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, enabling us to track how hidden representations evolve during training and how they support task performance. These training dynamics are reminiscent of ideas from information bottleneck theory, where intermediate representations are encouraged to retain task-relevant information about the target while discarding irrelevant variability in the input (Fischer, 2020; Tishby & Zaslavsky, 2015; Cheng et al., 2019). In contrast to classical information bottleneck approaches, we do not optimize or estimate a formal bottleneck objective; instead, we empirically characterize "expansion" and "compression" phases via the geometry and entropy of hidden-state trajectories revealed by MM-PHATE. Our results demonstrate that MM-PHATE preserves more dynamic details essential for understanding RNN performance compared to existing methods such as PCA (Wold et al., 1987), t-SNE (Maaten & Hinton, 2008), Isomap (Tenenbaum et al., 2000), Locally Linear Embedding (LLE) (Roweis & Saul, 2000), UMAP (McInnes et al., 2020), 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 distinct training-phase changes in RNN representations via time-resolved geometric and entropy-based summaries, and we quantitatively validate that these embedding signatures track task-relevant information using linear probes, time-step ablations, and label–state mutual information.

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 $i \in \{1, \ldots, m\}$ 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:

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

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

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

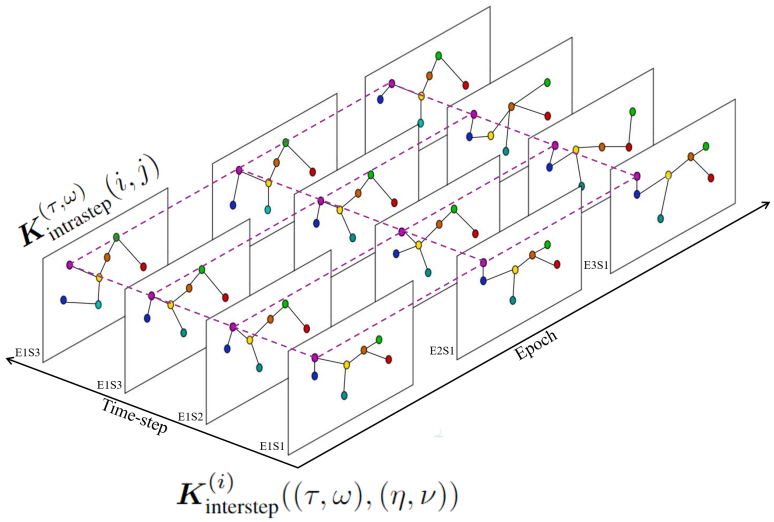

Figure 1: Example schematic of the multiway multislice graph used in MM-PHATE for RNNs. The intra-step kernels represent the similarities between the graph nodes at the same time-steps. The inter-step kernels represent the similarities between the nodes and themselves at different time-steps and epochs.

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 $\tau \in \{1,\ldots,n\}$ 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 $w \in \{1,\ldots,s\}$ 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 $\tau \in \{1 \ldots n\}$ and time-step $w \in \{1 \ldots s\}$ of each hidden unit $\boldsymbol{R}_i$

($i \in \{1 \dots m\}$) with respect to each sample $\boldsymbol{Y}_k \subset \boldsymbol{X}$. 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:

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

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 intra-step bandwidth for unit i at time-step w and epoch τ: $\sigma_{(\tau,\omega,i)}$, and the fixed inter-step bandwidth ϵ, we define:

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

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

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

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

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 inter-step affinities $\boldsymbol{K}^{(i)}_{\text{inter-step}}$, 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 intra-step affinities form a block diagonal matrix and the inter-step affinities form off-diagonal blocks composed of diagonal matrices (Fig. A.1).

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

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 evaluate MM-PHATE in one controlled setting (Hopf bifurcation) and two real RNN applications (Area2Bump neural spiking Chowdhury et al. (2022); HAR kinematics Reyes-Ortiz et al. (2012)) to assess: (i) fidelity to known dynamical structure, (ii) performance relative to standard embeddings, and (iii) whether embedding geometry tracks learning-relevant changes via neighborhood and entropy-based summaries (and probes/ablations/MI where available). We compare against PCA, t-SNE, Isomap, LLE and UMAP. We further compare to M-PHATE, however, unlike the other methods that all operate on the same tensor-structured data, M-PHATE does not use the multiple time scales of learning and network evolution meaning that it fundamentally embeds the data in a different space. We therefore include those comparisons in Appendix A.4.

5.1 Controlled Dynamical Benchmark: Hopf Bifurcation

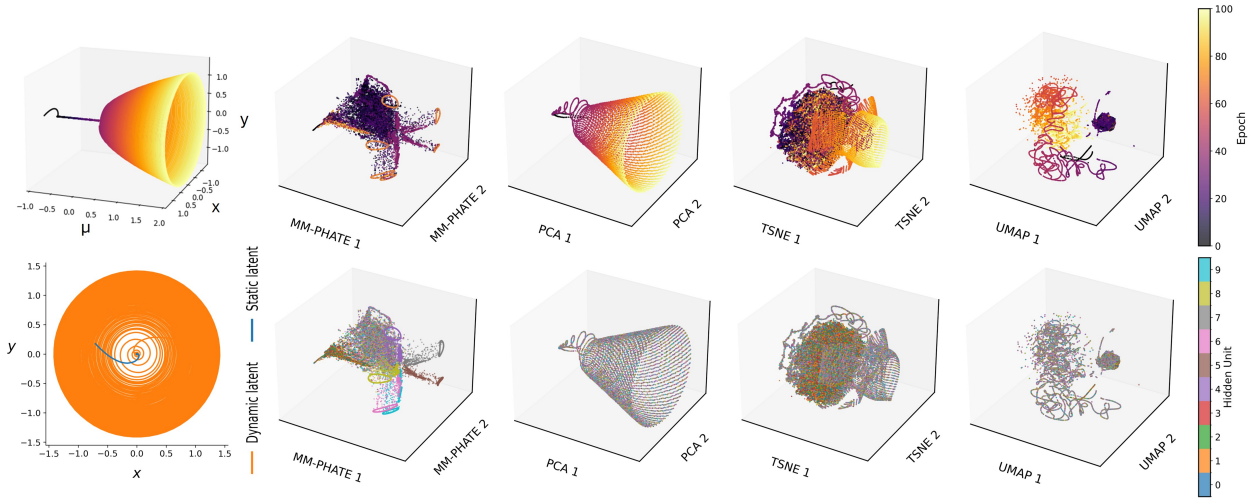

Figure 2: **Visualization of supercritical Hopf bifurcation dynamics and their embeddings.** The first column shows ground-truth latent trajectories: *top*—latent (x, y) flows transitioning from a contracting fixed point ($\mu < 0$) to a stable limit cycle ($\mu > 0$), crossing $\mu = 0$ around epoch 34; *bottom*—example static and dynamic hidden-unit traces illustrating the difference between a collapsing trajectory and an expanding spiral. Remaining columns show embeddings produced by MM-PHATE, PCA, t-SNE, and UMAP, applied to the full hidden-state tensor and colored by epoch (top row) or unit identity (bottom row). MM-PHATE recovers the underlying bifurcation geometry as a single coherent post-bifurcation orbit, whereas conventional methods primarily reflect changes in activation magnitude.

Interpreting RNN training dynamics is challenging because the latent geometry of hidden states is unknown; and visualizations often reflect activation magnitude rather than true dynamical structure. To determine whether a visualization method preserves actual dynamical structure, we construct a controlled benchmark based on a supercritical Hopf bifurcation—a canonical two-dimensional system with analytically understood geometry and phase transitions.

In this system, the control parameter μ governs a transition from a contracting spiral ($\mu < 0$) to loss of stability at the bifurcation point ($\mu = 0$), and finally to a stable limit cycle with radius $r^*(\mu) = \sqrt{\mu}$ for $\mu > 0$. These transitions provide a ground-truth analogue of high-level representation phases that later appear in RNN training: contraction, expansion near the bifurcation, and stabilization on a low-dimensional manifold.

To mimic the multiway structure of RNN activations, we generate two groups of noisy readouts across epochs and time-steps: (i) a *dynamic* group where μ linearly sweeps from negative to positive values, crossing the

bifurcation around epoch 34, and (ii) a *static* group where μ remains fixed in the pre-bifurcation regime. Each hidden "unit" is a noisy projection of the same latent dynamics (Appendix A.5), producing an activation tensor that mirrors RNN structure while preserving fully known latent geometry.

Figure 2 compares embeddings produced by MM-PHATE and several baselines. Each point in the visualization represents a hidden unit at a given time-step and epoch. MM-PHATE recovers the underlying dynamical topology: all post-bifurcation epochs align onto a single coherent periodic orbit, epochs remain globally ordered, and units follow parallel trajectories on the same manifold. Crucially, MM-PHATE treats amplitude growth as a nuisance dimension and organizes points according to the geometry and temporal progression of the dynamical system.

In contrast, PCA, t-SNE, Isomap, LLE, and UMAP produce a family of concentric limit cycles across epochs (Fig. A.6). This behavior arises because these variance-dominated methods encode the increasing oscillation amplitude (i.e., the monotonically increasing μ-dependent radius) as the dominant feature of the embedding. As a result, they emphasize inter-epoch amplitude differences rather than the shared periodic topology, obscuring the dynamical transition and mixing static and dynamic units. When a tanh nonlinearity is applied to the readout, variance-driven distortions become even more pronounced due to saturation, whereas MM-PHATE remains robust (Fig. A.12).

This controlled benchmark demonstrates that MM-PHATE is the only method that recovers the contraction-expansion-stabilization motifs intrinsic to the Hopf dynamics. It also provides the conceptual foundation for our entropy analysis: the Hopf bifurcation experiment exhibits characteristic low-entropy (contracting), high-entropy (expanding), and stabilized (limit-cycle) regimes, and MM-PHATE alone faithfully resolves these regimes (Fig. A.10-A.11). These ground-truth validations justify interpreting entropy changes in later sections as markers of representational change rather than as artifacts of amplitude scaling.

5.2 Neural activity

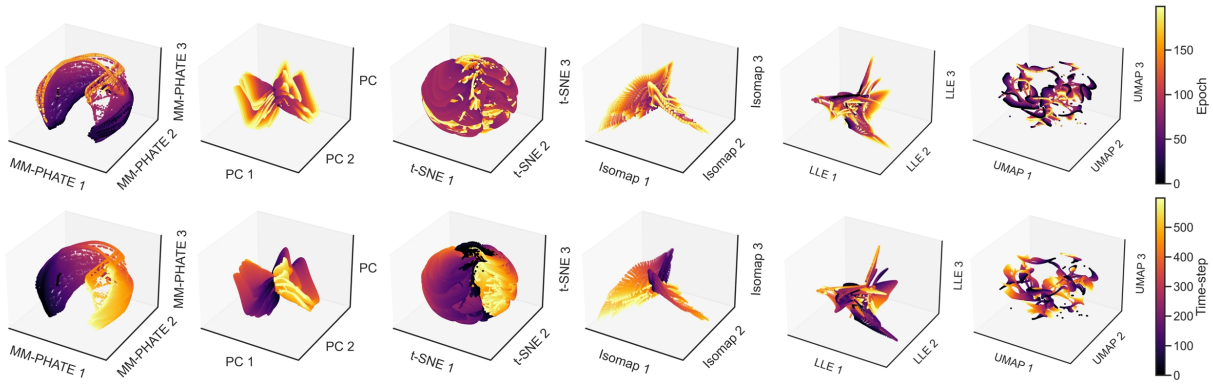

Figure 3: Area2Bump: Visualization of a 20-unit LSTM network trained for 200 epochs. Visualizations are generated using MM-PHATE, PCA, t-SNE, Isomap, Locally Linear Embedding (LLE), and UMAP (left to right). Points are colored by epoch (top row) or time-step (bottom row).

We next apply MM-PHATE to RNNs trained on the Area2Bump dataset, which contains spiking activity recordings from macaques during a reaching task (Chowdhury et al., 2022). We trained a single-layer, 20-unit LSTM to classify reach direction and analyzed how its hidden representations evolve over training. In an example session, the model reached a validation accuracy of 74%. Let $\boldsymbol{T}$ denote the activation tensor across epochs, time-steps, and hidden units (with an activation vector for each unit obtained across trials; see Appendix A.3 for implementation details). Each point in the visualization corresponds to a specific unit–time-step–epoch triple (Fig. 3); in practice, we subsample epochs and time-steps to reduce memory load.

We compare MM-PHATE to five standard embeddings—PCA, t-SNE, Isomap, LLE, and UMAP—and to M-PHATE (Appendix A.4), using the same set of points. For the baselines, we reshape $\boldsymbol{T}$ into a matrix

whose rows are unit–time-step–epoch points and whose columns are trial-indexed activations, and then apply each method to this flattened representation.

Qualitatively, MM-PHATE organizes the activity into a single manifold that simultaneously preserves progression across epochs (top row) and continuity across time-steps (bottom row). In particular, MM-PHATE reveals a clear reorganization that becomes pronounced at later time-steps and emerges over mid training (around epoch 40 to 90), roughly coinciding with the main rise and stabilization of validation performance (Fig. 3). By contrast, PCA and Isomap primarily capture a global drift dominated by high-variance directions and do not clearly separate training phases. t-SNE emphasizes local neighborhoods that are strongly driven by time-step variability and often yields discontinuous structure across epochs. LLE and UMAP preserve some within-unit trajectories but still do not clearly expose the mid-training transition visible in MM-PHATE. This motivates the quantitative evaluations below.

5.2.1 Neighborhood preservation

To quantitatively compare how well each embedding preserves the structure of the activation tensor, we compute neighborhood preservation between the original activation space and the low-dimensional embedding. For each point (unit–time-step–epoch triple), we identify its k-nearest neighbors in the original space and in the embedding space, and measure the fraction of neighbors that are shared (Appendix A.6). We report the average overlap separately for: (i) *intra-step* neighborhoods (within a fixed epoch and time-step, comparing units), and (ii) *inter-step* neighborhoods (within a fixed epoch and unit, comparing time-steps).

Table 1 summarizes results for the Area2Bump LSTM. MM-PHATE achieves the highest intra-step preservation at informative neighborhood sizes (e.g., $k = 5, 10, 15$), indicating that it best captures within-time-step unit communities. For inter-step neighborhoods, MM-PHATE yields the highest preservation for all but the smallest neighborhood size ($k = 5$), where LLE is marginally higher. We note that intra-step preservation necessarily saturates when $k \geq m$ (here $m = 20$ units), explaining the ceiling at $k = 20$ and $k = 40$; those rows serve mainly as a sanity check.

An additional variant omitting Isomap (due to memory constraints at larger sample sizes) is reported in Appendix Table 2; the relative ranking of methods remains unchanged. Overall, these results support the qualitative impression from Fig. 3: MM-PHATE preserves both *within-time-step* unit neighborhoods and *across-time* trajectories more faithfully than standard alternatives.

Table 1: Neighborhood preservation of visualization methods applied to an RNN trained on Area2Bump.

k		MM-PHATE	PCA	t-SNE	Isomap	LLE	UMAP
5	Intra-step	**0.621**	0.417	0.440	0.290	0.293	0.291
5	Inter-step	0.794	0.709	0.685	0.770	**0.812**	0.749
10	Intra-step	**0.757**	0.659	0.669	0.549	0.550	0.546
10	Inter-step	**0.820**	0.680	0.684	0.778	0.798	0.796
15	Intra-step	**0.852**	0.835	0.834	0.802	0.806	0.806
15	Inter-step	**0.834**	0.639	0.657	0.758	0.769	0.800
20	Intra-step	1.000	1.000	1.000	1.000	1.000	1.000
20	Inter-step	**0.853**	0.613	0.640	0.750	0.755	0.804
40	Intra-step	1.000	1.000	1.000	1.000	1.000	1.000
40	Inter-step	**0.826**	0.671	0.675	0.636	0.626	0.671

5.2.2 Intra-step entropy and time-resolved information analysis

To probe how representational geometry changes within an epoch, we measure how dispersed the hidden units are *at each time-step* in the embedding. For a fixed epoch e and time-step ω, we take the m embedded points $\{\mathbf{y}_{e,\omega,i}\}_{i=1}^{m}$ (one per unit) and estimate the differential entropy of this point cloud using a KDE-based estimator. We refer to this quantity as *intra-step entropy*. Importantly, this is an *embedding-space* measure of geometric dispersion (not a direct estimate of the Shannon entropy of the representation distribution in the information-theoretic sense); we use it as a proxy for how strongly units diversify vs. collapse at a given time-

step, and we validate its interpretability in a controlled setting. In the Hopf bifurcation experiment (§A.5), MM-PHATE intra-step entropy accurately recovers known contraction/expansion/stabilization regimes of the latent dynamics, whereas variance-dominated baselines do not (Fig. A.10).

On the Area2Bump LSTM, MM-PHATE reveals a structured temporal progression (Fig. 4). Entropy is relatively stable across time-steps until approximately epoch 30. Between epochs 30 and 50, entropy decreases for earlier time-steps while increasing for later ones, coinciding with the initial rise in training and validation accuracy. This pattern is consistent with the network compressing weakly informative early-sequence activity while expanding later representations that more strongly align with the task. From epochs 50 to 90, entropy rises across most time-steps, with later time-steps peaking near epoch 90. Around this point, validation accuracy plateaus and the train–test loss gap widens, indicating the onset of overfitting. Thereafter, entropy for early time-steps continues to increase while entropy for later time-steps declines, consistent with increasing representational variability in the early sequence that does not translate into generalization.

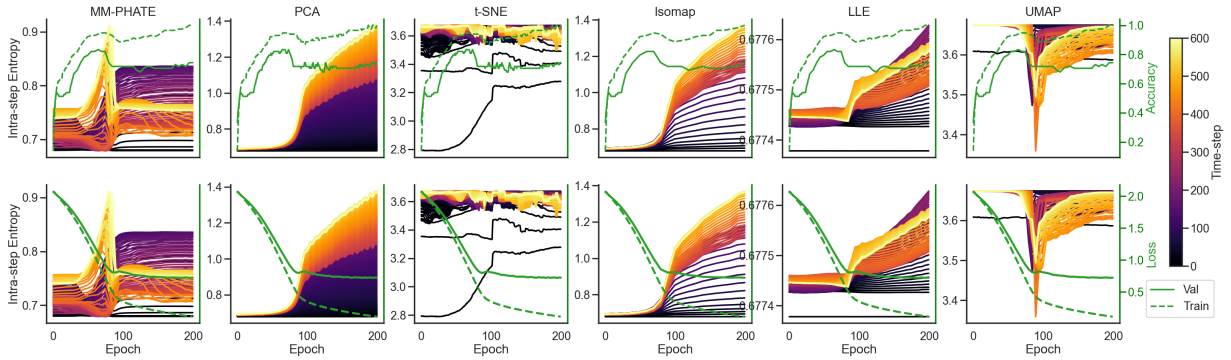

Figure 4: Intra-step entropy of hidden units in embedding space at each time-step and epoch, compared to training/validation accuracy (top) and losses (bottom), for MM-PHATE and baseline embeddings.

Linear probes, ablations, and mutual information. To test whether the intra-step entropy trends reflect *task-relevant* structure (rather than geometric drift), we evaluate label information in the recurrent state h_t using three complementary analyses (Appendix A.7, Appendix A.8, Appendix A.9): time-resolved linear probes, time-step ablations, and time-resolved mutual information (MI). All results are summarized in Fig. 5.

Linear probes quantify how *linearly decodable* the label is from h_t at each time-step during training. They show that early time-steps provide modest test accuracy early in training, peak around epochs 50–60, and then collapse toward chance with a widening train–test gap, whereas later time-steps improve steadily and reach a higher, more stable test plateau. The weak change in probe accuracy before epoch 30 suggests that the hidden states have not yet undergone a substantial reorganization into task-informative subspaces, which is consistent with the absence of clear transitions in intra-step entropy over the same interval. Time-step ablations test how the *trained classifier* relies on different parts of the sequence and reveal a temporal migration in reliance: masking late inputs (early-only) yields an early rise and a mid-training peak near the onset of overfitting, consistent with the model exploiting brittle early-sequence shortcuts that do not generalize, followed by a decline; in contrast, masking early inputs (late-only) improves gradually but remains well below the intact model. After roughly epoch 80, early-only and late-only accuracies converge to a similar moderate level, while intact performance remains substantially higher. This indicates that strong final accuracy depends on integrating evidence across the full sequence rather than any segment alone.

Time-resolved MI corroborates this picture by measuring total label information present in h_t, independent of the readout. MI increases over training for both early and late segments but is consistently higher for later time-steps and reaches a larger final value. Together, probes, ablations, and MI align with the MM-PHATE entropy patterns: later time-steps increasingly encode task-relevant information, while early time-steps eventually accumulate high-entropy variability that does not translate into improved generalization.

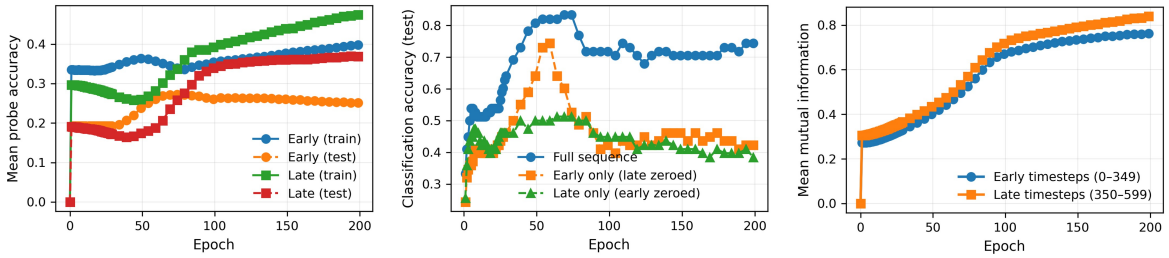

Figure 5: Combined analysis of time-resolved representations. **Left:** Linear probe accuracy for early and late time-steps on train and test sets. **Middle:** Time-step ablation accuracy for intact sequences, early-only sequences, and late-only sequences. **Right:** Time-resolved mutual information between labels and hidden states. All three analyses indicate that later time-steps carry the dominant task-relevant signal, while early time-steps increasingly encode variability that does not support generalization.

Comparison with baseline embeddings. Repeating the intra-step entropy analysis using PCA, t-SNE, Isomap, LLE, and UMAP (Fig. 4) yields substantially weaker alignment with probes, ablations, and MI. In particular, the baselines do not clearly reproduce the early compression / late expansion regime or the later divergence between early vs. late time-steps; their entropy changes are delayed and/or dominated by global geometric effects. This mirrors the controlled Hopf bifurcation experiment, where standard embeddings fail to recover the known entropy transitions. In contrast, MM-PHATE provides a time-resolved view in which entropy trends, separability in the embedding, probe performance, ablation performance, and MI jointly track how different segments of the input sequence contribute to learning and overfitting.

5.2.3 Inter-step entropy

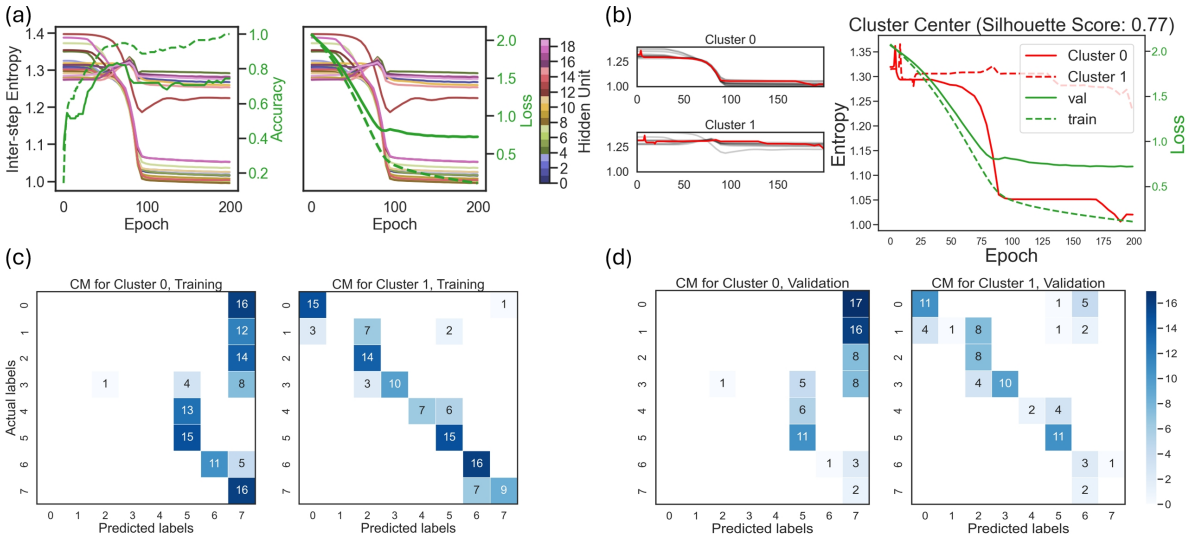

Figure 6: a) Inter-step entropy of each hidden unit across epochs, alongside model accuracy (left) and loss (right). b) Clusters of hidden units from inter-step entropy trajectories across epochs. Left: trajectories of units in cluster 0 (12 units) and cluster 1 (8 units). Right: cluster center trajectories across epochs. c–d) Confusion matrices of sub-networks constructed from each cluster on training and validation data.

The analyses above characterize how representations evolve *across units* at a fixed time-step. We now ask a complementary question: *how temporally selective is each unit across the sequence?* Whereas intra-step entropy summarizes diversity across units at a given (e, ω), inter-step entropy measures (for each unit) how strongly its representation differentiates among time-steps within an epoch, revealing temporal sensitivity.

Inter-step entropy reveals two robust groups of units (Fig. 6a). One subset maintains higher inter-step entropy throughout training, peaking around epoch 80, indicating sustained temporal differentiation. The other subset shows a pronounced decline in inter-step entropy later in training, indicating a collapse in temporal selectivity. To formalize this distinction, we apply time-series k-means with Dynamic Time Warping (DTW) distance (Bringmann et al., 2023) to the inter-step entropy trajectories (Fig. 6b), so that units are grouped by their *temporal pattern* of entropy changes even when the precise epochs of increase or decrease are slightly shifted. We emphasize that this DTW-based clustering is used descriptively to summarize the heterogeneity already visible in Fig. 6a, and our main conclusions do not depend on a specific choice of cluster number or assignments. Sub-networks constructed from these clusters exhibit large performance differences (Fig. 6c–d): models using only the high-temporal-selectivity cluster achieve substantially higher training and validation accuracy than models using only the low-temporal-selectivity cluster, confirming that the temporally differentiating units carry the most task-relevant signal.

This unit-level decomposition complements the intra-step entropy, probe, ablation, and MI results. Together they indicate that the example learning is accompanied by (i) increased task-relevant structure at later time-steps and (ii) an emerging specialization of hidden units, with a subset sustaining temporal selectivity while another subset becomes less informative later in training. Among baseline dimensionality reduction techniques, PCA, t-SNE, Isomap, LLE, and UMAP fail to recover this functional separation or the associated training-phase transitions (Fig. A.31). In contrast, MM-PHATE preserves temporal continuity while exposing distinct unit-level roles, yielding a coherent view of representational reorganization during learning.

Results for additional architectures (GRU, Vanilla RNN) and LSTM sizes (10–50 units) are provided in the appendix and show similar qualitative trends.

5.3 Analysis with Human Activity Recognition Model

To test whether the training-phase structure observed in Area2Bump generalizes beyond neural spiking data, we trained a 30-unit LSTM for kinematics-based Human Activity Recognition (HAR) (Reyes-Ortiz et al., 2012). The model was trained for 1000 epochs and reached a final validation accuracy of 84%. We applied MM-PHATE to the full hidden-state activation tensor and repeated the intra-step and inter-step analyses from Section 5.2.

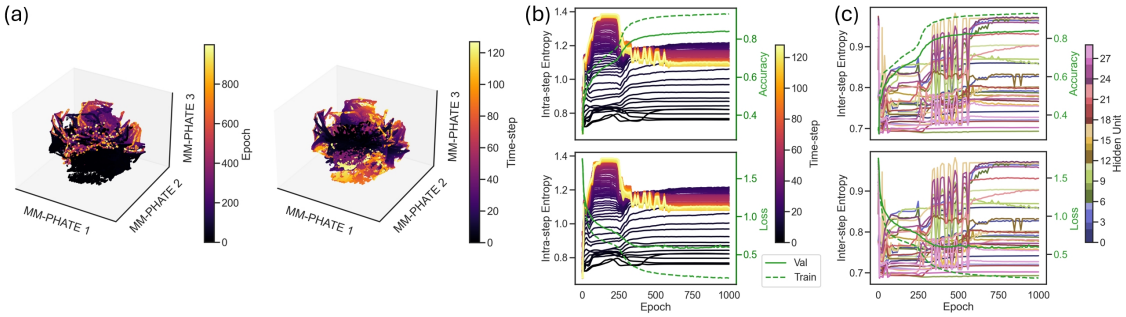

Figure 7: a) MM-PHATE visualization of a 30-unit LSTM network trained on HAR data, colored by epoch (left) and time-step (right). b) Intra-step entropy, c) inter-step entropy.

Geometry reveals a training-phase reorganization. MM-PHATE exposes a clear qualitative transition in how within-sequence dynamics are organized across epochs (Fig. 7a). In the embedding colored by epoch, early epochs occupy the lower portion of the manifold and later epochs progressively move upward, indicating a globally ordered training trajectory. In the embedding colored by time-step, however, the direction of the within-sequence flow changes across training: in early epochs (roughly the first ∼100 epochs), increasing time-step progresses predominantly *downward* along the manifold, whereas in later epochs the within-sequence progression shifts to an *upward* direction. This reversal indicates that the network is not merely drifting in activation magnitude, but is reorganizing the geometry of its temporal computation—

the mapping from time-step progression to movement on the representation manifold changes sign across training.

Intra-step entropy links this geometric transition to learning dynamics. We next quantify how representational diversity across units evolves within each epoch using intra-step entropy (Fig. 7b). Consistent with the qualitative reversal above, the entropy patterns suggest a redistribution of representational capacity across the sequence as training progresses. After an initial phase in which intra-step entropy gradually increases across time-steps, a transition becomes apparent (emerging after early training and becoming pronounced later): entropy at later time-steps (approximately after time-step $\sim$40) plateaus and then drops substantially, while entropy at earlier time-steps increases. This regime shift coincides with a marked improvement in training and validation accuracy and a decrease in both losses (Fig. 7b), suggesting that performance gains are accompanied by a structured reorganization of where (in time) the network expresses representational diversity. In light of the Hopf bifurcation experiment, where MM-PHATE entropy changes track known contraction/expansion/stabilization regimes, we interpret these HAR entropy transitions as geometric signatures of a real representational phase change rather than variance-driven artifacts.

Inter-step entropy shows increased temporal specialization at the same transition. Inter-step entropy (Fig. 7c) provides a complementary unit-level view by measuring how strongly each unit differentiates among time-steps within an epoch. Around the same training phase in which intra-step entropy reorganizes, many units exhibit a sharp increase in inter-step entropy, indicating that they become more temporally selective and more strongly encode time-resolved structure. Together with the intra-step results, this suggests a coordinated transition: the network increases temporal specialization at the unit level (higher inter-step entropy for many units) while the overall within-epoch geometry reallocates diversity across early vs. late portions of the sequence (Fig. 7a–b).

Relation to expansion–compression dynamics and comparison to Area2Bump. The combination of (i) a coherent epoch-ordered manifold, (ii) a reversal in the direction of time-step flow across epochs, and (iii) coupled intra-/inter-step entropy transitions is consistent with the expansion–compression picture of representation learning reported in trained RNNs (Farrell et al., 2022). Notably, this HAR behavior contrasts with Area2Bump, where early performance gains occur with weaker evidence of an analogous compression phase (Section 5.2). A plausible contributor is dataset complexity: HAR is substantially more low-dimensional than Area2Bump as measured by PCA (6 vs. 35 PCs to explain 95% variance; Fig. A.2), making it easier for a fixed-capacity network to consolidate task-relevant structure into a compact representation. In this sense, MM-PHATE highlights a key qualitative difference across datasets: HAR exhibits a pronounced representational reorganization during training (including a reversal of within-sequence flow direction), whereas Area2Bump shows comparatively weaker consolidation under the same style of analysis.

Overall, these results extend our findings from controlled dynamics (Hopf) and neural spiking data (Area2Bump) to a distinct kinematic domain, showing that MM-PHATE can reveal coherent, time-resolved training-phase transitions and unit-level specialization in RNN representations.

6 Conclusion

We introduced MM-PHATE, a visualization framework for RNNs that embeds hidden-state dynamics jointly across time-steps and training epochs via a multiway multislice graph. This construction preserves within-time-step unit communities and across-time trajectories in a single coherent manifold, addressing failure modes of standard embeddings that collapse or distort one or more of these axes.

Across experiments, MM-PHATE produced geometry that aligned with known or independently measured structure. In the Hopf bifurcation experiment, it uniquely recovered the correct dynamical topology and the associated contraction/expansion/stabilization regimes, providing a ground-truth basis for interpreting entropy trends as markers of representational reorganization. On Area2Bump, MM-PHATE achieved the strongest neighborhood preservation and revealed time-resolved training-phase changes that aligned with probes, ablations, and mutual information, while inter-step entropy identified functionally distinct unit

groups with different temporal selectivity. On HAR, MM-PHATE generalized these observations and exposed a pronounced training-phase transition (including a reversal of within-sequence flow direction) with coordinated intra-/inter-step entropy changes that coincided with performance improvements, suggesting task-dependent consolidation consistent with dataset complexity differences.

Limitations and future directions. MM-PHATE encodes continuity across time and epochs; highly non-smooth training dynamics (e.g., abrupt restarts or extreme learning rates) may reduce interpretability. The $nsm \times nsm$ kernel can also limit scale, but PHATE is robust to subsampling and scalable approximations (e.g., subsampling, coarsening, partitioning) are practical routes forward. Extending the multiway multislice construction to richer architectures (e.g., deeper recurrent stacks or attention-based models) is an important direction for future work (Katharopoulos et al.).

References

D. Anguita, Alessandro Ghio, L. Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *The European Symposium on Artificial Neural Networks*, 2013. URL https://api.semanticscholar.org/CorpusID:6975432.

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.

Karl Bringmann, Nick Fischer, Ivor van der Hoog, Evangelos Kipouridis, Tomasz Kociumaka, and Eva Rotenberg. Dynamic dynamic time warping, 2023. URL https://arxiv.org/abs/2310.18128.

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.

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].

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention.

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.

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.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. (arXiv:1802.03426), 2020. doi: 10.48550/arXiv.1802.03426. URL `http://arxiv.org/abs/1802.03426`. arXiv:1802.03426 [stat].

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.

Felix Pei, Joel Ye, David Zoltowski, Anqi Wu, {Raeed H.} Chowdhury, Hansem Sohn, {Joseph E.} O'Doherty, {Krishna V.} Shenoy, {Matthew T.} Kaufman, Mark Churchland, Mehrdad Jazayeri, {Lee E.} Miller, Jonathan Pillow, {Il Memming} Park, {Eva L.} Dyer, and Chethan Pandarinath. Neural latents benchmark '21: Evaluating latent variable models of neural population activity. *Advances in Neural Information Processing Systems*, 2021. ISSN 1049-5258. Publisher Copyright: © 2021 Neural information processing systems foundation. All rights reserved.; 35th Conference on Neural Information Processing Systems - Track on Datasets and Benchmarks, NeurIPS Datasets and Benchmarks 2021 ; Conference date: 06-12-2021 Through 14-12-2021.

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.

Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, December 2000. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.290.5500.2323.

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.

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.

Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1–3):37–52, August 1987. ISSN 01697439. doi: 10.1016/0169-7439(87)80084-9.

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 (Area2Bump as a neuroscience-motivated benchmark of population dynamics; HAR as a real-world sequential classification dataset).

1. Area2Bump is a part of the well-known "Neural Latents Benchmark" Challenge (Pei et al., 2021) in the neuroscience community. 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 HAR dataset was originally introduced in Anguita et al. (2013) which has been cited 3k times, and is also part of the UCI ML repository (where it has been viewed 196k times). 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 5.2 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 5.3 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

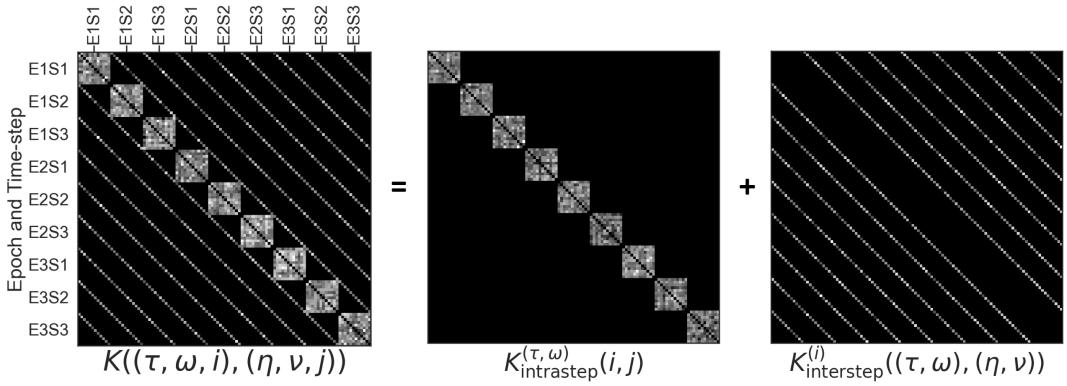

Figure A.1: Example schematic of the multiway multislice kernel used in MM-PHATE for RNNs. The intra-step kernels represent the similarities between the graph nodes at the same time-steps. The inter-step kernels represent the similarities between the nodes and themselves at different time-steps and epochs.

Common tensor and reshaping. For each trained network or dynamical system, we record hidden activations into a tensor with axes (epoch τ) $\times$ (unit i) $\times$ (time-step ω) $\times$ (trial/sample index). For baseline embeddings that operate on a matrix, we reshape the sampled tensor into

$$\boldsymbol{T}_{\text{mat}} \in \mathbb{R}^{N_{\text{pts}} \times p},$$

where each row corresponds to one unit–time-step–epoch triple and the p columns index trial-wise activations, as in the main text.

Hopf bifurcation dataset. For the Hopf experiment, the main activation tensor is

$$\texttt{trace\_data} \in \mathbb{R}^{(ET)\times H\times S},$$

where E is the number of epochs, T the number of intrinsic time-steps per epoch, H the number of hidden units, and S the number of samples (trials) per unit. Unless otherwise noted, Hopf embeddings use *all* epochs and time-steps; we do not subsample τ or ω for MM-PHATE, PCA, t-SNE, or UMAP. LLE and Isomap operate on a random subset of points, but still draw from the full (τ, ω, i) grid.

Area2Bump subsampling used for MM-PHATE and most baselines. Due to memory constraints, for Area2Bump we compute MM-PHATE on a subsampled tensor $\boldsymbol{T}_{\mathrm{MM}}$ defined by: (i) epochs τ sampled as the first 29 epochs followed by every 5th epoch thereafter (up to 200 total epochs), and (ii) time-steps ω sampled uniformly over the 600-step sequence using 100 evenly spaced indices. Unless stated otherwise below, t-SNE, LLE, UMAP are computed on this same sampled tensor $\boldsymbol{T}_{\mathrm{MM}}$.

HAR subsampling used for MM-PHATE. For HAR, we sample epochs as the first 29 epochs followed by every 10th epoch thereafter (up to 1000 epochs) and compute MM-PHATE on the resulting tensor. For baseline embeddings we reshape the corresponding sampled tensor into $\boldsymbol{T}_{\mathrm{mat}}$ in the same way.

A.3.1 MM-PHATE

For Area2Bump and HAR, MM-PHATE is applied to the sampled tensors described above (Area2Bump: $\boldsymbol{T}_{\mathrm{MM}}$; HAR: the sampled HAR tensor). We construct the multiway multislice kernel over units, time-steps, and epochs and then run PHATE via the M-PHATE package with `n_components = 3`.

For the Hopf bifurcation dataset, we apply MM-PHATE directly to `trace_data` of shape (ET, H, S) (no epoch or time-step subsampling), again with `n_components = 3`.

A.3.2 PCA

Due to PCA's low computational demand, we did not subsample any data before applied PCA using `sklearn.decomposition.PCA`.

Figure A.2 reports how many principal components are required to explain 95% of the variance for Area2Bump and HAR.

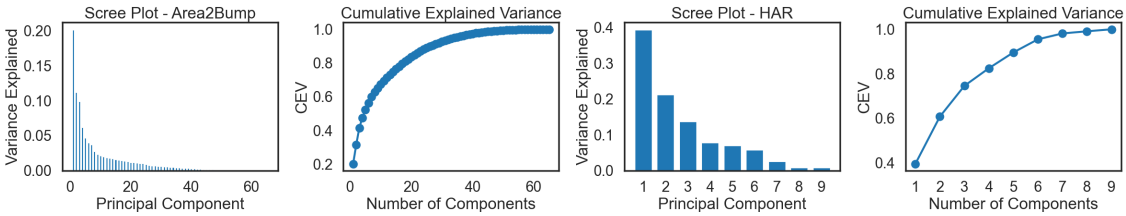

Figure A.2: PCA on the two datasets. Area2Bump requires 35 PCs (left) to explain 95% of the variance, whereas HAR requires 6 PCs (right).

For the Hopf dataset, we flatten `trace_data` to

$$X \in \mathbb{R}^{(ETH)\times S}$$

by stacking all (τ, ω, i) combinations and treating the S samples as features. We mean-center X across samples and fit PCA with `n_components = 3` to obtain a 3D embedding used in the Hopf figures (epoch-, time-, unit-, and group-colored views).

A.3.3 t-SNE

For Area2Bump, we first reduce $\boldsymbol{T}_{\mathrm{mat}}$ to 15 principal components (explaining 99.93% of the variance) and then run Barnes–Hut t-SNE with `n_components = 3`, `perplexity = 50`, and `n_iter = 2000` using `sklearn.manifold.TSNE`.

For Hopf, we run the same t-SNE configuration on the centered matrix $X \in \mathbb{R}^{(ETH)\times S}$ described above (no epoch or time-step subsampling). Because ETH is on the order of 8×10^4 in our configuration, this run is heavy but tractable on our cluster.

A.3.4 Isomap

Due to scalability constraints, Isomap is only applied on subsampled data.

For Area2Bump, we build a more aggressively subsampled tensor $\boldsymbol{T}_{\mathrm{ISO}}$ by sampling (i) epochs as the first 29 epochs followed by every 10th epoch thereafter, and (ii) 50 evenly spaced time-steps from the 600-step sequence. We then reduce $\boldsymbol{T}_{\mathrm{ISO}}$ to 15 PCs and apply `sklearn.manifold.Isomap` with `n_neighbors = 30` and `n_components = 3`.

For Hopf, we do not subsample epochs or time-steps, but we do randomly subsample points across the flattened matrix $X \in \mathbb{R}^{(ETH)\times S}$ to form

$$X_{\mathrm{sub}} \in \mathbb{R}^{N_{\mathrm{sub}}\times S},$$

where $N_{\mathrm{sub}} = \min(10{,}000, ETH)$ (with a fixed random seed). We then run `Isomap(n_neighbors = 30, n_components = 3)` on X_{sub}. The label vectors for epoch, time-step, unit, and group are restricted to the same subsample to produce unit-, epoch-, and group-colored Hopf panels.

A.3.5 Locally Linear Embedding (LLE)

For Area2Bump, LLE is computed on the same sampled tensor $\boldsymbol{T}_{\mathrm{MM}}$ used for MM-PHATE. We reshape to $\boldsymbol{T}_{\mathrm{mat}}$ and apply `LocallyLinearEmbedding(n_neighbors = 100, n_components = 3, method = "modified")`.

For Hopf, as with Isomap, we use the random subsample X_{sub} of size N_{sub} drawn from the full flattened matrix X. We then run `LocallyLinearEmbedding(n_neighbors = 20, n_components = 3, method = "modified")` to obtain a 3D embedding on the subsampled points, and use the corresponding epoch/time/unit/group labels to generate the Hopf LLE panels.

A.3.6 Uniform Manifold Approximation and Projection (UMAP)

For Area2Bump, UMAP is applied to $\boldsymbol{T}_{\mathrm{mat}}$ formed from $\boldsymbol{T}_{\mathrm{MM}}$. We use `umap.UMAP` with `n_neighbors = 50`, `min_dist = 0.1`, `n_components = 3`, Euclidean metric, and `random_state = 24`.

For Hopf, UMAP is run on the full centered matrix $X \in \mathbb{R}^{(ETH)\times S}$ with `n_neighbors = 30`, `min_dist = 0.1`, `n_components = 3`, and Euclidean metric. No epoch or time-step subsampling is used here; each (τ, ω, i) point is included.

A.4 M-PHATE as a Qualitative Baseline on a single time-step

To provide a qualitative point of reference, we also apply M-PHATE to the Hopf bifurcation experiment and the Area2Bump LSTM. Importantly, M-PHATE operates on a *single snapshot* of network activity: it embeds units based solely on their activations at one time-step per epoch (typically the final step). We select the final intrinsic time-step $\omega = T - 1$ for all epochs, yielding a tensor

$$\texttt{trace\_last} \in \mathbb{R}^{E\times H\times S}$$

and flatten it to

$$X_{\mathrm{last}} \in \mathbb{R}^{(EH)\times S}.$$

After mean-centering across samples, we run `phate.PHATE(n_components = 3, knn = 30)` to obtain a 3D embedding of epoch–unit pairs at the final time-step. This differs fundamentally from MM-PHATE, which embeds the full multislice tensor (units $\times$ time $\times$ epoch) using a diffusion process that jointly couples temporal and epoch-wise transitions.

Why M-PHATE is not an appropriate comparison. Because M-PHATE ignores all but a single snapshot of the time-steps, it discards the within-epoch temporal dynamics that are essential for understanding recurrent computation. RNNs process input sequences through evolving trajectories, and many of their representational transitions—including contraction, expansion, or bifurcation-like behavior—occur *within* an epoch's trajectory rather than only at its final state. Consequently:

- M-PHATE cannot represent how individual hidden units transition across time-steps.
- It cannot reveal dynamical geometry such as pre-bifurcation collapse, transition structure near $\mu \approx 0$, or post-bifurcation limit-cycle organization.
- It embeds a different mathematical object (units $\times$ epochs), whereas MM-PHATE embeds units $\times$ time $\times$ epochs; the two inhabit incompatible geometric spaces, making direct quantitative comparison inappropriate.

Thus M-PHATE serves only as a *qualitative* baseline showing how standard diffusion-based embeddings behave when restricted to a single time-slice, rather than an algorithm targeting dynamical systems.

Hopf control experiment. Figure A.3 shows M-PHATE applied to the last time-step of the Hopf dataset across epochs and units. While it captures a smooth trajectory as μ increases, it misses the characteristic change in *within-epoch* flow geometry that differentiates pre- and post-bifurcation dynamics (e.g., noise-dominated collapse vs. growing limit cycles). This demonstrates that restricting to the final state obscures the dynamical phenomena MM-PHATE is designed to uncover (Fig. A.6).

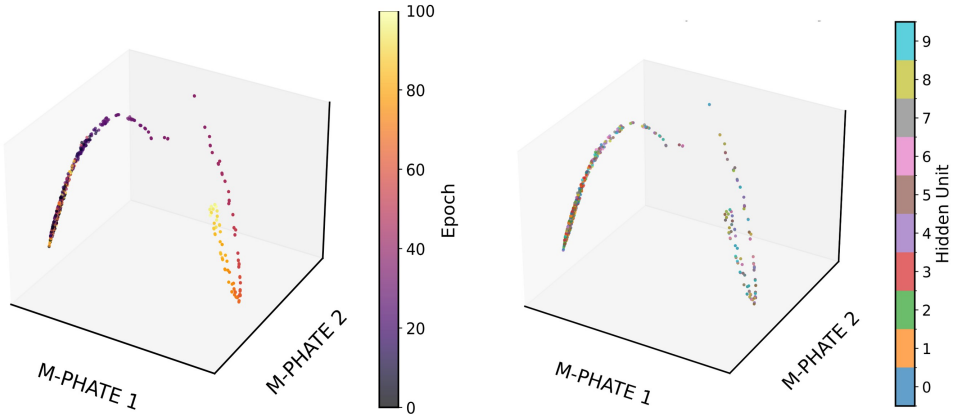

Figure A.3: M-PHATE applied to the Hopf bifurcation experiment (last time-step only). Each point corresponds to a hidden unit at the final time-step of an epoch. The embedding captures a smooth progression across epochs but fails to reveal the underlying bifurcation geometry, which manifests primarily in the *temporal* (within-epoch) dynamics that M-PHATE does not incorporate.

Area2Bump LSTM. Figure A.4 shows the same M-PHATE procedure applied to the 20-unit LSTM trained on the Area2Bump dataset. The structure is smooth across epochs, and hidden units display modest differentiation; however, the embedding lacks any representation of the sequential processing dynamics that occur within each trial. As a result, M-PHATE provides a useful snapshot-level visualization, but it cannot address questions related to temporal geometry, representational flow, or information propagation that MM-PHATE is explicitly designed to capture.

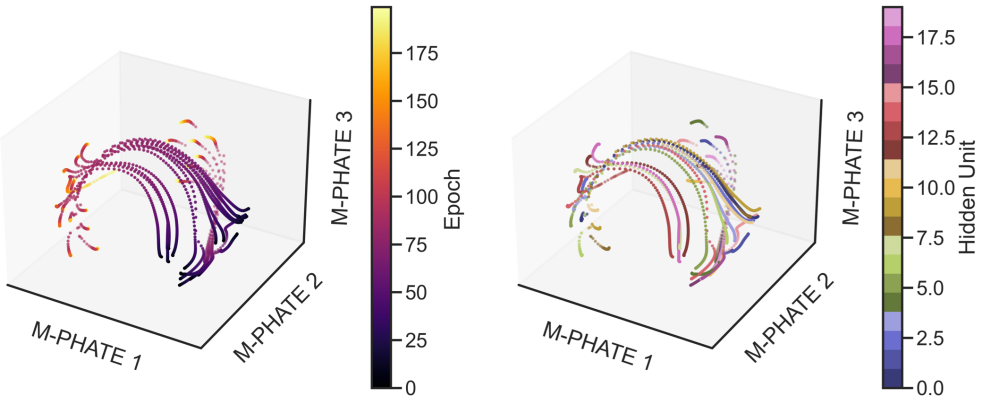

Figure A.4: M-PHATE applied to the Area2Bump LSTM (last time-step across epochs). Points represent hidden units at the final time-step of each epoch, colored by epoch (left) and by unit identity (right). The visualization captures smooth epoch progression but omits the temporal dynamics that are central to recurrent computation.

A.5 Hopf Bifurcation as a Controlled Dynamical Benchmark

A.5.1 Data Generation

To construct a controlled dynamical benchmark, we simulate a two–dimensional Hopf bifurcation system. The latent state of each sample at time t is $(x(t), y(t)) \in \mathbb{R}^2$, evolving according to

$$\begin{aligned} \dot{x} &= \mu x \; - \; y \; - \; (x^2 + y^2)x, \\ \dot{y} &= x \; + \; \mu y \; - \; (x^2 + y^2)y, \end{aligned} \tag{3}$$

where μ is the bifurcation control parameter. For $\mu < 0$, trajectories decay toward the fixed point at the origin; for $\mu > 0$, they converge to a stable limit cycle, implementing a canonical supercritical Hopf bifurcation.

Epochs. Each epoch τ corresponds to a fixed value of μ. Across the dataset, the dynamical group uses a linearly swept parameter

$$\mu_{\text{dynamic}}(\tau) = \mu_{\text{start}} + \frac{\tau}{n-1}(\mu_{\text{end}} - \mu_{\text{start}}), \qquad \tau = 0, \ldots, n-1,$$

where n is the total number of epochs. Thus the system crosses the bifurcation point ($\mu \approx 0$) once during the sweep. A second "static" group uses a fixed $\mu_{\text{static}} < 0$ for all epochs, providing a non-bifurcating baseline.

Time-steps. Within each epoch τ, the system is integrated forward for s time-steps (matching the notation for total RNN steps in Table 3) using a fourth-order Runge–Kutta method with step size Δt. Thus each epoch yields a short trajectory at constant μ.

Readout mapping (latent → hidden-unit activation). At each time-step w we map the latent state (x, y) into a scalar activation,

$$h_{\tau,w} = \alpha\,(\boldsymbol{w}^\top [x, y]) \; + \; \gamma\, r_{\tau,w}, \qquad r_{\tau,w} = \sqrt{x^2 + y^2}, \tag{4}$$

where $\boldsymbol{w} \in \mathbb{R}^2$ is a shared random projection direction (avoiding axis-aligned angles), α is a linear mixing weight, and γ scales the radius component. Optionally, $h_{\tau,w}$ is passed through $\tanh(\cdot)$ to mimic an RNN nonlinearity.

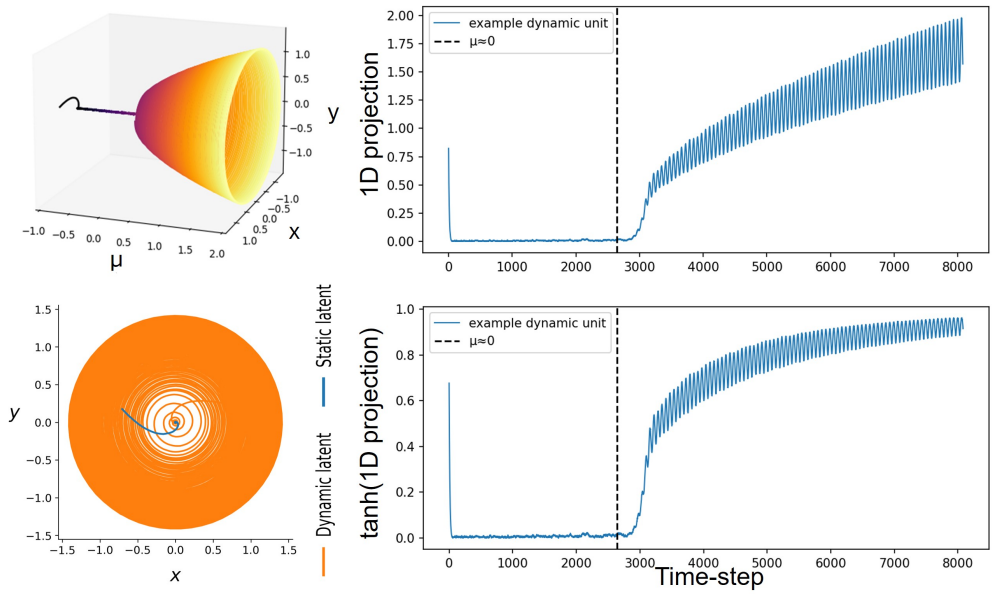

Figure A.5: Latent and readout dynamics of the Hopf bifurcation experiment. (Top left) Latent (x, y) trajectories of the Hopf system as μ moves from negative (stable focus) to positive (stable limit-cycle), illustrating the bifurcation. (Bottom left) Example dynamic and static hidden-unit trajectories visualized in the 2D latent plane. (Top right) The corresponding one-dimensional readout obtained from the projection $h = \alpha(\boldsymbol{w}^\top [x, y]) + \gamma r$ without applying a nonlinearity. (Bottom right) The same readout after applying a tanh nonlinearity, which compresses large-amplitude oscillations and highlights saturation in the post-bifurcation regime.

Hidden units and group structure. We construct m hidden units by replicating the readout $h_{\tau,w}$ across units with small i.i.d. noise (and optional gain/bias jitter), producing tight but non-identical clusters. Units are divided into two groups:

- **Dynamic group:** uses $\mu_{\text{dynamic}}(\tau)$ and undergoes the Hopf bifurcation.
- **Static group:** uses a fixed $\mu_{\text{static}} < 0$ and always decays to the fixed point.

Samples. We generate p independent samples (trials), each with its own initial condition and noise. All samples share the same μ schedule but produce distinct trajectories.

Tensor structure. The final activation tensor matches the notation of Table 3:

$$T \in \mathbb{R}^{(n\,s)\times m\times p},$$

where the first dimension indexes flattened epoch–time-step pairs (τ, w). A typical configuration used in the paper is

$$n = 101, \qquad s = 80, \qquad m = 10, \qquad p = 10,$$

with a static group of 4 units and a dynamic group of 6 units. The bifurcation occurs around epoch $\tau \approx 34$. Unless stated otherwise, μ_{dynamic} is swept from -1 to 2, while $\mu_{\text{static}} = -1$ (Fig. A.5).

A.5.2 Results

The Hopf bifurcation experiment provides a controlled setting for evaluating how embedding methods capture latent geometry, readout distortions, and bifurcation structure.

Epoch-colored embeddings (Figs. A.6, A.12). As the dynamic group crosses the Hopf bifurcation, the oscillatory radius increases substantially. PCA, t-SNE, Isomap, LLE, and UMAP primarily encode this amplitude growth, inflating post-bifurcation trajectories and obscuring the bifurcation geometry. MM-PHATE instead collapses post-bifurcation epochs onto a single coherent orbit, revealing the latent limit-cycle structure, rather than its raw magnitude. With a `tanh` readout, conventional methods additionally show saturation effects, while MM-PHATE remains qualitatively unchanged. (Static units are limited for Isomap/LLE due to scalability constraints.)

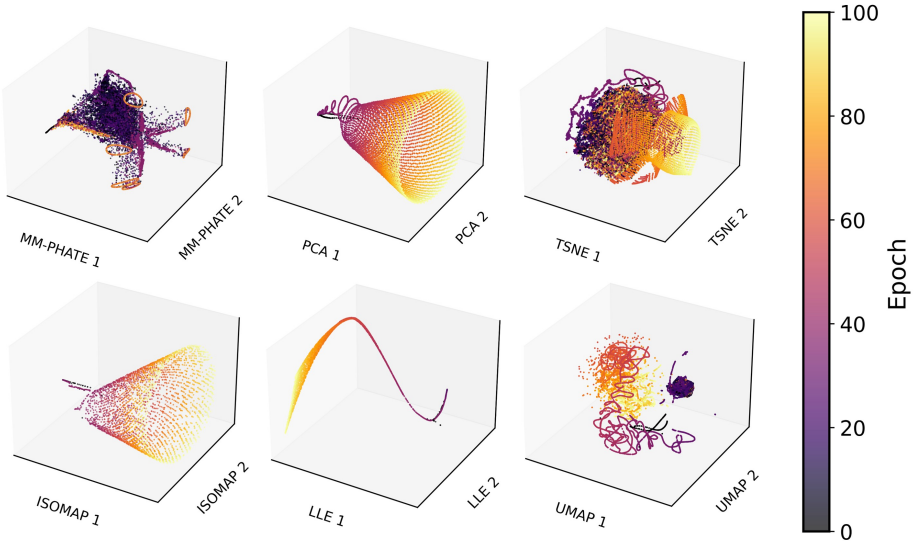

Figure A.6: Embeddings of the Hopf RNN traces colored by epoch using MM-PHATE, PCA, t-SNE, Isomap, LLE, and UMAP (left to right, top to bottom). *Note: For Isomap and LLE, we were unable to include sufficiently many static units due to poor scalability and subsampling constraints.*

Unit-colored embeddings (Figs. A.7, A.13). All units read out the same underlying Hopf dynamics but acquire subtle unit-specific variations after the bifurcation. MM-PHATE uniquely resolves these differences, separating individual units along the shared manifold. PCA, t-SNE, UMAP, Isomap, and LLE instead collapse unit trajectories together, both with and without the `tanh` nonlinearity. They show only minor deviations from injected noise and fail to capture the structured unit-level dynamical variation. (Isomap/LLE omit some static units as above.)

Single–dynamic-unit zoom (Fig. A.8). MM-PHATE organizes the evolution of a single dynamic unit across the entire sweep: contraction for $\mu < 0$, unfolding geometry near $\mu \approx 0$, and a clean limit cycle for $\mu > 0$, with the magnified view showing a smooth circular orbit.

Group-colored embeddings (Fig. A.9). Before bifurcation, dynamic units form a diffuse cloud reflecting heterogeneous transients en route to the emerging limit cycle, whereas static units (constant $\mu < 0$) repeatedly collapse to the same fixed point and form a compact cluster. MM-PHATE cleanly differentiates the two groups and reveals their distinct trajectory geometries across the Hopf bifurcation. Before bifurcation ($\mu < 0$), the dynamic units form a broad cloud, reflecting noise-dominated dynamics near the collapsing fixed point and the fact that these units are progressing toward the emergent limit cycle. In contrast, the static units (whose μ remains fixed < 0 across all epochs) accumulate into a denser, more compact structure, since their temporal evolution is repeatedly drawn toward the same contracting fixed point. Post-bifurcation, MM-PHATE continues to separate individual dynamic-unit trajectories, demonstrating that the method captures the full temporal progression of each unit rather than collapsing them into a single manifold. In contrast,

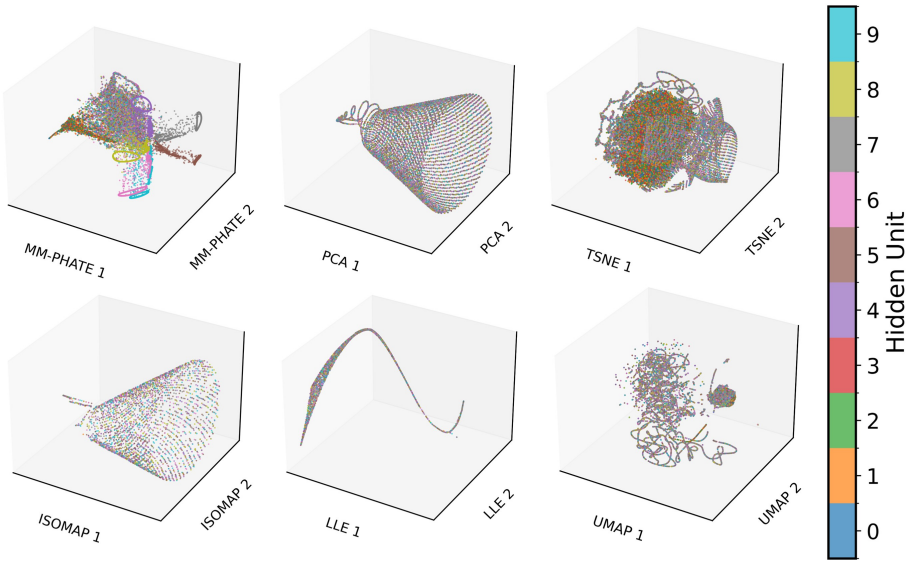

Figure A.7: Embeddings of the Hopf RNN traces colored by hidden-unit identity across six methods. *Note: For Isomap and LLE, we were unable to include sufficiently many static units due to poor scalability and subsampling constraints.*

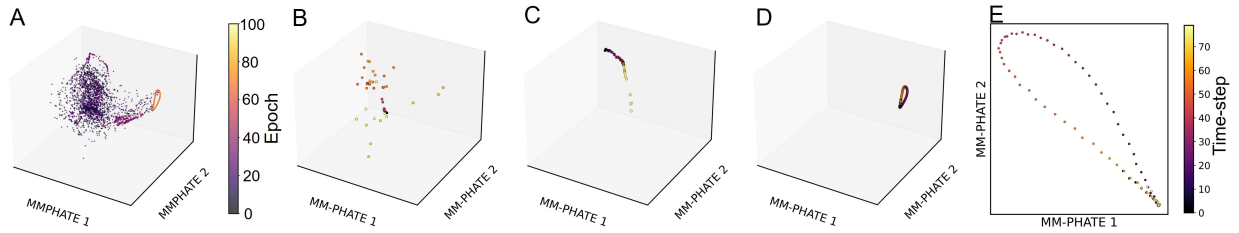

Figure A.8: Zoomed-in views of the MM-PHATE embedding for a single dynamic hidden unit across the Hopf bifurcation. (1) Full trajectory across all epochs, showing the global transition from the pre-bifurcation contraction to the post-bifurcation limit cycle (colored by epochs). (2) An early epoch with $\mu < 0$, where the trajectory collapses toward the fixed point with minimal rotational structure (colored by time-steps). (3) An epoch near the bifurcation point ($\mu \approx 0$), where the geometry begins to unfold and organization emerges (colored by time-steps). (4) A late epoch with $\mu > 0$, where the dynamics form a clear and stable limit cycle in the embedding (colored by time-steps). (5) A magnified view of a single post-bifurcation limit cycle, illustrating the smooth and consistent circular structure recovered by MM-PHATE (colored by time-steps).

the other methods are dominated by amplitude changes and small stochastic differences between units: they produce concentric, trumpet-shaped loops in which trajectories from different units largely overlap, with only subtle separations that primarily reflect noise rather than genuinely distinct dynamical orbits.

Intra-step entropy (Fig. A.10). Latent intra-step entropy (computed across samples) shows low entropy for static samples and a sharp rise for dynamic samples near the bifurcation. The readout inverts this pattern due to radius-dominated projection. MM-PHATE is the only embedding that qualitatively reproduces the latent structure; PCA, t-SNE, and UMAP instead mirror the readout, indicating their sensitivity to activation magnitudes rather than dynamical organization.

Inter-step entropy (Fig. A.11). Latent inter-step entropy exhibits consistently low entropy for static samples and a clear low-to-high transition for dynamic samples as μ crosses zero. The readout again shows

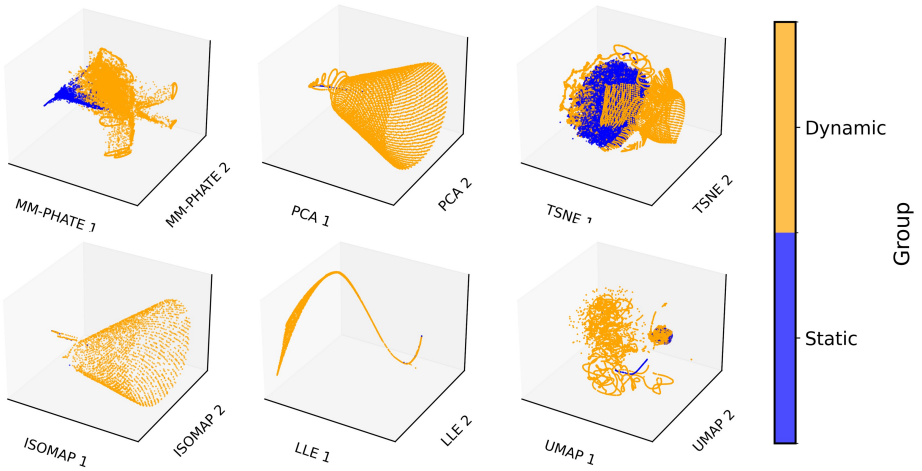

Figure A.9: Embeddings of the Hopf RNN colored by group identity (dynamic vs. static hidden units). *Note: For Isomap and LLE, we were unable to include sufficiently many static units due to poor scalability and subsampling constraints.*

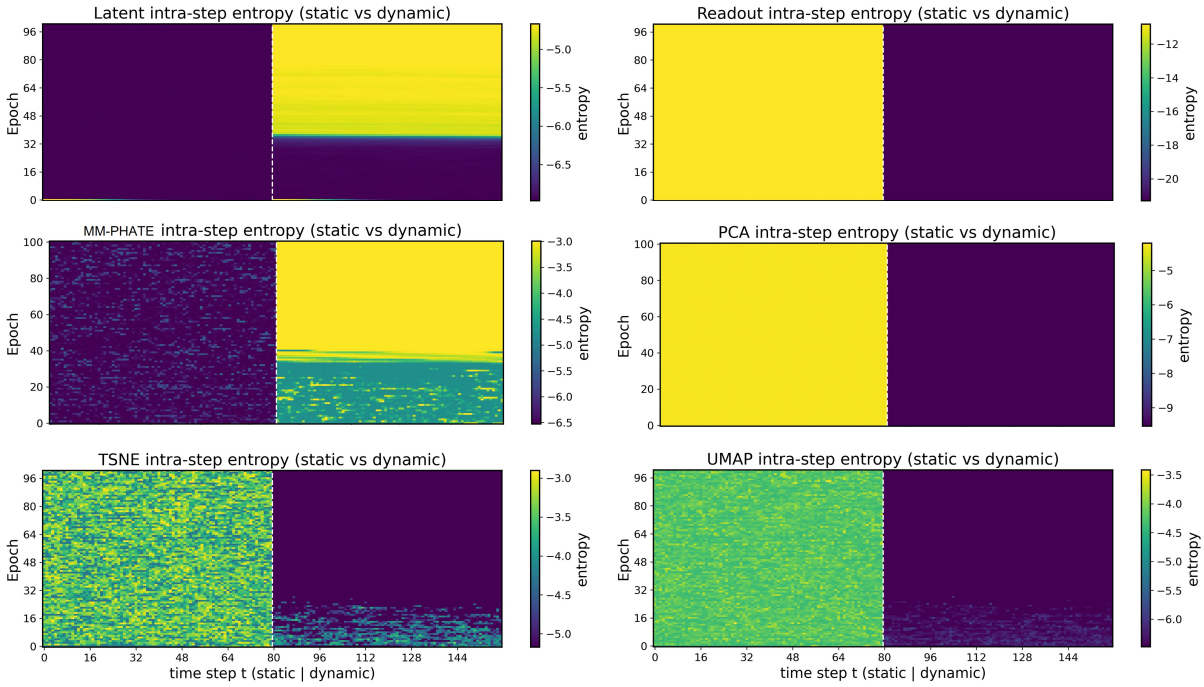

Figure A.10: Intra-step entropy analysis across the Hopf bifurcation. The first column shows ground-truth quantities: (top-left) latent intra-step entropy computed across samples (the latent space has no notion of hidden units), and (top-right) readout intra-step entropy computed across hidden units. The remaining columns show intra-step entropy computed from the MM-PHATE, PCA, t-SNE, and UMAP embeddings. MM-PHATE most closely reproduces the latent-space entropy dynamics: it preserves the consistently low entropy of the static group and captures the sharp transition in the dynamic group from low to high entropy as μ approaches the bifurcation point. By contrast, PCA, t-SNE, and UMAP fail to reflect the organization present in the latent dynamics, yielding high entropy for the static units and persistently low entropy for the dynamic units—mirroring the readout-level distortions rather than the underlying system dynamics.

the opposite pattern. MM-PHATE recovers the latent trend: static units remain low entropy, and dynamic units shift from high entropy in the noise-dominated regime ($\mu < 0$) to low entropy in the limit-cycle regime ($\mu > 0$). t-SNE captures part of this transition but spreads it broadly across epochs, while PCA and UMAP mostly follow the readout's magnitude-driven structure.

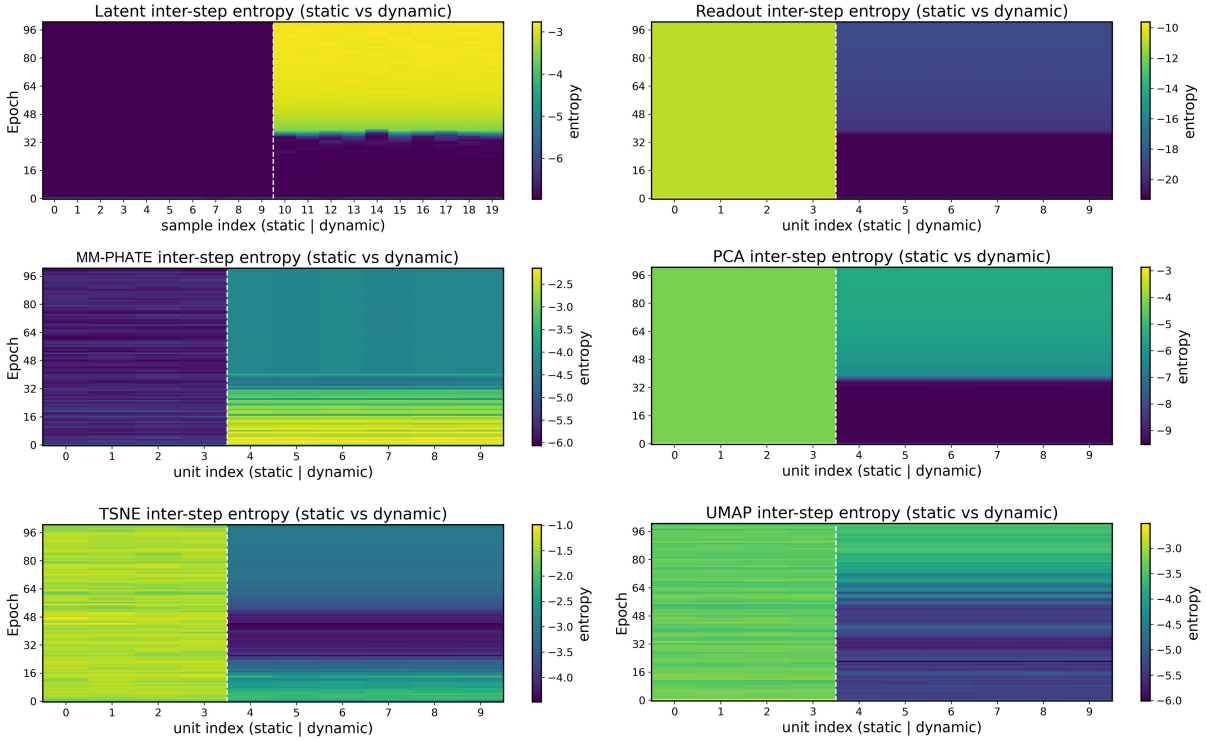

Figure A.11: Inter-step entropy across latent space (top-left), readout space (top-right), and four embedding methods (MM-PHATE, PCA, t-SNE, UMAP; columns 2–3). In the latent ground truth (top-left), the x-axis indexes samples, whereas in the readout and all embedding methods the x-axis indexes hidden units. Latent trajectories exhibit consistently low entropy for the static group, and a clear transition in the dynamic group from low entropy for $\mu < 0$ to high entropy for $\mu > 0$. The readout shows the opposite static–dynamic pattern, with high entropy for static units and a low-to-high transition for dynamic units, driven by the radius-dominated projection. MM-PHATE is the only embedding that qualitatively recovers the latent structure: static units remain low-entropy across epochs, and dynamic units transition from high entropy in the noise-dominated regime ($\mu < 0$) to low entropy in the limit-cycle regime ($\mu > 0$), indicating that MM-PHATE embeds units according to their full temporal trajectories rather than instantaneous magnitude. PCA and UMAP largely mirror the readout's magnitude-based structure, maintaining high entropy for static units and a low-to-high transition for dynamic units. t-SNE partially captures a high-to-low transition in the dynamic group but spreads this transition across many epochs. In contrast, MM-PHATE produces a sharp and well-localized transition around $\mu \approx 0$ and preserves the distinction between static and dynamic units, faithfully reflecting the underlying bifurcation geometry.

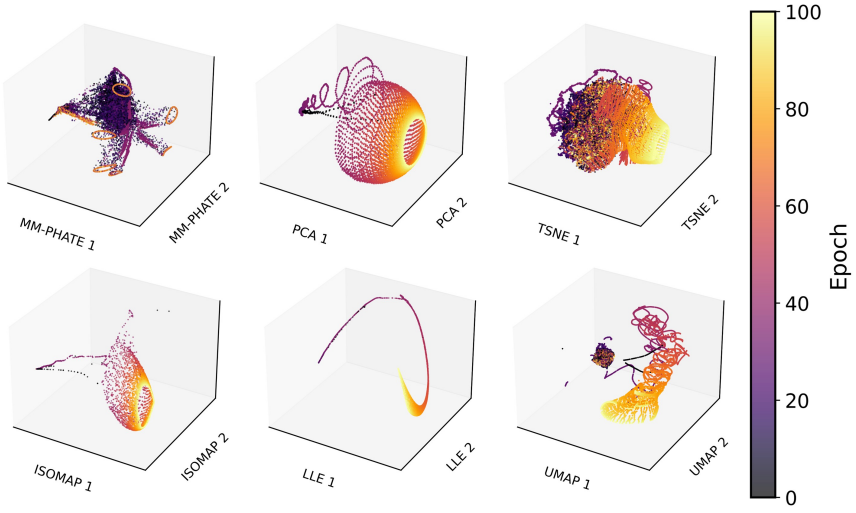

Figure A.12: Embeddings of the Hopf RNN traces with a `tanh` nonlinearity applied to the readout, colored by epoch using MM-PHATE, PCA, t-SNE, Isomap, LLE, and UMAP. The saturation induced by the `tanh` compresses the amplitudes of late-epoch trajectories, causing PCA, t-SNE, Isomap, LLE, and UMAP to exhibit visibly capped or flattened embeddings that primarily reflect this value saturation rather than the underlying dynamical progression. MM-PHATE, however, remains qualitatively unchanged: it continues to collapse post-bifurcation trajectories onto a single coherent orbit and preserves the bifurcation geometry despite the nonlinear distortion of the readout. Although the global appearance of all embeddings remains similar to the non-`tanh` case, these results highlight that conventional methods are sensitive to the absolute scale of activations, whereas MM-PHATE retains robustness to such transformations. *Note: Isomap and LLE cannot include sufficient static units due to poor scalability and subsampling constraints.*

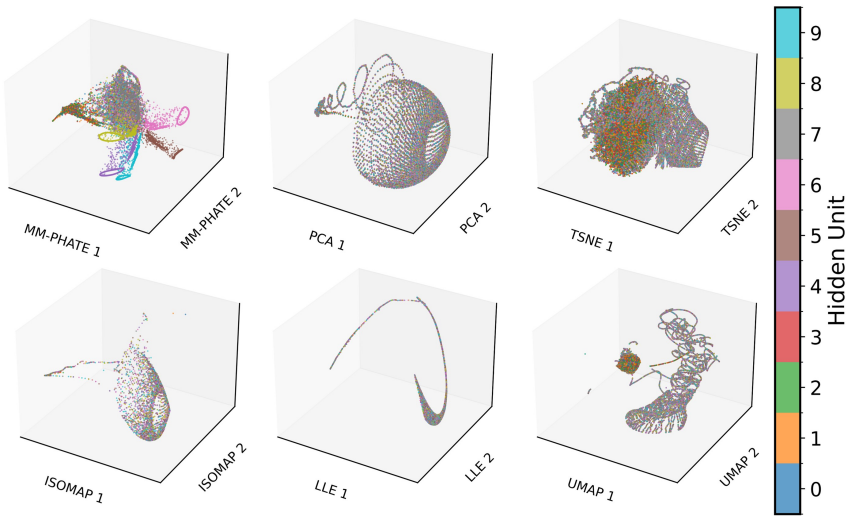

Figure A.13: Embeddings of the Hopf RNN traces with a `tanh` readout nonlinearity, colored by hidden-unit identity. Despite the saturation effects introduced by the `tanh`, MM-PHATE continues to cleanly separate the trajectories of individual dynamic units after the bifurcation, reflecting its ability to organize samples according to their underlying dynamical evolution rather than their raw activation magnitudes. In contrast, PCA, t-SNE, UMAP, Isomap, and LLE mix the unit trajectories together, with embeddings largely shaped by value saturation rather than dynamical structure. *As in the epoch-colored plots, Isomap and LLE omit some static units due to scalability and subsampling constraints.*

A.6 Neighborhood Preservation Analysis

We assess how faithfully each embedding preserves the local geometric relationships present in the activation tensor $T \in \mathbb{R}^{(ns)\times m\times p}$ by computing *intra-step* and *inter-step* neighborhood preservation. Each time–epoch pair (τ, ω) corresponds to an intra-step slice $T_{\tau,\omega} \in \mathbb{R}^{m\times p}$ containing the activations of all m hidden units across p samples.

Common evaluation grid. Because different methods operate on different subsampled sets of (τ, ω) pairs (e.g., UMAP and LLE require subsampling, Isomap requires even more aggressive subsampling), we evaluate all methods on the *intersection* of available slices. Isomap is excluded from the main comparison due to its extremely small intersection set under feasible subsampling.

Intra-step neighborhood preservation. For each slice (τ, ω) and hidden unit i, we compute its k nearest neighbors among $\{\boldsymbol{t}_{\tau,\omega,j}\}_{j=1}^{m}$ in the z-scored activation space and among $\{\boldsymbol{y}_{\tau,\omega,j}\}_{j=1}^{m}$ in the embedding. The intra-step preservation score is the average fraction of overlapping neighbors across all slices and units:

$$\mathrm{NP}_{\text{intra-step}} = \mathbb{E}_{\tau,\omega,i}\left[\frac{|\,\mathcal{N}_k^{\text{orig}}(\tau,\omega,i) \cap \mathcal{N}_k^{\text{emb}}(\tau,\omega,i)\,|}{k}\right].$$

Inter-step neighborhood preservation. For each hidden unit i, we consider its trajectory across time and epochs,

$$\{\boldsymbol{t}_i(\tau,\omega)\}_{(\tau,\omega)},$$

and compute the k nearest neighbors of each slice in both the original and embedded spaces (excluding self-neighbors). The inter-step score is the average neighbor overlap across units and slices:

$$\mathrm{NP}_{\text{inter-step}} = \mathbb{E}_{i,\tau,\omega}\left[\frac{|\,\mathcal{N}_k^{\text{orig}}(i;\tau,\omega) \cap \mathcal{N}_k^{\text{emb}}(i;\tau,\omega)\,|}{k}\right].$$

Reported results. For each method and $k \in \{5, 10, 15, 20, 40\}$, we report

$$(\mathrm{NP}_{\text{intra-step}},\ \mathrm{NP}_{\text{inter-step}})$$

computed on the exact same intersection of (τ, ω) pairs, ensuring that differences reflect the embedding quality rather than differences in sampling or data coverage.

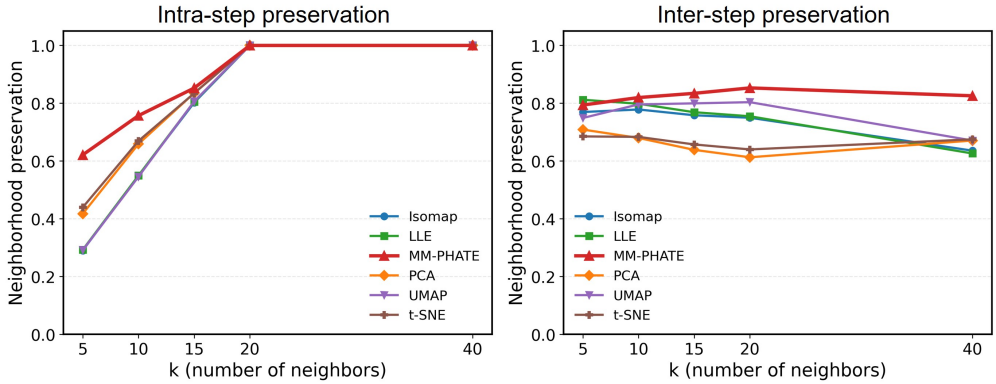

Figure A.14: Area2Bump: Neighborhood preservation of visualization methods including Isomap.

Table 2: Neighborhood preservation of visualization methods (Without Isomap).

k		MM-PHATE	PCA	t-SNE	UMAP	LLE
5	Intra-step	**0.649**	0.475	0.515	0.300	0.299
5	Inter-step	0.671	0.519	0.574	0.657	**0.703**
10	Intra-step	**0.789**	0.717	0.741	0.546	0.546
10	Inter-step	**0.672**	0.473	0.544	0.659	0.647
15	Intra-step	**0.859**	0.860	0.863	0.814	0.812
15	Inter-step	**0.667**	0.466	0.528	0.630	0.599
20	Intra-step	1.000	1.000	1.000	1.000	1.000
20	Inter-step	**0.660**	0.471	0.517	0.594	0.560
40	Intra-step	1.000	1.000	1.000	1.000	1.000
40	Inter-step	**0.634**	0.511	0.534	0.476	0.426

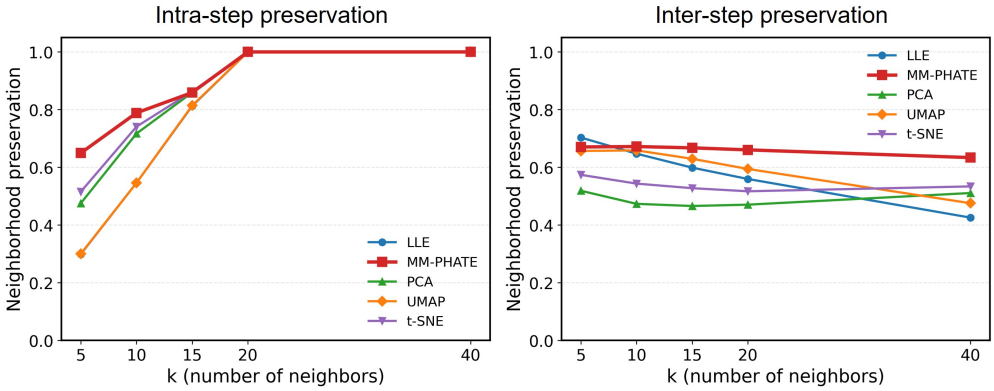

Figure A.15: Area2Bump: Neighborhood preservation without Isomap

A.7 Time-resolved linear probing analysis

To quantify how much task-relevant information is *linearly decodable* from recurrent representations at different positions in the sequence, we performed a time-resolved linear probe analysis on the hidden states h_t of the recurrent network $\boldsymbol{R}$. Let $\boldsymbol{X}$ denote the supervised dataset with labels $\boldsymbol{L}$, and let s be the total number of time-steps in the sequence. At a collection of training epochs $\tau \in \{1, \ldots, n\}$ (subsampled for efficiency), we evaluated the trained network on the fixed training and test splits and extracted the sequence of hidden states $\{h_t\}_{t=1}^{s}$ for each input example.

Per-time-step linear probes. For each probed epoch τ and each time-step $\omega \in \{1, \ldots, s\}$, we formed a feature matrix by collecting the hidden state at that time-step across examples, yielding $H^{(\tau,\omega)} \in \mathbb{R}^{N \times m}$, where m is the number of hidden units and N is the number of evaluated examples. We then fit a multinomial logistic regression (linear classifier) to predict the class labels $\boldsymbol{L}$ from $H^{(\tau,\omega)}$ using the training split, and evaluated the resulting probe on both the training and test splits. This yields time-resolved probe accuracies $\mathrm{Acc}^{\mathrm{train}}(\tau, \omega)$ and $\mathrm{Acc}^{\mathrm{test}}(\tau, \omega)$, which quantify the extent to which label information is linearly accessible from the representation at a specific time-step and epoch.

Heatmap summaries over epoch and time. We visualize $\mathrm{Acc}^{\mathrm{test}}(\tau, \omega)$ and $\mathrm{Acc}^{\mathrm{train}}(\tau, \omega)$ as heatmaps (Fig. A.16) with axes (epoch τ) $\times$ (time-step ω). These summaries reveal where in the sequence and when during training label information becomes linearly separable in the hidden state. In our experiments, later time-steps exhibit a steady increase in test decodability over training and reach a higher, more stable plateau, whereas early time-steps show more transient test decodability that peaks mid-training and later degrades.

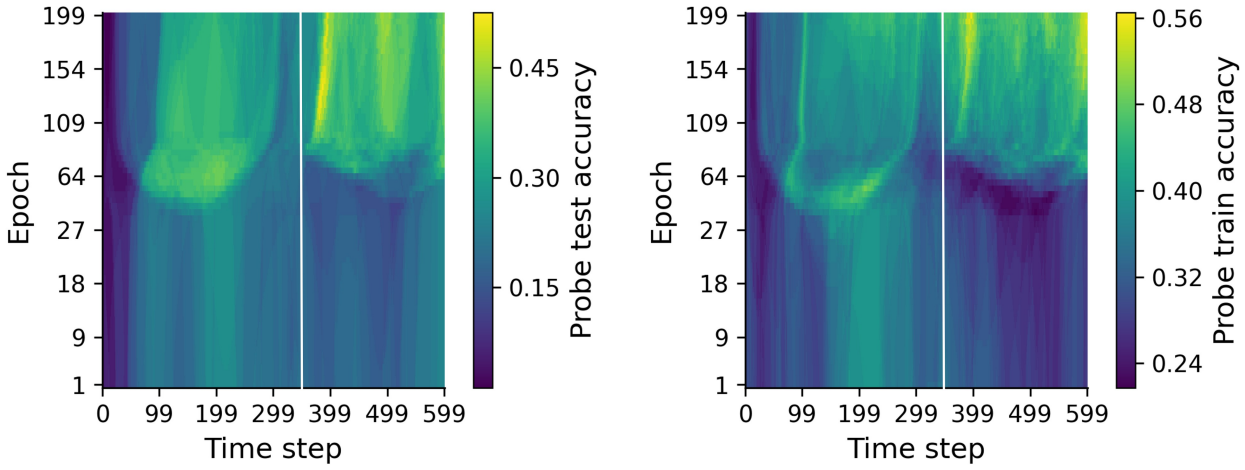

Figure A.16: Heatmaps showing linear probe accuracies across training epochs and time-steps. Left: Probe test accuracy, illustrating the model's performance on the test set for each time-step at different epochs. Right: Probe train accuracy, showing the model's performance on the training set across epochs and time-steps. Both heatmaps visualize the evolution of probe accuracy over time, with the color intensity representing accuracy values.

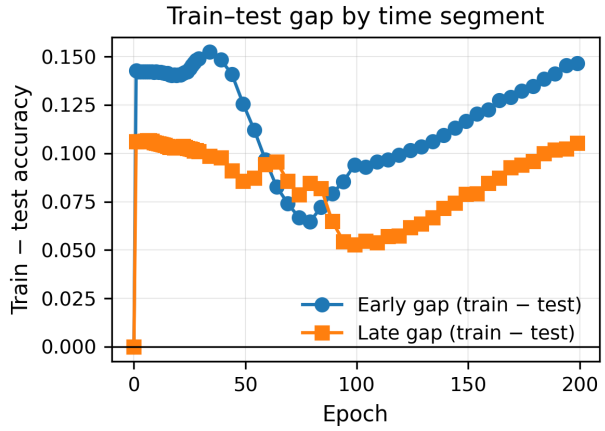

Figure A.17: Train-test accuracy gap over training epochs, separated into early and late time segments. The plot compares the difference between training and testing accuracies for early and late time steps, illustrating how the gap evolves throughout the training process. Early time steps are represented by circles, while late time steps are shown with squares. A larger gap typically indicates overfitting, while a smaller gap suggests more generalization.

Early/late aggregation and train–test gap. To align the probe results with the early-vs-late comparisons used elsewhere in the paper, we further aggregate probe accuracies over an "early" and "late" segment of the sequence (defined by a fixed boundary time-step = 350 based on divergence of intra-step entropy trends). For each epoch τ, we compute mean probe accuracies within each segment for both training and test splits, and report the corresponding generalization gaps (Fig. A.17):

$$\mathrm{Gap}_{\mathrm{early}}(\tau) = \overline{\mathrm{Acc}}^{\mathrm{train}}_{\mathrm{early}}(\tau) - \overline{\mathrm{Acc}}^{\mathrm{test}}_{\mathrm{early}}(\tau), \qquad \mathrm{Gap}_{\mathrm{late}}(\tau) = \overline{\mathrm{Acc}}^{\mathrm{train}}_{\mathrm{late}}(\tau) - \overline{\mathrm{Acc}}^{\mathrm{test}}_{\mathrm{late}}(\tau).$$

We visualize these quantities in Fig. A.17 to highlight when and where overfitting emerges in the sequence. Consistent with the main-text findings, the early-segment gap increases substantially after mid-training,

coinciding with the deterioration of early time-step test probe accuracy, whereas the late segment maintains higher test decodability with a smaller train–test gap.

Overall, the time-resolved probe analysis provides a complementary validation that the representational patterns observed in the MM-PHATE entropy analyses track task-relevant information: label signal becomes increasingly decodable at later time-steps, while early time-steps eventually exhibit higher variability that does not translate into improved generalization.

A.8 Time-step ablation analysis

To assess how the *trained* recurrent classifier $\boldsymbol{R}$ uses information across the sequence, we performed a time-step ablation analysis in which portions of the input sequence were masked at evaluation time while keeping the network weights W, b fixed. Let $\boldsymbol{X}$ denote the evaluation dataset with labels $\boldsymbol{L}$, and let s be the total number of time steps. For a set of training epochs τ (subsampled consistently with the entropy and probing analyses), we evaluated classification accuracy under three input conditions:

1. **Full sequence (intact):** the original input sequence is provided to the network.
2. **Early-only:** time steps $\omega > \omega_0$ are zeroed (late segment ablated), preserving only the early portion of the sequence.
3. **Late-only:** time steps $\omega \leq \omega_0$ are zeroed (early segment ablated), preserving only the late portion of the sequence.

Here ω_0 (equals to 350 in the example) is a fixed boundary time step defining the early/late split (the same split used to summarize probe results). For each epoch τ, we load the corresponding trained parameters and compute accuracy on both the training and test splits under each masking condition. Because masking is applied only at evaluation time, differences in accuracy directly reflect the extent to which the *existing trained decision rule* relies on early versus late parts of the sequence, rather than changes in training dynamics.

Figure A.18 reports the training-set accuracy across epochs for the intact model and the two ablated settings. Consistent with the main-text interpretation, early-only performance rises during early training but does not sustain the intact model's final performance, while late-only performance improves gradually over training yet remains below the intact sequence. These results not only indicate that final performance depends on integrating information across the full sequence but also provide insight into how the network's reliance on different sequence segments evolves during training. Early time-steps provide shortcuts that are initially leveraged by the classifier, but as training progresses, the reliance shifts toward later time-steps, culminating in a model that integrates information across the entire sequence for robust generalization.

Taken together, these results indicate that although both segments can support nontrivial classification when isolated, the strongest performance depends on coherent integration across the full sequence, and the network's reliance gradually migrates from early to later time-steps as training continues.

A.9 Time-resolved mutual information analysis

To measure the *total* task-relevant label information present in the recurrent representation, independent of any particular readout, we computed a time-resolved mutual information (MI) between class labels $\boldsymbol{L}$ and the hidden state h_t of the recurrent network $\boldsymbol{R}$. Let s denote the total number of time steps and m the number of hidden units. At a set of probed training epochs τ (subsampled consistently with the entropy and probing analyses), we extracted the hidden-state tensor on the training set and estimated MI at each time step and hidden unit.

Per-time-step, per-unit MI. For each epoch τ and time step $\omega \in \{1, \ldots, s\}$, we considered the hidden state vector across examples and treated each unit as a continuous feature. We estimated the mutual information between $\boldsymbol{L}$ and each unit activity at (τ, ω) using a nonparametric estimator for continuous features, producing a time–unit MI array. We then summarized the overall label information at each time

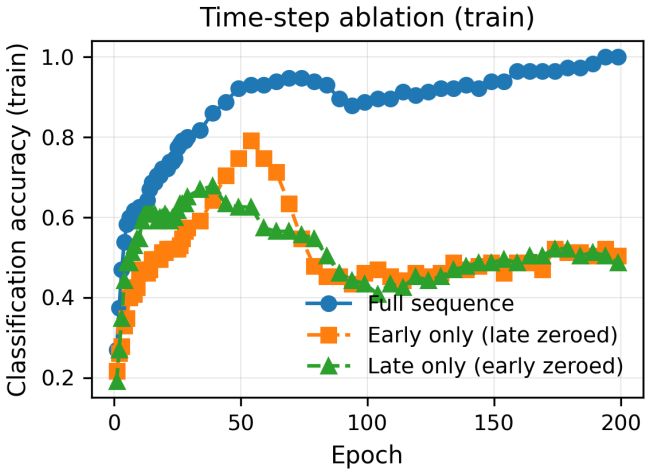

Figure A.18: Time-step ablation on the training split across training epochs. We compare the intact model (full sequence) to *early-only* evaluation (late time steps zeroed) and *late-only* evaluation (early time steps zeroed). The intact model consistently achieves higher accuracy than either ablated condition, indicating that final performance relies on integrating information across the full sequence rather than any single segment alone. The gradual shift from early reliance to full-sequence integration reveals the network's evolving strategy for learning and generalization.

step by averaging MI across units, yielding a time-resolved MI curve $I_\tau(\omega)$ that reflects how much label information is present in h_ω at epoch τ, regardless of whether the trained classifier exploits it.

Heatmap summary across epochs and time steps. Figure A.19 visualizes the mean-over-units MI $I_\tau(\omega)$ as a heatmap with axes (epoch τ) $\times$ (time step ω). This representation highlights how label information accumulates and redistributes across the sequence during training. In our experiments, MI increases over training and is consistently higher at later time steps, corroborating the probe and ablation results that later portions of the sequence carry more task-relevant information in the learned hidden dynamics (Fig. 5).

A.10 Random Label Analysis

A.10.1 Experimental setup

To probe how our MM-PHATE and entropy summaries behave in an extreme overfitting regime, we performed a random-label experiment inspired by Fischer (2020). The idea is to progressively destroy the correspondence between inputs and labels and ask whether the resulting representational changes are reflected in MM-PHATE geometry and intra-/inter-step entropy.

We trained multiple 1-layer LSTM models with 100 hidden units on the Area2Bump dataset under different label-shuffling conditions. For each condition, a specified number of classes had their labels randomly reassigned while the remaining classes retained their true labels. For each shuffle level, we drew 10 independent random class-permutation configurations. All models were trained with the same hyperparameters (learning rate 10^{-4}, batch size and optimizer as in Section 5.2), and most runs reached (near-)zero training loss within 200 epochs, while training on shuffled labels is known to severely degrade test performance.

We focused on the final training epoch and extracted hidden activations across all 600 time-steps. On this epoch we computed: (i) MM-PHATE embeddings of the hidden units across time and epoch, (ii) intra-step and inter-step entropies in the MM-PHATE space (as defined in Section 5.2), and (iii) summary statistics and distributional distances of the intra-step entropy across time-steps, using a clean-label model as a reference.

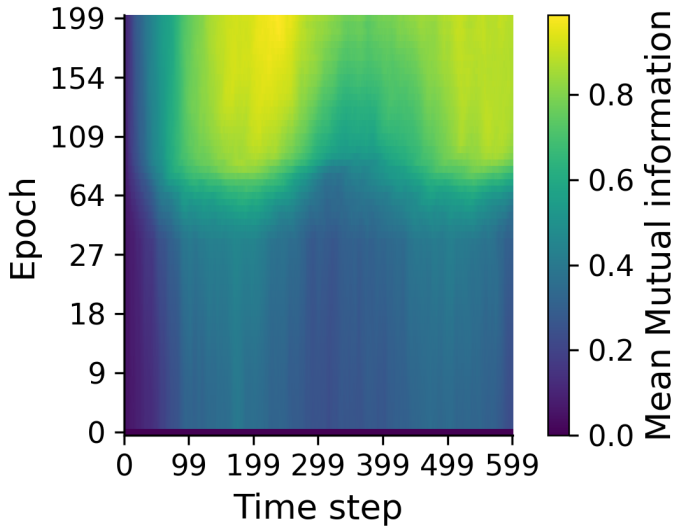

Figure A.19: Time-resolved mutual information between labels $\boldsymbol{L}$ and the recurrent hidden state h_ω, summarized by averaging MI across hidden units (m) at each epoch τ and time step ω. Warmer colors indicate greater label information present in the representation at that time and training stage. The heatmap shows increasing label information over training, with consistently higher MI at later time steps.

Specifically, for each shuffle level we summarized the intra-step entropy at the last epoch by: (i) maximum entropy across time-steps, (ii) mean entropy across time-steps, (iii) entropy at the last time-step, and (iv) variance of entropy across time-steps. We then compared the entropy distributions over time-steps between shuffle levels using Jensen–Shannon divergence (JSD) and total variation distance (TVD), both computed relative to a reference model trained on correct labels.

A.10.2 Metrics and analysis

Intra-step entropy. As in Section 5.2, intra-step entropy is computed at each time-step by applying a KDE-based estimator to the cloud of embedded hidden units at that time-step and epoch. It is therefore a measure of geometric dispersion across units in the MM-PHATE embedding, not a direct estimate of information-theoretic entropy of neural responses. In this experiment, we use its summary statistics as a compact way to track how the spread of unit representations across time-steps changes as label noise increases.

Inter-step entropy. Inter-step entropy is computed per unit and epoch, capturing how dispersed a unit's embedded trajectory is across time-steps. Larger inter-step entropy indicates that the unit differentiates more strongly across time within an epoch; smaller values indicate more temporally homogeneous activity. Here we examine how these unit-level temporal profiles change with label shuffling.

Distributional distances (JSD, TVD). To quantify how much the overall entropy profile at the last epoch deviates from the clean-label regime, we treat the intra-step entropy values across time-steps as a one-dimensional empirical distribution and compute: (i) the Jensen–Shannon divergence between this distribution and that of the clean-label model, and (ii) the corresponding total variation distance. These distances do not by themselves identify "good" or "bad" representations, but they provide a simple way to measure how strongly the entropy profile shifts as labels are progressively randomized.

A.10.3 Results

MM-PHATE geometry. With no or minimal label shuffling, MM-PHATE produces embeddings that show structured trajectories across epochs and a relatively organized arrangement of hidden units over

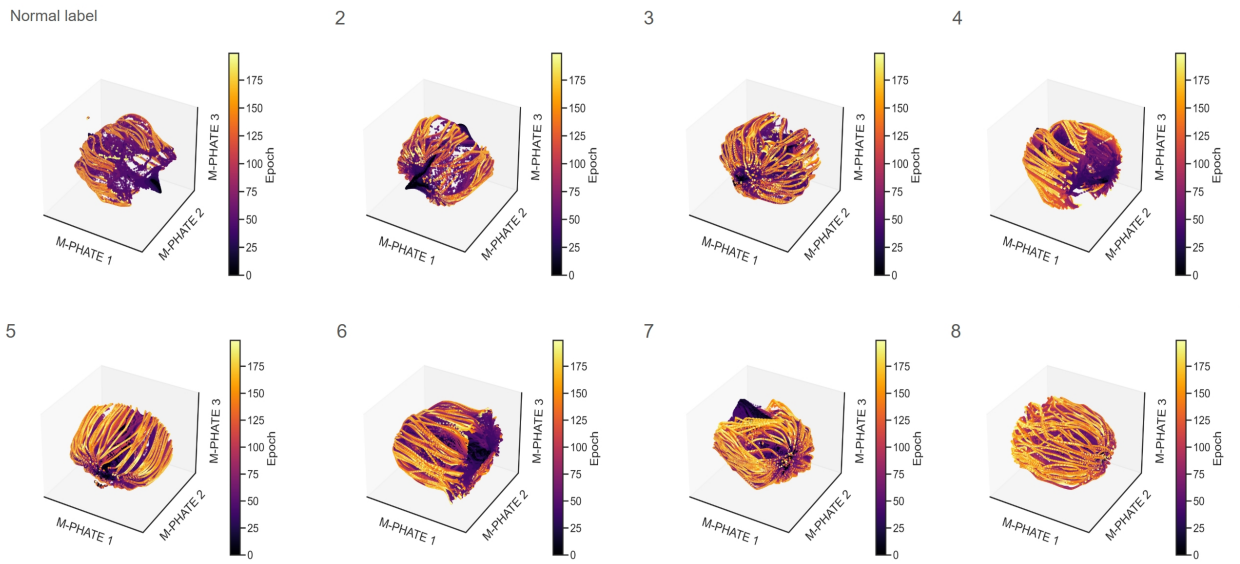

Figure A.20: Random label experiment: MM-PHATE visualization of 100-unit LSTM networks trained for 200 epochs on Area2Bump training data with different numbers of shuffled classes (indicated in the legend). Each point is a hidden unit at a given time-step and epoch. Points are colored by epoch.

time (Fig. A.20). As more classes are shuffled, these structures become progressively less differentiated: trajectories appear more tangled across epochs and units, and the embedding shows weaker separation consistent with the loss of stable, task-aligned structure. This qualitative change is not meant as a fine-grained diagnostic of class structure, but it demonstrates that MM-PHATE geometry is sensitive to the transition from meaningful learning to fitting random targets.

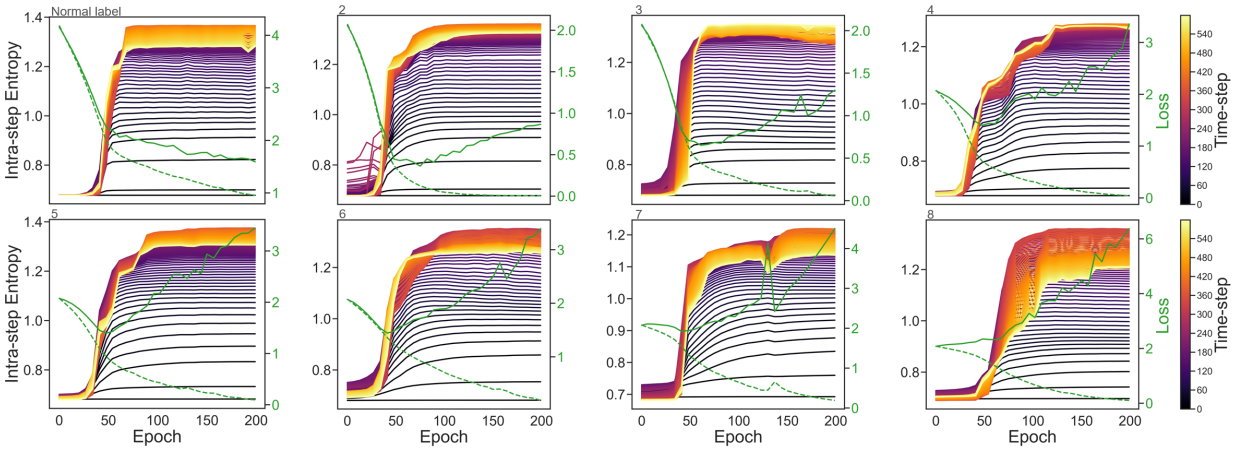

Figure A.21: Random label experiment: intra-step entropy of the MM-PHATE embedding for 100-unit LSTM networks trained for 200 epochs on Area2Bump, with different numbers of shuffled classes. Each curve shows the entropy at a given time-step across training epochs.

Intra-step and inter-step entropy trends. Across shuffle levels, intra-step entropy and inter-step entropy exhibit systematic changes (Figs. A.21–A.22). In the clean-label condition, intra-step entropy over training shows the structured temporal patterns described in Section 5.2. As more classes are shuffled, these patterns are attenuated and the entropy trajectories across epochs and time-steps become more homogeneous. At the final epoch, summary statistics of the intra-step entropy (maximum, mean, last time-step value, and

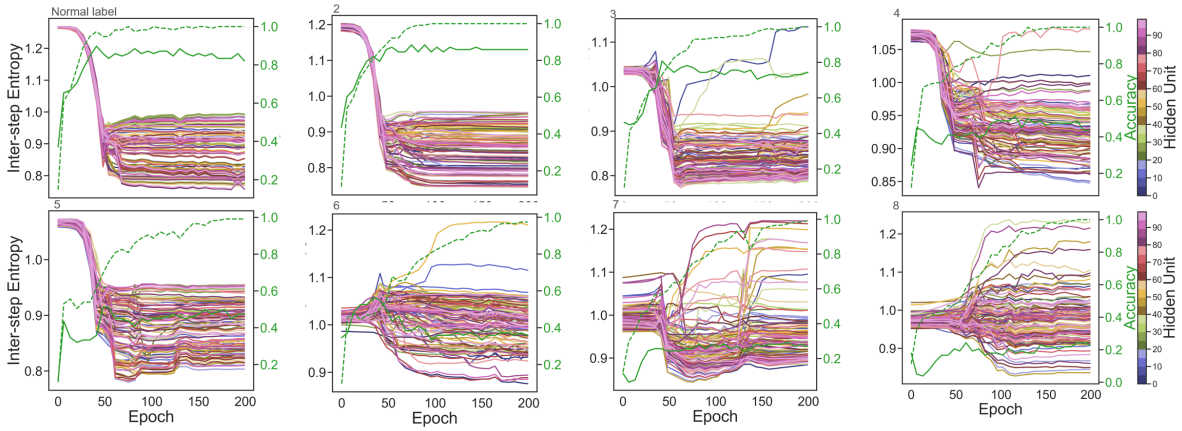

Figure A.22: Random label experiment: inter-step entropy of the MM-PHATE embedding for 100-unit LSTM networks trained for 200 epochs on Area2Bump, with different numbers of shuffled classes. Each curve shows the entropy trajectory of a hidden unit across time-steps at different epochs.

variance) decrease with increasing label shuffling (Fig. A.23), indicating that the embedding-space spread of unit representations across time-steps becomes more concentrated and less temporally differentiated under heavy label noise.

Inter-step entropy traces also change with shuffling. With true labels, units exhibit distinct temporal profiles over time-steps; as shuffling increases, many inter-step entropy trajectories become larger and less structured across epochs (Fig. A.22), consistent with units varying over time in ways that are less clearly aligned with the training phases seen in the clean-label regime. We interpret these trends as evidence that the temporal organization captured by MM-PHATE entropy summaries is disrupted when the network is forced to fit inconsistent labels.

Distributional distances. Finally, JSD and TVD between the intra-step entropy distributions (over time-steps) at the last epoch and the corresponding clean-label reference increase as more classes are shuffled (Fig. A.24). A similar trend is observed when computing these distances on the embedded point clouds themselves. These distances capture, in a single scalar per condition, how far the entropy profile has moved away from the structure observed in the clean-label model.

Overall, the random-label experiment serves as a stress test: when we deliberately destroy the input–label relationship, MM-PHATE geometry and the associated entropy summaries change in a systematic way. While we do not claim these measures uniquely quantify "representation quality," their sensitivity to this controlled overfitting regime supports their use as indicators of regime changes and loss of task-aligned structure in the main experiments.

A.11 Area2Bump with GRU

Here is the same analysis as section 5.2 using GRU (Fig. A.25, A.26, A.27). 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 5.2, 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.

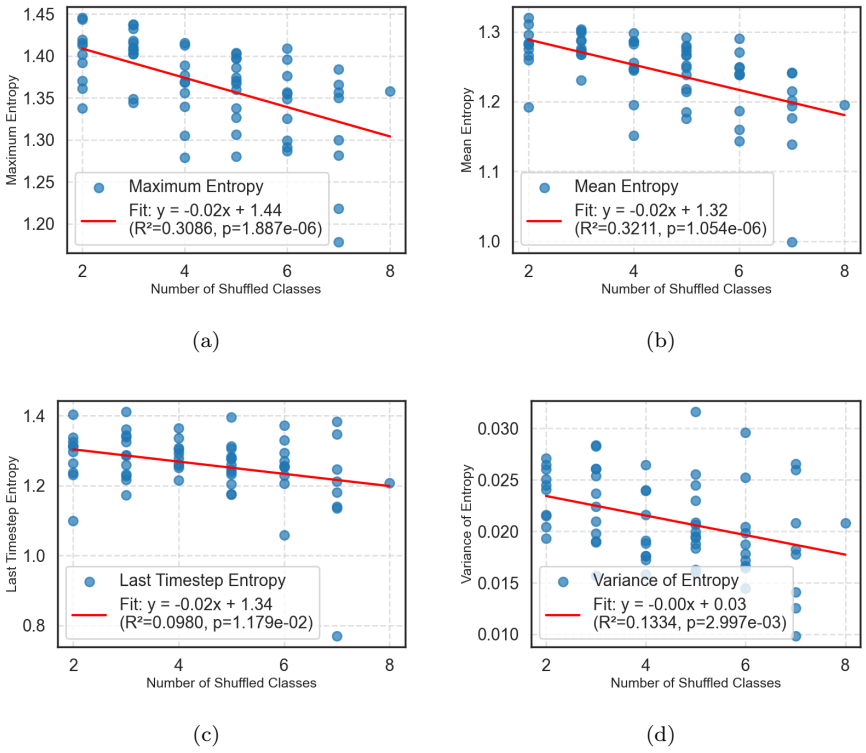

Figure A.23: Random label experiment: (a) maximum, (b) mean, (c) last time-step, and (d) variance of intra-step entropy at the last epoch for models trained with different numbers of shuffled classes.

A.12 Area2Bump with Vanilla RNN

Here is the same analysis as section 5.2 using vanilla RNN (Fig. A.28, A.29, A.30). 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.13 Area2Bump with LSTM of Various Sizes

Here we repeat the same MM-PHATE analysis as in section 5.2 using LSTM of various sizes (Fig. A.32, A.33, A.34). 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, 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.

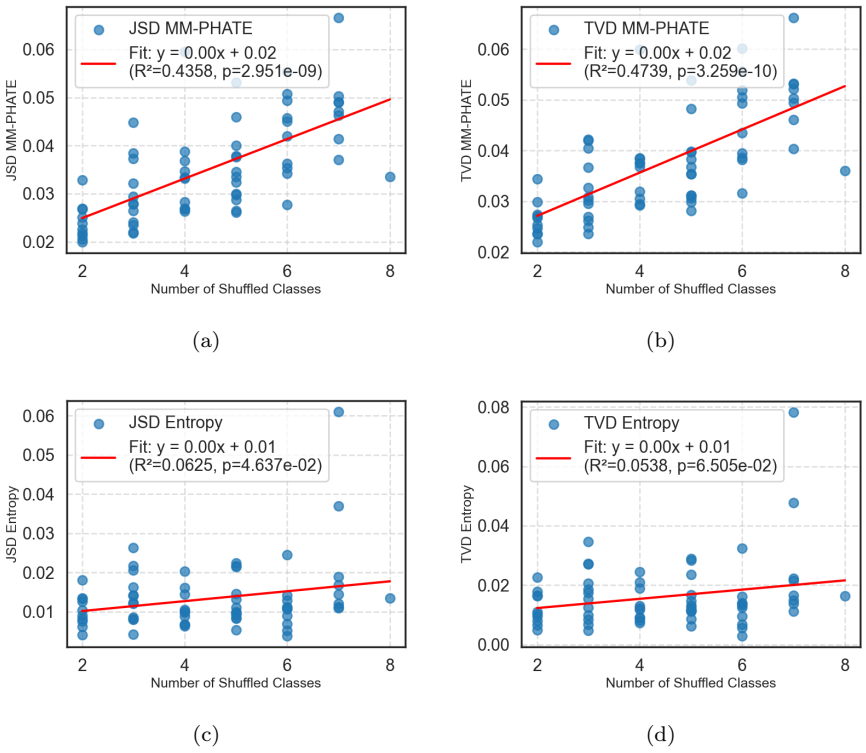

Figure A.24: Random label experiment: Jensen–Shannon divergence (JSD) and total variation distance (TVD) of the MM-PHATE embedding and intra-step entropy distribution at the last epoch, comparing models trained with different numbers of shuffled classes to a clean-label reference model.

B Computing Infrastructure

All but the t-SNE Area2Bump 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 Area2Bump 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.

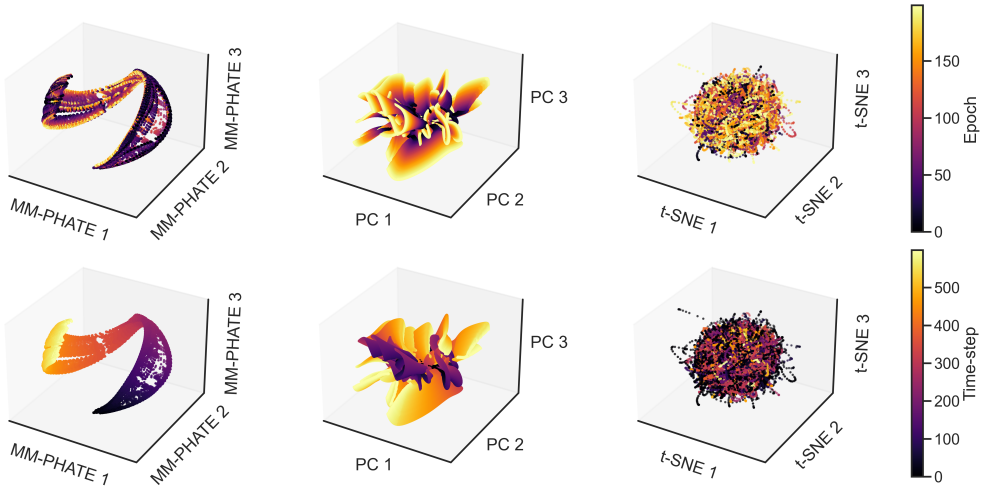

Figure A.25: Area2Bump GRU: Visualization of a 20-unit GRU network trained for 200 epochs. Each point represents a hidden unit at a specific time-step in a given epoch throughout the entire training process. The visualizations are generated using MM-PHATE, PCA, and t-SNE, from left to right, respectively. Points are colored based on epoch (top row) or time-step (bottom row)

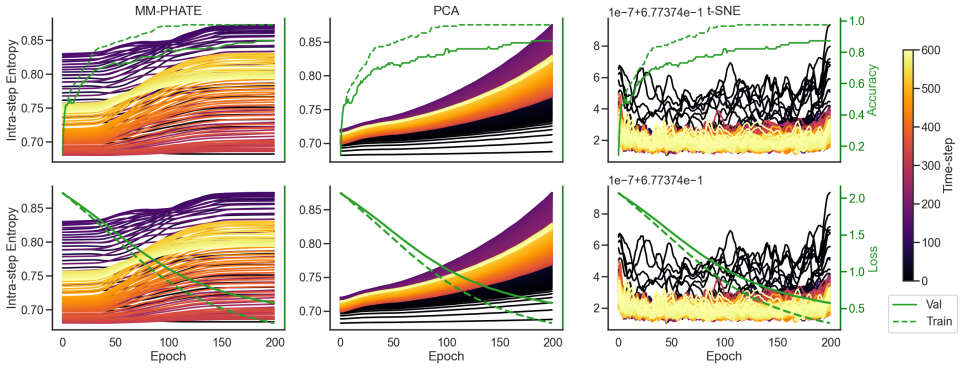

Figure A.26: Area2Bump GRU: Intra-step entropy of all hidden units in embedding space at each time-step in each epoch, compared to training and validation accuracy (top) and losses (bottom), comparing embeddings of MM-PHATE, PCA, and t-SNE.

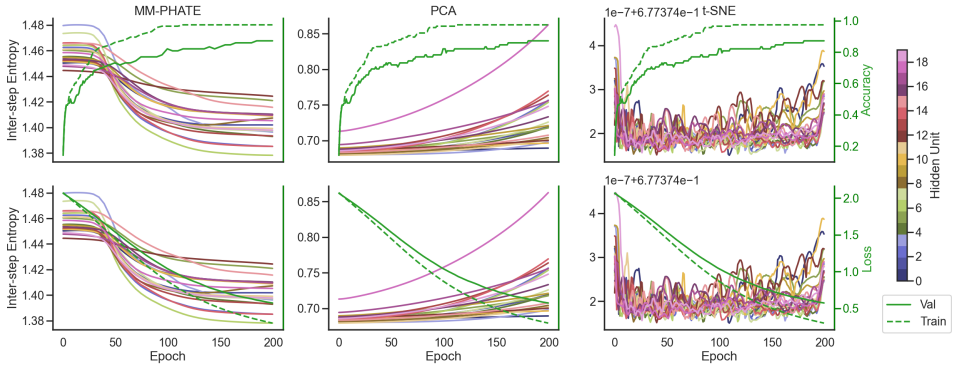

Figure A.27: Area2Bump GRU: Inter-step entropy of all hidden units in embedding space of the Area2Bump model at each time-step in each epoch, compared to training and accuracies (top) and losses (bottom). From left to right, the dimensionality reduction metrics used are MM-PHATE, PCA, and t-SNE.

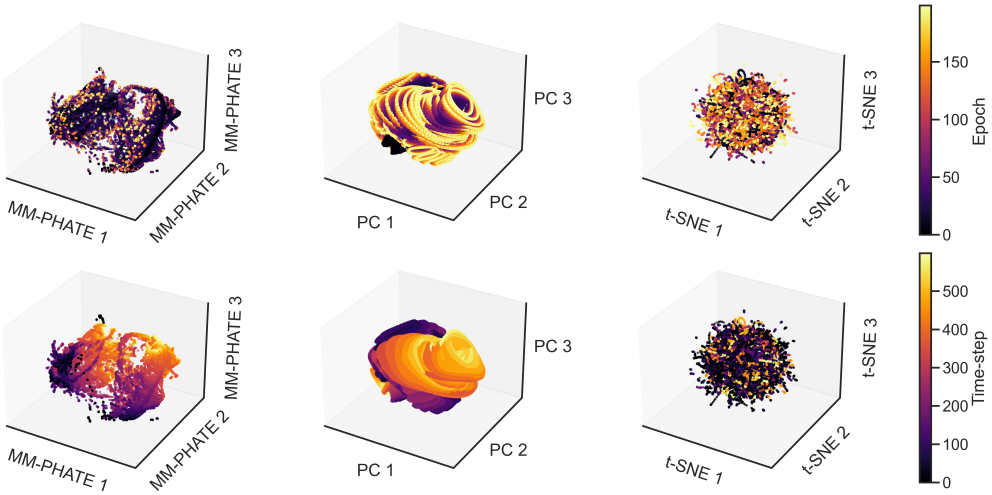

Figure A.28: Area2Bump Vanilla: Visualization of a 20-unit Vanilla RNN trained for 200 epochs. Each point represents a hidden unit at a specific time-step in a given epoch throughout the entire training process. The visualizations are generated using MM-PHATE, PCA, and t-SNE, from left to right, respectively. Points are colored based on epoch (top row) or time-step (bottom-row)

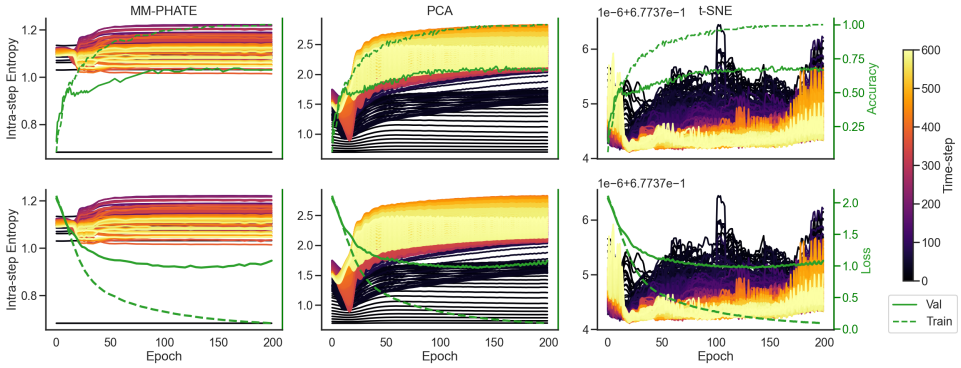

Figure A.29: Area2Bump Vanilla: Intra-step entropy of all hidden units in embedding space at each time-step in each epoch, compared to training and validation accuracy (top) and losses (bottom), comparing embeddings of MM-PHATE, PCA, and t-SNE.

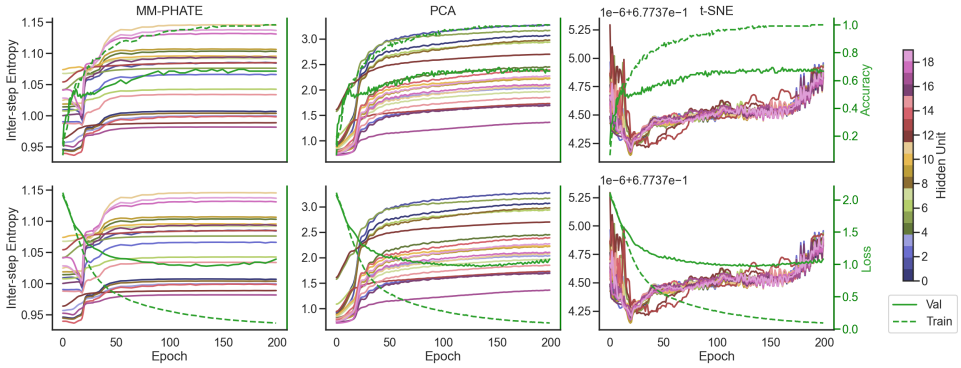

Figure A.30: Area2Bump Vanilla: Inter-step entropy of all hidden units in embedding space of the Area2Bump model at each time-step in each epoch, compared to training and accuracies (top) and losses (bottom). From left to right, the dimensionality reduction metrics used are MM-PHATE, PCA, and t-SNE.

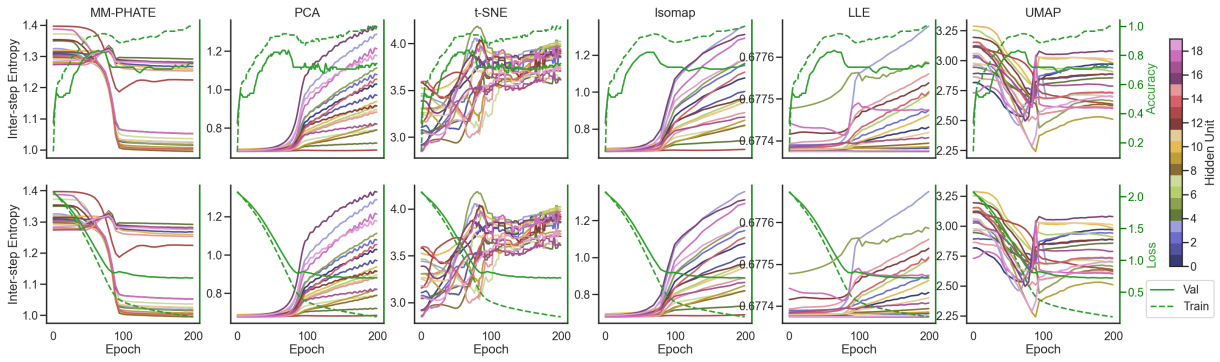

Figure A.31: Area2Bump: Inter-step entropy of all hidden units in embedding space of the Area2Bump model at each time-step in each epoch, compared to training and accuracies (top) and losses (bottom). From left to right, the dimensionality reduction metrics used are MM-PHATE, PCA, t-SNE, Isomap, LLE, and UMAP.

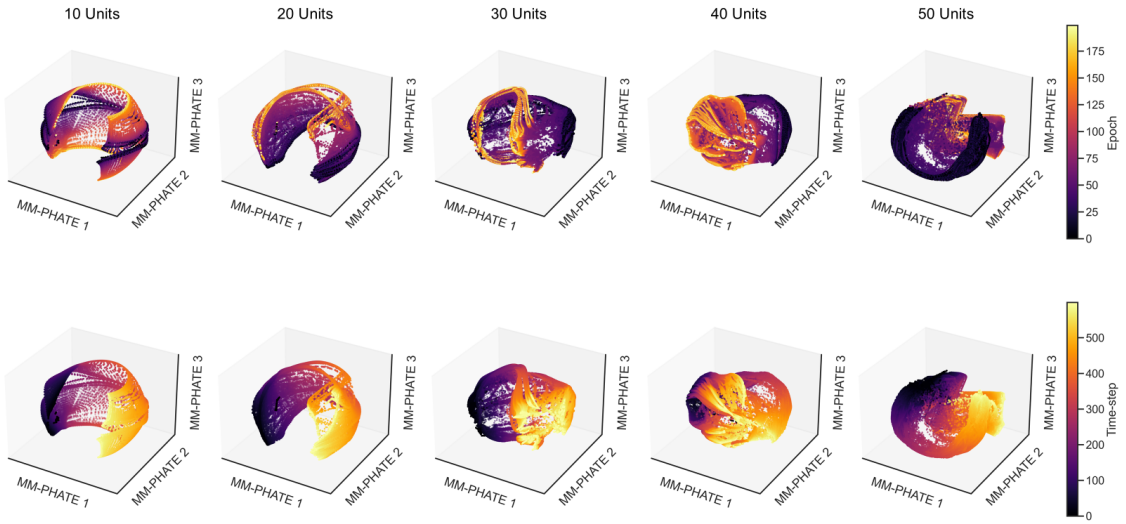

Figure A.32: Area2Bump LSTM: MM-PHATE visualization of networks of size 10 to 50 (left to right). Each point represents a hidden unit at a specific time-step in a given epoch. Points are colored based on epoch (top) or time-step (bottom).

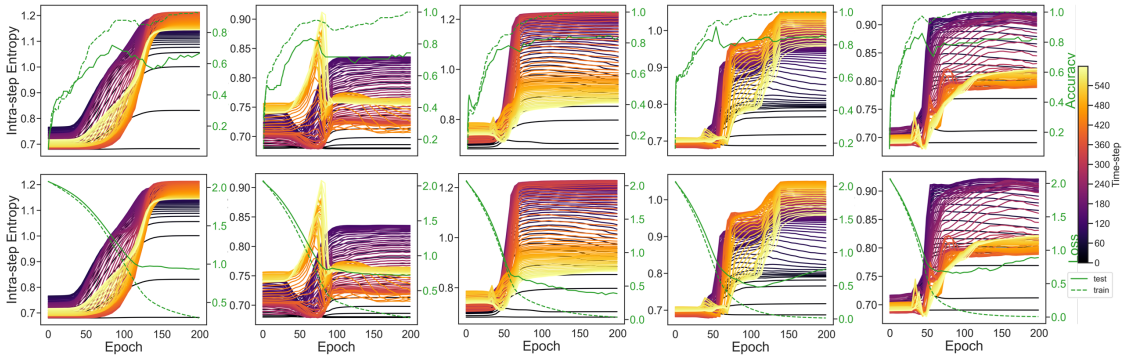

Figure A.33: Area2Bump LSTM: Intra-step entropy of all hidden units in MM-PHATE embedding space at each time-step in each epoch, compared to accuracies (top) and losses (bottom). Network sizes range from 10 to 50 (left to right).

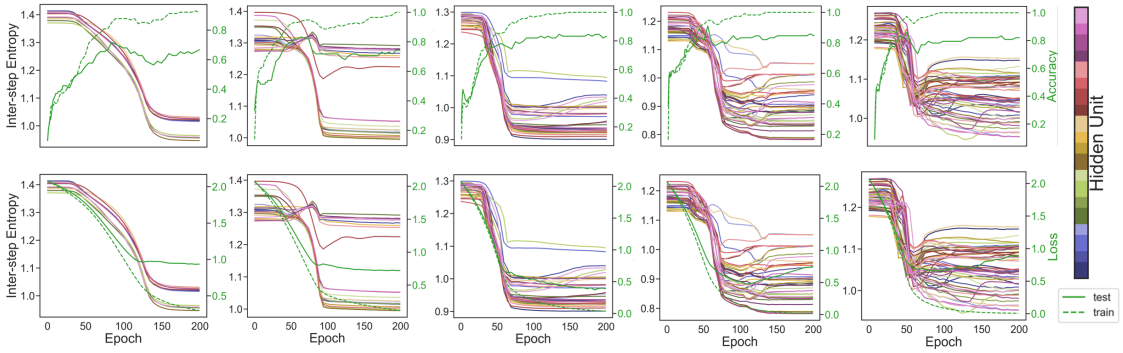

Figure A.34: Area2Bump LSTM: Inter-step entropy of all hidden units in MM-PHATE embedding space of the Area2Bump model at each time-step for each unit in each epoch, compared to accuracies (top) and losses (bottom). Network sizes range from 10 to 50 (left to right).

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 $\boldsymbol{Y}$
$\boldsymbol{K}^{(\tau)}_{\text{intraslice}}(i,j)$	Intraslice affinities between pairs of hidden units within an epoch τ
$\boldsymbol{K}^{(i)}_{\text{interslice}}(\tau,\upsilon)$	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
$\boldsymbol{K}^{(\tau,\omega)}_{\text{intra-step}}(i,j)$	Intra-step affinities between hidden units i and j at time-step ω in epoch τ
$\boldsymbol{K}^{(i)}_{\text{inter-step}}((\tau,\omega),(\eta,\nu))$	Inter-step affinities between a hidden unit i and itself at different time-steps and epochs
$\sigma_{(\tau,\omega,i)}$	Intra-step bandwidth for unit i at time-step w and epoch τ
k	Number of nearest neighbors
$\boldsymbol{L}$	Labels of $\boldsymbol{X}$

Table 3: Notations