Converting Cascade-Correlation Neural Nets into Probabilistic Generative Models

Ardavan S. Nobandegani[1,3] **Thomas R. Shultz**[2,3]
{ardavan.salehinobandegani@mail.mcgill.ca, thomas.shultz@mcgill.ca}
[1]Department of Electrical and Computer Engineering, McGill University
[2]School of Computer Science, McGill University
[3]Department of Psychology, McGill University

Abstract

Humans are not only adept in recognizing what class an input instance belongs to (i.e., classification task), but perhaps more remarkably, they can imagine (i.e., *generate*) plausible instances of a desired class with ease, when prompted. Inspired by this, we propose a framework which allows transforming Cascade-Correlation Neural Networks (CCNNs) into probabilistic generative models, thereby enabling CCNNs to generate samples from a category of interest. CCNNs are a well-known class of deterministic, discriminative NNs, which autonomously construct their topology, and have been successful in accounting for a variety of psychological phenomena. Our proposed framework is based on a Markov Chain Monte Carlo (MCMC) method, called the Metropolis-adjusted Langevin algorithm, which capitalizes on the gradient information of the target distribution to direct its explorations towards regions of high probability, thereby achieving good mixing properties. Through extensive simulations, we demonstrate the efficacy of our proposed framework. Importantly, our framework bridges computational, algorithmic, and implementational levels of analysis.

Keywords: Deterministic Discriminative Neural Networks; Probabilistic Generative Models; Markov Chain Monte Carlo

1 Introduction

A green-striped elephant! Probably no one has seen such a thing—no surprise. But what is a surprise is our ability to easily imagine one. Humans are not only adept in recognizing what class an input instance belongs to (i.e., classification task), but more remarkably, they can imagine (i.e., *generate*) plausible instances of a desired class, when prompted. In fact, humans can generate instances of a desired class, say, elephant, that they have never encountered before, like, a green-striped elephant.[1] In this sense, humans' generative capacity goes beyond merely retrieving from memory. In computational terms, the notion of generating examples from a desired class can be formalized in terms of *sampling* from some underlying probability distribution, and has been extensively studied in machine learning under the rubric of probabilistic generative models.

Cascade-Correlation Neural Networks (CCNNs) (Fahlman & Lebiere, 1989) are a well-known class of discriminative (as opposed to generative) models that have been successful in simulating a variety of phenomena in the developmental literature, e.g., infant learning of word-stress patterns in artificial languages (Shultz & Bale, 2006), syllable boundaries (Shultz & Bale, 2006), visual concepts (Shultz, 2006), and have also been successful in capturing important developmental regularities in a variety of tasks, e.g., the balance-scale task (Shultz, Mareschal, & Schmidt, 1994; Shultz & Takane, 2007), transitivity (Shultz & Vogel, 2004), conservation (Shultz, 1998), and seriation (Mareschal & Shultz, 1999). Also, CCNNs exhibit several similarities with known brain functions: distributed representation, self-organization of network topology, layered hierarchical topologies, both cascaded and direct pathways, an S-shaped activation function, activation modulation via integration of neural inputs, long-term potentiation, growth at the newer end of the network via synaptogenesis or neurogenesis, pruning, and weight freezing (Westermann, Sirois, Shultz, & Mareschal, 2006). Nonetheless, in virtue of being deterministic and discriminative, CCNNs have so far lacked the capacity to probabilistically generate examples from a category of interest.

In this work, we propose a framework which allows transforming CCNNs into probabilistic generative models, thereby enabling CCNNs to generate samples from a category. Our proposed framework is based on a Markov Chain Monte Carlo (MCMC) method, called the Metropolis-Adjusted Langevin (MAL) algorithm, which employs the gradient of the target distribution to guide its explorations towards regions of high probability, thereby significantly reducing the undesirable random walk often observed at the beginning of an MCMC run (a.k.a. the burn-in period). MCMC methods are a family of algorithms for sampling from a desired probability distribution, and have been successful in simulating important aspects of a wide range of cognitive phenomena, e.g., temporal dynamics of multistable perception (Gershman, Vul, & Tenenbaum, 2012; Moreno-Bote, Knill, & Pouget, 2011), developmental changes in cognition (Bonawitz, Denison, Griffiths, & Gopnik, 2014), category learning (Sanborn, Griffiths, & Navarro, 2010), causal reasoning in children (Bonawitz, Denison, Gopnik, & Griffiths, 2014), and accounting for many cognitive biases (Dasgupta, Schulz, & Gershman, 2016).

Furthermore, work in theoretical neuroscience has shed light on possible mechanisms according to which MCMC methods could be realized in generic cortical circuits (Buesing, Bill, Nessler, & Maass, 2011; Moreno-Bote et al., 2011; Pecevski, Buesing, & Maass, 2011; Gershman & Beck, 2016). In particular, Moreno-Bote et al. (2011) showed how an attractor neural network implementing MAL can account for multistable perception of drifting gratings, and Savin and Deneve (2014) showed how a network of leaky integrate-and-fire neurons can implement MAL in a biologically-realistic

[1] In *counterfactual* terms: Had a human seen a green-striped elephant, s/he would have yet recognized it as an elephant. Geoffrey Hinton once told a similar story about a pink elephant!

manner.

2 Cascade-Correlation Neural Networks

CCNNs are a special class of deterministic artificial neural networks, which construct their topology in an autonomous fashion—an appealing property simulating developmental phenomena (Westermann et al., 2006) and other cases where networks need to be constructed. CCNN training starts with a two-layer network (i.e., the input and the output layer) with no hidden units, and proceeds by recruiting hidden units one at a time, as needed. Each new hidden unit is trained to maximally correlate with residual error in the network built so far, and is recruited into a hidden layer of its own, giving rise to a deep network with as many hidden layers as the number of recruited hidden units. CCNNs use sum-of-squared error as an objective function, and typically use symmetric sigmoidal activation functions with range -0.5 to $+0.5$ for hidden and output units.[2] Some variants have been proposed: Sibling-Descendant Cascade-Correlation (SDCC) (Baluja & Fahlman, 1994) and Knowledge-Based Cascade-Correlation (KBCC) (Shultz & Rivest, 2001). Although in this work we focus on standard CCNNs, our proposed framework can handle SDCC and KBCC as well.

3 The Metropolis-Adjusted Langevin Algorithm

MAL (Roberts & Tweedie, 1996) is a special type of MCMC method, which employs the gradient of the target distribution to guide its explorations towards regions of high probability, thereby reducing the burn-in period. More specifically, MAL combines the two concepts of Langevin dynamics (a random walk guided by the gradient of the target distribution), and the Metropolis-Hastings algorithm (an accept/reject mechanism for generating a sequence of samples the distribution of which asymptotically converges to the target distribution).

We denote random variables with small bold-faced letters, random vectors by capital bold-faced letters, and their corresponding realizations by non-bold-faced letter. The MAL algorithm is outlined in Algorithm 1 wherein $\pi(\mathbf{X})$ denotes the target probability distribution, τ is a positive real-valued parameter specifying the time-step used in the Euler-Maruyama approximation of the underlying Langevin dynamics, N denotes the number of samples generated by the MAL algorithm, q denotes the proposal distribution (a.k.a. transition kernel), $\mathcal{N}(\mu,\Sigma)$ denotes the multivariate normal distribution with mean vector μ and covariance matrix Σ, and $\mathbb{I}$ denotes the identity matrix. The sequence of samples generated by the MAL algorithm, $\mathbf{X}^{(0)},\mathbf{X}^{(1)},\ldots$, is guaranteed to converge in distribution to $\pi(\mathbf{X})$ (Robert & Casella, 2013). It is worth noting that work in theoretical neuroscience has shown that MAL, outlined in Algorithm 1, can be implemented in a neurally-plausible manner (Savin & Deneve, 2014; Moreno-Bote et al., 2011).[3] In the following section, we propose a target distribution $\pi(\mathbf{X})$, allowing CCNNs to generate samples from a category of interest.

Algorithm 1 The Metropolis-Adjusted Langevin Algorithm

```
Input: Target distribution $\pi(\mathbf{X})$, parameter $\tau \in \mathbb{R}_+$, number of samples $N$.
Output: Samples $\mathbf{X}^{(0)},\ldots,\mathbf{X}^{(N-1)}$.
Pick $\mathbf{X}^{(0)}$ arbitrarily.
for $i = 0,\ldots,N-1$ do
Sample $\mathbf{u} \sim$ Uniform[0,1]
Sample $\mathbf{X}^* \sim q(\mathbf{X}^*|\mathbf{X}^{(i)}) = \mathcal{N}(\mathbf{X}^{(i)} + \tau\nabla\log\pi(\mathbf{X}^{(i)}), 2\tau\mathbb{I})$
    if $\mathbf{u} < \min\{1, \frac{\pi(\mathbf{X}^*)q(\mathbf{X}^{(i)}|\mathbf{X}^*)}{\pi(\mathbf{X}^{(i)})q(\mathbf{X}^*|\mathbf{X}^{(i)})}\}$ then
        $\mathbf{X}^{(i+1)} \leftarrow \mathbf{X}^*$
    else
        $\mathbf{X}^{(i+1)} \leftarrow \mathbf{X}^{(i)}$
    end if
end for
return $\mathbf{X}^{(0)},\ldots,\mathbf{X}^{(N-1)}$
```

4 The Proposed Framework

In what follows, we propose a framework which transforms CCNNs into probabilistic generative models, thereby enabling them to generate samples from a category of interest. The proposed framework is based on the MAL algorithm given in Sec. 3. Let $f(X;W^*)$ denote the input-output mapping learned by a CCNN, and W^* denote the set of weights for a CCNN after training.[4] Upon termination of training, presented with input X, a CCNN outputs $f(X;W^*)$. Note that, in case a CCNN possesses multiple output units, $f(X;W^*)$ will be a vector rather than a scalar. To convert a CCNN into a probabilistic generative model, we use the MAL algorithm with its target distribution $\pi(\mathbf{X})$ being set as follows:

$$\begin{aligned} \tilde{\pi}(\mathbf{X}) &\triangleq p(\mathbf{X}|\mathbf{Y}=L_j) \\ &= \frac{1}{Z}\exp(-\beta||L_j - f(\mathbf{X};W^*)||_2^2), \end{aligned} \tag{1}$$

where $||\cdot||_2$ denotes the l_2-norm, $\beta \in \mathbb{R}_+$ is a *damping factor*, Z is the normalizing constant, and L_j is a vector whose element corresponding to the desired class is $+0.5$ (i.e., its j^{th} element) and the rest of its elements are -0.5s. The intuition behind Eq. (1) can be articulated as follows: For an input instance $\mathbf{X} = X$ belonging to the desired class j,[5] the output of

[2] Fahlman and Lebiere (1989) also suggest linear, Gaussian, and asymmetric sigmoidal (with range 0 to $+1$) activation functions as alternatives. Our proposed framework can be straightforwardly adapted to handle all such activation functions.

[3] More precisely, it has been shown how the continuous-time version of MAL, Langevin dynamics, can be implemented in a neurally-plausible manner. But note that MAL amounts to sampling from the underlying Langevin dynamics.

[4] Formally, $f(\cdot;W^*) : \prod_{i=1}^{n} D_i \to \prod_{j=1}^{m} R_j$ where D_i and R_j denote the set of values that input unit i and output unit j can take on, respectively.

[5] In counterfactual terms, this is equivalent to saying: Had input instance X been presented to the network, it would have classified X in class j.

the network $f(X;W^*)$ is expected to be close to L_j in l_2-norm sense. In this light, Eq. (1) is adjusting the likelihood of input instance X to be inversely proportional to the base-e exponent of the said l_2 distance.

For a reader familiar with probabilistic graphical models, the expression in Eq. (1) looks similar to the expression for the joint probability distribution of Markov random fields and probabilistic energy-based models, e.g., Restricted Boltzman Machines and Deep Boltzman Machines. However, there is a crucial distinction: The normalizing constant Z, the computation of which is intractable in general, renders learning in those models computationally intractable.[6] The appropriate way to interpret Eq. (1) is to see it as a Gibbs distribution for a non-probabilistic energy-based model whose energy is defined as the square of the prediction error (LeCun, Chopra, Hadsell, Ranzato, & Huang, 2006). Section 1.3 of (LeCun et al., 2006) discusses the topic of Gibbs distribution for non-probabilistic energy-based models in the context of discriminitive learning, computationally modeled by $p(\mathbf{Y}|\mathbf{X})$ (i.e., to predict a class given an input), and raises the same issue that we highlighted above regarding the intractability of computing the normalizing constant Z in general. In sharp contrast to (LeCun et al., 2006), our framework is proposed for the purpose of generating examples from a desired class, as evidenced by Eq. (1) being defined in terms of $p(\mathbf{X}|\mathbf{Y})$. Also crucially, the intractability of computing Z raises no issue for our proposed framework due to an intriguing property of the MAL algorithm according to which the normalizing constant Z need not be computed at all.[7]

Due to Line 4 of Algorithm 1, MAL's proposal distribution q requires the computation of $\nabla \log \tilde{\pi}(\mathbf{X}^{(i)})$, which essentially involves computing $\nabla f(\mathbf{X}^{(i)};W^*)$ (note that the gradient is operating on $\mathbf{X}^{(i)}$, and W^* is treated as a set of fixed parameters). The multi-layer structure of CCNN ensures that $\nabla f(\mathbf{X}^{(i)};W^*)$ can be efficiently computed using Backpropagation. Alternatively, in settings where CCNNs recruit a small number of input units (hence, the cardinality of $\mathbf{X}^{(i)}$ is small), $\nabla f(\mathbf{X}^{(i)};W^*)$ can be obtained by introducing negligible perturbation to a component of input signal $\mathbf{X}^{(i)}$, dividing the resulting change in the network's outputs by the introduced perturbation, and repeating this process for all components of input signal $\mathbf{X}^{(i)}$. It is worth noting that although the idea of computing gradients through introducing small perturbations would lead to a computationally inefficient approach for *learning* CCNNs, it leads to a computationally efficient approach for *generation*, as the number of input units are typically much fewer than the number of weights in CCNNs (and artificial neural networks in general). It is crucial to note that the normalizing constant Z plays no role in the computation of $\nabla \log \tilde{\pi}(\mathbf{X}^{(i)})$.

5 Simulations

In this section we demonstrate the efficacy of our proposed framework through simulations. We particularly focus on learning which can be accomplished by two input and one output units. This permits visualization of the input-output space, which lies in $\mathbb{R}^3$. Note that our proposed framework can handle arbitrary number of input and output units; this restriction is solely for ease of visualization.

5.1 Continuous-XOR Problem

In this section, we show how our proposed framework allows a CCNN, trained on the continuous-XOR classification task, to generate examples from a category of interest. The output unit has a symmetric sigmoidal activation function with range -0.5 and $+0.5$. The training set consists of 100 samples in the unit-square $[0,1]^2$, paired with their corresponding labels. More specifically, the training set is comprised of all the ordered-pairs starting from $(0.1,0.1)$ and going up to $(1,1)$ with equal steps of size 0.1, paired with their corresponding labels (i.e., $+0.5$ for positive samples and -0.5 for negative samples); see Fig. 1(top-left). After training, a CCNN with 6 hidden layers is obtained whose input-output mapping, $f(x_1,x_2;W^*)$, is shown in Fig. 1(top-right).[8]

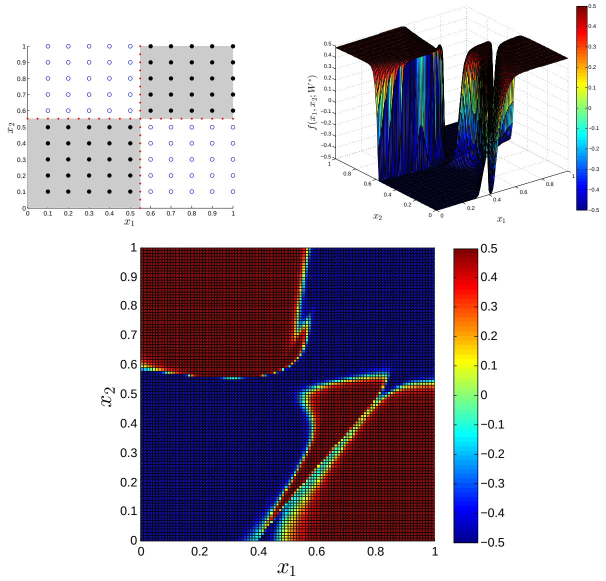
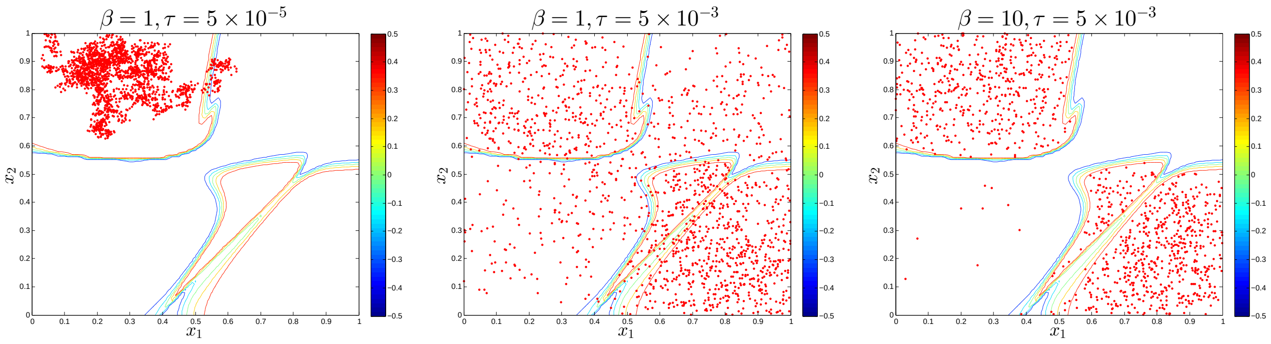


Figure 1: A CCNN trained on the continuous-XOR classification task. Top-left: Training patterns. All the patterns in the gray quadrants are negative examples with label -0.5, and all the patterns in the white quadrants are positive examples with label $+0.5$. Red dotted lines depict the boundaries. Top-right: The input-output mapping, $f(x_1,x_2;W^*)$, learned by a CCNN, along with a colorbar. Bottom: The top-down view of the curve depicted in top-right, along with a colorbar.

[6] More specifically, Z renders the computation of the gradient of the log-likelihood for those models intractable.

[7] The MAL algorithm inherits this property from the Metropolis-Hasting algorithm, which it uses as a subroutine.

[8] Due to the inherent randomness in CCNN construction, training could lead to networks with different structures. However, since in this work we are solely concerned with generating examples using



(a) $N = 2000, AR = 99.55\%$ (b) $N = 2000, AR = 75.25\%$ (c) $N = 2000, AR = 57.85\%$

Figure 2: Generating example for the positive category, under various choices for MAL parameter τ and damping factor β. Contour-plot of the learned mapping, $f(x_1, x_2; W^*)$, along with its corresponding colorbar is shown in each sub-figure. Generated samples are depicted by red dots. N denotes the total number of samples generated by MAL, and AR denotes the corresponding acceptance rate. **(a)** $\tau = 5 \times 10^{-5}$ leads to a very slow exploration of the input space. **(b)** $\tau = 5 \times 10^{-3}$ leads to an adequate exploration of the input space, however, $\beta = 1$ is not penalizing undesirable input regions severely enough. **(c)** A desirable performance is achieved by $\tau = 5 \times 10^{-3}$ and $\beta = 10$.

Fig. 2 shows the efficacy of our proposed framework in enabling CCNNs to generate samples from a category of interest, under various choices for MAL parameter τ (see Algorithm 1) and damping factor β (see Eq. (1)); generated samples are depicted by red dots. For the results shown in Fig. 2, the category of interest is the category of positive examples, i.e., the category of input patterns which, upon being presented to the (learned) network, would be classified as positive by the network. Because τ controls the amount of jump between consecutive proposals made by MAL, the following behavior is expected: For small τ (Fig. 2(a)) consecutive proposals are very close to one another, leading to a slow exploration of the input domain. As τ increases, bigger jumps are made by MAL (Fig. 2(b)).[9] Parameter β controls how severely deviations from the desired class label (here, $+0.5$) are penalized. The larger the parameter β, the more severely such deviations are penalized and the less likely MAL moves toward such regions of input space. Acceptance Rate (AR), defined as the number of accepted moves divided by the total number of suggested moves, is also presented for the results shown in Fig. 2. Fig. 2(c) shows that for $\tau = 5 \times 10^{-3}$ and $\beta = 10$, our proposed framework demonstrates desirable performance: virtually all of the generated samples fall within the desired input regions (i.e., the regions associated with hot colors, signaling the closeness of network's output to $+0.5$ in those regions; see Fig. 1(bottom)) and the desired regions are adequately explored (i.e., all hot-colored input regions being visited and almost evenly explored).

Fig. 2 depicts all the first $N = 2000$ samples generated by MAL, without excluding the so-called burn-in period. In that light, the result shown in Fig. 2(c) nicely demonstrates how MAL—by directing its suggestions toward the direction of gradient and therefore moving toward regions with high likelihood—could alleviate the need for discarding a (potentially large) number of samples generated at the beginning of an MCMC which are assumed to be unrepresentative of equilibrium state, a.k.a. the burn-in period. Fig. 3 shows the performance of our framework in enabling the learned CCNN to generate from the category of negative examples, with $\tau = 5 \times 10^{-3}$ and $\beta = 10$.

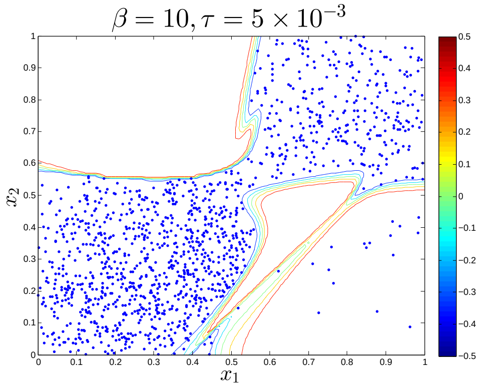


Figure 3: Generating example for the negative category, with $\tau = 5 \times 10^{-3}, \beta = 10$. Generated samples are shown by blue dots. Total number of samples generated is $N = 2000$, with $AR = 65.13\%$.

5.2 Two-Spirals Problem

Next, we show how our proposed framework allows a CCNN, trained on the famously difficult two-spirals classification task (Fig. 4), to generate examples from a category of interest. The output unit has a symmetric sigmoidal activation

CCNNs rather than how well CCNNs could learn a given discriminitive task, we arbitrarily pick a learned network. Note that our proposed framework can handle CCNNs with arbitrary structures; in that light, the choice of network is without loss of generality.

[9] Yet, too large a β is not good either, leading to a sparse and coarse-grained exploration of the input space. Some measures have been proposed in computational statistics for properly choosing τ; cf. (Roberts & Rosenthal, 1998).

function with range -0.5 and $+0.5$. The training set consists of 194 samples (97 samples per spiral), in the square $[-6.5, 6.5]^2$, paired with their corresponding labels ($+0.5$ and -0.5 for positive and negative samples, respectively). The training patterns are shown in Fig. 4(top-left); cf. (Chalup & Wiklendt, 2007) for details. After training, a CCNN with 14 hidden layers is obtained whose input-output mapping, $f(x_1, x_2; W^*)$, is depicted in Fig. 4(top-right).

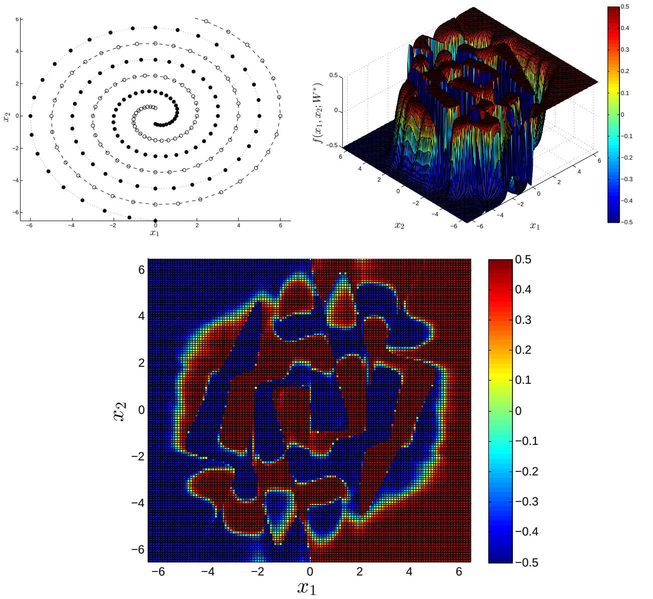


Figure 4: A CCNN trained on the two-spirals classification task. Top-left: Training patterns. Positive patterns (associated with label $+0.5$) are shown by hollow circles, and negative patterns (associated with label -0.5) by black circles. Positive spiral is depicted by a dashed line, and negative spiral by a dotted line. Top-right: The input-output mapping, $f(x_1, x_2; W^*)$, learned by a CCNN, along with a colorbar. Bottom: The top-down view of the curve depicted in top-right, along with a colorbar.

Fig. 5(top) and Fig. 5(bottom) show the efficacy of our proposed framework in enabling CCNNs to generate samples from the positive and negative categories, respectively. Although similar patterns of behavior observed in Sec. 5.1 due to increasing/decreasing β and τ are observed here as well, due to the lack of space such results are omitted. The results in Fig. 5 depict all the first $N = 15000$ samples generated by MAL, without excluding the burn-in period. In that light, these results again demonstrate the efficacy of MAL in alleviating the need for discarding a (potentially large) number samples generated at the beginning of an MCMC run.

Interestingly, our proposed framework also allows CCNNs to generate samples subject to some forms of constraints. For example, Fig. 6 demonstrates how our proposed framework enables a CCNN, trained on the continuous-XOR classification task (see Sec. 5.1), to generate examples from the positive category, under the following constraint: Generated samples must lie on the curve $x_2 = 0.25\sin(8\pi x_1) + 0.5$. To generate samples from the positive category while satisfying this constraint, MAL adopts our proposed target distribution given in Eq. (1), and treats x_1 as an independent and x_2 as a dependent variable.

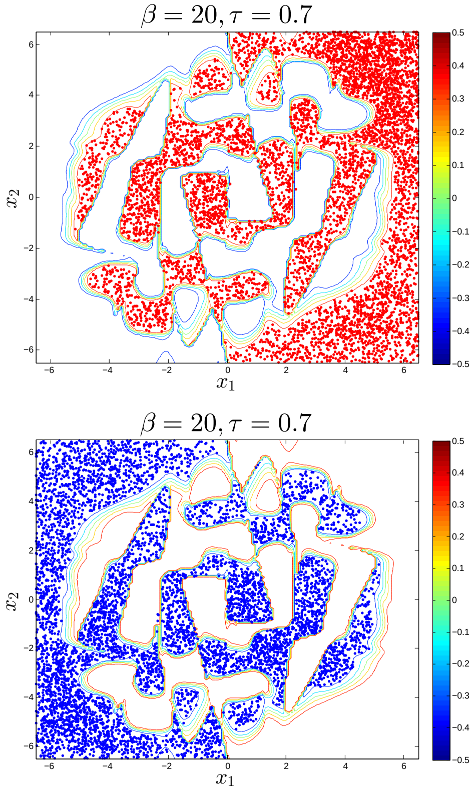


Figure 5: Generating example for the positive and negative categories, with $\beta = 20$ and $\tau = 0.7$. Contour-plot of the learned mapping, $f(x_1, x_2; W^*)$, along with its corresponding colorbar is shown in each sub-figure. N denotes the total number of samples generated by MAL, and AR denotes the corresponding acceptance rate. Top: Generated example for the positive category, with $N = 15000$ and $AR = 40.69\%$; generated samples are depicted by red dots. Bottom: Generated example for the negative category, with $N = 15000$ and $AR = 40.28\%$; generated samples are depicted by blue dots.

6 General Discussion

Although we discussed our proposed framework in the context of CCNNs, it can be straightforwardly extended to handle some other kinds of artificial neural networks, e.g. Multilayer Perceptron and Deep Convolutional Neural Networks. Furthermore, our proposed framework, together with recent work in theoretical neuroscience showing possible neurally-plausible implementations of MAL (Savin & Deneve, 2014; Moreno-Bote et al., 2011), suggests an intriguing modular hypothesis according to which generation could result from two separate modules interacting with each other (in our case, a CCNN and a neural network implementing MAL). This

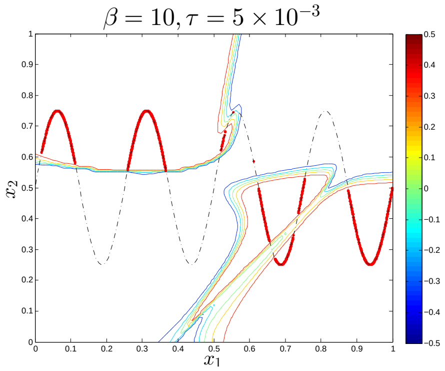


Figure 6: Generating examples for the positive category, under constraint $x_2 = 0.25\sin(8\pi x_1) + 0.5$ (dash-dotted curve), with $N = 5000$ and $AR = 39.82\%$. Contour-plot of the learned mapping, $f(x_1, x_2; W^*)$, along with its corresponding colorbar is depicted. Generated samples are shown by red dots, which appear mainly as solid red curves due to high density.

hypothesis yields the following prediction: There should be some brain impairments which lead to a marked decline in a subject's performance in generative tasks (i.e., tasks involving imagery, or imaginative tasks in general) but leave the subject's learning abilities (nearly) intact. Studies on learning and imaginative abilities of hippocampal amnesic patients already provide some supporting evidence for this idea (Hassabis, Kumaran, Vann, & Maguire, 2007; Spiers, Maguire, & Burgess, 2001; Brooks & Baddeley, 1976).

According to Line 4 of Algorithm 1, to generate the i^{th} sample, MAL requires access to a fine-tuned, Gaussian noise with mean $\mathbf{X}^{(i)} + \tau\nabla \log \pi(\mathbf{X}^{(i)})$ for its proposal distribution q. Recently Savin and Deneve (2014) showed how a network of leaky integrate-and-fire neurons can implement MAL in a neurally-plausible manner. However, as Gershman and Beck (2016) point out, Savin and Deneve leave unanswered what the source of that fine-tuned Gaussian noise could be. Our proposed framework may provide an explanation, not for the source of Gaussian noise, but for its fine-tuned mean value. According to our modular account, the main component of the mean value, which is $\nabla \log \pi(\mathbf{X}^{(i)})$, may come from another module (in our case, a CCNN) which has learned some input-output mapping $f(X; W^*)$, based on which the target distribution $\pi(\mathbf{X}^{(i)})$ is defined (see Eq. (1)).

The idea of sample generation under constraints could be an interesting line of future work. Humans clearly have the capacity to engage in imaginative tasks under a variety of constraints, e.g., when given incomplete sentences or fragments of a picture people can generate possible completions (Sanborn & Chater, 2016). Also, our proposed framework can be used to let a CCNN generate samples from a category of interest at any stage during CCNN construction. In that light, our proposed framework, along with a neurally-plausible implementation of MAL, gives rise to a *self-organized generative model*: a generative model possessing the self-constructive property of CCNNs. Such self-organized generative models could provide a wealth of developmental hypotheses as to how the imaginative capacities of children change over development, and models with quantitative predictions to compare against. We see our work as a step towards such models. Last but not least, our framework strongly suggests that, contrary to conventional wisdom, the boundary between discriminative and generative models is blurry—perhaps they are just two sides of the same coin!

Acknowledgments. This work is funded by an operating grant to TRS from the Natural Sciences and Engineering Research Council of Canada. We would like to thank Kevin Da Silva Castanheira for helpful comments on an earlier draft of this work.

References

Baluja, S., & Fahlman, S. E. (1994). Reducing network depth in the cascade-correlation learning architecture. Technical Report # CMU-CS-94-209, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA..

Bonawitz, E., Denison, S., Gopnik, A., & Griffiths, T. L. (2014). Win-stay, lose-sample: A simple sequential algorithm for approximating bayesian inference. *Cognitive Psychology*, *74*, 35–65.

Bonawitz, E., Denison, S., Griffiths, T. L., & Gopnik, A. (2014). Probabilistic models, learning algorithms, and response variability: sampling in cognitive development. *Trends in Cognitive Sciences*, *18*(10), 497–500.

Brooks, D., & Baddeley, A. (1976). What can amnesic patients learn? *Neuropsychologia*, *14*(1), 111–122.

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 Comput Biol*, *7*(11), e1002211.

Chalup, S. K., & Wiklendt, L. (2007). Variations of the two-spiral task. *Connection Science*, *19*(2), 183–199.

Dasgupta, I., Schulz, E., & Gershman, S. J. (2016). Where do hypotheses come from? Center for Brains, Minds and Machines (CBMM) Memo No. 056.

Fahlman, S. E., & Lebiere, C. (1989). The cascade-correlation learning architecture. In *Adv. in Neural Information Processing Systems*, pp. 524-532.

Gershman, S. J., & Beck, J. M. (2016). Complex probabilistic inference: From cognition to neural computation. In *Computational Models of Brain and Behavior,* ed A. Moustafa (Hoboken, NJ: Wiley-Blackwell).

Gershman, S. J., Vul, E., & Tenenbaum, J. B. (2012). Multistability and perceptual inference. *Neural Computation*, *24*(1), 1–24.

Hassabis, D., Kumaran, D., Vann, S. D., & Maguire, E. A. (2007). Patients with hippocampal amnesia cannot imagine new experiences. *Proceedings of the National Academy of Sciences*, *104*(5), 1726–1731.

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., & Huang, F. (2006). A tutorial on energy-based learning. *Predicting Structured Data*, *1*, 0.

Mareschal, D., & Shultz, T. R. (1999). Development of children's seriation: A connectionist approach. *Connection Science*, *11*(2), 149–186.

Moreno-Bote, R., Knill, D. C., & Pouget, A. (2011). Bayesian sampling in visual perception. *Proceedings of the National Academy of Sciences*, *108*(30), 12491–12496.

Pecevski, D., Buesing, L., & Maass, W. (2011). Probabilistic inference in general graphical models through sampling in stochastic networks of spiking neurons. *PLoS Comput Biol*, *7*(12), e1002294.

Robert, C., & Casella, G. (2013). *Monte Carlo statistical methods*. Springer Science & Business Media.

Roberts, G. O., & Rosenthal, J. S. (1998). Optimal scaling of discrete approximations to langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *60*(1), 255–268.

Roberts, G. O., & Tweedie, R. L. (1996). Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli*, 341–363.

Sanborn, A. N., & Chater, N. (2016). Bayesian brains without probabilities. *Trends in Cognitive Sciences*, *20*(12), 883–893.

Sanborn, A. N., Griffiths, T. L., & Navarro, D. J. (2010). Rational approximations to rational models: alternative algorithms for category learning. *Psychological Review*, *117*(4), 1144.

Savin, C., & Deneve, S. (2014). Spatio-temporal representations of uncertainty in spiking neural networks. In *Adv. in Neural Information Processing Systems*.

Shultz, T. R. (1998). A computational analysis of conservation. *Developmental Science*, *1*(1), 103–126.

Shultz, T. R. (2006). Constructive learning in the modeling of psychological development. *Processes of Change in Brain and Cognitive Development: Attention and Performance*, *21*, 61–86.

Shultz, T. R., & Bale, A. C. (2006). Neural networks discover a near-identity relation to distinguish simple syntactic forms. *Minds and Machines*, *16*(2), 107–139.

Shultz, T. R., Mareschal, D., & Schmidt, W. C. (1994). Modeling cognitive development on balance scale phenomena. *Machine Learning*, *16*(1-2), 57–86.

Shultz, T. R., & Rivest, F. (2001). Knowledge-based cascade-correlation: Using knowledge to speed learning. *Connection Science*, *13*(1), 43–72.

Shultz, T. R., & Takane, Y. (2007). Rule following and rule use in the balance-scale task. *Cognition*, *103*(3), 460–472.

Shultz, T. R., & Vogel, A. (2004). A connectionist model of the development of transitivity. In *Proceedings of the 26th Annual Conference of the Cognitive Science Society* (pp. 1243–1248).

Spiers, H. J., Maguire, E. A., & Burgess, N. (2001). Hippocampal amnesia. *Neurocase*, *7*(5), 357–382.

Westermann, G., Sirois, S., Shultz, T. R., & Mareschal, D. (2006). Modeling developmental cognitive neuroscience. *Trends in Cognitive Sciences*, *10*(5), 227–232.