Understanding and comparing learned features in CLIP models via Sparse Autoencoders

Franklin Zhu
Stanford University
fzhu22@stanford.edu

Logan Graves
Stanford University
gravesl@stanford.edu

Abstract

Vision models learn complex semantic relationships within their training data, but it is not well-understood what concepts those models contain, nor how architectural differences influence the concepts a model can learn. In this work, we (plan to) train Sparse Autoencoders (SAEs) on a ResNet and a Vision Transformer which have been pretrained similarly, towards two ends: (1) understanding how SAEs behave in the context of Convolutional Neural Networks, as compared to previous work with Vision Transformers; and (2) undertaking a comparative analysis of human-interpretable concepts contained in similar visual networks with different architectures (CLIP-ResNet50 and CLIP-ViT-L/14). By training SAEs on these two similar but distinct networks and comparing what they learn, we aim to develop an understanding of how architectural differences are involved in the kinds of concepts a model can learn, advancing theoretical and practical understanding of these architectures.

1. Introduction

1.1. Related Work

Deep neural networks exhibit exceptional performance across many domains, but their internal representations (e.g. activations in latent layers) are in many cases opaque. To explain this phenomenon, previous work such as [4] has introduced the so-called *"superposition hypothesis"*, in brief that models want to learn more features than they have dimensions to represent, so they pack these features in "almost-orthogonal" directions in their high-dimensional latents. Thus, while individual dimensions in the latent spaces of a deep learning model may not be interpretable, with the right techniques this 'superposition' of features can be undone and understood.

To achieve this comprehension, one prominent approach has been the use of Sparse Autoencoders (SAEs). Sparse Autoencoders were largely popularized by research done for language model interpretability by the Anthropic team in [5], [2] and in the past two years have continued to receive significant attention. SAEs are autoencoders trained to reconstruct the latents of another model, but with a much larger ("overcomplete") latent space of their own and a strong sparsity penalty. Intuitively, by incentivizing most dimensions in the model to stay inactive, SAEs are more likely to learn to represent each superimposed feature in its own dimension, as opposed to spreading them between dimensions, thus undoing the superposition.

Though initially popularized in the context of language model interpretability, SAEs are exceptionally effective in vision as well. In [3], it was shown that sparse autoencoders trained on the vision transformer submodel in OpenAI's CLIP[1] produce robust, highly human-interpretable features. Here we replicate these results and extend them, training SAEs both on the vision transformer and the ResNet versions of the CLIP model, and undertaking a comparative analysis of features.

2. Dataset

We intend to use images from ImageNet-1k to get activations from the selected CLIP ViT and ResNet models, which are used as inputs to our SAEs. ImageNet-1k is a large-scale vision dataset that includes 1000 different classes of categories, encompassing animals, vehicles, machines, etc. It includes 1.28 million training images, 50,000 validation images, and 100,000 test images. Since the CLIP models were trained on extremely large-scale data, we hope to recover as many of its activations as possible by using as large of a dataset as possible with our limited compute.

With too small of a dataset, problems can arise with a large portion of the CLIP's activations being recoverable with a small number of neurons in the SAE, which oversimplifies the features the CLIP models are learning and fails to capture all the features, and therefore all the expressive capacity, of the origiinal model. It would be ideal to use a dataset similar in scale to the dataset used to train the CLIP models, but once again, we are limited in compute.

2.1. Pre-processing

In order to pass in the images from ImageNet-1k to our ResNet and Vit, we pre-process them using provided functions that come with each OpenAI CLIP model. These functions take care of scaling each image to the correct input size for each respective model, as well as cropping and image normalization.

2.2. Baseline

As a baseline, we plan to compare our CLIP ViT SAE to a model trained in [3]. For their SAE, they trained on ImageNet-1k, using 2,621,440 images with batch size 1024.

3. Methods

Our aim with this project is to develop two working SAEs, one trained on CLIP-ResNet50 and one on CLIP-ViT-Large14, and then to conduct a comparative analysis of the features learned from both.

3.1. SAE Architecture

In line with previous work[3], we set up an overcomplete linear SAE with input and output dimensions d (for the ViT CLIP, $d = 768$, and for ResNet50 CLIP $d = 1024$). Our overcomplete latent layer expansion factor of 16 yields a latent layer of dimension $16d$. We choose 16 for the expansion factor as opposed to 64 or 32 for multiple reasons. Previous work [3][5] discovered so-called "ultra-low-density clusters" of SAE features that fire extremely rarely, yet which make up a significant proportion of model features. The author of [3] described achieving relatively similar performance in an SAE with an expansion factor of 16 as opposed to 64, indicating to us that 16 is empirically viable as an expansion factor, reducing training costs significantly without sacrificing significant performance.

For loss we use MSE loss with strong L1 regularization, and train using the Adam optimizer. (We will elaborate further on our hyperparameters once we have experimented further; previous work [3] in the area used an L1 factor of 8e-5 and a learning rate of 0.0004 for a ViT; ResNet parameters may be significantly different.)

3.2. Hook layers

In order to train SAEs, one must choose a specific part of the model to 'hook', i.e. choose which activations, at which layer, to feed into the SAEs. The choice of hook layers is not straightforward, and involves multiple factors, among them computational cost and susceptibility of representations to SAE learning.

We chose the CLS token in layer 22/24 in the ViT (768 dimensions), in line with previous work; this is a sufficiently abstract layer to have useful representations, while later layers suffer from issues related to normalization. In the ResNet, we hook the output of the Attention Pool (1024 dimensions); though it is slightly more processed than the features we are comparing it to in the ViT, choosing an earlier layer would quadruple the computational cost with unclear gains.

3.3. SAE Evaluation metrics

Once we have trained our SAEs using Adam, there are a number of metrics we intend to use to evaluate our SAEs. In terms of performance, we obviously care about the loss; if our model cannot competently reconstruct its inputs, then its latent representations are not faithful.

Assuming we have low loss, we evaluate SAE sparsity using a number of metrics:

1. *Average activation.* By averaging across inputs, we can see the distribution of average activations which will give a general albeit limited idea of the distribution of activations in our SAE.

2. *Average fraction of near-zero activations.* Averaging across the input batch, we can measure the proportion of activations that is near zero for any given input (this should be high if the SAE latent dimension is sparse).

3. *Identifying dead units.* Sparse learning can cause some neurons in the latent layer to 'die' i.e. fire very rarely; this is undesirable. We can track the number of dead units visually by plotting activation heatmaps across a batch (dead units will show up as dark rows) and quantify it using a threshold (e.g. if the total activation across the batch is less than some threshold ϵ we consider it to be dead).

4. *Similarity with original model.* Using cosine similarity between MLP outputs from the CLIP ViT/ResNet and the SAE encoder vetors, we can compare the extent to which these SAE features are aligned with neurons in the original model.

5. *Qualitative human interpretability.* Lastly, in addition to quantitative metrics we can use qualitative observations to confirm that our sparse features are indeed corresponding to some important concept. For example, we can visualize the top 9 images that activate a certain feature and see if there is a conceptual theme among them as we expect there to be.

3.4. Feature Comparisons

To measure the similarity of features learned in the ViT SAE and the ResNet SAE, we use a number of metrics:

1. *Average top-k image similarity.* For each feature in the SAEs, save the indices of the top k activating images for those features. Then for each ResNet SAE feature, compute the average overlap each SAE feature

has with its most similar feature in the ViT. Since some or many features may be dead, we may limit the number of features across which we compute the average.

2. *Pairing.* By pairing each ViT feature with a corresponding ResNet feature based on the similarity of its top-k images, we can create a list of possible 'same feature' relationships; then we can compute the percent overlap between them.

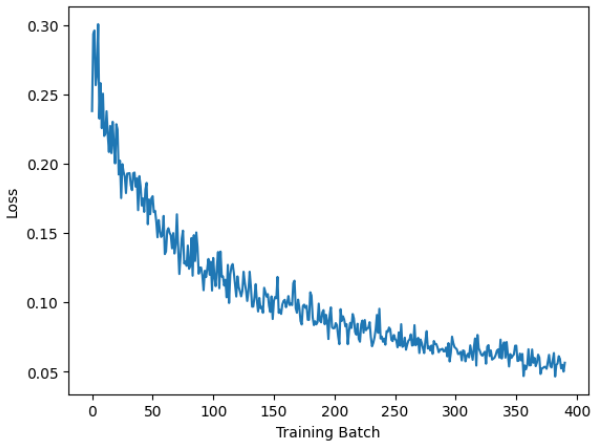
3. *Qualitative similarity.* By looking at visualizations of quantitatively similar features we can confirm or falsify the notion that these features are indeed similar in a human-interpretable way.

4. Results

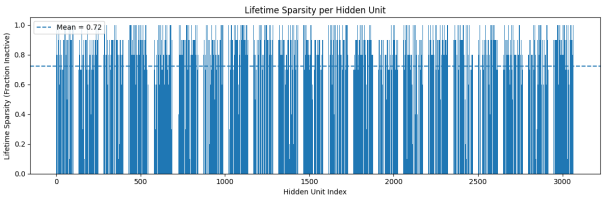
4.1. Preliminary Results

Though we have not had time yet to run a large training run as is required to train a performant SAE, we *have* set up the training loop and run a number of test runs.

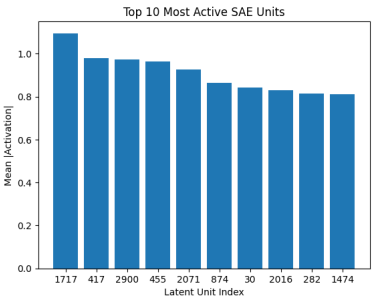
To ensure that our training is effective and our evaluations are functional, we trained a model with a very short run on CIFAR-10 (a much smaller dataset compared to ImageNet-1k). After confirming that the SAE is capable of overfitting a very small batch, we trained our SAE on one-eighth of the CIFAR-10 train set, achieving an average loss of 0.1 after one epoch. Though insubstantial in itself, this result indicates that our SAE setup functions as expected, in preparation for our larger training run.



Additionally, from this trained SAE, we were able to evaluate population and lifetime sparsity, which measure the proportion of active neurons on each image for the SAE. Population sparsity is specifically measured by calculating the per sample sparsity (proportion of active neurons) and then taking an average over the population. On the other hand, lifetime sparsity measures how often a specific neuron fires over across the entire training data. As a sanity check, the average lifetime sparsity is also equivalent to the population sparsity. For SAEs, the lower the sparsity the better, as this indicates that a low number of neurons are firing for each image, indicating that the neuron to feature mapping is 1 to 1. Our population sparsity was 0.724, which aligned with the average lifetime sparsity, and the individual lifetime sparsity is plotted below. The sparsity is rather high, but we trained on a very small proportion of the training data and for only a very small period of time, so these results were expected.



Then, we visualized the top 20 images with the highest activation scores, along with the neurons that are activated the most for those images. The plots are shown below.



4.2. Baseline Results

After a longer training run we intend to compare our model against results achieved in [3]. They were able to achieve a sparsity of around 0.4, and by analyzing a histogram of the lifetime sparsities, they confirmed that none of the SAE features were dead. They also plotted the mean activation values of each neuron with their sparsity to identify low and high label entropy bands. Besides sparsity, they calculate the cosine similarity between the MLP outputs from the CLIP ViT and the SAE encoder vectors. This reflects how strongly the SAE captures the features the MLP uses to decide classes. From their plot, they concluded that

about a half of the features are not neuron aligned at all, and the other half are aligned mostly to a small number of MLP neurons.

References

[1] Learning transferable visual models from natural language supervision, 2021.

[2] Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet, 2024.

[3] Towards multimodal interpretability: Learning sparse interpretable features in vision transformers, 2024.

[4] C. O. N. S. T. H. S. K. Z. H.-D. R. L. D. D. C. C. R. G. S. M. J. K. D. A. M. W. C. O. Nelson Elhage, Tristan Hume. Toy models of superposition, 2022.

[5] J. B. B. C. A. J. T. C. N. L. T. C. A. C. D. A. A. R. L. Y. W. S. K. N. S. T. M. N. J. A. T. K. N. B. M. J. E. B. T. H. S. C. T. H. C. O. Trenton Bricken*, Adly Templeton*. Towards monosemanticity: Decomposing language models with dictionary learning, 2023.