Feature Hedging: Correlated Features Break Narrow Sparse Autoencoders

Anonymous Author(s)
Affiliation
Address
email

Abstract

It is assumed that sparse autoencoders (SAEs) decompose polysemantic activations into interpretable linear directions, as long as the activations are composed of sparse linear combinations of underlying features. However, we find that if an SAE is more narrow than the number of underlying "true features" on which it is trained, and there is correlation between features, the SAE will merge components of correlated features together, thus destroying monosemanticity. In LLM SAEs, these two conditions are almost certainly true. This phenomenon, which we call *feature hedging*, is caused by SAE reconstruction loss, and is more severe the narrower the SAE. In this work, we introduce the problem of feature hedging and study it both theoretically in toy models and empirically in SAEs trained on LLMs. We suspect that feature hedging may be one of the core reasons that SAEs consistently underperform supervised baselines. Finally, we use our understanding of feature hedging to propose an improved variant of matryoshka SAEs. Our work shows there remain fundamental issues with SAEs, but we are hopeful that that highlighting feature hedging will catalyze future advances that allow SAEs to achieve their full potential of interpreting LLMs at scale.

1 Introduction

As large language models (LLMs) are deployed in real-world applications, it is increasingly important to understand their internal workings. Sparse autoencoders (SAEs) decompose the dense, polysemantic activations of LLMs into interpretable latent features [6, 2] using sparse dictionary learning [19]. SAEs have the advantage of operating completely unsupervised, and can easily be scaled to millions of neurons in its hidden layer (hereafter called "latents") [22, 11].

While SAEs showed promising results, recent work has cast doubt on the performance of SAEs relative to baseline techniques. Wu et al. [24] show that SAEs underperform on both concept steering and detection relative to baselines, and Kantamneni et al. [13] show that SAEs underperform simple linear probes on both in-domain and out-of-domain detection, even when the probes have very few training samples. The question, then, is why do SAEs underperform relative to other techniques? And if we can identify the problems holding back SAEs, can we then fix those problems?

One fundamental issue with SAEs is the problem of feature absorption [5], where a more specific latent suppresses the firing a more general latent. For instance, an SAE may have a latent that appears to track "Cities in USA" but that arbitrarily fails to fire on the specific cities "New York" and "Detroit", where a city-specific latent fires instead. Feature absorption requires underlying features to exist in a hierarchy, with a parent feature f_p and a child feature f_c, where f_c can only fire if f_p is firing ($f_c \implies f_p$). Feature absorption is caused by SAE sparsity penalty, and becomes more severe the wider the SAE. An SAE encoder/decoder under feature absorption is shown in Figure 1b.

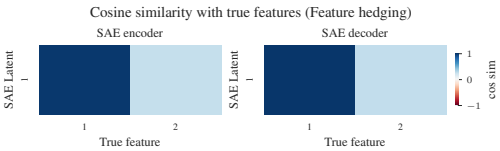
Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

Table 1: Comparing feature hedging and feature absorption

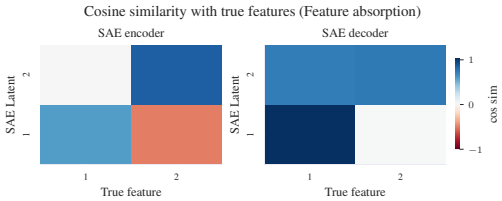
Feature absorption	Feature hedging
Learns gerrymandered latents	Mixes correlated features into latents
Caused by sparsity loss	Caused by MSE reconstruction loss
Features are all tracked in the SAE	One feature is in the SAE, the other is not
Affects the encoder and decoder asymmetrically	Affects encoder and decoder symmetrically
Gets worse the wider the SAE	Gets worse the narrower the SAE
Requires hierarchical features	Requires only correlation between features

In this paper, we identify another fundamental issue with SAEs which we call feature hedging. In hedging, an SAE is too narrow to represent both features f_a and f_b with their own latents l_a and l_b. Ideally, an SAE should assign a latent l to either f_a or f_b, and ignore the feature not being tracked. However, if f_a and f_b are either hierarchical as in absorption, or (anti-)correlated, then the SAE latent l can reduce reconstruction error by incorrectly mixing in components of both f_a and f_b. A sample SAE encoder and decoder experiencing hedging is shown in Figure 1a. In an LLM SAE, hedging will look like each SAE latent has noise mixed into it, likely reducing the performance of the latent for both detection and steering. Unlike with absorption, hedging becomes worse the narrower the SAE: thus trying to reduce absorption by making the SAE narrower will simply result in more hedging instead. The differences between feature hedging and feature absorption are shown in Table 1.

In LLM SAEs, the SAE is almost certainly narrower than the number of underlying features, as even extremely wide LLM SAEs appear to still miss features [22]. Furthermore, we should expect that nearly every feature in an LLM has some correlation to other features. We thus expect that hedging is the norm in SAEs trained on LLM activations and will distort the performance of LLM SAEs.



(a) When the SAE is only wide enough to represent one of the two features, we see feature hedging. Latent l_1 mainly tracks f_1, but a small component of f_2 is incorrectly mixed into the latent l_1 as well. f_2 is mixed symmetrically into both the encoder and decoder.



(b) If the SAE is wide enough to track both features, we see feature absorption. The decoder for l_1 perfectly tracks f_1, but its encoder turns off if f_2 is also active. l_2 tracks f_2, but its decoder merges together both f_1 and f_2. This asymmetry between encoder and decoder is characteristic of feature absorption.

Figure 1: SAE encoder and decoder patterns for hierarchical features f_1 and f_2, where $f_1 \implies f_2$. These features lead to either hedging or absorption depending on the width of the SAE.

A solution to feature absorption has been proposed in the form of matryoshka SAEs [4]. Matryoshka SAEs use nested SAE loss terms to enforce a hierarchy on the SAE latents, solving absorption by forcing the narrow inner levels of the SAE to reconstruct inputs on their own. However, as we show in this paper, matryoshka SAEs suffer more from hedging due to the inner matryoshka levels essentially being very narrow SAEs. Matryoshka SAEs thus trade off absorption for hedging.

In this work, we define and study feature hedging both theoretically in toy models and empirically in LLM SAEs. We show that hedging is worse the more narrow the SAE, and introduce a technique to characterize the amount of hedging present in a given SAE. We also study hedging and absorption in matryoshka SAEs, and show that it is possible to improve the monosemanticity of matryoshka SAEs by tuning the relative loss coefficients in each level of the matryoshka SAE to better balance the competing forces of absorption and hedging—though both problems remain present.

2 Background

Sparse autoencoders (SAEs). An SAE decomposes an input activation $a \in \mathbb{R}^D$ into a hidden state f consisting of L hidden neurons, called "latents". An SAE is composed of an encoder $W_{\text{enc}} \in \mathbb{R}^{L \times D}$, a decoder $W_{\text{dec}} \in \mathbb{R}^{D \times L}$, a decoder bias $b_{\text{dec}} \in \mathbb{R}^D$, and encoder bias $b_{\text{enc}} \in \mathbb{R}^L$, and a nonlinearity σ, typically ReLU or a variant like JumpReLU [20], TopK [11] or BatchTopK [3].

$$f = \sigma(W_{\text{enc}}(a - b_{\text{dec}}) + b_{\text{enc}}) \tag{1}$$

$$\hat{a} = W_{\text{dec}} f + b_{\text{dec}} \tag{2}$$

The SAE is trained with a reconstruction loss, typically Mean Squared Error (MSE), and a sparsity-inducing loss consisting of a function $\mathcal{S}$ that penalizes non-sparse representation with corresponding sparsity coefficient λ. For standard L1 SAEs, $\mathcal{S}$ is the L1 norm of f. For TopK and BatchTopK SAEs, there is no sparsity-inducing loss ($\mathcal{S} = 0$) as the TopK function directly induces sparsity. There is sometimes also an additional auxiliary loss $\mathcal{L}_{aux}$ with coefficient α to ensure all latents fire. Standard L1 SAEs typically do not have an auxiliary loss [18]. The general SAE loss is

$$\mathcal{L} = \|a - \hat{a}\|_2^2 + \lambda \mathcal{S} + \alpha \mathcal{L}_{\text{aux}}. \tag{3}$$

Tied SAEs. A tied SAE has $W_{\text{enc}} = W_{\text{dec}}^{\mathsf{T}}$. The biases have different dimensions and are untied.

Matryoshka SAEs. A matryoshka SAE [4] extends the SAE definition by summing losses created by prefixes of SAE latents. This forces each sub-SAE to reconstruct input activations on its own, and incentivizes the SAE to place more common, general concepts into latents with smaller index number. A matryoshka SAE uses nested prefixes with sizes $\mathcal{M} = m_1, m_2, ...m_n$ where $m_1 < m_2 < \ldots < m_n = L$, where L is the number of latents in the full dictionary. Matryoshka SAE loss is:

$$\mathcal{L} = \sum_{m \in \mathcal{M}} \left(\|a - \hat{a}_m\|_2^2 + \lambda \mathcal{S}_m \right) + \alpha \mathcal{L}_{\text{aux}} \tag{4}$$

Where $\hat{a}_m$ is the reconstruction for the SAE using the first m latents, and $\mathcal{S}_m$ is the sparsity penalty applied to the first m latents. For TopK and BatchTopK Matryoshka SAEs, there is no sparsity penalty ($\mathcal{S}_m = 0$) as the TopK function directly imposes sparsity.

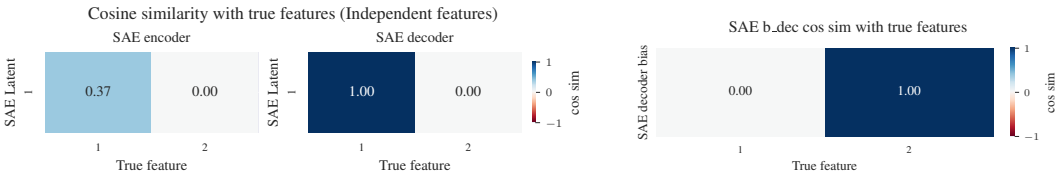
3 Studying hedging in single-latent SAEs

We begin by investigating hedging in the simplest possible toy SAE setting: an SAE with a single latent. We use a model with two true features f_1 and f_2. Each true feature f is a random direction with unit-norm in $\mathbb{R}^{50}$, and $f_1 \perp f_2$. Each feature fires with magnitude 1.0. Since we only have two features, an activation a can consist of $a \in \{0, f_1, f_2, f_1 + f_2\}$. There is no bias term added to the activations. Unless otherwise specified, f_1 fires with probability 0.25, and f_2 fires with probability 0.2. We use SAELens [1] to train a single-latent SAE on these activations.

3.1 Fully independent features

We first study the case when f_1 and f_2 fire independently. We find that the SAE correctly represents f_1 without any interference from f_2. However, the decoder bias has incorrectly learned to represent the direction of f_2, but with magnitude 0.2, equal to the probability of f_2 firing. The cosine similarities of the single SAE latent and SAE bias term with the true features is shown in Figure 2.

We consistently find this pattern of the decoder bias merging in positive components of features not tracked by their own latent. In this sense, the decoder bias can be thought of as tracking an always-on feature, and thus is in a hierarchical relationship with every other feature of the model.



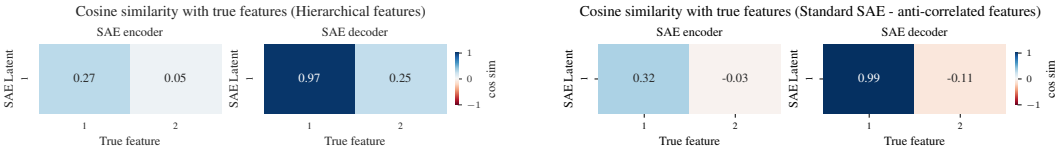
(a) The SAE encoder and decoder both correctly learn just f_1.

(b) The decoder bias incorrectly learns f_2.

Figure 2: Encoder, decoder, and decoder bias patterns for a toy model with 2 independent features.

3.2 Hierarchical features

Next, we investigate what happens if f_1 and f_2 are in a hierarchy, so f_2 can only fire if f_1 fires, but f_1 can still fire on its own ($f_2 \implies f_1$). We adjust the firing probability of f_2 so that $P(f_2|f_1) = 0.2$, and $P(f_2|\neg f_1) = 0$ (thus, $P(f_2) = 0.05$). In a two-latent SAE this setup would cause feature absorption. We plot the cosine similarities of our single latent with f_1 and f_2 in Figure 3a.



(a) When features f_1 and f_2 form a hierarchy ($f_2 \implies f_1$), the SAE incorrectly merges a component of f_2 into its single latent l_1.

(b) When features f_1 and f_2 are anti-correlated, the SAE incorrectly merges a *negative* component of f_2 into its single latent l_1.

Figure 3: Hedging occurs with hierarchical features or anti-correlated features.

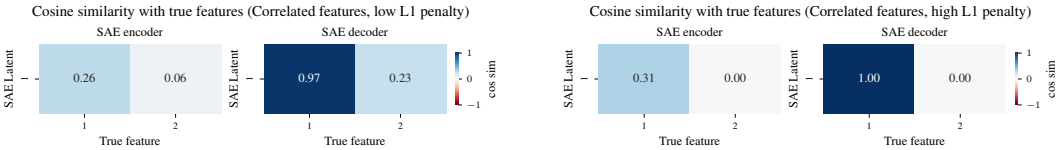
Here we clearly see feature hedging. The single SAE latent has now merged in a component of f_2 into its single latent, so it's now a mixture of f_1 and f_2. f_2 is merged roughly symmetrically into both the encoder and decoder of the SAE latent ($\cos(f_2, l_1)$ is about ¼ of $\cos(f_1, l_1)$ in both encoder and decoder). This is unlike in feature absorption where there is an asymmetry in the encoder and decoder. This merging of features reduces the MSE loss of the SAE despite being a degenerate solution.

Increasing the L1 penalty of the SAE does not solve this problem. f_2 only fires if f_1 fires, so adding a positive component of f_2 into the encoder does not cause the latent to fire any more often.

3.3 Positively correlated features

Next, we change our setup so that $P(f_2|\neg f_1) = 0.1$ instead of 0. We still keep $P(f_2|f_1) = 0.2$, so that f_2 is more likely to fire if f_1 fires, but it can still fire on its own as well. The features are now merely correlated rather than following a strict hierarchy.

We now see hedging depending on the strength of the L1 penalty. When the L1 penalty is low, hedging is apparent. However, if the L1 penalty is high enough and the level of correlation is low enough, then the SAE will learn the correct features, as positive hedging increases the L0 of the SAE slightly relative to learning just f_1. Plots of the cosine similarity of the SAE encoder and decoder compared to true features are shown in Figure 4 with high and low sparsity penalties. If we use a full-width SAE, the SAE learns the true features despite the correlation (see Appendix A.1).



(a) Hedging still occurs with L1 coefficient of 0.001.

(b) No hedging occurs with L1 coefficient of 0.1.

Figure 4: Hedging occurs with positively-correlated features depending on the sparsity penalty.

3.4 Anti-correlated features

Next, we reverse the conditional probabilities of f_2 so that $P(f_2|f_1) = 0.1$ and $P(f_2|\neg f_1) = 0.2$. Now f_2 is more likely to fire on its own than it is to fire along with f_1. A plot of the cosine similarity of the SAE with the true features is shown in Figure 3b.

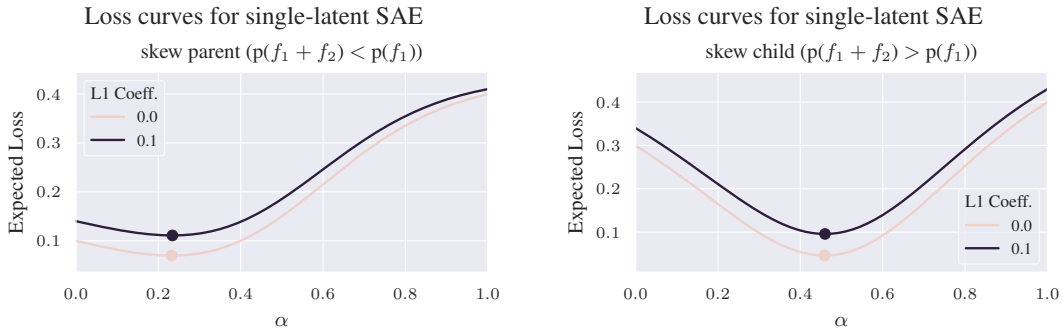
Now the SAE latent has actually merged a negative component of f_2 into its single latent instead of a positive component. Furthermore, increasing L1 penalty does nothing to solve this, as the negative component of hedging in the encoder does not increase L0 of the SAE. If we use a full-width SAE, we again see the SAE learns the true features despite the correlation (see Appendix A.1).

3.5 Hedging is caused by reconstruction loss: curves for single-latent SAEs

What causes hedging? We hypothesize that it is a combination of not enough latents to represent every feature, and the fact that MSE loss incentivizes reconstructing multiple features imperfectly as opposed to only one feature perfectly.

To test this, we analyze the loss curves for a single-latent tied SAE with a parent-child relationship between the two features f_1 and f_2, so $f_2 \implies f_1$. The ideal SAE latent must be some combination of these two features. As there are no other interfering features to break the symmetry between encoder and decoder, the SAE can be expressed by a single unit norm latent. We set the SAE latent l to an interpolation of these two features, $l = \alpha f_2 + (1 - \alpha) f_1$ (adjusted to have unit norm). We calculate expected SAE loss consisting of MSE + L1 loss for $0 \leq \alpha \leq 1$.

First, we set $P(a = f_1) = 0.3$ and $P(a = f_1 + f_2) = 0.1$. We characterize the probabilities this way since there are only two firing possibilities we need to consider: either f_1 is firing on its own or f_1 and f_2 are firing together. We use L1 coefficient of 0 and 0.1 to explore the effect of the sparsity penalty on loss. We also consider the case where both features fire together more than they fire on their own, with $P(a = f_1) = 0.1$ and $P(a = f_1 + f_2) = 0.3$. Loss curves are shown in Figure 5.



(a) Loss curves when the parent feature f_1 fires more on its own than with child feature f_2. Loss is minimized between f_1 and f_2 rather than at f_1 ($\alpha = 0$). Sparsity penalty does not change the minimum.

(b) Loss curves when the parent feature f_1 fires less on its own than it does with the child feature f_2. Loss is incorrectly minimized between f_1 and f_2. Sparsity penalty does not change the minimum.

Figure 5: Loss curves for an SAE with a single latent l and 2 hierarchical features, where $f_2 \implies f_1$. The minimum loss is indicated with a dot on each plot. $\alpha = 0$ means that $l = f_1$, and $\alpha = 1$ means $l = f_2$. In all cases, loss is minimized when the latent l is a combination of f_1 and f_2.

In these plots, $\alpha = 0$ corresponds to the SAE latent being exactly f_1, and $\alpha = 1$ corresponds to the latent being f_2, and $\alpha = 0.5$ corresponds to $f_1 + f_2$. We clearly see that the SAE loss has a single minimum between f_1 and $f_1 + f_2$, showing that the MSE minimum is attained with feature hedging.

4 Quantifying hedging in LLM SAEs

While we have demonstrated hedging in a synthetic setting, it remains a question how much hedging occurs in LLM SAEs. We next study the effect of adding new latents to an existing SAE. Based on our understanding of hedging in toy models, we expect that when a new latent is added to an SAE, this should pull the component of the new feature out of existing SAE latents. Thus if hedging

occurs, the change in existing latents after a new latent is added should project onto that new latent. If hedging did not exist, then adding a new latent should not have any effect on existing latents.

Hedging affects the encoder and decoder of the SAE symmetrically, so we should be able to detect hedging in either the encoder or decoder. We look at the decoder to distinguish hedging from absorption, as absorption affects the encoder. Under feature absorption, if a newly added latent is a child feature of an existing latent, then the encoder for the parent latent adds a negative component of the new child latent to avoid firing when the child is active, but the parent decoder remains unchanged. This corresponds to adding a new latent to Figure 1a and arriving at Figure 1b. Thus, any change to existing decoder latents cannot be attributed to absorption and must be due to hedging.

We expect that even if there were no hedging at all, simply due to noise, existing SAE decoder latents may undergo a change that has some small projection onto new added latents. We want to make sure that anything we quantify as hedging must be larger than what we would expect from random noise.

Hedging degree Taking this into account, we define a metric called hedging degree, h. We take an existing SAE s_0 with L latents and add N new latents to the SAE. After adding these latents, we continue training the SAE and arrive at a new SAE, s_1, with $L + N$ latents. We also continue training s_0 on the same tokens that we train s_1 on to ensure that any difference between s_0 and s_1 is due only to the newly added latents. W^0_{dec} refers to the new decoder of s_0, and W^1_{dec} refers to the decoder of s_1. We define the difference in the original L latents between s_0 and s_1 as:

$$\delta_L = W^1_{\text{dec}}[0:L] - W^0_{\text{dec}}[0:L] \tag{5}$$

where $W^1_{\text{dec}}[L : L+N]$ refers to the newly added decoder latents. $W_{\text{rand}}[0:N]$ refers to a decoder consisting of N randomly initialized latents. All decoders are normalized to have latents of unit norm. We define the projection of a vector v onto a subspace spanned by W as:

$$\text{Proj}(v, W) = \|W(W^T W)^{-1} W^T v\| \tag{6}$$

The hedging degree h is then defined as:

$$h = \frac{1}{L} \sum_i^L \underbrace{\|\text{Proj}(\delta_L[i], W^1_{\text{dec}}[L : L+N])\|}_{\text{Projection of } \delta_L \text{ onto N new latents}} - \underbrace{\|\text{Proj}(\delta_L[i], W_{\text{rand}}[0:N])\|}_{\text{Projection of } \delta_L \text{ onto N random latents}} \tag{7}$$

Any value of $h > 0$ corresponds to hedging above what we would expect from random noise, as h subtracts the projection along N randomly initialized latents as part of the computation.

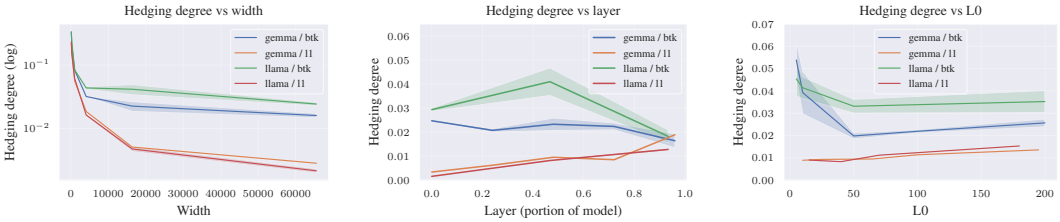
The choice of the number of new latents N is a hyperparameter of hedging degree. We use $N = 64$ for our hedging degree calculation. We explore the effect of different choices on N in Appendix A.3.

4.1 Results

We experiment with SAEs trained on Gemma-2-2b [21], as this model is commonly used for SAE research due to the thoroughness of the Gemma Scope suite of SAEs [15], as well as Llama-3.2-1b [7] to validate results on another LLM. All SAEs are trained first on 250M tokens of the Pile uncopyrighted [10]. After adding $N = 64$ latents, we continue training for another 250M tokens. The version of the SAE without latents added is also trained for another 250M tokens, so each SAE is trained for 500M tokens total. The pair of extended and non-extended SAEs is used to calculate hedging degree. SAE training details are in Appendix A.2.

We first calculate hedging degree vs SAE width in Figure 6a, with widths ranging from 128 to 65536. Hedging degree is dramatically higher at narrower widths, especially at 4096 width and below. While the hedging rate drops a lot with increasing SAE width, even at our max width of 65536 no SAE achieves 0 hedging degree, indicating there is still hedging occurring.

We next calculate hedging degree vs L0 (the average number of active latents) in Figure 6c, with L0 ranging from about 5 to 200. Very low L0 seems to lead to more hedging for BatchTopK SAEs, but the effect is minor compared with the effect of SAE width on hedging degree.



(a) Hedging degree vs width. No SAE tested reached 0 hedging.

(b) Hedging degree vs layer, normalized by number of LLM layers.

(c) Hedging degree vs L0.

Figure 6: Hedging degree for SAEs trained on Gemma-2-2b layer 12. Unless otherwise specified, SAEs have width 8192, BatchTopK SAEs have K=25. Shaded area in plots is 1 std.

Finally, we calculate hedging degree vs layer in Figure 6b. The hedging degree for L1 and TopK SAEs appears to merge around the end of the SAE, but overall the layer does not appear to have a massive effect on hedging degree.

It also appears that BatchTopK SAEs have more hedging than L1 SAEs. This may be due to L1 loss reducing hedging from positively correlated features, as we saw in Section 3.3.

5 Case study: adding a new latent to an existing SAE

We next explore how hedging affects a real SAE. We trained a L1 SAE on Gemma-2-2b layer 12 with width 8192 for 250M tokens on the Pile [10], then add a new latent to the SAE, and continue training both the original SAE and the extended SAE for another 250M tokens.

```
0.3/ css / bootstrap.min .
/ bootstrap.min . css _integrity="sha3"
/ bootstrap.min . css ">____ link
```

(a) Newly added case-study latent, latent 8192. The latent appears to track CSS scripts in HTML.

```
>____< link _rel =" stylesheet" _type ="
8"/>< link _rel = stylesheet_href="..//doc
png"><link _rel =" manifest"_href="
```

(b) Latent 3094, which had the largest negative δ-projection after adding latent 8192. This latent tracks "rel" in HTML, used for CSS scripts in HTML.

Figure 7: Sample top activating examples for case study latents.

We examine inputs that cause the newly added latent to fire to get a sense of what it represents. We reproduce a portion of the top activating examples for the new latent in Figure 7a. This latent appears to fire on CSS scripts included in HTML. A larger set of inputs is shown in Appendix A.4.

Next, we look at the magnitude of change in existing latents projected on the new latent. Based on our understanding of hedging, if a latent loses a large component of the newly added latent, this corresponds to a likely hierarchical relationship with the new latent. The latent which lost the largest component of the new latent is latent 3094, which seems to track the "rel" HTML attribute used mainly for linking CSS scripts. We show top activating examples for latent 3094 in Figure 7b.

Since CSS scripts are just one type of asset that can be linked using "rel", this appears to be exactly the sort of hierarchical relationship we expect to be heavily impacted by hedging.

6 Balancing hedging and absorption in matryoshka SAEs

Matryoshka SAEs [4] combat absorption with nested SAE loss prefixes. Each level acts like a small SAE, and is forced to reconstruct the input on its own. This forces the SAE to learn more general concepts in earlier levels, and makes it difficult for the SAE to make holes in the recall of parent latents for absorption, as this would hurt the reconstruction of earlier matryoshka levels.

However, since early matryoshka levels are effectively narrow SAEs, they suffer from feature hedging. As we saw in Section 4.1, the more narrow an SAE is, the more the severe the feature hedging. Matryoshka SAEs thus solve feature absorption at the expense of exacerbating feature hedging.

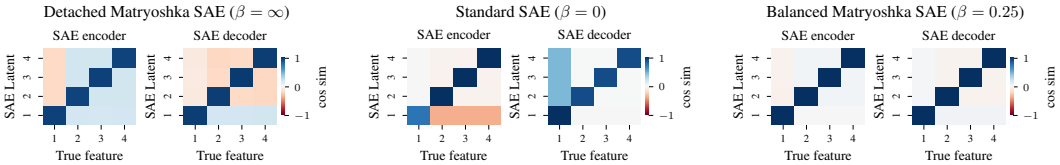
Inspecting the effect of hedging and absorption on the SAE encoder in Figure 1b shows that hedging and absorption have opposite effects. For hierarchical features, hedging adds a positive component of child features into the parent encoder latent, but absorption does the opposite and adds a negative component of child features into the parent latent. If we balance the negative component of child latents from absorption with the positive component from hedging, these effects can cancel out.

Balance matryoshka SAE We extend the definition of a matryoshka SAE from Equation 4 to allow applying a scaling coefficient β_m to the loss for each matryoshka level:

$$\mathcal{L} = \sum_{m \in \mathcal{M}} \beta_m \left(\|a - \hat{a}_m\|_2^2 + \lambda \mathcal{S}_m \right) + \alpha \mathcal{L}_{\text{aux}} \tag{8}$$

We refer to this extension as a *balance matryoshka SAE*, where each $\beta_m \geq 0$ controls the relative balance of each level. If each $\beta_m = 1$ this is a standard matryoshka SAE. If $\beta_m = 0$ for all matryoshka levels except the outer-most level, this reduces to a standard (non-matryoshka) SAE.

We demonstrate this balancing in a toy model of hierarchical features. The toy model has 4 features, with feature 1 being the parent feature and features 2-4 being children (features 2-4 can only fire if feature 1 is also firing). Feature 1 fires with probability 0.25, and each child feature fires with probability 0.15 if feature 1 is firing. We train a matryoshka SAE with a single inner level consisting of only latent 1 with balance coefficient β. For more details on this toy setup, see Appendix A.5.



(a) Matryoshka SAE with detached loss (equivalent a matryoshka SAE with $\beta = \infty$). Hedging adds positive components of the child features 2-4 to the encoder of latent 1.

(b) Standard SAE (equivalent a matryoshka SAE with $\beta = 0$). Absorption adds negative components of the child features 2-4 to the encoder of latent 1.

(c) Roughly balanced matryoshka SAE with $\beta = 0.25$. The positive and negative contributions hedging and absorption roughly cancel out, leaving a nearly perfect SAE.

Figure 8: Balancing hedging and absorption in a toy model of hierarchical features. Child features 2-4 only fire if parent feature 1 fires. The matryoshka SAE has a single inner level with 1 latent.

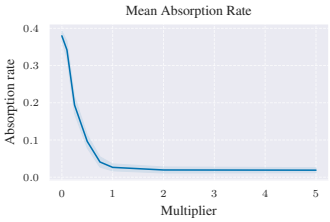
We show results in Figure 8. When β is too high or too low this results in hedging or absorption, respectively. When $\beta = 0.25$, these balance out and the SAE learns a near perfect representation.

Next, we train LLM balance matryoshka SAEs with different balance ratios on Gemma-2-2b layer 12. The SAEs are BatchTopK with k=40, trained on 500M tokens. The SAEs have 5 matryoshka levels of sizes 128, 512, 2048, 8192, and 32768 (so the full SAE has width 32768). We set the outermost $\beta_5 = 1$, and set a constant multiplier between each subsequent β_m, so multiplier $= \beta_m / \beta_{m+1}$. If the multiplier is 0.5, then $\beta_m = 0.5^{(5-m)}$.

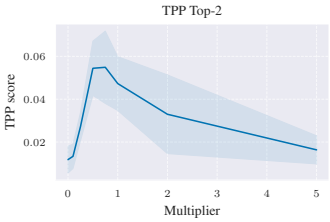
We train 10 seeds for each multiplier and show results in Figure 9 for absorption rate, targeted probe pertubation (TPP), Spurious Concept Removal (SCR), K-sparse probing, and feature-splitting metrics from SAEBench [14], and k=1 sparse probing results [12] for a Parts of Speech (POS) dataset we created using Treebank POS tagged sentences [16]. We add a POS dataset for probing since POS are very general concepts, and should be learned in the earliest levels of a matryoshka SAE.

For TPP, feature splitting, and sparse probing, using a compound multiplier of around 0.75 achieves better results than either a standard matryoshka SAE or a standard (non-matryoshka) SAE, providing evidence that balancing matryoshka losses can improve the performance. Using a multiplier of 0.75 still scores well on the absorption metric as well. Strangely, the SCR metric appears to perform better at higher multipliers. SCR and TPP should measure the same thing, so we do not fully understand the discrepancy between these metrics. We provide further results and more details in Appendix A.7.

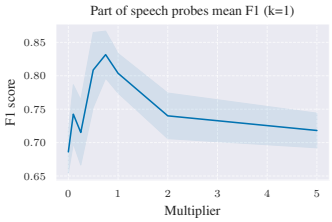
While balancing each β_m can improve performance on most metrics, we do not expect this to perfectly solve absorption and hedging. We show in Appendix A.6 that balancing all hedging and absorption



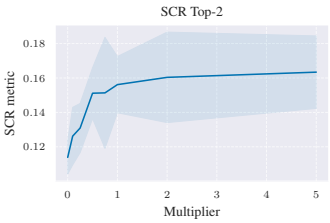
(a) SAEBench Absorption rate. Lower is better.



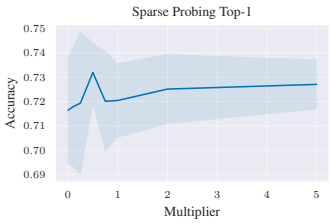
(b) SAEBench TPP metric. Higher is better.



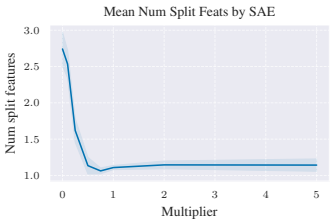
(c) k=1 sparse probing F1 score on Parts of Speech (POS).



(d) SAEBench SCR top-2 metric. Higher is better.



(e) SAEBench K=1 sparse probing accuracy.



(f) Feature splitting (SAEBench absorption). Lower is better.

Figure 9: Performance of balance matryoshka SAEs vs multiplier. The shaded area is 1 std. Multiplier=0 is equivalent to a standard SAE, and multiplier=1 is a standard matryoshka SAE.

with a single β_m is not always possible. We thus expect that it may be possible to further improve SAE performance by learning different balancing coefficients per latent, but this is left to future work.

7 Related work

Other work has highlighted theoretical problems with SAEs. Till [23] investigated a problem where SAEs may increase sparsity by inventing features. For instance, an SAE may fabricate a "red triangle" feature in addition to "red" and "triangle" features. Templeton et al. [22] dicuss the problem of feature splitting, where an SAE may not learn features at a desired level of specificity. Engels et al. [8] investigates SAE errors and finds that SAE error may be pathological and non-linear. Engels et al. [9] further shows that there are features that cannot be expressed as a simple linear direction, and thus SAEs may struggle to represent these features. Wu et al. [24] and Kantamneni et al. [13] both investigate the empirical performance of SAEs and find that SAEs underperform baselines.

8 Discussion

SAEs remain a promising technique for decomposing the residual stream of LLMs in an unsupervised manner. However, given recent work showing that SAEs underperform relative to baselines [24, 13], it is imperative that we understand the reasons for this underperformance so they can be addressed.

In this work, we introduced the problem of feature hedging in SAEs, showing it both theoretically in toy models, and empirically in SAEs trained on real LLMs. We suspect that hedging, along with absorption, may be one of the core theoretical problems leading to poor SAE performance.

Using our understanding of hedging, we introduced the balance matryoshka SAE architecture, allowing balancing of hedging and absorption against each other, improving interpretability. We view balance matryoshka SAEs as a starting point, and expect this architecture can be improved by optimizing the balance coefficients. There may not be a single coefficient that perfectly balances hedging and absorption for all features, so we expect there may be further gains from learning a different balancing coefficients per latent in the SAE. We leave these improvements to future work.

9 Limitations

We only test hedging in SAEs up to 65k latents on LLMs with 2b parameters due to compute constraints. Our method for detecting hedging requires fine-tuning SAEs, which is expensive.

References

[1] Joseph Bloom, Curt Tigges, Anthony Duong, and David Chanin. Saelens. https://github.com/jbloomAus/SAELens, 2024.

[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] Bart Bussmann, Patrick Leask, and Neel Nanda. Batchtopk sparse autoencoders. *arXiv preprint arXiv:2412.06410*, 2024.

[4] Bart Bussmann, Noa Nabeshima, Adam Karvonen, and Neel Nanda. Learning multi-level features with matryoshka sparse autoencoders. *arXiv preprint arXiv:2503.17547*, 2025.

[5] David Chanin, James Wilken-Smith, Tomáš Dulka, Hardik Bhatnagar, and Joseph Bloom. A is for absorption: Studying feature splitting and absorption in sparse autoencoders. *arXiv preprint arXiv:2409.14507*, 2024.

[6] Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=F76bwRSLeK.

[7] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen,

Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vítor Albiero, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. The llama 3 herd of models, 2024. URL `https://arxiv.org/abs/2407.21783`.

[8] Joshua Engels, Logan Riggs, and Max Tegmark. Decomposing the dark matter of sparse autoencoders. *arXiv preprint arXiv:2410.14670*, 2024.

[9] Joshua Engels, Eric J Michaud, Isaac Liao, Wes Gurnee, and Max Tegmark. Not all language model features are one-dimensionally linear. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=d63a4AM4hb.

[10] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[11] Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

[12] Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. Finding neurons in a haystack: Case studies with sparse probing. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=JYs1R9IMJr.

[13] Subhash Kantamneni, Joshua Engels, Senthooran Rajamanoharan, Max Tegmark, and Neel Nanda. Are sparse autoencoders useful? a case study in sparse probing. *arXiv preprint arXiv:2502.16681*, 2025.

[14] Adam Karvonen, Can Rager, Johnny Lin, Curt Tigges, Joseph Bloom, David Chanin, Yeu-Tong Lau, Eoin Farrell, Callum McDougall, Kola Ayonrinde, Matthew Wearden, Arthur Conmy, Samuel Marks, and Neel Nanda. Saebench: A comprehensive benchmark for sparse autoencoders in language model interpretability, 2025. URL https://arxiv.org/abs/2503.09532.

[15] 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, August 2024.

[16] Mitch Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[17] Samuel Marks, Can Rager, Eric J Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=I4e82CIDxv.

[18] Chris Olah, Adly Templeton, Trenton Bricken, and Adam Jermyn. April update. https://transformer-circuits.pub/2024/april-update/index.html, 2024. URL https://transformer-circuits.pub/2024/april-update/index.html.

[19] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997.

[20] Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders. *arXiv preprint arXiv:2407.14435*, 2024.

[21] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozińska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucińska,

Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Görner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nenshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Cogan, Sarah Perrin, Sébastien M. R. Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomas Kocisky, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. Gemma 2: Improving open language models at a practical size, 2024. URL `https://arxiv.org/abs/2408.00118`.

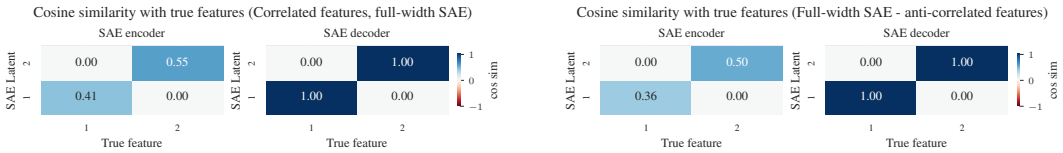
[22] Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, Alex Tamkin, Esin Durmus, Tristan Hume, Francesco Mosconi, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. `https://transformer-circuits.pub/2024/scaling-monosemanticity/`, May 2024. Accessed on May 21, 2024.

[23] Demian Till. Do sparse autoencoders find true features? *LessWrong*, 2024. URL `https://www.lesswrong.com/posts/QoR8noAB3Mp2KBA4B/do-sparse-autoencoders-find-true-features`.

[24] Zhengxuan Wu, Aryaman Arora, Atticus Geiger, Zheng Wang, Jing Huang, Dan Jurafsky, Christopher D Manning, and Christopher Potts. Axbench: Steering llms? even simple baselines outperform sparse autoencoders. *arXiv preprint arXiv:2501.17148*, 2025.

A Technical Appendices and Supplementary Material

A.1 Full-width SAE toy model results



(a) Full-width SAE with correlated features. The SAE is still able to perfectly learn the underlying features despite the correlation.

(b) Full-width SAE with anti-correlated features. The SAE is still able to perfectly learn the underlying features despite the correlation.

Figure 10: Full-width SAE results on correlated and anti-correlated toy models.

We extend the discussion of single-latent SAEs to explore what happens if the SAE has two latents, the same number of latents as the number of true features. We use the same toy model as in Section 3.3 for the positive correlation case, and the same toy model as in Section 3.4 for the anti-correlated case. We use L1 penalty of 1e-3 for the positive correlation case, the same as the L1 penalty that caused hedging in single-latent SAEs.

We plot the results in Figure 10. In both cases, the full-width SAEs are able to perfectly recover the true features despite the correlation, and despite the low L1 penalty. This shows that hedging is caused by the SAE being too narrow, as increasing the width of the SAE solves the problem.

A.2 Training details for LLM SAEs

All SAEs are trained on the Pile uncopyrighted [10], using a batch size of 4096 activations and context length of 1024 tokens. For L1 SAEs, we use a linear L1 warm-up of 10k steps. SAEs are trained on a single 80gb Nvidia H100 GPU. Model weights are loaded in fp32 precisions, but autocast to bfloat16 during training. We initialize the SAE so that the encoder and decoder are identical, where each latent has norm 0.1, following the procedure described in [18]. All L1 SAEs are trained with learning rate 7e-5, and BatchTopK SAEs are trained with learning rate 3e-4. SAEs are trained using SAELens [1].

Unless otherwise specified, BatchTopK SAEs use k=25. For SAEs trained on Gemma-2-2b, we conduct most experiments at layer 12 (roughly in the middle), and L1 SAEs trained on Gemma-2-2b use L1 coefficient of 10. This coefficient does not reuslt in dead extension latents, and yields a L0 around 50. For SAEs trained on Llama-3.2-1b, we conduct most experiments at layer 7 (roughly in the middle of the model), and for L1 SAEs trained on Llama-3.2-1b, we use L1 coefficient of 0.5. This coefficient does not result in dead extension latents, and yields a L0 around 50.

A.3 Choice of hedging hyperparameter N

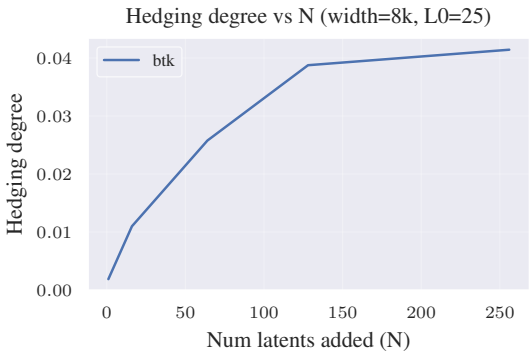


Figure 11: Hedging degree vs N

Our hedging degree metric requires adding N new latents onto an existing SAE to extend it, naturally leading to the question of what is a reasonable choice of N. We plot hedging degree vs N for Gemma-2-2b layer 12, given an initial BatchTopK SAE of width 8192 in Figure 11. We find that hedging degree increases until about N=250. We choose N=64 for our experiments, as 64 is still a small number of latents relative to the size of the residual stream (2304 for Gemma-2-2b), while still being large enough to hopefully reduce noise from any specific latent that gets added. Furthermore, as we see in the plot, the hedging degree from N=64 is about in the middle of the curve, further validating that this is a reasonable choice.

A.3.1 Extending LLM SAEs

We train two versions of extension SAEs - one for L1 loss SAEs and one for BatchTopK SAEs. In both cases, we begin with a pretrained SAE and add N latents randomly initialized with norm 0.1, and with the same encoder and decoder directions, following Olah et al. [18]. For the BatchTopK SAEs, we simply train the SAE from this point as normal, as the TopK auxiliary loss [11] will naturally ensure that the newly added latents do not simply die off.

For L1 SAEs with high L1 penalty, dead latents become a more serious problem. We find that most of the newly added extension latents will rapidly be killed off if we simply train as normal. To combat this, we re-warm-up the L1 penalty. However, we cap the minimum L1 penalty at λ_{min}, so for the portion of the warm-up where the L1 penalty would normally be below λ_{min}, the L1 penalty is left at λ_{min} instead. This capping helps ensure the existing SAE latents are not very disturbed by this change in the L1 penalty. If the final L1 penalty is λ_{min} or below, then we do not perform this warm-up at all, as the L1 penalty is not strong enough to immediately kill off the newly added latents.

For Gemma-2-2b SAEs, we set $\lambda_{\text{min}} = 10.0$. For Llama-3.2-1b SAEs, we set $\lambda_{\text{min}} = 0.5$.

This warm-up procedure is only used for the high-L1 variants in Figure 6c - for all other plots the L1 coefficient used is less than λ_{min}, so no warmup is needed.

A.4 Additional case study dashboards

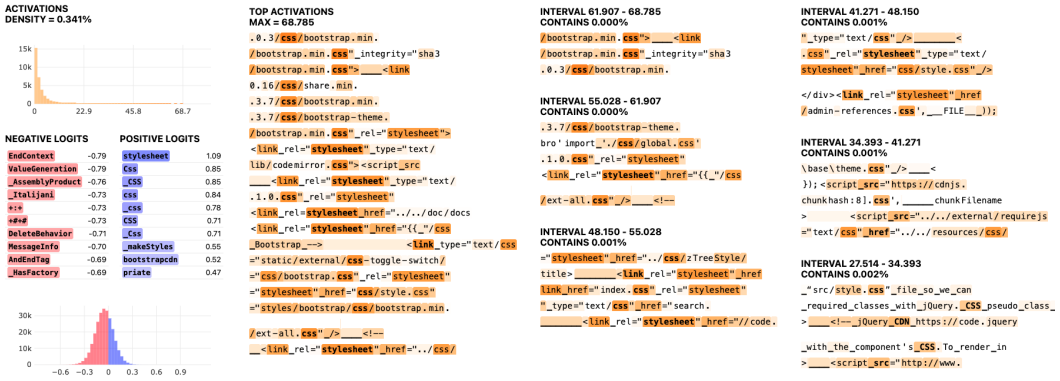


Figure 12: Dashboard for the newly added case study latent representing CSS scripts in HTML.

A.5 Toy balance matryoshka SAEs

To explore the effect of balancing matryoshka losses in a simple toy setting, we create a toy model with 4 true features, all mutually orthogonal and with unit norm in a 50 dimensional space. We set up a hierarchical relationship between these features, so feature 1 fires with probability 0.25, and features 2, 3, and 4 all fire with probability 0.15 only if feature 1 fires. Thus, feature 1 is the parent feature in the hierarchy and features 2, 3, and 4 are all child features.

We train a matryoshka SAE with 4 latents on 100,000,000 samples from this toy model. The matryoshka SAE has a single inner level consisting of 1 latent, to match the number of parent latents in our hierarchy. Since our goal with this toy is just to build intuition, we initialize the SAE to the correct solution and allow the training to thus pull it away from this correct solution. This also ensures

Figure 13: Dashboard for latent 3094, representing the "rel" HTML attribute used for CSS scripts. This latent has the highest negative δ-projection on the newly added case study latent.

that each variation of our SAE with different balancing co-efficients learns the same latents in the same order, so visual comparison is easy.

A.6 Toy unbalanceable matryoshka SAEs

The situation above where each child feature has the same probability of firing is unrealistic - we would expect that child features all fire with different probabilities from each other. Can we still balance the SAE perfectly in this situation? We adjust the toy model from above so that the 3 child features fire with probabilities 0.02, 0.2, and 0.5 for features f_2, f_3, and f_4, respectively. We then try to manually balance this SAE, finding that $\beta = 0.17$ gives roughly the best balance. We plots the resulting encoder/decoder cosine similarities in Figure 14.

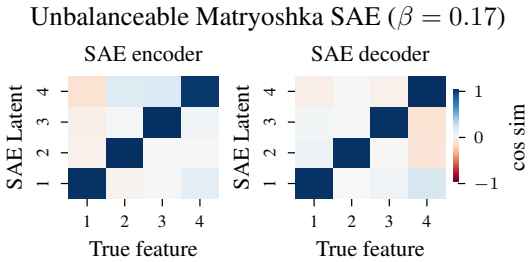


Figure 14: SAE encoder and decoder vs true feature cosine similarities for a balance matryoshka SAE where the child features fire with different probabilities. It's no longer possible to perfectly balance all 3 child features with the same β, but we can still do reasonably well.

We now see it is no longer possible to choose a single β that perfectly balances all 3 children. We see slight hedging of feature 4 in latent 1, and slight absorption of feature 2 in latent 1. Still, this looks decent compared to the full hedging or full absorption scenario, so we still expect that while balancing is not a perfect solution, it should be an improvement. We believe it should be possible to finding ways of better balancing the contribution of each outer latent on each inner latent, but this is left to future work.

A.7 SAE evaluation

A.7.1 SAEBench evals

We evaluate our SAEs mainly using SAEBench [14]. All evals are performed using default settings. We run all evaluations on an Nvidia H100 GPU with 80gb GPU memory. We evaluate on the following SAEBench tasks:

K-sparse probing k-sparse probing builds on the work of Gurnee et al. [12], where the goal is to create a linear probe from model activations using only k neurons as input to the probe. This was adapted for use as an SAE evaluation technique by Gao et al. [11]. We focus mainly on $k = 1$ sparse probing, which means finding the single best SAE latent that serves as a classifier for a given concept, and evaluating the accuracy of that latent. SAEBench includes supervised classification datasets on which k-sparse probing is evaluated.

Feature absorption The feature absorption metric in SAEBench is a variation on the metric defined in the original feature absorption work [5]. This metric uses a first-letter spelling task and first identifies the "main" latents for that task using k-sparse probing [12]. Then, the metric identifies cases where a linear probe is able to correctly perform the first-letter classification task, but the "main" SAE latents fail to perform the task. The metric also looks for other latents that project onto the linear probe direction and fire in place of the "main' latents. Lower absorption is better.

The SAEBench absorption metric also includes "absorptions fraction", "feature splitting", and "first-letter k=1 sparse probing" results as well, which we include in our extended results. Absorption fraction detects partial absorption, where a parent latent can still fire but weaker when an absorbing child latent fires as well. Feature splitting detects the amount of interpretable feature splitting occurring in the SAE. Interpretable feature splitting is still considered negative, as we would prefer that features do not split at all and the SAE can still represent general, high-level concepts. The k-sparse probing results for the first-letter spelling task is calculated as part of the absorption metric, but is an interesting sparse-probing result in and of itself.

Spurious concept removal (SCR) SCR builds on the SHIFT method from Marks et al. [17] to detect how well an SAE isolates concepts. The metric uses datasets where two properties are perfectly entangled, for instance "profession" and "gender", and trains a biased probe on these concepts. The SCR metric then detects how well k SAE latents can be ablated to de-bias the probe. If the SAE latents learn disentangled concepts, then it should only take a few SAE latents to perfectly de-bias the probe. A high SCR score means the SAE latents represent disentangled concepts.

Targeted probe perturbation (TPP) The TPP metric extends SCR to multi-class labels. Binary probes are trained for each class, and TPP measures how well ablating k SAE latents can degrade the performance of just one of the probes without degrading performance on the other probes. A high TPP score means that concepts are represented by distinct sets of SAE latents, rather than latents being entangled with many concepts.

A.7.2 Parts of speech (POS) probing dataset

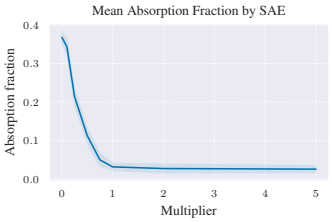
We are interested as well in general, high-frequency concepts that we expect should be learned in the inner-most levels of a matryoshka SAE. These concepts should be the most affected by both absorption and hedging, as these concepts can be considered parent concepts to most other concepts. Parts of speech (POS) is a great test-case for these general concepts, and are not covered by the SAEBench sparse probing task. As such, we create our own custom POS dataset using the Penn Treebank tagged sentences [16].

We simplify the Treebank parts of speech to the following core set:

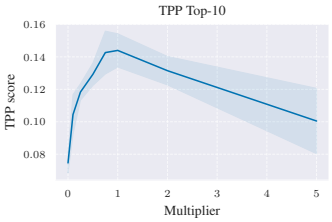
`"TO", "IN", "DT", "CC", "NNS", "PRP", "POS"`

We pass these tagged sentences through an LLM, and then collect activations for the final token of position of each word at a given layer in the LLM. We create a binary classification dataset for each of these parts of speech, and perform k-sparse probing [12] on SAE latents to find the top k latents that act as a classifier for each of these parts of speech.

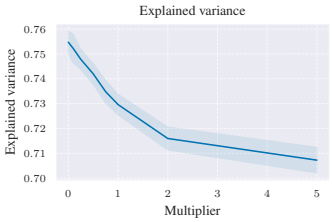
A.7.3 Balance matryoshka SAE full results



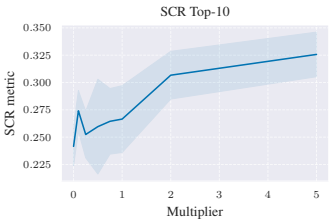
(a) SAEBench Absorption fraction. Lower is better.



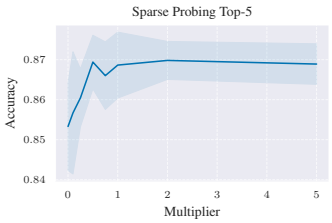
(b) SAEBench TPP top-10 metric. Higher is better.



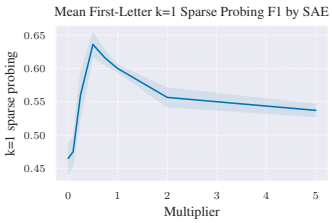
(c) Explained variance.



(d) SAEBench SCR top-10 metric. Higher is better.



(e) SAEBench K=5 sparse probing accuracy.



(f) K=1 first-letter sparse probing F1 score (SAEBench absorption).

Figure 15: Performance of balance matryoshka SAEs vs multiplier for extended metrics. The shaded area in the plots refers to 1 std. Multiplier=0 is equivalent to a standard non-matryoska SAE, and multiplier=1 is equivalent to a standard matryoshka SAE.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We provide evidence of the claims in the abstract and introduction in the toy models in Section 3, in LLM SAEs in Section 4, and balance matryoshka SAEs in Section 6.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in Section 9.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.

- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: We do not include theoretical results, only empirical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We include all relevant hyperparameters and training software / datasets used for all cases where we train an SAE, and include all code necessary to reproduce our results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often

one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We include all code necessary to reproduce our results, and all datasets used for training are also open source.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We include all training and test details of all evaluations we run.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We run multiple seeds for all experiments and provide errors bars on all results. We mention that the error bars are 1 standard deviation.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We mention that all experiments are performed on an Nvidia H100 with 80gb memory.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

Justification: Our research conforms to the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: Our work does not have a societal impact.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our work does not pose such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All datasets and models used in the work are properly credited and their license is respected.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: No new assets are introduced in the paper.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: We did not perform any crowdsourcing experiments or any experiments involving human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: We do not perform any experiments on human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: We do not use LLMs as an important component of the core methods of this research.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.