Poisson balanced spiking networks

Camille E. Rullán Buxó
Center for Neural Science
New York University

Jonathan W. Pillow
Princeton Neuroscience Institute &
Dept. of Psychology, Princeton University

December 1, 2019

Abstract

An important problem in computational neuroscience is to understand how networks of spiking neurons can carry out various computations underlying behavior. Balanced spiking networks (BSNs) provide a powerful framework for implementing arbitrary linear dynamical systems in networks of integrate-and-fire neurons (Boerlin et al. [1]). However, the classic BSN model requires near-instantaneous transmission of spikes between neurons, which is biologically implausible. Introducing realistic synaptic delays leads to an pathological regime known as "ping-ponging", in which different populations spike maximally in alternating time bins, causing network output to overshoot the target solution. Here we document this phenomenon and provide a novel solution: we show that a network can have realistic synaptic delays while maintaining accuracy and stability if neurons are endowed with conditionally Poisson firing. Formally, we propose two alternate formulations of Poisson balanced spiking networks: (1) a "local" framework, which replaces the hard integrate-and-fire spiking rule within each neuron by a "soft" threshold function, such that firing probability grows as a smooth nonlinear function of membrane potential; and (2) a "population" framework, which reformulates the BSN objective function in terms of expected spike counts over the entire population. We show that both approaches offer improved robustness, allowing for accurate implementation of network dynamics with realistic synaptic delays between neurons. Moreover, both models produce positive correlations between similarly tuned neurons, a feature of real neural populations that is not found in the original BSN. This work unifies balanced spiking networks with Poisson generalized linear models and suggests several promising avenues for future research.

1 Introduction

The brain carries out a wide variety of computations that can be implemented by dynamical systems, from sensory integration [2–5], to working memory [6–8], to movement planning and execution [9–11]. Although the existence of such computations in the brain is well established, the mechanisms by which these computations are implemented in networks of neurons remains poorly understood. One approach to this problem involves statistical modeling, which uses descriptive statistical methods to infer the dynamics of neural activity from recorded spike trains [11–22]. A second approach involves theoretical modeling, which seeks to identify strategies for implementing dynamical systems with networks of idealized model neurons [3, 23–31]. An important example of this second approach is the *balanced spiking network* (BSN) framework introduced by Boerlin *et al* [1].

The BSN model consists of a network of coupled leaky integrate-and-fire (LIF) neurons that can emulate an arbitrary linear dynamical system (LDS). The motivating idea is to design a network that approximates the output of a target LDS with a weighted combination of filtered spike trains. The population is divided into "excitatory" and "inhibitory" populations of neurons, based on whether they contribute positively or negatively to the output. This leads to an intuitive spiking rule: a neuron should spike whenever doing so will reduce the error between the output of the target LDS and the network output, i.e., the weighted combination of filtered spikes emitted so far. To make this work, each neuron has to maintain an internal representation of the error between the desired LDS

output and the current network output. Boerlin *et al* showed that, remarkably, this computation can be mapped precisely onto the dynamics of an LIF neuron. A neuron's membrane potential is a local representation of the network-wide error between target output and current network output, and its spike threshold is proportional to the amount by which adding a spike will reduce this error.

The BSN framework has many appealing characteristics. Spiking is efficient, in the sense that every spike contributes meaningfully to reducing error between target and actual output. The computations performed by the model are robust to perturbations and to the loss of neurons. The model also generates irregular spiking activity with intervals that that resemble those observed in real neurons.

However, the original BSN model has an important shortcoming that limits its plausibility as a model for information processing in real neural circuits. Namely, the model requires unrealistically fast propagation of information between neurons. Because every neuron's membrane potential is tracking the overall error between target and actual output, the membrane potential of *all* neurons has to reset whenever any neuron emits a spike. Failure to impose this reset leads to increased activity as multiple neurons attempt to correct same error. In fact, implementations of the BSN model typically impose a rule enforcing that only one neuron is allowed to spike in a single time bin, effectively allowing spikes to propagate faster than the temporal resolution of the simulation (e.g., 0.1ms). Without this rule, the network can easily enter unstable modes in which excitatory and inhibitory populations emit massive spike bursts in alternating time bins, overshooting the target in an attempt to correct the error from the previous time bin.

Here we show that a probabilistic spiking rule can overcome the need for unrealistically fast propagation of spikes in the BSN framework. The basic intuition for our solution is that instead of making neurons spike deterministically whenever doing so will reduce error, we can allow multiple neurons to spike probabilistically such that error will be reduced on average.

We propose two alternate formulations of BSN with Poisson spiking, distinguished from each other by the level at which the network is attempting to minimize the decoding error. First, we describe a 'local' framework, which preserves the original BSN model dynamics but replaces the hard integrate-and-fire spiking rule with the soft firing threshold of the Poisson generalized linear model (GLM) [32–35]. This spiking rule generates stochastic spiking conditioned only on each neuron's local copy of the error which, on average, leads to a reduction in the population-level read-out error.

Second, we propose a 'population' framework that replaces the greedy, single-neuron perspective of the local and BSN models with a rule based on minimizing the expected error at the population level. A vector of spike rates is generated by calculating the expected spike counts that minimizes the total decoding error. The probability of a single neuron spiking depends on its own weight, as with the local rule, but also takes into account the activity of the entire population of neurons and their weights. This coordination leads to spiking activity that is efficient and invariant to network size. Finally, we show that both the local- and population-level Poisson frameworks make the BSN robust to synaptic delays.

This paper is organized as follows. We begin with a pedagogical review to the BSN model (Sec. 2). We then examine the model's dependence on instantaneous spike propagation, and document the unstable behavior that arises if multiple spikes are allowed in a single time bin (Sec. 3). To address this shortcoming, we introduce local and population BSN models with conditionally Poisson spiking (Sec. 4). Finally, we illustrate the accuracy and robustness of these models to synaptic transmission delays (Sec. 5).

2 Background: balanced spiking network model

Here we provide a brief introduction to the original balanced spiking network (BSN) framework introduced by Boerlin, Machens, & Denève [1]. The goal is to design a spiking network that can accurately implement an arbitrary linear dynamical system. Consider a linear dynamical system defined by:

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + \mathbf{c}(t), \qquad \textit{(target dynamics)} \tag{1}$$

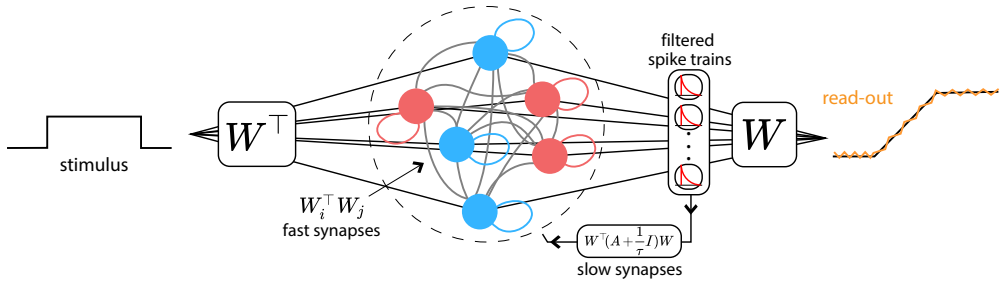

Figure 1: A simple diagram illustrating the BSN. Neurons receive stimulus input projected onto the transpose of a set of linear weights, $W^\top$, and the output is reconstructed by filtering spikes through the same weights, W. Neurons are connected via two coupling weights: fast synapses, $W^\top W$, which instantaneously propagate individual spikes through the network, and slow synapses, $W^\top(A+\frac{1}{\tau})W$, which implement network dynamics by feeding the filtered spike trains back into all neurons in the network. The network is divided into two equal populations of positive (red) and negative (blue) output weights, whose spikes have opposite effects on network output.

where $\mathbf{x}(t) = (x_1(t)...x_J(t))^\top$ is a vector of J dynamic variables that we will refer to as the *target*, A is the $J \times J$ linear dynamics matrix, and $\mathbf{c}_t = (c_i(t), ...c_J(t))^\top$ is a J-dimensional vector of inputs. The BSN model consists of a spiking network of N neurons that attempts to approximate the target output $\mathbf{x}(t)$ via a weighted combination of filtered spike trains:

$$\hat{\mathbf{x}}(t) = W\mathbf{r}(t) \qquad \textit{(network readout)} \tag{2}$$

where $\mathbf{r}(t)$ is the set of spike trains convolved with an exponential decay function, and W are $J \times N$ readout weights. (See Fig. 1 for a schematic.) In general, for a 1-D dynamical system, the population is divided into equal pools of 'positive' and 'negative' neurons (depending on the signs of their individual weight components) although this is not a strict requirement. For $J > 1$, the 'positive' vs 'negative' distinction does not necessarily apply as the signs of the weights need not be consistent across dimensions.

The i'th component of the vector $\mathbf{r}(t)$ is given by

$$r_i(t) = s_i(t) * h(t) = \int_0^t e^{-t'/\tau}\, s_i(t')dt', \qquad \textit{(filtered spike trains)} \tag{3}$$

where $s_i(t) = \sum_{t^i_{sp}} \delta(t - t^i_{sp})$ denotes the i'th neuron's spike train, defined by a series of delta functions at spike times $\{t^i_{sp}\}$, and τ is the time constant of the exponential filter $h(t)$.

From this starting assumption, Boerlin *et al* introduce a greedy update rule that causes a neuron to spike whenever doing so will reduce the squared error between target $\mathbf{x}(t)$ and network output $\hat{\mathbf{x}}(t)$,

$$E(t) = ||\mathbf{x}(t) - \hat{\mathbf{x}}(t)||_2^2, \qquad \textit{(error function)} \tag{4}$$

which is mathematically equivalent to the threshold-crossing spiking rule in a leaky integrate-and-fire (LIF) neuron.

Here we recapitulate the derivation of this spiking rule in discrete time, for clarity and ease of implementation. Let $\hat{\mathbf{x}}_t = W\mathbf{r}_t$ denote the network output at time bin t. The effect of adding a spike from neuron i in this time bin would be to augment the output vector by that neuron's decoding weight vector $\mathbf{w}_i$, which is given by i'th row of the decoding weight matrix W. Thus, network output is $\hat{\mathbf{x}}_t$ if neuron i is silent and $\hat{\mathbf{x}}_t + \mathbf{w}_i$ if it spikes. This suggests that the neuron should spike if doing so will result in smaller error, or

$$||\mathbf{x}_t - (\hat{\mathbf{x}}_t + \mathbf{w}_i)||_2^2 < ||\mathbf{x}_t - \hat{\mathbf{x}}_t||_2^2. \tag{5}$$

Simplifying this expression yields the condition that the neuron should spike if projection of the error vector onto $\mathbf{w}_i$ is greater than half the squared L_2 norm of $\mathbf{w}_i$:

$$\mathbf{w}_i{}^\top(\mathbf{x}_t - \hat{\mathbf{x}}_t) > \tfrac{1}{2}||\mathbf{w}_i||_2^2. \qquad \textit{(spike condition for neuron } i\textit{)} \tag{6}$$

Boerlin *et al* therefore suggest regarding the time-dependent left hand side of (eq. 6) as the membrane potential for neuron i, and the right hand side as its spike threshold T_i:

$$v_{[i]t} = \mathbf{w}_i{}^\top(\mathbf{x}_t - \hat{\mathbf{x}}_t) \quad \textit{(membrane potential)} \tag{7}$$

$$T_i = \frac{1}{2}||\mathbf{w}_i||_2^2 \qquad \textit{(threshold).} \tag{8}$$

Under this view, each neuron is computing a local approximation to the difference between the true target $\mathbf{x}_t$ and the network's current output $\hat{\mathbf{x}}_t = W\mathbf{r}_t$, projected onto that neuron's weight vector $\mathbf{w}_i$.

The only missing piece from this expression is that the neurons do not of course have access to the true value of $\mathbf{x}_t$. But they do have implicit access to A, and thus to the dynamics governing $\mathbf{x}_t$. Boerlin *et al* therefore propose that the the network output $\hat{\mathbf{x}}_t$ is sufficiently close to the target output $\mathbf{x}_t$ that it can be used to accurately approximate the desired dynamics: $A\mathbf{x}_t \approx A\hat{\mathbf{x}}_t$. To make this explicit, we introduce a proxy variable $\mathbf{z}_t$, which denotes the network's own (internal) approximation to the true target $\mathbf{x}_t$. This proxy variable evolves according to

$$\begin{aligned} \mathbf{z}_t &= \mathbf{z}_{t-1} + \Delta(A\hat{\mathbf{x}}_{t-1} + \mathbf{c}_t) \\ &= \mathbf{z}_{t-1} + \Delta(AW\mathbf{r}_{t-1} + \mathbf{c}_t), \quad \textit{(proxy variable)} \end{aligned} \tag{9}$$

where for simplicity we use a forward Euler method for integrating the dynamics equation (eq. 1) with time bin size Δ. Higher accuracy can be achieved using exponential Euler integration (see Methods). Of course $\mathbf{z}_t$ is never represented explicitly; the network tracks $\mathbf{z}_t$ via its projection onto the decoding weights W, as we will see shortly.

2.1 Simulating the BSN model

Simulating the BSN model for a single time bin can be described by a sequence of three steps:

1. Calculate the "pre-spike" membrane potential for each neuron by combining inputs from the previous time step and external input.
2. Apply the threshold to determine which neurons (if any) emit spikes.
3. Reset to obtain "post-spike" membrane potentials $\mathbf{v}_t$ and update filtered spike trains $\mathbf{r}_t$.

We will describe each of these steps in turn. First, the update rule for the pre-spike membrane potential (consistent with eq. 7) is:

$$\begin{aligned} v_{[i]t}^{(pre)} &= \mathbf{w}_i{}^\top(\mathbf{z}_t - \hat{\mathbf{x}}_t^{(pre)}) \\ &= \mathbf{w}_i{}^\top\Big(\mathbf{z}_{t-1} + \Delta(AW\mathbf{r}_{t-1} + \mathbf{c}_t) - (1 - \Delta\tfrac{1}{\tau})W\mathbf{r}_{t-1}\Big) \\ &= v_{[i]t-1} + \Delta\mathbf{w}_i{}^\top\Big((A + \tfrac{1}{\tau}I)W\mathbf{r}_{t-1} + \mathbf{c}_t\Big) \quad \textit{(pre-spike membrane potential)} \end{aligned} \tag{10}$$

where $v_{[i]t}^{(pre)}$ denotes the pre-spike membrane potential for neuron i at time bin t, $\hat{\mathbf{x}}_t^{(pre)} = (1 - \Delta\frac{1}{\tau})W\mathbf{r}_{t-1}$ denotes the network output for the current time bin *before spiking*, and $v_{[i]t-1} = \mathbf{w}_i{}^\top(\mathbf{z}_{t-1} - W\mathbf{r}_{t-1})$ denotes the (post-spike) membrane potential from the previous time step.

Second, spikes for the current time bin are computed by determining whether pre-spike membrane potential exceeds threshold ($v_{[i]t}^{(pre)} > T_i$). When this occurs, the neuron records a spike: $s_{[i]t} = 1$.

Lastly, the filtered spike trains $r_{[i]t}$ are augmented and membrane potential is reset:

$$r_{[i]t} = (1 - \Delta\tfrac{1}{\tau})r_{[i]t-1} + s_{[i]t} \qquad \textit{(filtered spike train update)} \tag{11}$$

$$v_{[i]t} = \mathbf{w}_i{}^\top(\mathbf{z}_t - \hat{\mathbf{x}}_t) = v_{[i]t}^{(pre)} - \mathbf{w}_i{}^\top W\mathbf{s}_t, \quad \textit{(post-spike membrane potential)} \tag{12}$$

which ensures that post-spike membrane potential equals the difference between the projected proxy variable and network output.

2.2 Vector update rules

For convenience, we can rewrite the BSN update equations in vector form. The pre-spike membrane potential is given by:

$$\begin{aligned}\mathbf{v}_t^{(pre)} &= W^\top(\mathbf{z}_t - \hat{\mathbf{x}}_t^{(pre)}) \\ &= \mathbf{v}_{t-1} + \Delta W^\top(A + \tfrac{1}{\tau}I)W\mathbf{r}_{t-1} + \Delta W^\top \mathbf{c}_t \\ &= \mathbf{v}_{t-1} + \Delta\Big(\Omega\mathbf{r}_{t-1} + W^\top\mathbf{c}_t\Big) \qquad \textit{(vector pre-spike membrane potential)}\end{aligned} \tag{13}$$

where $\Omega = W^\top(A + \frac{1}{\tau})W$ are the coupling weights from $\mathbf{r}_{t-1}$ to the pre-spike membrane potential, which implement computation of the divergence between the target $\mathbf{z}_t$ and passive decay of $\mathbf{r}_t$ in the absence of spiking.

Once the spike vector $\mathbf{s}_t$ has been computed, the filtered spike trains and network output for the current time bin are given by:

$$\mathbf{r}_t = (1 - \Delta\tfrac{1}{\tau})\mathbf{r}_{t-1} + \mathbf{s}_t \qquad \textit{(vector of filtered spike trains)} \tag{14}$$

$$\hat{\mathbf{x}}_t = W\mathbf{r}_t, \qquad \textit{(vector network output)} \tag{15}$$

and the vector of post-spike membrane potentials is given by

$$\begin{aligned}\mathbf{v}_t &= W^\top(\mathbf{z}_t - \hat{\mathbf{x}}_t) \\ &= \mathbf{v}_t^{(pre)} - W^\top W\mathbf{s}_t \\ &= \mathbf{v}_{t-1} + \Delta\Big(\Omega\mathbf{r}_{t-1} + W^\top\mathbf{c}_t\Big) - W^\top W\mathbf{s}_t, \quad \textit{(vector post-spike membrane potential)}\end{aligned} \tag{16}$$

which reflects reset of the pre-spike membrane potential after spiking, but can equivalently be seen to be the projected difference between the proxy variable $\mathbf{z}_t$ and the current network output $\hat{\mathbf{x}}_t$ in the current time bin.

It is worth noting that this model requires the instantaneous propagation of spikes between neurons. After a spike, the membrane potential reset (eq. 16) updates $\mathbf{v}_t$ for all neurons based on the spikes in the current time bin via the fast weights, $-W^\top W$. Although Boerlin *et al* refer to the weights $-W^\top W$ as "fast synapses" and the Ω as "slow synapses", note that the $\Omega\mathbf{r}_{t-1}$ term also involves near-instantaneous propagation of information, since the exponentially-filtered spike trains $\mathbf{r}$ jump by 1 after every spike.

The full BSN model first described in [1] contained additional penalties on $\mathbf{r}_t$ in the objective function, which had the effect of reducing spiking by trading off minimization of error (eq. 4) against a cost of inserting spikes. Although we have left these terms out our derivation here for simplicity, including them has the limited effect of changing the spike threshold and post-spike reset and does not change the nature of our findings. The simulations shown in the following sections use the full version as described in the Methods section (8). Simulation parameters for each figure are also included in section (8).

3 Limitations of the BSN model

A key limitation of the BSN model is that it requires unrealistically fast communication between neurons. In the standard integrate-and-fire model, a spike resets only the membrane potential of the neuron that emitted it. In the BSN model, by contrast, the membrane potentials of *all* neurons reset following a spike from any neuron via the $-W^\top W\mathbf{s}_t$ term in (eq. 16). The instantaneous reset of all membrane potentials following a spike is necessary to ensure that each neuron's membrane potential maintains an accurate representation of the read-out after each spike. From a normative standpoint, the hard LIF threshold entails that maintaining an accurate local copy of the error is critical for the network to encode the target.

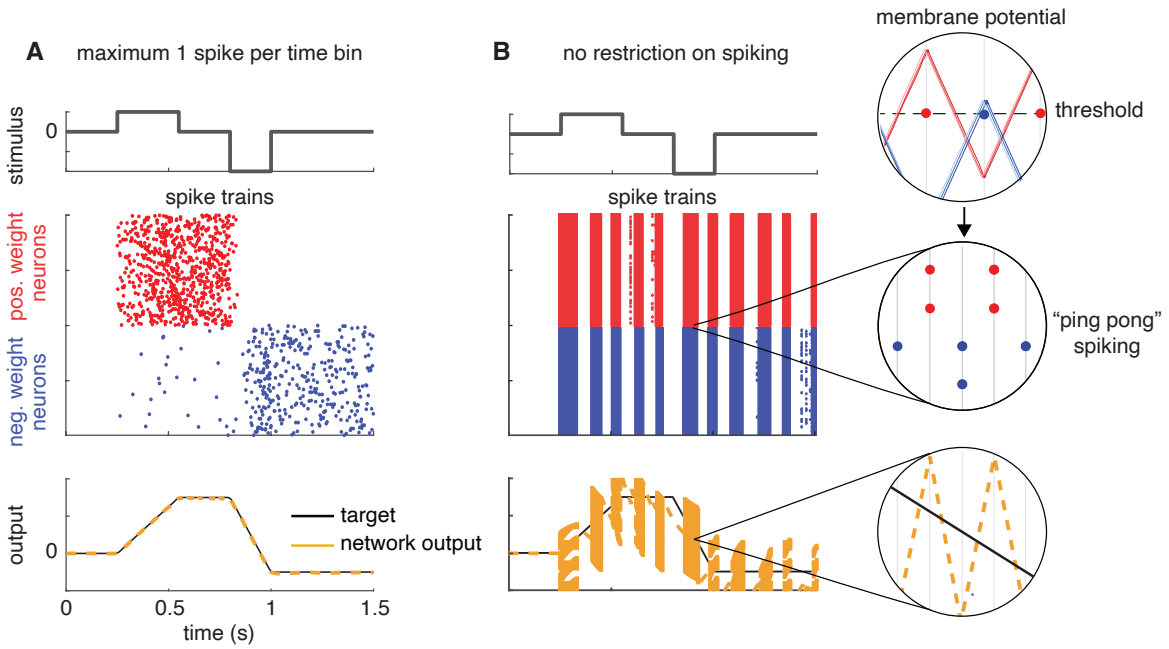

Figure 2: Balanced spiking network implementing a perfect integrator. The network consists of 400 neurons, divided into two populations with output weights of $+0.1$ (red) and -0.1 (blue). **(A)** Simulation results under the condition that only one neuron is allowed to fire per discrete time bin. **(B)** Simulation results when all neurons whose membrane potential is above threshold in a single time bin are allowed to fire, leading to "ping-pong" behavior. Insets show that read-out (yellow) is alternating between large over- and under-estimates of the target (in black). Inset shows the voltage traces of neurons in both positively and negatively weighted populations. Since the weights and inputs are identical across populations, so are the voltage traces. As a consequence, during ping-ponging, all neurons within a population cross the threshold in the same time bin.

In fact, the problem is slightly more complicated: standard implementations of the BSN model include a constraint that only one neuron is allowed to spike in a single time bin. Without this constraint, neurons with similar output weights tend to spike in the same time bin, when a spike from any one of them would have sufficed to compensate for error in network output. This causes the network output to dramatically overshoot the target. In the subsequent time bin, neurons with opposite-sign readout weights fire to compensate for this error, and overshoot the target by a large amount in the opposite direction. This sets up a pathological pattern of oscillatory firing known as "ping-ponging", in which two populations spike maximally in alternating time bins of the discrete simulation [1].

Fig 2 illustrates how ping-pong behavior can arise if multiple spikes are allowed in a single time bin. We set the BSN model to implement a 1-dimensional perfect integrator, $\dot{x}(t) = c(t)$, using the same parameters as the example from figure 1C of Boerlin et al. [1]. In brief, the network contained 400 neurons, divided into two equal sized populations with output weights of +0.1 and -0.1, respectively, which we refer to as positive-output and negative-output neurons (see Methods for complete details). When the rule forbidding multiple spikes per time bin is imposed, the network accurately tracks the target output variable (Fig. 2A). However, removing this constraint — allowing all neurons with membrane potential above threshold to fire — results in ping-pong behavior and large errors in tracking the target (Fig. 2B).

Ping-ponging can in principle be eliminated by adding noise to the membrane potential and using extremely small time bins, which increases the probability that at most one neuron will cross threshold in a single time bin. However, we found that this required a reduction in time bin size by several orders of magnitude. Moreover, this solution relies even more heavily on instantaneous synaptic communication, since it allows neurons to reset even more rapidly after a spike from any single neuron.

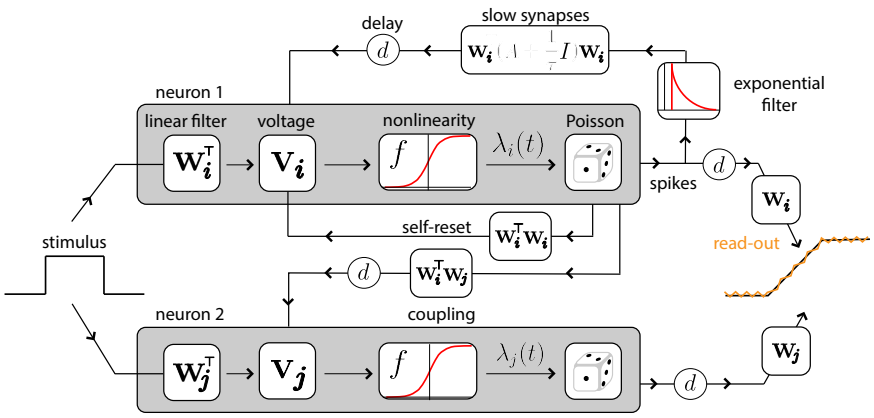

Figure 3: Schematic of BSN with conditionally Poisson neurons. The stimulus influences each neuron's membrane potential $\mathbf{v}_i$ via a set of input weights $W^\top$. The neurons reset themselves via instantaneous, fast synapses. Fast connections to other neurons propagate the effects of spikes with a synaptic time delay d. The desired linear dynamics are implemented via slow weights (filtered through an exponential) also with a time delay d. Within each neuron, spiking is probabilistic with an instantaneous probability of firing $\lambda_i(t) = f(v_i(t))$, where $f(\cdot)$ is a nonlinear function of voltage.

4 BSN with conditionally Poisson neurons

To overcome the problems of instantaneous transmission and network instability, we propose two novel formulations of balanced spiking networks with conditionally Poisson neurons: (1) a local framework, where each neuron spikes independently based on its local estimate of network error; and (2) a population-level framework, which sets firing rates to reduce expected error for the entire population.

The key idea in both frameworks is to replace the deterministic integrate-and-fire spiking rule with probabilistic spiking. Under this modified spiking rule, spiking is governed by an instantaneous probability of spiking λ_t, also known as the *conditional intensity*, such that spiking is independent with probability $\Delta\lambda_t$ in any small time window of width Δ. This results in an auto-regressive Poisson generalized linear model (GLM), also known as a Cox process [33–35]. This model has a quasi-realistic biophysical interpretation [32, 36, 37], and recent work has shown that it can capture a wide range of dynamical behaviors found in real neurons [35].

4.1 Local framework

A simple way to introduce probabilistic spiking to the BSN framework is to replace the hard spike threshold of the integrate-and-fire model with a soft threshold, so that spike probability grows as a nonlinear function of membrane potential, an approach also known as the "escape-rate approximation" [32]. Specifically, we define each neuron's conditional intensity function to be a sigmoidal function of membrane potential:

$$\lambda_t = f(v_t) = \frac{F_{max}}{1 + F_{max} e^{-\alpha(v_t - T)}} + F_{min}, \qquad \textit{(nonlinearity)} \tag{17}$$

where v_t is the membrane potential at time t, T is the spike threshold, F_{max} is the maximal firing rate, F_{min} is a baseline firing rate, and α is a slope parameter governing the sharpness of the threshold. The probability of spiking in a small time interval is proportional to λ_t, which models the spike response as an inhomogenous Poisson process. We refer to this as the local Poisson framework because, like the BSN, spikes are generated by internal dynamics that evolve according to local copies of the representation error.

Although it is common to use exponential nonlinearities for Poisson GLMs, here we have used a scaled sigmoid function to control both the suddenness of firing onset and the maximum achievable firing rate. The parameter α controls the precision of firing onset, while F_{max} and F_{min} control the range of firing rates within a small time

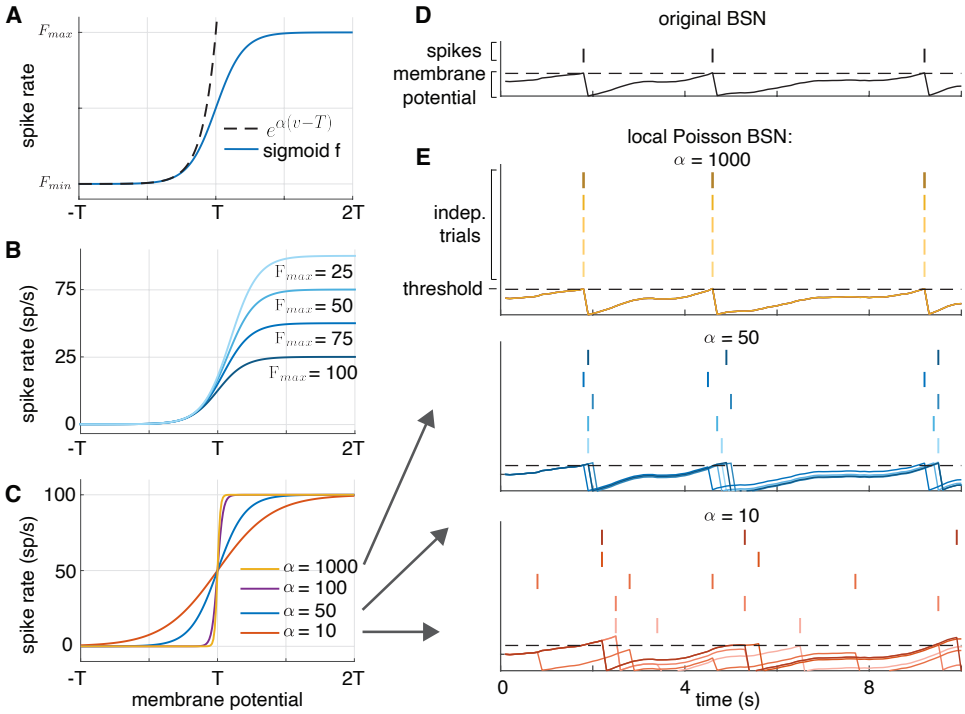

Figure 4: **(A)** The conditional intensity for the exponential non-linearity (dashed lines) and the sigmoid non-linearity (solid lines). The conditional intensity of the sigmoidal non-linearity closely follows that of the exponential non-linearity for sub-threshold voltages, but levels off after threshold, keeping firing rates stable. **(B)** Family of nonlinearities with varying F_{max}. Increasing F_{max} raises the firing rate at which the nonlinearity saturates. **(C)** Family of nonlinearities with varying α. Increasing α increases the steepness of the nonlinearity, which approaches a hard-threshold function as $\alpha \to \infty$ (like the BSN). **(D)** Simulation of the original BSN implementing a perfect integrator, showing membrane potential and spikes of a single example neuron. **(E)** Spikes and membrane potential of the same neuron in a local Poisson BSN implementation of the same system. High α simulations (yellow) replicate the behavior of the BSN integrator. Lowering α to 50 (blue) or 10 (red) results in a spread of spikes centered around the deterministic BSN spikes.

window. The resulting function (eq. 17) resembles an exponential function at low firing rates but saturates at a maximum of F_{max} (see Fig. 4). If we let both $\alpha, F_{max} \to \infty$, we recover the original (hard-threshold) integrate-and-fire rule, in which a spike occurs probability 1 when $v_t > T$. However, for finite α and F, the onset of spiking is more gradual.

As a practical matter, we do not wish to allow a single neuron to fire multiple spikes in a single time bin, because the first spike would preclude additional spikes in the same time bin due to "reset" of the membrane potential. We therefore simulate the model with the spiking rule:

$$P(s_t = 1|\lambda_t) = 1 - \exp(-\Delta\lambda_t), \qquad \textit{(local Poisson framework firing rule)} \tag{18}$$

where $\exp(-\Delta\lambda_t)$ is the probability of observing no spikes in a time bin of size Δ under the Poisson model. For each time-step, we update $\mathbf{v}$ as in the original BSN model, pass it through the nonlinear function $f(\cdot)$ to obtain the vector of Poisson firing rates, $\boldsymbol{\lambda}_t$, and draw spikes as independent Bernoulli random variables with probability as given above. See Fig. 3 for a schematic.

Fig. 4D shows an illustration of how α affects spiking precision and Fig. 4E shows a comparison between spike trains generated by a single-neuron integrating a noisy stimulus for different values of α and those of the deterministic LIF model (in black). As we expect, for high values of α we recover the precise spiking behavior of the BSN model. As α decreases, spike times spread around the ideal BSN spikes. The sub-threshold voltage traces also change with respect to α, indicating that the spiking behavior is a reflection of an underlying error-driven probabilistic process.

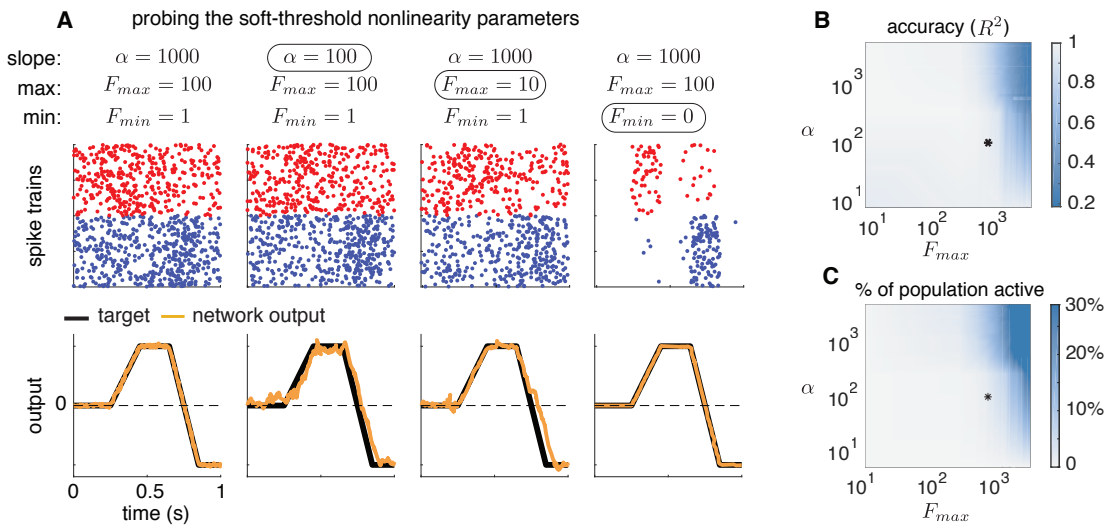

Figure 5: **(A)** Simulations of local Poisson model showing the effects of varying the parameters of the soft-threshold nonlinearity on performance. Relevant parameters are the slope α, maximal firing rate F_{max}, and baseline firing rate F_{min}. Network dynamics implemented a perfect 1D integrator and the stimulus was the same as Fig. 2. Red and blue dots indicate spikes from neurons with positive and negative output weights, respectively. **(B)** Network performance as quantified by R^2 across a range of parameter settings with baseline fixed at $F_{min} = 1$. Black asterisk indicates the values for the rightmost column of A (α = 1000, F_{max} = 100). Accuracy remains high across a broad range of parameter values, falling substantially below 1 when slope and maximum firing rate are both large, which gives rise to ping-ponging. **(C)** Percent of the neural population active as a function of α and F_{max}, showing ping-ponging behavior in upper right corner, where the model approaches a deterministic, hard-threshold firing rule.

Robustness to parameters of the nonlinearity

We studied the effects of varying α, F_{max}, and F_{min} on the performance of the homogeneous integrator network (5) and found that there exists a wide range of values for which the network error is low and the spiking activity is efficient (5E-F). Figures 5A shows raster plots and corresponding read-outs for 'optimal' parameter settings (defined as being in the low error and activity range, denoted by the * in 5E-F), and in 5B-D we modify these parameters to show qualitative changes in spiking activity and read-out accuracy. Decreasing α and F_{max} negatively impacts read-out quality, while removing background spiking returns the network to a high-precision, synchronized regime.

Network performance noticeably deteriorates for very large values of α and F_{max} because we are forcing the precision to be too high while allowing neurons to spike too frequently. Since the spiking rule is localized, large proportions of the population are active at the same time (5F), much like the ping-ponging seen in the BSN model. However, the values at which this happens (e.g., $F_{max} > 10^3$ spikes/second) are well above what we would expect to see in a biological system. Although for the sake of clarity we present the simulation for a relatively simple target function, we still observe a wide ranges of stable F_{max} and α settings for more complex or multi-dimensional simulations.

The computational advantage of our probabilistic spiking rule is that it introduces uncertainty into spike timing, therefore preventing all neurons from firing at once. The parameters controlling this asynchrony, F_{max}, F_{min} and α, have direct physical interpretations (maximal and minimal/background firing rate and error tolerance, respectively) which can be mapped on to characteristics of real neural circuits. By introducing extra degrees of freedom, we are losing the strict normative angle of the original BSN framework. However, what we gain in the process - a considerably expanded space of stable network configurations - makes it possible to introduce realistic communication delays between neurons (see section 5).

4.2 Population framework

We now describe a second framework for implementing BSNs with conditionally Poisson neurons. In this approach, we take a population-level instead of a neuron-level view of the optimization problem to be solved. Instead of assigning each neuron to carry an independent representation of the error between the target and actual network output, we assign each neuron an analog probability of firing such that expected number of spikes across the network compensates appropriately for the total error. We refer to this as the population framework.

The derivation of this framework starts from an error function describing the discrepancy between target and actual network output. However, instead of specifying that each neuron should spike whenever doing so will reduce error (eq. 5), we compute a vector of spike rates $\boldsymbol{\lambda}$, such that the expected spike response across the population over some time window of length κ will minimize error. This leads to the following network objective function:

$$E_t = ||\mathbf{x}_t - (\hat{\mathbf{x}}_t + \kappa W \boldsymbol{\lambda})||_2^2, \qquad \textit{('population' error function)} \tag{19}$$

where $\mathbf{x}_t$ is the target output at time t, $\hat{\mathbf{x}}_t$ is the actual network output at time t, W are the decoding weights, and $\boldsymbol{\lambda}$ is the vector of firing rates (conditional intensities) of a network of Poisson spiking neurons. In this expression, $\kappa W \boldsymbol{\lambda}$ is expected contribution to network output over a time window of size κ. For implementation in discrete time, κ should be an integer multiple of the bin size Δ.

To minimize the above error, we set the instantaneous spike rate vector equal to the least-squares solution:

$$\boldsymbol{\lambda}_t = \frac{1}{\kappa} \tilde{W}(\mathbf{x}_t - \hat{\mathbf{x}}_t) \tag{20}$$

where $\tilde{W} = W^\top (WW^\top)^{-1}$ is the Moore-Penrose pseudo-inverse of the decoding weight matrix W. Poisson neurons firing independently with conditional intensity $\boldsymbol{\lambda}_t$ will therefore minimize the expected error between target and actual network output. Note that if W has orthogonal unit-vector rows, such that $WW^\top = I$, then $\tilde{W} = W^\top$ and we obtain the same encoding weights as the original BSN framework.

In the local framework, increasing the population size means more neurons are competing to reduce the read-out error in a single time window. This increased activity can lead to ping-ponging. The population level view of the problem scales the probability of spiking, $\boldsymbol{\lambda}_{[i]}$, by $WW^\top$, which increases with population size. The responsibility of correcting an error is thus spread across the entire population and activity is stable with respect to network size.

However, the solution in (eq. 19) is not valid generally because the right-hand-side can take on negative values, whereas the conditional intensity for a Poisson process must be positive. To overcome this, we create two mirrored copies of the population. Positive firing rates are assigned to one copy with weight vector W, and negative firing rates are assigned to the other copy with weight vector $-W$. We distinguish between these neurons and 'anti-neurons' by the sign of their membrane potential as determined by the least squared solution. Similarly to the local and BSN models, spikes from either population will have opposite effects on the output variable. However, in this case these designations are not fixed labels and do not apply to the actual sign of $\mathbf{w}_i$. For example, spikes from the i'th neuron will contribute $\mathbf{w}_i$ to the network output, while a spike from its 'anti-neuron' counterpart will have a contribution of $-\mathbf{w}_i$ to network output, but $\mathbf{w}_i$ itself may be positive or negative.

Formally, we define the population framework in terms of the update equations:

$$\mathbf{v}_t = \tilde{W}(\mathbf{z}_t - \hat{\mathbf{x}}_t) \tag{21}$$

$$= \mathbf{v}_{t-1} + \Delta\Big(\tilde{\Omega}\mathbf{r}_{t-1} + \tilde{W}\mathbf{c}_t\Big) \qquad \textit{(pre-spike membrane potential)} \tag{22}$$

$$\boldsymbol{\lambda}_t^{(+)} = \frac{1}{\kappa}\max(\mathbf{v}_t, 0) \qquad \textit{(neuron spike rate)} \tag{23}$$

$$\boldsymbol{\lambda}_t^{(-)} = \frac{1}{\kappa}\max(-\mathbf{v}_t, 0) \qquad \textit{(anti-neuron spike rate)} \tag{24}$$

$$\mathbf{s}_t^{(+)} \sim \mathrm{Poiss}(\Delta\boldsymbol{\lambda}_t^{(+)}) \qquad \textit{(neuron spikes)} \tag{25}$$

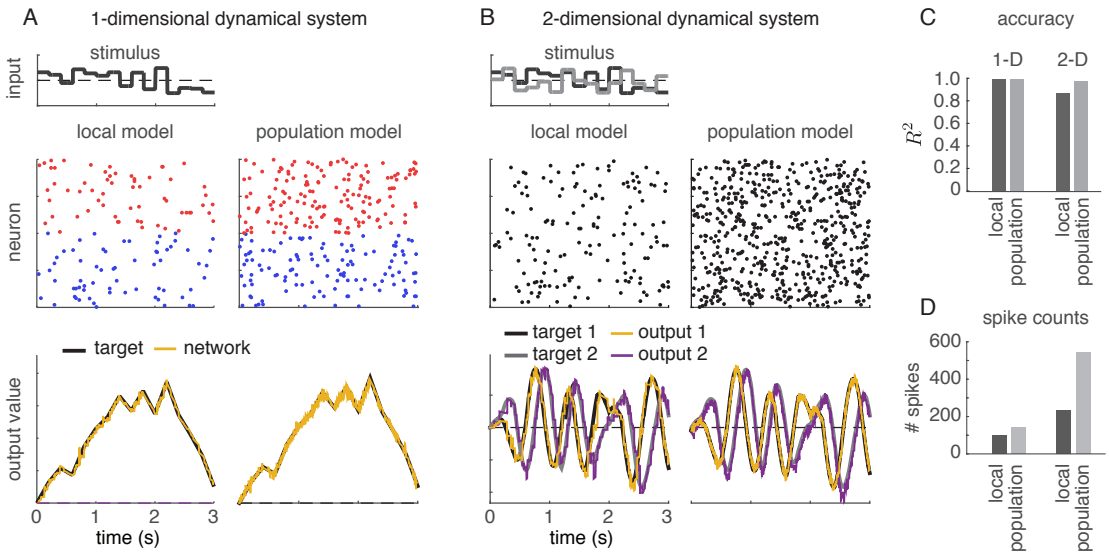

Figure 6: Simulations of the local and population frameworks implementing a 1D and 2D dynamical system. (**A**) The target was a 1-dimensional integrator: $\dot{x}(t) = c(t)$. Left side shows spikes and outputs from local Poisson model, while right side shows spikes and outputs for population Poisson model. As in previous figures, red dots indicate spikes from neurons with positive output weights, blue dots indicate spikes from neurons with negative weights. (**B**) The target was a 2-dimensional oscillator $\dot{x}_1(t) = -x_1(t) - 10x_2(t) + c(t); x_2(t) = 10x_1(t) - x_2(t) + c(t)$. For the population model, the time window for computing expected spike count was κ = 5ms (50 time bins). Weights were randomized to be positive or negative in either dimension, such that neurons are no longer divided into strictly positive- or negative-weight groups. **(C)** Accuracy (R^2) of the two models for 1D and 2D systems. (**D**) Number of spikes emitted by each model during simulations.

$$\mathbf{s}_t^{(-)} \sim \mathrm{Poiss}(\Delta \boldsymbol{\lambda}_t^{(-)}) \qquad \textit{(anti-neuron spikes)} \tag{26}$$

$$\mathbf{r}_t = (1 - \Delta \tfrac{1}{\tau})\mathbf{r}_{t-1} + \mathbf{s}_t^{(+)} - \mathbf{s}_t^{(-)} \qquad \textit{(population filtered spike rates)} \tag{27}$$

where $\tilde{\Omega} = \tilde{W}(A + \frac{1}{\tau})W$ are the coupling weights from $\mathbf{r}_{t-1}$ to the pre-spike membrane potentials. This differs from the standard BSN framework in that spikes, rather than being driven by deterministic threshold crossing, arise from a Poisson process with conditional intensity $\boldsymbol{\lambda}_t^{(+)}$ or $\boldsymbol{\lambda}_t^{(-)}$. If $\mathbf{v}_{[i]t}$ is negative, then $\boldsymbol{\lambda}_{[i]t}^{(+)}$ is set to zero, and the corresponding anti-neuron's firing rate is positive. The voltage updates and spiking resets are identical to the local and BSN models.

Figure 6 shows a comparison of local and population-level frameworks. The population model achieved higher accuracy than the local model for both the one-dimensional and two-dimensional dynamical systems, although at the expense of an increased number of spikes. However, the performance of the local model depends on the parameters of the firing rate nonlinearity, and we could have increased accuracy by increasing steepness α and maximal firing rate F_{max}.

5 Incorporating synaptic time delays

The original BSN model relies on near-instantaneous synaptic communication between neurons due to the fact that all neurons in the population reset immediately after a spike in any neuron. A more realistic model would require that synaptic inputs arrive only after a brief synaptic delay; only the reset of a neuron's own membrane potential following a spike could be considered instantaneous.

To test the robustness of the two Poisson BSN frameworks introduced above, we altered synaptic currents to incorporate a synaptic delays between neurons. In the revised model, spike trains and filtered spike trains received by other neurons are updated only after a synaptic delay d. Thus, if neuron fires at time t, it resets its own

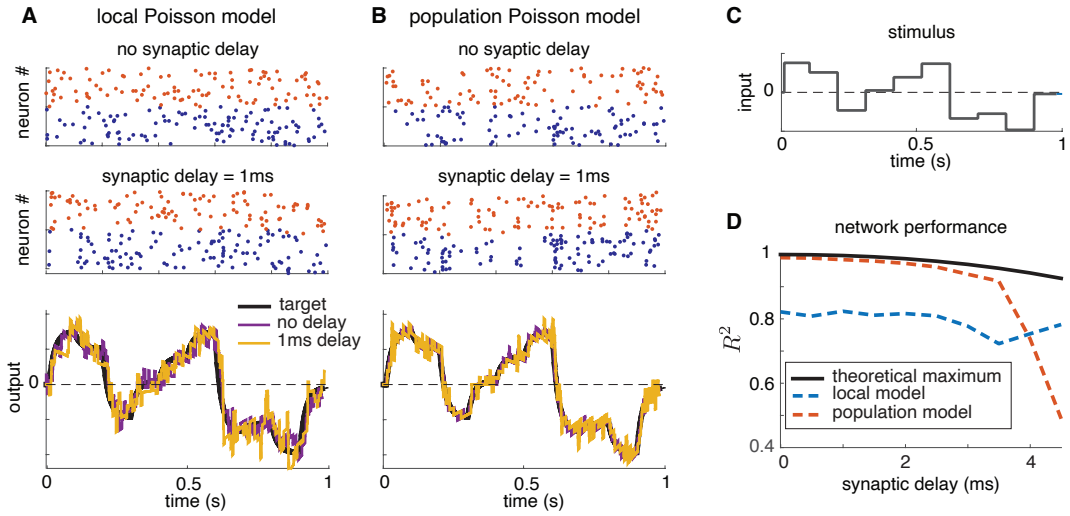

Figure 7: Illustration of local and population conditionally Poisson BSN frameworks with synaptic delays. **(A)** Spike trains simulated from the local Poisson framework implementing a 1D perfect integrator, both without (top) and with a 1-ms synaptic delay (middle). The network output accurately tracked the target variable for both models (bottom). As before, red/blue spike trains indicate neurons with positive/output weights. **(B)** Analogous plots for population Poisson framework. **(C)** Stimulus used for simulations shown in A and B. **(D)** Coefficient of determination (R^2) computed using 50 simulations of each framework. Black trace indicates the maximum possible R^2 value that could be obtained given the exponential Euler integration rule for computing the future target at time $t + d$.

membrane potential in the next time bin, but we will update spike trains and filtered spike trains received by other neurons only at time $t + d$.

To compensate for synaptic delays, we altered the network dynamics so that membrane potential reflects the network error extrapolated d time steps into the future:

$$\mathbf{v}_t = W^\top(\mathbf{z}_{t+d} + \hat{\mathbf{x}}_{t+d}), \qquad \textit{(membrane potential with synaptic delay)} \tag{28}$$

where $\mathbf{z}_{t+d} = \exp(-Ad)\mathbf{z}_t$ represents the network target at time $t + d$, computed by solving the homogeneous differential equation for $\mathbf{z}_{t+d}$ given an initial condition of $\mathbf{z}_t$ and linear dynamics matrix A, and $\hat{\mathbf{x}}_{t+d} = \exp(d/\tau)\hat{\mathbf{x}}_t$ represents the network output at time $t + d$ given by assuming passive decay of the membrane potential. Note here that $\exp(-Ad)$ denotes matrix exponential, and that this is equivalent to using exponential Euler integration to compute future values of $\mathbf{z}$ and $\hat{\mathbf{x}}$ (see Methods for details).

Under these revised dynamics, a membrane potential at time t represents the extrapolated error between true and desired network output at time $t + d$ instead of the instantaneous error at time t. This is equivalent to minimizing an objective function (eqs. 4 or 19) defined in terms of $\mathbf{x}_{t+d}$ and $\hat{\mathbf{x}}_{t+d}$ instead of $\mathbf{x}_t$ and $\hat{\mathbf{x}}_t$. For the population-level Poisson framework, the encoding weights $\tilde{W}$ replace $W^\top$ in (eq. 28).

Figure 7 shows an analysis of the accuracy of the local and population-level Poisson frameworks with synaptic delays. For both models, a 1ms synaptic delay does not have pronounced effects on the spiking activity or the quality of the read-out (Fig. 7A-B). Fig. 7D shows the R^2 values as a function of time delay for both frameworks. We compare it against a theoretical upper bound on the coding accuracy (in black), which is a consequence of the exponential Euler approximation. To determine this bound, we integrated the target dynamics with and without exponential Euler integration and calculated R^2 for all values of d.

For the local framework, the R^2 value is lower than in figure 6 because the parameters were chosen to make the network more robust to synaptic delays. Otherwise, requiring high precision with synaptic delays results in ping-ponging, similar to the higher error regime shown in figure 5E. By contrast, the population framework can maintain high levels of accuracy for a large range of synaptic delays.

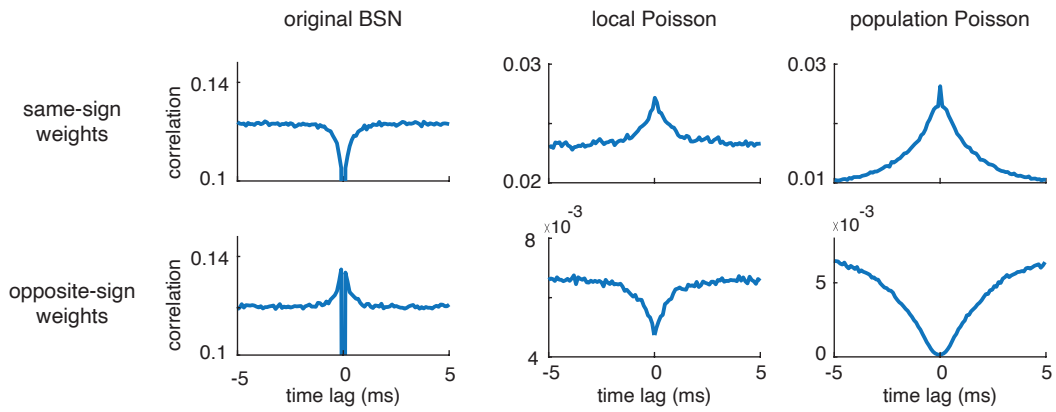

Figure 8: Cross-correlations for the original BSN, local Poisson and population Poisson BSN models with synaptic delay. Top row shows average correlations across pairs of neurons with the same sign output weight (i.e., both positive or both negative). Bottom row shows average correlations across pairs of neurons with opposite-sign output weights (i.e., one positive and one negative neuron). The original BSN network exhibits negative correlations between neurons with the same sign, and positive correlations between neurons with opposite sign. The local and Population Poisson models show the opposite pattern, which more closely resembles correlations found in neural populations in (e.g.) visual cortex.

6 Cross-correlations

Lastly, we compared the statistics of spike trains generated by the original BSN with those of the local and population Poisson BSN models. Figure 8 shows the cross-correlations of spike trains generated by a network of forty neurons implementing a one-dimensional integrator, $\dot{x}(t) = c(t)$, with a white-noise stimulus $c(t)$. Local and population Poisson BSN models both enforced a synaptic delay of 1 ms.

To compute cross-correlations, we divided the neurons into positive-output and negative-output groups. We then computed average within-group (positive-positive and negative-negative) and across-group (positive-negative) cross-correlations. These curves show substantial differences between the original BSN model and the two Poisson models. First, cross-correlations of the original BSN model are 0 at lag zero, due to the rule that only one neuron can spike in a single time bin. More importantly, the within-group correlations for the BSN model exhibit a trough at time zero, meaning that neurons with the same output weight are anti-correlated. Conversely, across-group correlations exhibit an increase at small lags, meaning that neurons with opposite sign output weights are more likely to fire together in a small time window.

This relationship is at odds with correlations in both retina and visual cortex, where studies have reported that correlations are highest for neurons with similar tuning, and lowest for neurons with dissimilar tuning [34, 38–40]. By contrast, the local and population Poisson models successfully recapitulate this pattern of correlations, with a peak in the cross-correlations between pairs of neurons with the same sign weights, and a trough for pairs of opposite-sign neurons. Cross-correlations from these models also exhibit no trough at zero due to the lack of a rule prohibiting simultaneous spiking. Thus, cross-correlations represent an additional dimension of biological plausibility of the proposed Poisson frameworks.

7 Discussion

In this paper, we have highlighted a shortcoming of the balanced spiking network (BSN) paradigm, namely the requirement of near-instantaneous communication between neurons, which arises from the fact that a spike in any neuron causes an instantaneous reset of membrane potential in all other neurons. In practice, the BSN model is often implemented with the additional rule that only one neuron can spike in a single time bin. When synaptic delays are introduced, or multiple spikes are allowed per bin, the model easily enters a ping-ponging

regime in which the network output overshoots and undershoots the target output on alternating time bins.

To address this problem, we proposed two extensions to the BSN model that incorporate conditionally Poisson spiking. Our proposed models both preserve the readout structure of the original BSN, in which a linear combination of exponentially filtered spike trains approximates a linear dynamical system of interest, and the desirable coding characteristics, such as E/I balance. However, they both replace of "hard threshold" integrate-and-fire spiking of the original BSN with a spiking process governed by an instantaneous spike rate or conditional intensity.

In the "local" Poisson BSN framework, the conditional intensity arises from passing the membrane potential through a sigmoidal nonlinearity. The accelerating phase of this nonlinearity is consistent with nonlinearities observed in neural data [41–44] and closely resembles the exponential nonlinearity commonly used in generalized linear modeling analyses [33, 34], while the saturating phase is consistent with saturation in real neural firing rates.

In the "population" Poisson BSN framework, the conditional intensity is obtained by setting the vector of expected spike counts to the least-squares solution for the total output error. This model differs from the original BSN in that the encoding weights, the linear mapping from output error to membrane potential, uses the pseudo-inverse of the decoding weights, $\tilde{W}$, whereas the original BSN used the transpose $W^\top$. This change ensures that the spike rate of each neuron takes account of how many other neurons in the population have similar decoding weights, so that the expected spike count across the entire population in some finite time window compensates optimally for the output error. These modifications make both frameworks robust to both parameter settings and synaptic delays on realistic time scales (1-3ms).

7.1 Related work

Our paper is not the first to address the issue of instability in the BSN. Recent work from Koren and Denève [45] examined the use of penalties on spiking to reduce ping-ponging (referred to in that paper as "up states"). We found that this strategy required fine-tuning and succeeded in a relatively narrow parameter regime compared to the solutions we proposed here. Other work has argued that oscillations in the brain activity may arise from BSNs with synaptic delays, suggesting that a substantially damped form of ping-ponging may be a signature of efficient computation in neural circuits [46].

Recent literature has explored a variety of other extensions and applications of the BSN framework, including nonlinear dynamical systems and the learning of synaptic weights [47, 48], synaptic plasticity rules [49], and biological extensions like finite timescale synapses [50] and synaptic delays [45]. The BSN framework has also been adapted to other computational problems such as probabilistic computation [51] and sensory adaptation [52].

The topic of balanced networks has also received considerable attention outside the specific BSN framework introduced by Boerlin et al. [1]. Balanced networks have been proposed as a substrate for working memory [53, 54], probabilistic inference [55, 56], and the control of complex movements [57]. Excitatory-inhibitory balance is also a key topic in the mathematical theory of neural circuit dynamics, where it has been proposed as an explanation for the correlations found in large-scale population activity [58–60]. Finally, a rich literature has focused on the training of spiking neural networks in more general supervised and reinforcement learning settings, where the objective involves task performance or can only be evaluated at the end of a trial [61–65].

Our work also connects to a rich literature on point process models of neural spike trains. The local Poisson framework draws direct inspiration from the work of Plesser and Gerstner [32], which sought to approximate a noisy integrate-and-fire model with an inhomogeneous Poisson process via the so-called "escape-rate approximation", which refers to the instantaneous probability of noisy membrane potential crossing threshold in a small time window. Subsequent work on the spike response model [36, 66–69] and Poisson generalized linear model [33–35, 70–72] further explored the connection between integrate-and-fire and conditionally Poisson spike train models. The latter are sometimes referred to as "soft-threshold" integrate-and-fire model [73], making the local

Poisson model a natural extension of the original BSN model.

7.2 Future challenges

Although our proposed frameworks are a step in the direction of biological plausibility, there remain a variety of open challenges. One such challenge is the development of neurally plausible learning rules and weight patterns. The network we proposed has a static weight matrix with all-to-all connectivity. A more realistic model would allow for sparse connectivity, sign constraints forcing neurons to be purely excitatory or inhibitory, and plausible learning rules that allow weights to change over time as a function of reward signals. The mathematical treatment of learning in the context of the BSN has proven difficult, although there have been successes learning the fast, slow and feed-forward weights through non-local, supervised, control theoretic approaches [47, 48]. The probabilistic formulation of the Poisson BSN frameworks makes implementing local, Hebbian plasticity rules more tractable, as it opens up the possibility of applying unsupervised learning techniques from traditional machine learning methodology. Finally, we hope to implement time delays in a more principled way, as in [50], and to address the biological plausibility of 'anti-neurons' in the population framework.

A second challenge is the incorporation of nonlinear dynamics. Although the original BSN model was designed to implement linear dynamical systems, it is well known that a wide variety of neural computations are nonlinear. Recent work has proposed an extension of the BSN framework to nonlinear dynamics [31, 47]; combining this approach with conditionally Poisson spiking therefore represents a promising avenue for future work.

Finally, the conditionally Poisson extensions we have proposed provide new opportunities for applying the BSN framework to the interpretation and analysis of real neural data sets. Both the original BSN model and ours assume access to the precise spiking patterns of all the neurons in a population, but real neural recordings typically record only a small fraction of the neurons in a population. Previous work has shown that latent BSN dynamics can be recovered from spike trains in the fully observed case [51]. Other work has discussed the recovery of Poisson generalized linear models from partial recordings [74, 75]. This motivates the development of new methods for identifying balanced network dynamics and computations from partially observed data sets, which may offer fundamental insights into spike-based computation in the brain.

8 Methods

Exponential Euler integration

Exponential Euler integration is a method for solving first-order differential equations of the form

$$\dot{y}_t = -Ay + g(y) \tag{29}$$

where $-Ay$ is a linear term and the nonlinear terms are grouped in $g(y)$. For equations without nonlinear terms, it above can be solved exactly from time 0 to a later time t as

$$y_t = e^{At} y_0 \tag{30}$$

We use this method to approximate the value of a differential equation at a later time, t', by approximating the value of a function y_t at time $t + t'$ as

$$y_{t+t'} = e^{At'} y_t \tag{31}$$

where the accuracy of the approximation decreases as t' increases.

Simulation parameters

For all simulations, time is measured in seconds and dt = 0.1ms. All other simulation parameters are shown in the table below.

Figure 2 was generated using the parameter settings described in Boerlin *et al*, fig. 1C (included below for comparison). The cost terms are $\mu = 10^{-6}$ and $\nu = 10^{-5}$ and the voltage decay constant is $\tau_v = 20$. The noise added to the voltage dynamics and the stimulus was Gaussian with $\sigma_v = 10^{-3}$ and $\sigma_c = 0.01$, respectively.When enforcing the constraint that one neuron should spike per time bin, we selected the neuron with the highest voltage above threshold. Another option with similar results would be to select randomly between the ones above threshold.

Figure	N	W	α	F_{max}	F_{min}	κ	A	τ	d
2	400	$\pm$ 0.1	N/A	N/Ax	N/A	N/A	0	10	0
4D, BSN	1	1	N/A	N/A	N/A	N/A	0	20	0
4D, GLM	1	1	Varies	50	0	N/A	0	20	0
5	400	$\pm$.1	1000 (ACD) 100 (B)	100 (ABD) 10 (C)	1 (A-C) 0 (D)	N/A	0	10	0
6A-B	400	$\pm$.2 + noise (μ = .4)	80	1	0	50	0	10	0 (A) 1ms (B)
6C-D	400	$\pm$.2 + noise (μ = .4)	80	1	0	50	$\begin{pmatrix} -1 & -10 \\ 10 & -1 \end{pmatrix}$	10	0 (C) 1ms (D)
7	200	$\pm$ 0.02 + noise (μ = .01)	500	100	0	25	-50	5	0 and 1ms
8	40	$\pm$ 0.0025 + noise (μ = .01)	500	10	0	20	0	2	.1ms

Table 1: Simulation parameters

As a reminder, N is the number of neurons in the network and W is the vector of read-out weights. For the local framework, α is the slope of the exponential nonlinearity, F_{max} is the saturation, and F_{min} is the minimum firing rate. For the population framework, κ is the time window over which the network minimizes the error. Finally, A is the dynamics matrix of the linear dynamical system, τ is the decay time constant of the filtered spike trains $\mathbf{r}$, and d is the time delay.

A software implementation of Poisson BSNs, along with code to re-generate all simulation figures shown in the manuscript, is available at `https://github.com/pillowlab/PoissonBalancedNets`.

Cost Terms

The original BSN model objective function incorporated two additional cost terms to penalize spiking: a quadratic cost term μ and the linear cost term ν. These terms encouraged the network to use fewer spikes and to distribute spiking more evenly across neurons with large and small output weights. We did not incorporate these in our derivation for clarity, but they are included in our simulations of the BSN.

Including both cost terms into the derivation in Section 2, equation (4) becomes

$$E_t = ||\mathbf{x}_t - \hat{\mathbf{x}}_t||_2^2 + \nu||\mathbf{r}_t||_1 + \mu||\mathbf{r}_t||_2^2 \tag{32}$$

and the voltage and threshold equations become

$$\mathbf{v}_t = W^\top(\mathbf{x}_t - \hat{\mathbf{x}}_t) - \frac{\mu}{\tau}\mathbf{r}_t \tag{33}$$

$$\dot{\mathbf{v}}_t = -\frac{1}{\tau_v}\mathbf{v}_{t-1} + \Omega\mathbf{r}_t - (W^\top W + \frac{\mu}{\tau^2})\mathbf{s}_t + W^\top\mathbf{c}_t \tag{34}$$

$$T_i = \frac{\frac{\nu}{\tau} + \frac{\mu}{\tau^2} + ||W_i||^2}{2} \tag{35}$$

The linear cost term (ν) is proportional to the L1 norm of $\mathbf{r}(t)$, or $||\mathbf{r}||_1 = \sum_{i=1}^N r_{[i]t}$. This cost term penalizes the network's total activity. The quadratic cost term (μ) limits individual neuron firing rates, forcing a spread of activity

across all neurons in a population. Over time, the network transfers activity from precise, costly neurons with high firing rates to imprecise, larger weighted neurons to maintain a compromise between efficiency and accuracy of the read-out. Boerlin *et al* also include a voltage leak term for biological realism.

In our Poisson models, we did not observe the ping-pong effects described in Boerlin *et al* for the range of parameters we considered, so we don't need cost terms for network stability. For the local Poisson framework, the cost terms can be included when α and F_{max} are high enough to cause ping-ponging.

References

[1] Martin Boerlin, Christian K. Machens, and Sophie Denève. Predictive coding of dynamical variables in balanced spiking networks. *PLoS Comput Biol*, 9(11):e1003258, 11 2013.

[2] J. I. Gold and M. N. Shadlen. Neural computations that underlie decisions about sensory stimuli. *Trends Cogn Sci*, 5(1):10–16, Jan 2001.

[3] K. F. Wong, A. C. Huk, M. N. Shadlen, and X. J. Wang. Neural circuit dynamics underlying accumulation of time-varying evidence during perceptual decision making. *Front Comput Neurosci*, 1:6, 2007. doi: $10.3389/\mathrm{neuro}.10.006.2007$.

[4] Bingni W Brunton, Matthew M Botvinick, and Carlos D Brody. Rats and humans can optimally accumulate evidence for decision-making. *Science*, 340(6128):95–98, 2013.

[5] V. Mante, D. Sussillo, K. V. Shenoy, and W. T Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature*, 503(7474):78–84, 2013.

[6] Christian K Machens, Ranulfo Romo, and Carlos D Brody. Flexible control of mutual inhibition: a neural model of two-interval discrimination. *Science*, 307(5712):1121–1124, 2005.

[7] Mark S. Goldman. Memory without feedback in a neural network. *Neuron*, 61(4):621 – 634, 2009. ISSN 0896-6273.

[8] Omri Barak, David Sussillo, Ranulfo Romo, Misha Tsodyks, and L.F. Abbott. From fixed points to chaos: Three models of delayed discrimination. *Progress in Neurobiology*, 103(0):214–222, 2013. ISSN 0301-0082. Conversion of Sensory Signals into Perceptions, Memories and Decisions.

[9] M. M. Churchland, J. P. Cunningham, M. T. Kaufman, S. I. Ryu, and K. V. Shenoy. Cortical preparatory activity: representation of movement or first cog in a dynamical machine. *Neuron*, 68(3):387 – 400, 2010. ISSN 0896-6273.

[10] Mark M Churchland, John P Cunningham, Matthew T Kaufman, Justin D Foster, Paul Nuyujukian, Stephen I Ryu, and Krishna V Shenoy. Neural population dynamics during reaching. *Nature*, 487(7405):51–56, 2012.

[11] Chethan Pandarinath, Daniel J. O'Shea, Jasmine Collins, Rafal Jozefowicz, Sergey D. Stavisky, Jonathan C. Kao, Eric M. Trautmann, Matthew T. Kaufman, Stephen I. Ryu, Leigh R. Hochberg, Jaimie M. Henderson, Krishna V. Shenoy, L. F. Abbott, and David Sussillo. Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature Methods*, 15(10):805–815, 2018. doi: $10.1038/\mathrm{s41592\text{-}018\text{-}0109\text{-}9}$.

[12] Jakob H Macke, Lars Buesing, John P Cunningham, M Yu Byron, Krishna V Shenoy, and Maneesh Sahani. Empirical models of spiking in neural populations. In *Advances in neural information processing systems*, volume 24, pages 1350–1358, 2011.

[13] B. Petreska, M Yu Byron, J. P Cunningham, G. Santhanam, S. I Ryu, K. V. Shenoy, and M. Sahani. Dynamical segmentation of single trials from population neural data. In *Advances in neural information processing systems*, pages 756–764, 2011.

[14] Lars Buesing, Jakob Macke, and Maneesh Sahani. Spectral learning of linear dynamics from generalised-linear observations with application to neural population data. In *Advances in Neural Information Processing Systems 25*, pages 1691–1699, 2012.

[15] E. W. Archer, U. Koster, J. W. Pillow, and J. H. Macke. Low-dimensional models of neural population activity in sensory cortical circuits. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 343–351. Curran Associates, Inc., 2014.

[16] E. Archer, Il Memming Park, L. Buesing, J. Cunningham, and L. Paninski. Black box variational inference for state space models. *arXiv preprint arXiv:1511.07367*, 2015.

[17] Yuanjun Gao, Evan W Archer, Liam Paninski, and John P Cunningham. Linear dynamical neural population models through nonlinear embeddings. In *Advances in Neural Information Processing Systems*, pages 163–171, 2016.

[18] Yuan Zhao and Il Memming Park. Variational latent gaussian process for recovering single-trial dynamics from population spike trains. *Neural Computation*, 29(5):1293–1316, 2017.

[19] S. Linderman, M. Johnson, A. Miller, R. Adams, D. Blei, and L. Paninski. Bayesian Learning and Inference in Recurrent Switching Linear Dynamical Systems. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54, pages 914–922, 2017.

[20] D. Hernandez, A. K. Moretti, Z. Wei, S. Saxena, J. Cunningham, and L. Paninski. A novel variational family for hidden nonlinear markov models. *arXiv preprint arXiv:1811.02459*, 2018.

[21] Yuan Zhao and Il Memming Park. Variational joint filtering. *arXiv preprint arXiv:1707.09049*, 2019. Memming's latest on nonlinear latent dynamics w/ Poisson noise.

[22] Lea Duncker, Gergo Bohner, Julien Boussard, and Maneesh Sahani. Learning interpretable continuous-time models of latent stochastic dynamical systems. *arXiv preprint arXiv:1902.04420*, 2019.

[23] H. Sompolinsky, A. Crisanti, and H. J. Sommers. Chaos in random neural networks. *Phys. Rev. Lett.*, 61: 259–262, Jul 1988. doi: $10.1103/\mathrm{PhysRevLett.61.259}$.

[24] C. Vreeswijk and H. Sompolinsky. Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science*, 274(5293):1724, 1996.

[25] N. Brunel. Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *Journal of computational neuroscience*, 8(3):183–208, 2000.

[26] S. Denève, P. E. Latham, and A. Pouget. Efficient computation and cue integration with noisy population codes. *Nat Neurosci*, 4(8):826–831, Aug 2001. doi: $10.1038/90541$.

[27] S. Ganguli, D. Huh, and H. Sompolinsky. Memory traces in dynamical systems. *Proceedings of the National Academy of Sciences*, 105(48):18970–18975, 2008.

[28] David Sussillo and L. F. Abbott. Generating coherent patterns of activity from chaotic neural networks. *Neuron*, 63(4):544–557, Aug 2009.

[29] C. Eliasmith, T.C. Stewart, X. Choo, T. Bekolay, T. DeWolf, C. Tang, and D. Rasmussen. A large-scale model of the functioning brain. *science*, 338(6111):1202–1205, 2012.

[30] Dimitry Fisher, Itsaso Olasagasti, David W Tank, Emre RF Aksay, and Mark S Goldman. A modeling framework for deriving the structural and functional architecture of a short-term memory microcircuit. *Neuron*, 79 (5):987–1000, 2013.

[31] Larry F Abbott, Brian DePasquale, and Raoul-Martin Memmesheimer. Building functional networks of spiking model neurons. *Nature neuroscience*, 19(3):350, 2016.

[32] H. E. Plesser and W. Gerstner. Noise in integrate-and-fire neurons: from stochastic input to escape rates. *Neural Comput*, 12(2):367–384, Feb 2000.

[33] W. Truccolo, U. T. Eden, M. R. Fellows, J. P. Donoghue, and E. N. Brown. A point process framework for relating neural spiking activity to spiking history, neural ensemble and extrinsic covariate effects. *J. Neurophysiol*, 93(2):1074–1089, 2005.

[34] J. W. Pillow, J. Shlens, L. Paninski, A. Sher, A. M. Litke, and E. P. Chichilnisky, E. J. Simoncelli. Spatio-temporal correlations and visual signaling in a complete neuronal population. *Nature*, 454:995–999, 2008.

[35] Alison I. Weber and Jonathan W. Pillow. Capturing the dynamical repertoire of single neurons with generalized linear models. *Neural Computation*, 29(12):3260–3289, 2017.

[36] R. Jolivet, T. Lewis, and W. Gerstner. The spike response model: a framework to predict neuronal spike trains. *Springer Lecture notes in computer science*, 2714:846–853, 2003.

[37] Kenneth W Latimer, E. J. Chichilnisky, Fred Rieke, and Jonathan W Pillow. Inferring synaptic conductances from spike trains with a biophysically inspired point process model. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 954–962. Curran Associates, Inc., 2014.

[38] Jonathon Shlens, Fred Rieke, and Ej Chichilnisky. Synchronized firing in the retina. *Curr Opin Neurobiol*, 18 (4):396–402, Aug 2008.

[39] A. Kohn and M. A. Smith. Stimulus dependence of neuronal correlation in primary visual cortex of the macaque. *J Neurosci*, 25(14):3661–3673, Apr 2005.

[40] Matthew A Smith and Adam Kohn. Spatial and temporal scales of neuronal correlation in primary visual cortex. *The Journal of Neuroscience*, 28(48):12591–12603, 2008.

[41] D. J. Heeger. Normalization of cell responses in cat striate cortex. *Vis Neurosci*, 9(2):181–197, Aug 1992.

[42] Kenneth H Britten, Michael N Shadlen, William T Newsome, and J Anthony Movshon. Responses of neurons in macaque mt to stochastic motion signals. *Visual neuroscience*, 10(6):1157–1169, 1993.

[43] W. S. Geisler and D. G. Albrecht. Bayesian analysis of identification performance in monkey visual cortex: nonlinear mechanisms and stimulus certainty. *Vision research*, 35(19):2723–2730, 1995.

[44] E. J. Chichilnisky. A simple white noise analysis of neuronal light responses. *Network: Computation in Neural Systems*, 12:199–213, 2001.

[45] Veronika Koren and Sophie Denève. Computational account of spontaneous activity as a signature of predictive coding. *PLoS computational biology*, 13(1):e1005355, 2017.

[46] Matthew Chalk, Boris Gutkin, and Sophie Deneve. Neural oscillations as a signature of efficient coding in the presence of synaptic delays. *Elife*, 5:e13824, 2016.

[47] Alireza Alemi, Christian Machens, Sophie Denève, and Jean-Jacques Slotine. Learning arbitrary dynamics in efficient, balanced spiking networks using local plasticity rules. *arXiv preprint arXiv:1705.08026*, 2017.

[48] S. Denève, A. Alemi, and R. Bourdoukan. The brain as an efficient and robust adaptive learner. *Neuron*, 94 (5):969–977, 2017.

[49] Ralph Bourdoukan, David Barrett, Sophie Deneve, and Christian K Machens. Learning optimal spike-based representations. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2285–2293. Curran Associates, Inc., 2012.

[50] Michael A Schwemmer, Adrienne L Fairhall, Sophie Denéve, and Eric T Shea-Brown. Constructing precisely computing networks with biophysical spiking neurons. *Journal of Neuroscience*, 35(28):10112–10134, 2015.

[51] Cristina Savin and Sophie Deneve. Spatio-temporal representations of uncertainty in spiking neural networks. In *Advances in Neural Information Processing Systems*, pages 2024–2032, 2014.

[52] Gabrielle J Gutierrez and Sophie Denève. Population adaptation in efficient balanced networks. *eLife*, 8, 2019.

[53] Sukbin Lim and Mark S Goldman. Balanced cortical microcircuitry for maintaining information in working memory. *Nature neuroscience*, 16(9):1306–1314, 2013.

[54] Sukbin Lim and Mark S Goldman. Balanced cortical microcircuitry for spatial working memory based on corrective feedback control. *Journal of Neuroscience*, 34(20):6790–6806, 2014.

[55] Martin Boerlin and Sophie Denève. Spike-based population coding and working memory. *PLoS Comput Biol*, 7(2):e1001080, 02 2011.

[56] Guillaume Hennequin, Laurence Aitchison, and Máté Lengyel. Fast sampling-based inference in balanced neuronal networks. In *Advances in neural information processing systems*, pages 2240–2248, 2014.

[57] Guillaume Hennequin, Tim P Vogels, and Wulfram Gerstner. Optimal control of transient dynamics in balanced networks supports generation of complex movements. *Neuron*, 82(6):1394–1406, 2014.

[58] B. Haider, A. Duque, A. R. Hasenstaub, and D. A. McCormick. Neocortical network activity in vivo is generated through a dynamic balance of excitation and inhibition. *Journal of Neuroscience*, 26(17):4535–4545, 2006.

[59] Ashok Litwin-Kumar and Brent Doiron. Slow dynamics and high variability in balanced cortical networks with clustered connections. *Nature neuroscience*, 15(11):1498, 2012.

[60] Brent Doiron, Ashok Litwin-Kumar, Robert Rosenbaum, Gabriel K Ocker, and Krešimir Josić. The mechanics of state-dependent neural correlations. *Nature neuroscience*, 19(3):383, 2016.

[61] H Sebastian Seung. Learning in spiking neural networks by reinforcement of stochastic synaptic transmission. *Neuron*, 40(6):1063–1073, 2003.

[62] Ioana Sporea and André Grüning. Supervised learning in multilayer spiking neural networks. *Neural computation*, 25(2):473–509, 2013.

[63] Wilten Nicola and Claudia Clopath. Supervised learning in spiking neural networks with force training. *Nature communications*, 8(1):2208, 2017.

[64] Brian DePasquale, Christopher J. Cueva, Kanaka Rajan, G. Sean Escola, and L. F. Abbott. full-force: A target-based method for training recurrent networks. *PLOS ONE*, 13(2):1–18, 02 2018. doi: 10.1371/journal.pone.0191527.

[65] F. Zenke and S. Ganguli. Superspike: Supervised learning in multilayer spiking neural networks. *Neural computation*, 30(6):1514–1541, 2018.

[66] W. Gerstner. A framework for spiking neuron models: The spike response model. In F. Moss and S. Gielen, editors, *The Handbook of Biological Physics*, volume 4, pages 469–516, 2001.

[67] L. Paninski, J. W. Pillow, and E. P. Simoncelli. Maximum likelihood estimation of a stochastic integrate-and-fire neural model. *Neural Computation*, 16:2533–2561, 2004.

[68] J. W. Pillow, L. Paninski, V. J. Uzzell, E. P. Simoncelli, and E. J. Chichilnisky. Prediction and decoding of retinal ganglion cell responses with a probabilistic spiking model. *The Journal of Neuroscience*, 25:11003–11013, 2005.

[69] Skander Mensi, Richard Naud, and Wulfram Gerstner. From stochastic nonlinear integrate-and-fire to generalized linear models. In *NIPS*, pages 1377–1385, 2011.

[70] R. E. Kass and V. Ventura. A spike-train probability model. *Neural computation*, 13(8):1713–1720, 2001.

[71] B. Aguera y Arcas and A. L. Fairhall. What causes a neuron to spike? *Neural Computation*, 15(8):1789–1807, 2003.

[72] Kenneth Latimer, Fred Rieke, and Jonathan W Pillow. Inferring synaptic inputs from spikes with a conductance-based neural encoding model. *bioRxiv*, page 281089, 2018.

[73] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski. *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge University Press, New York, NY, USA, 2014. ISBN 1107635195, 9781107635197.

[74] J. W. Pillow and P. Latham. Neural characterization in partially observed populations of spiking neurons. In J. C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1161–1168. MIT Press, Cambridge, MA, 2008.

[75] D. J. Rezende and W. Gerstner. Stochastic variational learning in recurrent spiking networks. *Frontiers in Computational Neuroscience*, 8, 2014.