ARM: AUGMENT-REINFORCE-MERGE GRADIENT FOR STOCHASTIC BINARY NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

To backpropagate the gradients through stochastic binary layers, we propose the augment-REINFORCE-merge (ARM) estimator that is unbiased and has low variance. Exploiting data augmentation, REINFORCE, and reparameterization, the ARM estimator achieves adaptive variance reduction for Monte Carlo integration by merging two expectations via common random numbers. The variance-reduction mechanism of the ARM estimator can also be attributed to antithetic sampling in an augmented space. Experimental results show the ARM estimator provides state-of-the-art performance in auto-encoding variational Bayes and maximum likelihood inference, for discrete latent variable models with one or multiple stochastic binary layers. Python code is available at https://github.com/ABC-anonymous-1.

1 INTRODUCTION

Given a function $f(\boldsymbol{z})$ of a random variable $\boldsymbol{z} = (z_1, \ldots, z_V)^T$, which follows a distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z})$ parameterized by $\boldsymbol{\phi}$, there has been significant recent interest in estimating $\boldsymbol{\phi}$ to maximize (or minimize) the expectation of $f(\boldsymbol{z})$ with respect to $\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z})$, expressed as

$$\mathcal{E}(\boldsymbol{\phi}) = \int f(\boldsymbol{z}) q_{\boldsymbol{\phi}}(\boldsymbol{z}) d\boldsymbol{z} = \mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z})}[f(\boldsymbol{z})]. \tag{1}$$

In particular, maximizing the marginal likelihood of a hierarchal Bayesian model (Bishop, 1995) and maximizing the evidence lower bound (ELBO) for variational inference (Jordan et al., 1999; Blei et al., 2017), two fundamental problems in statistical inference, both boil down to maximizing an expectation as in (1). To maximize (1), if $\nabla_{\boldsymbol{z}} f(\boldsymbol{z})$ is tractable to compute and $\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z})$ can be generated via reparameterization as $\boldsymbol{z} = \mathcal{T}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}),\ \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$, where $\boldsymbol{\epsilon}$ are random noises and $\mathcal{T}_{\boldsymbol{\phi}}(\cdot)$ denotes a deterministic transform parameterized by $\boldsymbol{\phi}$, then one may apply the reparameterization trick (Kingma & Welling, 2013; Rezende et al., 2014) to compute the gradient as

$$\nabla_{\boldsymbol{\phi}} \mathcal{E}(\boldsymbol{\phi}) = \nabla_{\boldsymbol{\phi}} \mathbb{E}_{\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})}[f(\mathcal{T}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}))] = \mathbb{E}_{\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\phi}} f(\mathcal{T}_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}))]. \tag{2}$$

Unfortunately, this trick is often not applicable to discrete random variables, which are widely used to construct discrete latent variable models such as the sigmoid belief net (Neal, 1992; Saul et al., 1996).

To maximize (1) for discrete $\boldsymbol{z}$, using the score function $\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{z}) = \nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\boldsymbol{z}) / q_{\boldsymbol{\phi}}(\boldsymbol{z})$, one may compute $\nabla_{\boldsymbol{\phi}} \mathcal{E}(\boldsymbol{\phi})$ via REINFORCE (Williams, 1992) as

$$\nabla_{\boldsymbol{\phi}} \mathcal{E}(\boldsymbol{\phi}) = \mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z})}[f(\boldsymbol{z}) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{z})] \approx \frac{1}{K} \sum\nolimits_{k=1}^{K} f(\boldsymbol{z}^{(k)}) \nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{z}^{(k)}),$$

where $\boldsymbol{z}^{(k)} \overset{iid}{\sim} q_{\boldsymbol{\phi}}(\boldsymbol{z})$ are independent, and identically distributed (iid). This unbiased estimator is also known as (a.k.a.) the score-function (Fu, 2006) or likelihood-ratio estimator (Glynn, 1990). While it is unbiased and only requires drawing iid random samples from $q_{\boldsymbol{\phi}}(\boldsymbol{z})$ and computing $\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{z}^{(k)})$, its high Monte-Carlo-integration variance often limits its use in practice. Note that if $f(\boldsymbol{z})$ depends on $\boldsymbol{\phi}$, then we assume it is always true that $\mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z})}[\nabla_{\boldsymbol{\phi}} f(\boldsymbol{z})] = 0$. For example, in variational inference, we need to maximize the ELBO as $\mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z})}[f(\boldsymbol{z})]$, where $f(\boldsymbol{z}) = \log[p(\boldsymbol{x} \mid \boldsymbol{z}) p(\boldsymbol{z}) / q_{\boldsymbol{\phi}}(\boldsymbol{z})]$. In this case, although $f(\boldsymbol{z})$ depends on $\boldsymbol{\phi}$, since $\mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z})}[\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\boldsymbol{z})] = \int \nabla_{\boldsymbol{\phi}} q_{\boldsymbol{\phi}}(\boldsymbol{z}) d\boldsymbol{z} = \nabla_{\boldsymbol{\phi}} \int q_{\boldsymbol{\phi}}(\boldsymbol{z}) d\boldsymbol{z} = 0$, we have $\mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z})}[\nabla_{\boldsymbol{\phi}} f(\boldsymbol{z})] = 0$.

To address the high-variance issue, one may introduce appropriate control variates (a.k.a. baselines) to reduce the variance of REINFORCE (Paisley et al., 2012; Ranganath et al., 2014; Mnih & Gregor,

2014; Gu et al., 2016; Mnih & Rezende, 2016; Ruiz et al., 2016; Kucukelbir et al., 2017; Naesseth et al., 2017). Alternatively, one may first relax the discrete random variables with continuous ones and then apply the reparameterization trick to estimate the gradients, which reduces the variance of Monte Carlo integration at the expense of introducing bias (Maddison et al., 2017; Jang et al., 2017). Combining both REINFORCE and the continuous relaxation of discrete random variables, Tucker et al. (2017) and Grathwohl et al. (2018) both aim to produce a low-variance and unbiased gradient estimator by introducing a continuous relaxation based control variate, whose parameter, however, needs to be estimated at each mini-batch by minimizing the sample variance of the estimator with stochastic gradient descent (SGD), increasing not only the computational complexity, but also the risk of overfitting the training data. Another interesting variance-control idea applicable to discrete latent variables is using local expectation gradients, which estimates the gradients based on REINFORCE, by performing Monte Carlo integration using a single global sample together with exact integration of the local variable for each latent dimension (Titsias & Lázaro-Gredilla, 2015).

Distinct from the usual idea of introducing control variates to reduce the estimation variance of REINFORCE, we propose the augment-REINFORCE-merge (ARM) estimator, a novel unbiased and low-variance gradient estimator for binary latent variables. We show by rewriting the expectation with respect to Bernoulli random variables as one with respect to augmented exponential random variables and then expressing the gradient as an expectation via REINFORCE, with the assistance of appropriate reparameterization, one can derive the ARM estimator in the augmented space by using either the strategy of sharing common random numbers between two expectations or the strategy of applying antithetic sampling. Both strategies, as detailedly discussed in Owen (2013), can both be used to explain why the ARM estimator is unbiased and could lead to significant variance reduction.

Our experimental results on both auto-encoding variational Bayes and maximum likelihood inference for discrete latent variable models, with one or multiple discrete stochastic layers, show that the ARM estimator converges fast, has low computational complexity, and provides state-of-the-art out-of-sample prediction performance, suggesting the effectiveness of using the ARM estimator for gradient backpropagation through stochastic binary layers.

2 ARM: AUGMENT-REINFORCE-MERGE ESTIMATOR

In this section, we first present the key theorem of the paper, and then provide its derivation. With this theorem, we summarize ARM gradient ascent for multivariate binary latent variables in Algorithm 1, as shown in the Appendix. Let us denote $\sigma(\phi) = e^{\phi}/(1+e^{\phi})$ as the sigmoid function and $\mathbf{1}_{[\cdot]}$ as an indicator function that equals to one if the argument is true and zero otherwise.

Theorem 1 (ARM). *For a vector of V binary random variables $\boldsymbol{z} = (z_1, \ldots, z_V)^T$, the gradient of*

$$\mathcal{E}(\boldsymbol{\phi}) = \mathbb{E}_{\boldsymbol{z} \sim \prod_{v=1}^{V} \text{Bernoulli}(z_v; \sigma(\phi_v))}[f(\boldsymbol{z})] \tag{3}$$

with respect to $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_V)^T$, the logits of the Bernoulli probability parameters, can be expressed as

$$\nabla_{\boldsymbol{\phi}} \mathcal{E}(\boldsymbol{\phi}) = \mathbb{E}_{\boldsymbol{u} \sim \prod_{v=1}^{V} \text{Uniform}(u_v;0,1)} \left[\left(f(\mathbf{1}_{[\boldsymbol{u} > \sigma(-\boldsymbol{\phi})]}) - f(\mathbf{1}_{[\boldsymbol{u} < \sigma(\boldsymbol{\phi})]}) \right) (\boldsymbol{u} - 1/2) \right], \tag{4}$$

where $\mathbf{1}_{[\boldsymbol{u} > \sigma(-\boldsymbol{\phi})]} := \left(\mathbf{1}_{[u_1 > \sigma(-\phi_1)]}, \ldots, \mathbf{1}_{[u_V > \sigma(-\phi_V)]} \right)^T$.

For simplicity, we will first present the ARM estimator for a univariate binary latent variable ($i.e.$, $V = 1$), and then generalize it to a multivariate one ($i.e.$, $V > 1$). In the univariate case, we need to evaluate the gradient of $\mathcal{E}(\phi) = \mathbb{E}_{z \sim \text{Bernoulli}(\sigma(\phi))}[f(z)]$ with respect to ϕ. Let us denote $t \sim \text{Exp}(\lambda)$ as an exponential distribution, whose probability density function is defined as $p(t \,|\, \lambda) = \lambda e^{-\lambda t}$, where $\lambda > 0$ and $t > 0$. The mean and variance are $\mathbb{E}[t] = \lambda^{-1}$ and $\text{var}[t] = \lambda^{-2}$, respectively. The exponential random variable $t \sim \text{Exp}(\lambda)$ can be reparameterized as $t = \epsilon/\lambda,\ \epsilon \sim \text{Exp}(1)$. It is well known, $e.g.$, in Ross (2006), that if $t_1 \sim \text{Exp}(\lambda_1)$ and $t_2 \sim \text{Exp}(\lambda_2)$ are two independent exponential random variables, then the probability that t_1 is smaller than t_2 can be expressed as $P(t_1 < t_2) = \lambda_1/(\lambda_1 + \lambda_2)$; moreover, since $t_1 \overset{d}{=} \epsilon_1/\lambda_1$ and $t_2 \overset{d}{=} \epsilon_2/\lambda_2$, where $\epsilon_1, \epsilon_2 \overset{iid}{\sim} \text{Exp}(1)$ and the symbol "$\overset{d}{=}$" denotes "equal in distribution," we have

$$P(t_1 < t_2) = P(\epsilon_1/\lambda_1 < \epsilon_2/\lambda_2) = P(\epsilon_1 < \epsilon_2 \lambda_1/\lambda_2) = \lambda_1/(\lambda_1 + \lambda_2). \tag{5}$$

2.1 AUGMENTATION OF A BERNOULLI RANDOM VARIABLE AND REPARAMETERIZATION

From (5) it becomes clear that the Bernoulli random variable $z \sim \text{Bernoulli}(\sigma(\phi))$ can be reparameterized by racing two augmented exponential random variables as

$$z = \mathbf{1}_{[\epsilon_1 < \epsilon_2 e^{\phi}]},\ \epsilon_1 \sim \text{Exp}(1),\ \epsilon_2 \sim \text{Exp}(1). \tag{6}$$

Consequently, the expectation with respect to the Bernoulli random variable can be reparameterized as one with respect to two augmented exponential random variables as

$$\mathcal{E}(\phi) = \mathbb{E}_{z \sim \text{Bernoulli}(\sigma(\phi))}[f(z)] = \mathbb{E}_{\epsilon_1, \epsilon_2 \overset{iid}{\sim} \text{Exp}(1)}[f(\mathbf{1}_{[\epsilon_1 e^{-\phi} < \epsilon_2]})]. \tag{7}$$

2.2 REINFORCE ESTIMATOR IN THE AUGMENTED SPACE

Since the indicator function $\mathbf{1}_{[\epsilon_1 e^{-\phi} < \epsilon_2]}$ is not differentiable, the reparameterization trick in (2) is not directly applicable to computing the gradient of (7). Fortunately, as $t_1 = \epsilon_1 e^{-\phi},\ \epsilon_1 \sim \text{Exp}(1)$ is equal in distribution to $t_1 \sim \text{Exp}(e^{\phi})$, the expectation in (7) can be further reparameterized as

$$\mathcal{E}(\phi) = \mathbb{E}_{\epsilon_1, \epsilon_2 \overset{iid}{\sim} \text{Exp}(1)}[f(\mathbf{1}_{[\epsilon_1 e^{-\phi} < \epsilon_2]})] = \mathbb{E}_{t_1 \sim \text{Exp}(e^{\phi}),\ \epsilon_2 \sim \text{Exp}(1)}[f(\mathbf{1}_{[t_1 < \epsilon_2]})], \tag{8}$$

and hence, via REINFORCE and then another reparameterization, we can express the gradient as

$$\begin{aligned}
\nabla_{\phi}\mathcal{E}(\phi) &= \mathbb{E}_{t_1 \sim \text{Exp}(e^{\phi}),\ \epsilon_2 \sim \text{Exp}(1)}[f(\mathbf{1}_{[t_1 < \epsilon_2]}) \nabla_{\phi} \log \text{Exp}(t_1; e^{\phi})] \\
&= \mathbb{E}_{t_1 \sim \text{Exp}(e^{\phi}),\ \epsilon_2 \sim \text{Exp}(1)}[f(\mathbf{1}_{[t_1 < \epsilon_2]})(1 - t_1 e^{\phi})] \\
&= \mathbb{E}_{\epsilon_1, \epsilon_2 \overset{iid}{\sim} \text{Exp}(1)}[f(\mathbf{1}_{[\epsilon_1 e^{-\phi} < \epsilon_2]})(1 - \epsilon_1)].
\end{aligned} \tag{9}$$

Similarly, we have $\mathcal{E}(\phi) = \mathbb{E}_{\epsilon_1, \epsilon_2 \overset{iid}{\sim} \text{Exp}(1)}[f(\mathbf{1}_{[\epsilon_1 < \epsilon_2 e^{\phi}]})] = \mathbb{E}_{\epsilon_1 \sim \text{Exp}(1),\ t_2 \sim \text{Exp}(-e^{\phi})}[f(\mathbf{1}_{[\epsilon_1 < t_2]})]$, and hence can also express the gradient as

$$\begin{aligned}
\nabla_{\phi}\mathcal{E}(\phi) &= \mathbb{E}_{\epsilon_1 \sim \text{Exp}(1),\ t_2 \sim \text{Exp}(e^{-\phi})}[f(\mathbf{1}_{[\epsilon_1 < t_2]}) \nabla_{\phi} \log \text{Exp}(t_2; e^{-\phi})] \\
&= -\mathbb{E}_{\epsilon_1 \sim \text{Exp}(1),\ t_2 \sim \text{Exp}(e^{-\phi})}[f(\mathbf{1}_{[\epsilon_1 < t_2]})(1 - t_2 e^{-\phi})] \\
&= -\mathbb{E}_{\epsilon_1, \epsilon_2 \overset{iid}{\sim} \text{Exp}(1)}[f(\mathbf{1}_{[\epsilon_1 e^{-\phi} < \epsilon_2]})(1 - \epsilon_2)].
\end{aligned} \tag{10}$$

2.3 MERGE OF REINFORCE GRADIENTS

A key observation of the paper is that by swapping the indices of the two iid standard exponential random variables in (10) , the gradient $\nabla_{\phi}\mathcal{E}(\phi)$ can be equivalently expressed as

$$\nabla_{\phi}\mathcal{E}(\phi) = -\mathbb{E}_{\epsilon_1, \epsilon_2 \overset{iid}{\sim} \text{Exp}(1)}[f(\mathbf{1}_{[\epsilon_2 e^{-\phi} < \epsilon_1]})(1 - \epsilon_1)]. \tag{11}$$

As the term inside the expectation in (9) and that in (11) could be highly positively correlated, we are motivated to merge (9) and (11) by sharing the same set of standard exponential random variables for Monte Carlo integration, which provides a new opportunity to well control the estimation variance (Owen, 2013). More specifically, simply taking the average of (9) and (11) leads to

$$\nabla_{\phi}\mathcal{E}(\phi) = \mathbb{E}_{\epsilon_1, \epsilon_2 \overset{iid}{\sim} \text{Exp}(1)} \left[\left(f(\mathbf{1}_{[\epsilon_1 e^{-\phi} < \epsilon_2]}) - f(\mathbf{1}_{[\epsilon_2 e^{-\phi} < \epsilon_1]}) \right) (1/2 - \epsilon_1/2) \right]. \tag{12}$$

Note one may also take a weighted average of (9) and (11), and optimize the combination weight to potentially further reduce the variance of the estimator. We leave that for future study.

Note that letting $\epsilon_1, \epsilon_2 \overset{iid}{\sim} \text{Exp}(1)$ is the same in distribution as letting

$$\epsilon_1 = \epsilon u,\ \epsilon_2 = \epsilon(1 - u), \quad \text{where } u \sim \text{Uniform}(0, 1),\ \epsilon \sim \text{Gamma}(2, 1), \tag{13}$$

which can be proved using $\text{Exp}(1) \overset{d}{=} \text{Gamma}(1, 1)$, $(u, 1 - u)^T \overset{d}{=} \text{Dirichlet}(\mathbf{1}_2)$, where $u \sim \text{Uniform}(0, 1)\}$, and Lemma IV.3 of Zhou & Carin (2012). Thus, (12) can be reparameterized as

$$\nabla_{\phi}\mathcal{E}(\phi) = \mathbb{E}_{u \sim \text{Uniform}(0,1),\ \epsilon \sim \text{Gamma}(2,1)} \left[\left(f(\mathbf{1}_{[u > \sigma(-\phi)]}) - f(\mathbf{1}_{[u < \sigma(\phi)]}) \right) (\epsilon u/2 - 1/2) \right],$$

Applying Rao Blackwellization (Casella & Robert, 1996), we can further express the gradient as

$$\nabla_{\phi}\mathcal{E}(\phi) = \mathbb{E}_{u \sim \text{Uniform}(0,1)} \left[\left(f(\mathbf{1}_{[u > \sigma(-\phi)]}) - f(\mathbf{1}_{[u < \sigma(\phi)]}) \right) (u - 1/2) \right], \tag{14}$$

which will be referred to as the Augment-REINFORCE-merge (ARM) estimator.

2.4 RELATIONSHIP TO ANTITHETIC SAMPLING

While we use augmentation, REINFORCE, and merge steps to obtain (12) and another reparameterization and marginalizing step to obtain the ARM gradient in (14), we find it can also be obtained by performing augmentation, REINFORCE, reparameterization, and antithetic sampling steps. More specifically, using the equivalence between $\epsilon_1, \epsilon_2 \overset{iid}{\sim} \text{Exp}(1)$ and (13), we can reparameterize (9) as

$$\begin{aligned}\nabla_\phi \mathcal{E}(\phi) &= \mathbb{E}_{u\sim\text{Uniform}(0,1),\ \epsilon\sim\text{Gamma}(2,1)}\big[f(\mathbf{1}_{[u<\sigma(\phi)]})(1-\epsilon u)\big] \\ &= \mathbb{E}_{u\sim\text{Uniform}(0,1)}\big[f(\mathbf{1}_{[u<\sigma(\phi)]})(1-2u)\big]. \end{aligned} \tag{15}$$

Further using antithetic sampling (Owen, 2013) with $\tilde{u} = 1-u$, we have

$$\nabla_\phi \mathcal{E}(\phi) = \mathbb{E}_{u\sim\text{Uniform}(0,1)}\big[f(\mathbf{1}_{[u<\sigma(\phi)]})(1/2-u)\big] + \mathbb{E}_{\tilde{u}\sim\text{Uniform}(0,1)}\big[f(\mathbf{1}_{[\tilde{u}<\sigma(\phi)]})(1/2-\tilde{u})\big] \tag{16}$$

$$= \mathbb{E}_{u\sim\text{Uniform}(0,1)}\big[f(\mathbf{1}_{[u<\sigma(\phi)]})(1/2-u) + f(\mathbf{1}_{[\tilde{u}<\sigma(\phi)]})(1/2-\tilde{u})\big] \tag{17}$$

which becomes the same as the ARM estimator in (14).

2.5 MULTIVARIATE GENERALIZATION

The ARM estimator for univariate binary, however, is of little use in practice, as one may first analytically solve the expectation and then compute its gradient. Below we show how to generalize the univariate ARM estimator to a multivariate one. Let us denote $(\cdot)_{\setminus v}$ as a vector whose vth element is removed. For the expectation in (3), applying the univariate ARM estimator in (14), we have

$$\begin{aligned}\nabla_{\phi_v}\mathcal{E}(\boldsymbol{\phi}) &= \mathbb{E}_{\boldsymbol{z}_{\setminus v}\sim\prod_{\nu\neq v}\text{Bernoulli}(z_\nu;\sigma(\phi_\nu))}\big\{\nabla_{\phi_v}\mathbb{E}_{z_v\sim\text{Bernoulli}(\sigma(\phi_v)}[f(\boldsymbol{z})]\big\} \\ &= \mathbb{E}_{\boldsymbol{z}_{\setminus v}\sim\prod_{\nu\neq v}\text{Bernoulli}(z_\nu;\sigma(\phi_\nu))}\big\{\mathbb{E}_{u_v\sim\text{Uniform}(0,1)}\big[(u_v-1/2) \\ &\qquad \times\big(f(\boldsymbol{z}_{\setminus v}, z_v=\mathbf{1}_{[u_v>\sigma(-\phi_v)]}) - f(\boldsymbol{z}_{\setminus v}, z_v=\mathbf{1}_{[u_v<\sigma(\phi_v)]})\big)\big]\big\}. \end{aligned} \tag{18}$$

Since $\boldsymbol{z}_{\setminus v}\sim\prod_{\nu\neq v}\text{Bernoulli}(z_\nu;\sigma(\phi_\nu))$ can be equivalently generated as $\boldsymbol{z}_{\setminus v} = \mathbf{1}_{[\boldsymbol{u}_{\setminus v}<\sigma(\boldsymbol{\phi}_{\setminus v})]}$ or as $\boldsymbol{z}_{\setminus v} = \mathbf{1}_{[\boldsymbol{u}_{\setminus v}>\sigma(-\boldsymbol{\phi}_{\setminus v})]}$, where $\boldsymbol{u}_{\setminus v}\sim\prod_{\nu\neq v}\text{Uniform}(u_\nu;0,1)$, exchanging the order of the two expectations in (18) and applying reparameterization, we have

$$\begin{aligned}\nabla_{\phi_v}\mathcal{E}(\boldsymbol{\phi}) &= \mathbb{E}_{u_v\sim\text{Uniform}(0,1)}\big\{(u_v-1/2)\,\mathbb{E}_{\boldsymbol{z}_{\setminus v}\sim\prod_{\nu\neq v}\text{Bernoulli}(z_\nu;\sigma(\phi_\nu))}\big[\\ &\qquad f(\boldsymbol{z}_{\setminus v}, z_v=\mathbf{1}_{[u_v>\sigma(-\phi_v)]}) - f(\boldsymbol{z}_{\setminus v}, z_v=\mathbf{1}_{[u_v<\sigma(\phi_v)]})\big]\big\} \\ &= \mathbb{E}_{\boldsymbol{u}\sim\prod_{v=1}^V\text{Uniform}(u_v;0,1)}\big[(u_v-1/2)\big(f(\mathbf{1}_{[\boldsymbol{u}>\sigma(-\boldsymbol{\phi})]}) - f(\mathbf{1}_{[\boldsymbol{u}<\sigma(\boldsymbol{\phi})]})\big)\big], \end{aligned} \tag{19}$$

which concludes the proof for (4) shown in Theorem 1.

Alternatively, instead of generalizing the univariate ARM gradient as in (18) and (19), we can first do a multivariate generalization of the univariate Augment-REINFORCE gradient in (15) as

$$\begin{aligned}\nabla_{\phi_v}\mathcal{E}(\boldsymbol{\phi}) &= \mathbb{E}_{\boldsymbol{z}_{\setminus v}\sim\prod_{\nu\neq v}\text{Bernoulli}(z_\nu;\sigma(\phi_\nu))}\big\{\nabla_{\phi_v}\mathbb{E}_{z_v\sim\text{Bernoulli}(\sigma(\phi_v)}[f(\boldsymbol{z})]\big\} \\ &= \mathbb{E}_{\boldsymbol{z}_{\setminus v}\sim\prod_{\nu\neq v}\text{Bernoulli}(z_\nu;\sigma(\phi_\nu))}\big\{\mathbb{E}_{u_v\sim\text{Uniform}(0,1)}\big[(1-2u_v)f(\boldsymbol{z}_{\setminus v}, z_v=\mathbf{1}_{[u_v<\sigma(\phi_v)]})\big]\big\} \\ &= \mathbb{E}_{\boldsymbol{u}\sim\prod_{v=1}^V\text{Uniform}(u_v;0,1)}\big[(1-2u_v)f(\mathbf{1}_{[\boldsymbol{u}<\sigma(\boldsymbol{\phi})]})\big], \end{aligned} \tag{20}$$

and then add an antithetic sampling step to arrive at (4).

2.6 EFFECTIVENESS OF ARM IN VARIANCE REDUCTION

Let us denote $g_v(\boldsymbol{u}) = f(\mathbf{1}_{[\boldsymbol{u}<\sigma(\boldsymbol{\phi})]})(1-2u_v)$ and $\tilde{\boldsymbol{u}} = 1-\boldsymbol{u}$. With $\boldsymbol{u}^{(k)}\overset{iid}{\sim}\prod_{v=1}^V\text{Uniform}(0,1)$, we define the ARM estimate of $\nabla_{\phi_v}\mathcal{E}(\boldsymbol{\phi})$ with K Monte Carlo samples, denoted as $g_{\text{ARM},v}$, and the augment-REINFORCE (AR) estimate with $2K$ Monte Carlo samples, denoted as $g_{\text{AR},v}$, using

$$g_{\text{ARM},v} = \tfrac{1}{2K}\textstyle\sum_{k=1}^K(g_v(\boldsymbol{u}^{(k)}) + g_v(\tilde{\boldsymbol{u}}^{(k)})), \quad g_{\text{AR},v} = \tfrac{1}{2K}\textstyle\sum_{k=1}^{2K} g_v(\boldsymbol{u}^{(k)}).$$

Similar to the analysis in Owen (2013), the amount of variance reduction brought by the ARM estimator can be reflected by the ratio of the variance of $g_{\text{ARM},v}$ to that of $g_{\text{AR},v}$ as

$$\frac{\text{var}[g_{\text{ARM},v}]}{\text{var}[g_{\text{AR},v}]} = \frac{\text{var}[g_v(\boldsymbol{u})] - \text{cov}(-g_v(\boldsymbol{u}), g_v(\tilde{\boldsymbol{u}}))}{\text{var}[g_v(\boldsymbol{u})]} = 1-\rho_v, \quad \rho_v = \text{Corr}(-g_v(\boldsymbol{u}), g_v(1-\boldsymbol{u})).$$

Note $-g_v(\boldsymbol{u}) = f(\mathbf{1}_{[\boldsymbol{u}<\sigma(\boldsymbol{\phi})]})(2u_v-1)$, $g_v(1-\boldsymbol{u}) = f(\mathbf{1}_{[\boldsymbol{u}>\sigma(-\boldsymbol{\phi})]})(2u_v-1)$, and $P(\mathbf{1}_{[u_v<\sigma(\phi_v)]} = \mathbf{1}_{[u_v>\sigma(-\phi_v)]}) = \sigma(|\phi_v|) - \sigma(-|\phi_v|)$, thus a strong positive correlation (*i.e.*, $\rho_v\to 1$) and hence noticeable variance reduction is likely especially if ϕ_v moves far away from zero during training.

3 BACKPROPAGATION THROUGH DISCRETE STOCHASTIC LAYERS

A latent variable model with multiple stochastic hidden layers can be constructed as

$$\boldsymbol{x} \sim p_{\boldsymbol{\theta}_0}(\boldsymbol{x} \,|\, \boldsymbol{b}_1),\ \boldsymbol{b}_1 \sim p_{\boldsymbol{\theta}_1}(\boldsymbol{b}_1 \,|\, \boldsymbol{b}_2), \ldots, \boldsymbol{b}_t \sim p_{\boldsymbol{\theta}_t}(\boldsymbol{b}_t \,|\, \boldsymbol{b}_{t+1}), \ldots, \boldsymbol{b}_T \sim p_{\boldsymbol{\theta}_T}(\boldsymbol{b}_T), \tag{21}$$

whose joint likelihood given the distribution parameters $\boldsymbol{\theta}_{0:T} = \{\boldsymbol{\theta}_0, \ldots, \boldsymbol{\theta}_T\}$ is expressed as

$$p(\boldsymbol{x}, \boldsymbol{b}_{1:T} \,|\, \boldsymbol{\theta}_{0:T}) = p_{\boldsymbol{\theta}_0}(\boldsymbol{x} \,|\, \boldsymbol{b}_1) \Big[\prod\nolimits_{t=1}^{T-1} p_{\boldsymbol{\theta}_t}(\boldsymbol{b}_t \,|\, \boldsymbol{b}_{t+1}) \Big] p_{\boldsymbol{\theta}_T}(\boldsymbol{b}_T). \tag{22}$$

In comparison to deterministic feedforward neural networks, stochastic ones can represent complex distributions and show natural resistance to overfitting (Neal, 1992; Saul et al., 1996; Tang & Salakhutdinov, 2013; Raiko et al., 2014; Gu et al., 2016; Tang & Salakhutdinov, 2013). However, the training, especially if there are stochastic discrete layers, is often much more challenging. Below we show for both auto-encoding variational Bayes and maximum likelihood inference, how to apply the ARM estimator for gradient backpropagation in stochastic binary networks.

3.1 ARM VARIATIONAL AUTO-ENCODER

For auto-encoding variational Bayes inference (Kingma & Welling, 2013; Rezende et al., 2014), we construct a variational distribution as

$$q_{\boldsymbol{w}_{1:T}}(\boldsymbol{b}_{1:T} \,|\, \boldsymbol{x}) = q_{\boldsymbol{w}_1}(\boldsymbol{b}_1 \,|\, \boldsymbol{x}) \Big[\prod\nolimits_{t=1}^{T-1} q_{\boldsymbol{w}_{t+1}}(\boldsymbol{b}_{t+1} \,|\, \boldsymbol{b}_t) \Big], \tag{23}$$

with which the ELBO can be expressed as

$$\begin{aligned} &\mathcal{E}(\boldsymbol{w}_{1:T}) = \mathbb{E}_{\boldsymbol{b}_{1:T} \sim q_{\boldsymbol{w}_{1:T}}(\boldsymbol{b}_{1:T} \,|\, \boldsymbol{x})} \left[f(\boldsymbol{b}_{1:T}) \right], \text{ where} \\ &f(\boldsymbol{b}_{1:T}) = \log p_{\boldsymbol{\theta}_0}(\boldsymbol{x} \,|\, \boldsymbol{b}_1) + \log p_{\boldsymbol{\theta}_{1:T}}(\boldsymbol{b}_{1:T}) - \log q_{\boldsymbol{w}_{1:T}}(\boldsymbol{b}_{1:T} \,|\, \boldsymbol{x}). \end{aligned} \tag{24}$$

Proposition 2 (ARM backpropagation)**.** *For a stochastic binary network with T binary stochastic hidden layers, constructing a variational auto-encoder (VAE) defined with $\boldsymbol{b}_0 = \boldsymbol{x}$ and*

$$q_{\boldsymbol{w}_t}(\boldsymbol{b}_t \,|\, \boldsymbol{b}_{t-1}) = \text{Bernoulli}(\boldsymbol{b}_t; \sigma(\mathcal{T}_{\boldsymbol{w}_t}(\boldsymbol{b}_{t-1}))) \tag{25}$$

for $t = 1, \ldots, T$, the gradient of the ELBO with respect to $\boldsymbol{w}_t$ can be expressed as

$$\begin{aligned} &\nabla_{\boldsymbol{w}_t} \mathcal{E}(\boldsymbol{w}_{1:T}) = \mathbb{E}_{q(\boldsymbol{b}_{1:t-1})} \left[\mathbb{E}_{\boldsymbol{u}_t \sim \text{Uniform}(0,1)} [f_\Delta(\boldsymbol{u}_t, \mathcal{T}_{\boldsymbol{w}_t}(\boldsymbol{b}_{t-1}), \boldsymbol{b}_{1:t-1})(\boldsymbol{u}_t - 1/2)] \nabla_{\boldsymbol{w}_t} \mathcal{T}_{\boldsymbol{w}_t}(\boldsymbol{b}_{t-1}) \right], \\ &\textit{where } f_\Delta(\boldsymbol{u}_t, \mathcal{T}_{\boldsymbol{w}_t}(\boldsymbol{b}_{t-1}), \boldsymbol{b}_{1:t-1}) = \mathbb{E}_{\boldsymbol{b}_{t+1:T} \sim q(\boldsymbol{b}_{t+1:T} \,|\, \boldsymbol{b}_t),\ \boldsymbol{b}_t = \mathbf{1}_{[\boldsymbol{u}_t > \sigma(-\mathcal{T}_{\boldsymbol{w}_t}(\boldsymbol{b}_{t-1}))]}} [f(\boldsymbol{b}_{1:T})] \\ &\qquad\qquad - \mathbb{E}_{\boldsymbol{b}_{t+1:T} \sim q(\boldsymbol{b}_{t+1:T} \,|\, \boldsymbol{b}_t),\ \boldsymbol{b}_t = \mathbf{1}_{[\boldsymbol{u}_t < \sigma(\mathcal{T}_{\boldsymbol{w}_t}(\boldsymbol{b}_{t-1}))]}} [f(\boldsymbol{b}_{1:T})] \end{aligned} \tag{26}$$

The gradient presented in (26) can be estimated with a single Monte Carlo sample as

$$\hat{f}_\Delta(\boldsymbol{u}_t, \mathcal{T}_{\boldsymbol{w}_t}(\boldsymbol{b}_{t-1}), \boldsymbol{b}_{1:t-1}) = \begin{cases} 0, & \text{if } \boldsymbol{b}_t^{(1)} = \boldsymbol{b}_t^{(2)} \\ f(\boldsymbol{b}_{1:t-1}, \boldsymbol{b}_{t:T}^{(1)}) - f(\boldsymbol{b}_{1:t-1}, \boldsymbol{b}_{t:T}^{(2)}), & \text{otherwise} \end{cases}, \tag{27}$$

where $\boldsymbol{b}_t^{(1)} = \mathbf{1}_{[\boldsymbol{u}_t > \sigma(-\mathcal{T}_{\boldsymbol{w}_t}(\boldsymbol{b}_{t-1}))]}$, $\boldsymbol{b}_{t+1:T}^{(1)} \sim q(\boldsymbol{b}_{t+1:T} \,|\, \boldsymbol{b}_t^{(1)})$, $\boldsymbol{b}_t^{(2)} = \mathbf{1}_{[\boldsymbol{u}_t < \sigma(\mathcal{T}_{\boldsymbol{w}_t}(\boldsymbol{b}_{t-1}))]}$, and $\boldsymbol{b}_{t+1:T}^{(2)} \sim q(\boldsymbol{b}_{t+1:T} \,|\, \boldsymbol{b}_t^{(2)})$. The proof of Proposition 2 is provided in the Appendix.

3.2 ARM MAXIMUM LIKELIHOOD INFERENCE

For maximum likelihood inference, the log marginal likelihood can be expressed as

$$\begin{aligned} \log p_{\boldsymbol{\theta}_{0:T}}(\boldsymbol{x}) &= \log \mathbb{E}_{\boldsymbol{b}_{1:T} \sim p_{\boldsymbol{\theta}_{1:T}}(\boldsymbol{b}_{1:T})} [p_{\boldsymbol{\theta}_0}(\boldsymbol{x} \,|\, \boldsymbol{b}_1)] \\ &\geq \mathcal{E}(\boldsymbol{\theta}_{1:T}) = \mathbb{E}_{\boldsymbol{b}_{1:T} \sim p_{\boldsymbol{\theta}_{1:T}}(\boldsymbol{b}_{1:T})} [\log p_{\boldsymbol{\theta}_0}(\boldsymbol{x} \,|\, \boldsymbol{b}_1)]. \end{aligned} \tag{28}$$

Generalizing Proposition 2 leads to the following proposition.

Proposition 3. *For a stochastic binary network defined as*

$$p_{\boldsymbol{\theta}_t}(\boldsymbol{b}_t \,|\, \boldsymbol{b}_{t+1}) = \text{Bernoulli}(\boldsymbol{b}_t; \sigma(\mathcal{T}_{\boldsymbol{\theta}_t}(\boldsymbol{b}_{t+1}))), \tag{29}$$

the gradient of the lower bound in (28) *with respect to $\boldsymbol{\theta}_t$ can be expressed as*

$$\begin{aligned} &\nabla_{\boldsymbol{\theta}_t} \mathcal{E}(\boldsymbol{\theta}_{1:T}) = \mathbb{E}_{p(\boldsymbol{b}_{t+1:T})} \left[\mathbb{E}_{\boldsymbol{u}_t \sim \text{Uniform}(0,1)} [f_\Delta(\boldsymbol{u}_t, \mathcal{T}_{\boldsymbol{\theta}_t}(\boldsymbol{b}_{t+1}), \boldsymbol{b}_{t+1:T})(\boldsymbol{u}_t - 1/2)] \nabla_{\boldsymbol{\theta}_t} \mathcal{T}_{\boldsymbol{\theta}_t}(\boldsymbol{b}_{t+1}) \right], \\ &\textit{where } f_\Delta(\boldsymbol{u}_t, \mathcal{T}_{\boldsymbol{\theta}_t}(\boldsymbol{b}_{t+1}), \boldsymbol{b}_{t+1:T}) = \mathbb{E}_{\boldsymbol{b}_{1:t-1} \sim p(\boldsymbol{b}_{1:t-1} \,|\, \boldsymbol{b}_t),\ \boldsymbol{b}_t = \mathbf{1}_{[\boldsymbol{u}_t > \sigma(-\mathcal{T}_{\boldsymbol{\theta}_t}(\boldsymbol{b}_{t+1}))]})} [\log p_{\boldsymbol{\theta}_0}(\boldsymbol{x} \,|\, \boldsymbol{b}_1)] \\ &\qquad\qquad - \mathbb{E}_{\boldsymbol{b}_{1:t-1} \sim p(\boldsymbol{b}_{1:t-1} \,|\, \boldsymbol{b}_t),\ \boldsymbol{b}_t = \mathbf{1}_{[\boldsymbol{u}_t < \sigma(\mathcal{T}_{\boldsymbol{\theta}_t}(\boldsymbol{b}_{t+1}))]})} [\log p_{\boldsymbol{\theta}_0}(\boldsymbol{x} \,|\, \boldsymbol{b}_1)]. \end{aligned}$$

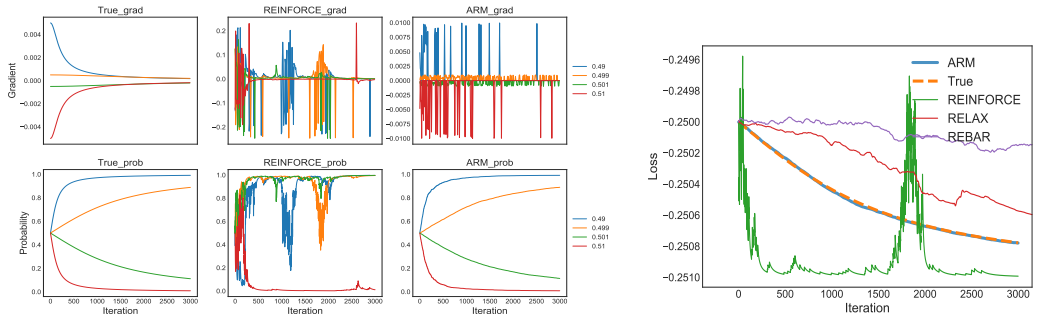


Figure 1: Left: Trace plots of the true/estimated gradients and estimated Bernoulli probability parameters for $p_0 \in \{0.49, 0.499, 0.501, 0.51\}$; Right: Trace plots of the loss functions for $p_0 = 0.499$.

4 Experimental results

To illustrate the working mechanism of the ARM estimator, related to Tucker et al. (2017) and Grathwohl et al. (2018), we consider learning ϕ to maximize $\mathcal{E}(\phi) = \mathbb{E}_{z\sim\text{Bernoulli}(\sigma(\phi))}[(z - p_0)^2]$, where $p_0 \in \{0.49, 0.499, 0.501, 0.51\}$, or equivalently, minimize the loss as $-\mathcal{E}(\phi)$. The optimal solution is $\sigma(\phi) = \mathbf{1}(p_0 < 0.5)$. The closer p_0 is to 0.5, the more challenging the optimization becomes. We compare the ARM estimator to the true gradient as $g_\phi = (1 - 2p_0)\sigma(\phi)(1 - \sigma(\phi))$ and three previously proposed unbiased estimators, including REINFORCE, REBAR (Tucker et al., 2017), and RELAX (Grathwohl et al., 2018). With a single random sample $u \sim \text{Uniform}(0, 1)$ for Monte Carlo integration, the ARM gradient can be expressed as

$$g_{\phi,\text{ARM}} = \left[(\mathbf{1}_{[u>\sigma(-\phi)]} - p_0)^2 - (\mathbf{1}_{[u<\sigma(\phi)]} - p_0)^2\right](u - 1/2),$$

while the REINFORCE gradient can be expressed as

$$g_{\phi,\text{REINFORCE}} = (\mathbf{1}_{[u<\sigma(\phi)]} - p_0)^2(\mathbf{1}_{[u<\sigma(\phi)]} - \sigma(\phi)).$$

See Tucker et al. (2017) and Grathwohl et al. (2018) for the details about REBAR and RELAX, respectively, which both introduce stochastically estimated control variates to improve REINFORCE.

As shown in Figure 1 (a), the REINFORCE gradients have large variances. Consequently, a REINFORCE based gradient ascent algorithm may diverge. For example, when $p_0 = 0.501$, the optimal value for the Bernoulli probability $\sigma(\phi)$ is 0, but the algorithm infers it to be close to 1 at the end of 3000 iterations of a random trial. By contrast, the univariate ARM estimator well approximates the time-varying true gradients by adjusting the frequencies, amplitudes, and signs of its gradient estimates, with larger and more frequent spikes for larger true gradients. As shown in Figure 1 (b), using the ARM estimator is indistinguishable from using the true gradient for updating ϕ to minimize the loss $-\mathbb{E}_{z\sim\text{Bernoulli}(\sigma(\phi))}[(z - 0.499)^2]$, significantly outperforming not only REINFORCE, which has a large variance, but also both REBAR and RELAX, which improve on REINFORCE by introducing carefully constructed control variates that are stochastically updated for variance reduction. We further plot in Figure 3 of the Appendix the gradient estimated with multiple Monte Carlo samples against the true gradient at each iteration, showing the ARM estimator has significant lower variance than REINFORCE given the same number of Monte Carlo samples.

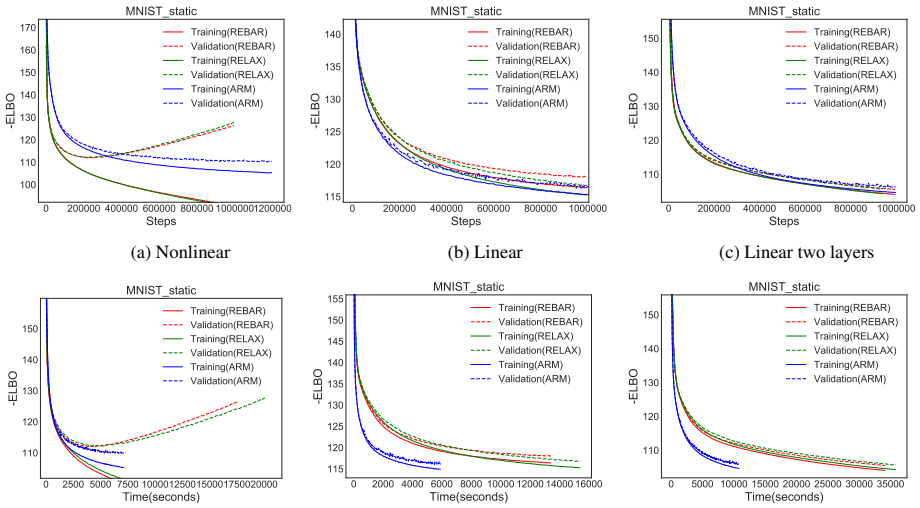
4.1 Discrete variational auto-encoders

To optimize a variational auto-encoder (VAE) for a discrete latent variable model, existing solutions often rely on biased but low-variance stochastic gradient estimators (Bengio et al., 2013; Jang et al., 2017), unbiased but high-variance ones (Mnih & Gregor, 2014), or unbiased REINFORCE combined with computationally expensive control variates, whose parameters are estimated by minimizing the sample variance of the estimator with SGD (Tucker et al., 2017; Grathwohl et al., 2018). Comparing to previously proposed methods for discrete latent variables, the ARM estimator exhibits low variance and is unbiased, computationally efficient, and simple to implement.

For discrete VAEs, we compare ARM with a variety of representative stochastic gradient estimators for discrete latent variables, including Wake-Sleep (Hinton et al., 1995), NVIL (Mnih & Gregor,

Table 1: The constructions of three differently structured discrete variational auto-encoders. The following symbols "→", "]",)", and "⇝" represent deterministic linear transform, leaky rectified linear units (LeakyReLU) (Maas et al., 2013) nonlinear activation, sigmoid nonlinear activation, and random sampling respectively, in the encoder (a.k.a. recognition network); their reversed versions are used in the decoder (a.k.a. generator).

	Nonlinear	Linear	Linear two layers
Encoder	784→200]→200]→200)⇝200	784→200)⇝200	784→200)⇝200→200)⇝200
Decoder	784⇜(784←[200←[200←200	784⇜(784←200	784⇜(784←200⇜(200←200



(a) Nonlinear (b) Linear (c) Linear two layers

(d) Nonlinear (e) Linear (f) Linear two layers

Figure 2: Test negative ELBOs on MNIST-static with respect to training iterations, shown in the top row, and wall clock times on Tesla-K40 GPU, shown in the bottom row, for three differently structured Bernoulli VAEs.

2014), LeGrad (Titsias & Lázaro-Gredilla, 2015), MuProp (Gu et al., 2016), Concrete (Gumbel-Softmax) (Jang et al., 2017; Maddison et al., 2017), REBAR (Grathwohl et al., 2018), and RELAX (Tucker et al., 2017). Following the settings in Tucker et al. (2017) and Grathwohl et al. (2018), for the encoder defined in (22) and decoder defined in (23), we consider three different network architectures, as summarized in Table 1, including "Nonlinear" that has one stochastic but two Leaky-ReLU (Maas et al., 2013) deterministic hidden layers, "Linear" that has one stochastic hidden layer, and "Linear two layers" that has two stochastic hidden layers. We consider a widely used binarization (Salakhutdinov & Murray, 2008; Larochelle & Murray, 2011), referred to as MNIST-static and available at http://www.dmi.usherb.ca/∼larocheh/mlpython/_modules/datasets/binarized_mnist.html, making our numerical results directly comparable to those reported in the literature. In addition to MNIST-static, we also consider MNIST-threshold (van den Oord et al., 2017), which binarizes MNIST by thresholding each pixel value at 0.5, and the binarized OMNIGLOT dataset.

We train discrete VAEs with 200 conditionally iid Bernoulli random variables as the hidden units of each stochastic binary layer. We maximize a single-Monte-Carlo-sample ELBO using Adam (Kingma & Ba, 2014), with the learning rate selected from $\{5, 1, 0.5\} \times 10^{-4}$ by the validation set. We set the batch size as 50 for MNIST and 25 for OMNIGLOT. For each dataset, using its default training/validation/testing partition, we train all methods on the training set, calculate the validation log-likelihood for every epoch, and report the test negative log-likelihood when the validation negative log-likelihood reaches its minimum within a predefined maximum number of iterations.

We summarize the test negative log-likelihoods in Table 2 for MNIST-static. We also summarize the test negative ELBOs in Table 4 of the Appendix, and provide related trace plots of the training and validation negative ELBOs on MNIST-static in Figure 2, and these on MNIST-threshold and OMNIGLOT in Figures 5 and 6 of the Appendix, respectively. For these trace plots, for a fair comparison of convergence speed between different algorithms, we use publicly available code from

Table 2: Test negative log-likelihoods of discrete VAEs trained with a variety of stochastic gradient estimators on MNIST-static and OMNIGLOT, where $*$, $\star$, $\dagger$, $\ddagger$ represent the results reported in Mnih & Gregor (2014), Tucker et al. (2017), Gu et al. (2016), and Grathwohl et al. (2018), respectively. The results for LeGrad (Titsias & Lázaro-Gredilla, 2015) are obtained by running the code provided by the authors. We report the results of ARM using the sample mean and standard deviation over five independent trials with random initializations.

(a) MNIST

Linear		Nonlinear		Two layers	
Algorithm	$-\log p(x)$	Algorithm	$-\log p(x)$	Algorithm	$-\log p(x)$
REINFORCE	$= 164.0$	REINFORCE	$= 114.6$	REINFORCE	$= 159.2$
Wake-Sleep*	$= 120.8$	Wake-Sleep*	-	Wake-Sleep*	$= 107.7$
NVIL *	$= 113.1$	NVIL *	$= 102.2$	NVIL*	$= 99.8$
LeGrad	≤ 117.5	LeGrad	-	LeGrad	-
MuProp†	≤ 113.0	MuProp*	$= 99.1$	MuProp†	≤ 100.4
Concrete*	$= 107.3$	Concrete*	$= 99.6$	Concrete*	$= 95.6$
REBAR*	$= 107.7$	REBAR*	$= 101.4$	REBAR*	$=$ **95.4**
RELAX‡	≤ 113.6	RELAX‡	≤ 119.2	RELAX‡	≤ 100.9
ARM	$=$ **107.2** $\pm$ **0.1**	ARM	$=$ **98.4** $\pm$ **0.3**	ARM	$= 96.7 \pm 0.3$

(b) OMNIGLOT

Linear		Nonlinear		Two layers	
Algorithm	$-\log p(x)$	Algorithm	$-\log p(x)$	Algorithm	$-\log p(x)$
NVIL*	$= 117.6$	NVIL*	$=$ **116.6**	NVIL*	$= 111.4$
MuProp*	$= 117.6$	MuProp*	$= 117.5$	MuProp*	$= 111.2$
Concrete*	$= 117.7$	Concrete*	$= 116.7$	Concrete*	$= 111.3$
REBAR*	$= 117.7$	REBAR*	$= 118.0$	REBAR*	$= 110.8$
RELAX‡	≤ 122.1	RELAX‡	≤ 128.2	RELAX‡	≤ 115.4
ARM	$=$ **115.8** $\pm$ **0.2**	ARM	$= 117.6 \pm 0.4$	ARM	$=$ **109.8** $\pm$ **0.3**

the authors and setting the learning rate of ARM the same as that selected by REBAR/RELAX in Grathwohl et al. (2018).

These results show that ARM provides state-of-the-art performance in delivering not only fast convergence, but also low negative log-likelihoods and negative ELBOs on both the validation and test sets, with low computational cost, for all three different network architectures. In comparison to the vanilla REINFORCE on MNIST-static, as shown in Table 2 (a), ARM achieves significantly lower test log-likelihoods, which can be explained by having much lower variance in its gradient estimation, while only costing 20% to 30% more computation time to finish the same number of iterations.

The trace plots in Figures 2, 5, and 6 show that ARM achieves its objective better or on a par with the state-of-the-art methods in all three different network architectures. In particular, the performance of ARM on MNIST-threshold is significantly better, suggesting ARM is more robust, better resists overfitting, and has better generalization ability. On OMNIGLOT, with "Nonlinear" network architecture, both REBAR and RELAX exhibit severe overfitting, which could be caused by their training procedure that updates the parameters of the control variates, which are designed to minimize the true variance of the gradient estimator, by minimizing the sample variance of the gradient estimator using SGD. For less overfitting linear and two-stochastic-layer networks, ARM overall performs better than both REBAR and RELAX and converges significantly faster (about 6-8 times faster) in terms of computation time.

4.2 Maximum likelihood inference for a stochastic binary network

Denoting $\boldsymbol{x}_l, \boldsymbol{x}_u \in \mathbb{R}^{394}$ as the lower and upper halves of an MNIST digit, respectively, we consider a standard benchmark task of estimating the conditional distribution $p_{\boldsymbol{\theta}_{0:2}}(\boldsymbol{x}_l \mid \boldsymbol{x}_u)$ (Raiko et al., 2014; Bengio et al., 2013; Gu et al., 2016; Jang et al., 2017; Tucker et al., 2017), using a stochastic binary network with two stochastic binary hidden layers, expressed as

$$\boldsymbol{x}_l \sim \text{Bernoulli}(\sigma(\mathcal{T}_{\boldsymbol{\theta}_0}(\boldsymbol{b}_1))),\ \boldsymbol{b}_1 \sim \text{Bernoulli}(\sigma(\mathcal{T}_{\boldsymbol{\theta}_1}(\boldsymbol{b}_2))),\ \boldsymbol{b}_2 \sim \text{Bernoulli}(\sigma(\mathcal{T}_{\boldsymbol{\theta}_2}(\boldsymbol{x}_u))). \quad (30)$$

Table 3: For the MNIST conditional distribution estimation benchmark task, comparison of the test negative log-likelihood between ARM and various gradient estimators in Jang et al. (2017) is reported here.

Gradient estimator	ARM	ST	DARN	Annealed ST	ST Gumbel-S.	SF	MuProp
$-\log p(\boldsymbol{x}_l \mid \boldsymbol{x}_u)$	$\mathbf{57.9 \pm 0.1}$	58.9	59.7	58.7	59.3	72.0	58.9

We set the network structure as 392-200-200-392 which means both $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are 200 dimensional binary vectors and the transformation $\mathcal{T}_{\boldsymbol{\theta}}$ are linear so the results are directly comparable with those in Jang et al. (2017). We approximate $\log p_{\boldsymbol{\theta}_{0:2}}(\boldsymbol{x}_l \mid \boldsymbol{x}_u)$ with $\log \frac{1}{K}\sum_{k=1}^{K} \text{Bernoulli}(\boldsymbol{x}_l; \sigma(\mathcal{T}_{\boldsymbol{\theta}_0}(\boldsymbol{b}_1^{(k)})))$, where $\boldsymbol{b}_1^{(k)} \sim \text{Bernoulli}(\sigma(\mathcal{T}_{\boldsymbol{\theta}_1}(\boldsymbol{b}_2^{(k)})))$, $\boldsymbol{b}_2^{(k)} \sim \text{Bernoulli}(\sigma(\mathcal{T}_{\boldsymbol{\theta}_2}(\boldsymbol{x}_u)))$. We perform training with $K = 1$, which can also be considered as optimizing on a single-Monte-Carlo-sample estimate of the lower bound of the log likelihood shown in (28). We use Adam (Kingma & Ba, 2014), with the learning rate set as 10^{-4}, mini-batch size as 100, and number of epochs for training as 2000. Given the inferred point estimate of $\boldsymbol{\theta}_{0:2}$ after training, we evaluate the accuracy of conditional density estimation by estimating the negative log-likelihood as $-\log p_{\boldsymbol{\theta}_{0:2}}(\boldsymbol{x}_l \mid \boldsymbol{x}_u)$, averaging over the test set using $K = 1000$. We show example results of predicting the activation probabilities of the pixels of $\boldsymbol{x}_l$ given $\boldsymbol{x}_u$ in Figure 4 of the Appendix.

As shown in Table 3, optimizing a stochastic binary network with the ARM estimator, which is unbiased and computationally efficient, achieves the lowest test negative log-likelihood, outperforming previously proposed biased stochastic gradient estimators on similarly structured stochastic networks, including DARN (Gregor et al., 2013), straight through (ST) (Bengio et al., 2013), slope-annealed ST (Chung et al., 2016), and ST Gumbel-softmax (Jang et al., 2017), and unbiased ones, including score-function (SF) and MuProp (Gu et al., 2016).

5 Conclusions

To train a discrete latent variable model with one or multiple stochastic binary layers, we propose the augment-REINFORCE-merge (ARM) estimator to provide unbiased and low-variance gradient estimates of the parameters of Bernoulli distributions. With a single Monte Carlo sample, the estimated gradient is the product of uniform random noises and the difference of a function of two vectors of correlated binary latent variables. Without relying on learning control variates for variance reduction, it maintains efficient computation and avoids increasing the risk of overfitting. Applying the ARM gradient leads to not only fast convergence, but also low test negative log-likelihoods (and low test negative evidence lower bounds for variational inference), on both auto-encoding variational Bayes and maximum likelihood inference for stochastic binary feedforward neural networks. Some natural extensions of the proposed ARM estimator include generalizing it to multivariate categorical latent variables, combining it with a control-variate or local-expectation based variance reduction method, and applying it to reinforcement learning whose action space is discrete.

References

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Christopher M Bishop. *Neural Networks for Pattern Recognition*. Oxford university press, 1995.

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

George Casella and Christian P Robert. Rao-blackwellisation of sampling schemes. *Biometrika*, 83 (1):81–94, 1996.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704*, 2016.

Michael C Fu. Gradient estimation. *Handbooks in operations research and management science*, 13: 575–616, 2006.

Peter W Glynn. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10):75–84, 1990.

Will Grathwohl, Dami Choi, Yuhuai Wu, Geoff Roeder, and David Duvenaud. Backpropagation through the Void: Optimizing control variates for black-box gradient estimation. In *ICLR*, 2018.

Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra. Deep autoregressive networks. *arXiv preprint arXiv:1310.8499*, 2013.

Shixiang Gu, Sergey Levine, Ilya Sutskever, and Andriy Mnih. MuProp: Unbiased backpropagation for stochastic neural networks. In *ICLR*, 2016.

Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-softmax. In *ICLR*, 2017.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M Blei. Automatic differentiation variational inference. *Journal of Machine Learning Research*, 18(14):1–45, 2017.

Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 29–37, 2011.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, 2013.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *ICLR*, 2017.

Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. In *ICML*, pp. 1791–1799, 2014.

Andriy Mnih and Danilo J Rezende. Variational inference for monte carlo objectives. *arXiv preprint arXiv:1602.06725*, 2016.

Christian Naesseth, Francisco Ruiz, Scott Linderman, and David Blei. Reparameterization gradients through acceptance-rejection sampling algorithms. In *AISTATS*, pp. 489–498, 2017.

R. M. Neal. Connectionist learning of belief networks. *Artificial Intelligence*, pp. 71–113, 1992.

Art B. Owen. *Monte Carlo Theory, Methods and Examples*, chapter 8 Variance Reduction. 2013.

John Paisley, David M Blei, and Michael I Jordan. Variational Bayesian inference with stochastic search. In *ICML*, pp. 1363–1370, 2012.

Tapani Raiko, Mathias Berglund, Guillaume Alain, and Laurent Dinh. Techniques for learning binary stochastic feedforward neural networks. *arXiv preprint arXiv:1406.2989*, 2014.

Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *AISTATS*, pp. 814–822, 2014.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, pp. 1278–1286, 2014.

Sheldon M. Ross. *Introduction to Probability Models*. Academic Press, 10th edition, 2006.

Francisco J. R. Ruiz, Michalis K. Titsias, and David M. Blei. The generalized reparameterization gradient. In *NIPS*, pp. 460–468, 2016.

Ruslan Salakhutdinov and Iain Murray. On the quantitative analysis of deep belief networks. In *ICML*, pp. 872–879, 2008.

Lawrence K Saul, Tommi Jaakkola, and Michael I Jordan. Mean field theory for sigmoid belief networks. *Journal of Artificial Intelligence Research*, 4:61–76, 1996.

Yichuan Tang and Ruslan R Salakhutdinov. Learning stochastic feedforward neural networks. In *NIPS*, pp. 530–538, 2013.

Michalis K Titsias and Miguel Lázaro-Gredilla. Local expectation gradients for black box variational inference. In *NIPS*, pp. 2638–2646. MIT Press, 2015.

George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. In *NIPS*, pp. 2624–2633, 2017.

Aaron van den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pp. 6306–6315, 2017.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pp. 5–32. Springer, 1992.

Mingyuan Zhou and Lawrence Carin. Negative binomial process count and mixture modeling. *arXiv preprint arXiv:1209.3442v1*, 2012.

Appendix

A THE ARM ALGORITHM

We summarize the algorithm to compute ARM gradient for binary latent variables. Here we show the gradient with respect to the logits associated with the probability of Bernoulli random variables. If the logits are further generated by deterministic transform such as neural networks, the gradient with respect to the transform parameters can be directly computed by the chain rule. For stochastic transforms, the implementation of ARM gradient is discussed in Section 3.

Algorithm 1: ARM gradient for V-dimensional binary latent vector

input : Bernoulli distribution $\{q_{\phi_v}(z_v)\}_{v=1:V}$ with probability $\{\sigma(\phi_v)\}_{v=1:V}$, target $f(\boldsymbol{z})$;
$\boldsymbol{z} = (z_1, \cdots, z_V)$, $\boldsymbol{\phi} = (\phi_1, \cdots, \phi_V)$
output : $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ that maximize $\mathcal{E}(\boldsymbol{\phi}, \boldsymbol{\psi}) = \mathbb{E}_{\boldsymbol{z} \sim \prod_{v=1}^{V} q_{\phi_v}(z_v)}[f(\boldsymbol{z}; \boldsymbol{\psi})]$

Initialize $\boldsymbol{\phi}$, $\boldsymbol{\psi}$ randomly;
while *not converged* **do**
  Sample $z_v \sim \text{Bernoulli}(\sigma(\phi_v))$ for $v = 1, \cdots, V$;
  sample $u_v \sim \text{Uniform}(0, 1)$ for $v = 1, \cdots, V$, $\boldsymbol{u} = (u_1, \cdots, u_V)$;
  $g_{\boldsymbol{\psi}} = \nabla_{\boldsymbol{\psi}} f(\boldsymbol{z}; \boldsymbol{\psi})$;
  $f_\Delta(\boldsymbol{u}, \boldsymbol{\phi}) = f\big(\mathbf{1}_{[\boldsymbol{u} > \sigma(-\boldsymbol{\phi})]}\big) - f\big(\mathbf{1}_{[\boldsymbol{u} < \sigma(\boldsymbol{\phi})]}\big)$;
  $g_{\boldsymbol{\phi}} = f_\Delta(\boldsymbol{u}, \boldsymbol{\phi})(\boldsymbol{u} - \frac{1}{2})$
  $\boldsymbol{\phi} = \boldsymbol{\phi} + \rho_t g_{\boldsymbol{\phi}}, \quad \boldsymbol{\psi} = \boldsymbol{\psi} + \eta_t g_{\boldsymbol{\psi}}$ with step-size ρ_t, η_t
end

B MULTILAYER DISCRETE STOCHASTIC NETWORK

We present the derivation of ARM gradient in multiple stochastic layers discussed in main text Proposition2.

Proof of proposition 2. First, to compute the gradient with respect to $\boldsymbol{w}_1$, since

$$\mathcal{E}(\boldsymbol{w}_{1:T}) = \mathbb{E}_{q(\boldsymbol{b}_1)} \mathbb{E}_{q(\boldsymbol{b}_{2:T} \mid \boldsymbol{b}_1)}[f(\boldsymbol{b}_{1:T})] \tag{31}$$

we have

$$\nabla_{\boldsymbol{w}_1} \mathcal{E}(\boldsymbol{w}_{1:T}) = \mathbb{E}_{\boldsymbol{u}_1 \sim \text{Uniform}(0,1)}[f_\Delta(\boldsymbol{u}_1, \mathcal{T}_{\boldsymbol{w}_1}(\boldsymbol{x}))(\boldsymbol{u}_1 - 1/2)] \nabla_{\boldsymbol{w}_1} \mathcal{T}_{\boldsymbol{w}_1}(\boldsymbol{x}), \tag{32}$$

where

$$\begin{aligned} f_\Delta(\boldsymbol{u}_1, \mathcal{T}_{\boldsymbol{w}_1}(\boldsymbol{x})) = {} & \mathbb{E}_{\boldsymbol{b}_{2:T} \sim q(\boldsymbol{b}_{2:T} \mid \boldsymbol{b}_1),\ \boldsymbol{b}_1 = \mathbf{1}_{[\boldsymbol{u}_1 > \sigma(-\mathcal{T}_{\boldsymbol{w}_1}(\boldsymbol{x}))]})}[f(\boldsymbol{b}_{1:T})] \\ & - \mathbb{E}_{\boldsymbol{b}_{2:T} \sim q(\boldsymbol{b}_{2:T} \mid \boldsymbol{b}_1),\ \boldsymbol{b}_1 = \mathbf{1}_{[\boldsymbol{u}_1 < \sigma(\mathcal{T}_{\boldsymbol{w}_1}(\boldsymbol{x}))]})}[f(\boldsymbol{b}_{1:T})] \end{aligned} \tag{33}$$

Second, to compute the gradient with respect to $\boldsymbol{w}_t$, where $2 \leq t \leq T-1$, since

$$\mathcal{E}(\boldsymbol{w}_{1:T}) = \mathbb{E}_{q(\boldsymbol{b}_{1:t-1})} \mathbb{E}_{q(\boldsymbol{b}_t \mid \boldsymbol{b}_{t-1})} \mathbb{E}_{q(\boldsymbol{b}_{t+1:T} \mid \boldsymbol{b}_t)}[f(\boldsymbol{b}_{1:T})] \tag{34}$$

we have

$$\nabla_{\boldsymbol{w}_t} \mathcal{E}(\boldsymbol{w}_{1:T}) = \mathbb{E}_{q(\boldsymbol{b}_{1:t-1})} \left[\mathbb{E}_{\boldsymbol{u}_t \sim \text{Uniform}(0,1)}[f_\Delta(\boldsymbol{u}_t, \mathcal{T}_{\boldsymbol{w}_t}(\boldsymbol{b}_{t-1}), \boldsymbol{b}_{1:t-1})(\boldsymbol{u}_t - 1/2)] \nabla_{\boldsymbol{w}_t} \mathcal{T}_{\boldsymbol{w}_t}(\boldsymbol{b}_{t-1}) \right], \tag{35}$$

where

$$\begin{aligned} f_\Delta(\boldsymbol{u}_t, \mathcal{T}_{\boldsymbol{w}_t}(\boldsymbol{b}_{t-1}), \boldsymbol{b}_{1:t-1}) = {} & \mathbb{E}_{\boldsymbol{b}_{t+1:T} \sim q(\boldsymbol{b}_{t+1:T} \mid \boldsymbol{b}_t),\ \boldsymbol{b}_t = \mathbf{1}_{[\boldsymbol{u}_t > \sigma(-\mathcal{T}_{\boldsymbol{w}_t}(\boldsymbol{b}_{t-1}))]})}[f(\boldsymbol{b}_{1:T})] \\ & - \mathbb{E}_{\boldsymbol{b}_{t+1:T} \sim q(\boldsymbol{b}_{t+1:T} \mid \boldsymbol{b}_t),\ \boldsymbol{b}_t = \mathbf{1}_{[\boldsymbol{u}_t < \sigma(\mathcal{T}_{\boldsymbol{w}_t}(\boldsymbol{b}_{t-1}))]})}[f(\boldsymbol{b}_{1:T})] \end{aligned} \tag{36}$$

Finally, to compute the gradient with respect to $\boldsymbol{w}_T$, we have

$$\nabla_{\boldsymbol{w}_T} \mathcal{E}(\boldsymbol{w}_{1:T}) = \mathbb{E}_{q(\boldsymbol{b}_{1:T-1})} \left[\mathbb{E}_{\boldsymbol{u}_T \sim \text{Uniform}(0,1)}[f_\Delta(\boldsymbol{u}_T, \mathcal{T}_{\boldsymbol{w}_T}(\boldsymbol{b}_{T-1}), \boldsymbol{b}_{1:T-1})(\boldsymbol{u}_T - 1/2)] \nabla_{\boldsymbol{w}_T} \mathcal{T}_{\boldsymbol{w}_T}(\boldsymbol{b}_{T-1}) \right], \tag{37}$$

$$\begin{aligned} f_\Delta(\boldsymbol{u}_T, \mathcal{T}_{\boldsymbol{w}_T}(\boldsymbol{b}_{T-1}), \boldsymbol{b}_{1:T-1}) = {} & f(\boldsymbol{b}_{1:T-1}, \boldsymbol{b}_T = \mathbf{1}_{[\boldsymbol{u}_T > \sigma(-\mathcal{T}_{\boldsymbol{w}_T}(\boldsymbol{b}_{T-1}))]}) \\ & - f(\boldsymbol{b}_{1:T-1}, \boldsymbol{b}_T = \mathbf{1}_{[\boldsymbol{u}_T < \sigma(\mathcal{T}_{\boldsymbol{w}_T}(\boldsymbol{b}_{T-1}))]}) \end{aligned} \tag{38}$$

□

C MULTIPLE SAMPLES GRADIENT IN TOY EXAMPLE

The variance of Monte-Carlo estimation with sample n decreases at rate $\frac{1}{n}$. Comparing $K = 5000$ for REINFORCE and $K = 10$ for ARM in Figure 3 indicates in this example ARM gradient can reduce variance to around 1/500 of the REINFORCE one.

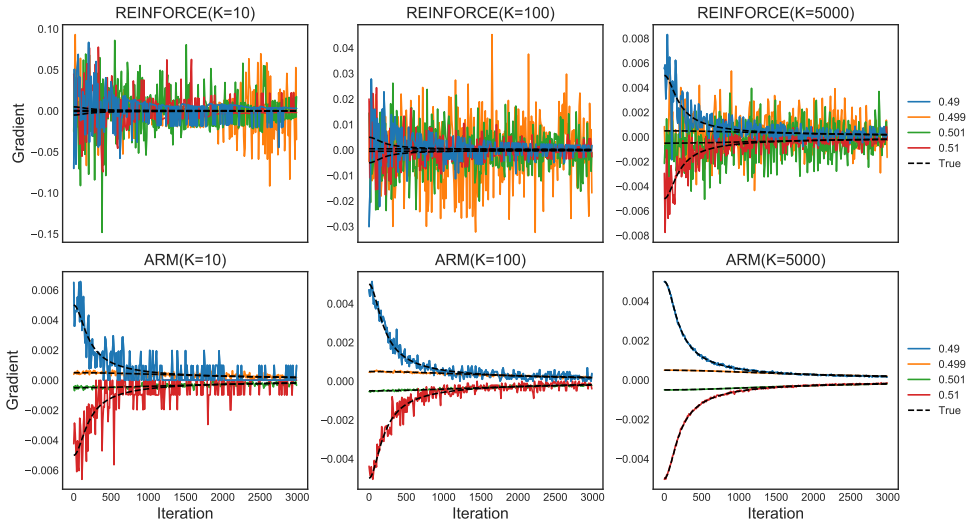


Figure 3: Estimation of the true gradient at each iteration using $K > 1$ Monte Carlo samples, using REINFORCE, shown in the top row, or ARM, shown in the bottom row. The ARM estimator exhibits significant lower variance given the same number of Monte Carlo samples.

D IMAGE COMPLETION AND VAE

The image completions in Figure 4 are obtained by random propagating the same upper half image through SBN for two times. The probability of the lower half digit pixels are ploted as the completion and suitble variations can be captured by the latent variables.

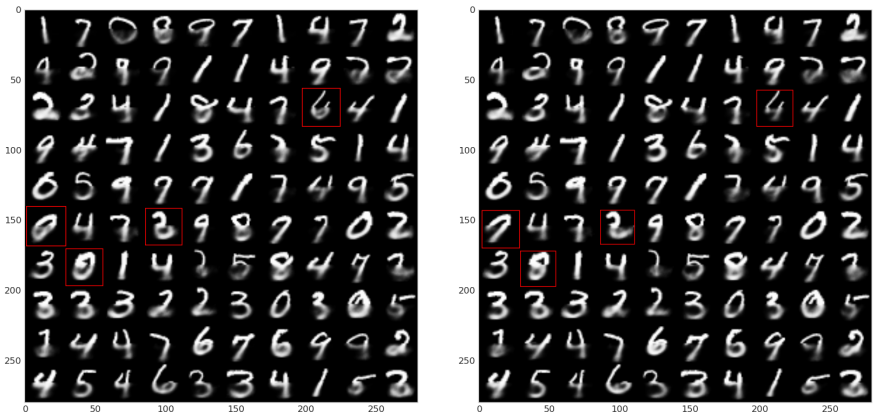
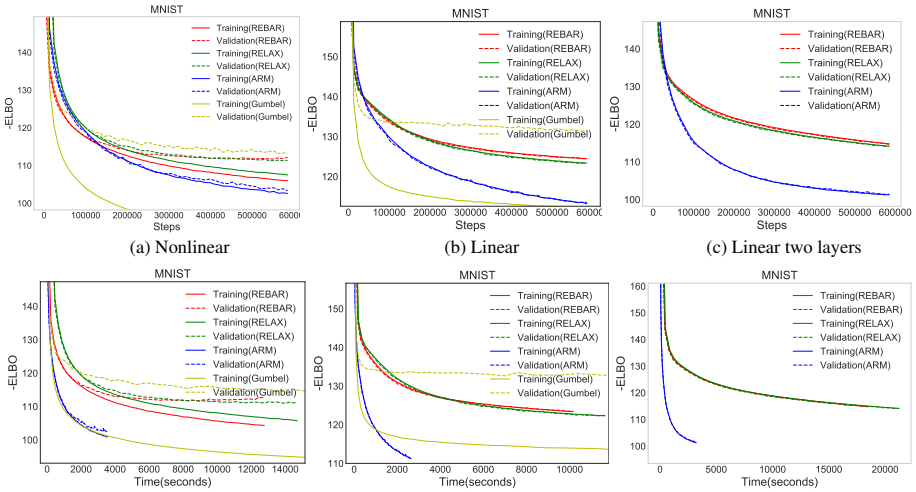


Figure 4: Randomly selected example results of predicting the lower half of a MNIST digit given its upper half, using a binary stochastic network, which has two binary linear stochastic hidden layers and is trained by ARM maximum likelihood inference. Red squares mark out notable variations between two random draws.

			ARM	RELAX	REBAR	ST Gumbel-Softmax
Bernoulli	Nonlinear	MNIST-threshold	**101.3**	110.9	111.6	112.5
		MNIST-static	**109.9**	112.1	111.8	-
		OMNIGLOT	129.5	**128.2**	128.3	140.7
	Linear	MNIST-threshold	**110.3**	122.1	123.2	129.2
		MNIST-static	**116.2**	116.7	117.9	-
		OMNIGLOT	**124.2**	124.4	124.9	129.8
	Two layers	MNIST-threshold	**98.2**	114.0	113.7	-
		MNIST-static	105.8	105.6	**105.5**	-
		OMNIGLOT	**118.3**	119.1	118.8	-

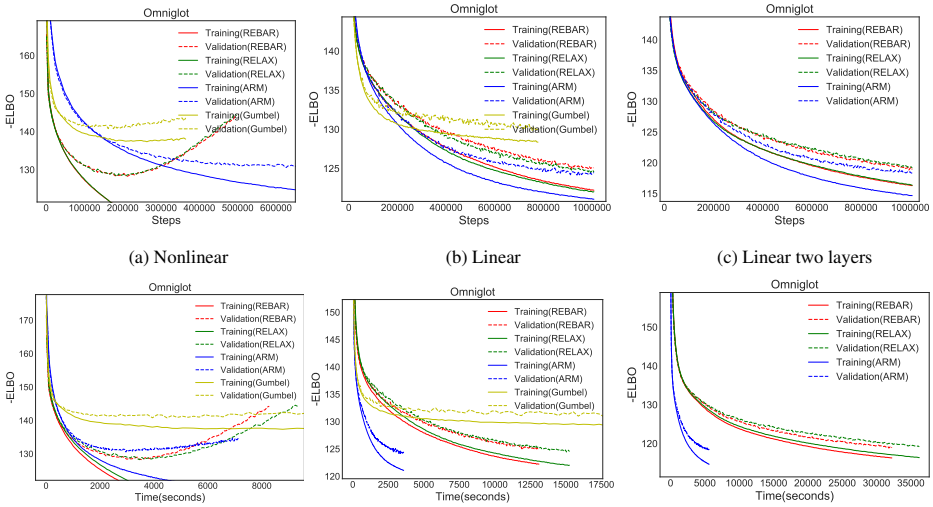
Table 4: Test negative ELBOs of discrete VAEs trained with four different stochastic gradient estimators. MNIST-threshold is the binarized MNIST thresholded at 0.5 and MNIST-static is the static binarized MNIST.



(a) Nonlinear (b) Linear (c) Linear two layers

(d) Nonlinear (e) Linear (f) Linear two layers

Figure 5: Test negative ELBOs on MNIST-threshold with respect to training iterations, shown in the top row, and wall clock times on Tesla-K40 GPU, shown in the bottom row, for three differently structured Bernoulli VAEs.



(a) Nonlinear (b) Linear (c) Linear two layers

(d) Nonlinear (e) Linear (f) Linear two layers

Figure 6: Test negative ELBOs on OMNIGLOT with respect to training iterations, shown in the top row, and wall clock times on Tesla-K40 GPU, shown in the bottom row, for three differently structured Bernoulli VAEs.