Reinforcable information maximization with local growth criteria in recurrent networks

Sensen Liu[1] and ShiNung Ching[1,2]

***Abstract*—We consider the design of recurrent networks that can achieve both unsupervised and goal-oriented modes of learning. Of particular interest is how such learning might be realized through local dynamics, without each neuron relying on global awareness of the entire recurrent network state. To this end, we bring together notions of reinforcement learning (RL) and information maximization (Infomax) to construct a local, dynamic synaptic learning rule that has versatility in achieving different learning regimes.**

Our model follows from our prior work, where we developed a learning algorithm that optimizes information retention in a recurrent network. Here, we introduce an augmented version of this learning rule by employing a reinforcement mechanism that can modulate the alternations between Hebbian and anti-Hebbian learning regimes on the basis of a prescribed objective function. In parallel, we consider not only plasticity at the level of synaptic weights, but also the issue of network growth, i.e., adding neurons on-the-fly, or 'neurogenesis'. Here, the idea is that learning arises not only through synaptic modification, but also from activity-dependent construction and elimination of neurons and synapses, thus obviating the need to specify network size *a priori*. Our result in this regard is to supplement the learning rule with additional local dynamics that enables both growth and pruning based on the contribution of each neuron to the overall Infomax objective. By using this strategy, learning and neurogenisis can occur side-by-side in a scalable way. Examples of the proposed framework are provided that highlight the ability of the proposed learning rule to handle both extrinsic rewards and unsupervised coding of input information.

***Index Terms*—recurrent neural network, information maximization , reinforcement learning.**

I. Introduction

Information maximization, or Infomax, is a popular framework in the study of neural coding, a form of unsupervised learning involving the representation of stimuli by neurons [1], [2]. It has been shown that networks built under Infomax objectives (nominally, the maximization of Shannon mutual information) exhibit learning dynamics and emergent activity patterns that are consistent with those observed in biology [3]. Recently, the Infomax framework has been extended to the study of recurrent networks, wherein the objective is to represent *and retain* information about a stimulus [4]. However, recurrent networks present a conceptual hurdle for Infomax, since optimization requires manipulation of the joint distribution of activity over the entire network state space [4]. Thus, any given neuron must be able to 'know' the state of every other neuron, a biologically problematic assumption. Setting aside biological interpretability, tracking a joint distribution of activity over a recurrent network is challenging computationally due to issues of scalability. In our prior work [5], we developed a learning strategy for recurrent Infomax that obviates this issue by endowing each neuron with a set of dynamical variables, evolving at multiple time-scales that allow for local estimation of global (whole-network) information.

In this brief paper, we present two novel extensions of these prior learning dynamics. First, we consider how the unsupervised Infomax framework might coexist with learning based on an extrinsic reward function. In this regard, we show how a reinforcement learning mechanism can be integrated into the Infomax learning policy, leading to a network that can fluidly alternate with unsupervised and reward-based regimes. Such a network is able to not only encode stimuli but use the salient information to engage in tasks. Second, we treat the problem of how to appropriately choose the size of a network based on stimulus and/or task characteristics. While such a question has been treated in the literature, most paradigms are based on examination of correlation between neurons (as a measure of redundancy) [6] and have been mostly applied to feedforward network architectures. In contrast, we show how networks can be constructed (and pruned) 'on-the-fly,' based only on locally held dynamical variables that are overtly related to the Infomax cost. In other words, each neuron is able to split or prune itself, based on a local criterion.

Thus, the major contributions of this paper are: (i) the development of a local, dynamical synaptic learning rule that incorporates a reinforcement mechanism within an already information-optimal paradigm, in a recurrent, stochastic spiking network, (ii) the derivation of dynamics that enable online network growth/pruning that fluidly integrated with the Infomax cost and, (iii) the illustration of the learning rule through prototypical examples. These results are significant because they show how the use of dynamical variables over multiple time-scales within Infomax can allow for flexible and scalable modes of learning.

The remainder of the paper is organized as follows. In Section II we provide the basic network and learning framework and review prior results. In Section III we provide the main results, developing first the reinforcable Infomax learning rule, and then the policy for local network growth and pruning. Examples are provided to illustrate the efficacy of the proposed framework, and Discussion and Conclusions are presented in Section IV.

II. Formulation and Preliminary Results

A. Recurrent Information Maximization

We consider a discrete, network of N stochastic McCulloch-Pitts type neurons. Specifically, we denote $\mathbf{x}^t = [x_1^t, x_2^t, ...x_N^t]^T \in \Re^N$ as the state vector of the whole network, and $x_i^t = \{0, 1\}$ characterizes the absence/presence of a spike at time t for neuron i. The variable x_i^t is obtained as $P(x_i^t = 1) = g(v_i^t)$, where v_i^t is an underlying state variable that aggregates input from pre-synaptic neurons via $v_i^t = \sum_{j=1}^N w_{ij} x_j^{t-1} + u_i^t$, and $g(\cdot)$ is a sigmoidal function, w_{ij} is the connection strength between neurons i and j, and u_i^t is an extrinsic input.

In [5] we considered the problem of optimization of recurrent Shannon mutual information in this network, i.e.,

$$\max_{\mathbf{w}} MI(\mathbf{x}^t; X_1^{t-1}), \tag{1}$$

S. Ching holds a Career Award at the Scientific Interface from the Burroughs-Wellcome Fund. This work was partially supported by AFOSR 15RT0189, NSF ECCS 1509342, NSF CMMI 1537015 and NSF CMMI 1653589, from the US Air Force Office of Scientific Research and the US National Science Foundation, respectively.

[1] S. Liu and S. Ching are with the Department of Electrical and Systems Engineering, Washington University in St. Louis, St. Louis, MO, USA

[2] S. Ching is with the Department of Biomedical Engineering and the Division of Biology and Biomedical Sciences, Washington University in St. Louis, St. Louis, MO, USA

where $\mathbf{w}$ denotes the network connectivity weights and

$$MI(\mathbf{x}^t; X_1^{t-1}) = \sum_{\mathbf{x}^t, X_1^{t-1}} P(\mathbf{x}^t, X_1^{t-1}) \log \frac{P(\mathbf{x}^t | X_1^{t-1})}{P(\mathbf{x}^t)}, \quad (2)$$

that is, the mutual information between the current state of the network and its history. Here, the matrix $X_1^{t-1} \in \Re^{N \times (t-1-1)}$ represents the history of the network up to time step $t-1$. The underlying idea behind this formulation is that such optimization may allow the network to code prior information about afferent inputs, thus enabling time-dependent computations.

B. A local dynamical plasticity rule for global, recurrent objectives

Due to the recurrence, the optimization of (2) with respect to the network synaptic weights involves global dependency, wherein the modification of any given weight depends on the entire joint distribution of the network as a whole. Thus, this form of learning is computationally arduous and does not scale gracefully.

However, this global dependence can be overcome if we incorporate local processes at the level of each neuron that serve to estimate global quantities. Specifically, we postulated the learning dynamics:

$$\Delta w_{ij}^t = \gamma \left[\bar{\phi_{ij}}^h(\mathbf{s}(t)) + \bar{\phi_{ij}}^a(\mathbf{s}(t)) \right], \quad (3)$$

where $\bar{\phi_{ij}}^h(t)$ is a Hebbian modification function that promotes co-activation of neurons, while $\bar{\phi_{ij}}^a(t)$ is anti-Hebbian. Here, $\mathbf{s}(t) = (s_{1,i}^t, s_{2,i}^t, s_{3,i}^t, s_{4,ij}^t, s_{5,ij}^t, s_{6,ij}^t)$ contains local dynamical variables that evolve based on a combination of pre- and post-synaptic neuronal activity. Each of these variables can be understood as a recursive estimate of an expectation toward the overall estimate of the joint distribution. Appendices A and B provide details of these variables and their connection to the underlying Infomax problem, while a full exposition can be found in [5].

III. Main Results

A. Combining Infomax with Reinforcement Learning

We showed in [5] that the local learning rule (3) can indeed enable networks to retain information about afferent stimuli over a receding temporal horizon. We sought to further generalize this learning rule by combining it with reinforcement learning (RL), so that networks could not only represent stimuli but also learn to perform tasks through active interaction with an environment.

RL requires feedback from the environment that is used to modulate a network's learning process. If we denote a (time-varying) reward as R^t, then the typical RL objective is to maximize the discounted sum of R^t over a future horizon. We realized that the functional form our of recurrent Infomax rule (3) could be used to incorporate RL into the learning process. In particular, recall that the Hebbian and anti-Hebbian modification functions in the rule serve to strengthen (correlate) or weaken (de-correlate) neurons, respectively. Thus, an extrinsic reward can bias the learning between these two regimes. More specifically, we introduce reinforcement variables $c_h^t, c_a^t \in \Re$, such that the new learning rule is:

$$\Delta w_{ij}^t = \gamma [c_h^t \bar{\phi_{ij}}^h(t) + c_a^t \bar{\phi_{ij}}^a(t)]. \quad (4)$$

The variables c_h^t, c_a^t modulate the weights of Hebbian and anti-Hebbian learning components in the synaptic learning rule and evolve according to:

$$\begin{aligned} \tau_h \Delta c_h^t &= -c_h^t + (\tau_h - 1) R^t; \\ \tau_a \Delta c_a^t &= -c_a^t - (\tau_a - 1) R^t. \end{aligned} \quad (5)$$

Here, $R^t \in \Re$ is a bounded real-valued reward received by the network at time t, based on the current task state. Commensurate with reinforcement, c_h^t is potentiated for larger R^t via (5). Oppositely, the anti-Hebbian term promotes forgetting or correction of actions that lower the reward, i.e., c_a^t is strengthened if R^t gets lower. Nominally, R^t is specified as a function that maps the current task state and prior actions into a quantitative reward. This specification will vary depending on the task in question, and a typical formulation may entail the error between the current task state and some prescribed target state. However, it is important to note that this rule does not require an explicit functional form of R^t.

The dynamics in (5) amount to a filtering operation on the reward R^t, which in turns modulates either the strengthening of weakening of connections within the Hebbian/anti-Hebbian learning framework. The time constants τ_h, τ_a in (5) represent a forgetting factor in the low-pass-filtering dynamics in (5). They determine how much immediate reward R^t is preserved at the current time. Larger time constants lead to actions that benefit more immediate outcomes.

B. Synapse Construction and Pruning based on Information Criteria

In the previous section, we introduced a reward-based recurrent Infomax learning rule. A separate challenge with such networks is to select the size (i.e., number of neurons) for a particular task or application *a priori*. Ideally, a network should be just large enough to meet task needs, without excessive redundancy. To this end, we propose a network growth and pruning policy based on the contribution of each neuron to the overall recurrent Infomax objective. More specifically, to add a new hidden neuron to an existing neuron x_i, we introduce the quantity:

$$G_i^t = a\mu^t - \sum_i^n |w_{ij}|, \quad (6)$$

where μ^t is the discounted MI associated with the i^{th} neuron, recursively specified as

$$\tau_\mu \mu^t = -(1 - \tau_\mu)\, \mu^{(t-1)} + MI(x_i^t; X_1^{t-1}), \quad (7)$$

where $\tau_\mu > 1$. Thus, the quantity (6) reflects the amount of information attributable to a neuron, penalized by its outgoing synaptic weights. We proceed to introduce a threshold criteria on (6) that allows the network to continue growing are gaining sufficient information relative to their synaptic weights, the latter approximating the 'cost' of building new connections.

Specifically, a neuron is added if $G_i^t > \kappa$ for some $\kappa \geq 0$. Similarly, pruning action is performed when $G_i^t < -\kappa$. With $\kappa > 0$, the interval $-\kappa < G_i^t < \kappa$ allows for convergence of the growth/pruning process.

The intuition behind (6) and the above threshold criteria is that we provide the network with more capacity if the i^{th} neuron is 'overloaded' with information. Here $a > 0$ is a rescaling constant, since the accumulated information μ^t can be much smaller than the cost of building connections emitting from that neuron.

C. Local Dynamics for Information-based Network Growth

The quantity (6) is 'global' quantity, since calculating the $MI(x_i^t; X_1^{t-1})$ requires the full joint distribution of the network. However, as in our prior work, we can approximate this quantity via recursive estimation, amounting to the addition of slow dynamics at the level of each neuron. Specifically, we introduce the approximate local MI as

$$MI(x_i^t; X_1^{t-1}) \approx \hat{M}^t = s_{a,i}^t - s_{b,i}^t + s_{c,i}^t - \log(1 - s_{2,i}^t). \quad (8)$$

where

$$\begin{aligned}\tau_a \Delta s_{a,i}^t &= -s_{a,i}^{t-1} + x_i^t \log \frac{g(v_i^t)}{1-g(v_i^t)};\\ \tau_b \Delta s_{b,i}^t &= -s_{b,i}^{t-1} + x_i^t \log \frac{s_{2,i}^t}{1-s_{2,i}^t};\\ \tau_c \Delta s_{c,i}^t &= -s_{c,i}^{t-1} + \log(1-p_i^t).\end{aligned} \tag{9}$$

and $s_{2,i}^t$ was a part of the previously specified local learning dynamics. The local dynamics proceed by making the substitution (8) into (7), resulting in

$$G_i^t \approx a\hat{\mu}^t - \sum_i^n |w_{ij}|, \tag{10}$$

$$\tau_\mu \hat{\mu}^t = -\left(1-\tau_\mu\right)\hat{\mu}^{(t-1)} + \hat{M}^t. \tag{11}$$

The dynamics (8)-(11) constitute a local growth/pruning process because the criteria is determined by activity at the level of each neuron. The complete derivation of these local dynamics is found in Appendix C.

IV. Examples

To proceed to illustrate both new capabilities (RL and autonomous growth/pruning) of the proposed network through two tasks: cartpole balancing and computing the sum of MNIST digits.

A. Inverted Pendulum

We consider the problem of balancing an inverted pendulum on a cart, simulated in the cartpole environment within open-ai gym [7]. The goal of this task is to apply a bidirectional force to a moving cart so as to balance an inverted pendulum, a hallmark control engineering application. Here, following the reward function is specified as:

$$R^t(\theta) = \begin{cases} 1, & \text{if } \theta < 15^\circ. \\ -1, & \text{otherwise,} \end{cases} \tag{12}$$

where θ is the pole angle from vertical.

We performed learning based on the hybrid RL/Infomax rule in (4) and also allowed the network to grow via the dynamics in (9)-(10). Learning proceeded in episodes wherein each episode ends when $\theta \geq 15^\circ$. The weight matrix at the end of an episode is retained at the start of the subsequent episode. While the network was fully recurrent, 16 neurons were reserved as an 'input layer' to encode the pole angle from $\theta = 0^\circ$ to 15°, while another 16 neurons for angular velocity $\dot{\theta}$. Finally, we used 5 output neurons to encode each of the two output actions (positive or negative forcing) with max pooling.

Figure 1A illustrates the performance of the network as a function of learning duration (averaged over 400 independent learning trials). As seen, the pendulum is kept stable for over 200 time steps after around 80 episodes. Figure 1B shows the average growth process during the first 60 training episodes. We notice that the network size starts to plateau around 60 episodes. However, it takes another 40 episodes for the network weights to converge, so that performance continues to increase even after the network reaches a steady size.

B. 1-step MNIST Summation

The cart pole example illustrates the RL capability of our proposed learning dynamics. However, because of its underlying Infomax objective, such learning should also be amenable to tasks that carry implicit information coding and memory requirements. To this end, we consider the 1-step MNIST summation task. The goal of this task is for the network to produce the sum of two sequentially presented MNIST digits [8], and thus requires coding the numerical meaning of the images while also learning the summation operation.

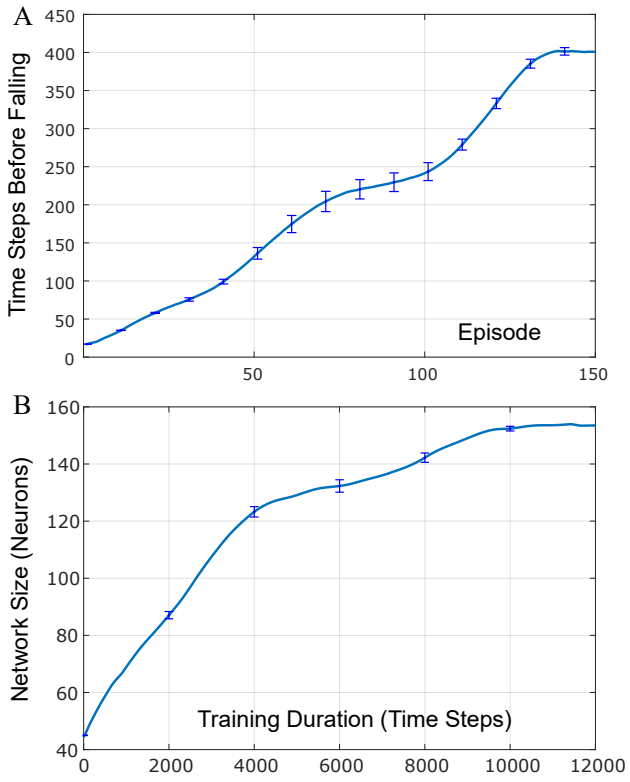

Fig. 1. **Inverted pendulum performance and network size.** (A) Duration inverted as function of learning episodes (mean and standard error over 400 trials). (B) Network growth during learning (mean and standard error over 400 trials).

We randomly chose 10000 MNIST images for training and another 10000 for validation. We allocated $28 \times 28 = 784$ neurons to receive pixel-wise input, and 19 output neurons to encode the possible sums ($\{0, ..., 18\}$).

Learning proceeded according to our proposed RL-Infomax rule with growth dynamics. The reward R^t was specified as:

$$R^t(\mathbf{z}^t) = R_{max}/2 - \|\mathbf{z}^t - \bar{\mathbf{z}}^t\|, \tag{13}$$

where $\mathbf{z}^t$ and $\bar{\mathbf{z}}^t$ are, respectively, the predicted and correct sum of input images at time t and $t-1$. Here $R_{max} = 18$ is the maximum possible error. Thus, the reward R^t takes integer values over a range of $[-9, 9]$. This form of reward function allows for a small error between $\bar{\mathbf{z}}^t$ and $\mathbf{z}^t$ to generate positive reward, whereas large distance leads to negative reward.

After the learning, we evaluated the performance of the network by applying a sequence of validation images to the input layer only, allowing the output to emanate from the network without further modulation of the weights.

Figure 2A shows the prediction accuracy (fraction of correct summation) over the validation-set, averaged over 100 independent learning trials. Two curves are shown in this figure. The blue curve is a network that is naive with respect to hyperparameters (here, especially the time constants of the dynamical variables) and without any pooling at the output neurons. We found that the performance in this case shows an almost linear increase as the network gets larger, but saturates at around 75 % accuracy for a relatively large network. Figure 2B shows the growth process associated with this learning scheme.

In contrast, we also considered the network embedded within a broader network architecture. Here, the 784 dimensional MNIST

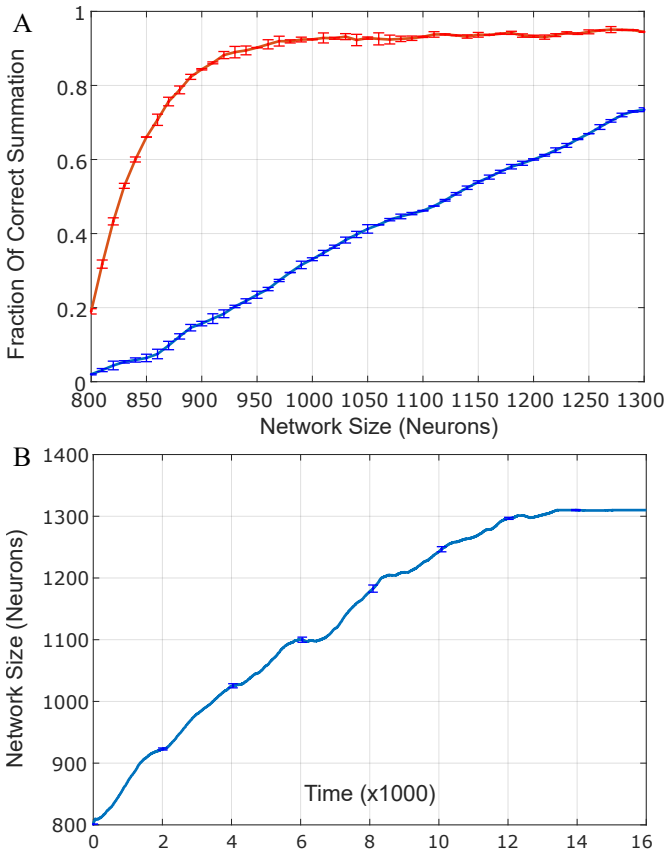

Fig. 2. **1-step MNIST summation performance.** (a) Task performance (fraction correct) as function of network size after learning. The blue curve is a vanilla network with the proposed learning rule, while the red curve represents a network with optimized dynamics and pooling. Mean and standard error plotted for 100 trials. (b) Network growth as a function of time steps (each associated with a sequential presentation of a digit).

digits were reduced to 100-dimension representations by means of a front-end encoder, hyperparameters were optimized *a priori* and max pooling was used at the output layer. The blue curve in Figure 2A shows performance in this case, wherein prediction accuracy approaches 95%.

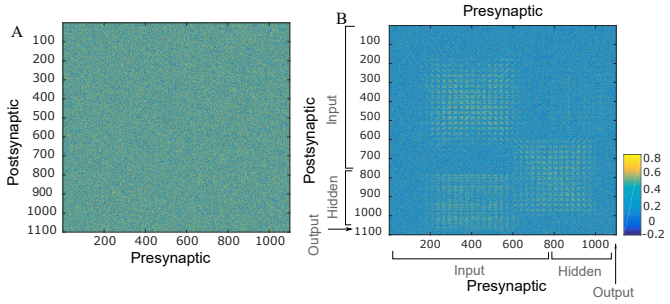

Fig. 3. **Self-organized structure in the network after learning for the 1-step MNIST task.** (A) Initially, the connectivity matrix is a random matrix. (B) Feedforward structure from input to hidden layer neurons appear in the model network, together with recurrence subnetworks within the input and hidden layers.

Figure 3 shows the emergent structure of the weights as a result of the learning process. The weights go from an initial random matrix (Figure 3A) to an organized structure with mostly feedforward connections from input layer to hidden layer and recurrent subnetwork structures elsewhere, as reflected by the checkerboard pattern of connections along the diagonal (Figure 3B).

We also examined the encoding properties of the neurons after learning. For this, we examined the spike-triggered average (STA) responses associated with 256 hidden neurons (Figure 4). We found that most neurons exhibit relatively low firing rates, with a few neurons producing more dense activation (Fig. 4A). The STAs suggest that the (recurrently connected) neurons reflect a 'complex cell' profile, wherein neurons encode specific digits at the immediate and prior time-step.

V. Discussion and Conclusion

A. Reinforcable Infomax

In this work, we develop a dynamical learning rule that incorporates a reinforcement mechanism within a recurrent Infomax paradigm. The reinforcement feature in our derived rule is different from most RL algorithms that explicitly address the interactions between the agent and the environment, such as policy optimization [9], [10] or policy evaluation through value functions [11]. Here, we started from the premise of recurrent mutual information maximization and obtained a learning rule that allows for both Hebbian/anti-Hebbian learning regimes. The reinforcement mechanism is realized via a reward-based modulation of the correlating and decorrelating actions of these underlying Infomax dynamics. Thus, the obtained rule uses reinforcement to bias an unsupervised learning schema, allowing for different modes of learning depending on task needs. This is reflected in the examples, wherein we considered tasks that are based purely on reward (cartpole) versus one that requires representing latent information (1-step MNIST).

We also note that the network model we assumed is Markovian, since the firing probability of each neuron only depends on the one-step previous network activity. However, whereas many canonical examples of RL treat the case of Markovian decision processes (e.g., Q-learning [11]), for the proposed network the task itself need not be Markovian.

B. Local Network Growth via Artificial Neurogenesis

The proposed growth algorithm provides a local solution for adding and removing neurons. Here, locality is interpreted as a criteria held at each neuron that involves only interactions between that neuron and its neighbors. This differs from the other network growth methods, where the criteria usually depends on information at the population level, such as the training error [12]–[14], or measures of redundancy between neurons [15]–[18]. A related proposed strategy includes randomly splitting nodes and then merging those that become highly correlated [6]. This latter technique has been applied primarily to feedforward networks. Further, many existing growth-pruning algorithms process node addition and removal separately ([15], [19], [20]), whereas our dynamics allow for fluid alternation between these operations.

Since our growth process is defined in terms of information quantities related to the underlying Infomax objective, the network can learn and grow at the same time. Therefore, such an algorithm is a general solution for learning and auto and potential for scalable computation in larger network. Indeed, the dynamics of our rule can be interpreted as a form of artificial 'neurogenesis' [21] in recurrent networks, with synapses and neurons being added through a process of need-based development.

C. Conclusion

From the standpoint of recurrent neural networks for computation, our results are interesting insofar as they provide a way to solve both

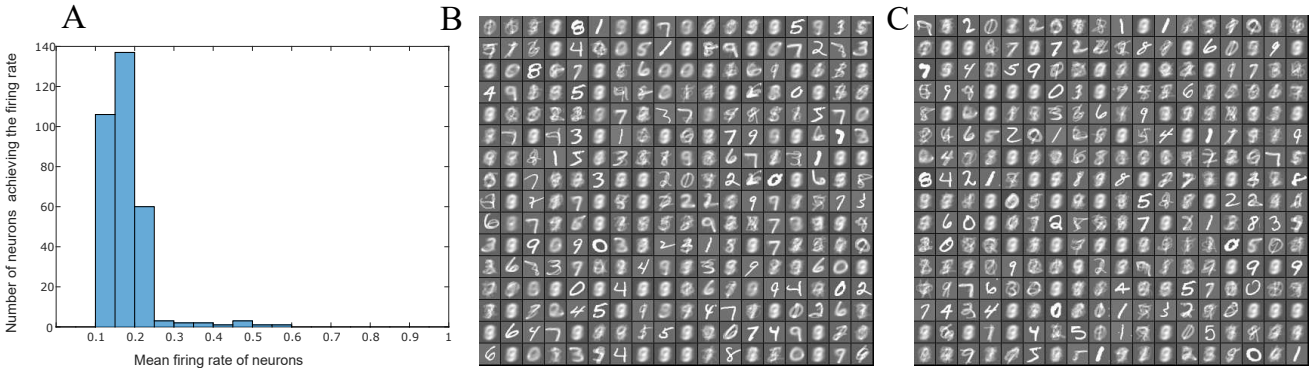

Fig. 4. **Neuron response properties.** (A) The majority of neurons have low mean firing rates. (B) Spike triggered averages associated with immediate input images (at time t), and (C) the same as (B) except for time $t-1$. Neurons appear to have the properties of complex cells, encoding specific digits at the current and previous time step.

unsupervised and goal-driven tasks with a unified learning scheme. Moreover, our proposed method obviates the need to parametrize the size of a network prior to engaging the task in question. It is harder to draw biological insights from our framework, since ultimately the model we have used is quite abstract. However, our results to at least emphasize the potential functional importance of dynamics at the level of neurons and their connections. Indeed, our solution involves the formulation of auxiliary dynamical processes that evolve as different time-scales than the activity of the neurons themselves. It is through these supportive processes that the local (and thus, more biophysically plausible) solutions can be realized.

APPENDIX A
EXACT FORM OF THE LEARNING RULE

From [5], learning rule that solves (1) can be written as:

$$\Delta w_{ij}^t = \gamma \mathbf{E}_{X^t}\left[\phi_{ij}^h(t) + \phi_{ij}^a(t)\right], \tag{14}$$

where the Hebbian and anti-Hebbian components in the information optimal learning rule (14) are:

$$\begin{aligned}\phi_{ij}^h(t) &= \phi^h(p_i^t, x_i^t, x_j^{t-1}) + \phi^h(p_i^t, 1-x_i^t, x_j^{t-1}),\\ \phi_{ij}^a(t) &= \phi^a(p_i^t, x_i^t, x_j^{t-1}) + \phi^a(p_i^t, 1-x_i^t, x_j^{t-1}).\end{aligned} \tag{15}$$

The two terms in the anti-Hebbian $\phi_{ij}^a(t)$ are:

$$\begin{aligned}\phi^a(p_i^t, x_i^t, x_j^{t-1}) &= 2\beta\left(\frac{\mathbf{E}[(p_i^t)^2] - \mathbf{E}[p_i^t]}{\mathbf{E}[p_i^t]}\right) x_j^{t-1}x_i^t + \\ &\quad 2\beta(1-p_i^t)x_j^{t-1}x_i^t;\\ \phi^a(p_i^t, 1-x_i^t, x_j^{t-1}) &= 2\beta\left(-\frac{\mathbf{E}[(p_i^t)^2] - \mathbf{E}[p_i^t]}{1-\mathbf{E}[p_i^t]}\right)\times \\ &\quad x_j^{t-1}(1-x_i^t) + 2\beta(-p_i^t)x_j^{t-1}(1-x_i^t).\end{aligned} \tag{16}$$

Similarly, the Hebbian part $\phi_{ij}^h(t)$ consists of:

$$\begin{aligned}\phi^h(p_i^t, x_i^t, x_j^{t-1}) &= 2\beta\left((1-p_i^t)\log\frac{p_i^t}{\mathbf{E}[p_i^t]}\right)x_j^{t-1}x_i^t;\\ \phi^h(p_i^t, 1-x_i^t, x_j^{t-1}) &= 2\beta\left(-p_i^t\log(\frac{1-p_i^t}{1-\mathbf{E}[p_i^t]})\right)x_j^{t-1}(1-x_i^t).\end{aligned} \tag{17}$$

APPENDIX B
LOCAL RECURSIVE APPROXIMATION OF THE RECURRENT INFOMAX RULE.

The above rule is global, since the expectations rely on the joint distribution of the entire network. However, under an ergodicity assumption, we can use local recursive estimation to approximate these functions. For this, we introduce new dynamical synaptic state variables, defined as:

$$\begin{aligned}s_{1,i}^t &= \mathbf{E}[p_i^t];\\ s_{2,i}^t &= \mathbf{E}[(p_i^t)^2];\\ s_{3,i}^t &= \mathbf{E}_{\mathbf{x}^t|X^{t-1}}(x_i^t);\\ s_{4,ij}^t &= \mathbf{E}\left(\mathbf{E}_{\mathbf{x}^t|X^{t-1}}(x_i^t)x_j^{t-1}\right);\\ s_{5,ij}^t &= \mathbf{E}\left[(1-p_i^t)\log\left(\frac{p_i^t}{\mathbf{E}[p_i^t]}\right)\mathbf{E}_{\mathbf{x}^t|X^{t-1}}(x_i^t)x_j^{t-1}\right];\\ s_{6,ij}^t &= \mathbf{E}\left[p_i^t\log\left(\frac{1-p_i^t}{1-\mathbf{E}[p_i^t]}\right)\mathbf{E}_{\mathbf{x}^t|X^{t-1}}(x_i^t)x_j^{t-1}\right].\end{aligned} \tag{18}$$

Learning rule (14) is thus approximated by (3), where

$$\begin{aligned}\bar{\phi_{ij}}^h(t) &= (\frac{s_{2,i}^t}{s_{1,i}^t})s_{4,ij}^t - (\frac{s_{2,i}^t - s_{1,i}^t}{1-s_{1,i}^t})s_{4,ij}^t,\\ \bar{\phi_{ij}}^a(t) &= s_{5,ij}^t + s_{6,ij}^t.\end{aligned} \tag{19}$$

Noting that $\mathbf{E}[p_i^t] = g(v_i^t)$, we can then develop recursive estimators for each of the above quantities via:

$$\begin{aligned}\tau_1\Delta s_{1,i}^t &= -s_{1,i}^{t-1} + g(v_i^t);\\ \tau_2\Delta s_{2,i}^t &= -s_{2,i}^{t-1} + (g(v_i^t))^2;\\ \tau_3\Delta s_{3,i}^t &= -s_{3,i}^{t-1} + g(v_i^t)x_i^t;\\ \tau_4\Delta s_{4,ij}^t &= -s_{4,ij}^{t-1} + s_{3,i}^t x_j^{t-1};\\ \tau_5\Delta s_{5,ij}^t &= -s_{5,ij}^{t-1} + (-g(v_i^t))s_{3,i}^t x_j^{t-1} + \\ &\quad (1-g(v_i^t))\log\left(\frac{g(v_i^t)}{s_{1,i}^t}\right)s_{3,i}^t x_j^{t-1};\\ \tau_6\Delta s_{6,ij}^t &= -s_{6,ij}^{t-1} + (-g(v_i^t))(1-s_{3,i}^t)x_j^{t-1} + \\ &\quad (-g(v_i^t))\log\left(\frac{1-g(v_i^t)}{1-s_{1,i}^t}\right)(1-s_{3,i}^t)x_j^{t-1}.\end{aligned} \tag{20}$$

APPENDIX C
RECURSIVE ESTIMATION OF THE GROWTH CRITERIA.

Following the notation in II-A, we let x_i^t be the state at time step t and X_1^{t-1} be the history of the recurrent network. Then, given the

history of network activity, the probability of a postsynaptic spike of node x_i^t is determined by a Bernoulli distribution:

$$P(x_i^t|X_1^{t-1}) = (p_i^t)^{x_i^t}(1-p_i^t)^{1-x_i^t}, \tag{21}$$

and similarly the marginal probability of x_i^t is:

$$P(x_i^t) = \sum_{X_1^{t-1}} P(x_i^t|X_1^{t-1})P(X_1^{t-1}) = (\mathbf{E}[p_i^t])^{x_i^t}(1-\mathbf{E}[p_i^t])^{1-x_i^t}, \tag{22}$$

where $\mathbf{E}[p_i^t] = \sum_{X_1^{t-1}} P(X_1^{t-1})p_i^t$ is the expectation with respect to X_1^{t-1}.

We recall the growth criteria (10) :

$$G_i^t = a\mu^t - \sum_i^n |w_{ij}|, \tag{23}$$

where μ^t defines the averaged MI associated with the i^{th} neuron, recursively specified as

$$\tau_\mu \mu^t = -(1-\tau_\mu)\,\mu^{(t-1)} + MI(x_i^t; X_1^{t-1}), \tag{24}$$

for $\tau_\mu > 1$.

Thus calculation of μ^t can be reduced to estimation of the immediate Shannon mutual information $MI(x_i^t; X_1^{t-1})$ of x_i^t with respect to the network, defined by (2):

$$MI(x_i^t; X_1^{t-1}) = \sum_{x_i^t, X_1^{t-1}} P(x_i^t, X_1^{t-1}) \log \frac{P(x_i^t|X_1^{t-1})}{P(x_i^t)}. \tag{25}$$

Substituting (21) and (22) to (2) yields

$$\begin{aligned} MI(x_i^t; X_1^{t-1}) = &\sum_{x_i^t, X_1^{t-1}} P(x_i^t, X_1^{t-1}) x_i^t \log \frac{p_i^t}{\mathbf{E}[p_i^t]} + \\ &\sum_{x_i^t, X_1^{t-1}} P(x_i^t, X_1^{t-1})(1-x_i^t) \log \frac{1-p_i^t}{1-\mathbf{E}[p_i^t]}. \end{aligned} \tag{26}$$

Denoting the expectation $\sum_{x_i^t, X_1^{t-1}} P(x_i^t, X_1^{t-1})$ by $\mathbf{E}[\cdot]$, (26) can be rewritten as:

$$\begin{aligned} MI(x_i^t; X_1^{t-1}) = &\mathbf{E}\left[x_i^t(\log p_i^t - \log \mathbf{E}[p_i^t])\right] + \\ &\mathbf{E}\left[(1-x_i^t)(\log(1-p_i^t) - \log(1-\mathbf{E}[p_i^t]))\right]. \end{aligned} \tag{27}$$

Rearranging (27) gives:

$$\begin{aligned} MI(x_i^t; X_1^{t-1}) = &\mathbf{E}\left[x_i^t \log \frac{p_i^t}{1-p_i^t}\right] - \\ &\mathbf{E}\left[x_i^t \log \frac{\mathbf{E}[p_i^t]}{1-\mathbf{E}[p_i^t]}\right] + \\ &\mathbf{E}\log(1-p_i^t) - \log(1-\mathbf{E}[p_i^t]). \end{aligned} \tag{28}$$

All of the expectations can be estimated via a recursive filter similar to (18) and (20), if we let:

$$\begin{aligned} s_{a,i}^t &= \mathbf{E}\left[x_i^t \log \frac{p_i^t}{1-p_i^t}\right]; \\ s_{b,i}^t &= \mathbf{E}\left[x_i^t \log \frac{\mathbf{E}[p_i^t]}{1-\mathbf{E}[p_i^t]}\right]; \\ s_{c,i}^t &= \mathbf{E}\log(1-p_i^t). \end{aligned} \tag{29}$$

Recall that $p_i^t = g(v_i^t)$ and $\mathbf{E}[p_i^t] = s_{2,i}^t$. Thus

$$MI(x_i^t; X_1^{t-1}) \approx \hat{M}^t = s_{a,i}^t - s_{b,i}^t + s_{c,i}^t - \log(1-s_{2,i}^t). \tag{30}$$

where these variables can be updated through

$$\begin{aligned} \tau_a \Delta s_{a,i}^t &= -s_{a,i}^{t-1} + x_i^t \log \frac{g(v_i^t)}{1-g(v_i^t)}; \\ \tau_b \Delta s_{b,i}^t &= -s_{b,i}^{t-1} + x_i^t \log \frac{s_{2,i}^t}{1-s_{2,i}^t}; \\ \tau_c \Delta s_{c,i}^t &= -s_{c,i}^{t-1} + \log(1-p_i^t). \end{aligned} \tag{31}$$

Then, (9) constitute recursive estimators for the quantities in (29), thus completing the derivation.

REFERENCES

[1] R. Linsker, "How to generate ordered maps by maximizing the mutual information between input and output signals," *Neural computation*, vol. 1, no. 3, pp. 402–411, 1989.

[2] E. Oja, "Simplified neuron model as a principal component analyzer," *Journal of mathematical biology*, vol. 15, no. 3, pp. 267–273, 1982.

[3] T. Toyoizumi, J.-P. Pfister, K. Aihara, and W. Gerstner, "Generalized bienenstockcoopermunro rule for spiking neurons that maximizes information transmission." *PNAS*, vol. 102, no. 14, pp. 5239–4244, April 2005.

[4] T. Tanaka, T. Kaneko, and T. Aoyagi, "Recurrent infomax generates cell assemblies, neuronal avalanches, and simple cell-like selectivity," *Neural computation*, vol. 21, no. 4, pp. 1038–1067, 2009.

[5] S. Liu and S. Ching, "Recurrent information optimization with local, metaplastic synaptic dynamics." *Neural Computation*, vol. 29, no. 9, pp. 2528–2552, 2017.

[6] M. M. Islam, M. A. Sattar, M. F. Amin, and X. Yao, "A new adaptive merging and growing algorithm for designing artificial neural networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 39, no. 3, Jun 2009.

[7] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[8] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, vol. 86, no. 11, p. 22782324, 1998.

[9] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning." *Machine Learning*, vol. 8, no. 3, pp. 229–256, 1992.

[10] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation." *Neural Information Processing Systems (NIPS)*, 2000.

[11] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 2017.

[12] M. Lehtokangas, "Modified cascade-correlation learning for classification," *IEEE Transactions on Neural Networks*, vol. 11, p. 795 798, 2000.

[13] R. Setiono and L. C. K. Hui, "Use of a quasi-newton method in a feedforward neural network construction algorithm," *IEEE Trans. Neural Netw.*, vol. 6, no. 1, pp. 273–277, Jan 1995.

[14] T. Ash, "Dynamic node creation in backpropagation networks," *Connect. Sci.*, vol. 1, no. 4, pp. 365–375, 1989.

[15] Y. Hirose, K. Yamashita, and S. Hijiya, "Backpropagation algorithm which varies the number of hidden units," *Neural Networks*, vol. 4, pp. 61–66, 1991.

[16] M. Wynne-Jones, "Node splitting: A constructive algorithm for feedforward neural networks," *Neural Comput. Appl.*, vol. 1, no. 1, pp. 17–22, 1993.

[17] T. M. Nabhan and A. Y. Zomaya, "Toward generating neural network structures for function approximation," *Neural Netw.*, vol. 7, no. 1, pp. 89–99, 1994.

[18] M. I. B. Shahid, M. M. Islam, M. A. H. Akhand, and K. Murase, "A new algorithm to design multiple hidden layered artificial neural networks," in *Proc. Joint Int. Conf. SCIS&ISIS*, Sep 2006, p. 463468.

[19] M. Islam and K. Murase, "A new algorithm to design compact two-hidden-layer artificial neural networks," *Neural Networks*, vol. 14, no. 9, p. 12651278, 2001.

[20] M. M. Islam and K. Murase, "An algorithm for automatic design of two hidden layered artificial neural networks," in *Proc. IJCNN*, 2000, pp. 467–472.

[21] G. Kempermann, L. Wiskott, and F. H. Gage, "Functional significance of adult neurogenesis," *Current opinion in neurobiology*, vol. 14, no. 2, pp. 186–191, 2004.