Multiway Multislice PHATE: Visualizing Hidden Dynamics of RNNs through Training

Anonymous authors
Paper under double-blind review

Abstract

Recurrent neural networks (RNNs) are a widely used tool for sequential data analysis; however they are still often seen as black boxes. Visualizing the internal dynamics of RNNs is a critical step in understanding the functional principles of these networks and developing ideal model architectures and optimization strategies. Previous studies typically only emphasize the network representation post-training, overlooking their evolution process throughout training. Here, we present Multiway Multislice PHATE (MM-PHATE), a novel method for visualizing the evolution of RNNs' hidden states. MM-PHATE is a graph-based embedding using structured kernels across the multiple dimensions spanned by RNNs: time, training epoch, and units. We demonstrate on various datasets that MM-PHATE uniquely preserves hidden representation community structure among units and identifies information processing and compression phases during training. The embedding allows users to look under the hood of RNNs across training and provides an intuitive and comprehensive strategy for understanding the network's internal dynamics, such as why and how one model outperforms another or how specific architectures impact an RNN's learning ability.

1 Introduction

Recurrent neural networks (RNNs) are designed to handle sequential data by modeling input sequences and retaining memory of past elements through recurrent connections or memory units (Lipton et al., 2015; Kaur & Mohta, 2019). Unlike feedforward neural networks (FNNs), which process each input independently, RNNs update their internal state dynamically, enabling them to capture contextual relationships across sequences. This makes RNNs particularly effective for tasks that rely on the order and relationships between elements, such as time-series analysis and action recognition (Hewamalage et al., 2021).

Since gaining popularity in the 1990s, various RNN variants and training strategies have been developed to improve training stability and long-range dependency learning. Architectures such as Long Short-Term Memory (LSTM) networks (Hochreiter & Schmidhuber, 1996) and Structurally Constrained Recurrent Networks (SCRN)(Mikolov et al., 2014) were designed to address challenges like the vanishing gradient problem(Salehinejad et al., 2018). Additionally, RNNs excel at handling irregular or incomplete sequential data due to their flexibility with variable-length inputs. These properties, along with ongoing architectural improvements (Salehinejad et al., 2018), have enabled RNNs to achieve exceptional performance across domains such as natural language processing (NLP), neuroscience, and biomedical signal processing (Barak, 2017; Chen & Li, 2021; Khalifa et al., 2021). Notably, RNNs remain the state-of-the-art for neural decoding in intracortical brain-computer interfaces (Deo et al., 2024).

Despite their extensive use, the learning dynamics and internal representations of RNNs remain difficult to interpret. This opacity complicates performance evaluation, model design, and parameter selection, hindering the development of more effective architectures. While significant progress has been made in interpreting FNNs (Wang et al., 2021; Yu et al., 2014), similar advances for RNNs have been limited. Most RNN analyses focus on either fixed-point analysis in dynamical systems (Sussillo & Barak, 2013) or performance evaluations across different architectures and components (Chung et al., 2014; Greff et al., 2017). Consequently, there is a clear need for new methods that facilitate the interpretation of RNNs' latent representations during training.

In explainable deep learning, dimensionality reduction is widely used to visualize high-dimensional data, helping researchers gain intuition about its structure and relationships (Karpathy et al., 2015; Hidaka & Kurita, 2017; Rauber et al., 2016; Gigante et al., 2019). However, existing techniques often have limitations. Methods like t-SNE emphasize local structure at the expense of global patterns (Maaten & Hinton, 2008), while methods such as PCA and Isomap focus on global structures and may miss important local details (Maćkiewicz & Ratajczak, 1993; Tenenbaum et al., 2000). These methods can also be sensitive to noise and outliers (Moon et al., 2019). Visualizing RNNs is particularly challenging since they require preserving structures across multiple dimensions, including time, epochs, and hidden units. Consequently, traditional dimensionality reduction techniques often fail to capture the complexity of RNN learning dynamics.

To address these challenges, Gigante et al.(Gigante et al., 2019) introduced Multislice PHATE (M-PHATE), which visualizes FNN hidden states during training. M-PHATE constructs a multislice graph where each slice represents the network's state at a specific training epoch, capturing both temporal relationships and community structures through PHATE(Moon et al., 2019). This approach effectively captures performance-related features, such as task-related specialization, without requiring external validation data, making it particularly useful in data-limited settings.

However, M-PHATE is designed for FNNs and does not account for the sequential nature of RNNs, where hidden states across all time steps play a critical role in representation learning (Su & Shlizerman, 2020). To address this limitation, we propose Multiway Multislice PHATE (MM-PHATE), which extends M-PHATE by capturing RNN hidden states across both time steps and epochs. MM-PHATE provides a comprehensive view of RNN learning dynamics, offering deeper insights into how information is processed and represented during training. Our results demonstrate that MM-PHATE preserves more dynamic details essential for understanding RNN performance compared to existing methods such as PCA, t-SNE, Isomap, Locally Linear Embedding (LLE), UMAP, and M-PHATE.

Our main contributions are as follows:

- We introduce MM-PHATE, a novel framework for visualizing the hidden dynamics of RNNs across both time steps and epochs. This method provides new insights into the learning trajectory, learned representations, and model performance.
- We show that MM-PHATE preserves the community structure of hidden units by tracking their learning trajectories and capturing correlations among their activations throughout training.
- MM-PHATE reveals phases of information processing and compression during training, consistent with the information bottleneck theory. We perform experiments to quantitatively confirm that the information retained by the network is reflected in the MM-PHATE embedding.

2 Related Work

Existing methods for interpreting RNNs can be categorized into performance-oriented and application-oriented post-training analyses. Performance-oriented approaches focus on evaluating network-level performance by comparing architectural components and training parameters. For example, Chung et al. (2014) compared gated RNNs (e.g., GRUs and LSTMs), while Greff et al. (2017) conducted a detailed analysis of LSTM components. However, these studies primarily emphasize performance outcomes and provide limited insights into the hidden state dynamics and learned representations within RNNs.

In contrast, application-oriented approaches focus on visualizing and interpreting hidden state activations after training, often in the context of specific tasks. In NLP, Karpathy et al. (2015) overlaid activation maps on texts, revealing interpretable unit behaviors such as tracking text structure. Li et al. (2016) used saliency heat maps to identify critical words in learned representations, while Strobelt et al. and Ming et al. developed interactive tools to correlate hidden state patterns with phrases (Strobelt et al., 2018; Ming et al., 2017). Similar techniques have been applied to domains such as speech recognition (Tang et al., 2017), earth sciences (Titos et al., 2022), and medical applications (Kwon et al., 2019). While these studies provide intuitive, task-specific insights, they often lack generalizability and do not capture training dynamics over time.

Other works have explored general RNN behavior using techniques such as Proper Orthogonal Decomposition (POD) to analyze Seq2Seq internal states (Su & Shlizerman, 2020) and PCA to link recurrent activations to model generalization (Farrell et al., 2022). However, these approaches focus on post-training analysis and do not visualize how hidden states evolve across both time-steps and epochs during training. To our knowledge, no existing methods provide a unified view of RNN hidden dynamics over temporal and training dimensions.

3 Background

PHATE: PHATE is a data visualization technique that can capture both the local and global structure of data using diffusion processes (Moon et al., 2019). The PHATE algorithm optimizes the diffusion kernel (Coifman & Lafon, 2006) for the visualization of high-dimensional data. Let $\boldsymbol{x}_i$ be a point in a high-dimensional dataset. PHATE begins by computing the Euclidean distance matrix $\boldsymbol{E}$ between all data points, where $\boldsymbol{E}_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$. These distances are then transformed into affinities using an adaptive α-decay kernel $\boldsymbol{K}_{k,\alpha}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{1}{2}\exp\left(-\left(\frac{\boldsymbol{E}_{ij}}{\epsilon_k(\boldsymbol{x}_i)}\right)^{\alpha}\right) + \frac{1}{2}\exp\left(-\left(\frac{\boldsymbol{E}_{ij}}{\epsilon_k(\boldsymbol{x}_j)}\right)^{\alpha}\right)$, which adapts to the data density around each point and captures local information. The parameters $\epsilon_k(x_i)$ and $\epsilon_k(x_j)$ are the k-nearest-neighbor distance of x_i and x_j, and α controls the decay rate. The affinities are then row-normalized to obtain the diffusion operator $\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{K}_{k,\alpha}$ that represents the single-step transition probabilities between data points, where $\boldsymbol{D}$ is a diagonal matrix whose entries are row sums of $\boldsymbol{K}_{k,\alpha}$. PHATE calculates the information distance between points based on their transition probabilities: $\text{dist}_{ij} = \sqrt{\|\log \boldsymbol{P}_i^t - \log \boldsymbol{P}_j^t\|^2}$, where $\boldsymbol{P}^t$ captures the transition probabilities of a diffusion process on the data over t steps and i and j are rows in the matrix. These distances are embedded into low dimensions using Multidimensional Scaling (MDS) (Ramsay, 1966) for visualization. Local and global distances within the data's manifold are represented in PHATE by multistep diffusion probabilities. The diffusion probability of each point captures its local context, enabling pairwise comparisons between all points (both neighboring and distant points) that represent the entire global context. For further details, see (Moon et al., 2019). We will use PHATE to embed RNN training dynamics, however we alter the initial graph construction to emphasize certain structures in the data we wish to visualize.

M-PHATE: Gigante et al. (2019) model the evolution of the hidden units in a feedforward neural network and their community structure using a multislice graph. Each slice corresponds to the network at an epoch during training, and the collection of graphs represents the dynamical system resulting from the evolution of the network's hidden states. Let $\boldsymbol{F}$ be an FNN with a total of m hidden units, and let $\boldsymbol{F}^{(\tau)}$ be the representation of the network after being trained for $\tau \in \{1, ..., n\}$ epochs on the training data $\boldsymbol{X}$ sampled from a larger dataset $\boldsymbol{\Pi}$. The algorithm first calculates a shared feature space using the normalized activations of all hidden units $i \in \{1, \dots, m\}$ on the input data, as a 3-dimensional tensor:

$$\boldsymbol{T}(\tau, i, k) = \frac{\boldsymbol{F}_i^{(\tau)}(\boldsymbol{Y}_k) - \frac{1}{p}\sum_\ell \boldsymbol{F}_i^{(\tau)}(\boldsymbol{Y}_\ell)}{\sqrt{\text{Var}_\ell[\boldsymbol{F}_i^{(\tau)}(\boldsymbol{Y}_\ell)]}},$$

where $\boldsymbol{F}_i^{(\tau)}(\boldsymbol{Y}_k) : \mathbb{R}^d \to \mathbb{R}$ denotes the activation of the i-th hidden unit of $\boldsymbol{F}$ for the k-th sample from input data $\boldsymbol{Y}$. Here, $\boldsymbol{Y}$ is a subset of p samples from the d-dimensional training data $\boldsymbol{X}$, with an equal number of samples from each input class, and $p \ll |\boldsymbol{X}|$. This activation tensor $\boldsymbol{T}$ is then used to calculate intraslice affinities between pairs of hidden units within an epoch τ during the training, as well as the interslice affinities between a hidden unit i and itself at different epochs:

$$\boldsymbol{K}_{\text{intraslice}}^{(\tau)}(i, j) = \exp\left(\frac{-\|\boldsymbol{T}(\tau, i) - \boldsymbol{T}(\tau, j)\|_2^{\alpha}}{\sigma_{(\tau,i)}^{\alpha}}\right)$$

$$\boldsymbol{K}_{\text{interslice}}^{(i)}(\tau, \upsilon) = \exp\left(\frac{-\|\boldsymbol{T}(\tau, i) - \boldsymbol{T}(\upsilon, i)\|_2^{2}}{\epsilon^2}\right)$$

where α is the α-decay parameter, $\sigma_{(\tau,i)}$ is the intraslice bandwidth for unit i in epoch τ, and ϵ is the fixed interslice bandwidth. These matrices are combined to form an $nm \times nm$ multislice kernel matrix $\boldsymbol{K}$, which is then symmetrized, row-normalized, and visualized using PHATE in 2D or 3D.

4 Multiway Multislice PHATE

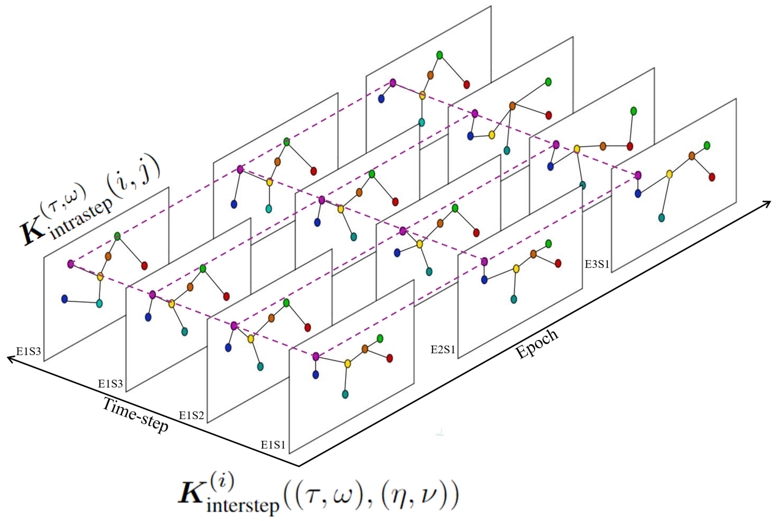

$\boldsymbol{K}^{(\tau,\omega)}_{\text{intrastep}}(i,j)$
E1S3
E1S3
E1S2
E1S1
E2S1
E3S1
Time-step
Epoch
$\boldsymbol{K}^{(i)}_{\text{interstep}}((\tau,\omega),(\eta,\nu))$

Figure 1: Example schematic of the multiway multislice graph used in MM-PHATE for RNNs. The intrastep kernels represent the similarities between the graph nodes at the same time-steps. The interstep kernels represent the similarities between the nodes and themselves at different time-steps and epochs.

M-PHATE was shown to be a powerful tool for visualizing FNNs. However, to effectively visualize the evolution of RNNs' hidden representations, we need to consider hidden state dynamics across *time-steps* within the sequence and training epochs concurrently (Su & Shlizerman, 2020). In RNNs, the output from previous *time-steps* is fed as an input to current *time-steps*. This is useful in the treatment of sequences and building a memory of the previous inputs into the network. The network iteratively updates a hidden state h. At each *time-step* t, the next hidden state h_{t+1} is computed using the input x_t and the current hidden state h_t. Importantly, the network uses the same weights W and biases b for each *time-step*. Thus the output y_t at *time-step* t is $y_t = f(W \cdot h_t + b)$, where f is some activation function.

Let $\boldsymbol{R}^{(\tau)}$ be the representation of an m-unit RNN after being trained for $\tau \in \{1, \dots, n\}$ epochs on the training data $\boldsymbol{X} \subset \boldsymbol{\Pi}$. We denote $\boldsymbol{R}^{(\tau)}_{i,w}(\boldsymbol{Y}_k) : \mathbb{R}^d \to \mathbb{R}$ the activation of the i-th hidden unit of $\boldsymbol{R}$ at time-step $w \in \{1, \dots, s\}$ in epoch τ for the k-th sample of $\boldsymbol{Y}$, where $\boldsymbol{Y}$ consists of p samples from the training data $\boldsymbol{X}$. We construct the 4-way tensor $\boldsymbol{T}$ using the hidden unit activations as a shared feature space, which we use to calculate unit affinities across all epochs and time-steps. The tensor $\boldsymbol{T}$ is an $n \times s \times m \times p$ tensor containing the activations at each epoch $\tau \in \{1 \dots n\}$ and time-step $w \in \{1 \dots s\}$ of each hidden unit $\boldsymbol{R}_i$ ($i \in \{1 \dots m\}$) with respect to each sample $\mathbf{Y}_k \subset \boldsymbol{X}$. To eliminate the variability in $\boldsymbol{T}$ due to the bias term b, we z-score the activation of each hidden unit at time-step w and epoch τ:

$$\boldsymbol{T}(\tau,\omega,i,k) = \frac{\boldsymbol{R}^{(\tau)}_{i,\omega}(\mathbf{Y}_k) - \frac{1}{p}\sum_\ell \boldsymbol{R}^{(\tau)}_{i,\omega}(\mathbf{Y}_\ell)}{\sqrt{Var_\ell[\boldsymbol{R}^{(\tau)}_{i,\omega}(\mathbf{Y}_\ell)]}}. \tag{1}$$

We construct a kernel over $\boldsymbol{T}$ utilizing our prior knowledge of the temporal aspect of $\boldsymbol{T}$ to capture its dynamics over epochs and time-steps. This constructed kernel, denoted $\boldsymbol{K}$, represents the weighted edges in the multislice graph of the hidden units(Fig. 1). In this representation, each unit has two types of connections: edges between the unit to itself across epochs and time steps and, within a fixed epoch and time-step, edges between a unit and its community—the other units which have the most similar representation. The edges are weighted by the similarity in activation pattern. We define $\boldsymbol{K}$ as a $nsm \times nsm$ kernel matrix between all m hidden units at all s time-steps in all n training epochs. The $((\tau-1)sm+(\omega-1)m+j)_{th}$ row or column of $\boldsymbol{K}$ refers to the j_{th} unit at time-step w in epoch τ. We henceforth refer to the row as $\boldsymbol{K}((\tau,\omega,j),:)$ and the column as $\boldsymbol{K}(:,(\tau,\omega,j))$. In order to capture the evolution of hidden units of R across time-steps and epochs, while preserving the unit's community structure, we construct a multiway multislice kernel matrix reflecting two types of connections simultaneously. Given the α-decay parameter α, the intrastep bandwidth for unit i at time-step w and epoch τ: $\sigma_{(\tau,\omega,i)}$, and the fixed interstep bandwidth ϵ, we define:

- Intrastep affinities between hidden units i and j at time-step ω in epoch τ:

$$\boldsymbol{K}^{(\tau,\omega)}_{\text{intrastep}}(i,j) = \exp\left(-\frac{\| \boldsymbol{T}(\tau,\omega,i) - \boldsymbol{T}(\tau,\omega,j) \|_2^{\alpha}}{\sigma^{\alpha}_{(\tau,\omega,i)}}\right)$$

- Interstep affinities between a hidden unit i and itself at different time-steps and epochs:

$$\boldsymbol{K}^{(i)}_{\text{interstep}}((\tau,\omega),(\eta,\nu)) = \exp\left(-\frac{\| \boldsymbol{T}(\tau,\omega,i) - \boldsymbol{T}(\eta,\nu,i) \|_2^{2}}{\epsilon^2}\right)$$

The bandwidth $\sigma_{(\tau,\omega,i)}$ of the α-decay kernel is set to be the distance of unit i at time-step w from epoch n to its k-th nearest neighbor across units at that time-step and epoch: $\sigma_{(\tau,\omega,i)} = d_k(\boldsymbol{T}(\tau,\omega,i), \boldsymbol{T}(\tau,\omega,:))$, where $d_k(z,Z)$ denotes the ℓ_2 distance from z to its k-th nearest neighbor in Z. We used $k=5$ in all the results presented. The use of this adaptive bandwidth means the kernel is not symmetric and thus requires symmetrization. In the interstep affinities $\boldsymbol{K}^{(i)}_{\text{interstep}}$, we use a fixed-bandwidth Gaussian kernel $\epsilon = \frac{1}{nsm}\sum_{\tau=1}^{n}\sum_{\omega=1}^{s}\sum_{i=1}^{m} d_k(\boldsymbol{T}(\tau,\omega,i), \boldsymbol{T}(:,:,i))$, the average across all time-steps in all epochs and all units of the distance of unit i at time-step t to its k_{th} nearest neighbor among the set consisting of the same unit i at all steps.

The combined kernel matrix of these two matrices contains one row and column for each unit at each time-step in each epoch, such that the intrastep affinities form a block diagonal matrix and the interstep affinities form off-diagonal blocks composed of diagonal matrices (Fig. A.1).

$$\boldsymbol{K}((\tau,\omega,i),(\eta,\nu,j)) = \begin{cases} \boldsymbol{K}^{(\tau,\omega)}_{\text{intrastep}}(i,j) & \text{if } (\tau,\omega) = (\eta,\nu) \\ \boldsymbol{K}^{(i)}_{\text{interstep}}((\tau,\omega),(\eta,\nu)) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

We symmetrize this final kernel as $\boldsymbol{K'} = \frac{1}{2}(\boldsymbol{K} + \boldsymbol{K}^T)$, and row-normalize it to obtain $\boldsymbol{P} = \boldsymbol{D}^{-1}\boldsymbol{K'}$, where $\boldsymbol{D}$ is a diagonal matrix whose entries are row sums of $\boldsymbol{K'}$ and which $\boldsymbol{P}$ represents a random walk over all units at all time-steps in all epochs, where propagating from one state to another is conditional on the transition probabilities between time-step ω in epoch τ and time-step ν in epoch η. PHATE is applied to $\boldsymbol{P}$ to visualize the tensor $\boldsymbol{T}$ in two or three dimensions. This resulting visualization thus simultaneously captures information regarding the evolution of the units across both time-steps and epochs.

5 Results

We demonstrate the ability of MM-PHATE to capture useful properties of RNN learning on two datasets: 1) The Area2Bump dataset (Chowdhury et al., 2022) consisting of neural activity recorded from Brodmann's area 2 of the somatosensory cortex while macaque monkeys performed a slightly modified version of a standard center-out reaching task and 2) The Human Activity Recognition (HAR) Using Smartphones dataset (Reyes-Ortiz et al., 2012), kinematic recordings of 30 subjects performing daily activities with a smartphone embedded with an inertial measurement unit.

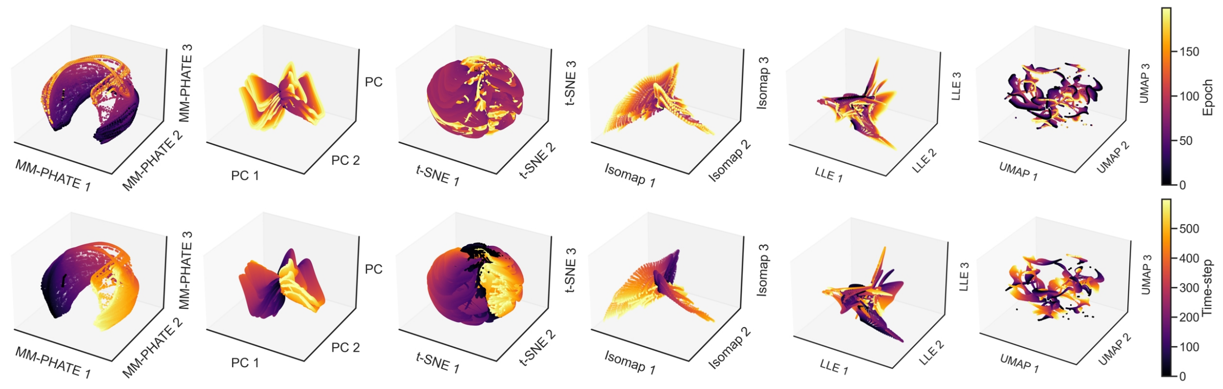


Figure 2: Area2Bump: Visualization of a 20-unit LSTM network trained for 200 epochs. The visualizations are generated using MM-PHATE, PCA, t-SNE, Isomap, Locally Linear Embedding(LLE), and Uniform Manifold Approximation and Projection(UMAP) from left to right, respectively. Points are colored based on epoch (top row) or time-step (bottom row).

5.1 Neural activity

We begin with the Area2Bump dataset, which consists of spiking activity data from macaques (Chowdhury et al., 2022). To analyze the learning dynamics of an LSTM network, we trained a single-layer, 20-unit LSTM on the Area2Bump dataset to classify arm-reaching directions. We then applied MM-PHATE to visualize how its hidden representations evolved during training. In an example session where our network achieved a validation accuracy of 74%, we applied the MM-PHATE visualization to the tensor $\boldsymbol{T}$ that consisted of the network activations of all 20 units over 600 time-steps for each of 200 training epochs. In practice, we sampled time-steps and epochs to reduce memory load. Each point in the visualization represents a hidden unit at a given time-step in a given epoch (Fig. 2). Here we compare MM-PHATE to five other dimensionality reduction techniques: PCA, t-SNE, Isomap, LLE, UMAP, as well as M-PHATE in the supplement (Fig. A.7), using the same $\boldsymbol{T}$ tensor. We first flattened $\boldsymbol{T}$ along the epoch, time-step, and unit axis, and embedded it with each of the dimensionality reduction methods.

Notably, the MM-PHATE visualization reveals a distinct split in representations during the later time-steps and epochs, highlighting unique learning patterns as the model converges. For instance, there should be visible shifts in representation before epoch 100, corresponding to the significant performance change during the initial training phase (Fig. 3). PCA and Isomap, while capturing a seemingly smooth transition, fail to reveal distinct differences between early and late epochs. In contrast, t-SNE produced a visualization that lacked continuity across time steps and epochs, likely due to its high sensitivity to noise. Both LLE and UMAP were more effective at clustering each hidden unit's trajectory. However, like PCA and Isomap, these methods do not clearly highlight the transformation in representation between early and late epochs.

5.1.1 Intra-step entropy

To evaluate the effectiveness of our embedding in capturing meaningful structures within the network's hidden dynamics, we next analyzed the flow of information during training (Fig. 3). Specifically, we aimed to analyze the spread of points in the MM-PHATE space, which corresponds to the diversity of the network's internal representations. To quantify this property, we computed the entropy between hidden units in the embedded space at each time-step ω and compared these intra-step entropies with the accuracy and loss metrics recorded at the end of each training epoch. Conceptually, we model a general RNN trained on dataset $\boldsymbol{X}$ with label $\boldsymbol{L}$ as a Markov chain ($\boldsymbol{L} \to \boldsymbol{X} \to \boldsymbol{R}$). This allows us to estimate the mutual information between $\boldsymbol{X}$ and $\boldsymbol{R}$ as $I(\boldsymbol{X}, \boldsymbol{R}) = H(\boldsymbol{R}) - H(\boldsymbol{R}|\boldsymbol{X})$, where $H(\boldsymbol{R})$ and $H(\boldsymbol{R}|\boldsymbol{X})$ are the marginal and conditional entropies. Given the deterministic nature of RNNs, $H(\boldsymbol{R}|\boldsymbol{X})$ equates to zero, indicating that $H(\boldsymbol{R})$ at each time-step ω reflects the input information the network retains during training (Tishby & Zaslavsky, 2015; Cheng et al., 2019).

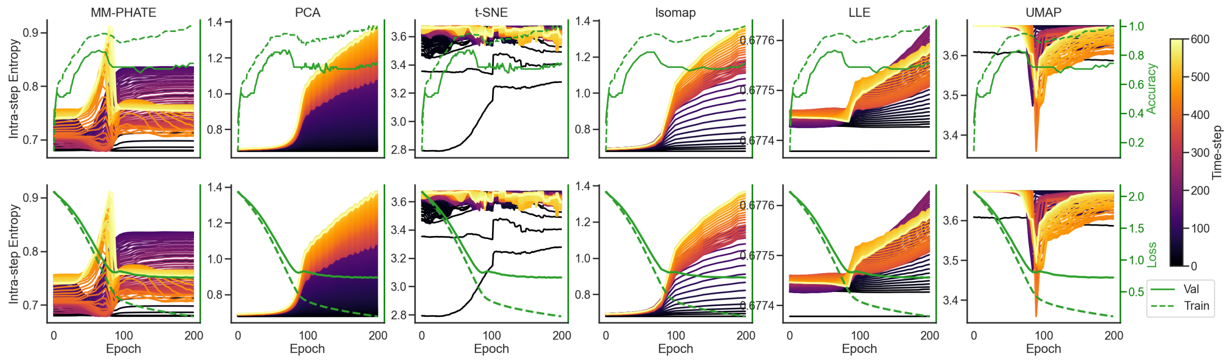


Figure 3: Intra-step entropy of all hidden units in embedding space at each time-step in each epoch, compared to training and validation accuracy (top) and losses (bottom), comparing embeddings of MM-PHATE, PCA, t-SNE, Isomap, LLE, and UMAP.

Our analysis of the MM-PHATE embedding revealed a general increase in entropy (Fig. 3) around epoch 100, where the network begins to overfit. We hypothesize that this rise is due to the memorization of noise or other nuisance variance in the training data. Changes in entropy at specific time steps and epochs coincided with shifts in model performance, suggesting that MM-PHATE effectively captures and retains dynamic information critical to the learning process. For example, fluctuations in validation accuracy before epoch 30 were not reflected in the training loss curve but were detected by MM-PHATE. This indicates that MM-PHATE offers insights beyond those provided by accuracy and loss metrics alone.

MM-PHATE also highlights information processing patterns across time steps, which are inaccessible through epoch-level accuracy and loss measurements. By generating intra-step entropy plots for each time step, MM-PHATE enables comparisons of temporal dynamics within the network. We observed that entropy at earlier time steps increases and plateaus around epoch 100, while later time steps show spiking activity around the same epoch before returning to earlier levels. Although the reason behind these diverging patterns remains unclear, MM-PHATE clearly reveals additional insights into the network's internal dynamics.

We conducted the same intra-step entropy analysis using other dimensionality reduction methods (Fig. 3). MM-PHATE captures an earlier rise in entropy compared to PCA and Isomap, which only show entropy changes after overfitting has begun, thus reflecting a more nuanced transition into memorization. PCA failed to capture early performance transitions, likely due to its linear nature and emphasis on global structure. Similarly, Isomap, LLE, and UMAP did not detect significant performance-related dynamics before overfitting. Although PCA and Isomap showed an increase in entropy, this occurred only after the training loss had plateaued. By contrast, MM-PHATE detected an earlier rise in entropy, corresponding to the transition into the plateau phase and more accurately reflecting the underlying dynamics. Meanwhile, t-SNE struggled to capture both global structure and entropy changes, underscoring MM-PHATE's superior ability to reveal hidden learning dynamics.

Additionally, to verify that the MM-PHATE embedding and intra-step entropy indeed reflect the information retained by the network, we conducted a random label analysis. Inspired by the experiments in Fischer (2020), which link mutual information to model performance and robustness, we manipulated the level of overfitting by training models with varying degrees of label shuffling. We then tracked changes in the representation structure over training. With minimal or no shuffling, the embeddings formed distinct clusters representing meaningful class structure. As overfitting increased, these clusters blurred and eventually collapsed, reflecting a loss of task-relevant information (Fig. A.2). This was further quantified using Jensen-Shannon Divergence (JSD) and Total Variation Distance (TVD) (Fig. A.6). Additional details and results are provided in Appendix A.4.

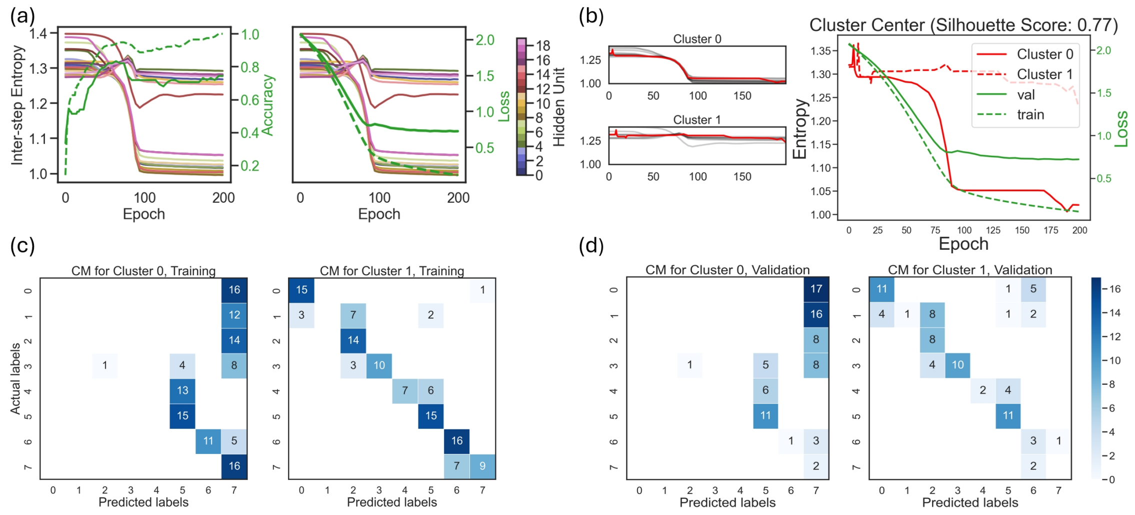


Figure 4: a) Inter-step entropy of each hidden unit across epochs, alongside model accuracy (left) and loss (right). b) Clusters of hidden units from the model's inter-step entropy trajectories across epochs. Left: Trajectories of the units in cluster 0 (12 units) and cluster 1 (8 units). Right: cluster center trajectories across epochs. c-d) Confusion matrices of clusters on training and validation data.

5.1.2 Inter-step entropy

To further evaluate the quality of our embedding, we quantified the inter-step entropy of hidden unit activations across various time steps ω (Fig. 4a). While intra-step entropy measures diversity within a single time step, inter-step entropy tracks how unit activations change across time. This allows us to identify units that contribute more to temporal dependencies. Our analysis revealed distinct patterns among these units. Certain units exhibited significantly higher inter-step entropy, indicating greater sensitivity to input changes over time and a potential role in capturing temporal dependencies. These units peaked in inter-step entropy around epoch 80, aligning with peaks in intra-step entropy at later time steps, suggesting their contribution to increased mutual information during these periods.

To validate these findings, we clustered the hidden units into two groups using k-means clustering with the Dynamic Time Warping (DTW) metric, expecting one cluster to exhibit higher inter-step entropy (Fig. 4b). Clustering revealed distinct inter-step entropy trajectories across epochs. Cluster 1 (8 units) consisted of the high-entropy units identified earlier, while units in cluster 0 (12 units) showed a sharp decline in inter-step entropy around epoch 100.

To further explore these clusters, we created sub-networks based on the units from each cluster, maintaining the original network architecture. We assessed their learning capabilities by comparing confusion matrices and evaluating their performance on both training and validation data (Fig. 4c-d). Interestingly, despite having fewer units, cluster 1 achieved significantly better performance on both datasets. This supports our earlier hypothesis that cluster 1 captures more complex temporal dependencies from the input. These results demonstrate that our visualization and clustering approach effectively reveals critical differences in how information is processed and represented within the network.

Previous studies have emphasized the importance of clustering hidden unit activations to understand the quality of learned representations (Su & Shlizerman, 2020; Oliva & Lago-Fernández, 2021; Ming et al., 2017). Our findings highlight the importance of capturing each unit's temporal dynamics and community structure, as well as how these structures evolve across time steps and epochs.

Comparisons with other dimensionality reduction techniques revealed significant limitations (Fig. A.14). PCA, Isomap, LLE, and UMAP were unable to capture subtle dynamic variations, particularly during the early epochs before overfitting. While t-SNE reflected general performance transitions, its trajectories were noisy and lacked precision. Most importantly, none of these methods effectively differentiated the learning behavior of individual units or captured their community structure. These results underscore the superior ability of MM-PHATE to preserve and reveal the hidden dynamics of RNNs during training.

In the appendix, we present results for different RNN architectures, namely GRU (A.6) and Vanilla RNN (A.7), both with 20 hidden units. Using the Area2Bump dataset, we additionally tried different LSTM sizes (10, 20, 30, 40, 50) (A.8). We observed consistent visualizations when varying network size. Changing the learning rate helped confirm that the visualization indeed reflects the model learning, i.e. the resulting change of entropies should always follow the change of model performance.

5.2 Analysis with Human Activity Recognition Model

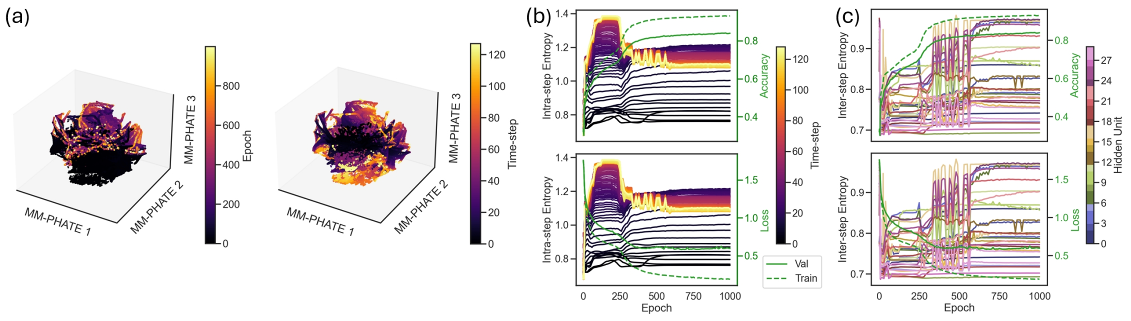


Figure 5: a) MM-PHATE visualization of a 30-unit LSTM network trained on HAR data, colored by epoch (left) and time-step (right). b) Intra-step entropy, c) and inter-step entropy.

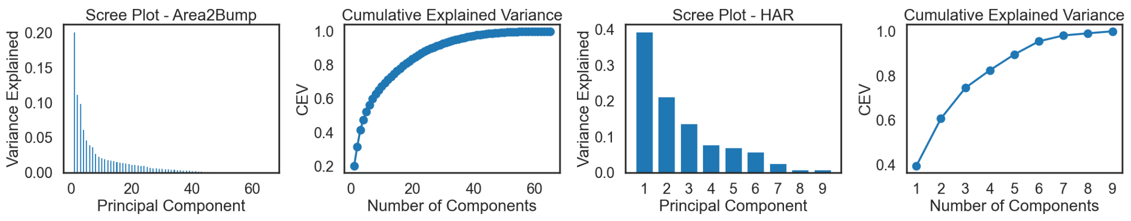


Figure 6: PCA on the 2 datasets. The Area2Bump data requires 35 PCs (left) to cover 95% of the variance. The HAR data (right) requires 6 PCs to cover 95% of variance.

We next analyzed an action recognition dataset by training a 30-unit LSTM network designed for kinematics-based Human Activity Recognition (HAR) (Reyes-Ortiz et al., 2012). The model was trained for 1000 epochs, achieving a final validation accuracy of 84% (Fig. 5). We applied MM-PHATE to the tensor of hidden unit activations and repeated the analysis performed on the Area2Bump dataset.

Similar to the Area2Bump model, the HAR network exhibited an increase in intra-step entropy at the onset of overfitting. Notably, entropy gradually increased across time steps before epoch 300. After this point, entropy at later time steps dropped significantly, while entropy at earlier time steps remained stable throughout training. This pattern suggests a reduction in mutual information between the input and hidden states as the network processed more input over time. This behavior aligns with findings by Farrell et al. (2022), who reported similar dynamics of information expansion and compression in trained RNNs. Their work demonstrated how gradient-based learning mechanisms balance expansion and compression processes to develop robust representations.

Inter-step entropy analysis provided further insights. Entropy in many hidden units began to increase sharply around epoch 300, coinciding with a decrease in intra-step entropy and an improvement in model performance. High inter-step entropy suggests that these units become more sensitive to input changes across time and play a critical role in learning complex dependencies. Despite this increase in time-dependent information, the network appears to compress this information into a more compact overall representation. This process is consistent with the information bottleneck theory, which posits that deep networks undergo a fitting phase followed by a compression phase to distill useful information and improve generalization (Cheng et al., 2019; Tishby & Zaslavsky, 2015; Butakov et al., 2023).

Given these observations, this raises the question of why the Area2Bump model does not exhibit a similar compression phase. Cheng et al. (2019) suggest that models with limited generalizability often fail to display a compression phase, particularly when applied to complex datasets with relatively simple network architectures. Since both models have comparable structures, we examined the complexity of the two datasets by analyzing the number of principal components (PCs) needed to explain 95% of the variance. The Area2Bump data required 35 PCs, compared to only 6 for the HAR data (Fig. 6). This significant difference in data complexity likely accounts for the absence of a compression phase in the Area2Bump model.

Our analysis demonstrates that information compression in RNNs occurs both across time steps and epochs, closely aligning with performance improvements. These findings affirm the utility of the information bottleneck theory in understanding RNN learning dynamics and confirm that MM-PHATE effectively uncovers detailed insights into the evolution of hidden representations. This aligns with insights from the conditional entropy bottleneck framework, which suggests that balancing the retention of essential input information while discarding irrelevant noise is crucial for improving both model generalization and robustness (Fischer, 2020).

6 Conclusion

In this paper, we introduced MM-PHATE, a dimensionality reduction technique designed to visualize the hidden dynamics of RNNs during training. MM-PHATE captures the evolution of these dynamics across both time steps and epochs, offering insights beyond traditional metrics like accuracy and loss curves, as well as other dimensionality reduction methods. Our entropy-based analysis shows that MM-PHATE can reveal distinct learning behaviors and the roles of hidden units in information flow, consistent with principles from the information bottleneck theory. We also quantitatively confirmed that the information retained by the network is reflected in the MM-PHATE embedding. This approach is particularly valuable in data-limited settings, as it does not require validation data. Additionally, by analyzing the hidden state dynamics throughout training, MM-PHATE provides deeper insights into the model's learning trajectory and the evolving structure of its representation.

Despite its strengths, our approach relies on several assumptions. One key assumption is the continuity of time and epochs, encoded through the structured graph kernel. This assumption may not hold in cases of large learning rates, multiple significant restarts, or discontinuous activation functions, potentially leading to less informative visualizations (Moon et al., 2019).

Additionally, while MM-PHATE builds on a computationally efficient PHATE implementation, the memory complexity of its multiway-multislice kernel poses challenges when scaling to larger architectures or datasets. However, PHATE has been shown to be robust to sub-sampling, maintaining strong correlations even with significant data reduction (Moon et al., 2019). For example, their results indicate that PHATE performs well with as little as 5% of data points retained. This supports kernel sub-sampling as a practical strategy for improving scalability. Other solutions, such as graph partitioning and data point merging (Kuchroo et al., 2022; Holtz et al., 2022), can further alleviate memory issues without compromising structural integrity.

Furthermore, MM-PHATE does not yet incorporate internal model structures such as attention mechanisms found in transformers (Vaswani et al., 2017). Extending MM-PHATE to analyze transformers—now dominant in sequential data analysis—remains a promising area for future work. However, RNNs continue to play a crucial role in many fields (Deo et al., 2024), particularly in scenarios with limited data where transformers may be impractical. Therefore, while MM-PHATE is optimized for RNNs, future adaptations could offer new insights into the hidden dynamics of transformer architectures.

References

Omri Barak. Recurrent neural networks as versatile tools of neuroscience research. *Current Opinion in Neurobiology*, 46:1–6, October 2017. ISSN 09594388. doi: 10.1016/j.conb.2017.06.003.

Ivan Butakov, Aleksander Tolmachev, Sofia Malanchuk, Anna Neopryatnaya, Alexey Frolov, and Kirill Andreev. Information bottleneck analysis of deep neural networks via lossy compression, May 2023. URL `http://arxiv.org/abs/2305.08013`.

Yuexing Chen and Jiarun Li. Recurrent neural networks algorithms and applications. In *2021 2nd International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE)*, pp. 38–43, Zhuhai, China, September 2021. IEEE. ISBN 978-1-66542-709-8. doi: 10.1109/ICBASE53849.2021.00015. URL `https://ieeexplore.ieee.org/document/9696155/`.

Hao Cheng, Dongze Lian, Shenghua Gao, and Yanlin Geng. Utilizing information bottleneck to evaluate the capability of deep neural networks for image classification. *Entropy*, 21(5):456, May 2019. ISSN 1099-4300. doi: 10.3390/e21050456.

Raeed Chowdhury, Joshua Glaser, and Lee Miller. Data from: Area 2 of primary somatosensory cortex encodes kinematics of the whole arm, 2022. URL `https://doi.org/10.5061/dryad.nk98sf7q7`. Dataset.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, December 2014. URL `http://arxiv.org/abs/1412.3555`. arXiv:1412.3555 [cs].

Ronald R. Coifman and Stéphane Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30, July 2006. ISSN 10635203. doi: 10.1016/j.acha.2006.04.006.

Darrel R. Deo, Francis R. Willett, Donald T. Avansino, Leigh R. Hochberg, Jaimie M. Henderson, and Krishna V. Shenoy. Brain control of bimanual movement enabled by recurrent neural networks. *Scientific Reports*, 14(1):1598, January 2024. ISSN 2045-2322. doi: 10.1038/s41598-024-51617-3. URL `https://www.nature.com/articles/s41598-024-51617-3`.

Matthew Farrell, Stefano Recanatesi, Timothy Moore, Guillaume Lajoie, and Eric Shea-Brown. Gradient-based learning drives robust representations in recurrent neural networks by balancing compression and expansion. *Nature Machine Intelligence*, 4(6):564–573, June 2022. ISSN 2522-5839. doi: 10.1038/s42256-022-00498-0.

Ian Fischer. The conditional entropy bottleneck. *Entropy*, 22(9):999, September 2020. ISSN 1099-4300. doi: 10.3390/e22090999.

Scott Gigante, Adam S. Charles, Smita Krishnaswamy, and Gal Mishne. Visualizing the PHATE of neural networks, August 2019. URL `http://arxiv.org/abs/1908.02831`. arXiv:1908.02831 [cs, stat].

Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, October 2017. ISSN 2162-237X, 2162-2388. doi: 10.1109/TNNLS.2016.2582924. arXiv:1503.04069 [cs].

Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1):388–427, January 2021. ISSN 01692070. doi: 10.1016/j.ijforecast.2020.06.008.

Akinori Hidaka and Takio Kurita. Consecutive dimensionality reduction by canonical correlation analysis for visualization of convolutional neural networks. In *Proceedings of the ISCIE international symposium on stochastic systems theory and its applications*, volume 2017, pp. 160–167. The ISCIE Symposium on Stochastic Systems Theory and Its Applications, 2017.

Sepp Hochreiter and Jürgen Schmidhuber. Lstm can solve hard long time lag problems. *Advances in neural information processing systems*, 9, 1996.

Chester Holtz, Gal Mishne, and Alexander Cloninger. Evaluating disentanglement in generative models without knowledge of latent factors. In *Topological, Algebraic and Geometric Learning Workshops 2022*, pp. 161–171. PMLR, 2022.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks, November 2015. URL `http://arxiv.org/abs/1506.02078`. arXiv:1506.02078 [cs].

Manjot Kaur and Aakash Mohta. A review of deep learning with recurrent neural network. In *2019 International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pp. 460–465, Tirunelveli, India, November 2019. IEEE. ISBN 978-1-72812-119-2. doi: 10.1109/ICSSIT46314.2019.8987837. URL `https://ieeexplore.ieee.org/document/8987837/`.

Yassin Khalifa, Danilo Mandic, and Ervin Sejdić. A review of hidden markov models and recurrent neural networks for event detection and localization in biomedical signals. *Information Fusion*, 69:52–72, May 2021. ISSN 15662535. doi: 10.1016/j.inffus.2020.11.008.

Manik Kuchroo, Jessie Huang, Patrick Wong, Jean-Christophe Grenier, Dennis Shung, Alexander Tong, Carolina Lucas, Jon Klein, Daniel B. Burkhardt, Scott Gigante, Abhinav Godavarthi, Bastian Rieck, Benjamin Israelow, Michael Simonov, Tianyang Mao, Ji Eun Oh, Julio Silva, Takehiro Takahashi, Camila D. Odio, Arnau Casanovas-Massana, John Fournier, Yale IMPACT Team, Abeer Obaid, Adam Moore, Alice Lu-Culligan, Allison Nelson, Anderson Brito, Angela Nunez, Anjelica Martin, Anne L. Wyllie, Annie Watkins, Annsea Park, Arvind Venkataraman, Bertie Geng, Chaney Kalinich, Chantal B. F. Vogels, Christina Harden, Codruta Todeasa, Cole Jensen, Daniel Kim, David McDonald, Denise Shepard, Edward Courchaine, Elizabeth B. White, Eric Song, Erin Silva, Eriko Kudo, Giuseppe DeIuliis, Haowei Wang, Harold Rahming, Hong-Jai Park, Irene Matos, Isabel M. Ott, Jessica Nouws, Jordan Valdez, Joseph Fauver, Joseph Lim, Kadi-Ann Rose, Kelly Anastasio, Kristina Brower, Laura Glick, Lokesh Sharma, Lorenzo Sewanan, Lynda Knaggs, Maksym Minasyan, Maria Batsu, Maria Tokuyama, M. Cate Muenker, Mary Petrone, Maxine Kuang, Maura Nakahata, Melissa Campbell, Melissa Linehan, Michael H. Askenase, Michael Simonov, Mikhail Smolgovsky, Nathan D. Grubaugh, Nicole Sonnert, Nida Naushad, Pavithra Vijayakumar, Peiwen Lu, Rebecca Earnest, Rick Martinello, Roy Herbst, Rupak Datta, Ryan Handoko, Santos Bermejo, Sarah Lapidus, Sarah Prophet, Sean Bickerton, Sofia Velazquez, Subhasis Mohanty, Tara Alpert, Tyler Rice, Wade Schulz, William Khoury-Hanold, Xiaohua Peng, Yexin Yang, Yiyun Cao, Yvette Strong, Shelli Farhadian, Charles S. Dela Cruz, Albert I. Ko, Matthew J. Hirn, F. Perry Wilson, Julie G. Hussin, Guy Wolf, Akiko Iwasaki, and Smita Krishnaswamy. Multiscale PHATE identifies multimodal signatures of covid-19. *Nature Biotechnology*, 40(5):681–691, May 2022. ISSN 1087-0156, 1546-1696. doi: 10.1038/s41587-021-01186-x.

Bum Chul Kwon, Min-Je Choi, Joanne Taery Kim, Edward Choi, Young Bin Kim, Soonwook Kwon, Jimeng Sun, and Jaegul Choo. RetainVis: Visual analytics with interpretable and interactive recurrent neural networks on electronic medical records. *IEEE Transactions on Visualization and Computer Graphics*, 25 (1):299–309, January 2019. ISSN 1077-2626, 1941-0506, 2160-9306. doi: 10.1109/TVCG.2018.2865027. arXiv:1805.10724 [cs, stat].

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and understanding neural models in NLP, January 2016. URL `http://arxiv.org/abs/1506.01066`. arXiv:1506.01066 [cs].

Zachary C. Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning, October 2015. URL `http://arxiv.org/abs/1506.00019`. arXiv:1506.00019 [cs].

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

Andrzej Maćkiewicz and Waldemar Ratajczak. Principal components analysis (PCA). *Computers & Geosciences*, 19(3):303–342, 1993. ISSN 0098-3004. doi: https://doi.org/10.1016/0098-3004(93)90090-R.

Tomas Mikolov, Armand Joulin, Sumit Chopra, Michael Mathieu, and Marc'Aurelio Ranzato. Learning longer memory in recurrent neural networks. *arXiv preprint arXiv:1412.7753*, 2014.

Yao Ming, Shaozu Cao, Ruixiang Zhang, Zhen Li, Yuanzhe Chen, Yangqiu Song, and Huamin Qu. Understanding hidden memories of recurrent neural networks. In *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 13–24, Phoenix, AZ, October 2017. IEEE. ISBN 978-1-5386-3163-8. doi: 10.1109/VAST.2017.8585721. URL `https://ieeexplore.ieee.org/document/8585721/`.

Kevin R. Moon, David Van Dijk, Zheng Wang, Scott Gigante, Daniel B. Burkhardt, William S. Chen, Kristina Yim, Antonia Van Den Elzen, Matthew J. Hirn, Ronald R. Coifman, Natalia B. Ivanova, Guy Wolf, and Smita Krishnaswamy. Visualizing structure and transitions in high-dimensional biological data. *Nature Biotechnology*, 37(12):1482–1492, December 2019. ISSN 1087-0156, 1546-1696. doi: 10.1038/s41587-019-0336-3.

Christian Oliva and Luis F. Lago-Fernández. Stability of internal states in recurrent neural networks trained on regular languages. *Neurocomputing*, 452:212–223, September 2021. ISSN 09252312. doi: 10.1016/j.neucom.2021.04.058.

James O. Ramsay. Some statistical considerations in multidimensional scaling. *ETS Research Bulletin Series*, 1966(1):i–96, 1966. doi: https://doi.org/10.1002/j.2333-8504.1966.tb00976.x. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/j.2333-8504.1966.tb00976.x`.

Paulo E Rauber, Samuel G Fadel, Alexandre X Falcao, and Alexandru C Telea. Visualizing the hidden activity of artificial neural networks. *IEEE transactions on visualization and computer graphics*, 23(1): 101–110, 2016.

Jorge Reyes-Ortiz, Davide Anguita, Alessandro Ghio, Luca Oneto, and Xavier Parra. Human Activity Recognition Using Smartphones. UCI Machine Learning Repository, 2012. DOI: https://doi.org/10.24432/C54S4K.

Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent advances in recurrent neural networks, February 2018. URL `http://arxiv.org/abs/1801.01078`. arXiv:1801.01078 [cs].

Hendrik Strobelt, Sebastian Gehrmann, Hanspeter Pfister, and Alexander M. Rush. LSTMVis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):667–676, January 2018. ISSN 1077-2626. doi: 10.1109/TVCG.2017.2744158.

Kun Su and Eli Shlizerman. Clustering and recognition of spatiotemporal features through interpretable embedding of sequence to sequence recurrent neural networks. *Frontiers in Artificial Intelligence*, 3:70, September 2020. ISSN 2624-8212. doi: 10.3389/frai.2020.00070.

David Sussillo and Omri Barak. Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Computation*, 25(3):626–649, March 2013. ISSN 0899-7667. doi: 10.1162/NECO_a_00409.

Zhiyuan Tang, Ying Shi, Dong Wang, Yang Feng, and Shiyue Zhang. Memory visualization for gated recurrent neural networks in speech recognition, February 2017. URL `http://arxiv.org/abs/1609.08789`. arXiv:1609.08789 [cs].

Joshua B. Tenenbaum, Vin De Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.290.5500.2319.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle, March 2015. URL `http://arxiv.org/abs/1503.02406`. arXiv:1503.02406 [cs].

Manuel Titos, Luz Garcia, Milad Kowsari, and Carmen Benitez. Toward knowledge extraction in classification of volcano-seismic events: Visualizing hidden states in recurrent neural networks. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 15:2311–2325, 2022. ISSN 1939-1404, 2151-1535. doi: 10.1109/JSTARS.2022.3155967.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL `https://arxiv.org/pdf/1706.03762.pdf`.

Zijie J. Wang, Robert Turko, Omar Shaikh, Haekyu Park, Nilaksh Das, Fred Hohman, Minsuk Kahng, and Duen Horng Polo Chau. CNN explainer: Learning convolutional neural networks with interactive visualization. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1396–1406, February 2021. ISSN 1077-2626, 1941-0506, 2160-9306. doi: 10.1109/TVCG.2020.3030418.

Wei Yu, Kuiyuan Yang, Yalong Bai, Hongxun Yao, and Yong Rui. Visualizing and comparing convolutional neural networks, December 2014. URL `http://arxiv.org/abs/1412.6631`. arXiv:1412.6631 [cs].

A Appendix

A.1 Datasets

We employed two datasets.

1. The Area2Bump dataset (Chowdhury et al., 2022) consists of neural activity recorded from Brodmann's area 2 of the somatosensory cortex while macaque monkeys performed a slightly modified version of a standard center-out reaching task. Dataset license: CC0 1.0. According to the authors, data was collected consistently "with the guide for the care and use of laboratory animals and approved by the institutional animal care and use committee of Northwestern University under protocol #IS00000367". This dataset, collected from monkeys thus does not contain personally identifiable information. Since it is neural data, we do not consider it to be offensive content.

   Data Statistics: The dataset includes 193 samples, split into 115 training samples and 78 testing samples. Each sample consists of 600 time steps with 65 features per time step. For the training set, the mean values across features range from 0.0001 to 0.03, with standard deviations between 0.001 and 0.02, and maximum values ranging from 0.003 to 0.12. In the test set, the mean values range from 0.00008 to 0.03, standard deviations from 0.0007 to 0.018, and maximum values from 0.0007 to 0.13.

2. The Human Activity Recognition (HAR) Using Smartphones dataset (Reyes-Ortiz et al., 2012), kinematic recordings of 30 subjects performing daily living activities with a smartphone embedded with an inertial measurement unit. Dataset license: CC BY 4.0. Information relating to participant consent was not found relating to this dataset. This dataset does not reveal participant name or identifiable information. The kinematics contained in the dataset are not considered offensive content.

   Data Details: The dataset includes six activity classes (e.g., Walking, Sitting) based on accelerometer and gyroscope data sampled at 50 Hz. The sensor data has been pre-processed with noise filtering and separated into gravitational and body motion components. Data is windowed into 2.56-second segments (128 data points) with 50% overlap, resulting in 561-dimensional feature vectors per window. The dataset is split into 70% training (21 subjects, 7352 samples) and 30% testing (9 subjects, 2947 samples).

   Data Statistics: For the training set, mean values across features range from -0.0008 to 0.8, with standard deviations between 0.1 and 0.41, and values spanning from -5.97 to 5.75. For the test set, mean values range from -0.013 to 0.8, with standard deviations between 0.095 and 0.41, and values spanning from -3.43 to 3.47.

A.2 Model Training

We used TensorFlow's Keras API for all model training and validation.

The network in Section 5.1 was trained as follows. The Area2Bump dataset was randomly split into training and validation subsets containing an equal number of samples for each input class with an 8 to 2 ratio. Additional samples that would have made the training data uneven were added back to the validation subset to utilize all available samples. The network consists of an LSTM layer with 20 units. This is followed by a Flatten layer that converts the LSTM's output into a one-dimensional vector. Finally, a Dense layer with 8 units and a softmax activation function produces the output for the 8-class classification tasks. The network was trained with a batch size of 64. We used an Adam optimizer with a learning rate of $1e^{-4}$. During the training process, we recorded the activations from the LSTM layer into the activation tensor $\boldsymbol{T}$ for visualization.

The network in Section 5.2 was trained as follows. The HAR dataset was preprocessed and split by the authors into training and validation subsets according to the subjects with a 7 to 3 ratio. The network consists of an LSTM layer with 30 units and a Dense layer with 6 units and a softmax activation function produces the output for the 6-class classification tasks. The network was trained with a batch size of 32. We used an Adam optimizer with a learning rate of $2e^{-5}$. During the training process, we recorded the activations from the LSTM layer into the activation tensor $\boldsymbol{T}$ for visualization.

A.3 Implementation of visualization methods

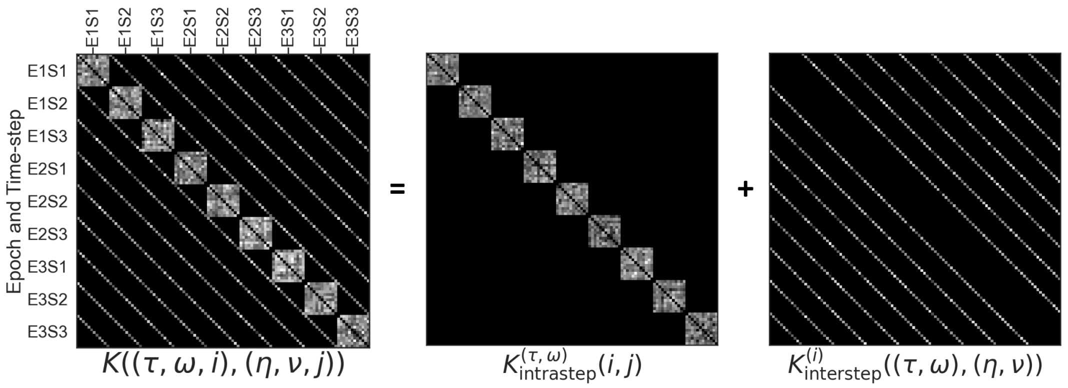

E1S1
E1S2
E1S3
E2S1
E2S2
E2S3
E3S1
E3S2
E3S3
Epoch and Time-step
=
+
$K((\tau,\omega,i),(\eta,\nu,j))$
$K^{(\tau,\omega)}_{\text{intrastep}}(i,j)$
$K^{(i)}_{\text{interstep}}((\tau,\omega),(\eta,\nu))$

Figure A.1: Example schematic of the multiway multislice kernel used in MM-PHATE for RNNs. The intrastep kernels represent the similarities between the graph nodes at the same time-steps. The interstep kernels represent the similarities between the nodes and themselves at different time-steps and epochs.

A.3.1 MM-PHATE

Due to memory constraints, we only used a subset of the tensor $\boldsymbol{T}$ for MM-PHATE computation. Specifically, for the Area2Bump model in section 5.1, epochs were sampled using an array combining the first 29 epochs with every 5th epoch thereafter to cover the initial rapid learning phase, and intrinsic steps were sampled using a linear space from 0 to the end (600), resulting in 100 evenly spaced steps. For the HAR model in Section 5.2, we sampled the epochs using an array combining the first 29 epochs with every 10th epoch thereafter. We utilized the M-PHATE package to construct our multiway multislice graphs and for the application of PHATE.

A.3.2 PCA

PCA was performed using the "PCA" class from the "sklearn.decomposition" package to reduce the dimensionality of the time trace tensor $\boldsymbol{T}$—recorded during training of the Area2Bump model—to three principal components.

A.3.3 t-SNE

In this analysis, we first conducted an initial dimensionality reduction on the same time trace tensor $\boldsymbol{T}$ with PCA to 15 principal components, which explains 99.93% of the variance in the activations. Subsequently, t-SNE with the Barnes-Hut approximation was performed using the "sklearn.manifold.TSNE" class.

A.3.4 Isomap

Due to memory constraints, we only used a subset of the tensor $\boldsymbol{T}$ for Isomap computation. Specifically, epochs were sampled using an array combining the first 29 epochs with every 10th epoch thereafter, and intrinsic steps were sampled using a linear space from 0 to the end (600), resulting in 50 evenly spaced steps. We first conducted an initial dimensionality reduction on the sampled tensor with PCA to 15 principal components using "sklearn.decomposition.PCA" package. Then, we applied Isomap using "sklearn.manifold.Isomap" class to reduce the dimensionality to 3.

A.4 Random Label Analysis

A.4.1 Experimental Setup

To investigate the effects of overfitting on hidden unit representations, we conducted a random label analysis, inspired by experiments in Fischer (2020). The goal was to observe how varying levels of label shuffling influence both the information retained in the network and the structure of the hidden representations, as captured by MM-PHATE embeddings and intra-step entropy measures.

We trained multiple LSTM models (1 layer, 100 hidden units) under different levels of label shuffling. For each experiment, a specified number of classes had their labels randomly reassigned, increasing the noise in the target variable. We randomly selected 10 combinations of shuffled classes for each condition. Models were trained using a learning rate of 0.0001, with other hyperparameters held constant. After 200 training epochs, most models achieved 0 training loss. We evaluated representations in the last training epoch across 600 time steps. The following metrics were extracted: Mean entropy and variance of entropy at each time step, computed over all hidden unit activations. We also looked at the maximum entropy and the entropy of the last time-step in the last epoch. Jensen-Shannon Divergence (JSD) and Total Variation Distance (TVD) of the entropy distribution compared to a reference model trained with correct labels.

A.4.2 Metrics and Analysis

Intra-step Entropy: Entropy at each time step was calculated to quantify the uncertainty in hidden unit activations. Mean entropy reflects the overall uncertainty, while variance of entropy indicates how much this uncertainty fluctuates over time.

Jensen-Shannon Divergence (JSD): JSD was used to measure the distributional divergence of entropy between models with shuffled and correct labels. Higher JSD values indicate that the network's representations have deviated significantly due to overfitting.

Total Variation Distance (TVD): TVD quantifies the absolute shift in entropy distribution. Like JSD, higher TVD values imply that the model's representation has undergone drastic changes, suggesting increased memorization.

A.4.3 Results

The results confirmed our hypothesis that overfitting reduces representation quality:

MM-PHATE embeddings: With minimal or no shuffling, the embeddings formed distinct clusters representing meaningful class structure. As overfitting increased, these clusters blurred and eventually collapsed, reflecting a loss of task-relevant information (Fig. A.2). The inter-step entropy (A.4) shows that the hidden units become more and more sensitive to temporal change in the input sequence as the level of overfitting increases.

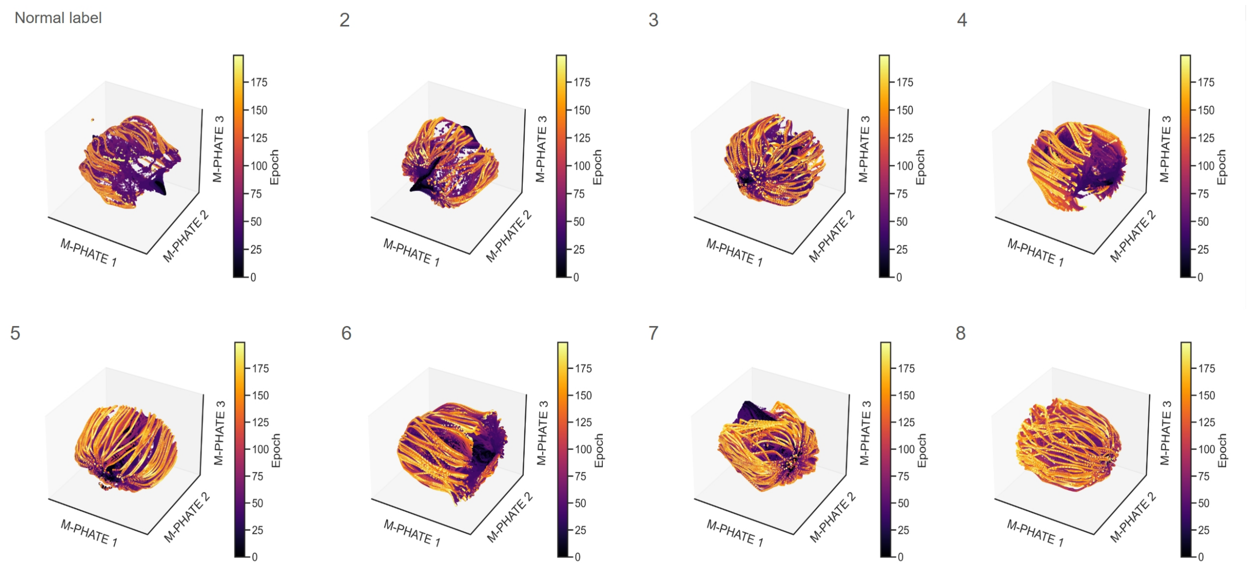


Figure A.2: Random Label: Visualization of 100-unit LSTM networks trained for 200 epochs on training data with different percentages of random labels. The number indicates the number of shuffled classes. Each point represents a hidden unit at a specific time-step in a given epoch throughout the entire training process. Points are colored based on epoch (top row) or time-step (bottom row)

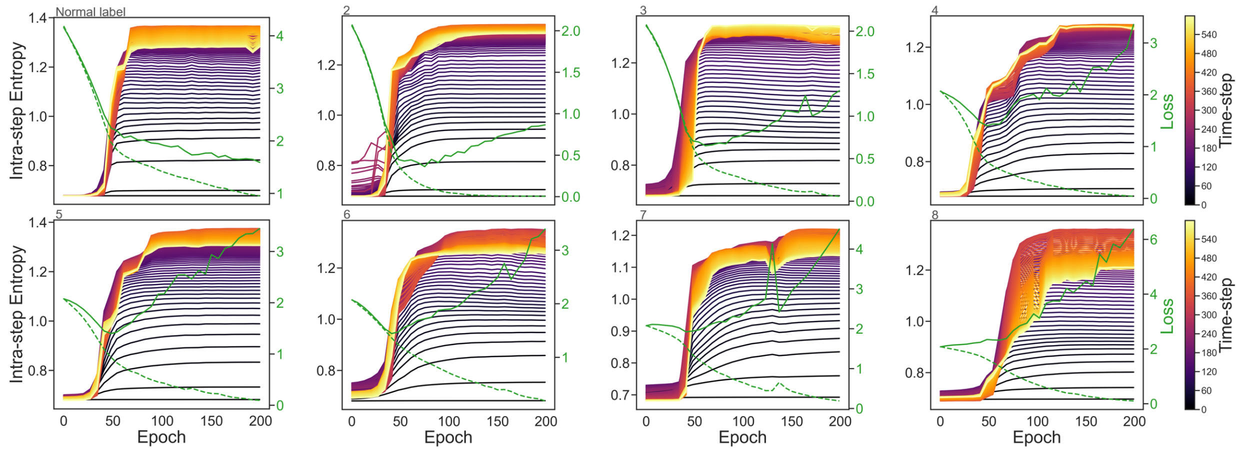


Figure A.3: Random Label: Intra-step entropy of the MM-PHATE embedding of 100-unit LSTM networks trained for 200 epochs on training data with different percentages of random label. The number indicates the number of shuffled classes. Each line represents the entropy at a given time-step across training epochs.

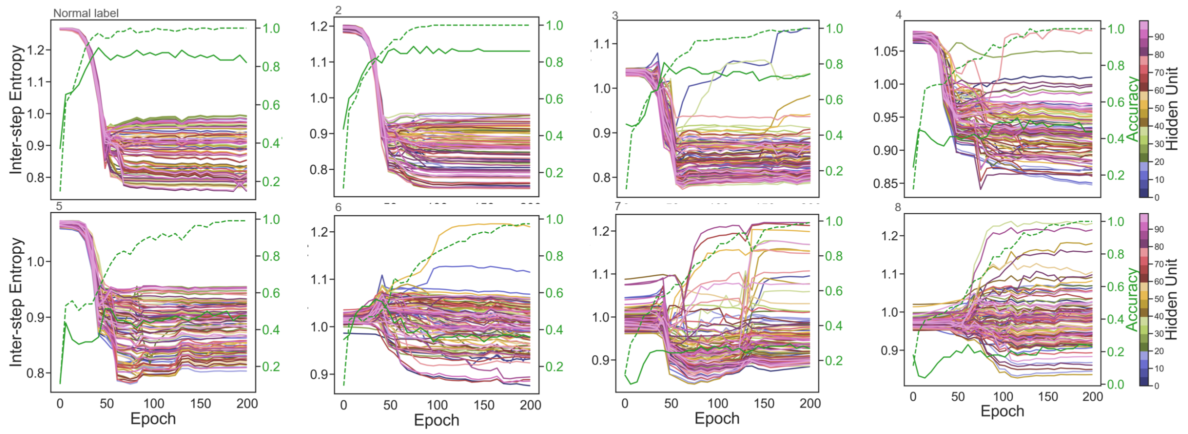


Figure A.4: Random Label: Inter-step entropy of the MM-PHATE embedding of 100-unit LSTM networks trained for 200 epochs on training data with different percentages of random label. The number indicates the number of shuffled classes. Each line represents the entropy of a hidden unit across time-steps at different epochs.

Intra-step entropy trends: Maximum, mean entropy, last-timestep intra-step entropy and variance decreased as more classes were shuffled, indicating that the network became increasingly rigid and deterministic (Fig. A.5).

Divergence metrics: We calculated JSD and TVD on both the MM-PHATE embedding and the entropy distribution across time-steps in the last epoch. Both JSD and TVD increased with shuffling, confirming significant shifts in the entropy distribution. These metrics demonstrate that overfitting alters the model's capacity to maintain structured representations, and that MM-PHATE successfully retained these dynamics in the lower dimension space (Fig. A.6).

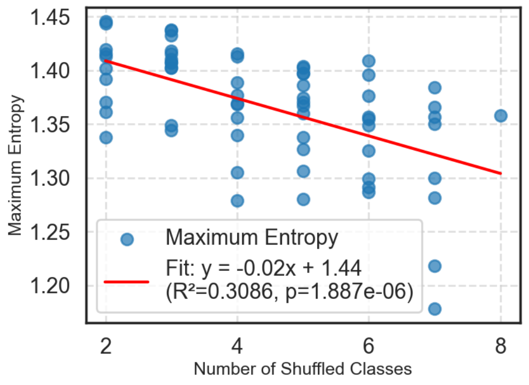
A.5 M-PHATE

M-PHATE was applied to the same 20-unit LSTM network trained on the Area2Bump dataset. The algorithm only incorporated the final state, or time-step, from each epoch. While the resulting visualization captures smooth transitions across epochs, it omits critical information from earlier time-steps. This loss of temporal resolution obscures insights into how the network processes input sequences over time—an essential aspect for understanding RNNs, for instance, how much sequential input information is retained by the network.

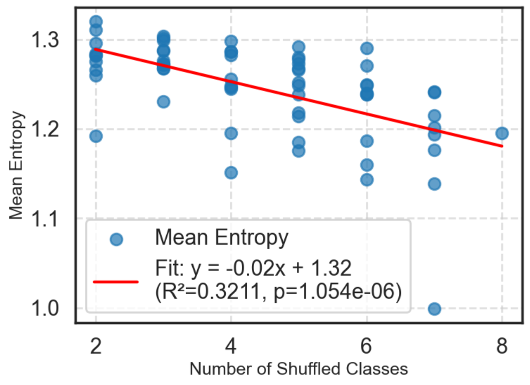
A.6 Area2Bump with GRU

Here is the same analysis as section 5.1 using GRU (Fig. A.8, A.9, A.10). Other parameters were kept the same.

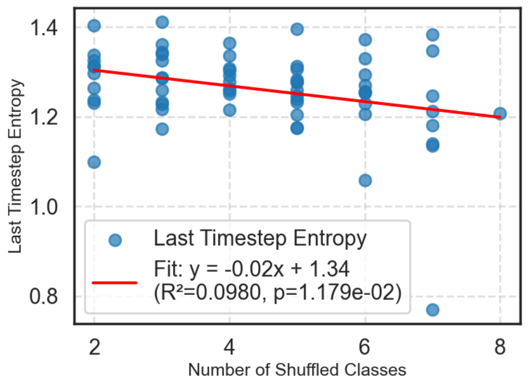
From these figures, it is evident that PCA and t-SNE present similar visualizations of the hidden dynamics across different network architectures, while MM-PHATE distinctly captures the unique learning behaviors of each model. Consistent with Section 5.1, PCA displays an increasing intra-step entropy even after model accuracy has plateaued, and t-SNE produces a noisy visualization. In contrast, MM-PHATE uniquely aligns its transitions well with the learning curve. Notably, the GRU model's representation appears more compact and organized compared to the LSTM model, potentially reflecting its superior performance and reduced overfitting.



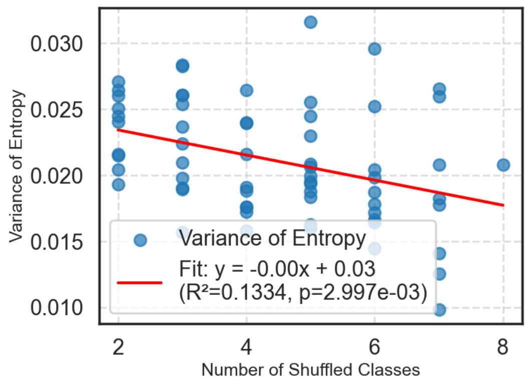
(a)



(b)



(c)



(d)

Figure A.5: Maximum, mean, last time-step intra-step entropy at the last epoch for models trained with different label shuffling levels.

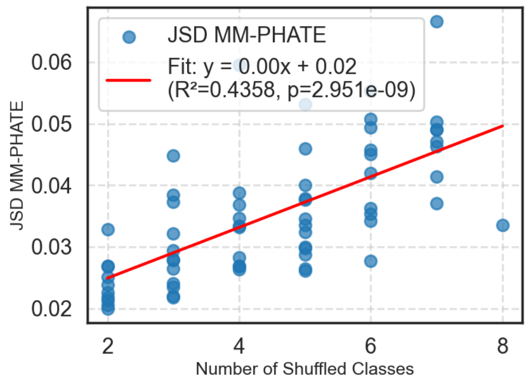
A.7 Area2Bump with Vanilla RNN

Here is the same analysis as section 5.1 using vanilla RNN (Fig. A.11, A.12, A.13). Other parameters were kept the same.

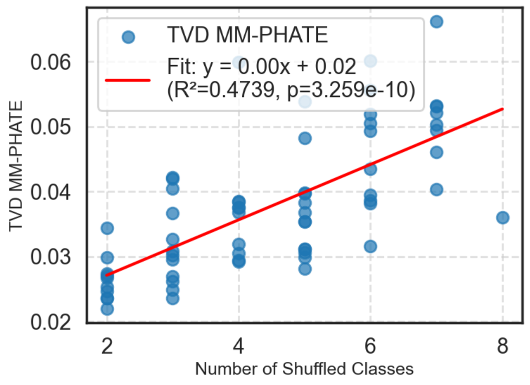
From these figures, it is evident that regardless of the network architectures, PCA exhibits a revolving pattern with overly smooth transitions across epochs, while t-SNE produces a noisy visualization. In contrast, the MM-PHATE visualization reveals that the Vanilla RNN displays a more chaotic pattern compared to the LSTM and GRU models, which is likely associated with its reduced performance and increased overfitting. Furthermore, the intra-step entropies of MM-PHATE show reduced variation across time-steps, indicating that the model struggles to process the input data effectively to generate meaningful representations.

A.8 Area2Bump with LSTM of Various Sizes

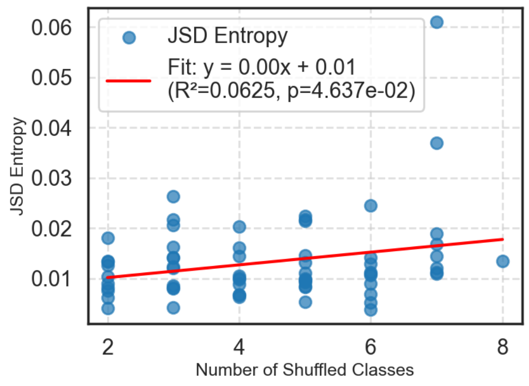
Here we repeat the same MM-PHATE analysis as in section 5.1 using LSTM of various sizes (Fig. A.15, A.16, A.17). Other parameters were kept the same. These results demonstrate that MM-PHATE consistently captures smooth yet distinct transitions across epochs and time steps, regardless of the LSTM network size. The intra- and inter-step entropy analyses further reveal that these transitions closely correlate with performance changes throughout training. Specifically, we observe a general increase in intra-step entropy as models begin to overfit, suggesting that the networks increasingly memorize input information. In contrast,



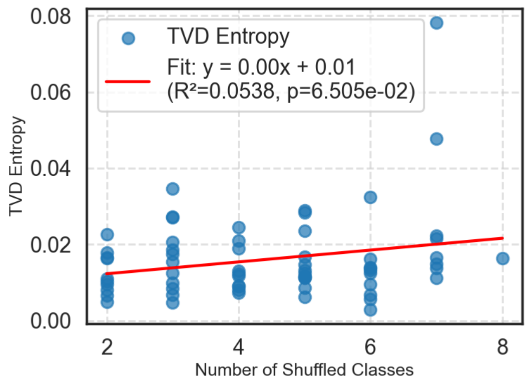
(a)



(b)



(c)



(d)

Figure A.6: JSD and TVD of the MM-PHATE embedding and intra-step entropy distribution in the last epoch for models trained with different label shuffling levels.

inter-step entropy shows a significant decline as overfitting progresses, reflecting a loss of sensitivity to input changes over time. The loss curve shows worse overfitting as the network size increases, and the networks' inter-step entropy becomes less structured.

B Computing Infrastructure

All but the t-SNE computation was carried out on a 14-core laptop running Windows 11 Home with a NVIDIA GeForce RTX 3070 Ti Laptop graphics card and 40GB of RAM. The t-SNE visualization was conducted on a single 95-core internal cluster running Ubuntu 18.04.6 LTS with 10 Quadro RTX 5000 graphics cards and 755GB of RAM.

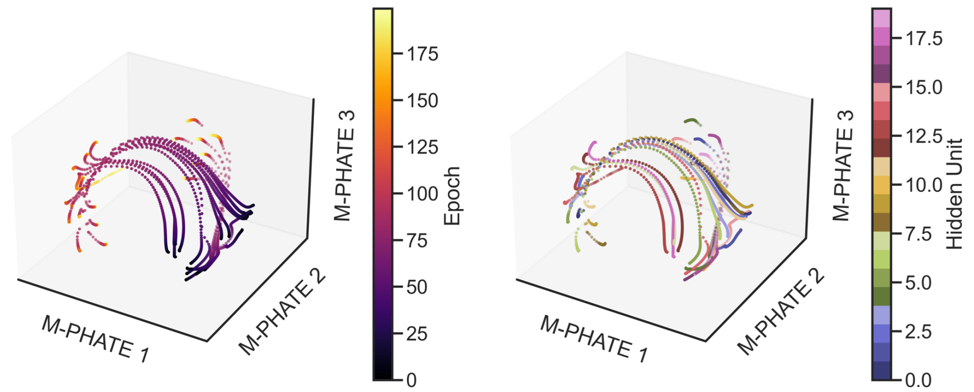


Figure A.7: M-PHATE on Area2Bump LSTM: Visualization of a 20-unit LSTM network trained for 200 epochs. Each point represents a hidden unit at the last time-step in a given epoch throughout the entire training process. The points are colored based on epoch (left) or hidden unit (right).

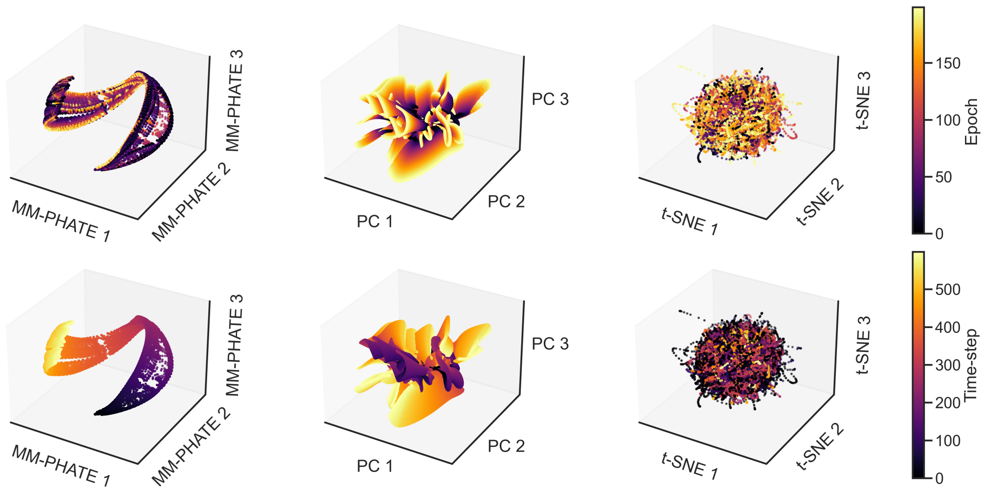


Figure A.8: Area2Bump GRU: Visualization of a 20-unit GRU network trained for 200 epochs. Each point represents a hidden unit at a specific time-step in a given epoch throughout the entire training process. The visualizations are generated using MM-PHATE, PCA, and t-SNE, from left to right, respectively. Points are colored based on epoch (top row) or time-step (bottom row)

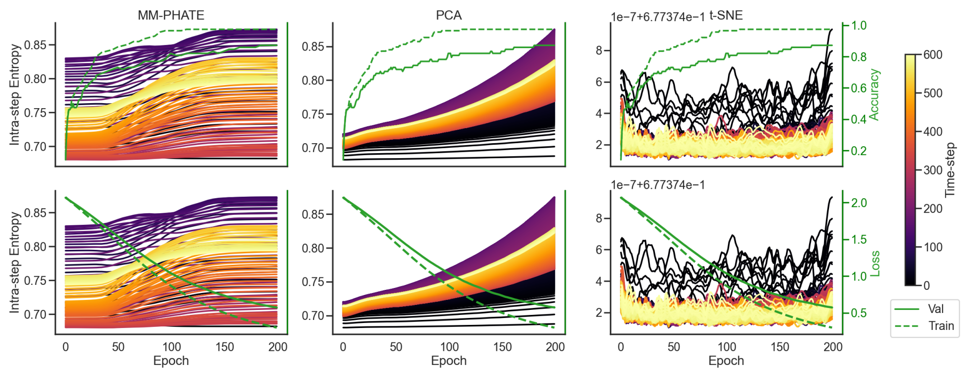


Figure A.9: Area2Bump GRU: Intra-step entropy of all hidden units in embedding space at each time-step in each epoch, compared to training and validation accuracy (top) and losses (bottom), comparing embeddings of MM-PHATE, PCA, and t-SNE.

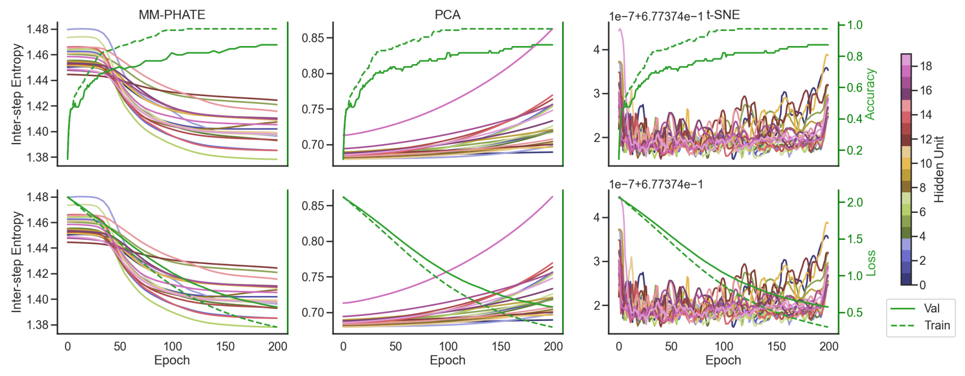


Figure A.10: Area2Bump GRU: Inter-step entropy of all hidden units in embedding space of the Area2Bump model at each time-step in each epoch, compared to training and accuracies (top) and losses (bottom). From left to right, the dimensionality reduction metrics used are MM-PHATE, PCA, and t-SNE.

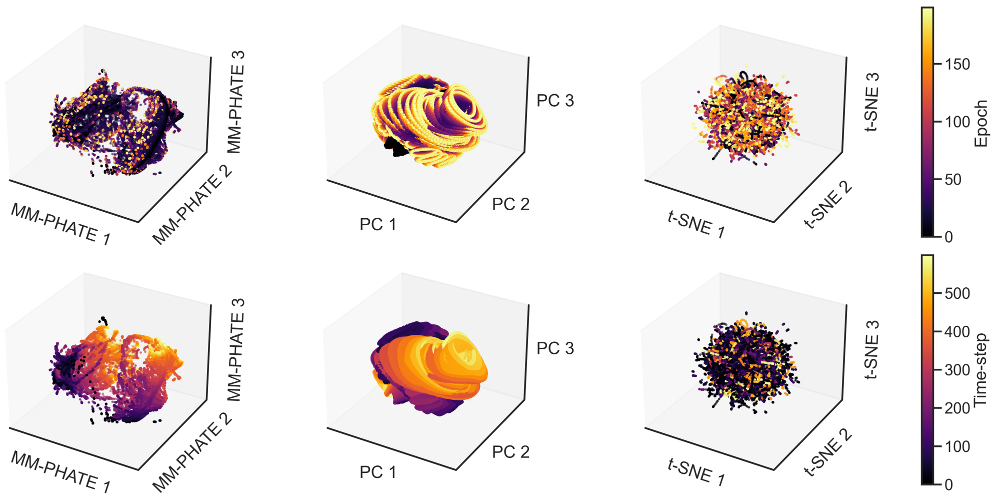


Figure A.11: Area2Bump Vanilla: Visualization of a 20-unit Vanilla RNN trained for 200 epochs. Each point represents a hidden unit at a specific time-step in a given epoch throughout the entire training process. The visualizations are generated using MM-PHATE, PCA, and t-SNE, from left to right, respectively. Points are colored based on epoch (top row) or time-step (bottom-row)

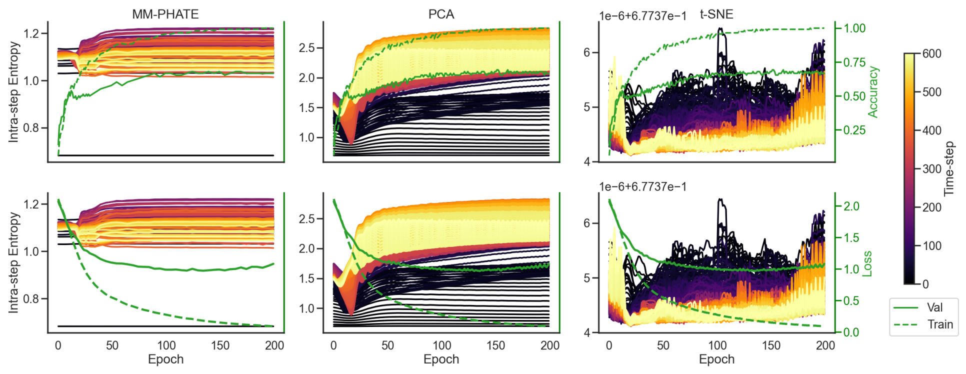


Figure A.12: Area2Bump Vanilla: Intra-step entropy of all hidden units in embedding space at each time-step in each epoch, compared to training and validation accuracy (top) and losses (bottom), comparing embeddings of MM-PHATE, PCA, and t-SNE.

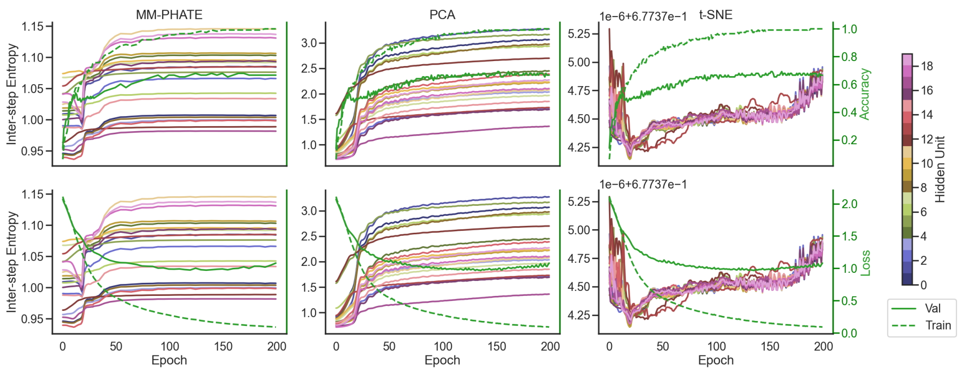


Figure A.13: Area2Bump Vanilla: Inter-step entropy of all hidden units in embedding space of the Area2Bump model at each time-step in each epoch, compared to training and accuracies (top) and losses (bottom). From left to right, the dimensionality reduction metrics used are MM-PHATE, PCA, and t-SNE.

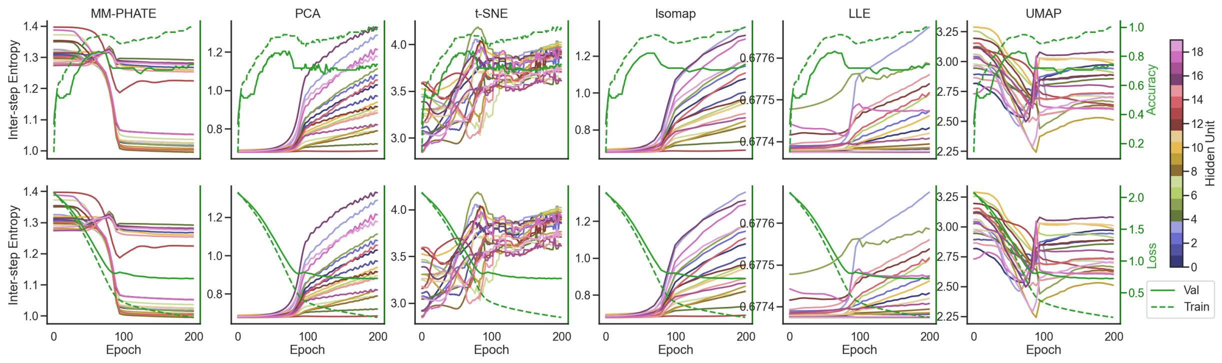


Figure A.14: Area2Bump: Inter-step entropy of all hidden units in embedding space of the Area2Bump model at each time-step in each epoch, compared to training and accuracies (top) and losses (bottom). From left to right, the dimensionality reduction metrics used are MM-PHATE, PCA, t-SNE, Isomap, LLE, and UMAP.

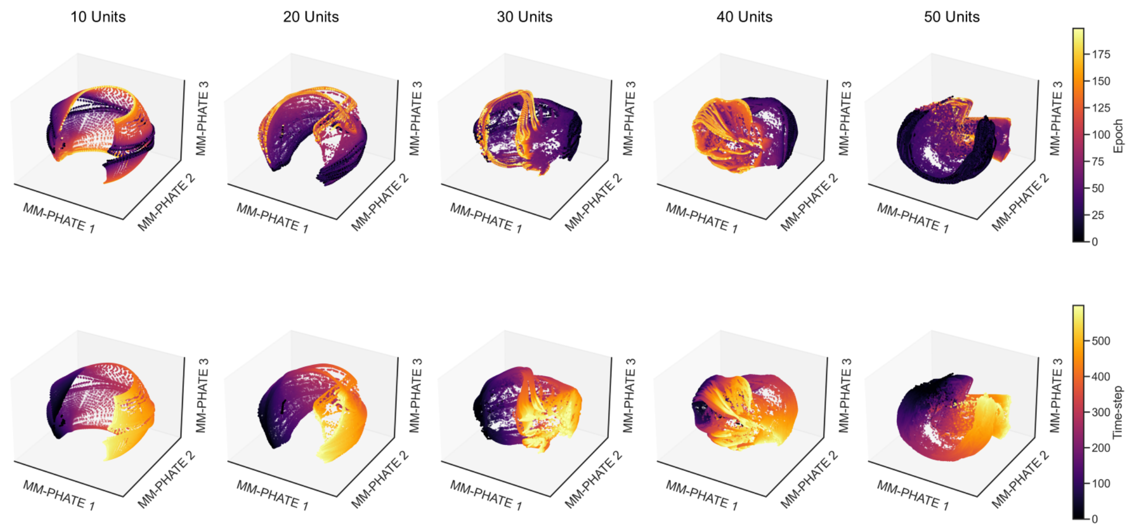


Figure A.15: Area2Bump LSTM: MM-PHATE visualization of networks of size 10 to 50 (left to right). Each point represents a hidden unit at a specific time-step in a given epoch. Points are colored based on epoch (top) or time-step (bottom).

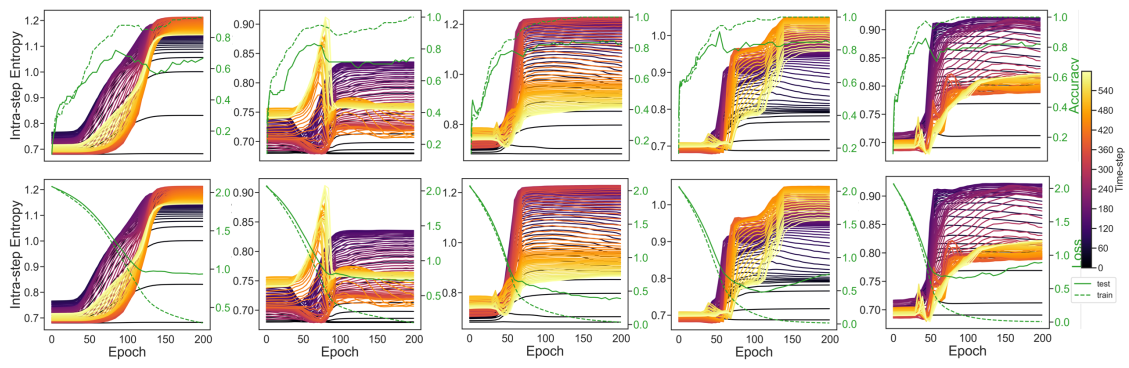


Figure A.16: Area2Bump LSTM: Intra-step entropy of all hidden units in MM-PHATE embedding space at each time-step in each epoch, compared to accuracies (top) and losses (bottom). Network sizes range from 10 to 50 (left to right).

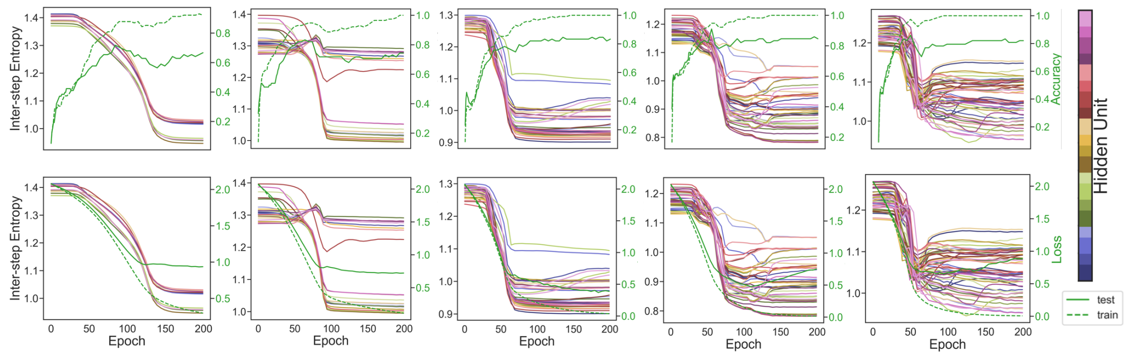


Figure A.17: Area2Bump LSTM: Inter-step entropy of all hidden units in MM-PHATE embedding space of the Area2Bump model at each time-step for each unit in each epoch, compared to accuracies (top) and losses (bottom). Network sizes range from 10 to 50 (left to right).

C Mathematical Notations

Notation	Definition
$\boldsymbol{x}$	Point in high dimensional space
$\boldsymbol{E}$	Euclidean distance matrix between all data points $\boldsymbol{x}$
$\boldsymbol{K}$	Affinity kernel matrix
$\epsilon_k(x_i)$	k-nearest-neighbor distance of x_i
α	Parameter controlling the decay rate
$\boldsymbol{P}$	Diffusion operator
$\boldsymbol{D}$	Diagonal matrix of row sums of $\boldsymbol{K}$
$\boldsymbol{P}^t$	Transition probabilities of a diffusion process over t steps
n	Total number of epochs the network is trained for
$\boldsymbol{F}$	Feed-forward neural network
m	Total number of hidden units in the network
$\boldsymbol{X}$	Training data, subset of $\boldsymbol{\Pi}$
$\boldsymbol{\Pi}$	Larger dataset
T	Activation tensor
$\boldsymbol{Y}$	Input data, subset of $\boldsymbol{X}$ with equal number of samples per class
p	Number of samples in $\boldsymbol{Y}$
$\boldsymbol{K}_{\text{intraslice}}^{(\tau)}(i,j)$	Intraslice affinities between pairs of hidden units within an epoch τ
$\boldsymbol{K}_{\text{interslice}}^{(i)}(\tau,\upsilon)$	Interslice affinities between a hidden unit i and itself at different epochs
$\sigma_{(\tau,i)}$	Intraslice bandwidth for unit i in epoch τ
ϵ	Fixed interslice bandwidth
τ	Index for given epoch
i,j	Index for given hidden unit
h_t	RNN hidden state at time step t
W	RNN weights
b	RNN biases
y_t	RNN output at time step t
f	RNN activation function
$\boldsymbol{R}$	Recurrent neural network
w	Index for given time step
s	Total number of time steps in the RNN
$\boldsymbol{K}_{\text{intrastep}}^{(\tau,\omega)}(i,j)$	Intrastep affinities between hidden units i and j at time-step ω in epoch τ
$\boldsymbol{K}_{\text{interstep}}^{(i)}((\tau,\omega),(\eta,\nu))$	Interstep affinities between a hidden unit i and itself at different time-steps and epochs
$\sigma_{(\tau,\omega,i)}$	Intrastep bandwidth for unit i at time-step w and epoch τ
k	Number of nearest neighbors
$\boldsymbol{L}$	Labels of $\boldsymbol{X}$

Table 1: Notations