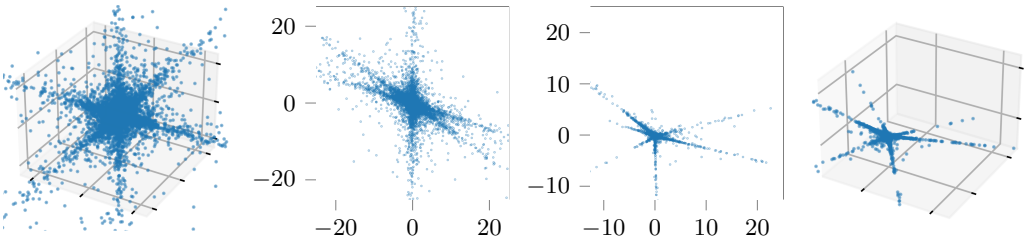


(a) Superposed inputs (b) MLP inputs (c) MLP outputs (d) Superposed outputs

Figure 3: An example of the effect of a randomly initialized neural network on superposed input data. We take 10K samples of $n_s = 3$ sparse input features from a Lomax distribution with shape $\alpha = 1$ and scale $\lambda = 1$ and project these to $n_d = 2$ dense input features by an i.i.d. standard normal matrix. Then, we pass the dense outputs to a two-layer MLP with ReLU activation and hidden size of $4n_d$ and recover $n_s = 3$ sparse outputs by the inverse of the previously generated projection matrix.

occurrence of the token, i.e., a token does not have a consistent embedding vector (Section 3). For the trained variant, the entropy increases across layers, i.e., the further into the model, the less likely the maximally activating examples for each latent contain activations concentrated on a single token. This is also expected: at later layers, we expect more abstract features that are less similar to token embeddings. Finally, the entropy for randomized models tends to be lower than for either the trained or control variants, indicating that latents are activated specifically at one or a few IDs.

In combination with the preceding results, this suggests that standard SAE quality and auto-interpretability metrics are missing an important aspect of SAE features: their 'abstractness'. While the token distribution entropy is not a direct measure of 'abstractness', it suggests that the randomized variants, viewed in the context of their similar auto-interpretability scores to the trained variant, remain able to learn simple, single-token features. However, unlike the trained variant, the features of the randomized variants do not become more complex as the layer index increases.

4 A TOY MODEL OF SUPERPOSITION IN RANDOM NETWORKS

We speculated in Section 1 that the apparently high degree of sparsity and interpretability in the activations of randomized transformers might be because the input data exhibits superposition, which neural networks preserve, or neural networks somehow amplify or even introduce superposition into the input data. In this section, we examine both possibilities through the lens of toy models. We find some evidence to support each potential cause, but we leave the question of which predominates in the case of randomized transformers and the results detailed in the main text to future work.

4.1 MATRIX MULTIPLICATIONS PRESERVE SUPERPOSITION

First, we consider a simplified model to demonstrate that multiplication by a weight matrix W preserves superposition. Imagine that we generate superposed input data x by first generating n_s i.i.d. 'sparse' features z from a heavy-tailed Lomax distribution $z \sim \mathrm{Lomax}(\alpha, \lambda)$. We can project the higher-dimensional, sparse z down to lower-dimensional, dense x with a matrix D, then add Gaussian noise with a small variance Σ, $x \sim \mathcal{N}(x; Dz, \Sigma)$. Importantly, if we multiply x by some matrix W, then $x' = Wx$ is *also* superposed: it is generated by the same model as x, except with different noise covariances and mappings from z to x, namely $x' \sim \mathcal{N}(z; WDz, W\Sigma W^T)$.

We can see that the same intuition might extend to neural networks with nonlinearities by visualizing the results of passing the dense activations through a simple feed-forward network (MLP). Figure 3 shows an example where $n_s = 3$ sparse features are projected down to $n_d = 2$ dense features, and the MLP outputs appear superposed despite the non-linearity. Moreover, it suggests that NNs might amplify superposition rather than only preserving it: comparing the inputs (Figures 3a and 3b) to the outputs (Figures 3d and 3c), there are fewer points between the 'arms' of the outputs.

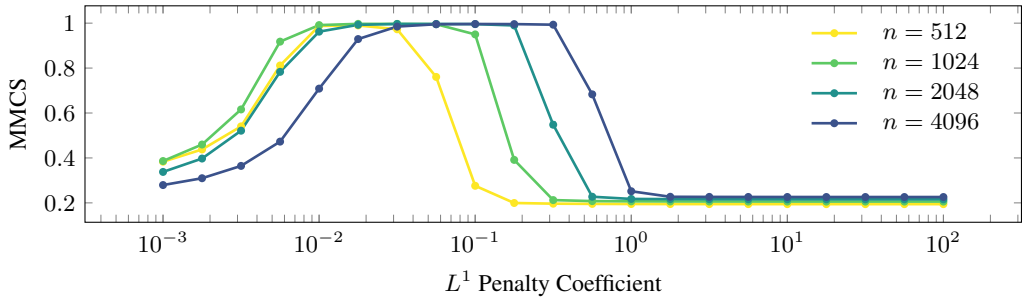


Figure 4: The mean max cosine similarity (MMCS) between the features learned by a standard SAE (decoder weight vectors) and the data-generating features against the L^1 penalty coefficient in the training loss, following Sharkey et al. (2022). There is a 'Goldilocks zone' where SAEs near-perfectly recover the data-generating features, given enough latents to represent them.

4.2 Do random NNs preserve or amplify superposition?

We investigated this suggestion by generating toy data with the same procedure as Sharkey et al. (2022), i.e., sampling ground-truth features on a hypersphere and generating correlated feature coefficients such that only a small number are active (Appendix I.1). We then passed these inputs to a two-layer MLP at initialization and trained SAEs on both the inputs and outputs individually. As a control, we used Gaussian-distributed inputs with a mean and standard deviation equal to the superposed toy data. We used standard SAEs with an L^1 sparsity penalty (Appendix I.2).

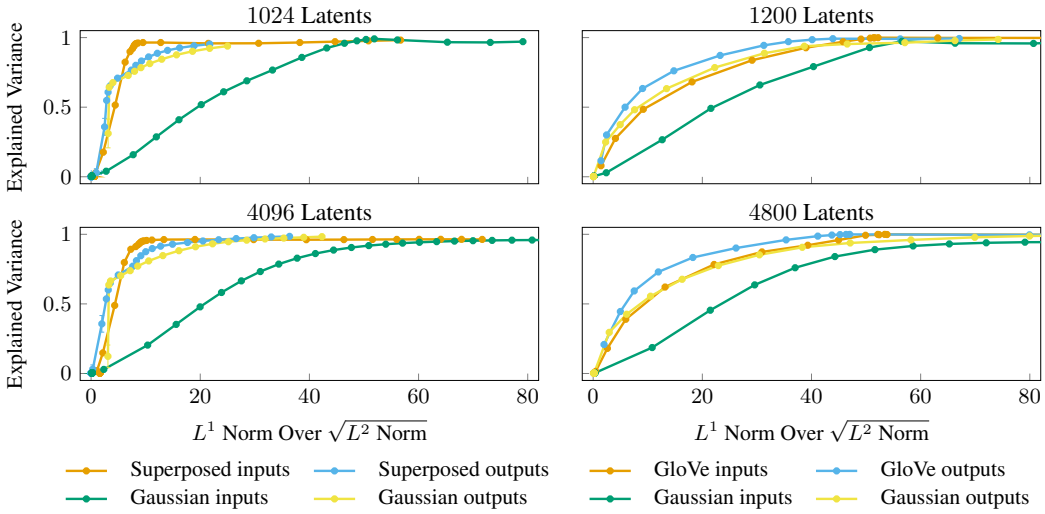
Following Sharkey et al. (2022), we confirmed that SAEs can recover the ground-truth features that generated the data (Figure 4). In particular, we measured the mean max cosine similarity (MMCS): for every data-generating feature, we found its maximum cosine similarity with the features learned by the SAE (its decoder weight vectors) and took the average over data-generating features. However, the MMCS only applies to the MLP inputs, where we have access to the data-generating features – a different approach is required to analyze the MLP outputs. To this end, we took the ability of SAEs to achieve low reconstruction error with high sparsity as a proxy for the degree to which the training data exhibits superposition. Specifically, we vary the L^1 penalty coefficient to obtain Pareto frontiers of the explained variance against sparsity measures (Figure 5a).

As expected, we found that SAEs achieved much greater sparsity at a given level of explained variance for the superposed inputs relative to the Gaussian control (Figure 5a; orange and blue-green). Interestingly, the difference between the superposed outputs, i.e., the outputs of the MLP given the superposed inputs, and the Gaussian outputs is much smaller, with only slightly greater sparsity at a given level of explained variance. This suggests that the outputs of randomly initialized MLPs have a relatively high level of sparsity insensitive to the input distribution. We consider other sparsity measures and hyperparameters in Appendix I.

4.3 Do token embeddings exhibit superposition?

To the extent that randomly initialized neural networks preserve or amplify superposition, our results (Section 3) could be explained by the degree to which the inputs to transformer language models exhibit superposition. We study this question by applying the procedure described in Section 4.2 to language data. In particular, we train SAEs on pre-trained GloVe word vectors, the embedding matrices of Pythia models, the results of passing these inputs to a randomly initialized two-layer MLP, and Gaussian controls. The setup is unchanged from Section 4.2, except that the number of data points is fixed by the number of word embeddings or tokens, and we use a single random seed.

We find that the gap between the Pareto frontiers of the GloVe word vectors and the corresponding Gaussian controls (Figure 5b) is smaller than that observed for the toy superposed datasets described in Section 4.2 (Figure 5a). More interestingly, we again see that the Pareto frontiers for both inputs improve when they are passed to a randomly initialized two-layer MLP, emphasizing the possibility that random NNs 'sparsify' their inputs (i.e., increase the degree of apparent superposition).



(a) Toy datasets following Sharkey et al. (2022). (b) 300-dim. GloVe vectors (Pennington et al., 2014).

Figure 5: Pareto frontiers of the explained variance against the L^1 norm divided by the square root of the L^2 norm (sparsity) for datasets perhaps exhibiting superposition, Gaussian controls with the same mean and variance, and the corresponding outputs when these are passed to a randomly initialized two-layer MLP (Section 4.2. Each point denotes a choice of L^1 penalty coefficient.

5 Limitations

In this work, we demonstrate that auto-interpretability measures can produce apparently meaningful, interpretable results for SAEs trained on randomly initialized models, which are unlikely to exhibit computationally interesting features. Given the impossibility of testing across all datasets and model architectures, we strategically focused on the Pythia family of models, widely adopted in mechanistic interpretability research (e.g., Paulo and Belrose, 2025; Ghilardi et al., 2025; Mueller, 2024), and the RedPajama-V2 dataset, representing typical pre-training data for language models and SAEs.

While we used the default model for generating explanations in the EleutherAI auto-interpretability framework (Caden et al., 2025), exploring alternative models could yield valuable insights into aggregate behaviors and the quality of generated explanations. Importantly, we do not claim that SAEs fail to capture information from trained Transformers above and beyond randomly initialized transformers; only that aggregate auto-interpretability measures do not necessarily indicate the existence of interesting underlying features.

6 Conclusion

In this work, we applied sparse autoencoders to both trained and randomly initialized transformers and evaluated them with a suite of common quantitative metrics. Our central empirical finding is that, under certain conditions, these metrics – particularly aggregate auto-interpretability scores – can be surprisingly similar in both settings. While we observe that features derived from trained transformers are qualitatively more complex and abstract, especially in later layers, these aggregate metrics often fail to capture this distinction.

This result does not imply that SAEs trained on real models fail to learn meaningful computational features. Rather, it reveals a limitation in our current evaluation methods. High aggregate auto-interpretability scores are insufficient proof for the discovery of complex, learned computations: they may instead reflect simpler structure inherent in the data or model architecture that is preserved even by random weights. Our analysis of token distribution entropy, while preliminary, serves as a proof-of-concept: it successfully revealed differences in feature 'abstractness' that aggregate auto-interpretability scores missed. Future work should focus on developing more robust metrics that can quantify the computational significance of the features SAEs discover. Our work reaffirms