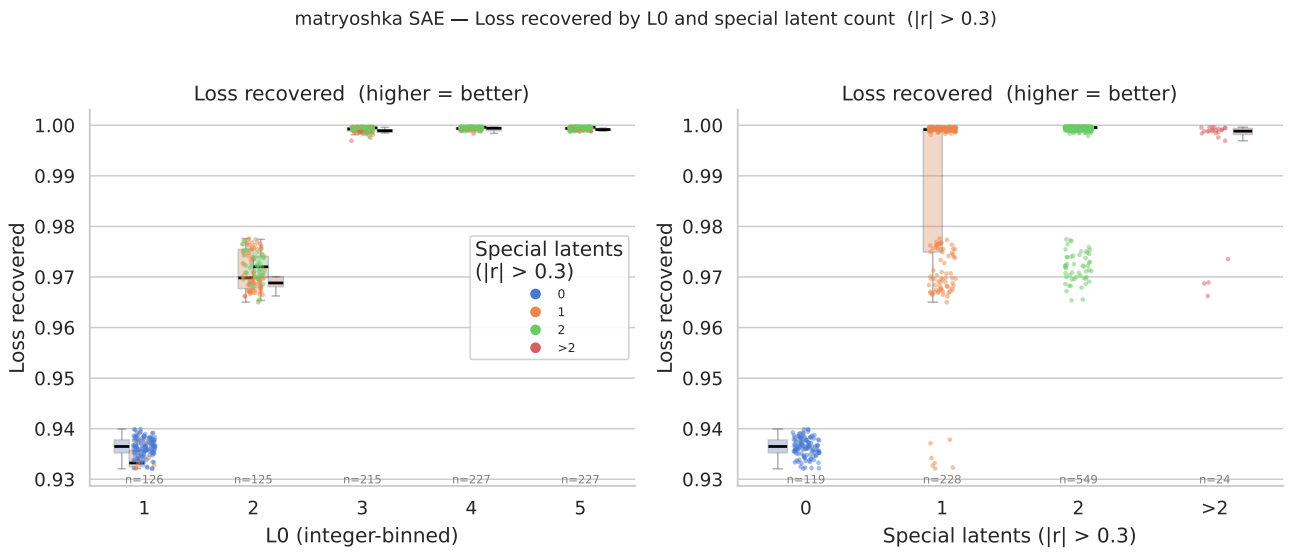

Figure 15. These strip plots show how the number of special latents (identified by the latent activation's correlation with $\Delta\alpha$ using threshold $|r| > 0.3$, as in Section 3.1) differs across all **Matryoshka SAEs** from the sweep in Appendix B, Table 5.

- For o_1: since o_1 logits are empirically linear in p, fit lines to $\ell_{d_1}(p)$ and $\ell_{d_2}(p)$ and compute the analytical intersection. Inputs for which either logit series fails an $R^2 \geq 0.999$ linearity check, or for which the intersection falls on a negative scale, are flagged and excluded.
- For o_2: detect sign changes in $\ell_{d_1}(p) - \ell_{d_2}(p)$ on the coarse $[0, 10]$ grid, then refine each crossing to three decimal places using bisection search. All bisection tasks across the batch are executed in parallel to avoid sequential per-input forward passes.

5. Determine a valid swap zone $[p_{\text{lo}}, p_{\text{hi}}]$ per bigram by intersecting the o_1 and o_2 crossover constraints with the scale ranges over which $\text{argmax}(o_1) = d_2$ and $\text{argmax}(o_2) = d_1$ hold simultaneously. If no contiguous window satisfying both conditions exists, the bigram is recorded as a failure with a specific reason (see Section 3.2).

6. Verify each swap zone by running the model at the midpoint $p^* = (p_{\text{lo}} + p_{\text{hi}})/2$ and confirming that the model output is (d_2, d_1).

7. Report the fraction of bigrams for which a valid swap zone was found and the swap verified, along with a breakdown of failure reasons for the remainder.

Steps 3–5 of the pipeline are very computationally expensive. We therefore run the steering analysis on the primary BatchTopK SAE and six additional selected high-performing checkpoints (Appendix B, Table 6), rather than on all SAEs in Figure 12. These checkpoints include two BatchTopK, two JumpReLU and two Matryoshka SAEs, while varying dictionary size and sparsity.

Table 10 shows the results when trying to swap model outputs using the above pipeline on a selected subset of SAEs. In future work, this study should be extended to further configurations for robust statistical significance.

Table 10. Generalisation of special latent steering to the selected SAE checkpoints from Appendix B, Table 6. For each identified special latent ($|r| > 0.3$), we report its bias, Pearson correlation r with $\Delta\alpha$, and the percentage of inputs where the latent was successfully steered to swap outputs (Success), did not activate (No Act.), or activated but could not be swapped (Active & No Swap). Cond. Success is success conditional on the latent activating: $\text{Success}/(\text{Success} + \text{Active No Swap})$.

SAE Type	d_{SAE}	L_0	Latent	Type	r	Steering Outcomes (%)			
						Success	No Act.	Active No Swap	Cond. Success
BatchTopK (Studied)	128	3.00	11	d_1-favouring	$+0.807$	49.8%	36.0%	14.2%	77.8%
			56	d_2-favouring	-0.321	2.0%	90.1%	7.9%	20.2%
BatchTopK	192	3.36	47	d_2-favouring	-0.836	52.9%	34.4%	12.7%	80.6%
			116	d_1-favouring	$+0.365$	0.5%	91.7%	7.8%	6.0%
	128	4.90	68	d_1-favouring	$+0.845$	32.9%	44.7%	22.4%	59.5%
			64	d_2-favouring	-0.730	6.0%	62.0%	32.0%	15.8%
JumpReLU	256	4.03	195	d_1-favouring	$+0.849$	38.8%	39.8%	21.4%	64.5%
			179	d_2-favouring	-0.721	5.8%	60.2%	34.0%	14.6%
	128	3.24	105	d_2-favouring	-0.816	28.6%	37.8%	33.6%	46.0%
			4	d_1-favouring	$+0.756$	6.9%	62.3%	30.8%	18.3%
Matryoshka	256	3.32	48	d_2-favouring	-0.796	26.6%	50.6%	22.8%	53.8%
			62	d_1-favouring	$+0.591$	6.0%	71.2%	22.8%	20.8%
	192	3.98	30	d_1-favouring	$+0.866$	72.5%	16.5%	11.0%	86.8%

F. Limitations

A. **Minimal toy model.** The transformer is trained on a single narrow task and omits components standard in pretrained LLMs (MLPs, LayerNorm, multi-head attention, value and output projections), with a vocabulary of only 100 symbols. Whether relative magnitude-based composition appears in less constrained models remains an open question.

B. **Single-latent steering.** The steering pipeline handles one special latent at a time. The co-activation experiments in Appendix D suggest that simultaneous multi-latent steering could recover additional swaps.

C. **SAE selection bias.** Our primary SAE was selected for having few dead latents, biasing toward high-quality SAEs. A fully representative study would require analysis across a random sample of all trained SAEs, not just well-performing ones.

D. **Limited SAE steering generalisation.** Special latents emerge across BatchTopK, JumpReLU, and Matryoshka architectures in the broader correlation analysis, but steering was tested on only seven selected checkpoints (including the primary BatchTopK SAE). Broader statistical significance testing across more configurations is needed.