The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Learning Nonlinear Dynamics in Efficient, Balanced Spiking Networks Using Local Plasticity Rules

Aireza Alemi*
École Normale Supérieure
& UC Davis

Christian K. Machens
Champalimaud Centre

Sophie Denève†
École Normale Supérieure

Jean-Jacques Slotine†
MIT

Abstract

The brain uses spikes in neural circuits to perform many dynamical computations. The computations are performed with properties such as spiking efficiency, i.e. minimal number of spikes, and robustness to noise. A major obstacle for learning computations in artificial spiking neural networks with such desired biological properties is due to lack of our understanding of how biological spiking neural networks learn computations.

Here, we consider the credit assignment problem, i.e. determining the local contribution of each synapse to the network's global output error, for learning nonlinear dynamical computations in a spiking network with the desired properties of biological networks. We approach this problem by fusing the theory of efficient, balanced neural networks (EBN) with nonlinear adaptive control theory to propose a local learning rule. Locality of learning rules are ensured by feeding back into the network its own error, resulting in a learning rule depending solely on presynaptic inputs and error feedbacks. The spiking efficiency and robustness of the network are guaranteed by maintaining a tight excitatory/inhibitory balance, ensuring that each spike represents a local projection of the global output error and minimizes a loss function. The resulting networks can learn to implement complex dynamics with very small numbers of neurons and spikes, exhibit the same spike train variability as observed experimentally, and are extremely robust to noise and neuronal loss.

Recurrent networks in the nervous system perform a variety of tasks that could be formalized as dynamical systems. In many cases, these dynamical systems are learned based on examples ("desired" trajectories), a form of supervised learning. For example, to learn to control an arm, sensorymotor circuits can learn to predict both the arm state trajectories and the sensory feedbacks, that are caused by specific motor commands.

Such learning occurs under several constraints. First, synapses have only access to local information. Because any local change in a synapse could have unpredictable effects on the rest of the network, previous approaches have often used non-local, biologically implausible learning rules such as temporal backpropagation (Werbos 1990) or FORCE learning (Sussillo and Abbott 2009).

Second, information in the nervous system is communicated with spikes. In order to obey the constraints imposed by the brain that is extremely costly to our metabolism (Laughlin, de Ruyter van Steveninck, and Anderson 1998), spiking neural networks should work with reasonably small number of spikes per dimension of the internal state dynamics. Such efficiency requires that there is no redundancy in the representation, so that spike trains of different neurons are uncorrelated.

Third, learning has to be able to resist various perturbations, such as varying levels of noise or the loss of neurons. Such robustness requires some level of degeneracy in neural populations, so that a given network has more neurons than strictly needed to learn a task.

These constraints are also important for implementation of dynamical computation in robotic systems with neuromorphic hardware. The goal in these systems is to design and implement micro-electronic systems that emulate the structure and function of the brain (Schuman et al. 2017). Spiking networks make the transmission of information among units less costly and more robust to noise than firing rate transmission. They can also trade off accuracy for speed or latency. Spiking efficiency make sure the consumption of energy due to emitting spikes is minimal. Robustness makes the system resilient to damage and noise, and locality of learning rules makes the physical implementation of plasticity rules easier in neuromorphic hardware.

No previous approach has been able to meet all three constraints at the same time. A few recent approaches based on adaptive control theory (Slotine and Coetsee 1986; Slotine and Li 1991) have been able to provide online learning rules for spiking networks. In (DeWolf et al. 2016), a spiking network controller was trained to be adaptive to unknown changes in kinematics and dynamics. In (Gilra and Gerstner 2017), a spiking network was trained to predict nonlinear dynamics using a local learning rule. However, none of the works enforce spiking efficiency i.e., they do not require smallest possible number of spikes. Spiking efficiency and robustness were introduced in efficient

*Email: alemi AT ucdavis.edu
†Co-PI, alphabetic order

balanced networks (EBN) (Boerlin, Machens, and Deneve 2013; Deneve and Machens 2016), but supervised learning in these networks has so far been limited to non-local rules (Memmesheimer et al. 2014) or to linear dynamical systems (Bourdoukan and Deneve 2015).

In this study, we fuse the EBN framework (Boerlin, Machens, and Deneve 2013; Deneve and Machens 2016) with adaptive nonlinear control theory (Slotine and Coetsee 1986; Slotine and Li 1991; Sanner and Slotine 1992) in order to derive local learning rules for arbitrary, nonlinear dynamics, while resulting in highly efficient and robust spiking networks. More specifically, we approximate nonlinear dynamics using a set of basis functions. The coefficients of the basis functions are then learned using a rule that utilizes correlation between the coefficients and the error signal (the difference between the desired trajectory and the approximated one). Adaptive control theory provides concepts such as Lyapunov functions and related theorems for quantifying convergence properties of this rule. Based on these concepts we devise a local learning rule and construct an EBN that inherit these tools from adaptive control theory.

Adaptive nonlinear control theory: a teacher-student scenario

The learning task that we consider is presented in Fig. 1A. An input signal $\mathbf{c}(t)$ drives a recurrent spiking network. The task is to learn the network connectivity such that a desired dynamics (the smooth curve in blue with the state variable $\mathbf{x}$) can be decoded from the network spike trains (the zigzag curve in red with the reconstructed state $\widehat{\mathbf{x}}$). This task could correspond to a sensorimotor learning task, where the network learns a *forward internal model* that receives the *efference copy* (i.e. a copy of the input motor command) to predict the body position given the sensory error feedback and the motor command (Wolpert and Ghahramani 2000).

More formally, we assume that the desired dynamics stem from a "teacher" dynamical system of the general form [1]

$$\dot{\mathbf{x}} = f(\mathbf{x}) + \mathbf{c}(t) \tag{1}$$

in which $\mathbf{x}$ is a time-dependent dynamic vector of continuous, real-valued variables, $\mathbf{c}(t)$ is a time-varying input or command signal chosen as a filtered random signal unless otherwise stated, and $f(.)$ is an arbitrary, vector-valued function. Note that if the teacher dynamical system is of higher order, we can transform it into a higher-dimensional, first-order dynamical system.

To understand our approach to learning this teacher system, we will first ignore the neural network, and briefly recapitulate how the problem is solved in adaptive nonlinear control theory (Slotine and Coetsee 1986; Slotine and Li 1991; Sanner and Slotine 1992). Here, the goal is to estimate the parameters of a "student" dynamical system such that its dynamics matches the dynamics of the "teacher" system (See Fig. 1B). The student system has the following form:

$$\dot{\widehat{\mathbf{x}}} = -\lambda\widehat{\mathbf{x}} + \mathbf{W}^\top \boldsymbol{\psi}(\widehat{\mathbf{x}}) + \mathbf{c}(t) + k\,\mathbf{e}, \tag{2}$$

[1] Our dynamic variables $\mathbf{x}(t)$ and input signals $\mathbf{c}(t)$ are a function of time but sometimes for simplicity we omit the time variable t, writing them as $\mathbf{x}$ and $\mathbf{c}$

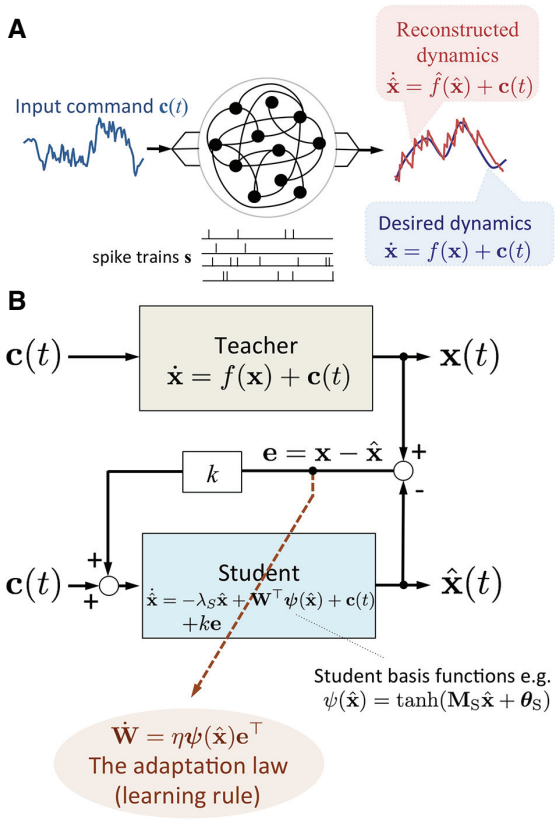

Figure 1: Schematics of the learning task and the control-theoretic approach. A. The task of learning an arbitrary dynamical system. The network is presented with a random input command with certain statistics and it needs to produce a trajectory (the zigzag curve in red) close to the trajectory given by the teacher (the smooth curve in blue). The supervisory error signal, i.e. $\mathbf{e} = \mathbf{x} - \widehat{\mathbf{x}}$, should be used to train the connections of the recurrent network to perform this task. B. Adaptive nonlinear control theory for solving an estimation problem formulated as a teacher-student scenario.

where $\widehat{\mathbf{x}}$ is the dynamic vector variable of the student, $-\lambda\widehat{\mathbf{x}}$ is a leak-term, $\mathbf{W}$ are the parameters of the student dynamics, $\boldsymbol{\psi}(\widehat{\mathbf{x}})$ are the student basis functions (e.g. $\boldsymbol{\psi}(\widehat{\mathbf{x}}) = \tanh(\mathbf{M}_S^\top\widehat{\mathbf{x}} + \boldsymbol{\theta}_S)$), $\mathbf{e} = \mathbf{x} - \widehat{\mathbf{x}}$ is the tracking error, and k is the feedback gain. The goal of learning is to adapt the parameters $\mathbf{W}$ such that the tracking error is minimized. In the initial phases of learning, the feedback gain k is usually set to a large value, which causes the student to follow the desired teaching dynamics closely. In turn, the student variable $\widehat{\mathbf{x}}$ already traverses the state space correctly which provides the opportunity for adjusting the adaptive parameters $\mathbf{W}$ in order to learn the dynamics. Once these parameters have been learned (at least approximately), the feedback gain is decreased, which also helps to get good generalization behavior and reduce the number of training iterations. Once the system is learned, the feedback gain can be set to zero, which is what we did in the test phases ($k_{\text{test}} = 0$).

If the student has sufficiently many rich basis functions to

approximate the function $f(.)$, then there exists a solution called $\mathbf{W}^{\text{true}}$. If we define a Lyapunov function as

$$V = \frac{1}{2}\mathbf{e}^\top\mathbf{e} + \frac{1}{2\eta}\,\mathrm{Tr}\left(\widetilde{\mathbf{W}}^\top\widetilde{\mathbf{W}}\right), \tag{3}$$

where $\widetilde{\mathbf{W}} = \mathbf{W}^{\text{true}} - \mathbf{W}$ is the estimation error and $\mathrm{Tr}(.)$ is the matrix trace operator, it can be shown (see Supplementary Materials) that the following adaptation law (or the learning rule)

$$\dot{\mathbf{W}} = \eta\,\boldsymbol{\psi}(\widehat{\mathbf{x}})\mathbf{e}^\top, \tag{4}$$

where η is a learning rate, will decrease the Lyapunov function, i.e. $\dot{V} = -(k+1)\,\mathbf{e}^\top\mathbf{e} \leq 0$. This result together with boundedness conditions for V and $\ddot{V}$ guarantees that $\dot{V}$, and hence the tracking error $\mathbf{e}$, will asymptotically go to zero and that the system is asymptotically stable (Slotine and Li 1991). Moreover, if the input $\mathbf{c}(t)$ is rich enough, then $\mathbf{W} \to \mathbf{W}^{\text{true}}$.

Learning a functional spiking network

We want to translate our student into a recurrent network of N leaky integrate-and-fire (LIF) neurons. Previous work has shown how to implement arbitrary linear or nonlinear dynamical systems in efficient balanced networks (Boerlin, Machens, and Deneve 2013; Thalmeier et al. 2016). EBN theory is based on two assumptions. First, an estimate of the K-dimensional state variable, $\mathbf{x}$, can be extracted from the filtered spike trains, $\mathbf{r}$, using a linear decoder, such that

$$\widehat{\mathbf{x}} = \mathbf{D}\mathbf{r}, \tag{5}$$

where $\mathbf{D}$ is a fixed decoding weight matrix of size $K \times N$ and the relation between $\mathbf{r}$ and the spike trains $\mathbf{s}$ is $\dot{\mathbf{r}} = -\lambda\mathbf{r} + \mathbf{s}$. Filtered spike trains are the convolution of the spike trains with a synaptic kernel and can be interpreted as instantaneous firing rate. These slower variables than the spike trains are used for dynamic computation (and readout of the implemented dynamics) on a different time scale than the fast one, which is used for making the system have a robust representation and a tight E-I balance.

Second, neurons in the network fire spikes such that this estimate, $\widehat{\mathbf{x}}$, closely follows the true state variable, $\mathbf{x}$, under cost constraints. Specifically, the network minimizes the following objective function,

$$\mathcal{L} = \langle L\rangle = \left\langle \|\mathbf{x} - \widehat{\mathbf{x}}\|^2 + \mu\|\mathbf{r}\|_2^2 + \nu\|\mathbf{r}\|_1 \right\rangle. \tag{6}$$

where $\|\cdot\|_p$ denotes the L_p-norm, $\langle\cdot\rangle$ is an average over time, and μ and ν determine the costs associated with spiking. The first term on the right-hand side of Eq. 6 ensures a good reconstruction, whereas the L2 and L1 cost functions on neural activity ensure a distributed and sparse spiking activity, respectively (Boerlin, Machens, and Deneve 2013).

The resulting network consists of LIF neurons whose membrane potentials, u_i's, obey the equation (Boerlin, Machens, and Deneve 2013; Bourdoukan et al. 2012) (see also Supplementary Materials)

$$u_i = \mathbf{D}_i^\top(\mathbf{x} - \widehat{\mathbf{x}}) - \mu r_i, \tag{7}$$

and whose firing thresholds are given by $T_i = \frac{1}{2}(\|\mathbf{D}_i\|_2^2 + \mu + \nu)$, where $\mathbf{D}_i$ is the i-th column of $\mathbf{D}$. The thresholds ensure that each neuron fires only when its spike decreases the objective function (Eq. 6). Differentiating Eq. 7 and simplifying it yields the following membrane potential dynamics

$$\dot{\mathbf{u}} = -\lambda\mathbf{u} + \mathbf{D}^\top\mathbf{c} - (\mathbf{D}^\top\mathbf{D} + \mu\mathbf{I})\mathbf{s} + \mathbf{D}^\top(\lambda\mathbf{x} + f(\mathbf{x})), \tag{8}$$

which shows the optimal weight matrix for the fast connections is $\mathbf{W}^{\text{fast}} = \mathbf{D}^\top\mathbf{D} + \mu\mathbf{I}$, where $\mathbf{I}$ is the identity matrix, and the optimal encoding weight matrix is $\mathbf{D}^\top$. In order to implement arbitrary dynamical systems in this framework, we need to include two approximations. First, we can approximate $\lambda\mathbf{x} + f(\mathbf{x})$ as a sum of the "student" basis functions (similar to Eq. 2), so that $\lambda\mathbf{x} + f(\mathbf{x}) = \sum_i \mathbf{W}_i\phi_i(\mathbf{x})$, where $\mathbf{W}_i$'s are the columns of $\mathbf{W}$, and $\phi(.)$ can be any sigmoidal nonlinearity set to $\tanh(.)$ in the simulations. Second, we can self-consistently replace the state $\mathbf{x}$ by its estimate based on network activity (Eq. 5). In turn, we obtain (Fig. 2B)

$$\dot{\mathbf{u}} = -\lambda\mathbf{u} + \mathbf{D}^\top\mathbf{c} - \mathbf{W}^{\text{fast}}\mathbf{s} + \mathbf{D}^\top\mathbf{W}^\top\boldsymbol{\phi}(\mathbf{D}\mathbf{r}). \tag{9}$$

From a biological point-of-view, the last term corresponds to nonlinear and highly structured dendrites. There is evidence that dendrites can perform nonlinear operations (Poirazi, Brannon, and Mel 2003a; 2003b). While dendrites are often nonlinear, the required detailed spatial organization of their inputs is rather speculative. To relax this latter constraint, we can take advantage of the fact that the network's internal state dynamics is extremely robust to large amounts of noise (as ensured by its constant, greedy minimization of its own coding errors), as long as $N > 2K$. Thus, we can replace $\boldsymbol{\phi}(\mathbf{D}\mathbf{r})$ with an unstructured, nonlinear dendrite that receives random connections from other neurons, $\Psi_i(\mathbf{r}) = \boldsymbol{\phi}(\mathbf{M}_i^\top\mathbf{r} + \boldsymbol{\theta}_i)$ where $\mathbf{M}_i$'s are the columns of the matrix $\mathbf{M}$. Indeed, the quantity $\mathbf{M}^\top\mathbf{r}$ can be written as the sum of its projection onto the decoder $\mathbf{D}$, and its projection onto the null space of $\mathbf{D}$ (denoted by $\mathbf{D}^\emptyset$), so that $\mathbf{M}^\top\mathbf{r} = \mathcal{M}^\top\widehat{\mathbf{x}} + \mathcal{M}^\top\mathbf{D}^\emptyset\mathbf{r}$. Given that EBNs do not control spiking in the null space of the decoder, the second term acts as unstructured noise. [2]

More concretely, now we have a network in which the axon of a neuron makes contact with different parts of dendritic trees of other neurons. This dendritic tree structure provides nonlinearities (any sigmoidal functional form with random slope and threshold) instead of just linearly summing the inputs. This nonlinear operation in the dendrites is utilized by the network as basis functions in order to approximate the unknown function f(.). The details of nonlinear operation in dendritic trees are not clearly understood to our knowledge. Therefore, we used random values for the parameters of dendritic nonlinearity operation ($\mathbf{M}$ and $\boldsymbol{\theta}$).

[2] We note that it would be possible to train the parameters $\mathbf{M}$ using unsupervised learning, since $\mathbf{D}^\top$ corresponds to the eigenvectors with non-zero eigenvalues in the neural correlation matrix. This should render the network even more efficient and robust as will be explored in the future.

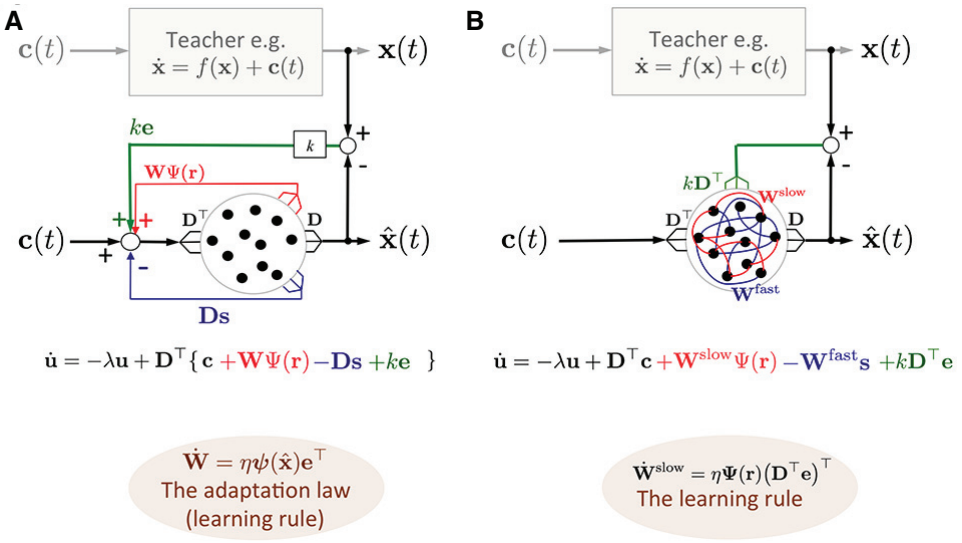

Figure 2: Building a spiking network approximating arbitrary dynamical systems. A. The unfolded version of the network of LIF neurons (with membrane potentials $\mathbf{u}$ and a threshold and reset mechanism) with three feedback loops implementing desired dynamics of $\mathbf{x}$ in the network: the role of fast (red) loop is to implement nonlinear dynamical system, the slow (blue) one provides efficiency and robustness, and the feedback (green) one feeds the error back into the network such that $\hat{\mathbf{x}}$ closely tracks $\mathbf{x}$. B. The folded version of the network with slow and fast connections. The corresponding learning rules are shown at the bottom.

To clarify the learning problem, we will write the resulting student network dynamics in the general form

$$\dot{\mathbf{u}} = -\lambda\mathbf{u} + \mathbf{Fc} - \mathbf{W}^{\text{fast}}\mathbf{s} + \mathbf{W}^{\text{slow}}\mathbf{\Psi}(\mathbf{r}). \quad (10)$$

We note that the network has three sets of synaptic connections. The feedforward connections, $\mathbf{F}$, receive and weight the external signal input, $\mathbf{c}$. The fast connections, $\mathbf{W}^{\text{fast}}$, guarantee the proper and efficient distribution of spikes across the network. Finally, the slow connections, $\mathbf{W}^{\text{slow}}$, implement the dynamics of the student system. In previous work, it is shown how to learn the feedforward connections (Brendel et al. 2017) and the fast recurrent connections (Brendel et al. 2017; Bourdoukan et al. 2012). Though all connections could be trained simultaneously, we here concentrate on how to train the slow connections based on example trajectories from an unknown (teacher) dynamical system. We use a fixed random decoding weight matrix $\mathbf{D}$ (which is close to optimal in the case of uncorrelated command signals) and set the feedforward connections and recurrent connections to their optimal values, $\mathbf{F} = \mathbf{D}^\top$ and $\mathbf{W}^{\text{fast}} = \mathbf{D}^\top\mathbf{D} + \mu\mathbf{I}$.

A direct translation of the adaptive control to the neural network will finally permit us to define a local learning rule for the slow connections. Rather than feeding back the error into the student network by adding it to its input, we directly inject the errors as feedback to each neuron with connections $\mathbf{D}^\top$, which is mathematically equivalent. In the presence of this feedback control loop, the network equation becomes

$$\dot{\mathbf{u}} = -\lambda\mathbf{u} + \mathbf{Fc} - \mathbf{W}^{\text{fast}}\mathbf{s} + \mathbf{W}^{\text{slow}}\mathbf{\Psi}(\mathbf{r}) + k\mathbf{D}^\top\mathbf{e}. \quad (11)$$

Moreover $\mathbf{W}^{\text{slow}} = \mathbf{D}^\top\mathbf{W}^\top$ is directly related to the coefficients $\mathbf{W}$ of the basis functions in the control theory framework. This allows us to map the adaptation law (Eq. 4) for the coefficients of the basis function to the following learning rule for the slow connections. Replacing $\mathbf{e}$ in Eq. 4 with $\mathbf{D}^\top\mathbf{e}$, we obtain

$$\dot{\mathbf{W}}^{\text{slow}} = \eta\,\mathbf{\Psi}(\mathbf{r})(\mathbf{D}^\top\mathbf{e})^\top. \quad (12)$$

In other words, the resulting learning rule is the product of a pre-synaptic input (passed through the dendritic nonlinearity) and the projection of error feedback. Hence, the learning is local.

For the case of sensorimotor learning, the quantity $k\mathbf{e}$ could correspond to visual or somatosensory "prediction errors", e.g. the difference between the sensory input and its prediction based on the efference copy of the motor commands. Over the course of learning, the errors become smaller and would eventually vanish in the absence of motor noise or sensory noise. Consequently, the feedback would become silent and could be removed entirely.

The general idea of this derivation, and the mapping from the control-theoretical framework onto efficient balanced networks, is also illustrated in Fig. 2A.

Implementing nonlinear dynamics

We employed the framework described above in the following example tasks.

Bistable attractor

The bistable attractor is an important nonlinear dynamical system which is widely studied in systems neuroscience. We implemented the following one-dimensional attractor dynamics

$$\dot{x} = x(0.5 - x)(0.5 + x) + c(t), \quad (13)$$

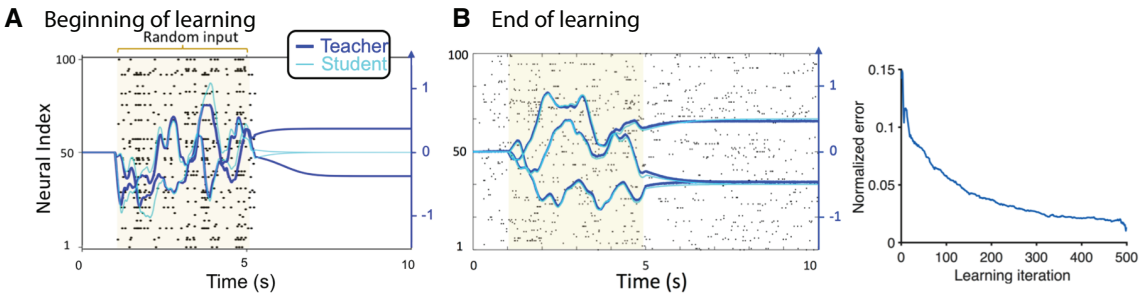

Figure 3: Result of learning a bistable attractor in an spiking network. The desired dynamics given by a teacher is $\dot{x} = x(0.5 - x)(0.5 + x) + c(t)$ where $c(t)$ is a random input signal. A. In the beginning of learning, the reconstructed dynamics of the student network (thin cyan trace) during the presentation of the input (shaded area) is due to the random inputs (two trials shown). After the input is turned off, the reconstruction does not go to any of the attractor states. B. At the end of learning, when the input is off, the reconstructed signal falls into one of the stable attractors (three trials). C. Learning curve showing the smoothed training error after 500 learning iterations.

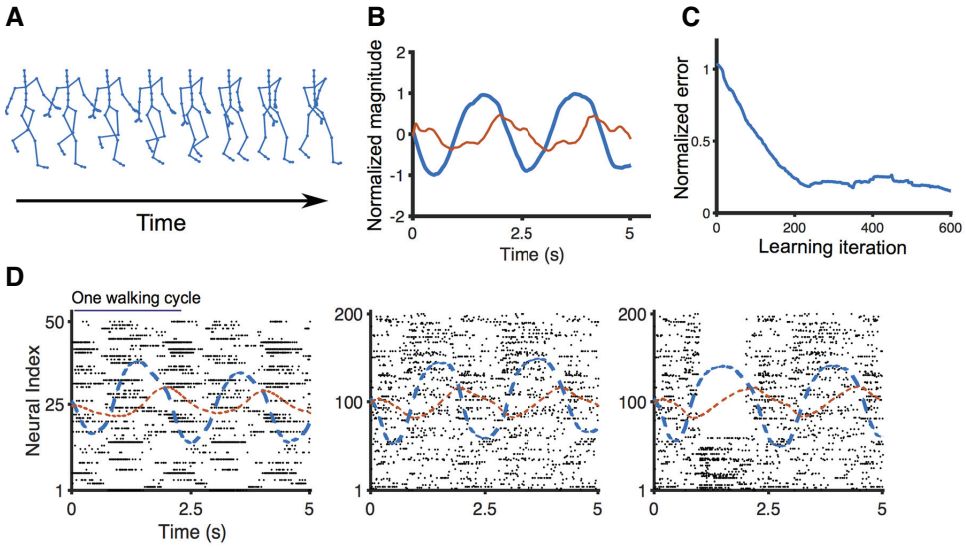

Figure 4: Result of learning to walk in spiking networks. A. The dynamics of the walk of a sticky man implemented by a network of spiking model neurons. The training data is from CMU Motion Caption Library. B. Two first principal components of the desired trajectory of the walk. The original dataset for the walk has 62 channels. C. Learning curve showing the smoothed normalized error during training as a function of learning iterations. D. Raster plots for networks that learned a walking dynamics based on four principal components of a trajectory of the walk. The left panel shows the plot for a network of 50 neurons. Four principal components were provided to the network as desired trajectories. The reconstruction of the first two principal components are shown overlaid on the raster plot. As the number of neurons in the network increases to 200 neurons (middle panel) the firing rates decrease and spiking activity becomes more asynchronous and irregular and the reconstruction improves. The network is drastically robust to silencing neurons as shown in the right-most panel where 70% of the neurons are silenced for a period of around one second but the network continues to generate walking with a short, negligible distortion. Note that after silencing the neurons, the network weights do not undergo any learning, but other active neurons compensate for neuronal loss in the network.

where $c(t)$ is a random input command. This system has two stable fixed-point attractors at $x = \pm 0.5$ and a saddle-point at $x = 0$. We used $N = 50$ neurons with random readout weights $\mathbf{D}$ to implement this system. As shown in Fig. 3A, the random input $c(t)$ drives the student network in the beginning of learning. Once this input is zero the desired dynamics given by the teacher (in thick blue trace) goes to one of the stable attractors. But the decoded dynamical variable of the network (in cyan) has not yet learned the desired attractors. After learning (Fig 3B), the network closely follows the desired trajectories in response to a random unseen input (not used during training) and, more importantly, once the

input is turned off, it settles into the correct attractors $\%90$ of the times. It took around 500 iterations to learn the bistable attractor as the learning curve shows the smoothed normalized training error as a function of learning iterations in Fig 3C. Note that the network exhibits realistic spiking activity with low firing rates and asynchronous, irregular spiking activity. The overall number of spikes is drastically lower than that of spiking networks that effectively use *rate codes* (Eliasmith and Anderson 2004; Eliasmith 2005; Gilra and Gerstner 2017).

Motion capture example

We also implemented the walking dynamics from the database of Carnegie Mellon University Motion Capture Library (MOCAP) http://mocap.cs.cmu.edu/. We used a similar preprocessing procedure as in (Sussillo and Abbott 2009). Briefly, the data was taken from file "08_01.amc" which consisted of 62 channels out of which three were set to zero, converting the movement to a walk on a treadmill. We preprocessed the data by a moving average.

In order to reduce the dimensions of the input signal, we took the first four principle components. This reduced input closely follows the original walking dynamics with minor loss. As the system is a mechanical system, and therefore of second order, we require both velocity and position information. Therefore, in order to be able to model it, we fed both the trajectories and its derivatives to the network, so that the overall input was 8-dimensional. Similarly, we provided a linear combination of the position error and the velocity error as the error feedback to the network. We then learned the dynamics of the 8-dimensional input with two different networks, one of size $N = 50$ and and one of size $N = 200$ (compare with (Sussillo and Abbott 2009)).

Fig. 4A shows the learned walking and Fig. 4B shows the first two principle components of the full 62-channel dynamics.The left and middle panels of Fig. 4D show the raster plot of the 50-neuron and 200-neuron networks, respectively. The learning curve in Fig. 4C shows the smoothed normalized training error as a function of learning iterations. The normalized test error reached to 0.16 ± 0.08 in the walking example after ~ 500 learning iterations with 200 neurons. An important feature of our model is its expansion: the number of input channels (K) must be smaller than the number of neurons (N) in the network. In order for the network to have the desired aforementioned properties, as a rule of thumb, the expansion ratio ($\Lambda = N/K$) should be larger than 5–10. This expansion provides many possible solutions for the network to perform a task. Thanks to the efficiency principle, the network is able to choose the most efficient one. It should be noted that the realistic spiking activity is closely linked to the expansion: increasing the expansion ratio results in lower firing rate and more irregular spiking activity (compare left and right panels of Fig. 4D). This is one of the main differences with other spiking networks (but in fact they use a *rate code*) implementing complex dynamics such as the Neural Engineering Framework (NEF) (Eliasmith and Anderson 2004; Eliasmith 2005), as they would typically need firing rates in the order of the inverse of synaptic time-scales to learn properly. Another striking feature of the model is its robustness to neural death and noise. Thanks to the presence of the fast connections, even if $\%70$ of the neurons in the network are silenced the network would still be able to perform the task with minimal loss in the performance (right-most panel in Fig. 4D and videos in Supplementary Materials). This is consistent with the expected robustness of the EBN framework (Barrett, Deneve, and Machens 2016).

Discussion

We have proposed a local learning rule in an spiking neural network of LIF neurons for learning arbitrary complex dynamics. The resulting networks exhibit low firing rates with asynchronous, irregular spiking activity where the spiking representation is as efficient as possible. The efficiency principle that we have exploited has direct consequences: the inhibitory input currents in each neuron closely track the excitatory input (E-I balance); and the network is highly robust to noise, neural elimination, and uncertainty in desired dynamics (Boerlin, Machens, and Deneve 2013; Barrett, Deneve, and Machens 2016). The learning rule is obtained thanks to concepts and tools in nonlinear adaptive control theory. This approach has the benefit of providing a systematic way to study convergence and stability properties of the learning process which currently is not pursued in the mainstream learning procedures in neuroscience for spiking neurons (Sussillo and Abbott 2009; Abbott, DePasquale, and Memmesheimer 2016; Thalmeier et al. 2016). Studying the effect of synaptic delays needs to be addressed in future versions of our model, although previous work suggests that increasing the amount of noise may help avoiding oscillation and synchronization (Chalk, Gutkin, and Deneve 2016). EBNs can be implemented in non-spiking networks (see e.g. (Vertechi, Brendel, and Machens 2014)), therefore our framework should also be implemented with rate-based neurons as well.

Spiking efficiency in our framework can be translated to computational efficiency if proper neuromorphic hardware is utilized for implementing this framework. The neuromorphic community has recently started shown interest for emulating efficient balanced networks in silicon (Boahen 2017) and it would be interesting to compare computational efficiency of EBNs with their rivals in a neuromorphic setup (Voelker et al. 2017; Voelker and Eliasmith 2017). Furthermore, the fast inhibition in neuromorphic hardware can easily be achieved so as to provide the required spiking efficiency in EBNs.

The learning rule can learn any complex dynamics $f(.)$. However, it may not learn the target dynamics if the input is not sufficently rich (Slotine and Li 1991). Mathematically, sufficient richness means that $\exists \alpha > 0, \exists t_0 > 0, \exists T > 0, \forall t$ such that the following matrix quantity $\mathbf{Q}$ be positive semi-definite: $\mathbf{Q} = 1/T \int_t^{t+T} \mathbf{\Psi}\mathbf{\Psi}^\top - \alpha \mathbf{I}$, where $\mathbf{I}$ is the identity matrix and $\mathbf{\Psi}\mathbf{\Psi}^\top$ is the outer product of the basis function outputs.

If function $f(.)$ has singularities that make the dynamics unstable, the learning may fail. Another requirement is that the states (for example, position and velocity in the motion

capture example) need to be well-defined. Any task that can be cast as a well-behaved dynamical system can be learned in the current framework.

The different parts of the spiking network have straightforward biological interpretations. Fast connections could correspond to interneurons which would mostly be driven by monosynaptic, fast AMPA synapses from excitatory neurons, targeting the soma and relying on ionic (GABA-A) neurotransmission. Slow connections could be implemented by slower metabolic channels (e.g. NMDA/GABA-B) and correspond either to direct connections between pyramidal cells, or disynaptic inhibition using another type of interneurons, in the case of negative weights.

The network seems to be highly structured in the current framework. However, it has already been shown that most of the other connections in the network can be trained using local spike-time-dependent plasticity rules with the exception of $\mathbf{W}^{\text{slow}}$ which is the contribution of the present work. In particular, the fast connections $\mathbf{W}^{\text{fast}}$ can be trained using local plasticity rules, while the feedforward connections (and thus, the decoding weights) can be trained using a Hebbian spike-time-dependent rule (Brendel et al. 2017; Bourdoukan et al. 2012) . Note that these inhibitory fast connections could be thought of as implementing a kind of 'dynamic' Winner-Take-All (WTA) network where once a neuron wins and spikes, then other neurons have a chance to fire as well. This is a different scheme from a distributed WTA of inhibitory neurons (Wang and Slotine 2006).

In our general framework, nonlinear dendrites compute the basis functions where we used $\tanh(.)$ nonlinearity. However, the learning rule is not limited to that and can work with any basis functions (nonlinearities) that can approximate arbitrary functions e.g. Gaussian, sigmoid basis functions, though sigmoids are more biologically plausible. Apart from the form of nonlinearity, we have a large degree of freedom in the choice of parameters of basis functions $\mathbf{M}$ i.e., in the design of the dendritic tree. The case that might be easiest to map to the control-theoretic framework is the case where $\mathbf{M} = \mathbf{D}\mathbf{M}'$ is a low rank matrix. But other cases such as random $\mathbf{M}$ also work in practice. We provided intuitions for why such a suboptimal architecture still functions: the network activity is forced to be as efficient as possible due to the presence of the fast connections (E/I balance). This defines an optimal combination of neural activities for each state $\hat{\mathbf{x}}$. In other words, neural firing rates become almost completely determined by the internal state estimate (up to their "Poisson-like" variability). This neural activity projected onto an arbitrary matrix will also be a function of $\hat{\mathbf{x}}$, plus some unstructured input that may be considered as noise. This noise is automatically compensated by the network robustness. In the case of the walking example, we tested a diagonal matrix $\mathbf{M}$ (in which case, the dendritic nonlinearity is replaced by a nonlinear transfer function for each neuron) and the network was still able to learn without any difficulty. The best choice of $\mathbf{M}$ (including the possibility of learning these parameters) remains to be explored further elsewhere. Our approach is aligned with other works taking similar strategies for implementing nonlinearities in recurrent networks (Abbott, DePasquale, and Memmesheimer 2016; Thalmeier et al. 2016).

Nonlinear adaptive control theory yields a local learning rule for the Neural Engineering Framework (Gilra and Gerstner 2017) to learn complex dynamics. However, the resulting networks do not implement fast connections to provide spiking efficiency and balance. This in turn implies that they do not scale easily when applied to large dynamics (Sussillo and Abbott 2009; Gilra and Gerstner 2017), making it more challenging for these networks to exhibit realistic spiking activity and robustness. Our work aims to seamlessly blend nonlinear adaptive control with earlier work on EBNs, which exploits the spiking nature of neural activity rather than treating it as a hindrance.

In conclusion, we argue for a close relationship between spiking efficiency, robustness and the tight E-I balance observed in cortical circuits. We suggest that experimentally observed spike trains, with low-firing rate and asynchronous, irregular spike trains, are a signature of an efficient spike-based coding, and not a noisy rate-based population code. The network effectively implements dimensionality reduction: regardless of its size, the dimensionality of its population dynamics and recurrent weights eventually becomes restricted to the dimensionality of the task, while neural fluctuations occur in direction orthogonal to the task. Thus, we predict that such low-dimensional dynamics emerge through experience in biological neural circuits. Finally, learning in biological circuits would require that feedback connections monitoring the network performance both drive the neurons and modulate learning. Each slow connection is learned as a function of the correlation between presynaptic input rate and postsynaptic error feedback, until this error feedback is canceled. Thus, only neurons with correlations to the error (and presumably contributing to such error) see their synaptic weights change. In contrast, backpropagation would result in diffuse change in the entire network. These predictions have broad implications for Brain Machine Interfaces. In the near future, this theory may pave the way for implementing more complex tasks and for spike based unsupervised, hierarchical and reinforcement learning, in both biological and artificial spiking networks.

The framework presented here can have engineering applications for example in light-weight robots or robots that are sent to space missions where efficiency matters. Furthermore, we have used the framework to estimate the parameters of a desired system but it can also be used for adaptive nonlinear control applications — where the controller needs to adapt to unknown changes in dynamics and kinematics of robots (Cheah, Liu, and Slotine 2006; DeWolf et al. 2016) — to give an efficient adaptive spiking controller. The current framework is obtained for deterministic dynamical systems – future work will extend it to stochastic dynamical systems.

Acknowledgments

A.A. and S.D. acknowledge funding from Agence Nationale de la Recherche (ANR) grant ANR-10-LABX-0087 IEC and ANR-10-IDEX-0001-02 PSL, European Research Council

(ERC) grant “Predispike”. S.D. was also supported by the James McDonnell Foundation award “Human Cognition”.

References

Abbott, L. F.; DePasquale, B.; and Memmesheimer, R.-M. 2016. Building functional networks of spiking model neurons. *Nature Publishing Group* 19(3):350–355.

Barrett, D.; Deneve, S.; and Machens, C. K. 2016. Optimal compensation for neuron loss. *eLife* 5:e12454.

Boahen, K. 2017. A neuromorph’s prospectus. *Computing in Science & Engineering* 19(2):14–28.

Boerlin, M.; Machens, C. K.; and Deneve, S. 2013. Predictive Coding of Dynamical Variables in Balanced Spiking Networks. *PLoS computational biology* 9(11):e1003258.

Bourdoukan, R., and Deneve, S. 2015. Enforcing balance allows local supervised learning in spiking recurrent networks. In *Advances in Neural Information Processing Systems (NIPS)*, 982–990.

Bourdoukan, R.; Barrett, D.; Machens, C. K.; and Deneve, S. 2012. Learning optimal spike-based representations. *Advances in Neural Information Processing Systems (NIPS)*.

Brendel, W.; Bourdoukan, R.; Vertechi, P.; Machens, C. K.; and Deneve, S. 2017. Learning to represent signals spike by spike. *arXiv.org:1703.03777*.

Chalk, M.; Gutkin, B.; and Deneve, S. 2016. Neural oscillations as a signature of efficient coding in the presence of synaptic delays. *eLife*.

Cheah, C. C.; Liu, C.; and Slotine, J. 2006. Adaptive tracking control for robots with unknown kinematic and dynamic properties. *The International Journal of Robotics Research* 25(3):283–296.

Deneve, S., and Machens, C. K. 2016. Efficient codes and balanced networks. *Nature Neuroscience*.

DeWolf, T.; Stewart, T. C.; Slotine, J.-J.; and Eliasmith, C. 2016. A spiking neural model of adaptive arm control. 283(1843):20162134.

Eliasmith, C., and Anderson, C. H. 2004. *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT press.

Eliasmith, C. 2005. A unified approach to building and controlling spiking attractor networks. 1–39.

Gilra, A., and Gerstner, W. 2017. Predicting non-linear dynamics: a stable local learning scheme for recurrent spiking neural networks. *arXiv preprint arXiv:1702.06463*.

Laughlin, S. B.; de Ruyter van Steveninck, R. R.; and Anderson, J. C. 1998. The metabolic cost of neural information. *Nature Neuroscience* 1(1):36–41.

Memmesheimer, R.-M.; Rubin, R.; Ölveczky, B. P.; and Sompolinsky, H. 2014. Learning Precisely Timed Spikes. *Neuron* 82(4):925–938.

Poirazi, P.; Brannon, T.; and Mel, B. W. 2003a. Arithmetic of subthreshold synaptic summation in a model CA1 pyramidal cell. *Neuron* 37(6):977–987.

Poirazi, P.; Brannon, T.; and Mel, B. W. 2003b. Pyramidal neuron as two-layer neural network. *Neuron* 37(6):989–999.

Sanner, R. M., and Slotine, J.-J. 1992. Gaussian networks for direct adaptive control. *IEEE Transactions on neural networks* 3(6):837–863.

Schuman, C. D.; Potok, T. E.; Patton, R. M.; Birdwell, J. D.; Dean, M. E.; Rose, G. S.; and Plank, J. S. 2017. A survey of neuromorphic computing and neural networks in hardware. *arXiv preprint arXiv:1705.06963*.

Slotine, J.-J. E., and Coetsee, J. A. 1986. Adaptive sliding controller synthesis for non-linear systems. *International Journal of Control* 43(6):1631–1651.

Slotine, J.-J. E., and Li, W. 1991. *Applied Nonlinear Control*. Prentice Hall.

Sussillo, D., and Abbott, L. F. 2009. Generating Coherent Patterns of Activity from Chaotic Neural Networks. *Neuron* 63(4):544–557.

Thalmeier, D.; Uhlmann, M.; Kappen, H. J.; and Memmesheimer, R.-M. 2016. Learning Universal Computations with Spikes. *PLoS computational biology* 12(6):e1004895.

Vertechi, P.; Brendel, W.; and Machens, C. K. 2014. Unsupervised learning of an efficient short-term memory network. *Advances in Neural Information Processing Systems (NIPS)*.

Voelker, A. R., and Eliasmith, C. 2017. Methods for applying the neural engineering framework to neuromorphic hardware. *arXiv preprint arXiv:1708.08133*.

Voelker, A. R.; Benjamin, B. V.; Stewart, T. C.; Boahen, K.; and Eliasmith, C. 2017. Extending the neural engineering framework for nonideal silicon synapses. In *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*, 1–4. IEEE.

Wang, W., and Slotine, J.-J. E. 2006. Fast computation with neural oscillators. *Neurocomputing* 69(16-18):2320–2326.

Werbos, P. J. 1990. Backpropagation through time: what it does and how to do it. In *Proceedings of the IEEE*.

Wolpert, D. M., and Ghahramani, Z. 2000. Computational principles of movement neuroscience. *Nature Neuroscience*.