Expressive dynamics models with nonlinear injective readouts enable reliable recovery of latent features from neural activity

Christopher Versteeg[1] **Andrew R. Sedler**[1,2] **Jonathan D. McCart**[1,2]
Chethan Pandarinath[1,2]

[1] **Wallace H. Coulter Department of Biomedical Engineering**
Emory University and Georgia Institute of Technology
Atlanta, GA, USA

[2] **Center for Machine Learning**
Georgia Institute of Technology
Atlanta, GA, USA

Editors: Sophia Sanborn, Christian Shewmake, Simone Azeglio, Nina Miolane

Abstract

An emerging framework in neuroscience uses the rules that govern how a neural circuit's state evolves over time to understand the circuit's underlying computation. While these *neural dynamics* cannot be directly measured, new techniques attempt to estimate them by modeling observed neural recordings as a low-dimensional latent dynamical system embedded into a higher-dimensional neural space. How these models represent the readout from latent space to neural space can affect the interpretability of the latent representation – for example, for models with a linear readout could make simple, low-dimensional dynamics unfolding on a non-linear neural manifold appear excessively complex and high-dimensional. Additionally, standard readouts (both linear and non-linear) often lack injectivity, meaning that they don't obligate changes in latent state to directly affect activity in the neural space. During training, non-injective readouts incentivize the model to invent dynamics that misrepresent the underlying system and computation. To address the challenges presented by non-linearity and non-injectivity, we combined a custom readout with a previously developed low-dimensional latent dynamics model to create the Ordinary Differential equations autoencoder with Injective Nonlinear readout (ODIN). We generated a synthetic spiking dataset by non-linearly embedding activity from a low-dimensional dynamical system into higher-D neural activity. We show that, in contrast to alternative models, ODIN is able to recover ground-truth latent activity from these data even when the nature of the system and embedding are unknown. Additionally, we show that ODIN enables the unsupervised recovery of underlying dynamical features (e.g., fixed points) and embedding geometry (e.g., the neural manifold) over alternative models. Overall, ODIN's ability to recover ground-truth latent features with low dimensionality make it a promising method for distilling interpretable dynamics that can explain neural computation.

Keywords: Dynamical systems, Manifold, Modeling, Neural Networks

1. Introduction

When artificial recurrent neural networks are trained to perform tasks, the rules that govern how activity evolves over time (i.e., the network dynamics) provide insight into the underlying computation [Sussillo and Barak (2013); Remington et al. (2018)]. As artificial networks are conceptually similar to biological neural circuits, these dynamical analyses could also help to understand how the brain performs complex sensory, cognitive, and motor computations [Vyas et al. (2020); Shenoy et al. (2013)]. However, without direct access to dynamics of biological circuits, we are forced to estimate them from observed neural activity.

Fortunately, advances in recording technology have dramatically increased the number of neurons that can be simultaneously recorded, supporting novel population-level analyses of neural activity [Stevenson and Kording (2011); Steinmetz et al. (2021); Demas et al. (2021)]. In these datasets, the activity of hundreds or thousands of neurons often falls on a relatively low-dimensional latent subspace (i.e., a neural manifold) [Gao and Ganguli (2015); Gallego et al. (2017)], orders-of-magnitude smaller than the total number of neurons. Neural activity in these latent spaces evolves according to consistent sets of rules (i.e., latent dynamics) [Duncker and Sahani (2021); Shenoy et al. (2013)], which, assuming no external inputs, can be expressed mathematically as:

$$\mathbf{z}_{t+1} = \mathbf{z}_t + f(\mathbf{z}_t) \tag{1}$$

$$\mathbf{y}_t = \exp g(\mathbf{z}_t) \tag{2}$$

$$\mathbf{x}_t \sim \text{Poisson}(\mathbf{y}_t) \tag{3}$$

where $\mathbf{z}_t \in \mathbb{R}^D$ represents the latent state at time t, $f(\cdot) : \mathbb{R}^D \to \mathbb{R}^D$ is the vector field governing the dynamical system, $\mathbf{y}_t \in \mathbb{R}^N$ denotes the firing rates of the N neurons, $g(\cdot) : \mathbb{R}^D \to \mathbb{R}^N$ maps latent activity into log-firing rates, and $\mathbf{x}_t \in \mathbb{R}^N$ denotes the observed spike counts at time t, assuming the spiking activity follows a Poisson distribution with time-varying rates given at each moment t by $\mathbf{y}_t$.

Unfortunately, any given observed neural activity can be equivalently described by many combinations of dynamics f and embeddings g, which makes the search for a unique latent system futile. However, versions of a latent system's dynamics f and embedding g that are less complex and use fewer latent dimensions can be easier to interpret than alternative representations that are more complex and/or higher-dimensional.

Neural population dynamics models (NPDMs) are a popular approach to estimate neural dynamics [Sussillo et al. (2016); Schimel et al. (2021); Sedler et al. (2023)]. NPDMs model neural activity as a latent dynamical system embedded into neural activity. We refer to the components of an NPDM that learn the dynamics and embedding as the generator $\hat{f}$ and the readout $\hat{g}$, respectively. The generator and readout are trained jointly to infer firing rates $\hat{\mathbf{y}}$ that maximize the likelihood of the observed neural activity $\mathbf{x}$.

Using NPDMs to estimate the underlying dynamics and embedding implicitly assumes that good reconstruction performance (i.e., $\hat{\mathbf{x}} \approx \mathbf{x}$) implies interpretable estimates of the underlying system (i.e., $\hat{\mathbf{z}} \approx \mathbf{z}$, $\hat{f} \approx f$, $\hat{g} \approx g$). However, recent work has shown that when the state dimensionality of the generator $\hat{D}$ is larger than a system's latent dimensionality D, high reconstruction performance may actually correspond to estimates of the latent system that are overly complex or misleading and therefore harder to interpret [Sedler et al. (2023)]. At present, reconstruction performance is seemingly an unreliable indicator for the interpretability of the learned dynamics.

This vulnerability to learning overly complex latent features might emerge from the fact that, without constraints on the readout $\hat{g}$, changes in the latent state are not obligated to affect predicted neural activity. Thus, NPDMs can be rewarded for inventing latent activity that boosts reconstruction performance, even if that latent activity has no direct correspondence to neural activity. A potential solution is to make $\hat{g}$ injective, which would obligate all latent activity to affect neural reconstruction. This would penalize any latent activity that is not reflected in the observed neural activity, thereby putting pressure on the generator $\hat{f}$ and readout $\hat{g}$ to learn a more interpretable (i.e., simpler and lower dimensional) representation of the underlying system.

In addition, most previously used readouts $\hat{g}$ were not expressive enough to model diverse mappings from latent space to neural space, assuming the embedding g to be a relatively simple (often linear) transformation (though there are exceptions [Gao et al. (2016); Zhao and Park (2020))]. Capturing nonlinear embeddings is important because neural activity often lives on a lower-dimensional neural manifold that is nonlinearly embedded into the higher-dimensional neural space [Jazayeri and Ostojic (2021)]. Therefore, assumptions of linearity are likely to prevent NPDMs from capturing dynamics in their simplest and lowest-dimensional form, making them less interpretable than the latent features learned by NPDMs that can approximate these nonlinearities.

To address these challenges, we propose a novel architecture called Ordinary Differential equation autoencoder with Injective Nonlinear readout (ODIN), which implements $\hat{f}$ using a Neural ODE (NODE [Chen et al. (2019)]) and $\hat{g}$ using a network inspired by invertible ResNets [Dinh et al. (2014); Chen et al. (2019); Behrmann et al. (2019)]. ODIN approximates an injective nonlinear mapping between latent states and neural activity, obligating all latent state variance to appear in the predicted neural activity and penalizing the model for using excessively complex or high-dimensional dynamics to model the underlying system. On synthetic data, ODIN learns representations of the latent system that are more interpretable, with simpler and lower-dimensional latent activity and dynamical features (e.g., fixed points) than alternative readouts. ODIN's interpretability is also more robust to overestimates of latent dimensionality and can recover the nonlinear embedding of synthetic data that evolves on a simulated neural manifold. In summary, ODIN estimates interpretable latent features from synthetic data, making it a promising tool for understanding how the brain performs computation.

2. Related Work

Many previous models have attempted to understand neural activity through the lens of neural dynamics. Early efforts limited model complexity by constraining both $\hat{f}$ and $\hat{g}$ to be linear [Macke et al. (2011); Archer et al. (2015)]. While these models were relatively straightforward to analyze, they often failed to adequately explain neural activity patterns.

Other approaches increased the expressiveness of the modeled dynamics $\hat{f}$. RNNs can learn to approximate complex nonlinear dynamics, and have been shown to substantially outperform linear dynamics models in reconstructing neural activity [Pei et al. (2022)], but their ability to model complex dynamics relies on having a high-dimensional latent state. In contrast, NODEs can model arbitrarily complex dynamics of embedded dynamical systems at the dimensionality of the system [Kim et al. (2021); Sedler et al. (2023)]. In contrast to our approach, previous NODE-based models used a linear readout $\hat{g}$. This can make the accuracy of estimated latent activity vulnerable to overestimates of the latent dimensionality (i.e., when $\hat{D} > D$) and/or fail to capture potential nonlinearities in the embedding g (i.e., the neural manifold).

Early efforts to allow greater flexibility in $\hat{g}$ preserved linearity in $\hat{f}$, using feed-forward neural networks to nonlinearly embed linear dynamical systems in high-dimensional neural firing rates [Gao et al. (2016)]. More recently, models have used Gaussian processes to approximate nonlinear mappings from latent state to neural firing with tuning curves Wu et al. (2017). Other models have combined nonlinear dynamics models and nonlinear embeddings for applications in behavioral tracking [Johnson et al. (2017)] and neural reconstruction [Zhao and Park (2020)]. While nonlinear, these models lacked injectivity in their mapping from latent activity to neural activity.

Some latent variable models have used invertible networks to approximate the mapping from the latent space to neural activity [Zhou and Wei (2020)] or for generative models of visual cortex activity [Bashiri et al. (2021)]. However, these previous approaches did not have explicit dynamics models, making our study, to our knowledge, the first to test whether injective readouts improve the interpretability of neural population dynamics models.

3. Methods

3.1. Synthetic Neural Data

To determine whether different models can distill an interpretable latent system from observed population activity, we first used reference datasets that were generated using simple ground-truth dynamics f and embedding g. Our synthetic test cases emulate the empirical properties of neural systems, specifically low-dimensional latent dynamics observed through noisy spiking activity [Sussillo et al. (2016); Smith et al. (2021); Jensen et al. (2021)]. We sampled latent trajectories from the Arneodo system (f, $D = 3$) and nonlinearly embedded these trajectories into neural activity via an embedding g (i.e., a neural manifold). We consider models that can recover the dynamics f and embedding g used to generate these data as providing an interpretable description of the latent system and its relation to the neural activity. Additional detail on data generation, models, and metrics can be found in the Supplementary Material.

We generated activations for N neurons ($N = 12$) by projecting the simulated latent trajectories $\mathbf{Z}$ through a $3 \times N$ matrix whose columns were random encoding vectors with elements sampled from a uniform distribution $U[-0.5, 0.5]$ (Fig. 1A, left). We standardized these activations to have zero mean and unit variance and applied a different scaled sigmoid function to each neuron, yielding a matrix of non-negative time-varying firing rates $\mathbf{Y}$. The scaling of each sigmoid function was evenly spaced on a logarithmic scale between $10^{0.2}$ and 10. This process created a diverse set of activation functions ranging from quasi-linear to nearly step-function-like (Fig. 1A, Activation Functions).

We simulated spiking activity $\mathbf{X}$ by sampling from inhomogeneous Poisson processes with time-varying rate parameters equal to the firing rates $\mathbf{Y}$ of the simulated neurons (Fig. 1A, right). We randomly split 70-point segments of these trials into training and validation datasets (training and validation proportions were 0.8 and 0.2, respectively).

3.2. Model Architecture

We used three sequential autoencoder (SAE) variants in this study, with the main difference being the choice of readout, $\hat{g}(\cdot)$. In brief, a sequence of binned spike counts $\mathbf{x}_{1:T}$ was passed through a bidirectional GRU encoder, whose final hidden states were converted to an initial

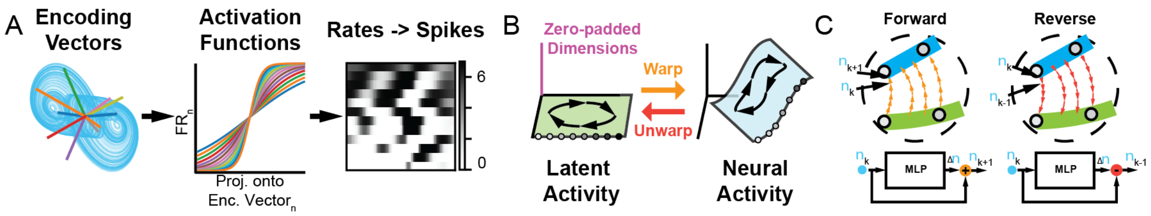

Figure 1: A) Synthetic data generation (left to right). Trajectories projected onto encoding vectors produce activations. Sigmoidal nonlinearity converts activations into firing rates. B) Zero-padded latent dynamics (green) are reversibly warped to higher-dimensional neural activity (blue). C) Flow applies a sequence of K small updates (parameterized by MLP). Reverse pass maps from neural to latent spaces via serial subtraction of updates from same MLP.

condition $\hat{\mathbf{z}}_0$ via a mapping $\phi(\cdot)$. A modified NODE generator unrolled the initial condition into time-varying latent states $\hat{\mathbf{z}}_{1:T}$. These were subsequently mapped to inferred rates via the readout $\hat{g}(\cdot) \in \{\text{Linear}, \text{MLP}, \text{Flow}\}$. All models were trained for a fixed number of epochs to infer firing rates $\hat{\mathbf{y}}_{1:T}$ that minimize the negative Poisson log-likelihood of the observed spikes $\mathbf{x}_{1:T}$. The computations each SAE performs are summarized by the following set of equations:

$$\mathbf{h}_T = \left[\mathbf{h}_{fwd} \middle| \mathbf{h}_{bwd}\right] = \text{BiGRU}(\mathbf{x}_{1:T}) \tag{4}$$

$$\hat{\mathbf{z}}_0 = \phi(\mathbf{h}_T) \tag{5}$$

$$\hat{\mathbf{z}}_{t+1} = \hat{\mathbf{z}}_t + \alpha \cdot \text{MLP}(\hat{\mathbf{z}}_t) \tag{6}$$

$$\hat{\mathbf{y}}_t = \exp \hat{g}(\hat{\mathbf{z}}_t) \tag{7}$$

For models with Linear and MLP readouts, $\phi(\cdot)$ was a linear map to $\mathbb{R}^{\hat{D}}$. For models with Flow readouts, $\phi(\cdot)$ was a linear map to $\mathbb{R}^N$ followed by the reverse pass of the Flow (see Section 3.2.1). We unrolled the NODE using Euler's method with a fixed step size equal to the bin width and trained using standard backpropagation for efficiency. A scaling factor ($\alpha = 0.1$) was applied to the output of the NODE's MLP to stabilize the dynamics during early training. Readouts were implemented as either a single linear layer (Linear), an MLP with two 150-unit ReLU hidden layers (MLP), or a Flow readout (Flow) which contains an MLP with two 150-unit ReLU hidden layers. We refer to these three models as Linear-NODE, MLP-NODE, and ODIN, respectively.

3.2.1. Flow Readout

The Flow readout resembles a simplified invertible ResNet [Behrmann et al. (2019)]. Flow learns a vector field that can reversibly transform data between latent and neural representations (Figure 1B). The Flow readout has three steps: first, we increase the dimensionality of the latent activity $\mathbf{z}_t$ to match that of the neural activity by padding the latent state with zeros. This corresponds to an initial estimate of the log-firing rates, $\log \hat{\mathbf{y}}_{t,0}$. Note that zero-padding makes our mapping injective rather than fully invertible (see [Zhou and Wei (2020)]). The Flow network then uses an MLP to iteratively refine $\log \hat{\mathbf{y}}_{t,k}$ over K steps ($K = 20$) after which we apply an exponential to produce the final firing rate predictions, $\hat{\mathbf{y}}_t$. A scaling factor ($\beta = 0.1$) was applied to the output of the Flow's MLP, which prevents the embedding from becoming unstable during the early training period. This process is summarized by the equations below:

$$\log \hat{\mathbf{y}}_{t,0} = [\hat{\mathbf{z}}_t|\mathbf{0}]^T \tag{8}$$

$$\log \hat{\mathbf{y}}_{t,k+1} = \log \hat{\mathbf{y}}_{t,k} + \beta \cdot \text{MLP}(\log \hat{\mathbf{y}}_{t,k}) \tag{9}$$

$$\hat{g}\left(\hat{\mathbf{z}}_t\right) = \log \hat{\mathbf{y}}_{t,K} = \log \hat{\mathbf{y}}_t \tag{10}$$

We also use a reverse pass of the Flow to transform the output of the encoders to initial conditions in the latent space via $\phi(\cdot)$, approximating the inverse function $\hat{g}^{-1}$. Our method subtracts the output of the MLP from the state rather than adding it as in the forward mode (Fig 1C), a simplified version of the fixed-point iteration procedure described in [Behrmann et al. (2019)]. We then trim the excess dimensions to recover $\hat{z} \in \mathbb{R}^{\hat{D}}$ (in effect, removing the zero-padding dimensions).

$$\log \hat{\mathbf{y}}_{t,k-1} = \log \hat{\mathbf{y}}_{t,k} - \beta \cdot \text{MLP}(\log \hat{\mathbf{y}}_{t,k}) \tag{11}$$

$$\hat{g}^{-1}\left(\log \hat{\mathbf{y}}_t\right) = [\log \hat{y}_{t,0,1}, \ldots, \log \hat{y}_{t,0,\hat{D}}]^T = \hat{\mathbf{z}}_t \tag{12}$$

The Flow mapping is only guaranteed to be injective if changes in the output of the MLP are sufficiently small relative to changes in the input (i.e., Lipschitz constant for the MLP that is strictly less than 1) [Behrmann et al. (2019)]. The model could be made fully injective by restricting the weights of the MLP (e.g., with spectral normalization). In practice, we found that using a moderate number of steps allows Flow to preserve approximate injectivity of the readout at all tested dimensionalities (Supp. Figs. S2), in contrast with MLP and Linear readouts (Fig 2D, Supp. Fig. S3).

3.3. Metrics and characterization of dynamics

We assessed model performance in five domains: 1) reconstruction performance, 2) latent accuracy, 3) dynamical accuracy, 4) embedding accuracy, and 5) injectivity. All metrics were evaluated on validation data. Critically, on biological data without a ground-truth system, only the reconstruction performance and readout injectivity can be assessed, since all the other metrics rely on full observability of the underlying system. Therefore, we need models for which good performance on the observable metrics (reconstruction, injectivity) implies good performance on the unobservable metrics (latent, dynamical, and embedding accuracy). Reconstruction performance was assessed using two key metrics. The first, spike negative log-likelihood (Spike NLL), was defined as the Poisson NLL employed during model training. The second, Rate R^2, was the coefficient of determination between the inferred and true firing rates, averaged across neurons. We used Spike NLL to assess how well the inferred rates explain the spiking activity, while Rate R^2 reflects the model's ability to find the true firing rates. These metrics quantify how well the model captures the embedded system's dynamics (i.e., that $\hat{f}, \hat{g}$ captures the system described by f, g), but give no indication of the interpretability of the learned latent representation (i.e., that the learned $\hat{f}, \hat{g}$ are simple and low-dimensional).

To determine whether a model's inferred latent activity contains features that are not in the simulated latent activity, we used a previously published metric called the State R^2 [Sedler et al. (2023)]. State R^2 is defined as the coefficient of determination (R^2) of a linear regression from simulated latent trajectories $\mathbf{z}$ to the inferred latent trajectories $\hat{\mathbf{z}}$. State R^2

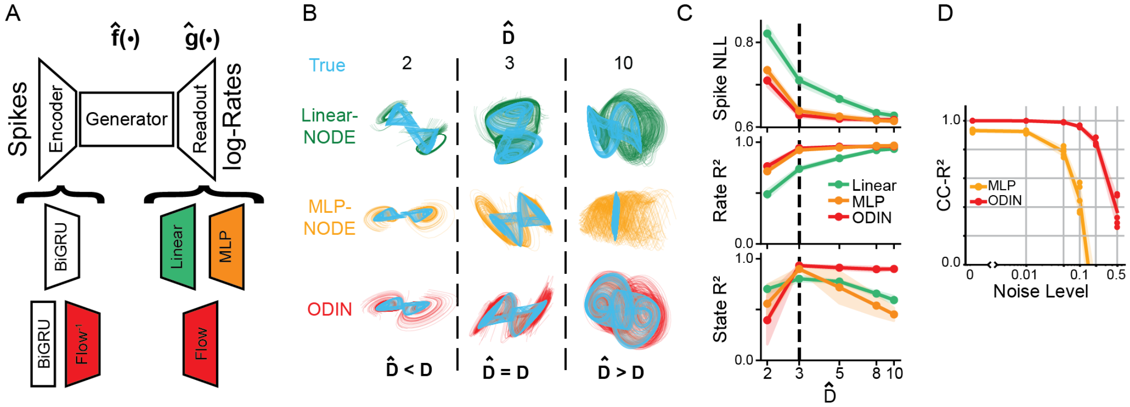

Figure 2: ODIN recovers latent activity and is robust to overestimated $\hat{D}$. A) Model diagram: Linear-NODE (green), MLP-NODE (orange), ODIN (red). B) Inferred latent activity at $\hat{D}$ with true latent activity (cyan). C) Model metrics vs. $\hat{D}$. Shaded areas represent one std. dev. around the mean. Dashed line indicates $\hat{D} = 3$ Top: Spike NLL, Middle: Rate R^2, Bottom: State R^2. D) Cycle-consistency R^2 for ODIN and MLP-NODE as a function of noise corruption.

will be low if the inferred latent trajectories contain features that cannot be explained by an affine transformation of the true latent trajectories. Importantly, State R^2 alone cannot ensure latent accuracy because a model can achieve high State R^2 trivially if the inferred latent activity $\hat{\mathbf{z}}$ is a low-dimensional projection of the simulated activity $\mathbf{z}$. Therefore, only models that have *both* good reconstruction performance (Spike NLL, Rate R^2) and State R^2 can be said to accurately reflect the simulated latent dynamics without extra features that make the model harder to interpret (i.e., $\hat{\mathbf{z}} \approx \mathbf{z}$).

As a direct comparison of the estimated dynamics $\hat{f}$ to the simulated dynamics f, we extracted the fixed-point (FP) structure from our trained models and compared it to the FP structure of the underlying system. We used previously published FP-finding techniques [Golub and Sussillo (2018)] to identify regions of the generator's dynamics where the magnitude of the vector field was close to zero, calling this set of locations the putative FPs. We linearized the dynamics around the FPs and computed the eigenvalues of the Jacobian of $\hat{f}$ to characterize each FP. Capturing FP location and character gives an indication of how closely the estimated dynamics resemble the simulated dynamics (i.e., $\hat{f} \approx f$).

To determine how well our embedding $\hat{g}$ captures the simulated embedding g, we projected the encoding vectors used to generate the synthetic neural activity from the ground-truth system into our model's latent space using the affine transformation from ground-truth to inferred latent activity. We projected inferred latent activity onto each neuron's affine-transformed encoding vector to find the predicted activation of each synthetic neuron. We then related the predicted firing rates of each neuron to its corresponding activations to estimate each neuron's activation function. Because the inferred latent activity is arbitrarily scaled relative to the true latent activity, we fit an affine transformation from the predicted activation function to the ground-truth. The coefficient of determination R^2 quantifies how well our models recovered the synthetic warping applied to each neuron (i.e., $\hat{g} \approx g$).

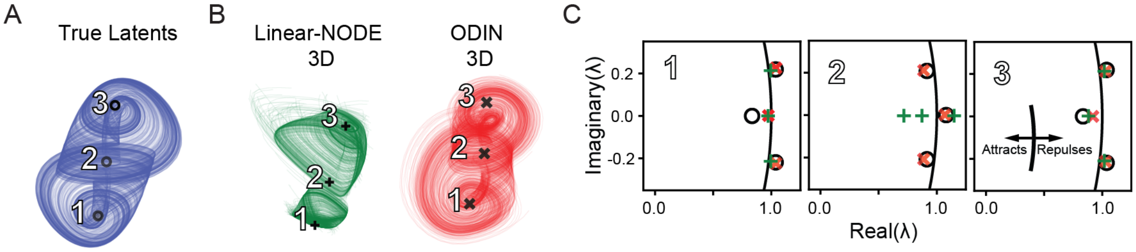

Figure 3: ODIN recovers accurate fixed points. A,B) Representative latent activity and fixed-points from the true (blue, ∘), ODIN (red, ×), and Linear-NODE (green, +) systems. Each fixed point is labeled with reference to C. C) Real vs. imaginary part of the eigenvalues of Jacobian evaluated at each fixed point. Black curve shows boundary between attractive and repulsive behavior (indicated by inset).

We compared the injectivity of the Flow readout to MLP readouts using cycle-consistency. Cycle-consistency quantifies how well the inferred latent activity $\hat{\mathbf{z}}$ can be recovered from the predicted log-firing rates $\log \hat{\mathbf{y}}$.

4. Results

4.1. Accurate latent activity across state dimensionalities with ODIN

As the latent dimensionality D is unknown for biological datasets, we tested the robustness of each model to choices of state dimensionality $\hat{D}$. We trained Linear/MLP -NODE, and ODIN (Fig 2A) to reconstruct synthetic neural activity from the Arneodo system and compared reconstruction performance (i.e. Spike NLL and Rate R^2) and latent recovery (i.e. State R^2). We trained 5 different random seeds for each of the 3 model types and 5 state dimensionalities (75 total models, hyperparameters in Supp. Table 1, representative hyperparameter sweeps in Supp. Fig. S1).

First, we observed that latent activity inferred by Linear-NODE did not closely resemble the simulated latent activity, with all tested dimensionalities performing worse than either ODIN or the MLP-NODE at $\hat{D} = 3$ (Fig 2B,C, mean State $R^2 = 0.70$ for Linear-NODE vs. 0.89, 0.93 for MLP-NODE, ODIN respectively). We also found that Linear-NODE required many more dimensions to reach the peak reconstruction performance (Fig 2C, Rate R^2). These results show that models that are unable to account for nonlinear embeddings learn more complex and higher dimensional dynamics than models with nonlinear readouts.

Next, we found that at the correct dimensionality ($\hat{D} = 3$), ODIN and MLP-NODE had similar performance in both reconstruction and latent recovery. However, as the dimensionality increased beyond the true dimensionality ($\hat{D} > 3$), latent recovery of the MLP-NODE degraded rapidly while ODIN's latent recovery remained high (Fig 2C, as $\hat{D} > 3$).

We then used cycle consistency – how well inputs to a function can be recovered from the function's outputs – to test whether ODIN's relative robustness in latent recovery was associated with its injective readout. We trained a separate MLP to predict inferred latents $\hat{\mathbf{z}}$ from predicted log-firing rates $\log \hat{\mathbf{y}}$ for each 10D MLP-NODE and ODIN model. We found that ODIN's cycle consistency was consistently higher than MLP-NODE (Fig. 2D), in both the noise-free and noise corrupted conditions. As the true latent dimensionality D of biological datasets is unknown, NPDMs with non-injective readouts (like MLPs) may be predisposed to learning misleading latent activity that can be more difficult to interpret.

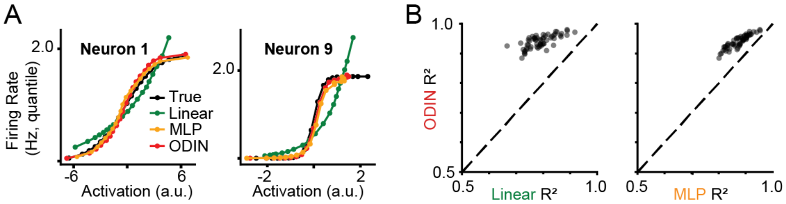

Figure 4: ODIN recovers nonlinear embeddings. A) Inferred activation function for two example neurons, Linear, MLP, ODIN, True = green, orange, red, black, respectively). Predicted firing rate vs. neural activation. B) Comparison of the R^2 values across all neurons for models with $\hat{D} = 3$.

4.2. Recovering fixed point structure with ODIN

A common method for evaluating NPDMs is to compare the character and structure of the inferred fixed points (FPs) to the FPs of the ground-truth system [Sedler et al. (2023)]. FPs provide a concise description of the dynamics in the region around the FP, and can collectively provide a qualitative picture of the overall dynamical landscape. To obtain a set of candidate FPs, we found points in the latent space at which the magnitude of the vector field $\|\hat{f}\|$ is minimized (as in [Sussillo and Barak (2013); Golub and Sussillo (2018)]). We computed the eigenvalues of the Jacobian of $\hat{f}$ at each FP location. The real and imaginary components of these eigenvalues identify each FP as attractive, repulsive, etc.

We found that 3D ODIN models and 3D Linear-NODEs were both able to recover three fixed points that generally matched the location of the three fixed points of the Arneodo system (Fig 3A, B), However, ODIN was also able to capture the eigenspectra of all three FPs (Fig. 3C, red ×), while the Linear-NODE failed to capture the rotational dynamics of the central FP (Fig 3C, middle column). We found that the MLP-NODE was also able to find FPs with similar accuracy to ODIN at 3D. Failing to capture the nonlinear embedding can apparently lead to impoverished estimates of the underlying dynamics $\hat{f}$.

4.3. Recovering simulated activation functions with ODIN

A primary goal of this work is to find models that allow unsupervised recovery of the embedding geometry (i.e., the neural manifold), as these features may provide additional insight about the computations performed by the neural system [Gardner et al. (2021); Jazayeri and Ostojic (2021)]. For this section, we considered a representative model from each readout class with the correct number of latent dimensions ($D = 3$). We derived an estimate of the activation function for each neuron, and compared this estimate to the ground-truth activation function.

We found, as expected, that Linear-NODE was unable to approximate the sigmoidal activation function of individual neurons (Fig 4A, green). On the other hand, both ODIN and MLP-NODE were able to capture activation functions ranging from nearly linear to step function-like in nature (Fig 4A, red, orange). Across all simulated neurons for models with $D = 3$, we found that ODIN more accurately estimated the activation function of individual neurons compared to both Linear- and MLP-NODEs (Fig 4B), suggesting that ODIN's injectivity allows more accurate estimation of nonlinear embeddings (two-sided paired t-test, p-val for ODIN vs. Linear-, MLP-NODE < 1e-10).

5. Discussion

Dynamics models have had great success in reproducing neural activity patterns and relating brain activity to behavior [Pandarinath et al. (2018); Smith et al. (2023)], but these models have not yet been used to investigate neural computation directly. If NPDMs could be trusted to find accurate representations of latent dynamics, then recent techniques that can uncover computation in artificial networks could help to interpret computations in the brain [Sussillo and Barak (2013); Driscoll et al. (2022)].

A primary limitation of this work is that ODIN is not yet able to account for external inputs to the system. Inferring inputs is difficult due to ambiguity in the role and timecourse of inputs compared to internal dynamics for driving the state of the system. While some RNN-based models can infer inputs [Pandarinath et al. (2018)], more work is needed to infer inputs to NODE-based models. Injective readouts are an important step towards addressing the fundamental difficulties of input inference, as models without injective readouts can be incentivized to imagine latent features in place of an inferred input.

References

Evan Archer, Il Memming Park, Lars Buesing, John Cunningham, and Liam Paninski. Black box variational inference for state space models, November 2015. URL http://arxiv.org/abs/1511.07367. arXiv:1511.07367 [stat].

A Arneodo, P Coullet, and C Tresser. Occurence of strange attractors in three-dimensional Volterra equations. *Physics Letters A*, 79(4):259–263, October 1980. ISSN 0375-9601. doi: 10.1016/0375-9601(80)90342-4. URL https://www.sciencedirect.com/science/article/pii/0375960180903424.

Mohammad Bashiri, Edgar Walker, Konstantin-Klemens Lurz, Akshay Jagadish, Taliah Muhammad, Zhiwei Ding, Zhuokun Ding, Andreas Tolias, and Fabian Sinz. A flow-based latent state generative model of neural population responses to natural images. In *Advances in Neural Information Processing Systems*, volume 34, pages 15801–15815. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/84a529a92de322be42dd3365afd54f91-Abstract.html.

Jens Behrmann, Will Grathwohl, Ricky T. Q. Chen, David Duvenaud, and Joern-Henrik Jacobsen. Invertible Residual Networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 573–582. PMLR, May 2019. URL https://proceedings.mlr.press/v97/behrmann19a.html. ISSN: 2640-3498.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural Ordinary Differential Equations. Technical Report arXiv:1806.07366, arXiv, December 2019. URL http://arxiv.org/abs/1806.07366. arXiv:1806.07366 [cs, stat] type: article.

Jeffrey Demas, Jason Manley, Frank Tejera, Kevin Barber, Hyewon Kim, Francisca Martínez Traub, Brandon Chen, and Alipasha Vaziri. High-speed, cortex-wide volumetric recording of neuroactivity at cellular resolution using light beads microscopy. *Nature Methods*, 18(9):1103–1111, September 2021. ISSN 1548-7105. doi: 10.1038/s41592-021-01239-8. URL https://www.nature.com/articles/s41592-021-01239-8. Number: 9 Publisher: Nature Publishing Group.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Laura Driscoll, Krishna Shenoy, and David Sussillo. Flexible multitask computation in recurrent networks utilizes shared dynamical motifs, August 2022. URL https://www.biorxiv.org/content/10.1101/2022.08.15.503870v1. Pages: 2022.08.15.503870 Section: New Results.

Lea Duncker and Maneesh Sahani. Dynamics on the manifold: Identifying computational dynamical activity from neural population recordings. *Current Opinion in Neurobiology*, 70:163–170, October 2021. ISSN 0959-4388. doi: 10.1016/j.conb.2021.10.014. URL https://www.sciencedirect.com/science/article/pii/S0959438821001264.

Juan A. Gallego, Matthew G. Perich, Lee E. Miller, and Sara A. Solla. Neural Manifolds for the Control of Movement. *Neuron*, 94(5):978–984, June 2017. ISSN 1097-4199. doi: 10.1016/j.neuron.2017.05.025.

Peiran Gao and Surya Ganguli. On simplicity and complexity in the brave new world of large-scale neuroscience. *Current Opinion in Neurobiology*, 32:148–155, June 2015. ISSN 0959-4388. doi: 10.1016/J.CONB.2015.04.003. URL https://www.sciencedirect.com/science/article/pii/S0959438815000768. Publisher: Elsevier Current Trends.

Yuanjun Gao, Evan Archer, Liam Paninski, and John P. Cunningham. Linear dynamical neural population models through nonlinear embeddings. Technical Report arXiv:1605.08454, arXiv, October 2016. URL http://arxiv.org/abs/1605.08454. arXiv:1605.08454 [q-bio, stat] type: article.

Richard J. Gardner, Erik Hermansen, Marius Pachitariu, Yoram Burak, Nils A. Baas, Benjamin A. Dunn, May-Britt Moser, and Edvard I. Moser. Toroidal topology of population activity in grid cells. Technical report, bioRxiv, February 2021. URL https://www.biorxiv.org/content/10.1101/2021.02.25.432776v1. Section: New Results Type: article.

William Gilpin. Chaos as an interpretable benchmark for forecasting and data-driven modelling. *Advances in Neural Information Processing Systems*, 2021. URL http://arxiv.org/abs/2110.05266.

Matthew D. Golub and David Sussillo. Fixedpointfinder: A tensorflow toolbox for identifying and characterizing fixed points in recurrent neural networks. *Journal of Open Source Software*, 3(31):1003, 2018. doi: 10.21105/joss.01003. URL https://doi.org/10.21105/joss.01003.

Mehrdad Jazayeri and Srdjan Ostojic. Interpreting neural computations by examining intrinsic and embedding dimensionality of neural activity. Technical Report arXiv:2107.04084, arXiv, August 2021. URL http://arxiv.org/abs/2107.04084. arXiv:2107.04084 [q-bio] type: article.

Kristopher Jensen, Ta-Chu Kao, Jasmine Stone, and Guillaume Hennequin. Scalable Bayesian GPFA with automatic relevance determination and discrete noise models. In *Advances in Neural Information Processing Systems*, volume 34, pages 10613–10626. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/58238e9ae2dd305d79c2ebc8c1883422-Abstract.html.

Matthew J. Johnson, David Duvenaud, Alexander B. Wiltschko, Sandeep R. Datta, and Ryan P. Adams. Composing graphical models with neural networks for structured representations and fast inference, July 2017. URL http://arxiv.org/abs/1603.06277. arXiv:1603.06277 [stat].

Timothy D Kim, Thomas Z Luo, Jonathan W Pillow, and Carlos Brody. Inferring latent dynamics underlying neural population activity via neural differential equations. In *International Conference on Machine Learning*, pages 5551–5561. PMLR, 2021.

Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

Jakob H Macke, Lars Buesing, John P Cunningham, Byron M Yu, Krishna V Shenoy, and Maneesh Sahani. Empirical models of spiking in neural populations. In *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL https://papers.nips.cc/paper/2011/hash/7143d7fbadfa4693b9eec507d9d37443-Abstract.html.

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 autoencoders. *Nature Methods*, 15(10):805–815, October 2018. ISSN 1548-7105. doi: 10.1038/s41592-018-0109-9. URL https://www.nature.com/articles/s41592-018-0109-9. Number: 10 Publisher: Nature Publishing Group.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. Technical Report arXiv:1912.01703, arXiv, December 2019. URL http://arxiv.org/abs/1912.01703. arXiv:1912.01703 [cs, stat] type: article.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

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. Technical Report arXiv:2109.04463, arXiv, January 2022. URL http://arxiv.org/abs/2109.04463. arXiv:2109.04463 [cs, q-bio] type: article.

Evan D. Remington, Devika Narain, Eghbal A. Hosseini, and Mehrdad Jazayeri. Flexible Sensorimotor Computations through Rapid Reconfiguration of Cortical Dynamics. *Neuron*, 98(5):1005–1019.e5, June 2018. ISSN 1097-4199. doi: 10.1016/j.neuron.2018.05.020.

Olivier Roy and Martin Vetterli. The Effective Rank: a Measure of Effective Dimensionality. *European Association for Signal Processing*, 2007.

Marine Schimel, Ta-Chu Kao, Kristopher T. Jensen, and Guillaume Hennequin. iLQR-VAE : control-based learning of input-driven dynamics with applications to neural data. Technical report, bioRxiv, October 2021. URL https://www.biorxiv.org/content/10.1101/2021.10.07.463540v1. Section: New Results Type: article.

Andrew R. Sedler, Christopher Versteeg, and Chethan Pandarinath. Expressive architectures enhance interpretability of dynamics-based neural population models, February 2023. URL http://arxiv.org/abs/2212.03771. arXiv:2212.03771 [cs, q-bio].

Krishna V. Shenoy, Maneesh Sahani, and Mark M. Churchland. Cortical control of arm movements: a dynamical systems perspective. *Annual Review of Neuroscience*, 36:337–359, July 2013. ISSN 1545-4126. doi: 10.1146/annurev-neuro-062111-150509.

Jimmy T. H. Smith, Scott W. Linderman, and David Sussillo. Reverse engineering recurrent neural networks with Jacobian switching linear dynamical systems. Technical Report arXiv:2111.01256, arXiv, November 2021. URL http://arxiv.org/abs/2111.01256. arXiv:2111.01256 [cs] type: article.

Jimmy T. H. Smith, Andrew Warrington, and Scott W. Linderman. Simplified State Space Layers for Sequence Modeling, March 2023. URL http://arxiv.org/abs/2208.04933. arXiv:2208.04933 [cs].

Nicholas A Steinmetz, Cagatay Aydin, Anna Lebedeva, Michael Okun, Marius Pachitariu, Marius Bauza, Maxime Beau, Jai Bhagat, Claudia Böhm, Martijn Broux, Susu Chen, Jennifer Colonell, Richard J Gardner, Bill Karsh, Fabian Kloosterman, Dimitar Kostadinov, Carolina Mora-Lopez, John O'Callaghan, Junchol Park, Jan Putzeys, Britton Sauerbrei, Rik J J van Daal, Abraham Z Vollan, Shiwei Wang, Marleen Welkenhuysen, Zhiwen Ye, Joshua T Dudman, Barundeb Dutta, Adam W Hantman, Kenneth D Harris, Albert K Lee, Edvard I Moser, John O'Keefe, Alfonso Renart, Karel Svoboda, Michael Häusser, Sebastian Haesler, Matteo Carandini, and Timothy D Harris. Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings. *Science*, 372(6539), April 2021.

Ian H. Stevenson and Konrad P. Kording. How advances in neural recording affect data analysis. *Nature Neuroscience*, 14(2):139–142, February 2011. ISSN 1546-1726. doi: 10.1038/nn.2731.

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 1530-888X. doi: 10.1162/NECO_a_00409.

David Sussillo, Rafal Jozefowicz, L. F. Abbott, and Chethan Pandarinath. LFADS - Latent Factor Analysis via Dynamical Systems. Technical Report arXiv:1608.06315, arXiv, August 2016. URL http://arxiv.org/abs/1608.06315. arXiv:1608.06315 [cs, q-bio, stat] type: article.

Saurabh Vyas, Matthew D. Golub, David Sussillo, and Krishna V. Shenoy. Computation Through Neural Population Dynamics. *Annual Review of Neuroscience*, 43(1): 249–275, 2020. doi: 10.1146/annurev-neuro-092619-094115. URL https://doi.org/10.1146/annurev-neuro-092619-094115. _eprint: https://doi.org/10.1146/annurev-neuro-092619-094115.

Anqi Wu, Nicholas A. Roy, Stephen Keeley, and Jonathan W Pillow. Gaussian process based nonlinear latent structure discovery in multivariate spike train data. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://papers.nips.cc/paper_files/paper/2017/hash/b3b4d2dbedc99fe843fd3dedb02f086f-Abstract.html.

Yuan Zhao and Il Memming Park. Variational Online Learning of Neural Dynamics. *Frontiers in Computational Neuroscience*, 14, 2020. ISSN 1662-5188. URL https://www.frontiersin.org/article/10.3389/fncom.2020.00071.

Ding Zhou and Xue-Xin Wei. Learning identifiable and interpretable latent models of high-dimensional neural activity using pi-VAE, November 2020. URL http://arxiv.org/abs/2011.04798. arXiv:2011.04798 [cs, q-bio, stat].

Expressive dynamics models with nonlinear injective readouts enable reliable recovery of latent features from neural activity

Supplementary Material

Appendix A. Datasets

A.1. Simulated neural data

A.1.1. Latent trajectories

We used the Arneodo system [Arneodo et al. (1980)] to generate synthetic data because it exhibits mildly chaotic behavior (Lyapunov exponent equal to 0.243), it has a low-dimensional state space, and the regions around its fixed points are well-sampled by trajectories of the system. As demonstrated by [Sedler et al. (2023)], these properties allow recovery of latent dynamics in the absence of a nonlinear embedding. The Arneodo system is described by the following system of equations

$$\dot{x} = y \tag{13}$$

$$\dot{y} = z \tag{14}$$

$$\dot{z} = -ax - by - cz + dx^3 \tag{15}$$

where $a = -5.5$, $b = 4.5$, $c = 1.0$, and $d = -1.0$ [Arneodo et al. (1980)].

The system was simulated using the `dysts` Python package, which offered well-reasoned standards for initial conditions, integration steps, and resampling frequency [Gilpin (2021)]. Initial conditions had been determined by running the model until the moments of the autocorrelation function were stationary. Integration steps had been chosen based on the highest significant frequency observed in the power spectrum. After integration, trajectories were resampled to contain 35 points per period, where period was based on the dominant frequency in the power spectrum.

A.1.2. Embedding low-dimensional trajectories on a nonlinear manifold

We simulated neural activity by nonlinearly embedding the Arneodo trajectories as firing rates in the neural space. First, the trajectories were linearly projected into the neural space via a set of encoding vectors γ_i and standardized for each neuron (see Methods). These activations $\mathbf{a}_i$ were passed through a sigmoid with input scaling η_i and output scaling $b = 2$ to produce reasonable firing rates as follows:

$$\eta_{\mathbf{i}} = 10^{0.8 \cdot \frac{i-1}{N-1} + 0.2}, \tag{16}$$

$$\mathbf{y}_i = \psi_i(\mathbf{a}_i) = b \times \sigma(\eta_i \times \mathbf{a}_i), \quad i = 1, 2, \ldots, N. \tag{17}$$

where $\sigma(\cdot)$ denotes the sigmoid function. This resulted in a set of activation functions $\psi_i(\cdot)$ ranging from quasi-linear to step-like. The resulting rates $\mathbf{y}_i$ were used to parameterize a Poisson process, which was sampled to obtain spiking data for N neurons ($N = 12$).

A.1.3. Embedding low-dimensional trajectories onto linear manifold

For Supp. Figure S3, we tested whether Linear-NODEs fit to linearly-embedded data would find non-injective readouts when $\hat{D} > D$. We simulated an alternative dataset with the same procedure as above, except instead of passing the activations $\mathbf{a}_i$ through the sigmoidal nonlinearity, we exponentiated them to find the rate parameter $\mathbf{y}_i$ of a Poisson process, which was sampled to obtain spiking data for N neurons ($N = 12$).

Appendix B. Model training

B.1. Simulated neural data

All weights were initialized from $\mathcal{U}(-\sqrt{k}, \sqrt{k})$, where $k = 1/\text{in\_features}$ for linear layers and $k = 1/\text{hidden\_size}$ for the GRU encoder weights. Dropout layers ($p = 0.05$) were inserted before and after the initial condition linear projection during training. We used the average Poisson negative log-likelihood (NLL) across neurons and time points as our training objective. Models were trained incrementally to improve the stability of training: rather than compute loss on the whole trajectory, we added groups of 5 new time steps every 75 epochs, up to the max of 70 steps. Models were trained by stochastic gradient descent using Adam for 3000 epochs. A single learning rate was shared for the optimizer of the encoder, generator, and readout weights for each model. Each generator was a NODE that contained an MLP with six hidden layers, each with 128 ReLU units.

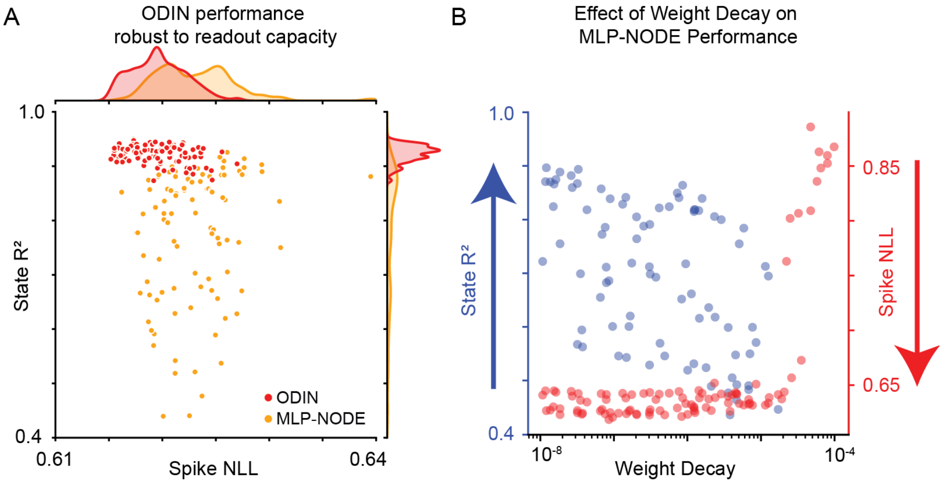

Figure S1: Example hyperparameter sweeps for ODIN and MLP-NODE

We performed initial hyperparameter sweeps to determine ranges that resulted in good reconstruction performance as measured by Spike NLL (see Methods), and used the same hyperparameter setting for models across state dimensionalities. Two example hyperparameter sweeps testing the effect of readout capacity (100 model initializations with readout hidden sizes in the range [60,200] and number of hidden layers in [1,3]) and weight decay (100 model initializations with weight decay drawn log-uniformly from the range [1e-8, 1e-4], Supp. Fig. S1). We found that across readout capacities, good reconstruction performance implied good latent recovery for ODIN but not MLP-NODE. Additionally, we found that increasing weight decay on MLP-NODE tended to degrade rather than improve latent recovery. Across all HPs tested, we found no hyperparameter settings for which ODIN had good reconstruction performance but poor latent recovery.

HPs for models trained on the Arneodo system are given in Table 1.

Table 1: Training hyperparameters (Synthetic Data)

	Arneodo		
	Linear	MLP	ODIN
Batch Size	650	650	650
Learning Rate	2e-3	1.88e-4	1.88e-4
Encoder Hidden Size	100	100	100
Dropout	0.05	0.05	0.05
NODE Hidden Layers	6	6	6
NODE Hidden Size	128	128	128
Readout Hidden Layers	0	2	2
Readout Hidden Size	-	150	150

Appendix C. Injectivity estimation

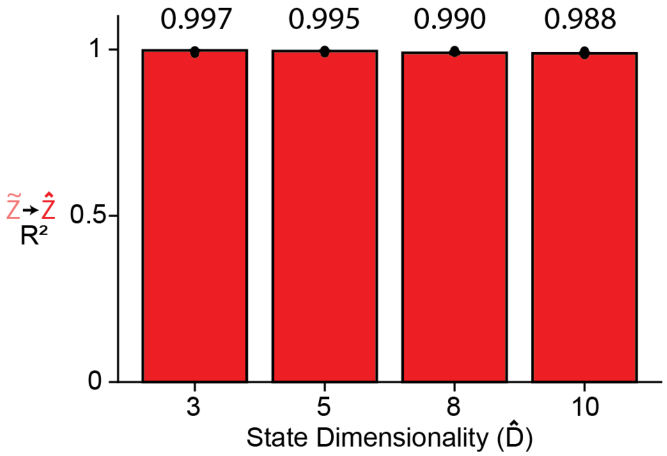

Figure S2: Reconstructed Latent R^2 (measure of injectivity) of the Flow readout across state dimensionalities. Each bar indicates the mean value of 5 randomly initialized ODIN models for each state dimensionality. Results from individual models are plotted as points.

To demonstrate the approximate injectivity of the Flow readout, we tested whether the readout could be inverted to recover the inferred latent activity. The readout mapping $\hat{g}$ should satisfy the following equations

$$\tilde{\mathbf{z}}_t = \hat{g}^{-1}(\hat{g}(\hat{\mathbf{z}}_t)) \tag{18}$$

$$\tilde{\mathbf{z}}_t \approx \hat{\mathbf{z}}_t \tag{19}$$

where $\hat{\mathbf{z}}_t$ is the inferred latent activity and $\tilde{\mathbf{z}}_t$ is the latent activity recovered by the reverse pass of the Flow.

We computed the R^2 between the inferred and recovered $\mathbf{z}_t$ for these models and found that our mappings were able to recover the inferred $\hat{\mathbf{z}}_t$ with average R^2 values across randomly initialized models of 0.997, 0.996, 0.990, and 0.988 at $\hat{D} = 3, 5, 8, 10$, respectively (Supplementary Figure S2).

C.1. Effective Rank

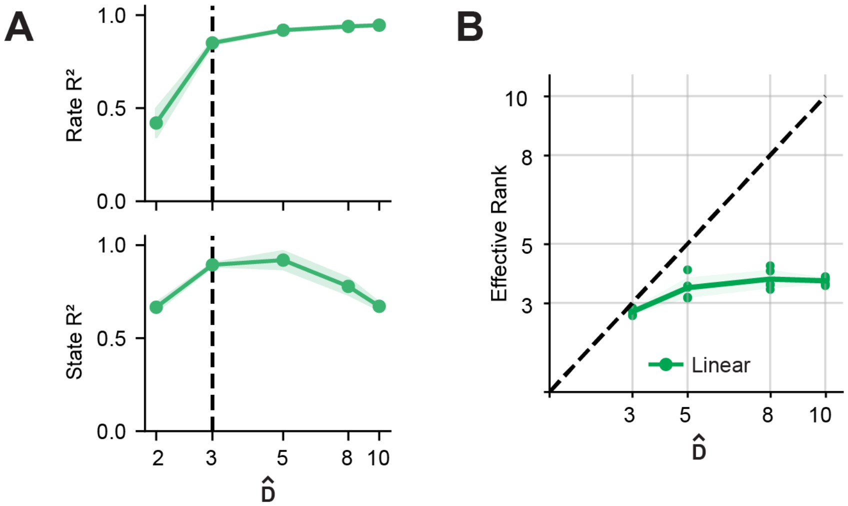

Figure S3: Linear-NODE trained on synthetic neural activity from linearly-embedded Arneodo system. A) Reconstruction performance (Rate R^2) and latent state reconstruction (State R^2) across $\hat{D}$. B) Effective rank for Linear-NODEs as a function of $\hat{D}$.

To assess the injectivity of the Linear readout, we used a previously published method that determines the approximate number of significant singular values of a given matrix A [Roy and Vetterli (2007)]. Let A be a complex-valued, non-all-zero matrix of size $N \times \hat{D}$, where $N > \hat{D}$ that acts as the weight matrix of a readout from inferred latents $\hat{\mathbf{z}}$ to predicted log-rates $\log \hat{\mathbf{y}}$ in the equation $\log \hat{\mathbf{y}} = A\hat{\mathbf{z}} + \mathbf{b}$. We perform a singular value decomposition (SVD) on A, such that $A = U\Delta V$, where U and V are unitary matrices of size $N \times N$ and $\hat{D} \times \hat{D}$, respectively, and Δ is an $N \times \hat{D}$ rectangular diagonal matrix containing the real non-negative singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{\hat{D}} \geq 0$.

For simplicity, let us define $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_{\hat{D}})^T$. We then compute the singular value distribution p_k, for $k = 1, 2, \ldots, \hat{D}$, as

$$p_k = \frac{\sigma_k}{\|\sigma\|_1}, \tag{20}$$

where $\|\sigma\|_1$ is the L_1-norm. Using this singular value distribution, we can calculate the Shannon entropy H as

$$H(p_1, p_2, \ldots, p_{\hat{D}}) = -\sum_{k=1}^{\hat{D}} p_k \log(p_k). \tag{21}$$

The authors in Roy and Vetterli (2007) define the effective rank of the matrix A, denoted as erank(A), using the Shannon entropy H as follows:

$$\text{erank}(A) = \exp\left(H(p_1, p_2, \ldots, p_{\hat{D}})\right). \tag{22}$$

The effective rank gives us a measure of the number of significant singular values in A. As traditional rank counts a matrix as being "full-rank" even if it has negligibly small but non-zero singular values, the effective rank provides a more informative assessment of the matrix's rank when used as the readout from a NPDM. We assessed the effective rank of the linear readout for 5 Linear-NODE models (with state dimensionality of $\hat{D} = 2, 3, 5, 8, 10$, respectively) trained on synthetic neural data generated by linearly embedding trajectories from the Arneodo system (Section A.1.1) into log-firing rates, and found that while the reconstruction performance improved as $\hat{D}$ increased (Supp. Fig. S3A), the effective rank plateaued at erank ≈ 4 (Supp. Fig S3B).

C.2. Cycle Consistency

To directly compare injectivity of the Flow readout versus the MLP, we quantified how well each model's inferred latent activity could be recovered from the reconstructed log-rates. To do this, we took our fully-trained 10D ODIN and MLP-NODE models (shown in Fig. 2C, $\hat{D} = 10$) and obtained the inferred latent activity $\hat{\mathbf{z}}$ and predicted log-firing rates $\log \hat{\mathbf{y}}$ from the Arneodo dataset. Then, we trained a separate network $h : \log \mathbf{Y} \to \mathbf{Z}$ to minimize the mean squared error between its output $\tilde{\mathbf{z}}$ and the model-inferred latent activity $\hat{\mathbf{z}}$ (see Table 2 for hyperparameters).

$$\tilde{\mathbf{z}} = h\left((\log \hat{\mathbf{y}})\right) \tag{23}$$

Table 2: Training hyperparameters (Cycle-Consistency MLP, h)

Parameter	Value
Batch Size	2048
Learning Rate	1e-3
Hidden Layers	3
Hidden Size	128
Epochs	1000

We computed the coefficient of determination between the re-generated latent activity $\tilde{\mathbf{z}}$ and inferred latent activity $\hat{\mathbf{z}}$. If this performance is high, the inferred latents can be recovered from the log-rates suggesting that the readout is approximately injective.

$$\text{Cycle Consistency} = R^2(\tilde{\mathbf{z}}, \hat{\mathbf{z}}) \tag{24}$$

It is possible for a readout to be fully injective (i.e., that $\hat{g}^{-1}$ exists), but still compress some features of latent activity into negligibly small contributions to the predicted firing rates, making the readout effectively, if not technically, non-injective. We reasoned that if this were the case, the inverse mapping h, in order to properly invert the warping applied by $\hat{g}$, would be highly sensitive to noise. We expect that such noise perturbations would be warped by h into large changes in the predicted latents. Using the models trained without noise, we computed the R^2 of re-generated latents $\tilde{\mathbf{z}}$ compared to the inferred latents $\hat{\mathbf{z}}$. We therefore consider both the noise-free and noise-corrupted cycle consistency scores as indicators of the approximate injectivity of each readout, taking into consideration undue distortion applied in the process of learning the injective mapping.

$$\tilde{\mathbf{z}}_\sigma = h(\log \hat{\mathbf{y}} + \epsilon_\sigma), \quad \epsilon_\sigma = \mathcal{N}(0, \sigma), \quad \sigma \in [0.01, 0.05, 0.1, 0.2, 0.5]) \tag{25}$$

$$ccR^2_\sigma = R^2(\tilde{\mathbf{z}}_\sigma, \hat{\mathbf{z}}) \tag{26}$$

C.3. Alternative injective readout

As an additional confirmation that injectivity was the critical addition to non-linear readouts that made latent recovery more robust, we tested an alternative injective architecture — an invertible neural network (INN) [Dinh et al. (2014)]. We found that using a 6-layer INN in place of the Flow readout had comparable Rate R^2 and State R^2 to ODIN, and that, like ODIN, State R^2 was stable as $\hat{D}$ increased beyond $D = 3$ (see Supp. Fig. S5). Each INN layer was composed of coupling, permutation and affine transformations. Additional training parameters are noted in Table 3. This result further supports our claims that injective networks empirically promote robust latent recovery.

Unfortunately, the INN hidden layer size is obligated to be the size of either the input or output dimensionalities, whichever is larger. Therefore, in realistic biological datasets where the number of neurons can be highly variable across datasets, the capacity of the INN readout is intrinsically linked to the number of recorded neurons. For this reason, we chose to use the Flow readout, which decouples the computational capacity of the injective transformation from the dimensionality of the neural space.

Table 3: Training hyperparameters INN (Synthetic Arneodo Data)

Parameter	Value
Batch Size	650
Learning Rate	1.88e-4
Encoder Hidden Size	100
Dropout	0.05
NODE Hidden Layers	6
NODE Hidden Size	128
Readout Hidden Layers	6
Readout Hidden Size	12

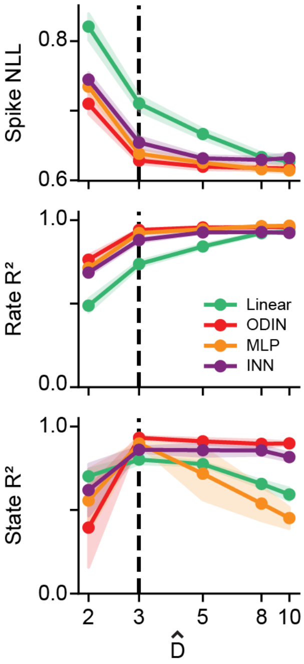

Figure S5: Invertible Neural Network readouts produce qualitatively similar results to Flow readout models. Data shown is the same as Fig. 2C, except overlaid with INN readout model (purple)

Appendix D. Fixed point finding and characterization

For each model (Linear-NODE, MLP-NODE and ODIN), we located fixed points (FPs) by finding the positions in the latent space that minimized the norm of the vector field via the objective $q = \frac{1}{2}\|\hat{f}\|_2^2$ [Sussillo and Barak (2013); Golub and Sussillo (2018)]. We initialized our search with 1024 randomly sampled initial states from along inferred latent trajectories. We used Adam with a learning rate of 5e-2 to minimize the q-value for each point independently over 10,000 iterations. Candidate points that did not achieve a q-value less than a magnitude of 7e-3 were excluded. As more than one candidate can approach the same FP, we combined candidate points that were within a specified distance, $\epsilon = 1$, from one another. In practice, points that were excluded had much larger q-values than the putative fixed points. We then linearized the dynamics around each FP and computed the system Jacobian to determine the stability and rotational character of the system around these FPs.

Appendix E. Metrics

E.1. Synthetic data metrics

E.1.1. Rate reconstruction (Rate R^2)

We computed the coefficient of determination between true ($\mathbf{Y}$) and predicted ($\hat{\mathbf{Y}}$) rates for each neuron, and reported the average value across neurons.

$$\text{Rate } R^2 = R^2(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{N}\sum_{i=0}^{N} 1 - \frac{\sum(\mathbf{y}_i - \hat{\mathbf{y}}_i)^2}{\sum(\mathbf{y}_i - \bar{\mathbf{y}}_i)^2}$$

E.1.2. Latent state reconstruction (State R^2)

To compute State R^2, we concatenated a vector of ones with the true latent states ($\mathbf{Z_1}$), then used the pseudoinverse to find the optimal affine transformation from the true latents to the inferred latents ($\hat{\mathbf{Z}}$) (i.e., optimal linear estimation). We computed the coefficient of determination (R^2) between the true and inferred latent activity with the same equation as in E.1.1.

$$\mathbf{W}_z = \mathbf{Z}_1^{\dagger}\hat{\mathbf{Z}} \tag{27}$$

$$\text{State } R^2 = R^2(\hat{\mathbf{Z}}, \mathbf{Z}_1\mathbf{W}_z) \tag{28}$$

E.1.3. Activation function comparison

We developed a method for deriving an estimate of the inferred activation functions $\hat{\psi}_i(\cdot)$ for a comparison to the true activation functions $\psi_{\mathbf{i}}(\cdot)$ (see Equation 17). We projected the true encoding vectors $\boldsymbol{\gamma}_i$ into the latent space of the model via the affine transformation $\mathbf{W}_z$ (see section E.1.2). We then used these encoding vectors $\hat{\boldsymbol{\gamma}}_i \in \mathbb{R}^{\hat{D}}$ to convert inferred latent states $\hat{\mathbf{Z}} \in \mathbb{R}^{T\times\hat{D}}$ into an activation $\hat{\mathbf{a}}_i \in \mathbb{R}^T$ for each neuron.

$$\hat{\boldsymbol{\gamma}}_i = \boldsymbol{\gamma}_{1,i}\mathbf{W}_z, \text{ for } i = 1, 2, \cdots, N \tag{29}$$

$$\hat{\mathbf{a}}_i = \hat{\mathbf{Z}} \cdot \hat{\boldsymbol{\gamma}}_i \tag{30}$$

To estimate the activation function for a given neuron i, we need pairs of inferred activations $\hat{\mathbf{a}}_i$ and firing rates $\hat{\mathbf{y}}_i$. For each neuron, we split firing rates into 20 quantiles and computed the corresponding median activation $\hat{\mathbf{a}}_{i,1:20}^{med}$ and firing rate $\hat{\mathbf{y}}_{i,1:20}^{med}$ within each quantile.

$$\hat{\mathbf{y}}_{i,1:20}^{med}, \hat{\mathbf{a}}_{i,1:20}^{med} = \text{Quantize}(\hat{\mathbf{y}}_i, \hat{\mathbf{a}}_i, 20) \tag{31}$$

We represented the inferred activation function $\hat{\psi}_i(\cdot)$ using these activation-firing rate pairs. We then performed the same procedure on the true rates and activations to find a similar representation of the true activation function $\psi_i(\cdot)$ for each neuron. To compare the true activation function $\psi(\cdot)$ to the estimated activation function $\hat{\psi}(\cdot)$, we combined the activations of each neuron i and its corresponding firing rate as the columns of the matrices:

$$\hat{\boldsymbol{\Psi}}_i = \begin{pmatrix}\hat{\mathbf{a}}_i^{med} & \hat{\mathbf{y}}_i^{med}\end{pmatrix}, \quad \boldsymbol{\Psi}_i = \begin{pmatrix}\mathbf{a}_i^{med} & \mathbf{y}_i^{med}\end{pmatrix}$$

Because the inferred latent activity can be scaled and translated arbitrarily with respect to the true latent activity, we found the optimal affine transformation between $\hat{\mathbf{\Psi}}_i$ and $\mathbf{\Psi}_i$. We used the R^2 of this mapping to quantify the correspondence between the two activation functions $\hat{\psi}_i(\cdot)$ and $\psi_i(\cdot)$ for each neuron.

Appendix F. Compute resources

We used an internal computing cluster with a total of 30 Nvidia GeForce RTX 2080 Ti GPUs for model training. Each model trained on simulated neural data took approximately 3 hours to train. With 2 models training on each GPU, the 100 models included in Figs. 2, 3, and 4 took approximately 150 GPU-hours. FP finding was fast, requiring 1 minute for each model.

Appendix G. Open-source packages used

- `torch` Paszke et al. (2019) (BSD license): Deep learning framework providing layer definitions, GPU acceleration, automatic differentiation, optimization, and more.
- `pytorch_lightning` (Apache 2.0 license): Lightweight wrappers for model training.
- `ray.tune` Liaw et al. (2018) (Apache 2.0 license): Distributed hyperparameter tuning.
- `dysts` Gilpin (2021) (Apache 2.0 license): Implementations for modeled dynamical systems.
- `fixed_point_finder` Golub and Sussillo (2018) (Apache 2.0 license): Inspiration for `torch`-based fixed point finder.
- `FrEIA` (MIT license): Implementation of alternative Invertible Neural Network architecture.
- `scikit-learn`Pedregosa et al. (2011) (BSD License): Implementations of linear regression models and principal component analysis.