BundDLe-Net: Neuronal Manifold Learning Meets behaviour

Akshey Kumar[1,3], Aditya Gilra[2], Mauricio Gonzalez-Soto[3], and Moritz Grosse-Wentrup[3,4,5]

[1]UniVie Doctoral School Computer Science DoCS, University of Vienna
[2]Centrum Wiskunde & Informatica, Amsterdam, Netherlands
[3]Research Group Neuroinformatics, University of Vienna
[4]Vienna Cognitive Science Hub
[5]Data Science @ Uni Vienna

August 8, 2023

Abstract

Neuronal manifold learning techniques represent high-dimensional neuronal dynamics in low-dimensional embeddings to reveal the intrinsic structure of neuronal manifolds. Common to these techniques is their goal to learn low-dimensional embeddings that preserve all dynamic information in the high-dimensional neuronal data, i.e., embeddings that allow for reconstructing the original data. We introduce a novel neuronal manifold learning technique, BundDLe-Net, that learns a low-dimensional Markovian embedding of the neuronal dynamics which preserves only those aspects of the neuronal dynamics that are relevant for a given behavioural context. In this way, BundDLe-Net eliminates neuronal dynamics that are irrelevant to decoding behaviour, effectively de-noising the data to reveal better the intricate relationships between neuronal dynamics and behaviour. We demonstrate the quantitative superiority of BundDLe-Net over commonly used and state-of-the-art neuronal manifold learning techniques in terms of dynamic and behavioural information in the learned manifold on calcium imaging data recorded in the nematode *C. elegans*. Qualitatively, we show that BundDLe-Net learns highly consistent manifolds across multiple worms that reveal the neuronal and behavioural motifs that form the building blocks of the neuronal manifold.

1 Introduction

Advances in neuronal imaging techniques have increased the number of neurons that can be recorded simultaneously by several orders of magnitude [2, 8]. While these advances greatly expand our abilities to study and understand brain function, the complexities of the resulting high-dimensional data sets pose non-trivial challenges for data analysis and visualization.

This work was supported by the CHIST-ERA grant CHIST-ERA-19-XAI-002, by the Austrian Science Fund (FWF) (grant number I 5211-N), as part of the Causal Explanations in Reinforcement Learning (CausalXRL) project.

Fortunately, individual neurons are embedded in brain networks that collectively organise their high-dimensional neuronal activity patterns into lower-dimensional neuronal manifolds [4, 19]. To understand the collective organization of individual neurons into brain networks, we require algorithms that learn neuronal manifolds from empirical data.

The goal of neuronal manifold learning is to find low-dimensional representations of data that preserve particular data properties. In neuroscience, a broad range of classical dimensionality reduction techniques is being employed, including but not limited to principal component analysis (PCA), multi-dimensional scaling (MDS), Isomap, locally linear embedding (LLE), Laplacian eigenmaps (LEM), t-SNE, and uniform manifold approximation and projection (UMAP) [15]. More recently, advances in artificial intelligence in general and deep learning methods, in particular, have given rise to a new class of (often non-linear) dimensionality reduction techniques based on autoencoder architectures [16, 21, 22].

Common to all these techniques is their focus on finding low-dimensional data representations that allow a good (or even perfect) reconstruction of the original, high-dimensional data. In contrast, we argue that reconstruction quality is only one out of several desirable features for neuronal manifold learning. First, and in line with the argument by Krakauer et al. [11] that neuroscience needs behaviour, we argue that a neuronal manifold learning algorithm should not aspire to represent all but only those characteristics of high-dimensional neuronal activity patterns that are relevant in a given behavioural context. For instance, when studying an animal's ability to navigate a maze using visual cues, neuronal activity patterns that carry auditory or olfactory information are irrelevant in the behavioural context and should be abstracted away to better reveal the intricate relationships between neuronal representations of the visual cues and motor behaviour. Second, we argue that the reconstruction of the dynamics of the neuronal activity patterns should also take into account whether the low-dimensional embedding is causally sufficient in terms of the system's dynamics. To elaborate on this issue, consider the example of using a dimensionality reduction technique to learn the physical state description of a simple pendulum from a video stream showing the pendulum in action. Ideally, the dimensionality reduction technique should learn to represent the position and momentum of the pendulum for each video frame because these two variables constitute a full description of the system's physical state. In contrast, a dimensionality reduction technique that learns to represent the positions of the pendulum in the current and the past video frame only (without representing the pendulum's momentum) would also allow for a good reconstruction of the dynamics of the pendulum. This is the case because the pendulum's momentum, which is required to predict in which direction it will swing, can be approximately reconstructed from the difference in position across two video frames. However, this representation would not constitute a complete description of the actual physical state of the system. In analogy, a neuronal manifold learning technique should attempt to learn a complete physical state description of the underlying neuronal dynamics. Mathematically, this goal can be formulated as learning neuronal state trajectories that form a Markov chain because, in a Markov chain, the current state of the chain is causally sufficient for predicting the next state (in mathematical terms, the past and future states of the chain are statistically independent given the current state).

Here, we introduce a novel framework for neuronal manifold learning, termed the Behaviour and Dynamics Learning Network (BundDLe-Net). BundDLe-Net learns a low-dimensional Markovian representation of the neuronal dynamics while retaining all information about a given behavioural context. It is based on the architecture shown in Fig. 1, which consists of two branches. In the lower branch, the high-dimensional neuronal trajectories X_t are first projected via a mapping τ

to a lower-dimensional, latent trajectory Y_t^L. A first-order transition model T_Y then predicts the difference ΔY_t^L between the current and the next state to arrive at an estimate Y_{t+1}^L of the latent state at time $t+1$. This predicted latent state is compared to the true latent state at time $t+1$ in the upper branch, which is obtained by mapping the observed neuronal state X_{t+1} at time $t+1$ via the same τ as in the lower branch to the latent state Y_{t+1}^U via the loss function $\mathcal{L}_{\text{Markov}}$. By jointly learning the mapping τ and the first-order transition model T_Y that minimise the loss function $\mathcal{L}_{\text{Markov}}$ we obtain a latent, low-dimensional time-series Y_t that is Markovian by construction. This is the case because the transition model T_Y acts as a bottleneck that constrains the class of functions for τ for which the current state of the system is sufficient to predict the next state, in the sense that previous states do not provide any additional information. However, this architecture is not yet sufficient to learn a meaningful latent data representation because a mapping τ that projects the neuronal state trajectories to a constant ($Y_t = c$) would also fulfill the criterion of Markovianity. To obtain a meaningful latent representation, we also require that the behavioural context must be decodable from the latent representation Y_t by adding the loss function $\mathcal{L}_{\text{behaviour}}$ that measures the reconstruction error between the true behavioural labels (B_{t+1}) and those predicted from the latent representation ($\tilde{B}_{t+1}$). By jointly learning that mapping τ and the first-order state transition model T_Y that minimise the two loss functions $\mathcal{L}_{\text{Markov}}$ and $\mathcal{L}_{\text{behaviour}}$, the BundDLe-Net architecture learns low-dimensional Markovian representations of those aspects of the high-dimensional neuronal state trajectories that are relevant for a given behavioural context.

We remark that BundDLe-Net is a generic architecture in the sense that each of its modules (the mapping τ, the state transition model T_Y, and the prediction model for the behaviour) can be realised by whatever models, e.g., linear or non-linear mappings which may be realised via (deep) neuronal networks or other modeling techniques, are most suitable for a certain type of neuronal data. The BundDLe-Net architecture is available as a *Python* toolbox at https://github.com/akshey-kumar/BunDLe-Net.

In the following, we compare the BundDLe-Net architecture with other state-of-the-art neuronal manifold learning techniques on calcium imaging data recorded in the nematode *C. elegans* [7] and demonstrate its ability to uncover intricate relationships between neuronal activity patterns and behaviour that are not revealed by competing techniques.

2 Results

In the following, we demonstrate how BundDLe-Net preserves vital information about the behavioural dynamics while simultaneously enabling visually interpretable insights into the data. We start with a quantitative evaluation of BundDLe-Net and compare it with existing state-of-the-art neuronal manifold learning techniques. We then examine the visual interpretability of the embeddings of BundDLe-Net and competing algorithms. To ensure the robustness of our findings, we apply BundDLe-Net to five different worms and analyse the consistency of the embeddings in terms of their topology. The results highlight the generalisation abilities of BundDLe-Net, revealing similar patterns while maintaining individual differences across recordings. Finally, we show that BundDLe-Net is capable of embedding behaviours in distinct motifs based on the neuronal basis of the behaviour and its dynamics.

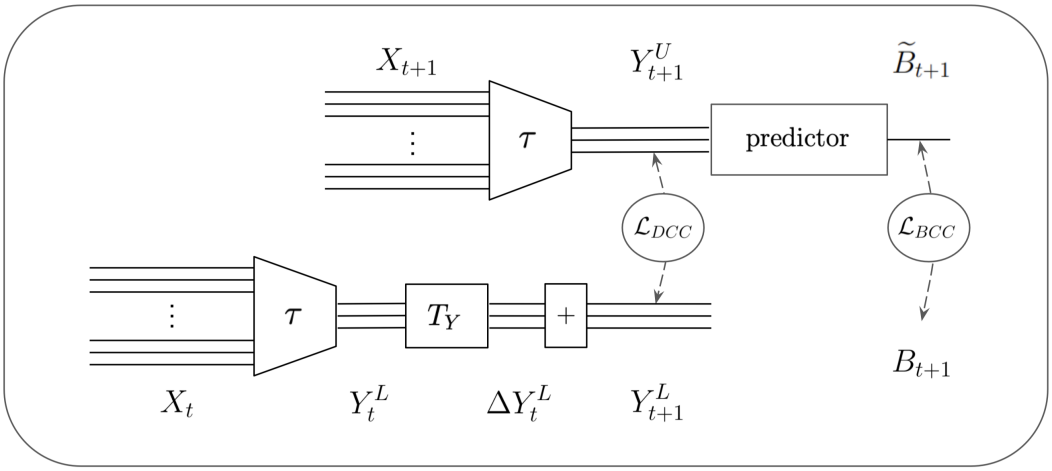

Figure 1: The BundDLe-Net architecture

2.1 Description of data

We apply BundDLe-Net to calcium-imaging whole brain data from the nematode *C. elegans* from the work by Kato et al. [7]. This dataset is ideal for demonstrating the capabilities of BundDLe-Net due to its high-dimensional neuronal recordings labelled with motor behaviour[1], multiple animal recordings, eight different behavioural states, and multiple repetitions of behavioural states over time. It includes time-series recordings of neuronal activation from five worms with human-annotated behaviours for each time frame. The recordings consist of approximately 2500-3500 time samples, spanning around 18 minutes (sampled at ~ 2.9 Hz) in which around 100 - 200 neurons are recorded. A low-pass filter with a cut-off frequency of 0.07 Hz is applied to mitigate high-frequency noise in the raw neuronal traces. Not all recorded neurons could be identified; hence, only a subset is labeled for each worm, with different yet overlapping subsets identified across worms. The human-annotated behaviours B denote the motor state of the worm at a given instant of time and can take on one of eight states: forward, slowing, dorsal turn, ventral turn, no-state, sustained reversal, reversal-1, and reversal-2.

2.2 Quantitative evaluation against competing methods

We evaluate a latent space representation based on how well it preserves behavioural and dynamical information. To estimate the *behavioural information* of an embedding, we train a simple feed-forward neural network in a supervised setting[2] to predict behaviour from the embedding. The decoding accuracy is then used as a metric for the information content about B in the embedding, with the decoding accuracy obtained on the raw, high-dimensional neuronal traces serving as the baseline. To evaluate the *dynamical information* in the embedding, we train an ANN autoregressor to predict Y_{t+1} from Y_t. The mean squared error between the predicted and

[1]The motor behavioural labels were inferred from the activity of the neurons AVAR, AVAL, SMDVR, SMDVL, SMDDR, SMDDL, RIBR, RIBL while the worms were immobilised. Hence, we removed these neurons from the dataset to ensure we are not inferring behaviours directly from these neurons.

[2]We use a simple architecture consisting of a single linear layer since it already demonstrated a high decoding accuracy (~ 0.94) on the raw neuronal traces. Hence, more complex models are not required to evaluate the embeddings.

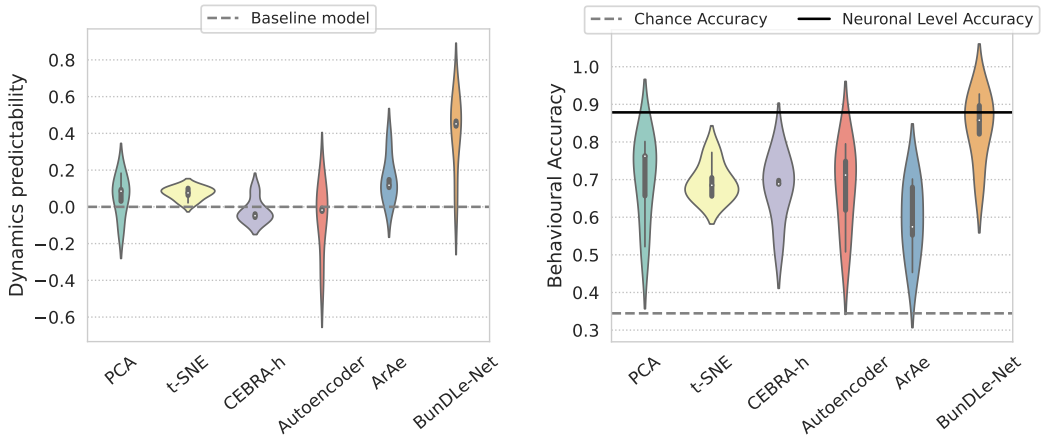

Figure 2: Quantitative comparison of BundDLe-Net with other commonly-used manifold learning techniques. (left) Evaluation of how well dynamical information is preserved in the embedding. The dashed line represents a baseline autoregressor which copies its input to the output. (right) Evaluation with respect to how well behavioural information is preserved. The dashed line represents the chance decoding accuracy, estimated by randomly shuffling the behavioural labels. The solid line represents the behaviour decoding accuracy from the raw neuronal traces.

true Y_{t+1} is estimated. From this, we compute a predictability metric for the dynamics, defined as $1 - \text{MSE}_m/\text{MSE}_{io}$, where MSE_m is the mean squared error of the model, and MSE_{io} is the mean squared error of a trivial autoregressor that copies its input to the output. We trained all evaluation models on a training set of the embedded data and performed the evaluation on a held-out test set to prevent overfitting (for more details see Model validation in section 4.3).

With the stage for evaluation set, we compare BundDLe-Net with other algorithms that are commonly used to learn high-level representations in the field of neuroscience such as PCA, t-SNE, autoencoder, an ANN autoregressor with an autoencoder architecture (ArAe)[3] and CEBRA-hybrid[4]. A description of these methods can be found in Section 4.4. All embedding spaces were chosen to be 3-dimensional for ease of comparison across algorithms and visualisation purposes.

Figure 2 presents the outcomes of our quantitative comparison, showcasing dynamical and behavioural prediction metrics in the left and right panels, respectively. Each panel depicts the predictability metric on the y-axis and the manifold learning technique on the x-axis, while the violin plots portray the metric's distribution across all five worm datasets. The substantial variability across these plots underscores the diverse behavioural and dynamical attributes inherent in each worm dataset. For the dynamics evaluations, we compare all the models to a baseline model which simply copies the input Y_t to the output as the predicted value for Y_{t+1}. For the behaviour evaluation, we compare with a chance level behavioural decoding accuracy obtained by randomly shuffling the behavioural labels.

[3]The ArAe would preserve dynamical information and embed it in a lower dimensional space due to the autoencoder architecture.

[4]Note that CEBRA as an algorithm was designed for continuous-valuled behaviours. We cast our categorical behaviour (int) into a continuous behaviour (floating-point) and ran CEBRA on it.

Turning to the results, we see that BundDLe-Net outperforms all other methods, including the state-of-the-art CEBRA, by a large margin. In the left panel, unsupervised methods like PCA, t-SNE, and the autoencoder show limited improvement over the baseline in predicting dynamics. Since they try to preserve maximum variance in the data in a low-dimensional space, they neglect to preserve minor details that may be crucial in determining the future time dynamics. CEBRA-hybrid which also takes temporal information into consideration, does not perform significantly better than the baseline model. The autoregressive-autoencoder, which seeks to reconstruct X_{t+1} from X_t preserves some dynamical information, and is seen to outperform PCA, t-SNE, and the autoencoder. Nonetheless, ArAe's reconstruction of the entire neuronal state at time $t+1$ can lead to irrelevant details persisting in latent space embedding. In contrast, BundDLe-Net's design focuses exclusively on retaining information pertinent to latent space state at time $t+1$, which results in a markedly superior performance even compared to ArAe.

Shifting our attention to the right panel, most models surpass chance-level behaviour decoding accuracy. Notably, both CEBRA-h and the unsupervised methods (PCA, t-SNE, autoencoder) exhibit roughly the same performance on average. Despite this, their average decoding accuracy remains notably lower than neuronal-level decoding accuracy, indicating an inability to fully capture behavioural information present at the neuronal level. Although ArAe worked slightly better at preserving dynamical information, it falls short in preserving behavioural information. This suggests that unsupervised preservation of dynamical attributes alone does not suffice for constructing behaviourally relevant models. In this regard, BundDLe-Net stands out, manifesting heightened behavioural information capture relative to other models, to the extent that, on average it rivals decoding performance at the neuronal level.

Of particular interest is the comparison between CEBRA-h and BundDLe-Net in terms of their respective performances. Despite incorporating behavioural information in addition to dynamics, CEBRA-h demonstrates only marginal improvements over other models. In contrast, BundDLe-Net rises above all other methods, excelling in both behavioural and dynamical metrics. This highlights BundDLe-Net's proficiency in effectively retaining crucial neuronal-level information relevant to behaviour analysis and modelling.

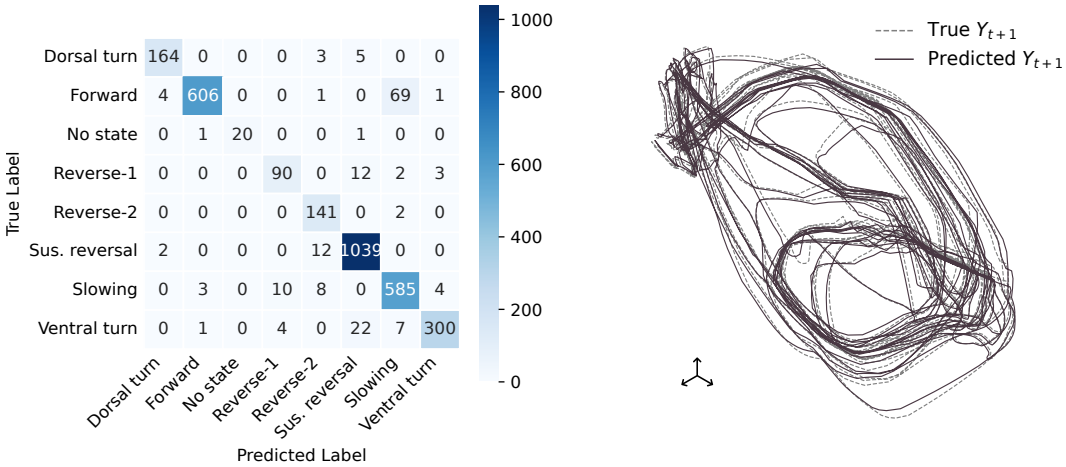

Figure 3: (left) Confusion matrix of behaviour predictions from BundDLe-Net's embedding for Worm-1 (right) True dynamics and dynamics predicted by BundDLe-Net for Worm-1.

In the following, we provide further information to build an intuition for the behavioural and dynamic prediction performance of BundDLe-Net. In Figure 3 a), we present the confusion matrix for BundDLe-Net's behavioural *prediction layer* from the ANN architecture. BundDLe-Net achieves a decoding accuracy of 94.3%, with the few decoding errors dominated by confusion of forward and slowing, two behaviours that are qualitatively similar and only quantitatively differ in the speed of the motion. To evaluate the dynamical performance of the model, we use the *transition model layer* T_Y to estimate Y_{t+1} from Y_t and compare it with the true Y_{t+1}, obtained as $\tau(X_{t+1})$. Figure 3 b) shows that the predicted dynamics indeed track the true dynamics rather well. These results indicate that the behaviour predictor and transition model within BundDLe-Net do well to preserve dynamical and behavioural information as intended.

2.3 Visual interpretability of embeddings

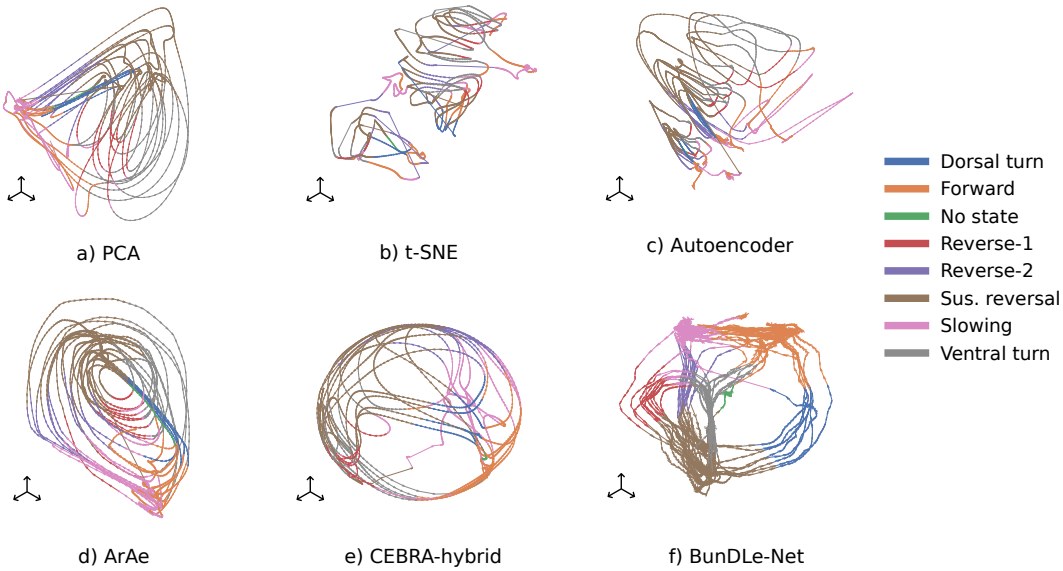

Figure 4: Neuronal manifolds learnt by various algorithms viz. a) PCA b) t-SNE c) Autoencoder d) Autoregressor-Autoencoder (ArAe) e) CEBRA-hybrid f) BundDLe-Net.

In this section, we analyse the embeddings of BundDLe-Net and other competing neuronal manifold learning techniques. We visualise the embeddings of the Worm-1 in 3D and evaluate them qualitatively based on their structure and interpretability. We generalise the insights to all worms in the next section. Figure 4 shows the embeddings of Worm-1 by a) PCA, b) t-SNE, c) autoencoder, d) Autoregressor-Autoencoder (ArAe), e) CEBRA-hybrid, and f) BundDLe-Net. In a), b) and c), we observe a noticeable drift in the PCA, t-SNE, and autoencoder embeddings. This drift drags out the dynamics in time, which is undesirable since we are searching for consistent mappings independent of time. The drift is also seen to obscure the recurrent nature of the dynamics to a large extent in b). The source of this drift could be a calcium imaging artifact or some neuronal dynamics irrelevant to our behaviour of interest. Since these models aim to preserve maximum variance for full-state reconstruction, they inadvertently embed the drift.

In contrast, in Figure 4 d), e), f), we see that this drift is largely absent, and the recurrent

dynamics are more evident. These models have a common characteristic: they consider dynamics without attempting to reconstruct the entire neuronal state. Among the three methods shown, ArAe is unsupervised, while CEBRA and BundDLe-Net take behaviour into account. In both d) and e), we observe reasonably separated behaviours with minor trajectory overlaps. However, both embeddings demonstrate high variance *within* a trajectory of a given behaviour. In contrast, BundDLe-Net produces compact bundles that are well-separated from one another. The variance is low within each bundle, while a high variance is observed between different bundles. Consequently, BundDLe-Net's embedding exhibits distinct behavioural trajectories that are well-separated and along which the dynamics recur in an orbit-like fashion.

Additionally, in e), we observe that CEBRA-h tends to embed the neuronal activity on the surface of a sphere, which may be an artifact resulting from the contrastive learning paradigm. As a consequence, trajectories can intersect at certain points. Such intersection points are generally undesirable because they introduce ambiguity regarding the future trajectory. Ideally, intersection points should only occur when there is genuinely no information available about the subsequent behavioural trajectory.

In stark contrast, BundDLe-Net's trajectories demonstrate a markedly different pattern, characterised by high compactness and sparse intersections. Figure 4 f) reveals precisely three intersection points: sustained reversal ●, ventral turn ●, and forward ●. (See supplementary material for rotating 3-D plots) Importantly, these intersections signify instances where BundDLe-Net indeed encountered a lack of information about future trajectories. Hence, such bifurcations and intersections represent truly probabilistic regions within the trajectory, where uncertainty genuinely manifests in the data.

2.4 Consistency of neuronal manifolds across worms

Here, we apply BundDLe-Net to all five worms in the dataset to visually compare the embeddings regarding their consistency and/or any differences that arise across worms.

To produce comparable embeddings, we first trained a model on each worm separately. We then extracted the T_Y layer and behaviour predictor layer from the model with the least loss (Worm-1, in this case). We then trained fresh models on each worm, with the chosen T_Y and behaviour predictor layers from Worm-1 frozen in. Thus the new models would only have to learn the mapping τ for each worm while the other layers remained unchanged throughout the learning process. Notably, this approach was feasible despite the recording of different neurons from each worm. By adopting this strategy, we ensured consistent geometries across the worms, allowing us to effectively compare differences in topology, should they be present.

The embeddings are illustrated in Figure 5. A latent dimension of three was again chosen for ease of visualization, and can also be justified by a graph-theoretical argument detailed in Section 4.3. Examining Figure 5, we observe a branching structure in the trajectories of all the worms. For now, let us consider Worm-1. The dynamics exhibit bundling of several segments, leading to recurring patterns along these bundles. Within each branch, the dynamics are predominantly deterministic, while probabilistic *decisions* occur only at specific bifurcation points in the trajectories. We disregard bundles consisting of fewer than one or two segments and identify five prominent bundles in Worm-1, which can be described as follows,

$(\boldsymbol{C_1})$: sustained reversal ● $\rightarrow$ ventral turn ●

$(\boldsymbol{C_2})$: ventral turn ● $\rightarrow$ slowing ● $\rightarrow$ reversal-1 ● $\rightarrow$ sustained reversal ●

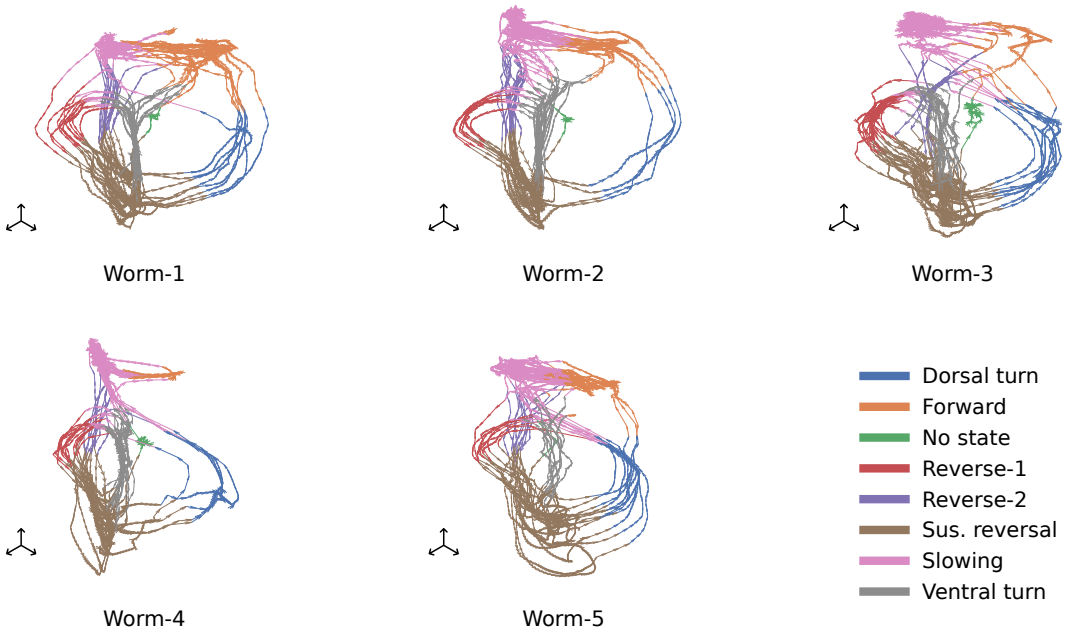

Figure 5: BundDLe-Net embeddings on five different *C. elegans* worm datasets which include neuronal recordings and behavioural labels

$(\boldsymbol{C_3})$: ventral turn ● → forward ●

$(\boldsymbol{C_4})$: sustained reversal ● → dorsal turn ● → forward ●

$(\boldsymbol{C_5})$: forward ● → slowing ● → reversal-2 ● → sustained reversal ●

These five motifs define the generic building blocks of the neuronal manifold in the sense that the neuronal trajectories are almost deterministic within each motif and probabilistic bifurcations occur at the transitions between motifs. As can be readily checked in Figure 5, these building blocks are highly consistent across worms, with similar behavioural motifs emerging across all worms. For example, motif $\boldsymbol{C_2}$ is consistently present in the embeddings of all worms, forming a loop (● → ● → ● → ●). The same holds true for motifs $\boldsymbol{C_1}$ and $\boldsymbol{C_5}$. However, motif $\boldsymbol{C_4}$ is not present in all worms and is notably absent in Worm-4. Instead, both Worm-4 and Worm-5 exhibit a slightly different motif (sustained reversal ● → dorsal turn ● → slowing ●).

It is noteworthy that even though the individual worm recordings do not share an identical subset of neurons, the embeddings share a basic topological structure with only minor variations in transitions and bifurcation points. These results demonstrate consistency in the embeddings across worms while preserving individuality in the behavioural dynamics in each worm and recording session.

2.5 Embedding of states in distinct behavioural motifs

Figure 6: (left) BundDLe-Net trajectory of Worm-1 highlighting the embedding of slowing behaviour (in pink) within two distinct bundles or behavioural motifs. (right) In contrast, the sustained reversal state (in brown) is represented by a single intersection point on the right side.

behaviour can be modeled at different levels of granularity. In the present data set, the worms' behaviour is described in terms of high-level behavioural patterns, e.g., forward and reversal movements. Alternatively, one could analyze the angular positions and velocities of the various segments of the worms' bodies, resulting in a more fine-grained representation. Both fine-grained and coarse-grained models hold value in specific contexts. However, it is crucial to maintain consistency within a model's state space to describe the dynamics accurately. If we utilise a model to understand fine-grained elements but only have access to coarse-grained information lacking essential details, the resulting model will be incomplete or inconsistent. Here, we demonstrate how BundDLe-Net adeptly handles coarse-graining of data while still preserving the crucial distinctions between states that are instrumental in explaining the overall dynamics.

We present the discovery of two distinct behavioural states with identical labels based on BundDLe-Net's neuronal embedding concerning the given set of behaviours. Consider branch $\boldsymbol{C_2}$ (● → ● → ● → ●) and $\boldsymbol{C_5}$ (● → ● → ● → ●) of the trajectory in Figure 6. The *slowing* behaviour (in pink) occurs in both these branches, i.e., they are represented distinctly in the latent space and are not fused together even though they have been assigned the same behavioural label. Branch $\boldsymbol{C_2}$ has a much shorter *slowing* segment than branch $\boldsymbol{C_5}$. We name the new behavioural states corresponding to $\boldsymbol{C_2}$ and $\boldsymbol{C_5}$ as *slowing 1* and *slowing 2*, respectively. These different types of slowing movements are embedded in distinct behavioural motifs since they differ in their neuronal realization and their relevance for the model dynamics, i.e., one would predict different future trajectories depending on whether the state is *slowing 1* or *slowing 2*. We note that this is not the case for other the behavioural states, e.g., the sustained reversal (in brown) for which all trajectories form one coherent bundle in the embedding. This implies that in the behavioural state of a sustained reversal BundDLe-Net found no information at the neuronal level to predict whether a dorsal or ventral turn is more likely to occur next. In summary, BundDLe-Net can maintain distinct representations or fuse trajectories depending on whether dynamical information about future behaviours is present. Accordingly, if provided with a set of behaviours that are not consistent or complete for the construction of a full dynamical model, BundDLe-Net can discover extra distinctions or *states* that complete this set of behaviours, provided this information is present in the neuronal level.

3 Discussion

In this work, we have presented a generic architecture to learning consistent state representations from neuronal data based on a few simple but vital principles. It is noteworthy that our architecture is similar to some of those used in the field of contrastive learning on image data [6, 10] but with an added behavioural context. Having demonstrated BundDLe-Net on *C. elegans* data we see that it outperforms other standard algorithms. This work could be useful in gaining automated insights from neuroscience data. There is also potential in extending this work in the direction of decision making based on the bifurcation points in the learned representations. Finally, this work is a small step in creating causal cognitive models with a neuronal basis. Further work would involve developing a causal framework for such an algorithm.

We started out with a dataset comprised of non-Markovian neuronal activity and non-Markovian behaviour. However, through the application of BundDLe-Net, we obtained an embedding that facilitated the identification of distinct bundles. Each bundle corresponds to a deterministic behavioural motif, while transitions between motifs maintain a probabilistic nature. Consequently, by employing BundDLe-Net, we were able to transform the non-Markovian dynamics at the neuronal and behavioural level into a Markovian dynamics between the motifs $(\boldsymbol{C_1}, ..., \boldsymbol{C_5})$.

In contrast to previous work that typically learns causal features on *iid* data with respect to a single target variable [13], our method is applicable on time-series data. The architecture bears some similarity to some algortihms used in the field of reinforcement learning [10, 22]. However, our goal is to learn macrovariables that preserve information about its own dynamics *and* the behaviour, while discarding any irrelevant information. A simple autoencoder framework would not be adequate for this, since it is based on state-reconstruction and would try to preserve details that are irrelevant to behavioural dynamics [23].

In the neuroscience setting, we assume that the behaviour is always an effect of the neuronal activity, i.e. $B_t = g(X_t, \epsilon_B)$, and not vice-versa. Making the assumption allows the macroscopic level to be causal with respect to the behaviour as well. Subsequent work could extend this framework to scenarios involving feedback loops where the behaviour can further act as a stimulus and induces neuronal activity.

4 Methods

In this section, we first provide further information on the theoretical principles that motivate BundDLe-Net. Subsequently, we elaborate on the architectural framework that arises from these principles. We then proceed to provide a comprehensive overview of BundDLe-Net's implementation, encompassing the learning modules and the details of the training process. Finally, we present the competing methods that serve as benchmarks for evaluating the performance of BundDLe-Net.

4.1 Theoretical principle

BundDLe-Net employs a fundamental theoretical principle to embed neuronal data with respect to a given set of behaviours. The core idea is to ensure that the resulting embedding Y contains all information about the dynamics and behaviour that is present at the neuronal-level X. To elucidate this concept, consider the diagram in Figure 7, where T_X denotes a transition model at the X level. For illustrative purposes, we presently assume that the X level is Markov, but

will later relax this assumption. The embedding Y is obtained by applying a function τ on the X level. Generally, the resulting transition model at the Y level may not be Markov, implying that Y_t might not fully capture the information about Y_{t+1} present in the system, either at the X level and/or in the past states Y_{t-n}, where $n \in \mathbb{Z}^+$. Such an embedding would be of limited use since one might need to refer back to the X level to answer certain questions about the Y level.

$$\begin{tikzcd} X_t \arrow[r, "\mathbf{T}_X"] \arrow[d, dashed, "\tau"] & X_{t+1} \arrow[d, dashed, "\tau"] \\ Y_t \arrow[r, "\mathbf{T}_Y"] & Y_{t+1} \end{tikzcd}$$

Figure 7: Commutativity diagram where the X and Y level signify the neuronal and latent space dynamics respectively.

To ensure a more comprehensive and self-contained embedding, we aim for Y to be Markov and independent of the X level. This requires the diagram (Figure 7) to commute, i..e. it should not make a difference if we first time-evolve and then transform with τ, or the other way round. Put in terms of conditional independence, our requirement takes the form $Y_{t+1} \perp X_t | Y_t$, meaning that knowledge of X_t provides no additional information about Y_{t+1} beyond what is already known from Y_t. In this way, the dynamics at the Y level are self-contained and *sealed-off* from the details at X level. This is what makes our embedding so useful and interpretable: our embedding has all the relevant information from the X level, enabling it to be viewed as a distinct and meaningful dynamical process in its own right.

Non-Markovian neuronal dynamics To handle non-Markov neuronal dynamics at X, we consider time windows that include the previous n time steps, i.e., $(X_t, ..., X_{t-n})$ as input to our model. By choosing a large enough value for n, we can ensure that the resulting process becomes Markov [20], allowing us to model it in the same way as described above. Note that while earlier we were mapping a single time slice to a point in latent space, now we are mapping an entire time window of length n to a single point in latent space. Such a transformation does not merely coarse-grain over the neuronal or *spatial* level of granularity but also over the *temporal* domain of patterns.

Learning meaningful embeddings While the requirement of a Markov embedding may be very useful in terms of elegance and interpretability, it is not sufficient to ensure meaningful embeddings. For example, consider a transformation τ that uniformly maps every neuronal state to a constant. In this scenario, the resultant process would exhibit Markov dynamics as a single-state process. However, such an embedding fails to yield any meaningful insights regarding the underlying dynamics or behaviour. Remarkably, for BundDLe-Net, such a process would yield a perfect $\mathcal{L}_{\text{Markov}}$ loss, irrespective of the input data.

An additional requirement must be imposed to avoid such *trivial* embeddings. We demand that the behaviour B can be decoded from the embedding, thereby preventing the transformation from reducing everything to a mere constant. By upholding this crucial condition, we preserve the behavioural intricacies that render the embedding purposeful and informative, aligning with the ideals espoused by Krakauer et al. [11].

4.2 BundDLe-Net architecture

Here, we explain how the BundDLe-Net's architecture in Figure 1 arises from the commutativity diagram of Figure 7. The upper and lower arms in the architecture correspond to the possible paths from X_t to Y_{t+1} in the commutativity diagram. The lower arm in the architecture involves first coarse-graining X_t, followed by implementing a transition model on the Y-level. In practice, the transition model outputs ΔY_t from which Y_{t+1} is estimated as $Y_t + \Delta Y_t$. Since the transition model T_Y outputs Y_{t+1} with only Y_t as input, the Y-level is first-order Markov by construction. The upper arm of BundDLe-Net coarse-grains the time-evolved X_{t+1}[5]. Both arms result in estimates of Y_{t+1} which we distinguish by upper indices Y^L_{t+1} and Y^U_{t+1}. We add a mean-squared error term to our loss function $\mathcal{L}_{\text{Markov}}$ that forces Y^L_{t+1} and Y^U_{t+1} to be equalbehaviourd, thus ensuring that our requirement of commutativity in Figure 7 is satisfied,

$$\mathcal{L}_{\text{Markov}}(Y^U_{t+1}, Y^L_{t+1}) = \left\| Y^U_{t+1} - Y^L_{t+1} \right\|^2.$$

The estimated Y_{t+1} is then passed through a predictor layer which learns to output the behaviour B_{t+1} given Y_{t+1}. Correspondingly, we add a term $\mathcal{L}_{\text{Behaviour}}$ to our loss function which forces the predicted behaviour to match the true behaviour. This ensures that Y_t contains the same amount of information about B_t as X_t. Here, we use the cross-entropy loss where $B^{(j)}_{t+1}$ represents the j-th component of a one-hot encoded label vector of B_{t+1}, and $\tilde{B}^j_{t+1}$ is the softmax output of the predicted $\tilde{B}_{t+1}$.

$$\mathcal{L}_{\text{Behaviour}}(B_{t+1}, \tilde{B}_{t+1}) = -\sum_{j=1}^{8} B^{(j)}_{t+1} \log(\tilde{B}^{(j)}_{t+1}).$$

Both terms are weighted by a hyper-parameter γ and the loss function is given as,

$$\mathcal{L} = (1-\gamma)\mathcal{L}_{\text{Markov}} + \gamma \mathcal{L}_{\text{Behaviour}}.$$

All the layers in BundDLe-Net are learned simultaneously, and both loss terms ensure that the learned τ and T_Y preserve information about the behavioural dynamics. A open-source Python implementation of the BundDLe-Net architecture is available at https://github.com/akshey-kumar/BunDLe-Net.

4.3 Learning modules

Architecture parameters The τ layer (encoder) of our network consists of a series of ReLU layers [1], followed by a normalization layer. An encoder of identical architecture is used later in the autoregressor-autoencoder (ArAe) model to facilitate comparison across models. For the *predictor* and T_Y layer, we use a single dense layer each. In the case of our dataset, this sufficed to achieve good performance. For other data sets, more complex layers may be required. For T_Y, we also add a normalization layer so that the output remains in the scaling of the latent space learned by τ. The details of the individual layers are provided in the Python code in Appendix A.

[5]Since we have time-series data, we need not learn T_X of the commutativity diagram,but simply feed X_{t+1} directly into the network.

Gaussian noise against overfitting To safeguard against overfitting of the model, we introduce Gaussian white noise in the latent space by incorporating it in the τ layer. The Gaussian white noise serves as a means of regularisation which makes the model robust to overfitting. Overfitting becomes a concern when the model learns a discontinuous, bijective mapping from a higher-dimensional space to a lower-dimensional one. Such a mapping fails to capture the essence of a true coarse-graining, as the resulting embedding merely replicates all the information present at the neuronal level. To avoid this, we aim to learn a continuous embedding of the time-series, where the trajectory changes smoothly from one data point to the next. In order to learn a continuous embedding, we introduce a small amount of Gaussian noise to the final layer of τ. This perturbation encourages continuous embeddings to be learnt, since they would be more robust to the added noise. In this way we ensure that bijective mappings are avoided and our model truly coarse-grains the data by discarding irrelevant information.

Latent space dimensionality We choose the dimensionality of the Y-level to be three. This is because in 3-D, we can connect any finite number of points without the edges crossing each other. This allows for embeddings of neuronal activity in the form of trajectories with nodes and edges that do not intersect. This might not always be possible in 2-D, where one can have a constellation of data points that cannot be connected without crossings. Three dimensions, however are sufficient to ensure that one can always draw edges (curved, if necessary) between any configuration of points without intersections.

Intersection points are undesirable for the embedding of a dynamical process due to the ambiguity they introduce. A meaningful embedding should exhibit smooth trajectories without self-intersections. When intersection points occur, the future path becomes uncertain, leading to reduced predictability and interpretability. Avoiding such intersections enhances the interpretability of the embedded data and allows an enhanced prediction of future dynamics.

Model validation / parameter tuning To determine the optimal parameters for the model, including the number and types of layers, we use a held-out validation set on Worm-1. The neuronal and behavioural data of Worm-1 is partitioned into seven folds along the time axis, and one fold is randomly selected as the validation set from the time-ordered dataset. The remaining data form the training set. By choosing an entire fold in the data as a validation set, we ensure that the model performs as well on unseen data. This would not be the case if we created our validation set by *iid* (independent and identically distributed) sampling due to high time-correlations in the time-series. After selecting the optimal model parameters through validation on Worm-1, we train models with the same parameters on the other worms. Since we only use Worm-1 for parameter tuning, if the model performs well on other worms, we can be confident that its success is not due to overfitting.

Training details Since the neuronal data was found to be non-Markovian[6], we use time-windows of length 15 as input to BundDLe-Net. Reducing the window length decreased model performance, while increasing it further had no significant effect. Training was performed with the ADAM optimiser [9] with a learning rate of 0.001 and batch size of 100. The γ parameter of BundDLe-Net was chosen to be 0.9 to ensure that $\mathcal{L}_{\text{Markov}}$ and $\mathcal{L}_{\text{Behaviour}}$ are roughly the same order of magnitude during training (see Figure 8). We trained BundDLe-Net until the losses converged.

[6]We tested for non-Markovianity using an autoregressor model and found that including multiple time steps from the past boosted the prediction performance of the model.

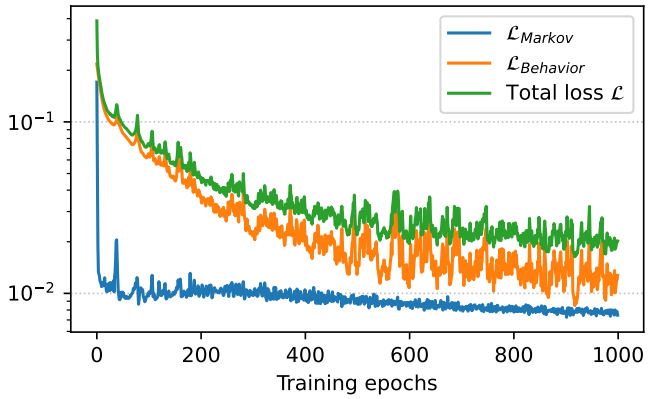

Figure 8: Markov and behavioural loss during training of BundDLe-Net on a log plot

4.4 Description of competing methods

Here, we describe the other commonly-used neuronal manifold learning algorithms used in the comparison. All models are used to project the *C. elegans* data to a three-dimensional space for purposes of fair comparison. A full implementation of the various models, training process, and evaluation procedures can be found at https://github.com/akshey-kumar/comparison-algorithms/tree/main.

PCA Principal component analysis [17] has been applied to neuronal datasets to enable visualization and interpretation of the data. It is a linear transformation that aligns the data along the directions of maximum variance. Typically, the first three principal components are chosen and plotted in 3-D space [7]. The resulting trajectories can provide a rough perspective of the neuronal dynamics at a high level. Since this is a commonly-used method to coarse-grain data, we use PCA as our first baseline model.

t-SNE t-distributed stochastic neighbour embedding is a popular tool for visualizing high-dimensional data, including neuronal data [5, 14]. It is essentially a non-linear dimensionality reduction method that tries to preserve distances between the data points.

Autoencoder Arguably, autoencoders (or some variant thereof) are currently one the most predominant method for learning low-dimensional representations of data [21, 22]. Typically, an autoencoder learns a representation by attempting to reconstruct the training data using an ANN composed of an encoder and decoder [12]. Here, we consider the deterministic vanilla autoencoder with a deep encoder and decoder. The depth of the layers, number of neurons, and other training-related hyperparameters were tuned to obtain reasonably optimal performance.

Autoregressor-autoencoder (ArAe) An autoregressor is generally used on time-series data to predict the future state based on the past. Here we implemented an autoregressor with an ANN with an autoencoder-like architecture[7] and refer to it as ArAe. Such architectures have been used before to learn low-dimensional representations of time-series data [3, 21]. We implement our ArAe as an ANN with a deep encoder and decoder that tries to predict X_{t+1} given X_t as input with Y_t as the latent space as seen in Figure 9.

CEBRA CEBRA [18] is a state-of-the-art neuronal manifold technique. It uses contrastive learning to optimise the encoding of data by maximizing the similarity between related samples and minimizing the similarity between unrelated samples. The algorithm employs neural network encoders and a similarity measure to optimise the embeddings based on user-defined or time-only labels. In our experiments, we used CEBRA-hybrid, which takes both behaviour and time dynamics into account for the embedding.

References

[1] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[2] Misha B Ahrens, Michael B Orger, Drew N Robson, Jennifer M Li, and Philipp J Keller. Whole-brain functional imaging at cellular resolution using light-sheet microscopy. *Nature Methods*, 10(5):413–420, 2013.

[3] Boyuan Chen, Kuang Huang, Sunand Raghupathi, Ishaan Chandratreya, Qiang Du, and Hod Lipson. Automated discovery of fundamental variables hidden in experimental data. *Nature Computational Science*, 2(7):433–442, July 2022. doi: 10.1038/s43588-022-00281-6. URL https://doi.org/10.1038/s43588-022-00281-6.

[4] Mark M Churchland, John P Cunningham, Matthew T Kaufman, Justin D Foster, Paul Nuyujukian, Stephen I Ryu, and Krishna V Shenoy. Neural population dynamics during reaching. *Nature*, 487(7405):51–56, 2012.

[5] George Dimitriadis, Joana P. Neto, and Adam R. Kampff. t-SNE visualization of large-scale neural recordings. *Neural Computation*, 30(7):1750–1774, July 2018. doi: 10.1162/neco_a_01097. URL https://doi.org/10.1162/neco_a_01097.

[6] Bernd Illing, Jean Ventura, Guillaume Bellec, and Wulfram Gerstner. Local plasticity rules can learn deep representations using self-supervised contrastive predictions. In *Advances in Neural Information Processing Systems*, volume 34, pages 30365–30379. Curran Associates, Inc., 2021.

[7] Saul Kato, Harris S. Kaplan, Tina Schrödel, Susanne Skora, Theodore H. Lindsay, Eviatar Yemini, Shawn Lockery, and Manuel Zimmer. Global brain dynamics embed the motor command sequence of caenorhabditis elegans. *Cell*, 163(3):656–669, October 2015. doi: 10.1016/j.cell.2015.09.034. URL https://doi.org/10.1016/j.cell.2015.09.034.

[8] Jason ND Kerr and Winfried Denk. Imaging in vivo: watching the brain in action. *Nature Reviews Neuroscience*, 9(3):195–205, 2008.

[7] We use an autoencoder architecture since an autoregressor, in general, need not map the data to a low-dimensional space. Hence, we use an encoder to obtain an embedding that can be compared with the other methods

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. URL https://arxiv.org/abs/1412.6980.

[10] Thomas N. Kipf, Elise van der Pol, and Max Welling. Contrastive learning of structured world models. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=H1gax6VtDB.

[11] John W Krakauer, Asif A Ghazanfar, Alex Gomez-Marin, Malcolm A MacIver, and David Poeppel. Neuroscience needs behavior: correcting a reductionist bias. *Neuron*, 93(3): 480–490, 2017.

[12] Mark A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991. doi: https://doi.org/10.1002/aic.690370209. URL https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.690370209.

[13] Krzysztof Chalupka and Frederick Eberhardt and Pietro Perona. Causal feature learning: an overview. *Behaviormetrika*, 44(1):137–164, 2016. doi: 10.1007/s41237-016-0008-2. URL https://doi.org/10.1007/s41237-016-0008-2.

[14] Jinke Liu and Martin Vinck. Improved visualization of high-dimensional data using the distance-of-distance transformation. *PLOS Computational Biology*, 18(12):e1010764, December 2022. doi: 10.1371/journal.pcbi.1010764. URL https://doi.org/10.1371/journal.pcbi.1010764.

[15] Rufus Mitchell-Heggs, Seigfred Prado, Giuseppe P Gava, Mary Ann Go, and Simon R Schultz. Neural manifold analysis of brain circuit dynamics in health and disease. *Journal of Computational Neuroscience*, 51(1):1–21, 2023.

[16] Chethan Pandarinath, Daniel J. O'Shea, Jasmine Collins, Rafal Jozefowicz, Sergey D. Stavisky, Jonathan C. Kao, Eric M. Trautmann, Matthew T. Kaufman, Stephen I. Ryu, Leigh R. Hochberg, Jaimie M. Henderson, Krishna V. Shenoy, L. F. Abbott, and David Sussillo. Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature Methods*, 15(10):805–815, September 2018. doi: 10.1038/s41592-018-0109-9. URL https://doi.org/10.1038/s41592-018-0109-9.

[17] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11): 559–572, 1901. doi: 10.1080/14786440109462720.

[18] Steffen Schneider, Jin Hwa Lee, and Mackenzie Weygandt Mathis. Learnable latent embeddings for joint behavioural and neural analysis. *Nature*, 617(7960):360–368, May 2023. doi: 10.1038/s41586-023-06031-6.

[19] Carsen Stringer, Marius Pachitariu, Nicholas Steinmetz, Matteo Carandini, and Kenneth D Harris. High-dimensional geometry of population responses in visual cortex. *Nature*, 571 (7765):361–365, 2019.

[20] Floris Takens. Detecting strange attractors in turbulence. In David Rand and Lai-Sang Young, editors, *Dynamical Systems and Turbulence*, volume 898 of *Lecture Notes in Mathematics*, pages 366–381. Springer-Verlag, 1981. doi: 10.1007/BFb0091924.

[21] N. Wahlström, Schön T. B., and M. P. Desienroth. From pixels to torques: Policy learning

with deep dynamical models. *Deep Learning Workshop at the 32nd International Conference on Machine Learning ICML 2015*, 2015.

[22] Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/a1afc58c6ca9540d057299ec3016d726-Paper.pdf.

[23] Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=-2FCwDKRREu.

A BundDLe-Net architecture

```python
class BunDLeNet(Model):
    """Behaviour and Dynamical Learning Network (BunDLeNet) model.

    This model represents BunDLe-Net's architecture for deep
    learning and is based on the commutativity diagrams. The
    resulting model preserves information relevant to the
    behavioural dynamics

    Args:
        latent_dim (int): Dimension of the latent space.
    """
    def __init__(self, latent_dim):
        super(BunDLeNet, self).__init__()
        self.latent_dim = latent_dim
        self.tau = tf.keras.Sequential([
            layers.Flatten(),
            layers.Dense(50, activation='relu'),
            layers.Dense(30, activation='relu'),
            layers.Dense(25, activation='relu'),
            layers.Dense(10, activation='relu'),
            layers.Dense(latent_dim, activation='linear'),
            layers.Normalization(axis=-1),
            layers.GaussianNoise(0.05)
        ])
        self.T_Y = tf.keras.Sequential([
            layers.Dense(latent_dim, activation='linear'),
            layers.Normalization(axis=-1),
        ])
        self.predictor = tf.keras.Sequential([
            layers.Dense(8, activation='linear')
        ])

    def call(self, X):
        # Upper arm of commutativity diagram
        Yt1_upper = self.tau(X[:,1])
        Bt1_upper = self.predictor(Yt1_upper)

        # Lower arm of commutativity diagram
        Yt_lower = self.tau(X[:,0])
        Yt1_lower = Yt_lower + self.T_Y(Yt_lower)

        return Yt1_upper, Yt1_lower, Bt1_upper
```

A.1 BundDLe-Net loss function

```
def bccdcc_loss(yt1_upper, yt1_lower, bt1_upper, b_train_1, gamma):
    """Calculate the loss for the BunDLe Net

    Args:
        yt1_upper: Output from the upper arm of the BunDLe Net.
        yt1_lower: Output from the lower arm of the BunDLe Net.
        bt1_upper: Predicted output from the upper arm of the BunDLe Net.
        b_train_1: True output for training.
        gamma (float): Tunable weight for the DCC loss component.

    Returns:
        tuple: A tuple containing the DCC loss, behaviour loss, and total
                                                loss.
    """
    mse = tf.keras.losses.MeanSquaredError()
    scce = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)

    DCC_loss = mse(yt1_upper, yt1_lower)
    behaviour_loss = scce(b_train_1, bt1_upper)
    total_loss = gamma*DCC_loss + (1-gamma)*behaviour_loss
    return gamma*DCC_loss, (1-gamma)*behaviour_loss, total_loss
```

B Other architectures of ANN models

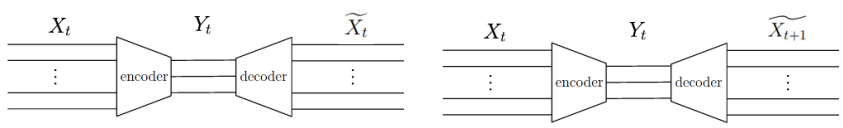

Figure 9: Architecture of Autoencoder and autoencoder-autoregressor (ArAe)

C Learning process

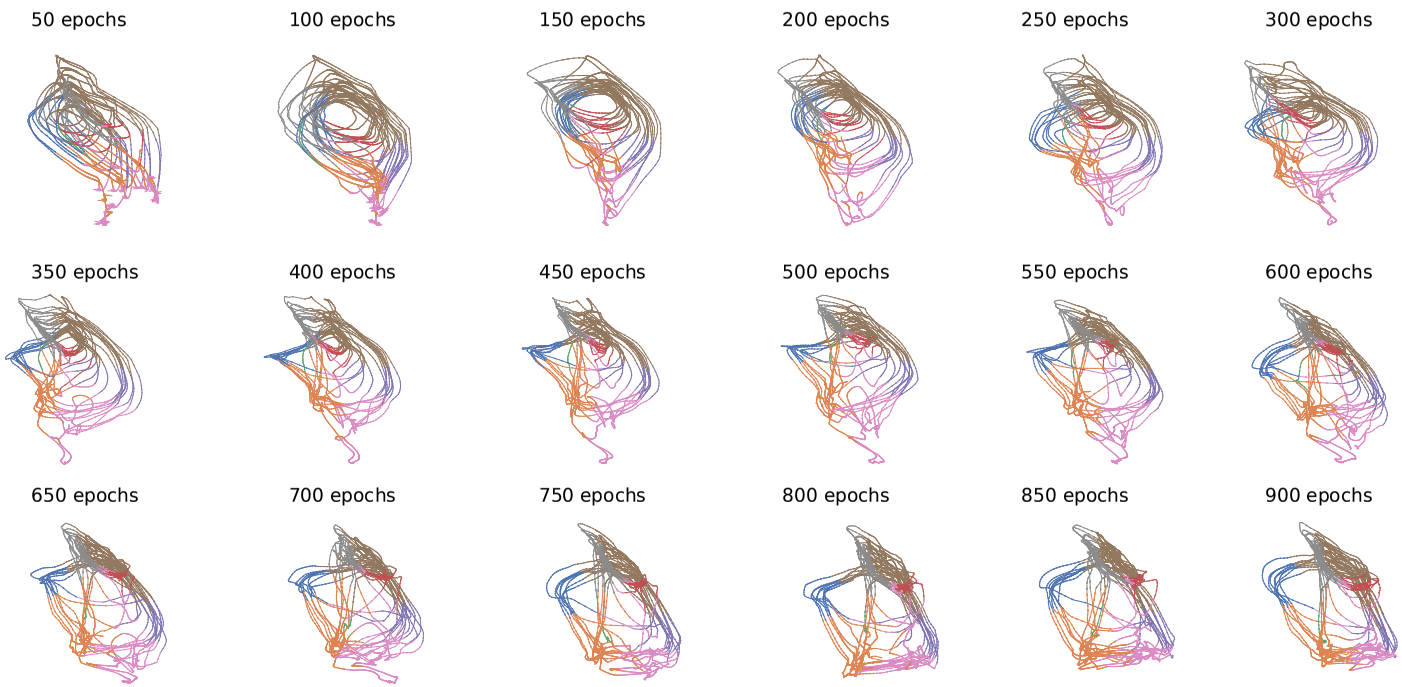

Figure 10: Visualisation of learning process as a function of epochs