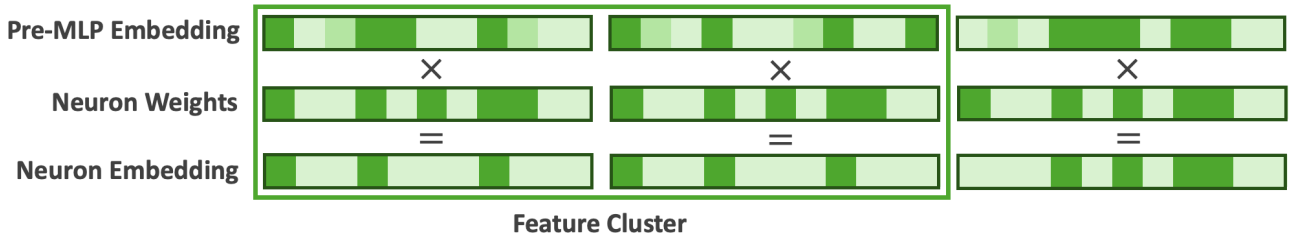

Figure 1. An illustration of the neuron embedding process. We compute the element-wise products of the pre-MLP embedding of the inputs and the neuron input weights to produce the neuron embeddings. These are then clustered based on similarity. The neuron weights select the relevant information from the embedding, such that the neuron embeddings of two different inputs can be brought together or pushed apart.

where $S_C(\cdot)$ is the cosine similarity. We use Hierarchical Agglomerative Clustering (HAC) on the distance matrix with a distance threshold that controls when clusters merge, which we set to 0.5. HAC also provides a full cluster hierarchy, which can give additional insight into a neuron's behaviour. In particular, neurons may have some semantically related but distinct features, as well as completely unrelated features. The hierarchy clearly captures these relationships between clusters, as well as between sub-clusters.

3.3. Monosemantic Training

3.3.1. Measuring Polysemanticity

By computing the neuron embeddings for a neuron's dataset examples, we can naturally compute metrics on the embedded points, allowing us to measure proxies for monosemanticity. We compute simple metrics like the maximum distance between any pair of points, the mean distance between points, and metrics on the clustering such as mean inter- and intra-cluster distance, and the number of clusters.

We can also apply this method to the neurons in Sparse Auto-Encoders (SAEs), which are trained to disentangle the features represented in MLP layers. Evaluating SAEs currently relies on a few heuristics that tend to correlate with interpretable, monosemantic neurons, such as the average number of SAE neurons activating per example, in combination with manual interpretation of neurons (Bricken et al., 2023). These neuron embeddings metrics more directly measure neuron monosemanticity, and so would be a natural addition to SAE evaluation to bridge the gap between fast but loosely-correlated heuristics, and slow manual interpretation.

3.3.2. Sparse Auto-Encoder Loss

When training SAEs, we want each SAE neuron to represent a single, interpretable feature. This is achieved by pushing the neurons to activate sparsely by including the $L1$ norm of the neuron activations in the loss. However, this does not directly penalise neurons for responding to multiple features. We show that we can utilise the ability of neuron embeddings to capture the similarity between features to directly penalise polysemantic neurons via a term in the SAE loss.

Intuitively, we want to push neurons to respond to a single feature, which corresponds to a neuron having a single dense cluster in it's neuron embedding space. To do this efficiently during training, for each SAE neuron we maintain a neuron embedding of the inputs which activated the neuron, and then convert this to a neuron embedding on the fly to measure the similarity of the combined neuron embedding with the neuron embedding of the current input. We then take the sum of these similarities for all neurons over a batch of inputs and include it in the SAE loss.

Assume we're training an SAE layer with dimensionality d, inserted after layer L_i of the original model. Concretely, at a given training step, for each input $\mathbf{x}_b$ in a batch of size n, $\{\mathbf{x}_0 \cdots \mathbf{x}_{n-1}\}$, we collect all the SAE neurons with a non-zero activation. For each neuron, we check if it's activated before. If it hasn't, we store the pre-SAE embedding h_i in a lookup mapping the neuron to an embedding. If it has, we retrieve the averaged embedding $h_{i,j,avg}$ stored in the lookup and convert it to a neuron embedding $\mathbf{e}_{i,j,avg}$ by taking it's Hadamard product with $\mathbf{w}_{i,j}^{SAE}$, the weights of SAE neuron $N_{i,j}^{SAE}$. We similarly compute the neuron embedding $\mathbf{e}_{i,j,b}$ of the current input $\mathbf{x}_b$ which has caused the neuron to activate, using it's pre-SAE embedding $h_{i,b}$.

We then compute the distance between the two neuron embeddings, $1 - S_C(\mathbf{e}_{i,j,avg}, \mathbf{e}_{i,j,b})$, which measures whether the feature the neuron is responding to in the current input is similar to the average feature the neuron has responded to throughout training. We take the sum of the distances across all neurons and all inputs in the batch to compute the neuron embedding loss $\mathcal{L}_N$. In a single equation:

$$\mathcal{L}_N = \sum_{b=0}^{n} \sum_{j=0}^{d} 1 - S_C(h_{i,j,avg} \odot \mathbf{w}_{i,j}^{SAE}, h_{i,b} \odot \mathbf{w}_{i,j}^{SAE}) \quad (4)$$