Biologically Plausible Sequence Learning with Spiking Neural Networks

Zuozhu Liu,1,3 Thiparat Chotibut,2,3,* Christopher Hillar,4,5 Shaowei Lin3

1Department of Statistics and Applied Probability, National University of Singapore,
2Department of Physics, Faculty of Science, Chulalongkorn University, Thailand,
3Engineering Systems and Design, Singapore University of Technology and Design, 4Awecom, Inc,
5Redwood Center for Theoretical Neuroscience, University of California, Berkeley,
{lcowen.hn, *thiparatc, hillarmath}@gmail.com, shaowei_lin@sutd.edu.sg

Abstract
Motivated by the celebrated discrete-time model of nervous activity outlined by McCulloch and Pitts in 1943, we propose a novel continuous-time model, the McCulloch-Pitts network (MPN), for sequence learning in spiking neural networks. Our model has a local learning rule, such that the synaptic weight updates depend only on the information directly accessible by the synapse. By exploiting asymmetry in the connections between binary neurons, we show that MPN can be trained to robustly memorize multiple spatiotemporal patterns of binary vectors, generalizing the ability of the symmetric Hopfield network to memorize static spatial patterns. In addition, we demonstrate that the model can efficiently learn sequences of binary pictures as well as generative models for experimental neural spike-train data. Our learning rule is consistent with spike-timing-dependent plasticity (STDP), thus providing a theoretical ground for the systematic design of biologically inspired networks with large and robust long-range sequence storage capacity.

Introduction
Experimental evidence from neurobiology reveals that changes in synaptic weights of some neurons depend on the timing difference between presynaptic and postsynaptic spikes (Markram and Sakmann 1995; Gerstner et al. 1996; Bi and Poo 2001), a concept termed spike-timing-dependent plasticity (STDP). While deep learning is tremendously successful in numerous machine learning tasks, its underlying backpropagation learning algorithm is biologically implausible (Bengio et al. 2015). Recent works attempt to bridge this gap by proposing STDP-consistent training rules for deep learning (Bengio et al. 2015; 2017; Liu, Quek, and Lin 2018; Scellier and Bengio 2017), with the hope of uncovering computationally efficient learning algorithms inspired by biological brains. Thorough understanding of STDP-consistent learning rules and their computing capabilities can also benefit the design of modern bioinspired computing hardware such as neuromorphic chips.

On the other hand, machine learning can assist in the quest for understanding how brains perform computation, especially when viewed through the lens of a spiking neural network (SNN). For example, variational inference and reinforcement learning unveil SNN architectures that perform probabilistic inference consistent with STDP rules (Nessler et al. 2013; Pecevski and Maass 2016; Rezende, Wierstra, and Gerstner 2011). Learning-to-learn methods and Long Short-Term Memory help identify a novel SNN architecture with the capability to store long spatiotemporal sequences of spikes during computation (Bellec et al. 2018). Moreover, building generative models of SNNs from experimental time-series data of action potentials yields insights into the network structure generating such spikes (Tyrcha et al. 2013; Nasser, Marre, and Cessac 2013; Zeng et al. 2013).

With the goal of building a biologically plausible generative model of SNNs from spatiotemporal patterns, such as neural spike-train data, we propose a novel continuous-time learning algorithm that is consistent with experimentally observed STDP. We define simple rules for updating states and learning weights of a continuous-time SNN demonstrate that the expected synaptic changes given a pair of presynaptic and postsynaptic spikes reproduce the STDP curve. As a consequence of our model, we show that the stochasticity of the refractory period is an important ingredient for reproducing STDP. Namely, by assuming that the time between the spiking of a neuron and its recovery is stochastic and follows an exponential distribution, we prove that the expected increase or decrease in synaptic weights follows STDP-like curves.

In addition to its biological plausibility, our SNN model is capable of learning sequences of binary vectors, or spatiotemporal patterns of spikes. In the deterministic limit, this network, a McCulloch-Pitts network (McCulloch and Pitts 1943), is a generalization of the Hopfield network that allows asymmetric connections between fully-visible binary units. As a result, deterministic dynamics in the state space can converge to an attractor of period larger than one (a cycle), as opposed to a fixed-point corresponding to a static spatial memory in the Hopfield network. We demonstrate that the network can be trained to memorize repeating sequences of binary vectors and that the spatiotemporal memory is robust to perturbation. In particular, given an arbitrary initial state of the spike pattern, the dynamics generated by the trained network converge to a memorized cycle. Thus, the...
STDP-consistent McCulloch-Pitts network performs a spatiotemporal pattern completion, generalizing the classical Hopfield network that performs spatial pattern completion. In the noisy limit, the trained network can be regarded as a generative model for spatiotemporal patterns of spikes that performs sampling of both the time to the next spiking event and the neuron to next spike. We illustrate that this network efficiently performs inference on continuous-time data from a neurobiology experiment, and we discuss its applicability to inference problems in neuroscience.

McCulloch-Pitts Networks
To begin, suppose we are presented with a time series $D(t)$ of binary vectors with single bit-flip transitions:

$$D(t) = x(0) \quad \text{for } 0 = t_0 \leq t < t_1,$$

$$D(t) = x(n-1) \quad \text{for } t_{N-1} \leq t < t_N,$$

$$D(t) = x(N) \quad \text{for } t = t_N.$$

Our goal is to train a continuous-time stochastic process that mimics $D(t)$.

In this section, we define our model of a neural stochastic process, which we term a McCulloch-Pitts network. Relationships to biological neural networks will be discussed in the Spike-Timing-Dependent Plasticity section. Let $G = (V, E)$ be a weighted directed graph with vertices $V = \{1, \ldots, d\}$, edges $ij \in E$ from $i$ to $j$, vertex biases $b_i$, and weights $w_{ij} \in \mathbb{R}$ for each edge $ij \in E$. Let $\theta$ denote the vector of parameters. Self-loops and directed cycles are allowed in $G$.

Dynamics
Let $\mathcal{X} = \{0, 1\}^d$ be the state space of binary vectors on $V$, and $X(t)$ be a stochastic process indexed by continuous time $t \geq 0$ with states $\mathcal{X}$. Let $\tau > 0$ be the temperature, and define $\text{flip}_j(x)$ to be the binary vector $x$ whose $i$-th coordinate is flipped. To generate $X(t)$, we select an initial condition $X(0)$ from $\mathcal{X}$, and repeat the following two steps:

1. Suppose $X(T) = x$. For each $i \in V$, we sample a holding time $\xi_i$ from the exponential distribution $\text{Exp}(\lambda_i)$ where $\lambda_i = \text{exp}(\sigma_i z_i / \tau)$, $\sigma_i = 1 - 2x_i$, and $z_i = \sum_{j \in E} w_{ij} x_j + b_i$.

2. Let $\xi_j$ be the smallest of the holding times $\xi_i$, and set $T' = T + \xi_j$. We set $X(t) = x$ for all $T \leq t < T'$, and $X(T') = \text{flip}_j(x)$. Finally, we update $T \leftarrow T'$.

More generally, a rate scaling parameter $r_i$, if introduced in the rate $\lambda_i = r_i \text{exp}(\sigma_i z_i / \tau)$. However, $r_i$ can be absorbed into $b_i$ and $w_{ij}$, see Supplementary Materials

\begin{align*}
\text{(SM) A. Thus, we can set } r_i = 1 \text{ for simplicity and assume the graph } G \text{ has self-loops. We term } X(t) \text{ the McCulloch-Pitts network (MPN) associated to } (G, \theta, \tau). \text{ Observe that the transitions in } X(t) \text{ involve flipping only one bit at a time.}

\text{Alternatively, we can reformulate an equivalent yet simpler algorithm to generate } X(t) \text{ using softmax; given an initial condition } X(0) \in \mathcal{X}, \text{ we repeat the following steps:}

1. Let $X(T) = x$. Sample $\xi \sim \text{Exp}(\lambda)$ where $\lambda = \sum_{i \in V} \lambda_i$.

2. Let $T' = T + \xi$ and set $X(t) = x$ for all $T \leq t < T'$. Pick $j \in V$ according to $P_j = \lambda_j / \lambda$, and set $X(T') = \text{flip}_j(x)$. Finally, we update $T \leftarrow T'$.

The multinomial distribution $P_j = \lambda_j / \lambda$ in step 2 is the softmax, since $P_j \propto \exp(\sigma_j z_i / \tau)$.

Training
For brevity, let us define the following data statistics:

$$\delta_i^{(n)} = x_i^{(n+1)} - x_i^{(n)}, \quad \sigma_i^{(n)} = 1 - 2x_i^{(n)}$$

$$z_i^{(n)} = \sum_{j \in E} w_{ij} x_j^{(n)} + b_i$$

$$\lambda_i^{(n)} = \exp\left(\sigma_i^{(n)} z_i^{(n)} / \tau\right), \quad \lambda(n) = \sum_i \lambda_i^{(n)}.$$

The log-conditional likelihood, $\mathcal{L}_D(\theta)$, measuring similarity between model distribution $X(t)$ and $D(t)$, amounts to:

$$\log p\left(X(t) = D(t), 0 < t < t_{N+1} \mid X(0) = x(0)\right) =$$

$$\sum_{n=0}^{N-1} \log p\left(X(t) = D(t), t_n < t < t_{n+1} \mid X(t_n) = x(n)\right)$$

$$= \sum_{n=0}^{N-1} \log \left\{ \prod_i \left(\lambda_i^{(n)} \right)^{\delta_i^{(n)}} \lambda^{(n)} \exp\left( (t_n - t_{n+1}) \lambda^{(n)} \right) \right\}$$

$$= \sum_{n=0}^{N-1} \left\{ T_n(\theta) + H_n(\theta) \right\},$$

with transition terms $T_n(\theta)$ and holding terms $H_n(\theta)$:

$$T_n(\theta) = \sum_{i=1}^d \frac{\delta_i^{(n+1)}}{\tau} z_i^{(n)} / \tau,$$

$$H_n(\theta) = -(t_{n+1} - t_n) \sum_{i=1}^d \lambda_i^{(n)}.$$

Since $-\mathcal{L}_D(\theta)$ is convex (it consists of sums and exponentiations of linear/convex functions), we will adopt gradient methods for parameter optimization. Straightforward calculation yields the gradients of $T_n(\theta)$ and $H_n(\theta)$:

$$\frac{\partial T_n}{\partial w_{jk}} = \frac{1}{\tau} \frac{\delta_j^{(n)}}{\tau}, \quad \frac{\partial T_n}{\partial b_k} = \frac{1}{\tau} \delta_k^{(n)}$$

$$\frac{\partial H_n}{\partial w_{jk}} = -\frac{1}{\tau} \frac{\delta_j^{(n)}}{\tau} \lambda_k^{(n)} (t_{n+1} - t_n),$$

$$\frac{\partial H_n}{\partial b_k} = -\frac{1}{\tau} \delta_k^{(n)} \lambda_k^{(n)} (t_{n+1} - t_n).$$

Note that these rules are highly local, in the sense that the update for a synaptic weight $w_{ik}$ depends only on the presynaptic state $x_i$, the postsynaptic rate $\lambda_k$, and the postsynaptic
transition $\delta_k$. Similarly, the update for the bias $b_k$ depends only on the rate $\lambda_k$ and transition $\delta_k$. Moreover, any parameter update that agrees in sign with the above gradients will increase both the transition and holding terms of the log-conditional likelihood. Therefore, the training algorithm is robust to noise (up to sign) in the parameter updates. See SM C for the training algorithm pseudocode.

**Spike-Timing-Dependent Plasticity**

Recent work in neuroscience (Ermentrout and Terman 2010) has shown that individual neurons often do not behave as single spiking units. In fact, different compartments of a neuron, such as the apical and basal regions of a pyramidal neuron, can spike apart from the soma. To model such neural dynamics with a McCulloch-Pitts network, we represent the compartments of a neuron by the vertices of the graph $G$, and refer to them as units rather than neurons to avoid confusion.

We assume that each spiking unit $i$ is either an armed state $x_i = 0$, capable of spiking, or in a refractory state $x_i = 1$, representing a unit that just spikd and is incapable of spiking again until it recovers. We assume that a unit influences the spiking of a neighboring unit if and only if it is in the refractory state, a reasonable condition also assumed in (Peceski and Maass 2016). In addition, to account for a well-known observation that the transmission of spikes across synapses is unreliable (Faisal, Selen, and Wolpert 2008), we model the random chance of spiking events that depend on synaptic strengths $w_{ij}$ and unit biases $b_i$ as stochastic processes. Specifically, the wait time to the next spiking event of unit $i$ is assumed to be exponentially distributed with rate

$$\lambda_i(\theta) = r_i \exp \left\{ \sigma_i z_i / \tau \right\},$$

where

$$\sigma_i = 1 - 2x_i, z_i = \sum_{j \in E} w_{ji} x_j + b_i,$$

and where the parameter $r_i$ controls the background spiking rate of the unit when $z_i$ vanishes. Note that when the unit is armed, the expected time to fire exponentially decreases with $z_i$; it is thus tempting to call $z_i$ the “membrane potential” because of its similarity to classical integrate-and-fire neurons. However, spike events are not stochastic when the membrane potential is known (Zador 1998). To avoid confusion, we shall therefore call $z_i$ the unsigned activity and $\sigma_i z_i$ the signed activity. Note also that while most neuron models assume a fixed refractory period, in our model it is stochastic, given by an exponential distribution whose expected time to recovery is inversely proportional to the exponential of $z_i$.

Learning in the network is implemented by two sets of update rules. Let $\eta$ be the learning rate. When a spike occurs, we apply transition updates:

$$\Delta w_{jk} = \frac{\eta}{\tau} \sigma_j^{(n)} \delta_k^{(n)}, \quad \Delta b_k = \frac{\eta}{\tau} \lambda_k^{(n)}. \tag{1}$$

On the other hand, when there are no spikes over a period of time, we apply holding updates:

$$\Delta w_{jk} = -\frac{\eta}{\tau} \sigma_j^{(n)} \lambda_k^{(n)} (t_{n+1} - t_n),$$

$$\Delta b_k = -\frac{\eta}{\tau} \lambda_k^{(n)} (t_{n+1} - t_n). \tag{2}$$

We may also interpret the holding updates as a fixed-rate decay of the weights and biases over time.

In Bi and Poo’s experiments (Bi and Poo 1998) on spiketime-dependent plasticity (STDP), the behavior of a pair of interconnected neurons, in which the synaptic weight is not strong enough for a presynaptic spike to cause a postsynaptic spike, is studied. The presynaptic neuron is initially stimulated to spike, and the amplitude of the excitatory postsynaptic current (EPSC) is measured. After five minutes, the presynaptic neuron is stimulated again to spike. Next, after $\varepsilon$ time, the postsynapse is stimulated to spike. This pair of stimulations is repeated every second for 60 seconds. Finally, after twenty minutes, the amplitude of the EPSC is measured again and the percentage change is recorded.

We prove that the average case behavior of the learning algorithm for an MPN agrees with the experimentally-observed synaptic potentiation and depression in biological neural networks. To simplify the mathematical analysis, we shall make the following assumptions:

1. Consider a synapse with presynaptic and postsynaptic neurons $i, j$ with states $x_i, x_j$. Let $w$ be the synaptic weight (directed weight from $i$ to $j$), and $b_i, b_j$ the neural biases. Let $z_i = b_i z_j = w x_i + b_j$ be the unsigned activities. Let $\lambda_i = r_i \exp(\sigma_i z_i / \tau), \lambda_j = r_j \exp(\sigma_j z_j / \tau)$ be the firing rates, where we set the background rate $r_i = r_j = \tau$, and $\tau$ is the temperature controlling the degree of stochasticity.

2. The neurons do not spike unless they are manually stimulated, but they may recover from their refractory state on their own accord.

3. The weight $w$ is updated according to equations (1) and (2), but the biases $b_i, b_j$ are fixed.

4. The ratio $\eta / \tau$ is much smaller than 1.

5. The refractory periods are on average shorter than the armed periods.

**Theorem 0.1.** Assuming the above conditions, let $\varepsilon$ be the timing of the presynaptic spike subtracted from that of the postsynaptic spike. If $\varepsilon$ is small and positive, then the expected synaptic weight change is

$$E[\Delta w] \approx C_1 e^{-\lambda_1|\varepsilon|}, \quad \lambda_i = re^{-b_i / \tau},$$

but if $\varepsilon$ is small and negative, then the expected synaptic weight change is

$$E[\Delta w] \approx -C_2 e^{-\lambda_j|\varepsilon|}, \quad \lambda_j = re^{-(w+b_j) / \tau},$$

where $C_1, C_2$ are positive constants.

**Proof.** See SM B. □

**Experiments**

In this section, we showcase the biologically plausible features and capabilities of the MPN: We first numerically verify the prediction of Theorem 0.1, reproducing STDP-like curves from our learning rule. The MPN is then shown to be self-consistent, i.e., accurate inference of its own generative model is achievable. We next demonstrate the ability of MPN to robustly memorize spatiotemporal patterns. Namely, we

---

3The codes are available at https://github.com/owen94/MPNets.
When the dynamics are stochastic, we show that stochastic and the video in SM) and learning a generative model of vice versa. Fig.1 displays simulation results for both initial synaptic weight updates of 60 consecutive spikes are accumulated, as sug-
in the refractory state and neuron conditions, which reveals that the expected (average) weight changes discussed in the Spike-Timing-Dependent Plasticity section, curves from our MPN learning rule. Following the algorithm postsynaptic neuron By simulating the dynamics of a presynaptic neuron STDP and MPN Learning Rule

Experimental neural spike-train data. The latter demonstrates that MPNs can effectively reproduce spike-timing statistics in neurobiology experiments.

**STDP and MPN Learning Rule**

By simulating the dynamics of a presynaptic neuron \(i\) and a postsynaptic neuron \(j\), we now verify the emergence of STDP curves from our MPN learning rule. Following the algorithm discussed in the Spike-Timing-Dependent Plasticity section, the expected synaptic weight changes \(\Delta w_{ij}\) can be measured as a function of the spike timing difference \(\epsilon\).

We simulate 10 trials for each \(\epsilon\), while for each trial the updates of 60 consecutive spikes are accumulated, as suggested in (Bi and Poo 1998), to finally compute the new synaptic weight \(\hat{w}_{ij}\). When there are no longer weight updates for every spike, the update will be terminated. Here, we set the learning rate for the transition and holding updates to \(\eta_T = 0.05\) and \(\eta_H = 0.001\), respectively. Other parameters are \(w_{ij} = 1, b_i = b_j = 0\).

Two initial conditions are considered: either neuron \(i\) is in the refractory state and neuron \(j\) is in the armed state or vice versa. Fig.1 displays simulation results for both initial conditions, which reveals that the expected (average) weight change consists of two exponential branches as a function of

\[
E(\Delta w) = \frac{1}{100} |w - \hat{w}|, \\
\epsilon = \text{in agreement with Theorem 0.1. This result is consistent with the STDP rule discussed in (Bi and Poo 1998), and the two branches do not need to be symmetric as assumed in (Scellier and Bengio 2017).}

**Consistency Check:** Here, we numerically show that MPNs can consistently learn the parameters of another MPN that generates training samples. 100,000 samples were generated from the pre-defined MPN of 10 neurons with weights \(w\) and biases \(b\). Next, we train an MPN to fit the generated data. For the reconstruction error using \(\ell_1\) norm defined by \(|\text{error}(w, \hat{w})| = \frac{1}{100} |w - \hat{w}|\), we find \(\text{error}(w, \hat{w}) = 0.03\) after 30 epochs of training. This consistency check shows that MPN is self-consistent. Note also that the training algorithm converges fast, plateauing after 5 training epochs (see SM D.1).

**Robust Spatiotemporal Memory**

**Spatiotemporal memory:** The advantage of weight asymmetry is the flexibility to memorize repeating sequences of spikes (cycles) regarded as spatiotemporal memories. Here, we show that MPN can be trained to memorize a cycle using the deterministic update rule (the noiseless limit is identical to setting \(\tau = 0\)). This update rule amounts to flipping the neuron \(i\) with the highest rate, i.e., \(i = \arg \max \lambda_j\). Fig.2 (top) shows the cycle of period 16 to memorize, which is the repeating sequence to be stored by an MPN with 8 neurons. After training with this sequence, MPN can successfully gen-

\[
E(\Delta w) = \frac{1}{100} |w - \hat{w}|, \\
\epsilon = \text{in agreement with Theorem 0.1. This result is consistent with the STDP rule discussed in (Bi and Poo 1998), and the two branches do not need to be symmetric as assumed in (Scellier and Bengio 2017).}

**Consistency Check:** Here, we numerically show that MPNs can consistently learn the parameters of another MPN that generates training samples. 100,000 samples were generated from the pre-defined MPN of 10 neurons with weights \(w\) and biases \(b\). Next, we train an MPN to fit the generated data. For the reconstruction error using \(\ell_1\) norm defined by \(|\text{error}(w, \hat{w})| = \frac{1}{100} |w - \hat{w}|\), we find \(\text{error}(w, \hat{w}) = 0.03\) after 30 epochs of training. This consistency check shows that MPN is self-consistent. Note also that the training algorithm converges fast, plateauing after 5 training epochs (see SM D.1).

**Robust Spatiotemporal Memory**

**Spatiotemporal memory:** The advantage of weight asymmetry is the flexibility to memorize repeating sequences of spikes (cycles) regarded as spatiotemporal memories. Here, we show that MPN can be trained to memorize a cycle using the deterministic update rule (the noiseless limit is identical to setting \(\tau = 0\)). This update rule amounts to flipping the neuron \(i\) with the highest rate, i.e., \(i = \arg \max \lambda_j\). Fig.2 (top) shows the cycle of period 16 to memorize, which is the repeating sequence to be stored by an MPN with 8 neurons. After training with this sequence, MPN can successfully gen-

\[
E(\Delta w) = \frac{1}{100} |w - \hat{w}|, \\
\epsilon = \text{in agreement with Theorem 0.1. This result is consistent with the STDP rule discussed in (Bi and Poo 1998), and the two branches do not need to be symmetric as assumed in (Scellier and Bengio 2017).}

**Consistency Check:** Here, we numerically show that MPNs can consistently learn the parameters of another MPN that generates training samples. 100,000 samples were generated from the pre-defined MPN of 10 neurons with weights \(w\) and biases \(b\). Next, we train an MPN to fit the generated data. For the reconstruction error using \(\ell_1\) norm defined by \(|\text{error}(w, \hat{w})| = \frac{1}{100} |w - \hat{w}|\), we find \(\text{error}(w, \hat{w}) = 0.03\) after 30 epochs of training. This consistency check shows that MPN is self-consistent. Note also that the training algorithm converges fast, plateauing after 5 training epochs (see SM D.1).

**Robust Spatiotemporal Memory**

**Spatiotemporal memory:** The advantage of weight asymmetry is the flexibility to memorize repeating sequences of spikes (cycles) regarded as spatiotemporal memories. Here, we show that MPN can be trained to memorize a cycle using the deterministic update rule (the noiseless limit is identical to setting \(\tau = 0\)). This update rule amounts to flipping the neuron \(i\) with the highest rate, i.e., \(i = \arg \max \lambda_j\). Fig.2 (top) shows the cycle of period 16 to memorize, which is the repeating sequence to be stored by an MPN with 8 neurons. After training with this sequence, MPN can successfully gen-
which is the repeating sequence (right) stochastic dynamics generated by an MPN that is trained to memorize cycles of period 4; see Fig. 2 in SM for topologies. Fig. 3 (a) which are cycle diagram generated by these MPNs is shown in Fig. 3; and cycle on cycle cycle straightforward and will be reported elsewhere. Generalization to multi-cycle storage in larger networks is spatiotemporal memories in a small network with 4 neurons. Multiple static patterns. Here, we demonstrate the simplest patterns, analogous to standard Hopfield nets that can store multiple static patterns. To train an MPN to store multiple spatiotemporal repeating sequences of period 4 in Fig. 3 (b), which are cycle A \[\{(1, 0, 0, 0), (1, 1, 0, 0), (0, 1, 0, 0), (0, 0, 0, 0)\}\] and cycle B \[\{(0, 0, 0, 1), (0, 0, 1, 1), (0, 1, 0, 0), (0, 1, 1, 0)\}\]. This notation represents an ordered set, in which a sequence generated by the trained MPN proceeds from left to right. In the presence of noise \(\tau > 0\), fluctuations may destroy the robustness of cycles. Namely, rather than remaining in one of the cycles (deterministically stable periodic attractors), states could be driven away from one robust cycle to another through a series of rare one flip transitions. Fig. 3 shows the response of the trained MPNs as the strength of intrinsic noise \(\tau\) increases. Thus, one useful role of noise in MPN is to facilitate transitions between stored spatiotemporal patterns of spikes that would not otherwise be possible in the deterministic limit. Whether one could exploit this intrinsic noise-induced memory switching phenomena to design controllable memory switching networks is worthy of future investigation.

Application: Robustly Memorizing Sequences of Binary Pictures

Robust spatiotemporal memory is an appealing feature of the MPN; however, successive elements of a preferable sequence to store might not be one-hop different as assumed in the previous section. For instance, the successive elements in the sequence of digits “2019” represented as binary pictures of Fig. 5 (left) differ by multi-hop transitions. Here, we show that MPNs can be trained to learn sequences of pictures that differ by multi-hop transitions, illustrated in Figs. 4-5, and also in the video in SM.

The multi-hop away sequence to learn consists of \([\text{the all-zero state, “2”, “0”, “1”, “9”}]\), in this specific order. Each static binary picture in the sequence is represented as a 20 \(\times\) 20 binary matrix. To transit among these static memories in the correct order, we randomly assign 100 one-hop transitions.
with a higher rate will generate the cycles containing multi-hop away memories. As a consequence of our training procedure, the trained MPN produces a sequence that both memories already share. The rates at the other neurons, see Fig.5. This is consistent with the fact that “2” will transit to “0” through multi-hop transitions and that the spiked neurons of “2” will recover with higher rates than those neurons remaining in the armed state. However, the neurons that spike at both the memory “2” and “0”, i.e. the bottom horizontal line, will prefer to not flip, consistent with the almost negligible rates at the neurons that both memories already share. Note finally that while Hopfield networks can be used for recalling or denoising static memories (Hopfield 1982; Hillar and Tran 2018), we have shown that MPNs can be applied to recall spatiotemporal memories or complete spatiotemporal patterns. Thus, the MPN with asymmetric weights can be regarded as an STDP-consistent and non-equilibrium generalization of the dynamics arising from the symmetric Hopfield network. Regarding weight asymmetry, we observe from the trained MPN that the learned weights are considerably asymmetric. The asymmetry measure $|w_{ij} - w_{ji}|/(|w_{ij}| + |w_{ji}|)/2) = 1.38, 1, 53, 1.58$ for training parameters $\tau = 0.5, 1, 2$, respectively.

Application: Inferring Generative Models for Neural Spike-train Data

We now apply the MPN to learn spike-train statistics of a neurobiology experiment, which reports the neuronal activities of cat primary visual cortex \(^3\). The experiment recorded spike-train dataset of 25 neurons. We preprocess the data and

\(^3\)The dataset is available on https://crcns.org/data-sets/vc/pvc-3
While both (Brea, Senn, and Pfister 2013) and this work at-
tempts to learn the distribution of sequences, they differ in
model assumptions, (Brea, Senn, and Pfister 2013) assumes a
continuous-time neuron model with exponentially distributed refractory periods and stochastic neurons. Such different models and assumptions lead to distinct learning rules. Finally, while (Brea, Senn, and Pfister 2013) exploits hidden units to learn a sequence, the MPN reported here consist of only visible neurons. We expect that, by adding hidden neurons, MPN will become even more expressive. We leave hidden neurons problem to be explored in future work.

Biological plausibility (Rezende, Wierstra, and Gerstner
2011; Bengio et al. 2015; 2017; Scellier and Bengio 2017;
Liu, Quek, and Lin 2018) has also been emphasized in the
machine learning community, with the hope to uncover ef-
cient learning algorithms inspired by neuroscience. How-
ever, many proposed STDP-consistent algorithms assume
symmetric weights, a constraint generally considered biolog-
ically implausible. For example, (Scellier and Bengio 2017)
introduces a novel learning paradigm termed Equilibrium
Propagation that offers a biologically plausible mechanism
for backpropagation and credit assignment in Deep Learning.
The learning algorithm implements a form of STDP for neu-
rnal networks with symmetric weights, i.e. \( w_{ij} = w_{ji} \), such
that, in one of the learning phases, the weight update rule
satisfies \( \frac{\partial w_{ij}}{\partial t} \propto \rho(u_i) \frac{\partial p(x_i)}{\partial u_i} + \rho(u_j) \frac{\partial p(x_j)}{\partial u_j} \), where \( u_i \) and \( u_j \) are the membrane potential and its firing rate. In con-
trast, our model possesses a different weight update rule (see
Spike-Timing-Dependent Plasticity section), does not pose
the symmetric weight condition, and can reproduce asymmet-
ric STDP-like curves (see the STDP and MPN Learning Rule
section).

Lastly, our learning algorithm is partly inspired by Min-
imum probability flow (MPF) (Sohl-Dickstein, Battaglino,
and DeWeese 2011; Hillar, Sohl-Dickstein, and Koepsell
2012), a symmetric-weight discrete-time algorithm that ef-
ciently learns distributions of static variables. Although
the update rule of MPF is akin to the the transition up-
dates of an MPN with symmetric weights, an asymmetric-
weight continuous-time MPN can also learn distributions of
sequences.

### Conclusion

We introduce a biologically plausible continuous-time se-
quence learning algorithm for spiking neural networks and
demonstrate its capability to learn neural spike-train data as
well as to robustly store and recall spatiotemporal memories.
The hallmark of our local learning rule is that it provides a
normative explanation of STDP. There are several direc-
tions for future investigations. Firstly, akin to the Hopfield
network capacity problem, we could explore the capacity of
MPNs for storing random repeating sequences. More interest-
ingly, given a set of transitions between states that is not
necessarily a cycle, is it feasible to train the network to mem-
orize those transitions? The answer could hint at whether
MPNs possess large long-sequence storage capacity, a highly
sought-after characteristic of modern sequence learning mod-
els. Furthermore, while the current MPN is a fully-visible
network, extensions to a deep network with hidden layers
for large-scale computer vision and natural language tasks
would be worth investigating. Lastly, implementing the local

![Figure 6: Histogram of the ISI of the spikes generated from the trained MPN is in excellent agreement with that of the test data.](image-url)
learning rule on neuromorphic computing hardware could be a near-term application of our neuro-inspired framework.

Acknowledgement
Zuo Zhu Liu is supported by the Singapore MOE grant MOE2016-T2-2-135 under PI Prof. Alexandre Thiery. Thiparat Chotibut and Shaowei Lin are supported by SUTD-ZJU grant ZJURP1600103. We also thank Andrew Tan, and Andreas Herz for helpful discussions.

References


