Deep Switch Networks for Generating Discrete Data and Language

Payam Delgosha
University of California, Berkeley
pdelgosha@eecs.berkeley.edu

Naveen Goela
Tanium, Data Science
ngoela@alum.mit.edu

Abstract

Multilayer switch networks are proposed as artificial generators of high-dimensional discrete data (e.g., binary vectors, categorical data, natural language, network log files, and discrete-valued time series). Unlike deconvolution networks which generate continuous-valued data and which consist of upsampling filters and reverse pooling layers, multilayer switch networks are composed of adaptive switches which model conditional distributions of discrete random variables. An interpretable, statistical framework is introduced for training these nonlinear networks based on a maximum-likelihood objective function. To learn network parameters, stochastic gradient descent is applied to the objective, and is stable until convergence. This direct optimization does not involve back-propagation over separate encoder and decoder networks, or adversarial training of dueling networks. While training remains tractable for moderately sized networks, Markov-chain Monte Carlo (MCMC) approximations of gradients are derived for deep networks which contain latent variables. The statistical framework is evaluated on synthetic data, high-dimensional binary data of handwritten digits, and web-crawled natural language data. Aspects of the model's framework such as interpretability, computational complexity, and generalization ability are discussed.

1 Introduction

Several deep generative models have been proposed in the literature, including stacked Restricted Boltz-mann Machines (RBMs), pixel convolutional neural networks (pixel CNNs), recurrent neural networks (RNNs), variational auto-encoders (VAEs), and generative adversarial networks (GANs). Many generative models produce continuous-valued data such as images. The present paper establishes a statistical model for generating high-dimensional discrete data. To learn discrete distributions, higher-order multivariate and long-range dependencies must be modeled effectively. A smooth interpolation between discrete samples is not guaranteed. For this reason, we propose an adaptive switch, instead of a filter, as the fundamental element of a generative model.

1.1 Generative Modeling

RBMs: Discrete samples can be generated by latent-variable models such as RBMs using block Gibbs sampling. RBMs may be trained efficiently via contrastive divergence learning procedures (Hinton, 2002). Although training more sophisticated multilayer RBMs is feasible (Salakhutdinov and Hinton, 2009), most approaches do not result in simple statistical models. The log-likelihood, even for single-layer RBMs, must be approximated due to the intractability of the partition function (Tieleman, 2008).

Pixel-CNNs, RNNs, Language Models: Several generative models of sequences involve optimizing convolutional filters via back-propagation of gradients. Pixel-CNNs are defined by filter transformations tailored for continuous-valued data (van den Oord et al., 2016). To harness existing training methods for discrete-valued data, language models represent each word in a vocabulary by a real-valued embedding vector (Bengio et al., 2003; Jozefowicz et al., 2016; Shazeer et al., 2017). Embedding vectors may not be applicable to all discrete distributions. Many language models also view the generation of data conditioned on past variables as a problem of prediction or classification. In the absence of log-likelihoods, perplexity scores and performance indicators from downstream tasks serve as standards for assessment.

Latent-Variable Models: VAEs and variational RNNs specify joint distributions over observed and latent variables (Kingma and Welling, 2014; Chung et al., 2015). The posterior conditional density over latent variables is approximated using variational inference (Blei et al., 2017). Training VAEs involves the optimization of both encoder and decoder networks, approximations such as the evidence lower bound (ELBO), and the reparameterization trick for gradients which usually implies continuous latent variables. VAEs have been modified to include discrete latent variables and components (van den Oord et al., 2017; Rolfe, 2017; Vahdat et al., 2018).

GANs: Generative adversarial networks are optimized by adversarial training between a generator and a critic network (Goodfellow et al., 2014). More stable variants have been devised, including the Wasserstein GAN (Salimans et al., 2016; Arjovsky et al., 2017). Progressively-trained GANs have generated images with improved quality, resolution, and diversity (Karras et al., 2018). Modifying GAN architectures for producing discrete data has yielded partial successes; e.g., the maximum-likelihood augmented GAN (Che et al., 2017) and boundary-seeking GAN (Hjelm et al., 2018). Several theoretical questions regarding the generalization and equilibrium of GANs have been studied, including whether the trained distribution is close to the target, and whether mode collapse can be prevented (Arora et al., 2017; Arora et al., 2018).

1.2 Deep Switch Networks

The present paper introduces nonlinear switching mechanisms as a means for learning discrete distributions. The switch model is simple, interpretable, and easily trained via a direct maximum-likelihood optimization. The training of deep switch networks which contain latent variables is feasible due to MCMC gradient approximations. We note that highway networks (Srivastava et al., 2015; Zilly et al., 2017), and maxout networks (Goodfellow et al., 2013) contain network elements similar to adaptive switches. However, to the best of our knowledge, deep switch networks have not been studied previously for generative modeling of discrete data.

2 Modeling Discrete Distributions

The joint distribution of a vector of discrete random variables $X_{[1:n]} \equiv [X_1, X_2, \ldots, X_n]^T$ can be decomposed into conditional distributions via the chain rule,

$$p_{X_{[1:n]}}(x_{[1:n]}) = \prod_{k=0}^{n-1} p_{X_{k+1}|X_{[1:k]}}\left(x_{k+1}|x_{[1:k]}\right). \quad (1)$$

Modeling the joint distribution in (1) implies modeling successive conditional distributions. Each conditional distribution captures higher-order interactions between random variables.

2.1 Limitations of Sigmoid Functions

A parametric model for each conditional distribution in (1) could be the logistic or sigmoid function. However, this model is too simplistic. Consider the following elementary example. Let $\{X_1, X_2\}$ denote two independent Bernoulli $(1/2)$ random variables. Let $X_3 = X_1 \oplus X_2$ denote a third binary random variable, which introduces higher-order dependencies. The XOR operation $\oplus$ represents addition modulus 2. Consider a parametric model for the conditional distribution $p_{X_3|X_1,X_2}(x_3|x_1, x_2)$,

$$p_{X_3|X_1,X_2}(1|x_1, x_2) = \sigma(w_1 x_1 + w_2 x_2 + b). \quad (2)$$

For this specific example, if parameters $\{w_1, w_2, b\}$ are learned via maximum-likelihood estimation, it is impossible to achieve a likelihood greater than $1/2$.

To see this, note that the logistic form of (2) represents a linear separator in $\mathbb{R}^2$ in a probabilistic sense (Figure 1). The half-plane $w_1 x_1 + w_2 x_2 + b > 0$ represents points (x_1, x_2) that are more likely to generate the event $X_3 = 1$, and the half-plane $w_1 x_1 + w_2 x_2 + b < 0$ represents points (x_1, x_2) that are more likely to generate the event $X_3 = 0$. In order to model the deterministic function $X_3 = X_1 \oplus X_2$ correctly, the linear separator must partition the space of integral lattice points $\{(0,0), (1,1), (1,0), (0,1)\}$ correctly. Illustrated in blue, lattice points $\{(0,0), (1,1)\}$ should imply a higher likelihood of generating the event $X_3 = 0$. Illustrated in red, lattice points $\{(1,0), (0,1)\}$ should imply a higher likelihood of generating the event $X_3 = 1$. However, no linear separator in $\mathbb{R}^2$ is able to differentiate between the blue and red sets of lattice points.

We note that the insufficiency of a single linear separator has been observed for binary logistic regression for classification tasks, thereby motivating the use of deep neural networks. However, the context of our analysis is the statistical modeling of discrete distributions, by maximizing the likelihood of the data given the model.

2.2 A Nonlinear, Adaptive Switch

We construct a statistical model which utilizes a nonlinear, adaptive switch. This model is designed to be simple enough to train efficiently. For the example of Figure 1, consider *two* linear separators in $\mathbb{R}^2$. Let Y_j, $j \in \{1, 2\}$, denote two binary random variables. Their conditional distributions are specified as follows:

$$p_{Y_j|X_1,X_2}(1|x_1, x_2) = \sigma\left(w_1^{(j)} x_1 + w_2^{(j)} x_2 + b^{(j)}\right).$$

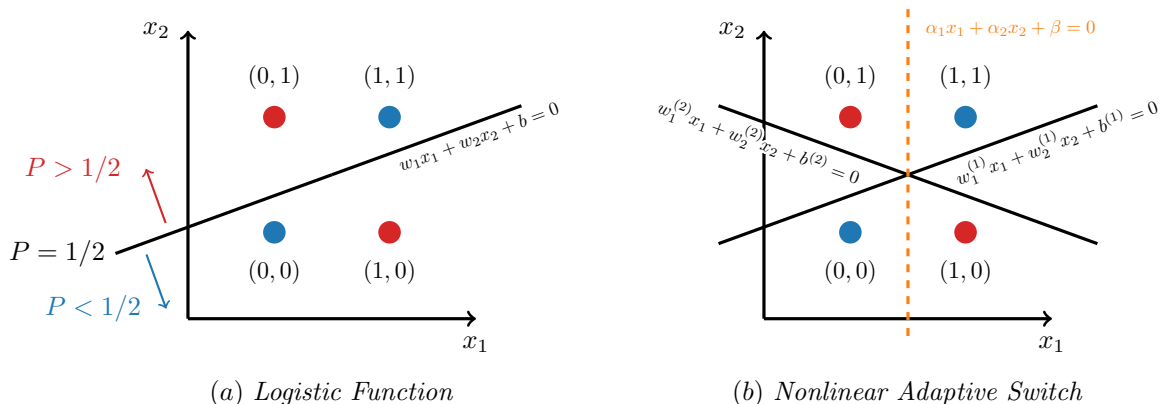(a) Logistic Function   (b) Nonlinear Adaptive Switch

Figure 1: *Elementary example:* Let $\{X_1, X_2\}$ denote two independent Bernoulli(1/2) random variables, and let $X_3 = X_1 \oplus X_2$. Towards the goal of modeling the joint distribution of $\{X_1, X_2, X_3\}$, we model the conditional distribution $p_{X_3|X_1,X_2}(x_3|x_1, x_2)$. (a) If the conditional is modeled by a single logistic function, the corresponding linear separator partitions $\mathbb{R}^2$ into two half-planes. Accurate modeling requires distinguishing the set of red points from the set of blue points, which is impossible with a linear separator (Section 2.1). (b) An adaptive switch is optimal for generative modeling. For this specific example, the adaptive switch itself is modeled as a linear separator, $\alpha_1 x_1 + \alpha_2 x_2 + \beta = 0$ (orange dashed line). Subject to the input configuration $(x_1, x_2) \in \mathbb{R}^2$, the switch separates left and right half-planes of $\mathbb{R}^2$, and subsequently selects one of two linear separators (black lines) to correctly distinguish between the upper and lower lattice points (Section 2.2).

Auxiliary variables Y_j have corresponding linear separators in $\mathbb{R}^2$ which depend on the input configuration (x_1, x_2). By selecting either Y_1 or Y_2 based on the input configuration (x_1, x_2), it is possible to significantly improve the modeling of $p_{X_3|X_1,X_2}(x_3|x_1, x_2)$. More precisely, consider a discrete (non-binary) random variable Z taking values in $\{1, 2\}$, which represents an adaptive switch. The switch also depends on the input configuration (x_1, x_2). This switch can itself be modeled by a logistic function[1] as follows:

$$p_{Z|X_1,X_2}(1|x_1, x_2) = \sigma(\alpha_1 x_1 + \alpha_2 x_2 + \beta).$$

Note that $p_{Z|X_1,X_2}(2|x_1, x_2) = 1 - p_{Z|X_1,X_2}(1|x_1, x_2)$. Random variable Y_Z selects either Y_1 or Y_2 adaptively. An adaptive switch improves upon the limited sigmoid function of (2), and models the target conditional distribution as follows,

$$p_{X_3|X_1,X_2}(1|x_1, x_2)$$
$$= \sum_{j=1,2} p_{Z|X_1,X_2}(j|x_1, x_2) p_{Y_j|X_1,X_2}(1|x_1, x_2).$$

This switch can achieve a maximum likelihood of nearly 1 after stochastic gradient optimization. As illustrated in Figure 1, there exist three separator

---

[1]The logistic function is not required for this elementary example of a switch. However, we introduce it to be consistent with the softmax function used to model adaptive switches in multilayer networks.

lines for this adaptive switch. The slanted black lines indicate the two separators for Y_j, $j \in \{1, 2\}$, and the dashed orange line indicates the separator for the switch variable Z. A likelihood of nearly 1 is obtained as a result of the switch variable Z correctly distinguishing between the two half-planes corresponding to $X_1 = 0$ and $X_1 = 1$. Within each half-plane, the slanted black lines correctly distinguish between the upper and lower integral lattice points.

3 Network Architecture

The example of Section 2.2 can be generalized to include switching between multiple linear separators in high-dimensional spaces. Experiments show that the adaptive switch is sufficient for non-trivial representation of discrete data. Furthermore, we devise deep network architectures for learning high-dimensional discrete distributions. Deep switch networks exhibit an improved performance in generative modeling compared to single-layer networks.

3.1 Single-layer Networks

To model the distribution of X_{k+1} conditioned on $X_{[1:k]}$, we define m binary auxiliary random variables $Y_1^{(k)}, \ldots, Y_m^{(k)}$. The conditional distribution of these auxiliary random variables is specified in parametric

form as follows. For $1 \leq j \leq m$,

$$p\big(Y_j^{(k)} = 1 \big| x_{[1:k]}; \theta^{(k)}\big) = \sigma\big(x_{[1:k]}^T w_{Y_j^{(k)}} + b_{Y_j^{(k)}}\big). \quad (3)$$

Each auxiliary variable represents a linear separator in the probabilistic sense as discussed in Section 2.2. A single-layer switch network chooses between these m linear separators in high-dimensional space, thereby partitioning sets of integral lattice points. In (3), $\theta^{(k)}$ signifies all model parameters for the k-th conditional distribution. The number of parameters to learn for m auxiliary variables for the k-th conditional is $\mathcal{O}(mk)$. Specifically, these parameters are

$$w_{Y_j^{(k)}} \in \mathbb{R}^k, \ b_{Y_j^{(k)}} \in \mathbb{R}, \text{for } 1 \leq j \leq m.$$

Since $0 \leq k \leq n - 1$ for all conditionals, the total number of parameters is $\mathcal{O}(mn^2)$. For m a constant, we note that this complexity in parameters to learn is similar to that of a fully-connected single-layer neural network, and similar to that of RBMs with an equivalent number of hidden latent variables.

3.2 Statistical Model of Adaptive Switching

To model the k-th conditional, we define a non-binary switch random variable, $Z^{(k)}$, which takes values in the set $\{1, \ldots, m\}$. The switch random variable selects a particular linear separator out of m separators. The switch value $Z^{(k)}$ is determined in an adaptive fashion based on the value of the input variables $X_{[1:k]}$. The switching mechanism is modeled by a softmax distribution. More precisely, for $1 \leq j \leq m$,

$$p\left(Z^{(k)} = j \big| x_{[1:k]}; \theta^{(k)}\right)$$
$$= \frac{\exp\left(x_{[1:k]}^T \alpha_{Z^{(k)}}^{(j)} + \beta_{Z^{(k)}}^{(j)}\right)}{\displaystyle\sum_{j'=1}^{m} \exp\left(x_{[1:k]}^T \alpha_{Z^{(k)}}^{(j')} + \beta_{Z^{(k)}}^{(j')}\right)}. \quad (4)$$

In (4), $\theta^{(k)}$ signifies all model parameters for the k-th conditional distribution. These parameters include those for the switch defined above, namely,

$$\alpha_{Z^{(k)}}^{(j)} \in \mathbb{R}^k, \ \beta_{Z^{(k)}}^{(j)} \in \mathbb{R}, \text{for } 1 \leq j \leq m.$$

Thus, the adaptive switch also requires learning of $\mathcal{O}(mk)$ parameters for the k-th conditional, and a total of $\mathcal{O}(mn^2)$ parameters for all conditional distributions.

3.3 Log-Likelihood Objective

Based on the defined auxiliary and switch random variables, the k-th target conditional distribution is modeled by selecting one of the variables $Y_1^{(k)}, \ldots, Y_m^{(k)}$
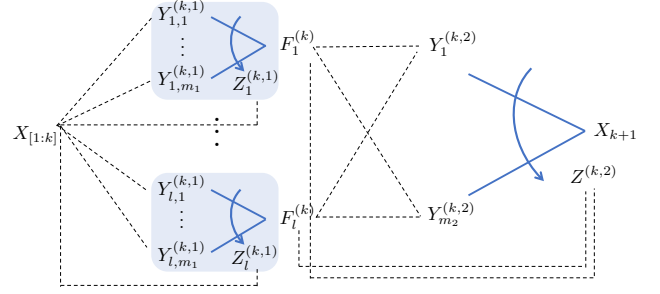

Figure 2: *The architecture of two-layer switch networks:* We have a total of l intermediate variables $F_1^{(k)}, \ldots, F_l^{(k)}$, each of which is the output of a single-layer switch network. Subsequently, these variables are fed into another single-layer switch network to generate the final output X_{k+1}. Dotted lines indicate dependencies between variables.

based on the value of $Z^{(k)}$. More precisely, utilizing the equation terms written in (3) and (4),

$$p\big(X_{k+1} = 1 | x_{[1:k]}; \theta^{(k)}\big) =$$
$$\sum_{j=1}^{m} p\big(Z^{(k)} = j \big| x_{[1:k]}; \theta^{(k)}\big) p\big(Y_j^{(k)} = 1 \big| x_{[1:k]}; \theta^{(k)}\big). \quad (5)$$

We optimize the model parameters $\theta^{(1)}, \ldots, \theta^{(n-1)}$ in order to maximize the empirical log likelihood. More precisely, consider I data samples $\vec{x}^{(1)}, \ldots, \vec{x}^{(I)}$ obtained from the true distribution. Each individual data sample is n-dimensional, $\vec{x}^{(i)} = (x_1^{(i)}, \ldots, x_n^{(i)})$. For $0 \leq k \leq n - 1$, we optimize for the following empirical log-likelihood,

$$L^{(k)} := \frac{1}{I} \sum_{i=1}^{I} \log p\left(X_{k+1} = x_{k+1}^{(i)} \big| x_{[1:k]}^{(i)}; \theta^{(k)}\right). \quad (6)$$

The conditional probabilities have the specific form given in (5), and the log-likelihood is differentiable. Thus, stochastic gradient descent allows for efficient training. Note that the optimization can be done in a distributed fashion in parallel. The objectives $L^{(1)}, \ldots, L^{(n-1)}$ can be optimized independently in order to find the optimal parameters $\theta^{(1)}, \ldots, \theta^{(n-1)}$.

4 Multilayer Switch Networks

In this section, we propose a deep architecture, extending the single-layer adaptive switch described in Section 3. To avoid excessive mathematical notation, we focus on the architecture of a two-layer switch network, depicted in Figure 2. Generalization to more layers follows inductively and naturally by replicating switch network primitives.

4.1 Combining Switch Networks

To better model the k-th conditional distribution of discrete random variables, we first replicate and instantiate l single-layer switch networks, as described in Section 3. Each switch network utilizes m_1 auxiliary random variables. More precisely, the i-th switch network contains m_1 auxiliary variables denoted by $\{Y_{i,1}^{(k,1)}, \ldots, Y_{i,m_1}^{(k,1)}\}$, and a non-binary switch random variable denoted by $Z_i^{(k,1)}$. For indices $1 \le i \le l$ and $1 \le j \le m_1$, the conditional distributions of the auxiliaries are specified as logistic functions,

$$p\big(Y_{i,j}^{(k,1)}{=}1\big|x_{[1:k]};\theta_1^{(k)}\big)= \sigma\big(x_{[1:k]}^T w_{Y_{i,j}^{(k,1)}}+b_{Y_{i,j}^{(k,1)}}\big). \quad (7)$$

The conditional distribution for the adaptive switch for the i-th single-layer network is specified by a softmax function,

$$p\Big(Z_i^{(k,1)} = j\big|x_{[1:k]};\theta_1^{(k)}\Big)$$
$$= \frac{\exp\Big(x_{[1:k]}^T \alpha_{Z_i^{(k,1)}}^{(j)} + \beta_{Z_i^{(k,1)}}^{(j)}\Big)}{\displaystyle\sum_{j'=1}^{m_1} \exp\Big(x_{[1:k]}^T \alpha_{Z_i^{(k,1)}}^{(j')} + \beta_{Z_i^{(k,1)}}^{(j')}\Big)}. \quad (8)$$

The model parameters for the first layer are denoted by $\theta_1^{(k)}$, and consist of the following coefficients,

$$w_{Y_{i,j}^{(k,1)}} \in \mathbb{R}^k,\ b_{Y_{i,j}^{(k,1)}} \in \mathbb{R},$$
$$\alpha_{Z_i^{(k,1)}}^{(j)} \in \mathbb{R}^k, \beta_{Z_i^{(k,1)}}^{(j)} \in \mathbb{R},$$

for $1 \le i \le l$, $1 \le j \le m_1$. The total number of parameters is $\mathcal{O}(lm_1 k)$ for the k-th conditional, and $\mathcal{O}(lm_1 n^2)$ for all conditionals.

4.2 Intermediate Variables Between Layers

As shown in Figure 2, the outputs of the l single-layer networks are labeled by variables $F_1^{(k)}, \ldots, F_l^{(k)}$. These outputs are called intermediary variables prior to the second layer. The conditional distributions of these intermediary variables are specified as follows,

$$p\big(F_i^{(k)} = 1|x_{[1:k]};\theta_1^{(k)}\big)= \sum_{j=1}^{m_1} p\big(Z_i^{(k)} = j|x_{[1:k]};\theta_1^{(k)}\big) \times$$
$$p\big(Y_{i,j}^{(k,1)} = 1|x_{[1:k]};\theta_1^{(k)}\big). \quad (9)$$

We assume that the $F_i^{(k)}$, $1 \le i \le l$ are conditionally independent given $X_{[1:k]}$. More precisely, for any configuration of the intermediate variables $f_{[1:l]}^{(k)} \in \{0,1\}^l$, we assume the following product decomposition,

$$p\big(f_{[1:l]}^{(k)}|x_{[1:k]};\theta_1^{(k)}\big)= \prod_{i=1}^l p\big(f_i^{(k)}|x_{[1:k]};\theta_1^{(k)}\big).$$

4.3 Processing of the Second Layer

Having constructed the intermediate variables $F_1^{(k)}, \ldots, F_l^{(k)}$, the second and final output layer is once again an adaptive switch network. This switch network is modeled in parametric form with m_2 auxiliary random variables $\{Y_1^{(k,2)}, \ldots, Y_{m_2}^{(k,2)}\}$ and switch random variable $Z^{(k,2)}$. More precisely, for $1 \le j \le m_2$, conditioned on any configuration of the intermediate variables $f_{[1:l]}^{(k)} \in \{0,1\}^l$,

$$p\big(Y_j^{(k,2)} = 1\big|f_{[1:l]}^{(k)};\theta_2^{(k)}\big)= \sigma\big((f_{[1:l]}^{(k)})^T w_{Y_j^{(k,2)}}+ b_{Y_j^{(k,2)}}\big). \quad (10)$$

The conditional distribution for the adaptive switch of the second layer is given by a softmax function,

$$p\Big(Z_i^{(k,1)} = j\big|x_{[1:k]};\theta_1^{(k)}\Big)$$
$$= \frac{\exp\Big((f_{[1:l]}^{(k)})^T \alpha_{Z^{(k,2)}}^{(j)} + \beta_{Z^{(k,2)}}^{(j)}\Big)}{\displaystyle\sum_{j'=1}^{m_2} \exp\Big((f_{[1:l]}^{(k)})^T \alpha_{Z^{(k,2)}}^{(j')} + \beta_{Z^{(k,2)}}^{(j')}\Big)}. \quad (11)$$

The model parameters for the second layer are denoted by $\theta_2^{(k)}$, and consist of the following coefficients,

$$w_{Y_j^{(k,2)}} \in \mathbb{R}^l,\ b_{Y_j^{(k,2)}} \in \mathbb{R},$$
$$\alpha_{Z^{(k,2)}}^{(j)} \in \mathbb{R}^l,\ \beta_{Z^{(k,2)}}^{(j)} \in \mathbb{R},$$

for $1 \le j \le m_2$. The total number of parameters is $\mathcal{O}(m_2 l)$ for the k-th conditional, and $\mathcal{O}(m_2 ln)$ for all conditionals. Thus, a two-layer switch network can be characterized by the triplet (m_1, l, m_2) which specifies the number of parameters in both layers.

4.4 Log-Likelihood Objective

The target distribution of X_{k+1} conditioned on the intermediate variables is governed by

$$p\big(X_{k+1} = 1|f_{[1:l]}^{(k)};\theta_2^{(k)}\big)= \sum_{j=1}^{m_2} p\big(Y_j^{(k,2)} = 1|f_{[1:l]}^{(k)};\theta_2^{(k)}\big) \times$$
$$p\big(Z^{(k,2)} = j|f_{[1:l]}^{(k)};\theta_2^{(k)}\big). \quad (12)$$

The empirical log-likelihood of the k-th conditional given I discrete samples $\vec{x}^{(1)}, \ldots, \vec{x}^{(I)}$ is given by,

$$L^{(k)} := \frac{1}{I}\sum_{i=1}^I \log p\big(X_{k+1} = x_{k+1}^{(i)}|x_{[1:k]}^{(i)};\theta^{(k)}\big)$$
$$= \frac{1}{I}\sum_{i=1}^I \log \sum_{f_{[1:l]}^{(k)}} p\big(f_{[1:l]}^{(k)}|x_{[1:k]}^{(i)};\theta_1^{(k)}\big) \times \quad (13)$$
$$p\big(X_{k+1} = x_{k+1}^{(i)}|f_{[1:l]}^{(k)};\theta_2^{(k)}\big).$$

Model	Epochs trained	-(Log-likelihood)	TV distance	JS distance
One layer, $m = 4$	2910	6.887280	0.303341	0.020051
One layer, $m = 16$	3120	6.833584	0.156232	0.006034
Two layer, $(m_1, l, m_2) = (4, 4, 8)$	1090	6.82794115	0.13848737	0.00460584
Two layer, $(m_1, l, m_2) = (2, 8, 32)$	2320	6.820030	0.102167	0.002584

Table 1: Comparing the performance of switch network models for a synthetic discrete distribution. The log-likelihood is computed using an expectation over the true distribution. Total variation distance (TV) and Jensen–Shannon distance (JS) between the learned distributions and the true distribution are computed exactly.
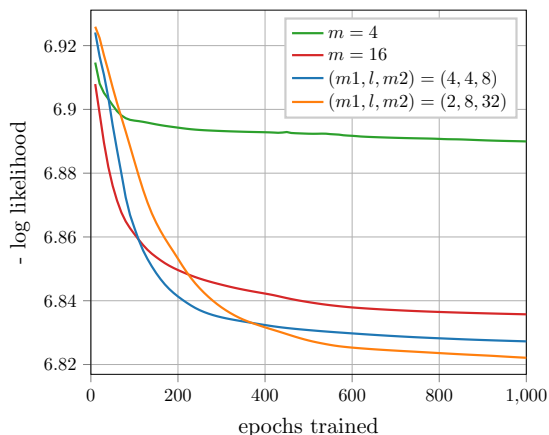

Figure 3: Log-likelihood (LL) curves of single-layer and multi-layer switch networks after SGD training over many epochs, on data samples from a synthetic distribution. The optimal LL value is 6.81.

4.5 MCMC Gradient Approximations

The log-likelihood given in (13) contains a summation $\sum_{f_{[1:l]}^{(k)}}$ over all possible configurations of the intermediate variables $F_{[1:l]}^{(k)}$ in $\{0, 1\}^l$. In two-layer experiments, we set $l = 4$ or $l = 8$ which produces reasonable results. The log-likelihood is differentiable and stochastic gradient descent (SGD) can be applied to the objective. However, in general, in order to overcome the computational difficulty in computing this objective and its gradients due to the exponential number of configurations of the intermediate variables when l is large, we discuss methods in the supplement to approximate the gradient of $L^{(k)}$. Specifically, Monte-Carlo Markov Chain (MCMC) gradient approximations are derived which provide a *trade-off* in the desired accuracy of approximation versus computational complexity.

5 Experimental Results

We evaluate single-layer and multilayer switch network architectures, as described in Sections 3 and 4, by conducting experiments on diverse sets of data.

5.1 Synthetic Discrete Distribution

We generated a synthetic distribution on $n = 10$ binary variables $X_{[1:10]}$, where the probability of each of the 2^n possible configurations is proportional to a uniform random number in the interval $[0.1, 1]$. The probabilities were normalized to ensure a valid probability distribution. Then, 10^5 i.i.d. samples were generated from this distribution as the training data.

Training Setup: The generated database was used to train several instances of the single-layer and two-layer switch networks. More precisely, two instances of a one-layer network with parameters $m = 4$ and $m = 16$, and two instances of a two-layer network with parameters $(m_1, l, m_2) = (4, 4, 8)$ and $(m_1, l, m_2) = (2, 8, 32)$ were trained. We used stochastic gradient descent (SGD) to optimize for the log-likelihood objectives in (6) and (13), using batch size 1000 and learning rate 10. For the two-layer objective, we optimized directly for the exact likelihood in (13) without employing an approximation, since l was not too large to cause computational difficulties.

Training Log-likelihood Performance: Figure 3 illustrates the empirical (negative) log-likelihood, i.e. $-\sum_{k=0}^{n-1} L^{(k)}$, after 1000 training epochs. Note that due to the minus sign, we should minimize the objective. It is easy to see that, using Gibbs' inequality, the value of the objective cannot go below the entropy of the empirical distribution of the database, which is approximately 6.81 in this example. As shown in Figure 3, the likelihood performance improves by increasing the hyper-parameter m for single-layer architectures, and also by updating from one-layer to two-layer structures. The two-layer network with $(m_1, l, m_2) = (2, 8, 32)$ nearly achieves the optimal value of 6.81.

Generalization Ability: Note that since the ground truth synthetic distribution is known, we can compare the joint distribution learned by our model using (5) and (12), and compare it with the ground truth. For the network architectures described in our training setup, we compared the total variation (TV) distance and Jensen–Shannon (JS) distance between the
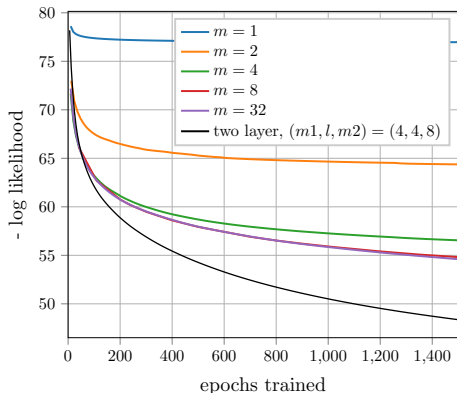
Figure 4: Empirical log-likelihood performance of training on a binary version of the MNIST handwritten digits dataset for different configurations of single-layer and two-layer switch networks. Single-layer architectures do not exhibit improvement beyond $m = 8$. However, the two-layer network structure provides an immense improvement over single-layer networks. These log-likelihood results may be compared to baseline results in the literature (Tieleman, 2008).

Figure 5: Generated binary digits from a simple 2–layer switch network with parameters $(m_1, l, m_2) = (4, 4, 8)$ after approximately 1500 epochs of training.

learned distributions and the ground truth. Table 1 illustrates these metrics. The TV and JS distances decrease as m increases, and also if the model is updated to a two-layer architecture.

5.2 High-Dimensional Binary MNIST

We converted each data sample of the MNIST dataset of handwritten digits into a discrete binary sequence. In its original form, each sample is a 28×28 image consisting of pixels which have values in the interval $[0, 255]$. We converted each pixel to a bit by treating values above 150 as 1 and values below 150 as 0. We then concatenated the rows of each image from top to bottom to form a binary vector of length $n = 784$. Thus, each data sample is represented by a high-dimensional binary vector in $\{0, 1\}^{784}$. This binary version of the MNIST dataset does not have an entropy of 784 bits. The handwritten variations occurring in the dataset likely constitute less than 50 bits of entropy.

Training Setup: We trained the discretized MNIST dataset using a one-layer switch network with the value of the hyperparameter m ranging in the set $\{1, 2, 4, 8, 32\}$. A single-layer network with $m = 1$ is essentially a logistic conditional model similar to that of (2). Also, we trained a two layer network with $(m_1, l, m_2) = (4, 4, 8)$. We used SGD for optimizing the log-likelihood in (6) for the one layer architecture, and also the exact likelihood objective in (13). In both cases, we used a batch size of 300 and learning rate 10.
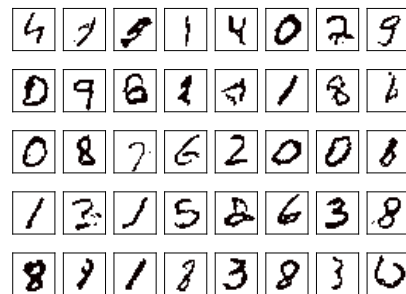
Training Log-Likelihood Performance: Figure 4 illustrates minus the training log likelihood objective, i.e. $-\sum_{k=0}^{n-1} L^{(k)}$, for the above configurations of the one layer and the two layer network. Increasing m from 1 to 2, i.e. adding one more logistic variable, greatly improves the likelihood performance. Increasing m above 8 does not improve the training likelihood. However, the two layer configuration exhibits a significant improvement in the training likelihood.

Generalization Ability: Figure 5 illustrates samples generated by a 2–layer switch network with parameters $(m_1, l, m_2) = (4, 4, 8)$. We also computed the test error performance for single-layer networks on the MNIST test dataset which consists of $10,000$ test samples. Figure 6 illustrates the test log-likelihood performance versus the model complexity (i.e. the hyperparameter m) for single-layer switch networks. This plot obeys the general bias-variance trade-off.

5.3 Web-Crawled Language Data

We crawled the web.mit.edu website and extracted words of length up to 8 characters. This dataset consists of $65,568$ words in total, and 686 distinct words. We processed each character as lowercase, implying 27 possibilities for characters, including a space and characters $\{a, \ldots, z\}$. Each character was converted into an integer in the set $\{0, \ldots 26\}$ by mapping a space to 0, a to 1, b to 2, and so forth. We concatenated the 5-bit binary representation of the integers corresponding to each character to form a binary sequence of length $n = 8 \times 5 = 40$. Thus, each word was mapped to a high-dimensional binary vector in $\{0, 1\}^{40}$.

Training Setup: We trained two instances of the single-layer switch network with parameters $m = 4$ and $m = 16$ and one instance of the two layer switch
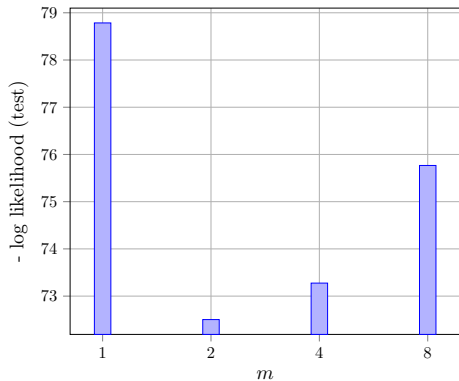
Figure 6: Test likelihood performance versus model complexity for single-layer switch networks trained on binary `MNIST` data.
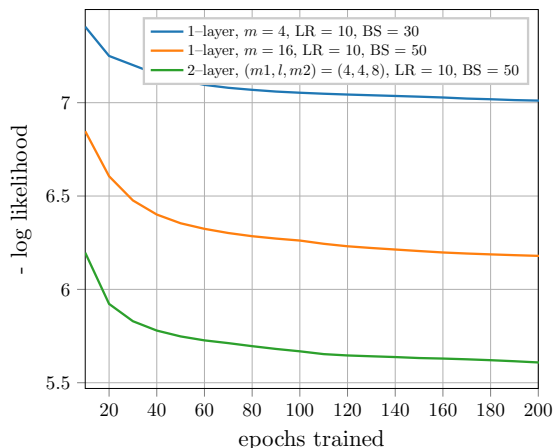

Figure 7: Log-likelihood performance of training for learning character-level (i.e., characters encoded into bits) generation of web-crawled words. LR = learning rate, and BS = batch size. Hyper-parameter search was conducted to achieve improved log-likelihoods. Log-likelihoods provide a direct assessment of the training performance of the language model.

network with $(m_1, l, m_2) = (4, 4, 8)$. For the single-layer configurations, we optimize the objective as in (6), and for the two layer configuration, we optimize the exact objective as in (13), using SGD.

Training Log-Likelihood Performance Figure 7 illustrates the likelihood performance for the above mentioned configurations. Note that the two layer architecture which has roughly $m_1 \times l = 16$ many parameters achieves a better likelihood compared to the one layer architecture with roughly the same $m = 16$ number of parameters.

Generalization Ability: Figure 8 illustrates representative words generated using three different network configurations. The two-layer network model generates several recognizable words, with the exception of `srereou`, which nevertheless still follows the vowel and consonant structure of language.

6   Conclusion

We have proposed deep switch networks for learning high-dimensional discrete distributions using maximum-likelihood optimization. The adaptive switch model is simple and interpretable. Comparisons and assessments of performance for several network structures were provided. Results were obtained for diverse, high-dimensional discrete distributions.

1–**layer** $m = 4$	1–**layer** $m = 16$	2–**layer** (m_1, l, m_2) $= (4, 4, 8)$
LR = 10	LR = 10	LR = 10
BS = 30	BS = 50	BS = 50
200 epochs	240 epochs	220 epochs
aieed in research riuu and the mealeted roc the of vishimn saydese	a the elmc the of the robou syr?kt mahh nev and to	a alumni we the function help research its true know srereou world
48/100	74/100	91/100

Figure 8: Sample words generated by single-layer and two-layer switch networks. Correctly generated words are colored in dark blue, and non-dictionary words are colored in orange. LR = learning rate, and BS = batch size. The final row indicates the ratio of the number of dictionary words to the total number of words generated per model.

References

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223.

Arora, S., Ge, R., Liang, Y., Ma, T., and Zhang, Y. (2017). Generalization and equilibrium in generative adversarial nets (GANs). In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 224–232, International Convention Centre, Sydney, Australia. PMLR.

Arora, S., Risteski, A., and Zhang, Y. (2018). Do GANs learn the distribution? some theory and empirics. In *International Conference on Learning Representations*.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

Che, T., Li, Y., Zhang, R., Hjelm, R. D., Li, W., Song, Y., and Bengio, Y. (2017). Maximum-likelihood augmented discrete generative adversarial networks. *CoRR*, abs/1702.07983.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., and Bengio, Y. (2015). A recurrent latent variable model for sequential data. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 2980–2988. Curran Associates, Inc.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013). Maxout networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning*, pages III–1319–III–1327.

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8):1771–1800.

Hjelm, R. D., Jacob, A. P., Trischler, A., Che, G., Cho, K., and Bengio, Y. (2018). Boundary seeking GANs. In *International Conference on Learning Representations*.

Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., and Wu, Y. (2016). Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2018). Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. *Proceeding of International Conference on Learning Representations 2014*.

Rolfe, J. T. (2017). Discrete variational autoencoders. In *ICLR*.

Salakhutdinov, R. and Hinton, G. (2009). Deep boltzmann machines. In van Dyk, D. and Welling, M., editors, *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, volume 5 of *Proceedings of Machine Learning Research*, pages 448–455, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA. PMLR.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. (2017). Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR*.

Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015). Highway networks. *CoRR*, abs/1505.00387.

Tieleman, T. (2008). Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1064–1071, New York, NY, USA. ACM.

Vahdat, A., Andriyash, E., and Macready, W. (2018). Dvae#: Discrete variational autoencoders with relaxed boltzmann priors. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 1864–1874. Curran Associates, Inc.

van den Oord, A., Kalchbrenner, N., Espeholt, L., kavukcuoglu, k., Vinyals, O., and Graves, A. (2016). Conditional image generation with pixel-cnn decoders. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 4790–4798. Curran Associates, Inc.

van den Oord, A., Vinyals, O., and kavukcuoglu, k. (2017). Neural discrete representation learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 6306–6315. Curran Associates, Inc.

Zilly, J. G., Srivastava, R. K., Koutník, J., and Schmidhuber, J. (2017). Recurrent highway networks. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 4189–4198.