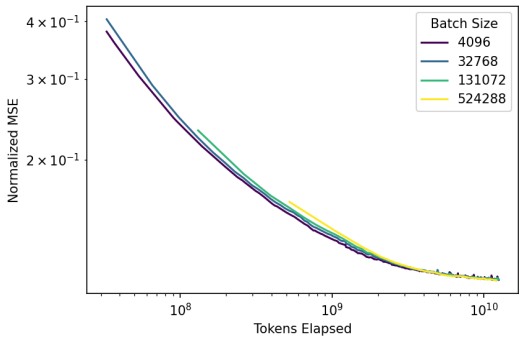


Figure 12: With correct hyperparameter settings, different batch sizes converge to the same $L(N)$ loss (gpt2small).

We use a bias-correction similar to that used in Kingma and Ba [2014]. Despite this, the early steps of EMA are still generally worse than the original model. Thus for the $L(C)$ experiments, we take the min of the EMA model's and non-averaged model's validation losses.

A.6 Other details

- For the main MSE loss, we compute an MSE normalization constant once at the beginning of training, and do not do any loss normalization per batch.
- For the AuxK MSE loss, we compute the normalization per token, because the scale of the error changes throughout training.
- In theory, the b_{pre} lr should be scaled linearly with the norm of the data to make the autoencoder completely invariant to input scale. In practice, we find it to tolerate an extremely wide range of values with little impact on quality.
- Anecdotally, we noticed that when decaying the learning rate of an autoencoder previously training at the L(C) loss, the number of dead latents would decrease.

B Other training details

Unless otherwise noted, autoencoders were trained on the residual activation directly after the layernorm (with layernorm weights folded into the attention weights), since this corresponds to how residual stream activations are used. This also causes importance of input vectors to be uniform, rather than weighted by norm[20].

B.1 TopK training details

We select k_{aux} as a power of two close to $\frac{d_{model}}{2}$ (e.g. 512 for GPT-2 small). We typically select $\alpha = 1/32$. We find that the training is generally not extremely sensitive to the choice of these hyperparameters.

We find empirically that using AuxK eliminates almost all dead latents by the end of training.

Unfortunately, because of compute constraints, we were unable to train our 16M latent autoencoder to $L(N)$, which made it not possible to include the 16M as part of a consistent $L(N)$ series.

B.2 Baseline hyperparameters

Baseline ReLU autoencoders were trained on GPT-2 small, layer 8. We sweep learning rate in [5e-5, 1e-4, 2e-4, 4e-4], L_1 coefficient in [1.7e-3, 3.1e-3, 5e-3, 1e-2, 1.7e-2] and train for 8 epochs of 6.4 billion tokens at a batch size of 131072. We try different resampling periods in [12.5k, 25k] steps,

[20]We believe one of the main impacts of normalizing is that it the first token positions are downweighted in importance (see Section F.2 for more discussion).

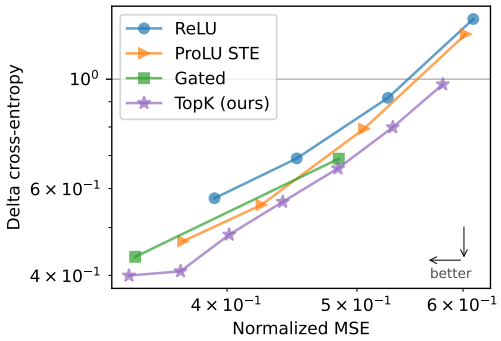
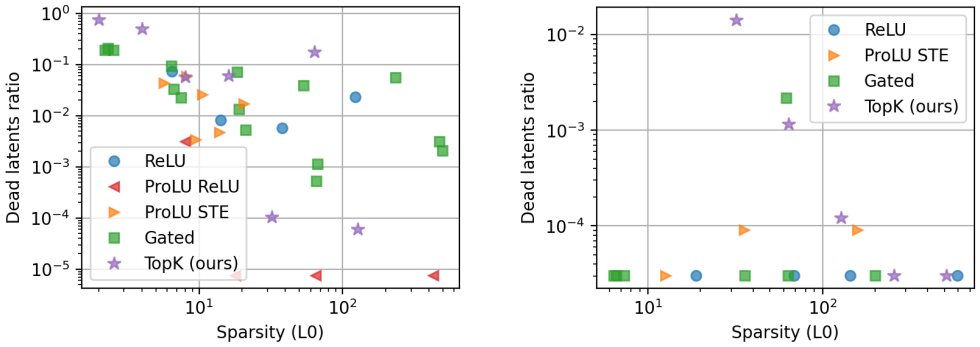


Figure 13: For a fixed sparsity level ($L_0 = 128$), a given MSE leads to a lower downstream-loss for TopK than for other activations functions. (GPT-4). We are less confident about these runs than the corresponding ones for GPT-2, yet the results are consistent (see Figure 5).



(a) Dead latent ratios for GPT-2

(b) Dead latent ratios for GPT-4

Figure 14: Our baselines generally have few dead latents, similar or less than our TopK models.

and choose to resample 4 times throughout training. We consider a feature dead if it does not activate for 10 million tokens.

For Gated SAE [Rajamanoharan et al., 2024], we sweep L_1 coefficient in [1e-3, 2.5e-3, 5e-3, 1e-2, 2e-2], learning rate in [2.5e-5, 5e-5, 1e-4], train for 6 epochs of 6.4 billion tokens at a batch size of 131072. We resample 4 times throughout training.

For ProLU autoencoders [Taggart, 2024], we sweep L_1 coefficient in [5e-4, 1e-3, 2.5e-3, 5e-3, 1e-2, 2e-2], learning rate in [2.5e-5, 5e-5, 1e-4], train for 6 epochs of 6.4 billion tokens at a batch size of 131072. We resample 4 times throughout training. For the ProLU gradient, we try both ProLU-STE and ProLU-ReLU. Note that, consistent with the original work, ProLU-STE autoencoders all have $L_0 < 25$, even for small L_1 coefficients.

We used similar settings and sweeps for autoencoders trained on GPT-4. Differences include: replacing the resampling of dead latents with a L_1 coefficient warm-up over 5% of training [Conerly et al., 2024]; removing the decoder unit-norm constraint and adding the decoder norm in the L_1 penalty [Conerly et al., 2024].

Our baselines generally have few dead latents, similar or less than our TopK models (see Figure 14).

C Training ablations

C.1 Dead latent prevention

We find that the reduction in dead latents is mostly due to a combination of the AuxK loss and the tied initialization scheme.

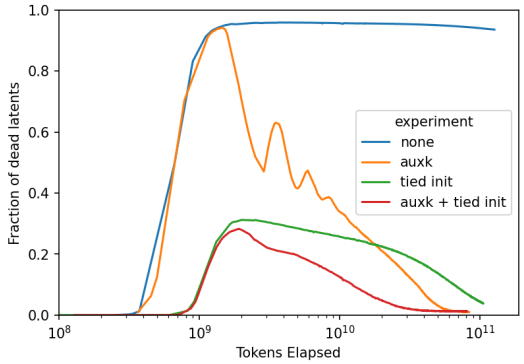


Figure 15: Methods that reduce the number of dead latents (gpt2sm 2M, k=32). With AuxK and/or tied initialization, number of dead latents generally decreases over the course of training, after an early spike.

C.2 Initialization

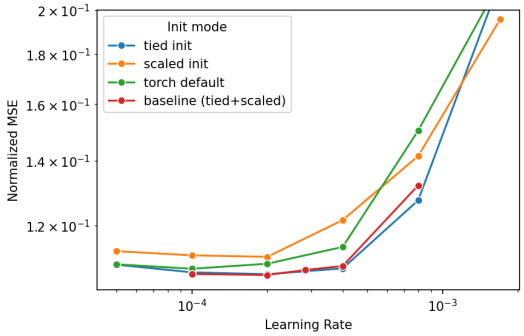


Figure 16: Initialization ablation (gpt2sm 128k, k=32).

We find that tied initialization substantially improves MSE, and that our encoder initialization scheme has no effect when tied initialization is being used, and hurts slightly on its own.

C.3 b_{enc}

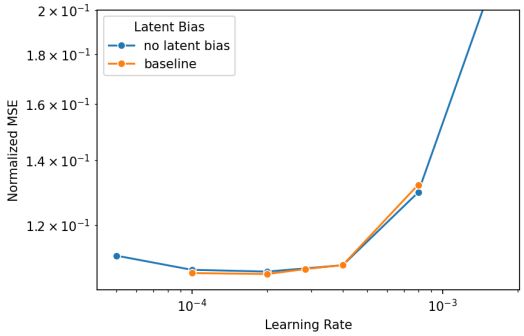


Figure 17: b_{enc} does not strongly affect loss (gpt2sm 128k, k=32).

We find that b_{enc} does not affect the MSE at convergence. With b_{enc} removed, the autencoder is equivalent to a JumpReLU where the threshold is dynamically chosen per example such that exactly k latents are active. However, convergence is slightly slower without b_{enc}. We believe this may be confounded by encoder learning rate but did not investigate this further.