Stitching Sparse Autoencoders of Different Sizes

Anonymous Author(s)
Affiliation
Address
email

Abstract

Sparse autoencoders (SAEs) are a promising method for decomposing the activations of language models into a learned dictionary of latents, the size of which is a key hyperparameter. However, the effect of the dictionary size hyperparameter on the learned latents remains poorly understood. In this work, we investigate how increasing the dictionary size of SAEs trained on the activations of GPT-2 and Pythia-410M affects their latents. We find that latents in SAEs fall into two distinct categories. There are reconstruction latents that are either present in smaller SAEs or are more fine-grained versions of them, but we also find novel latents that capture information missed by smaller SAEs. Novel latents can be inserted into a smaller SAE to improve performance, while reconstruction latents degrade it. The existence of novel latents when larger SAEs are trained suggests that researchers may be using SAEs which miss out on features crucial to the task studied. The category of a latent can be effectively predicted with the cheap proxy of taking the maximum cosine similarity with each latent in the smaller SAE's decoder: novel latents have low cosine similarity, whereas reconstruction have high. Utilizing this insight, we introduce SAE stitching: a method that inserts or swaps novel latents from a larger SAE into a smaller one, allowing for smooth interpolation between SAE sizes with monotonically decreasing reconstruction error. Our findings shed light on the trade-offs between dictionary size, sparsity, and reconstruction performance in SAEs, enhancing the understanding of feature learning in these models.

1 Introduction

Mechanistic interpretability aims to reverse-engineer neural networks into human-interpretable algorithms [5]. Sparse autoencoders (SAEs) have emerged as a promising tool for recovering monosemantic and interpretable features from the activations of large language models [2, 3]. A key hyperparameter in SAEs is the dictionary size, which determines the number of latents the SAE can learn. Despite its importance, the impact of dictionary size on the learned latents remains understudied.

Previous work has shown mixed findings regarding how SAEs scale with dictionary size. For instance, [7] observed that larger SAEs learn latents absent in smaller ones, such as specific chemical elements. Conversely, [2] found similar latents across various SAE sizes, noting that latents in smaller SAEs sometimes split into multiple latents as the dictionary size increases (Appendix A.1 includes such examples taken from our SAEs). This raises important questions about how latents evolve with dictionary size and how to effectively leverage larger SAEs for improved performance and interpretability.

[0*] These authors contributed equally to this work.

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

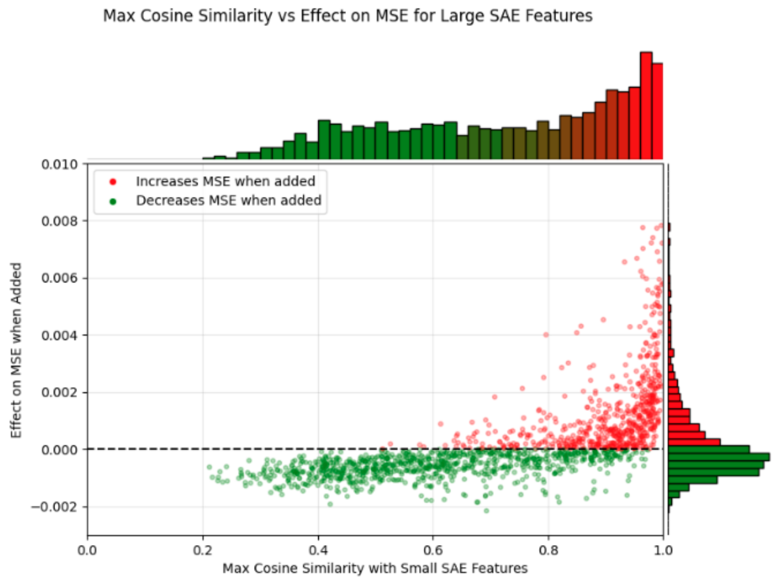

Figure 1: Change in MSE when adding each feature from GPT2-1536 to GPT2-768, plotted against the maximum cosine similarity of that feature to any feature in GPT2-768. Features with cosine similarity less than 0.7 tend to improve MSE, while more redundant features hurt performance. A few extreme outliers with very high cosine similarity and effect on MSE are not visible in this plot.

We extend this investigation to a range of SAE sizes trained on GPT-2-small and Pythia-410M (see Appendix A.2 for the full list of SAEs). In particular, we demonstrate it is possible to stitch SAEs of different sizes together by replacing latents in one with latents in another. This analysis provides evidence of two classes of latent in pairs of larger SAEs and smaller SAEs [1]:

1. **Novel latents** that capture information entirely absent in smaller SAEs. These latents can be freely introduced into smaller SAEs, often improving reconstruction performance without degradation.
2. **Reconstruction latents** that are either already present in smaller SAEs or are more precise versions of them. Introducing these latents into smaller SAEs without degrading performance requires the removal of corresponding latents from the smaller model.

We find that the maximum decoder cosine similarity between a latent and a target SAE effectively predicts whether a latent is novel or a reconstruction latent. Furthermore it is computationally cheap, as it does not require evaluating the SAE.

Building on these insights, we propose a method called SAE stitching, which allows for the interpolation between SAEs of different sizes and their reconstruction performance. By using the decoder cosine similarity to identify which latents to insert or swap, we can construct hybrid SAEs that benefit from the strengths of both smaller and larger models. This approach could enable domain specialization in small, generic SAEs by replacing general latents in smaller SAEs with specialized ones from larger SAEs.

Our contributions in this paper include:

- A characterization of how SAE latents evolve with dictionary size, providing insights into latents learning in these SAEs;
- The introduction of maximum decoder cosine similarity as an effective and cheap metric for identifying related latents across different SAE sizes;

[1]Throughout this paper we refer to SAEs in pairs of larger and smaller SAEs, and our results relate to these pairs, rather than a broader concept of what constitutes a small or large SAE. Furthermore, we refer to the learned elements of the SAE as latents, rather than as features, which is how we describe properties of the data.

- The development of SAE stitching, a method for interpolating between SAEs of different sizes to improve performance and enable domain specialization.

Our findings highlight the trade-offs between dictionary size, sparsity, and reconstruction performance in SAEs. By enhancing the understanding of latents learning and providing practical methods for combining dictionaries, we contribute to the science of sparse dictionary learning.

2 Method

We follow the setup from [2] to train SAEs that reconstruct the residual stream of LLMs. The encoding function is defined as $f_i(\mathbf{x}) = \mathrm{ReLU}(\mathbf{W}_i^{\text{enc}}\mathbf{x} + b_i^{\text{enc}})$, and the reconstruction is given by $\hat{\mathbf{x}} = \mathbf{b}^{\text{dec}} + \sum_{i=1}^{F} f_i(\mathbf{x})\mathbf{W}_i^{\text{dec}}$.

The encoder and decoder weights ($\mathbf{W}^{\text{enc}}$, $\mathbf{b}^{\text{enc}}$, $\mathbf{W}^{\text{dec}}$, $\mathbf{b}^{\text{dec}}$) are optimized to minimize the loss function:

$$\mathcal{L} = \mathbb{E}_{\mathbf{x}}\left[\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 + \lambda \sum_{i=1}^{F} f_i(\mathbf{x})\right],$$

which combines an L_2 reconstruction penalty and an L_1 activation penalty.

To study the impact of adding latents from one SAE to another, consider two base SAEs:

$$SAE_1(\mathbf{x}) = \mathbf{b}_1^{\text{dec}} + \sum_{i=1}^{F_1} f_{1,i}(\mathbf{x}), \quad \text{and} \quad SAE_2(\mathbf{x}) = \mathbf{b}_2^{\text{dec}} + \sum_{i=1}^{F_2} f_{2,i}(\mathbf{x}).$$

We construct a hybrid SAE by introducing a latent from one to the other—for example, adding latent 38 from SAE_1 to SAE_2:

$$SAE_2^{\star}(\mathbf{x}) = f_{1,38}(\mathbf{x}) + \mathbf{b}_2^{\text{dec}} + \sum_{i=1}^{F_2} f_{2,i}(\mathbf{x}).$$

More generally, we add latents from SAE_1 to SAE_2 and possibly remove latents from SAE_2. We define a latent as **novel** if its introduction reduces the reconstruction loss without needing to remove any latents from SAE_2. Conversely, a latent is a **reconstruction** latent if its addition increases the reconstruction loss unless certain latents in SAE_2 are removed.

Determining the latents that should be removed when introducing reconstruction latents requires testing all combinations of latents. This is computationally infeasible due to the exponential number of candidate groups. Therefore we propose using decoder cosine similarity to identify similar features, which is correlated with the change in reconstruction when adding a feature, as shown in Figure 1. We classify a latent as belonging to the reconstruction group if its maximum decoder cosine similarity is greater than 0.7, and otherwise to the novel group. We expand on how we chose this threshold in Appendix A.4.

3 Experiments

In our experiments we used SAEs trained on the residual stream of GPT2-Small [6] and Pythia-410M-deduped [1]. In this section we focus on the results from GPT2-Small, but have replicated with Pythia-410m. Full details of the SAE sizes are given in Appendix Table 1.

Appendix Figure 9 shows the impact of adding latents in random order from an SAE to a different SAE half its size, separately for the novel latent group (green) and the reconstruction group (red). We observe that across SAE sizes, the novel group generally leads to a decrease in the reconstruction error, whereas the reconstruction group generally leads to an increase in the reconstruction error. In particular we see a 10% decrease in the reconstruction MSE of GPT2-768 just from introducing the novel latents from GPT2-1536 with no fine-tuning required.

In order to insert reconstruction features from a larger SAE into a smaller SAE, we must swap them with their similar features. To find groups of latents to swap, we construct a bipartite graph where the latents in the smaller SAE form one vertex set, and the latents in the larger SAE form the other.

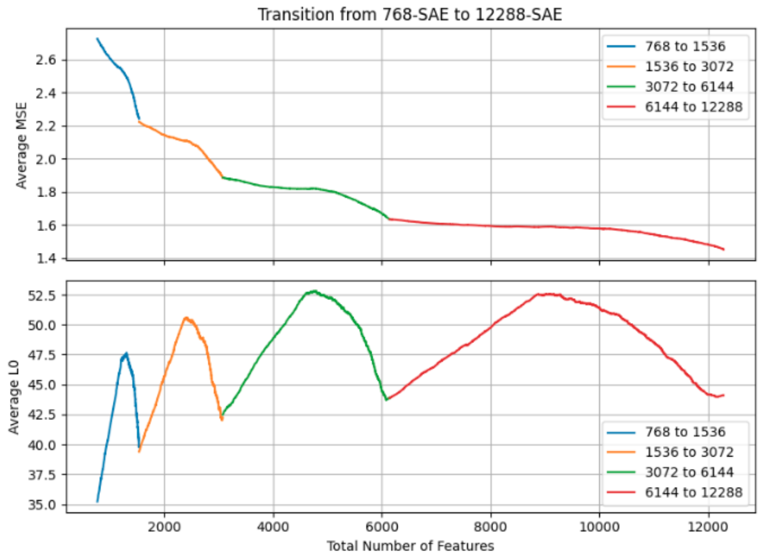

Figure 2: It is possible to smoothly interpolate between sparse autoencoders of different sizes by inserting or switching latents, where every insertion or switch results in a strict improvement in reconstruction (MSE). First, novel latents are added result in an increase in the L0; then the remaining latents are replaced with their similar latents in the larger SAE, leading to a decrease in the L0.

Latents in the two sets are connected if their decoder cosine similarity is greater than the threshold. Then, for every connected subgraph, we say that the subgraph latents from the smaller SAE and the subgraph latents from the larger SAE may be swapped (see Appendix A.7 for examples).

As shown in Appendix Figure 8, the effect of swapping the subgraph latents generally slightly worsens the reconstruction of the smaller SAE but also increases sparsity; this contrasts with novel latents, which improve reconstruction at the cost of L0.

Combining our methods for adding and swapping latents, we first add the novel latents to the smaller SAE and then swap in the remaining latents. The resulting reconstruction loss are shown in Figure 2, where we see this method interpolates reconstruction performance between SAEs of different sizes.

We also briefly explored whether stitching could be used to construct better performing SAEs at a smaller dictionary size, however this resulted in a trade-off between sparsity and dictionary size. This approach is described in Appendix A.6.

4 Conclusion

In this brief investigation into latents in SAEs of different sizes, we identified the existence of two classes of latent in larger SAEs in comparison to smaller ones: a group of novel latents that are entirely missing from the smaller SAE, and a reconstruction group of similar latents that sparsify latents in the smaller SAE. We demonstrate that decoder cosine similarity is a simple and effective similarity metric, that we used to interpolate between SAEs of different sizes. Furthermore, stitching could allow for the domain specialisation of small SAEs without needing to construct domain specific datasets.

Our results provide new insight into how the dictionary size hyperparameter effects the latents learned by SAEs, in particular the existence of the novel and reconstruction categories of latents. Whilst we focused on GPT-2 and Pythia-410m, we would encourage future work that validates these results on a more SAEs and base models, such as [4].

References

[1] Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR, 2023.

[2] Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, et al. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2, 2023.

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

[4] 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.

[5] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001, 2020.

[6] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[7] Adly Templeton, Tom Conerly, and et al. *Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet*. Anthropic, 2024.

A Appendix / supplemental material

A.1 Example latents

Figure 3 shows a histogram of the maximum decoder cosine similarity for each latent in GPT2-1536 over all latents in GPT2-768. On the right-hand-side, there is a cluster of latents with high cosine similarity.

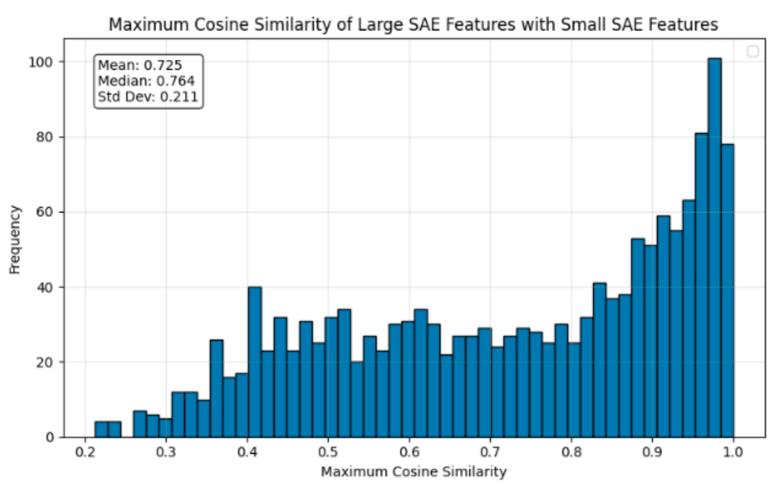

Figure 3: Distribution of maximum cosine similarities between decoder weights of latents in GPT2-1536 and GPT2-768. Many latents in the larger SAE have high similarity to latents in the smaller SAE, but there is also a long tail of novel latents.

Figure 4 shows an example of a latent from GPT-1536 and a latent from GPT-768 that have a cosine similarity of 0.99. We see that both of these latents activate strongly on the same inputs, and boost similar logits.

Figure 4: Examples latents with high cosine similarity (Redacted URL)

However, GPT2-1536 has a latent for "make sure" that has no counterpart in GPT-768. The nearest latents have a decoder cosine similarity of around 0.3, and are shown in

Figure 5: Example GPT2-1536 latent with no similar latent in GPT-768, with the three most similar latents shown (Redacted URL)

We evaluate the reconstruction performance of the two SAEs on inputs where this latent is active and inactive. The reconstruction performance of the smaller SAE is considerably worse on inputs where this larger SAE latent is active, compared to inputs where the latent is not active.

	Latent inactive	**Latent active**	**Difference**
GPT2-1536	2.225	2.518	0.293
GPT2-768	2.703	3.292	0.589

Averaging this metric across all 657 latents in GPT-1536 that have low maximum cosine similarity with all latents in GPT-768, we see a similar pattern (Figure 6)

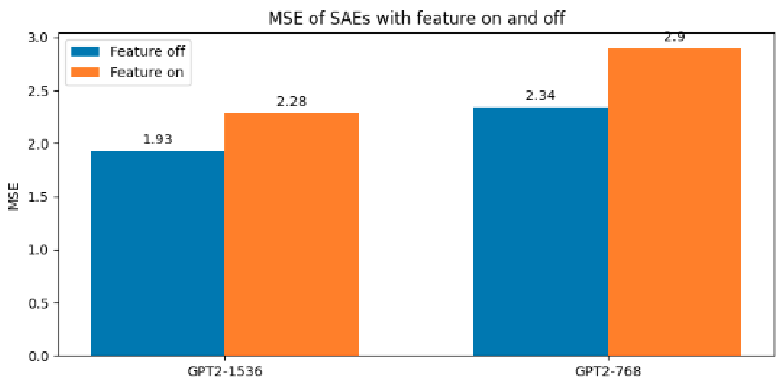

Figure 6: Reconstruction MSE of SAEs on inputs where novel latents in the larger SAE are active and inactive

A.2 Open source SAE weights

Table 1: The SAEs used in this study. All GPT2-small SAEs were trained on the layer 8 residual stream, and the Pythia-410m SAEs were trained on the layer 3 residual stream. CELR is the cross entropy loss recovered from either zero or mean ablation. The GPT2 SAEs are available on Neuronpedia at Redacted URL. We used the TransformerLens (`https://transformerlensorg.github.io/TransformerLens/`) implementations of GPT2 and Pythia.

Name	Model	Dict. size	L0	MSE	CELR Zero	CELR Mean
GPT2-768	gpt2-small	768	35.2	2.72	0.915	0.876
GPT2-1536	gpt2-small	1536	39.5	2.22	0.942	0.915
GPT2-3072	gpt2-small	3072	42.4	1.89	0.955	0.937
GPT2-6144	gpt2-small	6144	43.8	1.631	0.965	0.949
GPT2-12288	gpt2-small	12288	43.9	1.456	0.971	0.958
GPT2-24576	gpt2-small	24576	42.9	1.331	0.975	0.963
GPT2-49152	gpt2-small	49152	42.4	1.210	0.978	0.967
GPT2-98304	gpt2-small	98304	43.9	1.144	0.980	0.970
Pythia-8192	pythia-410m-deduped	8192	51.0	0.030	0.977	0.972
Pythia-16384	pythia-410m-deduped	16384	43.2	0.024	0.983	0.979

A.3 Comparison between latent similarity measures

[2] measure latent similarity via masked cosine similarity of activations, we suggest using the cosine similarity between latent decoder weights. We find that decoder weight cosine similarity is correlated with high latent similarity (Figure 7) and is more efficient to compute.

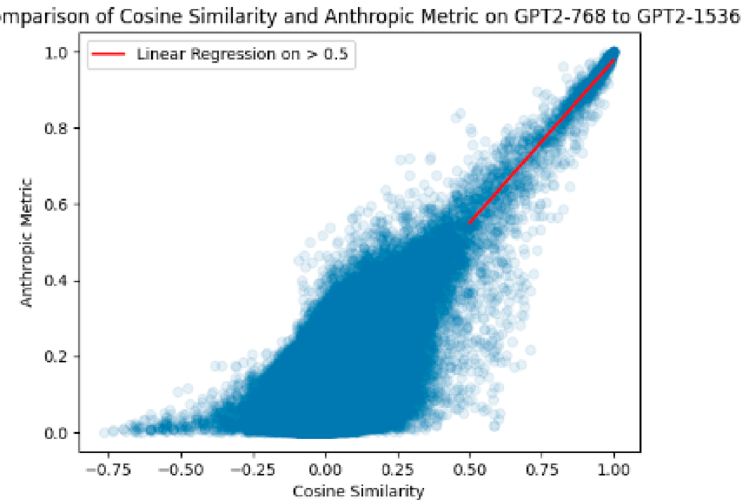

Figure 7: Comparison between decoder cosine similarity and masked activation similarity as used by [2]

A.4 Selecting a cosine similarity threshold

The cosine similarity threshold for related latents is manually set to provide a balance between labeling reconstruction latents as novel latents and vice versa, however values ± 0.1 give similar results. Figure 1, plots the maximum cosine similarity of each latent in GPT2-1536 with latents in GPT-768 against the change in reconstruction loss of GPT2-768 when adding that latent.

Based on the selected threshold value, we find that a small proportion of latents labeled as novel, which should decrease reconstruction MSE, result in an increase in the reconstruction MSE; and a larger proportion of latents labeled as reconstruction latents, which should increase reconstruction MSE, result in a decrease in MSE. These results are displayed in Table 2.

Table 2: Number of latents in GPT2-1536 grouped by whether they reduce or increase GPT-768 reconstruction, and whether their maximum cosine similarity is below the 0.7 threshold.

	# Novel latents	**# Reconstruction latents**
δ MSE <0	626	281
δ MSE >0	29	598

A.5 Swapping latents

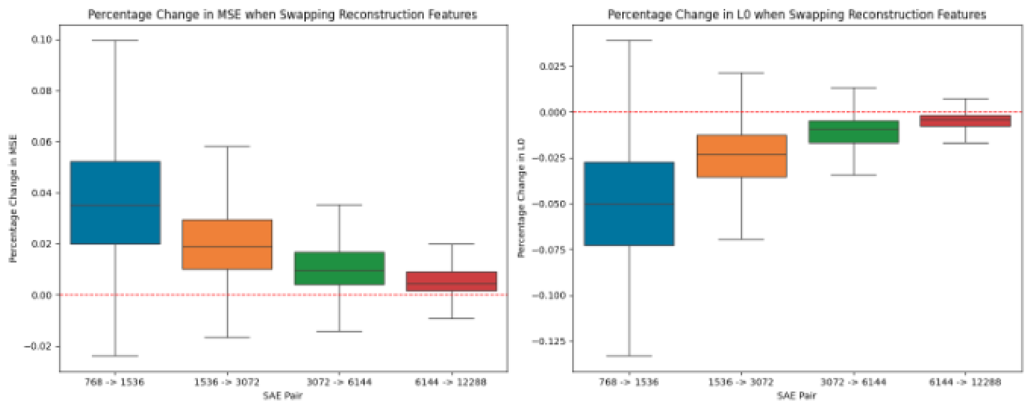

Figure 8: Effects on MSE and L0 when swapping reconstruction latents from larger SAEs to smaller ones. Swapping latent structures generally increases the MSE but almost always decreases L0. Outliers are not shown. The percentual effects per swap get smaller for larger models as the effects are distributed over more swaps.

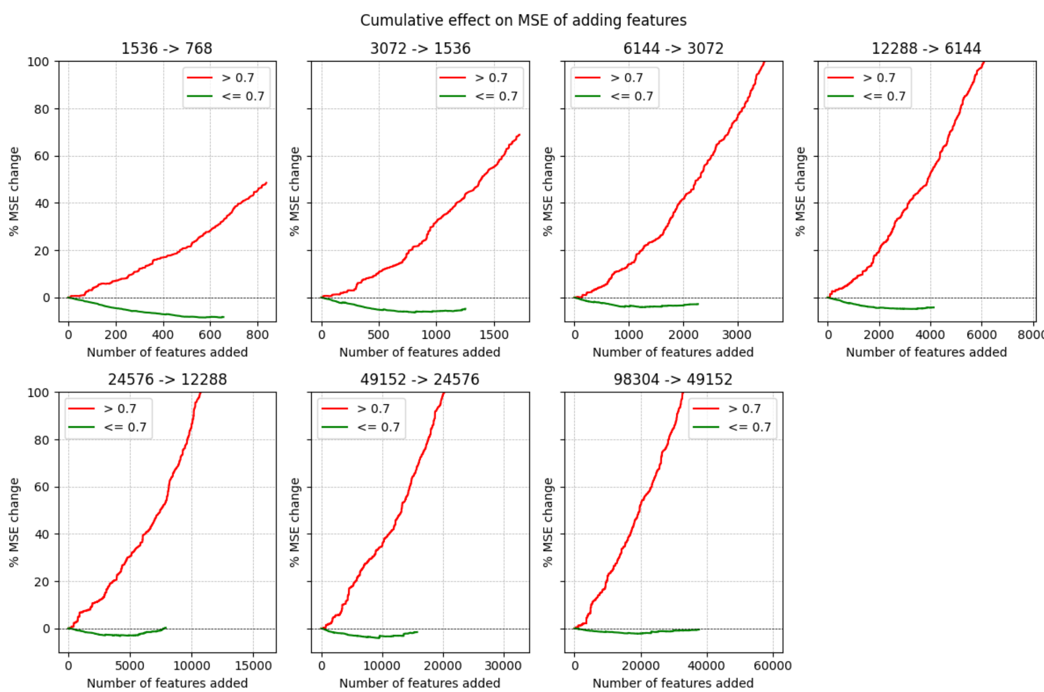

Figure 9: Percentage change of MSE of adding in latents from a larger SAE to a smaller SAE in a random order. Adding in all the latents with cosine <= 0.7 from GPT-1536 in GPT-768 reduces the MSE by almost 10%.

A.6 Frankenstein's SAEs

We briefly explored whether these methods can be used to construct better performing models at the same dictionary size as existing SAEs by choosing better latents to introduce to the dictionary. We

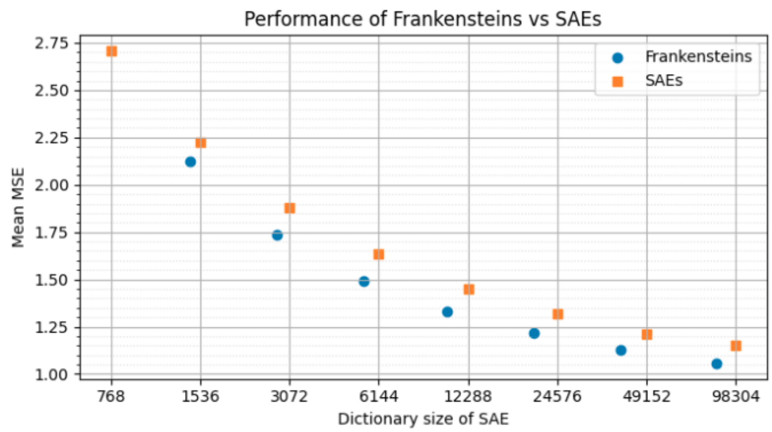

Figure 10: Reconstruction performance (MSE) of Stitched SAEs compared to the original SAEs of various sizes. The Stitched SAEs achieve lower MSE than comparably sized normal SAEs.

construct a Frankenstein's SAE from our base SAE model (GPT2-768) by iteratively adding latents from larger SAEs that have low cosine similarity with the stitched SAE latents, as described in 1.

Algorithm 1 Constructing a Frankenstein's SAE

Require: Base SAE model M_0 with n_0 latents
Require: Set of larger SAEs $M_1, M_2, ..., M_k$ with $n_1 < n_2 < ... < n_k$ latents
Require: Cosine similarity threshold $\theta = 0.7$
  $M_{enhanced} \leftarrow M_0$
  for $i \leftarrow 1$ to k **do**
    $F_{novel} \leftarrow \emptyset$
    for each latent $f \in M_i$ **do**
      if $\max_{g \in M_{enhanced}}(\text{CosineSimilarity}(f, g)) < \theta$ **then**
        $F_{novel} \leftarrow F_{novel} \cup f$
      end if
    end for
    $M_{enhanced} \leftarrow M_{enhanced} \cup F_{novel}$
  end for
  Retrain decoder weights of $M_{enhanced}$ for 100M tokens
  return $M_{enhanced}$

We find that these stitched SAEs have lower reconstruction MSE at a given dictionary size than the base SAEs, roughly achieving the same reconstruction performance as an SAE twice their size. However, they do this at a higher L0 than the base SAEs, making direct comparisons between SAEs and stitched SAEs difficult.

182 ## A.7 Latent families

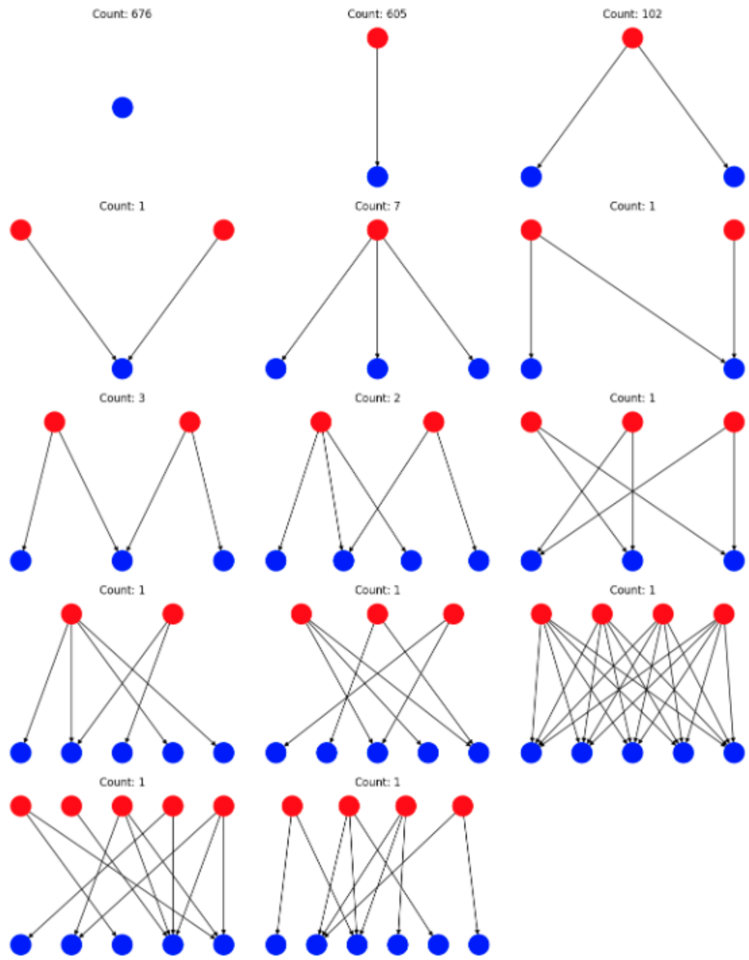

Figure 11: Connected subgraphs of the bipartite graph of latent in GPT2-768 and GPT2-1536.