Gradient Estimation with Discrete Stein Operators

Anonymous Author(s)
Affiliation
Address
email

Abstract

Gradient estimation—approximating the gradient of an expectation with respect to the parameters of a distribution—is central to the solution of many machine learning problems. However, when the distribution is discrete, most common gradient estimators suffer from excessive variance. To improve the quality of gradient estimation, we introduce a variance reduction technique based on Stein operators for discrete distributions. We then use this technique to build flexible control variates for the REINFORCE leave-one-out estimator. Our control variates can be adapted online to minimize variance and do not require extra evaluations of the target function. In benchmark generative modeling tasks such as training binary variational autoencoders, our gradient estimator achieves substantially lower variance than state-of-the-art estimators with the same number of function evaluations.

1 Introduction

Modern machine learning relies heavily on gradient methods to optimize the parameters of a learning system. However, exact gradient computation is often difficult. For example, in variational inference for training latent variable models [29, 42] and policy gradient algorithms in reinforcement learning [63] the exact gradient features an intractable sum or integral introduced by an expectation under an evolving probability distribution. To make progress, one resorts to estimating the gradient by drawing samples from that distribution [39].

The two main classes of gradient estimators used in machine learning are the pathwise or reparameterization gradient estimators [29, 46, 56] and the REINFORCE or score function estimators [18, 63]. The pathwise estimators have shown great success in training variational autoencoders [29] but are only applicable to continuous probability distributions. The REINFORCE estimators are more general-purpose and easily accommodate discrete distributions but often suffer from excessively high variance.

To improve the quality of REINFORCE estimators, we develop a new variance reduction technique for discrete expectations. Our method is based upon Stein operators [see, e.g., 1, 53], computable functionals that generate mean-zero functions under a target distribution and provide a natural way of designing control variates (CVs) for stochastic estimation.

We first provide a general recipe for constructing practical Stein operators for discrete distributions (Table 1), generalizing the prior literature [6, 8, 9, 16, 25, 45, 65]. We then develop a gradient estimation framework—RODEO—that augments REINFORCE estimators with mean-zero CVs generated from Stein operators. Finally, inspired by Double CV [58], we extend our method to develop CVs for REINFORCE leave-one-out estimators [30, 48] to further reduce the variance.

The benefits of using Stein operators to construct discrete CVs are twofold. First, the operator structure permits us to learn CVs with a flexible functional form such as those parameterized by neural networks. Second, since our operators are derived from Markov chains on the discrete support, they naturally incorporate information from neighboring states of the process for variance reduction.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

Table 1: Discrete Stein operators that generate mean-zero functions under q (i.e., $\mathbb{E}_q[(Ah)(x)] = 0$).

Stein Operator	$(\boldsymbol{Ah})(\boldsymbol{x})$
Gibbs (7)	$\frac{1}{d}\sum_{i=1}^{d}\sum_{y_{-i}=x_{-i}} q(y_i\|x_{-i})h(y) - h(x)$
MPF (9)	$\sum_{y\in\mathcal{N}_x, y\neq x}\sqrt{q(y)/q(x)}(h(y)-h(x))$
Barker (9)	$\sum_{y\in\mathcal{N}_x, y\neq x}\frac{q(y)}{q(x)+q(y)}(h(y)-h(x))$
Difference (11)	$\frac{1}{d}\sum_{i=1}^{d} h(\texttt{dec}_i(x)) - \frac{q(\texttt{inc}_i(x))}{q(x)}h(x)$

We evaluate RODEO on 15 benchmark tasks, including training binary variational autoencoders (VAEs) with one or more stochastic layers. In most cases and with the same number of function evaluations, RODEO delivers lower variance and better training objectives than the state-of-the-art gradient estimators DisARM [14, 67], ARMS [13], Double CV [58], and RELAX [20].

2 Background

We consider the problem of maximizing the objective function $\mathbb{E}_{q_\eta}[f(x)]$ with respect to the parameters η of a discrete distribution $q_\eta(x)$. Throughout the paper we assume $f(x)$ is a differentiable function of real-valued inputs $x \in \mathbb{R}^d$ but is only evaluated at a discrete subset $\mathcal{X}^d$ due to the discreteness of q_η.[1] Exact computation of the expectation is typically intractable due to the complex nature of $f(x)$ and $q_\eta(x)$. The standard workaround is to rewrite the gradient as $\nabla_\eta \mathbb{E}_{q_\eta}[f(x)] = \mathbb{E}_{q_\eta}[f(x)\nabla_\eta \log q_\eta(x)]$ and employ the Monte Carlo gradient estimator known as the score function or REINFORCE estimator [18, 63]:

$$\frac{1}{K}\sum_{k=1}^{K}(f(x^{(k)}) - b)\nabla_\eta \log q_\eta(x^{(k)}), \quad x^{(1)}, \ldots, x^{(K)} \overset{i.i.d.}{\sim} q_\eta. \tag{1}$$

Here, b is a constant called the "baseline" introduced to reduce the variance of the estimator by reducing the scaling effect of $f(x^{(k)})$. Since $\mathbb{E}_{q_\eta}[\nabla_\eta \log q_\eta(x)] = 0$, the REINFORCE estimator is unbiased for any choice of b, and the term $b\nabla_\eta \log q_\eta(x)$ is known as a *control variate* (CV) [41, Ch. 8]. The optimal baseline can be estimated using additional function evaluations [7, 44, 62]. A simpler approach is to use moving averages of f from historical evaluations or to train function approximators to mimic those values [37]. When $K \geq 2$, a powerful variant of (1) is obtained by replacing b with the leave-one-out average of function values:

$$\frac{1}{K}\sum_{k=1}^{K}\left(f(x^{(k)}) - \frac{1}{K-1}\sum_{j\neq k} f(x^{(j)})\right)\nabla_\eta \log q_\eta(x^{(k)}). \tag{2}$$

This approach is often called the REINFORCE leave-one-out (RLOO) estimator [30, 47, 48] and was recently observed to have very strong performance in training discrete latent variable models [14, 47].

All the above methods construct a baseline that is independent of the point $x^{(k)}$ under consideration, but there are other ways to preserve the unbiasedness of the estimator. We are free to use a sample-dependent baseline $b(x^{(k)})$ as long as the expectation $c \triangleq \mathbb{E}_{q_\eta}[b(x^{(k)})\nabla_\eta \log q_\eta(x^{(k)})]$ is easily computed or has a low-variance unbiased estimate. In this case, we can correct for any bias introduced by $b(x^{(k)})$ by adding in this expectation: $\frac{1}{K}\sum_{k=1}^{K}(f(x^{(k)}) - b(x^{(k)}))\nabla_\eta \log q_\eta(x^{(k)}) + c$. For example, $b(x^{(k)})$ can be a lower bound or a Taylor expansion of $f(x^{(k)})$ [22, 42]. Taking this a step further, the Double CV estimator [58] proposed to treat $f(x^{(k)}) - b(x^{(k)})$ as the effective objective function and apply the leave-one-out idea:

$$\frac{1}{K}\sum_{k=1}^{K}\left[\left(f(x^{(k)}) - \boldsymbol{b_k(x^{(k)})}\right) - \frac{1}{K-1}\sum_{j\neq k}\left(f(x^{(j)}) - \boldsymbol{b_k(x^{(j)})}\right)\right]\nabla_\eta \log q_\eta(x^{(k)}) + c. \tag{3}$$

The resulting estimator adds two CVs to RLOO: the *global* CV $b_k(x^{(k)})\nabla_\eta \log q_\eta(x^{(k)})$ and the *local* CV $b_k(x^{(j)})$. Intuitively, $b_k(x^{(j)})$ is aimed at reducing the variance of the LOO average. This is motivated by the fact that replacing $\frac{1}{K-1}\sum_{j\neq k} f(x^{(j)})$ with $\mathbb{E}_{q_\eta}[f]$ in (2) leads to lower variance [58, Prop 1]. To obtain a tractable correction term c, Titsias and Shi [58] adopt a linear design of b: $b_k(x) = \frac{1}{K-1}\sum_{j\neq k} f(x_j)(x-\mu)$ for $\mu = \mathbb{E}_{q_\eta}[x]$.

[1] This assumption holds for many discrete probabilistic models [21] including the binary VAEs of Section 6.

Although the Double CV framework points to a promising new direction for developing better REINFORCE estimators, one only obtains significant reduction in variance when b_k is strongly correlated with f. This may fail to hold for the above linear b_k and a highly nonlinear f, especially in applications like training deep generative models.

In the following two sections, we will introduce a method that allows us to use very flexible CVs while still maintaining a tractable correction term. Our method enables online adaptation of CVs to minimize gradient variance (similar to RELAX [21]) but does not assume q_η has a continuous reparameterization. We then apply it to generalize the linear CVs in Double CV to very flexible ones such as neural networks. Moreover, we provide an effective CV design based on surrogate functions that requires no additional evaluation of f compared to RLOO.

3 Control Variates from Discrete Stein Operators

At the heart of our new estimator is a technique for generating flexible discrete CVs, that is for generating a rich class of functions that have known expectations under a given discrete distribution q. One way to achieve this is to identify any discrete-time Markov chain $(X^{(t)})_{t=0}^{\infty}$ with stationary distribution q. Then, the transition matrix P, defined via $P_{xy} = P(X^{(t+1)} = y | X^{(t)} = x)$, satisfies $P^\top q = q$ and hence

$$\mathbb{E}_q[(P - I)h] = 0, \tag{4}$$

for any integrable function h. We overload our notation so that, for any suitable matrix A, $(Ah)(x) \triangleq \sum_y A_{xy} h(y)$. In other words, for any integrable h, the function $(P - I)h$ is a valid CV as it has known mean under q. Moreover, the linear operator $P - I$ is an example of a *Stein operator* [1, 53] in the sense that it generates mean-zero functions under q. In fact, both Stein et al. [54] and Dellaportas and Kontoyiannis [12] developed CVs of the form $(P - I)h$ based on reversible discrete-time Markov chains and linear input functions $h(x) = x_i$.

More generally, Barbour [3] and Henderson [23] observed that if we identify a continuous-time Markov chain $(X^{(t)})_{t \geq 0}$ with q as its stationary distribution, then the *generator* A, defined via

$$(Ah)(x) = \lim_{t \to 0} \frac{\mathbb{E}[h(X^{(t)})] - h(x)}{t}, \tag{5}$$

satisfies $A^\top q = 0$ and hence $\mathbb{E}_q[Ah] = 0$ for all integrable h. Therefore, A is also a Stein operator suitable for generating CVs. Moreover, since any discrete-time chain with transition matrix P can be embedded into a continuous-time chain with transition rate matrix $A = P - I$, this continuous-time construction is strictly more general [55, Ch. 4].

3.1 Discrete Stein operators

We next present several examples of broadly applicable discrete Stein operators (summarized in Table 1) that can serve as practical defaults.

Gibbs Stein operator The transition kernel of the random-scan Gibbs sampler with stationary distribution q [see, e.g., 17] is

$$P_{xy} = \tfrac{1}{d} \textstyle\sum_{i=1}^{d} q(y_i | x_{-i}) \mathbf{1}(y_{-i} = x_{-i}), \tag{6}$$

where $\mathbf{1}(\cdot)$ is the indicator function. The associated Stein operator is $A = P - I$ with

$$(Ah)(x) = \tfrac{1}{d} \textstyle\sum_{i=1}^{d} \sum_{y_{-i} = x_{-i}} q(y_i | x_{-i}) h(y) - h(x). \tag{7}$$

In the binary variable setting, (7) recovers the operator Bresler and Nagaraj [8], Reinert and Ross [45] used to bound distances between the stationary distributions of Glauber dynamics Markov chains.

Zanella Stein operator Zanella [68] studied continuous-time Markov chains with generator

$$A_{xy} = \kappa \left(q(y)/q(x) \right) \mathbf{1}(y \in \mathcal{N}_x, y \neq x) - \textstyle\sum_{z \neq x} A_{xz} \, \mathbf{1}(y = x), \tag{8}$$

where $\mathcal{N}_x$ denotes the transition neighborhood of x and κ is a continuous positive function satisfying $\kappa(t) = t\kappa(1/t)$. Conveniently, the neighborhood structure can be arbitrarily sparse. Moreover, the κ

constraint ensures that the chain satisfies detailed balance (i.e., $q(x)A_{xy} = q(y)A_{yx}$) and hence that $A^\top q = 0$. Hodgkinson et al. [24] call the associated operator the *Zanella Stein operator*:

$$(Ah)(x) = \sum_{y \in \mathcal{N}_x, y \neq x} \kappa\big(\tfrac{q(y)}{q(x)}\big)(h(y) - h(x)). \tag{9}$$

Important special cases include the *minimum probability flow [51] (MPF) Stein operator* ($\kappa(t) = \sqrt{t}$) discussed in Barp et al. [6] and the *Barker [4] Stein operator* ($\kappa(t) = \frac{t}{t+1}$).

Birth-death Stein operator Let $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_d$ represent the standard basis vectors on $\mathbb{R}^d$. For any finite-cardinality space, we may index the elements of $\mathcal{X} = \{s_0, \ldots, s_{m-1}\}$, let $\texttt{idx}(x_i)$ return the index of the element that x_i represents, and define the increment and decrement operators

$$\texttt{inc}_i(x) = x + \boldsymbol{e}_i(s_{(\texttt{idx}(x_i)+1) \bmod m} - x_i) \quad \text{and} \quad \texttt{dec}_i(x) = x + \boldsymbol{e}_i(s_{(\texttt{idx}(x_i)-1) \bmod m} - x_i).$$

Then the product *birth-death process* [28] on $\mathcal{X}^d$ with birth rates $b_{i,x} = \frac{q(\texttt{inc}_i(x))}{q(x)}$, death rates $d_{i,x} = 1$, and generator

$$A_{xy} = \tfrac{1}{d}\textstyle\sum_{i=1}^d b_{i,x}\mathbf{1}(y = \texttt{inc}_i(x)) + d_{i,x}\mathbf{1}(y = \texttt{dec}_i(x)) - (b_{i,x} + d_{i,x})\mathbf{1}(y = x),$$

has q as a stationary distribution, as $A^\top q = 0$. This construction yields the birth-death Stein operator

$$(Ag)(x) = \tfrac{1}{d}\textstyle\sum_{i=1}^d b_{i,x}(g(\texttt{inc}_i(x)) - g(x)) - d_{i,x}(g(x) - g(\texttt{dec}_i(x))). \tag{10}$$

An analogous operator is available for countably infinite $\mathcal{X}$ [24], and Brown and Xia [9], Eichelsbacher and Reinert [16], Holmes [25] used related operators to characterize convergence to discrete target distributions. Moreover, by substituting $h(x) = g(x) - g(\texttt{inc}(x))$ in (10), we recover the *difference Stein operator* proposed by Yang et al. [65] without reference to birth-death processes:

$$(Ah)(x) = \tfrac{1}{d}\textstyle\sum_{i=1}^d h(\texttt{dec}_i(x)) - b_{i,x}h(x). \tag{11}$$

Choosing a Stein operator Despite being better-known, the difference and MPF operators often suffer from large variance and numerical instability due to their use of unbounded probability ratios $q(y)/q(x)$. As a result, we recommend the numerically stable Gibbs operator when each component of x is binary or takes on a small number of values. When x_i takes on a large number of values (m), the Gibbs operator suffers from linear scaling with m. In this case, we recommend the Barker operator where a sparse neighborhood structure can be specified, such as $\mathcal{N}_x = \{\texttt{inc}_i(x), \texttt{dec}_i(x) \text{ for } i \in [d]\}$. The Barker operator is numerically stable as its $\kappa(\frac{q(y)}{q(x)}) = \frac{q(y)}{q(x)+q(y)}$.

4 Gradient Estimation with Discrete Stein Operators

Recall that REINFORCE estimates the gradient $\mathbb{E}_{q_\eta}[f(x)\nabla_\eta \log q_\eta(x)]$. Due to the existence of $\nabla_\eta \log q_\eta(x)$ as a weighting function, we apply a discrete Stein operator to a vector-valued function $\tilde{h} : \mathcal{X} \to \mathbb{R}^d$ per dimension to construct the following estimator with a mean-zero CV:

$$\mathbb{E}_{q_\eta}[f(x)\nabla_{\eta_i} \log q_\eta(x) + (A\tilde{h}_i)(x)]. \tag{12}$$

Ideally, we want to choose the $\tilde{h}$ such that $A\tilde{h}_i$ will be strongly correlated with $f(x)\nabla_{\eta_i} \log q_\eta(x)$ to reduce its variance. The optimal $\tilde{h}_i$ is given by the solution of Poisson equation

$$\mathbb{E}_{q_\eta}[f\nabla_{\eta_i} \log q_\eta] - f\nabla_{\eta_i} \log q_\eta = A\tilde{h}_i. \tag{13}$$

We could learn a separate $\tilde{h}_i$ per dimension to approximate the solution of (13). However, this poses a difficult optimization problem that requires simultaneously solving d Poisson equations. Instead, we will incorporate knowledge about the structure of the solution to simplify the optimization.

To determine a candidate functional form for $\tilde{h}$, we consider an "optimal" Markov chain which generates i.i.d. sample points from q_η. In this case, $(P - I)\tilde{h}_i$ becomes $\mathbb{E}_{q_\eta}[\tilde{h}_i] - \tilde{h}_i$, and the optimal solution is $\tilde{h}_i = f\nabla_{\eta_i} \log q_\eta$. Inspired by this, we consider $\tilde{h}$ of the form

$$\tilde{h}(x) = h(x)\nabla_\eta \log q_\eta(x), \tag{14}$$

Algorithm 1 Optimizing $\mathbb{E}_{q_\eta}[f_\theta(x)]$ with RODEO gradients

input: Objective f_θ, sample points $x^{(1:K)} \overset{i.i.d.}{\sim} q_\eta$, Stein operator A, step sizes α_t, β_t
for $t = 1 : T$ **do**
1: $\{f_\theta(x^{(k)}), \nabla_\theta f_\theta(x^{(k)}), \nabla f_\theta(x^{(k)})\}_{k=1}^K \leftarrow$ autodiff$(f_\theta, x^{(1:K)})$.
2: Compute the surrogate $h_k(x^{(j)}), h_k^*(x^{(j)})$ of (16), (18) **for** $j \neq k$ and $j, k = 1, \cdots, K$.
3: Compute the RODEO gradient estimator $g_\gamma(x^{(1:K)})$ using (17).
4: $\theta \leftarrow \theta + \alpha_t \frac{1}{K} \sum_{k=1}^K \nabla_\theta f(x^{(k)})$.
5: $\eta \leftarrow \eta + \alpha_t g_\gamma(x^{(1:K)})$.
6: Update hyperparameters: $\gamma \leftarrow \gamma - \beta_t \nabla_\gamma \|g_\gamma(x^{(1:K)})\|_2^2$.

where we now only need to learn a scalar-valued function h. Notably, when h exactly equals f and $A = P - I$ for any discrete time Markov chain kernel P, our CV adjustment amounts to Rao-Blackwellization [41, Sec. 8.7], as we end up replacing $f(x)\nabla_{\eta_i} \log q_\eta(x)$ with its conditional expectation $P(f\nabla_{\eta_i} \log q_\eta)(x) = \mathbb{E}_{X_{t+1}|X_t=x}[f(X_{t+1})\nabla_{\eta_i} \log q_\eta(X_{t+1})]$. This yields a guaranteed variance reduction.

Surrogate function design Based on the above reasoning, we can view h as a surrogate for f. We avoid directly setting $h = f$ because our Stein operators evaluate h at all neighbors of the sample points, and f can be very expensive to evaluate [see, e.g., 49]. To avoid this problem, we first observe that $\tilde{h}$ (14) can be made sample-specific, i.e., we can use a different $\tilde{h}_k$ for each sample point $x^{(k)}$:

$$\frac{1}{K} \sum_{k=1}^K [f(x^{(k)})\nabla_\eta \log q_\eta(x^{(k)}) + (A\tilde{h}_k)(x^{(k)})]. \tag{15}$$

We then consider the following choices of h_k that are informed about f while being cheap to evaluate:

$$h_k(y) = \frac{1}{K-1} \sum_{j \neq k} H(f(x^{(j)}), \nabla f(x^{(j)})^\top (y - x^{(j)})). \tag{16}$$

Here y will take on the values of $x^{(k)}$ and its neighbors in the Markov chain. We omit $x^{(k)}$ in the sum (16) to ensure that the function h_k is independent of $x^{(k)}$ and hence that $\mathbb{E}_{q_\eta}[\frac{1}{K} \sum_{k=1}^K (A\tilde{h}_k)(x^{(k)})] = 0$. Moreover, this surrogate function design introduces no additional evaluations of f beyond those required for the usual RLOO estimator. Also, as observed by Titsias and Shi [58], for many applications, including VAE training, where f has parameters θ learned through gradient-based optimization, $\{\nabla f(x^{(k)})\}_{k=1}^K$ can be obtained "for free" from the same backpropagation used to compute $\nabla_\theta f_\theta(x^{(k)})$ (see Algorithm 1).

RODEO We can further improve the estimator by leveraging discrete Stein operators and the above surrogate function design to construct both the global and local CVs in the Double CV framework [58] (see Section 2). We call our final estimator **RODEO** for **R**L**O**O with **D**iscrete St**E**in **O**perators:

$$\frac{1}{K} \sum_{k=1}^K [(f(x^{(k)}) - \frac{1}{K-1} \sum_{j \neq k} (f(x^{(j)}) + (Ah_j)(x^{(j)})))\nabla_\eta \log q_\eta(x^{(k)}) + (A\tilde{h}_k^*)(x^{(k)})], \tag{17}$$

where $\tilde{h}_k^*(y) = h_k^*(y)\nabla_\eta \log q_\eta(y)$ and $\{h_k, h_k^*\}_{k=1}^K$ are scalar-valued functions. Here, $(Ah_j)(x^{(j)})$ is a scalar-valued CV introduced to reduce the variance of the leave-one-out baseline $\frac{1}{K-1} \sum_{j \neq k} f(x^{(j)})$, while $(A\tilde{h}_k^*)(x^{(k)})$ acts as a global CV to further reduce the variance of the gradient estimate. We adopt the aforementioned design of h (16) and $\tilde{h}$ (14) for the two CVs. In Appendix B, we show that the RODEO estimator (17) is unbiased for $\nabla_\eta \mathbb{E}_{q_\eta}[f(x)]$ when each h_k is defined as in (16) and

$$h_k^*(y) = \frac{1}{K-1} \sum_{j \neq k} H^*(f(x^{(j)}), \nabla f(x^{(j)})^\top (y - x^{(j)})). \tag{18}$$

Optimization with RODEO In practice, we let the functions H and H^* share a neural network architecture with two output units. Since the estimator is unbiased, we can optimize the parameters γ of the network in an online fashion to minimize the variance of the estimator (similar to [20]). Specifically, denoting the RODEO gradient estimate (17) by $g_\gamma(x^{(1:K)})$, then $\nabla_\gamma \mathrm{Tr}(\mathrm{Var}(g_\gamma(x^{(1:K)}))) = \mathbb{E}[\nabla_\gamma \|g_\gamma(x^{(1:K)})\|_2^2]$. In Algorithm 1, we use an unbiased estimate of this hyperparameter gradient, $\nabla_\gamma \|g_\gamma(x^{(1:K)})\|_2^2$, to update γ after each optimization step of η.

Table 2: Training binary latent VAEs with $K = 2, 3$ (except for RELAX which uses 3 evaluations) on MNIST, Fashion-MNIST, and Omniglot. We report the average ELBO (± 1 standard error) on the training set after 1M steps over 5 independent runs. Test data bounds are reported in Table 4.

		Bernoulli Likelihoods			Gaussian Likelihoods		
		MNIST	Fashion-MNIST	Omniglot	MNIST	Fashion-MNIST	Omniglot
$K = 2$	DisARM	-102.75 ± 0.08	-237.68 ± 0.13	-116.50 ± 0.04	668.03 ± 0.61	182.65 ± 0.47	446.61 ± 1.16
	Double CV	-102.14 ± 0.06	-237.55 ± 0.16	-116.39 ± 0.10	676.87 ± 1.18	186.35 ± 0.64	447.65 ± 0.87
	RODEO (Ours)	$\mathbf{-101.89 \pm 0.17}$	$\mathbf{-237.44 \pm 0.09}$	$\mathbf{-115.93 \pm 0.06}$	$\mathbf{681.95 \pm 0.37}$	$\mathbf{191.81 \pm 0.67}$	$\mathbf{454.74 \pm 1.11}$
$K = 3$	ARMS	-100.84 ± 0.14	-237.05 ± 0.12	-115.21 ± 0.07	683.55 ± 1.01	193.07 ± 0.34	457.98 ± 1.03
	Double CV	-100.94 ± 0.09	-237.40 ± 0.11	-115.06 ± 0.12	686.48 ± 0.68	193.93 ± 0.20	457.44 ± 0.79
	RODEO (Ours)	$\mathbf{-100.46 \pm 0.13}$	$\mathbf{-236.88 \pm 0.12}$	$\mathbf{-115.01 \pm 0.05}$	$\mathbf{692.37 \pm 0.39}$	$\mathbf{196.56 \pm 0.42}$	461.87 ± 0.90
RELAX (3 evals)		-101.99 ± 0.04	-237.74 ± 0.12	-115.70 ± 0.08	688.58 ± 0.52	196.38 ± 0.66	$\mathbf{462.23 \pm 0.63}$

5 Related Work

As we have seen in Section 2, there is a long history of designing CVs for REINFORCE estimators using "baselines" [7, 37, 44, 62]. Recent progress is mostly driven by leave-one-out [30, 38, 47, 48] and sample-dependent baselines [20, 22, 42, 58, 60]. REBAR [60] constructs the baseline through continuous relaxation of the discrete distribution [26, 35] and applies reparameterization gradients to the correction term. As a result REBAR uses three evaluations of f for each $x^{(k)}$ instead of usual single evaluation (see Appendix A for a detailed explanation). The RELAX [20] estimator generalizes REBAR by noticing that their continuous relaxation can be replaced with a free-form CV. However, in order to get strong performance, RELAX still includes the continuous relaxation in their CV and only adds a small deviation to it. Therefore, RELAX also uses three evaluations of f for each $x^{(k)}$ and is usually considered more expensive than other estimators.

An attractive property of the RODEO estimator (17) is that it incorporates information from neighboring states thanks to the Stein operator while avoiding additional costly f evaluations thanks to the learned surrogate functions h_k. Estimators based on analytic local expectation [57, 59] and GO gradients [11] also use neighborhood information but only at the cost of many additional target function evaluations. In fact, the local expectation gradient [57] can be viewed as a Stein CV adjustment (17) with a Gibbs Stein operator and the target function f used directly instead of the surrogate h. The downside of these approaches is that f must be evaluated Kd times per training step instead of K times as in RODEO, a prohibitive cost when f is expensive and $d \geq 200$ as in Section 6.

Prior work has also studied variance reduction methods based on sampling without replacement [31] and antithetic sampling [13–15, 66] for gradient estimation. DisARM [14, 67] was shown to outperform RLOO estimators when $K = 2$ and ARMS [13] generalizes DisARM to $K > 2$.

Stein operators for continuous distributions have also been leveraged for effective variance reduction in a variety of learning tasks [2, 5, 36, 40, 50, 52] including gradient estimation [27, 34]. In particular, the gradient estimator of Liu et al. [34] is based on the Langevin Stein operator [19] for continuous distributions and coincides with the continuous counterpart of RELAX [20]. In contrast, our approach considers discrete Stein operators for Monte Carlo estimation in discrete distributions with exponentially large state spaces. Recently, Parmas and Sugiyama [43, App. E.4] used a probability flow perspective to characterize *all* unbiased gradient estimators satisfying a mild technical condition; our estimators fall into this broad class but were not specifically investigated.

6 Experiments

Python code replicating all experiments can be found in the supplementary material.

6.1 Training Bernoulli VAEs

Following Dong et al. [14] and Titsias and Shi [58], we conduct experiments on training variational auto-encoders [29, 46] (VAEs) with Bernoulli latent variables. VAEs are models with a joint density $p_\theta(y, x)$, where x is the latent variable and θ denotes model parameters. They are typically learned through maximizing the *evidence lower bound* (ELBO) $\mathbb{E}_{q_\eta(x|y)}[f(x)]$ for an auxiliary inference

network $q_\eta(x|y)$ and $f(x) \triangleq \log p_\theta(y, x) - \log q_\eta(x|y)$. In our experiments, $p(x)$ and $q_\eta(x|y)$ are high-dimensional distributions where each dimension of the random variable is an independent Bernoulli. Since exact gradient computations are intractable, we will use gradient estimators to learn the parameters η of the inference network. The VAE architecture and training experimental setup follows Titsias and Shi [58], and details are given in Appendix D. The dimensionality of the latent variable x is $d = 200$. The functions H (16) and H' (18) share a neural network architecture with two output units and a single hidden layer with 100 units. For the numerical stability and variance reasons discussed in Section 3, we use the Gibbs Stein operator (7) as a default choice in our experiments, but we revisit this choice in Section 6.3.

We consider the MNIST [33], Fashion-MNIST [64] and Omniglot [32] datasets using their standard train, validation, and test splits. We use both binary and continuous VAE outputs (y) as in Titsias and Shi [58]. In the binary output setting, data are dynamically binarized, and the Bernoulli likelihood is used; in the continuous output setting, data are centered between $[-1, 1]$, and the Gaussian likelihood with learnable diagonal covariance parameters is used.

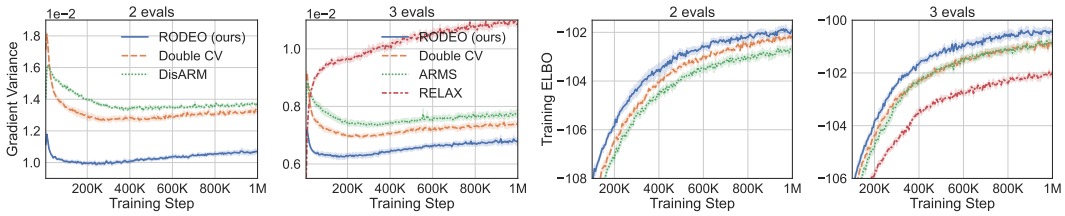


Figure 1: Training binary latent VAEs with 2 or 3 f evaluations per step on binarized MNIST.

$\boldsymbol{K = 2}$ In the first set of experiments we focus on the most common setting of $K = 2$ sample points and compare our variance reduction method to its counterparts including DisARM [14] and Double CV [58]. Results for RLOO are omitted since it is consistently outperformed by Double CV [see 58]. Table 2 shows, on all three datasets, RODEO achieves the best training ELBOs. In Figure 1 (left), we plot the gradient variance and average training ELBOs against training steps for all estimators on dynamically binarized MNIST. RODEO outperforms DisARM and Double CV by a large margin in gradient variance.

Next, we consider VAEs with Gaussian likelihoods trained on non-binarized datasets. In this case the gradient estimates suffer from even higher variance due to the large range of values $f(x)$ can take. The results are plotted in Figure 2. We see that RODEO has substantially lower variance than DisARM and Double CV, leading to significant improvements in training ELBOs. In Appendix Figure 7, we find that RODEO also consistently yields the best test set performance in all six settings.

$\boldsymbol{K = 3}$ In the second set of experiments we compare RELAX [20], which uses three evaluations of f per training step, with RODEO, Double CV, and ARMS [13] for $K = 3$. Figure 1 (right) and Table 2 demonstrate that RODEO outperforms the three previous methods and generally leads to the lowest variance. Although RELAX was often observed to have very strong performance in prior work [14, 58], our results in Figure 1 suggest that, for dynamically binarized datasets, much larger gains can be achieved by using the same number of function evaluations in other estimators.

For the experiments mentioned above, we report final training ELBOs in Table 2, test log-likelihood bounds in Appendix Table 4, and binarized MNIST average running time in Appendix Table 5. For $K = 3$, RODEO has the best final performance in 5 out of 6 tasks and runtime nearly identical to RELAX. For $K = 2$, RODEO has the best final performance for all tasks and is 1.5 times slower than Double CV and DisARM. We attribute the runtime gap to the Gibbs operator (7) which performs $2d$ evaluations of the auxiliary functions H and H' in (16) and (18). While this results in some extra cost, the network parameterizing H and H' takes only 2-dimensional inputs, produces scalar outputs, and is typically small relative to the VAE model. As a result, the evaluation of H and H' is significantly cheaper than that of f, and its relative contribution to runtime shrinks as the cost of f grows.

6.2 Training hierarchical Bernoulli VAEs

To investigate the performance of RODEO when scaling to hierarchical discrete latent variable models, we follow DisARM [14, 66] to train VAEs with 2/3/4 stochastic layers, each of which consists of

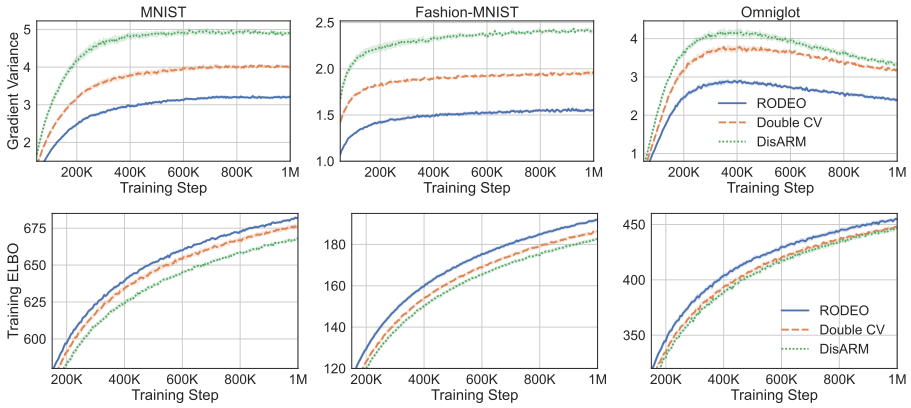


Figure 2: Training binary latent VAEs with Gaussian likelihoods, $K = 2$, and non-binarized datasets.

200 Bernoulli variables. We set $K = 2$ and compare our estimator with DisARM and Double CV on dynamically binarized MNIST, Fashion-MNIST, and Omniglot. For each stochastic layer, we use a different CV network which has the same architecture as those in our VAE experiments from the previous section. More details are presented in Appendix D.

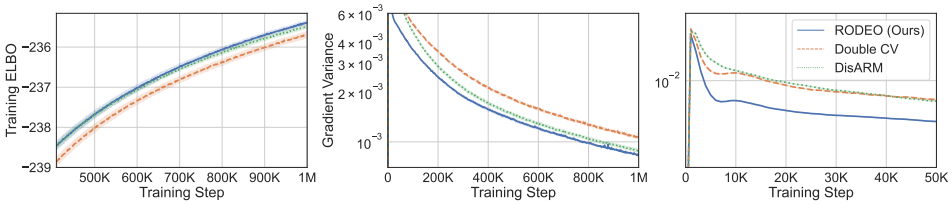


Figure 3: Training hierarchical binary latent VAEs with four stochastic layers on Fashion-MNIST. In this experiment, the estimators have very different behaviors towards the beginning and the end of training. We show this on the right by zooming into the first 50K steps of the gradient variance plot.

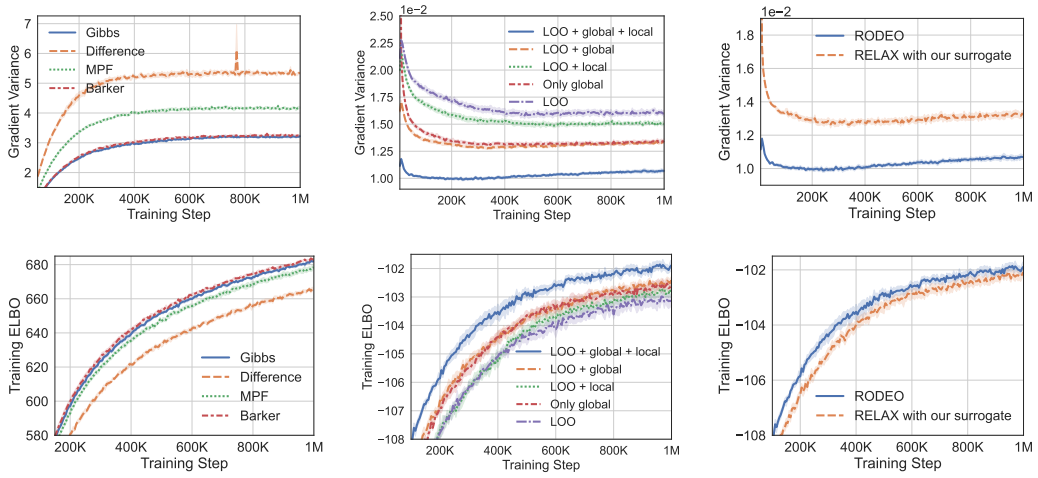
We plot the results on VAEs with four stochastic layers on Fashion-MNIST in Figure 3. The results for other datasets and for 2 and 3 stochastic layers can be found in Appendix E. RODEO generally achieves the best training ELBOs. One difference we noticed when comparing the variance results with those obtained from single-stochastic-layer VAEs is that these estimators have very different behaviors towards the beginning and the end of training. For example, in Figure 3, Double CV starts with lower variance than DisARM, but the gap diminishes after around 100K steps, and DisARM start to perform better as the training proceeds. In contrast, RODEO has the lowest variance in both phases for all datasets save Omniglot, where DisARM overtakes it in the long run.

6.3 Ablation study

Finally, we conduct an ablation study to gain more insight into the impact of each component of RODEO. In each experiment we train binary latent VAEs with $K = 2$.

Impact of Stein operator Figure 4a explores the impact of RODEO Stein operator choice on non-binarized MNIST. As expected, the less stable difference (11) and MPF (9) operators lead to significantly higher gradient variances and worse training ELBOs. In fact, the same operators led to divergent training on binarized MNIST. The Barker operator (9) with neighborhoods defined by having one differing coordinate yields results very similar to Gibbs (7) as the operators themselves are very similar for binary variables (notice that $\frac{q(y)}{q(y)+q(x)} = q(y_i|x_{-i})$ when $x_{-i} = y_{-i}$).

Impact of LOO baseline and global and local CVs Figure 4b explores binarized MNIST performance when RODEO (17) is modified to retain a) only the LOO baseline (this is equivalent to standard RLOO), b) only the global CV, c) the LOO baseline + the global CV, or d) the LOO baseline + the local CV. We observe that the global and local Stein CVs both contribute to variance reduction



(a) (b) (c)

Figure 4: Ablation study of impact of RODEO components: (a) Stein operators, (b) LOO baseline and global and local CVs, (c) surrogate functions on binary VAE training performance.

and have complementary effects. Moreover, remarkably, the global Stein CV alone outperforms RLOO.

Impact of surrogate functions To tease apart the benefits of our new surrogate functions (16) and the remaining RODEO components, we replace the surrogate function $c_\phi(z)$ in RELAX [21] with our surrogates, which only requires a single evaluation of f per $x^{(k)}$ (see Appendix A for more details). Figure 4c compares the performance of RODEO and this modified RELAX on binarized MNIST. Since the surrogate functions are matched, the consistent improvements over modified RELAX can be attributed to the Stein and double CV components of RODEO.

7 Conclusions, Limitations, and Future Work

This work tackles the gradient estimation problem for discrete distributions. We proposed a variance reduction technique which exploits Stein operators to generate control variates for REINFORCE leave-one-out estimators. Our RODEO estimator does not rely on continuous reparameterization of the distribution, requires no additional function evaluations per sample point, and can be adapted online to learn very flexible control variates parameterized by neural networks.

One potential drawback of our surrogate function constructions (16) and (18) is the need to evaluate an auxiliary function (H or H') at $K-1$ locations. This cost can be comfortably borne when H and H' are much cheaper to evaluate than f, and, in our experiments with the most common sample sizes, the runtime of RODEO was no worse than that of RELAX. To obtain a more favorable cost for large K, one could employ alternative surrogates that require only a constant number of auxiliary function evaluations, e.g.,

$$h_k(y) = H\big(\tfrac{1}{K-1}\textstyle\sum_{j\neq k} f(x^{(j)}), \tfrac{1}{K-1}\textstyle\sum_{j\neq k} \nabla f(x^{(j)})^\top (y - x^{(j)})\big).$$

The runtime of RODEO could also be improved by varying the Stein operator employed. For example, the Gibbs operator $(Ah)(x)$ (7) used in our experiments evaluated its surrogate function h at d neighboring locations of x. This evaluation complexity could be reduced by subsampling neighbors, resulting in a cheaper but still valid Stein operator, or by employing the numerically stable Barker operator (9) with fewer neighbors. Either strategy would introduce a speed-variance trade-off worthy of study in follow-up work. Finally, we have restricted our focus in this work to differentiable target functions f. In future work, this limitation could be overcome by designing effective surrogate functions that make no use of derivative information.

References

[1] Andreas Anastasiou, Alessandro Barp, François-Xavier Briol, Bruno Ebner, Robert E Gaunt, Fatemeh Ghaderinezhad, Jackson Gorham, Arthur Gretton, Christophe Ley, Qiang Liu, and Lester Mackey. Stein's method meets statistics: A review of some recent developments. *arXiv preprint arXiv:2105.03481*, 2021.

[2] Roland Assaraf and Michel Caffarel. Zero-variance principle for Monte Carlo algorithms. *Phys. Rev. Lett.*, 83:4682–4685, Dec 1999.

[3] A. D. Barbour. Stein's method and Poisson process convergence. *J. Appl. Probab.*, (Special Vol. 25A): 175–184, 1988. ISSN 0021-9002. A celebration of applied probability.

[4] Av A Barker. Monte Carlo calculations of the radial distribution functions for a proton? electron plasma. *Australian Journal of Physics*, 18(2):119–134, 1965.

[5] Alessandro Barp, Chris Oates, Emilio Porcu, and Mark Girolami. A Riemann-Stein kernel method. *arXiv preprint arXiv:1810.04946*, 2018.

[6] Alessandro Barp, Francois-Xavier Briol, Andrew Duncan, Mark Girolami, and Lester Mackey. Minimum Stein discrepancy estimators. *Advances in Neural Information Processing Systems*, 32:12964–12976, 2019.

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

[8] Guy Bresler and Dheeraj Nagaraj. Stein's method for stationary distributions of Markov chains and application to Ising models. *The Annals of Applied Probability*, 29(5):3230–3265, 2019.

[9] Timothy C. Brown and Aihua Xia. Stein's Method and Birth-Death Processes. *The Annals of Probability*, 29(3):1373 – 1403, 2001.

[10] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *International Conference on Learning Representations*, 2015.

[11] Yulai Cong, Miaoyun Zhao, Ke Bai, and Lawrence Carin. GO gradient for expectation-based objectives. In *International Conference on Learning Representations*, 2019.

[12] Petros Dellaportas and Ioannis Kontoyiannis. Control variates for estimation based on reversible Markov chain Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(1):133–161, 2012.

[13] Aleksandar Dimitriev and Mingyuan Zhou. ARMS: Antithetic-REINFORCE-Multi-Sample gradient for binary variables. In *International Conference on Machine Learning*, volume 139, pages 2717–2727, 2021.

[14] Zhe Dong, Andriy Mnih, and George Tucker. DisARM: An antithetic gradient estimator for binary latent variables. In *Advances in Neural Information Processing Systems*, volume 33, pages 18637–18647, 2020.

[15] Zhe Dong, Andriy Mnih, and George Tucker. Coupled gradient estimators for discrete latent variables. *arXiv preprint arXiv:2106.08056*, 2021.

[16] Peter Eichelsbacher and Gesine Reinert. Stein's method for discrete Gibbs measures. *The Annals of Applied Probability*, 18(4):1588–1618, 2008.

[17] Charles J. Geyer. Introduction to Markov chain Monte Carlo. In Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng, editors, *Handbook of Markov Chain Monte Carlo*, pages 3–48. CRC Press, 2011.

[18] Peter W. Glynn. Likelihood ratio gradient estimation for stochastic systems. *Commun. ACM*, 33(10): 75–84, October 1990.

[19] Jackson Gorham and Lester Mackey. Measuring sample quality with Stein's method. In *Advances in Neural Information Processing Systems*, pages 226–234, 2015.

[20] W. Grathwohl, D. Choi, Y. Wu, G. Roeder, and D. Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *International Conference on Learning Representations*, 2018.

[21] Will Grathwohl, Kevin Swersky, Milad Hashemi, David Duvenaud, and Chris Maddison. Oops I took a gradient: Scalable sampling for discrete distributions. In *International Conference on Machine Learning*, pages 3831–3841, 2021.

[22] Shixiang Gu, Sergey Levine, Ilya Sutskever, and Andriy Mnih. MuProp: Unbiased backpropagation for stochastic neural networks. In *International Conference on Learning Representations*, 2016.

[23] Shane G Henderson. *Variance reduction via an approximating Markov process*. PhD thesis, Stanford University, 1997.

[24] Liam Hodgkinson, Robert Salomone, and Fred Roosta. The reproducing stein kernel approach for post-hoc corrected sampling. *arXiv preprint arXiv:2001.09266*, 2020.

[25] Susan Holmes. Stein's method for birth and death chains. In Persi Diaconis and Susan Holmes, editors, *Stein's Method: Expository Lectures and Applications*, pages 42–65. 2004.

[26] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-softmax. *International Conference on Learning Representations*, 2017.

[27] Martin Jankowiak and Fritz Obermeyer. Pathwise derivatives beyond the reparameterization trick. In *International Conference on Machine Learning*, pages 2235–2244. PMLR, 2018.

[28] Samuel Karlin and James McGregor. The classification of birth and death processes. *Transactions of the American Mathematical Society*, 86(2):366–400, 1957.

[29] Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

[30] W. Kool, H. V. Hoof, and M. Welling. Buy 4 REINFORCE samples, get a baseline for free! In *DeepRLStructPred@ICLR*, 2019.

[31] Wouter Kool, Herke van Hoof, and Max Welling. Estimating gradients for discrete random variables by sampling without replacement. In *International Conference on Learning Representations*, 2020.

[32] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[33] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[34] Hao Liu, Yihao Feng, Yi Mao, Dengyong Zhou, Jian Peng, and Qiang Liu. Action-depedent control variates for policy optimization via Stein's identity. *arXiv preprint arXiv:1710.11198*, 2017.

[35] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017.

[36] Antonietta Mira, Reza Solgi, and Daniele Imparato. Zero variance Markov chain Monte Carlo for Bayesian estimators. *Statistics and Computing*, 23(5):653–662, 2013.

[37] Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. In *International Conference on Machine Learning*, pages 1791–1799, 2014.

[38] Andriy Mnih and Danilo Rezende. Variational inference for Monte Carlo objectives. In *International Conference on Machine Learning*, pages 2188–2196, 2016.

[39] Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte Carlo gradient estimation in machine learning. *Journal of Machine Learning Research*, 21(132):1–62, 2020.

[40] Chris J Oates, Mark Girolami, and Nicolas Chopin. Control functionals for Monte Carlo integration. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(3):695–718, 2017.

[41] Art B. Owen. *Monte Carlo theory, methods and examples*. 2013.

[42] John Paisley, David M Blei, and Michael I Jordan. Variational Bayesian inference with stochastic search. In *International Conference on Machine Learning*, pages 1363–1370, 2012.

[43] Paavo Parmas and Masashi Sugiyama. A unified view of likelihood ratio and reparameterization gradients. In *International Conference on Artificial Intelligence and Statistics*, pages 4078–4086. PMLR, 2021.

[44] Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *International Conference on Artificial Intelligence and Statistics*, page 814–822, 2014.

[45] Gesine Reinert and Nathan Ross. Approximating stationary distributions of fast mixing Glauber dynamics, with applications to exponential random graphs. *The Annals of Applied Probability*, 29(5):3201–3229, 2019.

[46] Danilo J. Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, 2014.

[47] Lorenz Richter, Ayman Boustati, Nikolas Nüsken, Francisco Ruiz, and Omer Deniz Akyildiz. VarGrad: A low-variance gradient estimator for variational inference. In *Advances in Neural Information Processing Systems*, volume 33, pages 13481–13492, 2020.

[48] Tim Salimans and David A Knowles. On using control variates with stochastic approximation for variational Bayes and its connection to stochastic linear regression. *arXiv preprint arXiv:1401.1022*, 2014.

[49] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[50] Shijing Si, Chris Oates, Andrew B Duncan, Lawrence Carin, and François-Xavier Briol. Scalable control variates for Monte Carlo methods via stochastic optimization. *arXiv preprint arXiv:2006.07487*, 2020.

[51] Jascha Sohl-Dickstein, Peter Battaglino, and Michael R DeWeese. Minimum probability flow learning. In *International Conference on Machine Learning*, pages 905–912, 2011.

[52] Leah F South, Chris J Oates, Antonietta Mira, and Christopher Drovandi. Regularised zero-variance control variates for high-dimensional variance reduction. *arXiv preprint arXiv:1811.05073*, 2018.

[53] Charles Stein. A bound for the error in the normal approximation to the distribution of a sum of dependent random variables. In *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 2: Probability Theory*. The Regents of the University of California, 1972.

[54] Charles Stein, Persi Diaconis, Susan Holmes, and Gesine Reinert. Use of exchangeable pairs in the analysis of simulations. *Lecture Notes-Monograph Series*, pages 1–26, 2004.

[55] David Stirzaker. *Stochastic processes and models*. OUP Oxford, 2005.

[56] Michalis K. Titsias and Miguel Lázaro-Gredilla. Doubly stochastic variational Bayes for non-conjugate inference. In *International Conference on Machine Learning*, 2014.

[57] Michalis K Titsias and Miguel Lázaro-Gredilla. Local expectation gradients for black box variational inference. *Advances in neural information processing systems*, 28:2638–2646, 2015.

[58] Michalis K Titsias and Jiaxin Shi. Double control variates for gradient estimation in discrete latent variable models. *International Conference on Artificial Intelligence and Statistics*, 2022.

[59] Seiya Tokui and Issei Sato. Evaluating the variance of likelihood-ratio gradient estimators. In *International Conference on Machine Learning*, pages 3414–3423, 2017.

[60] G. Tucker, A. Mnih, C. J. Maddison, and J. Sohl-Dickstein. REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models. In *Advances in Neural Information Processing Systems*, 2017.

[61] Arash Vahdat and Jan Kautz. NVAE: A deep hierarchical variational autoencoder. *Advances in Neural Information Processing Systems*, 33:19667–19679, 2020.

[62] Lex Weaver and Nigel Tao. The optimal reward baseline for gradient-based reinforcement learning. In *Conference on Uncertainty in Artificial Intelligence*, pages 538–545, 2001.

[63] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[64] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[65] Jiasen Yang, Qiang Liu, Vinayak Rao, and Jennifer Neville. Goodness-of-fit testing for discrete distributions via Stein discrepancy. In *International Conference on Machine Learning*, pages 5561–5570. PMLR, 2018.

[66] Mingzhang Yin and Mingyuan Zhou. ARM: Augment-REINFORCE-merge gradient for stochastic binary networks. In *International Conference on Learning Representations*, 2019.

[67] Mingzhang Yin, Nhat Ho, Bowei Yan, Xiaoning Qian, and Mingyuan Zhou. Probabilistic best subset selection via gradient-based optimization, 2020.

[68] Giacomo Zanella. Informed proposals for local MCMC in discrete spaces. *Journal of the American Statistical Association*, 115(530):852–865, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 7.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] MIT-licensed open-source Python code replicating all experiments can be found in the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix D.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See, e.g., Table 2.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See, e.g., Table 5.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] MIT-licensed open-source Python code replicating all experiments can be found in the supplementary material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Sample-dependent Baselines in REBAR and RELAX

We start with the REINFORCE estimator with the sample-dependent baseline b_k:

$$\frac{1}{K}\sum_{k=1}^{K}(f(x^{(k)})-b_k)\nabla_\eta \log q_\eta(x^{(k)})+\mathbb{E}[b_k\nabla_\eta \log q_\eta(x^{(k)})]. \tag{19}$$

REBAR [60] introduces indirect independence on $x^{(k)}$ in b_k through the continuous reparameterization $x=H(z), z^{(k)}\sim q_\eta(z|x=x^{(k)})$, where z is a continuous variable and H is an argmax-like thresholding function. Specifically, $b_k=f(\sigma_\lambda(z^{(k)}))$, where σ_λ is a continuous relaxation of H controlled by the parameter λ. The correction term decomposes into two parts:

$$\begin{aligned}&\mathbb{E}_{x^{(k)}}[\mathbb{E}_{z^{(k)}|x^{(k)}}[f(\sigma_\lambda(z^{(k)}))]\nabla_\eta \log q_\eta(x^{(k)})]\\ =&\nabla_\eta\mathbb{E}_{q_\eta(z)}[f(\sigma_\lambda(z))]-\mathbb{E}_{x^{(k)}}[\nabla_\eta\mathbb{E}_{q_\eta(z^{(k)}|x^{(k)})}[f(\sigma_\lambda(z^{(k)}))]].\end{aligned}$$

Both parts can be estimated with the reparameterization trick [29, 46, 56] which often has low variance. The RELAX [20] estimator generalizes REBAR by noticing that $f(\sigma_\lambda(z))$ can be replaced with a free-form differentiable function $c_\phi(z)$. However, RELAX still relies on parameterizing $c_\phi(z)$ as $f(\sigma_\lambda(z))+r_\theta(z)$ to achieve strong performance, as noted in Dong et al. [14].

To form modified RELAX in Section 6.3, we replace $b_k=c_\phi(z^{(k)})$ with $b_k=h_k(\sigma_\lambda(z^{(k)}))$ for h_k defined in (16).

B Proof of Unbiasedness of RODEO

Recall our estimator defined in (17) is

$$\frac{1}{K}\sum_{k=1}^{K}[(f(x^{(k)})-\frac{1}{K-1}\sum_{j\neq k}(f(x^{(j)})+(Ah_j)(x^{(j)})))\cdot\nabla_\eta \log q_\eta(x^{(k)})+(A\tilde{h}_k^*)(x^{(k)})]. \tag{20}$$

To show the unbiasedness, we compute its expectation under q_η as

$$\begin{aligned}&\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}_{q_\eta}[f(x^{(k)})\nabla_\eta \log q_\eta(x^{(k)})]\\ -&\frac{1}{K(K-1)}\sum_{k=1}^{K}\sum_{j\neq k}\mathbb{E}_{q_\eta}[(f(x^{(j)})+(Ah_j)(x^{(j)}))\nabla_\eta \log q_\eta(x^{(k)})]\\ +&\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}_{q_\eta}[(A\tilde{h}_k^*)(x^{(k)})].\end{aligned}$$

Since the first term is the desired gradient $\nabla_\eta\mathbb{E}_{q_\eta}[f(x)]$ and the third term is zero, it suffices to show that the second term also vanishes. Using the law of total expectations, we find for $j\neq k$,

$$\begin{aligned}&\mathbb{E}_{q_\eta}[(f(x^{(j)})+(Ah_j)(x^{(j)}))\nabla_\eta \log q_\eta(x^{(k)})]\\ &=\mathbb{E}_{x^{(k)}\sim q_\eta}[\mathbb{E}_{q_\eta}[f(x^{(j)})+(Ah_j)(x^{(j)})\mid x^{(k)}]\nabla_\eta \log q_\eta(x^{(k)})]\\ &=\mathbb{E}_{x^{(k)}\sim q_\eta}[\mathbb{E}_{q_\eta}[f(x^{(j)})\mid x^{(k)}]\nabla_\eta \log q_\eta(x^{(k)})]\\ &=\mathbb{E}_{q_\eta}[f(x^{(j)})\nabla_\eta \log q_\eta(x^{(k)})]\\ &=\mathbb{E}_{x^{(j)}\sim q_\eta}[f(x^{(j)})\mathbb{E}_{x^{(k)}\sim q_\eta}[\nabla_\eta \log q_\eta(x^{(k)})\mid x^{(j)}]]=0,\end{aligned}$$

which completes the proof.

C Additional Experiments

C.1 Wall clock time comparison with RLOO

Besides necessary target function evaluations, the RODEO estimator comes with the additional cost of evaluating the neural network-based H, H^*. Therefore, RODEO is most suited to the problems

Table 3: Architecture of the ResNet VAE in Appendix C.1. 3x3xC means kernel size 3x3 and C output channels. Each (De)conv Res block is composed of two (de)convolutional layers with strides 1, same padding, and ReLU activations, plus a skip connection with identity map. For Res blocks with downsample and upsample functions, the first convolutional layer has strides 2, and the skip connection is replaced by a convolutional layer with 2x2 kernel size and strides 2.

Encoder	Decoder
Conv 3x3x16, strides 1, padding 1	Fully connected, 7x7x64 units
Conv Res block 3x3x16	Deconv Res block 3x3x64
Conv Res block 3x3x16	Deconv Res block 3x3x64
Conv Res block 3x3x32 (downsample by 2)	Deconv Res block 3x3x32 (upsample by 2)
Conv Res block 3x3x32	Deconv Res block 3x3x32
Conv Res block 3x3x64 (downsample by 2)	Deconv Res block 3x3x16 (upsample by 2)
Conv Res block 3x3x64	Deconv Res block 3x3x16
Fully connected, 200 units	Deconv 3x3x1, strides 1, padding 1

where evaluating f is very expensive and its cost dominates those of the H, H^*. This is often the case in practice. For example, state-of-the-art variational autoencoders [e.g., 61] are often built on expensive neural architectures such as deep residual networks (ResNet). Here, to demonstrate the practical advantage of our method as the complexity of f grows, we replace the two-layer MLP VAEs used in previous experiments with a ResNet VAE (architecture shown in Table 3), while the neural network used by H, H^* remains a single-layer MLP with 100 hidden units. We then compare the wall clock performance of RODEO with RLOO. The latent variables in this experiment remain binary and have 200 dimensions.

The results are shown in Figure 5. RODEO achieves better training ELBOs than RLOO in the same amount of time. In fact, for this VAE architecture, the per-iteration time of RODEO is **25.2ms**, which is very close to the **23.1ms** of RLOO. This indicates that the cost of f is significant higher than H, H^*.

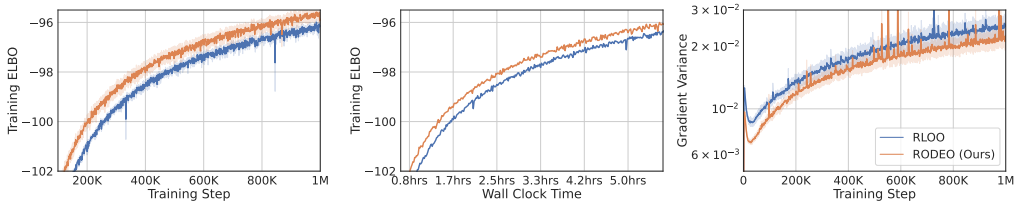


Figure 5: Comparing the performance of RODEO and RLOO on more expensive ResNet VAE models trained on binarized MNIST with $K = 2$. The middle plot shows the average wall clock performance over 5 trials.

C.2 Impact of neural network architectures of surrogate functions

We conduct one more ablation study to investigate the impact of neural network architectures used by H, H^*. Specifically, we replace the single-hidden-layer control variate network used in previous experiments with a two-hidden-layer MLP (each layer has 100 units) and compare their performance on binarized MNIST with $K = 2$. We keep other settings the same as in Section 6.1. The results are plotted in Figure 6. We do not observe significant difference between the two versions of RODEO.

D Experimental Details

Our implementation is based on the open-source code of DisARM [14] (Apache license) and Double CV [58] (MIT license). Our figures display 1M training steps and our tables report performance after 1M training steps to replicate the experimental settings of DisARM [14].

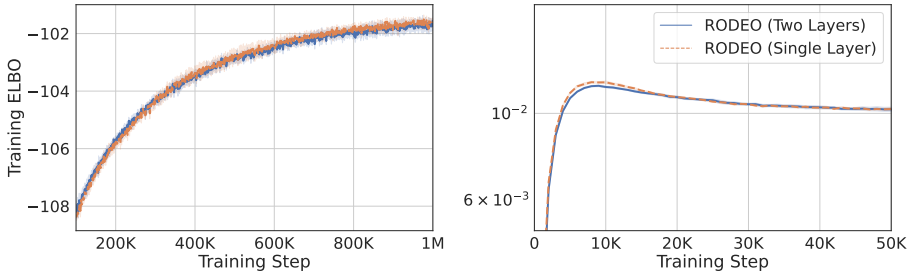


Figure 6: Comparing the performance of RODEO with single-hidden-layer and two-hidden-layer neural network architectures for H, H^* on binary VAEs.

D.1 Details of VAE experiments

VAEs are models with a joint density $p(y, x) = p(y|x)p(x)$, where x denotes the latent variable. x is assigned a uniform factorized Bernoulli prior. The likelihood $p_\theta(y|x)$ is parameterized by the output of a neural network with x as input and parameters θ. The VAE has two hidden layers with 200 units activated by LeakyReLU with the coefficient 0.3. To optimize the VAE we use Adam with base learning rate 10^{-4} for non-binarized data and 10^{-3} for dynamically binarized data, except for binarized Fashion-MNIST we decreased the learning rate to 3×10^{-4} because otherwise the training is very unstable for all estimators. We use Adam with the same learning rate 10^{-3} for adapting our control variate network in all experiments. The batch size is 100. The settings of other estimators are kept the same with Titsias and Shi [58].

In the minimum probability flow (MPF) and Barker Stein operator (9), we choose the neighborhood $\mathcal{N}_x$ to be the states that differ in only one coordinate from x. Let $y \in \mathcal{N}_x$ be an element in this neighborhood such that $y_i \neq x_i$ and $y_{-i} = x_{-i}$. For the MPF Stein estimator and the difference Stein estimator (11), the density ratio $\frac{q(y)}{q(x)}$ can be simplified to $\frac{q(y_i|x_{-i})}{q(x_i|x_{-i})}$. We further replace it with $\frac{q(y_i|x_{-i})}{q(x_i|x_{-i})+10^{-3}}$ to alleviate numerical instability. The Barker Stein estimator does not suffer from the numerical issue since the coefficient is bounded in the Bernoulli case: $\frac{q(y)}{q(x)+q(y)} = \frac{q(y_i|x_{-i})}{q(x_i|x_{-i})+q(y_i|x_{-i})} = q(y_i|x_{-i})$. In our experiments, we find that the difference Stein estimator is highly unstable and may diverge as the iteration proceeds.

D.2 Details of hierarchical VAE experiments

We optimize the hierarchical VAE using Adam with base learning rate 10^{-4}. Our control variate network is optimized using Adam with learning rate 10^{-3}. Settings of training multilayer VAEs are kept the same with Dong et al. [14], except that we do not optimize the prior distribution of the VAE hidden layer and use a larger batch size 100.

E Additional Results

In this section, we measure test set performance using 100 test points and the marginal log-likelihood bound of Burda et al. [10], which provides a tighter estimate of marginal log likelihood than the ELBO. Throughout, we call this the "test log-likelihood bound."

Table 4: Average 100-point test log-likelihood bounds of binary latent VAEs trained with $K = 2, 3$ (except for RELAX which uses 3 evaluations per step) on MNIST, Fashion-MNIST, and Omniglot. We report the average value ± 1 standard error after 1M steps over 5 independent runs.

		Bernoulli Likelihoods			Gaussian Likelihoods		
		MNIST	Fashion-MNIST	Omniglot	MNIST	Fashion-MNIST	Omniglot
$K = 2$	DisARM	-101.61 ± 0.07	-239.11 ± 0.11	-118.34 ± 0.05	669.26 ± 0.53	163.40 ± 0.59	305.32 ± 0.89
	Double CV	-100.91 ± 0.04	-239.00 ± 0.17	-118.45 ± 0.09	677.02 ± 0.93	164.99 ± 0.71	304.72 ± 1.39
	RODEO (Ours)	$\mathbf{-100.78 \pm 0.16}$	$\mathbf{-238.97 \pm 0.09}$	$\mathbf{-118.09 \pm 0.05}$	$\mathbf{681.11 \pm 0.31}$	$\mathbf{168.26 \pm 0.73}$	$\mathbf{308.55 \pm 1.02}$
$K = 3$	ARMS	-99.08 ± 0.12	-238.19 ± 0.11	-116.78 ± 0.13	688.61 ± 0.84	174.14 ± 0.44	320.45 ± 1.07
	Double CV	-99.16 ± 0.12	-238.54 ± 0.16	-116.75 ± 0.15	690.28 ± 0.49	173.67 ± 0.30	322.88 ± 1.10
	RODEO (Ours)	$\mathbf{-98.72 \pm 0.14}$	$\mathbf{-237.97 \pm 0.12}$	$\mathbf{-116.69 \pm 0.09}$	$\mathbf{695.11 \pm 0.33}$	$\mathbf{174.57 \pm 0.30}$	$\mathbf{323.92 \pm 1.24}$
RELAX (3 evals)		-100.80 ± 0.09	-239.03 ± 0.11	-117.60 ± 0.06	686.21 ± 0.57	171.43 ± 0.61	317.78 ± 1.25

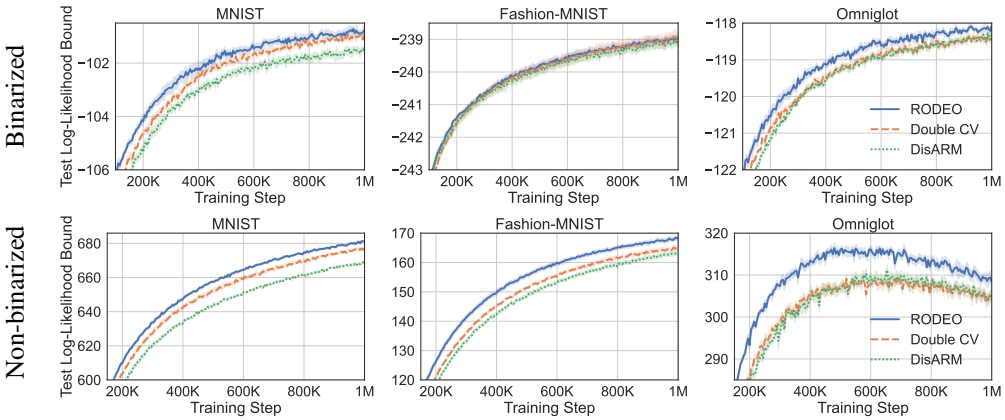


Figure 7: Average 100-point test log-likelihood bounds for binary latent VAEs trained on (top) dynamically binarized and (bottom) non-binarized MNIST, Fashion-MNIST, and Omniglot using $K = 2$.

Table 5: Average running time across 10^4 steps on an NVIDIA 3080Ti GPU with an AMD 5950X CPU for the VAE experiment on binary MNIST in Section 6.1.

	Double CV	DisARM/ARMS	RODEO (Ours)	RELAX (3 evals)
$K = 2$	2.11 ms/step	1.89 ms/step	3.08 ms/step	4.71 ms/step
$K = 3$	2.28 ms/step	1.91 ms/step	4.72 ms/step	

Table 6: Average running time across 1K steps on an NVIDIA A40 GPU with an Intel 4214X CPU for the ResNet VAE experiment in Appendix C.1.

Method	RODEO (Ours)	RLOO
Time	25.2 ms/step	23.1 ms/step

Table 7: Average running time across 10^4 steps on an NVIDIA 3080Ti GPU with an AMD 5950X CPU when training hierarchical VAEs with $K = 2$.

	Double CV	DisARM	RODEO (Ours)
Two layers	4.33 ms/step	3.54 ms/step	6.79 ms/step
Three layers	7.69 ms/step	6.09 ms/step	10.61 ms/step
Four layers	11.67 ms/step	9.53 ms/step	14.91 ms/step

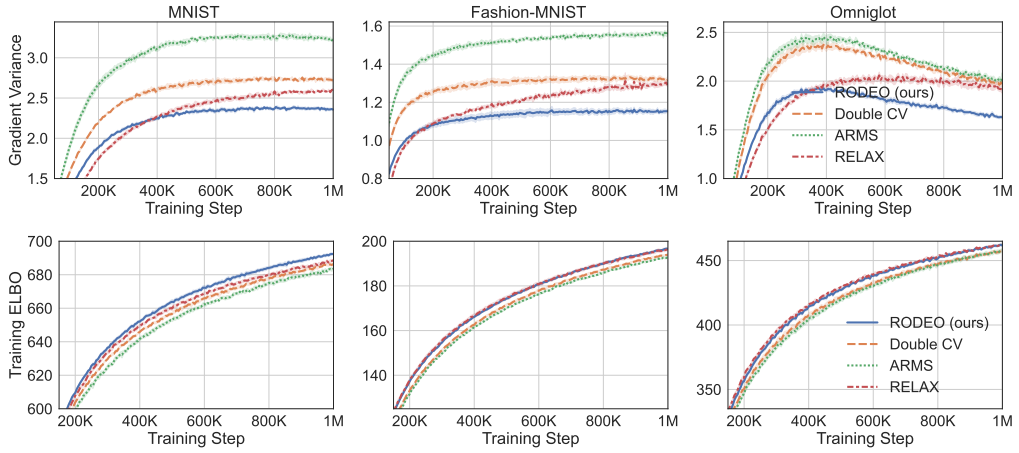


Figure 8: Training binary latent VAEs with Gaussian likelihoods with three evaluations of f per step using RODEO/Double CV/ARMS with $K = 3$ or RELAX on non-binarized MNIST, Fashion-MNIST, and Omniglot. (Top) variance of gradient estimates. (Bottom) the plot of average ELBO on training examples against training steps.

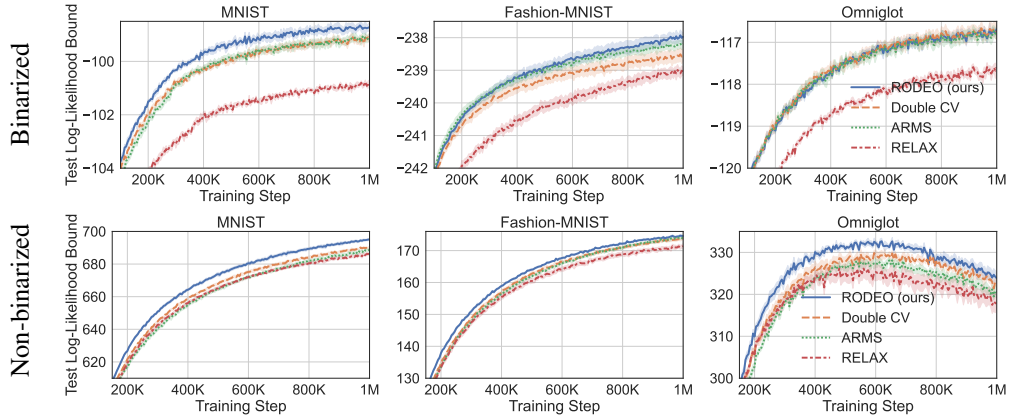


Figure 9: Average 100-point test log-likelihood bounds for binary latent VAEs trained on (top) dynamically binarized and (bottom) non-binarized MNIST, Fashion-MNIST, and Omniglot with three evaluations of f per step using RODEO/Double CV/ARMS with $K = 3$ or RELAX.

Table 8: Training hierarchical binary latent VAEs on dynamically binarized MNIST, Fashion-MNIST, and Omniglot. We report the average (± 1 standard error) training ELBOs and 100-point test log-likelihood bounds after 1M steps over 5 independent runs.

		Training ELBO			Test Log-Likelihood Bound		
		MNIST	Fashion-MNIST	Omniglot	MNIST	Fashion-MNIST	Omniglot
Two layers	Double CV	-103.52 ± 0.06	-239.82 ± 0.07	-114.06 ± 0.04	-97.62 ± 0.08	-237.65 ± 0.06	-110.48 ± 0.04
	DisARM	-103.39 ± 0.12	$\mathbf{-239.67 \pm 0.06}$	$\mathbf{-113.67 \pm 0.05}$	-97.56 ± 0.07	$\mathbf{-237.61 \pm 0.06}$	$\mathbf{-110.15 \pm 0.04}$
	RODEO (Ours)	$\mathbf{-103.15 \pm 0.07}$	-239.76 ± 0.09	-113.84 ± 0.11	$\mathbf{-97.43 \pm 0.03}$	-237.63 ± 0.07	-110.32 ± 0.10
Three layers	Double CV	-97.59 ± 0.15	-234.34 ± 0.07	-108.66 ± 0.06	-93.71 ± 0.12	-234.34 ± 0.07	-107.48 ± 0.07
	DisARM	-97.95 ± 0.30	-234.45 ± 0.05	-108.60 ± 0.08	-94.12 ± 0.28	-234.46 ± 0.06	-107.32 ± 0.10
	RODEO (Ours)	$\mathbf{-97.21 \pm 0.17}$	$\mathbf{-234.11 \pm 0.10}$	$\mathbf{-108.51 \pm 0.04}$	$\mathbf{-93.52 \pm 0.16}$	$\mathbf{-234.19 \pm 0.07}$	$\mathbf{-107.26 \pm 0.06}$
Four layers	Double CV	-98.73 ± 0.06	-235.69 ± 0.07	-110.92 ± 0.06	-93.28 ± 0.03	-234.63 ± 0.03	-107.86 ± 0.03
	DisARM	-98.97 ± 0.02	-235.50 ± 0.04	-110.85 ± 0.07	-93.56 ± 0.04	-234.52 ± 0.04	-107.87 ± 0.05
	RODEO (Ours)	$\mathbf{-98.67 \pm 0.14}$	$\mathbf{-235.39 \pm 0.05}$	$\mathbf{-110.79 \pm 0.03}$	$\mathbf{-93.27 \pm 0.09}$	$\mathbf{-234.39 \pm 0.06}$	$\mathbf{-107.77 \pm 0.02}$

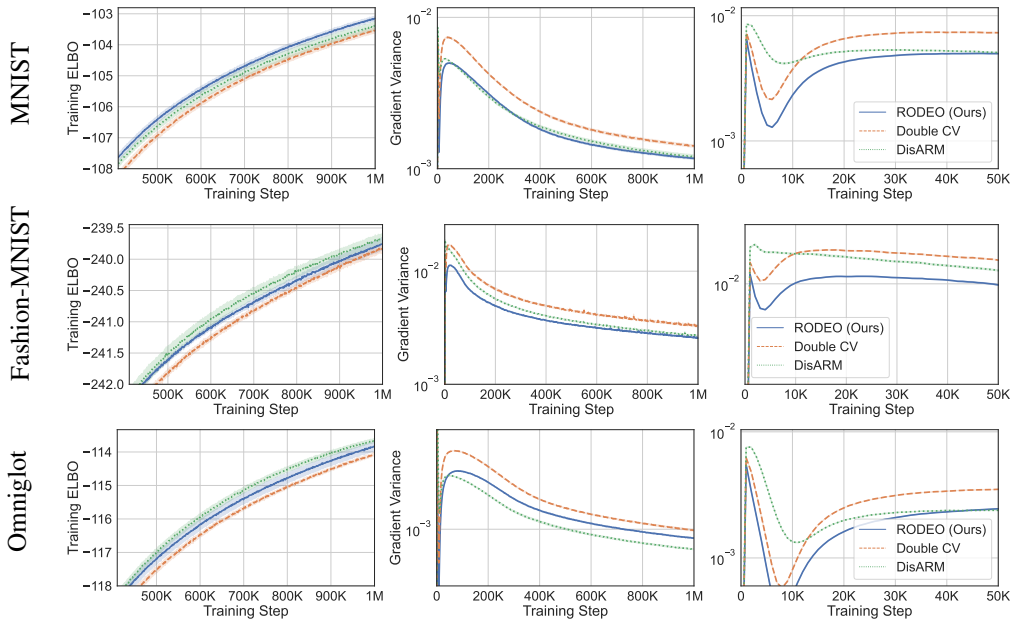


Figure 10: Training hierarchical binary latent VAEs with **two** stochastic layers on dynamically binarized MNIST, Fashion-MNIST and Omniglot. We plot (left) the average ELBO on training examples and (middle) variance of gradient estimates. We zoom into the first 50K steps of the variance plot on the right figure.

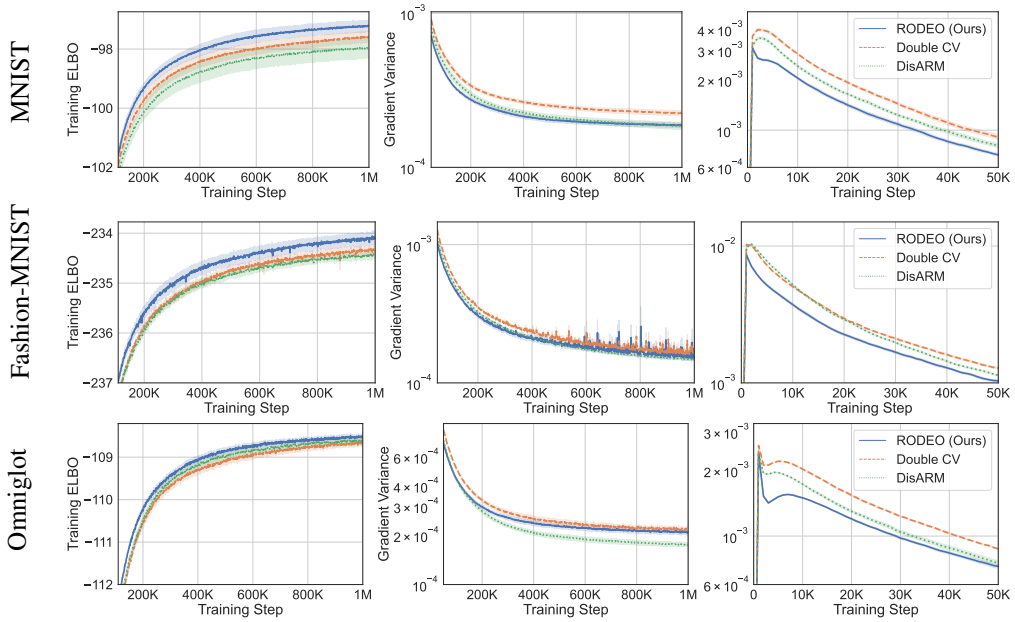


Figure 11: Training hierarchical binary latent VAEs with **three** stochastic layers. on dynamically binarized MNIST, Fashion-MNIST and Omniglot. We plot the average ELBO on training examples (left) and variance of gradient estimates (middle). We zoom into the first 50K steps of the variance plot on the right figure.

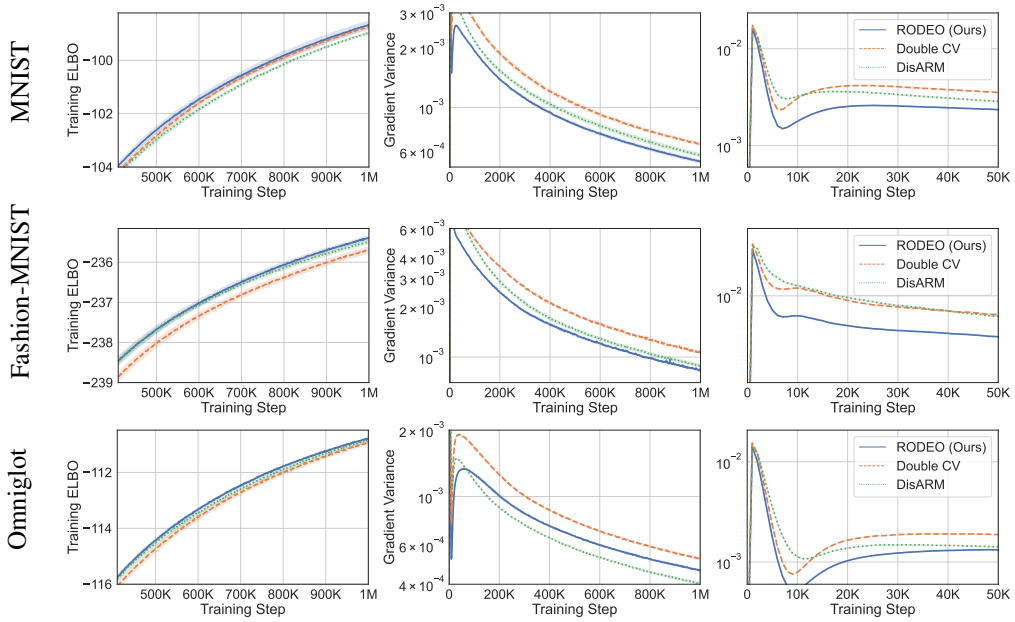


Figure 12: Training hierarchical binary latent VAEs with **four** stochastic layers on dynamically binarized MNIST, Fashion-MNIST and Omniglot. We plot the average ELBO on training examples (left) and variance of gradient estimates (middle). We zoom into the first 50K steps of the variance plot on the right figure.