Improving Sparse Decomposition of Language Model Activations with Gated Sparse Autoencoders

Senthooran Rajamanoharan [1 *] **Arthur Conmy** [1 *] **Lewis Smith** [1] **Tom Lieberum** [1 †] **Vikrant Varma** [1 †]
János Kramár [1] **Rohin Shah** [1] **Neel Nanda** [1]

Abstract

Recent work has found that sparse autoencoders (SAEs) are an effective technique for unsupervised discovery of interpretable features in language models' (LMs) activations, by finding sparse, linear reconstructions of those activations. We introduce the Gated Sparse Autoencoder (Gated SAE), which achieves a Pareto improvement over training with prevailing methods. In SAEs, the L1 penalty used to encourage sparsity introduces many undesirable biases, such as *shrinkage* – systematic underestimation of feature activations. The key insight of Gated SAEs is to separate the functionality of (a) determining which directions to use and (b) estimating the magnitudes of those directions: this enables us to apply the L1 penalty only to the former, limiting the scope of undesirable side effects. Through training SAEs on LMs of up to 7B parameters we find that, in typical hyper-parameter ranges, Gated SAEs solve shrinkage, are similarly interpretable, and require half as many firing features to achieve comparable reconstruction fidelity.

1 Introduction

Mechanistic interpretability aims to explain how neural networks produce outputs in terms of the learned algorithms executed during a forward pass (Olah, 2022; Olah et al., 2020). Much work makes use of the fact that many concept representations appear to be linear (Elhage et al., 2021; Gurnee et al., 2023; Olah et al., 2020; Park et al., 2023), i.e. that they correspond to interpretable directions in activation space. However, finding the set of all interpretable directions is a highly non-trivial problem. Classic approaches, like interpreting neurons (i.e. directions in the standard basis) are insufficient, as many are polysemantic and tend to activate for a range of different seemingly unrelated concepts (Bolukbasi et al., 2021; Elhage et al., 2022a;b). Within the field, there has recently been much interest (Bricken et al., 2023; Cunningham et al., 2023; Kissane et al., 2024a;b; Bloom, 2024) in using sparse autoencoders (SAEs; (Ng, 2011)) as an unsupervised method for finding causally relevant, and ideally interpretable, directions in a language model's activations.

Although SAEs show promise in this regard (Marks et al., 2024; Nanda et al., 2024), the L1 penalty used in the prevailing training method to encourage sparsity also introduces biases that harm the accuracy of SAE reconstructions, as the loss can be decreased by trading-off some reconstruction accuracy for lower L1. In this paper, we introduce a modification to the baseline SAE architecture – a *Gated SAE* – along with an accompanying loss function, which partially overcomes these limitations. Our key insight is to use separate affine transformations for (a) determining which dictionary elements to use in a reconstruction and (b) estimating the coefficients of active elements, and to apply the sparsity penalty only to the former task. We share a subset of weights between these transformations to avoid significantly increasing the parameter count and inference-time compute requirements of a Gated SAE compared to a baseline SAE of equivalent width.[1]

We evaluate Gated SAEs on multiple models: a one layer GELU activation language model (Nanda, 2022), Pythia-2.8B (Biderman et al., 2023) and Gemma-7B (Gemma Team et al., 2024), and on multiple sites within models: MLP layer outputs, attention layer outputs, and residual stream activations. Across these models and sites, we find Gated SAEs to be a Pareto improvement over baseline SAEs holding training compute fixed (Fig. 1): they yield sparser decompositions at any desired level of reconstruction fidelity. We also conduct further follow up ablations and investigations on a subset of these models and sites to better understand the differences between Gated SAEs and baseline SAEs.

Overall, the key contributions of this work are that we:

[1]Google DeepMind. Correspondence to: Senthooran Rajamanoharan <srajamanoharan@google.com>, Neel Nanda <neelnanda@google.com>.

41st International Conference on Machine Learning Mechanistic Interpretability Workshop, Vienna, Austria. PMLR 235, 2024.
Copyright 2024 by the author(s).

[1]Although due to an auxiliary loss term, computing the Gated SAE loss for training purposes does require 50% more compute than computing the loss for a matched-width baseline SAE.
*: Equal contribution. †: Core infrastructure contributor.

1. Introduce the Gated SAE, a modification to the standard SAE architecture that decouples detection of which features are present from estimating their magnitudes (Section 3.2);
2. Show that Gated SAEs Pareto improve the sparsity and reconstruction fidelity trade-off compared to baseline SAEs (Section 4.1);
3. Confirm that Gated SAEs overcome shrinkage while outperforming other methods that also address this problem (Section 5.2);
4. Provide evidence from a small double-blind study that Gated SAE features are comparably interpretable to baseline SAE features (Section 4.2).

2 Preliminaries

In this section we summarise the concepts and notation necessary to understand existing SAE architectures and training methods following Bricken et al. (2023), which we call the *baseline SAE*. We define Gated SAEs in Section 3.2.

As motivated in Section 1, we wish to decompose a model's activation $\mathbf{x} \in \mathbb{R}^n$ into a sparse, linear combination of feature directions:

$$\mathbf{x} \approx \mathbf{x}_0 + \sum_{i=1}^{M} f_i(\mathbf{x})\mathbf{d}_i, \tag{1}$$

where $\mathbf{d}_i$ are dictionary of $M \gg n$ latent unit-norm *feature directions*, and the sparse coefficients $f_i(\mathbf{x}) \geq 0$ are the corresponding *feature activations* for $\mathbf{x}$.[2] The right-hand side of Eq. (1) naturally has the structure of an autoencoder: an input activation $\mathbf{x}$ is encoded into a (sparse) feature activations vector $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^M$, which in turn is linearly decoded to reconstruct $\mathbf{x}$.

Baseline architecture Using this correspondence, Bricken et al. (2023) and subsequent works attempt to learn a suitable sparse decomposition by parameterizing a single-layer autoencoder $(\mathbf{f}, \hat{\mathbf{x}})$ defined by:

$$\mathbf{f}(\mathbf{x}) := \text{ReLU}\left(\mathbf{W}_{\text{enc}}\left(\mathbf{x} - \mathbf{b}_{\text{dec}}\right) + \mathbf{b}_{\text{enc}}\right) \tag{2}$$

$$\hat{\mathbf{x}}(\mathbf{f}) := \mathbf{W}_{\text{dec}}\mathbf{f} + \mathbf{b}_{\text{dec}} \tag{3}$$

and training it using gradient descent to reconstruct samples $\mathbf{x} \sim \mathcal{D}$ from a large dataset $\mathcal{D}$ of activations collected from a single site and layer of a trained language model, constraining the hidden representation $\mathbf{f}$ to be sparse. Once the sparse autoencoder has been trained, we obtain a decomposition of the form of Eq. (1) by identifying the (suitably normalised) columns of the decoder weight matrix $\mathbf{W}_{\text{dec}} \in \mathbb{R}^{M \times n}$ with the dictionary of feature directions $\mathbf{d_i}$, the decoder bias $\mathbf{b}_{\text{dec}} \in \mathbb{R}^n$ with the centering term $\mathbf{x}_0$, and the (suitably normalised) entries of the latent representation $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^M$ with the feature activations $f_i(\mathbf{x})$.

Baseline training methodology To train sparse autoencoders, Bricken et al. (2023) use a loss function with two terms that respectively encourage faithful reconstruction and sparsity:[3]

$$\mathcal{L}(\mathbf{x}) := \|\mathbf{x} - \hat{\mathbf{x}}(\mathbf{f}(\mathbf{x}))\|_2^2 + \lambda \left\|\mathbf{f}(\mathbf{x})\right\|_1. \tag{4}$$

Since it is possible to arbitrarily reduce the L1 sparsity loss term without affecting reconstructions or sparsity by simply scaling down encoder outputs and scaling up the norm of the decoder weights, it is important to constrain the norms of the columns of $\mathbf{W}_{\text{dec}}$ during training. Following Bricken et al. (2023), we constrain norms to one. See Appendix F for further details on SAE training.

Evaluating SAEs Two metrics are primarily used to get a sense of SAE quality (Bricken et al., 2023): *L0*, a measure of SAE sparsity, and *loss recovered*, a measure of SAE reconstruction fidelity. L0 measures the average number of features used by a SAE to reconstruct input activations. Loss recovered is a normalised measure of the increase induced in a LM's cross entropy loss when we replace its original activations with the corresponding SAE reconstructions during the model's forward pass. Both these metrics are formally defined in Appendix A. Since it is possible for SAEs to score well on these metrics and still fail to be useful for interpretability-related tasks (Templeton et al., 2024), we perform manual analysis of SAE interpretability in Section 4.2.

3 Gated SAEs

3.1 Motivation

The intention behind how SAEs are trained is to maximise reconstruction fidelity at a given level of sparsity, as measured by L0, although in practice we optimize a mixture of reconstruction fidelity and L1 regularization. This difference is a source of unwanted bias in the training of a sparse autoencoder: for any fixed level of sparsity, a trained SAE can achieve lower loss (as defined in Eq. (4)) by trading off a little reconstruction fidelity to perform better on the L1 sparsity penalty.

The clearest consequence of this bias is *shrinkage* (Wright and Sharkey, 2024). Holding the decoder $\hat{\mathbf{x}}(\bullet)$ fixed, the L1 penalty pushes feature activations $\mathbf{f}(\mathbf{x})$ towards zero, while

[2] In this work, we use the term *feature* to refer only to the *learned features* of SAEs, i.e. the overcomplete basis directions that are linearly combined to produce reconstructions. In particular, *learned features* are always linear and not necessarily interpretable.

[3] Note that we cannot directly optimize the L0 norm (i.e. the number of active features) since this is not a differentiable function. We do however use the L0 norm to evaluate SAE sparsity.

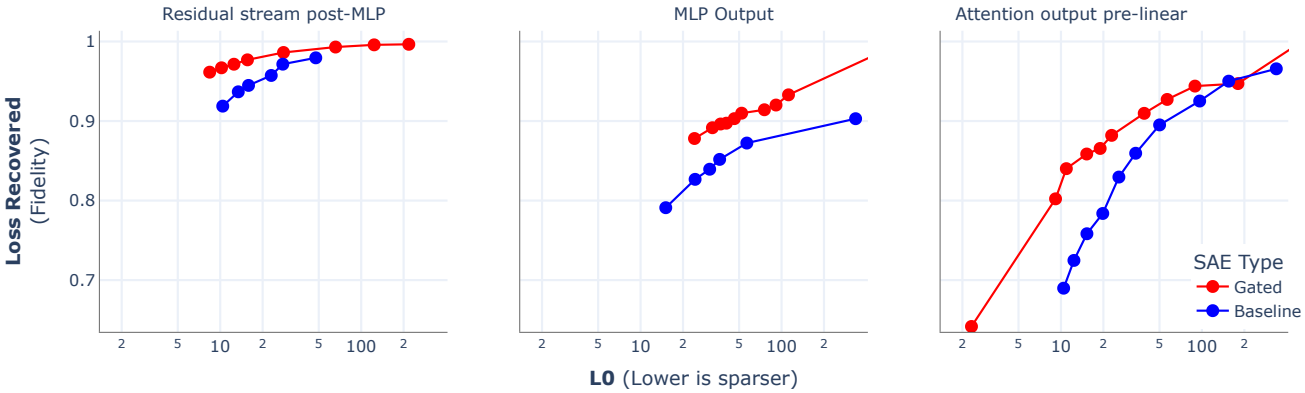


Figure 1: Gated SAEs consistently offer improved reconstruction fidelity for a given level of sparsity compared to prevailing (baseline) approaches. These plots compare Gated SAEs to baseline SAEs at Layer 20 in Gemma-7B. Gated SAEs' dictionaries are of size $2^{17} \approx 131$k whereas baseline dictionaries are 50% larger, so that both types are trained with equal compute. This performance improvement holds in layers throughout GELU-1L, Pythia-2.8B and Gemma-7B (see Appendix D).

the reconstruction loss pushes $\mathbf{f}(\mathbf{x})$ high enough to produce an accurate reconstruction. Thus, the optimal value falls somewhere in between, and as a result the SAE systematically underestimates the magnitude of feature activations, without necessarily providing any compensatory benefit for sparsity.[4]

How can we reduce the bias introduced by the L1 penalty? The output of the encoder $\mathbf{f}(\mathbf{x})$ of a baseline SAE (Section 2) has two roles:

1. It *detects* which features are active (according to whether the outputs are zero or strictly positive). For this role, the L1 penalty is necessary to ensure the decomposition is sparse.
2. It *estimates* the magnitudes of active features. For this role, the L1 penalty is a source of unwanted bias.

If we could separate out these two functions of the SAE encoder, we could design a training loss that narrows down the scope of SAE parameters that are affected (and therefore to some extent biased) by the L1 sparsity penalty to precisely those parameters that are involved in feature detection, minimising its impact on parameters used in feature magnitude estimation.

3.2 Gated SAEs

Architecture How should we modify the baseline SAE encoder to achieve this separation of concerns? Our solution is to replace the single-layer ReLU encoder of a baseline SAE with a *gated* ReLU encoder. Taking inspiration from Gated Linear Units (Shazeer, 2020; Dauphin et al., 2017), we define the gated encoder as

$$\tilde{\mathbf{f}}(\mathbf{x}) := \underbrace{\mathbb{1}[\overbrace{(\mathbf{W}_{\text{gate}}(\mathbf{x} - \mathbf{b}_{\text{dec}}) + \mathbf{b}_{\text{gate}})}^{\boldsymbol{\pi}_{\text{gate}}(\mathbf{x})} > \mathbf{0}]}_{\mathbf{f}_{\text{gate}}(\mathbf{x})} \odot \underbrace{\text{ReLU}(\mathbf{W}_{\text{mag}}(\mathbf{x} - \mathbf{b}_{\text{dec}}) + \mathbf{b}_{\text{mag}})}_{\mathbf{f}_{\text{mag}}(\mathbf{x})}, \tag{5}$$

where $\mathbb{1}[\bullet > \mathbf{0}]$ is the (pointwise) Heaviside step function and $\odot$ denotes elementwise multiplication. Here, $\mathbf{f}_{\text{gate}}$ determines which features are deemed to be active, while $\mathbf{f}_{\text{mag}}$ estimates feature activation magnitudes (which only matter for features that have been deemed to be active); $\boldsymbol{\pi}_{\text{gate}}(\mathbf{x})$ are the $\mathbf{f}_{\text{gate}}$ sub-layer's pre-activations, which are used in the gated SAE loss, defined below.

Naively, we appear to have doubled the number of parameters in the encoder, increasing the total number of parameters by 50%. We mitigate this through weight sharing: we parameterize these layers so that the two layers share the same projection directions, but allow the norms of these directions as well as the layer biases to differ. Concretely, we define $\mathbf{W}_{\text{mag}}$ in terms of $\mathbf{W}_{\text{gate}}$ and an additional vector-valued rescaling parameter $\mathbf{r}_{\text{mag}} \in \mathbb{R}^M$ as follows:

$$(\mathbf{W}_{\text{mag}})_{ij} := (\exp(\mathbf{r}_{\text{mag}}))_i \cdot (\mathbf{W}_{\text{gate}})_{ij}\,. \tag{6}$$

See Fig. 2 for an illustration of the tied-weight Gated SAEs architecture. With this weight tying scheme, the Gated SAE has only $2 \times M$ more parameters than a baseline SAE. In Section 5.1, we show that this weight tying scheme does not harm performance.

[4] Conversely, rescaling the shrunk feature activations (Wright and Sharkey, 2024) is not necessarily enough to overcome the bias induced by by L1 penalty: a SAE trained with the L1 penalty could have learnt sub-optimal encoder and decoder directions that are not improved by such a fix. In Section 5.2 and Fig. 7 we provide empirical evidence that this is true in practice.

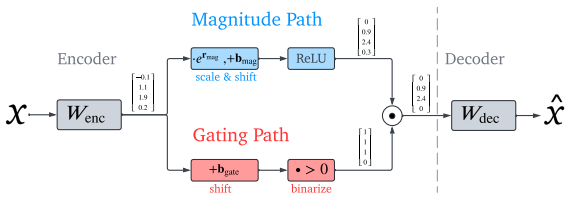


Figure 2: The Gated SAE architecture with weight sharing between the gating and magnitude paths, shown with an example input.

With tied weights, the gated encoder can be reinterpreted as a single-layer linear encoder with a non-standard and discontinuous "Jump ReLU" activation function (Erichson et al., 2019), $\sigma_\theta(z)$, illustrated in Fig. 12. To be precise, using the weight tying scheme of Eq. (6), $\tilde{\mathbf{f}}(\mathbf{x})$ can be re-expressed as $\tilde{\mathbf{f}}(\mathbf{x}) = \sigma_{\boldsymbol{\theta}}(\mathbf{W}_{\text{mag}} \cdot \mathbf{x} + \mathbf{b}_{\text{mag}})$, with the Jump ReLU gap given by $\boldsymbol{\theta} = \mathbf{b}_{\text{mag}} - e^{\mathbf{r}_{\text{mag}}} \odot \mathbf{b}_{\text{gate}}$; see Appendix G for an explanation. We think this is a useful intuition for reasoning about how Gated SAEs reconstruct activations in practice.

Training A naive guess at a loss function for training Gated SAEs would be to replace the sparsity penalty in Eq. (4) with the L1 norm of $\mathbf{f}_{\text{gate}}(\mathbf{x})$. Unfortunately, due to the Heaviside step activation function in $\mathbf{f}_{\text{gate}}$, no gradients would propagate to $\mathbf{W}_{\text{gate}}$ and $\mathbf{b}_{\text{gate}}$. To mitigate this, we instead apply the L1 norm to the positive parts of the preactivation, $\text{ReLU}\left(\boldsymbol{\pi}_{\text{gate}}(\mathbf{x})\right)$. To ensure $\mathbf{f}_{\text{gate}}$ aids reconstruction by detecting active features, we add an auxiliary task requiring that these same rectified preactivations can be used by the decoder to produce a good reconstruction:

$$\mathcal{L}_{\text{gated}}(\mathbf{x}) := \underbrace{\left\|\mathbf{x} - \hat{\mathbf{x}}\left(\tilde{\mathbf{f}}(\mathbf{x})\right)\right\|_2^2}_{\mathcal{L}_{\text{reconstruct}}} + \underbrace{\lambda \left\|\text{ReLU}(\boldsymbol{\pi}_{\text{gate}}(\mathbf{x}))\right\|_1}_{\mathcal{L}_{\text{sparsity}}} + \underbrace{\left\|\mathbf{x} - \hat{\mathbf{x}}_{\text{frozen}}\left(\text{ReLU}\left(\boldsymbol{\pi}_{\text{gate}}(\mathbf{x})\right)\right)\right\|_2^2}_{\mathcal{L}_{\text{aux}}} \tag{7}$$

where $\hat{\mathbf{x}}_{\text{frozen}}$ is a frozen copy of the decoder, $\hat{\mathbf{x}}_{\text{frozen}}(\mathbf{f}) := \mathbf{W}_{\text{dec}}^{\text{copy}}\mathbf{f} + \mathbf{b}_{\text{dec}}^{\text{copy}}$, to ensure that gradients from $\mathcal{L}_{\text{aux}}$ do not propagate back to $\mathbf{W}_{\text{dec}}$ or $\mathbf{b}_{\text{dec}}$. This can be implemented by stop gradient operations rather than creating copies. See Appendix I for pseudo-code for the forward pass and loss function.

To calculate this loss (or its gradient), we have to run the decoder twice: once to perform the main reconstruction for $\mathcal{L}_{\text{reconstruct}}$ and once to perform the auxiliary reconstruction for $\mathcal{L}_{\text{aux}}$. This leads to a 50% increase in the compute required to perform a training update step. However, the increase in overall training time is typically much less, as in our experience much of the training wall clock time goes to generating language model activations (if these are being generated on the fly) or disk I/O (if training on saved activations).

4 Evaluating Gated SAEs

In this section we benchmark Gated SAEs against baseline SAEs across a large variety of models and at different sites. We show that they produce more faithful reconstructions at equal sparsity and that they resolve shrinkage. Through a double-blind manual interpretability study, we find that Gated SAEs produce features that are similarly interpretable to baseline SAE features.

4.1 Benchmarking Gated SAEs

Methodology We trained a suite of Gated and baseline SAEs, a family of each type to reconstruct each of the following activations:

1. The MLP neuron activations in GELU-1L, which is the closest direct comparison to Bricken et al. (2023);
2. The MLP outputs, attention layer outputs (taken pre-W_O (Kissane et al., 2024a)) and residual stream activations in 5 different layers throughout Pythia-2.8B and four different layers in the Gemma-7B base model.

For each model and reconstruction site, we trained multiple SAEs using different values of λ (and therefore L0), allowing us to compare the Pareto frontiers of L0 and loss recovered between Gated and baseline SAEs. We also use the *relative reconstruction bias* metric, γ, defined in Appendix B to measure shrinkage in our trained SAEs. This metric measures the relative bias in the norm of an SAE's reconstructions; unbiased SAEs obtain $\gamma = 1$, whereas SAEs affected by shrinkage (which causes reconstruction norms to be systematically too small) have $\gamma < 1$.

Since Gated SAEs require at most 1.5× more compute to train than regular SAEs (Section 3.2) of the same width, we compare Gated SAEs to baseline SAEs that have a 50% larger dictionary (hidden dimension M) to ensure fair comparison in our evaluations.[5]

Results We plot sparsity against reconstruction fidelity for SAEs with different values of λ. Higher λ corresponds to increased sparsity and worse reconstruction, so as in Bricken et al. (2023) we observe a Pareto frontier of possible trade-offs. We plot Pareto curves for GELU-1L in Fig. 3a and Pythia-2.8B and Gemma-7B in Appendix D. At

[5]Since wider SAEs provide better reconstructions (all else being equal), the gap between Gated SAEs' and baseline SAEs' performance is even wider when we use baseline SAEs with equal width in the comparison. This can be seen in the difference between the "1.5× width" and "equal width" baseline curves in Fig. 5.

all sites tested, Gated SAEs are a Pareto improvement over regular SAEs: they provide better reconstruction fidelity at any fixed level of sparsity.[6] For some sites in Pythia-2.8B and Gemma-7B, loss recovered does not monotonically increase with L0; we attribute this to difficulties training SAEs (Appendix F.1.3).

As shown in Fig. 3b, Gated SAEs' reconstructions are unbiased, with $\gamma \approx 1$, whereas baseline SAEs exhibit shrinkage ($\gamma < 1$), with the impact of shrinkage getting worse as the L1 coefficient λ increases (and L0 consequently decreases). Fig. 10 shows that this result generalizes to Pythia-2.8B.

4.2 Interpretability

Although Gated SAEs provide more faithful reconstructions than baselines at equal sparsity, it does not necessarily follow that these reconstructions are better suited to downstream interpretability-related tasks. Currently, there is no consensus on how to systematically assess the degree to which a SAE's features are useful for downstream tasks, but a plausible proxy is to assess the extent to which these features are human interpretable (Bricken et al., 2023). Therefore, to gain a more qualitative understanding of the differences between their learned features, we conduct a blinded human study in which we rate and compare the interpretability of randomly sampled Gated and baseline SAE features.

Methodology We study a variety of SAEs from different layers and sites. For Pythia-2.8B we had 5 raters, who each rated one feature from baseline and Gated SAEs trained on each (site, layer) pair from Fig. 8, for a total of 150 features. For Gemma-7B we had 7 raters; one rated 2 features each, and the rest 1 feature each, from baseline or Gated SAEs trained on each (site, layer) pair from Fig. 9, for a total of 192 features.

For each model, raters are shown the features in random order, without revealing which SAE, site, or layer they came from.[7] To assess a feature, the rater decides whether there is an explanation of the feature's behavior, in particular for its highest activating examples. The rater then enters that explanation (if applicable) and selects whether the feature is interpretable ('Yes'), uninterpretable ('No') or maybe interpretable ('Maybe'). All raters are either authors of this paper or colleagues, who have prior experience interpreting SAE features. As an interface we use an open source SAE visualizer library (McDougall, 2024); representative screenshots of the interface may be found at the library's GitHub page.

[6] Although both Gated and baseline SAEs have loss recovered tending to one for high enough L0.

[7] Although due to a debugging issue, Gemma-7B attention SAEs were rated separately, so raters were not blind to that.

Results & analysis Fig. 4 shows interpretability rating distributions by SAE type and LM, marginalising over layers, sites and raters.[8] To test whether Gated SAEs may be more interpretable and estimate the difference, we pair our datapoints according to all covariates (model, layer, site, rater); this lets us control for all of them without making any parametric assumptions, and thus reduces variance in the comparison. We use a one-sided paired Wilcoxon-Pratt signed-rank test, and provide a 90% BCa bootstrap confidence interval for the mean difference between Baseline and Gated labels, where we count 'No' as 0, 'Maybe' as 1, and 'Yes' as 2. Overall the test of the null hypothesis that Gated SAEs are at most as interpretable as Baseline SAEs gets $p = 0.060$ (estimate 0.13, mean difference CI $[0, 0.26]$). This breaks down into $p = 0.15$ on just the Pythia-2.8B data (mean difference CI $[-0.07, 0.33]$), and $p = 0.13$ on just the Gemma-7B data (mean difference CI $[-0.04, 0.29]$).

A Mann-Whitney U rank test on the label differences, comparing results on the two models, fails to reject ($p = 0.95$) the null hypothesis that they're from the same distribution; the same test directly on the labels similarly fails to reject ($p = .84$) the null hypothesis that they're similarly interpretable overall.

The contingency tables used for these results are shown in Fig. 13. The overall conclusion is that, while we can't definitively say the Gated SAE features are more interpretable than those from the Baseline SAEs, they are at least comparable. We provide more analysis of how these break down by site and layer in Appendix H.

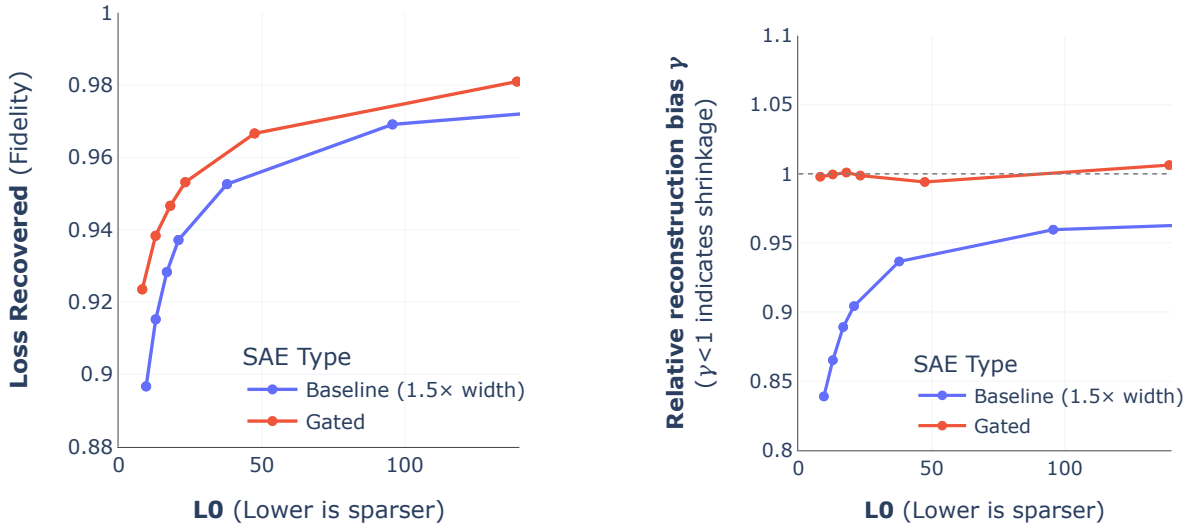
5 Why do Gated SAEs improve SAE training?

5.1 Ablation study

In this section, we vary several parts of the Gated SAE training methodology to gain insight into which aspects of the training drive the observed improvement in performance. Gated SAEs differ from baseline SAEs in many respects, making it easy to incorrectly attribute the performance gains to spurious details without a careful ablation study. Fig. 5a shows Pareto frontiers for these variations; below we describe each variation in turn and discuss our interpretation of the results.

Unfreeze decoder: Here we unfreeze the decoder weights in $\mathcal{L}_{\text{aux}}$ – i.e. allow this auxiliary task to update the decoder weights in addition to training $\mathbf{f}_{\text{gate}}$'s parameters. Although this (slightly) simplifies the loss, there is a reduction in per-

[8] 95% error bars were obtained by modelling each frequency shown as binomial, with p set to the sample frequency, and calculating the 2.5% and 97.5% quantiles.



(a) (b)

Figure 3: (a) Gated SAEs offer better reconstruction fidelity (as measured by loss recovered) at any given level of feature sparsity (as measured by L0); (b) Gated SAEs address shrinkage. These plots compare Gated and baseline SAEs trained on GELU-1L neuron activations; see Appendix D for comparisons on Pythia-2.8B and Gemma-7B.

formance, suggesting that it is beneficial to limit the impact of the L1 sparsity penalty to just those parameters in the SAE that need it – i.e. those used to detect which features are active.

No $\mathbf{r}_{\text{mag}}$: Here we remove the $\mathbf{r}_{\text{mag}}$ scaling parameter in Eq. (6), effectively setting it to zero, further tying $\mathbf{f}_{\text{gate}}$'s and $\mathbf{f}_{\text{mag}}$'s parameters together. With this change, the two encoder sublayers' preactivations can at most differ by an elementwise shift.[9] There is a slight drop in performance, suggesting $\mathbf{r}_{\text{mag}}$ contributes somewhat to the improved performance of the Gated SAE.

Untied encoders: Here we check whether our choice to share the majority of parameters between the two encoders has meaningfully hurt performance, by training Gated SAEs with gating and ReLU encoder parameters completely untied. Despite the greater expressive power of an untied encoder, we see no improvement in performance – in fact a slight deterioration. This suggests our tying scheme (Eq. (6)) – where encoder directions are shared, but magnitudes and biases aren't – is effective at capturing the advantages of using a gated SAE while avoiding the 50% increase in parameter count and inference-time compute of using an untied SAE.

5.2 Is it sufficient to just address shrinkage?

As explained in Section 3.1, SAEs trained with the baseline architecture and L1 loss systematically underestimate the magnitudes of latent features' activations (i.e. shrinkage). Gated SAEs, through modifications to their architecture and loss function, overcome these limitations.

It is natural to ask to what extent the performance improvement of Gated SAEs is solely attributable to addressing shrinkage. Although addressing shrinkage would – all else staying equal – improve reconstruction fidelity, it is not the only way to improve SAEs' performance: for example, Gated SAEs could also improve upon baseline SAEs by learning better encoder directions (for estimating when features are active and their magnitudes) or by learning better decoder directions (i.e. better dictionaries for reconstructing activations).

Here we try to answer this question by comparing Gated SAEs trained as described in Section 3.2 with an alternative (architecturally equivalent) approach that also addresses shrinkage, but in a way that uses frozen encoder and decoder directions from a baseline SAE of equal dictionary size.[10] Any performance improvement over baseline SAEs obtained by this alternative approach (which we dub "baseline + rescale & shift") can only be due to better estimations of active feature magnitudes, since by construction an SAE parameterized by "baseline + rescale & shift" shares the same encoder and decoder directions as a baseline SAE.

[9] Because the two biases $\mathbf{b}_{\text{gate}}$ and $\mathbf{b}_{\text{mag}}$ can still differ.

[10] Concretely, we do this by training baseline SAEs, freezing their weights, and then learning additional rescale and shift parameters (similar to Wright and Sharkey (2024)) to be applied to the (frozen) encoder pre-activations before estimating feature magnitudes.

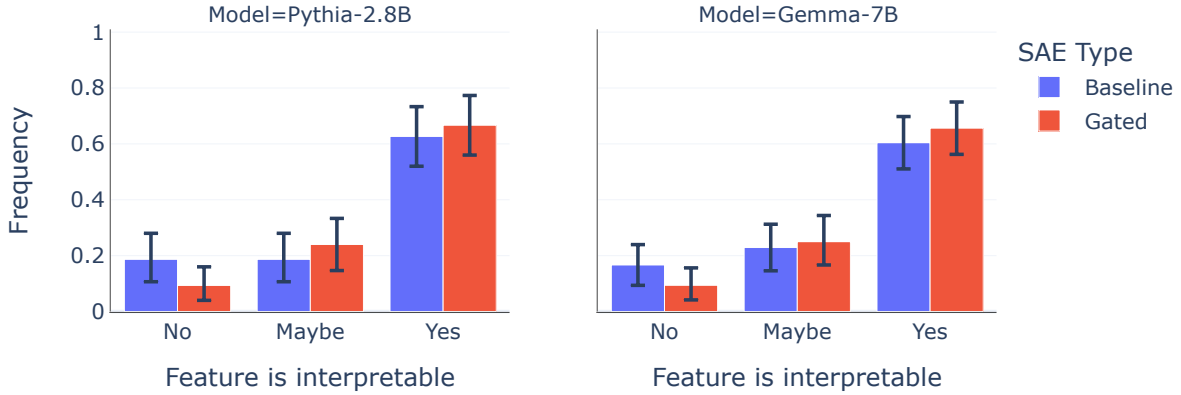


Figure 4: Proportions of SAE features rated as interpretable / uninterpretable / maybe interpretable by SAE type (Gated or baseline) and language model. Gated and baseline SAEs are similarly interpretable, with a mean difference (in favor of Gated SAEs) of 0.13 (95% CI $[0, 0.26]$) after aggregating ratings for both models.

As shown in Fig. 5b, although resolving shrinkage only ("baseline + rescale & shift") does improvement baseline SAEs' performance a little, a significant gap remains with respect to the performance of Gated SAEs. This suggests that the benefit of the gated architecture and loss comes from learning better encoder and decoder directions, not just from overcoming shrinkage. In Appendix C we explore further how Gated and baseline SAEs' decoders differ by replacing their respective encoders with an optimization algorithm at inference time.

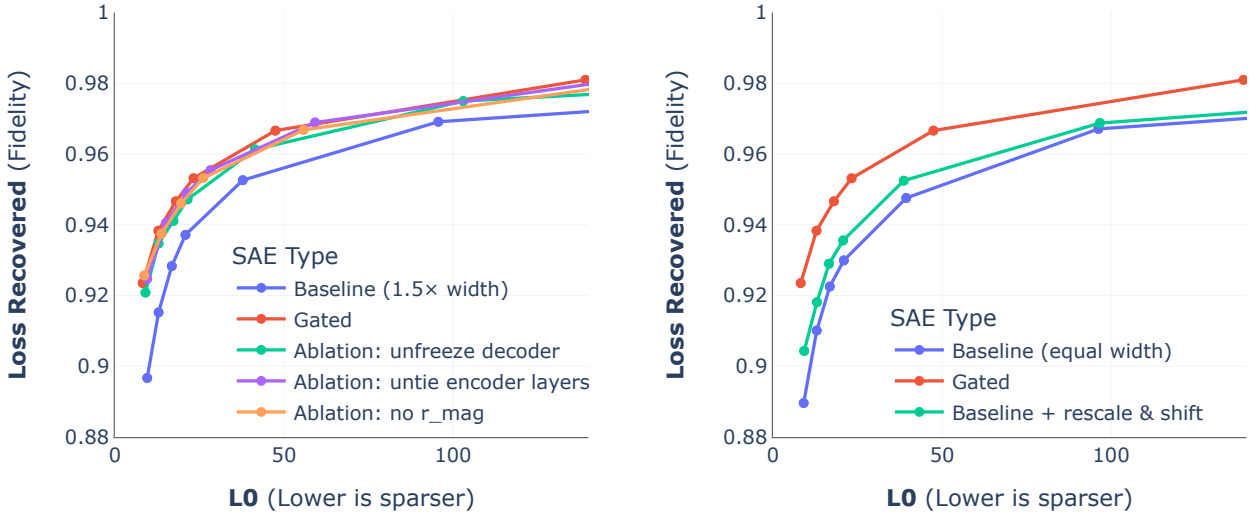
6 Related work

Mechanistic interpretability Recent work in mechanistic interpretability has found recurring components in small and large LMs (Olsson et al., 2022), identified computational subgraphs that carry out specific tasks in small LMs (circuits; (Wang et al., 2023)) and reverse-engineered how toy tasks are carried out in small transformers (Nanda et al., 2023). A central difficulty in this kind of work is choosing the right units of analysis. Sparse linear features have been identified as a promising candidate in prior work (Yun et al., 2023; Tamkin et al., 2023). The superposition hypothesis outlined by Elhage et al. (2022b) also provided a theoretical basis for this theory, sparking a new interest in using SAEs specifically to learn a feature basis (Sharkey et al., 2022; Bricken et al., 2023; Cunningham et al., 2023; Kissane et al., 2024a;b; Bloom, 2024), as well as using SAEs directly for circuit analysis (Marks et al., 2024). Other work has drawn awareness to issues or drawbacks with SAE training for this purpose, some of which our paper mitigates. Wright and Sharkey (2024) raised awareness of shrinkage and proposed addressing this via fine-tuning. Gated SAEs, as discussed, resolve shrinkage during training. (Olah et al., 2024a; Templeton et al., 2024; Batson et al., 2024; Olah et al., 2024b) have also proposed general SAE training methodology improvements, which are mostly orthogonal to the architectural changes discussed in this work. In parallel work, Taggart (2024) finds early improvements using a Jump ReLU (Erichson et al., 2019), but with a different loss function, and without addressing the problems of the L1 penalty.

Classical dictionary learning Research into the general problem of sparse dictionary learning precedes transformers, and even deep learning. For example, sparse coding (Elad, 2010) studies how discrete and continuous representations can involve more representations than basis vectors, and sparse representations are also studied in neuroscience (Thorpe, 1989; Olshausen and Field, 1997). One dictionary learning algorithm, k-SVD (Aharon et al., 2006) also uses two stages to learn a dictionary like Gated SAEs. Although classical dictionary learning algorithms can be more powerful than SAEs (Appendix C), they are less suited for downstream uses like weights-based circuit analysis or attribution patching (Syed et al., 2023; Kramár et al., 2024), because they typically use an iterative algorithm to decompose activations, whereas SAEs make feature extraction explicit via the encoder. Bricken et al. (2023) have also argued that classical algorithms may be 'too strong', in the sense they may learn features the LM itself could not access, whereas SAEs uses components similar to a LM's MLP layer to decompose activations.

7 Conclusion

In this work we introduced Gated SAEs which are a Pareto improvement in terms of reconstruction quality and sparsity compared to baseline SAEs (Section 4.1), and are comparably interpretable (Section 4.2). We showed via an ablation study that every key part of the Gated SAE methodology was necessary for strong performance (Section 5.1).



(a) (b)

Figure 5: (a) Our ablation study on GELU-1L MLP neuron activations indicates: (i) the importance of freezing the decoder in the auxiliary task $\mathcal{L}_{\text{aux}}$ used to train $\mathbf{f}_{\text{gate}}$'s parameters; (ii) tying encoder weights according to Eq. (6) is slightly beneficial for performance (in addition to yielding a significant reduction in parameter count and inference compute); (iii) further simplifying the encoder weight tying scheme in Eq. (6) by removing $\mathbf{r}_{\text{mag}}$ is mildly harmful to performance. (b) Evidence from GELU-1L that the performance improvement of gated SAEs does not solely arise from addressing shrinkage (systematic underestimation of latent feature activations): taking a frozen baseline SAE's parameters and learning $\mathbf{r}_{\text{mag}}$ and $\mathbf{b}_{\text{mag}}$ parameters on top of them (green line) does successfully resolve shrinkage, by decoupling feature magnitude estimation from active feature detection; however, it explains only a small part of the performance increase of gated SAEs (red line) over baseline SAEs (blue line).

This represents significant progress on improving Dictionary Learning on LMs – at many sites, Gated SAEs require half the L0 to achieve the same loss recovered (Fig. 8). This is likely to improve work that uses SAEs to steer language models (Nanda et al., 2024), interpret circuits (Marks et al., 2024), or understand LM components across the full distribution (Bricken et al., 2023).

Limitations & future work. Our benchmarking study focused on GELU-1L and models in the Pythia and Gemma families. It is therefore not certain that these results will generalise to other model families. On the other hand, the theoretical underpinnings of the Gated SAE architecture (Section 3) make no assumptions about LM architecture, suggesting Gated SAEs should be a Pareto improvement more generally. While we have confirmed that Gated SAE features are comparably interpretable to baseline SAE features, it does not necessarily follow that Gated SAE decompositions are equally useful for mechanistic interpretability. It is certainly possible that human interpretability of SAE features is only weakly correlated with either: (i) identification of the causally meaningful directions in a LM's activations; or (ii) usefulness on downstream tasks like circuit analysis or steering. A framework for scalably and objectively evaluating the usefulness of SAE decompositions (gated or otherwise) is still in its early stages (Makelov et al., 2024) and further progress in this area would be highly valuable. It is plausible that some of the performance gap between Gated and baseline SAEs could be closed by inexpensive inference-time interventions that prune the many low activating features that tend to appear in baseline SAEs, mimicking Gated SAEs' thresholding mechanism. Finally, we would be most excited to see progress on using dictionary learning techniques to further interpretability in general, such as to improve circuit finding (Conmy et al., 2023; Marks et al., 2024) or steering (Turner et al., 2023) in language models, and hope that Gated SAEs can serve to accelerate such work.

Impact Statement We see this work as foundational interpretability research, without direct applications, and thus without direct positive or negative ethical considerations, except those that all machine learning research on language models has.

References

M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006. doi: 10.1109/TSP.2006.881199.

J. Batson, B. Chen, A. Jones, A. Templeton, T. Conerly, J. Marcus, T. Henighan, N. L. Turner, and A. Pearce. Circuits Updates - March 2024. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/mar-update/index.html.

S. Biderman, H. Schoelkopf, Q. G. Anthony, H. Bradley, K. O'Brien, E. Hallahan, M. A. Khan, S. Purohit, U. S. Prashanth, E. 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. Apache License 2.0.

J. Bloom. Open Source Sparse Autoencoders for all Residual Stream Layers of GPT-2 Small, 2024. https://www.alignmentforum.org/posts/f9EgfLSurAiqRJySD.

T. Blumensath and M. E. Davies. Gradient pursuits. *IEEE Transactions on Signal Processing*, 56(6):2370–2382, 2008.

T. Bolukbasi, A. Pearce, A. Yuan, A. Coenen, E. Reif, F. Viégas, and M. Wattenberg. An interpretability illusion for bert. *arXiv preprint arXiv:2104.07143*, 2021.

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

A. Conmy. My best guess at the important tricks for training 1L SAEs, Dec 2023. https://www.lesswrong.com/posts/yJsLNWtmzcgPJgvro/my-best-guess-at-the-important-tricks-for-training-1l-saes.

A. Conmy, A. N. Mavor-Parker, A. Lynch, S. Heimersheim, and A. Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability, 2023.

H. Cunningham, A. Ewart, L. Riggs, R. Huben, and L. Sharkey. Sparse autoencoders find highly interpretable features in language models, 2023.

Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 933–941. JMLR.org, 2017.

M. Elad. *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer, New York, 2010. ISBN 978-1-4419-7010-7. doi: 10.1007/978-1-4419-7011-4.

N. Elhage, N. Nanda, C. Olsson, T. Henighan, N. Joseph, B. Mann, A. Askell, Y. Bai, A. Chen, T. Conerly, N. DasSarma, D. Drain, D. Ganguli, Z. Hatfield-Dodds, D. Hernandez, A. Jones, J. Kernion, L. Lovitt, K. Ndousse, D. Amodei, T. Brown, J. Clark, J. Kaplan, S. McCandlish, and C. Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. URL https://transformer-circuits.pub/2021/framework/index.html.

N. Elhage, T. Hume, C. Olsson, N. Nanda, T. Henighan, S. Johnston, S. ElShowk, N. Joseph, N. DasSarma, B. Mann, D. Hernandez, A. Askell, K. Ndousse, A. Jones, D. Drain, A. Chen, Y. Bai, D. Ganguli, L. Lovitt, Z. Hatfield-Dodds, J. Kernion, T. Conerly, S. Kravec, S. Fort, S. Kadavath, J. Jacobson, E. Tran-Johnson, J. Kaplan, J. Clark, T. Brown, S. McCandlish, D. Amodei, and C. Olah. Softmax linear units. *Transformer Circuits Thread*, 2022a. https://transformer-circuits.pub/2022/solu/index.html.

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

N. B. Erichson, Z. Yao, and M. W. Mahoney. Jumprelu: A retrofit defense strategy for adversarial attacks, 2019.

Gemma Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, L. Sifre, M. Rivière, M. S. Kale, J. Love, P. Tafti, L. Hussenot, and et al. Gemma, 2024. URL https://www.kaggle.com/m/3301. Apache License 2.0.

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

N. P. Jouppi, D. H. Yoon, G. Kurian, S. Li, N. Patil, J. Laudon, C. Young, and D. Patterson. A domain-specific supercomputer for training deep neural networks. *Communications of the ACM*, 63(7):67–78, 2020.

C. Kissane, R. Krzyzanowski, A. Conmy, and N. Nanda. Sparse autoencoders work on attention layer outputs. Alignment Forum, 2024a. https://www.alignmentforum.org/posts/DtdzGwFh9dCfsekZZ.

C. Kissane, R. Krzyzanowski, A. Conmy, and N. Nanda. Attention SAEs scale to GPT-2 Small. Alignment Forum, 2024b. https://www.alignmentforum.org/posts/FSTRedtjuHa4Gfdbr.

J. Kramár, T. Lieberum, R. Shah, and N. Nanda. Atp*: An efficient and scalable method for localizing llm behaviour to components. *arXiv preprint arXiv:2403.00745*, 2024.

A. Makelov, G. Lange, and N. Nanda. Towards principled evaluations of sparse autoencoders for interpretability and control. In *ICLR 2024 Workshop on Secure and Trustworthy Large Language Models*, 2024. URL https://openreview.net/forum?id=MHIX9H8aYF.

S. Marks, C. Rager, E. J. Michaud, Y. Belinkov, D. Bau, and A. Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models, 2024.

C. McDougall. SAE Visualizer, 2024. https://github.com/callummcdougall/sae_vis.

N. Nanda. GELU-1L, 2022. URL https://huggingface.co/NeelNanda/GELU_1L512W_C4_Code. MIT License.

N. Nanda. Open Source Replication & Commentary on Anthropic's Dictionary Learning Paper, Oct 2023. https://www.alignmentforum.org/posts/aPTgTKC45dWvL9XBF.

N. Nanda, L. Chan, T. Lieberum, J. Smith, and J. Steinhardt. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=9XFSbDPmdW.

N. Nanda, A. Conmy, L. Smith, S. Rajamanoharan, T. Lieberum, J. Kramár, and V. Varma. [Summary] Progress Update #1 from the GDM Mech Interp Team. Alignment Forum, 2024. https://www.alignmentforum.org/posts/HpAr8k74mW4ivCvCu.

A. Ng. Sparse autoencoder, 2011. CS294A Lecture notes, http://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf.

C. Olah. Mechanistic interpretability, variables, and the importance of interpretable bases, 2022. https://www.transformer-circuits.pub/2022/mech-interp-essay.

C. Olah, N. Cammarata, L. Schubert, G. Goh, M. Petrov, and S. Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001.

C. Olah, S. Carter, A. Jermyn, J. Batson, T. Henighan, T. Conerly, J. Marcus, A. Templeton, B. Chen, and N. L. Turner. Circuits Updates - January 2024. *Transformer Circuits Thread*, 2024a. URL https://transformer-circuits.pub/2024/jan-update/index.html.

C. Olah, S. Carter, A. Jermyn, J. Batson, T. Henighan, J. Lindsey, T. Conerly, A. Templeton, J. Marcus, and T. Bricken. Circuits Updates - April 2024. *Transformer Circuits Thread*, 2024b. URL https://transformer-circuits.pub/2024/april-update/index.html.

B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37(23):3311–3325, 1997. doi: 10.1016/S0042-6989(97)00169-7.

C. Olsson, N. Elhage, N. Nanda, N. Joseph, N. DasSarma, T. Henighan, B. Mann, A. Askell, Y. Bai, A. Chen, T. Conerly, D. Drain, D. Ganguli, Z. Hatfield-Dodds, D. Hernandez, S. Johnston, A. Jones, J. Kernion, L. Lovitt, K. Ndousse, D. Amodei, T. Brown, J. Clark, J. Kaplan, S. McCandlish, and C. Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html.

K. Park, Y. J. Choe, and V. Veitch. The linear representation hypothesis and the geometry of large language models, 2023.

Y. Pati, R. Rezaiifar, and P. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*, pages 40–44 vol.1, 1993. doi: 10.1109/ACSSC.1993.342465.

L. Sharkey, D. Braun, and B. Millidge. [interim research report] taking features out of superposition with sparse autoencoders, 2022. https://www.alignmentforum.org/posts/z6QQJbtpkEAX3Aojj.

N. Shazeer. GLU variants improve transformer. *CoRR*, abs/2002.05202, 2020. URL https://arxiv.org/abs/2002.05202.

A. Syed, C. Rager, and A. Conmy. Attribution patching outperforms automated circuit discovery. *arXiv preprint arXiv:2310.10348*, 2023.

G. M. Taggart. Prolu: A nonlinearity for sparse autoencoders, 2024. https://www.lesswrong.com/posts/HEpufTdakGTTKgoYF/prolu-a-pareto-improvement-for-sparse-autoencoders.

A. Tamkin, M. Taufeeque, and N. D. Goodman. Codebook features: Sparse and discrete interpretability for neural networks, 2023.

A. Templeton, J. Batson, T. Henighan, T. Conerly, J. Marcus, A. Golubeva, T. Bricken, and A. Jermyn. Circuits Updates - February 2024. *Transformer Circuits Thread*, 2024. https://transformer-circuits.pub/2024/feb-update/index.html.

S. J. Thorpe. Local vs. distributed coding. *Intellectica*, 8: 3–40, 1989.

A. M. Turner, L. Thiergart, D. Udell, G. Leech, U. Mini, and M. MacDiarmid. Activation addition: Steering language models without optimization, 2023.

K. R. Wang, A. Variengien, A. Conmy, B. Shlegeris, and J. Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=NpsVSN6o4ul.

B. Wright and L. Sharkey. Addressing feature suppression in SAEs, Feb 2024. https://www.alignmentforum.org/posts/3JuSjTZyMzaSeTxKk/addressing-feature-suppression-in-saes.

Z. Yun, Y. Chen, B. A. Olshausen, and Y. LeCun. Transformer visualization via dictionary learning: contextualized embedding as a linear superposition of transformer factors, 2023.

Appendix

A Metrics for evaluating SAEs

SAEs are expected to decompose input activations sparsely, and yet in a manner that allows for faithful reconstruction. L0 and loss recovered are two metrics typically used (Bricken et al., 2023) to measure sparsity and reconstruction fidelity respectively. These are defined as follows:

- The **L0** of a SAE is defined by the average number of active features on a given input, i.e $\mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \|\mathbf{f}(\mathbf{x})\|_0$.
- The **loss recovered** of a SAE is calculated from the average cross-entropy loss of the language model on an evaluation dataset, when the SAE's reconstructions are spliced into it. If we denote by $\text{CE}(\phi)$ the average loss of the language model when we splice in a function $\phi : \mathbb{R}^n \to \mathbb{R}^n$ at the SAE's site during the model's forward pass, then loss recovered is

$$1 - \frac{\text{CE}(\hat{\mathbf{x}} \circ \mathbf{f}) - \text{CE}(\text{Id})}{\text{CE}(\zeta) - \text{CE}(\text{Id})}, \tag{8}$$

where $\hat{\mathbf{x}} \circ \mathbf{f}$ is the autoencoder function, $\zeta : \mathbf{x} \mapsto \mathbf{0}$ the zero-ablation function and $\text{Id} : \mathbf{x} \mapsto \mathbf{x}$ the identity function. According to this definition, a SAE that always outputs the zero vector as its reconstruction would get a loss recovered of 0%, whereas a SAE that reconstructs its inputs perfectly would get a loss recovered of 100%.

B Measuring shrinkage

As described in Section 3.1, the L1 sparsity penalty used to train baseline SAEs causes feature activations to be systematically underestimated, a phenomenon called *shrinkage*. Since this in turn shrinks the reconstructions produced by the SAE decoder, we can observe the extent to which a trained SAE is affected by shrinkage by measuring the average norm of its reconstructions.

Concretely, the metric we use is the *relative reconstruction bias*,

$$\gamma := \arg\min_{\gamma'} \mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \left[\|\hat{\mathbf{x}}_{\text{SAE}}(\mathbf{x})/\gamma' - \mathbf{x}\|_2^2 \right], \tag{9}$$

i.e. γ^{-1} is the optimum multiplicative factor by which an SAE's reconstructions should be rescaled in order to minimise the L2 reconstruction loss; $\gamma = 1$ for an unbiased SAE and $\gamma < 1$ when there's shrinkage.[11] Explicitly solving the optimization problem in Eq. (9), the relative reconstruction bias can be expressed analytically in terms of the mean SAE reconstruction loss, the mean squared norm of input activations and the mean squared norm of SAE reconstructions, making γ easy to compute and track during training:

$$\gamma = \frac{\mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \left[\|\hat{\mathbf{x}}_{\text{SAE}}(\mathbf{x})\|_2^2 \right]}{\mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \left[\hat{\mathbf{x}}_{\text{SAE}}(\mathbf{x}) \cdot \mathbf{x} \right]} \tag{10}$$

$$= \frac{2\,\mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \left[\|\hat{\mathbf{x}}_{\text{SAE}}(\mathbf{x})\|_2^2 \right]}{\mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \left[\|\hat{\mathbf{x}}_{\text{SAE}}(\mathbf{x})\|_2^2 \right] + \mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \left[\|\mathbf{x}\|_2^2 \right] - \mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \left[\|\hat{\mathbf{x}}_{\text{SAE}}(\mathbf{x}) - \mathbf{x}\|_2^2 \right]}, \tag{11}$$

where the second equality makes use of the identity $2\mathbf{a} \cdot \mathbf{b} \equiv \|\mathbf{a}\|_2^2 + \|\mathbf{b}\|_2^2 - \|\mathbf{a} - \mathbf{b}\|_2^2$. Notice from the second expression for γ that an unbiased reconstruction ($\gamma = 1$) therefore satisfies

$$\mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \left[\|\hat{\mathbf{x}}_{\text{SAE}}(\mathbf{x})\|_2^2 \right] = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \left[\|\mathbf{x}\|_2^2 \right] - \mathbb{E}_{\mathbf{x}\sim\mathcal{D}} \left[\|\hat{\mathbf{x}}_{\text{SAE}}(\mathbf{x}) - \mathbf{x}\|_2^2 \right].$$

In other words, an unbiased but imperfect SAE (i.e. one that has non-zero reconstruction loss) must have mean squared reconstruction norm that is strictly *less than* the mean squared norm of its inputs *even without shrinkage*. Shrinkage makes the mean squared reconstruction norm even smaller.

[11] We have defined γ this way round so that $\gamma < 1$ intuitively corresponds to shrinkage.

C Inference-time optimization

The task SAEs perform can be split into two sub-tasks: sparse coding, or learning a set of features from a dataset, and sparse approximation, where a given datapoint is approximated as a sparse linear combination of these features. The decoder weights are the set of learned features, and the mapping represented by the encoder is a sparse approximation algorithm. Formally, sparse approximation is the problem of finding a vector $\boldsymbol{\alpha}$ that minimises;

$$\boldsymbol{\alpha} = \arg\min \|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 \quad s.t. \quad \|\boldsymbol{\alpha}\|_0 < \gamma \tag{12}$$

i.e. that best reconstructs the signal $\mathbf{x}$ as a linear combination of vectors in a dictionary $\mathbf{D}$, subject to a constraint on the L0 pseudo-norm on $\boldsymbol{\alpha}$. Sparse approximation is a well studied problem, and SAEs are a *weak* sparse approximation algorithm. SAEs, at least in the formulation conventional in dictionary learning for language models, in fact solve a slightly more restricted version of this problem where the weights $\boldsymbol{\alpha}$ on each feature are constrained to be non-negative, leading to the related problem

$$\boldsymbol{\alpha} = \arg\min \|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 \quad s.t. \quad \|\boldsymbol{\alpha}\|_0 < \gamma, \boldsymbol{\alpha} > 0 \tag{13}$$

In this paper, we do not explore using more powerful algorithms for sparse coding. This is partly because we are using SAEs not just to recover *a* sparse reconstruction of activations of a LM; ideally we hope that the learned features will coincide with the linear representations actually used by the LM, under the superposition hypothesis. Prior work (Bricken et al., 2023) has argued that SAEs are more likely to recover these due to the correspondence between the SAE encoder and the structure of the network itself; the argument is that it is implausible that the network can make use of features which can only be recovered from the vector via an iterative optimisation algorithm, whereas the structure of the SAE means that it can only find features whose presence can be predicted well by a simple linear mapping. Whether this is true remains, in our view, an important question for future work, but we do not address it in this paper.

In this section we discuss some results obtained by using the dictionaries learned via SAE training, but replacing the encoder with a different sparse approximation algorithm at inference time. This allows us to compare the dictionaries learned by different SAE training regimes independently of the quality of the encoder. It also allows us to examine the gap between the sparse reconstruction performed by the encoder against the baseline of a more powerful sparse approximation algorithm. As mentioned, for a fair comparison to the task the encoder is trained for, it is important to solve the sparse approximation problem of Eq. (13), rather than the more conventional formulation of Eq. (12), but most sparse approximation algorithms can be modified to solve this with relatively minor changes.

Solving Eq. (13) exactly is equivalent to integer linear programming, and is NP hard. The integer linear programs in question would be large, as our SAE decoders routinely have hundreds of thousands of features, and solving them to guaranteed optimality would likely be intractable. Instead, as is commonly done, we use iterative greedy algorithms to find an approximate solution. While the solution found by these sparse approximation algorithms is not guaranteed to be the global optimum, these are significantly more powerful than the SAE encoder, and we feel it is acceptable in practice to treat them as an upper bound on possible encoder performance.

For all results in this section, we use gradient pursuit, as described in Blumensath and Davies (2008), as our inference time optimisation (ITO) algorithm. This algorithm is a variant of orthogonal matching pursuit (Pati et al., 1993) which solves the orgothonalisation of the residual to the span of chosen dictionary elements approximately at every step rather than exactly, but which only requires matrix multiplies rather than matrix solves and is easier to implement on accelerators as a result. It is possibly not crucial for performance that our optimisation algorithm be implementable on TPUs, but being able to avoid a host-device transfer when splicing this into the forward pass allowed us to re-use our existing evaluation pipeline with minimal changes.

When we use a sparse approximation algorithm at test time, we simply use the decoder of a trained SAE as a dictionary, ignoring the encoder. This allows us to sweep the target sparsity at test time without retraining the model, meaning that we can plot an entire Pareto frontier of loss recovered against sparsity for a single decoder, as in done in Fig. 7.

Fig. 6 compares the loss recovered when using ITO for a suite of SAEs decoders trained with both methods at three different test time L0 thresholds. This graph shows a somewhat surprising result; while Gated SAEs learn better decoders generally, and often achieve the best loss recovered using ITO close to their training sparsity, SAE decoders are often outperformed by decoders which achieved a higher test time L0; it's better to do ITO with a target L0 of 10 with an decoder with an achieved L0 of around 100 during training than one which was actually trained with this level of sparsity. For instance, the left hand panel in Fig. 6 shows that SAEs with a training L0 of 100 are better than those with an L0 of around 10 at almost every sparsity level in terms of ITO reconstruction. However, gated SAE dictionaries have a small but real advantage over standard SAEs in terms of loss recovered at most target sparsity levels, suggesting that part of the advantage of gated SAEs is that they learn better dictionaries as well as addressing

issues with shrinkage. However, there are some subtleties here; for example, we find that baseline SAEs trained with a lower sparsity penalty (higher training L0) often outperform more sparse baseline SAEs according to this measure, and the best performing baseline SAE (L0 ≈ 99) is comparable to the best performing Gated SAE (L0 ≈ 20).

Fig. 7 compares the Pareto frontiers of a baseline model and a gated model to the Pareto frontier of an ITO sweep of the best performing dictionary of each. Note that, while the Pareto curve of the baseline dictionary is formed by several models as each encoder is specialised to a given sparsity level, as mentioned, ITO lets us plot a Pareto frontier by sweeping the target sparsity with a single dictionary; here we plot only the best performing dictionary from each model type to avoid cluttering the figure. This figure suggests that the performance gap between the encoder and using ITO is smaller for the gated model. Interestingly, this cannot solely be explained by addressing shrinkage, as we demonstrate by experimenting with a baseline model which learns a rescale and shift with a frozen encoder and decoder directions.

D More loss recovered / L0 Pareto frontiers

In Fig. 8 we show that Gated SAEs outperform baseline SAEs. In Fig. 9 we show that Gated SAEs ourperform baseline SAEs at all but one MLP output or residual stream site that we tested on.

In Fig. 9 at the attention output pre-linear site at layer 27, loss recovered is bigger than 1.0. On investigation, we found that the dataset used to train the SAE was not identical to Gemma's pretraining dataset, and at this site it was possible to mean ablate this quantity and decrease loss – explaining why SAE reconstructions had lower loss than the original model.

E Further shrinkage plots

In Fig. 10, we show that Gated SAEs resolve shrinkage, as measured by relative reconstruction bias (Appendix B), in Pythia-2.8B.

F Training and evaluation: hyperparameters and other details

F.1 Training

F.1.1 General training details

Other details of SAE training are:

- **SAE Widths**. Our SAEs have width 2^{17} for most baseline SAEs, 3×2^{16} for Gated SAEs, except for the (Pythia-2.8B, Residual Stream) sites we used 2^{15} for baseline and 3×2^{14} for Gated since early runs at these sites had lots of learned feature death.
- **Training data**. We use activations from hundreds of millions to billions of activations from LM forward passes as input data to the SAE. Following Nanda (2023), we use a shuffled buffer of these activations, so that optimization steps don't use data from highly correlated activations.[12]
- **Resampling**. We used *resampling*, a technique which at a high-level reinitializes features that activate extremely rarely on SAE inputs periodically throughout training. We mostly follow the approach described in the 'Neuron Resampling' appendix of Bricken et al. (2023), except we reapply learning rate warm-up after each resampling event, reducing learning rate to 0.1x the ordinary value, and, increasing it with a cosine schedule back to the ordinary value over the next 1000 training steps.
- **Optimizer hyperparameters**. We use the Adam optimizer with $\beta_2 = 0.999$ and $\beta_1 = 0.0$, following Templeton et al. (2024), as we also find this to be a slight improvement to training. We use a learning rate warm-up. See Appendix F.1.2 for learning rates of different experiment.
- **Decoder weight norm constraints**. Templeton et al. (2024) suggest constraining columns to have *at most* unit norm (instead of exactly unit norm), which can help distinguish between productive and unproductive feature directions (although it should have no systematic impact on performance). However, we follow the original approach of constraining columns to have exact unit norms in this work for the sake of simplicity.
- **Compute resources**. Individual SAEs were each trained on TPU-v3 slices with a 2x2 topology (Jouppi et al., 2020). The same chips were used to generate LM activations on-the-fly, train SAE parameters and evaluate SAEs during training, using up to 8-way model parallelism. With this setup, the time to train a SAE varies by SAE width, LM residual stream dimension, sequence length, layer and site.[13] We also used a negligible amount of compute on resampling (Appendix F), evaluation (e.g. Figure 1) and interpretability experiments (Section 4.2). Training wall clock time ranges from around 7 hours to train on GELU-1L MLP activations to around 47 hours to train

[12] In contrast to earlier findings (Conmy, 2023), we found that when using Pythia-2.8B's activations from sequences of length 2048, rather than GELU-1L's activations from sequences of length 128, it was important to shuffle the 10^6 length activation buffer used to train our SAEs.

[13] The FLOPs required to compute LM activations increase with layer; SAEs trained on MLP activations have a higher parameter count than those trained on MLP outputs, attention outputs or the residual stream.

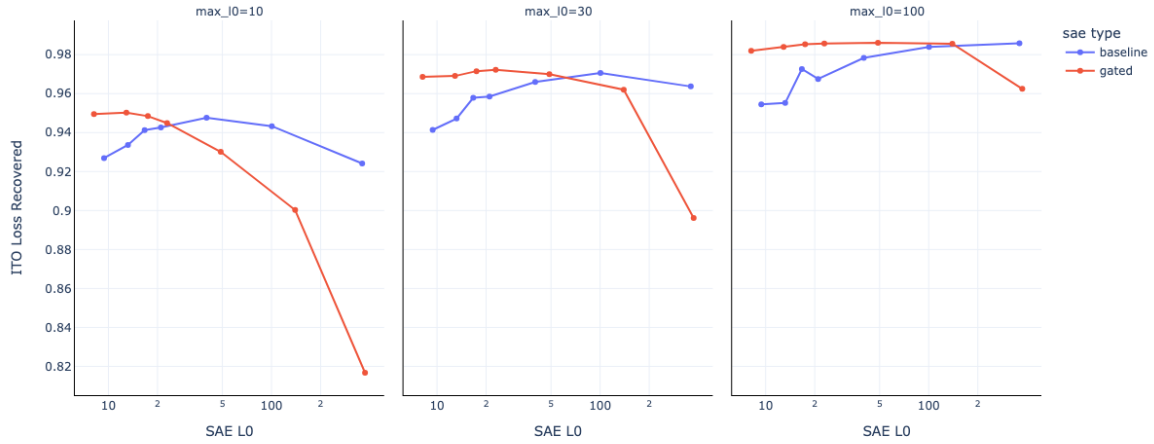


Figure 6: This figure compares the ITO performance of different decoders across a sweep for decoders trained using a baseline SAE and the gated method, at three different test time target sparsities. Gated SAEs trained at lower target sparsities consistently achieve better dictionaries by this measure. Interestingly, the best performing baseline dictionary by this measure often has a much higher test time sparsity than the target; for instance, at a test time sparsity of 30, the best baseline SAE was the one that had a test time sparsity of more like 100. This could be an artifact of the fact that the L0 measure is quite sensitive to noise, and standard SAE architectures tend to have a reasonable number of features with very low activation.

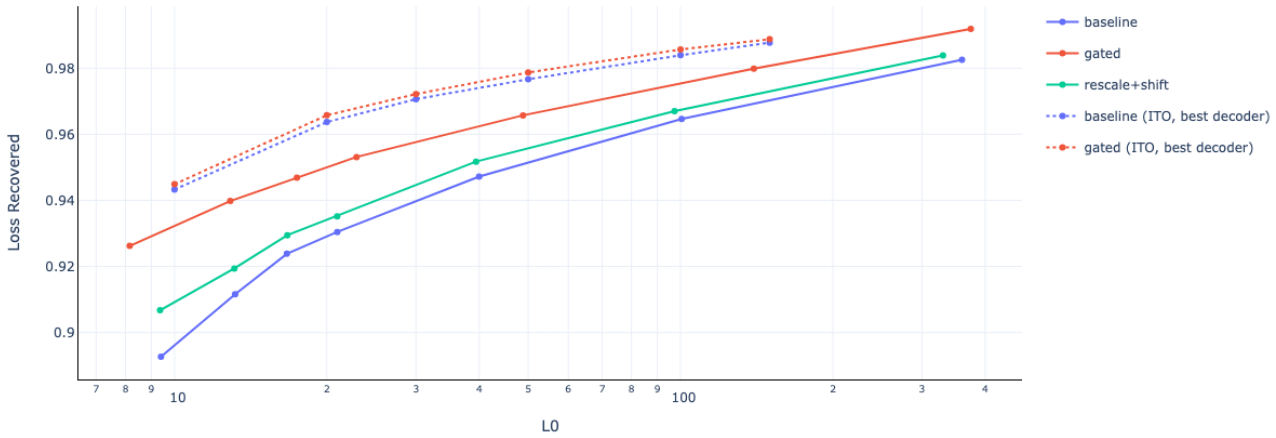


Figure 7: Pareto frontiers of a baseline SAE, a baseline SAE with learned rescale and shift (to account for shrinkage) and a gated SAE across different sparsity lambdas, compared to the ITO Pareto frontier of the best decoder of each type with ITO, varying the target sparsity. The best gated encoder is better than the best standard encoder by this measure, but the difference is marginal. As shown in the plot above, the best baseline encoder by the ITO measure had a much larger test time sparsity (around 100) than the best gated model (around 30). This figure suggests that the gap between SAE performance and 'optimal' performance, if we assume that ITO is close to the maximum possible reconstruction using the given encoder, is much smaller for the gated model.

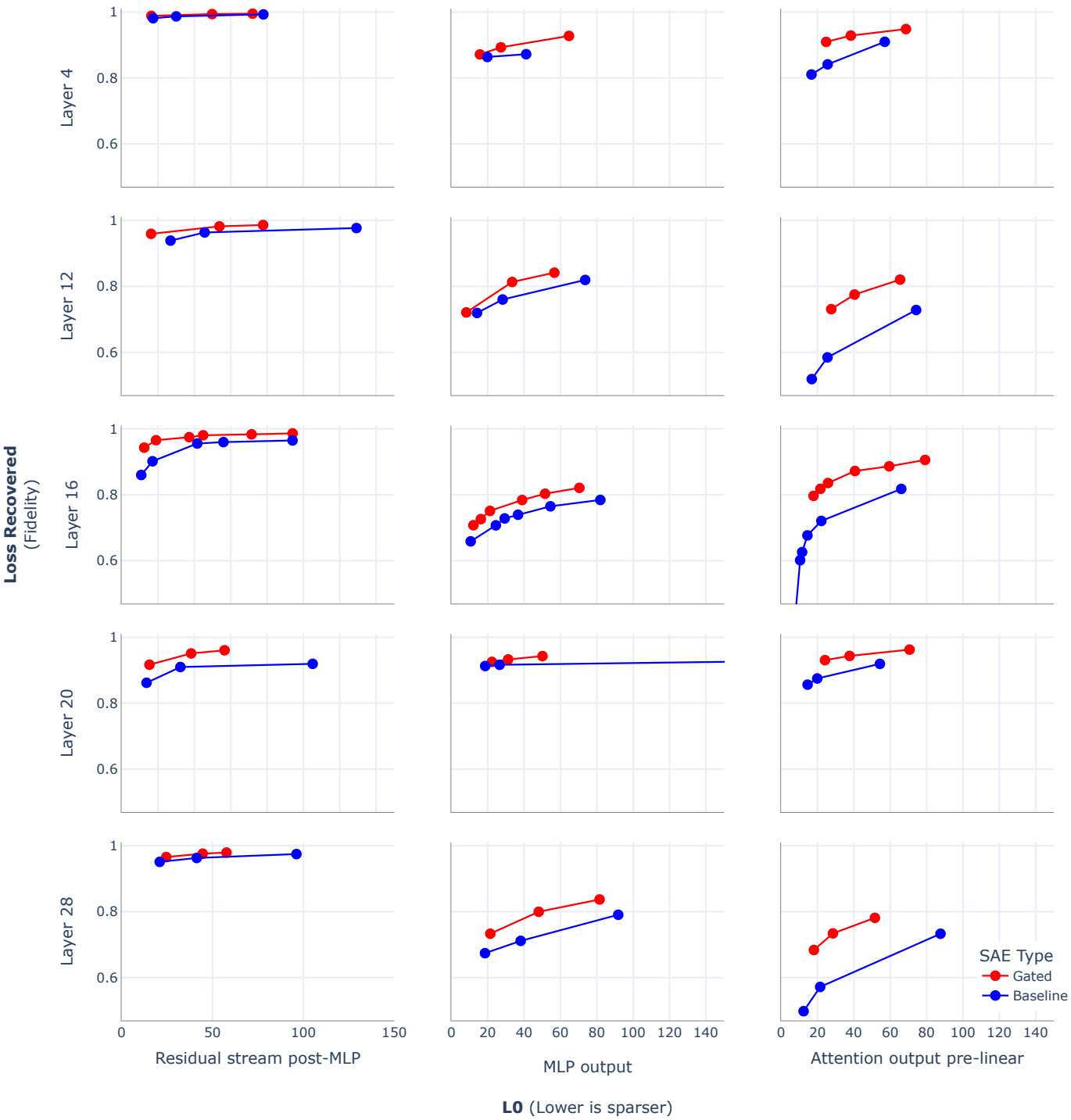


Figure 8: Gated SAEs throughout Pythia-2.8B. At all sites we tested, Gated SAEs are a Pareto improvement. In every plot, the SAE with maximal loss recovered was a Gated SAE.

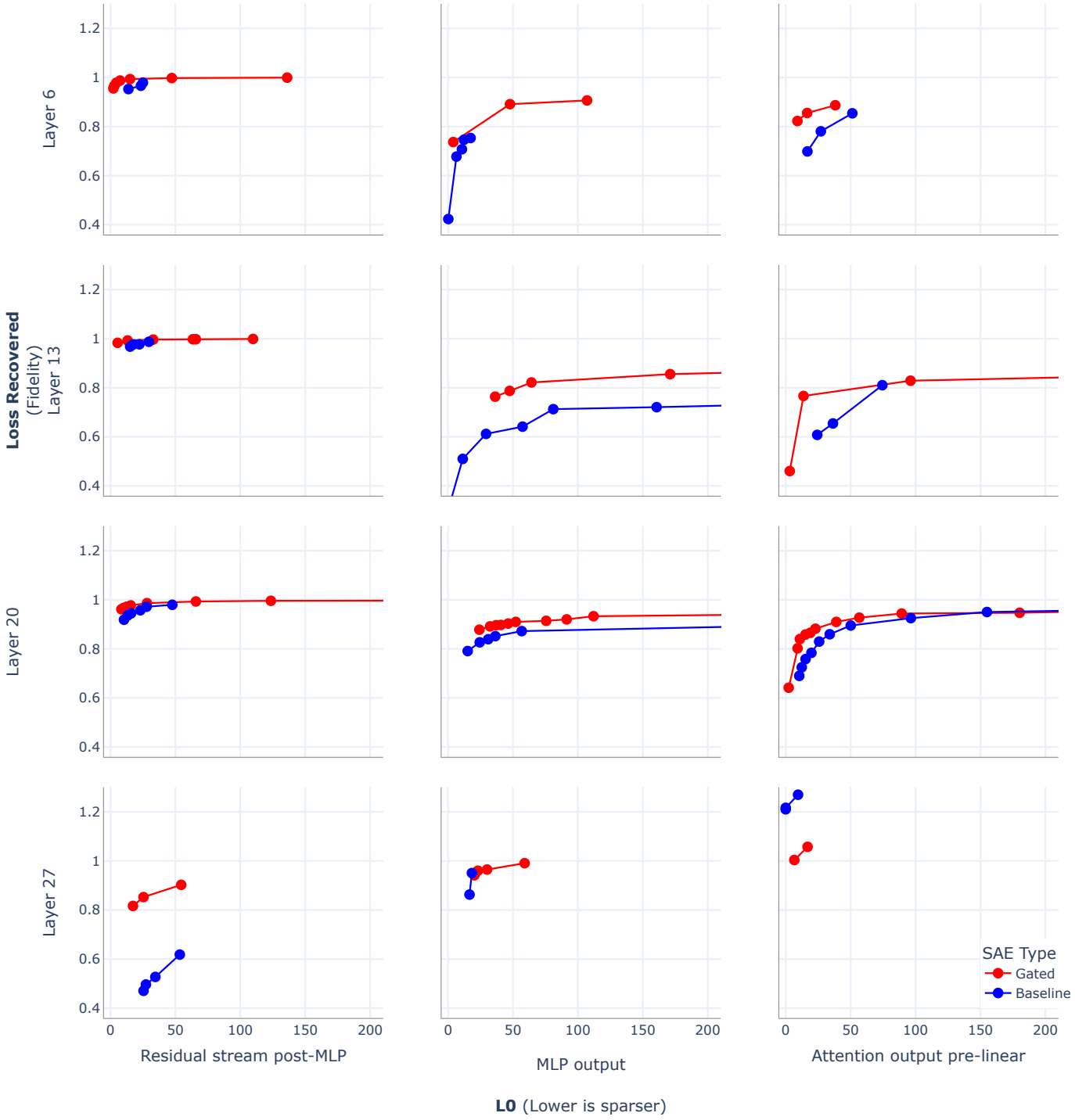


Figure 9: Gated and Normal Pareto-Optimal SAEs for Gemma-7B – see Appendix D for a discussion of the anomalies (such as the Layer 27 attention output SAEs).

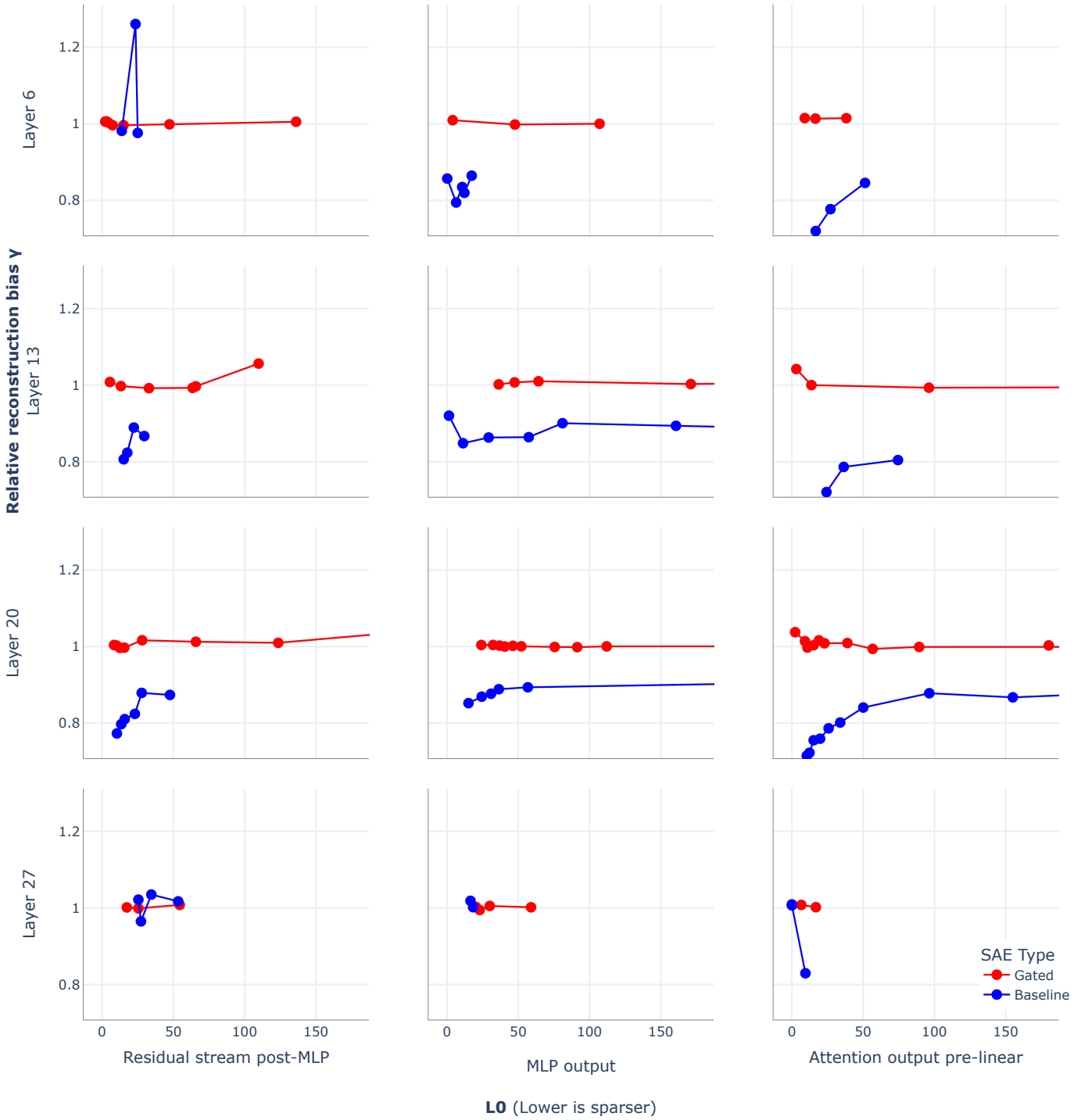


Figure 10: Gated SAEs address the problem of shrinkage in Pythia-2.8B.

on Gemma-7B sites at layer 27. We estimate that we used twice as much compute as used in the paper on preliminary experiments.

F.1.2 Experiment-specific training details

- We use learning rate 0.0003 for all Gated SAE experiments, and the GELU-1L baseline experiment. We swept for optimal baseline learning rates for the GELU-1L baseline to generate this value. For the Pythia-2.8B and Gemma-7B baseline SAE experiments, we divided the L2 loss by $\mathbb{E}||x||_2$, motivated by better hyperparameter transfer, and so changed learning rate to 0.001 and 0.00075. We didn't see noticeable difference in the Pareto frontier and so did not sweep this hyperparameter further.
- We generate activations from sequences of length 128 for GELU-1L, 2048 for Pythia-2.8B and 1024 for Gemma-7B.
- We use a batch size of 4096 for all runs. We use 300,000 training steps for GELU-1L and Gemma-7B runs, and 400,000 steps for Pythia-2.8B runs.

F.1.3 Lessons learned scaling SAEs

- **Learned feature death is unpredictable**. In Fig. 11 there are few patterns that can be gleaned from staring at which runs have high numbers of dead learned features (called dead neurons in Bricken et al. (2023)).
- **Resampling makes hyperparameter sweeps difficult**. We found that resampling caused L0 and loss recovered to increase, similar to Conmy (2023).
- **Training appears to converge earlier than expected**. We found that we did not need 20B tokens as in Bricken et al. (2023), as generally resampling had stopped causing gains and loss curves plateaued after just over one billion tokens.

F.2 Evaluation

We evaluated the models on over a million held-out tokens.

G Equivalence between gated encoder with tied weights and linear encoder with non-standard activation function

In this section we show under the weight sharing scheme defined in Eq. (6), a gated encoder as defined in Eq. (5) is equivalent to a linear layer with a non-standard (and parameterized) activation function.

Without loss of generality, consider the case of a single latent feature ($M = 1$) and set the pre-encoder bias to zero. In this case, the gated encoder is defined as

$$\tilde{f}(\mathbf{x}) := \mathbb{1}_{\mathbf{w}_{\text{gate}} \cdot \mathbf{x} + b_{\text{gate}} > 0} \operatorname{ReLU}\left(\mathbf{w}_{\text{mag}} \cdot \mathbf{x} + b_{\text{mag}}\right) \quad (14)$$

and the weight sharing scheme becomes

$$\mathbf{w}_{\text{mag}} := \rho_{\text{mag}} \mathbf{w}_{\text{gate}} \quad (15)$$

with a non-negative parameter $\rho_{\text{mag}} \equiv \exp(\mathbf{r}_{\text{mag}})$.

Substituting Eq. (15) into Eq. (14) and re-arranging, we can re-express $\tilde{f}(\mathbf{x})$ as a single linear layer

$$\tilde{f}(\mathbf{x}) := \sigma_{b_{\text{mag}} - \rho_{\text{mag}} b_{\text{gate}}}\left(\mathbf{w}_{\text{mag}} \cdot \mathbf{x} + b_{\text{mag}}\right) \quad (16)$$

with the parameterized activation function

$$\sigma_\theta(z) := \mathbb{1}_{z > \theta} \operatorname{ReLU}(z). \quad (17)$$

called JumpReLU in a different context (Erichson et al., 2019). Fig. 12 illustrates the shape of this activation function.

H Further analysis of the human interpretability study

We perform some further analysis on the data from Section 4.2, to understand the impact of different sites, layers, and raters.

H.1 Sites

We first pose the question of whether there's evidence that the sites had different interpretability outcomes. A Friedman test across sites shows significant differences (at $p = 0.047$) between the Gated-vs-Baseline differences, though not ($p = 0.92$) between the raw labels.

Breaking down by site and repeating the Wilcoxon-Pratt one-sided tests and computing confidence intervals, we find the result on MLP outputs is strongest, with mean 0.40, significance $p = 0.003$, and CI [0.18, 0.63]; this is as compared with the attention outputs ($p = 0.47$, mean .05, CI [-0.16, 0.26]) and final residual ($p = 0.59$, mean -0.07, CI [-0.28, 0.12]) SAEs.

H.2 Layers

Next we test whether different layers had different outcomes. We do this separately for the 2 models, since the layers aren't directly comparable. We run 2 tests in each setting: Page's trend test (which tests for a monotone trend across layers) and the Friedman test (which tests for any difference, without any expectation of a monotone trend).

Results are presented in Table 1; they suggest there are some significant nonmonotone differences between layers. To elucidate this, we present 90% BCa bootstrap confidence intervals of the mean raw label (where 'No'=0, 'Maybe'=1, 'Yes'=2) and the Gated-vs-Baseline difference, per layer, in Fig. 14 and Fig. 15, respectively.

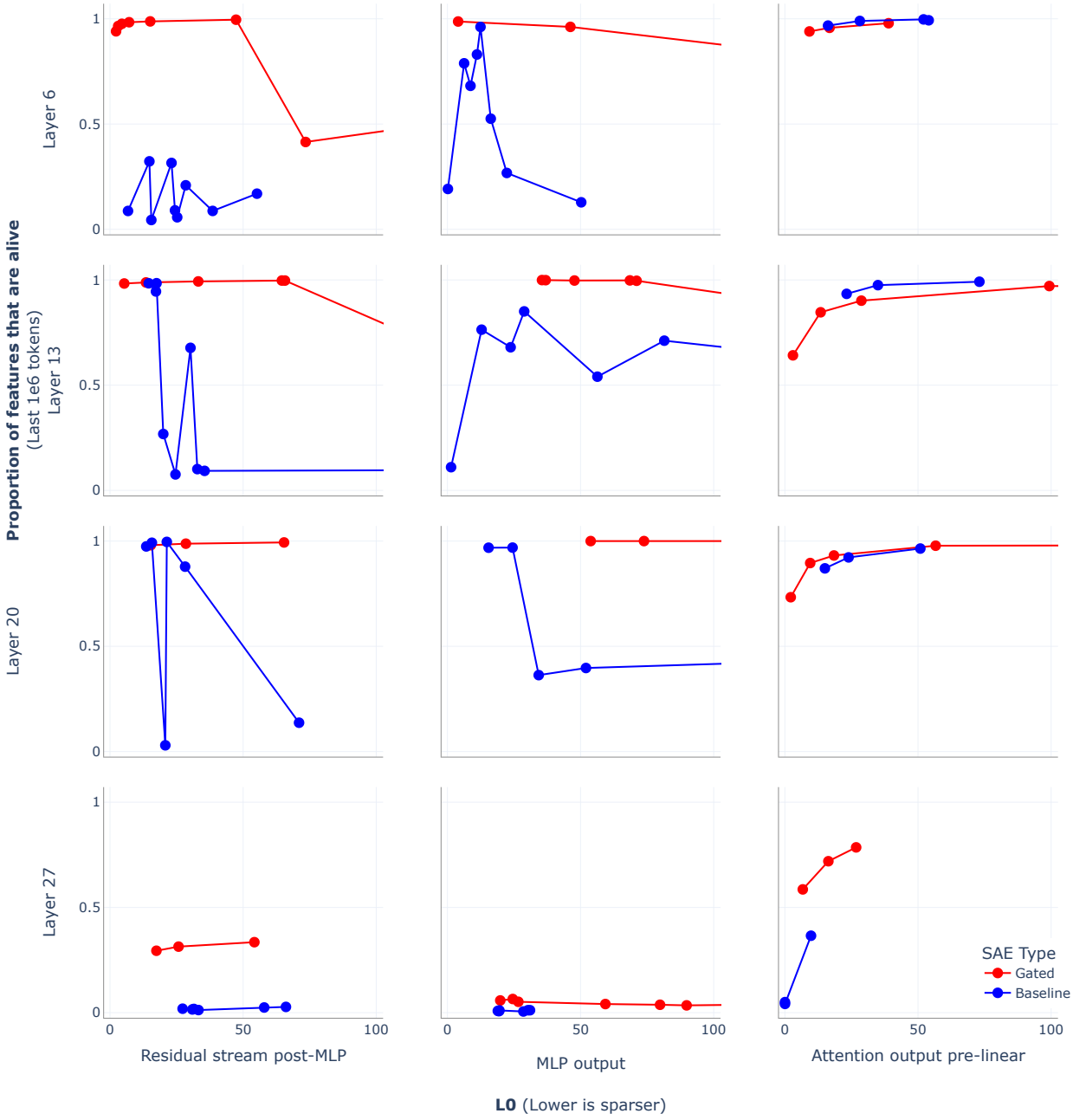


Figure 11: Feature death in Gemma-7B.

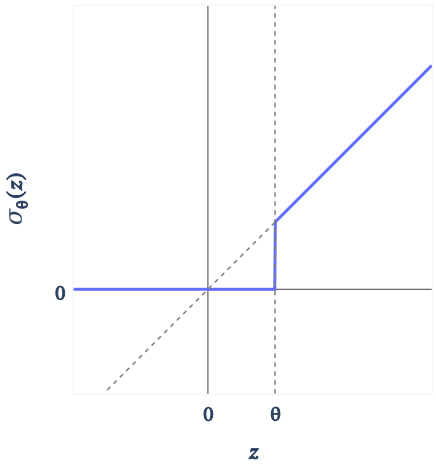


Figure 12: After applying the weight sharing scheme of Eq. (6), a gated encoder becomes equivalent to a single layer linear encoder with a Jump ReLU (Erichson et al., 2019) activation function σ_θ, illustrated above.

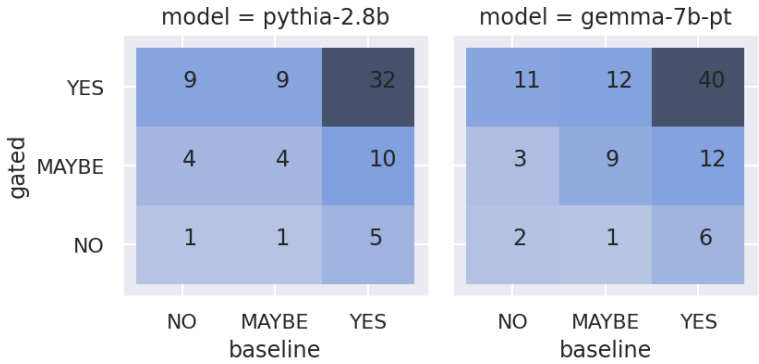


Figure 13: Contingency table showing Gated vs Baseline interpretability labels from our paired study results, for Pythia-2.8B and Gemma-7B.

p-values	Raw label	Delta from Baseline
Pythia-2.8B (Page's trend test)	0.50	0.13
Pythia-2.8B (Friedman test)	0.57	0.05
Gemma-7B (Page's trend test)	0.037	0.31
Gemma-7B (Friedman test)	0.003	0.64

Table 1: Layer significance tests

p-values	Raw label	Delta from Baseline
Across models (Kruskal-Wallis H-test)	0.01	0.71
Pythia-2.8B (Friedman test)	0.13	0.05
Gemma-7B (Friedman test)	0.03	0.76

Table 2: Rater significance tests

H.3 Raters

In Table 2 we present test results weakly suggesting that the raters differed in their judgments. This underscores that there's still a significant subjective component to this interpretability labeling. (Notably, different raters saw different proportions of Pythia vs Gemma features, so aggregating across the models is partially confounded by that.)

I Pseudo-code for Gated SAEs and the Gated SAE loss function

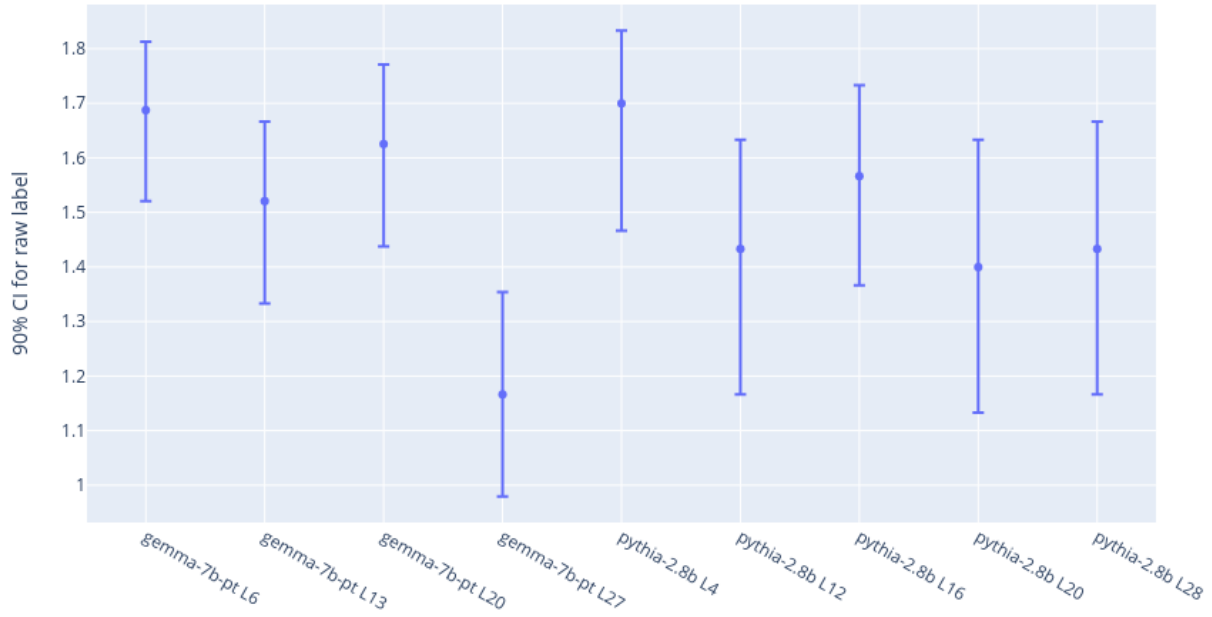


Figure 14: Per-layer 90% confidence intervals for the mean interpretability label

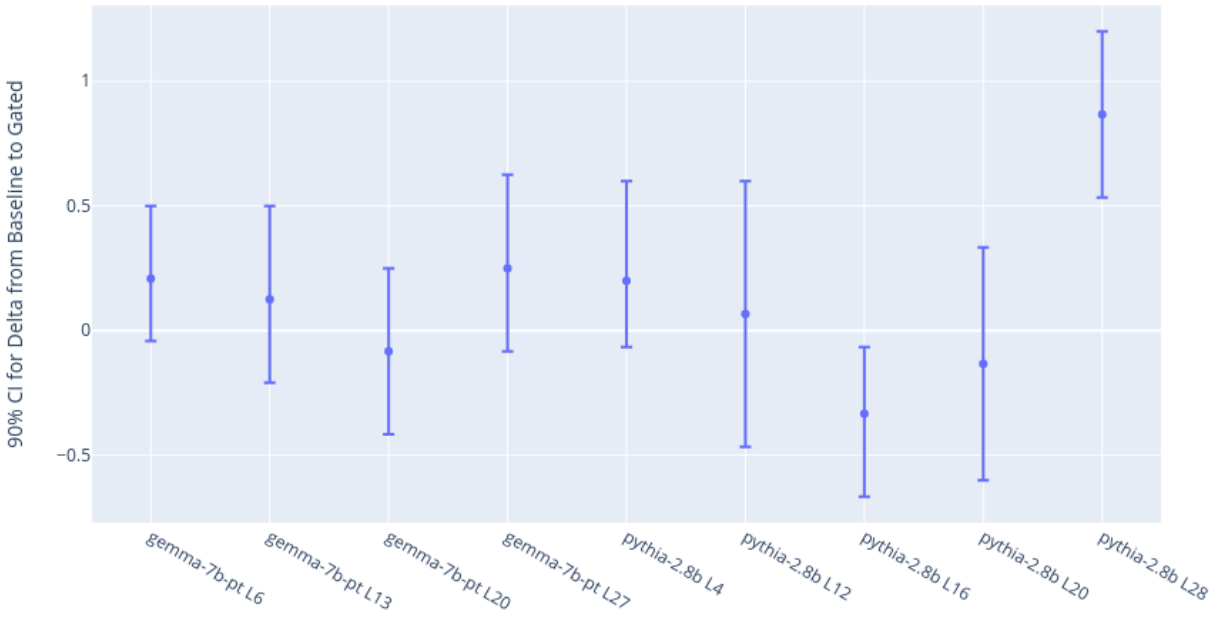


Figure 15: Per-layer 90% confidence intervals for the Gated-vs-Baseline label difference

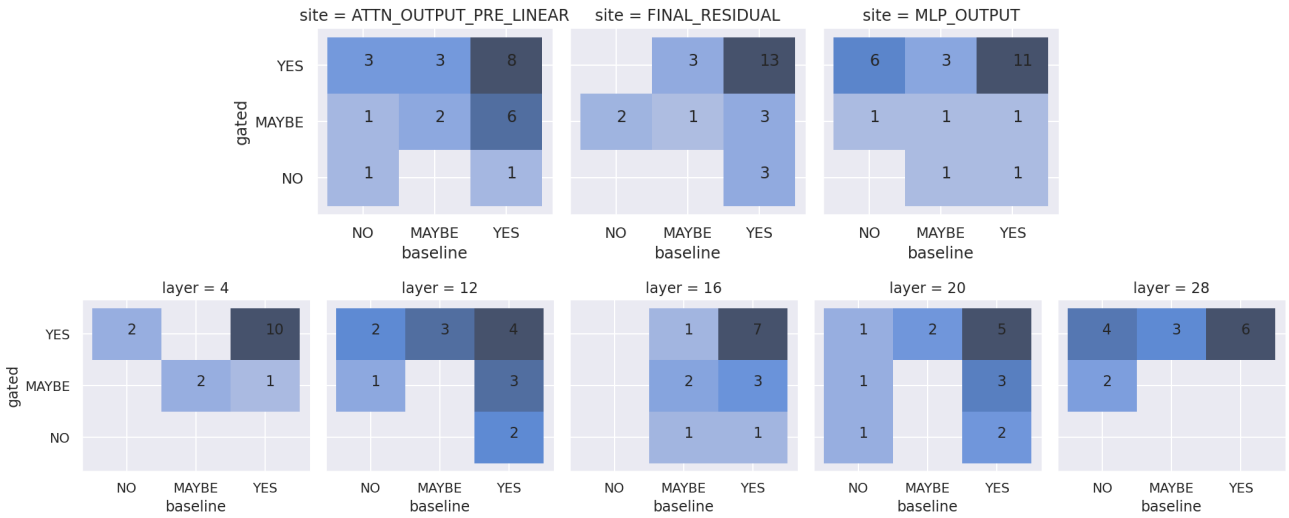


Figure 16: Contingency tables for the paired (gated vs baseline) interpretability labels, for Pythia-2.8B

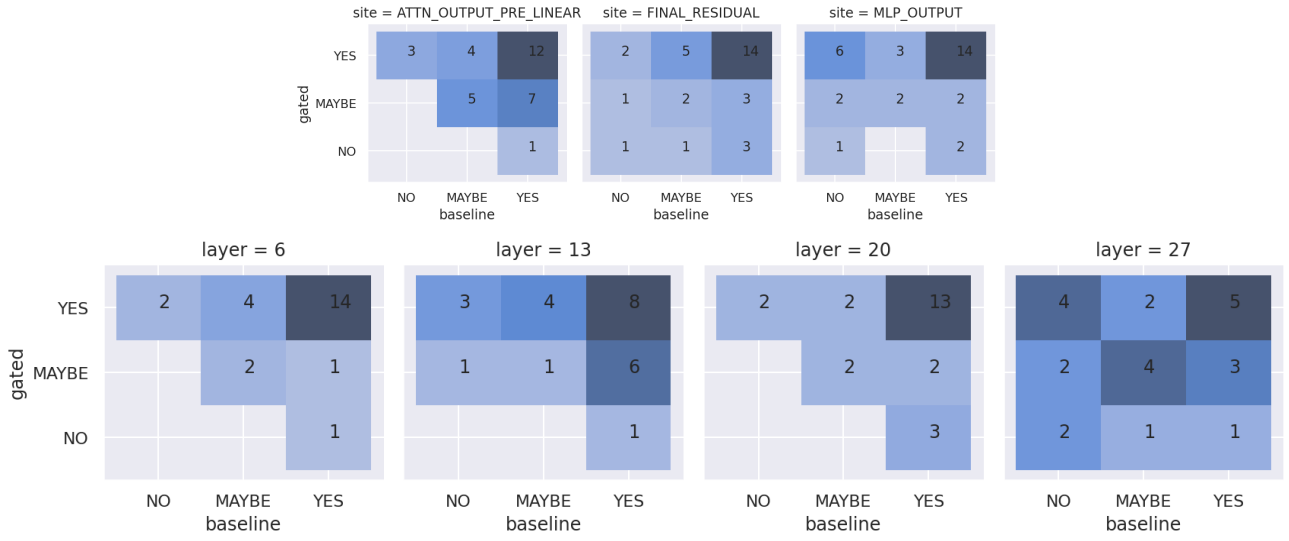


Figure 17: Contingency tables for the paired (gated vs baseline) interpretability labels, for Gemma-7B