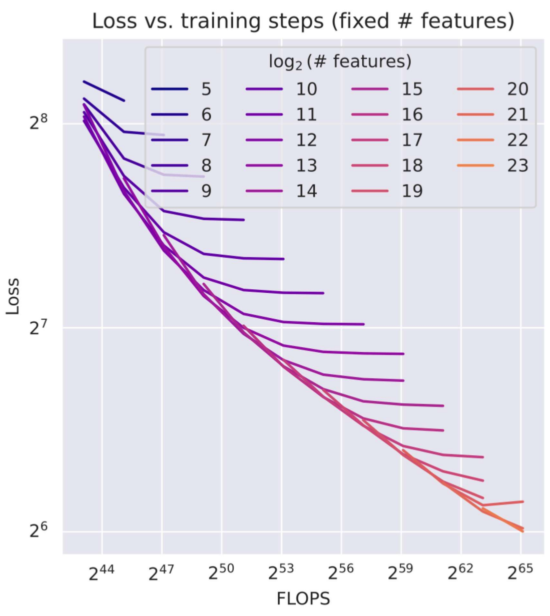
Scaling Laws for Dictionary Learning

Jack Lindsey, Tom Conerly, Adly Templeton, Jonathan Marcus, Tom Henighan

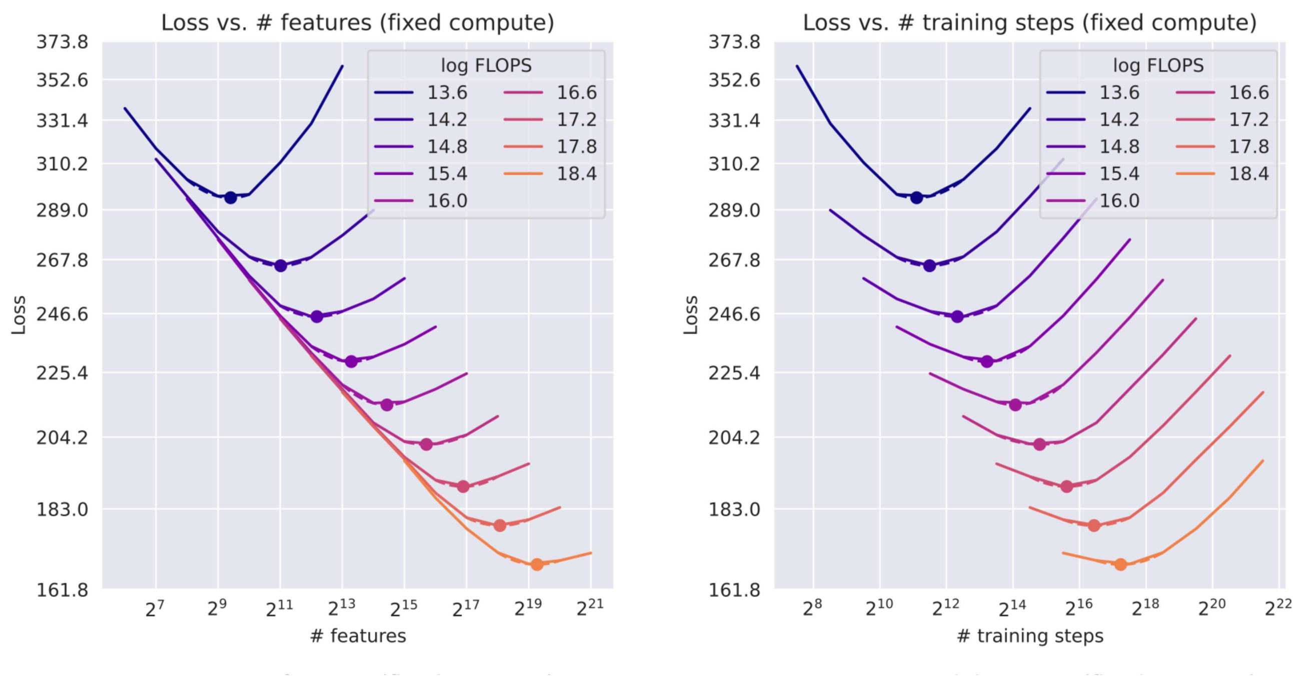
Training sparse autoencoders (SAEs) for dictionary learning on larger models can be computationally intensive. It is important to understand (1) the extent to which using additional compute improves dictionary learning results, and (2) how that compute should be allocated. Here we analyze these questions in depth. As a case study, we consider SAEs trained on the residual stream following the third layer of a four-layer transformer.

Though we lack a gold-standard method of assessing the quality of a dictionary learning run, we have found that the loss function used during training – a weighted combination of reconstruction mean-squared error (MSE) and an L1 penalty on feature activations – is a useful proxy. Unless otherwise indicated, we use $\mathrm{MSE} + 5\mathrm{L1}$ as our loss function during training and subsequent analysis. However, we find qualitatively similar results when we use other linear combinations of MSE and L1, or when we track linear combinations of MSE and the L0 "norm" of feature activations. We note that losses with different L1 coefficients are not comparable – ultimately, we select the L1 coefficients that produce the most useful features for downstream interpretability analyses.

Once we have chosen a loss function of interest, it allows us to treat dictionary learning as a standard machine learning problem, to which we can apply the "scaling laws" framework for hyperparameter optimization (*see e.g.* Kaplan *et al.* 2020, Hoffman *et al.* 2022). In an SAE, compute usage primarily depends on two key hyperparameters, the number of features being learned, and the number of steps used to train the autoencoder. The compute (in FLOPS) scales with the product of these parameters, if the input dimension and other hyperparameters are held constant. We conducted a thorough sweep over these parameters, fixing the values of other hyperparameters (learning rate, batch size, optimization protocol, etc.).

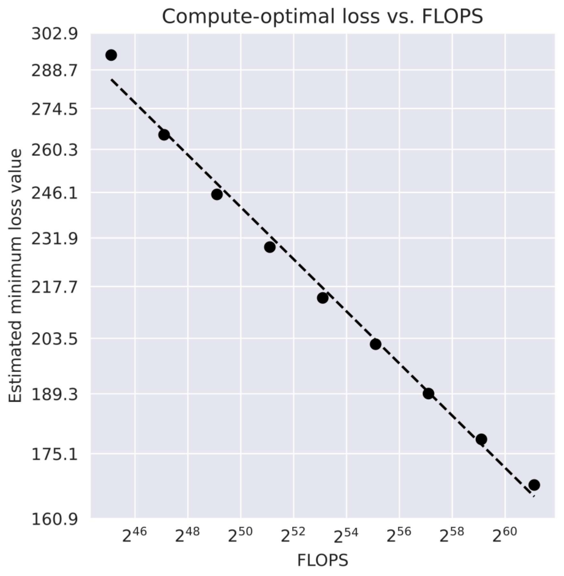


We are especially interested in keeping track of the compute-optimal values of the loss function and parameters of interest; that is, the lowest loss that can be achieved using a given number of FLOPS, and the number of training steps / features that achieve this minimum.

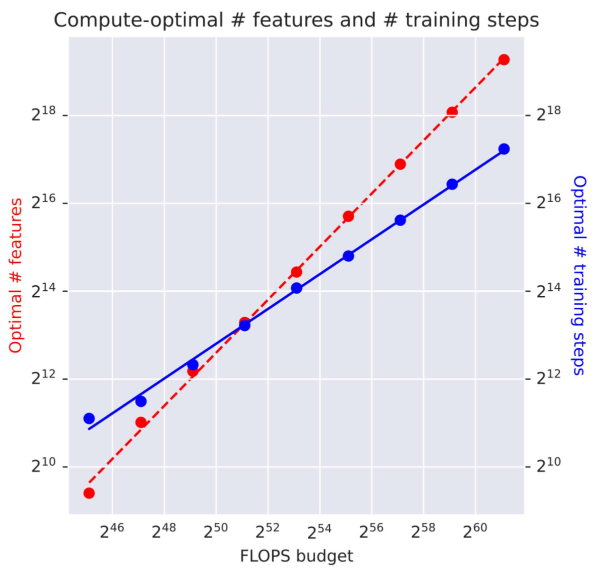


We have made the following observations:

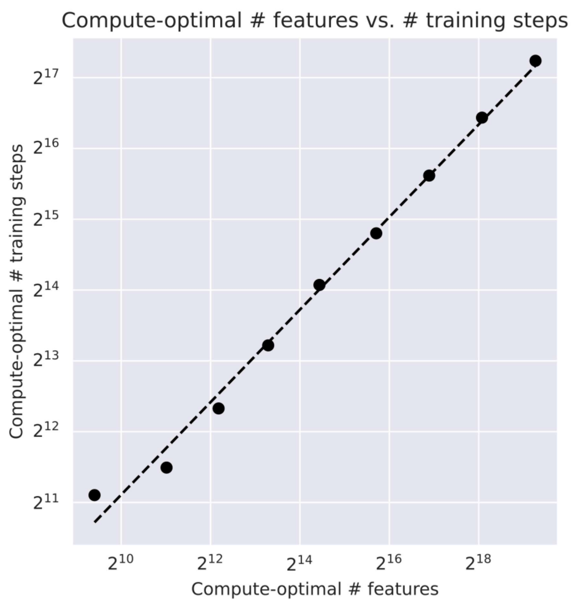
- Over the ranges we tested, loss functions decrease approximately according to a power law with respect to FLOPS, given the compute-optimal choice of training steps and number of features.



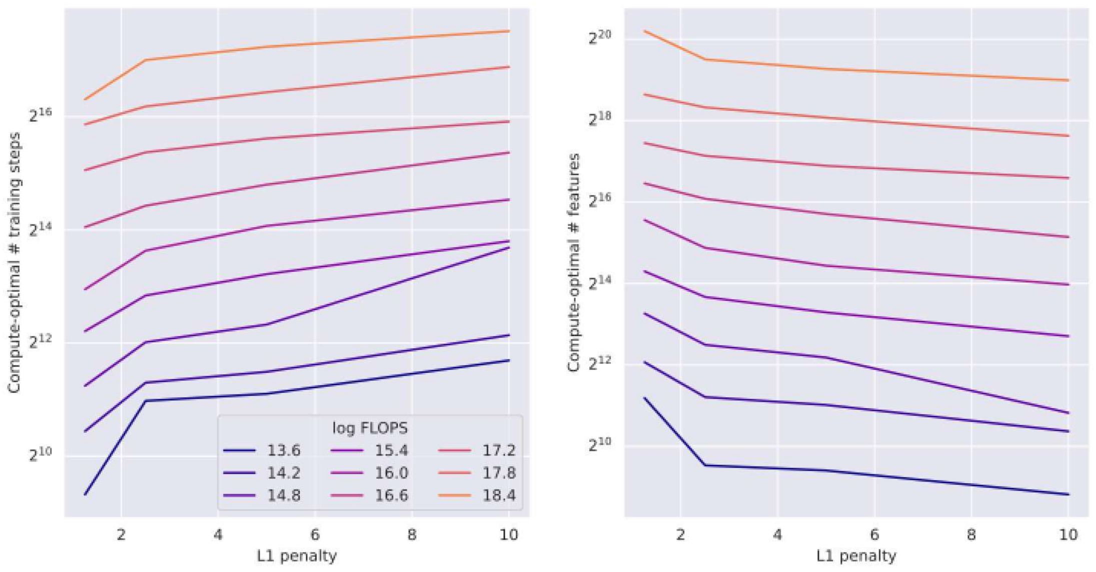
- As the compute budget increases, the optimal allocations of FLOPS to training steps and number of features both scale approximately as power laws.



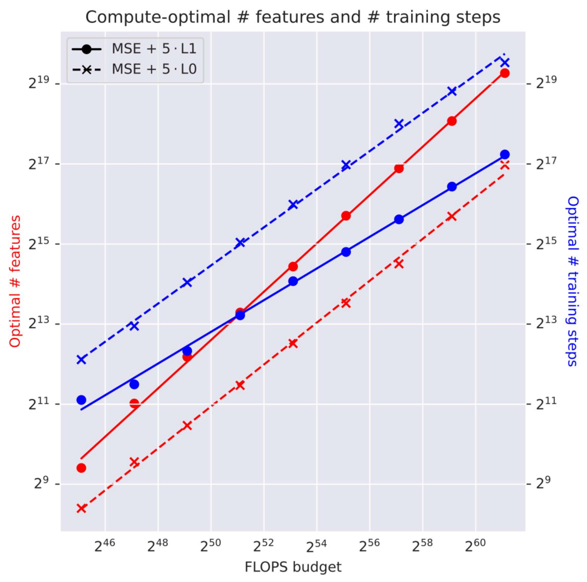
- In general, the optimal number of features appears to scale somewhat more quickly than the optimal number of training steps. For instance, as the number of features increases, the corresponding compute-optimal number of training steps scales as a function of the number of features with an exponent between 0.5 and 1. The specific parameters of the scaling trends vary between different loss functions.



- Emphasizing the sparsity penalty more in the loss function (i.e. increasing the L1 coefficient) leads to a larger number of training steps being compute-optimal.The compute-optimal number of training steps increases for loss functions that place a greater relative emphasis on the sparsity penalty. This suggests that reconstruction loss is optimized more quickly than sparsity over the course of SAE training, which we have observed empirically.



- In these experiments, we also investigate the parameters needed to optimize for L0-based loss functions (i.e. linear combinations of MSE and L0 “norm” of feature activations). Since these cannot be directly optimized with gradient descent, we instead sweep over a range of L1 penalty coefficients during training and select the value that minimizes the L0-based loss function. We find that minimizing L0-based loss functions requires a greater number of training steps for a given number of features, compared to minimizing L1-based loss functions. Though we have not precisely characterized the source of this difference, we suspect it arises because additional training steps allow the SAE to fully zero out small feature activations.



The details of these trends are likely to vary depending on the underlying model, the layer of the model being probed, and other optimization details. Optimizing other hyperparameters (such as learning rate) jointly with training steps and number of features may influence the scaling trends. However, we expect many of these qualitative trends to be broadly applicable. We suggest that conducting similar analyses will be useful to other groups working with SAEs, particularly as computational cost increases. Extrapolating trends inferred from smaller experiments enables more informed choices of hyperparameters for resource-intensive dictionary learning runs. We are also careful to note that qualitative inspection of SAE features remains important, as the relationship between SAE loss and qualitative usefulness of SAE features is imperfect and may break down at sufficient scale.