A TRAINING

We train each autoencoder on 1 billion tokens from the Pile (Gao et al., 2020), excluding the copyrighted Books3 dataset,[3] for a single epoch. Specifically, we concatenate a batch of 1024 text samples with the end-of-sentence token, tokenize the concatenated text, and divide the output into sequences of 2048 tokens, discarding the final incomplete sequence. We use an effective batch size of 131072 tokens (64 sequences) for all experiments.

We do not compute activation vectors and cache them to disk before training, which minimizes storage overhead at the expense of repeated computation. We construct a batch of activation vectors to input to the autoencoder by performing the forward pass of the underlying transformer for a sequence of tokens, collecting the residual stream activation vectors at every layer, and stacking them together. Following Lieberum et al. (2024), we exclude activation vectors corresponding to special tokens (end-of-sentence, beginning-of-sentence, and padding). Hence, each batch has an equal number of activation vectors from each layer, which is the number of non-special tokens.

Following the optimization guidelines in Bricken et al. (2023); Gao et al. (2024), we initialize the pre-encoder bias $\mathbf{b}_{\text{pre}}$ to the geometric median of the first training batch; we initialize the decoder weight matrix $\mathbf{W}_{\text{dec}}$ to the transpose of the encoder $\mathbf{W}_{\text{enc}}$; we scale the decoder weight vectors to unit norm at initialization and after each training step; and we remove the component of the gradient of the decoder weight matrix parallel to its weight vectors after each training step.

We use the Adam optimizer (Kingma & Ba, 2017) with the default β parameters, a constant learning rate of 1×10^{-4}, and $\epsilon = 6.25 \times 10^{-10}$. Unlike Gao et al. (2024), we do not use gradient clipping or weight averaging, and we use FP16 mixed precision to reduce memory use.

B EVALUATION

B.1 RECONSTRUCTION ERROR AND SPARSITY

While we use the FVU instead of MSE as the reconstruction error in the training loss, we record both metrics for the inputs from each transformer layer and the mean over all layers (Figure 8). The L^0 norm of the latents is fixed at k per activation vector (layer and token), but we record the L^1 norm (Figure 9). We report the values of these metrics over one million tokens from the test set.

For Pythia-70m, the FVU at each layer is comparable to Marks et al. (2024, p. 21), who trained separate SAEs with $n = 32768$ and L^0 norms between 54 and 108, as well as Cunningham et al. (2023, p. 13). For Pythia-160m, the FVU is similar to Gao et al. (2024), who report the normalized MSE on layer 8 of GPT-2 small.

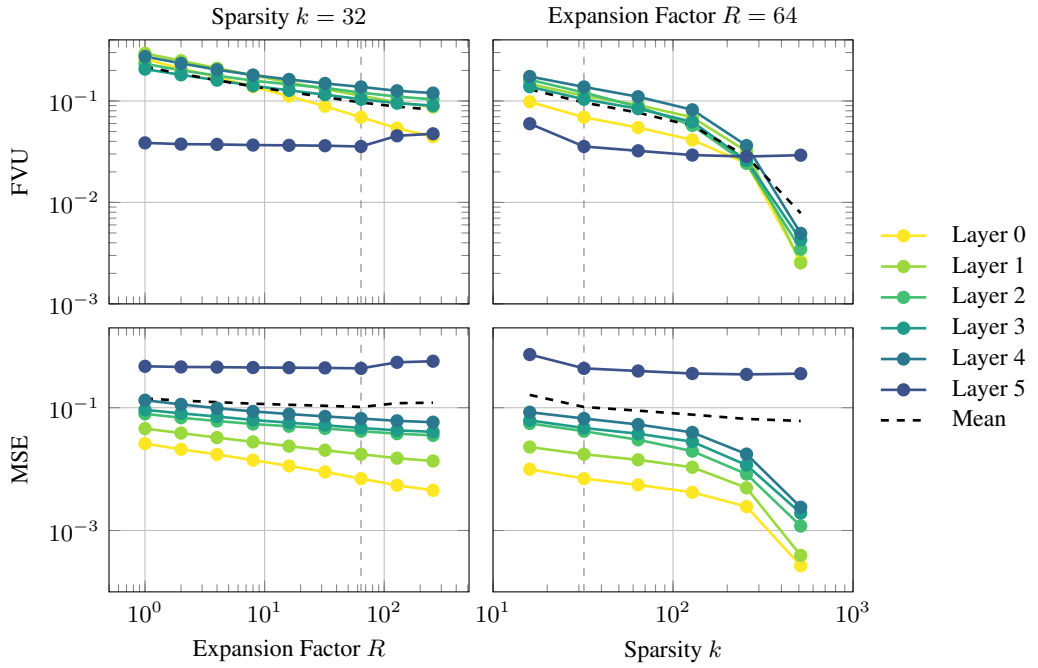
B.2 DOWNSTREAM LOSS

In addition to the increase in cross-entropy (CE) loss, we record the Kullback-Leibler (KL) divergence between probability distributions when the residual stream activations at a given layer are replaced by their reconstruction (Section 4.1). We report the values of these metrics over one million tokens from the test set (Figure 10). The increase in cross-entropy loss is comparable to Marks et al. (2024, p. 21) for Pythia-70m, Gao et al. (2024, p. 5) and Braun et al. (2024) for GPT-2 small, and Lieberum et al. (2024, p. 7-8) for layer 20 of Gemma 2 2B and 9B.

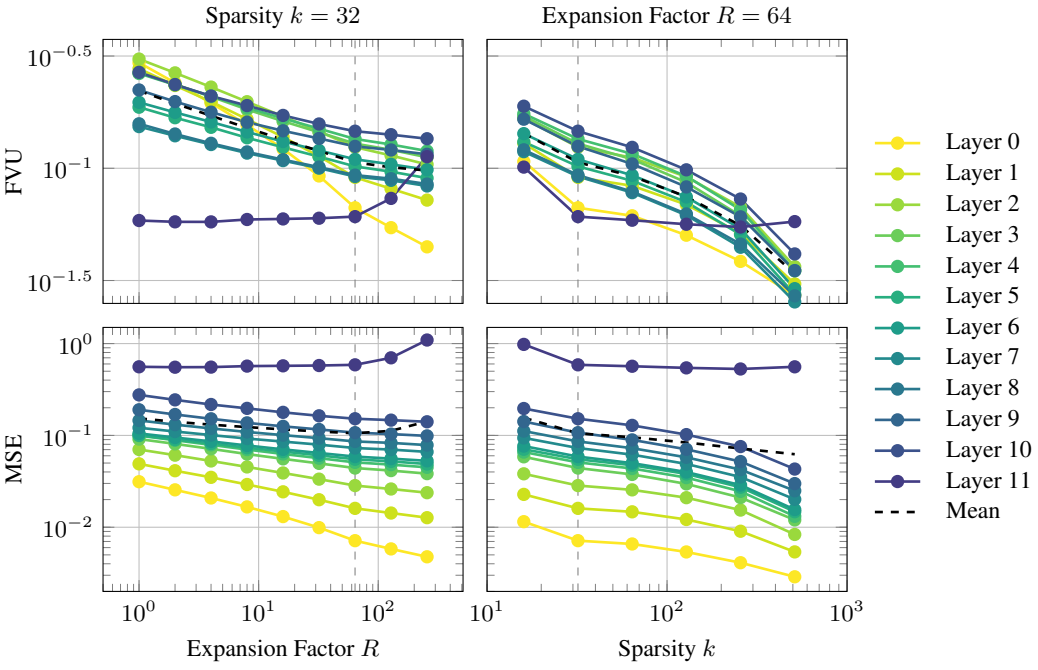
B.3 OTHER TRANSFORMER ARCHITECTURES

We predominantly study transformers from the Pythia suite because it affords a controlled setting to study how our results scale across model sizes (Biderman et al., 2023). While we do not expect our results to depend strongly on the underlying transformer architecture, we validated this assumption by training MLSAEs on Gemma 2 2B (Riviere et al., 2024), Llama 3.2 3B (Grattafiori et al., 2024), and GPT-2 small (Radford et al., 2019) with our default hyperparameters, i.e., an expansion factor of $R = 64$ and sparsity $k = 32$. We note that Gemma 2 2B and Llama 3.2 3B are similar in size to Pythia-2.8b, and GPT-2 small is similar in size to Pythia-160m.

[3] https://huggingface.co/datasets/monology/pile-uncopyrighted

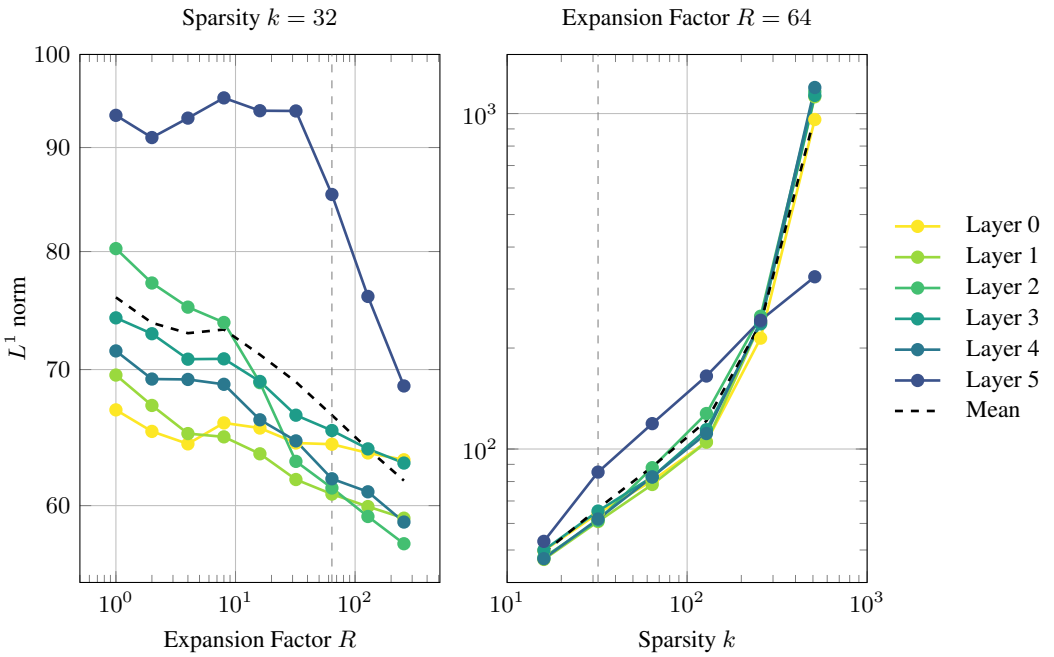


(a) Pythia-70m

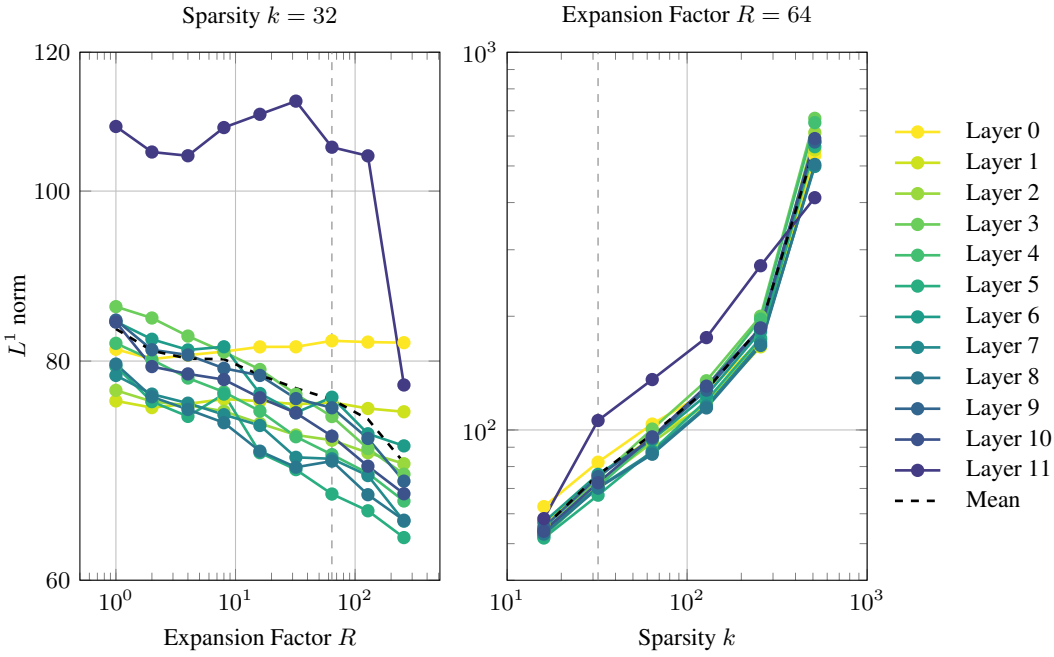


(b) Pythia-160m

Figure 8: With fixed sparsity $k = 32$, the FVU and MSE generally decrease as the expansion factor R increases. For inputs from the last layer, they increase for the largest expansion factors, which we attribute to fluctuations in the percentage of dead latents (Figure 11). With fixed expansion factor $R = 64$, the FVU and MSE decrease as the sparsity k increases. While all inputs are standardized before passing them to the encoder, the decoder outputs are rescaled afterward. Hence, the MSE increases across layers because it is not divided by the variance of the inputs.



(a) Pythia-70m



(b) Pythia-160m

Figure 9: With fixed sparsity $k = 32$, the L^1 norm per token (the sum of absolute activations) generally decreases as the expansion factor R increases. With fixed expansion factor $R = 64$, the L^1 norm increases as the sparsity k increases. Recall that the L^0 norm per token (the count of non-zero activations) is fixed at k.