Interpretable Latent Representations of Neural Activity via Sparse Autoencoders

Arjun Naik

University of Washington
Paul G. Allen School of Computer Science & Engineering
arjunsn@uw.edu

June 2025

1 Introduction

Methods in modern theoretical and computational neuroscience rely heavily on machine learning approaches to analyze high-dimensional neural recordings. Some notable examples are calcium imaging, extracellular electrophysiology, and more comprehensive behavioral recordings. Embeddings are widely used in machine learning to represent complex, high-dimensional data in a lower-dimensional latent space [2] while preserving important structure. In NLP, word embeddings (e.g. word2vec [3], BERT) are used to *capture semantic relationships*, and in computer vision, learned latent spaces and feature embeddings enable tasks such as feature classification and retrieval. These applications have inspired similar techniques in both biology (e.g. gene embeddings) and neuroscience. In neuroscience, embeddings are applied to neural recordings to decode behavior and visualize dynamics. Here, latent embeddings compress neural activity into a few dimensions while retaining features associated with a task. However, most such embeddings are dense and uninterpretable vectors and are limited in the connection between the meaning of dimensions or how specific neurons contribute as features. This limits their utility when the goal is to relate activity to specific stimuli. This points to a need for interpretable latent models that can reveal structure in a biologically meaningful way without affecting downstream performance.

One strategy to improve interpretability in latent embeddings is to impose sparsity constraints, which allow only a few dimensions to be active for a given input. Sparse representations might make it easier to attribute latent features to individual neurons, behavioral variables, or even structured population activity patterns. Sparse Autoencoders (SAEs) are a framework that we can use to enforce this. Sparse Autoencoders have been shown to uncover disentangled and interpretable features in downstream models [1]. SAEs use sparsity penalties to enforce the structure while retaining the reconstruction performance of embeddings [5, 6]. CEBRA (a neural embedding model suite) claims to produce interpretable embeddings by aligning embedding dimensions with specific auxiliary variables (such as behavior), which enables the latent space to preserve structure [7]. The interpretability claimed here is mainly at the level of global behavioral structure. A clear further direction is to interpret how individual latent units correlate with specific neurons or more

fine-grained behavioral components. In this project, we investigate whether post-hoc sparsification using SAEs can generate more interpretable latent representations of embeddings generated by CEBRA, emphasizing attribution at the neuron level. We will also assess how this transformation affects behavioral decoding performance, to ensure that interpretability does not tradeoff with utility of the embeddings. Having interpretable latent representations of neural activity will enable researchers to trace population level dynamics to individual neurons or neural circuits, which is critical for understanding neural coding. In the following sections, we describe our dataset and methods, report quantitative and qualitative evaluations of post-hoc sparsification, and discuss implications for biological interpretability and future directions.

2 Data Description

For this analysis, we used publicly available datasets from the `cebra.datasets.hippocampus` module [7]. The datasets are single session tetrode recordings from four rats: *Achilles, Buddy, Cicero, Gatsby*. The neural data were binned into 25 millisecond time windows, yielding spike train vectors for each timestep. Each sample is a high-dimensional vector that represents the firing activity of neurons at each time point, and the number of recorded neurons varies per rat. Each sample is paired with continuous behavioral labels: the position of the rat along a linear track, and the running direction, encoded as two binary dimensions: left and right. Overall the label is a 3D vector. This enables supervised learning of the embeddings and downstream decoding analysis. Each rat has a different number of total time points depending on session length and activity, and we partitioned the data into a 80%-20% train/validation split individually for each rat. Other than the preprocessing done by the CEBRA module behind the scenes, no manual preprocessing was applied.

3 Methods

3.1 Overview

To restate, our goal is to assess whether SAEs can enhance the interpretability of CEBRA-generated neural embeddings without significantly affecting decoding performance. First we tune CEBRA models to generate neural embeddings from hippocampal neural recordings, and then we use SAEs (tuned with multiple architectures and hyperparameters) in order to enforce post-hoc sparsifications. In order to pick SAEs to use in our analysis, we will be analyzing reconstruction fidelity and sparsity (trying to maximize sparsity and reconstruction fidelity simultaneously). Upon selecting specific SAEs, we analyze interpretability using custom metrics and determine how decoding accuracy changed.

3.2 Generating Neural Embeddings with CEBRA

To generate behaviorally relevant latent embeddings from the data, we configured CEBRA to operate in supervised mode, using position and direction labels from rat hippocampal recordings as alignment targets. We trained models independently for each rat dataset, but used the same hyperparameter sweep for all of them:

- **Output Dimensions:** `[8, 16, 32, 64]`

- **Model Architecture:** `"offset10-model"` (causal convolution with 10-bin context)
- **Time Offsets:** `5` (supervised positive pairs 5 bins apart)
- **Temperature Mode:** `"constant"`, **Temperature:** `[0.1]`
- **Max Iterations:** `2000`
- **Hidden Units:** `[32, 64]`
- **Distance Metric:** `"cosine"`
- **Batch Size:** `256`
- **Learning Rate:** `[1e-3, 5e-3]`

We used the `offset10-model` architecture provided by the CEBRA library, which applies a stack of causal convolutional layers to neural activity vectors. This CEBRA model captures temporal patterns over a 10-bin window and aligns embeddings across time using contrastive supervision. The use of causal convolutions makes sure that the learned representations are based on past and current activity only, and that they preserve biological plausibility.

We performed a coarse hyperparameter sweep to select effective defaults for training CEBRA embeddings across rat datasets. Each embedding is a fixed-length vector of the specified output dimension, computed per 25 ms neural activity bin. We chose output dimensions 8-64 to provide a good balance between encoding enough information while not being too noisy. Learning rates of 0.001 and 0.005 are standard learning rates for contrastive learning. We chose 32 and 64 as our number of hidden units, to avoid our model overfitting to small datasets.

After these preliminary sweeps, we analyzed how the specific hyperparameters chosen affected directional decoding accuracy as well as positional decoding accuracy. In order to decode position, we trained a Ridge regression model to predict the rat's position along a 1D track uing the latent embeddings as input. We report the accuracy using R^2 on a validation set. To classify direction, we trained a logistic regression model to classify whether the rat was running left or right using the embeddings again. Here we report performance based on classification accuracy on validation data. These decoding models are linear and deliberately simple in order to ensure that high decoding performance is reflective of structure captured by embeddings.

As shown qualitatively in Figure 1, decoding performance seems to strongly favor a learning rate of 0.001 and 64 hidden units, but doesn't seem to have a clear preference regarding output dimensions. For the purposes of this paper, we will fix the learning rate and number of hidden units, but will vary the embedding size for the set `[8, 16, 32, 64]`. We are varying embedding size to analyze how enforcing sparsity can lead to interpretability for various amounts of encoded information to begin with. So overall, we are training 16 CEBRA models (4 rats $\times$ 4 output dimensions).

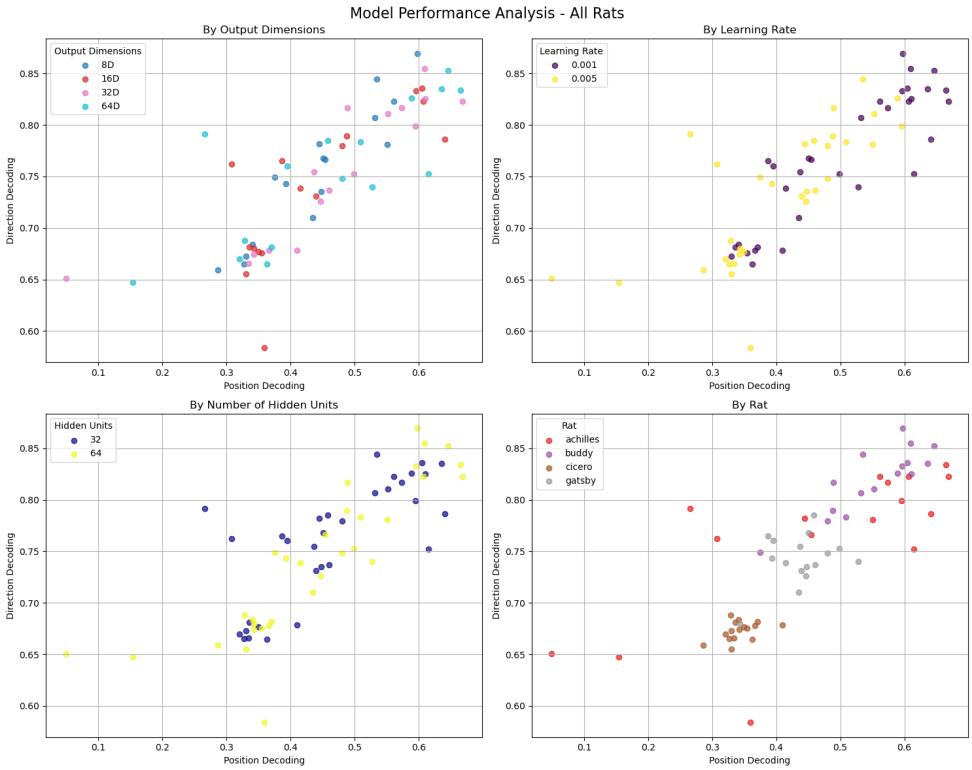

Figure 1: Decoding performance across embedding hyperparameters. Each point represents the R2 (position) and accuracy (direction) for a specific CEBRA model configuration. Color represents the value of a selected hyperparameter. More detailed graphs located on this paper's GitHub. [4]

3.3 Sparsification via Sparse Autoencoders

3.3.1 SAE Architectures

To induce sparsity and improve interpretability in the latent embeddings, we trained a set of sparse autoencoders (SAEs) that use CEBRA embeddings as inputs. Each SAE takes a fixed-dimensional embedding and learns a higher-dimensional latent representation that reconstructs the input with a sparsity constraint. We explored several architectural variants, each applying a different form of sparsity regularization:

- **Standard SAE:** A single hidden-layer autoencoder with ReLU activation and an L_1 penalty on the latent representation.
- **Top-k SAE:** Enforces sparsity by retaining only the top k largest activations in each latent vector and setting the rest to zero.
- **Jump ReLU SAE:** Introduces a sparsity penalty based on a nonlinear exponential function of activation magnitude, as described in prior literature.

Each SAE has an expansion factor hyperparameter, which controls the dimensionality of the latent space. For an input of dimension d, the latent representation is of size $d \times$ expansion factor. All SAEs share the same encoder-decoder structure with fully connected layers and ReLU activations, and were trained to minimize reconstruction loss plus a sparsity penalty term.

3.3.2 SAE Training

We trained each sparse autoencoder (SAE) to reconstruct latent embeddings produced by CEBRA, using a combination of mean squared error (MSE) reconstruction loss and an architecture-specific sparsity penalty. For a given input embedding $\mathbf{z} \in \mathbb{R}^d$, the SAE learns a transformation $f(\mathbf{z}) = \hat{\mathbf{z}}$ via a latent bottleneck $\mathbf{h} \in \mathbb{R}^{d \times \text{expansion factor}}$ that is constrained to be sparse. The general loss function used is:

$$\mathcal{L} = \|\mathbf{z} - \hat{\mathbf{z}}\|_2^2 + \lambda \cdot \mathcal{R}(\mathbf{h})$$

$\mathcal{R}(\mathbf{h})$ is a sparsity regularization term that depends on the chosen architecture (e.g., L_1 norm, top-k thresholding, or jump-based exponential penalty). λ is a tunable hyperparameter controlling the sparsity-strength tradeoff. For top-k SAEs, the penalty term is zero and sparsity is enforced manually.

We trained one SAE model per combination of rat, embedding dimensionality, architecture type, and relevant hyperparameters. All models were trained for 50 epochs using the Adam optimizer with a batch size of 256. Learning rate and sparsity-related hyperparameters were selected via grid search (described in the following subsection).

We trained the SAEs on the following hyperparameter sweep:

- **Embedding Dimensions:** 8, 16, 32, 64
- **Architectures:** `standard`, `topk`, `jumprelu`
- **Expansion Factors:** 2, 4, 8
- **Learning Rates:** 0.0001, 0.0003, 0.001
- **ℓ_1 Penalty Weights (Standard, JumpReLU):** 0.0001, 0.001, 0.01
- **Top-k Values (TopK):** 8, 16, 32
- **Bandwidth Values (JumpReLU):** 0.0005, 0.001, 0.005

For further analysis and hyperparameter selection, we saved the models from the sweep and recorded the latent representation.

3.3.3 Hyperparameter Selection

To evaluate the performance of each trained SAE model, we computed the reconstruction quality (R^2), and latent sparsity. These metrics helped guide our hyperparameter selection process.

Our primary goal was not to identify a single "best" model per se, but rather to determine which architecture and configuration consistently produced latent representations across rats and embedding sizes that had strong reconstruction fidelity while also being as sparse as possible. Based on qualitative scatter plots of explained variance versus sparsity, we observed that the `topk` architecture consistently achieved strong reconstruction performance while maintaining high sparsity. This pattern held more clearly at higher embedding dimensions (e.g., 32 and 64), which may indicate that top-k selection provides a robust inductive bias for interpretability.

Figures 2 and 3 show both extreme cases of the explained variance versus sparsity scatterplots. In the Figure 2, there isn't a clear separation, but top-k still clearly outperforms the other architecture. The separation is more clear in the Figure 3. To evaluate performance here, we are qualitatively looking for an architecture which sits close to the top left of the graph, where both sparsity and reconstruction fidelity are maximized. We also qualitatively observe that expansion factor of 8 sits closest to the top left of the graph consistently for top-k SAEs, indicating this as a good hyperparameter to stick with.

To limit the search space for downstream evaluation, we fixed the SAE architecture to `topk`, the expansion factor to 8, and the learning rate to 0.001 (see GitHub for how this parameter was tuned, it was very similar to the previous one). We then varied the top-k value across `[8, 16, 32]` to explore the sparsity-interpretability tradeoff. Other hyperparameters were not relevant in the `topk` case. This narrowing allowed us to systematically analyze the impact of top-k sparsity on both decoding and attribution metrics.

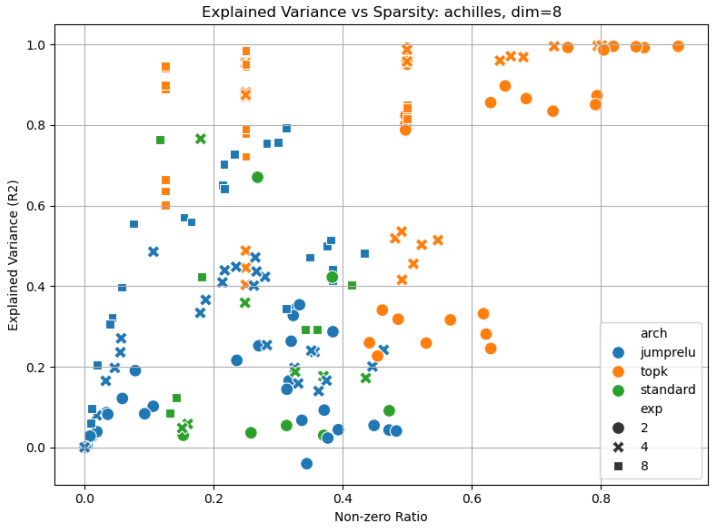

Figure 2: Explained variance (reconstruction fidelity) vs Nonzero ratio (inverse of sparsity) for Achilles and input dimension of 8. Color represents architecture while shape represents expansion factor. More detailed graphs located on this paper's GitHub. [4]

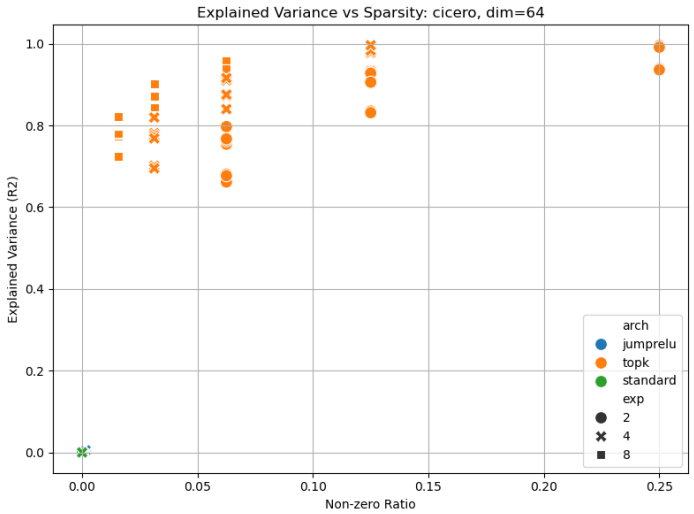

Figure 3: Explained variance (reconstruction fidelity) vs Nonzero ratio (inverse of sparsity) for Cicero and input dimension of 64. Color represents architecture while shape represents expansion factor. More detailed graphs located on this paper's GitHub. [4]

3.4 Post-hoc Evaluation

To quantitatively assess the quality and interpretability of both the original CEBRA embeddings as well as their SAE counterparts, we apply a range of evaluation metrics. These evaluations fall into two broad categories: decoding performance and interpretability.

3.4.1 Decoding Performance Metrics

We evaluated each embedding space by measuring how well it could linearly decode behavioral variables:

- **Position Decoding (Regression):** We trained a Ridge regression model to predict the rat's 1D position on the track. Performance was measured using R^2 on validation data:

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

- **Direction Decoding (Classification):** We trained a logistic regression classifier to determine whether the rat was moving left or right. Accuracy was measured on validation data:

$$\text{Accuracy} = \frac{1}{N} \sum_i \mathbb{1}[y_i = \hat{y}_i]$$

Interpretability Metrics

To evaluate interpretability, we used two scores to reflect how clearly latent dimensions relate to known neural or behavioral structure:

Neuron–Latent Attribution Score We computed the absolute Pearson correlation between each neuron's activity and each latent dimension across time:

$$\text{corr}_{ij} = \text{corr}(\text{neuron}_i, \text{latent}_j)$$

The neuron attribution score was then defined as the average of the strongest 10 absolute correlations for each latent dimension:

$$\text{Neuron Score} = \frac{1}{10} \sum_{j=1}^{1} 0 \max_{i \in \text{neurons}} |\text{corr}_{ij}|$$

Behavior–Latent Attribution Score For each latent dimension, we computed the R^2 of a Ridge regression predicting each of the three behavioral components: position, left, and right. The behavior score captures the strength of alignment between any latent unit and a behavioral feature:

$$\text{Behavior Score} = \frac{1}{3} \sum_{j=1}^{3} \max_{k \in \{\text{pos, left, right}\}} R^2(z_j \to \text{behavior}_k)$$

To quantify the structural compactness of embeddings, we computed the proportion of nonzero latent dimensions. These metrics were applied consistently to both pre- and post-SAE embeddings, enabling a direct comparison of interpretability and utility tradeoffs.

4 Results

4.1 Interpretability Improvements After SAE Sparsification

To compare interpretability before and after sparsification, we visualize changes in both neuron-level attribution and behavioral correlation scores. These scores are defined in Section 3.4. Figure 4 shows a consistent increase in interpretability for the neuron attribution score after applying sparse autoencoders, with results broken down by top-k values. For the behavior attribution score, CEBRA performs slightly better than $k = 8, 16$, but $k = 32$ beats CEBRA embeddings in terms of interpretability.

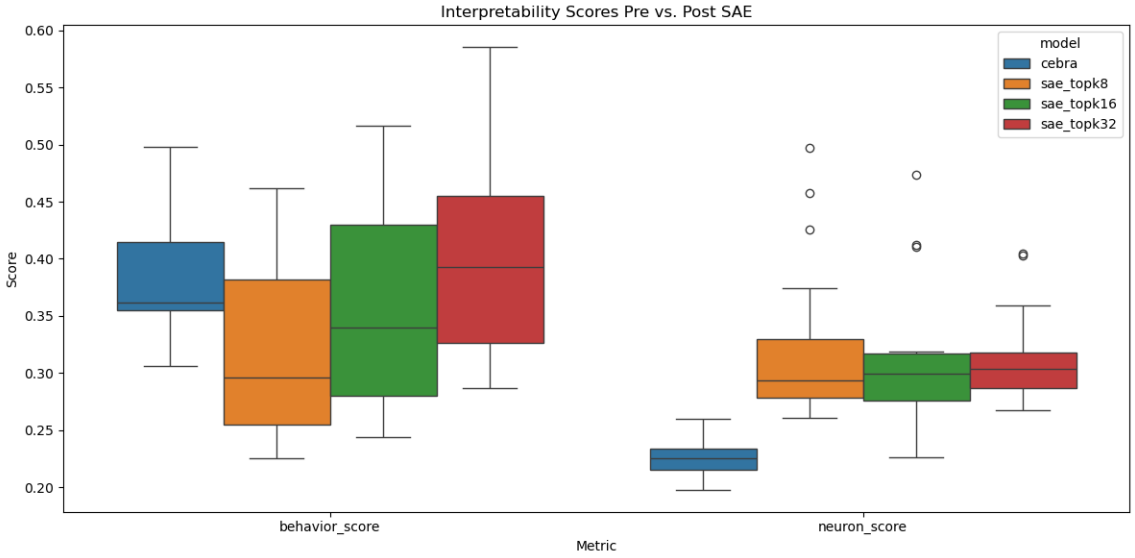

Figure 4

4.2 Decoding Performance Pre vs. Post Sparsification

To assess whether sparsification compromises decoding quality, we plot decoding accuracy for both position and direction tasks. Figure 5 indicates that decoding performance is largely preserved post-sparsification, even as sparsity increases, and is even enhanced slightly.

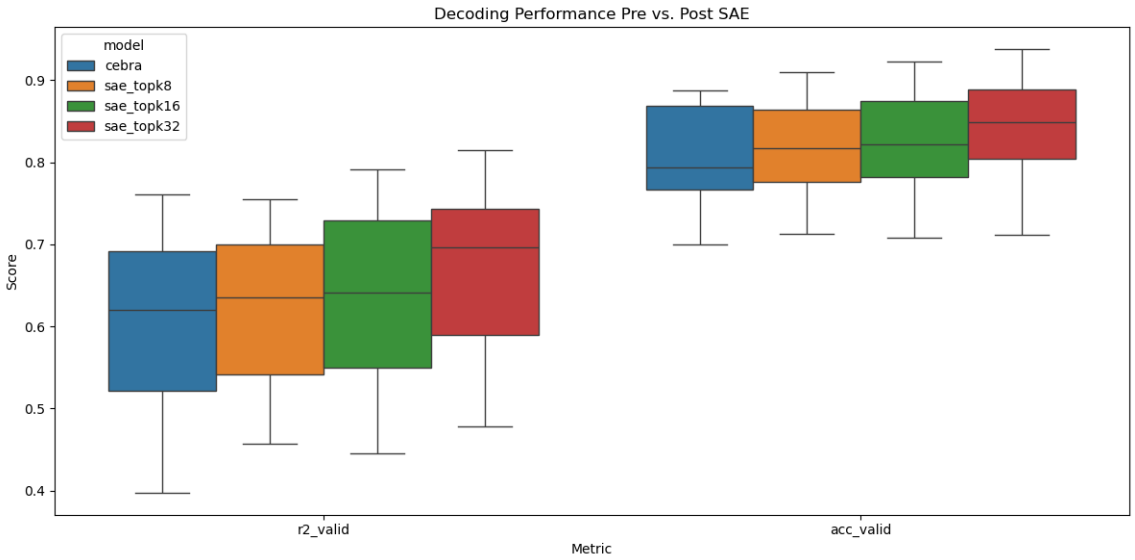

Figure 5

4.3 Effect of TopK and Embedding Size on Interpretability

To better understand how architectural and hyperparameter choices affect interpretability, we examine interpretability scores broken down by embedding dimension and top-k sparsity level. Figure 6 shows that higher embedding sizes tend to improve neural interpretability, while it is not affected by k. On the other hand, lower embedding sizes and higher k tend to improve behavioral interpretability.

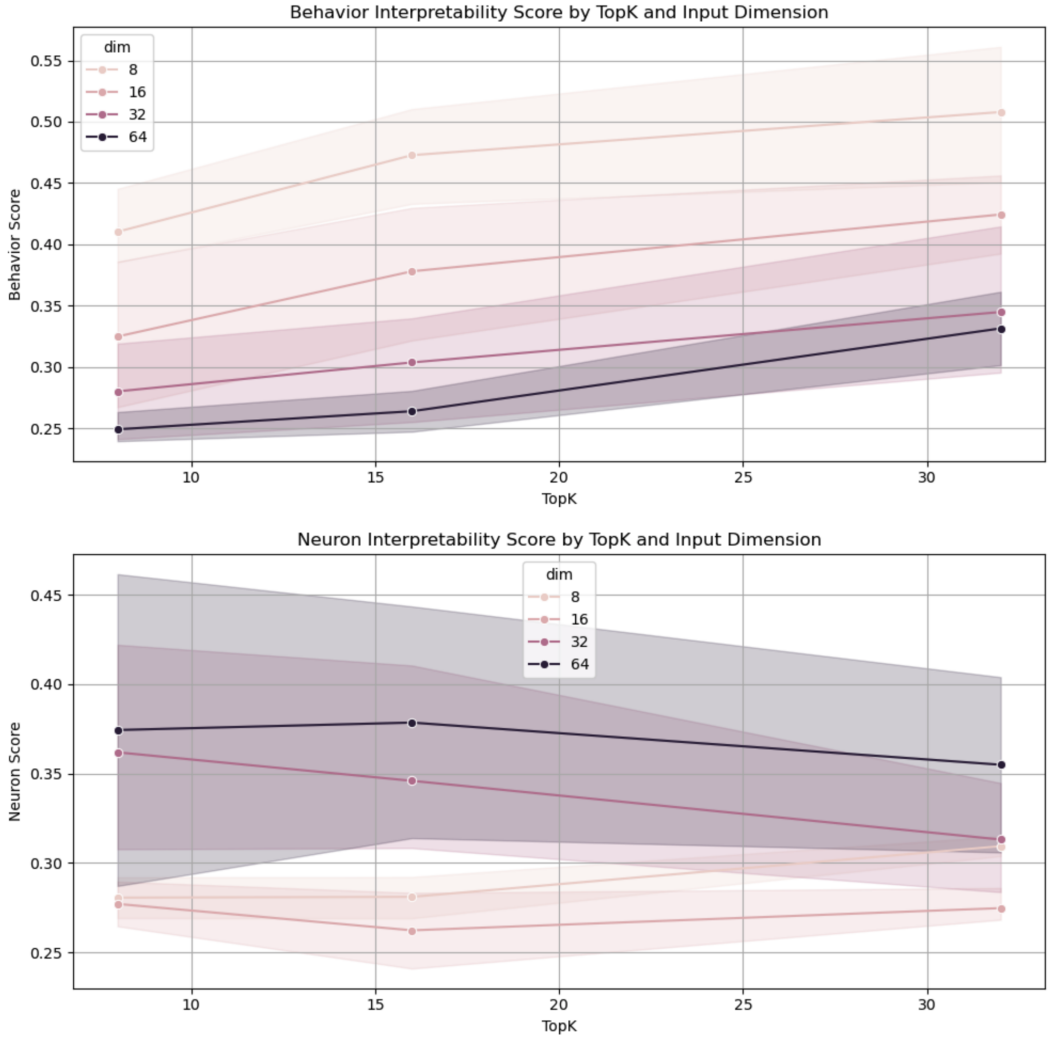

Figure 6

5 Discussion

We see that sparsifying CEBRA embeddings using sparse autoencoders (SAEs) enhances interpretability (mainly neural) without degrading downstream decoding performance. Figure 4 shows that neuron-level attribution improves reliably across SAE architectures, with top-k models showing a strong increase in interpretability as k increases. Behavior-level attribution displays a more

complicated, with $k = 32$ models outperforming raw CEBRA embeddings, suggesting that sparsity may balance compression and retention of behaviorally relevant information.

Decoding results (Figure 5) further show that sparsification does not sacrifice downstream representation quality. In fact, sparse embeddings sometimes achieve higher decoding scores, likely due to its regularization effects or better disentangling of latent dimensions.

Figure 6 illustrates how interpretability depends on both the embedding size and the degree of sparsity. Neural interpretability increases with larger embedding sizes, maybe because of a higher capacity for representing individual neurons distinctly. In contrast, behavior interpretability improves with both increased sparsity and reduced embedding dimensionality, suggesting that simpler representations may more directly align with task-relevant features.

5.1 Limitations and Future Directions

One limitation of our approach is that we assume we can capture interpretability using qualitative judgment and simple correlation metrics. These are helpful heuristics, but they may not fully reflect the causal and hierarchical structure of latent representations. Additionally, we restricted our SAE variants to a few mechanisms of sparsity. Future work could explore learned sparsity thresholds or dynamic gating mechanisms (Gated SAEs).

Another limitation is that our interpretability metrics are defined post hoc and may be sensitive to noise, especially in smaller latent spaces. More robust or unsupervised interpretability assessments (e.g., based on alignment with structure in the neural manifold) could improve generality.

6 Conclusion

We presented a pipeline for enhancing and evaluating the interpretability of latent neural representations by applying sparse autoencoders to supervised CEBRA embeddings. Our results show that sparse embeddings retain and even improve decoding performance while yielding gaining interpretability in their latent dimensions. They're interpretable in terms of their alignment with individual neurons and behavioral variables. These findings suggest that sparsity is a useful inductive bias for representation learning models with neuroscience in mind.

7 Acknowledgments

I would like to thank Prof. Mat Golub and the CSE 599 teaching staff for their guidance throughout this course and for support in this project. I am also grateful to Dr. Su-In Lee as well as Chris Lin for their mentorship in the application of explainable AI. Finally, I acknowledge the authors of the `cebra` library for making their tools available.

All code, analysis notebooks, and additional figures are available at our project GitHub:
`https://github.com/Aleph-Null-123/interpretable-neural-embeddings`.

References

[1] Hoagy Cunningham, Aidan Ewart, Logan Riggs Smith, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

[2] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[3] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[4] Arjun Naik. Interpretable latent representations of neural activity via sparse autoencoders. `https://github.com/Aleph-Null-123/interpretable-neural-embeddings`, 2025. Accessed: June 2025.

[5] Andrew Y Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78, 2004.

[6] Bruno A Olshausen and David J Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

[7] Steffen Schneider, Jin Hwa Lee, and Mackenzie Weygandt Mathis. Learnable latent embeddings for joint behavioural and neural analysis. *Nature*, 617:360–368, 2023.