Tokenized SAEs: Disentangling SAE Reconstructions

Thomas Dooms [*1] **Daniel Wilhelm** [*1]

Abstract

Sparse auto-encoders (SAEs) have become a prevalent tool for interpreting language models' inner workings. However, it is unknown how tightly SAE features correspond to computationally important directions in the model. This work empirically shows that many RES-JB SAE features predominantly correspond to simple input statistics. We hypothesize this is caused by a large class imbalance in training data combined with a lack of complex error signals. To reduce this behavior, we propose a method that disentangles token reconstruction from feature reconstruction. This improvement is achieved by introducing a per-token bias, which provides an enhanced baseline for interesting reconstruction. As a result, significantly more interesting features and improved reconstruction in sparse regimes are learned.

1. Introduction

The holy grail of mechanistic interpretability research is the ability to decompose a network into a semantically meaningful set of variables and algorithms. SAEs have emerged as a promising method to extract interpretable context (Cunningham et al., 2023; Kissane et al., 2024; Dunefsky et al., 2024). However, the importance of SAE features to model computation is still unknown. This paper specifically studies the importance of local context on the variety of learned features. This is enhanced by an SAE training token frequency imbalance resulting in bias toward local context.

We find that many features in medium-sized SAEs such as RES-JB (Lin & Bloom, 2024) are affected by this imbalance. This causes them largely to reconstruct a direction biased toward the direction of the most prevalent training data unigrams. Empirically, we estimate that between 35% and 45% of the features reconstruct common unigrams and almost 70% reconstruct common bigrams. We hypothesize these features then moreso reflect training token statistics than interesting internal model behavior. We attribute this phenomenon to the following two observations:

- Local context is a strong approximation for latent representations, even in deeper layers.
- There is a prominent class imbalance in the training data of SAEs. Certain local combinations will appear much more frequently than specific global interactions.

Given both their frequency and strength in the representation, these local contexts occupy the majority of the features an SAE uses to minimize its reconstruction error. We show this to hold for all kinds of common n-grams. Furthermore, we hypothesize this to be the cause for a range of pathological behaviors exhibited by SAEs, such as the inability to generalize out-of-distribution in certain contexts (Templeton et al., 2024; Gurnee, 2024).

Fortunately, these insights can be leveraged toward a solution; we propose a means to disentangle these "uninteresting" feature reconstruction tokens from the interesting features. This is accomplished by extending the SAE with a per-token bias, allowing the SAE to represent a "base" reconstruction for each token. This leaves room for more semantically useful features. Furthermore, the proposed bias lookup table is efficient, resulting in SAEs becoming less compute-intensive to train. Specifically, our contributions are:

- We identify and characterize the issue of SAEs learning token reconstruction features due to the input distribution and formulate why this is the case.
- We propose a technique to mitigate this behavior by separating token reconstruction from context reconstruction. We name this approach *Tokenized SAEs*.

2. Background

2.1. Notation.

For interpretability, it is important to relate a sequence of N input tokens $\mathbf{x} \in \mathbb{T}^N$ to activations at some location p. This mapping exists as a function A^p.

We define an n-gram as $[t_0, t_1, t_2, \ldots, t_n] \in \mathbb{T}^{n+1}$. In this paper, we assume $t_0 = \text{BOS} \in \mathbb{T}$, the beginning-of-sequence token.

[*]Equal contribution [1]Independent. Correspondence to: Thomas Dooms <doomsthomas@gmail.com>.

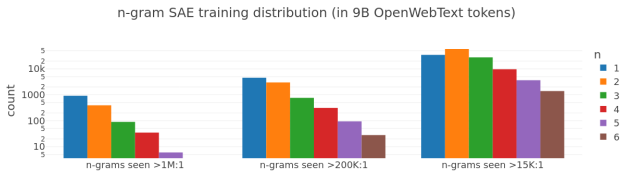

Figure 1. Particular n-grams are seen exponentially more often than others. Many combinations occur millions of times more than an arbitrary n-gram.

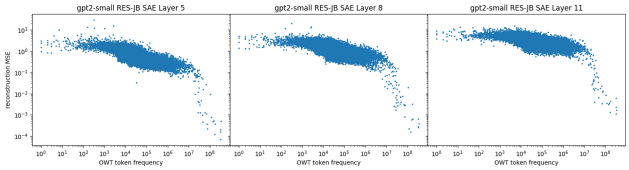

Figure 2. With increasing OpenWebText token frequency, the reconstruction MSE of unigrams in layers 5, 8, and 11 of the RES-JB SAE decreases. This indicates the SAE effectively memorizes the most common tokens. This effect does not occur with the most common bigrams, likely because they are composed of the most common unigrams and/or occupy unigram subspaces.

2.2. Imbalance.

We will examine sparse auto-encoders at some location p. These map each row vector of $A^p(\mathbf{x})$ to itself, reconstructing it. The sparsity of the hidden layer is minimized, leading to seemingly interpretable features.

During training, short n-grams are exponentially over-represented due to an imbalanced training distribution. This biases the SAE toward reconstructing these short n-gram inputs.

This occurs because the SAE is trained to reconstruct each row vector of $A^p(\mathbf{x})$. Due to attention, each is a function only of the prior tokens, i.e. for row i, $A^p(\mathbf{x})_i = A^p(\mathbf{x}_{\leq i})_i$. For example, for each training prompt the SAE is provided training examples $A^p(\text{BOS})$ and $A^p(\text{BOS}, t)$, where t follows the distribution of training set tokens.

For row vector i, there are at most $|\mathbb{T}|^i$ possible activations. However, in practice the degree of over-representation can be measured directly for a given training set. Assuming each training sequence begins at a random token, the n-gram frequency distribution follows the dataset's n-token frequency distribution. We show many n-grams are more than a million times more likely than baseline (Figure 1).

This results in an effect similar to "imbalanced regression"[1], where the target space distribution is sampled unevenly during training (Yang et al., 2021; Stocksieker et al., 2024). Each row vector follows an distribution based on its index, causing the SAE to become biased toward the highest-weighted regions of space (here the most common small n-gram activations). Such a class weighting causes a general MSE-trained regressor to underestimate rare labels (Ren et al., 2022).

We show experimentally this causes higher reconstruction loss for less common unigrams, since they must "overcome" the biases (Figure 2).

[1]The terminology "imbalanced" accurately describes its implications, although it may best be described as a weighted class.

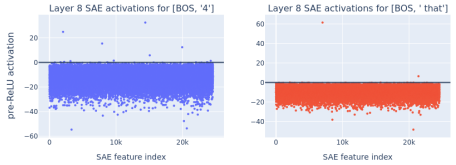

Figure 3. To memorize unigrams exactly and sparsely, the SAE represents each using a small subset of feature neurons that fire in response to the unigram. Due to the incorporation of prior token information, SAEs in later layers often also strongly memorize bigrams.

3. Sparse Auto-Encoders

The motivation for training SAEs is often presented as feature discovery. This is achieved by reconstructing the hidden representations through a sparse hidden basis, often called features. We show that SAEs memorize and organize themselves around the most common input n-grams, contributing to the observed correlation between them (Figure 4).

3.1. Memorization.

Suppose the most common n-gram inputs cause a training imbalance. Then we would expect to see (and observe) that with larger n-gram frequency, the reconstruction MSE decreases (Figure 2) and fewer features activate (Figure 3). In later layers, attention has likely consolidated information from other tokens, making the most common representations involve prior tokens. For example, many common words require multiple tokens to represent. We have observed evidence for this by noting that unigrams are most commonly activated in early layers and bigrams in later layers.

3.2. Token Reconstruction Features.

Suppose some SAE is represented by a set of features $\mathbb{F}$. Based on the prior experimental results and imbalance the-

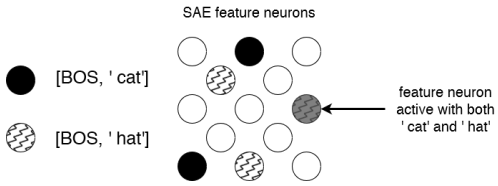

Figure 4. Illustrating experimental results, an individual feature neuron is activated when one of its associated n-grams is present. The most common tokens will occupy a full feature while less common tokens will share a feature. To maximize reconstruction, this sharing occurs between semantically similar tokens.

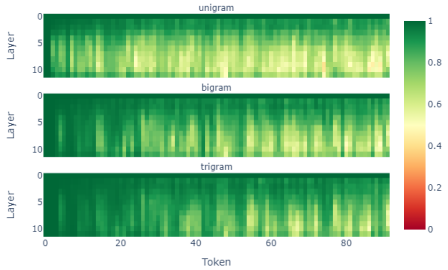

Figure 5. Measuring cosine similarity of hidden representations and a patched version which only includes the last n tokens in GPT-2 small. Trigrams are generally an adequate approximation across the network.

ory, we hypothesize that each common n-gram $\mathbf{x}$ maps to a subset of $\mathbb{F}$ which $A^p(\mathbf{x})$ activates. The set of increasingly most common n-grams approaches a cover of $\mathbb{F}$, with the exclusion of dead features. (Figure 4)

An SAE feature activates when a common activation pattern appears in $\mathbf{x}$, corresponding to some aspect of an n-gram. We show this experimentally by predicting which input tokens will activate a given feature. In RES-JB layer 8, of the 76% of features activated by a unigram, 39% matched the top unigram activation and 66% matched at least one. The 24% of features not activated by a unigram illustrate:

1. In later layers, some common SAE inputs may result from non-local information that more likely occurs in longer sequences. Experimental evidence shows that a minority of layer 8 GPT-2 features do not respond to any of 212K most-common ($n \leq 6$)-grams. A qualitative characterization of these features reveals these features exhibit more interesting semantic behavior.

2. This method operates under the assumption that some n tokens prior to row vector i are sufficient to mostly describe the SAE inputs, i.e. $A^p(\mathbf{x})_i \approx A^p(\mathbf{x}_{i-n})_i$. We show this to generally be the case in Figure 5, even in complex models and later layers (Figure 6). See Appendix C for additional discussion.

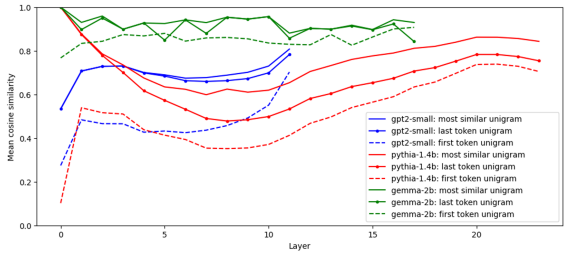

Figure 6. Tokenized SAEs assume residual activations are similar to those of individual unigrams, specifically the final input token. Toward this, we investigate whether unigram residuals are good approximations for 128-token residuals. We compare 38K 128-token residuals to all unigram residuals, recording the mean cosine similarity of (a) the most similar unigram, (b) the final-token unigram, and (c) the first-token unigram (an effectively random control). We find (a) residuals have very high cosine similarity to unigram residuals across all models and layers, and (b) the final token unigram is often nearly-closest.

4. Tokenized SAEs

To resolve the abovementioned issues, we propose a new method that separates token reconstruction features from the dictionary. This is achieved by adding a separate path to the SAE, which is only concerned with providing a base reconstruction of tokens. Concretely, we add a lookup table which acts as a per-token bias (Equation 2).

$$\mathbf{f}(\mathbf{a_t}) = \text{ReLU}(W_{\text{enc}}(\mathbf{a_t} - b_{dec}) + b_{enc}) \quad (1)$$

$$\hat{\mathbf{a}}_t = W_{\text{dec}}\mathbf{f}(\mathbf{a_t}) + b_{dec} + W_{lookup}(\mathbf{t}) \quad (2)$$

This lookup table has no impact on the encoding thus computing feature activations requires no change in setup. However, token information is necessary for the reconstruction. We provide further details in Appendix A.

4.1. Training

As with the encoder, sensibly initializing the lookup table leads to large improvements in learning speed and final convergence. We do so by using the activations on the target point by only sampling each token without context, or formulaically; $W_{lookup}(\mathbf{t}) = A^p(\text{BOS}, \mathbf{t})_1$.

Since the lookup table is essentially a hyper-sparse set of features, it is necessary to increase its learning rate to yield better reconstructions. A sensible approach is to multiply the learning rate by the approximate L0 of the SAE, this theoretically results in equal gradient updates. However, empirical results indicate using even higher learning rates is beneficial. More training-specific information can be found in Appendix A.

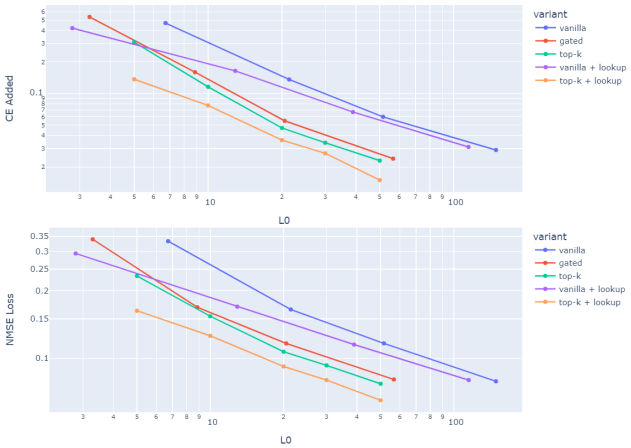

Figure 7. A Pareto frontier comparison of various SAEs on layer 8 of GPT-2 using an expansion factor of 16. The cross-entropy added and normalized MSE compared to the L0 norm are shown. All SAEs were trained on about 300M tokens until (close to) convergence. Due to human error, the 'vanilla + lookup' did not learn its lookup table.

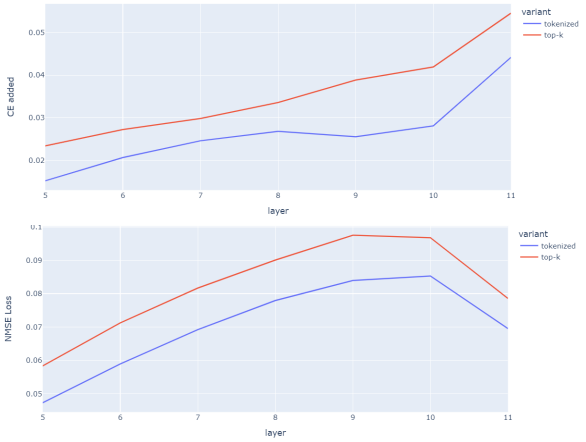

Figure 8. Comparison of a top-k SAE and its tokenized variant on layers 5 through 11 on GPT-2. All SAEs were trained with an expansion factor of 16, $k = 30$ and about 250M tokens were used.

4.2. Reconstruction

The experiments in this section are all performed on layer 8 of GPT-2 small. This is sufficiently deep in the model that we would expect complex behavior to have arisen. Furthermore, a breadth of public pre-trained SAEs can be used for comparison. We use the *added* cross-entropy (Equation 3) to measure the impact on the model prediction and *normalized* MSE (Equation 4) to measure reconstruction.

$$CE_{added}(\mathbf{x}) = \frac{CE_{patched}(\mathbf{x}) - CE_{clean}(\mathbf{x})}{CE_{clean}(\mathbf{x})} \tag{3}$$

$$\text{NMSE}(\mathbf{x}) = \frac{||\mathbf{x} - \text{SAE}(\mathbf{x})||_2}{||\mathbf{x}||_2} \tag{4}$$

Figure 7 shows a large-scale comparison of Pareto frontiers for various architectures. We benchmark vanilla SAEs (Cunningham et al., 2023), gated SAEs (Rajamanoharan et al., 2024) and Top-k SAEs (Gao et al., 2024). The vanilla and the gated SAEs are trained with decoder sparsity loss from Conerly et al. (2024). Beyond this, no additional training techniques (resampling, ghost gradients, ...) were used.

This indicates Tokenized SAEs outperform their non-tokenized counterparts by a significant margin. They achieve the same reconstruction while being about 25% sparser. Furthermore, in hyper-sparse regimes, they consistently yield good reconstructions and follow a clean linear pattern in contrast while their counterparts consistently deteriorate.

4.3. Suite

To demonstrate the generality of the approach, we train two suites of SAEs on layers 5 through 11 for GPT-2. Both use top-k as an activation function, one has a lookup table (tokenized) and one doesn't (top-k). This shows that the lookup table consistently enhances reconstructions, with no visible degradation in deeper layers (one could even argue the opposite).

One aspect not shown in the plots but still wish to highlight is the improved training speed with the lookup table. We consider the final value of baseline NMSE and CE added, tokenized SAEs reach that value 6-10x faster across all layers of GPT-2. Training a competitive SAE (according to these metrics) can be achieved in mere minutes on consumer hardware.

4.4. Scaling

One salient concern for this approach is the impact of deeper and more complex models on the utility of the token lookup table. To this end, we perform preliminary experiments on Pythia 1.4B for layers 12, 16 and 20 using the newly proposed top-k. Results indicate that tokenized SAEs still outperform their baselines. This leads us to believe token subspaces may be more salient than commonly believed.

	12	**16**	**20**
Top-k	0.076	0.081	0.155
Tokenized	0.045	0.055	0.121

Table 2. CE added across 3 layers of Pythia-1.4B using top-k SAEs with $k = 50$. Due to computation constraints, the SAEs are undertrained, using only 70M tokens. Qualitatively, the training progression showed no signs of the baseline 'catching up'. The NMSE (not shown) exhibits a similar improvement.

	RES-JB	Vanilla	Vanilla*	Top-k	Top-k*
Consistency	4.1	3.6	3.4	3.4	4.2
Complexity	2.5	1.1	2.9	1.7	3.0

Table 1. We manually score 20 features from multiple SAEs (tokenized denoted by an asterisk) and note their mean complexity and consistency according to Cunningham & Connerly (2024). In short, the complexity score ranges from 1 (unigrams) to 5 (deep semantics). The consistency ranges from 1 (no discernable pattern) to 5 (no deviations). Given the limited sample size, these results should be interpreted cautiously; they provide preliminary indications rather than definitive evidence. Scoring features manually is time-consuming.

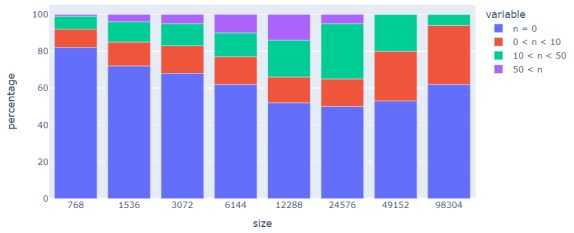

Figure 9. Approximate categorization of features by the number of tokens they activate on (above a threshold of 5). Experiments are performed on JB's suite for GPT-2 layer 8. In smaller SAEs, there is no bandwidth to represent individual or small sets of tokens. In medium-sized SAEs, we see features representing small sets of tokens. As size increases, it starts representing specific tokens.

5. Feature Comparison

5.1. Quantitative

We quantify the number of uninteresting features by sampling each possible unigram (pre-pended with BOS) and measuring the number of features that activate strongly for it. Features that strongly correspond to very few tokens are highly likely to be feature reconstruction tokens. We display which distribution they follow in Figure 9.

We perform the same experiment on the Tokenized SAEs from Figure 7. We find that the number of features that activate on any single unigram is below 5% for all of them. Appendix C contains more in-depth analyses regarding the differences.

5.2. Qualitative

We performed a blind study on five layer 8 GPT-2 SAEs. A top-k and vanilla SAE, with their tokenized counterpart and finally RES-JB (Lin & Bloom, 2024) as a baseline. The results are shown in Table 1 and suggest that our SAE features are about equally consistent, but their complexity is noticeably higher. Appendix B includes a list of cherry-picked features to corroborate these subjective findings. In summary, we find that features generated by Tokenized SAEs tend to be more semantically meaningful and contain fewer uninteresting features.

6. Future Work

Tokenized SAEs have a wide possible range of extensions. One obvious candidate is to incorporate n-gram statistics, instead of simply unigrams. We believe this to be mostly an engineering challenge; it requires efficiently making a sparse, multi-token lookup table. Furthermore, while this paper only considers the tokens as a sparse basis, one could consider a previous SAE as a basis. This would incentivize structuring around already-existing features, likely improving circuit analysis.

Additionally, a more thorough study into the quality of Tokenized SAE features is still to be performed. This should be done on both the dictionary and the lookup table. The former is related to the incorporated non-local context and the latter is related to the token reconstruction. Exactly characterizing this token reconstruction similarity in latent representations is undoubtedly useful.

7. Acknowledgements

This project originated as a MATS sprint. We thank Jacob Dunefsky and Neel Nanda for their insightful discussions and guidance. We also thank Michael Pearce for coining the project's name. This research received funding from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme.

8. Contributions

Thomas conceived the proposed approach, trained the SAEs, and analysed the TSAEs and their differences. Daniel noticed and researched the training imbalance and analysed the TSAEs. The paper was written in tandem.

References

Conerly, T., Templeton, A., Bricken, T., Maruc, J., and Henighan, T. Circuits updates - april 2024. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/april-update/index.html.

Cunningham, H. and Connerly, T. Circuits updates - june 2024. *Transformer Circuits Thread*, 2024.

URL https://transformer-circuits.pub/2024/june-update/index.html.

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

Dunefsky, J., Chlenski, P., and Nanda, N. Transcoders enable fine-grained interpretable circuit analysis for language models. Alignment Forum, 2024. URL https://www.lesswrong.com/posts/YmkjnWtZGLbHRbzrP.

Gao, L., la Tour, T. D., Tillman, H., Goh, G., Troll, R., Radford, A., Sutskever, I., Leike, J., and Wu, J. Scaling and evaluating sparse autoencoders, 2024. URL https://arxiv.org/abs/2406.04093.

Gurnee, W. Sae reconstruction errors are (empirically) pathological. Alignment Forum, 2024. URL https://www.lesswrong.com/posts/rZPiuFxESMxCDHe4B.

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

Lin, J. and Bloom, J. Announcing Neuronpedia: Platform for accelerating research into Sparse Autoencoders. March 2024.

Rajamanoharan, S., Conmy, A., Smith, L., Lieberum, T., Varma, V., Kramár, J., Shah, R., and Nanda, N. Improving dictionary learning with gated sparse autoencoders, 2024.

Ren, J., Zhang, M., Yu, C., and Liu, Z. Balanced mse for imbalanced visual regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Stocksieker, S., Pommeret, D., and Charpentier, A. Boarding for iss: Imbalanced self-supervised: Discovery of a scaled autoencoder for mixed tabular datasets, 2024.

Templeton, A., Conerly, T., Marcus, J., Lindsey, J., Bricken, T., Chen, B., Pearce, A., Citro, C., Ameisen, E., Jones, A., Cunningham, H., Turner, N. L., McDougall, C., MacDiarmid, M., Freeman, D., Sumers, T., Rees, E., Batson, J., Jermyn, A., Carter, S., Olah, C., and Henighan, T. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

Yang, Y., Zha, K., Chen, Y.-C., Wang, H., and Katabi, D. Delving into deep imbalanced regression. In *Proceedings of the 38th International Conference on Machine Learning (ICML 2021)*, 2021.

A. Training Setup

A.1. General

The setup is intentionally kept as simple as possible to avoid confounding factors. Specifically, no resampling or ghost gradients are used. All SAEs are trained on a subset of C4 tokens using a context length of 256. Tokens are collected in a buffer of size 128K and then sampled in batches of 4096 to train the SAE. We use the Adam optimizer with a learning rate of $1e^{-4}$ and a cosine annealing learning schedule. This base training setup is consistent across all experiments.

For all GPT-2 models, we use an expansion factor of 16 (12288 features). For the Pythia-1.4B, we use an expansion factor of 8 (16382 features).

A.2. Initialization

We initialize W_{enc} with the response of the W_{dec} for all SAEs. Further, as stated in subsection 4.1, we initialize the lookup table to the activations for that token without tokens. Both these initialization procedures attempt to attain the same goal; approximate an identity. We therefore "balance" the lookup $W_{lookup}(tok) = \alpha A^p(\text{BOS}, tok)$ and the encoder $W_{enc} = (1-\alpha)W_{dec}^T$ using a hyperparameter that sums to one. While all values outperform non-tokenized SAEs, we found $\alpha = 0.5$ to work well across all experiments.

Interestingly, we can approximate α during training using Equation 5 where $W_{original}$ is the lookup at the start of training. This reveals how much the SAE naturally steers towards learning balancing the lookup and SAE. We find that the middle layers of GPT-2 converge towards 0.6 and the later layers towards 0.5. The middle layers of Pythia-1.4B tens to 0.45 and the later ones to 0.4.

$$\hat{\alpha} = \frac{1}{n}\sum_{i}^{n} \frac{W_{original} \cdot W_{lookup}}{||W_{original}||_2^2} \tag{5}$$

A.3. Learning Rate

In subsection 4.1, we note that increasing the learning rate of the lookup table improves the reconstructions. The reasoning is that, due to the difference in sparsity, each entry in the lookup table is updated much less than the features in the SAE. We empirically find that increasing the learning rate to 0.01 (up to 100x higher than the global learning rate) yields good results. We attribute this to the lookup table being more stable to train and again to token frequency imbalance. One could also dynamically change the learning rate of each lookup entry based on this frequency, we did not try this.

A.4. Memory and Compute Overhead

Adding a lookup table does not impact training or inference time significantly since it is an extremely efficient operation. We noticed a 3-5% increase in training time by introducing the table. In terms of memory overhead, the lookup table has a larger impact. For common SAE sizes, the memory requirements double. We do not think this to be an issue since SAEs are generally not memory-heavy; a whole GPT-2 suite would consume about 3GB of memory. If this is an issue, one could consider using a truncated lookup table, containing only the n most common tokens.

B. Cherry-Picked Features

Potential Categories of First 25 Features (top-k TSAE, layer 8):

- **Overall thematic**: 16 (movie storylines)
- **Part of a word**: 10 (second token), 12 (second token), 17 (single letter in a Polish word), 19 ("i/fi/ani")
- **Thematic short n-grams**: 15 (" particular/Specific"), 23 (defense-related), 28 ("birth/death")
- **N-grams requiring nearby period/newline/comma**: 7 ("[punctuation] If"), 18 ("U/u"), 22 ("is/be")
- **Bigrams**: 2 ("site/venue"), 6 ("'s"), 8 ("shown that"/"found that"/"revealed that"), 14 ([punctuation] "A/An/a/ The")

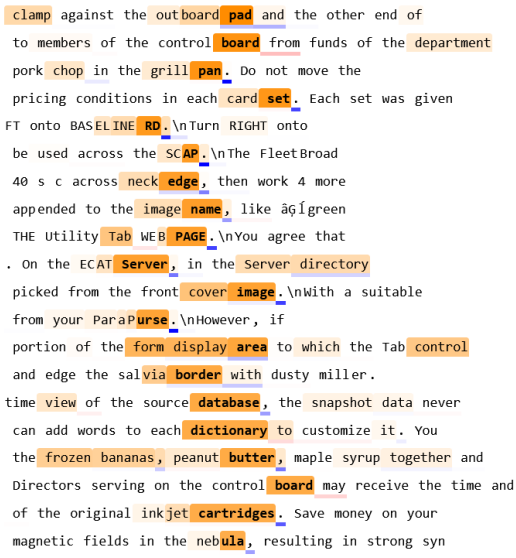

Figure 10. An end of sentence feature, boosting ”.”, ”,”, and ”and” tokens.

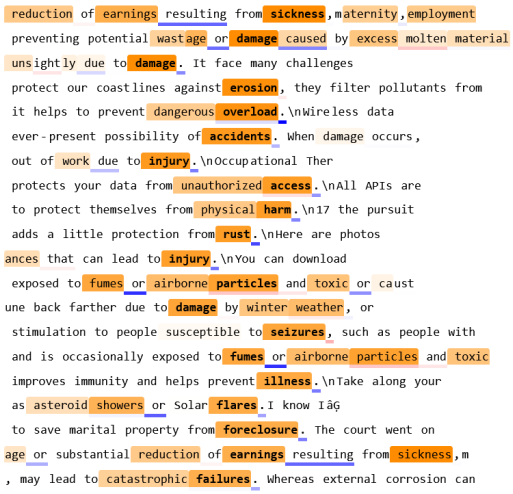

Figure 11. A health hazard feature.

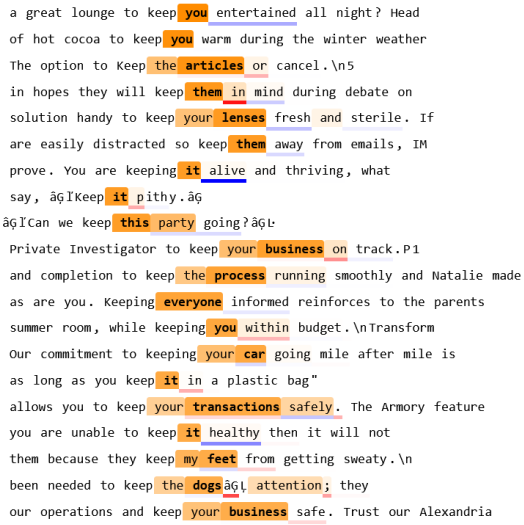

Figure 12. A direct object feature.

- **Categoric bigrams**: 13 ([NUM] ”feet/foot/meters/degrees”)
- **Skipgrams**: 1 (”in the [TOK]”), 21 (”to [TOK] and”)
- **Locally Inductive**: 11 (requires a sequence of punctuation/short first names)
- **Globally Inductive**: 24 (activates only when final token earlier in the prompt)
- **Less Than 10 Activation (implies low encoder similarity with input)**: 0, 4, 5, 9
- **Unknown**: 3, 20

Specific Interesting Features (top-k TSAE, layer 8):

- **36**: ”.\n[NUM].[NUM]”
- **40**: Colon in the hour/minute ”[1-12]:”
- **1200**: ends in ”([1-2 letters]”
- **1662**: ”out of [NUM]”/”[NUM] by [NUM]”/”[NUM] of [NUM]”/”Rated [NUM]”/”[NUM] in [NUM]”
- **1635**: credit/banks (bigrams/trigrams)
- **2167**: ”Series/Class/Size/Stage/District/Year” [number/roman numerals/numeric text]
- **2308**: punctuation/common tokens immediately following other punctuation
- **3527**: [currency][number][optional comma][optional number].
- **3673**: ” board”/” Board”/” Commission”/” Council”
- **5088**: full names of famous people, particularly politicians
- **5552**: ends in ”[proper noun(s)]([uppercase 1-2 letters][uppercase 1-2 letters]”
- **6085**: ends in ”([NUM])”
- **6913**: Comma inside parentheses

Many features were found to activate on exact copies of the final n-gram. It is unknown if this is a possibility for all features.

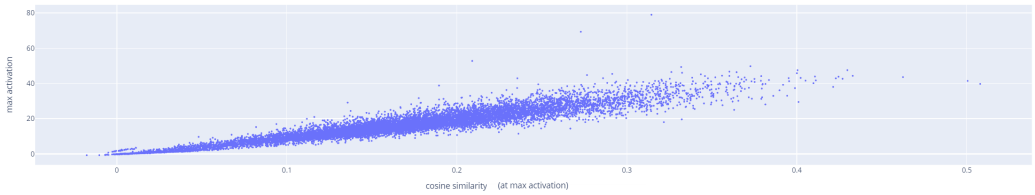

Figure 13. Due to how SAE activations are computed, feature activation strength is correlated with input vector cosine similarity with W_enc. Low-activating features likely are not detecting signal in the input. (Figure shows top-k tokenized layer 8.)

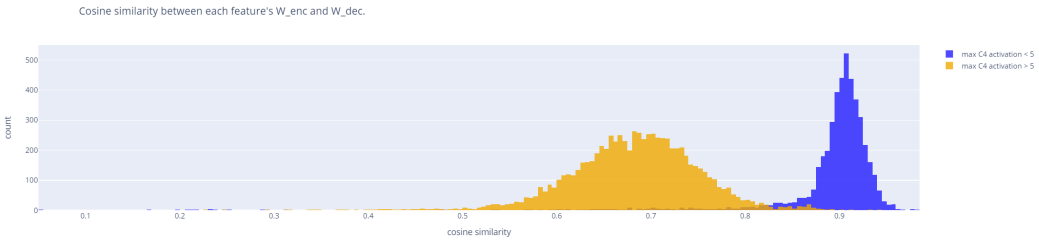

Figure 14. We initialize W_enc as the transpose of W_dec. This resulted in an easy post-facto test for dead features – simply comparing the cosine similarity of the encoder and decoder. As shown, the peak at 0.9 exactly corresponds to features which never activate more highly than 3 (in our top-k tokenized SAE layer 8).

C. Additional Analysis

C.1. Low feature activations imply low similarity with input vector

It is important to ask whether an activated feature is detecting something of significance or not. One method to detect this is by the strength of the activation. The mechanics of the encoder computation indicate that larger feature activations will correlate with larger cosine similarity between the input vector and W_enc (Figure 13). Hence, a small-magnitude activation likely indicates the feature has not detected a signal.

Due to this, a minimum activation threshold is advisable when evaluating features.

C.2. Recognizing dead features by encoder/decoder similarity

Because we pre-initialize each feature with W_enc and W_dec transposed, an interesting finding is that dead features correspond nearly exactly to features with high cosine similarity between each feature's encoder and decoder. This can be used post-facto to detect dead features:

Dead features are evidenced by high cosine similarity between W_enc and W_dec, since they were pre-initialized as transposes (Figure 14). Here, we show these groups correspond nearly exactly to low test set activations (in gpt2-small layer 5 TSAE).

We examined the high-similarity group using four metrics, concluding they are likely not valid features:

- Nearly all are completely dissimilar to RES-JB features ($<$0.2 max cosine similarity).
- Nearly all have a top activation $<$3 (activations are normally distributed about 0).
- Nearly all are rarely ($<$1-10%) in the the top 30 activations. (However, nearly all features with $<$0.85 similarity are sometimes in the top 30.)
- Manually looking at the activations, the features are often difficult to interpret.

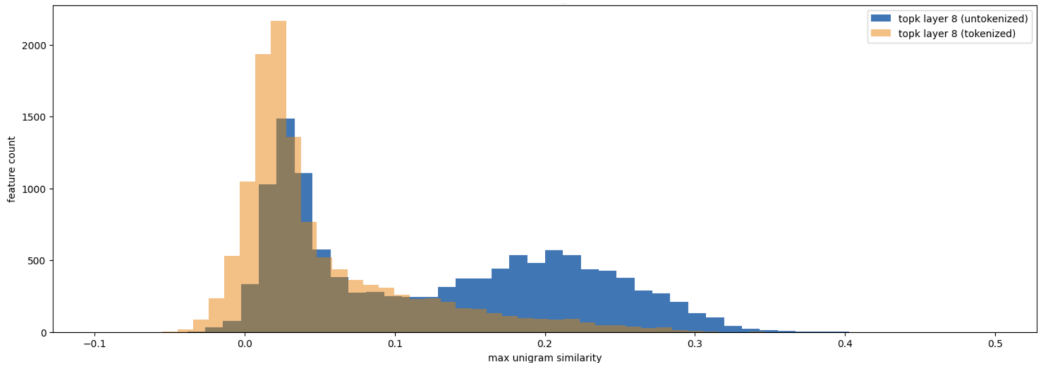

Figure 15. A central claim of TSAEs is that unigram-based features are reduced. Compared to a similarly-trained non-tokenized SAE, we see that significantly fewer TSAE features have significant max cosine similarity between the encoder weights and all unigram inputs. Hence, TSAE features will not respond as often to individual tokens.

C.3. Feature complexity of TSAEs

Measuring complexity is difficult, since feature activations may have multiple causes which are not yet fully understood. That said, a central motivation for TSAEs is that by excluding many ”simple” unigram-based features, features may potentially represent more complex concepts (yet still be interpretable).

- First, we show that TSAE features are largely no longer unigram-based when compared to an identically trained non-tokenized top-k SAE. To measure this, we determine the max cosine similarity between all unigram input vectors and each feature's encoder weights. We find that W_enc is drastically less similar to unigram features in a tokenized SAE (Figure 15).
- Second, we determine whether the additional features may be considered more ”complex”. To measure this, we examine features only with a minimum max activation to ensure they are not dead and properly detect some signal. Taking the top-activation prompt, we activate increasingly large suffix n-grams until the activation becomes (a) positive and (b) within 90% of the maximum activation (to avoid outlier maximum indices). The former often indicates the beginning of an increasing activation, while the latter indicates a strong encoder weight similarity to the input.

  Plotting the percentage of features at each minimum n (Figure 16), we notice that indeed TSAEs have more features activating at each $n > 2$ than a similarly-trained non-tokenized SAE. At least for this metric, we conclude that indeed the loss of unigram features translated into additional features requiring longer context.

C.4. More on final-token subspaces

Here, we provide additional support that resid_pre activations are strongly related to a token subspace. We find that regardless of model complexity and layer – and even with Gemma 2B's 256K vocabulary – >20% of the time a prompt's final-token (or a near-exact token) residual is closer than any other unigram residual (Figure 17).

D. Neuronpedia feature Study

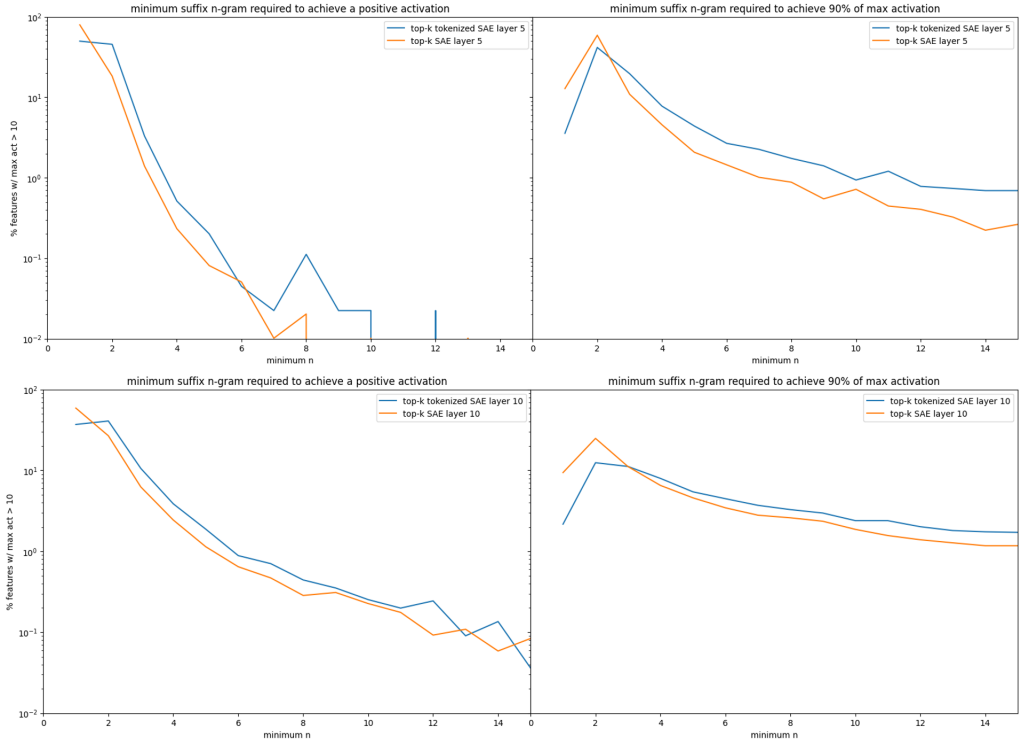

Figure 16. We measure feature complexity by finding the minimum suffix n-gram of each top-activating feature (>10) that results in a positive activation (left) or 90%-max activation (right). We note that a larger percentage of features in non-tokenized SAEs are unigrams ($n = 1$), while for $n > 2$ TSAEs generally have more "complex" features by this metric. Further, we see that layer 5 (top) achieves positive activations entirely with small n compared to layer 10 (bottom).

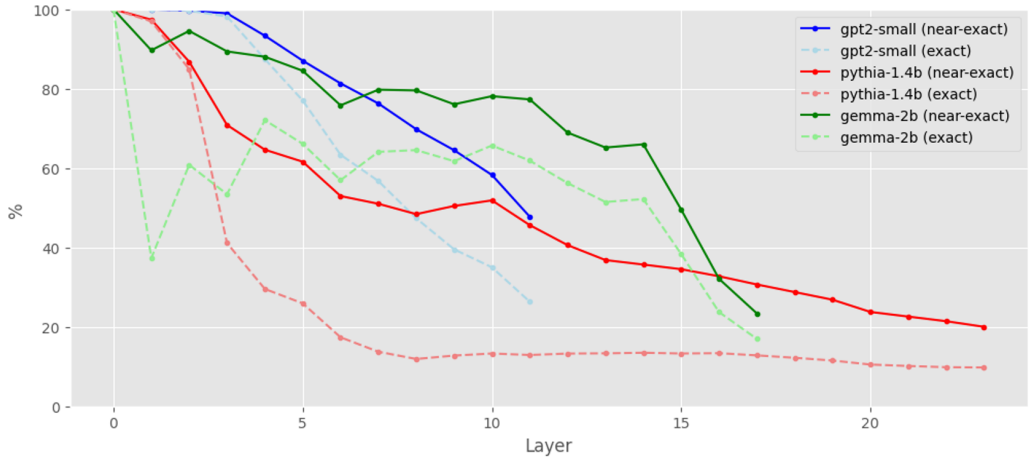

Figure 17. At each layer, we measure how many example prompts resid_pre activations are nearest to their final-token unigram activation than any other. Surprisingly, even in the final layer a large percentage are closest. When making a near-exact comparison, we compare token strings after stripping whitespace and lowercasing. (Typically, these unigram activations are nearby in space.)

Index	Term	Type
0	numbers	Unigram collection
1	“Pier”	**Unigram**
2	“weeks”/”months”/”years”	Unigram collection
3	Token after sorry/apologize	Bigram collection
4	separator/time	Attention
5	“in”	**Unigram**
6	Adjectives related to famousness	Unigram collection + attention
7	recipe(s)	**Unigram**
8	Causality (by a/due to)	Bigrams + attention
9	“Ļ”	**Unigram**
10	“Ļ” (again, look it up)	**Unigram**
11	Not sure	Attention
12	“told”	**Unigram**
13	solved, addressed, resolved	Unigram collection
14	“example”	**Unigram**
15	Really not sure…	*nan*
16	“With”	**Unigram**
17	“Ste”	**Unigram**
18	numerics in brackets (references)	Bigram collection
19	“s” after number (20s)	Bigram collection
20	Anglo + Alred + Pf	Unigram collection

Table 3. A qualitative study into the first 21 features of Joseph Blooms GPT-2 resid_pre SAE on layer 8. We show that more than half of the features represent uninteresting reconstructions.