AUTOMATED INTERPRETABILITY METRICS DO NOT DISTINGUISH TRAINED AND RANDOM TRANSFORMERS

Thomas Heap
University of Bristol
Bristol, UK
thomas.heap@bristol.ac.uk

Tim Lawson
University of Bristol
Bristol, UK

Lucy Farnik
University of Bristol
Bristol, UK

Laurence Aitchison
University of Bristol
Bristol, UK

ABSTRACT

Sparse autoencoders (SAEs) are widely used to extract sparse, interpretable latents from transformer activations. We test whether commonly used SAE quality metrics and automatic explanation pipelines can distinguish trained transformers from randomly initialized ones (e.g., where parameters are sampled i.i.d. from a Gaussian). Over a wide range of Pythia model sizes and multiple randomization schemes, we find that, in many settings, SAEs trained on randomly initialized transformers produce auto-interpretability scores and reconstruction metrics that are similar to those from trained models. These results show that high aggregate auto-interpretability scores do not, by themselves, guarantee that learned, computationally relevant features have been recovered. We therefore recommend treating common SAE metrics as useful but insufficient proxies for mechanistic interpretability and argue for routine randomized baselines and targeted measures of feature 'abstractness.'

1 INTRODUCTION

Sparse autoencoders (SAEs) are a popular tool in mechanistic interpretability research, with the aim of disentangling the internal representations of neural networks by learning sparse, interpretable features from network activations (Elhage et al., 2022; Sharkey et al., 2022; Cunningham et al., 2023; Bricken et al., 2023). An autoencoder with a high-dimensional hidden layer is trained to reconstruct activations while enforcing sparsity (Gao et al., 2024; Templeton et al., 2024; Lieberum et al., 2024), with the aim of discovering the underlying concepts or 'features' learned by the network (Park et al., 2023; Wattenberg and Viégas, 2024). Developing better SAEs relies on quantitative evaluation metrics like auto-interpretability scores that measure agreement between generated explanations and activation patterns (Bills et al., 2023; Paulo et al., 2024; Karvonen et al., 2024a).

For an interpretability method to be considered robust, its evaluation metrics should distinguish features learned through training from artifacts arising from the data or model architecture. A key sanity check is therefore to compare the method's output on a trained model against a strong null model, such as one with randomly initialized weights (Adebayo et al., 2020). We apply this sanity check to SAEs and find that several common quantitative metrics do not always clearly distinguish between the trained and randomized settings. In particular, we found that SAEs trained on transformers with random parameters can yield latents with auto-interpretability scores (Bills et al., 2023; Paulo et al., 2024) that are surprisingly similar to those from a fully trained model.

This result raises important questions about what we can glean from applying these metrics of SAE quality. High auto-interpretability scores alone do not guarantee that an SAE has identified complex, learned computations. Instead, such scores may sometimes reflect simpler statistical properties of the training data (Dooms and Wilhelm, 2024) or architectural inductive biases that are present even without training. Indeed, one could argue that a randomly initialized network still performs a basic form of computation, such as preserving or amplifying the sparse structure of its inputs (Section 4). From this perspective, SAEs might faithfully interpret this simple, inherent computation.

While some SAE features from trained models clearly arise from learned computation, the commonly used aggregate metrics are often insufficient for determining whether a given SAE has learned these more complex features. These results have important implications for mechanistic interpretability research. In particular, we suggest that more rigorous methods to distinguish between artifacts and genuinely learned computations are needed, and that interpretability techniques should be carefully validated against appropriate null models.

Finally, we speculate about why these patterns might emerge. At a high level, there are two hypotheses: (1) the input data already exhibits superposition, and randomly initialized neural networks largely preserve this superposition; and (2) randomly initialized neural networks amplify or even introduce superposed structure to the input data (e.g., given dense input generated i.i.d. from a Gaussian). We present toy models to demonstrate the plausibility of these hypotheses in Section 4 but defer conclusions as to the mechanism responsible to future work.

2 Related Work

Sparse dictionary learning Under a different name, 'superposition' in visual data is one of the foundational observations of computational neuroscience. Olshausen and Field (1996; 1997) showed that the receptive fields of simple cells in the mammalian visual cortex can be explained as a result of sparse coding, i.e., representing a relatively large number of signals (sensory information) by simultaneously activating a small number of elements (neurons). Coding theory offers a perspective on efforts to extract the 'underlying signals' responsible for neural network activations (Marshall and Kirchner, 2024).

Sparse dictionary learning (SDL) approximates a set of input vectors by linear combinations of a relatively small number of learned basis vectors. The learned basis is usually overcomplete: it has a greater dimension than the inputs. SDL algorithms include Independent Component Analysis (ICA), which finds a linear representation of the data such that the components are maximally statistically independent (Bell and Sejnowski, 1995; Hyvärinen and Oja, 2000). Sparse autoencoders (SAEs) are a simple neural network approach (Lee et al., 2006; Ng, 2011; Makhzani and Frey, 2014). Typically, an autoencoder with a single hidden layer that is many times larger than the input activation vectors is trained with an objective that imposes or incentivizes sparsity in its hidden layer activations to try to find this structure. A **latent** is a single neuron (dimension) in the autoencoder's hidden layer.

Mechanistic interpretability Recently, it has become common to understand 'features' or concepts in language models as low-dimensional subspaces of internal model activations (Park et al., 2023; Wattenberg and Viégas, 2024; Engels et al., 2024). If such sparse or 'superposed' structure exists, we expect to be able to 'intervene on' or 'steer' the activations, i.e., to modify or replace them to express different concepts and so influence model behavior (Meng et al., 2022; Zhang and Nanda, 2023; Heimersheim and Nanda, 2024; Makelov, 2024; O'Brien et al., 2024).

SAEs are a popular approach for discovering features, where one typically trains a single autoencoder to reconstruct the activations of a single neural network layer, e.g., the transformer residual stream (Sharkey et al., 2022; Cunningham et al., 2023; Bricken et al., 2023). Many SAE architectures have been suggested, which commonly vary the activation function applied after the linear encoder (Makhzani and Frey, 2014; Gao et al., 2024; Rajamanoharan et al., 2024b; Lieberum et al., 2024). SAEs have also been trained with different objectives (Braun et al., 2024; Farnik et al., 2025) and applied to multiple layers simultaneously (Yun et al., 2021; Lawson et al., 2024; Lindsey et al., 2024).

Besides reconstruction errors and preservation of the underlying model's performance, SAEs have been evaluated according to whether they capture specific concepts (Gurnee et al., 2023; Gao et al., 2024) or factual knowledge (Huang et al., 2024; Chaudhary and Geiger, 2024), and whether these can be used to 'unlearn' concepts (Karvonen et al., 2024b).

Automatic neuron description SAEs often learn tens of thousands of latents, which are infeasible to describe by hand. Yun et al. (2021) find the tokens that maximally activate a dictionary element from a text dataset and manually inspect activation patterns. Instead, researchers typically collect latent activation patterns over a text dataset and prompt a large language model to explain them (e.g. Bills et al., 2023; Foote et al., 2023). These methods have been widely adopted (e.g. Cunningham et al., 2023; Bricken et al., 2023; Gao et al., 2024; Templeton et al., 2024; Lieberum et al., 2024).

Bills et al. (2023) generate an explanation for the activation patterns of a language-model neuron over examples from a dataset, simulate the patterns based on the explanation, and score the explanation by comparing the observed and simulated activations. This method is commonly known as auto-interpretability (as in self-interpreting). Paulo et al. (2024) introduce classification-based measures of the fidelity of automatic descriptions that are inexpensive to compute relative to simulating activation patterns and an open-source pipeline to compute these measures. Choi et al. (2024) use best-of-k sampling to generate multiple explanations based on different subsets of the examples that maximally activate a neuron. Importantly, they fine-tune Llama-3.1-8B-Instruct on the top-scoring explanations to obtain inexpensive 'explainer' and 'simulator' models.

Polysemanticity Lecomte et al. (2024) noted that neurons may become polysemantic incidentally. A polysemantic neuron (basis dimension) of a network layer represents multiple interpretable concepts (Elhage et al., 2022; Scherlis et al., 2023); unsurprisingly, individual neurons in a randomly initialized network may be polysemantic. By contrast, our work studies *superposition* (Elhage et al., 2022; Chan, 2024), which pertains to the representations learned across a whole network layer as opposed to any individual neuron. In particular, superposition allows a network layer as a whole to represent a larger number of (sparse) features than the layer has (dense) neurons by sparse coding (only a few concepts are active at a time, i.e., a given token position).

Training only the embeddings Zhong and Andreas (2024) showed that transformers learn surprising algorithmic capabilities when only the embeddings are trained and no other parameters. These results demonstrate that the behavior of a randomly initialized transformer can be shaped to a surprising extent by training only a few parameters. However, our setting is very different: besides considering SAEs, we randomize *all* the parameters, including the embeddings, in our 'Step-0' and 'Re-randomized incl. embeddings' variants. Our 'Re-randomized excl. embeddings' variant uses pre-trained embeddings, but we do not train those embeddings with fixed, randomized weights. Instead, we freeze the pre-trained embeddings and randomize the other weights (Section 3).

Random transformers for board games Karvonen et al. (2024c) found that SAEs were considerably better at extracting meaningful structure from chess games using pre-trained transformers, as opposed to those with random weights. However, the data from board games is wildly different from language data. In particular, there is reason to expect that language is sparse (e.g., a particular concept such as 'serendipitous' appears only rarely), and that this sparse structure is 'aligned' with conceptual meaning. In contrast, in board games, this is not necessarily true: a useful concept such as a knight fork does not necessarily turn up sparsely in board games.

Random one-layer transformers Bricken et al. (2023) found that auto-interpretability scores discriminated effectively between random and trained one-layer transformers. Similarly, we found that auto-interpretability scores for randomized models were relatively low for smaller models (e.g., Pythia-70m) but that the gap was narrowed for larger models (e.g., Pythia-6.9b).

3 RESULTS

We trained per-layer SAEs on the residual stream activation vectors of transformer language models from the Pythia suite, with between 70M and 7B parameters (Biderman et al., 2023). We compared SAEs trained on different variants of the underlying transformers:

- **Trained:** The usual, trained model.
- **Re-randomized incl. embeddings:** All the model parameters, including the embeddings, are re-initialized by sampling Gaussian noise with mean and variance equal to the values for each of the original, trained weight matrices.
- **Re-randomized excl. embeddings:** As above, except the embedding and unembedding weight matrices are not re-initialized, i.e., are the same as the original, trained model.
- **Step-0:** For Pythia models, the `step0` revisions are available, which are the original model weights at initialization, i.e., before any learning (Biderman et al., 2023).
- **Control:** The original, trained model, except where the input token embeddings are replaced at inference time by sampling i.i.d. standard Gaussian noise for each token, such that