Learning the Synaptic and Intrinsic Membrane Dynamics Underlying Working Memory in Spiking Neural Network Models

Yinghao Li
yil574@ucsd.edu
Computational Neurobiology Laboratory, Salk Institute for Biological Studies, La Jolla, CA 92037, U.S.A.

Robert Kim
r8kim@health.ucsd.edu
Computational Neurobiology Laboratory, Salk Institute for Biological Studies, La Jolla, CA 92037, and Neurosciences Graduate Program and Medical Scientist Training Program, University of California San Diego, La Jolla, CA 92093, U.S.A.

Terrence J. Sejnowski
terry@salk.edu
Computational Neurobiology Laboratory, Salk Institute for Biological Studies, La Jolla, CA 92037, and Institute for Neural Computation and Division of Biological Sciences, University of California San Diego, La Jolla, CA 92093, U.S.A.

Recurrent neural network (RNN) models trained to perform cognitive tasks are a useful computational tool for understanding how cortical circuits execute complex computations. However, these models are often composed of units that interact with one another using continuous signals and overlook parameters intrinsic to spiking neurons. Here, we developed a method to directly train not only synaptic-related variables but also membrane-related parameters of a spiking RNN model. Training our model on a wide range of cognitive tasks resulted in diverse yet task-specific synaptic and membrane parameters. We also show that fast membrane time constants and slow synaptic decay dynamics naturally emerge from our model when it is trained on tasks associated with working memory (WM). Further dissecting the optimized parameters revealed that fast membrane properties are important for encoding stimuli, and slow synaptic dynamics are needed for WM maintenance. This approach offers a unique window into how connectivity patterns and intrinsic neuronal properties contribute to complex dynamics in neural populations.

1 Introduction

Neurons in the cortex form recurrent connections that give rise to the complex dynamic processes underlying computational functions (Goldman-Rakic, 1995; Chen & Aihara, 1995; Douglas & Martin, 2007; Wang, 2008). Previous studies have used models based on recurrent neural networks (RNNs) of continuous-rate units to characterize network dynamics behind neural computations and to validate experimental findings (Sompolinsky, Crisanti, & Sommers, 1988; Sussillo & Abbott, 2009; Rajan, Abbott, & Sompolinsky, 2010; Mante, Sussillo, Shenoy, & Newsome, 2013; Mastrogiuseppe & Ostojic, 2018; Rajan, Harvey, & Tank, 2016). However, these models do not explain how intrinsic membrane properties could also contribute to the emerging dynamics.

Rate-based encoding of information has been reliably observed in experimental settings (Mante et al., 2013). However, recent studies demonstrated that membrane potential dynamics along with spike-based coding are also capable of reliably transmitting information (VanRullen, Guyonneau, & Thorpe, 2005; Sippy, Lapray, Crochet, & Petersen, 2015; Pala & Petersen, 2018). In addition, the intrinsic membrane properties of inhibitory neurons, including the membrane time constant and rheobase (minimum current required to evoke a single action potential), were different in two higher-order cortical areas (Medalla, Gilman, Wang, & Luebke, 2017). These findings strongly indicate that neuronal intrinsic properties, often ignored in previous computational studies employing rate-based RNNs, are crucial for better understanding how distinct subtypes of neurons contribute to information processing.

Rate-based RNNs can be easily trained by stochastic gradient descent to perform specified cognitive tasks (Rumelhart, Hinton, & Williams, 1988). However, similar supervised learning methods cannot be used to train spiking RNNs due to the nondifferentiable behavior of action potentials (Tavanaei, Ghodrati, Kheradpisheh, Masquelier, & Maida, 2018). Thus, several methods have introduced differentiable approximations of the nondifferentiable spiking dynamics (Lee, Delbruck, & Pfeiffer, 2016; Huh & Sejnowski, 2018; Zhang & Li, 2019; Neftci, Mostafa, & Zenke, 2019). These studies directly applied backpropagation to tune synaptic connections for task-specific computations. Other methods that do not rely on gradient computations have been also utilized to train spiking networks. One such method is based on the first-order reduced and controlled error (FORCE) algorithm previously developed for rate RNNs (Sussillo & Abbott, 2009). The FORCE-based methods are capable of training spiking networks, but training all the parameters including recurrent connections could become computationally inefficient (Kim & Chow, 2018; Thalmeier, Uhlmann, Kappen, & Memmesheimer, 2016; Nicola & Clopath, 2017). Finally, recent studies successfully converted rate-based networks trained with a gradient-descent method to spiking networks for both convolutional and recurrent neural networks (Sengupta, Ye, Wang, Liu, & Roy, 2019; Kim, Li, & Sejnowski, 2019). Since these models are built on rate-coding networks, the resulting spiking models do not take advantage of the rich spiking dynamics. Moreover, these previous models assume that all the units in a trained network are equivalent, even though experimental evidence shows that neurons in biological neural networks are highly heterogeneous. Such diversity has a vital role in efficient neural coding (Chelaru & Dragoi, 2008).

Here, we present a new approach that can directly train not only recurrent synapses but also membrane-related parameters of a spiking RNN model. Our method utilizes mollifier functions (Ermoliev, Norkin, & Wets, 1995) to approximate the nondifferentiable gradient computation for discrete spiking dynamics, and a gradient-descent method is applied to tune the model parameters. These parameters are composed of synaptic parameters including recurrent connections and several important spiking-related parameters, such as membrane time constant and action potential threshold. Neurons with diverse and heterogeneous intrinsic parameters emerged from training our spiking model on a wide range of cognitive tasks. Furthermore, we observed that both synaptic and spiking parameters worked in a synergistic manner to perform complex tasks that required information integration and working memory.

2 Results

Here, we provide an overview of the method that we developed to directly train spiking recurrent neural network (RNN) models (for more details see section 4). Throughout the study, we considered recurrent network models composed of leaky integrate-and-fire (LIF) units whose membrane voltage dynamics were governed by

$$\tau_{m,i}\frac{dv_i}{dt} = -\left(v_i(t) - v_{\mathrm{rest}_i}\right) + R_i I_i(t), \tag{2.1}$$

where $\tau_{m,i}$ is the membrane time constant of unit i, $v_i(t)$ is the membrane voltage of unit i at time t, $v_{\mathrm{rest},i}$ is the resting potential of unit i, and R_i is the input resistance of unit i. $I_i(t)$ represents the current input to unit i at time t, which is given by

$$I_i(t) = \sum_{j=1}^{N} s_{ij}(t) + I_{\mathrm{ext}_i}(t), \tag{2.2}$$

where N is the total number of units in the network, $s_{ij}(t)$ is the synaptic input from unit j to unit i at time t, and $I_{\mathrm{ext},i}(t)$ is the external current source

Table 1: Parameter Values Used for This Study.

Parameter Name	Symbol	Minimum	Maximum
Input resistance	R	5 MΩ	1000 MΩ
Membrane time constant	τ_m	5 ms	50 ms
Action potential threshold	ϑ	−50 mV	−30 mV
Resting potential	v_{rest}	−80 mV	−60 mV
Reset voltage value	v_{reset}	$v_{\text{rest}} - 10$ mV	$v_{\text{rest}} - 1$ mV
Synaptic decay time	τ	5 ms	100 ms

Note: To keep the constraint $v_{\text{rest}} > v_{\text{reset}}$, we trained the afterhyperpolarization (AHP) potential with range from −10 mV to −1 mV, so the value of v_{reset} is dependent on the value of v_{rest}.

into unit i at time t. We used a single exponential synaptic filter to model the synaptic input (s),

$$\tau_{ij}\frac{ds_{ij}}{dt} = -s_{ij}(t) + \sum_{t_j^{(k)}<t} w_{ij}\delta(t - t_j^{(k)}), \tag{2.3}$$

where τ_{ij} is the decay time constant of the synaptic current from unit j to unit i, w_{ij} is the synaptic strength from unit j to unit i, $t_j^{(k)}$ denotes the time of the kth action potential of unit j, and $\delta(x)$ is the Dirac delta function. Once the membrane voltage of the unit i crosses its action potential threshold (ϑ_i), its membrane voltage is brought back down to its reset voltage ($v_{\text{reset},i}$).

Each LIF unit is characterized by five distinct parameters: membrane time constant ($\tau_{m,i}$), resting potential ($v_{\text{rest},i}$), input resistance (R_i), action potential threshold (ϑ_i), and reset potential ($v_{\text{reset},i}$). In addition, there are two trainable synaptic parameters: synaptic strength (w_{ij}) and synaptic decay time constant (τ_{ij}) from unit j to unit i.

In order to tune all the parameters described above to produce functional spiking RNNs capable of performing cognitive tasks, we employed the commonly used gradient-descent method known as backpropagation through time (BPTT; Werbos, 1990) with a few important modifications. We utilized mollifier gradient approximations to avoid the nondifferentiability problem associated with training spiking networks with backpropagation (Ermoliev et al., 1995). Furthermore, we optimized each of the model parameters (except for the synaptic connectivity weights) in a biologically plausible range (see section 4). We also employed the weight parameterization method proposed by Song et al. to impose Dale's principle (Song, Yang, & Wang, 2016a; see section 4). All the spiking RNN models trained in the study used the parameter value ranges listed in Table 1 unless otherwise noted.

2.1 Units with Diverse Parameter Values Emerge after Training. We applied our method to train spiking networks to perform the context-dependent input integration task previously employed by Mante et al. (2013). Briefly, Mante et al. trained rhesus monkeys to flexibly integrate sensory inputs (color and motion of randomly moving dots presented on a screen). A contextual cue was given to instruct the monkeys which sensory modality (color or motion) they should attend to. The monkeys were required to employ flexible computations as the same modality could be either relevant or irrelevant depending on the contextual cue. Several previous modeling studies have successfully implemented a simplified version of the task and reproduced the neural dynamics present in the experimental data with both continuous-rate RNNs and spiking RNNs converted from rate RNNs (Song et al., 2016a; Miconi, 2017; Kim et al., 2019). With our method, we were able to directly train the first, to our knowledge, spiking RNNs with heterogeneous units whose parameters were within biologically plausible limits.

In order to train spiking RNNs to perform the input integration task, we employed a task paradigm similar to the one used by previous computational studies (Mante et al., 2013; Song et al., 2016a; Miconi, 2017; Kim et al., 2019). A recurrently connected network received two streams of noisy input signals along with a constant-valued signal that encoded the contextual cue (see Figure 1A). The input signals were sampled from a standard gaussian distribution (i.e., with zero mean and unit variance) and then shifted by a positive or negative "offset" value to simulate the evidence presented in the input modalities. The network was trained to produce an output signal approaching either $+1$ or -1 depending on the cue and the evidence present in the input signal: if the cued input had a positive mean, the output signal approached $+1$, and vice versa (see Figure 1B, top). The input signal, 150 ms in duration, was given after a fixation period (300 ms), and the network was trained to produce an output signal immediately after the offset of the input signal.

We trained 20 spiking RNNs to perform the context-based input integration task. All of the trainable parameters were initialized with random numbers drawn from a standard gaussian distribution and rescaled to the biologically plausible ranges (see section 4 and Table 1). Each network was trained until the training termination criteria were satisfied (see section 4). On average, 508 ± 46 training trials were needed for a network to meet the training termination conditions. After training, a wide distribution of the parameters emerged for both excitatory and inhibitory populations (see Figure 1C, top).

Consistent with the previous experimental recordings from cortical neurons, the inhibitory units in our trained RNNs fired at a higher rate compared to the excitatory units (Peyrache et al., 2012). The higher average firing rates of the inhibitory units were largely due to the intrinsic properties that resulted from training. Compared to the excitatory population,

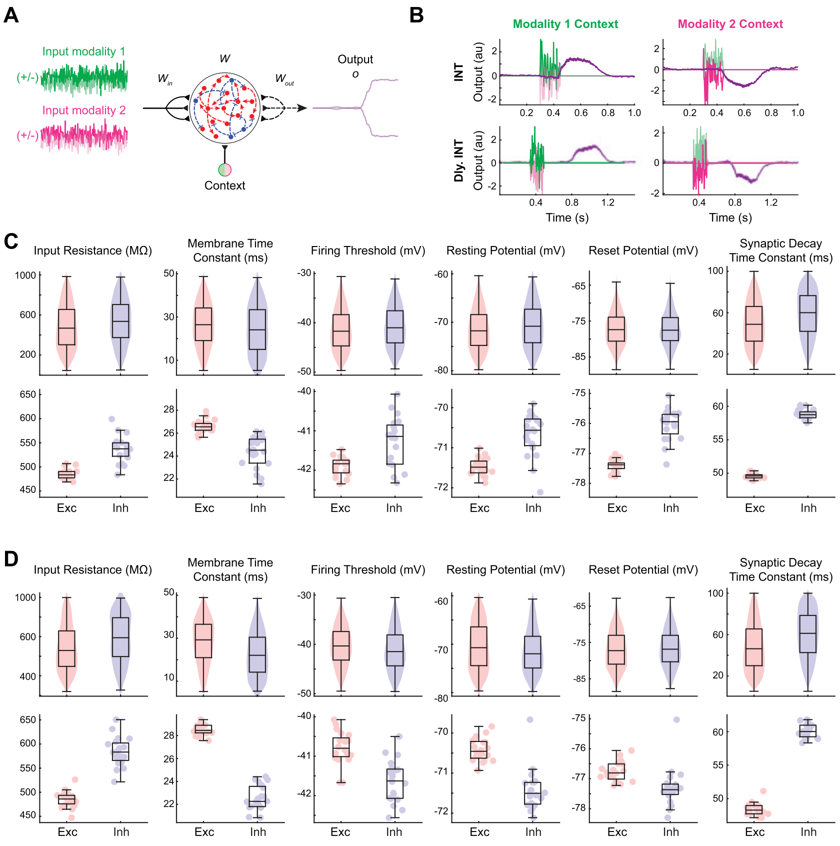

Figure 1: Biologically realistic spiking network performing a context-dependent input integration task. (A) Schematic diagram of the RNN model trained for the context-dependent integration task. Two streams of noisy input signals (green and magenta lines) along with a context signal were delivered to the LIF RNN. The network was trained to integrate and determine if the mean of the cued input signal (i.e., cued offset value) was positive (+ choice) or negative "−" choice) without or with a delay period at the end of the noisy input signals. (B) Example input and output signals from example RNNs trained to perform the task without (top row; INT) or with a delay period (bottom row; Dly. INT). (C) Distributions of the optimized parameters for the excitatory (red) and inhibitory (blue) units across all 20 models trained for the INT task. Top: Distributions pooled from all the units from 20 models. Bottom: Each dot represents the average value from one network. (D) Distributions of the optimized parameters for the excitatory (red) and inhibitory (blue) units across all 20 models trained for the Dly. INT task. Top: Distributions pooled from all the units from 20 models. Bottom: Each dot represents the average value from one network.

the inhibitory units in the trained RNNs had significantly larger input resistance, smaller membrane time constants, and more depolarized resting potential (see Figure 1C; $P < 0.0001$, two-sided Wilcoxon rank-sum test). The action potential thresholds and the reset potentials were significantly more depolarized for the inhibitory group. Furthermore, the time constants of the inhibitory synaptic current variable were significantly larger than the excitatory synaptic decay time constants (see Figure 1C).

2.2 Working Memory Requires Distinct Parameter Distributions. The context-dependent input integration task considered in the previous section did not require complex cognitive skills such as working memory (WM) computations. In order to explore what parameter values are essential for WM tasks, we modified the paradigm to incorporate a WM component by adding a delay period after the delivery of the input signals. The RNN model was trained to integrate the noisy input signals, sustain the integrated information throughout the 300 ms delay period, and produce an output signal (see Figure 1B, bottom). We again trained 20 models for the modified integration task with the same training termination criteria (see section 4). This task required more training trials (on average 1618 ± 346), but all the models were successfully trained within 2000 training trials.

Overall, the distributions of the trained parameters were similar to those observed from the RNNs trained on the non-WM version of the task (see Figure 1D). The parameters that were significantly different between the two RNN models were the membrane time constant and the synaptic decay time constant. The inhibitory units from the WM model displayed much faster membrane dynamics and slower synaptic decay compared to the inhibitory population of the non-WM model ($P < 0.0001$, two-sided Wilcoxon rank-sum test).

To ensure that the patterns of the trained parameters and the distinct distributions of the two parameters (τ_m and τ) observed from the delayed integration model were indeed associated with WM computations, we trained RNNs on two additional WM-related tasks: delayed matched-to-sample (DMS) and delayed discrimination (DIS) tasks. For each task, we again trained 20 RNNs. Both task paradigms included two sequential stimuli separated by a brief delay period. For the DMS task, the two input stimuli were either $+1$ or -1; if the two sequential had the same sign (i.e., $+1/+1$ or $-1/-1$), the network was trained to have an output signal approaching $+1$, while if the two stimuli had different signs (i.e., $+1/-1$ or $-1/+1$), the output signal approached -1 (see Figure 2A; see section 4). The two input stimuli for the DIS task were sinusoidal waves with different frequencies, modeled after the task used by Romo, Brody, Hernández, and Lemus (1999) where monkeys were trained to discriminate two vibratory stimuli. If the first stimulus had a higher (lower) frequency, our RNN model was trained to produce a positive (negative) output signal (see Figure 2B; see section 4).

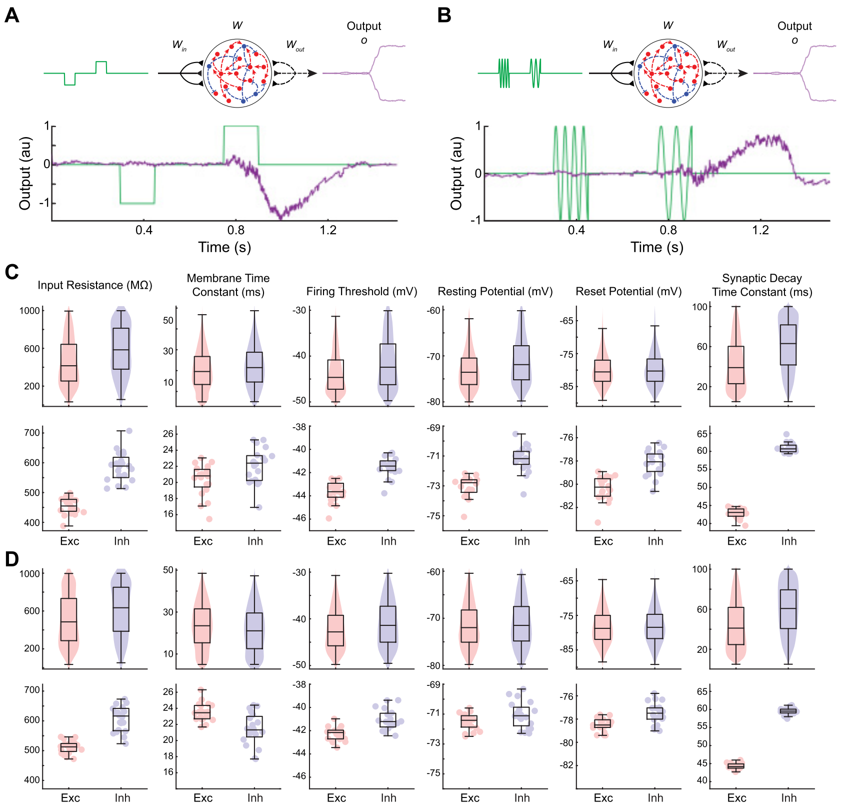

Figure 2: RNNs trained for two additional WM tasks. (A) Schematic illustrating the task paradigm for the delayed match-to-sample (DMS) task (top) and input and output signals from an example-trained RNN (bottom). (B) Schematic illustrating the task paradigm for the delayed discrimination (DIS) task (top) and input and output signals from an example-trained RNN (bottom). (C) Distributions of the optimized parameters for the excitatory (red) and inhibitory (blue) units across all 20 models trained for the DMS task. Top: Distributions pooled from all the units from 20 models. Bottom: Each dot represents the average value from one network. (D) Distributions of the optimized parameters for the excitatory (red) and inhibitory (blue) units across all 20 models trained for the DIS task. Top: Distributions pooled from all the units from 20 models. Bottom: Each dot represents the average value from one network.

It took longer to train our model on these two tasks compared to the delayed integration task (7104 ± 3739 trials for the DMS task and 6985 ± 2112 trials for the DIS task). The distributions of the tuned parameters from the two WM tasks were similar to the distributions obtained from the delayed

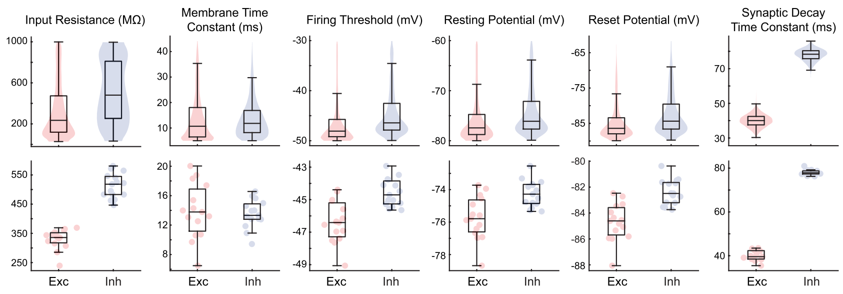

Figure 3: RNNs initialized with fixed parameter values were trained to perform the DMS task. We initialized all the trainable parameters to constant values (i.e., the average value of each parameter from previously trained models: $R_{\text{init}} = 628.44$ MΩ, $\tau_{m_{\text{init}}} = 21.92$ ms, $\theta_{\text{init}} = -42.32$ mV, $V_{\text{rest,init}} = -67.63$ mV, $V_{\text{reset,init}} = -74.49$ mV, $\tau_{\text{init}} = 64.16$ ms) and trained a group of RNNs ($n = 15$) to perform the DMS. Distributions of the optimized parameters for the excitatory (red) and inhibitory (blue) units across all 15 models trained for the DMS task. Top: Distributions pooled from all the units from 15 models. Bottom: Each dot represents the average value from one network.

integration task (see Figures 2C and 2D). More important, we again observed significantly faster membrane voltage dynamics and slower synaptic decay from the inhibitory units in the DMS and DIS models compared to the inhibitory units from the non-WM task. These findings strongly suggest that the two parameters (τ_m and τ) of the inhibitory group contribute to important dynamics associated with WM.

To ensure that the random initialization did not contribute to the heterogeneous distributions after training, we trained another group of RNNs with fixed initial values for the trainable parameters. The trained parameters from the RNNs trained with fixed initial parameter values still displayed heterogeneous values for each trained parameter, indicating that our initialization did not significantly contribute to the heterogeneity (see Figure 3).

2.3 Shared Intrinsic Properties across Different Working Memory Tasks. Prefrontal cortex and other higher-order cortical areas have been shown to integrate information in a flexible manner and switch between tasks seamlessly (Mante et al., 2013). Along this line of thought, we hypothesized that the intrinsic properties optimized for one WM task should be generalizable to other tasks that also require WM. In order to test this hypothesis, we retrained all the RNNs that were trained in the previous sections to perform the DMS task without tuning the intrinsic parameters. For example, given a network trained on the non-WM integration task, we froze its intrinsic (R, τ_m, v_{rest}, v_{reset} ϑ) along with the synaptic decay time constant (τ) and optimized the recurrent connections (W) only using BPTT (see section 4). Therefore, each of the 20 RNNs trained for each of the four tasks (non-WM integration, delayed integration, DMS, and DIS tasks) was retrained to perform the DMS task. As expected, the average number of trials required to successfully retrain the RNNs previously trained for the DMS task was low at 4409 ± 3596 (see Figure 4A). The number of trials required to retrain the RNNs from the DIS task was also low at 4180 ± 2693. The RNNs trained for the delayed integration task took longer to retrain, at 5392 ± 2198. The non-WM RNNs required the most number of training trials to perform the DMS task (9648 ± 2933). These findings indicate that the intrinsic properties from one WM model are transferable to other WM models.

Based on these previous results, the membrane time constant (τ_m) and the synaptic decay (τ) variables appeared to be the two most important parameters for the transferability of WM. To test this, we repeated the retraining procedure with both τ_m and τ either fixed ("frozen") or optimized ("tuned") for the non-WM RNNs (see section 4). For the frozen condition (i.e., τ_m and τ frozen while the other parameters optimized), the number of trials required to retrain the non-WM RNNs to perform the DMS task was high and not significantly different from the number of trials it took with the intrinsic parameters fixed (see Figure 4B). On the other hand, retuning only τ_m and τ with the other parameters fixed (i.e., tuned condition) resulted in a significant reduction in training time (see Figure 4B), suggesting that these two parameters are indeed critical for performing WM. Optimizing both τ_m and τ resulted in a significant decrease in τ_m for both excitatory and inhibitory populations (see Figure 4C). The synaptic decay values decreased for the excitatory units after retuning (see Figure 4D left). For the inhibitory population, τ was significantly increased (see Figure 4D right).

2.4 Membrane and Synaptic Decay Time Constants Critical for WM Maintenance. Pyramidal excitatory neurons and parvalbumin (PV) interneurons make up the majority of the neuronal cell population in the cortex, and they have been shown to specialize in fast and reliable encoding of information with high temporal precision (Tremblay, Lee, & Rudy, 2016). To further investigate if the fast membrane and slow synaptic dynamics of the units from our WM RNNs are aligned with previous experimental findings and to probe how they contribute to WM maintenance, we manipulated τ_m and τ during different epochs of the DMS task paradigm.

For each of the RNNs trained from the DMS task, we first divided the population into two subgroups based on their τ_m values (see section 4). The short τ_m group contained units whose τ_m was smaller than the lower quartile value, while the long τ_m group contained units whose τ_m was greater than the upper quartile. During each of the four epochs (fixation, first stimulus, delay, and second stimulus), we then inhibited the two τ_m subgroups separately by hyperpolarizing them and assessed the task performance (see

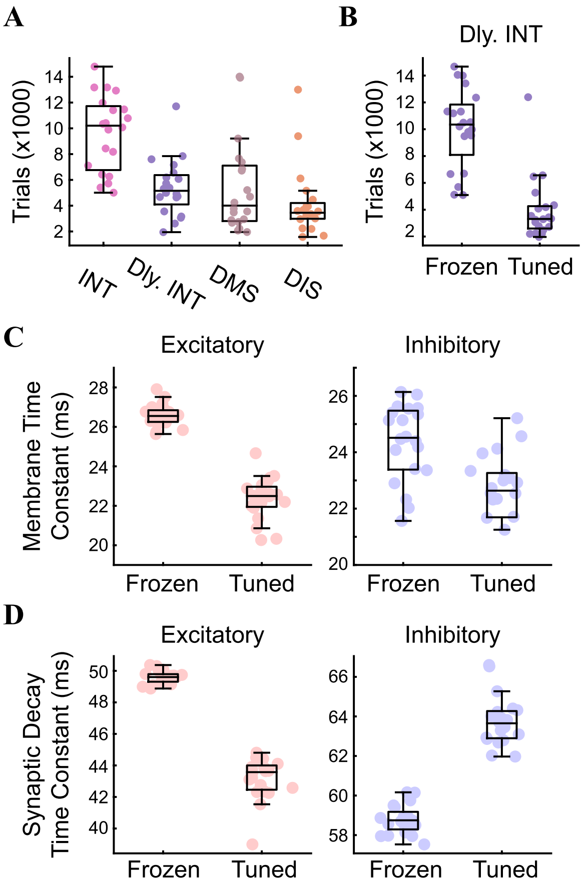

Figure 4: Retraining RNN models to perform the DMS task. (A) Number of training trials required to retrain the models previously trained for the INT, Dly. INT, DMS, or DIS tasks to perform the DMS task. (B) Number of training trials required to retrain the Dly. INT RNNs to perform the DMS task with the membrane time constant (τ_m) and synaptic decay time constant (τ) frozen or tuned. (C) Distribution of the membrane time constant values for the excitatory (red) and inhibitory (blue) units for the two conditions (frozen and tuned). Each dot represents the average value from one network. (D) Distribution of the synaptic decay time constant values for the excitatory (red) and inhibitory (blue) units for the two conditions (frozen and tuned). Each dot represents the average value from one network.

section 4). As shown in Figure 5, inhibiting the short τ_m subgroup during the two stimulus windows significantly impaired task performance (see Figures 5B and 5D), while disrupting the long τ_m group did not result in significant changes in task performance in all four task epochs.

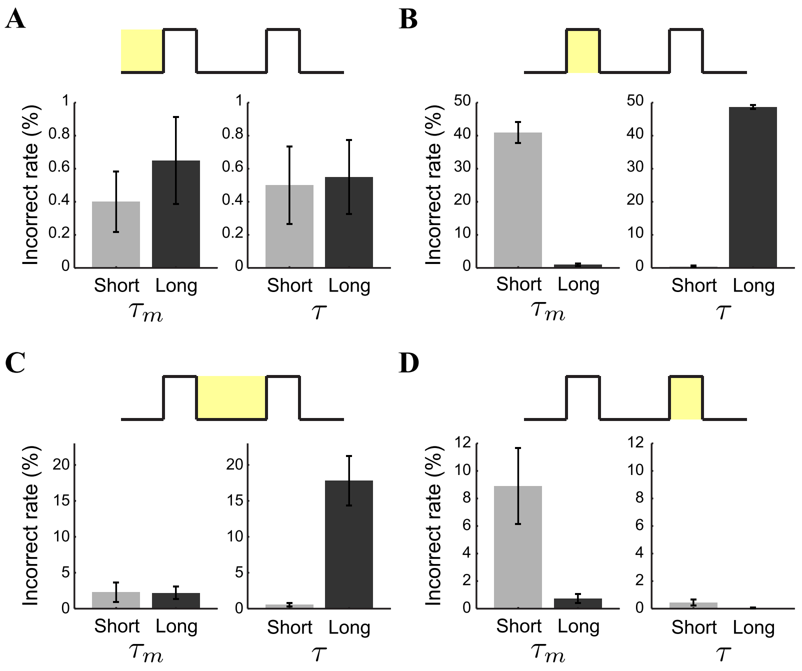

Figure 5: Membrane and synaptic time constants important for encoding stimuli and WM maintenance. (A–D) DMS task performance when short τ_m, long τ_m, short τ, or long τ units were inhibited during the fixation (A), first stimulus window (B), delay period (C), or second stimulus window (D).

We repeated the above analysis with two subgroups derived from a quartile split of the synaptic decay time constant (τ; see section 4). Suppressing the synaptic connections in the long τ subgroup during the first stimulus window and the delay period significantly impaired task performance (see Figures 5B and 5C). Inhibiting the short τ group at any of the four epochs did not affect the task performance.

Therefore, the units with the fast membrane voltage dynamics (τ_m) were important for encoding of stimuli, while the slow synaptic dynamics (τ) were critical for maintaining the first stimulus information throughout the period spanning from the first stimulus window to the end of the delay window.

3 Discussion

In this study, we presented a new method for directly training spiking RNNs with a gradient-based supervised training algorithm. Our approach allows optimizing not only the synaptic variables but also parameters intrinsic to spiking dynamics. By optimizing a wide range of parameters, we first demonstrated that units with diverse features emerged when the model was trained on a cognitive task (see Figures 1 and 2). We also showed that fast membrane dynamics combined with a slow synaptic property are critical for performing WM tasks (see Figures 4 and 5). Diversity is a basic biological principle that emerged here as a basic computational principle in spiking neural models.

Previous modeling studies have trained RNNs to perform cognitive tasks (Mante et al., 2013; Song, Yang, & Wang, 2016b; Miconi, 2017). Although some of these studies were able to train spiking RNN models, the intrinsic parameters of spiking neurons were not included as trainable variables. By using the mollifier approximation (Ermoliev et al., 1995), we developed a comprehensive framework that can tune both connectivity and spiking parameters using a gradient-descent method. Training spiking RNNs on multiple tasks using our method revealed functional specialization of excitatory and inhibitory neurons. More important, our approach allowed us to identify fast membrane voltage dynamics as an essential property required to encode incoming stimuli robustly for WM tasks.

Previous computational studies employing RNNs assumed that all the units in a network shared the same intrinsic parameters and optimized only synaptic connectivity patterns during training. Recent studies developed models that give rise to units with heterogeneous intrinsic properties. For example, a new activation function that is tunable for each neuron in a network has been proposed (Ramachandran, Zoph, & Le, 2017). In addition, we recently trained synaptic decay time constants in a rate RNN model (Kim et al., 2019). Although these methods produce heterogeneous units, they do not incorporate parameters inherent in spiking mechanisms. Our method not only allows direct training of synaptic weights of spiking RNNs that abide by Dale's principle but also enables training of synaptic and intrinsic membrane parameters for each neuron.

Although our method was successful at training spiking RNNs with biological constraints, the gradient-based method employed in the present study is not biologically plausible. In cortical neural networks, local learning rules, such as spike-timing-dependent plasticity (STDP), were observed, but the gradient-descent algorithm used in our method is not local to synapses or local in time (Tavanaei et al., 2018). However, this nonlocality allowed our method to train intrinsic membrane and connectivity parameters, creating biologically plausible neural architectures that solve specified problems. The learning algorithm for spiking neurons makes it possible to uncover neural dynamics hidden in experimental data (Mante et al., 2013; Song et al., 2016a; Remington, Narain, Hosseini, & Jazayeri, 2018), thus emphasizing that a biologically realistic model can be constructed by non-biological means.

Another limitation of our framework arises from our spiking neuron model. Although we were able to train models with heterogeneous neurons, the leaky integrate-and-fire model used in our study can only capture the dynamics of fast-firing neurons due to the lack of adaptation (Gerstner, Kistler, Naud, & Paninski, 2014). In particular, several other types of neurons, such as regular-firing and bursting neurons, are also common in cortical networks (Connors & Gutnick, 1990). Applying our method to spiking neuron models with adaptation dynamics, such as the model proposed by Bellec, Salaj, Subramoney, Legenstein, and Maass (2018), will be an interesting next step to further investigate the role of neurons from various firing classes in information processing.

In summary, we provide a novel approach for directly training both connectivity and membrane parameters in spiking RNNs. Training connectivity and intrinsic membrane parameters revealed distinct populations identifiable only by their parameter values, thus enabling investigation of the roles played by specific populations in the computation processes. This lays the foundation for uncovering how neural circuits process information with discrete spikes and building more power-efficient spiking networks.

4 Methods

4.1 Spiking Network Structure and Discretization.

Our spiking RNN model consisted of N integrate-and-fire (LIF) units is governed by

$$\tau_{m,i}\frac{dv_i}{dt} = -\left(v_i(t) - v_{\text{rest}_i}\right) + R_i I_i(t) + \xi, \tag{4.1}$$

where $\tau_{m,i}$ is the membrane time constant of unit i, $v_i(t)$ is the membrane voltage of unit i at time t, $v_{\text{rest},i}$ is the resting potential of unit i, and R_i is the input resistance of unit i, and ξ is the membrane voltage spontaneous fluctuation. $I_i(t)$ represents the current input to unit i at time t, which is given by

$$I_i(t) = \sum_{j=1}^{N} s_{ij}(t) + I_{\text{ext}_i}(t), \tag{4.2}$$

where N is the total number of units in the network, $s_{ij}(t)$ is the filtered spike train of unit j to unit i at time t, and $I_{\text{ext}_i}(t)$ is the external current source into unit i at time t. For this study, $N = 400$ for all tasks and networks trained.

The external current $\boldsymbol{I}_{\text{ext}}(t)$ encodes the task-specific input at time t,

$$\boldsymbol{I}_{\text{ext}}(t) = W_{\text{in}}\boldsymbol{u}(t), \tag{4.3}$$

where the time-varying stimulus signals $bmu(t) \in \mathbb{R}^{N_{\text{in}} \times 1}$ are fed into the network via $W_{\text{in}} \in \mathbb{R}^{N \times N_{\text{in}}}$, which can be viewed as presynaptic connections to the network that convert analog input into firing rates. N_{in} corresponds to the number of channels in the input signal.

We used a single exponential synaptic filter

$$\tau_{ij} \frac{ds_{ij}}{dt} = -s_{ij}(t) + \sum_{t_j^{(k)} < t} w_{ij} \delta(t - t_j^{(k)}), \tag{4.4}$$

where τ_{ij} is the synaptic decay time constant from unit j to unit i, w_{ij} is the synaptic strength from unit j to unit i, $t_j^{(k)}$ denotes the time of the kth action potential of unit j, and $\delta(x)$ is the Dirac delta function. Once the membrane voltage of the unit i crosses its action potential threshold (ϑ_i), its membrane voltage is brought back down to its reset voltage ($v_{\text{reset},i}$).

The output of our spiking model at time t is given by

$$o(t) = W_{\text{out}} \boldsymbol{r}(t), \tag{4.5}$$

where $W_{\text{out}} \in \mathbb{R}^{1 \times N}$ are the readout weights, and $\boldsymbol{r}(t) \in \mathbb{R}^{N \times 1}$, which can be interpreted as the firing rate of units, are given by

$$\tau_{r,i} \frac{dr_i}{dt} = -r_i(t) + \sum_{t_i^{(k)} < t} \delta(t - t_i^{(k)}), \tag{4.6}$$

where $\tau_{r,i}$ is the synaptic decay time constant of firing rate estimate for unit i.

We converted the continuous-time differential equations to discrete-time iterative equations and used numerical integration (Euler's method) to solve the equations. The membrane voltage $\boldsymbol{v} \in \mathbb{R}^{1 \times N}$ at step $n+1$ is given by

$$\boldsymbol{v}^{(n+1)} = \tilde{\boldsymbol{v}}^{(n)} + \frac{\Delta t}{\boldsymbol{\tau}_m} \left(- \left(\tilde{\boldsymbol{v}}^{(n)} - \boldsymbol{v}_{\text{rest}} \right) + \boldsymbol{I}^{(n+1)} \odot \boldsymbol{R} \right) + c\mathcal{N}(0, \Delta t), \tag{4.7}$$

where Δt is the sampling rate (or step size), which was set $\Delta t = 1$ ms for this study, $\boldsymbol{\tau}_m \in \mathbb{R}^{1 \times N}$ is the membrane time constant, $\boldsymbol{v}_{\text{rest}} \in \mathbb{R}^{1 \times N}$ is the resting potential, $\odot$ refers to Hadamard operation (element-wise multiplication), $\div$ refers to the element-wise division, and $\boldsymbol{R} \in \mathbb{R}^{1 \times N}$ is the input resistance. The term $c\mathcal{N}(0, \Delta t)$ injects spontaneous membrane fluctuations, where $\mathcal{N}(0, \Delta t) \in \mathbb{R}^{1 \times N}$ is a gaussian random vector consisting of N independent gaussian random variables with mean 0 and variance Δt, and c is the scaling constant for the amplitude of fluctuations, set as $c = 5$ throughout the study.

There are two time-varying terms in equation 2.7, the membrane voltage after reset ($\tilde{\boldsymbol{v}}^{(n)}$) and input current ($\boldsymbol{I}^{(n+1)}$). The voltage reset in the LIF model after action potentials at step n is formulated as

$$\tilde{\boldsymbol{v}}^{(n+1)} = \boldsymbol{v}^{(n+1)} + \left(\boldsymbol{v}_{\text{reset}} - \boldsymbol{v}^{(n+1)}\right) \odot H\left(\boldsymbol{v}^{(n+1)} - \boldsymbol{\vartheta}\right), \tag{4.8}$$

where $\boldsymbol{v}_{\text{reset}} \in \mathbb{R}^{1\times N}$ is the reset potential, $\boldsymbol{\vartheta} \in \mathbb{R}^{1\times N}$ is the action potential thresholds, and $H(x)$ is the element-wise Heaviside step function. The term $H\left(\boldsymbol{v}^{(n+1)} - \boldsymbol{\vartheta}\right)$ represents the spiking output activities at step $n+1$. The input current at step $n+1$ is given by

$$\boldsymbol{I}^{(n+1)} = S^{(n)} \cdot \mathbf{1} + W_{\text{in}}\boldsymbol{u}^{(n+1)}, \tag{4.9}$$

where $\mathbf{1} \in \mathbb{R}^{1\times N}$ is the column vector with all ones and $S^{(n)}$ is the filtered spike train matrix at step n, which follows the iteration

$$S^{(n)} = S^{(n-1)} + \frac{\Delta t}{T}\left(-S^{(n-1)} + W \odot H\left(\boldsymbol{v}^{(n)} - \boldsymbol{\vartheta}\right)\right), \tag{4.10}$$

where $T \in \mathbb{R}^{N\times N}$ is the matrix of synaptic decay time constants and $W \in \mathbb{R}^{N\times N}$ is the matrix of synaptic strengths. Here, $W \in \mathbb{R}^{N\times N}$ is a matrix, and $H\left(\boldsymbol{v}^{(n)} - \boldsymbol{\vartheta}\right) \in \mathbb{R}^{1\times N}$ is a row vector. The notation $A \odot \boldsymbol{v}$ refers to element-wise multiplication of matrix A row by row with the row vector $\boldsymbol{v}$.

The output at step $n+1$ is computed by

$$o^{(n+1)} = W_{\text{out}}\boldsymbol{r}^{(n+1)}, \tag{4.11}$$

in which

$$\boldsymbol{r}^{(n+1)} = \boldsymbol{r}^{(n)} + \frac{\Delta t}{\boldsymbol{\tau}_r}\left(-\boldsymbol{r}^{(n)} + H\left(\boldsymbol{v}^{(n+1)} - \boldsymbol{\vartheta}\right)\right), \tag{4.12}$$

where $\boldsymbol{\tau}_r \in \mathbb{R}^{1\times N}$ is the synaptic decay time constants of firing rate estimate.

4.2 Training Details. In this study, we used only the supervised back-propagation of errors learning algorithm. The loss function ($\mathcal{L}$) is defined in terms of the root mean square error (RMSE) with respect to a task-specific target signal ($\boldsymbol{z}$) and the network output signal ($\boldsymbol{o}$),

$$\mathcal{L} := \sqrt{\left(\sum_{n=1}^{M}\left(z^{(n)} - o^{(n)}\right)^2\right)}. \tag{4.13}$$

where M is the total time steps.

We used adaptive moment estimation (ADAM) stochastic gradient descent algorithm (Kingma & Ba, 2014) with mini-batch training. The mollifier gradient approximations were employed to address non-differentiability problem associated with the spiking process (see section 4.3). The learning rate was set to 0.01, the batch size was set to 10, and the first and second moment decay rates were 0.9 and 0.999, respectively. The trainable parameters include input weights (W_{in}), synaptic strengths (W), readout weights (W_{out}), synaptic decay time constants (T), membrane time constants ($\boldsymbol{\tau}_m$), input resistances ($\boldsymbol{R}$), resting potentials ($\boldsymbol{v}_{\text{rest}}$), reset voltages ($\boldsymbol{v}_{\text{reset}}$), action potential thresholds ($\boldsymbol{\vartheta}$), and synaptic decay time constants for firing rate estimates ($\boldsymbol{\tau}_r$).

A nonlinear projected gradient method was used to constrain parameters within the biologically realistic ranges described in Table 1. A linear projection map forces some solutions to be projected on the boundary. That is, there are always some units whose parameters take the min and max values of the constraint. On the other hand, a nonlinear projection guarantees that no values are on the boundary almost surely, a more realistic situation to consider. Specifically, to bound a parameter p at iteration $i+1$ into the range $[p_{\min}, p_{\max}]$, we have

$$\tilde{p}_{i+1} = \sigma(p_{i+1}) \cdot (p_{\max} - p_{\min}) + p_{\min}, \tag{4.14}$$

where $\tilde{p}_{i+1}$ is the projected solution of parameter p at iteration $i+1$, p_{i+1} is the unconstrained solution given by the gradient descent algorithm at iteration $i+1$, $p_{\max}$ and $p_{\min}$ are the maximum and minimum values of parameter p, and $\sigma(x)$ is the sigmoid function, defined as

$$\sigma(x) := \frac{1}{1+\exp(-x)}. \tag{4.15}$$

We initialized all parameters, except the input weights (W_{in}), as samples from the standard gaussian distribution with zero mean and unit variance, whereas the input weights were drawn from gaussian distribution with zero mean and variance 400. This is because our input signals were bounded within the range $[-1, 1]$, insufficient to bring the membrane voltage from the resting potential above the action potential threshold. Hence, to accelerate training, it was necessary to make sure units were excited by the input signals in the first place. The synaptic strength matrix (W) was also initialized sparse, with the percentage of connectivity being only 20%. We say the network successfully did the task if the output signal hits above $+0.8$ (or below -0.8) if the target output is $+1$ (or -1). We stopped training when the loss ($\mathcal{L}$) is less than 15 and the accuracy over 100 trials is above 95%.

The method proposed by Song et al. (2016a) was used to impose Dale's principle with separate excitatory and inhibitory populations. The synaptic connectivity matrix (W) in the model was parameterized by

$$\tilde{W}_{i+1} = [W_{i+1}]_+ \cdot D, \tag{4.16}$$

where $\tilde{W}_{i+1}$ is the resulting matrix that encoded separate populations at update step $i+1$, W_{i+1} is the solution given by the gradient descent algorithm at step $i+1$, and $[\cdot]_+$ is the rectified linear unit (ReLU) operation applied at the end of each update step. The ReLU operation is to ensure that entries of the matrix are always nonnegative before being multiplied by the matrix D, as the negative weight connections updated from gradient descent are pruned by the end of each update. The diagonal matrix ($D \in \mathbb{R}^{N \times N}$) encodes $+1$ for excitatory units and -1 for inhibitory units. The value of matrix (D) was randomly assigned before training according to a preset proportion between inhibitory and excitatory units, and the value D was fixed through the whole training process. The I/E units proportion in this study was 20% to 80%.

In order to capture the biologically realistic dynamics of SNNs, the temporal resolution (Δt) was set to be no longer than the duration of absolute refractory period to ensure that the spiking activities are not affected by the numerical integration process. Therefore, we set $\Delta t = 1$ ms during training. Due to the vanishing gradient problem occurring in training RNNs (Hochreiter, Bengio, Frasconi, & Schmidhuber, 2001), with $\Delta = 1$ ms, it is impossible to train tasks with duration longer than 1 second (i.e., $M > 1000$). It is notable that in the above formulation, only membrane time constant (τ_m) and synaptic time decay (τ) are dependent on the sampling rate (Δt; see equations 4.7 and 4.10). Hence, after the models are trained, we can make sampling rate (Δt) smaller (i.e., having finer temporal resolution) while still keeping the same dynamics of the trained networks. Increasing Δt by a factor is equivalent to decreasing τ and τ_m altogether by the same factor, as τ and τ_m are inversely proportional to Δt in equations 4.7 and 4.10. Hence, to train a network performing tasks with duration longer than 1 second, we need to make the temporal resolution coarser (i.e., increasing Δt by a factor s) so that with the same trainable range of time steps (i.e., a fixed $M \leq 1000$), the duration of task becomes longer by the same factor s. This "decrease in temporal resolution" can be interpreted as shortening τ and τ_m instead of an actual decrease in temporal resolution. Applying this trick enables us to train tasks with arbitrary duration by rescaling the ranges of τ and τ_m into a smaller one while still making the spiking activities biologically realistic. In practice, we simply scaled down τ and τ_m by a factor $s = 3$ with a fixed number of time steps (M), and later during the testing stage, we rescaled M, τ and τ_m up by the same factor s.

4.3 Mollifier Gradient Approximations. In the above formulation, the Heaviside step function $H(x)$ is not continuous. As a result, the loss function $\mathcal{L}$ is not differentiable. This poses a major problem when applying the traditional backpropagation algorithm for training neural networks because the backpropagation algorithm uses gradient descent methods that require the function being minimized to be differentiable, or at least to be continuous. However, the derivative of Heaviside step function $H(x)$ is Dirac delta function $\delta(x)$, which is 0 everywhere except at 0, where the function value is ∞. It is difficult to use this derivative for the gradient descent methods because the value of the gradients is 0 almost everywhere.

To address the discontinuity problem, we employed mollifier gradient method proposed by Ermoliev et al. (1995). The method can be applied to any strongly lower semicontinuous functions to find local minima following an iterative gradient descent in which the gradients change over iterations based on averaged functions derived from the original objective function. The family of averaged functions f_ε of function f is defined by convolution of f with a mollifier: ψ_ε

$$f_\varepsilon(x) := \int_{\mathbb{R}^n} f(x-z)\psi_\varepsilon(z)dz = \int_{\mathbb{R}^n} f(x)\psi_\varepsilon(x-z)dz = f * \psi_\varepsilon(x), \tag{4.17}$$

where $\psi_\varepsilon \in \{\psi_\varepsilon : \mathbb{R}^n \to \mathbb{R}_+, \varepsilon > 0\}$, a family of compactly supported (generalized) functions named mollifiers that satisfy

$$\int_{\mathbb{R}^n} \psi_\varepsilon(x)\,dx = 1, \quad \lim_{\varepsilon\to 0} \psi_\varepsilon(x) = \lim_{\varepsilon\to 0} \varepsilon^{-n}\psi_\varepsilon(x/\varepsilon) = \delta(x). \tag{4.18}$$

It was shown that for any strongly lower semicontinuous functions f, the averaged functions f_ε epi-converge to f as $\varepsilon \to 0$, a type of convergence that preserves the local minima and minimizers. Therefore, it is possible to use the gradients of averaged functions to minimize the original lower semicontinuous functions and find the local minima. We used the conventional family of mollifiers obtained by normalizing a probability density function ψ:

$$\psi_\varepsilon(z) := \frac{\psi(z/\varepsilon)}{\varepsilon^n}. \tag{4.19}$$

In our case, $n = 1$ as the domain of $H(x)$ is the real line:

$$H_\varepsilon(x) := \frac{1}{\varepsilon}\int_{-\infty}^{\infty} H\,(x-z)\,\psi(z/\varepsilon)\,dz. \tag{4.20}$$

For any $\varepsilon > 0$, the gradient of $H_\varepsilon\,(g(x))$ with respect to parameter p is given by

$$\nabla_p H_\varepsilon\left(g(x)\right) = \frac{1}{\varepsilon}\psi\left(g(x)/\varepsilon\right)\nabla_p g(x) = \psi_\varepsilon\left(g(x)\right)\nabla_p g(x), \tag{4.21}$$

where ψ is some symmetric density function and $g(x)$ is any function with $\mathbb{R}$ as its codomain. Since our goal was not to find a local minimum x^* that satisfies the optimality condition $\lim_{\varepsilon\to 0}\|\nabla f_\varepsilon(x^*)\| = 0$ as defined by Ermoliev et al. (1995), but rather to minimize the loss function for its value to be sufficiently small so that the network can perform the task correctly, we did not vary the gradients during the minimization process. Instead, we fixed an approximation of the gradient and used the approximation throughout the training process. We chose the normalized box function, that is, the density function of uniform distribution $\mathcal{U}(-\varepsilon/2, \varepsilon/2)$, as the kernel,

$$\psi(x) := \begin{cases} \frac{1}{\varepsilon} & \text{for } x \in [-\varepsilon/2, \varepsilon/2] \\ 0 & \text{otherwise} \end{cases} \tag{4.22}$$

and fixed $\varepsilon = 5$.

We found no difference in the trained models with different choices of ε as long as the value was large enough to keep the gradients active so that they did not vanish through time steps. There was also no difference between models trained with fixed ε and those trained with the original scheme in Ermoliev et al. (1995) where $\varepsilon \to 0$ as the number of iterations increases. The purpose for fixing the value of ε was to compare the training epochs (iterations) among the retraining paradigms (see Figure 3) with the same gradient.

4.4 Retraining Models for DMS Task. To test whether intrinsic properties optimized for one WM task are generalizable to other tasks that also require WM, we retrained our models to perform the DMS task with all intrinsic properties fixed. In contrast to the training paradigm described in the previous sections, the trainable parameters for retraining only include input weights (W_{in}), synaptic strengths (W), and readout weights (W_{out}). Each of the 20 RNNs trained for each of the four tasks (non-WM integration, delayed integration, DMS, and DIS tasks) used in this study was retrained to perform the DMS task.

To test whether synaptic decay time constants (τ) and membrane time constants (τ_m) are the most crucial parameters for transferability of WM tasks, we repeated the retraining procedure with both τ_m and τ either fixed or optimized for the non-WM RNNs. The RNNs optimized to perform the context-based input integration task were used for retraining under two schemes: the tuned scheme and the frozen scheme. For the tuned scheme, the trainable parameters include input weights (W_{in}), synaptic strengths (W), readout weights (W_{out}), synaptic decay time constants (T), membrane time constants ($\boldsymbol{\tau}_m$), and synaptic decay time constants for firing rate estimates ($\boldsymbol{\tau}_r$). For the frozen scheme, the trainable parameters include input weights (W_{in}), synaptic strengths (W), readout weights (W_{out}), input resistances ($\boldsymbol{R}$), resting potentials ($\boldsymbol{v}_{\text{rest}}$), reset voltages ($\boldsymbol{v}_{\text{reset}}$), and action potential thresholds ($\boldsymbol{\vartheta}$).

4.5 Units Function Analysis. For Figure 4, we manipulated τ_m and τ during different epochs of the DMS task paradigm to investigate if fast membrane and slow synaptic dynamics are responsible for WM maintenance. For each of the RNNs trained from the DMS task, we first divided the population into two subgroups based on their τ_m values. The short τ_m group contained units whose τ_m was smaller than the median value of τ_m of all units in the RNN, while the long τ_m group contained units whose τ_m was greater than the median value. The average median value of τ_m across all 20 models was 19.64 ± 2.45 ms. During each of the four epochs (fixation, first stimulus, delay, and second stimulus), we inhibited the two τ_m subgroups separately by hyperpolarizing them and then assessed the task performance. The hyperpolarization was done by setting the membrane voltage $v = -100$ mV for the intended subgroup of units. Similar to the training stage, we say that the network successfully did the task if the output signal hits above $+0.8$ (or below -0.8) if the target output is $+1$ (or -1). If the target output is between -0.8 and $+0.8$, the network is considered having no response. If the output signal is above $+0.8$ (or below -0.8) while the target output is -1 (or $+1$), we say that the network gives an incorrect response.

We conducted a similar analysis based on two subgroups of synapses derived from a quartile split of synaptic decay time constant (τ). The short τ group contained synapses whose τ was smaller than the 25th percentile of all τ in the RNN, while the long τ group contained synapses whose τ was greater than the 75th percentile. The average 25th percentile across all 20 models was 25.36 ± 2.40 ms, and the average 75th percentile was 66.18 ± 1.17 ms. The targeted subgroup of synapses was suppressed by setting the connection strength $w = 0$ during each of the four epochs of DMS task.

Code Availability

The implementation of our framework and the codes to generate all the figures in this work are available at https://github.com/y-inghao-li/SRNN/.

Data Availability

The trained models used in this study are available as Matlab-formatted data at https://github.com/y-inghao-li/SRNN/.

Acknowledgments

We are grateful to Jorge Aldana for assistance with computing resources. This research was funded by DARPA (W911NF1820259 to T.J.S.), the Office of Naval Research (N00014-16-1-2829 to T.J.S.), and the National Institute of Mental Health (F30MH115605-01A1 to R.K.). We also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Quadro P6000 GPU used for this research. The funders had no role in study design, data collection and analysis, decision to publish, or manuscript preparation.

References

Bellec, G., Salaj, D., Subramoney, A., Legenstein, R., & Maass, W. (2018). Lon short-term memory and learning-to-learn in networks of spiking neurons. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, K., N. Cesa-Bianchi, N., & R. Garnett (Eds.), *Advances in neural information processing systems*, *21*. Red Hook, NY: Curran

Chelaru, M. I., & Dragoi, V. (2008). Efficient coding in heterogeneous neuronal populations. In *Proceedings of the National Academy of Sciences*, *105*(42), 16344–16349. 10.1073/pnas.0807744105

Chen, L., & Aihara, K. (1995). Chaotic simulated annealing by a neural network model with transient chaos. *Neural Networks*, *8*(6), 915–930. 10.1016/0893-6080(95)00033-V

Connors, B. W., & Gutnick, M. J. (1990). Intrinsic firing patterns of diverse neocortical neurons. *Trends in Neurosciences*, *13*(3), 99–104. 10.1016/0166-2236(90)90185-D, PubMed: 1691879

Douglas, R. J., & Martin, K. A. (2007). Recurrent neuronal circuits in the neocortex. *Current Biology*, *17*(13), R496–R500. 10.1016/j.cub.2007.04.024, PubMed: 17610826

Ermoliev, Y. M., Norkin, V. I., & Wets, R. J. (1995). The minimization of semicontinuous functions: Mollifier subgradients. *SIAM Journal on Control and Optimization*, *33*(1), 149–167. 10.1137/S0363012992238369

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

Goldman-Rakic, P. S. (1995). Cellular basis of working memory. *Neuron*, *14*(3), 477–485. 10.1016/0896-6273(95)90304-6, PubMed: 7695894

Hochreiter, A., Bengio, Y, Frasconi, P., & Schmidhuber, J. (2001). *Gradient flow in recurrent nets: The difficulty of learning long term dependencies*. In J. F. Kolen & S. C. Kremer (Eds.), *A field guide to dynamical recurrent neural networks* (pp. 237–243). Piscataway, NJ: IEEE Press.

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* (pp. 1433–1443). Red Hook, NY: Curran.

Kim, C. M., & Chow, C. C. (2018). Learning recurrent dynamics in spiking networks. *eLife*, *7*.

Kim, R., Li, Y., & Sejnowski, T. J. (2019). Simple framework for constructing functional spiking recurrent neural networks. In *Proceedings of the National Academy of Sciences*, *116*(45), 22811–22820. 10.1073/pnas.1905926116

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

Lee, J. H., Delbruck, T., & Pfeiffer, M. (2016). Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*, *10*, 508.

Mante, V., Sussillo, D., Shenoy, K. V., & Newsome, W. T. (2013). Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, *503*(7474), 78. 10.1038/nature12742, PubMed: 24201281

Mastrogiuseppe, F., & Ostojic, S. (2018). Linking connectivity, dynamics, and computations in low-rank recurrent neural networks. *Neuron*, *99*(3), 609–623. 10.1016/j.neuron.2018.07.003, PubMed: 30057201

Medalla, M., Gilman, J. P., Wang, J.-Y., & Luebke, J. I. (2017). Strength and diversity of inhibitory signaling differentiates primate anterior cingulate from lateral prefrontal cortex. *Journal of Neuroscience*, *37*(18), 4717–4734. 10.1523/JNEUROSCI.3757-16.2017, PubMed: 28381592

Miconi, T. (2017). Biologically plausible learning in recurrent neural networks reproduces neural dynamics observed during cognitive tasks. *eLife*, *6*, e20899. 10.7554/eLife.20899, PubMed: 28230528

Neftci, E. O., Mostafa, H., & Zenke, F. (2019). Surrogate gradient learning in spiking neural networks. *CoRR*, abs/1901.09948.

Nicola, W., & Clopath, C. (2017). Supervised learning in spiking neural networks with force training. *Nature Communications*, *8*(1), 2208. 10.1038/s41467-017-01827-3, PubMed: 29263361

Pala, A., & Petersen, C. C. (2018). State-dependent cell-type-specific membrane potential dynamics and unitary synaptic inputs in awake mice. *eLife*, *7*, e35869. 10.7554/eLife.35869, PubMed: 30052198

Peyrache, A., Dehghani, N., Eskandar, E. N., Madsen, J. R., Anderson, W. S., Donoghue, J. A., . . . Destexhe, A. (2012). Spatiotemporal dynamics of neocortical excitation and inhibition during human sleep. In *Proceedings of the National Academy of Sciences*, *109*(5), 1731–1736. 10.1073/pnas.1109895109

Rajan, K., Abbott, L., & Sompolinsky, H. (2010). Stimulus-dependent suppression of chaos in recurrent neural networks. *Physical Review E*, *82*(1), 011903. 10.1103/PhysRevE.82.011903

Rajan, K., Harvey, C. D., & Tank, D. W. (2016). Recurrent network models of sequence generation and memory. *Neuron*, *90*(1), 128–142. 10.1016/j.neuron.2016.02.009, PubMed: 26971945

Ramachandran, P., Zoph, B., & Le, Q. V. (2017). *Searching for activation functions.* arXiv:1710.05941.

Remington, E. D., Narain, D., Hosseini, E. A., & Jazayeri, M. (2018). Flexible sensorimotor computations through rapid reconfiguration of cortical dynamics. *Neuron*, *98*(5), 1005–1019. 10.1016/j.neuron.2018.05.020, PubMed: 29879384

Romo, R., Brody, C. D., Hernández, A., & Lemus, L. (1999). Neuronal correlates of parametric working memory in the prefrontal cortex. *Nature*, *399*(6735), 470–473. 10.1038/20939, PubMed: 10365959

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1988). Learning representations by back-propagating errors. *Cognitive Modeling*, *5*(3), 1.

Sengupta, A., Ye, Y., Wang, R., Liu, C., & Roy, K. (2019). Going deeper in spiking neural networks: VGG and residual architectures. *Frontiers in Neuroscience*, *13*. 10.3389/fnins.2019.00095, PubMed: 30899212

Sippy, T., Lapray, D., Crochet, S., & Petersen, C. C. (2015). Cell-type-specific sensorimotor processing in striatal projection neurons during goal-directed behavior. *Neuron*, *88*(2), 298–305. 10.1016/j.neuron.2015.08.039, PubMed: 26439527

Sompolinsky, H., Crisanti, A., & Sommers, H.-J. (1988). Chaos in random neural networks. *Physical Review Letters*, *61*(3), 259. 10.1103/PhysRevLett.61.259, PubMed: 10039285

Song, H. F., Yang, G. R., & Wang, X.-J. (2016a). Training excitatory-inhibitory recurrent neural networks for cognitive tasks: A simple and flexible framework. *PLOS Computational Biology*, *12*(2), e1004792.

Song, H. F., Yang, G. R., & Wang, X.-J. (2016b). Training excitatory-inhibitory recurrent neural networks for cognitive tasks: A simple and flexible framework. *PLOS Computational Biology*, *12*(2), e1004792.

Sussillo, D., & Abbott, L. F. (2009). Generating coherent patterns of activity from chaotic neural networks. *Neuron*, *63*(4), 544–557. 10.1016/j.neuron.2009.07.018, PubMed: 19709635

Tavanaei, A., Ghodrati, M., Kheradpisheh, S. R., Masquelier, T., & Maida, A. (2018). Deep learning in spiking neural networks. *Neural Networks*, *111*, 47–63. 10.1016/j.neunet.2018.12.002, PubMed: 30682710

Thalmeier, D., Uhlmann, M., Kappen, H. J., & Memmesheimer, R.-M. (2016). Learning universal computations with spikes. *PLOS Computational Biology*, *12*(6), e1004895. 10.1371/journal.pcbi.1004895, PubMed: 27309381

Tremblay, R., Lee, S., & Rudy, B. (2016). GABAergic interneurons in the neocortex: From cellular properties to circuits. *Neuron*, *91*(2), 260–292. 10.1016/j.neuron.2016.06.033, PubMed: 27477017

VanRullen, R., Guyonneau, R., & Thorpe, S. J. (2005). Spike times make sense. *Trends in Neurosciences*, *28*(1), 1–4. 10.1016/j.tins.2004.10.010, PubMed: 15626490

Wang, X.-J. (2008). Decision making in recurrent neuronal circuits. *Neuron*, *60*(2), 215–234. 10.1016/j.neuron.2008.09.034, PubMed: 18957215

Werbos, P. J. (1990). Backpropagation through time: What it does and how to do it. In *Proceedings of the IEEE*, *78*(10), 1550–1560. 10.1109/5.58337

Zhang, W., & Li, P. (2019). Spike-train level backpropagation for training deep recurrent spiking neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E, Fox, & R. Garnett (Eds.), *Advances in neural information processing systems, 32* (pp. 7800–7811). Red Hook, NY: Curran.

Received November 3, 2020; accepted March 15, 2021.