LETTER **Communicated by Danilo Jimenez Rezende**

A Learning Framework for Winner-Take-All Networks with Stochastic Synapses

Hesham Mostafa
hmmostafa@ucsd.edu
Gert Cauwenberghs
gert@ucsd.edu
Institute of Neural Computation, University of California San Diego, La Jolla, CA 92093, U.S.A.

Many recent generative models make use of neural networks to transform the probability distribution of a simple low-dimensional noise process into the complex distribution of the data. This raises the question of whether biological networks operate along similar principles to implement a probabilistic model of the environment through transformations of intrinsic noise processes. The intrinsic neural and synaptic noise processes in biological networks, however, are quite different from the noise processes used in current abstract generative networks. This, together with the discrete nature of spikes and local circuit interactions among the neurons, raises several difficulties when using recent generative modeling frameworks to train biologically motivated models. In this letter, we show that a biologically motivated model based on multilayer winner-take-all circuits and stochastic synapses admits an approximate analytical description. This allows us to use the proposed networks in a variational learning setting where stochastic backpropagation is used to optimize a lower bound on the data log likelihood, thereby learning a generative model of the data. We illustrate the generality of the proposed networks and learning technique by using them in a structured output prediction task and a semisupervised learning task. Our results extend the domain of application of modern stochastic network architectures to networks where synaptic transmission failure is the principal noise mechanism.

1 Introduction

The hypothesis that sensory perception is a process of active inference is key to explaining various perceptual phenomena (Gregory, 1980; Lee & Mumford, 2003). One form of this hypothesis is that the brain maintains a probabilistic model of the environment, which it then uses to infer latent causes and fill missing or ambiguous details in the noisy sensory input it receives (Friston, 2003). Traditionally, there has been a clear distinction between probabilistic modeling approaches developed based on practical

Neural Computation **30, 1542–1572** (2018)
doi:10.1162/neco_a_01080

considerations (Ackley, Hinton, & Sejnowski, 1985; Dayan, Hinton, Neal, & Zemel, 1995; Kingma & Welling, 2013; Goodfellow et al., 2014) and probabilistic models developed as mechanistic explanations of how the brain represents and manipulates probability distributions (Deneve, 2008; Ma, Beck, Latham, & Pouget, 2006; Mostafa, Müller, & Indiveri, 2015). While the later models are more biologically relevant, they lack the scalability and power of the former. Finding a middle ground between these two types of models could prove useful in two ways: insights gained from developing practical, large-scale, and biologically motivated generative models could shed some light on how the brain is able to model complex high-dimensional distributions, and the constraints imposed by the biological substrate could inform the development of more computationally efficient types of generative models.

One example of an attempt to find such a middle ground used stochastic spiking networks to implement and sample from Boltzmann machines (Buesing, Bill, Nessler, & Maass, 2011). This model was developed further through the use of more realistic neuronal dynamics (Neftci, Das, Pedroni, Kreutz-Delgado, & Cauwenberghs, 2014) and stochastic synapses (Neftci, Pedroni, Joshi, Al-Shedivat, & Cauwenberghs, 2016; Al-Shedivat, Neftci, & Cauwenberghs, 2014). Establishing a link between Boltzmann machines and the dynamics of biologically realistic networks, however, is difficult due to the need for a symmetric synaptic connectivity matrix. Moreover, generating samples from the Boltzmann distribution embodied by the spiking network (either unconditional samples or samples from the posterior distribution) requires running the network for several steps in order to approach the equilibrium distribution. For highly multimodal distributions, a significant number of steps might be needed for the Markov chain to mix, making it computationally expensive to sample from the network.

Restricted Boltzmann machines were among the first large-scale, effectively trainable generative models (Ackley, Hinton, & Sejnowski, 1985; Hinton, 2002). Recently, however, effective variational training methods have been developed for training generative architectures with a feedforward hierarchical structure of latent variables (Kingma & Welling, 2013; Rezende, Mohamed, & Wierstra, 2014) where generative samples can be obtained in a single pass through the network. Recent variational methods rely on having an analytically tractable distribution over the variables in one layer conditioned on their parent variables in the previous layer. For continuous latent variables, the distributions are typically gaussians whose mean and variance are functions of the parent variables, while categorical/discrete variables typically follow a softmax distribution. Such distributions do not map naturally onto the dynamics of biologically realistic networks. Moreover, they are not computationally cheap because they involve multiplications and exponentiation operations.

The main contribution of this letter is the development of an analytically tractable approximation of the probability distribution over possible

network states in multilayer networks of winner-take-all (WTA) circuits, where neurons in different WTA circuits are connected using stochastic synapses. Since the state of each WTA is a discrete variable (the identity of the winner) and we use samples from these discrete variables/WTAs to approximate various intractable expectations, we need to be able to backpropagate error information through these stochastic samples. The approximate expression for the probability distribution over network states that we develop allows us to make use of the recently introduced Gumbel-softmax approximate reparameterization of discrete distributions (Jang, Gu, & Poole, 2017; Maddison, Mnih, & Teh, 2016) to enable backpropagation through stochastic network samples. We thus obtain a general learning framework for WTA networks with stochastic synapses that can be applied to a wide range of learning problems.

In order to obtain an analytically tractable approximation of the probability distribution over possible network states, it was necessary to use an abstract network model with no real temporal dynamics. This multilayer abstract network of WTAs is evaluated layer by layer in the typical manner used to evaluate multilayer artificial neural networks, and it can be trained using standard machine learning packages. We show that the parameters of this abstract network model can be directly mapped to the parameters of a network of leaky integrate-and-fire (LIF) neurons with stochastic conductance-based synapses. To our knowledge, this is the first general training method for feedforward non-rate-based spiking networks with stochastic synapses, allowing us to obtain spiking network versions of common stochastic architectures such as variational autoencoders. This opens the way for implementing these stochastic architecture on power-efficient asynchronous neuromorphic chips (Qiao et al., 2015; Furber, Galluppi, Temple, & Plana, 2014; Benjamin et al., 2014; Park, Ha, Yu, Neftci, & Cauwenberghs, 2014), as well as investigating their behavior on a biologically realistic neural substrate. The feedforward structure sidesteps the biologically unrealistic requirement of symmetric weights found in spiking neural implementations of Boltzmann machines (Buesing, Bill, Nessler, & Maass, 2011). The feedforward structure also allows very fast sampling with no need to run a long Markov chain to obtain unbiased samples. The training method we used is an offline training method based on backpropagation, however. Further work is needed to develop a more biologically motivated learning method, in the spirit of the learning method in Mostafa, Ramesh, and Cauwenberghs (2017), that learns online and changes synaptic weights based only on information in the pre- and postsynaptic neurons.

We first use the proposed networks to solve a structured output prediction task in order to illustrate the soundness of our network approximation and training method. We then apply the proposed learning framework to learning a generative model of the MNIST data set using variational methods. Following recent trends in variational methods, we use a stochastic neural network to implement the variational posterior. The

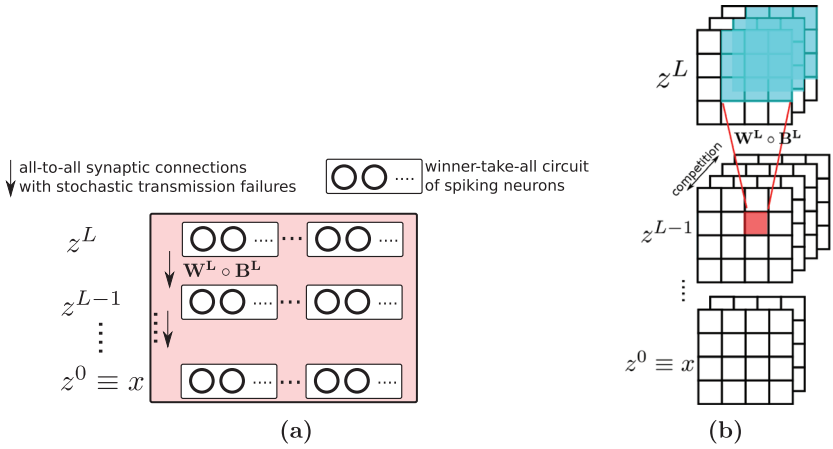

Figure 1: Multilayer networks of WTA circuits. (a) Network with fully connected layers of latent variables where each neuron projects to all neurons in the subsequent layer through connections that fail stochastically. (b) Network with convolutional layers where each layer is composed of a number of feature maps. Competition is across the feature maps dimension; all neurons at the same spatial position in one layer are part of the same WTA. At each spatial position, only one neuron can win the competition and generate a spike. As in standard convolutional layers, each neuron receives input from a local spatial receptive field, and the receptive field projection weights are tied across the spatial positions. The incidences of synaptic failure are not tied, however, and they are independent for each connection.

network we use to approximate the posterior distribution over the latent variables is also based on WTA circuits with stochastic synapses. Both the generative/decoder branch and the inference/encoder branch of the network are thus composed solely of discrete-valued WTA circuits. To illustrate the generality of the proposed networks, we also use them in a configuration that is inspired by ladder networks (Rasmus, Berglund, Honkala, Valpola, & Raiko, 2015) to solve a semisupervised learning task.

2 Model Description

We investigate multilayer networks of WTA circuits with the general structure shown in Figure 1. The network in Figure 1 has L layers where each layer can have a different number of WTA circuits. WTA circuits in the same layer all have the same number of neurons. This number may be different in different layers. Exactly one neuron in a WTA circuit spikes during one pass through the network, and this is the neuron receiving the largest input among the WTA neurons. Neurons are connected using stochastic synapses

where each synapse has an independent probability to fail to transmit a spike.

The interlayer connection pattern can be all-to-all as in Figure 1a or have a convolutional structure as in Figure 1b. The convolutional connection pattern is the same as in standard convolutional networks with tied weights at the different spatial positions. The synaptic failure incidences are not tied, however; two connections having the same tied weight fail independently. In a layer receiving all-to-all connections, the neurons are arbitrarily grouped into WTA circuits. In a convolutional layer, a WTA circuit is formed by all neurons having the same spatial position, that is, competition is across the feature maps dimension. Thus, at each spatial position, a spike is produced by only the most strongly activated feature map at that position. All convolutions are carried out using a stride of one and use zero padding to produce output feature maps having the same spatial dimensions as the input feature maps. For simplicity, the notation in the rest of the letter assumes all-to-all interlayer connections described by a 2D weight matrix. This notation easily accommodates the convolutional connections case if parts of the 2D weight matrix are assumed to be fixed at zero to reflect the spatial locality of the receptive fields.

Let $\mathbf{t^l}$ be the vector representing the total input to the neurons in layer l and $\mathbf{z^l}$ a binary 0/1 vector indicating whether each neuron in layer l has spiked (1) or not (0). Let $U^l(i)$ be a function that maps a neuron index i in layer l to the set of indices of all the neurons in the same WTA. For a convolutional layer, that would be all the neurons at the same spatial position. Then $\mathbf{t^l}$ and $\mathbf{z^l}$ are given by

$$\mathbf{t^l} = (\mathbf{W^{l+1}} \circ \mathbf{B^{l+1}})\mathbf{z^{l+1}} \qquad l = 0, \ldots, L-1 \tag{2.1a}$$

$$z_i^l = \begin{cases} 1 & \text{if} \quad t_i^l = \max\limits_{j \in U^l(i)} t_j^l \\ 0 & \text{otherwise} \end{cases} \qquad l = 0, \ldots, L-1, \tag{2.1b}$$

where z_i^l, t_i^l are the ith elements in vectors $\mathbf{z^l}$ and $\mathbf{t^l}$, respectively. $\mathbf{W^l}$ is the weight matrix from layer l to layer $l-1$, and $\mathbf{B^l}$ is a matrix of independent 0/1 Bernoulli random variables that is multiplied element-wise with $\mathbf{W^l}$. Each element in $\mathbf{B^l}$ can have a different probability of being 1. These probabilities are given by the matrices $\mathbf{Z^l} = \mathbb{E}\left[\mathbf{B^l}\right]$. Each connection thus has an independent probability to fail to transmit a spike. The lowest layer is the data layer, $z^0 \equiv x$. Note that network evaluation proceeds layer by layer and is thus an abstraction of the behavior of a dynamical spiking network.

We assume each neuron in the top layer has an equal probability to win the competition in its WTA circuit. This prior on the top-layer activity, together with the network weights and synaptic transmission failure probabilities, implicitly defines a probability distribution over the spike configurations generated by the network. This distribution can be written

as

$$\log\left(P_\theta(\mathbf{z^L}, \mathbf{z^{L-1}}, \ldots, \mathbf{z^1}, \mathbf{x})\right) = \log\left(P(\mathbf{z^L})\right) + \log\left(P_\theta(\mathbf{x} \mid \mathbf{z^1})\right) + \sum_{l=1}^{L-1} \log\left(P_\theta(\mathbf{z^l} \mid \mathbf{z^{l+1}})\right), \tag{2.2}$$

where θ is the collection of all the network parameters (weights and transmission failure probabilities) and $P(\mathbf{z^L})$ is the prior distribution over the top-layer activity.

It is easy to generate samples from the distribution given in equation 2.2 by sampling the top-layer prior and the failure probabilities of all the connections, then executing one pass through the network. In order to efficiently train these networks to optimize various expectations over the spike patterns generated by the network, we need to be able to quickly evaluate the probability of particular spike patterns, that is, we need an explicit expression for the distribution in equation 2.2. This means we need an explicit expression for the log-conditional distributions $\log\left(P_\theta(\mathbf{z^l} \mid \mathbf{z^{l+1}})\right)$. We derive this expression in section 2.1.

2.1 Deriving the Conditional Distribution of the Latent Variables. We consider a single WTA with C neurons. The WTA has C possible outputs where each output corresponds to a different neuron winning the competition and generating a spike. A neuron wins the competition in a WTA if it receives the largest input among the WTA neurons. Let t_i be the input to the ith neuron of the WTA and $\mathbf{z^{in}}$ the binary vector representing the input spike activity from the preceding layer. t_i and its first two moments are given by

$$t_i = (\mathbf{w_i} \circ \mathbf{b_i})^T \mathbf{z^{in}}, \tag{2.3}$$

$$\mu(t_i) = \mathbb{E}[t_i] = (\mathbf{w_i} \circ \boldsymbol{\zeta_i})^T \mathbf{z^{in}}, \tag{2.4}$$

$$\sigma^2(t_i) = \mathbb{E}\left[(t_i - \mu(t_i))^2\right] = (\mathbf{w_i}^2 \circ \boldsymbol{\zeta_i} \circ (1 - \boldsymbol{\zeta_i}))^T \mathbf{z^{in}}, \tag{2.5}$$

where $\mathbf{w_i}$ is the input weight vector for neuron i in the WTA. $\mathbf{b_i}$ is a vector of Bernoulli random variables, which model stochastic transmission failures and whose mean is $\boldsymbol{\zeta_i} = \mathbb{E}[\mathbf{b_i}]$. $\mathbf{b_i}$ is multiplied element-wise by $\mathbf{w_i}$. The mean and variance of t_i are given by equations 2.4 and 2.5, respectively. We make use of the central limit theorem to approximate the probability distribution of t_i (which is the sum of many independent random variables) by a gaussian having the same mean and variance. This approximation is quite accurate when the total number of nonzero inputs to the neuron is large. This is the number of nonzero entries in $\mathbf{z^{in}}$ that corresponds to the number of WTAs in the preceding layer. Figure 2 illustrates how the number of nonzero inputs affects the quality of the gaussian approximation. For 10 and 20 inputs, the discrete nature of the neuron's input distribution is evident,

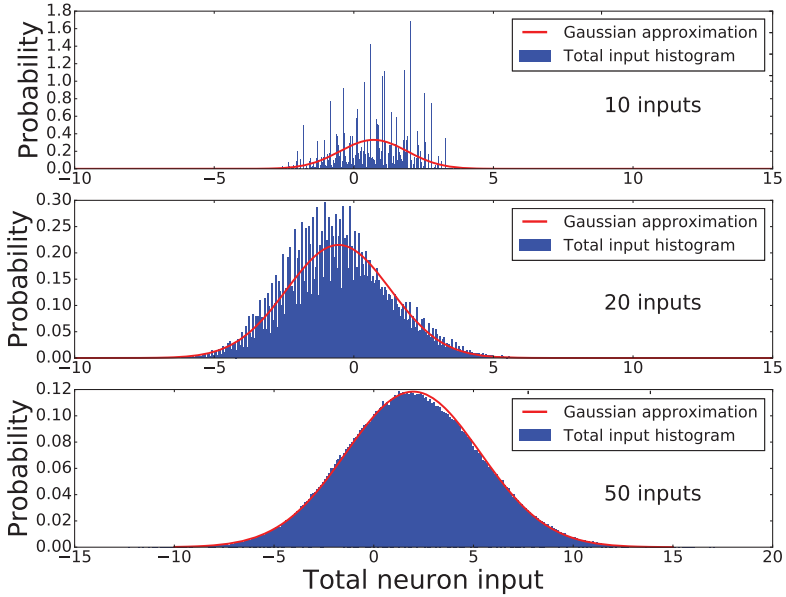

Figure 2: Quality of the gaussian approximation of the neuron's input distribution. We considered a neuron with 10, 20, and 50 inputs. The inputs are all 1 s. The input connection weights were drawn randomly from a standard normal distribution. Each connection has an independent probability to fail, which is drawn uniformly at random from [0, 1]. After fixing the connection weights and failure probabilities, we generate 500,000 samples for each of the three input sizes to construct the total input histogram.

as there are only 2^{10} and 2^{20} possible inputs, respectively, corresponding to the possible configurations of synaptic transmission failure. For 50 inputs, the input distribution becomes much smoother, and the gaussian approximation becomes more accurate.

We first consider a WTA with two neurons whose total inputs are t_1 and t_2. The probability that $t_1 > t_2$ is the probability that t_1 takes a particular value multiplied by the probability that t_2 takes a smaller value, averaged across all values. Alternatively, the probability that $t_1 > t_2$ is the probability that $r = t_1 - t_2 > 0$. Following the gaussian approximation of t_1 and t_2, r is also a gaussian with mean $\mu(t_1) - \mu(t_2)$ and variance $\sigma^2(t_1) + \sigma^2(t_2)$. The two equivalent ways of formulating $P(t_1 > t_2)$ are thus

$$P(t_1 > t_2) = \int_{-\infty}^{\infty} \phi\left(x; \mu(t_1), \sigma(t_1)\right) \mathrm{cdf}\left(x; \mu(t_2), \sigma(t_2)\right) dx, \tag{2.6}$$

$$P(t_1 > t_2) = P(r > 0) = \text{cdf}\left(\frac{\mu(t_1) - \mu(t_2)}{\sqrt{\sigma^2(t_1) + \sigma^2(t_2)}}; 0, 1\right), \tag{2.7}$$

$$\phi(x; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

$$\text{cdf}(x; \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{(y-\mu)^2}{2\sigma^2}} dy.$$

For a WTA with C neurons whose total inputs are $t_1, \ldots t_C$, and following the same reasoning as equation 2.6, the probability that neuron i receives the largest input is

$$p_i = P\left(\bigwedge_{j \neq i} t_i > t_j\right) = \int_{-\infty}^{\infty} \phi(x; \mu(t_i), \sigma(t_i)) \prod_{j \neq i} \text{cdf}\left(x; \mu(t_j), \sigma(t_j)\right) dx. \tag{2.8}$$

The integration in equation 2.8 is analytically intractable for $C > 2$. For $C = 2$, p_i reduces to the expressions in equations 2.6 and 2.7. To approximate the integration in equation 2.8, we approximate the product of the cdf functions by one of the cdf functions making up the product. That will usually be the cdf of the gaussian having the largest mean. This approximation is illustrated in Figure 3. The quality of the approximation improves when the means of the gaussians are well separated and the gaussians have low variance. This approximation is equivalent to approximating equation 2.8 as the minimum probability that neuron i wins a pairwise competition (governed by equations 2.6 and 2.7) with another neuron in the WTA:

$$\hat{p}_i = \min_{j \neq i} \left\{ \int_{-\infty}^{\infty} \phi(x, \mu(t_i), \sigma(t_i)) \, \text{cdf}\left(x; \mu(t_j), \sigma(t_j)\right) dx \right\}, \tag{2.9}$$

$$= \min_{j \neq i} \left\{ \text{cdf}\left(\frac{\mu(t_i) - \mu(t_j)}{\sqrt{\sigma^2(t_j) + \sigma^2(t_i)}}; 0, 1\right) \right\}, \tag{2.10}$$

$$\tilde{p}_i = \frac{\hat{p}_i}{\sum_j \hat{p}_i}. \tag{2.11}$$

In other words, we approximate the probability that a neuron in a WTA wins the competition (receives the largest input) by the probability that it wins the competition against one other neuron, where this other neuron is selected to be the neuron with the highest probability to receive a larger input than the first neuron. This approximation works in practice because winning the competition against the neuron that has the highest mean input implies winning the competition against other neurons as well with a high

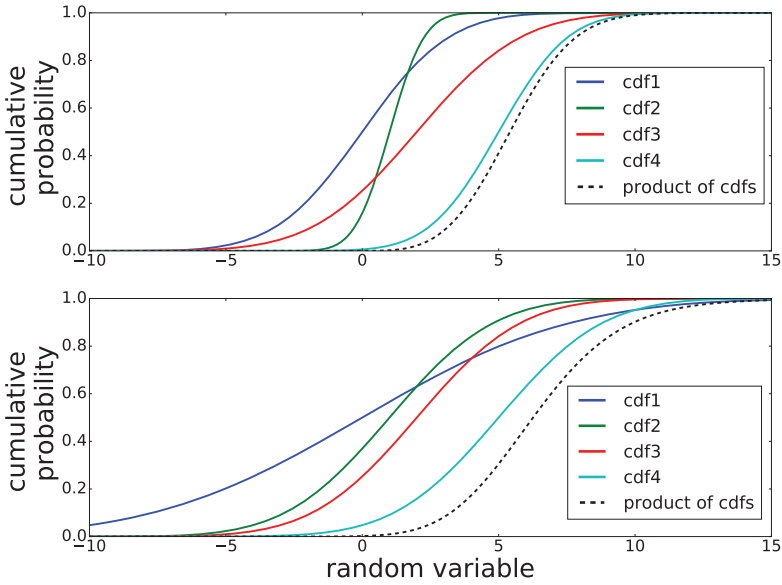

Figure 3: Approximating the product of four cumulative density functions (dashed black line) by one of the cdfs making up the product. In the top plot, this product is approximated reasonably well by cdf4. In the bottom plot, the approximation by cdf4 is worse due to the larger variance of some of the cdfs.

probability. This approximation yields an unnormalized probability $\hat{p}_i$. The normalized probability that neuron i wins the competition is given by $\tilde{p}_i$.

Equations 2.4, 2.5, 2.10, and 2.11 define an analytical, differentiable approximation to the probability of a neuron winning the competition in a WTA given the input layer binary activity vector $\mathbf{z^{in}}$. Since the WTAs in a layer are conditionally independent given the input layer activity, these equations allow us to obtain a differentiable expression for the winning probability of all the neurons in a layer with multiple WTAs. In a layer with multiple WTAs and where $\tilde{p}_i$ is the probability that neuron i wins the competition in its WTA given the input layer activity $\mathbf{z^{in}}$, the log probability of observing a particular spike pattern $\mathbf{z^{out}}$ (which is a 0/1 binary vector) is

$$\log\left(p(\mathbf{z^{out}} \mid \mathbf{z^{in}})\right) = \sum_i \log(\tilde{p}_i) z_i^{out}. \tag{2.12}$$

$\mathbf{z^{out}}$ has to be a valid spike pattern, with exactly one neuron in each WTA emitting a spike. Our approximation of a layer's conditional distribution can be used to obtain an explicit expression for the distribution in equation 2.2.

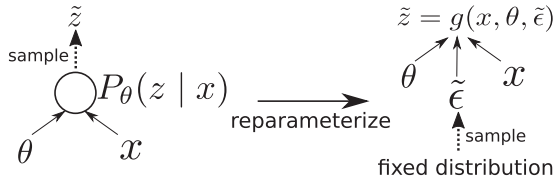

Figure 4: The reparameterization of probability distributions to obtain a differentiable relation between the samples and the distribution parameters. Instead of sampling directly from a distribution (left), the distribution is reparameterized so that a sample can be obtained using a differentiable deterministic transformation, g, where randomness is injected using stochastic samples $\tilde{\epsilon}$ from a fixed distribution (right). This results in a differentiable pathway from the stochastic sample $\tilde{z}$ to the distribution parameters θ and the conditioned variable x. For example, instead of sampling directly from a gaussian $\tilde{z} \sim \mathcal{N}(z; \mu(\theta, x), \sigma(\theta, x))$, we sample first from a standard normal distribution $\tilde{\epsilon} \sim \mathcal{N}(\epsilon, 0, 1)$ and then obtain $\tilde{z}$ using $\tilde{z} = g(x, \theta, \tilde{\epsilon}) = \mu(\theta, x) + \tilde{\epsilon}\sigma(\theta, x))$.

2.2 Reparameterizing the Latent Variables Distribution. When training stochastic WTA networks containing latent variables $\mathbf{z}$, we typically need to evaluate expectations of the form $\mathbb{E}_{P_\theta(\mathbf{z}|\mathbf{x_0})}(f(\mathbf{z}, \mathbf{x_0}))$, where $P_\theta(\mathbf{z} \mid \mathbf{x_0})$ is the probability distribution over the latent variables given an observed data point $\mathbf{x_0}$; θ is the vector of network parameters (synaptic weights and transmission failure probabilities), which implicitly define the distribution P; and f is a differentiable function of $\mathbf{z}$ and $\mathbf{x_0}$. f can be, for example, the log probability of observing a particular configuration of latent variables, or it can be a variational lower bound on the data log likelihood. Exact evaluation of this expectation is typically intractable, so it is often approximated using N samples from $P_\theta(\mathbf{z} \mid \mathbf{x_0})$:

$$\mathbb{E}_{P_\theta(\mathbf{z}|\mathbf{x_0})}(f(\mathbf{z}, \mathbf{x_0})) \approx \frac{1}{N} \sum_{\mathbf{z^i} \sim P_\theta(\mathbf{z}|\mathbf{x_0})} f(\mathbf{z^i}, \mathbf{x_0}). \tag{2.13}$$

In order to use gradient descent to maximize or minimize this expectation with respect to the the parameters θ, we need to be able to estimate $\nabla_\theta \mathbb{E}_{P_\theta(\mathbf{z}|\mathbf{x_0})}(f(\mathbf{z}, \mathbf{x_0}))$. Evaluating these derivatives is also analytically intractable, so we have to resort to sampling approaches. One of the classical algorithms for estimating gradients of stochastic expectations with respect to the parameters of the expectation probability density is the REINFORCE algorithm (Williams, 1992). Recently, however, better estimators with greatly reduced variance have been introduced (Kingma & Welling, 2013; Rezende et al., 2014; Ruiz, Titsias, & Blei, 2016). These estimators make use of the so-called reparameterization trick, which is illustrated in Figure 4. Through reparameterization, samples from a probability distribution can

be written as a differentiable transformation of the distribution parameters and stochastic samples from a fixed standard distribution. Thus, one can directly use the chain rule to obtain the derivative of the sample-based estimate of the expectation in equation 2.13 with respect to the distribution parameters θ since f is differentiable, and through the reparameterization trick, we have a differentiable relation between the stochastic samples $\mathbf{z^i}$ and the distribution parameters θ.

Not all distributions admit such a reparameterization. Most notably, reparameterizations of discrete distributions cannot be differentiable. An approximate differentiable reparameterization that works well in practice is the Gumbel-softmax reparameterization (Jang et al., 2017; Maddison et al., 2016), which we reproduce here. Given a discrete distribution described by the probability vector $\mathbf{p} = [p_1, \ldots, p_K]$ defining the probability of each of the distribution's K outcomes, a sample can be drawn from this discrete distribution using the following nondifferentiable reparameterization:

$$\tilde{\mathbf{z}} = \text{one\_hot}(\arg\max_j(g_j + \log(p_j))). \tag{2.14}$$

$\tilde{\mathbf{z}}$ is a one-hot vector with exactly one entry equal to one and the others to zero. The index of this entry is the sample outcome and is given by the arg max operator. $g_1, \ldots, g_K$ are independent samples from the $Gumbel(0, 1)$ distribution (Gumbel & Lieblein, 1954). Samples from $Gumbel(0, 1)$ can be obtained by first sampling from the uniform distribution $u_j \sim Uniform(0, 1)$ and then transforming the uniform samples using $g_j = -\log(-\log(u_j))$. $\tilde{\mathbf{z}}$ is a one-hot $K - dim$ vector that can take one of K possible values. In order to obtain a continuous differentiable reparameterization, the arg max operator is relaxed to the differentiable *softmax* function:

$$\hat{z}_j = \frac{\exp((g_j + \log(p_j))/\tau)}{\sum_{l=1}^{K} \exp((g_l + \log(p_l))/\tau)} \quad j = 1, \ldots, K. \tag{2.15}$$

The sample vector $\hat{\mathbf{z}} = [\hat{z}_1, \ldots, \hat{z}_K]$ is now differentiable with respect to the parameters of the discrete distribution, $\mathbf{p}$. $\hat{\mathbf{z}}$ is a sample from a continuous distribution that approaches the original discrete distribution as the temperature parameter τ approaches 0. While training the network, τ is gradually annealed toward 0. Instead of sampling $\mathbf{z^i}$ directly from $P_\theta(\mathbf{z} \mid \mathbf{x_0})$, we use the probabilities of the different discrete states of each WTA (given by equations 2.4, 2.5, 2.10, and 2.11) in equation 2.15 to sample the WTA states and use the resulting reparameterized continuous-valued samples, $\hat{\mathbf{z}}^\mathbf{i}$, to evaluate the expectation in equation 2.13. When sampling from multilayer networks such as the network in Figure 1, we use the reparameterized continuous-valued samples of one layer to evaluate the winning probabilities of the WTAs in the next layer, and then use these winning

probabilities to obtain reparameterized continuous-valued samples from the next layer. We do this layer by layer until we obtain reparameterized continuous-valued samples of all the WTAs in the network. Any differentiable cost function involving these continuous-valued samples can be optimized using backpropagation. During the annealing of τ, we always keep it well above zero to avoid instabilities in training since the derivatives of the stochastic samples with respect to the network parameters diverge as τ approaches zero. In the rest of this letter, all loss functions are optimized using ADAM (Kingma & Ba, 2014) within the Theano framework (Bastien et al., 2012; Bergstra et al., 2010).

In the exact case, only one neuron can win the competition in a WTA. Samples obtained using the continuous relaxation in equation 2.15, however, yield a continuous-valued activity vector for the WTA circuit, where neurons that have a higher probability of winning are more likely to have higher activities in the sample vectors. This can be understood as approximating the hard WTA circuit by a soft WTA circuit (Douglas & Martin, 2004) where the winning neuron does not completely shut down the activity in the other WTA neurons. We use this soft WTA mechanism only during training to allow gradients to flow back through stochastic samples. During testing, we use the hard WTA mechanism where only the neuron receiving the largest input emits a spike and use the exact network dynamics with stochastic synaptic transmission failures to generate samples from the network instead of the approximation in equation 2.11.

2.3 Networks with Realistic Neuronal and Synaptic Dynamics. The model presented previously abstracts away the dynamical nature of a realistic spiking network in favor of an analytically tractable behavior with no real temporal dynamics. Analyzing the behavior of a multilayer spiking network with realistic temporal dynamics is particularly challenging as an analytical formulation of the network behavior is typically not available. It has been common practice to first develop an abstract analytically tractable model and then map the parameters of the abstract model to that of a realistic spiking network model (Buesing et al., 2011; Pecevski, Buesing, & Maass, 2011; Nessler, Pfeiffer, & Maass, 2013). The discrepancy between the behavior of the two models is then empirically investigated to validate the ability of the abstract model to capture the relevant behavior of the more realistic model. This approach extends to training spiking network with realistic temporal dynamics where it has been very effective in training feedforward networks for classification tasks (O'Connor, Neil, Liu, Delbruck, & Pfeiffer, 2013; Diehl et al., 2015; Cao, Chen, & Khosla, 2015; Hunsberger & Eliasmith, 2015). This approach first trains an abstract artificial neural network using standard backpropagation techniques; then the weights are mapped to the realistic spiking network.

In this section, we introduce a biologically realistic spiking network model and introduce a modification to the training procedure of the

abstract model in order to obtain similar behavior from the two models once the parameters are mapped from the abstract model to the biologically realistic spiking model. We use LIF neurons with conductance-based synapses. One LIF neuron is used for each neuron in the abstract model. The interlayer connection patterns remain the same. Within the same layer, LIF neurons in the same WTA circuit have all-to-all inhibitory connections to obtain competitive behavior. The full neuron model, as well as the parameters used in the subsequent simulations, are given in the appendix.

There are two ways the behavior of the network model using LIF neurons can deviate from the behavior of the abstract model:

1. None of the LIF neurons in a WTA spike. LIF neurons have a spiking threshold mechanism, and no LIF neuron in a WTA will spike if all of them receive subthreshold input. In the abstract model, the neuron with the largest input in the WTA will spike regardless of the magnitude of that input.
2. The winning LIF neuron in a WTA is decided before all the WTAs in the previous layer have spiked. The asynchronous dynamics of the LIF spiking network can allow a WTA to decide a winner based on only partial input from the previous layer. This decision might be different from the winner decision obtained using the synchronous dynamics that evaluate the layers in a strict sequence where the neurons in one layer take into account input from all the winners in the previous layer.

We ameliorate the effects of the first source of deviation by adding a cost term to the training cost function that is minimized when the winning neuron in each WTA has an above-threshold mean input. More precisely, this additional cost term for an individual WTA has the form

$$threshold\_crossing\_cost = \begin{cases} -\max_i \mu(t_i) & \text{if} \quad \max_i \mu(t_i) < H \\ 0 & \text{otherwise} \end{cases}, \tag{2.16}$$

where $\mu(t_i)$ is the mean input to the ith neuron in the WTA (see equation 2.4) and H is chosen to be slightly larger than the total input needed to trigger a spike in an LIF neuron. Minimizing this cost term encourages the winning neuron in each WTA to have a superthreshold input allowing us to map the weights of the abstract model directly to a network of LIF neurons. The first source of deviation between the abstract model and the LIF network is not completely eliminated, though, as the additional cost term affects only the mean input. There is thus still a possibility that for some synaptic failure patterns, all neurons in a WTA will get subthreshold input and all fail to spike.

The second source of deviation can be ameliorated by scaling the synaptic weights by a factor greater than unity after training. Scaling the weights

does not change the winning probabilities in each WTA. Since the winning neurons are typically receiving a positive, superthreshold input due to the cost term in equation 2.16, this scaling results in all the winning neurons receiving a strong positive input that makes them spike in close temporal proximity. The network operation thus becomes similar to a synfire chain (Abeles, 1982), where a synchronous volley of spikes from the input layer triggers an almost synchronous volley of spikes from the next layer and so on (Rotter & Aertsen, 1998). Simulations of the LIF spiking network were carried out using NEST (Gewaltig & Diesmann, 2007).

3 Results

3.1 Structured Output Prediction. We first validate the soundness of our approximation of the WTA winning probabilities in a structured output prediction task where the goal is to predict the lower half of an MNIST image given the upper half. The MNIST data set contains 70,000 28×28 grayscale images of handwritten digits split into three groups of 50,000, 10,000, and 10,000 images for training, validation, and testing respectively. As in Raiko, Berglund, Alain, and Dinh (2014) and Gu, Levine, Sutskever, and Mnih (2015), we use a binarized version of the MNIST images obtained by thresholding the pixel intensities. We use two different types of feedforward networks, illustrated in Figures 5a and 5b, to predict the lower image half. The networks differ in the structure of the two latent variable layers: z^1 and z^2. z^1 and z^2 can either be fully connected layers that receive all-to-all connections from the previous layer (see Figure 5a) or convolutional layers where each neuron has a local receptive field (see Figure 5b). The final prediction layer, y, is always fully connected to the previous layer. We represent the binary inputs and outputs using WTA circuits with two neurons where the spike from one neuron codes for binary 0 and the spike from the other neuron for binary 1. Given a training pair x_{lower} and x_{upper}, the goal is to maximize the log probability of predicting the lower image half given the upper half, $\log(P_\theta(y = x_{lower} \mid x = x_{upper}))$, across all training pairs where P_θ is the probability distribution encoded by the network in Figure 5a. Exact evaluation of this training quantity would require marginalizing over the hidden variables z_1 and z_2. Instead, we estimate it using N samples from the hidden variables per example:

$$\frac{1}{N} \sum_{z_1^i \sim P_\theta(z_1 \mid x_{lower}), z_2^i \sim P_\theta(z_2 \mid z_1^i)} \log(P_\theta(y = x_{lower} \mid z_2^i)), \tag{3.1}$$

where we use $N = 1$ sample during training. We generate samples using the continuous reparameterization from equation 2.15. This way, we can backpropagate errors through these samples to the network parameters and to samples from earlier layers. During training, we use an annealing schedule

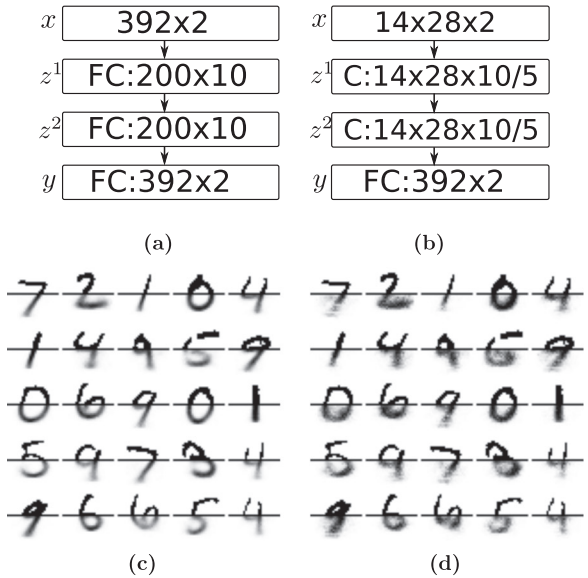

Figure 5: (a, b) Networks used in structured output prediction task. *C*: $A \times B \times C/D$ indicates a convolutional layer with *C* feature maps, $A \times B$ spatial dimensions, and $D \times D$ kernel size. *FC*: $A \times B$ indicates a fully connected layer with *A* WTAs and *B* neurons in each WTA. (c, d) Network prediction of the lower half of sample test digits. Each digit prediction used 100 samples. Horizontal line in each digit separates the upper (input) from the bottom (predicted) half. (c) Samples generated using the abstract fully connected network in panel a. (d) Samples generated using an LIF network with the same structure as panel a. The abstract network in panel a was trained with the threshold-crossing constraint and the parameters mapped to the LIF network.

to lower the temperature of the softmax used to approximate the hard WTA mechanism. The learning rate also decays during training. The validation set was used to tune the temperature and learning rate schedules. Although synaptic transmission failure can be a trainable parameter, we kept it fixed at 0.5 for all synapses and train only the synaptic weights.

The network is tested using samples generated in two different ways: samples generated using the approximate analytical distribution used during training where temperature is set to zero and samples generated using the exact network dynamics, that is, using the hard WTA mechanism and synaptic transmission failures. In both cases, 100 samples were used to evaluate the log likelihood of the lower half of the image given the upper half for each test digit. The results are shown in the first two rows of Table 1. The network performs better on the test set using the approximate analytical distribution used during training. The difference in performance

Table 1: Performance of Structured Output Prediction Networks (Negative Log-Likelihood).

	Samples Generated Using Approximate Analytical Distribution	Samples Generated Using Exact Network Dynamics
Fully connected abstract network (see Figure 5a)	60.1 ± 0.067 nats	61.8 ± 0.074 nats
Fully connected abstract network with threshold-crossing constraint	62.86 ± 0.29 nats	65.68 ± 0.27 nats
Convolutional abstract network (see Figure 5b)	66.98 ± 0.10 nats	71.77 ± 0.44 nats
Convolutional abstract network with threshold-crossing constraint	70.09 ± 0.35 nats	73.6 ± 0.39 nats
Fully connected LIF network	–	89.8 nats
Convolutional LIF network	–	118.5 nats

Notes: Mean and standard deviation from 10 runs. Only the result of one run is reported for the LIF networks.

compared to the exactly generated samples is slight, however, indicating that the approximation distribution reasonably matches the exact distribution over the test set and that effective learning can be achieved using the approximate distribution. The convolutional network significantly lags the fully connected network in performance. This is expected, as the local structure at a particular point in the upper digit half has very little to do with the local structure at the corresponding point in the lower half. The test set negative log likelihood of 61.8 ± 0.074 nats achieved by the fully connected network version is slightly worse than the test set negative log likelihood of 58.5 nats achieved with previous stochastic networks (Jang et al., 2017) that use discrete neural variables with activation noise rather than synaptic transmission noise.

We retrained the networks in Figures 5a and 5b using the threshold-crossing constraint in equation 2.16. As shown in Table 1, this leads to a degradation of performance as the network parameters now have an additional constraint that is unrelated to the objective. We mapped the abstract WTA networks trained using the threshold-crossing constraint to a spiking network with LIF neurons and evaluated the log likelihood of the lower image half using 50 samples. These samples were generated by simulating the network of LIF neurons 50 times using different synaptic failure patterns. There is a significant degradation in performance when going to an asynchronous network of LIF neurons due to the two sources of deviation outlined in section 2.3. The performance degradation is more severe in the case of convolutional networks due to their local receptive fields structure; by using a 5×5 kernel, a neuron in a convolutional layer receives input from only 25 active neurons. It is thus more probable that a synaptic

failure pattern will cause the neuron to receive a subthreshold input and fail to spike, compared to the fully connected network where each neuron receives input from 392 or 200 active neurons. Figures 5c and 5d show some examples of the network prediction of the lower halves of MNIST test set digits.

3.2 Learning Generative Models Using Variational Autoencoders. Our goal is to learn a generative model for a set of data points with discrete values $\mathbf{X} = \{\mathbf{x^1}, \mathbf{x^2}, \ldots, \mathbf{x^N}\}$ where $\mathbf{x^i} \in \{1, 2, \ldots, k\}^D$. We use a feedforward network of WTA circuits connected by stochastic synapses to encode the generative distribution. This is the generation network shown in Figure 6a. We lump all the latent variables together in one vector $\mathbf{z} = [\mathbf{z^1}, \ldots, \mathbf{z^L}]$, and the distribution over the network spike patterns is given by $P_\theta(\mathbf{x}, \mathbf{z})$ where θ represents the generation network parameters. For the prior on the top-layer variables, $\mathbf{z}^L$, we choose the uniform prior where each neuron has the same probability to win the competition in its WTA. Our goal is to maximize the data log likelihood $\log(P_\theta(\mathbf{X}))$. We use variational methods to maximize this likelihood. For a particular input point, $\mathbf{x}$,

$$\log(P_\theta(\mathbf{x})) = \sum_{\mathbf{z}} P_\theta(\mathbf{z} \mid \mathbf{x}) \log(P_\theta(\mathbf{x})) = \sum_{\mathbf{z}} P_\theta(\mathbf{z} \mid \mathbf{x}) \log\left(\frac{P_\theta(\mathbf{x}, \mathbf{z})}{P_\theta(\mathbf{z} \mid \mathbf{x})}\right). \tag{3.2}$$

$P_\theta(\mathbf{x}, \mathbf{z})$ is easy to evaluate using Equations 2.2 and 2.12. The posterior distribution $P_\theta(\mathbf{z} \mid \mathbf{x})$ is, however, intractable. We approximate the intractable posterior by the parameterized distribution $Q_\phi(\mathbf{z} \mid \mathbf{x})$. We use a multilayer network of WTAs connected using stochastic synapses to implement $Q_\phi(\mathbf{z} \mid \mathbf{x})$. This is the inference/recognition network shown in Figure 6a. It has an analogous structure to the generation network except that the connections between layers are going in the opposite direction. The weights and synaptic transmission failure probabilities in the two networks are independent. Given x, the inference network induces a probability distribution $Q_\phi(\mathbf{z} \mid \mathbf{x})$ over the latent variables, which can be expressed in an analogous fashion to the generative distribution using equations 2.2 and 2.12. The distribution parameters ϕ are the weights and synaptic transmission failure probabilities in the inference network. Note that unlike typical variational autoencoder architectures, the latent variable layers are directly connected and there are no deterministic layers in the network. The data loglikelihood can be written as

$$\log(P_\theta(\mathbf{x})) = \sum_{\mathbf{z}} Q_\phi(\mathbf{z} \mid \mathbf{x}) \log\left(\frac{P_\theta(\mathbf{x}, \mathbf{z})}{P_\theta(\mathbf{z} \mid \mathbf{x})} \frac{Q_\phi(\mathbf{z} \mid \mathbf{x})}{Q_\phi(\mathbf{z} \mid \mathbf{x})}\right) \tag{3.3}$$

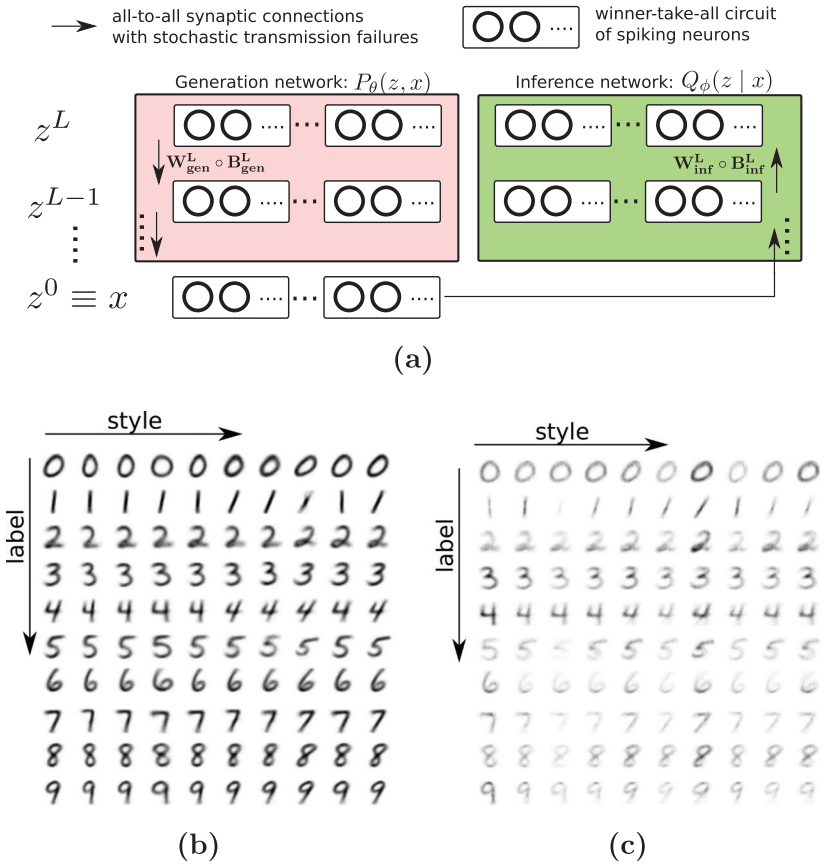

Figure 6: (a) Structure of the variational autoencoder network. (b, c). Digits generated using the generative network with separate style and label WTAs in the top layer. Each digit is constructed by fixing the top-layer activity, then averaging across 300 samples. (b) Digits generated by the abstract network. (c) Digits generated by an LIF network having the same structure. The corresponding abstract network was trained using the threshold-crossing constraint; then the parameters were directly mapped to the LIF network. The digits are less well defined since sometimes neurons in one WTA receive subthreshold input and all fail to spike, making it more likely that the visible layer WTAs also fail to spike.

$$= KL(Q_\phi(\mathbf{z} \mid \mathbf{x}) \| P_\theta(\mathbf{z} \mid \mathbf{x})) + \sum_{\mathbf{z}} Q_\phi(\mathbf{z} \mid \mathbf{x}) \log \left(\frac{P_\theta(\mathbf{x}, \mathbf{z})}{Q_\phi(\mathbf{z} \mid \mathbf{x})} \right) \tag{3.4}$$

$$= KL(Q_\phi(\mathbf{z} \mid \mathbf{x}) \| P_\theta(\mathbf{z} \mid \mathbf{x})) + \mathcal{L}(\mathbf{x}; \theta, \phi). \tag{3.5}$$

KL is the Kullback-Leibler divergence between two distributions, and since it is nonnegative, $\mathcal{L}$ is a lower bound on the data log likelihood given by $\mathcal{L} = \mathbb{E}_{Q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left(\frac{P_\theta(\mathbf{x},\mathbf{z})}{Q_\phi(\mathbf{z}|\mathbf{x})}\right)\right]$. Since it is intractable to evaluate $\mathcal{L}$ exactly due to the required summation over all $\mathbf{z}$, we approximate it using N samples from the inference network:

$$\mathcal{L} \approx \frac{1}{N} \sum_{\mathbf{z^i} \sim Q_\phi(\mathbf{z}|\mathbf{x})} \log\left(\frac{P_\theta(\mathbf{x}, \mathbf{z^i})}{Q_\phi(\mathbf{z^i} \mid \mathbf{x})}\right). \tag{3.6}$$

We maximize this sample-based lower bound through gradient descent on the parameters θ and ϕ, where we use the differentiable Gumbel-softmax reparameterization described in section 2.2 to backpropagate errors through the stochastic samples. A common problem with training variational autoencoders with multiple layers of stochastic hidden variables is the collapse of the latent approximate posterior $Q_\phi(\mathbf{z} \mid \mathbf{x})$ toward the latent prior $P_\theta(\mathbf{z})$, which stops the latent variables in upper layers from encoding information about the input during training (Sønderby et al., 2016; Chen et al., 2016). We adopt the solution proposed in Sønderby et al. (2016) in which we initially maximize only the reconstruction likelihood and gradually morph the cost function to optimize the variational lower bound $\mathcal{L}$. Our optimization objective is thus

$$\mathcal{J} = \frac{1}{N} \sum_{\mathbf{z^i} \sim Q_\phi(\mathbf{z}|\mathbf{x})} \log\left(P_\theta(\mathbf{x} \mid \mathbf{z^i})\right) + \beta \log\left(\frac{P_\theta(\mathbf{z^i})}{Q_\phi(\mathbf{z^i} \mid \mathbf{x})}\right), \tag{3.7}$$

where β gradually increases from 0 to 1 during training. When $\beta = 1$, $\mathcal{J}$ is equivalent to the right-hand side of equation 3.6; it becomes a sample-based estimate of $\mathcal{L}$.

We apply the variational autoencoder network based on WTA circuits and stochastic synapses to learning a generative model of MNIST digits. As in the structured output prediction task, we anneal the temperature of the softmax used in the Gumbel-softmax continuous reparameterization as well as the learning rate during training. We use $N = 1$ samples per data point during training.

We trained a network with three stochastic fully connected hidden layers of sizes 10×10, 20×10, and 30×10 going from the deepest layer downward (an $A \times B$ layer has A WTAs with B neurons in each WTA) in order to learn a generative model of the training set of MNIST digits. These layers were followed by the visible layer of 784×2 neurons. We used a uniform prior in the top layer where each neuron has an equal chance to win the competition in its WTA and used a fixed transmission failure probability of 0.5. After training, we evaluated the variational lower bound using the approximate distribution over WTA winners and $N = 50$ samples in

equation 3.6. The samples were also drawn from the approximate distribution. This evaluation was done on the test set. Under the approximate distribution, we obtain a variational lower bound of -98.9 ± 0.3 nats (mean and standard deviation from 10 runs), which is on par with discrete stochastic neural networks that use activation noise rather than synaptic transmission noise (Jang et al., 2017).

Estimating the variational lower bound using the exact network dynamics is particularly difficult in our case as exact evaluation of $P_\theta(\mathbf{x}, \mathbf{z})$ and $Q_\phi(\mathbf{z} \mid \mathbf{x})$ is intractable. Theoretically, we can estimate these probability distributions using samples, but we observed that the number of samples needed to reliably estimate the exact $P_\theta(\mathbf{x}, \mathbf{z})$ and $Q_\phi(\mathbf{z} \mid \mathbf{x})$ is computationally prohibitive. Note that this problem does not arise in conventional variational autoencoders as they have exact expressions for the conditional log probability of each layer's activity. There are two steps to our approximation of the winning probability of the neurons in a WTA: the first step assumes the input to each neuron is gaussian, and the second step approximates the intractable integration in equation 2.8 with the tractable integration in equation 2.9. The first step is practically unavoidable, as the exact distribution of the input to a neuron is an intractable discrete distribution where each point corresponds to one configuration of synaptic failures. We can dispense with the second step, however, by numerically evaluating the integration in equation 2.8.

Our estimate of the variational lower bound that is more faithful to the exact network dynamics uses exactly generated samples $\mathbf{z}^\mathbf{i}$ in equation 3.6, that is, samples of $\mathbf{z}$ generated by sampling the synaptic transmission failure probabilities, and it uses numerical integration of equation 2.8 to obtain accurate winning probabilities in each WTA. The accurate winning probabilities are used to evaluate $P_\theta(\mathbf{x}, \mathbf{z})$ and $Q_\phi(\mathbf{z} \mid \mathbf{x})$. Under this estimate, the variational lower bound is -104 ± 1.4 nats, which is slightly worse than the lower bound evaluated using the approximate distribution used during training.

We repeated the training and evaluation steps for the same network but using the threshold crossing constraint. As in the structured output prediction task, enforcing the threshold crossing constraint reduced the network performance. The results are summarized in Table 2. The parameters of the abstract network using the threshold crossing constraint can be directly mapped to the parameters of an LIF network. However, evaluating the variational lower bound in the LIF network is computationally intractable as it would require an infeasibly large number of samples where acquiring each sample involves the simulation of a large system of differential equations. Instead, we visually evaluate the LIF network performance on a generative task with style-label separation: we trained a variational autoencoder network in which a subset of the WTAs in the top layer was forced to represent the MNIST image label. The network had three stochastic hidden layers of sizes 20×10, 30×10, and 30×10 going from the deepest layer

Table 2: Variational Lower Bound Evaluated for a Network with Three Hidden Layers of Latent Variables.

	Evaluation Using Approximate Samples and Approximate Analytical Distribution	Evaluation Using Exact Samples and Numerical Integration of Equation 2.8
Training without threshold-crossing constraint	-98.9 ± 0.3 nats	-104 ± 1.4 nats
Training with threshold-crossing constraint	-102.46 ± 0.18 nats	-109.12 ± 0.37 nats

Note: Mean and standard deviation from 10 runs.

down. The top-layer prior used during training was label dependent: 5 of the 20 WTAs in the top layer had a delta function prior centered on the neuron whose index corresponds to the label of the current training example, while the remaining 15 WTAs used a uniform prior. During training, different values of the top 15 WTAs with uniform prior will represent different variants of each digit (where the digit is specified by the other 5 WTAs). Aesthetically, each configuration of the 15 WTAs will give rise to 10 digits (0 to 9) having similar style. We chose 5 WTAs to represent the label in order to have a significant label-dependent part in the top-layer activity. We trained the network twice—once without the threshold crossing constraint and once with the threshold crossing constraint. We then mapped the parameters of the network trained with the threshold crossing constraint to a LIF network. Figures 6b and 6c show the images generated by the abstract network trained without the threshold crossing constraint and the images generated by the LIF network. We sample from the generative network by first forcing the 5 label WTAs to represent one label and randomly choosing a winner in the remaining 15 WTAs in the top layer; that is, we choose a label and a random style. We then let the network dynamics sample from the remaining layers.

3.3 Semisupervised Learning Using Ladder Networks. Noise plays a crucial role in several autoencoding architectures (Vincent, Larochelle, Bengio, & Manzagol, 2008; Bengio, Laufer, Alain, & Yosinski, 2014; Valpola, 2015) where it limits the information flowing between network layers, thereby forcing networks to learn more compact and general representations in an unsupervised manner. In a semisupervised setting, the representations learned in an unsupervised manner are fed to a classifier, and few labeled examples are used to train the classifier (and the underlying representations as well). By simultaneously learning representations to optimize the unsupervised loss (reconstruction error) and the supervised loss

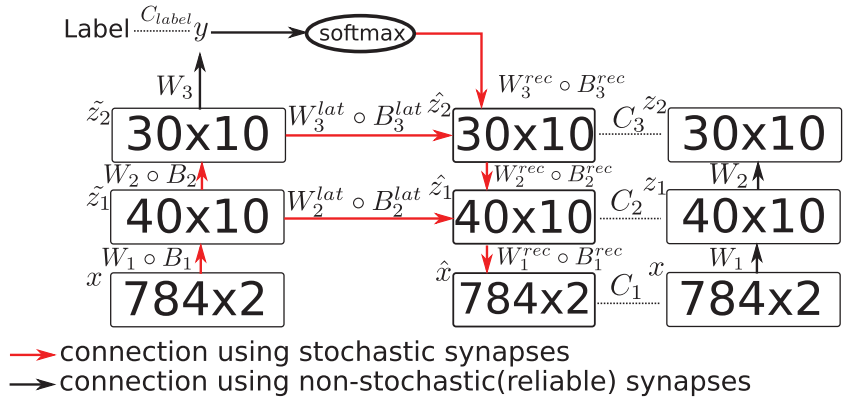

Figure 7: Ladder network used in the MNIST semisupervised learning task. There are two input pathways—one using stochastic synapses (leftmost pathway) and one using deterministic synapses (rightmost pathway). Both pathways use the same synaptic weights. For unlabeled examples, all the weights are trained to maximize the log probability of reconstructing the activity in the deterministic input pathway using the reconstruction pathway (middle pathway). These are the cost terms C_1, C_2, and C_3. For labeled examples, only the weights in the input pathway, W_1, W_2, and W_3, are trained to maximize the probability of the correct label in the upper classifier layer with 10 units.

(label prediction error), the network can learn to extract the label-relevant information using only a few labeled examples.

We used WTA networks with stochastic synapses to construct an autoencoder and added a linear classifier on top of the deepest hidden layer. We applied this architecture to an MNIST semisupervised learning task where only a subset of the labeled examples from the training set was used to train the classifier. The full (unlabeled) training set was used to train the autoencoder. We used a ladder architecture (Rasmus et al., 2015) for the autoencoder as shown in Figure 7 with lateral connections between the encoding and decoding branches. In the original ladder networks (Rasmus et al., 2015), the real-valued activations in the encoding branch (see the leftmost branch in Figure 7) are corrupted through the addition of gaussian noise. In our case, the corruption of the encoding branch is achieved through the use of stochastic synapses. The parallel, uncorrupted branch providing the reconstruction targets (see the rightmost branch in Figure 7) uses deterministic synapses and the same weight matrices as the corrupted encoding branch. For each training example, the encoding branch and reconstruction branch are sampled to obtain $\hat{\mathbf{p}}_\mathbf{x}$, $\hat{\mathbf{p}}_{\mathbf{1}}$, and $\hat{\mathbf{p}}_{\mathbf{2}}$, which are the probability vectors for generating a spike in the reconstruction branch layers. $\mathbf{x}$, $\mathbf{z_1}$, and $\mathbf{z_2}$

Table 3: MNIST Test Set Accuracy for Different Numbers of Labeled Training Examples.

	100	300	2000	5000
Abstract network without threshold crossing constraint	$92.8 \pm 0.9\%$	$94.7 \pm 0.26\%$	$95.1 \pm 0.13\%$	$95.7 \pm 0.22\%$
Abstract network with threshold crossing constraint	$92.2 \pm 1.7\%$	$94.7 \pm 0.34\%$	$95.3 \pm 0.083\%$	$95.7 \pm 0.13\%$
LIF network	90.0%	91.9%	92.1%	93.3%

Note: For the abstract networks, means, and standard deviations from five runs.

are the binary spike vectors in the corresponding layers in the clean input branch. The unsupervised reconstruction loss is then given by

$$L_{unsupervised} = -\left(\langle \mathbf{x}, log(\hat{\mathbf{p}}_{\mathbf{x}})\rangle + \langle \mathbf{z_1}, log(\hat{\mathbf{p}}_{\mathbf{1}})\rangle + \langle \mathbf{z_2}, log(\hat{\mathbf{p}}_{\mathbf{2}})\rangle\right), \tag{3.8}$$

where $\langle , \rangle$ is the dot product operator. $\mathbf{y}$ is the input vector to the top classifier layer. The supervised loss at the classifier layer is the cross-entropy loss:

$$L_{supervised} = -\log \frac{\exp(y_r)}{\sum_i \exp(y_i)}, \tag{3.9}$$

where r is the input label index. During training, a fixed labeled subset of the training set is used to minimize $L_{supervised}$, while the full training set was used to minimize $L_{unsupervised}$. As in previous experiments, we anneal the temperature of the softmax used in the Gumbel-softmax continuous reparameterization as well as the learning rate during training, and keep synaptic failure probabilities fixed at 0.5. When classifying the test set, a deterministic version of the encoder branch is used where no synaptic transmission failures occur. Table 3 shows the classification accuracy after training with different numbers of labeled examples.

We trained the abstract network using the threshold crossing constraint (see equation 2.16) and then mapped the parameters to an LIF network. We then evaluated the classification performance of both networks. The classification accuracy results of the deterministic version of both network with reliable synapses are shown in Table 3. Unlike the structured output prediction and variational autoencoder tasks, there is no drop in performance when using the threshold crossing constraint during training of the abstract network. Performance takes a hit, however, in the LIF network. In the deterministic case, the difference in activity between the LIF network and the

abstract network is due to the second source of deviation outlined in section 2.3: neurons that spike quickly before all the WTAs in the previous layer have spiked, thereby yielding a winner decision based only on partial input from the previous layer.

We are not aware of any previous work that used ladder networks with discrete neurons in a semisupervised task. A semisupervised learning approach that also uses discrete neurons is based on restricted Boltzmann machines and achieves worse performance (92.0% accuracy with 800 labeled examples) (Larochelle & Bengio, 2008). A convolutional spiking network was previously trained layer by layer in an unsupervised manner followed by a classifier that was trained using few labeled examples (Panda & Roy, 2016). Our approach results in much sparser activity and outperforms this previous work when using few labeled examples. This previous work, however, reaches significantly better accuracy when the number of labeled training examples increases.

4 Discussion

Local neural competition mediated by inhibitory populations (Douglas & Martin, 1992) is a ubiquitous phenomenon in biological networks. This competition can take one of several forms such as recurrent excitation that causes activity in the neural population receiving the largest input to ramp up and suppress the activity in the other populations through a common inhibitory population (Douglas & Martin, 2004). Alternatively, competition can take the form of a race to spike among the neurons, where the neuron receiving the strongest input spikes first, triggering a volley of spikes from inhibitory neurons that suppresses activity in the local circuit. The latter mechanism, which does not keep a persistent memory of the winner, can be regulated by oscillatory inhibition where the race to spike occurs during the decaying phase of the rhythmic inhibition. This synchronizes the excitatory spikes generated in the network to periodically recurring temporal windows in which the inhibition is low (Fries, Nikolić, & Singer, 2007). The WTA networks described in this letter are an abstraction of the later form of competitive dynamics where the network generates a synchronous volley of spikes during each pass/cycle.

Local neural competition can give rise to two distinct forms of the WTA mechanism. In one form, the winner's output is all or nothing (e.g., one spike) and does not encode the input to the winner; in the second form, the activity of the winning population/neuron encodes the input it received. The two forms roughly correspond to an argmax operation and a max operation, respectively. Networks employing the latter form of the WTA mechanism (the max form) can be directly trained using standard backpropagation techniques. The max WTA form thus finds application in a variety of networks (Goodfellow, Warde-Farley, Mirza, Courville, & Bengio, 2013; Srivastava, Masci, Kazerounian, Gomez, & Schmidhuber, 2013)

where it effectively partitions the network into many overlapping subnetworks. Each subnetwork corresponds to a different set of winners. Such virtual partitioning improves performance as subnetworks or clusters of subnetworks can specialize to different parts of the input space (Srivastava, Masci, Gomez, & Schmidhuber, 2014). The argmax form of the WTA mechanism, however, is simpler to implement using spiking networks as the winning neuron can simply emit one spike, and information is solely encoded in the identity of the winners. Moreover, the argmax form of the WTA enables faster processing in spiking networks compared to the max form, as the neuron does not have to transmit high-precision information using latency or rate codes. The virtual network partitioning property of the max form carries over to the argmax form. The argmax form is also more attractive for neuromorphic implementations, as network evaluation does not involve any multiplications. Due to its many advantages, we have focused mainly on the argmax form of the WTA mechanism in this letter. The main downside of the argmax form is that it is nondifferentiable. We circumvented this problem by using a softmax approximation and annealing the softmax temperature during training. Stochastic networks employing the max form of the WTA, however, can still be readily trained using the framework presented in this letter; instead of sampling the winners in each WTA, we would instead sample the input to each neuron (under the gaussian approximation) and allow only the neuron receiving the maximum sampled input in each WTA to transmit its input while the activity of all other neurons is suppressed. The sampled input would be obtained using a reparameterization of the gaussian distribution, allowing error information to backpropagate through the samples to the weights and transmission failure probabilities controlling the mean and variance of the gaussian inputs to the neurons.

Our training procedure is based on backpropagation through stochastic samples. Backpropagation is biologically unrealistic for several reasons, such as the need to interleave forward and backward passes and the use of symmetric weights in the forward and backward passes. More biologically plausible models have been proposed to address these issues that use contrastive learning in energy-based models (Xie & Seung, 2003; Bengio & Fischer, 2015; Scellier & Bengio, 2017), or that relax the symmetry requirement by using random weights in the backward pass (Lillicrap, Cownden, Tweed, & Akerman, 2016; Baldi, Sadowski, & Lu, 2016; Nøkland, 2016; Mostafa et al., 2017). These methods, however, have been applied in supervised learning settings, and their performance and applicability to learning in stochastic networks are unclear. Moreover, many of these approaches are based on neurons with smooth activation functions, so their applicability to a non-rate-based spiking network such as ours is questionable. The form of a biologically plausible learning rule for multilayer stochastic spiking networks is thus still an open question.

The stochastic neural network framework we developed in this letter makes use of synaptic transmission failure as the sole noise mechanism. Synaptic transmission in biological networks is highly unreliable in many cases (Allen & Stevens, 1994; Faisal, Selen, & Wolpert, 2008; Borst & Gerard, 2010). There is evidence that such unreliability is not due to biophysical constraints as synapses; even synapses with few release sites can be highly reliable (Volgushev, Voronin, Chistiakova, Artola, & Singer, 1995; Stratford, Tarczy-Hornoch, Martin, Bannister, & Jack, 1996). This suggests that synaptic transmission failure could potentially serve a useful computational role (Branco & Staras, 2009). Stochastic network architectures typically use noisy neurons rather than noisy synapses (Ackley et al., 1985; Valpola, 2015; Rezende et al., 2014; Kingma & Welling, 2013). The choice of injecting noise directly into the neurons is motivated by the analytical tractability of the neuronal noise model, which often leads to simple analytical expressions for the probability distribution over possible activity patterns. In this letter, we used the synaptic noise model and derived an approximate analytical expression relating the synaptic noise to the response variability in WTA networks. Both the mean and variance of the neuron's input are now direct functions of the same synaptic weights (see equations 2.4 and 2.5) and do not use the separate pathways for controlling mean and variance commonly used in abstract stochastic networks (Kingma & Welling, 2013; Rezende et al., 2014). By making use of a more biophysically explicit and measurable noise mechanism such as synaptic transmission failure, the probabilistic networks presented in this letter are better suited for developing mechanistic models of probabilistic computations in the brain compared to stochastic networks using abstract neuronal noise mechanisms.

Using the approximate expression for the probability distribution over network states, the proposed networks are effectively trainable using recent approaches for training discrete stochastic networks (Jang et al., 2017; Maddison et al., 2016). At test time, the difference between the expectations calculated using the network's exact and approximate probability distributions is minimal. The learning framework based on the approximate network distribution is thus accurate enough and general enough to train the proposed stochastic WTA networks in a wide range of scenarios.

The development of biologically inspired stochastic network models that can be applied to practical problems has mainly focused on approximating Boltzmann machines (Buesing et al., 2011; Neftci et al., 2014, 2016) using recurrently connected spiking neurons. Unbiased samples cannot be quickly obtained from these models as they require running a Markov chain Monte Carlo sampler. While Boltzmann distributions are quite general, multilayer feedforward stochastic networks trained end-to-end using backpropagation have in recent years displayed superior performance as generative models (Goodfellow et al., 2014; Kingma & Welling, 2013). Moreover, many powerful stochastic architectures for semisupervised learning tasks (Valpola, 2015) and autoencoding tasks (Vincent et al., 2008; Bengio et al.,

Table 4: Parameters of the LIF Spiking Neurons.

Parameter	Description	Value
C_m	Membrane capacitance	250.0 pF
g_L	Leak conductance	16.7 nS
E_L	Leak reversal potential	−70.0 mV
E_{exc}	Excitatory synapse reversal potential	0.0 mV
E_{exc}	Inhibitory synapse reversal potential	−85.0 mV
τ_{syn}	Synaptic conductance time constant	3.0 ms
V_{th}	Firing threshold	−65.0 mV
V_{reset}	Reset potential	−70.0 mV
T_{ref}	Absolute refractory period	10.0 ms

2014) do not fit into the Boltzmann machines framework. The framework of WTA networks with stochastic synapses that we present here, however, can implement a much wider class of modern stochastic network architectures while still maintaining reasonable biological realism in the noise model and nonlinearities used.

Appendix: Spiking Neuron Model and Simulation Parameters

The dynamics of the conductance-based leaky integrate and fire neuron are described by

$$\begin{aligned} C_m\dot{V}_m(t) &= g_L(E_l - V_m(t)) + g_{exc}(t)(E_{exc} - V_m(t)) \\ &\quad + g_{inh}(t)(E_{inh} - V_m(t)), \end{aligned} \tag{A.1}$$

$$\dot{g}_{exc}(t) = -\frac{g_{exc}(t)}{\tau_{syn}} + \sum_i \max(w_i, 0) \sum_r \delta(t - t_i^r), \tag{A.2}$$

$$\dot{g}_{inh}(t) = -\frac{g_{inh}(t)}{\tau_{syn}} + \sum_i \max(-w_i, 0) \sum_r \delta(t - t_i^r), \tag{A.3}$$

$$V_m(t) \leftarrow V_{reset} \quad \text{if} \quad V_m(t) > V_{th}, \tag{A.4}$$

where V_m is the membrane potential and g_{exc} and g_{inh} are the excitatory and inhibitory synaptic conductances, respectively. The summation over i runs over all presynaptic neurons. t_i^r is the time of the rth spike from presynaptic neuron i and w_i is the weight of the synapse from presynaptic neuron i. Positive weights trigger an increase in the excitatory synaptic conductance g_{exc}, while negative weights trigger an increase in the inhibitory synaptic conductance g_{inh}. The neuron has a refractory period of T_{ref}. The description of the neuron parameters and the parameter values used in simulation are given in Table 4. Neurons in the same WTA are connected using inhibitory

connections with weight $w_{inh}^{WTA} = -200.0$. The weights of the trained abstract model are scaled by a factor of 4 before using them in the LIF spiking network.

Acknowledgments

This work was supported by the Swiss National Science Foundation Early Postdoc Mobility grant P2ZHP2_164960, by the National Science Foundation under grant 1640081, and the Nanoelectronics Research Corporation, a wholly owned subsidiary of the Semiconductor Research Corporation, through Extremely Energy Efficient Collective Electronics, an SRC-NRI Nanoelectronics Research Initiative.

References

Abeles, M. (1982). *Local cortical circuits. An electrophysiological study*. Berlin: Springer.

Ackley, D., Hinton, G., & Sejnowski, T. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science*, *9*, 147–169.

Al-Shedivat, M., Neftci, E., & Cauwenberghs, G. (2014). *Learning non-deterministic representations with energy-based ensembles*. arXiv:1412.7272.

Allen, C., & Stevens, C. (1994). An evaluation of causes for unreliability of synaptic transmission. *Proceedings of the National Academy of Sciences*, *91*, 10380–10383.

Baldi, P., Sadowski, P., & Lu, Z. (2016). *Learning in the machine: Random backpropagation and the learning channel*. arXiv:1612.02734.

Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I., Bergeron, A., . . . Bengio, Y. (2012). *Theano: New features and speed improvements*. arXiv:1211.5590.

Bengio, Y., & Fischer, A. (2015). *Early inference in energy-based models approximates backpropagation*. arXiv:1510.02777.

Bengio, Y., Laufer, E., Alain, G., & Yosinski, J. (2014). Deep generative stochastic networks trainable by backprop. In *Proceedings of the International Conference on Machine Learning* (pp. 226–234).

Benjamin, B. V., Gao, P., McQuinn, E., Choudhary, S., Chandrasekaran, A. R., Bussat, J., . . . Boahen, K. (2014). Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proceedings of the IEEE*, *102*, 699–716.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., . . . Bengio, Y. (2010). Theano: A CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference* (vol. 4, p. 3). http://conference.scipy.org/proceedings/scipy2010/

Borst, J., & Gerard, G. (2010). The low synaptic release probability in vivo. *Trends in neurosciences*, *33*, 259–266.

Branco, T., & Staras, K. (2009). The probability of neurotransmitter release: Variability and feedback control at single synapses. *Nature Reviews Neuroscience*, *10*, 373–383.

Buesing, L., Bill, J., Nessler, B., & Maass, W. (2011). Neural dynamics as sampling: A model for stochastic computation in recurrent networks of spiking neurons. *PLoS Computational Biology*, *7*, e1002211.

Cao, Y., Chen, Y., & Khosla, D. (2015). Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, *113*, 54–66.

Chen, X., Kingma, D., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., & Abbeel, P. (2016). *Variational lossy autoencoder*. arXiv:1611.02731.

Dayan, P., Hinton, G., Neal, R., & Zemel, R. (1995). The Helmholtz machine. *Neural Computation*, *7*, 889–904.

Deneve, S. (2008). Bayesian spiking neurons I: Inference. *Neural Computation*, *20*, 91–117.

Diehl, P. U., Neil, D., Binas, J., Cook, M., Liu, S.-C., & Pfeiffer, M. (2015). Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *Proceedings of the International Joint Conference on Neural Networks*. Piscataway, NJ: IEEE.

Douglas, R., & Martin, K. (1992). A functional microcircuit for cat visual cortex. *J. Physiol.*, *440*, 735–769.

Douglas, R., & Martin, K. (2004). Neural circuits of the neocortex. *Annual Review of Neuroscience*, *27*, 419–456.

Faisal, A., Selen, L., & Wolpert, D. (2008). Noise in the nervous system. *Nature Reviews Neuroscience*, *9*, 292–303.

Fries, P., Nikolić, D., & Singer, W. (2007). The gamma cycle. *Trends in Neurosciences*, *30*, 309–316.

Friston, K. (2003). Learning and inference in the brain. *Neural Networks*, *16*, 1325–1352.

Furber, S., Galluppi, F., Temple, S., & Plana, L. (2014). The SpiNNaker project. *Proceedings of the IEEE*, *102*, 652–665.

Gewaltig, M.-O., & Diesmann, M. (2007). Nest (neural simulation tool). *Scholarpedia*, *2*, 1430.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., . . . Bengio, Y. (2014). Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems*, *27* (pp. 2672–2680). Red Hook, NY: Curran.

Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., & Bengio, Y. (2013). Maxout networks. In *Proceedings of the International Conference on Machine Learning* (pp. 1319–1327).

Gregory, R. (1980). Perceptions as hypotheses. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, *290*, 181–197.

Gu, S., Levine, S., Sutskever, I., & Mnih, A. (2015). *Muprop: Unbiased backpropagation for stochastic neural networks*. arXiv:1511.05176.

Gumbel, E., & Lieblein, J. (1954). *Statistical theory of extreme values and some practical applications: A series of lectures*. Washington, DC: U.S. Government Printing Office.

Hinton, G. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, *14*, 1771–1800.

Hunsberger, E., & Eliasmith, C. (2015). *Spiking deep networks with LIF neurons*. arXiv:1510.08829.

Jang, E., Gu, S., & Poole, B. (2017). Categorical reparameterization with Gumbel-softmax. *Stat*, *1050*, 1.

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

Kingma, D., & Welling, M. (2013). *Auto-encoding variational Bayes*. arXiv:1312.6114.

Larochelle, H., & Bengio, Y. (2008). Classification using discriminative restricted Boltzmann machines. In *Proceedings of the 25th International Conference on Machine Learning* (pp. 536–543). New York: ACM.

Lee, T., & Mumford, D. (2003). Hierarchical Bayesian inference in the visual cortex. *Journal of the Optical Society of America. A, Optics, Image Science, and Vision*, *20*, 1434–1448.

Lillicrap, T., Cownden, D., Tweed, D., & Akerman, C. (2016). Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, *7*.

Ma, W., Beck, J., Latham, P., & Pouget, A. (2006). Bayesian inference with probabilistic population codes. *Nature Neurosci.*, *9*, 1432–1438.

Maddison, C., Mnih, A., & Teh, Y. (2016). *The concrete distribution: A continuous relaxation of discrete random variables*. arXiv:1611.00712.

Mostafa, H., Müller, L. K., & Indiveri, G. (2015). Rhythmic inhibition allows neural networks to search for maximally consistent states. *Neural Computation*, *27*, 2510–2547.

Mostafa, H., Ramesh, V., & Cauwenberghs, G. (2017). *Deep supervised learning using local errors*. arXiv:1711.06756.

Neftci, E., Das, S., Pedroni, B., Kreutz-Delgado, K., & Cauwenberghs, G. (2014). Event-driven contrastive divergence for spiking neuromorphic systems. *Frontiers in Neuroscience*, *7*.

Neftci, E., Pedroni, B., Joshi, S., Al-Shedivat, M., & Cauwenberghs, G. (2016). Stochastic synapses enable efficient brain-inspired learning machines. *Frontiers in Neuroscience*, *10*.

Nessler, B., Pfeiffer, M., & Maass, W. (2013). Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity. *PLoS Computational Biology*, *9*, e1003037.

Nøkland, A. (2016). Direct feedback alignment provides learning in deep neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnet (Eds.), *Advances in neural information processing systems*, *29* (pp. 1037–1045). Red Hook, NY: Curran.

O'Connor, P., Neil, D., Liu, S.-C., Delbruck, T., & Pfeiffer, M. (2013). Real-time classification and sensor fusion with a spiking deep belief network. *Frontiers in Neuroscience*, *7*.

Panda, P., & Roy, K. (2016). Unsupervised regenerative learning of hierarchical features in spiking deep networks for object recognition. In *Proceedings of the 2016 International Joint Conference on Neural Networks* (pp. 299–306). Piscataway, NJ: IEEE.

Park, J., Ha, S., Yu, T., Neftci, E., & Cauwenberghs, G. (2014). A 65k-neuron 73-Mevents/s 22-pJ/event asynchronous micro-pipelined integrate-and-fire array transceiver. In *Proceedings of the 2014 Biomedical Circuits and Systems Conference* (pp. 675–678). Piscataway, NJ: IEEE.

Pecevski, D., Buesing, L., & Maass, W. (2011). Probabilistic inference in general graphical models through sampling in stochastic networks of spiking neurons. *PLoS Computational Biology*, *7*, e1002294.

Qiao, N., Mostafa, H., Corradi, F., Osswald, M., Stefanini, F., Sumislawska, D., & Indiveri, G. (2015). A re-configurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses. *Frontiers in Neuroscience*, *9*.

Raiko, T., Berglund, M., Alain, G., & Dinh, L. (2014). Techniques for learning binary stochastic feedforward neural networks. *Stat*, *1050*, 11.

Rasmus, A., Berglund, M., Honkala, M., Valpola, H., & Raiko, T. (2015). Semi-supervised learning with ladder networks. In C. Cortes, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems*, *28* (pp. 3546–3554). Red Hook, NY: Curran.

Rezende, D., Mohamed, S., & Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of Machine Learning Research* (pp. 1278–1286).

Rotter, S., & Aertsen, A. (1998). Accurate spike synchronization in cortex. *Zeitschrift für Naturforschung C*, *53*, 686–690.

Ruiz, F., Titsias, M., & Blei, D. (2016). The generalized reparameterization gradient. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems, 29* (pp. 460–468). Red Hook, NY: Curran.

Scellier, B., & Bengio, Y. (2017). Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in Computational Neuroscience*, *11*.

Sønderby, C., Raiko, T., Maaløe, L., Sønderby, S., & Winther, O. (2016). Ladder variational autoencoders. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems*, *29* (pp. 3738–3746). Red Hook, NY: Curran.

Srivastava, R., Masci, J., Gomez, F., & Schmidhuber, J. (2014). *Understanding locally competitive networks*. arXiv:1410.1165.

Srivastava, R., Masci, J., Kazerounian, S., Gomez, F., & Schmidhuber, J. (2013). Compete to compute. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems, 26* (pp. 2310–2318). Red Hook, NY: Curran.

Stratford, K., Tarczy-Hornoch, K., Martin, K., Bannister, N., & Jack, J. (1996). Excitatory synaptic inputs to spiny stellate cells in cat visual cortex. *Nature*, *382*, 258.

Valpola, H. (2015). From neural PCA to deep unsupervised learning. In E. Bingham, S. Kaski, J. Laaksonen, & J. Lampinen (Eds.), *Advances in independent component analysis and learning machines* (pp. 143–171).

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning* (pp. 1096–1103). New York: ACM.

Volgushev, M., Voronin, L., Chistiakova, M., Artola, A., & Singer, W. (1995). All-or-none excitatory postsynaptic potentials in the rat visual cortex. *European Journal of Neuroscience*, *7*, 1751–1760.

Williams, R. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*, 229–256.

Xie, X., & Seung, S. (2003). Equivalence of backpropagation and contrastive Hebbian learning in a layered network. *Neural Computation*, *15*, 441–454.

Received August 20, 2017; accepted January 11, 2018.