where $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^m$ is the sparse latent representation and $\hat{\mathbf{x}}(\mathbf{f}) \in \mathbb{R}^n$ is the reconstructed input. $\mathbf{W}^{\text{enc}}$ is the encoder matrix with dimension $n \times m$ and $\mathbf{b}^{\text{enc}}$ is a vector of dimension m; conversely $\mathbf{W}^{\text{dec}}$ is the decoder matrix with dimension $m \times n$ and $\mathbf{b}^{\text{dec}}$ is of dimension n. The activation function σ enforces non-negativity and sparsity in $\mathbf{f}(\mathbf{x})$, and a latent i is active on a sample $\mathbf{x}$ if $f_i(\mathbf{x}) > 0$.

SAEs are trained on the activations of a language model at a particular site, such as the residual stream, on a large text corpus, using a loss function of the form

$$\mathcal{L}(\mathbf{x}) := \underbrace{\|\mathbf{x} - \hat{\mathbf{x}}(\mathbf{f}(\mathbf{x}))\|_2^2}_{\mathcal{L}_{\text{reconstruct}}} + \underbrace{\lambda \mathcal{S}(\mathbf{f}(\mathbf{x}))}_{\mathcal{L}_{\text{sparsity}}} + \alpha \mathcal{L}_{\text{aux}} \tag{3}$$

where $\mathcal{S}$ is a function of the latent coefficients that penalizes non-sparse decompositions, and λ is a sparsity coefficient, where higher values of λ encourage sparsity at the cost of higher reconstruction error. Some architectures also require the use of an auxiliary loss $\mathcal{L}_{\text{aux}}$, for example to recycle inactive latents in TopK SAEs (Gao et al., 2024). We expand on the different SAE variants in Appendix A.2 and provide a glossary of terms in Appendix A.1.

3 Related Work

SAEs for Mechanistic Interpretability. SAEs have been demonstrated to recover sparse, monosemantic, and interpretable features from language model activations (Bricken et al., 2023; Cunningham et al., 2023; Templeton, 2024; Gao et al., 2024; Rajamanoharan et al., 2024a;b), however their application to mechanistic interpretability is nascent. After training, researchers often interpret the meaning of SAE latents by examining the dataset examples on which they are active, either through manual inspection using features dashboards (Bricken et al., 2023) or automated interpretability techniques (Gao et al., 2024). SAEs have been used for circuit analysis (Marks et al., 2024) in the vein of (Olah et al., 2020; Olsson et al., 2022); to study the role of attention heads in GPT-2 (Kissane et al., 2024); and to replicate the identification of a circuit for indirect object identification in GPT-2 (Makelov et al., 2024). Transcoders, a variant of SAEs, have been used to simplify circuit analysis and applied to the greater-than circuit in GPT-2 (Dunefsky et al., 2024). While these applications highlight SAEs as valuable tools for understanding language models, it remains unclear whether they identify canonical units of analysis.

Representational Structure. Language models trained on the next-token prediction task learn representations that model the generative process of the data. For example, Li et al. (2022) found that transformers trained by next-move prediction to play the board game Othello explicitly represent the board state; and Gurnee & Tegmark (2023) used linear probes to predict geographical and temporal information from language model activations. SAEs learn these structures as well, for example, the activations of a cluster of GPT-2 SAE latents form a cycle when reconstruction activations for weekday name tokens (Engels et al., 2024). Bricken et al. (2023) find evidence of convergent global structure by applying a 2-dimensional UMAP transformation to decoder directions of SAEs of different sizes. This results in a rich structure of latent projections with regions of latents relating to similar concepts close to each other.

Tuning Dictionary Size. Previous work on tuning dictionary size has mixed findings regarding how SAEs scale with dictionary size. For instance, Templeton (2024) observed that larger SAEs learn latents absent in smaller ones, such as specific chemical elements. Conversely, Bricken et al. (2023) 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.3 includes such examples taken from our SAEs). Currently, the effect of dictionary size on the learned latents has not been systematically studied.

Model Stitching. Model stitching is a method by which layers from one neural network are "stitched" or swapped into another neural network. Lenc & Vedaldi (2015); Bansal et al. (2021) propose that model stitching can be used to quantify representation similarity across different neural network architectures and training regimes by demonstrating that if two networks trained on the same task can be stitched together with minimal loss in performance, it suggests a high degree of alignment in their intermediate representations.

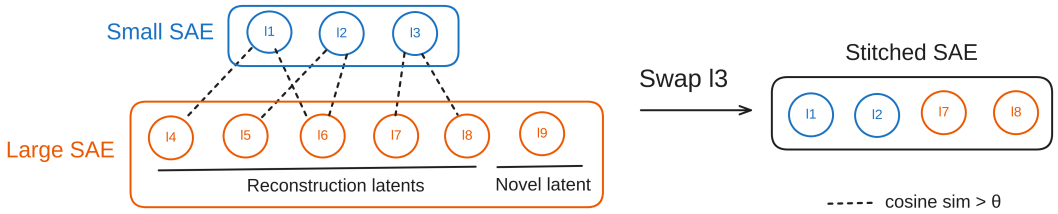

Figure 3: SAE stitching operation: connected subgraphs of latents can be swapped between SAEs based on cosine similarity

4 SAE STITCHING

Templeton (2024) finds that a larger SAE has latents that activate on certain specific individual chemical elements that a smaller SAE did not represent. Conversely, Bricken et al. (2023) observes that smaller SAEs learn latents activating on broad mathematical text, while larger SAEs learn latents activating on more specific mathematical categories. On the one hand, this suggests that training large SAEs is necessary to learn latents relating to all relevant concepts. On the other hand, this means that some of the capacity of larger SAEs is used to represent similar information as smaller SAEs, but with more latents.

To compare latents of SAEs with different dictionary sizes, we introduce SAE stitching. In SAE stitching, we add or replace latents in one SAE with latents from another and observe the effect on the reconstruction performance. We find that larger SAEs learn both finer-grained versions of latents from smaller SAEs (*reconstruction latents*) and entirely new latents that capture additional information (*novel latents*). SAE stitching allows us to identify these two categories of latents in larger SAEs.

4.1 STITCHING OPERATION

The output of an SAE can be expressed as a sum of the contributions from the individual latents:

$$\hat{\mathbf{x}} := \sum_{i=0}^{d} \mathbf{W}_i^{\text{dec}} f_i(\mathbf{x}) + \mathbf{b}^{\text{dec}} \tag{4}$$

where d is the size of the SAE dictionary, $\mathbf{W}_i^{\text{dec}}$ is an individual decoder direction, $f_i(\mathbf{x})$ is the activation value for the latent i, and $\mathbf{b}^{\text{dec}}$ is the decoder bias term. To stitch latents from one SAE into another, we modify the reconstruction as follows:

$$\hat{\mathbf{x}} := \alpha \mathbf{b}_0^{\text{dec}} + (1-\alpha)\mathbf{b}_1^{\text{dec}} + \sum_{l_0 \in L_0} \mathbf{W}_{0,l_0}^{\text{dec}} f_{0,l_0}(\mathbf{x}) + \sum_{l_1 \in L_1} \mathbf{W}_{1,l_1}^{\text{dec}} f_{1,l_1}(\mathbf{x}) \tag{5}$$

where $\alpha = \frac{|L_0|}{|L_0|+|L_1|}$, $\mathbf{W}_{0,l_0}^{\text{dec}}$ and $\mathbf{W}_{1,l_1}^{\text{dec}}$ represent individual decoder directions and L_0 and L_1 are the set of latents we include from the respective SAEs. Unlike with model stitching (Bansal et al., 2021), SAE decoder directions are privileged and require no transformations to stitch.

We experiment with stitching on eight SAEs trained on the residual stream of layer 8 of GPT 2 Small (Radford et al., 2019) with dictionary sizes ranging from 768 to 98,304 and two of the Gemma Scope SAEs (Lieberum et al., 2024) trained on Gemma 2 2B (Team et al., 2024) with dictionary size 16,384 and 32,768. For the full list of SAE properties and training details see Appendix A.5.

4.2 NOVEL AND RECONSTRUCTION LATENTS

Using SAE stitching, we want to find out whether latents in larger SAEs are just more fine-grained versions of latents of smaller SAEs, or whether they represent novel information that is missed by the smaller SAEs. If we stitch a latent from a larger SAE into a smaller SAE and the reconstruction

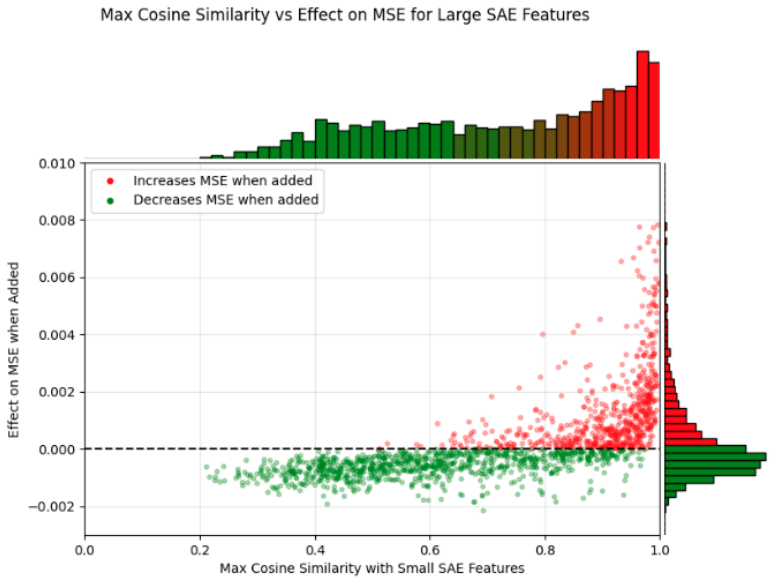

Figure 4: Change in MSE when adding each feature from GPT2-1536 to GPT2-768, plotted against the maximum cosine similarity of that feature to any feature in GPT2-768. Features with cosine similarity less than 0.7 tend to improve MSE, while more redundant features hurt performance. A few extreme outliers with very high cosine similarity and effect on MSE are not visible in this plot.

performance improves, this indicates that the latent is adding new information to the smaller SAE. If the reconstruction performance worsens instead, this indicates that the latent is overlapping and the stitched SAE is now representing the same information twice. In this case, we can instead swap the latent from the smaller SAE with its corresponding latents from the larger SAE.

However, evaluating all combinations of latents when stitching two SAEs is computationally infeasible. Bricken et al. (2023) finds that the decoder directions of latents with similar behavior across SAEs of different sizes form local clusters with high cosine similarity. As such, we propose thresholding on the maximum decoder cosine similarity metric as a heuristic to classify latents into the two categories, i.e. for an SAE decoder direction $\mathbf{W}^{\text{dec}}_{1,i}$, the feature is assigned to the novel group if

$$\max_j \left(\frac{\mathbf{W}^{\text{dec}}_{1,i} \cdot \mathbf{W}^{\text{dec}}_{0,j}}{\|\mathbf{W}^{\text{dec}}_{1,i}\| \|\mathbf{W}^{\text{dec}}_{0,j}\|} \right) < \theta \tag{6}$$

where $\mathbf{W}^{\text{dec}}_{0,j}$ represents any decoder direction in $\mathbf{W}^{\text{dec}}_{0}$. If the maximum cosine similarity is larger than the threshold, it is assigned to the reconstruction group. Figure 4 shows the relationship between cosine similarity and the effect on the reconstruction loss of independently stitching that feature for a pair of GPT-2 SAEs. In our experiments we set the cosine threshold to 0.7 for the GPT-2 SAEs, and 0.4 for the Gemmascope SAEs, based on our exploratory analysis. Appendix Figure 13 shows the ROC curve for using different thresholds to predict the effect of adding a latent based on the maximum cosine similarity.

Inserting reconstruction latents into the target SAE decreases the performance due to overlapping functionality. We resolve this by removing latents in the target SAE that have a high cosine similarity to the ones we are inserting, effectively swapping them with similar latents. To select which latents we swap, we construct a bipartite graph where latents from the smaller SAE form one set of vertices and latents from the larger SAE form the other. We connect latents if they have a decoder cosine similarity above the threshold. For each connected subgraph, the latents from the smaller SAE can be swapped by their corresponding connected latents in the larger SAE (see Appendix A.4 for examples of connected subgraphs).