Graph neural networks uncover structure and functions underlying the activity of simulated neural assemblies

Cédric Allier, Larissa Heinrich, Magdalena Schneider, and Stephan Saalfeld ✉

Janelia Research Campus, Howard Hughes Medical Institute, Ashburn, VA 20147, USA.

✉ saalfelds@janelia.hhmi.org

January 16, 2026

Abstract

Graph neural networks trained to predict observable dynamics can be used to decompose the temporal activity of complex heterogeneous systems into simple, interpretable representations. Here we apply this framework to simulated neural assemblies with thousands of neurons and demonstrate that it can jointly reveal the connectivity matrix, the neuron types, the signaling functions, and in some cases hidden external stimuli. In contrast to existing machine learning approaches such as recurrent neural networks and transformers, which emphasize predictive accuracy but offer limited interpretability, our method provides both reliable forecasts of neural activity and interpretable decomposition of the mechanisms governing large neural assemblies.

1 Introduction

We have shown that GNNs can decompose complex dynamical systems into interpretable representations (**allier˙decomposing˙2024**). We jointly learned pairwise interaction functions and local update rules together with a latent representation of the different objects present in the system, and an implicit representation of external stimuli. This approach can resolve the complexity arising from heterogeneity in large N-body systems that are affected by complex external inputs. Here, we leverage this technique to model the simulated activity of large heterogeneous neural assemblies. We retrieve the connectivity matrix, neuron types, signaling functions, local update rules, and external stimuli from activity data alone, yielding a fully functional mechanistic approximation of the original network with excellent roll-out performance.

Prior studies were successful in retrieving functional connectivity, functional properties of neurons, or predictive models of neural activity at the population level, but a method to infer an interpretable mechanistic model of complex neural assemblies is yet missing.

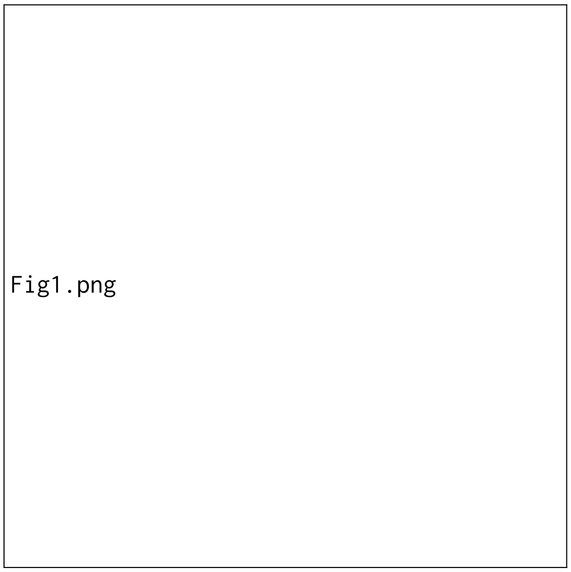


Figure 1: The temporal activity of a simulated neural network (**a**) is converted into densely connected graph (**b**) processed by a message passing GNN (**c**). Each neuron (node i) receives activity signals x_j from connected neurons (node j), processed by a transfer function ψ^* and weighted by the matrix $\boldsymbol{W}$. The sum of these messages is updated with functions ϕ^* and Ω^* to obtain the predicted activity rate $\widehat{\dot{x}}_i$. In addition to the observed activity x_i, the GNN has access to learnable latent vectors $\boldsymbol{a}_i$ associated with each node i.

mi˙connectome-constrained˙2021 infer the hidden voltage dynamics and functional connectivity from calcium recordings of parts of the *C. elegans* nervous system using a simple biophysical neuron model. While its predictive performance is modest, it demonstrates that integrating real activity recordings with connectivity can yield experimentally testable predictions beyond simulation. **pospisil˙fly˙2024** used the recently published connectome of *Drosophila melanogaster* (**dorkenwald˙neuronal˙2024**) to create simulations of full brain activity and showed that they can infer effective connectivity from known perturbations of individual neurons with a simple linear model. Lacking information about neural connectivity, **wang˙foundation˙2025** trained a foundation model of mouse visual cortex activity on real two-photon recordings from 135,000 neurons, which predicts responses to both natural videos and out-of-distribution stimuli across animals.

Our method learns the functional properties of individual neurons and their connections, enabling transfer of the learned dynamics to different network configurations with modified connectivity or different neuron type compositions.

2 Methods

Simulation of neural assemblies

We simulated the activity of neural assemblies with the model described by **stern˙reservoir˙2023**. We represent the neural system as a graph of N nodes where nodes correspond to neurons and edges represent weighted synaptic connections. The activity signal x_i represents continuous neural dynamics. Each neuron (node i) receives activity signals from connected neurons (nodes j) and updates its own activity x_i according to

$$\dot{x}_i = -\frac{x_i}{\tau_i} + s_i \phi\left(x_i\right) + g_i \sum_{\mathrm{j}=1}^{\mathrm{N}} \boldsymbol{W}_{ij} \psi_{ij}\left(x_j\right) + \eta_i\left(t\right). \qquad (1)$$

These systems can generate activity across a wide range of time scales similar to what is observed between cortex regions. The damping effect (first term) is parameterized by τ, the self-coupling (second term) is parameterized by s, and g scales the aggregated messages. The matrix $\boldsymbol{W}$ contains the synaptic weights multiplying the transfer function $\psi_{ij}(x_j)$. The weights were drawn from a Cauchy distribution with $\mu = 0$ and $\sigma^2 = \frac{1}{N}$. A positive and negative sign of the weights indicates excitation and inhibition, respectively. The last term $\eta_i(t)$ is Gaussian noise with zero mean. In our experiments, the number of neurons N was 1,000 or 8,000. We set g_i to 10, and used values between 0.25 and 8 for τ and s to test different self-coupling regimes as suggested by **stern˙dynamics˙2014**. First, we chose $\tanh(x)$ for both $\phi(x)$ and $\psi_{ij}(x)$. Later, we made the function ψ_{ij} dependent on the neuron or the interaction between two neurons by changing $\psi_{ij}(x_j)$ to $\tanh(\frac{x_j}{\gamma_i})$ or $\tanh(\frac{x_j}{\gamma_i}) - \theta_j x_j$, respectively, with γ and θ parameterizing different neuron types. Finally, we introduced external stimuli into the dynamics through a time-dependent function $\Omega_i(t)$ that scales the aggregated messages. The model used in our simulations is therefore

$$\begin{aligned} \dot{x}_i = & -\frac{x_i}{\tau_i} + s_i \tanh\left(x_i\right) \\ & + g_i \Omega_i\left(t\right) \sum_{\mathrm{j}=1}^{\mathrm{N}} \boldsymbol{W}_{ij} \left(\tanh\left(\frac{x_j}{\gamma_i}\right) - \theta_j x_j\right) + \eta_i\left(t\right). \end{aligned} \qquad (2)$$

In **??**, we list the parameters used for each experiment.

2.1 Graph neural networks

?? depicts the components of the GNNs trained on simulated data. The GNN learns the update rule

$$\widehat{\dot{\boldsymbol{x}}}_i = \phi^*\left(\boldsymbol{a}_i, x_i\right) + \Omega_i^*\left(t\right) \sum_{\mathrm{j}=1}^{\mathrm{N}} \boldsymbol{W}_{ij} \psi^*\left(\boldsymbol{a}_i, \boldsymbol{a}_j, x_j\right).$$

The optimized neural networks are ϕ^*, ψ^*, modeled as MLPs (ReLU activation, hidden dimension = 64, 3 layers, output size = 1), and Ω^* modeled as a coordinate-based MLP (**sitzmann˙implicit˙2020**). Other learnables are the two-dimensional latent vector $\boldsymbol{a}_i$ associated with each neuron, and the connectivity matrix $\boldsymbol{W}$. The optimization loss is

$$\begin{aligned} L = & \sum_{i=1}^{N} \|\widehat{\dot{\boldsymbol{x}}}_i - \dot{\boldsymbol{x}}_i\|^2 + \alpha \sum_{i=1}^{N} \|\phi^*\left(\boldsymbol{a}_i, 0\right)\|^2 \\ & + \beta \sum_{i=1}^{N} \|\mathrm{ReLU}\left(\frac{\partial \phi^*}{\partial x}\left(\boldsymbol{a}_i, x_i\right)\right)\|^2 \\ & + \gamma \sum_{i=1}^{N} \sum_{j=1}^{N} \|\mathrm{ReLU}\left(-\frac{\partial \psi^*}{\partial x}\left(\boldsymbol{a}_i, \boldsymbol{a}_j, x_j\right)\right)\|^2 + \zeta \|\boldsymbol{W}\|. \end{aligned}$$

The first term is the prediction error with $\widehat{\dot{\boldsymbol{x}}}_i$ being the GNN prediction, the second term encourages the steady state to be zero, the third term encourages exponential decay to avoid runaway excitations, the fourth term prevents ambiguity about the sign of the connectivity matrix, and the last term encourages sparsity of the connectivity matrix $\boldsymbol{W}$. In our experiments, we use different combinations of these regularization terms (**??**).

We implemented the GNNs using the PyTorch Geometric library (**fey˙fast˙2019**) and used AdamUniform gradient descent, with a learning rate of 10^{-4}. Each GNN was trained over 100 to 200 epochs, each epoch covering typically 10^5 time-points.

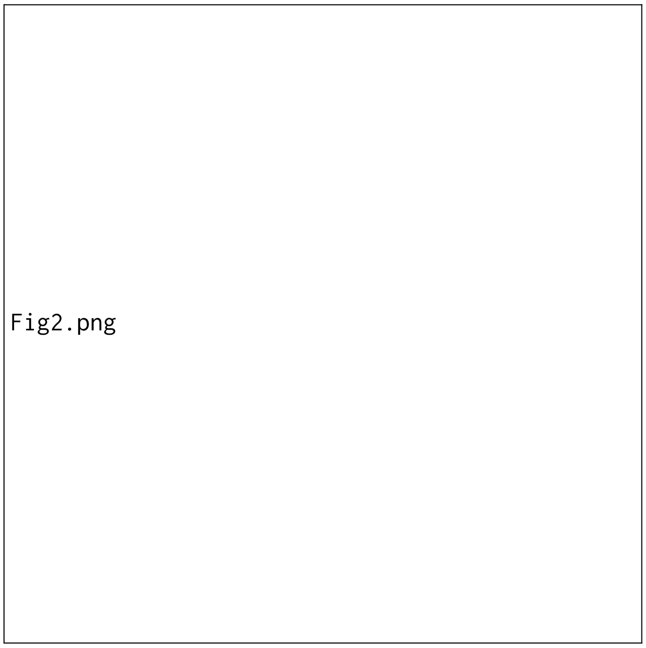


Figure 2: 1,000 densely connected neurons with 4 neuron-dependent update functions. (**a**) Activity time series used for GNN training. This dataset contains 10^5 time-points. (**b**) Sample time series taken from (**a**). (**c**) True connectivity $\boldsymbol{W}_{ij}$. The inset shows 20 × 20 weights. (**d**) Learned connectivity. (**e**) Comparison of learned and true connectivity (given $g_i = 10$ in **??**). (**f**) Learned latent vectors $\boldsymbol{a}_i$ of all neurons. Colors correspond to different neuron types. (**g**) Learned update functions $\phi^*(\boldsymbol{a}_i, x_i)$. The plot shows 1000 overlaid curves, one for each vector $\boldsymbol{a}_i$. True functions are overlaid in light gray. (**h**) Learned transfer function $\psi^*(x_i)$, normalized to a maximum value of 1. Colors indicate true neuron types. True functions are overlaid in light gray.

To improve neuron type clustering in the learned latent space, we used a modified version of our heuristic training schedule (**allier˙decomposing˙2024**).

3 Results

First, we simulated a noise-free neural network consisting of 1,000 densely connected neurons of four different types, each parameterized with different values of τ_i and s_i in **??** for 10^5 time points. **??** shows the simulated training series, and the results of the trained GNN. It successfully recovered the connectivity matrix ($R^2 = 1.00$, slope = 0.99 given $g_i = 10$), the common transfer function ψ, and the four distinct neuron-specific update functions $\phi^*(\boldsymbol{a})$. **??** visualizes the joint optimization of shared $\phi*$ and neuron-dependent $\boldsymbol{a}_i$. Neurons with identical latent parameters (neuron types) eventually form tight clusters of $\boldsymbol{a}$ leading to accurate approximations of the underlying function for each type.

Symbolic regression (PySR package (**cranmer˙interpretable˙2023**)) over samples generated by ϕ^* allowed us to retrieve their exact symbolic expression (**??**). K-means clustering of the learned latent vectors $\boldsymbol{a}_i$ recovered the neuron types with a classification accuracy of 1.00. We performed clustering for $K = 2, ..., 10$, and selected the one that yielded the maximum silhouette score (using the Euclidean distance). Increasing the dimension of the latent space did not improve the results, whereas using a one-dimensional latent space was detrimental (data not shown).

Then, to assess generalization capabilities, we performed rollout inference using the trained GNN model with initial activity values not seen during training (**??**). The trained GNN predicted the activity of 1,000 neurons for up to 400 time-points with excellent accuracy ($R^2 = 0.94$, slope = 1.00). The accuracy degrades at 800 time-points ($R^2 = 0.36$, slope = 0.9) due to error accumulation. To assess the importance of learning latent neuron types, we trained a GNN with fixed $\boldsymbol{a}_i$ on the same data (**??**) such that the learned update function was no longer neuron-dependent. This GNN recovered the connectivity matrix less well ($R^2 = 0.92$, slope = 0.93) and rollout accuracy was substantially worse (**??**). This suggests that models that ignore the heterogeneity of neural populations are poor approximations of the underlying dynamics of such systems.

Next, we evaluated generalization to fundamentally different network architectures. We changed the relative proportions of neuron types and the sparsity of the connectivity matrix. The modified GNN model still yielded excellent rollout results (**????**).

Performance scales with the length of the training series. Systematic sub-sampling of the 10^5 training series showed that results did not substantially improve beyond $\approx 2 \times 10^4$ time points for this 1,000-neuron system (**??**).

Reducing the density of the connectivity matrix leads to an effective reduction of informative training samples. We therefore tested performance for connectivity matrices with 5% to 100% non-zero connections. Even 5% connectivity matrices $\boldsymbol{W}$ could be recovered accurately if we enforced sparsity with an L_1-penalty on $\boldsymbol{W}$ (**????**). If all or some entries known to be zero can be masked, the results are even more robust (data not shown).

To assess robustness, we injected Gaussian noise into the simulated dynamics (**??**) to obtain an SNR of ∼10 dB. At this noise level, training remained stable and we recovered both the connectivity matrix and the signaling functions (**??**).

Next, we tested the ability to recover larger connectivity matrices, i.e. when the number of neurons was increased to 8,000. **??** shows that the 64 million weights of the connectivity matrix in a simulation with

Fig3.png

Figure 3: 2,048 densely connected neurons with different neuron-dependent update and transfer functions (4 neuron types), in the presence of external inputs. The training dataset contains 10^5 time points. (**a**) External inputs are represented by a time-dependent scalar field $\Omega_i(t)$ that scales the connectivity matrix $\boldsymbol{W}_{ij}$ (**??**). 1024 neurons (left), spatially ordered, are modulated by this field. The other 1024 neurons (right) are not affected ($\Omega_i = 1$). (**b**) Activity time values. (**c**) Sample of 10 time series used for training. (**d**) Comparison of learned and true connectivity $\boldsymbol{W}_{ij}$ (given $g_i = 10$ in **??**). (**e**) Comparison of learned and true $\Omega_i(t)$ values. (**f**) True field $\Omega_i(t)$ plotted at different time-points. (**g**) Learned field $\Omega_i^*(t)$ plotted at different time-points.

Gaussian noise (SNR of ~16 dB) were well recovered ($R^2 = 1.00$, slope= 1.00). Similarly, increasing the number of neuron types yielded excellent results (**??**, 32 different update functions, classification accuracy of 0.99).

We then introduced neuron-specific transfer functions of the form $\psi\big(x_j/\gamma_i\big)$ (**??**). Jointly optimizing the shared MLP ψ^* and the latent vectors $\boldsymbol{a}_i$ accurately identified these functions; symbolic regression recovered their analytical expressions (**??**). The learned connectivity closely matched the ground truth ($R^2 = 0.99$, slope = 0.99).

Next, we examined neuron–neuron–specific transfer functions $\psi\big(x_j/\gamma_i - \theta_i x_j\big)$ (**??**). Here, the multivariate MLP ψ^* takes pairwise latents $\big(\boldsymbol{a}_i, \boldsymbol{a}_j\big)$ as inputs (**??**). The four neuron types were well recovered, as were the four corresponding update functions and the 16 pairwise transfer functions. Symbolic regression yielded close approximations (**??**). The learned connectivity again matched the ground truth ($R^2 = 0.99$, slope= 1.03).

As neural networks in nature do not work in isolation but process external inputs, we tested whether we could recover both network structure and dynamics, as well as unknown external inputs in a final experiment. We introduced a time evolving function $\Omega_i(t)$ in the simulation (**??**), being spatially defined on a grid of 1,024 neurons (**??**). During training, $\Omega_i(t)$ is approximated by a coordinate-based MLP $\Omega^*(x, y, t)$. In addition to the grid of 1,024 neurons impacted by external stimuli, the simulation includes another set of 1,024 neurons for which $\Omega_i = 1$. The external inputs were well recovered ($R^2 = 0.99$, slope= 1.02), together with the connectivity matrix ($R^2 = 0.99$, slope= 0.94), neuron types, and signaling functions (**??**).

4 Discussion

To enable future applications to experimental data that captures neural activity of large complex networks (**lueckmann˙zapbench˙2025**), we designed simulations of complex neural assemblies. We covered systems with diverse functional connectivity, different neuron types affecting signal transfer between neurons and internal state update, and external inputs. With these simulations, we showed that message-passing GNNs can be used to recover the structure, functions, and the external inputs underlying the observed activity.

Our GNN-based approach is immediately interpretable, because it models the inputs, outputs, and behavior of individual neurons, and their functional connectivity. In our experiments, the learned latent embedding of individual neurons is two-dimensional, offering an excellent visualization that can be used to analyze the structure of the underlying parameterization (e.g. discrete neuron types vs. continuous parameterization). The signaling functions are approximated by simple MLPs that can be used as sample generators for symbolic regression to recover their analytical expressions.

We showed that the joint optimization of simple shared MLPs and low-dimensional latent embeddings for individual neurons is a powerful tool to account for the variability present in neural assemblies. It allows to distinguish different types of neurons, and to reveal their signaling functions, even in the presence of external inputs.

GNNs are an excellent tool to model known properties of the structure and function of complex dynamical systems such as neural networks. If partial knowledge about the connectivity of neurons is available, the connectivity matrix can be masked, neurons known to be of the same type can be forced to share learnable embeddings, expectations about the structure of the signaling and update functions can be used to constrain what the model can learn.

While our simulations capture core features of neural dynamics, biological applications will require additional features. Therefore, in future work, we will add time-dependent connectivity, signaling functions, and neural embeddings, time delays, diffusive exchange of chemical signals, and the fact that the neural dynamics are not directly observed but the result of a poorly characterized indirect biophysical process.

5 Conclusion

Our approach demonstrates the potential of GNNs to model complex neural activity, recovering key components such as connectivity matrices, signaling functions, and external inputs from observed dynamics alone. The method effectively decomposes the observed complexity into simple, interpretable components. Future work will focus on improving efficiency and incorporating additional key features of neural activity, such as changing connectivity over time and time delays, to enhance its application to experimental data.

Code Availability

The code is available at `https://github.com/saalfeldlab/NeuralGraph`. The repository includes demo scripts that reproduce the key results presented in this paper. `demo_1.py` reproduces **??**, training a GNN on 1000 densely connected neurons with 4 neuron types to recover the connectivity matrix, latent embeddings, and signaling functions. `demo_2.py` reproduces **??**, demonstrating the recovery of external inputs $\Omega_i(t)$ in addition to network structure for 2048 neurons.

Acknowledgements

We thank Tosif Ahamed and Tirthabir Biswas for helpful discussions and comments that improved this work and manuscript.

A Supplementary notes

A.1 Neuron type clustering used during training

In the main training loop, we jointly train the parameters of the MLPs ϕ^* and ψ^*, the coordinate-based network Ω^*, and the latent vectors $\boldsymbol{a}_i$ for each neuron. This does not guarantee that similar learned functions ϕ^* and ψ^* are produced by similar latent vectors $\boldsymbol{a}_i$, and vice versa. To encourage such a well-behaved mapping, we repeatedly cluster and re-initialize the vectors $\boldsymbol{a}$ for all neurons and retrain the corresponding function ϕ^*. We perform this clustering every 4 epochs of training:

Step 1: Sample function profiles

$$\boldsymbol{F}_i = \phi^*(\boldsymbol{a}_i, x_j), \quad x_j \in [-5, 5], \quad \text{1000 samples}$$

Step 2: Project profiles to 2D with UMAP

$$\boldsymbol{z}_i = \text{UMAP}(\boldsymbol{F}_i) \in \mathbb{R}^2$$

Step 3: Hierarchical clustering c of $\boldsymbol{z}_i$ (complete linkage with Euclidean distance threshold 0.01), resulting in m clusters

$$C_k = \{i : c(i) = k\}$$

Step 4: Define target functions

$$f_k^*(x) = \underset{i \in C_k}{\text{median}}(\phi^*(\boldsymbol{a}_i, x))$$

Step 5: Replace latent vectors with cluster medians

$$\boldsymbol{a}'_k = \underset{i \in C_k}{\text{median}}(\boldsymbol{a}_i), \quad \boldsymbol{a}_i \leftarrow \boldsymbol{a}'_k \quad \forall i \in C_k$$

Step 6: Retrain ϕ^* with loss

$$\sum_{k=1}^{m} \left\| \phi^*(\boldsymbol{a}'_k, x) - f_k^*(x) \right\|^2$$

for 20 epochs of 1000 samples $x \in [-5, 5]$.

The model used in our simulations is

$$\begin{aligned}\dot{x}_i = & -\frac{x_i}{\tau_i} + s_i \tanh(x_i) \\ & + g_i \Omega_i(t) \sum_{\mathrm{j=1}}^{\mathrm{N}} \boldsymbol{W}_{ij} (\tanh(\frac{x_j}{\gamma_i}) - \theta_j x_j) + \eta_i(t).\end{aligned} \tag{1}$$

Following table summarizes the simulation parameters.

Name	**Figure**	$\mathbf{N}_{\text{frames}}$	$\mathbf{N}_{\text{neurons}}$	$\mathbf{N}_{\text{types}}$	**Conn.**	$\boldsymbol{\sigma^2}$	$\boldsymbol{\Omega}$	$\mathbf{g_i}$	$\mathbf{s_i}$	$\boldsymbol{\tau_i}$	$\boldsymbol{\gamma_j}$	$\boldsymbol{\theta_j}$
Baseline	2, Supp. 1-7	100,000	1,000	4	100%	0	no	10	1,2	0.5,1	1	0
External inputs	3, Supp. 15	100,000	2,048	4	100%	1	yes	10	1,2	0.5,1	1,2,4,8	0
Sparse	Supp. 8-9	100,000	1,000	4	5%	0	no	10	1,2	0.5,1	1	0
High noise	Supp. 10	100,000	1,000	4	100%	7.2	no	10	1,2	0.5,1	1	0
Large scale	Supp. 11	100,000	8,000	4	100%	1	no	10	1,2	0.5,1	1	0
Many types	Supp. 12	100,000	1,000	32	100%	0	no	10	1-8	0.25-1	1	0
Transmitters	Supp. 13	100,000	1,000	4	100%	0	no	10	1,2	0.5,1	1,2,4,8	0
Transmitters & receptors	Supp. 14	100,000	1,000	4	100%	0	no	10	1,2	0.5,1	1,2,4,8	0-0.040

Supplementary Table 1: Simulation parameters. Connectivity indicates percentage of non-zero W_{ij}. Noise σ^2 is variance of $\eta_i(t)$ in **??**. Ω indicates presence of external inputs $\Omega_i(t)$. Parameters: g_i (coupling strength), s_i (self-coupling), τ_i (time constant), γ_j (scale in $\tanh(x_j/\gamma_i)$), θ_j (linear term in $\tanh(x_j/\gamma_i) - \theta_j x_j$).

The GNN learns the update rule

$$\widehat{\dot{x}}_i = \phi^*(\boldsymbol{a_i}, x_i) + \Omega_i^*(t) \sum_{\mathrm{j=1}}^{\mathrm{N}} \boldsymbol{W}_{ij} \psi^*(\boldsymbol{a_i}, \boldsymbol{a_j}, x_j). \tag{2}$$

The optimized neural networks are ϕ^*, ψ^*, modeled as MLPs (ReLU activation, hidden dimension = 64, 3 layers, output size = 1), and Ω^* modeled as a coordinate-based MLP (**sitzmann˙implicit˙2020**). Other learnables are the two-dimensional latent vector $\boldsymbol{a}_i$ associated with each neuron, and the connectivity matrix $\boldsymbol{W}$. The optimization loss is

$$\begin{aligned}L = & \sum_{i=1}^{N} \|\widehat{\dot{x}}_i - \dot{x}_i\|^2 + \alpha \sum_{i=1}^{N} \|\phi^*(\boldsymbol{a_i}, 0)\|^2 \\ & + \beta \sum_{i=1}^{N} \|\mathrm{ReLU}(\frac{\partial \phi^*}{\partial x}(\boldsymbol{a_i}, x_i))\|^2 \\ & + \gamma \sum_{i=1}^{N} \sum_{j=1}^{N} \|\mathrm{ReLU}(-\frac{\partial \psi^*}{\partial x}(\boldsymbol{a_i}, \boldsymbol{a_j}, x_j))\|^2 + \zeta \|\boldsymbol{W}\|.\end{aligned} \tag{3}$$

Following table summarizes the training parameters.

Name	**Figure**	$\boldsymbol{\alpha}$	$\boldsymbol{\beta}$	$\boldsymbol{\gamma}$	$\boldsymbol{\zeta}$	$\boldsymbol{\psi^*}$ **input**
Baseline	2, Supp. 1-7	1	0	0	0	x_j
External inputs	3, Supp. 15	1	5	10	10^{-5}	a_j, x_j
Sparse	Supp. 8-9	1	0	0	10^{-5}	x_j
High noise	Supp. 10	1	0	0	0	x_j
Large scale	Supp. 11	1	0	0	$5 \cdot 10^{-5}$	x_j
Many types	Supp. 12	1	0	0	0	x_j
Transmitters	Supp. 13	1	0	100	0	a_j, x_j
Transmitters & receptors	Supp. 14	1	0	500	0	a_i, a_j, x_j

Supplementary Table 2: Training parameters for loss function (**??**).

function	**true**	**learned**
ϕ_1	$-x + \tanh(x)$	$-0.998x + \tanh(x) - 0.0016$
ϕ_2	$-x + 2\tanh(x)$	$-0.998x + 1.996\tanh(x)$
ϕ_3	$-2x + \tanh(x)$	$-1.994x + \tanh(x)$
ϕ_3	$-2x + 2\tanh(x)$	$-1.996x + 1.997\tanh(x)$
ψ	$\tanh(x)$	$\tanh(x)$

Supplementary Table 3: Comparison of true and learned functions plotted in **??**. Symbolic regression (PySR package (**cranmer˙interpretable˙2023**)) is applied to the learned functions to retrieve their expressions.

function	**true**	**learned**
ϕ_1	$-x + \tanh(x)$	$-0.999x + \tanh(x)$
ϕ_2	$-x + 2\tanh(x)$	$-0.999x + 1.992\tanh(x)$
ϕ_3	$-2x + \tanh(x)$	$-1.994x + \tanh(x)$
ϕ_3	$-2x + 2\tanh(x)$	$-1.984x + 1.991\tanh(x)$
ψ_1	$\tanh(x)$	$\tanh(x)$
ψ_2	$\tanh(0.5x)$	$\tanh(0.489)x$
ψ_3	$\tanh(0.25x)$	$\tanh(0.247x)$
ψ_4	$\tanh(0.125x)$	$\tanh(0.128x)$

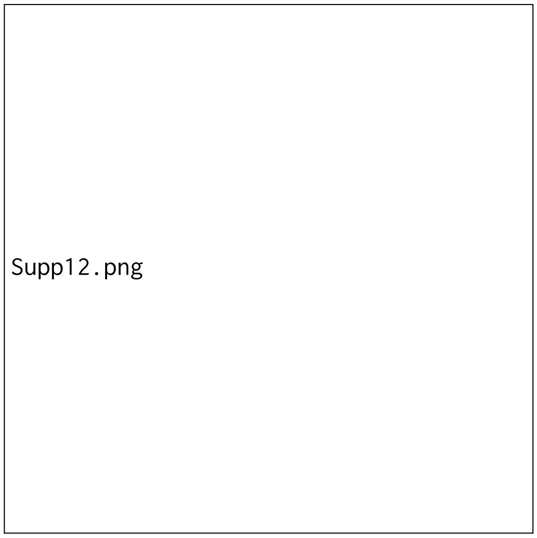
Supplementary Table 4: Comparison of true and learned functions plotted in **??**.

function	**true**	**learned**
ϕ_1	$-x + \tanh(x)$	$-0.999x + \tanh(x)$
ϕ_2	$-x + 2\tanh(x)$	$-0.993x + 1.999\tanh(x)$
ϕ_3	$-2x + \tanh(x)$	$-1.992x + \tanh(x)$
ϕ_3	$-2x + 2\tanh(x)$	$-1.974x + 1.989\tanh(x)$
ψ_{11}	$\tanh(x)$	$\tanh(x)$
ψ_{12}	$\tanh(x) - 0.013x$	$\tanh(x) - 0.017x$
ψ_{13}	$\tanh(x) - 0.027x$	$\tanh(x) - 0.028x$
ψ_{14}	$\tanh(x) - 0.040x$	$\tanh(x) - 0.053x$
ψ_{21}	$\tanh(0.5x)$	$\tanh(0.486)x$
ψ_{22}	$\tanh(0.5x) - 0.013x$	$\tanh(0.414x)$
ψ_{23}	$\tanh(0.5x) - 0.027x$	$0.814\tanh(0.603x)$
ψ_{24}	$\tanh(0.5x) - 0.040x$	$0.67\tanh(x)$
ψ_{31}	$\tanh(0.25x)$	$\tanh(0.222x)$
ψ_{32}	$\tanh(0.25x) - 0.013x$	$\tanh(0.204x)$
ψ_{33}	$\tanh(0.25x) - 0.027x$	$\tanh(0.163x)$
ψ_{34}	$\tanh(0.25x) - 0.040x$	$\tanh(0.172x)$
ψ_{41}	$\tanh(0.125x)$	$\tanh(0.118x)$
ψ_{42}	$\tanh(0.125x) - 0.013x$	$\tanh(0.110x)$
ψ_{43}	$\tanh(0.125x) - 0.027x$	$\tanh(0.097x)$
ψ_{44}	$\tanh(0.125x) - 0.040x$	$\tanh(0.081x)$

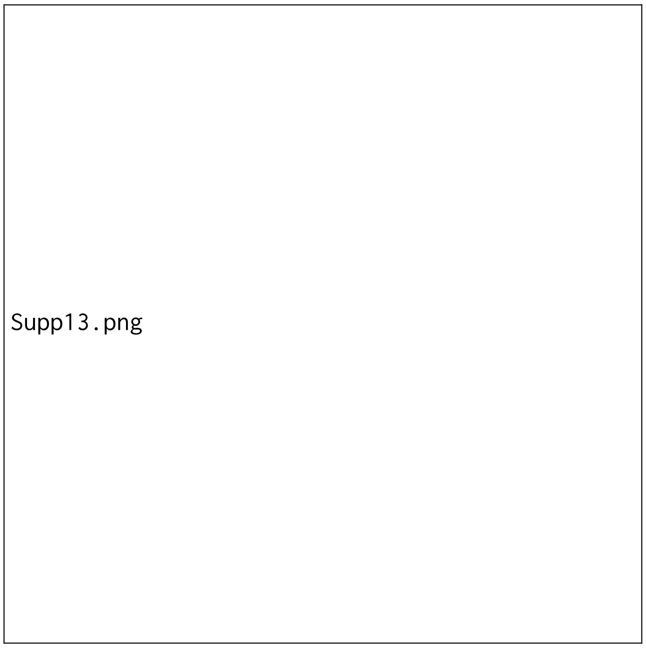
Supplementary Table 5: Comparison of true and learned functions plotted in **??**.

Supp1.png

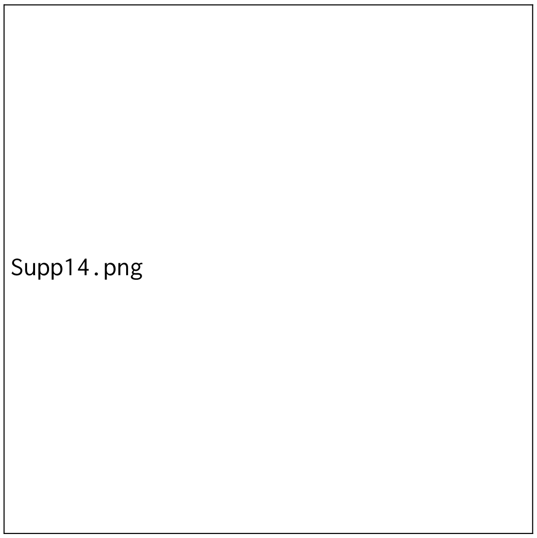
Supplementary Figure 1: 1,000 densely connected neurons with 4 neuron-dependent update functions. Results plotted over 20 epochs. (**a**) Learned latent vectors $\boldsymbol{a}_i$ of all neurons. (**b**) Learned update functions $\phi^*(\boldsymbol{a}, x)$. (**c**) Learned transfer function $\psi^*(x)$, normalized to a maximum value of 1. (**d**) Learned connectivity $\boldsymbol{W}_{ij}$. (**e**) Comparison of learned and true connectivity. Colors indicate true neuron types.



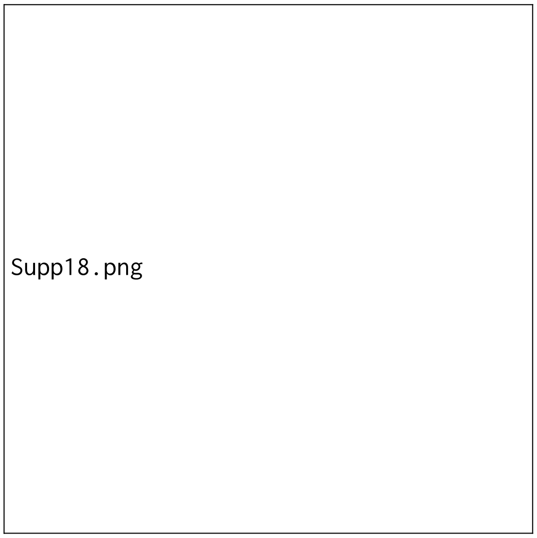
Supplementary Figure 2: Rollout inference performed with the GNN model trained with a simulation of 1,000 densely connected neurons (**??**). Results plotted at time step 400 and 800, respectively. (**a**) and (**c**) 25 learned activity traces plotted as a function of time-points. True activity traces are overlaid in light gray. (**b**) and (**d**) Comparison between true and learned activity values of 1000 neurons.



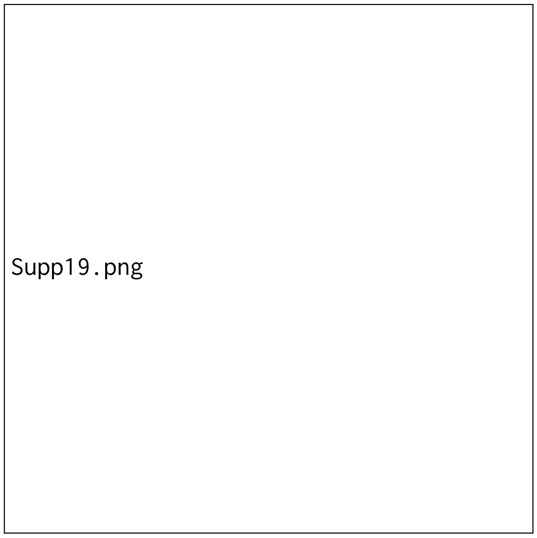
Supplementary Figure 3: 1,000 densely connected neurons with 4 neuron dependent update functions (first two terms in **??**). Results are obtained with a GNN trained with the hypothesis that all neurons are identical. To do so, we fixed the learnable latent vectors $\boldsymbol{a}_i$ to a unique vector. (**a**) Activity time series used for GNN training. This dataset contains 10^5 time-points. (**b**) Sample of 10 time series taken from (**a**). (**c**) True connectivity $\boldsymbol{W}_{ij}$. The inset shows 20 × 20 weights. (**d**) Learned connectivity. (**e**) Comparison of learned and true connectivity (given $g_i = 10$ in **??**). (**f**) Fixed latent vectors $\boldsymbol{a}_i$ of all neurons. (**g**) Learned update functions $\phi^*(\boldsymbol{a}, x)$. (**h**) Learned transfer function $\psi^*(x)$, normalized to a maximum value of 1. Colors indicate true neuron types. True functions are overlaid in light gray.



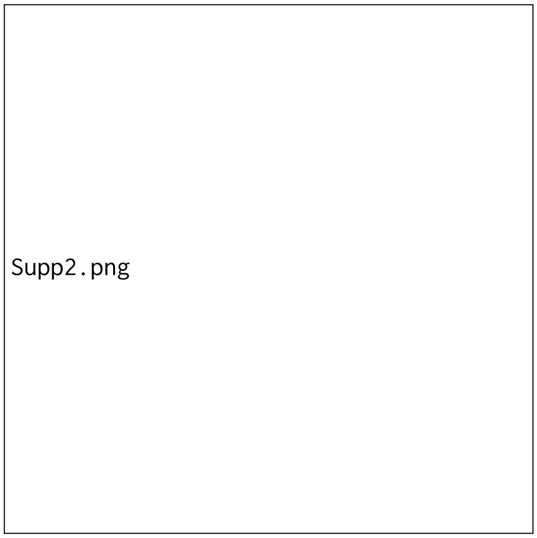
Supplementary Figure 4: Rollout inference performed with the GNN model trained with the hypothesis that all neurons are identical (**??**). Results plotted at time step 200 and 800, respectively. (**a**) and (**c**) 25 learned activity traces plotted as a function of time-points. True activity traces are overlaid in light gray. (**b**) and (**d**) Comparison between true and learned activity values of 1000 neurons.



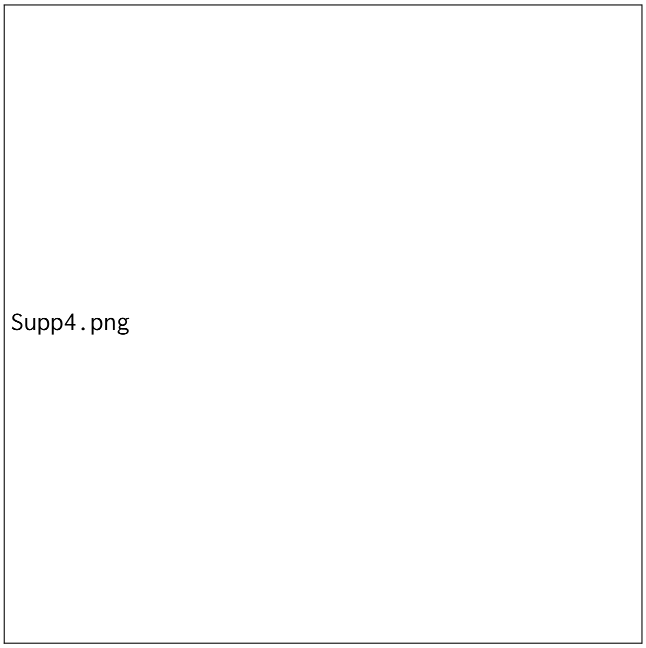
Supplementary Figure 5: Generalization test with modified network structure: performance evaluation after changing connectivity matrix and neuron type proportions. The GNN model was trained with 1,000 densely connected neurons. (**a**) Original relative proportions of neuron types (25% each). (**b**) Modified relative proportions of neuron types (10%, 20%, 30%, 40%). (**c**) Original connectivity matrix (10^6 weights, fully connected). (**d**) Modified sparse connectivity matrix (243,831 weights, ∼25% sparsity). (**e,f**) Rollout inference over 400 time-steps shows perfect performance ($R^2 = 1.0$, slope= 1.0). (**g,h**) Extended rollout over 800 time-steps maintains high accuracy ($R^2 = 0.996$, slope= 1.0).



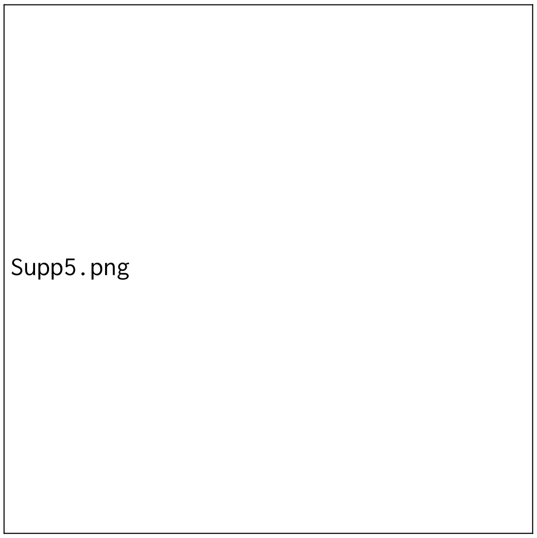
Supplementary Figure 6: Generalization test with modified network structure. (**a**) Original relative proportions of neuron types (25% each). (**b**) Modified relative proportions with only two types present (60%, 40%, types 2 and 3 eliminated). (**c**) Original connectivity matrix (10^6 weights, fully connected). (**d**) Modified sparse connectivity matrix (487,401 weights, 50% sparsity). (**e,f**) Rollout inference over 400 time-steps achieves excellent accuracy ($R^2 = 1.0$, slope= 1.0). (**g,h**) Extended rollout over 800 time-steps maintains high accuracy ($R^2 = 0.975$, slope= 0.99).



Supplementary Figure 7: 1,000 densely connected neurons with 4 neuron-dependent update functions. The plot displays R^2 for the comparison between true and learned connectivity matrices $\boldsymbol{W}_{ij}$ as a function of training epochs for different training dataset sizes (colors). Comparison is made at equal numbers of gradient descent iterations.



Supplementary Figure 8: 1,000 sparsely (5%) connected neurons with 4 neuron-dependent update functions. Results are obtained after 20 epochs. (**a**) Activity time series used for GNN training. The training dataset contains 10^5 time-points. (**b**) Time series of a sample of 10 representative neurons taken from (**a**). (**c**) True connectivity $\boldsymbol{W}_{ij}$. The inset shows 20 × 20 weights. (**d**) Learned connectivity. (**e**) Comparison of learned and true connectivity (given $g_i = 10$ in **??**). (**f**) Learned latent vectors $\boldsymbol{a}_i$ of all neurons. (**g**) Learned update functions $\phi^*(\boldsymbol{a}, x)$. (**h**) Learned transfer function $\psi^*(x)$, normalized to a maximum value of 1. Colors indicate true neuron types. True functions are overlaid in light gray.



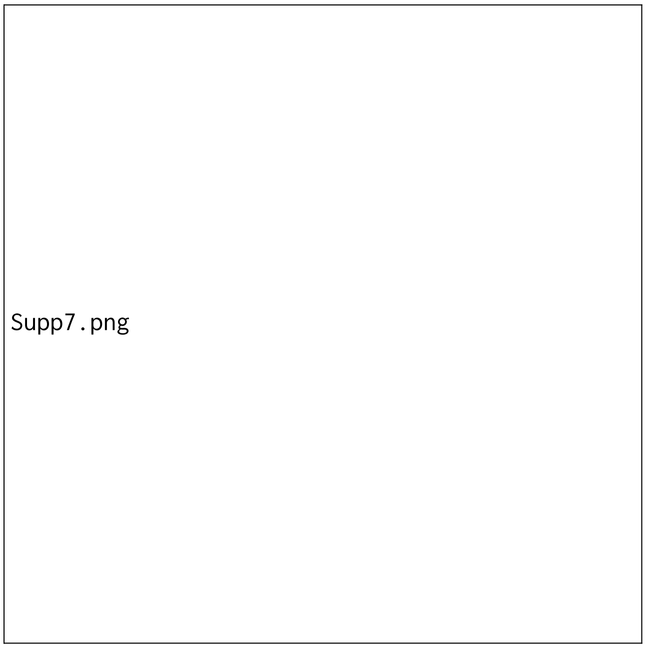
Supplementary Figure 9: 1,000 densely connected neurons with 4 neuron-dependent update functions. The plot displays R^2 for the comparison between true and learned connectivity matrices $\boldsymbol{W}_{ij}$ as a function of training epochs for different connectivity filling factors (colors). All comparisons are made at equal numbers of gradient descent iterations.

Supp6.png

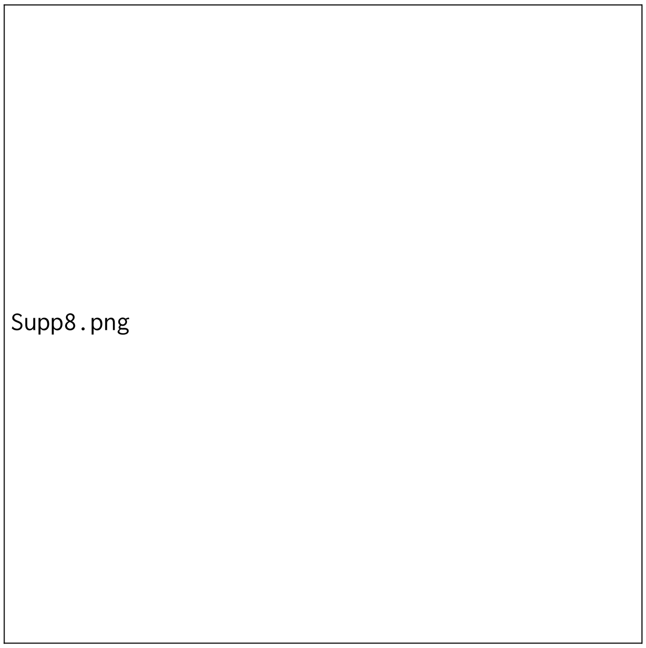
Supplementary Figure 10: 1,000 densely connected neurons with 4 neuron-dependent update functions in the presence of Gaussian noise (**??**). The signal-to-noise ratio is about 10 dB as measured by comparing filtered and raw signals. For comparison, corresponding signals without noise are shown in **??**. (**a**) Activity time series used for GNN training. The training dataset contains 10^5 time-points. (**b**) Sample of 10 time series taken from (**a**). (**c**) True connectivity $\boldsymbol{W}_{ij}$. The inset shows 20×20 weights. (**d**) Learned connectivity. (**e**) Comparison of learned and true connectivity (given $g_i = 10$ in **??**). (**f**) Learned latent vectors $\boldsymbol{a}_i$ of all neurons. (**g**) Learned update functions $\phi^*(\boldsymbol{a}, x)$. (**h**) Learned transfer function $\psi^*(x)$, normalized to a maximum value of 1. Colors indicate true neuron types. True functions are overlaid in light gray.

Supp3.png

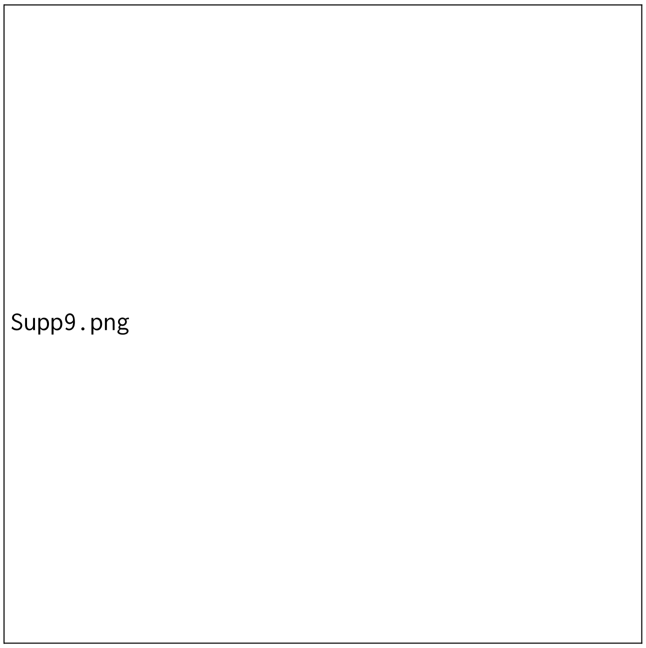
Supplementary Figure 11: 8,000 densely connected neurons with 4 neuron-dependent update functions. Results obtained after 14 epochs. (**a**) Activity time series used for GNN training. The training dataset contains 10^5 time-points. (**b**) Sample of 10 time series taken from (**a**). (**c**) True connectivity $\boldsymbol{W}_{ij}$. The inset shows 20×20 weights. (**d**) Learned connectivity. (**e**) Comparison of learned and true connectivity (given $g_i = 10$ in **??**). (**f**) Learned latent vectors $\boldsymbol{a}_i$ of all neurons. (**g**) Learned update functions $\phi^*(\boldsymbol{a}, x)$. (**h**) Learned transfer function $\psi^*(x)$, normalized to a maximum value of 1. Colors indicate true neuron types. True functions are overlaid in light gray.



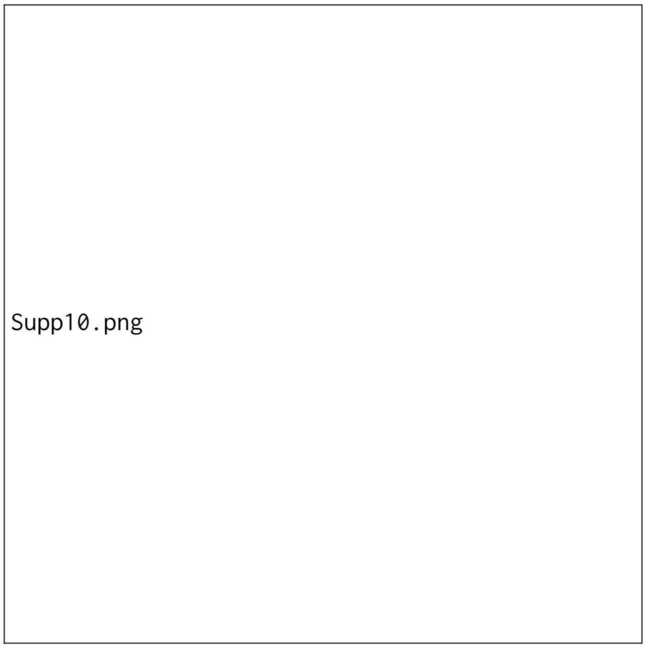
Supplementary Figure 12: 1,000 densely connected neurons with 32 neuron-dependent update functions. Results are obtained after 20 epochs. (**a**) Activity time series used for GNN training. The training dataset contains 10^5 time-points. (**b**) Sample of 10 time series taken from (**a**). (**c**) True connectivity $\boldsymbol{W}_{ij}$. (**d**) Learned connectivity. (**e**) Comparison between learned and true connectivity. (**f**) Learned latent vectors $\boldsymbol{a}_i$. (**g**) Learned update functions $\phi^*(\boldsymbol{a}, x)$. (**h**) Learned transfer functions $\psi^*(x)$. Colors indicate true neuron types. True functions are overlaid in light gray.



Supplementary Figure 13: 1,000 densely connected neurons with neuron-dependent update and transfer functions (4 neuron types). (**a**) Activity time series used for GNN training. The training dataset contains 10^5 time-points. (**b**) Sample of 10 time series taken from (**a**). (**c**) True connectivity $\boldsymbol{W}_{ij}$. (**d**) Learned connectivity. (**e**) Comparison between learned and true connectivity. (**f**) Learned latent vectors $\boldsymbol{a}_i$. (**g**) Learned update functions $\phi^*(\boldsymbol{a}, x)$. (**h**) Learned transfer functions $\psi^*(\boldsymbol{a}, x)$. Colors indicate true neuron types. True functions are overlaid in light gray.



Supplementary Figure 14: 1,000 densely connected neurons with neuron-dependent update and transfer functions (4 neuron types). (**a**) Activity time series used for GNN training. The training dataset contains 10^5 time-points. (**b**) Sample of 10 time series taken from (**a**). (**c**) True connectivity $\boldsymbol{W}_{ij}$. (**d**) Learned connectivity. (**e**) Comparison between learned and true connectivity. (**f**) Learned latent vectors $\boldsymbol{a}_i$. (**g**) Learned update functions $\phi^*(\boldsymbol{a}, x)$. (**h**) Learned transfer functions $\psi^*(\boldsymbol{a_i}, \boldsymbol{a_j}, x)$. Colors indicate true neuron types. True functions are overlaid in light gray.



Supplementary Figure 15: 2,048 densely connected neurons with neuron-dependent update and transfer functions (4 neuron types) in the presence of external stimuli. Results are obtained after 16 epochs. (**a**) Activity time series used for GNN training. The training dataset contains 10^5 time-points. (**b**) Sample of 10 time series taken from (**a**). (**c**) True connectivity $\boldsymbol{W}_{ij}$. (**d**) Learned connectivity. (**e**) Comparison between learned and true connectivity. (**f**) Learned latent vectors $\boldsymbol{a}_i$. (**g**) Learned update functions $\phi^*(\boldsymbol{a}, x)$. particleColors indicate true neuron types. True functions are overlaid in light gray.