Heterogeneity in Neuronal Dynamics Is Learned by Gradient Descent for Temporal Processing Tasks

Chloe N. Winston
wincnw@gmail.com
Departments of Neuroscience and Computer Science, University of Washington, Seattle, WA 98195, U.S.A., and University of Washington Computational Neuroscience Center, Seattle, WA 98195, U.S.A.

Dana Mastrovito
dana.mastrovito@alleninstitute.org
Allen Institute for Brain Science, Seattle, WA 98109, U.S.A.

Eric Shea-Brown
etsb@uw.edu
Stefan Mihalas
stefanm@alleninstitute.org
University of Washington Computational Neuroscience Center, Seattle, WA 98195, U.S.A.; Allen Institute for Brain Science, Seattle, WA 98109, U.S.A.; and Department of Applied Mathematics, University of Washington, Seattle, WA 98195, U.S.A.

Individual neurons in the brain have complex intrinsic dynamics that are highly diverse. We hypothesize that the complex dynamics produced by networks of complex and heterogeneous neurons may contribute to the brain's ability to process and respond to temporally complex data. To study the role of complex and heterogeneous neuronal dynamics in network computation, we develop a rate-based neuronal model, the generalized-leaky-integrate-and-fire-rate (GLIFR) model, which is a rate equivalent of the generalized-leaky-integrate-and-fire model. The GLIFR model has multiple dynamical mechanisms, which add to the complexity of its activity while maintaining differentiability. We focus on the role of after-spike currents, currents induced or modulated by neuronal spikes, in producing rich temporal dynamics. We use machine learning techniques to learn both synaptic weights and parameters underlying intrinsic dynamics to solve temporal tasks. The GLIFR model allows the use of standard gradient descent techniques rather than surrogate gradient descent, which has been used in spiking neural networks. After establishing the ability to optimize parameters using gradient descent in single neurons, we ask how networks of GLIFR neurons learn and perform on temporally challenging tasks, such as sequential MNIST. We find that these networks learn diverse parameters, which gives rise to

diversity in neuronal dynamics, as demonstrated by clustering of neuronal parameters. GLIFR networks have mixed performance when compared to vanilla recurrent neural networks, with higher performance in pixel-by-pixel MNIST but lower in line-by-line MNIST. However, they appear to be more robust to random silencing. We find that the ability to learn heterogeneity and the presence of after-spike currents contribute to these gains in performance. Our work demonstrates both the computational robustness of neuronal complexity and diversity in networks and a feasible method of training such models using exact gradients.

1 Introduction

1.1 Background. Artificial neural networks (ANNs) have been used to emulate the function of biological networks at a system level (Yamins et al., 2014; Rajan et al., 2016). Such models rely on the fantastic capacity of ANNs to be trained to solve a task via backpropagation, using tools developed by the machine learning community (Goodfellow et al., 2016). However, the way in which the neurons in recurrent ANNs integrate their inputs over time differs from neuronal computation in the brain (see Figures 1A and 1B).

Primarily, biological neurons are dynamic, constantly modulating internal states in a nonlinear way while responding to inputs. A biological neuron maintains a membrane potential that varies not only with input currents but also through history-dependent transformations. When its voltage exceeds some threshold, a neuron produces a spike, a rapid fluctuation in voltage that is considered the basis of neuronal communication. While these are the defining features of a neuron, neurons exhibit additional, more complex types of dynamics, including threshold variability and bursting. Threshold adaptation gives rise to threshold variability and thus modulates the sensitivity of a neuron to inputs. A proposed mechanism is that the threshold fluctuates in a voltage-dependent manner, possibly due to the gating mechanisms of various ion channels (Fontaine et al., 2014). Another type of dynamic is responsible for bursting and related spiking behaviors. Bursting results from depolarizing currents, which can be induced by prior spikes. A broader array of spiking patterns can be explained by considering both hyperpolarizing and depolarizing after-spike currents, currents that are modulated by a neuron's spiking activity (Mihalaş & Niebur, 2009).

In sum, biological neuronal dynamics can be thought of as continuous, nonlinear transformations of internal neural states, such as ionic currents and voltage. This is in contrast to a typical artificial neuron that maintains a single state resulting from a static, linear transformation of an input vector. Moreover, diversity in intrinsic neuronal mechanisms gives rise to heterogeneity in the dynamics that biological neurons express. For example, not all neurons in the brain display bursting behavior or threshold adaptation.

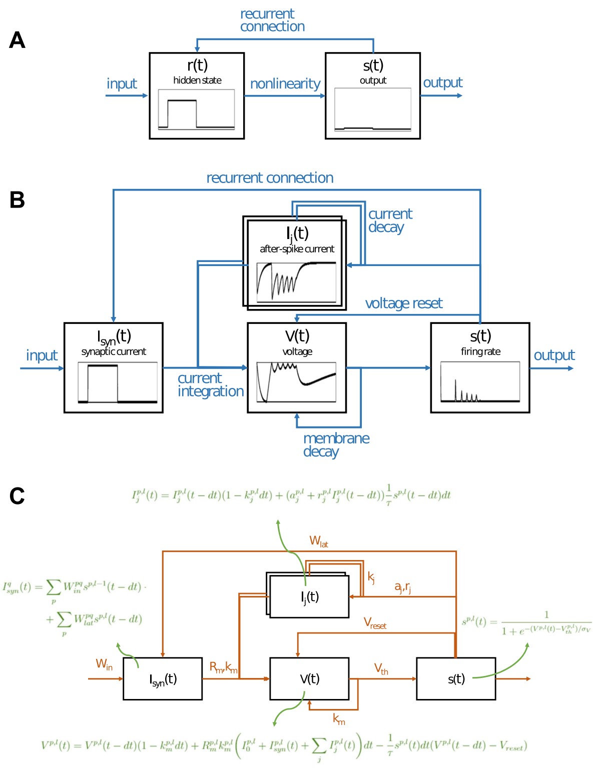

A
recurrent connection
input
r(t)
hidden state
nonlinearity
s(t)
output
output
B
recurrent connection
current decay
$I_j(t)$
after-spike current
voltage reset
input
$I_{syn}(t)$
synaptic current
current integration
V(t)
voltage
s(t)
firing rate
output
membrane decay
C
$I_j^{p,l}(t) = I_j^{p,l}(t-dt)(1-k_j^{p,l}dt) + (a_j^{p,l} + r_j^{p,l} I_j^{p,l}(t-dt))\frac{1}{\tau} s^{p,l}(t-dt)dt$
W_{lat}
k_j
a_j, r_j
$I_j(t)$
$I_{syn}^q(t) = \sum_p W_{in}^{pq} s^{p,l-1}(t-dt) \cdot + \sum_p W_{lat}^{pq} s^{p,l}(t-dt)$
V_{reset}
$s^{p,l}(t) = \frac{1}{1+e^{-(V^{p,l}(t)-V_{th}^{p,l})/\sigma_V}}$
W_{in}
$I_{syn}(t)$
R_m, k_m
V(t)
V_{th}
s(t)
k_m
$V^{p,l}(t) = V^{p,l}(t-dt)(1-k_m^{p,l}dt) + R_m^{p,l}k_m^{p,l}\left(I_0^{p,l} + I_{syn}^{p,l}(t) + \sum_j I_j^{p,l}(t)\right)dt - \frac{1}{\tau}s^{p,l}(t)dt(V^{p,l}(t-dt) - V_{reset})$

Figure 1: (A) Schematic of a "vanilla" RNN cell or neuron. An RNN neuron maintains a hidden state $r(t)$ that is computed at each timestep by linearly weighting the input signal and the previous output of itself and neighboring neurons through a recurrent connection. The output $S(t)$ is computed by applying a nonlinear transformation (e.g., ReLU, tanh, or sigmoid) to $r(t)$. (B, C) Schematics of a GLIFR neuron. Each neuron maintains a synaptic current I_{syn} that is computed at each timestep by linearly weighting the input signal by W_{in}, as well as the previous output of its neuron layer through a lateral or recurrent connection by W_{lat}. The neuron's voltage V decays over time according to membrane decay factor k_m and integrates synaptic currents and after-spike currents I_j over time based on membrane resistance R_m and the membrane decay factor. Additionally, the voltage tends toward V_{reset} through a continuous reset mechanism based on the firing rate at a given time. An exponential transformation of the difference between the voltage and the threshold voltage yields a continuous-valued normalized firing rate, which varies between 0 and 1. The normalized firing rate, along with terms a_j and r_j, is used to modulate the after-spike currents that decay according to decay factor k_j. The dynamics present in a GLIFR neuron give rise to its key differences from RNN neurons; GLIFR neurons can express heterogeneous dynamics, in contrast to the fixed static transformations utilized in RNN neurons.

The brain also exhibits diversity in neuronal properties, such as ionic conductances, threshold, and membrane properties. This diversity is evident from experiments fitting neuronal models to spiking behavior observed in the mouse cortex (Teeter et al., 2018). Heterogeneity of such neural characteristics further amplifies the diversity in neuronal dynamics. In contrast, the only diversity in typical recurrent neural networks (RNNs), ANNs that incorporate information over time, lies in synaptic weights. Typical RNN neurons do not exhibit intrinsic dynamics, and each neuron responds identically to input with a common, fixed activation function.

Vanilla RNNs, which employ linear weighting of inputs and nonlinear transformation of hidden states over time (see Figure 1A), have managed to perform quite well on complex tasks without incorporating the complexity and heterogeneity prominent in biological networks. RNNs have been successful at classifying text (Liu et al., 2016) and images from a pixel-by-pixel presentation (Goodfellow et al., 2016; Li et al., 2018; LeCun et al., 2015). This raises the question of what, if any, the advantage of neuronal dynamics and the diversity thereof is. Do neuronal dynamics enable the modeling of more temporally complex patterns? Does neuronal heterogeneity improve the learning capacity of neural networks?

We propose that neuronal dynamics, such as after-spike currents, as well as heterogeneity in the properties of these dynamics across a network, enhance a network's capacity to model temporally complex patterns. Moreover, we hypothesize that heterogeneity in dynamics, resulting from the diversity of single-neuron parameters, will be optimal when parameters are optimized in conjunction with synaptic weights during training. In order to evaluate these hypotheses, we develop and test a neuronal model whose parameters can be optimized through gradient descent.

A set of approaches has been previously used to develop neuronal models that encapsulate the above-mentioned biological complexities. These range from models that include a dynamical description of at the level of membrane channels (Hodgkin & Huxley, 1952; Morris & Lecar, 1981) to more compact models that synthesize the complexities into more abstract dynamical systems (Izhikevich, 2003) or hybrid systems (Mihalaş & Niebur, 2009). While networks of such neurons have been constructed (Markram et al., 2015; Billeh et al., 2020), they are often difficult to optimize, either to solve a task or to fit biological data.

Small increases in dynamical complexity can have significant benefits. Adaptation dynamics can help neural networks perform better on predictive learning tasks (Burnham et al., 2021) and achieve better fits of neuronal and behavioral data in mice (Hu et al., 2021). Inroads have been made to allow the optimization of spiking models using backpropagation via surrogate gradients (Huh & Sejnowski, 2018; Neftci et al., 2019). Such methods have revealed the importance of neuronal integration and synaptic timescales (Perez-Nieves et al., 2021) and adaptation (Salaj et al., 2021; Bellec et al., 2020) in network computation. Previous models (Perez-Nieves et al., 2021; Burnham et al., 2021) have also shown the importance of timescale diversity in temporal tasks. However, these models are still significantly simpler than those found to fit single-neuron data well (Teeter et al., 2018). To allow larger-scale dynamics to fit well, we want to be able to optimize neural networks with the type of dynamics proven to fit such complexities. Here we focus on the addition of after-spike currents, which are currents modulated by a neuron's spiking behavior leading to a wide variety of observed dynamics, including bursting (Gerstner et al., 2014; Mihalaş & Niebur, 2009).

While spiking neurons are more biologically realistic, they are generally more difficult to optimize. Thus, we develop a model that is typically differentiable but becomes a spiking model in the limit of taking a parameter to 0. We refer to this novel neuronal model as the generalized leaky-integrate-and-fire-rate (GLIFR) model. The GLIFR model is built on the spike-based generalized-leaky-integrate-and-fire (GLIF) models. GLIF models introduce several levels of complexity to leaky-integrate-and-fire models, which only model a neuron's leaky integration of input currents. We focus on the $GLIF_3$ model (Teeter et al., 2018), which accounts for after-spike currents through additive and multiplicative effects of spiking. Our proposed GLIFR model translates the $GLIF_3$ model into a rate-based scheme. Unlike the GLIF model, the differentiability of the GLIFR model enables the application of standard deep learning techniques to optimize parameters underlying intrinsic neuronal dynamics. Thus, we explore two aspects of the GLIFR model. First, we explore the addition of complex neural dynamics. After-spike currents act like a feedback mechanism in a neuron, being directly affected by a neuron's firing pattern and simultaneously being integrated into the voltage along with synaptic currents. Second, we study the effects of heterogeneity in dynamics resulting from heterogeneity in underlying neuronal parameters learned by gradient descent. For example, different after-spike currents may lead to different neuronal timescales. Beyond after-spike currents, differences in the membrane time constant and firing thresholds across neurons may add to the rich repertoire of firing patterns produced by a network.

To explore these two aspects introduced by the GLIFR neurons, we use gradient descent to optimize networks of GLIFR neurons on several temporal tasks and assess the performance and robustness of these networks in comparison to that of vanilla RNNs as well as long short-term-memory (LSTM) networks, which use a gating mechanism to promote memory over timescales (Hochreiter & Schmidhuber, 1997). While the GLIFR networks are outperformed by LSTM networks, our networks have mixed performance when compared to vanilla RNNs on a pattern generation task and a temporally complex sequential MNIST task. We find that it is possible to optimize both intrinsic neuronal parameters and synaptic weights using gradient descent. Optimization of neuronal parameters generally leads to diversity in parameters and dynamics across networks. Moreover, when we compare among several variations of the GLIFR model, we find that both the presence of after-spike currents and the heterogeneity in neuronal properties improve performance, suggesting an important computational role for neuronal complexity and heterogeneity in the brain. We provide code for creating and optimizing GLIFR models in Python.

1.2 Related Work. In our work, we assess how, if at all, the presence of neuronal dynamics, as well as the heterogeneity thereof, confers performance improvements in RNNs. Prior work has developed RNNs that express biologically realistic dynamics, giving rise to a class of networks known as spiking neural networks (SNNs). One example of an SNN (Zenke & Vogels, 2021) uses a leaky-integrate-and-fire dynamic across its neurons. In these neurons, the membrane potential was modulated according to equation 1.1 where S represents whether the neuron is spiking and V represents the neuron's membrane potential:

$$\begin{aligned} V(t+dt) &= (e^{-dt/\tau_m}V(t) + (1-e^{-dt/\tau})I(t))(1-S(t)), \\ S(t) &= \mathbb{H}(V(t)-V_{th}), \\ I(t+dt) &= e^{-dt/\tau_{syn}}I(t) + WS_{presynaptic}(t) + W_{rec}S(t). \end{aligned} \tag{1.1}$$

At each timestep, the voltage exponentially decays toward zero while being increased by the neurons' synaptic currents that also decay exponentially. Spiking drives the voltage to zero. In this work, only synaptic weights were trained, and a surrogate gradient was employed to address the difficulty presented by the undifferentiable Heaviside function. Specifically, when computing gradients of the loss with respect to parameters, the gradient of the Heaviside function was approximated by a smoother function (see equation 1.2). Using these approximated surrogate gradients, this SNN (Zenke & Vogels, 2021) achieved high performance on auditory tasks (SHD, RawHD, and RawSC) and an MNIST task where inputs are converted to spiking times:

$$f(x) = \frac{1}{(\beta|x|+1)^2}. \tag{1.2}$$

As an alternative to spiking models, a number of rate-based models have also been developed, including those that incorporate forms of after-spike currents. Muscinelli et al. (2019) and Beiran and Ostojic (2019) model after-spike currents in a form similar to equation 1.3 where I_j represents the after-spike current and s represents the firing rate. This form enables a neuron's firing behavior to have an additive effect on the after-spike current but not a multiplicative effect. In contrast, our model includes both an additive and multiplicative term as in Table 1:

$$\frac{dI_j(t)}{dt} = -k_jI_j(t) + a_js(t). \tag{1.3}$$

Table 1: Equations Describing the GLIFR Model.

$S(t) = \frac{1}{1+e^{-(V(t)-V_{th})/\sigma_V}}$
$\frac{dV(t)}{dt} = -k_m V(t) + R_m k_m I_{tot}(t) - (V(t) - V_{reset}) S_r(t)$
$\frac{dI_j}{dt} = -k_j I_j(t) + \left(a_j + r_j I_j(t)\right) S_r(t)$
$I_{syn}(t) = W_{in} S_{pre}(t) + W_{lat} S(t)$

In addition, in these works, parameter optimization by gradient descent was not explored. Other work has explored the effect of spike frequency adaptation on network performance using a different mode of spike frequency adaptation, mediated by threshold adaptation rather than after-spike currents (Salaj et al., 2021). Spike-frequency adaptation improved the performance of LIF SNNs on temporally challenging tasks, such as a sequential MNIST task, where the network must classify images of handwritten digits based on a pixel-by-pixel scan, an audio classification task where the network must identify silence or spoken words from the Google Speech Commands data set, and an XOR task where the network must provide the answer after a delay following the inputs. These networks were found to approach RNN performance on the second audio classification task.

While the work described thus far used spiking models of biological dynamics, it did not explore the advantage that heterogeneity could confer. Several approaches have been taken to this end. One approach is to initialize an SNN with heterogeneous dynamics but optimize only synaptic weights. This method achieved comparable or higher performance than ANNs that employed convolutional or recurrent transformations on an object detection task (She et al., 2021). A second approach is to optimize intrinsic neuronal parameters in addition to synaptic weights. Under the hypothesis that neuronal heterogeneity is computationally advantageous, the learned parameters will be diverse across the trained network. To this end, one study extended the surrogate gradient technique for LIF SNNs to also optimize membrane and synaptic time constants across networks (Perez-Nieves et al., 2021). It was found that these networks learned heterogeneous parameters across the network when trained on temporal MNIST tasks, a gesture classification task, and two auditory classification tasks. On some tasks, particularly the temporally complex tasks that relied on precise timing of spikes, learning parameters improved performance over learning synaptic weights alone. However, in this work, learning parameters relied on surrogate gradients, and a simpler neuron model (LIF) was used.

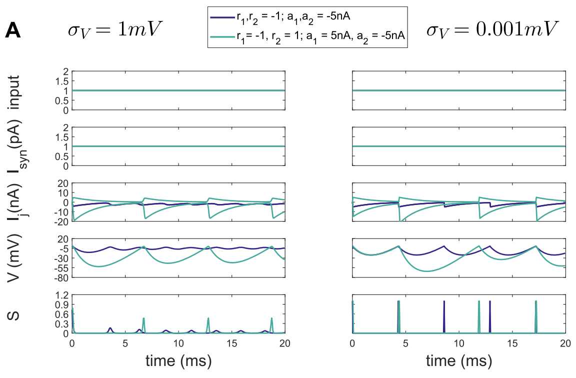
Recent work has proposed a novel approach, event-based backpropagation, to compute exact gradients despite the discrete spiking (Wunderlich & Pehle, 2021). While promising, this approach requires complex computational techniques. In contrast, our work establishes a method of training neuronal parameters with standard gradient descent by using a rate-based neuronal model; moreover, this model additionally expresses after-spike currents as schematized in Figures 1A and 1B.

To summarize, previous work has suggested that heterogeneity in integration and synaptic timescales improves network performance. This opens the door to the questions we address here: whether after-spike currents that produce complex dynamics within individual neurons can be trained, whether these after-spike currents improve network performance, and whether training naturally leads to their heterogeneity. It also invites the question of whether neuronal models with these complex dynamics can be designed that learn using standard gradient descent. In what follows, we show that this is possible, and illustrate how this results in heterogeneous neuronal dynamics as well as mixed performance relative to vanilla RNNs.

2 Generalized Leaky-Integrate-and-Fire-Rate (GLIFR) Model

2.1 Model Definition. We develop a neuronal model that exhibits after-spike currents and is conducive to training with gradient descent (see Figures 1B and 1C). To do this, we build on a previously described spike-based GLIF model that incorporates after-spike currents (Teeter et al., 2018). We transform this GLIF model into what we term a GLIFR model, which is rate-based to facilitate optimizing neuronal parameters with traditional gradient descent mechanisms. Specifically, we modify the GLIF$_3$ model from Teeter et al. (2018) to produce firing rates rather than discrete spikes. In order to do this, we define the normalized firing rate of a GLIFR neuron as a sigmoidal transformation of the voltage ($S(t) = \frac{1}{1+e^{-(V(t)-V_{th})/\sigma_V}}$). This may be interpreted as an instantaneous normalized firing rate (varying between 0 and 1), as a spiking probability, or as a smoothed spike (see Figure 2). A raw firing rate can be derived from $S(t)$ as follows: $S_r(t) = \frac{S(t)}{\tau}$ where τ (ms) represents how often we check if the neuron has crossed the threshold. The parameter σ_V (mV) controls the smoothness of the voltage-spike relationship, with low values ($\sigma_V \ll 1$ mV) enabling production of nearly discrete spikes and higher values enabling more continuous output. This rate-based approach enables the use of exact gradients rather than surrogate gradients across the spiking function. Additionally, it is more akin to vanilla RNNs that use smooth activation functions, thereby motivating the application of standard deep learning techniques.

We model voltage similar to its definition in the spiking GLIF model, but instead of the discrete reset rule in the GLIF model, we let voltage continuously tend toward the reset voltage V_{reset} (mV) at a rate proportional to the firing rate S_r. Thus, voltage is modeled as $\frac{dV(t)}{dt} = -k_m V(t) + R_m k_m I_{tot}(t) - (V(t) - V_{reset}) S_r(t)$, where $I_{tot}(t)$ is the total current produced by the sum of a constant baseline current I_0, the after-spike currents, and the synaptic current.



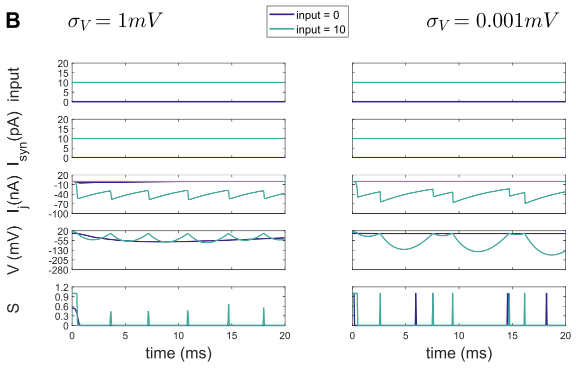


Figure 2: (A) Sample responses to constant input. These plots show example outputs of two neurons when provided with a constant input over a 40 ms time period. The two neurons share identical membrane-related parameters and decay factors for after-spike currents. The two neurons differ only in the multiplicative and additive terms for after-spike currents (r_j, a_j). The neuron whose output is traced in purple has the following after-spike current related parameters: $r_1, r_2 = -1$; $a_1, a_2 = -5000\ pA$. The neuron whose output is represented in green has the following parameters: $r_1 = -1, r_2 = 1$, $a_1 = 5000\ pA, a_2 = -5000\ pA$. The difference in after-spike current parameters gives rise to different types of dynamics. The lefthand column contains outputs produced by neurons with $\sigma_V = 1$, whereas the righthand column contains outputs produced by neurons with $\sigma_V = 0.001$ mV, demonstrating the ability of the GLIFR neuron to produce spike-like behavior while maintaining its general differentiability. Both $I_1(t)$ and $I_2(t)$ are plotted for each parameter combination. Thus, we plot two purple traces corresponding to $j = 1, 2$ and two green traces corresponding to $j = 1, 2$. (B) Sample responses to different amplitude inputs. These plots show example outputs of a neuron when provided with different magnitudes of constant input over a 40 ms time period. Larger inputs appear to yield higher-frequency oscillations in the neuron's output. The lefthand column contains outputs produced by neurons with $\sigma_V = 1$ mV, whereas the righthand column contains outputs produced by neurons with $\sigma_V = 0.001$ mV.

After-spike currents are modeled as a set of separate ionic currents indexed by j as follows: $\frac{dI_j}{dt} = -k_j I_j(t) + (a_j + r_j I_j(t))S_r(t)$. The decay factor k_j (ms^{-1}) captures the continuous decay of the current. The "spike-dependent" component of the current is determined by the combination of the multiplicative parameter r_j and the additive parameter a_j (pA), scaled by the raw firing rate $\frac{1}{\tau}s$.

Synaptic currents, those resulting from input from the previous layer as well as lateral connections, are modeled according to $I_{syn}(t) = W_{in}S_{pre}(t) + W_{lat}S(t)$, where S_{pre} represents either the normalized firing rate of the presynaptic neuron layer or the input to the network. The weights W_{in} describe how the input to the neuron layer should be weighted, and the lateral weights W_{in} describe the connections between neurons in the same layer. We do not include a temporal dependence for simplicity as we focus on cellular properties. These equations are summarized in Table 1.

Applying gradient descent, we simultaneously optimize both synaptic weights and parameters underlying the GLIFR model. The parameters we train are listed in Table 5. The GLIFR model is also publicly released as a PyTorch model (further details on its implementation are in section 6).

2.2 Effect of Parameters on Model Behavior. The effects of the parameters on model behavior have been previously described in the discrete spiking scheme (Mihalaş & Niebur, 2009). In this work, we translate to a continuous spiking scheme. Thus, we visualize the effect of σ_V on model behavior (see Figures 2A and 2B). As shown, lower values of σ_V approximate discrete spikes, while higher values of σ_V result in smoother changes in the firing rate.

Varying the values of a_j and r_j can give rise to a variety of complex patterns including bursting. As shown in Figure 2A, hyperpolarizing (negative) values enable the neuronal firing rate to oscillate slightly, and a combination of hyperpolarizing and depolarizing (positive) after-spike currents enables regular oscillations in firing rate. Because we model firing rates rather than individual spikes, we take this as a form of bursting. We furthermore find that for a given set of neuronal parameters, larger inputs yield higher-frequency oscillations in firing rate (see Figure 2B). We note that in these simulations and later in trained networks, the GLIFR model is theoretically capable of producing both biologically plausible patterns and less biologically realistic activity.

2.3 Optimization of Neuronal Parameters for Learning Realizable Signals. We first confirm the ability of neuronal parameters to be optimized through gradient descent in a theoretically simple task: learning a target that is known to be realizable by a GLIFR.

As shown in Figure 3A, we initialize a single GLIFR neuron and record its response to a constant input stimulus over a fixed period of time (10 ms

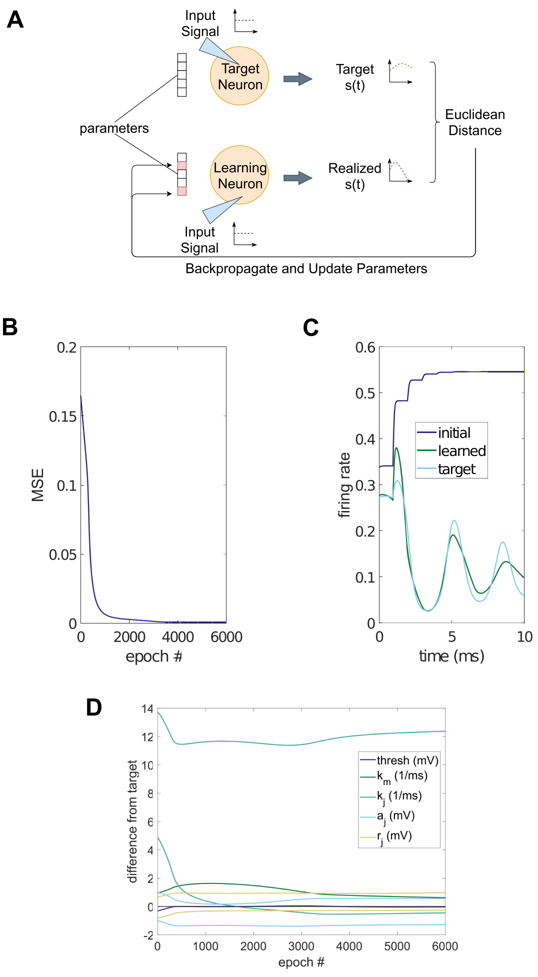


Figure 3: (A) Testing optimization in single neurons for realizable signals. In the realizable pattern generation task depicted here, a neuron (the target neuron) is used to create the target firing pattern. A second neuron (the learning neuron) is initialized with different parameters and learns parameters to produce the target. (B) Training loss. This plot shows an example trace of mean-squared error over training of a single neuron. (C) Output of the learning neuron. This plot shows the output of the learning neuron prior to training and after training, along with the target, demonstrating the ability of a single GLIFR neuron to learn simple patterns. (D) Learned parameters. This plot depicts the difference between the values of the trainable parameters in the learning neuron and the corresponding values in the target neuron over training epochs. Note that a value of zero represents equivalence to the target network in a particular parameter.

with timestep duration of 0.05 ms.) This is our target neuron. We then create a second neuron (learning neuron) that is equivalent to the target neuron only in the incoming weights used. We train the learning neuron to produce the target neuron's output, using Adam optimization and a learning rate of 0.001, allowing it to learn the neural parameters (Kingma & Ba, 2014).

We tested the ability of the learning neuron to reproduce the pattern of the target neuron. Figures 3B and 3C show that the learning neuron successfully learned to approximate the dynamics of the target neuron. However, the final parameters learned by the learning neuron differed from the parameters of the target neuron (see Figure 3D), illustrating that different internal mechanisms can lead to similar dynamics (Prinz et al., 2004). This supports the idea that for the proposed model, when optimizing independently the time constants and the additive and multiplicative parameters for after-spike currents, varying distributions of trained parameters across a network may nevertheless lead to similar network function.

3 Results

3.1 Strategy for Analyzing Task Learning and Performance. After verifying the theoretical ability to optimize neuronal parameters in single GLIFR systems, we turn to more complex tasks. As in previous work (Perez-Nieves et al., 2021), we aimed to assess the role of several factors in network computation while evaluating our networks: (1) the presence of biologically realistic dynamics (i.e., membrane dynamics, voltage reset), (2) the presence of after-spike currents, (3) random heterogeneity of neuronal parameters across the network, and (4) optimized heterogeneity of neuronal parameters across the network. Thus, we use multiple network types. As a baseline, we use a vanilla recurrent neural network (RNN). We also use the following variations of the GLIFR network: a GLIFR network with fixed heterogeneity with after-spike currents (FHetA) or without (FHet), a GLIFR network with refined heterogeneity with after-spike currents (RHetA) or without (RHet), and a GLIFR network with learned heterogeneity with after-spike currents (LHetA) or without (LHet). We define fixed heterogeneity as heterogeneity that a network is initialized with and does not alter over training, refined heterogeneity as heterogeneity that a network is initialized with but is altered ("fine-tuned") over training, and learned heterogeneity as heterogeneity that a network is not initialized with but is learned over training. To achieve heterogeneous initialization, we permute the parameters learned in a corresponding LHet/LHetA network and initialize a new network with the resulting parameters. These distinctions are illustrated in Figure 4. Finally, we use an LSTM network as another baseline whose mechanisms improve the ability to model complex temporal dynamics without emulating biology. For each experiment, we set the number of neurons for each network type to yield comparable numbers of learnable parameters.

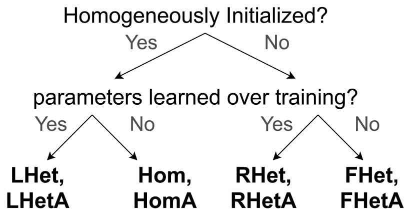


Figure 4: GLIFR schemes. A flowchart describes the variations of the GLIFR networks we explored. These are based on whether the network neuronal parameters were homogeneously initialized or heterogeneously initialized as well as whether these intrinsic neuronal parameters were learned over training. This classification enables us to isolate effects of the expression of after-spike currents (models with names ending in A), a complex type of dynamic, and the learning of heterogeneous parameters.

Each network, including RNNs, GLIFR networks, and LSTM networks, consists of a single recurrent hidden layer of the appropriate neuron type, whose outputs at each time point are passed through a fully connected linear transformation to yield the final output. The recurrence is incorporated into the GLIFR networks through the term $W_{lat}S(t)$ in the synaptic currents, equation 1.1, and the $W_{lat}^{pq}s^{p,l}(t-\Delta t)$ term in equation 6.5.

Code for the GLIFR model is publicly available at https://github.com/AllenInstitute/GLIFS_ASC. The model is implemented in PyTorch, and we rely on the underlying autodifferentiation engine to backpropagate through the model.

3.2 Performance on Tasks. In this work, we study five tasks: a sine generation task, a neuromorphic MNIST (N-MNIST) task, a line-by-line MNIST (L-MNIST) task, and a pixel-by-pixel MNIST (P-MNIST) task. We choose these tasks due to their varying complexity and timescales. The sine generation task tests the ability of a network to learn sine waves whose frequencies are determined by the amplitude of the input signal. This requires memory over a relatively small time period. The N-MNIST task is based on the scanning of an MNIST image using a digital vision sensor, which senses changes in pixel intensity along two dimensions. The resulting input to the model is along 15 timesteps, where for each timestep, the network is input a $34 \times 34 \times 2$ binary vector representing two channels capturing the change in pixel intensity over the 34×34 image. In the L-MNIST and P-MNIST tasks, networks are expected to classify the digit represented by an MNIST digit after receiving the input image either line-by-line or pixel-by-pixel. Thus, the L-MNIST task engages memory over 28 timesteps (one timestep is used to process each line of the image), whereas the P-MNIST task engages memory over 576 timesteps. The setup of each of these tasks is depicted in Figure 5, and details of each of these tasks, along with the

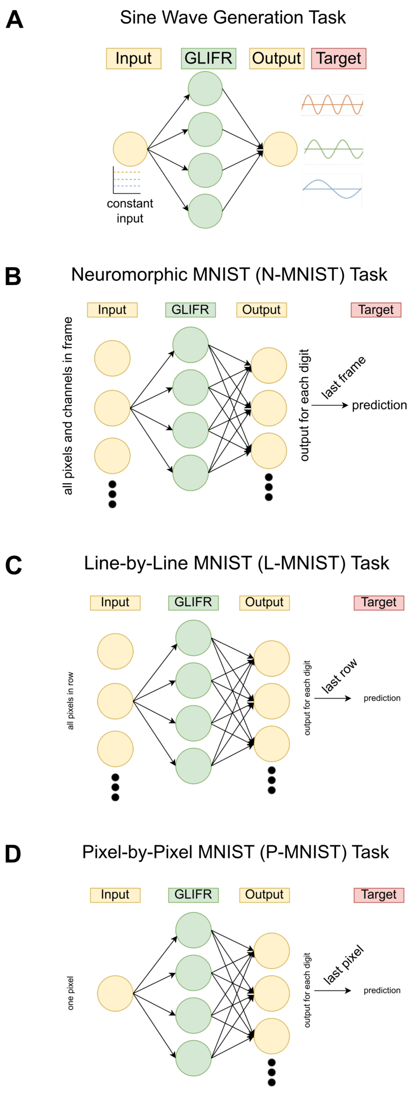


Figure 5: Setup of tasks explored. These schematics illustrate the setup of the sine wave generation task (A), the neuromorphic MNIST task (B), the line-by-line MNIST task (C), and the pixel-by-pixel task (D). In each schematic, the input to the network as well as the readout (produced by a linear transformation of the output of the neuron layer) are denoted in yellow, and the hidden layer of neurons (GLIFR, RNN, or LSTM neurons) is denoted in green. The lateral connections are not visualized for clarity, but in each task, lateral connections between neurons exist. Finally, the target signal is shown, along with how the readout is transformed.

hyperparameters used, are in section 6. Due to the computation time required to train networks on the P-MNIST task, we limit our training to 50 epochs and analyze the performance of only RNNs and the LHetA and RHetA GLIFR networks.

We run 10 training simulations for each task and network type, and we use two-sample t-tests to analyze the results ($\alpha = 0.05$). As shown in Figure 6 and Table 2 (see also Figures 10 and 11), all networks train to close to convergence within the given training epochs except on the P-MNIST task due to the limited training time. The GLIFR networks, regardless of their initialization and whether they learn parameters over training, outperform RNNs on the N-MNIST and P-MNIST tasks. Within the variations of the GLIFR networks, we find that after-spike currents appear to improve performance only in the L-MNIST task, particularly in the LHet, RHet, and FHet schemes. On the other hand, learned heterogeneity appears to have a much more consistent benefit to performance. On all tasks except for the N-MNIST task, RHetA networks outperform LHetA networks, indicating a benefit of heterogeneous initialization, and LHetA and RHetA networks outperform HomA networks, indicating a benefit of learned heterogeneity. Oddly, we find the reverse pattern in the N-MNIST task, where among the eight variations of the GLIFR networks studied, Hom networks perform best. We hypothesize that heterogeneous and complex dynamics are advantageous. However, due to the complicated error landscape, gradient descent is capable of optimizing the heterogeneity of neuronal parameters only in some settings. Exploring in detail the optimization setting that benefits such complex neuron models is beyond the scope of this study.

3.3 Robustness to Silencing. GLIFR dynamics enable more complex temporal interactions within and among neurons. What is the resulting impact on the robustness of GLIFR networks' random "deletions" (or silencing) of individual neurons? We test this by evaluating the robustness of networks to random silencing of neurons. Specifically, for various proportions p, we randomly select a subset of the neurons in the network with proportion p to silence in a trained network throughout testing. For each subset, we clamp the neurons' firing rates to zero, preventing their contribution in both forward and lateral connections, and compute the performance of the network on the task. We run this experiment on the sine wave generation, N-MNIST, and L-MNIST tasks (see Figure 7). In all tasks, silencing impaired performance, but we analyze the extent to which each network was impaired. Lower impairment implies greater robustness to silencing. In the sine wave generation task, several variations of the GLIFR network appear more robust than the RNN when small percentages of neurons are silenced. The GLIFR networks do not appear to maintain this advantage for the N-MNIST task. We see the greatest differences in the L-MNIST task, where we find that when silencing proportions of neurons with $p \geq 0.2$, all forms of GLIFR networks show an advantage over the RNN. In general, RHetA

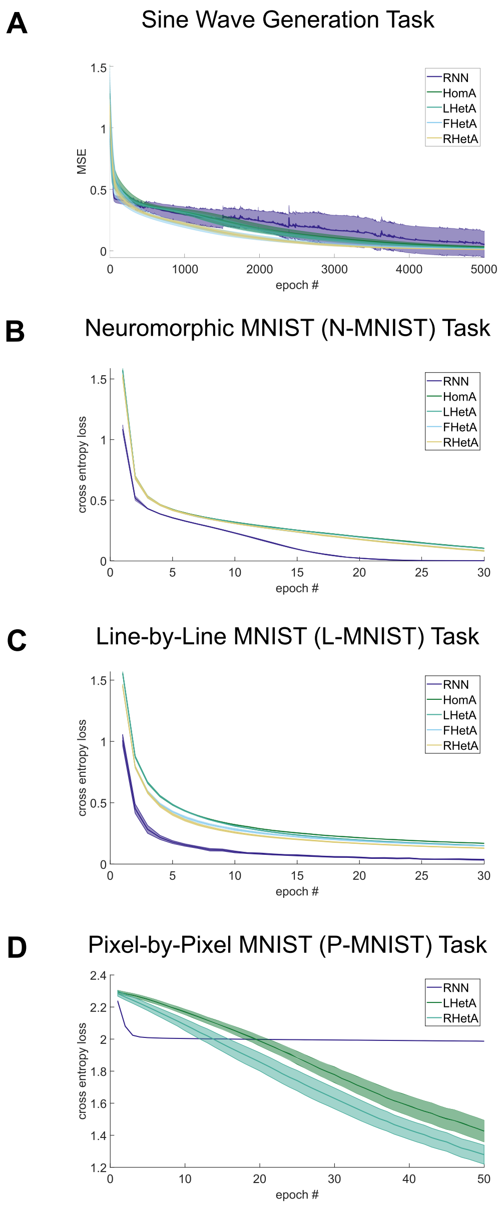


Figure 6: Training loss curves. The training mean-squared error (sine wave generation) or cross-entropy loss (MNIST tasks) of the network averaged over 10 random initializations is plotted over training epochs. The shading indicates a moving average of the standard deviation. On average, all network types converge on a solution within the training time except for the P-MNIST task.

Table 2: Performance on Tasks.

	Sine Generation Task	N-MNIST	L-MNIST	P-MNIST
RNN	0.0561	84.1740%	97.7920%	25.6400%
LSTM	0.0094	85.9230%	98.6620%	N/A
Hom	0.0348	88.0000%	93.6540%	N/A
HomA	0.0309	87.9560%	93.6420%	N/A
LHet	0.0233	87.7470%	93.7750%	N/A
LHetA	0.0227	87.8330%	94.1700%	61.6780%
FHet	0.0164	87.2160%	93.7220%	N/A
FHetA	0.0166	87.2240%	94.1160%	N/A
RHet	0.0115	87.4020%	93.7520%	N/A
RHetA	0.0121	87.3070%	94.8140%	67.2090%

Note: This table lists the testing performance (mean-squared error for the sine wave generation task and accuracy for the remaining tasks) of each task explored.

networks perform the best. This observation suggests that neuronal complexity and heterogeneity improve network robustness despite the reduced baseline performance induced by silencing when compared with vanilla RNNs.

Arguably, the poorer robustness of the RNNs may be anticipated due to the lack of dropout during training (Hinton et al., 2012). Thus, we performed additional experiments on the L-MNIST task. For each probability p, we trained each network with dropout with probability p and tested its performance when random sets of neurons (proportion p) were silenced through the entirety of each testing simulation. Indeed, we see that the performance of these networks trained with dropout is better than that observed without dropout, but the trends discussed above hold (see Figure 12).

3.4 Optimization in Discrete Spiking Regime. In all the experiments so far, we have kept σ_V a constant during training. As previously noted, the parameter σ_V can be modified such that as $\sigma_V \ll 1$, the model approaches discrete spiking dynamics.

We take a simulated annealing approach to assess whether this setup can be utilized to learn a nearly discrete model that would more closely reproduce biologically realistic, rapid spiking. Specifically, we train an LHetA GLIFR network on the L-MNIST task while gradually decreasing the σ_V parameter over training (from 1 to 10^{-3}). We find that the LHetA networks still learn the task well, achieving an average accuracy of 74.49% ($n = 10$; standard deviation of 1.90%). This is lower than the previously achieved accuracy of 94.17% using relatively high values of σ_V but better than when using a constant but low value of σ_V throughout training. We also note that this approach does not converge once σ_V is reduced past a certain amount

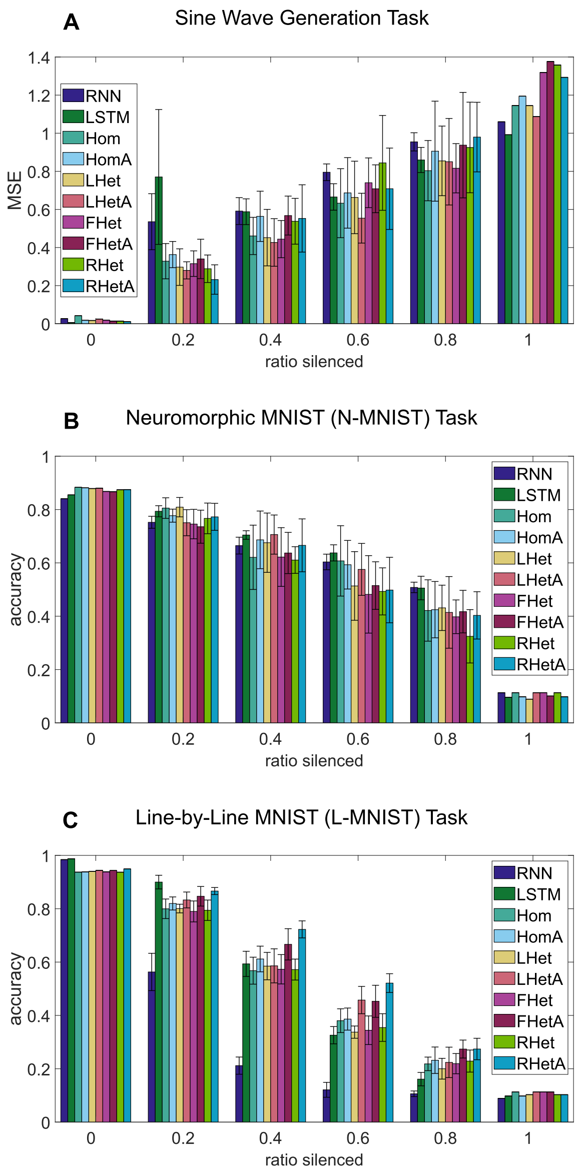


Figure 7: Model performance when neurons were silenced through testing. A single random network initialization was trained on the L-MNIST task. During testing, a random subset of neurons in the network was silenced, and the network was tested. This was repeated 10 times with different random subsets of the same percentage. The average accuracy of each network with varying percentages of their neurons silenced is shown for each task. Bars represent standard deviation across trials.

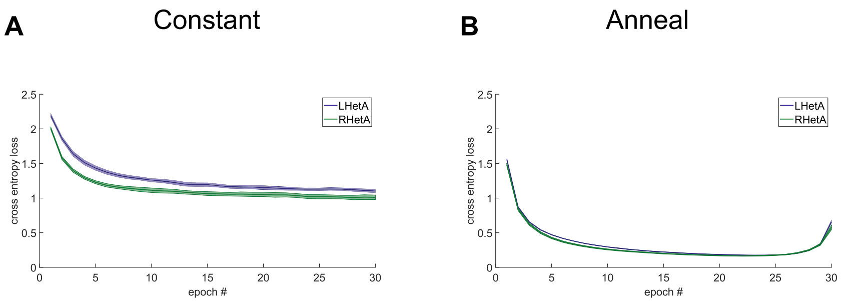


Figure 8: Training loss curves with lower σ_V. The training cross-entropy loss of the LHetA and RHetA networks on the L-MNIST task averaged over 10 random initializations is plotted over training epochs where either σ_V is maintained at a constant value of 10^{-3} (A) or the simulated annealing approach is taken for learning with small values of σ_V (B). The shading indicates a moving average of the standard deviation.

(see Figure 8). Thus, further work is needed to identify the optimal annealing procedure for learning tasks with low σ_V, but this annealing experiment demonstrates the ability of GLIFR networks to produce spiking behavior and still perform well despite being a fundamentally rate-based model.

3.5 Heterogeneity Learned over Training. Based on the performance values above, neuronal heterogeneity seemed to contribute to the GLIFR networks' performance. To further elucidate whether and how the learned heterogeneity in parameters may have reshaped neuronal dynamics, we determined the extent to which the GLIFR networks had learned truly diverse parameters in each task context. For the homogeneously initialized networks, both a_js and r_js had been initialized with limited diversity to promote neuronal stability, and the remaining neuronal parameters had been set homogeneously. We hypothesized that the diversity in all parameters would have developed over training. The heterogeneously initialized networks were set with distributions of parameters learned by the LHet and LHetA networks, but we expected the distribution to shift over the course of training as the model fine-tuned its shuffled parameters and weights to the particular task.

We found that training did result in heterogeneous parameter values for all tasks (data shown for L-MNIST; see Figures 13A and 13B) such that the variance of the trained a_j and r_j parameters is much larger than at initialization (standard deviation of initial distribution was 0.01). Similarly, we observe shifts in the distribution of some parameters when heterogeneity is "refined." We wanted to determine how well this diversity in neuronal parameters mapped to diversity in neuronal dynamics. To do this, we constructed f-I curves representing the average firing rate of each neuron over a time period (5 ms) when the neuron was injected with varying levels of current (see Figure 13B). We found a diversity in shapes of these curves, illustrating the diversity in the neuronal dynamics produced by neurons in the trained networks.

Finally, we used Ward's hierarchical clustering to capture in other ways the variation in neuronal dynamics. Specifically, we performed hierarchical clustering on the neuronal parameters of networks trained on the each task. We used the Calinski-Harabasz (CH) index as a measure of clustering. For the pattern generation task, we were unable to find a number of clusters that produced the optimal CH score. However, we found that the CH score was maximized using five clusters in the LHetA network trained on the N-MNIST task and using four clusters in the LHetA network trained on the L-MNIST task (see Figure 14A). After clustering these neurons, we examined their parameters. Arguably, the classes of neurons for both tasks appear to lie on a continuum rather than discretely clustering, but they appeared to be separated primarily based on after-spike current parameters a_j and k_j (see Figure 9B). In the networks trained on the L-MNIST task, one class (A) expresses slow depolarizing after-spike currents, a second class (B) expresses fast after-spike currents with hyperpolarizing values of a_j and depolarizing values of r_j, a third class (C) expresses smaller and faster after-spike currents, and a final class (D) expresses after-spike currents with small hyperpolarizing values of a_j and depolarizing values of r_j. As shown in Figure 15D, this results in saturating behavior for class A, oscillating or bursting behavior for class B, and relatively stable behavior for the other two classes. We noted that each pair of parameters corresponding to after-spike currents tended to be similar (i.e., for a given neuron, $a_1 \approx a_2$). We suggest that this is a result of the gradient descent technique. Potentially, the gradients over both after-spike currents were similar, resulting in both sets of parameters progressing in the same direction. We found similar clustering for the N-MNIST task but not the sine wave generation task, for which we were unable to find an optimal number of clusters.

We also found that the classes based on parameters separated the f-I curves (see Figure 9B). For example, the neurons with hyperpolarizing after-spike currents tended to exhibit low maximal firing rates. And the neurons with depolarizing after-spike currents tended to exhibit firing rates that rapidly saturated to relatively high values. While we cannot extend these results to identify each category as discrete cell types, this further suggests that the learned diversity in parameters enabled a diversity in dynamics as well.

4 Discussion

This work explored the role of neuronal complexity and heterogeneity in network learning and computation using a novel paradigm. Specifically, we developed the GLIFR model, a differentiable rate-based neuronal model

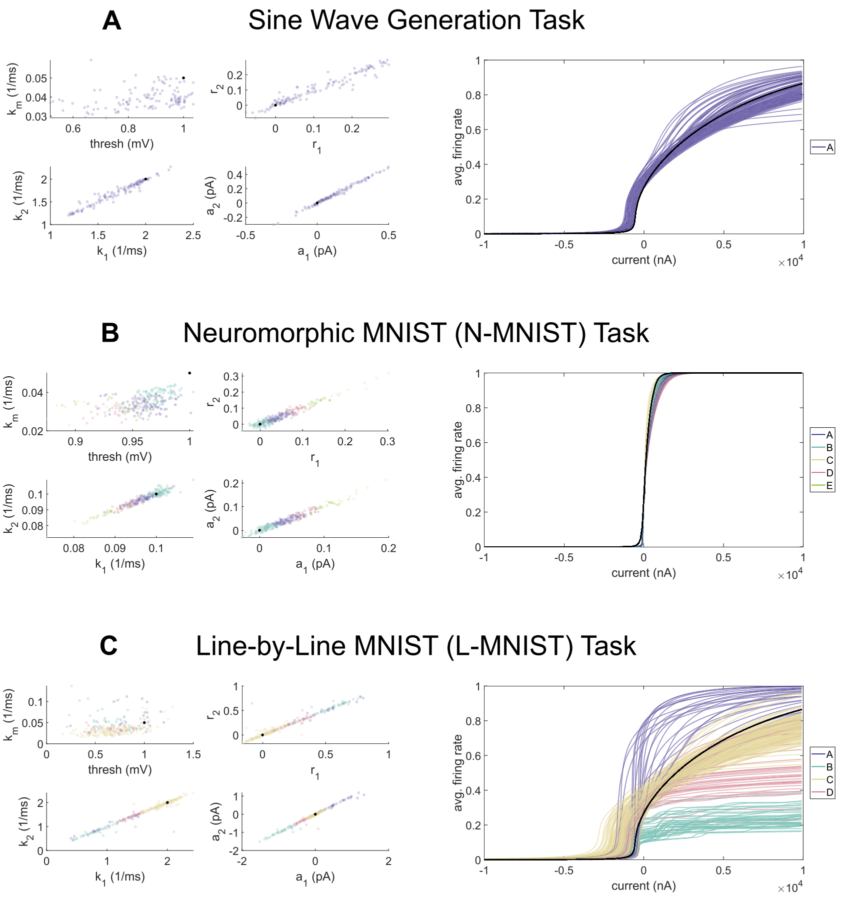


Figure 9: Heterogeneity in learned networks for the sine wave generation task (A), the N-MNIST task (B), and the L-MNIST task (C). For each task, various pairs of parameters in a representative trained LHetA network are plotted. The points are colored according to clusters produced by Ward's hierarchical clustering on learned parameters using optimal cluster number (based on the CH index). These plots are displayed in the left column of the figure. The black point in each scatterplot represents the parameter distribution the network was initialized to. On the right column of the figure, f-I curves are plotted for each neuron in a representative trained LHetA network. The black traces represent those produced by the initial network. Together, these data demonstrate the heterogeneity in parameters and dynamics that result from the training.

that expresses after-spike currents in addition to the simple types of dynamics. While past work (Perez-Nieves et al., 2021; Salaj et al., 2021) has studied neuronal complexity and heterogeneity, here we demonstrated the ability to learn both synaptic weights and individual neuronal parameters underlying intrinsic dynamics with traditional gradient descent for models with more complex internal neuron dynamics generated by after-spike currents. While it is generally rate-based, the GLIFR model retains the ability to produce spiking outputs and thus is a powerful model for studying neuronal dynamics.

We demonstrated the ability for networks of GLIFR neurons to learn tasks ranging in complexity from a sine wave generation task to the pixel-by-pixel MNIST task. In each task, heterogeneous parameters and dynamics were learned. We tested the effects of the ability to learn parameter heterogeneity and the ability to express after-spike currents on model performance. Learning heterogeneous parameter distributions generally improved performance, and modeling after-spike currents improved performance on the L-MNIST task. Regardless of whether heterogeneous parameters or after-spike currents were learned, the GLIFR models outperformed vanilla RNNs in the N-MNIST and P-MNIST tasks. We hypothesize that the differences in results among tasks reflect differences in complexity and engagement of memory between tasks. For instance, we only observe improvements in GLIFR networks over RNN networks in N-MNIST and P-MNIST tasks, but not in the sine wave generation task and the L-MNIST task. Since the N-MNIST and P-MNIST tasks theoretically engage more memory than the other two tasks, we suggest that the RNN may simply do well enough for the other two tasks that improvement over it would be unexpected. It is difficult to explain why neuronal heterogeneity appeared to be disadvantageous in the N-MNIST task, but perhaps neuronal heterogeneity complicated the error landscape and hindered the network's learning ability in this setting. We also found that GLIFR networks trained on the L-MNIST task were more robust to neuronal loss. Specifically, when we silenced fixed fractions of neurons, the GLIFR networks generally performed better than vanilla recurrent neural networks. Finally, we found that learning parameters across a network in response to a temporally challenging task enabled the network to develop neurons with differing intrinsic parameters and dynamics.

These findings support the hypothesis that neuronal heterogeneity and complexity have a computational role in learning complex tasks. The implications of this are two-fold. On one hand, neuronal heterogeneity may allow more powerful computing in artificial neural networks. Vanilla recurrent networks, for example, rely on a single type of dynamic—typically the ReLU, sigmoid, or tanh activation function. However, the use of activation functions that can be "learned" over time, such that the trained network exhibits diverse dynamics across its neurons, may confer a computational advantage (Geadah et al., 2020). Here we demonstrated several computational advantages of more biologically realistic neurons for some specific temporally challenging tasks while using traditional gradient descent training techniques. Other learning techniques may be required for other tasks, where the complex error landscape hinders the ability of gradient descent to optimize both the neuronal parameters and the synaptic weights.

On the other hand, our results provide further insight into the purpose of complexity and diversity of neural dynamics seen in the brain. Our brains are responsible for integrating various sensory stimuli, each varying in their temporal structures. Intuitively, heterogeneity in the types of dynamics used by neurons across the brain may enable robust encoding of a broad range of stimuli. The ability of diverse networks to encode information more efficiently is supported by several studies (Hunsberger et al., 2014; Shamir & Sompolinsky, 2006; Tripathy et al., 2013). Certain types of complex dynamics may also affect synaptic strengths and improve the network's robustness to damage.

We believe that with the GLIFR model in the research domain, we have opened the door to more intriguing studies that will identify further roles for complex neural dynamics in learning tasks. In future work, testing the GLIFR models on additional tasks may provide additional insight into the computational advantages of neuronal heterogeneity and complexity. Additionally, it would be interesting to pursue a theoretical explanation of the computational role of heterogeneity and complexity. Past experimental work (Tripathy et al., 2013) found that neuronal heterogeneity increased the amount of information carried by each neuron and reduced redundancy in the network. Exploring this in computational models would be valuable, as it may suggest additional ways in which the computational advantages of biological dynamics can be harnessed to improve artificial neural networks and yield insights into the mechanisms of computation in biological networks.

5 Code Availability

Modeling, training, and analysis code is available at https://github.com/AllenInstitute/GLIFS_ASC.

6 Methods

6.1 GLIFR Model.

6.1.1 Development of the Single-Neuron Model. As previously described, the GLIFR model encapsulates the concepts behind the GLIF$_3$ model (Teeter et al., 2018), while producing continuous outputs. We describe how state variables are computed at each timestep in the GLIFR model, comparing the computation to that in the GLIF model. We use a subscript s for the GLIF state variables and no subscript for the equivalent GLIFR variables. We use superscript p to represent the index of the neuron in consideration and l for the associated layer, where postsynaptic layers are indexed with larger numbers than are presynaptic layers. For example, $S^{p,l}(t)$ defines the normalized firing rate of the pth neuron in the lth layer.

The discrete spiking equation can be expressed as in equation 6.1. $\mathbb{H}$ refers to the Heaviside function where $\mathbb{H}(x) = 1$ when $x > 0$ and $\mathbb{H}(x) = 0$ when $x \leq 0$. Thus, spiking is a binary event, and neurons produce a spike when the voltage exceeds a certain threshold. In the GLIFR model, we define $S^{p,l}(t)$, the normalized firing rate that is unitless and varies continuously between 0 and 1 (see equation 6.2). A raw firing rate $S_r^{p,l}(t) = \frac{1}{\tau} S^{p,l}(t)$ may be derived where τ (ms) represents how often we check if the neurons has crossed the threshold, and we set τ to dt throughout the study. While S represents a normalized firing rate, for simplicity of terminology, we refer to it as the firing rate. The parameter σ_V (mV) controls the smoothness of the voltage-spiking relationship, and we used $\sigma_V = 1$ mV for all simulations unless otherwise noted. Thus, for the GLIF and GLIFR models, respectively, spiking is defined as

$$S_s^{p,l}(t) = \mathbb{H}(V^{p,l}(t) - V_{th}^{p,l}), \tag{6.1}$$

$$S^{p,l}(t) = \frac{1}{1 + e^{-(V^{p,l}(t) - V_{th}^{p,l})/\sigma_V}}. \tag{6.2}$$

After-spike currents are modeled as a set of separate ionic currents (corresponding to $j = 1, 2, \ldots$) as in equation 6.4. We use two after-spike currents with arbitrary ionic current equivalents, but this model can theoretically be extended to more currents. As in the GLIF model (see equation 6.3), decay factor $k_j^{p,l}$ (ms^{-1}) is used to capture the continuous decay of the current, and the "spike-dependent current" is determined by multiplicative parameter $r_j^{p,l}$ and additive parameter $a_j^{p,l}$ (pA). In the GLIFR model, though, this "spike-dependent current" is scaled by the raw firing rate $\frac{1}{\tau} s^{p,l}$:

$$\frac{dI_{s,j}^{p,l}(t)}{dt} = -k_j^{p,l} I_j^{p,l}(t) + (a_j^{p,l} + r_j^{p,l} I_j^{p}(t)) S_s^{p,l}(t), \tag{6.3}$$

$$\frac{dI_j^{p,l}(t)}{dt} = -k_j^{p,l} I_j^{p,l}(t) + (a_j^{p,l} + r_j^{p,l} I_j^{p,l}(t)) S_r^{p}(t). \tag{6.4}$$

Synaptic currents, currents resulting from input from the previous layer as well as lateral connections, are modeled according to equation 6.6. The first term in this equation represents the integration of firing inputs from the presynaptic layer, and the second term describes the integration of firing inputs from the same layer through lateral connections. W_{pq} (pA) defines how much the presynaptic spiking outputs (of neuron p) affect the postsynaptic current (of neuron q). While we model this synaptic transmission as being nearly instantaneous (the delay being just one timestep), we model the

Table 3: States in GLIFR Neurons and Their Units.

$s(t)$	normalized firing rate	1
$V(t)$	voltage	mV
$I_j(t)$	after-spike current	pA
$I_{syn}(t)$	synaptic current	pA

lateral transmission as having a synaptic delay denoted Δt. In other words, $I^q_{syn}(t)$ depends on $S^{p,l-1}(t)$ and $S^{p,l}(t-\Delta t)$:

$$I^q_{s,syn}(t) = \sum_p W^{pq}_{in} S^{p,l-1}_s(t-dt) + \sum_p W^{pq}_{lat} S^{p,l}_s(t-\Delta t), \tag{6.5}$$

$$I^q_{syn}(t) = \sum_p W^{pq}_{in} S^{p,l-1}(t-dt) + \sum_p W^{pq}_{lat} S^{p,l}(t-\Delta t). \tag{6.6}$$

Finally, voltage is modeled in much the same way as in a traditional GLIF model (see equation 6.7), but in order to facilitate gradient descent, we ignore the more complex reset rule that is used in the GLIF model (Teeter et al., 2018). Instead, we set $f_r = 1$ in equation 6.8:

$$\frac{dV^{p,l}_s(t)}{dt} = -k^{p,l}_m V^{p,l}(t-dt) + R_m k^{p,l}_m \left(I_0 + I^{p,l}_{syn}(t) + \sum_j I^{p,l}_j(t) \right) - f_r(V^{p,l}(t-dt) - V_{reset}) S^{p,l}_s(t), \tag{6.7}$$

$$\frac{dV^{p,l}(t)}{dt} = -k^{p,l}_m V^{p,l}(t-dt) + R_m k^{p,l}_m \left(I_0 + I^{p,l}_{syn}(t) + \sum_j I^{p,l}_j(t) \right) - (V^{p,l}(t-dt) - V_{reset}) S^{p,l}_r(t-dt). \tag{6.8}$$

Here, $V^{p,l}(t)$ decays according to k_m (ms^{-1}) and integrates current based on resistance R_m (GΩ). Voltage also tends toward the reset voltage V_{reset} (mV) at a rate proportional to the firing rate $s^{p,l}$. This is a continuous equivalent of the discrete voltage reset.

A summary of these equations and the states and parameters describing the GLIFR equations is in Tables 1, 3, and 4.

6.1.2 Effect of σ_V on Model Behavior. As described earlier, the parameter σ_V controls the smoothness of the relationship between voltage and firing rate. As σ_V approaches 0, the GLIFR equations in the continuous space approach the equivalent GLIF equations in the discrete space. This is established in equations 6.9 to 6.11:

Table 4: List of Parameters Describing GLIFR Networks, along with Their Units.

Spiking related		
$V_{th}^{p,l}$	threshold	mV
σ_V	smoothness	mV
Voltage related		
$k_m^{p,l}$	voltage decay factor	ms^{-1}
R_m	membrane resistance	GΩ
V_{reset}	reset voltage	mV
I_0	baseline current	pA
After-spike current related		
$a_j^{p,l}$	additive constant in after-spike current	pA
$r_j^{p,l}$	multiplicative constant in after-spike current	unitless
$k_j^{p,l}$	decay factor for after-spike current	ms^{-1}
Synaptic current related		
W_{in}^{pq}	input weight	pA
W_{lat}^{pq}	lateral weight	pA

Note: Here, indices p, l refer to neuron p in layer l.

$$\lim_{\sigma_V \to 0} s^{p,l}(t) = \lim_{\sigma_V \to 0} \frac{1}{1 + e^{-(V^{p,l}(t) - V_{th}^{p,l})/\sigma_V}} = \begin{cases} 1 & V^{p,l}(t) > V_{th}^{p,l} \\ 0 & V^{p,l}(t) < V_{th}^{p,l} \end{cases} = \mathbb{H}(V^{p,l}(t) - V_{th}^{p,l}) \tag{6.9}$$

$$\begin{aligned}\lim_{\sigma_V \to 0} V^{p,l}(t) &= \lim_{\sigma_V \to 0} \Bigg(V^{p,l}(t-dt)(1 - k_m^{p,l}dt) \\ &\quad + R_m^{p,l} k_m^{p,l} \left(I_0^{p,l} + I_{syn}^{p,l}(t) + \sum_j I_j^{p,l}(t) \right) dt \\ &\quad - \frac{dt}{\tau} s^{p,l}(V^{p,l}(t-dt) - V_{reset}) \Bigg) \\ &= V^{p,l}(t-dt)(1 - k_m^{p,l}dt) + R_m^{p,l} k_m^{p,l} \left(I_0^{p,l} + I_{syn}^{p,l}(t) + \sum_j I_j^{p,l}(t) \right) dt \\ &\quad - \frac{dt}{\tau} \lim_{\sigma_V \to 0} (s^{p,l})(V^{p,l}(t-dt) - V_{reset}) \\ &= V^{p,l}(t-dt)(1 - k_m^{p,l}dt) + R_m^{p,l} k_m^{p,l} \left(I_0^{p,l} + I_{syn}^{p,l}(t) + \sum_j I_j^{p,l}(t) \right) dt \\ &\quad - \frac{dt}{\tau} \mathbb{H}(V^{p,l}(t) - V_{th}^{p,l})(V^{p,l}(t-dt) - V_{reset})\end{aligned} \tag{6.10}$$

$$\begin{aligned}\lim_{\sigma_V\to 0} I_j^{p,l}(t) &= \lim_{\sigma_V\to 0}\Big(I_j^{p,l}(t-dt)(1-k_j^{p,l}dt)\\ &\quad + (a_j^{p,l} + r_j^{p,l} I_j^{p,l}(t-dt))\frac{dt}{\tau} s^{p,l}(t-dt)\Big)\\ &= \lim_{\sigma_V\to 0}\Big(I_j^{p,l}(t-dt)(1-k_j^{p,l}dt)\\ &\quad + (a_j^{p,l} + r_j^{p,l} I_j^{p,l}(t-dt))\frac{dt}{\tau}\lim_{\sigma_V\to 0}(s^{p,l}(t-dt))\Big)\\ &= \lim_{\sigma_V\to 0}\Big(I_j^{p,l}(t-dt)(1-k_j^{p,l}dt)\\ &\quad + (a_j^{p,l} + r_j^{p,l} I_j^{p,l}(t-dt))\frac{dt}{\tau}\mathbb{H}(V^{p,l}(t) - V_{th}^{p,l})\Big) \qquad (6.11)\end{aligned}$$

6.1.3 Discretization of GLIFR Equations. In order to compute efficiently across large networks of GLIFR neurons, we discretize the equations defined previously for the GLIFR model so that the required computations are approximated using forward Euler. The resulting equations are defined in equations 6.12 to 6.15:

$$S^{p,l}[t] = \frac{1}{1 + e^{-(V^{p,l}[t] - V_{th}^{p,l})/\sigma_V}}, \qquad (6.12)$$

$$\begin{aligned}V^{p,l}[t] &= V^{p,l}[t-dt](1-k_m^{p,l}dt) + R_m^{p,l}k_m^{p,l}\Big(I_0^{p,l} + I_{syn}^{p,l}[t] + \sum_j I_j^{p,l}[t]\Big)dt\\ &\quad - S_r^{p,l}[t]dt(V^{p,l}[t-dt] - V_{reset}), \qquad (6.13)\end{aligned}$$

$$I_j^{p,l}[t] = I_j^{p,l}[t-dt](1-k_j^{p,l}dt) + (a_j^{p,l} + r_j^{p,l}I_j^{p,l}[t-dt])S_r^{p,l}[t-dt]dt \qquad (6.14)$$

$$I_{syn}^{q}[t] = \sum_p W_{in}^{pq}S^{p,l-1}[t-dt] + \sum_p W_{lat}^{pq}S^{p,l}[t-\Delta t]. \qquad (6.15)$$

6.2 GLIFR Optimization. We use standard gradient descent to optimize networks of GLIFR neurons, training both synaptic weights and parameters in each neuron separately, similar to Perez-Nieves et al. (2021). In this way, the network can learn heterogeneous parameters across neurons.

We introduce several changes in variables to enable training multiplicative terms and promote stability. Because multiplicative terms can pose challenges for optimization by gradient descent, we keep resistance fixed and do not optimize it. Instead, while we do optimize k_m, we optimize the product $W_{pq}R_mk_qdt$ rather than W_{pq} (for either incoming weight or lateral weights) or R_m individually, and we refer to this product as ω_{pq}. Thus, the

Table 5: Trained Parameters.

$V_{th}^{p,l}$	threshold	mV
$\omega_{in}^{pq}, \omega_{lat}^{pq}$	combined synaptic weights	mV
$a_j^{p,l}$	additive constant in after-spike current	pA
$r_j^{p,l,*}$	transformed multiplicative constant in after-spike current	unitless
$k_j^{p,l,*}$	transformed decay factor for after-spike current	ms^{-1}
$k_m^{p,l,*}$	transformed voltage decay factor	ms^{-1}

Notes: List of trainable parameters in GLIFR networks, along with their units. Here, indices p, l refer to neuron p in layer l, so that individual neuron parameters are trained separately.

voltage and synaptic current equations can be combined as in the following equation:

$$\begin{aligned} V^{p,l}[t] &= V^{p,l}[t-dt](1-k_m^{p,l}dt) + R_m^{p,l}k_m^{p,l}\left(I_0^{p,l} + \sum_j I_j^{p,l}[t]\right)dt \\ &\quad + \sum_p \omega_{in}^{pq} S^{p,l-1}[t-dt] + \sum_p \omega_{lat}^{pq} S^{p,l}[t-\Delta t] \\ &\quad - S_r^{p,l}[t]dt(V^{p,l}[t-dt] - V_{reset}). \end{aligned} \tag{6.16}$$

We also add biological constraints to several parameters to maintain stability through training. In biological systems, k_m and k_j are restricted to be positive since they represent rates of decay. Moreover, to maintain stability, decay factors should be kept between 0 and $\frac{1}{dt}$ throughout training. To achieve this constraint without introducing discontinuities in the loss function, we optimize $t^* = \ln(\frac{kdt}{1-kdt})$ for each decay factor k. Thus, t^* can be any real number, whereas $k = \text{sigmoid}(t^*) \cdot \frac{1}{dt}$ is the decay factor we use when simulating neural activity. In this way, the value of k that is used is always between 0 and $\frac{1}{dt}$.

The term r_j can also introduce instability, so we use a similar method to constrain r_j to $[-1, 1]$. We optimize $t^* = \ln(\frac{1-r_j}{1+r_j})$ and retrieve r_j using $r_j = 1 - 2 \cdot \text{sigmoid}(t^*)$, which is bounded between -1 and 1. Other instabilities may be encountered, for example, by large synaptic weights, but we find these approaches sufficient for training networks of GLIFR neurons.

The parameters we optimize-after making the above changes are listed in Table 5. When training on complex tasks, we create networks of neurons, each consisting of a single hidden layer of GLIFR neurons whose outputs at each time point are passed through a fully connected linear transformation to yield the final output. The weights and biases used for this linear transformation are additionally optimized.

Table 6: Parameter Initialization.

Parameter	Homogeneous Initialization
V_{th}	1 mV
ω^{pq}	$U(-\frac{1}{N}, \frac{1}{N})$ mV
a_j^p	$U(-0.01, 0.01)$ pA
r_j^p	$U(-0.01, 0.01)$
k_j^p	$\frac{0.1}{dt}$
k_m^p	dt

Notes: This table describes how each trainable parameter was initialized in GLIFR neurons either homogeneously or heterogeneously across neurons. Note that in both schemes, ω, r_j, and k_j were initialized with at least some amount of heterogeneity.

6.3 Evaluation Setup. When optimizing our model on tasks, we use multiple network types. First, as a baseline, we use a vanilla recurrent neural network (RNN). The neurons used by RNNs are modeled as in equation 6.17. Here, W_{ih} represents the weights applied to the input signal, W_{hh} refers to the recurrent weights, and b refers to the bias term. In contrast to GLIFR neurons, we set the synaptic delay Δt to dt in RNNs for some tasks in order to enable the modeling of nonlinear transformations at the beginning of a simulation. In other words, instead of incorporating dynamics from several timesteps prior, the synaptic delay is a single timestep in duration. This enables the RNN to perform better than it would if we used $\Delta t = 1$ ms:

$$h(t) = \tanh(W_{ih}x(t) + W_{hh}h(t - \Delta t) + b). \tag{6.17}$$

We additionally study multiple variations of the GLIFR networks, varying in whether parameters are homogeneously initialized and whether parameters are learned over training. When initializing homogeneously, we employ a deterministic initialization scheme to decrease the likelihood that the network will learn parameters underlying exploding or saturating firing rates; specifically, we initialize the network with small decay constants and thresholds close to the reset voltage of 0 mV. The homogeneous initialization scheme is described in Table 6. Note that the neuron described is in a layer with N neurons. When initializing heterogeneously, we initialize both parameters and input and lateral weights by randomly sampling from the distribution of parameters and weights in a trained LHet or LHetA model. This also decreases the likelihood of learning parameters that lead to exploding or saturating firing rates while erasing any learned patterns in the parameters or connectivity patterns in the network.

Table 7: Size of Networks.

	Sine Generation	L-MNIST	P-MNIST	N-MNIST
RNN	128(16,769)	256(75,530)	256(75,530)	256(75,530)
LSTM	63(16,696)	122(75,406)	122(75,406)	122(75,406)
Hom	128(16,641)	256(75,274)	256(75,274)	256(75,274)
HomA	128(16,641)	256(75,274)	256(75,274)	256(75,274)
LHet	127(16,638)	255(75,235)	255(75,235)	255(75,235)
LHetA	124(16,617)	252(75,106)	252(75,106)	252(75,106)
FHet	128(16,641)	256(75,274)	256(75,274)	256(75,274)
FHetA	128(16,641)	256(75,274)	256(75,274)	256(75,274)
RHet	127(16,638)	255(75,235)	255(75,235)	255(75,235)
RHetA	124(16,617)	252(75,106)	252(75,106)	252(75,106)

Notes: This table lists the size of networks trained on each task. These are listed as number of neurons (number of parameters).

Table 8: Hyperparameters.

	Sine Generation	L-MNIST	P-MNIST	N-MNIST
Model hyperparameters				
Δt (GLIFR/RNN)	1 ms/0 ms	1 ms/0 ms	15 ms/15 ms	0 ms/0 ms
dt	0.05 ms	0.05 ms	0.05 ms	1 ms
Optimization technique				
Loss function	Mean-squared error		Cross-entropy loss	
Technique	Adam optimization (Kingma & Ba, 2014)			
Optimization hyperparameters				
Learning rate	0.0001	0.001	0.0001	0.001
Betas	(0.9, 0.999)			
Batch size	6	256	512	512
Number of epochs	5000	30	50	30

Notes: This table lists the hyperparameters used for each task. The only hyperparameter that varies between networks is the synaptic delay (Δt), and this is denoted as (Δt for GLIFR and LSTM/Δt for RNN).

6.4 Tasks. In this work, we explore multiple tasks, analyzing and comparing the performance of RNNs, LSTMs, and variations of GLIFR networks. The chosen tasks range in complexity. When training networks on these tasks, we parameter-match them, thus using the hidden sizes listed in Table 7. The hyperparameters used for each task are listed in Table 8. Below, we discuss each task in detail.

6.4.1 Sine Generation Task. In the sine generation task, each network was trained to generate 5 ms sinusoidal patterns whose frequencies are determined by the amplitudes of the corresponding constant inputs. We trained each network to produce six sinusoids with frequencies ranging from 80 Hz to 600 Hz. The *i*th sinusoid was associated with a constant input of amplitude $(i/6) + 0.25$. Previous work (Sussillo & Barak, 2013) tackles a more complex version of this task using RNNs but uses Hessian-free optimization, while we limit our training technique on both the RNN and the GLIFR networks to simple gradient descent.

6.4.2 Line-by-Line MNIST (L-MNIST) Task. In the L-MNIST task, a network takes in a 28-dimensional input at each timestep corresponding to a row of a MNIST image, which is sized 28×28. At each of the 28 timesteps, the network produces a 10-dimensional output (1 dimension for each digit 0–9), whose softmax would represent a one-hot encoding of the digit. The network is trained so that the 10-dimensional output at the last time point correctly indicates the input digit.

6.4.3 Pixel-by-Pixel MNIST (P-MNIST) Task. We analyze a variation of the L-MNIST task in which at each timestep, a network takes in a 1-dimensional input corresponding to a single pixel of an MNIST image. Thus, the number of timesteps processed for a single sample is $28 \times 28 = 784$. Similar to the L-MNIST setup, the network produces a 10-dimensional output for each timestep, and the network is trained so that the output at the last time point correctly indicates the input digit.

6.4.4 Neuromorphic MNIST (N-MNIST) Task. In the N-MNIST task, a network is trained to identify the digit represented by an MNIST image captured by a dynamic vision sensor (DVS) (Orchard et al., 2015). The image is smoothly moved in front of a DVS camera, while for each frame, the DVS camera senses changes in pixel intensity in two directions and translates these into spiking events. As a result, a single image is represented by a $34 \times 34 \times 2 \times T$ matrix, where T is the number of frames and each image is 34×34. Due to the large T accompanying samples from this data set, we employ the strategy taken in He et al. (2020). Specifically, for a timestep duration dt, we summarize each consecutive nonoverlapping window of length dt by computing $sign(\sum_t X_{ijk}(t))$ for each pixel coordinates i, j and channel k. We define $sign(x)$ to be 0 for $x = 0$, positive for $x > 0$, and negative for $x < 0$. After summarizing each consecutive window, we use the first 15 timesteps for training.

6.5 Hardware. To accelerate training networks on the L-MNIST and P-MNIST tasks, we use the NVIDIA RTX 2080Ti GPU (two GPUs and eight CPUs). With GPU acceleration, one epoch on the L-MNIST task takes under 1 minute, and one epoch on the P-MNIST task takes approximately 2 minutes. We trained the sine wave generation task, which was and the N-MNIST task, which took around 20 minutes per epoch, on CPU only.

7 Supplementary Data

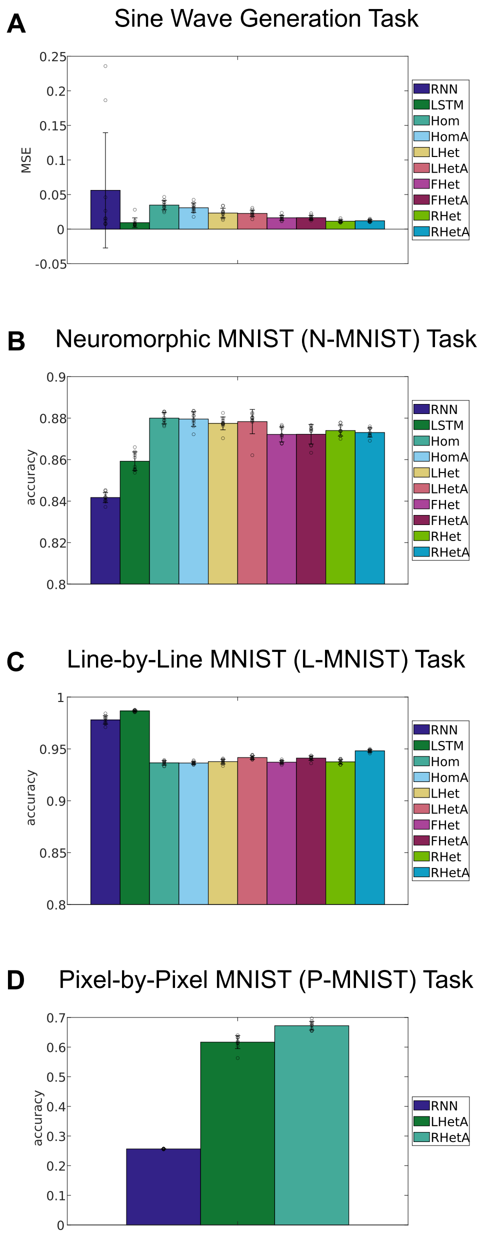


Figure 10: Distribution of final loss. For each result listed in Table 2, the distribution of MSEs/accuracies across the 10 trials conducted is displayed.

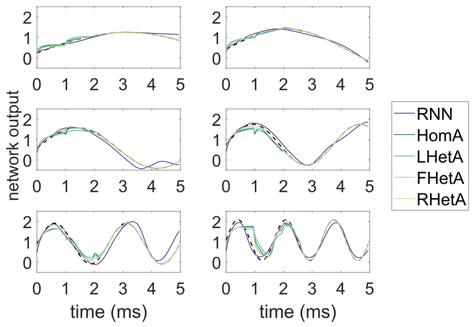


Figure 11: Predictions on sine wave generation task. A representative trace is shown for the predictions made by five networks trained on the sine wave generation task. The black dashed line denotes the target pattern. All network types perform reasonably on this task.

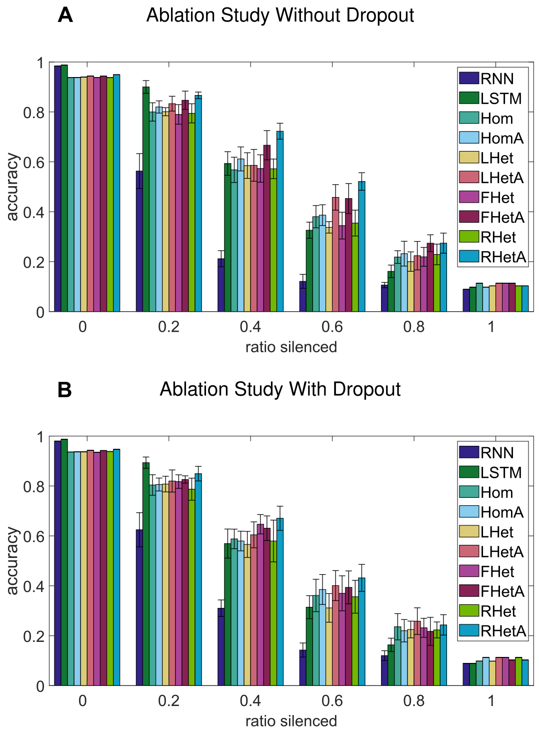
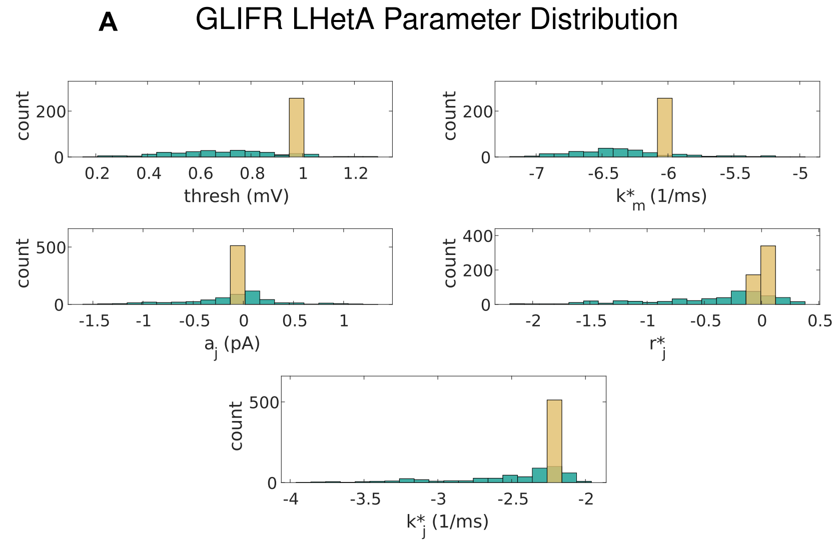


Figure 12: Effect of dropout on ablation results. Testing accuracy under random neuron silencing is recorded when dropout is used during training (B) and when it is not applied (A).



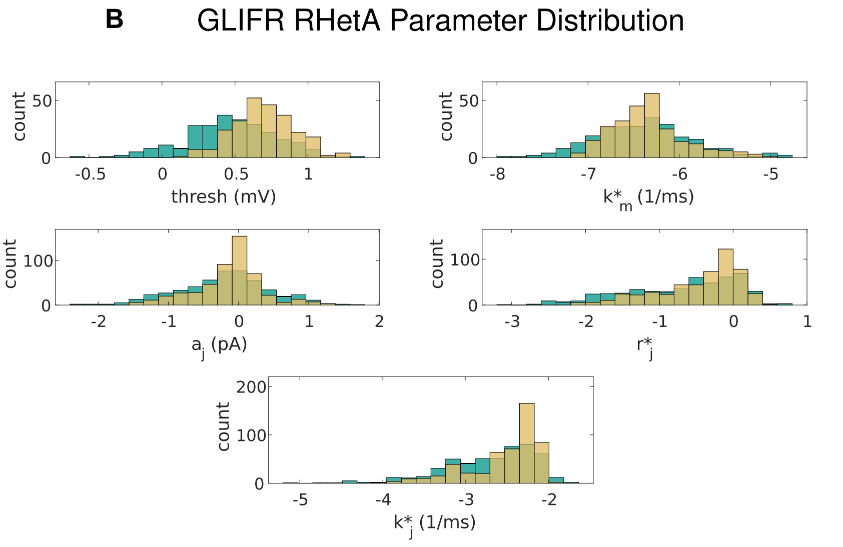


Figure 13: A–D. Learned parameter distributions. The initial (yellow) and learned (green) distributions of parameters across neurons in single example networks are shown for the L-MNIST task. Results are shown for both homogeneous initialization (A) and heterogeneous initialization (B). Under both initialization schemes, heterogeneity in neuronal parameters is learned.

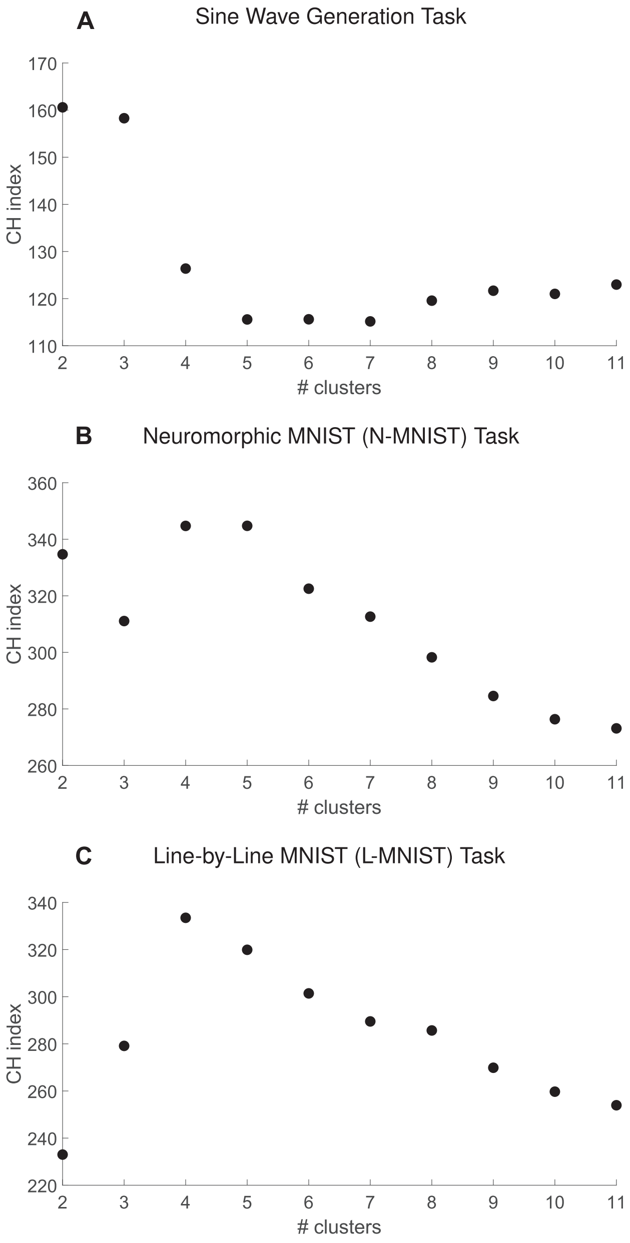


Figure 14: The CH scores for different cluster numbers are plotted for the parameters learned on the sine wave generation task (A), the N-MNIST task (B), and the L-MNIST task (C), demonstrating optimal cluster counts of one, five, and four, respectively.

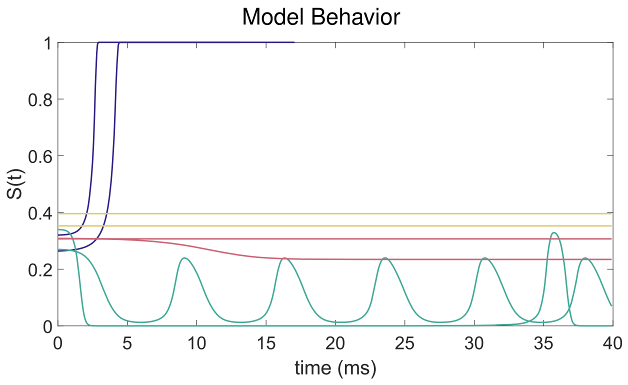


Figure 15: Sample responses. Two neurons from each cluster in the LHetA network trained on the L-MNIST task were chosen and exposed to a constant stimulus of magnitude 0.001 pA. The resulting $S(t)$ is plotted here. The colors represent the different clusters of neurons as identified in Figure 9.

Acknowledgments

We acknowledge the following sources of funding and support: the University of Washington Mary Gates Research Endowment, NIH training grant T90 DA 32436-10, and NIH BRAIN initiative grant 1RF1DA055669. We thank the Allen Institute founder, Paul G. Allen, for his vision, encouragement, and support.

References

Beiran, M., & Ostojic, S. (2019). Contrasting the effects of adaptation and synaptic filtering on the timescales of dynamics in recurrent networks. *PLOS Computational Biology*, *15*(3), e1006893. 10.1371/journal.pcbi.1006893

Bellec, G., Scherr, F., Subramoney, A., Hajek, E., Salaj, D., Legenstein, R., & Maass, W. (2020). A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature Communications*, *11*(1). 10.1038/s41467-020-17236-y

Billeh, Y. N., Cai, B., Gratiy, S. L., Dai, K., Iyer, R., Gouwens, N. W., . . . Arkhipov, A. (2020). Systematic integration of structural and functional data into multi-scale models of mouse primary visual cortex. *Neuron*, *106*(3), 388–403. 10.1016/j.neuron.2020.01.040, PubMed: 32142648

Burnham, D., Shea-Brown, E., & Mihalas, S. (2021). *Learning to predict in networks with heterogeneous and dynamic synapses.* bioRxiv.

Fontaine, B., Peña, J. L., & Brette, R. (2014). Spike-threshold adaptation predicted by membrane potential dynamics in vivo. *PLOS Computational Biology*, *10*(4), e1003560. 10.1371/journal.pcbi.1003560

Geadah, V., Kerg, G., Horoi, S., Wolf, G., & Lajoie, G. (2020). *Advantages of biologically-inspired adaptive neural activation in RNNs during learning*. arXiv:2006.12253.

Gerstner, W., Kistler, W. M., Naud, R., & Paninski, L. (2014). *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

He, W., Wu, Y., Deng, L., Li, G., Wang, H., Tian, Y., . . . Xie, Y. (2020). Comparing SNNs and RNNs on neuromorphic vision datasets: Similarities and differences. *Neural Networks*, *132*, 108–120. 10.1016/j.neunet.2020.08.001, PubMed: 32866745

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). *Improving neural networks by preventing co-adaptation of feature detectors*. arXiv:1207.0580.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. 10.1162/neco.1997.9.8.1735, PubMed: 9377276

Hodgkin, A. L., & Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, *117*(4). 10.1113/jphysiol.1952.sp004764

Hu, B., Garrett, M. E., Groblewski, P. A., Ollerenshaw, D. R., Shang, J., Roll, K., . . . Mihalas, S. (2021). Adaptation supports short-term memory in a visual change detection task. *PLOS Computational Biology*, *17*(9). 10.1371/journal.pcbi.1009246

Huh, D., & Sejnowski, T. J. (2018). Gradient descent for spiking neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems*, *31*. Curran.

Hunsberger, E., Scott, M., & Eliasmith, C. (2014). The competing benefits of noise and heterogeneity in neural coding. *Neural Computation*, *26*(8), 1600–1623. 10.1162/NECO_a_00621, PubMed: 24877735

Izhikevich, E. (2003). Simple model of spiking neurons. *IEEE Transactions on Neural Networks*, *14*(6), 1569–1572. 10.1109/TNN.2003.820440, PubMed: 18244602

Kingma, D. P., & Ba, J. (2014). *Adam: A method for stochastic optimization.* arXiv:1412.6980.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. 10.1038/nature14539, PubMed: 26017442

Li, S., Li, W., Cook, C., Zhu, C., & Gao, Y. (2018). *Independently recurrent neural network (INDRNN): Building a longer and deeper RNN.* arXiv:1803.04831.

Liu, P., Qiu, X., & Huang, X. (2016). *Recurrent neural network for text classification with multi-task learning.* arXiv:1605.05101.

Markram, H., Muller, E., Ramaswamy, S., Reimann, M. W., Abdellah, M., Sanchez, C. A., . . . Schürmann, F. (2015). Reconstruction and simulation of neocortical microcircuitry. *Cell*, *163*(2), 456–492. 10.1016/j.cell.2015.09.029

Mihalaş, Ş., & Niebur, E. (2009). A generalized linear integrate-and-fire neural model produces diverse spiking behaviors. *Neural Computation*, *21*(3), 704–718. 10.1162/neco.2008.12-07-680

Morris, C., & Lecar, H. (1981). Voltage oscillations in the barnacle giant muscle fiber. *Biophysical Journal*, *35*(1). 10.1016/S0006-3495(81)84782-0

Muscinelli, S. P., Gerstner, W., & Schwalger, T. (2019). How single neuron properties shape chaotic dynamics and signal transmission in random neural networks. *PLOS Computational Biology*, *15*(6), e1007122. 10.1371/journal.pcbi.1007122

Neftci, E. O., Mostafa, H., & Zenke, F. (2019). Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, *36*(6). 10.1109/MSP.2019.2931595

Orchard, G., Jayawant, A., Cohen, G. K., & Thakor, N. (2015). Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in Neuroscience*, *9*, 437. 10.3389/fnins.2015.00437

Perez-Nieves, N., Leung, V. C., Dragotti, P. L., & Goodman, D. F. (2021). Neural heterogeneity promotes robust learning. *Nature Communications*, *12*(1). 10.1038/s41467-021-26022-3

Prinz, A. A., Bucher, D., & Marder, E. (2004). Similar network activity from disparate circuit parameters. *Nature Neuroscience*, *7*(12), 1345–1352. 10.1038/nn1352, PubMed: 15558066

Rajan, K., Harvey, C. D., & Tank, D. W. (2016). Recurrent network models of sequence generation and memory. *Neuron*, *90*(1). 10.1016/j.neuron.2016.02.009

Salaj, D., Subramoney, A., Kraisnikovic, C., Bellec, G., Legenstein, R., & Maass, W. (2021). Spike frequency adaptation supports network computations on temporally dispersed information. *eLife*, *10*. 10.7554/eLife.65459

Shamir, M., & Sompolinsky, H. (2006). Implications of neuronal diversity on population coding. *Neural Computation*, *18*(8), 1951–1986. 10.1162/neco.2006.18.8.1951, PubMed: 16771659

She, X., Dash, S., Kim, D., & Mukhopadhyay, S. (2021). A heterogeneous spiking neural network for unsupervised learning of spatiotemporal patterns. *Frontiers in Neuroscience*, *14*, 1406. 10.3389/fnins.2020.615756

Sussillo, D., & Barak, O. (2013). Opening the black box: Low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Computation*, *25*(3), 626–649. 10.1162/NECO_a_00409, PubMed: 23272922

Teeter, C., Iyer, R., Menon, V., Gouwens, N., Feng, D., Berg, J., . . . Mihalas, S. (2018). Generalized leaky integrate-and-fire models classify multiple neuron types. *Nature Communications*, *9*(1). 10.1038/s41467-017-02717-4

Tripathy, S. J., Padmanabhan, K., Gerkin, R. C., & Urban, N. N. (2013). Intermediate intrinsic diversity enhances neural population coding. In *Proceedings of the National Academy of Sciences*, *110*(20), 8248–8253. 10.1073/pnas.1221214110

Wunderlich, T. C., & Pehle, C. (2021). Event-based backpropagation can compute exact gradients for spiking neural networks. *Scientific Reports*, *11*(1), 1–17. 10.1038/s41598-021-91786-z, PubMed: 34145314

Yamins, D. L. K., Hong, H., Cadieu, C. F., Solomon, E. A., Seibert, D., & DiCarlo, J. J. (2014). Performance-optimized hierarchical models predict neural responses in higher visual cortex. In *Proceedings of the National Academy of Sciences*, *111*(23), 8619–8624. 10.1073/pnas.1403112111

Zenke, F., & Vogels, T. P. (2021). The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *Neural Computation*, *33*(4), 899–925. 10.1162/neco_a_01367, PubMed: 33513328

Received January 19, 2022; accepted November 2, 2022.