# SUMO: UNBIASED ESTIMATION OF LOG MARGINAL PROBABILITY FOR LATENT VARIABLE MODELS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

The standard variational lower bounds used to train latent variable models produce biased estimates of most quantities of interest. We introduce an unbiased estimator of the log marginal likelihood and its gradients for latent variable models based on randomized truncation of infinite series. If parameterized by an encoder-decoder architecture, the parameters of the encoder can be optimized to minimize its variance of this estimator. We show that models trained using our estimator give better test-set likelihoods than a standard importance-sampling based approach for the same average computational cost. This estimator also allows use of latent variable models for tasks where unbiased estimators, rather than marginal likelihood lower bounds, are preferred, such as minimizing reverse KL divergences and estimating score functions.

## 1 INTRODUCTION

Latent variable models are powerful tools for constructing highly expressive data distributions and for understanding how high-dimensional observations might possess a simpler representation. Latent variable models are often framed as probabilistic graphical models, allowing these relationships to be expressed in terms of conditional independence. Mixture models, probabilistic principal component analysis (Tipping & Bishop, 1999), hidden Markov models, and latent Dirichlet allocation (Blei et al., 2003) are all examples of powerful latent variable models. More recently there has been a surge of interest in probabilistic latent variable models that incorporate flexible nonlinear likelihoods via deep neural networks (Kingma & Welling, 2013). These models can blend the advantages of highly structured probabilistic priors with the empirical successes of deep learning (Johnson et al., 2016). Moreover, these explicit latent variable models can often yield relatively interpretable representations, in which simple interpolation in the latent space can lead to semantically-meaningful changes in high-dimensional observations (e.g., Higgins et al. (2017)).

It can be challenging, however, to fit the parameters of a flexible latent variable model, since computing the marginal probability of the data requires integrating out the latent variables in order to maximize the likelihood with respect to the model parameters. Typical approaches to this problem include the celebrated expectation maximization algorithm (Dempster et al., 1977), Markov chain Monte Carlo, and the Laplace approximation. Variational inference generalizes expectation maximization by forming a lower bound on the aforementioned (log) marginal likelihood, using a tractable approximation to the unmanageable posterior over latent variables. The maximization of this lower bound—rather than the true log marginal likelihood—is often relatively straightforward when using automatic differentiation and Monte Carlo sampling. However, a lower bound may be ill-suited for tasks such as posterior inference and other situations where there exists an entropy maximization objective; for example in entropy-regularized reinforcement learning (Williams & Peng, 1991; Mnih et al., 2016; Norouzi et al., 2016) which requires minimizing the log probability of the samples under the model.

While there is a long history in Bayesian statistics of estimating the marginal likelihood (e.g., Newton & Raftery (1994); Neal (2001)), we often want high-quality estimates of the *logarithm* of the marginal likelihood, which is better behaved when the data are high dimensional; it is not as susceptible to underflow and it has gradients that are numerically sensible. However, the log transformation introduces some challenges: Monte Carlo estimation techniques such as importance sampling do not straightforwardly give unbiased estimates of this quantity. Nevertheless, there has been significant

work to construct estimators of the log marginal likelihood in which it is possible to explicitly trade off between bias against computational cost (Burda et al., 2016; Bamler et al., 2017; Nowozin, 2018). Unfortunately, while there are asymptotic regimes where the bias of these estimators approaches zero, it is always possible to optimize the parameters to increase this bias to infinity.

In this work, we construct an unbiased estimator of the log marginal likelihood. Although there is no theoretical guarantee that this estimator has finite variance, we find that it can work well in practice. We show that this unbiased estimator can train latent variable models to achieve higher test log-likelihood than lower bound estimators at the same expected compute cost. More importantly, this unbiased estimator allows us to apply latent variable models in situations where these models were previously problematic to optimize with lower bound estimators. Such applications include latent variable modeling for posterior inference and for reinforcement learning in high-dimensional action spaces, where an ideal model is one that is highly expressive yet efficient to sample from.

## 2 PRELIMINARIES

### 2.1 LATENT VARIABLE MODELS

Latent variable models (LVMs) describe a distribution over data in terms of a mixture over unobserved quantities. Let $p_\theta(x)$ be a family of probability density (mass) functions on a data space $\mathcal{X}$, indexed by parameters $\theta$. We will generally refer to this as a "density" for consistency, even when the data should be understood to be discrete; similarly we will use integrals even when the marginalization is over a discrete set. In a latent variable model, $p_\theta(x)$ is defined via a space of latent variables $\mathcal{Z}$, a family of mixing measures on this latent space with density denoted $p_\theta(z)$, and a conditional distribution $p_\theta(x \mid z)$. This conditional distribution is sometimes called an "observation model" or a conditional likelihood. We will take $\theta$ to parameterize both $p_\theta(x \mid z)$ and $p_\theta(z)$ in the service of determining the marginal $p_\theta(x)$ via the mixture integral:

$$p_\theta(x) := \int_{\mathcal{Z}} p_\theta(x \mid z) p_\theta(z)\, dz = \mathbb{E}_{z \sim p_\theta(z)} \left[ p_\theta(x \mid z) \right] . \tag{1}$$

This simple formalism allows for a large range of modeling approaches, in which complexity can be baked into the latent variables (as in traditional graphical models), into the conditional likelihood (as in variational autoencoders), or into both (as in structured VAEs). The downside of this mixing approach is that the integral may be intractable to compute, making it difficult to evaluate $p_\theta(x)$—a quantity often referred to in Bayesian statistics and machine learning as the *marginal likelihood* or *evidence*. Various Monte Carlo techniques have been developed to provide consistent and often unbiased estimators of $p_\theta(x)$, but it is usually preferable to work with $\log p_\theta(x)$ and unbiased estimation of this quantity has, to our knowledge, not been previously studied.

### 2.2 TRAINING LATENT VARIABLE MODELS

Fitting a parametric distribution to observed data is often framed as the minimization of a difference between the model distribution and the empirical distribution. The most common difference measure is the forward Kullback-Leibler (KL) divergence; if $p_{\mathsf{data}}(x)$ is the empirical distribution and $p_\theta(x)$ is a parametric family, then minimizing the KL divergence with respect to $\theta$ is equivalent to maximizing the likelihood:

$$D_{\mathrm{KL}}(p_{\mathsf{data}} \,||\, p_\theta) = \int_{\mathcal{X}} p_{\mathsf{data}}(x)\, \log \frac{p_{\mathsf{data}}(x)}{p_\theta(x)}\, dx = -\mathbb{E}_{\mathsf{data}} \left[ \log p_\theta(x) \right] + \text{const} . \tag{2}$$

Thus the optimization problem of finding the MLE parameters $\theta$ comes down to maximizing the expected log probability of the data:

$$\theta^{\mathsf{MLE}} = \arg\min_\theta D_{KL}(p_{\mathsf{data}} \,||\, p_\theta) = \arg\max_\theta \mathbb{E}_{\mathsf{data}} \left[ \log p_\theta(x) \right] . \tag{3}$$

Since expectations can be estimated in an unbiased manner using Monte Carlo procedures, simple subsampling of the data enables powerful stochastic optimization techniques, with stochastic gradient descent in particular forming the basis for learning the parameters of many nonlinear models.

However, this requires unbiased estimates of $\nabla_\theta \log p_\theta(x)$, which are not available for latent variable models. Instead, a stochastic lower bound of $\log p_\theta(x)$ is often used and then differentiated for optimization.

Though many lower bound estimators are applicable, we focus on an importance-weighted evidence lower bound (Burda et al., 2016). This lower bound is constructed by introducing a proposal distribution $q(x; z)$ and using it to form an importance sampling estimate of the marginal likelihood:

$$p_\theta(x) = \int_{\mathcal{Z}} p_\theta(x \mid z)\, p_\theta(z)\, dz = \int_{\mathcal{Z}} q(z; x) \frac{p_\theta(x \mid z)\, p_\theta(z)}{q(z; x)}\, dz = \mathbb{E}_{z\sim q}\left[\frac{p_\theta(x \mid z)\, p_\theta(z)}{q(z; x)}\right]. \quad (4)$$

If $K$ samples are drawn from $q(z; x)$ then this provides an unbiased estimate of $p_\theta(x)$ and the biased "importance-weighted autoencoder" estimator $\text{IWAE}_K(x)$ of $\log p_\theta(x)$ is given by

$$\text{IWAE}_K(x) := \log \frac{1}{K} \sum_{k=1}^{K} \frac{p_\theta(x \mid z_k)\, p_\theta(z_k)}{q(z_k; x)}, \quad z_k \overset{iid}{\sim} q(z; x)\,. \quad (5)$$

The special case of $K = 1$ generates an unbiased estimate of the evidence lower bound (ELBO), which is often used for performing variational inference by stochastic gradient descent. While the IWAE lower bound acts as a useful replacement of $\log p_\theta(x)$ in maximum likelihood training, it may not be suitable for other objectives such as those that involve entropy maximization. We discuss tasks for which a lower bound estimator would be ill-suited in Section 3.4.

There are two properties of IWAE that will allow us to modify it to produce an unbiased estimator: First, it is consistent in the sense that as the number of samples $K$ increases, the expectation of $\text{IWAE}_K(x)$ converges to $\log p_\theta(x)$. Second, is also monotonically non-decreasing in expectation:

$$\log p_\theta(x) = \lim_{K\to\infty} \mathbb{E}[\text{IWAE}_K(x)] \qquad \text{and} \qquad \mathbb{E}[\text{IWAE}_{K+1}(x)] \geq \mathbb{E}[\text{IWAE}_K(x)]\,. \quad (6)$$

These properties are sufficient to create an unbiased estimator using the Russian roulette estimator.

### 2.3 Russian roulette estimator

In order to create an unbiased estimator of the log probability function, we employ the Russian roulette estimator (Kahn, 1955). This estimator is used to estimate the sum of infinite series, where evaluating any term in the series almost surely requires only a finite amount of computation. Intuitively, the Russian roulette estimator relies on a randomized truncation and upweighting of each term to account for the possibility of not computing these terms.

To illustrate the idea, let $\Delta_k$ denote the $k$-th term of an infinite series. Assume the partial sum of the series $\sum_{k=1}^{\infty} \Delta_k$ converges to some quantity we wish to obtain. We can construct a simple estimator by always computing the first term then flipping a coin $b \sim \text{Bernoulli}(q)$ to determine whether we stop or continue evaluating the remaining terms. With probability $1 - q$, we compute the rest of the series. By reweighting the remaining future terms by $1/(1-q)$, we obtain an unbiased estimator:

$$\tilde{Y} = \Delta_1 + \left(\frac{\sum_{k=2}^{\infty} \Delta_k}{1-q}\right) \mathbb{1}_{b=0} + (0)\,\mathbb{1}_{b=1} \qquad \mathbb{E}[\tilde{Y}] = \Delta_1 + \frac{\sum_{k=2}^{\infty} \Delta_k}{1-q}(1-q) = \sum_{k=1}^{\infty} \Delta_k.$$

To obtain the "Russian roulette" (RR) estimator (Forsythe & Leibler, 1950), we repeatedly apply this trick to the remaining terms. In effect, we make the number of terms a random variable $\mathcal{K}$, taking values in $1, 2, \ldots$ to use in the summation (i.e., the number of successful coin flips) from some distribution with probability mass function $p(K) = \mathbb{P}(\mathcal{K} = K)$ with support over the positive integers. With $K$ drawn from $p(K)$, the estimator takes the form:

$$\hat{Y}(K) = \sum_{k=1}^{K} \frac{\Delta_k}{\mathbb{P}(\mathcal{K} \geq k)} \qquad \mathbb{E}_{K\sim p(K)}[\hat{Y}(K)] = \sum_{k=1}^{\infty} \Delta_k\,. \quad (7)$$

The equality on the right hand of equation 7 holds so long as (i) $\mathbb{P}(\mathcal{K} \geq k) > 0,\ \forall k > 0$, and (ii) the series converges absolutely, i.e., $\sum_{k=1}^{\infty} |\Delta_k| < \infty$ (Chen et al., 2019). This condition ensures that the average of multiple samples will converge to the value of the infinite series by the law of large numbers. However, the variance of this estimator depends on the choice of $p(K)$ and can potentially be very large or even infinite (McLeish, 2011; Rhee & Glynn, 2015; Beatson & Adams, 2019).

# 3 SUMO: UNBIASED ESTIMATION OF LOG PROBABILITY FOR LVMS

## 3.1 RUSSIAN ROULETTE TO TIGHTEN LOWER BOUNDS

We can turn any absolutely convergent series into a telescoping series and apply the Russian roulette randomization to form an unbiased stochastic estimator. We focus here on the IWAE bound described in Section 2.2. Let $\Delta_k(x) = \text{IWAE}_{k+1}(x) - \text{IWAE}_k(x)$. Then since IWAE converges absolutely, we apply equation 7 to construct our estimator, which we call SUMO (Stochastically Unbiased Marginalization Objective):

$$\text{SUMO}(x) = \text{IWAE}_1(x) + \sum_{k=1}^{K} \frac{\Delta_k(x)}{\mathbb{P}(\mathcal{K} \geq k)} \qquad \text{where} \quad K \sim p(\mathcal{K})\,. \tag{8}$$

The randomized truncation of the series using the Russian roulette estimator means that this is an unbiased estimator of the log marginal likelihood, regardless of the distribution $p(K)$:

$$\mathbb{E}\left[\text{SUMO}(x)\right] = \log p_\theta(x)\,. \tag{9}$$

Furthermore, we can prove that the gradients of SUMO also have finite expectation (Appendix A.3). Algorithm 1 shows how we compute a single sample of SUMO, which has the same expected cost as $\text{IWAE}_K$ where $K = \mathbb{E}[\mathcal{K} + 1]$.

## 3.2 GRADIENT VARIANCE AND THE CHOICE OF $p(K)$

To efficiently optimize a limit, one should choose an estimator to minimize the product of the second moment of the *gradient* estimates and the expected compute cost per evaluation. The choice of $p(K)$ effects both the variance and computation cost of our estimator. Denoting $\hat{G} := \nabla_\theta \hat{Y}$ and $\Delta_k^g := \nabla_\theta[\text{IWAE}_{k+1}(x) - \text{IWAE}_k(x)]$, the Russian roulette estimator is optimal across a broad family of unbiased randomized truncation estimators if and only if the $\Delta_k^g$ are statistically independent, in which case it has second moment $\mathbb{E}||\hat{G}||_2^2 = \sum_{k=1}^{\infty} \mathbb{E}||\Delta_k^g||_2^2 / \mathbb{P}(\mathcal{K} \geq k)$ (Beatson & Adams, 2019). For the following derivation of $p(K)$, we assume independence of $\Delta_k^g$ and restrict ourselves to the Russian roulette estimator. While the $\Delta_k^g$ are not in fact strictly independent with our sampling procedure (Algorithm 1), and other estimators within the family may perform better, we justify our choice by showing that $\mathbb{E}\Delta_i\Delta_j$ for $i \neq j$ converges to zero much faster than $\mathbb{E}\Delta_k^2$ (Appendices A.1 & A.2).

We show that $\mathbb{E}||\Delta_k^g||_2^2$ is $\mathcal{O}(1/k^2)$ (Appendix A.3). Therefore, the optimal compute-variance product (Rhee & Glynn, 2015; Beatson & Adams, 2019) is given by $\mathbb{P}(\mathcal{K} \geq k) \propto \sqrt{\mathbb{E}||\Delta_k^g||_2^2}$. In our case, this gives $\mathbb{P}(\mathcal{K} \geq k) = 1/k$, which results in an estimator with infinite expected computation and no finite bound on variance. In fact, any $p(K)$ which gives rise to provably finite variance requires a heavier tail than $\mathbb{P}(\mathcal{K} \geq k) = 1/k$ and will have infinite expected computation.

There are no theoretical guarantees for optimization with an infinite-variance gradient estimator, but we empirically find that convergence is stable. We plot $||\Delta_k||_2^2$ for the toy variational inference task used to assess signal to noise ratio in Tucker et al. (2018) and Rainforth et al. (2018), and find that they converge faster than $\frac{1}{k^2}$ in practice (Appendix A.4). While this indicates the variance is better than the theoretical bound, an estimator having infinite expected computation cost will always be an issue as it indicates significant probability of sampling arbitrarily large $K$. We therefore modify the

**Algorithm 1** Computing SUMO, an unbiased estimator of $\log p(x)$.

**Input:** $x$, $m \geq 1$, encoder $q(z;x)$, decoder $p(x,z)$, $p(K)$, `reverse_cdf`$(\cdot) = \mathbb{P}(\mathcal{K} \geq \cdot)$

1: Sample $K \sim p(\mathcal{K})$
2: Sample $\{z_k\}_{k=1}^{K+m} \overset{iid}{\sim} q(z;x)$
3: $\log w_k \leftarrow \log p(x, z_k) - \log q(z_k; x)$
4: `ks` $\leftarrow [1, \ldots, K+m]$
5: `cum_iwae` $\leftarrow$ `log_cumsum_exp`$(\log w_k) - \log($`ks`$[:k+1])$
6: `inv_weights` = $1/$`reverse_cdf(ks)`

**return** `cum_iwae`$[m\text{-}1]$ + sum(`inv_weights` * (`cum_iwae`$[m:]$ - `cum_iwae`$[m\text{-}1\text{:-}1]$))

tail of the sampling distribution such that the estimator has finite expected computation:

$$\mathbb{P}(\mathcal{K} \geq k) = \begin{cases} 1/k & \text{if } k < \alpha \\ 1/\alpha \cdot (1-0.1)^{k-\alpha} & \text{if } k \geq \alpha \end{cases} \tag{10}$$

We typically choose $\alpha = 80$, which gives an expected computation cost of approximately 5 terms.

#### 3.2.1 TRADING VARIANCE AND COMPUTE

One way to potentially improve the RR estimator is to construct it so that some minimum number of terms (denoted here as $m$) are always computed. This puts a lower bound on the computational cost, but can potentially lower variance, providing a design space for trading off estimator quality against computational cost. This corresponds to a choice of RR estimator in which $\mathbb{P}(\mathcal{K} = K) = 0$ for $K \leq m$. This computes the sum out to $m$ terms (effectively computing $\text{IWAE}_m$) and then estimates the remaining difference with Russian roulette:

$$\text{SUMO}(x) = \text{IWAE}_m(x) + \sum_{k=m}^{K} \frac{\Delta_k(x)}{\mathbb{P}(\mathcal{K} \geq k)}, \quad K \sim p(K) \tag{11}$$

In practice, instead of tuning parameters of $p(K)$, we set $m$ to achieve a given expected computation cost per estimator evaluation for fair comparison with IWAE and related estimators.

### 3.3 TRAINING $q(z;x)$ TO REDUCE VARIANCE

The SUMO estimator does not require amortized variational inference, but the use of an "encoder" to produce the approximate posterior $q(z;x)$ has been shown to be a highly effective way to perform rapid feedforward inference in neural latent variable models. We use $\phi$ to denote the parameters of the encoder $q_\phi(z;x)$. However, the gradients of SUMO with respect to $\phi$ are in expectation zero precisely because SUMO is an unbiased estimator of $\log p_\theta(x)$, regardless of our choice of $q_\phi(z;x)$. Nevertheless, we would expect the choice of $q_\phi(z;x)$ significantly impacts the variance of our estimator. As such, we optimize $q_\phi(z;x)$ to reduce the variance of the SUMO estimator. We can obtain unbiased gradients in the following way (Ruiz et al., 2016; Tucker et al., 2017):

$$\nabla_\phi \text{Var}[\text{SUMO}] = \nabla_\phi \mathbb{E}[\text{SUMO}^2] - \cancel{\nabla_\phi (\mathbb{E}[\text{SUMO}])^2} = \mathbb{E}[\nabla_\phi \text{SUMO}^2]\,. \tag{12}$$

Notably, the expectation of this estimator depends on the variance of SUMO, which we have not been able to bound. In practice, we observe gradients which are bounded but sometimes very large. We apply gradient clipping to the encoder to clip gradients which are excessively large in magnitude. This helps stabilize the training progress but does introduce bias into the encoder gradients. Fortunately, the encoder itself is merely a tool for variance reduction, and biased gradient with respect to the encoder can still significantly help optimization.

### 3.4 APPLICATIONS OF UNBIASED LOG PROBABILITY

Here we list some applications where an unbiased log probability is useful. Using SUMO to replace existing lower bound estimates allows latent variable models to be used for new applications where a lower bound is inappropriate. As latent variable models can be both expressive and efficient to sample from, they are frequently useful in applications where the data are high-dimensional and samples from the model are needed.

**Minimizing $\log p_\theta(x)$.** Some machine learning objectives include terms that seek to increase the entropy of the learned model. The "reverse KL" objective—often used for training models to perform approximate posterior inferences—minimizes $\mathbb{E}_{x \sim p_\theta(x)}[\log p_\theta(x) - \log \pi(x)]$ where $\pi(x)$ is a target density that may only be known up a normalization constant. Local updates of this form are the basis of the expectation propagation procedure (Minka, 2001). This objective has also been used for distilling autoregressive models that are inefficient at sampling (Oord et al., 2018). Moreover, reverse KL is connected to the use of entropy-regularized objectives (Williams & Peng, 1991; Ziebart, 2010; Mnih et al., 2016; Norouzi et al., 2016) in decision-making problems, where the goal is to encourage the decision maker toward exploration and prevent it from settling into a local minimum.

**Unbiased score function $\nabla_\theta \log p_\theta(x)$.** The score function is the gradient of the log-likelihood with respect to the parameters and has uses in estimating the Fisher information matrix and performing stochastic gradient Langevin dynamics (Welling & Teh, 2011), among other applications. Of particular note, the REINFORCE gradient estimator—which is generally applicable for optimizing objectives of the form $\min_\theta \mathbb{E}_{x\sim p_\theta(x)}[R(x)]$—is estimated using the score function. This can be replaced with the gradient of SUMO which itself is an estimator of the score function $\nabla_\theta \log p_\theta(x)$.

$$\begin{aligned}\nabla_\theta \mathbb{E}_{x\sim p_\theta(x)}[R(x)] &= \mathbb{E}_{x\sim p(x)}[R(x)\nabla_\theta \log p_\theta(x)] \\ &= \mathbb{E}_{x\sim p_\theta(x)}[R(x)\nabla_\theta \mathbb{E}[\text{SUMO}(x)]] \\ &= \mathbb{E}_{x\sim p_\theta(x)}[\mathbb{E}[R(x)\nabla_\theta \text{SUMO}(x)]]\end{aligned} \tag{13}$$

where the inner expectation is over the stochasticity of the SUMO estimator. Such estimators are often used in for reward maximization in reinforcement learning where $p_\theta(x)$ is a stochastic policy.

## 4 Related Work

There is a long history in Bayesian statistics of marginal likelihood estimation in the service of model selection. The harmonic mean estimator (Newton & Raftery, 1994), for example, has a long (and notorious) history as a consistent estimator of the marginal likelihood that may have infinite variance (Murray & Salakhutdinov, 2009). The Chib estimator (Chib, 1995), the Laplace approximation, and nested sampling (Skilling, 2006) are alternative proposals that can often have better properties (Murray & Salakhutdinov, 2009). Annealed importance sampling (Neal, 2001) probably represents the gold standard for marginal likelihood estimation.

Russian roulette also has a long history. It dates back to unpublished work from von Neumann and Ulam, who used it to debias Monte Carlo methods for matrix inversion (Forsythe & Leibler, 1950) and particle transport problems (Kahn, 1955). It has gained popularity in statistical physics (Spanier & Gelbard, 1969; Kuti, 1982; Wagner, 1987), for unbiased ray tracing in graphics and rendering (Arvo & Kirk, 1990), and for a number of estimation problems in the statistics community (Wei & Murray, 2017; Girolami et al., 2013; Rychlik, 1990; 1995; Jacob & Thiery, 2015; Jacob et al., 2017). It has also been independently rediscovered many times (Fearnhead et al., 2008; McLeish, 2011; Rhee & Glynn, 2012; Tallec & Ollivier, 2017).

The use of Russian roulette estimation in deep learning and generative modeling applications has been gaining traction in recent years. It has been used to solve short-term bias in optimization problems (Tallec & Ollivier, 2017; Beatson & Adams, 2019). Wei & Murray (2017) estimates the reciprocal normalization constant of an unnormalized density. Han et al. (2018) uses a similar random truncation approach to estimate the distribution of eigenvalues in a symmetric matrix. Along similar motivations with our work, Chen et al. (2019) uses this estimator to construct an unbiased estimator of the change of variables equation in the context of normalizing flows (Rezende & Mohamed, 2015), and Xu et al. (2019) uses it to construct unbiased log probability for a nonparameteric distribution in the context of variational autoencoders (Kingma & Welling, 2013).

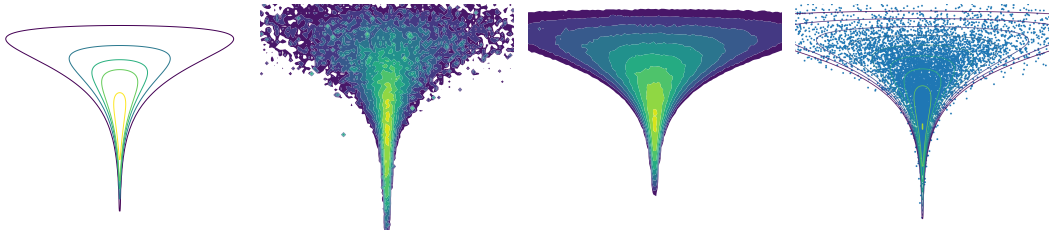
Though we extend latent variable models to applications that require unbiased estimates of log probability and benefit from efficient sampling, an interesting family of models already fulfill these requirements. Normalizing flows (Rezende & Mohamed, 2015; Dinh et al., 2017) offer exact log probability and certain models have been proven to be universal density estimators (Huang et al., 2018). However, these models often require restrictive architectural choices with no dimensionality-reduction capabilities, and make use of many more parameters to scale up than alternative generative models (Kingma & Dhariwal, 2018). Discrete variable versions of these models are still in their infancy and must use biased gradients (Tran et al., 2019; Hoogeboom et al., 2019), whereas latent variable models extend naturally to discrete observations.

## 5 Density Modeling Experiments

We first compare the performance of SUMO when used as a replacement to IWAE with the same expected cost on density modeling tasks. We make use of two benchmark datasets: dynamically binarized MNIST (LeCun et al., 1998) and binarized OMNIGLOT (Lake et al., 2015).

Table 1: Test negative log-likelihood of the trained model, estimated using IWAE($k$=5000). For SUMO, $k$ refers to the expected number of computed terms.

| | MNIST | | | OMNIGLOT | | |
|---|---|---|---|---|---|---|
| Training Objective | $k$=5 | $k$=15 | $k$=50 | $k$=5 | $k$=15 | $k$=50 |
| IWAE (Burda et al., 2016) | 85.54 | — | 84.78 | 106.12 | — | 104.67 |
| IWAE (Our impl.) | 85.28±0.01 | 84.89±0.03 | 84.50±0.02 | 104.96±0.04 | 104.53±0.05 | 103.99±0.12 |
| SUMO | **85.09±0.01** | **84.71±0.02** | **84.40±0.03** | **104.85±0.04** | **104.29±0.12** | 103.79±0.14 |



Target log probability | Training w/ IWAE (k=15) | Training w/ SUMO (k=15) | Model samples

Figure 1: We trained latent variable models for posterior inference, which requires minimizing log probability under the model. Training with IWAE leads to optimizing for the bias while leaving the true model in an unstable state, whereas training with SUMO—though noisy—leads to convergence.

We use the same neural network architecture as IWAE (Burda et al., 2016). The prior $p(z)$ is a 50-dimensional standard Gaussian distribution. The conditional distributions $p(x_i|z)$ are independent Bernoulli, with the decoder parameterized by two hidden layers, each with 200 $\tanh$ units. The approximate posterior $q(z;x)$ is also a 50-dimensional Gaussian distribution with diagonal covariance, whose mean and variance are both parameterized by two hidden layers with 200 $\tanh$ units. We reimplemented and tuned IWAE, obtaining strong baseline results which are better than those previously reported. We use the same hyperparameters to train with the SUMO estimator. We find clipping very large gradients can help performance, as large gradients may be infrequently sampled. This may introduce a small amount of bias into the gradients while reducing variance, but can nevertheless help achieve faster convergence and should still result in a less-biased estimator.

The averaged test log-likelihoods and standard deviations over 3 runs are summarized in Table 1. To be consistent with existing literature, we evaluate our model using IWAE with $5000$ samples. In all the cases, SUMO achieves slightly better performance than IWAE with the same expected cost. We also bold the results that are statistically significant according to an unpaired $t$-test with significance level $0.05$. However, we do see diminishing returns as we increase $k$, suggesting that as we increase compute, the variance of our estimator may impact performance more than the bias of IWAE.

## 6 Latent variables for entropy maximization

We move on to our first task for which a lower bound estimate of log probability would not suffice. The reverse KL objective is useful when we have access to a (possibly unnormalized) target distribution but no efficient sampling algorithm.

$$\min_\theta D_{\mathrm{KL}}(p_\theta(x)||p^*(x)) = \min_\theta \mathbb{E}_{x\sim p_\theta(x)}[\log p_\theta(x) - \log p^*(x)] \tag{14}$$

A major problem with fitting latent variables models to this objective is the presence of an entropy maximization term, effectively a minimization of $\log p_\theta(x)$. Estimating this log marginal probability with a lower bound estimator could result in the optimizing $\theta$ to maximize the bias of the estimator instead of the true objective. Our experiments demonstrate that this causes IWAE to often fail to optimize the objective.

**Modified IWAE for entropy maximization.** The bias of the IWAE estimator can be interpreted as the KL between an importance-weighted approximate posterior $q_{IW}(z;x)$ implicitly defined by

the encoder and the true posterior $p(z|x)$ (Domke & Sheldon, 2018). Both the encoder and decoder parameters can therefore affect this bias. In practice, we find that the encoder optimization proceeds at a faster timescale than the decoder optimization: i.e., the encoder can match $q(z;x)$ to the decoder's $p(x|z)$ more quickly than the latter can match an objective. For this reason, we train the encoder to reduce bias and use a minimax training objective

$$\max_{q(z;x)} \min_{p(x,z)} \mathbb{E}_{x\sim p(x)}[\text{IWAE}_K(x) - \log p^*(x)] \tag{15}$$

Though this is still a lower bound with unbounded bias, it makes for a stronger baseline than optimizing $q(z;x)$ in the same direction as $p(x,z)$. We find that this approach can work well in practice when $k$ is set sufficiently high.

We choose a "funnel" target distribution (Figure 1) similar to the distribution used as a benchmark for inference in Neal et al. (2003), where $p^*$ has support in $\mathbb{R}^2$ and is defined $p^*(x_1,x_2) = \mathcal{N}(x_1;0,1.35^2)\mathcal{N}(x_2;0,e^{2x_1})$ We use neural networks with one hidden layer of 200 hidden units and $\tanh$ activations for both the encoder and decoder networks. We use 20 latent variables, with $p(z)$, $p_\theta(x|z)$, and $q_\phi(z;x)$ all being Gaussian distributed.

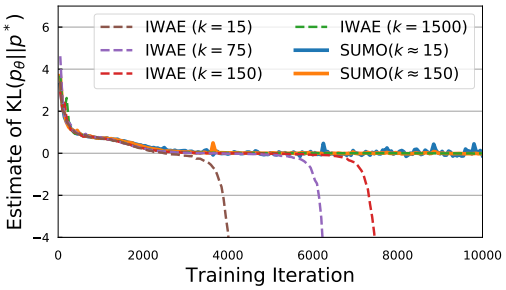



Figure 2: Training with reverse KL requires minimizing $\log p(x)$. SUMO estimates are unbiased and trains well, but minimizing the lower bound IWAE with small $k$ leads to estimates of $-\infty$.

Figure 2 shows the learning curves when using IWAE and SUMO. Unless $k$ is set very large, IWAE will at some point start optimizing the bias instead of the actual objective. The reverse KL is a non-negative quantity, so any estimate significantly below zero can be attributed to the unbounded bias. On the other hand, SUMO correctly optimizes for the objective even with a small expected cost. Increasing the expected cost $k$ for SUMO reduces variance. For the same expected cost, SUMO can optimize the true objective but IWAE cannot. We also found that if $k$ is set sufficiently large, then IWAE can work when we train using the minimax objective in equation 15, suggesting that a sufficiently debiased estimator can also work in practice. However, this requires much more compute and likely does not scale compared to SUMO. We also visualize the contours of the resulting models in Figure 1. For IWAE, we visualize the model a few iterations before it reaches numerical instability.

## 7 Latent variable policies for combinatorial optimization

Let us now consider the problem of finding the maximum of a non-differentiable function, a special case of reinforcement learning without an interacting environment. Variational optimization (Staines & Barber, 2012) can be used to reformulate this as the optimization of a parametric distribution,

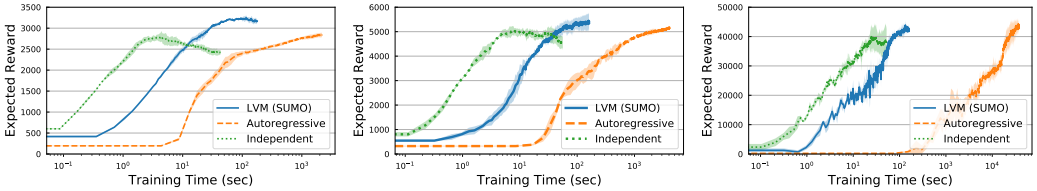
$$\max_x R(x) \geq \max_\theta \mathbb{E}_{x\sim p_\theta(x)}[R(x)], \tag{16}$$

which is now a differentiable function with respect to the parameters $\theta$, whose gradients can be estimated using the REINFORCE gradient estimator and the SUMO estimator (equation 13).

For concreteness, we focus on the problem of quadratic pseudo-Boolean optimization (QPBO) where the objective is to maximize

$$R(x) = \sum_{i=1} w_i(x_i) + \sum_{i<j} w_{ij}(x_i, x_j) \tag{17}$$

where $\{x_i\}_{i=1}^d \in \{0,1\}$ are binary variables. Without further assumptions, QPBO is NP-hard (Boros & Hammer, 2002). As there exist complex dependencies between the binary variables and optimization of equation 16 requires sampling from the policy distribution $p_\theta(x)$, a model that is both expressive and allows efficient sampling would be ideal. For this reason, we motivate the use of latent variable models with independent conditional distributions, which we trained using the SUMO objective. Our baselines are an autoregressive policy, which captures dependencies but for




QPBO w/ 500 variables | QPBO w/ 1000 variables | QPBO w/ 10000 variables

Figure 3: Solving combinatorial optimization with latent variable policies allow faster exploration than autoregressive policy models, while being more expressive than an independent policy.

which sampling must be performed sequentially, and an independent policy, which is easy to sample from but captures no dependencies.

$$p_{\text{LVM}}(x) := \int \prod_{i=1}^{d} p_\theta(x_i|z)p(z)dz \qquad p_{\text{Autoreg}}(x) := \prod_{i=1}^{d} p(x_i|x_{<i}) \qquad p_{\text{Indep}}(x) := \prod_{i=1}^{d} p(x_i)$$

We construct one problem instance for each $d \in \{500, 1000, 10000\}$. For each instance, we randomly sampled the weights $w_i$ and $w_{ij}$ uniformly from the interval $[-1, 1]$. Figure 3 shows the performance of each policy model in solving problems of different sizes, where training time (note the log scale) includes model sampling and updating, but not the computation of the reward function. Including the reward computation would simply increase each method by a constant amount of time, but would not allow us to compare across problem instances due to the quadratic nature of our test problem. All methods were trained with exactly $10000$ iterations. We use the same architecture as in the entropy maximization experiment, except with Bernoulli conditional distributions. The independent policy uses the same decoder architecture as the latent variable policy but always uses a fixed random $z$, whereas the autoregressive policy uses the a masked autoencoder architecture (Germain et al., 2015) with around the same number of parameters. For the masked autoencoder, evaluation of log probability only requires a single forward pass, but sampling requires $d$ forward passes.

We note that even with only $d = 500$ variables, the search space is intractably large for exact optimization. We find that at convergence, the independent policy performs worse than the latent variable and autoregressive policies. While more complex models may have a better grasp of the current "frontier" of reward distributions, the simple independent model must prematurely reduce its entropy in order to obey strong dependencies between variables. This reduction in entropy reduces the final reward due to insufficient exploration. This behavior is indicated by the dips in expected reward during optimization. On the other extreme, the autoregressive policy is slow to update, with each iteration taking more than 3 seconds on the $d = 10000$ problem instance and is roughly $100\times$ slower than the other models. In contrast, the latent variable and independent policies spend similar amounts of time solving all problem instances, and are parallelizable. While the use of SUMO with $k = 15$ makes our latent variable policy slower than the independent policy, the performance gap is roughly $0.5\times$ to $2\times$ the size of the problem instance, suggesting that models that capture dependencies can obtain non-trivial improvement in expected reward.

## 8 Conclusion

We introduced SUMO, a new unbiased estimator of the log probability for latent variable models, and demonstrated tasks for which this unbiased estimator performs better than standard lower bounds. In practice, the high variance of our estimator necessitates gradient clipping during optimization, which re-introduces a small amount of bias. However, even this clipped estimator provides lower-bias estimates than existing estimators at the same compute cost, and works well when minimizing log probability. We plan to investigate new families of gradient-based optimizers which can handle heavy-tailed stochastic gradients. It may be fruitful to investigate the use of convex combination of consistent estimators within the SUMO approach, as any convex combination is unbiased.

## REFERENCES

James Arvo and David Kirk. Particle transport and image synthesis. *ACM SIGGRAPH Computer Graphics*, 24(4):63–66, 1990.

Robert Bamler, Cheng Zhang, Manfred Opper, and Stephan Mandt. Perturbative black box variational inference. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, pp. 5079–5088. Curran Associates, Inc., 2017.

Alex Beatson and Ryan P. Adams. Efficient optimization of loops and limits with randomized telescoping sums. In *International Conference on Machine Learning*, 2019.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022, 2003.

Endre Boros and Peter L Hammer. Pseudo-Boolean optimization. *Discrete Applied Mathematics*, 123(1-3):155–225, 2002.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. 2016.

Ricky TQ Chen, Jens Behrmann, David Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. *arXiv preprint arXiv:1906.02735*, 2019.

Siddhartha Chib. Marginal likelihood from the Gibbs output. *Journal of the American Statistical Association*, 90(432):1313–1321, 1995.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1): 1–22, 1977.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *International Conference on Learning Representations*, 2017.

Justin Domke and Daniel R Sheldon. Importance weighting and variational inference. In *Advances in Neural Information Processing Systems*, pp. 4470–4479, 2018.

Paul Fearnhead, Omiros Papaspiliopoulos, and Gareth O Roberts. Particle filters for partially observed diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70 (4):755–777, 2008.

George E Forsythe and Richard A Leibler. Matrix inversion by a Monte Carlo method. *Mathematics of Computation*, 4(31):127–129, 1950.

Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pp. 881–889, 2015.

Mark Girolami, Anne-Marie Lyne, Heiko Strathmann, Daniel Simpson, and Yves Atchade. Playing Russian roulette with intractable likelihoods. Technical report, Citeseer, 2013.

Insu Han, Haim Avron, and Jinwoo Shin. Stochastic Chebyshev gradient descent for spectral optimization. In *Advances in Neural Information Processing Systems*, pp. 7386–7396, 2018.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew M Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Machine Learning*, 2017.

Emiel Hoogeboom, Jorn WT Peters, Rianne van den Berg, and Max Welling. Integer discrete flows and lossless compression. *arXiv preprint arXiv:1905.07376*, 2019.

Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows. In *International Conference on Machine Learning*, 2018.

Pierre E Jacob and Alexandre H Thiery. On nonnegative unbiased estimators. *The Annals of Statistics*, 43(2):769–784, 2015.

Pierre E Jacob, John O'Leary, and Yves F Atchadé. Unbiased Markov chain Monte Carlo with couplings. *arXiv preprint arXiv:1708.03625*, 2017.

Matthew Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and Sandeep R Datta. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in Neural Information Processing Systems*, pp. 2946–2954, 2016.

Herman Kahn. Use of different Monte Carlo sampling techniques. 1955.

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

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pp. 10215–10224, 2018.

Julius Kuti. Stochastic method for the numerical study of lattice fermions. *Physical Review Letters*, 49(3):183, 1982.

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

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

Don McLeish. A general method for debiasing a Monte Carlo estimator. *Monte Carlo Methods and Applications*, 17(4):301–315, 2011.

Thomas P Minka. Expectation propagation for approximate Bayesian inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pp. 362–369. Morgan Kaufmann Publishers Inc., 2001.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1928–1937, 2016.

Iain Murray and Ruslan Salakhutdinov. Evaluating probabilities under high-dimensional latent variable models. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou (eds.), *Advances in Neural Information Processing Systems 21*, pp. 1137–1144. 2009.

Radford M Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, 2001.

Radford M Neal et al. Slice sampling. *The Annals of Statistics*, 31(3):705–767, 2003.

Michael A Newton and Adrian E Raftery. Approximate Bayesian inference with the weighted likelihood bootstrap. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(1):3–26, 1994.

Mohammad Norouzi, Samy Bengio, Navdeep Jaitly, Mike Schuster, Yonghui Wu, Dale Schuurmans, et al. Reward augmented maximum likelihood for neural structured prediction. In *Advances In Neural Information Processing Systems*, pp. 1723–1731, 2016.

Sebastian Nowozin. Debiasing evidence approximations: On importance-weighted autoencoders and jackknife variational inference. In *International Conference on Learning Representations*, 2018.

Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C Cobo, Florian Stimberg, et al. Parallel Wavenet: Fast high-fidelity speech synthesis. In *International Conference on Machine Learning*, 2018.

Tom Rainforth, Adam R Kosiorek, Tuan Anh Le, Chris J Maddison, Maximilian Igl, Frank Wood, and Yee Whye Teh. Tighter variational bounds are not necessarily better. In *International Conference on Machine Learning*, 2018.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of Adam and beyond. In *International Conference on Learning Representations*, 2018.

Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, 2015.

Chang-han Rhee and Peter W Glynn. A new approach to unbiased estimation for SDEs. In *Proceedings of the Winter Simulation Conference*, pp. 17. Winter Simulation Conference, 2012.

Chang-han Rhee and Peter W Glynn. Unbiased estimation with square root convergence for SDE models. *Operations Research*, 63(5):1026–1043, 2015.

Francisco JR Ruiz, Michalis K Titsias, and David M Blei. Overdispersed black-box variational inference. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, 2016.

Tomasz Rychlik. Unbiased nonparametric estimation of the derivative of the mean. *Statistics & probability letters*, 10(4):329–333, 1990.

Tomasz Rychlik. A class of unbiased kernel estimates of a probability density function. *Applicationes Mathematicae*, 22(4):485–497, 1995.

John Skilling. Nested sampling for general Bayesian computation. *Bayesian Analysis*, 1(4):833–859, 2006.

Jerome Spanier and Ely M Gelbard. *Monte Carlo Principles and Neutron Transport Problems*. Addison-Wesley Publishing Company, 1969.

Joe Staines and David Barber. Variational optimization. *arXiv preprint arXiv:1212.4507*, 2012.

Corentin Tallec and Yann Ollivier. Unbiasing truncated backpropagation through time. *arXiv preprint arXiv:1705.08209*, 2017.

Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

Dustin Tran, Keyon Vafa, Kumar Krishna Agrawal, Laurent Dinh, and Ben Poole. Discrete flows: Invertible generative models of discrete data. *arXiv preprint arXiv:1905.10347*, 2019.

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 *Advances in Neural Information Processing Systems*, pp. 2627–2636, 2017.

George Tucker, Dieterich Lawson, Shixiang Gu, and Chris J Maddison. Doubly reparameterized gradient estimators for Monte Carlo objectives. In *International Conference on Learning Representations*, 2018.

Wolfgang Wagner. Unbiased Monte Carlo evaluation of certain functional integrals. *Journal of Computational Physics*, 71(1):21–33, 1987.

Colin Wei and Iain Murray. Markov chain truncation for doubly-intractable inference. 54:776–784, 20–22 Apr 2017.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning*, pp. 681–688, 2011.

Ronald J Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

Kai Xu, Akash Srivastava, and Charles Sutton. Variational Russian roulette for deep Bayesian nonparametrics. In *International Conference on Machine Learning*, pp. 6963–6972, 2019.

Brian D Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. PhD thesis, figshare, 2010.

# A APPENDIX

## A.1 CONVERGENCE OF $\text{IWAE}_{k+1} - \text{IWAE}_k$

Let $w_i = \frac{p(x|z_i)p(z_i)}{q(z_i|x)}$ and we define $Y_k \coloneqq \frac{1}{k}\sum_{i=1}^{k} w_i$ as the sample mean and we have $\mathbb{E}[Y_k] = \mathbb{E}[w] = \mu$.

$$\begin{aligned}\text{IWAE}_k &= \log Y_k = \log\left[\mu + (Y_k - \mu)\right] \\ &= \log\mu - \sum_{t=1}^{\infty}\frac{(-1)^t}{t\mu^t}(Y_k - \mu)^t \end{aligned} \tag{18}$$

We follow the notations in Nowozin (2018). We use the central moments $\gamma_t \coloneqq \mathbb{E}[(Y_k - \mu)^t]$ and $\mu_t \coloneqq \mathbb{E}[(w - \mu)^t]$ for $t \geq 2$.

$$\mathbb{E}\Delta_k^2 = \mathbb{E}(\text{IWAE}_{k+1} - \text{IWAE}_k)^2 \tag{19}$$

$$= \mathbb{E}\left[\log\mu - \sum_{t=1}^{\infty}\frac{(-1)^t}{t\mu^t}(Y_{k+1} - \mu)^t - \log\mu + \sum_{t=1}^{\infty}\frac{(-1)^t}{t\mu^t}(Y_k - \mu)^t\right]^2 \tag{20}$$

$$= \mathbb{E}\left[\sum_{t=1}^{\infty}\frac{(-1)^t}{t\mu^t}\left[(Y_k - \mu)^t - (Y_{k+1} - \mu)^t\right]\right]^2 \tag{21}$$

Expanding Eq. 21 to order two gives

$$\mathbb{E}\Delta_k^2 = \mathbb{E}\left[-\frac{1}{\mu}(Y_k - \mu - Y_{k+1} + \mu) + \frac{1}{2\mu^2}\left[(Y_k - \mu)^2 - (Y_{k+1} - u)^2\right]\right]^2 + o(k^{-2}) \tag{22}$$

$$= \frac{1}{\mu^2}\mathbb{E}\left[Y_{k+1} - Y_k + \frac{1}{2\mu}(Y_k + Y_{k+1} - 2\mu)(Y_k - Y_{k+1})\right]^2 + o(k^{-2}) \tag{23}$$

$$= \frac{1}{\mu^2}\mathbb{E}\left[2(Y_{k+1} - Y_k) + \frac{1}{2\mu}(Y_k + Y_{k+1})(Y_k - Y_{k+1})\right]^2 + o(k^{-2}) \tag{24}$$

Since we use cumulative sum to compute $Y_k$ and $Y_{k+1}$, we obtain $Y_{k+1} = \frac{kY_k + w_{k+1}}{k+1}$.

$$\implies \mathbb{E}\Delta_k^2 = \frac{1}{\mu^2}\mathbb{E}\left[2\frac{w_{k+1} - 1}{k+1} + \left(\frac{w_{k+1} + \frac{2k+1}{k+1}\sum_{i=1}^{k} w_k}{2k\mu}\right)\left(\frac{w_{k+1} - 1}{k+1}\right)\right]^2 + o(k^{-2}) \tag{25}$$

We note that $\frac{w_{k+1}-1}{k+1} = \mathcal{O}(\frac{1}{k})$ and $\frac{w_{k+1} + \frac{2k+1}{k+1}\sum_{i=1}^{k} w_k}{2k\mu} = \mathcal{O}(1)$. Therefore $\Delta_k$ is $\mathcal{O}(\frac{1}{k})$, and $\mathbb{E}\Delta_k^2 = \mathcal{O}(\frac{1}{k^2})$.

## A.2 CONVERGENCE OF $\Delta_k\Delta_j$

From the previous proof, we have

$$\Delta_k = 2\mu\frac{w_{k+1} - 1}{k+1} + \left(\frac{w_{k+1} + \frac{2k+1}{k+1}\sum_{i=1}^{k} w_k}{2k}\right)\left(\frac{w_{k+1} - 1}{k+1}\right)$$

Without loss of generality, suppose $j \geq k+1$,

$$\mathbb{E}\Delta_k\Delta_j = \mathbb{E}\left[\left(\sum_{t=1}^{\infty}\frac{(-1)^t}{t\mu^t}\left[(Y_k - \mu)^t - (Y_{k+1} - \mu)^t\right]\right)\left(\sum_{t=1}^{\infty}\frac{(-1)^t}{t\mu^t}\left[(Y_j - \mu)^t - (Y_{j+1} - \mu)^t\right]\right)\right] \tag{26}$$

For clarity, let $C_k = Y_k - \mu$ be the zero-mean random variable. Nowozin (2018) gives the relations

$$\mathbb{E}[C_k^2] = \gamma_2 = \frac{\mu_2}{k} \tag{27}$$

$$\mathbb{E}[C_k^3] = \gamma_3 = \frac{\mu_3}{k^2} \tag{28}$$

$$\mathbb{E}[C_k^4] = \gamma_4 = \frac{3\mu_2^2}{k^2} + \frac{\mu_4 - 3\mu_2^2}{k^3} \tag{29}$$

$$\mathbb{E}\Delta_k\Delta_j = \mathbb{E}\left[\left(\sum_{t=1}^{\infty}\frac{(-1)^t}{t\mu^t}(C_k^t - C_{k+1}^t)\right)\left(\sum_{t=1}^{\infty}\frac{(-1)^t}{t\mu^t}(C_j^t - C_{j+1}^t)\right)\right] \tag{30}$$

Expanding both the sums inside the brackets to order two:

$$\begin{aligned}
\mathbb{E}\Delta_k\Delta_j \approx\ & \mathbb{E}\frac{1}{\mu^2}(C_{k+1}-C_k)(C_{j+1}-C_j) && (1)\\
& - \mathbb{E}\frac{1}{2\mu^3}(C_{k+1}^2-C_k^2)(C_{j+1}-C_j) && (2)\\
& - \mathbb{E}\frac{1}{2\mu^3}(C_{k+1}-C_k)(C_{j+1}^2-C_j^2) && (3)\\
& + \mathbb{E}\frac{1}{4\mu^4}(C_{k+1}^2-C_k^2)(C_{j+1}^2-C_j^2) && (4)
\end{aligned}$$

We will proceed by bounding each of the terms $(1)$, $(2)$, $(3)$, $(4)$. First, we decompose $C_j$. Let $B_{k,j} \coloneqq \frac{1}{j}\sum_{i=k+1}^{j}(w_i - \mu)$.

$$C_j = \frac{1}{j}\left(kC_k + \sum_{i=k+1}^{j}(w_i-\mu)\right) = \frac{k}{j}C_k + \frac{1}{j}\sum_{i=k+1}^{j}(w_i-\mu) = \frac{k}{j}C_k + B_{k,j} \tag{31}$$

We know that $B_{k,j}$ is independent of $C_k$ and $\mathbb{E}[B_{k,j}] = 0$, implying $\mathbb{E}[C_kB_{k,j}] = 0$. Note $C_j^2 = \frac{k^2}{j^2}C_k^2 + 2\frac{k}{j}C_kB_{k,j} + B_{k,j}^2$.

Now we show that $(1)$ is zero:

$$\begin{aligned}
\mathbb{E}[\frac{1}{\mu^2}(C_{k+1}-C_k)(C_{j+1}-C_j)] &= \frac{1}{\mu^2}\mathbb{E}\Big[C_{k+1}\frac{k+1}{j+1} + C_{k+1}B_{j+1,k+1}\\
&\qquad - \frac{k+1}{j}C_{k+1}^2 - B_{j,k+1}C_{k+1} - C_k\frac{k}{j+1}\\
&\qquad - C_kB_{j+1,k} + \frac{k}{j}C_k^2 + C_kB_{j,k}\Big]\\
&= \frac{1}{\mu^2}\mathbb{E}[\ -\frac{k+1}{j(j+1)}C_{k+1}^2 + C_k^2\frac{k}{j(j+1)}]\\
&= \frac{1}{\mu^2}[-\frac{k+1}{j(j+1)}\frac{\mu_2}{k+1} + \frac{\mu_2}{k}\frac{k}{j(j+1)}] = 0
\end{aligned}$$

We now investigate $(2)$:

$$\begin{aligned}
\mathbb{E}[-\frac{1}{2\mu^3}(C_{k+1}^2-C_k^2)(C_{j+1}-C_j)] &= \frac{1}{2\mu^3}\mathbb{E}\Big[C_k^3\frac{k}{j+1} + C_k^2B_{j+1,k} - C_k^3\frac{k}{j} - C_k^2B_{j,k}\\
&\qquad + C_{k+1}^3\frac{k+1}{j+1} + C_{k+1}^2B_{j,k} + \frac{k}{j}C_k^2 + C_kB_{j+1,k}\Big]\\
&= \frac{1}{\mu^2}\mathbb{E}[-\frac{k+1}{j(j+1)}C_{k+1}^2 + C_k^2\frac{k}{j(j+1)}]\\
&= \frac{1}{2\mu^3}[-\frac{\mu_3}{kj(j+1)} + \frac{\mu_3}{(k+1)j(j+1)}] = -\frac{\mu_3}{2\mu^3}[\frac{1}{k(k+1)j(j+1)}]
\end{aligned}$$

We now show that $(3)$ is zero:

$$\begin{aligned}
\mathbb{E}[\frac{1}{2\mu^3}(C_{k+1}-C_k)(C_j^2-C_{j+1}^2)] &= \frac{1}{2\mu^3}\mathbb{E}[C_{k+1}C_j^2 - C_{k+1}C_{j+1}^2 - C_kC_j^2 + C_kC_{j+1}^2)]\\
&= \frac{1}{2\mu^3}[\frac{\mu_3}{j^2} - \frac{\mu_3}{(j+1)^2} - \frac{\mu_3}{j^2} - \frac{\mu_3}{(j+1)^2}]\\
&= 0
\end{aligned}$$

Finally, we investigate $(4)$:

$$\mathbb{E}[\frac{1}{4\mu^4}(C_{k+1}^2 - C_k^2)(C_{j+1}^2 - C_j^2)]$$

Using the relation in equation 29, we have

$$\mathbb{E}[C_k^2 C_j^2] = \mathbb{E}[C_k^2(\frac{k^2}{j^2}C_k^2 + \frac{2k}{j}C_k B_{j,k} + B_{j,k}^2)] \tag{32}$$

$$= \frac{k^2}{j^2}\gamma_4 + \gamma_2\frac{(j-k)\mu_2}{j^2} \tag{33}$$

$$= \frac{(2k+j-3)\mu_2^2 + \mu_4}{j^2 k} \tag{34}$$

$$\begin{aligned}
\mathbb{E}[\frac{1}{4\mu^4}(C_{k+1}^2 - C_k^2)(C_{j+1}^2 - C_j^2)] =& \frac{(2k+j-3)\mu_2^2+\mu_4}{j^2k} - \frac{(2k+j-2)\mu_2^2+\mu_4}{(j+1)^2k} \\
&- \frac{(2k+j-1)\mu_2^2+\mu_4}{j^2(k+1)} + \frac{(2k+j)\mu_2^2+\mu_4}{(j+1)^2(k+1)} \\
=& \frac{(j^2-5j-3)\mu_2^2}{j^2(j+1)^2k(k+1)} - \frac{\mu_4}{(j+1)^2k(k+1)} \\
=& \frac{j^2(\mu_2^2-\mu_4) - (5j+3)\mu_2^2}{j^2(j+1)^2k(k+1)} \\
=& \mathcal{O}(j^{-2}k^{-2})
\end{aligned}$$

In summary, $\mathbb{E}\Delta_k\Delta_j$ is $\mathcal{O}(k^{-2}j^{-2})$.

### A.3 Convergence of $\nabla\Big(\text{IWAE}_{k+1} - \text{IWAE}_k\Big)$

The IWAE log likelihood estimate is:

$$\mathcal{L}_k = \log\Big(\frac{1}{k}\sum_{i=1}^{k}\frac{p_\theta(x,z_i)}{q_\psi(z_i|x)}\Big)$$

The gradient of this with respect to $\lambda$, where $\lambda$ is either $\theta$ or $\psi$, is

$$\frac{d\mathcal{L}_k}{d\lambda} = \frac{1}{\sum_{i=1}^{k}\frac{p_\theta(x,z_i)}{q_\psi(z_i|x)}}\sum_{i=1}^{k}\frac{d}{d\lambda}\frac{p_\theta(x,z_i)}{q_\psi(z_i|x)}$$

We abbreviate $w_i := \frac{p_\theta(x,z_i)}{q_\psi(z_i|x)}$, and $\nu_i = \frac{dw_i}{d\lambda}$. In both $\lambda = \psi$ and $\lambda = \theta$ cases, it suffices to treat the $w_i$ and $\nu_i$ as i.i.d. random variables with finite variance and expectation. They are i.i.d. due to i.i.d. sampling of the $z_i$ from $q$, and the conditions on the moments are ensured for reasonable neural network architectures with bounded weights (enforcing a Lipschitz constraint), and are also required for the basic IWAE estimator to have finite variance and expectation.

Consider the differences between two gradients: we label $\Delta^g$ as follows:

$$\Delta_k^g := \frac{d\mathcal{L}_{k+1}}{d\lambda} - \frac{d\mathcal{L}_k}{d\lambda}$$

We have:

$$\begin{aligned}
\Delta_k^g &= \frac{1}{\sum_{i=1}^{k+1}w_i}\nu_{k+1} + \Big(\frac{1}{\sum_{i=1}^{k+1}w_i} - \frac{1}{\sum_{i=1}^{k}w_i}\Big)\sum_{i=1}^{k}\nu_i \\
&= \frac{1}{\sum_{i=1}^{k+1}w_i}\nu_{k+1} + \frac{w_{k+1}}{\big(\sum_{i=1}^{k+1}w_i\big)\big(\sum_{i=1}^{k}w_i\big)}\sum_{i=1}^{k}\nu_i
\end{aligned}$$

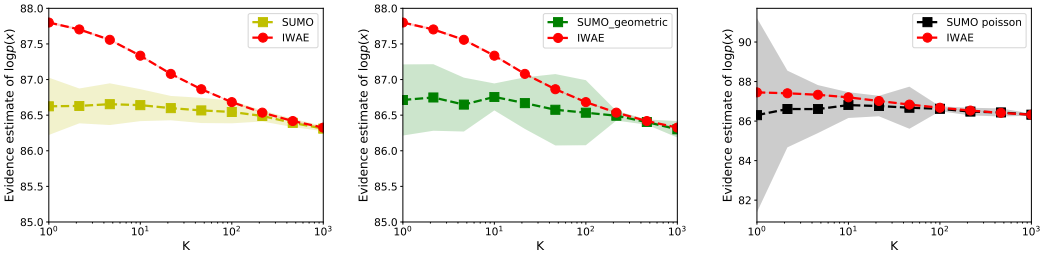



Figure 4: A comparison of SUMO with different distributions and IWAE estimations of test negative log-likelihood on a trained model with $\text{IWAE}_1$ objective on MNIST. The expected cost is $K + 5$ for each evaluation. The results are averaged over 100 runs (mean in bold and std shaded).

We again let $Y_k$ denote the $k$th sample mean $\frac{1}{k}\sum_i w_i$. Then:

$$\Delta_k^g = \frac{1}{kY_k}\nu_{k+1} + \frac{w_{k+1}}{(k+1)Y_kY_{k+1}}\bar{\nu}_k$$

The sample means $Y_k$ and $\bar{\mu}_k$ have finite expectation and variance. The variance vanishes as $k \to \infty$ (but the expectation does not change).

$$\mathbb{E}||\Delta_k^g||_2^2 = \frac{1}{k^2}\mathbb{E}||\frac{\nu_{k+1}}{Y_k} + \frac{k}{k+1}\frac{w_{k+1}\bar{\nu_k}}{Y_kY_{k+1}}||_2^2$$

$$\text{Let } \frac{\nu_{k+1}}{Y_k} + \frac{k}{k+1}\frac{w_{k+1}\bar{\nu_k}}{Y_kY_{k+1}} := \phi_k$$

$$\implies \mathbb{E}||\Delta_k^g||_2^2 = \frac{1}{k^2}||\mathbb{E}\phi_k||_2^2 + \frac{1}{k^2}\text{Var}(\phi_k)$$

The second term vanishes at a rate strictly faster than $\frac{1}{k^2}$: the variance of $\phi_k$ goes to zero as $k \to \infty$. But the first term does not: $\phi_k$ is a biased estimator of $\phi_\infty$ so $\mathbb{E}\phi_k$ does change with $k$, but it does not necessarily go to zero:

$$\mathbb{E}\phi_\infty = \mathbb{E}\Big[\frac{\nu}{\mathbb{E}w} + \frac{k}{k+1}\frac{w\mathbb{E}\nu}{(\mathbb{E}w)^2}\Big] = \frac{\mathbb{E}\nu}{\mathbb{E}w}$$

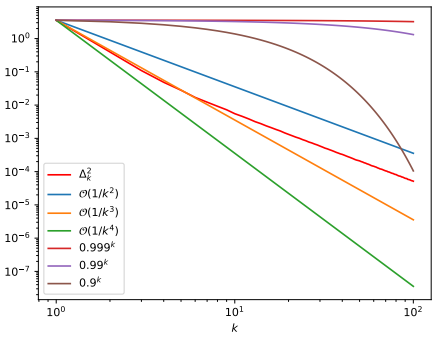
Thus, $\mathbb{E}||\Delta_k^g||_2^2$ is at most $\mathcal{O}(\frac{1}{k^2})$.

### A.4 Empirical Confirmation on the Convergence of $\mathbb{E}\Delta_k^2$ and $\mathbb{E}||\Delta_k^g||_2^2$

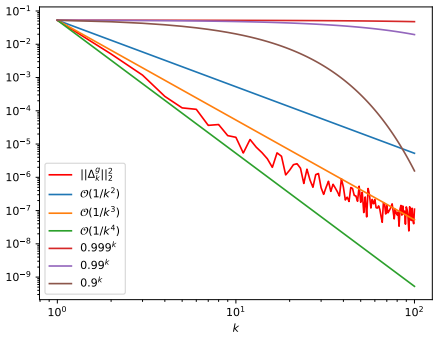
We measure the $\Delta_k^2$ and $||\Delta_k^g||_2^2$ on a toy example to verify the convergence rates empirically. We reimplement the toy Gaussian example from Rainforth et al. (2018); Tucker et al. (2018). The generative model is $p_\theta(x,z) = \mathcal{N}(z|\theta, I)\mathcal{N}(x|z, I)$. The encoder is $q_\phi(z|x) = \mathcal{N}(z|Ax + b, \frac{2}{3}I)$. Alongside $||\Delta_k||_2^2$, we plot several reference convergence rates such as $\mathcal{O}(1/k^c)$, $c > 1$, and $\mathcal{O}(c^k)$, $c < 1$, as a visual guide. The results are shown in Figure 5. Following the setup in Rainforth et al. (2018), we sample a group of model parameters close to the optimal values which are perturbed by Gaussian noise from $\mathcal{N}(0, 0.01^2)$. The gradient $\Delta_k^g$ is taken w.r.t. the model parameter $\theta$.

### A.5 Experimental Setup

All the models are trained using a batch size of 100 and an Amsgrad optimizer (Reddi et al., 2018) with parameters $lr = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-4}$. The learning rate is reduced by factor $0.8$ with a patience of 50 epochs. We use gradient norm scaling in both the inference and generative networks. We train SUMO using the same architecture and hyperparameters as IWAE except the gradient clipping norm. We set the gradient norm to 5000 for encoder and $\{20, 40, 60\}$ for decoder in SUMO. For IWAE, the gradient norm is fixed to 10 in all the experiments. We report the performance of models with early stopping if no improvements have been observed for 300 epochs on the validation set.




(a) Mean of estimated of $\mathbb{E}\Delta_k^2$ with increasing $k$ over ten random trials with 1000 samples per trial. X and Y axis are on log scale. Empirically the convergence rate of $\Delta_k^2$ is between $\mathcal{O}(1/k^2)$ and $\mathcal{O}(1/k^3)$.




(b) Mean of estimated $\mathbb{E}||\Delta_k^g||_2^2$ with increasing $k$ over ten trials with 1000 samples per trial. Empirically the convergence is faster than theoretical analysis $\mathcal{O}(1/k^2)$.

Figure 5: Empricial validation of the convergence rate of the norms of $\Delta$ and $\Delta^g$.

### A.5.1 Reverse KL and Combinatorial Optimization

These two tasks use the same encoder and decoder architecture: one hidden layer with $\tanh$ nonlinearities and 200 hidden units. We set the latent state to be of size 20. The prior is a standard Gaussian with diagonal covariance, while the encoder distribution is a Gaussian with parameterized diagonal covariance. For reverse KL, we used independent Gaussian conditional likelihoods for $p(x|z)$, while for combinatorial optimization we used independent Bernoulli conditional distributions. We found it helps stablize training for both IWAE and SUMO to remove momentum and used RMSprop with learning rate 0.00005 and epsilon 1e-3 for fitting reverse KL. We used Adam with learning rate 0.001 and epsilon 1e-3, plus standard hyperparameters for the combinatorial optimization problems. SUMO used an expected compute of 15 terms, with $m = 5$ and the tail-modified telescoping Zeta distribution.