RESIDUAL STREAM ANALYSIS WITH MULTI-LAYER SAES

Tim Lawson* **Lucy Farnik** **Conor Houghton** **Laurence Aitchison**
School of Engineering Mathematics and Technology
University of Bristol
Bristol, UK

ABSTRACT

Sparse autoencoders (SAEs) are a promising approach to interpreting the internal representations of transformer language models. However, SAEs are usually trained separately on each transformer layer, making it difficult to use them to study how information flows across layers. To solve this problem, we introduce the multi-layer SAE (MLSAE): a single SAE trained on the residual stream activation vectors from every transformer layer. Given that the residual stream is understood to preserve information across layers, we expected MLSAE latents to 'switch on' at a token position and remain active at later layers. Interestingly, we find that individual latents are often active at a single layer for a given token or prompt, but the layer at which an individual latent is active may differ for different tokens or prompts. We quantify these phenomena by defining a distribution over layers and considering its variance. We find that the variance of the distributions of latent activations over layers is about two orders of magnitude greater when aggregating over tokens compared with a single token. For larger underlying models, the degree to which latents are active at multiple layers increases, which is consistent with the fact that the residual stream activation vectors at adjacent layers become more similar. Finally, we relax the assumption that the residual stream basis is the same at every layer by applying pre-trained tuned-lens transformations, but our findings remain qualitatively similar. Our results represent a new approach to understanding how representations change as they flow through transformers. We release our code to train and analyze MLSAEs at `https://github.com/tim-lawson/mlsae`.

1 INTRODUCTION

Sparse autoencoders (SAEs) learn interpretable directions or 'features' in the representation spaces of language models (Elhage et al., 2022; Cunningham et al., 2023; Bricken et al., 2023). Typically, SAEs are trained on the activation vectors from a single model layer (Gao et al., 2024; Templeton et al., 2024; Lieberum et al., 2024). This approach illuminates the representations within a layer. However, Olah (2024); Templeton et al. (2024) believe that models may encode meaningful concepts by simultaneous activations in multiple layers, which SAEs trained at a single layer do not address. Furthermore, it is not straightforward to automatically identify correspondences between features from SAEs trained at different layers, which may complicate circuit analysis (e.g. He et al., 2024).

To solve this problem, we take inspiration from the residual stream perspective, which states that transformers (Vaswani et al., 2017) selectively write information to and read information from token positions with self-attention and MLP layers (Elhage et al., 2021; Ferrando et al., 2024). The results of subsequent circuit analyses, like the explanation of the indirect object identification task presented by Wang et al. (2022), support this viewpoint and cause us to expect the activation vectors at adjacent layers in the residual stream to be relatively similar (Lad et al., 2024).

To capture the structure shared between layers in the residual stream, we introduce the multi-layer SAE (MLSAE): a single SAE trained on the residual stream activation vectors from every layer of a transformer language model. Importantly, the autoencoder itself has a single hidden layer – it is

*Correspondence to `tim.lawson@bristol.ac.uk`.

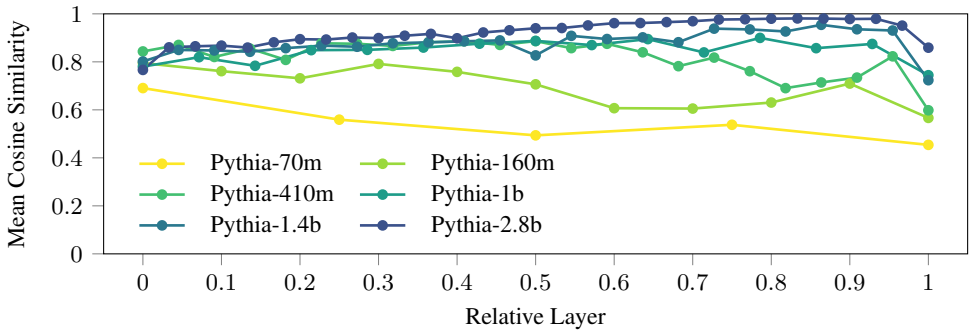

Figure 1: The mean cosine similarities between the residual stream activation vectors at adjacent layers of transformers, over 10 million tokens from the test set. To compare transformers with different numbers of layers, we divide the lower of each pair of adjacent layers by the number of pairs. This 'relative layer' is the x-axis of the plot. We subtract the dataset mean from the activation vectors at each layer before computing cosine similarities to control for changes in the norm between layers (Heimersheim & Turner, 2023), which we demonstrate in Figure 4.

multi-layer only in the sense that it is trained on activations from multiple layers of the underlying transformer. In particular, we consider the activation vectors from each layer as separate training examples, which is equivalent to training a single SAE at each layer individually but with the parameters tied across layers. We briefly discuss alternative methods in Section 5.

We show that multi-layer SAEs achieve comparable reconstruction error and downstream loss to single-layer SAEs while allowing us to directly identify and analyze features that are active at multiple layers (Section 4.1). When aggregating over a large sample of tokens, we find that individual latents are likely to be active at multiple layers, and this measure increases with the number of latents. However, for a single token, latent activations are more likely to be isolated to a single layer. For larger underlying transformers, we show that the residual stream activation vectors at adjacent layers are more similar and that the degree to which latents are active at multiple layers increases.

Finally, we relax the assumption that the residual stream basis is the same at every layer by applying pre-trained tuned-lens transformations to activation vectors before passing them to the encoder. Surprisingly, this does not obviously increase the extent of multi-layer latent activations.

2 Related work

A sparse code represents many signals, such as sensory inputs, by simultaneously activating a relatively small number of elements, such as neurons (Olshausen & Field, 1996; Bell & Sejnowski, 1997). Sparse dictionary learning (SDL) approximates each input vector by a linear combination of a relatively small number of learned basis vectors. The learned basis is usually overcomplete: it has a greater dimension than the inputs. Independent Component Analysis (ICA) achieves this aim by maximizing the statistical independence of the learned basis vectors by iterative optimization or training (Bell & Sejnowski, 1995; 1997; Hyvärinen & Oja, 2000; Le et al., 2011). Sparse autoencoders (SAEs) can be understood as ICA with the addition of a noise model optimized by gradient descent (Lee et al., 2006; Ng, 2011; Makhzani & Frey, 2014)

The activations of language models have been hypothesized to be a dense, compressed version of a sparse, expanded representation space (Elhage et al., 2021; 2022). Under this view, there are interpretable directions in the dense representation spaces corresponding to distinct semantic concepts, whereas their basis vectors (neurons) are 'polysemantic' (Park et al., 2023). It has been shown theoretically (Wright & Ma, 2022) and empirically (Elhage et al., 2022; Sharkey et al., 2022; Whittington et al., 2023) that SDL recovers ground-truth features in toy models, and that learned dictionary elements are more interpretable than the basis vectors of language models (Cunningham et al., 2023; Bricken et al., 2023) or dense embeddings (O'Neill et al., 2024). Notably, 'features' are not necessarily linear (Wattenberg & Viégas, 2024; Engels et al., 2024; Hernandez et al., 2024).

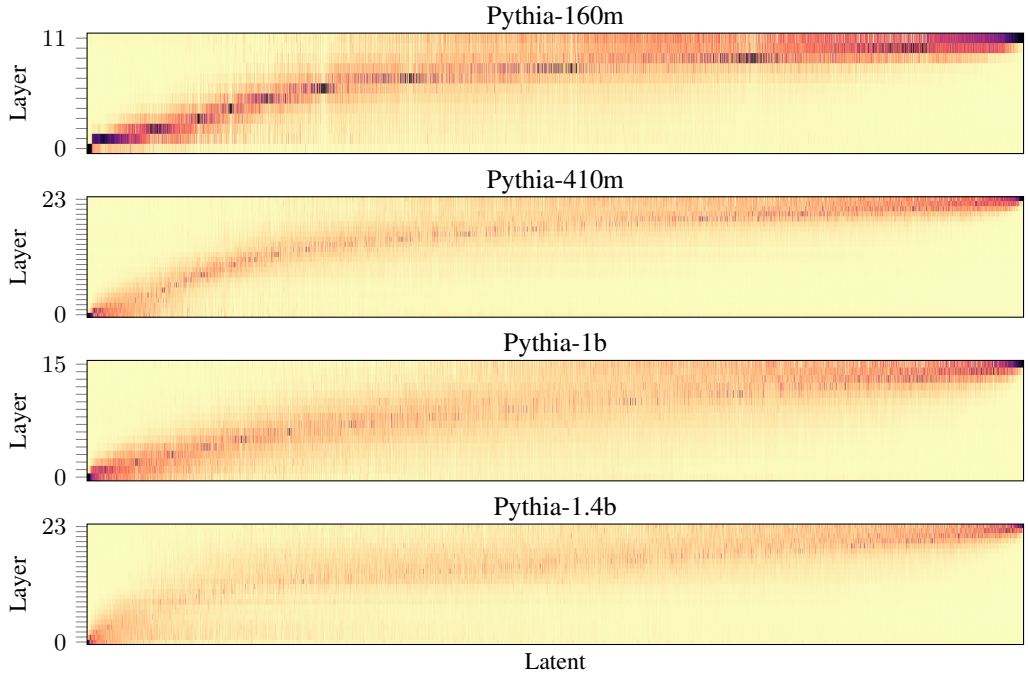

Figure 2: Heatmaps of the distributions of latent activations over layers when aggregating over 10 million tokens from the test set. Here, we plot the distributions for MLSAEs trained on Pythia models with an expansion factor of $R = 64$ and sparsity $k = 32$. The latents are sorted in ascending order of the expected value of the layer index (Eq. 10).

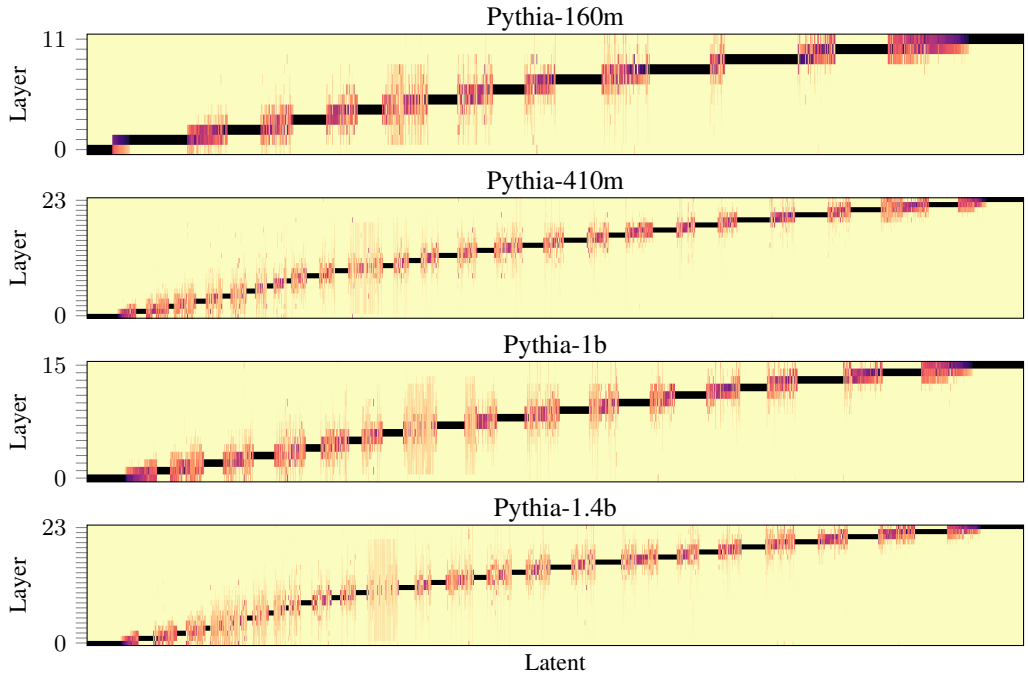

Figure 3: Heatmaps of the distributions of latent activations over layers for a single example prompt. Here, we plot the distributions for MLSAEs trained on Pythia models with an expansion factor of $R = 64$ and sparsity $k = 32$. The example prompt is "When John and Mary went to the store, John gave" (Wang et al., 2022). We exclude latents with maximum activation below 1×10^{-3} and sort latents in ascending order of the expected value of the layer index (Eq. 10).