NetFormer: An interpretable model for recovering identity and structure in neural population dynamics

Wuwei Zhang[*]
Princeton University

Ziyu Lu[*]
University of Washington

Trung Le
University of Washington

Hao Wang
Rutgers University

Uygar Sümbül
Allen Institute for Brain Science

Eric Shea-Brown[†]
University of Washington

Lu Mi[†]
Georgia Institute of Technology

Abstract

Neuronal dynamics are highly nonlinear and nonstationary. Traditional methods for extracting the underlying network structure from activity recordings mainly concentrate on modeling static connectivity, without accounting for key nonstationary aspects of biological neural systems, such as ongoing synaptic plasticity and neuronal modulation. To bridge this gap, we introduce the NetFormer model, an interpretable approach applicable to such systems. In our model, activity of each neuron across a series of historical time steps is defined as a token. These tokens are then linearly mapped through a query and key mechanism to generate a state- (and hence time-) dependent attention matrix that directly encodes nonstationary connectivity structures. We analyzed our formulation from the perspective of nonstationary and nonlinear networked dynamical systems. We then applied NetFormer to a large-scale, multi-modal dataset of neural activity patterns across populations of neurons in mouse visual cortex. By comparing against an allied dataset containing ground-truth baselines for connectivity between cell types, we demonstrated the effectiveness of NetFormer in predicting neural dynamics and recovering the underlying structural information about the molecular identity.

1 Introduction

Inferring the underlying connectivity of a network from observations of the activity of its units is a long-standing challenge. In the brain, this challenge is exacerbated by (i) different nonlinear dynamics present in individual neurons, (ii) experimental difficulty in sampling the full neuronal population simultaneously, and (iii) dynamic reconfiguration of effective connectivity, mediated by both synaptic plasticity and neuromodulation. This last issue carries significant practical importance in studying behavioral dynamics, learning and memory [1, 2, 3, 4, 5]. As such, it poses a (harder) generalization of the classical problem where the connectivity should no longer be considered as a static unknown, rather as a dynamical variable that needs to be inferred and tracked over time.

A surrogate, but not sufficient, measure of success in unsupervised inference of connectivity is the inferred network's success in fitting the observed dynamics (Appendix A.1). While traditional linear

[*]Equally contributing first authors.
[†]Equally contributing senior authors.
Correspondence to: wz1411@princeton.edu, luziyu@uw.edu, etsb@uw.edu, lmi7@gatech.edu.

1^{st} Workshop on NeuroAI @ 38th Conference on Neural Information Processing Systems (NeurIPS 2024).

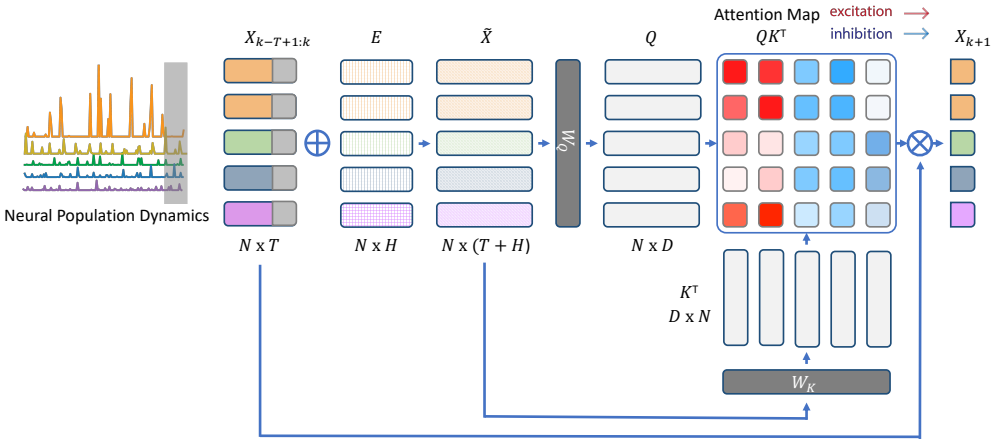

Figure 1: Overview of NetFormer. NetFormer learns to predict neural dynamics and infer connectivity through a linearized attention mechanism. It takes in as input activity of N neurons across T timesteps ($X_{k-T+1:k}$), and predicts their next-step activity (X_{k+1}). Queries Q and keys K are linearly mapped from $\tilde{X}$ ($X_{k-T+1:k}$ concatenated with positional embedding E), using W_Q and W_K. Linearized attention matrix is computed as $QK^\top$, which learns the neuron-level connectivity implicitly. Red and blue in the attention map indicate excitatory and inhibitory interactions, respectively.

dynamical models struggle to capture the essential nonlinear mechanisms of leaky integration and firing in biological neurons [6], more sophisticated nonlinear models typically suffer from a lack of interpretability, making it difficult to identify the underlying connectivity [7, 8, 9]. Moreover, traditional approaches often adopt a static perspective on connectivity [10, 11], failing to account for the nonstationary interactions, such as those produced by plasticity and modulation at synapses.

Here we propose an interpretable nonlinear and nonstationary dynamical model to represent interactions between neurons (Figure 1), based on the fast weight programming nature of the attention mechanism [12]. Prior research has suggested that the attention mechanism can reveal information about the underlying structure of a system [13, 14]. We further removed the softmax activation funtion in the attention mechanism, as the constraint of attention weights summing up to one is not biologically meaningful because neither the in-degrees nor the out-degrees of neuronal connectivity (nor their counterparts incorporating synaptic strength) are invariant across neurons [15]. We first demonstrated with both mathematical analysis and simulation study that even without the softmax activation, the core part of the attention mechanism – the dot-product between queries and keys – is capable of capturing nonstationary and nonlinear structural information. We then evaluated this novel approach on publicly available datasets of neuronal activity recordings, showing its potential for recovering meaningful identity and structural information. Our main contributions are as follows: (i) We formulated a transformer-inspired network model, the NetFormer, for which the core of the attention mechanism – the dot-product between queries and keys – directly encodes nonstationary and nonlinear structure of networks; (ii) We applied the NetFormer model to population activity recorded from mouse visual cortex, and showed that it can recover experimentally measured synaptic connectivity, while benchmarking it with standard recurrent models and other common statistical metrics; (iii) We provided a mathematical analysis on how the NetFormer can capture nonstationary interactions likely to exist in the brain, providing a hypothesis for the improved performance of the NetFormer on real neural data.

2 Model: Mathematical analysis on simulated systems

We consider an N-dimensional dynamical system

$$\frac{d}{dt}\boldsymbol{x}(t) = f\big(\boldsymbol{W}(t)\boldsymbol{x}(t)\big) \tag{2.1}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^N$, $f : \mathbb{R}^N \to \mathbb{R}^N$, and $\boldsymbol{W}(t)$ is an $N \times N$ matrix whose entries may vary across time. $W_{i,j}(t)$ prescribes how the i-th variable $\boldsymbol{x}^{(i)}$ is driven by the j-th variable $\boldsymbol{x}^{(j)}$ at time t.

Let $\boldsymbol{x}_k$ be observations of the system at discrete timesteps t_k. For each k, we train the NetFormer model (Figure 1) to predict $\boldsymbol{x}_{k+1}$ based on $\boldsymbol{X}_k = [\boldsymbol{x}_{k-T+1} \quad \cdots \quad \boldsymbol{x}_k] \in \mathbb{R}^{N\times T}$, the recent T-step history of the system up to timestep k. To encode neuronal identities, a learnable positional

embedding matrix $\boldsymbol{E} \in \mathbb{R}^{N\times H}$ is concatenated to $\boldsymbol{X}_k$, giving $\tilde{\boldsymbol{X}}_k = [\boldsymbol{X}_k \quad \boldsymbol{E}] \in \mathbb{R}^{N\times(T+H)}$. The queries $\boldsymbol{Q}_k$ and keys $\boldsymbol{K}_k$ are obtained through linear transformations of $\tilde{\boldsymbol{X}}_k$, and their product gives the linearized attention matrix $\boldsymbol{A}_k$:

$$\boldsymbol{Q}_k = \tilde{\boldsymbol{X}}_k \boldsymbol{W}_Q \in \mathbb{R}^{N\times D},\ \boldsymbol{K}_k = \tilde{\boldsymbol{X}}_k \boldsymbol{W}_K \in \mathbb{R}^{N\times D},\ \boldsymbol{A}_k = \boldsymbol{Q}_k \boldsymbol{K}_k^\top \in \mathbb{R}^{N\times N}. \tag{2.2}$$

It follows that entry (i, j) of $\boldsymbol{A}_k$ is computed from the history of $\boldsymbol{x}^{(i)}$ and $\boldsymbol{x}^{(j)}$, and thus describes the relationship between the i-th and j-th variables. To predict $\boldsymbol{x}_{k+1}$, we take $\boldsymbol{x}_k$ to be the values $\boldsymbol{v}_k$ and employ the residual connection [16], obtaining prediction as

$$\hat{\boldsymbol{x}}_{k+1} = \boldsymbol{v}_k + \boldsymbol{A}_k \boldsymbol{v}_k = \boldsymbol{x}_k + \boldsymbol{A}_k \boldsymbol{x}_k, \tag{2.3}$$

which is similar to the update rule we would get if Equation 2.1 were simulated using the classical forward Euler method with step size δ

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \delta f(\boldsymbol{W}_k \boldsymbol{x}_k). \tag{2.4}$$

Since neuronal connections can be either excitatory or inhibitory, but neither effect can be arbitrarily large, we choose f to be a sigmoidal function with

$$f(0) = 0,\ f(\bar{x}) = f(0) + f'(0)\bar{x} + O(\bar{x}^3) \text{ for } \bar{x} \text{ within some interval } (-\epsilon, \epsilon) \text{ around } 0^1$$

Equation 2.4 can thus be written as

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \delta f(\boldsymbol{W}_k \boldsymbol{x}_k) = \boldsymbol{x}_k + \delta f'(0) \boldsymbol{W}_k \boldsymbol{x}_k + \delta O(\boldsymbol{x}_k^3). \tag{2.5}$$

Comparing equations 2.3 and 2.5, we deduce that the linearized attention matrix $\boldsymbol{A}_k$ learned by the NetFormer may capture the true interactions between different variables $\boldsymbol{W}_k$ by approximating $\delta f'(0)\boldsymbol{W}_k$, especially when $\epsilon < 1$ and the first order term plays the most significant role. It is not hard to see that this hypothesis also extends to systems in the form of

$$\frac{d}{dt}\boldsymbol{x}(t) = -\boldsymbol{x}(t) + f\big(\boldsymbol{W}(t)\boldsymbol{x}(t)\big), \tag{2.6}$$

which includes the decaying effect that is commonly present in neural dynamics [17] (Appendix A.2).

We first considered four simplified simulated systems, with variations in the inclusion of nonlinearity and nonstationarity:

$$\textbf{(a)}\ \frac{d\boldsymbol{x}}{dt} = \boldsymbol{W}\boldsymbol{x},\ \textbf{(b)}\ \frac{d\boldsymbol{x}}{dt} = \tanh(\boldsymbol{W}\boldsymbol{x}),\ \textbf{(c)}\ \frac{d\boldsymbol{x}}{dt} = \boldsymbol{W}(\boldsymbol{x})\boldsymbol{x},\ \textbf{(d)}\ \frac{d\boldsymbol{x}}{dt} = \tanh(\boldsymbol{W}(\boldsymbol{x})\boldsymbol{x}),$$

where $\boldsymbol{W}(\boldsymbol{x}) = \boldsymbol{W}_0 + \boldsymbol{x}\boldsymbol{\omega}^\top$. Simulation details are in Appendix A.3.1. All trained NetFormer models are able to make accurate one-step-ahead predictions ($R^2 = 1.000$, Figure 2a-d left). Visually, the average linearized attention matrix across timesteps, $\bar{\boldsymbol{A}} = \frac{1}{K}\sum_{k=1}^{K}\boldsymbol{A}_k$, provides a good characterization of the average ground-truth dynamical association matrix across timesteps, $\bar{\boldsymbol{W}} = \frac{1}{K}\sum_{k=1}^{K}\boldsymbol{W}(\boldsymbol{x}_k)$ (Figure 2a-d right). As a baseline, we consider $\boldsymbol{A}_{\text{OLS}}$ from the linear ordinary least squares regression $\hat{\boldsymbol{x}}_{k+1} = \boldsymbol{A}_{\text{OLS}}\boldsymbol{x}_k$. We used the Spearman's rank correlation coefficient (ρ) between the off-diagonal entries of $\bar{\boldsymbol{A}}$ or $\boldsymbol{A}_{\text{OLS}}$ and $\bar{\boldsymbol{W}}$ to quantify how faithfully the learned connectivities reflect the ground-truth. $\bar{\boldsymbol{A}}$ achieved comparable performance as $\boldsymbol{A}_{\text{OLS}}$ in systems **(a) (b)**, but significantly outperformed $\boldsymbol{A}_{\text{OLS}}$ in systems **(c) (d)**, both visually (Figure 2a-d right) and quantitively (Appendix A.3.2). Moreover, in the nonstationary systems **(c) (d)**, the linearized attention matrix is able to track the majority of changes in $\boldsymbol{W}(\boldsymbol{x})$ across timesteps (Figure 2e-f and Appendix A.3.3). This ability to capture nonstationarity also explains why NetFormer can outperform the linear regression model which only accounts for static connectivity.

3 Experiments on connectivity-constrained simulation and neural data

Neurons form synapses based, in part, on their cell types. The transmission of information through these synapses is stochastic, which is shaped by activity history [18]. We applied the NetFormer on a recent multi-modal dataset from [19], where both activity and cell type of neurons in the mouse primary visual cortex are available. After training the NetFormer to predict activity, we inferred the connectivity strength between cell types from its attention matrix, and compared it against

[1] see Appendix A.3.4 for more discussion on the radius of convergence of this series representation

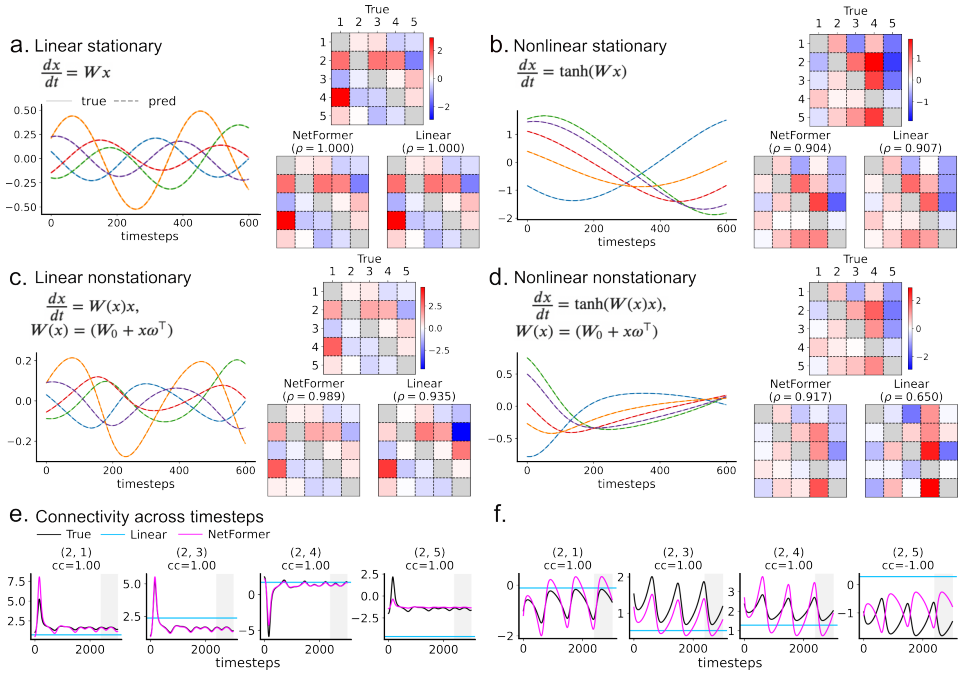

Figure 2: NetFormer provides accurate dynamics predictions, and recovers ground truth connectivity matrices. a-d Left: Test set predictions of NetFormer. Predicted trajectories were obtained by concatenating all one-step-ahead predictions. Predicted (dashed) trajectories overlap with the true (solid) trajectories. **a-d Right:** True and inferred connectivity from NetFormer's linearized attention matrix or linear regression weight matrix. For systems **c, d**, connectivity is averaged across test set timesteps for visualization. For NetFormer, the inferred connectivity shown is the one whose Spearman correlation ρ is closest to the average ρ across 10 random seeds. Colorbars: scale of off-diagonal entries. Diagonal entries are masked in grey. Inferred connectivity matrices were rescaled by the reciprocal of simulation stepsize for visualization. **e:** True and inferred temporal evolution of four example connections in the nonstationary system **c**. Timesteps used as test set are shaded in grey. **f:** True and inferred temporal evolution in system **d**.

the experimental ground truth: cell-type level postsynaptic potential (PSP) measured using patch-clamp experiments from [18], where stimuli were applied to each patched neuron and postsynaptic responses from other neurons were recorded. Additionally, following the experimental ground truth, we designed the connectivity of a synthetic neuronal population and simulated its activity. On this connectivity-constrained simulation dataset, we assessed the NetFormer's ability to infer both individual neuron-level and cell type-level connectivity. See below and Appendix A.4 for details about both datasets.

Multi-modal in-vivo neural recording: The dataset [19] includes spontaneous population activity recorded from the mouse primary visual cortex (V1) across layers 2 and 3 via two-photon calcium imaging. We trained NetFormer models on data from one animal (SB025), which includes recordings of 2481 neurons. The dataset also provides single-cell spatial transcriptomics data, enabling identification of excitatory and inhibitory neuron classes (Pvalb, Sst, and Vip).

Connectivity-constrained simulation: We generated activity of a synthetic neuron population with 200 neurons whose cell-type level connectivity is defined by results of patch clamp experiments [18], in which the probability and strength of connections between neurons are probed directly. Specifically, the population activity is generated following

$$\boldsymbol{x}_{k+1} = \tanh(\boldsymbol{W}\boldsymbol{x}_k + \boldsymbol{b}) + \boldsymbol{\epsilon}. \tag{3.1}$$

where ϵ is a Gaussian observation noise, b is a constant representing the baseline activity of each neuron, and W describes the neuronal connectivity. In our simulation, 76% of neurons are excitatory, with the remaining 24% being inhibitory. The inhibitory neurons are further subdivided into three cell types (Pvalb, Sst, and Vip) with equal size. To construct W, we first ensured that the connectivity proportion of each cell type pair aligns with the probabilities from [18]. For connected neurons, the connection strength is sampled from $\mathcal{N}(\mu, \sigma)$, where μ is the measured PSP from [18].

Neural dynamics prediction and connectivity inference: We benchmarked NetFormer against multiple methods and models. First, we compared it with linear recurrent model (referred as "linear

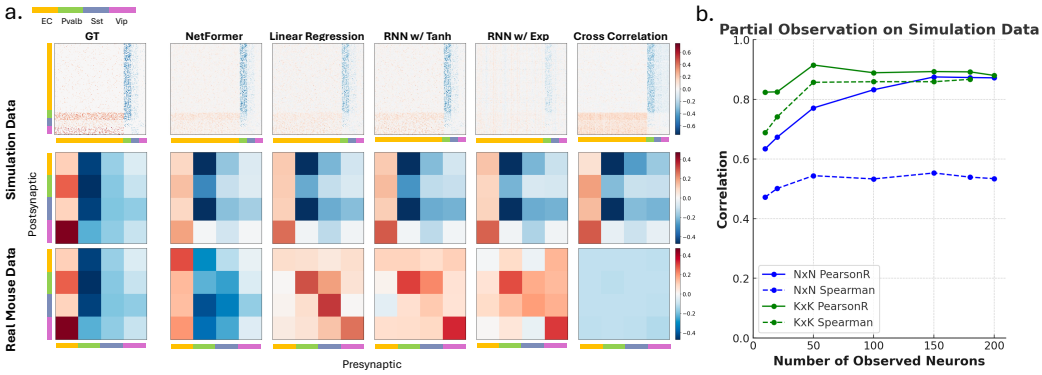

Figure 3: a. Visualization of ground truth and inferred connectivity matrices at both individual-neuron level and cell-type level. NetFormer is benchmarked with linear regression, RNNs with tanh and exponential nonlinearity, and standard statistics metrics in both simulation data and neural data. A positive linear transformation has been applied to standardize all matrices to the same range for better visualization. **b.** Experiment on partial observation with 200 neurons. Connectivity at both neuron-level and cell type-level are evaluated.

regression"): $\boldsymbol{x}_{k+1} = \boldsymbol{W}\boldsymbol{x}_k + \boldsymbol{b}$, where W represents neuronal coupling strengths and b accounts for baseline activity. Next, we compared it with two variants of nonlinear recurrent neural networks (referred as "RNNs"): (i) $\boldsymbol{x}_{k+1} = \tanh(\boldsymbol{W}\boldsymbol{x}_k + \boldsymbol{b})$, which closely matches our connectivity-constrained simulation and serves as an oracle approach to provide an upper bound on performance, (ii) $\boldsymbol{x}_{k+1} = \exp(\boldsymbol{W}\boldsymbol{x}_k + \boldsymbol{b})$, also known as the generalized linear model (GLM) [20], which models the case where the model nonlinearity does not match the one underlying the data. We also considered standard statistical metrics, including cross-correlation, covariance, mutual information, and transfer entropy (details in Appendix A.5), with results in Table 1.

We evaluated activity prediction using MSE, R^2, and Pearson correlation coefficient. We assessed the correlation between inferred and ground truth connectivity using Pearson and Spearman correlation coefficients, both at the $N \times N$ neuron level (N: number of recorded neurons) and the $K \times K$ cell type-level ($K = 4$ includes one excitatory and three inhibitory types: Pvalb, Sst, and Vip). Details on how to aggregate neuron-level connectivity to cell-level connectivity are provided in Appendix A.7.2. Qualitative comparison between the inferred connectivity and ground truth connectivity is shown in Figure 3**a**. All inferred connectivities have been linearly transformed for standardization and better visualization.

On the connectivity-constrained simulation, NetFormer, together with linear regression and cross correlation, performed comparably to the "oracle" model (RNN with tanh nonlinearity) in both neuron-wise and cell type-wise connectivity inference. RNN with exponential nonlinearity performed significantly worse, especially at individual neuron level, underscoring NetFormer's strength in not requiring prior knowledge of specific activation functions. Other statistical metrics such as covariance, mutual information, and transfer entropy all performed significantly worse than the NetFormer.

On the neural data, we only evaluated the inferred cell-type level connectivity, as we do not have access to ground truth connectivity at the level of individual neurons. Our NetFormer model consistently outperformed other methods in both inferring connectivity and predicting neural activity. Interestingly, linear regression and both variants of RNN failed to infer connectivity; we hypothesize that the reason is two-fold: the underlying connectivity is nonstationary rather than static as assumed by these models; the nonlinearity in the dynamics is different from the one used in these models. The statistical metrics were also unable to recover connectivity.

For most real neural systems, it is impossible to access activity of all neurons. To evaluate the robustness of the NetFormer model against such partial observation, in the simulated network, we randomly selected a subset of neurons and trained the NetFormer on their activity. In Figure 3**b**, we show that the performance in recovering the neuron-level connectivity does not significantly decrease with only half of the neurons observed. Moreover, cell-type level connectivity inference is less sensitive to partial observations, highlighting the potential of NetFormer to effectively derive cell-type level connectivity from real neural data. In addition, using the learned embedding matrix in NetFormer, we can decode whether a neuron is excitatory ot inhibitory (Appendix A.6).

			NetFormer	linear regression	RNN w/ tanh	RNN w/ exp	cross correlation	covariance	mutual information	transfer entropy
simulation	connectivity N × N	Pearson	**0.869**±0.002	0.817±0.002	0.905±0.000*	0.581±0.011	0.823	-0.029	0.539	0.600
		Spearman	**0.532**±0.001	0.507±0.001	0.546±0.000*	0.393±0.009	0.519	-0.015	0.262	0.339
	connectivity K × K	Pearson	0.879±0.001	0.885±0.001	0.908±0.000*	0.887±0.008	**0.888**	-0.438	0.371	0.419
		Spearman	**0.860**±0.002	0.852±0.005	0.866±0.002*	0.822±0.025	0.732	-0.334	0.018	0.353
in-vivo recording	connectivity K × K	Pearson	**0.777**±0.047	-0.395±0.020	-0.395±0.036	-0.407±0.006	-0.017	-0.162	-0.176	0.075
		Spearman	**0.847**±0.063	-0.409±0.051	-0.343±0.105	-0.191±0.300	-0.080	-0.190	-0.061	0.233
	activity prediction	MSE	**0.404**±0.004	0.443±0.001	0.560±0.001	0.476±0.003	–	–	–	–
		Pearson	**0.740**±0.003	0.720±0.001	0.639±0.000	0.699±0.002	–	–	–	–
		R^2	**0.548**±0.004	0.515±0.001	0.386±0.001	0.478±0.004	–	–	–	–

Table 1: Quantitative results from both connectivity-constrained simulation and in-vivo neural recording. An asterisk (*) indicates that RNN with tanh activation serves as the oracle model to provide an upper bound in the simulation data. The results of connectivity inference using mutual information and transfer entropy are assessed by comparing against the absolute values of the ground truth. Simulation data provides ground truth for neuron-level ($N \times N$) and cell type-level ($K \times K$) connectivity. Patch-clamp results serve as ground truth for real data cell-type connectivity. We assess performance using Spearman's and Pearson's coefficients. Next-step activity prediction on the test set is evaluated with mean squared error, Pearson's coefficient, and R^2.

4 Discussion and Conclusion

Experience and adaptation change the effective connectivity of the underlying neuronal network via mechanisms including synaptic plasticity and neuromodulation, at various time scales. This perspective poses connectivity as a dynamical variable that should be tracked, rather than inferred once. Here, we propose the NetFormer as a light-weighted model for dynamical connectivity inference. We began with a mathematical analysis that relates nonlinear and nonstationary dynamics to its linearized attention mechanism. We further demonstrated, on both simulated and in-vivo neural datasets, the strength of our model through comparison against various baselines.

5 Reproducibility

Implementation details and computing requirements are listed in Appendix A.7 and A.8. Our code is available at https://github.com/NeuroAIHub/NetFormer.

Acknowledgments

This work was supported by the NSF NCS-FO #2024364 (ZL and ESB) and the Shanahan Foundation Fellowship (WZ and LM). TL acknowledges the support in part by A3D3 NSF grant OAC-2117997 and the Department of Electrical and Computer Engineering at the University of Washington. HW is partially supported by Amazon Faculty Research Award, Microsoft AI & Society Fellowship, NSF CAREER Award IIS-2340125, NIH grant R01CA297832, and NSF grant IIS-2127918.

References

[1] Cornelia I Bargmann. Beyond the connectome: how neuromodulators shape neural circuits. *Bioessays*, 34(6):458–465, 2012.

[2] Danil Tyulmankov, Ching Fang, Annapurna Vadaparty, and Guangyu Robert Yang. Biological learning in key-value memory networks. *Advances in Neural Information Processing Systems*, 34:22247–22258, 2021.

[3] Eve Marder. Neuromodulation of neuronal circuits: Back to the future. *Neuron*, 76(1):1–11, 2012.

[4] Yuhan Helena Liu, Stephen Smith, Stefan Mihalas, Eric Shea-Brown, and Uygar Sümbül. Cell-type–specific neuromodulation guides synaptic credit assignment in a spiking neural network. *Proceedings of the National Academy of Sciences*, 118(51):e2111821118, 2021.

[5] Kyle Aitken and Stefan Mihalas. Neural population dynamics of computing with synaptic modulations. *Elife*, 12:e83035, 2023.

[6] L.F Abbott. Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain Research Bulletin*, 50(5):303–304, 1999.

[7] 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, et al. Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature methods*, 15(10):805–815, 2018.

[8] Trung Le and Eli Shlizerman. Stndt: Modeling neural population activity with spatiotemporal transformers. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 17926–17939. Curran Associates, Inc., 2022.

[9] Joel Ye, Jennifer L Collinger, Leila Wehbe, and Robert Gaunt. Neural data transformer 2: Multi-context pretraining for neural spiking activity. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[10] Alex Tank, Ian Covert, Nicholas Foti, Ali Shojaie, and Emily B Fox. Neural granger causality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4267–4279, 2021.

[11] Sindy Löwe, David Madras, Richard Zemel, and Max Welling. Amortized causal discovery: Learning to infer causal graphs from time-series data. In *Conference on Causal Learning and Reasoning*, pages 509–525. PMLR, 2022.

[12] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*, pages 9355–9366. PMLR, 2021.

[13] Ryan Singh and Christopher L Buckley. Attention as implicit structural inference. *Advances in Neural Information Processing Systems*, 36:24929–24946, 2023.

[14] Ziyu Lu, Anika Tabassum, Shruti Kulkarni, Lu Mi, J Nathan Kutz, Eric Shea-Brown, and Seung-Hwan Lim. Attention for causal relationship discovery from biological neural dynamics. *arXiv preprint arXiv:2311.06928*, 2023.

[15] Andrea Santuy, Laura Tomás-Roca, José-Rodrigo Rodríguez, Juncal González-Soriano, Fei Zhu, Zhen Qiu, Seth GN Grant, Javier DeFelipe, and Angel Merchan-Perez. Estimation of the number of synapses in the hippocampus and brain-wide by volume electron microscopy and genetic labeling. *Scientific Reports*, 10(1):14014, 2020.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] Wulfram Gerstner and Werner M Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[18] Luke Campagnola, Stephanie C Seeman, Thomas Chartrand, Lisa Kim, Alex Hoggarth, Clare Gamlin, Shinya Ito, Jessica Trinh, Pasha Davoudian, Cristina Radaelli, et al. Local connectivity and synaptic dynamics in mouse and human neocortex. *Science*, 375(6585):eabj5861, 2022.

[19] Stéphane Bugeon, Joshua Duffield, Mario Dipoppa, Anne Ritoux, Isabelle Prankerd, Dimitris Nicoloutsopoulos, David Orme, Maxwell Shinn, Han Peng, Hamish Forrest, Aiste Viduolyte, Charu Bai Reddy, Yoh Isogai, Matteo Carandini, and Kenneth D. Harris. A transcriptomic axis predicts state modulation of cortical interneurons. *Nature*, 607, 2022.

[20] Jonathan W Pillow, Jonathon Shlens, Liam Paninski, Alexander Sher, Alan M Litke, EJ Chichilnisky, and Eero P Simoncelli. Spatio-temporal correlations and visual signalling in a complete neuronal population. *Nature*, 454(7207):995–999, 2008.

[21] Saurabh Vyas, Matthew D Golub, David Sussillo, and Krishna V Shenoy. Computation through neural population dynamics. *Annual review of neuroscience*, 43:249–275, 2020.

[22] Liam Paninski, Jonathan Pillow, and Jeremy Lewi. Statistical models for neural encoding, decoding, and optimal stimulus design. *Progress in brain research*, 165:493–507, 2007.

[23] Sean Escola, Alfredo Fontanini, Don Katz, and Liam Paninski. Hidden markov models for the stimulus-response relationships of multistate neural systems. *Neural computation*, 23(5):1071–1132, 2011.

[24] Chengrui Li, Soon Ho Kim, Chris Rodgers, Hannah Choi, and Anqi Wu. One-hot generalized linear model for switching brain state discovery. In *The Twelfth International Conference on Learning Representations*, 2024.

[25] Omri Barak. Recurrent neural networks as versatile tools of neuroscience research. *Current opinion in neurobiology*, 46:1–6, 2017.

[26] Matthew G Perich, Charlotte Arlt, Sofia Soares, Megan E Young, Clayton P Mosher, Juri Minxha, Eugene Carter, Ueli Rutishauser, Peter H Rudebeck, Christopher D Harvey, et al. Inferring brain-wide interactions using data-constrained recurrent neural network models. *BioRxiv*, pages 2020–12, 2020.

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

[28] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[29] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International journal of forecasting*, 36(3):1181–1191, 2020.

[30] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 95–104, 2018.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[32] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.

[33] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021.

[34] Sarthak Jain and Byron C Wallace. Attention is not explanation. *arXiv preprint arXiv:1902.10186*, 2019.

[35] Samira Abnar and Willem Zuidema. Quantifying attention flow in transformers. *arXiv preprint arXiv:2005.00928*, 2020.

[36] Scott Linderman, Matthew Johnson, Andrew Miller, Ryan Adams, David Blei, and Liam Paninski. Bayesian learning and inference in recurrent switching linear dynamical systems. In *Artificial intelligence and statistics*, pages 914–922. PMLR, 2017.

[37] Joshua Glaser, Matthew Whiteway, John P Cunningham, Liam Paninski, and Scott Linderman. Recurrent switching dynamical systems models for multiple interacting neural populations. *Advances in neural information processing systems*, 33:14867–14878, 2020.

[38] Kenneth D Harris and Alexander Thiele. Cortical state and attention. *Nature reviews neuroscience*, 12(9):509–523, 2011.

[39] John G White, Eileen Southgate, J Nichol Thomson, Sydney Brenner, et al. The structure of the nervous system of the nematode caenorhabditis elegans. *Philos Trans R Soc Lond B Biol Sci*, 314(1165):1–340, 1986.

[40] Lu Mi, Richard Xu, Sridhama Prakhya, Albert Lin, Nir Shavit, Aravinthan Samuel, and Srinivas C Turaga. Connectome-constrained latent variable model of whole-brain neural activity. In *International Conference on Learning Representations*, 2021.

[41] Francesco Randi, Anuj K Sharma, Sophie Dvali, and Andrew M Leifer. Neural signal propagation atlas of caenorhabditis elegans. *Nature*, 623(7986):406–414, 2023.

[42] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*, 2019.

[43] Gino Brunner, Yang Liu, Damian Pascual, Oliver Richter, Massimiliano Ciaramita, and Roger Wattenhofer. On identifiability in transformers. *arXiv preprint arXiv:1908.04211*, 2019.

[44] Yu-An Wang and Yun-Nung Chen. What do position embeddings learn? an empirical study of pre-trained language model positional encoding. *arXiv preprint arXiv:2010.04903*, 2020.

[45] Lu Mi, Trung Le, Tianxing He, Eli Shlizerman, and Uygar Sümbül. Learning time-invariant representations for individual neurons from population dynamics. *Advances in Neural Information Processing Systems*, 36, 2024.

[46] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[47] William Falcon and The PyTorch Lightning team. PyTorch Lightning, March 2019.

A Appendix

A.1 Related Work

Dynamical models of neuronal activity: Dynamical models have been a powerful tool for high-dimensional neural data analysis [21]. Generalized linear models (GLMs), known for both interpretability and desirable convexity properties [22], have been widely used to model neuronal population activity as well as inter-neuronal interactions [20]. Nevertheless, unless stacked with an explicit state switching mechanism [23], in GLMs the temporal filters describing interactions among neurons are typically stationary across time [24]. Recurrent neural networks (RNNs) have been a popular alternative [25, 26]; while the connectivity in (trained) RNNs is typically given by a static connectivity matrix "W", variants including long short-term memory networks (LSTMs) [27] and gated recurrent neural networks (GRUs) [28], do include nonstationarities at the level of individual neural units. While this can enhance the model's expressivity and performance in predictive tasks [29, 30], it also introduces challenges for interpretability [10]. Recently, transformer models [31] have been observed to outperform RNNs in various time series forecasting tasks [32, 33], but their deep layered structures and nonlinear attention mechanisms also raise challenges in interpretation with respect to underlying connectivity structures in the orignal data [34, 35], as discussed more below. A closely related approach to the present work is the switching linear dynamical systems [36, 37]. These models have nonstationary connectivity matrices which switch among a number of discrete values according to a Markov process. Nevertheless, in vivo experimental recordings have revealed that cortical activity is more likely to go through a continuum of states instead of discrete switching [38]. This motivates us to propose a model capable of capturing continuous changes in connectivity.

Predicting activity from connectivity: The reverse direction, predicting activity from connectivity, is an allied approach for studying the complexities relating functional and structural information. A prominent line of study have focused on the worm C. elegans as its synaptic connectome was the first available among all species [39]. Using this, generative models of activity have been proposed [40]. Nevertheless, decades of electrophysiological analyses have emphasized the strong additional role of neuromodulators in shaping activity [41, 3]. As a result, the synaptic connectome alone predicts only partial information about recorded population dynamics [1], which directly motivates our study of nonstationary, dynamical connectivity which aims to infer connectivity from functional activity.

Interpretability of the attention mechanism: Attention weights and positional embeddings provide opportunities to understand the inner working of the transformer models. However, the interpretability of these components is a subject of debate. Findings supporting a certain level of interpretability, such as correlation to linguistic features, are common in the literature, with specialized metrics developed to quantify their interpretability [42, 35]. However, caution should be taken when equating attention with explanation [34], considering the lack of identifiability [43] and the wide variety of underlying architectures and implementations [44]. In this work, we seek to avoid these confounding aspects by focusing on the linearized attention mechanism [12].

A.2 Justification for linearized attention applied to "leaky" systems (Eqn 2.6)

Using the forward Euler method and step size δ, Equation 2.6 can be simulated as

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \delta\Big(-\boldsymbol{x}_k + f(\boldsymbol{W}_k \boldsymbol{x}_k)\Big) = \boldsymbol{x}_k - \delta \boldsymbol{x}_k + \delta f(\boldsymbol{W}_k \boldsymbol{x}_k). \tag{A.1}$$

Following the same sigmoidal assumption on f, Equation A.1 can be written as

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \delta(f'(0)\boldsymbol{W}_k - \boldsymbol{I})\boldsymbol{x}_k + \delta O(\boldsymbol{x}_k^3), \tag{A.2}$$

where $\boldsymbol{I}$ is the $N \times N$ identity matrix. Therefore, the linearized attention matrix $\boldsymbol{A}_k$ learned from Equation 2.3 may reflect the true interactions $\boldsymbol{W}_k$ by approximating $\delta(f'(0)\boldsymbol{W}_k - \boldsymbol{I})$, and can capture the interactions between different variables (off-diagonal entries of $\boldsymbol{W}_k$) up to a scaling factor $(\delta f'(0))$.

A.3 Additional details for nonlinear and nonstationary systems simulation (Sec ??)

A.3.1 Simulation details

In Figure 2**a,b**, ground-truth W were generated randomly, with real-part of each eigenvalue clipped at 0 to ensure stability of the system. W in **a,b** were also used as W_0 in **c,d**, respectively. ω in **c,d** were

picked randomly while maintaining stability of the system. In **a**, the system trajectory was simulated using the closed-form solution $x(t) = e^{Wt}\eta$. In **b-d**, trajectories were simulated using the forward Euler method: $x_{k+1} = x_k + \delta f(W_k x_k)$, where $W_k \equiv W$ for **b**, and $W_k = W_0 + x_k \omega^\top$ for **c, d**. All simulations consist of 3000 timesteps with stepsize $\delta = 0.01$, with the first 80% used as training set, and last 20% as test set. Time-averaged ground-truth across test set timesteps $\bar{W} = \sum_{k=2400}^{3000} W_k$ for **c, d**. Simulated trajectories are visualized in Figure 4. In all settings, the NetFormer model was trained to minimize the mean squared error on the trainining set for 1100 epochs using the Adam optimizer in Pytorch, with $T = 1, H = 5$, batch size = 80, initial learning rate = 0.01. In **b, d**, learning rate was decayed by a factor of 0.9 every 100 epochs. In **c**, learning rate was decayed by a factor of 0.8 every 100 epochs.

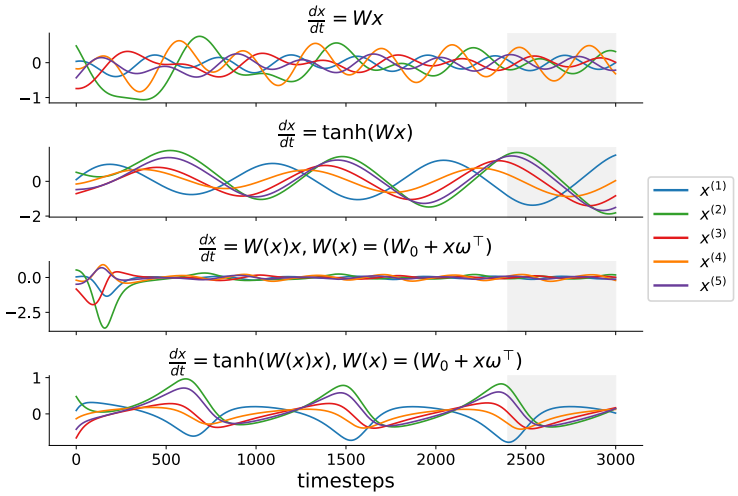

Figure 4: Simulated trajectories of toy models in Figure 2. Shaded regions represent timesteps used as test set.

A.3.2 Quantitive comparsion with linear regressoin model

For each toy system, we trained 10 NetFormer models with different random initializations, and computed the Spearman's rank correlation coefficient (ρ) and the Pearson correlation coefficient (r) between the off-diagonal entries of $\bar{A}$ and $\bar{W}$ for each trained model. In terms of ρ, $\bar{A}$ achieved comparable performance as A_{OLS} in systems **(a) (b)** ($p > 0.3$, two-sided one sample t test), but significantly outperformed A_{OLS} in systems **(c) (d)** ($p < 10^{-8}$) (figure5 left). Similar observations can be made with r (figure5 right). For each system, the attention matrix visualized in figure2 is the one whose ρ is the closet to the average ρ across 10 random initializations.

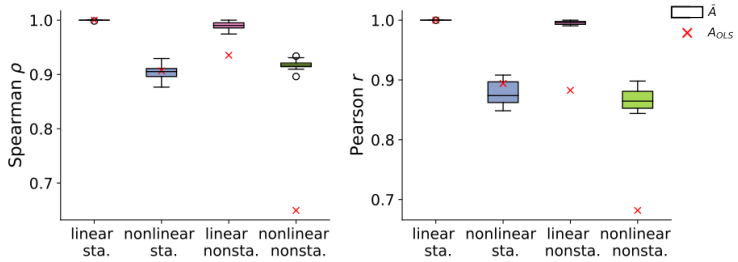

Figure 5: Comparison between A_{OLS} (red cross) and $\bar{A}$ from NetFormer models with 10 different random initializations (boxplots).

A.3.3 Nonstationarity connectivity tracking

On toy systems with nonstationary connectivity (Figure 2c, d), we evaluated how well linearized attention matrices across timesteps can track changes in the connectivity. For each pair (i, j), $i, j =$

$1, \ldots, 5, i \neq j$, we collected A_{ij} and W_{ij} across all test timesteps, resulting in two time-varying series $A_{ij}(t)$ and $W_{ij}(t)$, and computed the Pearson correlation coefficient between them. Results for 10 trained NetFormer models with different random seeds are shown in Figure 6. Distributions of the temporal correlation coefficients for all off-diagonal pairs (i, j) are shown as violin plots, where each violin corresponds to model trained with one random seed. The median of each distribution is marked with a black line. All medians are greater than 0.999.

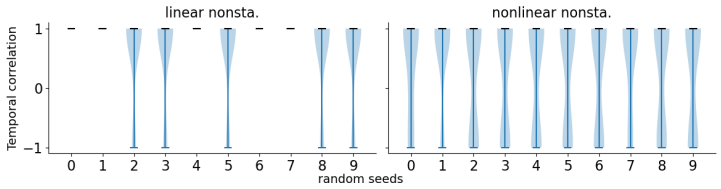

Figure 6: Distribution of test set temporal correlation between the linearized attention matrix and the true nonstationary connectivity. Each column shows result from NetFormer model with a different random seed. Median of each distribution is marked in black. All medians are greater than 0.999.

A.3.4 Further discussion on nonlinear dynamical systems

In section 2, we showed that when f is sigmoidal, A can reflect W through Taylor series approximation of $f(Wx_k)$. Take $f = \tanh$ as an example. When $|\vec{w}_i^\top x^k| < \frac{\pi}{2}\ \forall i = 1, \ldots, N$,

$$\tanh(Wx) = \tanh(0) + \tanh'(0)Wx + O(x^3).$$

As $\tanh(0) = 0$, the forward Euler method is

$$x_{k+1} = x_k + \delta \tanh(Wx_k) = x_k + \delta \tanh'(0)Wx_k + \delta O(x_k^3).$$

This analysis also applies to other sigmoidal functions f, such as $\arctan$, with

$$f(0) = 0, f(x) = f'(0)x + O(x^3) \text{ for } x \text{ within some interval around } 0.$$

Therefore, we hypothesize that A can capture W through learning $\delta f'(0)W$ for sigmoidal f. It is also clear that learning W becomes more challenging when $\vec{w}_i^\top x$ does not always stay within the radius of convergence of the Maclaurin series. Nonetheless, we note that if some $\vec{w}_i^\top x$ is constantly outside the convergence region, $f(\vec{w}_i^\top x)$ will be constantly positive or negative, and the system will either blow up or decay to zero. Therefore, for the systems of interest here, which are those with interesting persistent dynamics, from time to time $\vec{w}_i^\top x$ must fall within the convergence region where the Maclaurin series representation is valid. That being said, while A may still be able to capture some aspect of W, it could become less accurate, and may require more observations of the system to gather sufficient timestamps within the convergence region.

In the example nonlinear dynamical system shown in figure 2**b**, $\vec{w}_i^\top x^k$ stays within the radius of convergence of $\tanh$ for all i and k, which makes the Maclaurin series approximation valid for all timesteps. In figure 7, we provide another toy model example showing that the attention from NetFormer still bears considerable similarity to the ground-truth W even when $\vec{w}_i^\top x^k$ falls out of the convergence region for some i and k.

A.4 Connectivity-constrained simulation and neural data: Datasets and preprocessing

A.4.1 Connectivity-constrained simulation

We used the following procedure to construct the ground-truth connectivity matrix. For each cell-type pair and for every pair of neurons, we first drew a random sample from the uniform distribution between 0 and 1. Then, we used the connectivity probability from patch-clamp experiments [18] as a cutoff threshold to determine if two neurons are connected. For connected neurons, we sampled their connection strength from a normal distribuion $\mathcal{N}(\mu, 0.1)$, where μ is the measured post-synaptic potential from patch-clamp experiments. We simulated $30,000$ steps for 200 neurons, using the first 80% timesteps for training and the last 20% for testing.

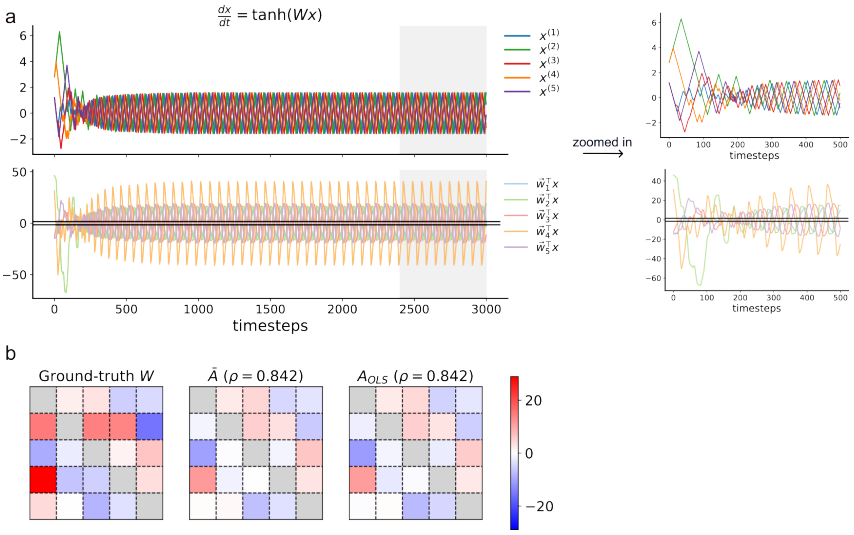

Figure 7: Demonstration of NetFormer on a nonlinear system $\frac{dx}{dt} = \tanh(Wx)$ where $\vec{w}_i^\top x^k$ does not always stay within the convergence region of the Maclaurin series of $\tanh$. **a. Top row:** Trajectories of the simulated system. Simulation was done using the forward Euler method with stepsize $\delta = 0.1$. Shaded regions represent timesteps used as test set. **Bottom row:** Visualization of $\vec{w}_i^\top x$ across simulated timesteps. Boundaries of the convergence region, $\pm\frac{\pi}{2}$, were marked with black horizontal lines. The right column provides a zoomed-in view of the first 500 simulation timesteps. **b. Left to right** Ground-truth W, average linearized attention matrix across test timesteps from NetFormer ($\bar{A}$), A_{OLS} fitted through least-squares regression. Colorbars indicate the scale of the off-diagonal entries, and the diagonal entries are masked in grey. $\bar{A}$, A_{OLS} were rescaled for visualization. Spearman's rank correlation coefficients (ρ) were computed between the off-diagonal entries of $\bar{A}$ or A_{OLS} and W. We trained 10 NetFormer models with different random initializations ($\rho = 0.841 \pm 0.02$, mean $\pm$ std), and $\bar{A}$ shown is the one whose ρ is the closet to the average ρ across 10 random initializations. NetFormer models achieved similar performance as A_{OLS} ($p = 0.9$, two-sided one sample t test). All NetFormer models were trained to minimize the mean squared error on the trainining set for 600 epochs using the Adam optimizer in Pytorch, with $T = 1, H = 5$, batch size = 80. Learning rate was initialized to 0.01, and was decayed by a factor of 0.9 every 100 epochs.

A.4.2 Patch-clamp dataset

The dataset released in [18] contains experimental results of connectivity probability and connectivity strength (Postsynaptic Potential (PSP)) at the cell-type level measured using patch-clamp. In each experiment, up to eight neurons were simultaneously subjected to whole-cell patch-clamp recording, mainly under current-clamp conditions, with some stimuli also tested under voltage-clamp conditions. Stimuli were applied to each patched neuron while recording the other neurons for postsynaptic responses. We mainly focus on the experimental results for layers 2/3 in mouse primary visual cortext (V1), to match the neurons recorded in the multimodal mouse datatset [19].

A.4.3 Multimodal mouse data

For functional activity recordings from neuronal populations, we used a recent, public multimodal dataset provided by [19]. This dataset includes spontaneous population activity recordings from the mouse primary visual cortex (V1) across layers 2/3 via 2-photon calcium imaging at a temporal sampling frequency of 4.3Hz across six 20-minute sessions, recording approximately 500 neurons per session. Spatial coordinates of the recorded neurons are also provided. We trained our models on data from one experimental subject (SB025), which includes recordings of 2481 neurons, with some neurons repeating across six sessions. The dataset also includes single-cell spatial transcriptomics, profiling mRNA expression for 72 selected genes to identify excitatory and inhibitory class labels of neurons. 51% of neurons in the inhibotiry class can further be identified to be one of Lamp5, Pvalb, Vip, Sncg, and Sst.

A.4.4 Data preprocessing

In the connectivity-constrained simulation, when using RNN with an exponential activation, we rescaled the data to ensure that all neuronal activities are nonnegative, as exponential activation produces only nonnegative outputs.

For the experimental neural recording, which is nonnegative, we normalized it using the mean and standard deviation calculated across all sessions and neurons involved in training. When using the RNN model with exponential activation, we normalized the data by dividing by the standard deviation only, without first subtracting the mean.

A.5 Baselines

Linear Regression: We denote neural activity data as $X \in \mathbb{R}^{N\times T}$ recorded from N neurons and T time steps. Let $x_{k+1} \in \mathbb{R}^N$ denote the neuronal activity at the $(k+1)$th time step. Given previous 1 time step, x_k, linear regression predicts current time step activity as

$$\hat{x}_{k+1} = Wx_k + b$$

Recurrent Neural Network (RNN) with tanh activation: Given neuronal activity across previous p time steps, $x_k, x_{k-1}, \ldots, x_{k-p+1}$, a RNN with predefined Tanh activation function predicts current time step activity as

$$\hat{x}_{k+1} = \sigma\left(W^{(0)}x_k + W^{(1)}x_{k-1} + \cdots + W^{(p-1)}x_{k-p+1} + b\right), \sigma = \tanh$$

where $W^{(l)}, l \in \{0, 1, \ldots, p-1\}$, represent how the previous l-th step affects the current step activity and each element $W^{(l)}_{ij}$ represents how the j-th neuron at the previous l-th time step influences the ith neuron in current step. The RNN is trained by minimizing mean squared errors (MSE) of the current time step activity prediction given previous time steps. $p = 1$ is commonly used for RNN. For modeling both simulation data and real mouse data, we chose $p = 1$, because the simulation data has exactly one timestep dependency. Using larger p did not improve performance in real data either.

Recurrent Neural Network (RNN) with exponential activation: Next, we change the predefined activation function of RNN to exponential function for modeling both simulation data and real mouse data.

$$\hat{x}_{k+1} = \sigma\left(W^{(0)}x_k + W^{(1)}x_{k-1} + \cdots + W^{(p-1)}x_{k-p+1} + b\right), \sigma = \exp$$

Cross correlation: Recall that $X_k = [x_{k-T+1} \quad \cdots \quad x_k] \in \mathbb{R}^{N\times T}$ denotes activity of N neurons across T time steps. Let $x^{(i)}, x^{(j)} \in \mathbb{R}^T$ denote the i-th and j-th neurons' activity across T time steps. For simpliticity, let $a = x^{(i)}[\tau :], b = x^{(j)}[: -\tau]$. Cross correlation with time delay τ reflects connectivity as

$$r^{i\leftarrow j} = \frac{a^\top b}{\|a - \overline{a}\|\|b - \overline{b}\|}$$

We choose $\tau = 1$ for inferring connectivity in both simulation and read data.

Covariance: Similar to above, let $x^{(i)}, x^{(j)} \in \mathbb{R}^T$ denote the i-th and j-th neurons' activity across T time steps. Covariance indicates the level to which two variables vary together. Covariance matrix is symmetric, which assumes that the influence from neuron i to neuron j is the same as influence from neuron j to neuron i. Covariance between neuron i and j is defined as

$$c^{i\leftarrow j} = c^{j\leftarrow i} = \frac{1}{T-1}\Sigma_{k=1}^{T} x_k^{(i)} x_k^{(j)}$$

Mutual information: Mutual information quantifies the amount of information that one random variable contains about another, which is also symmetric. When calculating the mutual information between activity history of two neurons, the computation involves estimating the entropy of each neuron's activity individually and the joint entropy of both neurons together. We used the Python package PyInform.mutualinfo to compute the mutual information. Let $x^{(i)}, x^{(j)} \in \mathbb{R}^T$ as defined above, then

$$I^{i\leftarrow j} = I^{j\leftarrow i} = I(x^{(i)}; x^{(j)}) = H(x^{(i)}) + H(x^{(j)}) - H(x^{(i)}, x^{(j)}),$$

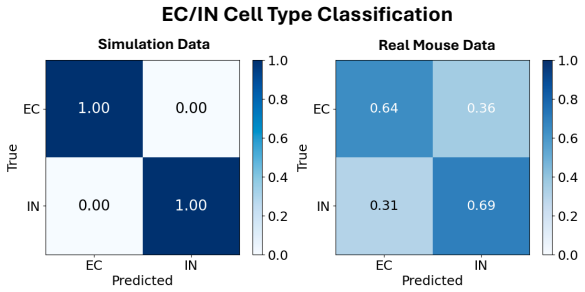

Figure 8: Confusion matrices of Excitatory/Inhibitory cell type classification using the learnable neuronal embedding from NetFormer on both connectivity-constrained simulation and real mouse data. The classifier is logistic regression. For real data, all neurons in one session are classified to be Excitatory or Inhibitory neurons.

where $H(x^{(i)})$ is the entropy of neuron i's activity, calculated as $H(x^{(i)}) = -\sum_{z \in x^{(i)}} p(z) \log p(z)$. $H(x^{(j)})$ is the entropy of neuron j's activity. $H(x^{(i)}, x^{(j)})$ is the joint entropy of neurons i and j, calculated as $H(x^{(i)}, x^{(j)}) = -\sum_{z \in x^{(i)}, \xi \in x^{(j)}} p(z, \xi) \log p(z, \xi)$.

Transfer entropy: Transfer entropy quantifies the amount of directed information transferred between systems, or in our case, between two neurons. We use python package PyInform.transferentropy to compute the transfer entropy, which is defined as

$$T^{i \leftarrow j} = \sum p\left(x_{k+1}^{(i)}, x_{(k+1-p):k}^{(i)}, x_{(k+1-l):k}^{(j)}\right) \log \left(\frac{p\left(x_{k+1}^{(i)} \mid x_{(k+1-p):k}^{(i)}, x_{(k+1-l):k}^{(j)}\right)}{p\left(x_{k+1}^{(i)} \mid x_{(k+1-p):k}^{(i)}\right)} \right),$$

where $x_{k+1}^{(i)}$ is the future activity of neuron i. $x_{(k+1-p):k}^{(i)}$ represents the past p activities of neuron i up to time k. $x_{(k+1-l):k}^{(j)}$ denotes the past l activities of neuron j up to time k. $p(\cdot)$ denotes the probability distributions calculated from the joint and conditional activities as observed in the data.

A.6 Neuronal identity information

We demonstrate the learned positional embeddings E for each neuron, as in [45]. We train a logistic regression for binary classification using embeddings as features for neurons in the training set. We classify unseen neurons in the test set as excitatory/inhibitory neurons, with 100% top-1 accuracy in simulation data, 71.21% top-1 accuracy and AUROC score of 0.700 in real mouse data (confusion matrixes in Figure 8). This shows that excitatory and inhibitory cell types are learned in a linearly separable way using the positional embeddings in the NetFormer.

A.7 Implementaion Details

A.7.1 Model framework for fitting connectivity-constrained simulation and real mouse data

Following Section 2, we train the NetFormer to predict x_{k+1} based on $X_k = [x_{k-T+1} \quad \cdots \quad x_k] \in \mathbb{R}^{N \times T}$. To encode neuron identities, a learnable positional embedding matrix $E \in \mathbb{R}^{N \times H}$ is concatenated to X, giving $\tilde{X}_k = [X_k \quad E] \in \mathbb{R}^{N \times (T+H)}$. The queries Q_k and keys K_k are obtained through linear transformations of $\tilde{X}_k$, $Q_k = \tilde{X}_k W_Q \in \mathbb{R}^{N \times D}$, and $K_k = \tilde{X}_k W_K \in \mathbb{R}^{N \times D}$. NetFormer model is trained to predict the next time-step activity x_{k+1}, defined as

$$\hat{x}_{k+1} = A_k x_k + x_k = \phi(\frac{Q_k K_k^T}{\sqrt{D}}) x_k + x_k = \frac{1}{\sqrt{D}} (\tilde{X}_k W_Q)(W_K^\top \tilde{X}_k^\top) x_k + x_k,$$

where A_k is the self-attention that we want to use for inferring connectivity, ϕ is the attention activation. In the standard Transformer model [31], softmax is used as the attention activation function, but here we set ϕ equal to identity for better interpretability. In fact, we experimented with different activation functions and empirically found that the identity activation yields the best results on recordings from the mouse cortex.

Although not used in the current experiments, it is possible to use an additional linear transformation on X_k to accommodate neuronal dynamics that can depend on multiple previous timesteps, that is,

$$\hat{x}_{k+1} = A_k(X_k w_{out}) + X_k w_{out}, w_{out} \in \mathbb{R}^{T\times 1}.$$

A.7.2 NetFormer training and evaluation

We first assign each unique neuron in all sessions an ID, which is later used to track positional embedding for each unique neuron, because same neuron can be recorded in more than one session. Then, within each session, we construct samples with window size 200 in simulation and 60 in real data, and we make sure samples in one batch should come from the same session so that the dimensions can match.

We use the first 80% time-steps in all sessions for training and the last 20% time-steps for validation. The model is trained using MSE as the loss function, comparing the predicted activity for the next time step with the ground-truth activity. We employ early stopping criteria, ceasing training if there are 20 epochs without improvement, with a hard limit of 100 epochs maximum.

After training is complete, we calculate the attention for each sample in the dataset. For each session, we aggregate the attentions from all samples to compute a single averaged attention. Averaged attentions from all sessions are then transformed into a final cell-type level attention. We achieve this by aggregating attention values according to their corresponding presynaptic and postsynaptic cell types and dividing by the total count of such pairings.

We also extract positional embeddings from the model and utilize each neuron's unique ID to determine the neuronal embedding for every unique neuron. These embeddings are then used as features for logistic regression to classify neurons as either excitatory or inhibitory.

For evaluation, we assess the inferred cell-type level connectivity against the Postsynaptic Potential (PSP) resting state amplitude obtained from patch-clamp experiments, which serves as the experimental ground-truth. Additionally, we evaluate the accuracy of the binary cell-type classification using experimental data from single-cell spatial transcriptomics, which provides a classification of neurons into excitatory and inhibitory types across all sessions.

A.7.3 Evaluation Metircs

We use python libraries and built-in functions for computing evaluation metrics.

For connectivity inference, we flatten the inferred 2-dimensional $N \times N$ or $K \times K$ connectivity matrix and grond-truth matrix.
Pearson correlation: scipy.stats.pearsonr()
Spearman rank correlation: scipy.stats.spearmanr().

For activity prediction, given the input matrix $\in \mathbb{R}^{B\times N\times T}$ for NetFormer and input matrix $\in \mathbb{R}^{B\times N}$ for RNN, where B is the batch size, NetFormer outputs $\in \mathbb{R}^{B\times N\times 1}$ and RNN outputs $\in \mathbb{R}^{B\times N}$. We flatten the predicted activity and the grond-truth.
MSE: torch.nn.functional.mse_loss()
Pearson correlation: scipy.stats.pearsonr()
$\mathbf{R^2}$ **:** sklearn.metrics.r2_score()

For binary classification, classifier predicts the probability for all neurons.
Top-1 accuracy: sklearn.metrics.accuracy_score()
Area Under the Receiver Operating Characteristic (AUROC): sklearn.metrics.roc_auc_score()

A.7.4 Hyperparameters

In connectivity-constrained simulation data, for training NerFormer, we use window size 100, embedding size 200, hidden dimension of query and key matrices is 300, learning rate 10^{-3}, and batch size 32. For training RNN, we use $p = 1$, batch size 32, and learning rate 10^{-3}.

In real data, for training NetFormer, we use window size 60, embedding size 30, hidden dimension of query and key matrices 90, learning rate 10^{-3}, and batch size 32. For training RNN, we use $p = 1$, batch size 32, and learning rate 10^{-4}.

We use PyTorch [46] and PyTorch Lightning [47] for model development and training, and Adam as the optimizer.

A.7.5 Pseudo Code

We train NetFormer and extract attentions and positional embeddings for connectivity inference and binary cell-type classification. The pseudo code for model training , connectivity inference and cell-type classification is provided as follows:

```
NetFormer(x, neuron_ids):
    if constraint == True:
        cell_type_level_mean = parameters(num_cell_type, num_cell_type)
        cell_type_level_var = parameters(num_cell_type, num_cell_type)

    embeddings = embedding_table(neuron_ids)
    input = layer_norm(concat(x, embeddings))
    x, embeddings = input[:, :, :T], input[:, :, T:]

    dim_x, dim_e = x.shape[-1], embeddings.shape[-1]
    scale = (dim_x + dim_e) ** -0.5

    logits = input @ W_Q_W_KT @ input.T
    logits = logits * scale

    if activation == softmax:
            attention = softmax(logits)
        elif activation == sigmoid:
            attention = sigmoid(logits)
        elif activation == tanh:
            attention = tanh(logits)
        elif activation == none:
            attention = logits

    output = layer_norm(attention @ x + x)

    if out_layer == True:
        # linear_out is a lienar transformation from dimension T to 1
        output = linear_out(output)
        return output, attention
    else:
        # Use the last column as prediction
        return output[:, :, -1], attention


NetFormer_Training(all_samples):
    all_inputs, all_neuron_ids, all_GT_targets = all_samples
    model = NetFormer()
    optimizer = Adam(model, learning_rate)

    all_predictions, all_attentions = model(all_inputs, all_neuron_ids)

    prediction_loss = MSE(all_predictions, all_GT_targets)
    loss = prediction_loss

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()


Connectivity_Inference(all_samples, trained_NetFormer, GT_connectivity):
```

```
    all_inputs, all_neuron_ids, all_GT_targets = all_samples
    all_predictions, all_attentions = trained_NetFormer(all_inputs, all_neuron_ids)

    avg_attention = mean(all_attentions, axis=0)

    pearson_corr = pearsonr(GT_connectivity, avg_attention)
    spearman_corr = spearmanr(GT_connectivity, avg_attention)


Cell_Type_Classification(trained_NetFormer, neuron_ids, cell_types):
    embeddings = trained_NetFormer.embedding_table(neuron_ids)

    X_train = embeddings[TRAIN_idx]
    y_train = cell_types[TRAIN_idx]
    X_test = embeddings[TEST_idx]
    y_test = cell_types[TEST_idx]

    # Train classifier
    classifier = LogisticRegression.fit(X_train, y_train)
    # Test on test set
    y_pred = classifier.predict(X_test)
```

A.8 Compute Resources

In Section 2, model training was done on a MacBook Pro with Apple M1 chip. In Section 3, using the NVIDIA A100 GPU, NetFormer model trained on connectivity-constrained simulation data took about 10min. NetFormer model trained on one mouse (SB025) in real mouse data took about 20min, which requires at least 60 GB of RAM and 16 GB of GPU memory.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and precede the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS paper checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We validate all claims with mathematical analysis and experimental results.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Please see our discussion section.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: In section 2 we provided a mathematical analysis on the connection between our model and the classical forward Euler method. In the main text, we noted explicitly that the theorical analysis may not extend easliy to nonlinear systems with inputs outside the convergence region. Meanwhile, we provided empirical evidence with an additional simulation example in A.3.4 to demonstrate that our model can still be applicable in this setting.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

In section 2 and model architecture figure 1, we included all details about the model architecture and the toy simulation. Appendix A.3.1 provided the mathematical formula for simulating the toy data, model architecture, and model training. Appendix A.4 described another simulation data, two real mouse datasets, and data preprocessing steps, while Appendix A.7 contains implementation details for modeling and evaluating on these datasets.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We have not yet released our code but plan to in the future. All data are publicly available.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.

- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: To the extent possible, we state all experimental setting and details including hyperparameters, architectures, optimizers, etc.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Where possible, we state sensitivity analysis in our results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: In Appendix A.8, we mentioned the use of NVIDIA A100 GPUs. We indicated that each experiment on one mouse (SB025) requires at least 60 GB of RAM and 16 GB of GPU memory and takes about 20min, while each experiment on connectivity-constrained simulation data takes about 10min.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: Our work does not raise any ethical concerns, and conforms with all ethical guidelines.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: This work studies fundamental concepts in machine learning and neuroscience, and we do not believe it has any immediate societal consequences.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We are not releasing high-risk models or data with this work.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have credited all those involved with the collection of the data we utilize.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: No new assets are introduced in this work.

    Guidelines:

    - The answer NA means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: No human subjects were involved in this work.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: No human subjects were involved in this work.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.