Revisiting Reweighted Wake-Sleep for Models with Stochastic Control Flow

Tuan Anh Le¹, Adam R. Kosiorek¹,², N. Siddharth¹, Yee Whye Teh², Frank Wood³
¹ Department of Engineering Science, University of Oxford
² Department of Statistics, University of Oxford
³ Department of Computer Science, University of British Columbia

Abstract

Stochastic control-flow models (SCFMs) are a class of generative models that involve branching on choices from discrete random variables. Amortized gradient-based learning of SCFMs is challenging as most approaches targeting discrete variables rely on their continuous relaxations—which can be intractable in SCFMs, as branching on relaxations requires evaluating all (exponentially many) branching paths. Tractable alternatives mainly combine REINFORCE with complex control-variate schemes to improve the variance of naïve estimators. Here, we revisit the reweighted wake-sleep (RWS) [5] algorithm, and through extensive evaluations, show that it outperforms current state-of-the-art methods in learning SCFMs. Further, in contrast to the importance weighted autoencoder, we observe that RWS learns better models and inference networks with increasing numbers of particles. Our results suggest that RWS is a competitive, often preferable, alternative for learning SCFMs.

1 INTRODUCTION

Stochastic control-flow models (SCFMs) describe generative models that employ branching (i.e., the use of if/else/cond statements) on choices from discrete random variables. Recent years have seen such models gain relevance, particularly in the domain of deep probabilistic programming [2][49][52][57], Ch. 7], which allows combining neural networks with generative models expressing arbitrarily complex control flow. SCFMs are encountered in a wide variety of tasks including tracking and prediction [29][41], clustering [45], topic modeling [8], model structure learning [11], counting [14], attention [60], differentiable data structures [17][19], speech & language modeling [7][24], and concept learning [25][30].

While a variety of approaches for amortized gradient-based learning (targeting model evidence lower bound (ELBO)) exist for models using discrete random variables, the majority rely on continuous relaxations of the discrete variables [e.g. [23][34][48][55][56][58], enabling gradient computation through reparameterization [27][46]. The models used by these approaches typically do not involve any control flow on discrete random choices, instead choosing to feed choices from their continuous relaxations directly into a neural network, thereby facilitating the required learning.

In contrast, SCFMs do not lend themselves to continuous-relaxation-based approaches due to the explicit branching requirement on choices from the discrete variables. Consider for example a simple SCFM—a two-mixture Gaussian mixture model (GMM). Computing the ELBO for this model involves choosing mixture identity (Bernoulli). Since a sample from a relaxed variable denotes a point on the surface of a probability simplex (e.g. [0.2, 0.8] for a Bernoulli random variable), instead of its vertices (0 or 1), computing the ELBO would need evaluation of both branches, weighting the resulting computation under each branch appropriately. This process can very quickly become intractable for more complex SCFMs, as it requires evaluation of all possible branches in the computation, of which there may be exponentially many, as illustrated in Figure 2.
Alternatives to continuous-relaxation methods mainly involve the use of the importance weighted autoencoder (IWAE) framework, employing the REINFORCE gradient estimator, combined with control-variate schemes to help decrease the variance of the naïve estimator. Although this approach ameliorates the problem with continuous relaxations in that it does not require evaluation of all branches, it has other drawbacks. Firstly, with more particles, the IWAE estimator adversely impacts inference-network quality, consequently impeding model learning. Secondly, its practical efficacy can still be limited due to high variance and the requirement to design and optimize a separate neural network.

Having characterized the class of models we are interested in (c.f. Figure 1), and identified a range of current approaches (along with their characteristics) that might apply to such models, we revisit reweighted wake-sleep (RWS) schemes coupled with such gradient-variance reduction. To this end, we first review state-of-the-art methods for learning in SCFMs, we demonstrate its efficacy in learning better generative models and inference networks, using lower variance gradient estimators, over a range of computational budgets. To this end, we first revisit RWS and present an extensive evaluation of these methods on i) a probabilistic context free grammar (PCFG) model on sentences, ii) the Attend, Infer, Repeat (AIR) model to perceive and localize multiple MNIST digits, and iii) a pedagogical GMM example that exposes a shortcoming of RWS which we then design a fix for. Our experiments confirm RWS as a competitive, often preferable, alternative for learning SCFMs.

2 BACKGROUND

Consider data \((x^{(n)})_{n=1}^{N}\) sampled from a true (unknown) generative model \(p(x)\), a family of generative models \(p_\theta(z, x)\) of latent variable \(z\) and observation \(x\) parameterized by \(\theta\) and a family of inference networks \(q_\phi(z|x)\) parameterized by \(\phi\). We aim to learn the generative model by maximizing the marginal likelihood over data:

\[
\theta^* = \arg \max_{\theta} \frac{1}{N} \sum_{n=1}^{N} \log p_\theta(x^{(n)}).
\]

Simultaneously, we would like to learn an inference network \(q_\phi(z|x)\) that amortizes inference given observation \(x\); i.e., \(q_\phi(z|x)\) maps an observation \(x\) to an approximation of \(p_\theta(z|x)\). Amortization ensures this function evaluation is cheaper than performing approximate inference of \(p_\theta(z|x)\) from scratch. Our focus here is on such joint learning of generative model and inference network, here referred to as “learning a deep generative model”, although we note that other approaches exist that learn the generative model or inference network in isolation.

We begin by reviewing IWAEs as a general approach for learning deep generative models using stochastic gradient descent (SGD) methods, focusing on generative-model families with discrete latent variables, for which the naïve gradient estimator’s high variance impedes learning. We also review control-variate and continuous-relaxation methods for gradient-variance reduction. IWAEs coupled with such gradient-variance reduction methods are currently the dominant approach for learning deep generative models with discrete latent variables.

2.1 IMPORTANCE WEIGHTED AUTOENCODERS

Burda et al. introduce the IWAE, maximizing the mean ELBOs over data, \(\frac{1}{N} \sum_{n=1}^{N} \text{ELBO}_{IS}^K(\theta, \phi, x^{(n)}),\) where, for \(K\) particles,

\[
\text{ELBO}_{IS}^K(\theta, \phi, x) = \mathbb{E}_{Q_\phi(z_{1:k}|x)} \left[ \log \left( \frac{1}{K} \sum_{k=1}^{K} w_k \right) \right],
\]

\[
Q_\phi(z_{1:k}|x) = \prod_{k=1}^{K} q_\phi(z_k|x), \quad w_k = \frac{p_\theta(z_k, x)}{q_\phi(z_k|x)}.
\]

When \(K = 1\), this reduces to the variational autoencoder (VAE). Burda et al. show that \(\text{ELBO}_{IS}^K(\theta, \phi, x)\) is a lower bound on \(\log p_\theta(x)\) and that increasing \(K\) leads to a tighter lower bound. Further, tighter lower bounds arising from increasing \(K\) improve learning of the generative model, but impair learning of the inference network, as the signal-to-noise ratio of \(\theta^{*}\)’s gradient estimator is \(O(\sqrt{K})\) whereas \(\phi^{*}\)’s is \(O(1/\sqrt{K})\). Note that although Tucker et al. solve this for reparameterizable distributions, the issue persists for discrete distributions. Consequently, poor learning of the inference network, beyond a certain point (large \(K\), can actually impair learning of the generative model as well; a finding we explore in §4.3.

Optimizing the IWAE objective using SGD methods requires unbiased gradient estimators of \(\nabla_\theta \text{ELBO}_{IS}^K(\theta, \phi, x)\) with respect to \(\theta\) and \(\phi\).

\[
\nabla_\theta \text{ELBO}_{IS}^K(\theta, \phi, x) \text{ is estimated by evaluating } \nabla_\theta \log \hat{Z}_K \text{ using samples } z_{1:k} \sim Q_\phi(\cdot|x), \text{ where } \hat{Z}_K = \frac{1}{K} \sum_{k=1}^{K} \hat{w}_k, \quad \nabla_\phi \text{ELBO}_{IS}^K(\theta, \phi, x)
\]
is estimated similarly for models with reparameterizable latents, discrete (and other non-reparameterizable) latents require the REINFORCE gradient estimator:\[59\]
\[
g_{\text{REINFORCE}} = \log \hat{Z}_K \nabla_\phi \log Q_\phi(z_{1:K}|x) + \nabla_\phi \log \hat{Z}_K. \tag{2}
\]

2.2 CONTINUOUS RELAXATIONS AND CONTROL VARIATES

Since the gradient estimator in \[2\] typically suffers from high variance, mainly due to the effect of \(\text{1}\), a number of approaches have been developed to ameliorate the issue. These can be broadly categorized into approaches that directly transform the discrete latent variables (continuous relaxations), or approaches that target improvement of the naïve \text{REINFORCE} estimator (control variates).

Continuous Relaxations: Here, discrete variables are transformed to enable reparameterization [27, 46], helping reduce gradient-estimator variance. Approaches span the Gumbel distribution [23, 34], spike-and-X transforms [48], overlapping exponentials [56], and generalized overlapping exponentials for tighter bounds [55].

Besides difficulties inherent to such methods, such as tuning temperature parameters, or the suitability of undirected Boltzmann machine priors, these methods are not well suited for learning SCFMs as they generate samples on the surface of a probability simplex rather than its vertices. For example, sampling from a transformed Bernoulli distribution yields samples of the form \([\alpha, (1 - \alpha)]\) rather than simply 0 or 1—the latter form required for branching. With relaxed samples, as illustrated in Figure 2, one would need to execute all the exponentially many discrete-variable driven branches in the model, weighting each branch appropriately—something that can quickly become infeasible for even moderately complex models. However, for purposes of comparison, for relatively simple SCFMs, one could apply methods involving continuous relaxations, as demonstrated in \[4.3\].

Control Variates: Here, approaches build on the REINFORCE estimator for the IWAE ELBO objective, designing control-variate schemes to reduce the variance of the naïve estimator. Variational inference for Monte Carlo objectives (VIMCO) [38] eschews designing an explicit control variate, instead exploiting the particle set obtained in IWAE. It replaces \(\text{1}\) with

\[
g_{\text{VIMCO}} = \sum_{k=1}^{K} (\log \hat{Z}_K - \Upsilon_{-k}) \nabla_\phi \log q_\phi(z_k|x), \tag{3}
\]

\[\Upsilon_{-k} = \log \frac{1}{K} \left( \exp \left( \frac{1}{K-1} \sum_{\ell \neq k} \log w_\ell \right) + \sum_{\ell \neq k} w_\ell \right)\]

where \(\Upsilon_{-k} \perp z_k\) and highly correlated with \(\log \hat{Z}_K\).

Finally, assuming \(z_k\) is a discrete random variable with \(C\) categories, \text{REBAR} [53] and \text{RELAX} [16] improve on Mnih and Gregor [37] and Gu et al. [20], replacing \(\text{1}\) as

\[
g_{\text{RELAX}} = \left( \log \hat{Z}_K - c_\rho(\hat{g}_1; K) \right) \nabla_\phi \log Q_\phi(z_{1:K}|x)
+ \nabla_\phi c_\rho(g_1; K) - \nabla_\phi c_\rho(\hat{g}_1; K), \tag{4}
\]

where \(g_k\) is a \(C\)-dimensional vector of reparameterized Gumbel random variates, \(z_k\) is a one-hot argmax function of \(g_k\), and \(\hat{g}_k\) is a vector of reparameterized conditional Gumbel random variates conditioned on \(z_k\). The conditional Gumbel random variates are a form of Rao-Blackwellization used to reduce variance. The control variate \(c_\rho\), parameterized by \(\rho\), is optimized to minimize the gradient variance estimates along with the main ELBO optimization, leading to state-of-the-art performance on, for example, sigmoid belief networks [40].

The main difficulty in using this method is choosing a suitable family of \(c_\rho\), as some choices lead to higher variance despite concurrent gradient-variance minimization.

3 REVISITING REWEIGHTED WAKE-SLEEP

Reweighted wake-sleep (RWS) [5] comes from a family of algorithms [11, 22] for learning deep generative models, eschewing a single objective over parameters \(\theta\) and \(\phi\) in favour of individual objectives for each. We review the RWS algorithm and discuss its pros and cons.

3.1 REWEIGHTED WAKE-SLEEP

Reweighted wake-sleep (RWS) [5] is an extension of the wake-sleep algorithm [11, 22] both of which, like IWAE, jointly learn a generative model and an inference network given data. While IWAE targets a single objective, RWS alternates between objectives, updating the generative model parameters \(\theta\) using a \text{wake-phase} \(\theta\) update and the inference network parameters \(\phi\) using either a \text{sleep}- or a \text{wake-phase} \(\phi\) update (or both).

Wake-phase \(\theta\) update. Given \(\phi, \theta\) is updated using an unbiased estimate of \(\nabla_\theta \log Q_\theta(\phi, x^{(n)}) = \sum_{n=1}^{N} \text{ELBO}_{15}(\theta, \phi, x^{(n)})\), obtained without reparameterization or control variates, as the sampling distribution \(Q_\phi(\cdot|x)\) is independent of \(\phi\).

\[\text{The assumption is needed only for notational convenience. However, using more structured latents leads to difficulties in picking the control-variate architecture.}\]

\[\text{We assume that the deterministic mappings induced by the parameters} \ \theta, \ \phi \ \text{are themselves differentiable, such that they are amenable to gradient-based learning.}\]
Sleep-phase $\phi$ update. Here, $\phi$ is updated to minimize the Kullback-Leibler (KL) divergence between the posteriors under the generative model and the inference network, averaged over the true data distribution and the current generative model

$$
E_{p(x)}[D_{KL}(p_\theta(z|x), q_\phi(z|x))]
= E_{p(x)}[\log p_\theta(z|x) - \log q_\phi(z|x)].
$$  

(5)

Its gradient, $E_{p(x)}[-\nabla_\phi \log q_\phi(z|x)]$, is estimated by evaluating $-\nabla_\phi \log q_\phi(z|x)$, where $z, x \sim p_\theta(z, x)$. The estimator’s variance can be reduced at a standard Monte Carlo rate by increasing the number of samples of $z, x$.

Wake-phase $\phi$ update. Here, $\phi$ is updated to minimize the KL divergence between the posteriors under the generative model and the inference network, averaged over the data distribution of the current generative model

$$
E_{p(x)}[D_{KL}(p_\theta(z|x), q_\phi(z|x))]
= E_{p(x)}[\log p_\theta(z|x) - \log q_\phi(z|x)].
$$  

(6)

The outer expectation $E_{p(x)}[E_{p(x)}[-\nabla_\phi \log q_\phi(z|x)]]$ of the gradient is estimated using a single sample $x$ from the true data distribution $p(x)$, given which, the inner expectation is estimated using self-normalized importance sampling with $K$ particles, using $q_\phi(z|x)$ as the proposal distribution. This results in the following estimator

$$
\sum_{k=1}^{K} \frac{w_k}{\sum_{\ell=1}^{K} w_\ell} \left( -\nabla_\phi \log q_\phi(z_k|x) \right),
$$  

where, similar to $\{1\}$, $x \sim p(x), z_k \sim q_\phi(z_k|x)$, and $w_k = p_\theta(z_k, x)/q_\phi(z_k|x)$. Note that (7) is the negative of the second term of the REINFORCE estimator of the IWAE ELBO in (2). The crucial difference between the wake-phase $\phi$ update and the sleep-phase $\phi$ update is that the expectation in (6) is over the true data distribution $p(x)$ and the expectation in (5) is under the current model distribution $p_\theta(x)$. The former is desirable from the perspective of amortizing inference over data from $p(x)$, and although its estimator is biased, this bias decreases as $K$ increases.

3.2 PROS OF REWEIGHTED WAKE-SLEEP

While the gradient update of $\theta$ targets the same objective as IWAE, the gradient update of $\phi$ targets the objective in (5) in the sleep case and (6) in the wake case. This makes RWS a preferable option to IWAE for learning inference networks because the $\phi$ updates in RWS directly target minimization of the expected KL divergences from the true to approximate posterior. With an increased computational budget, using more Monte Carlo samples in the sleep-phase $\phi$ update case and more particles $K$ in the wake-phase $\phi$ update, we obtain a better estimator of these expected KL divergences. This is in contrast to IWAE, where optimizing ELBO $K$ targets a KL divergence on an extended sampling space ($33$) which for $K > 1$ doesn’t correspond to a KL divergence between true and approximate posteriors (in any order). Consequently, increasing $K$ in IWAE leads to impaired learning of inference networks.

Moreover, targeting $D_{KL}(p, q)$ as in RWS can be preferable to targeting $D_{KL}(q, p)$ as in VAES. The former encourages mean-seeking behavior, having the inference network to put non-zero mass in regions where the posterior has non-zero mass, whereas the latter encourages mode-seeking behavior, having the inference network to put mass on one of the modes of the posterior.

In theory, these biases should not affect the fixed point $\theta^*, \phi^*$ where $p_\theta^*(x) = p(x)$ and $q_{\phi^*}(z|x) = p_\theta^*(z|x)$, First, if $\theta \to \theta^*$ through the wake-phase $\theta$ update, the data distribution bias reduces to zero. Second, although the wake-phase $\phi$ gradient estimator is biased, it is consistent—with large enough $K$, convergence of stochastic optimization is theoretically guaranteed on convex objectives and empirically on non-convex objectives. Further, this gradient estimator follows the central limit theorem, so its asymptotic variance decreases linearly with $K$, Eq. (9.8)). Thus, using larger $K$ improves learning of the inference network.
4 EXPERIMENTS

The IWAE and RWS algorithms have primarily been applied to problems with continuous latent variables and/or discrete latent variables that do not actually induce branching (such as sigmoid belief networks; [40]). The purpose of the following experiments is to compare RWS to IWAE combined with control variates and continuous relaxations (c.f. [3]) on models with conditional branching, and show that it outperforms such methods. We empirically demonstrate that increasing the number of particles $K$ can be detrimental in IWAE but advantageous in RWS, as evidenced by achieved ELBOS and average distance between true and amortized posteriors.

In the first experiment, we present learning and amortized inference in a PCFG [4], an example SCFM where continuous relaxations are inapplicable. We demonstrate that RWS outperforms IWAE with a control variate both in terms of learning and inference. The second experiment focuses on Attend, Infer, Repeat (AIR), the deep generative model of [14]. It demonstrates that RWS leads to better learning of the generative model in a setting with both discrete and continuous latent variables, for modeling a complex visual data domain (c.f. §4.2). The final experiment involves a GMM (§4.3), thereby serving as a pedagogical example. It explains the causes of why RWS might be preferable to other methods in more detail.

Notationally, the different variants of RWS will be referred to as wake-sleep (WS) and wake-wake (WW). The wake-phase $\theta$ update is always used. We refer to using it in conjunction with the sleep-phase $\phi$ update as WS and using it in conjunction with the wake-phase $\phi$ update as WW. Using both wake- and sleep-phase $\phi$ updates doubles the required stochastic sampling while yielding only minor improvements on the models we considered. The number of particles $K$ used for the wake-phase $\theta$ and $\phi$ updates is always specified, and computation between them is matched so a wake-phase $\phi$ update with batch size $B$ implies a sleep phase $\phi$ update with $KB$ samples.

4.1 PROBABILISTIC CONTEXT-FREE GRAMMAR

In this experiment we learn model parameters and amortize inference in a PCFG [4]. Each discrete latent variable in a PCFG chooses a particular child of a node in a tree.

Depending on each discrete choice, the generative model can lead to different future latent variables. A PCFG is an example of an SCFM where continuous relaxations cannot be applied—weighing combinatorially many futures by a continuous relaxation is infeasible and doing so for futures which have infinite latent variables is impossible.

While supervised approaches have recently led to state-of-the-art performance in parsing [9], PCFGs remain one of the key models for unsupervised parsing [35]. Learning in a PCFG is typically done via expectation-maximization [12] which uses the inside-outside algorithm [31]. Inference methods are based on dynamic programming [14, 61] or search [28]. Applying RWS and IWAE algorithms to PCFGs allows learning from large unlabeled datasets through SGD while inference amortization ensures linear-time parsing in the number of words in a sentence, at test-time. Moreover, using the inference network as a proposal distribution in IS provides asymptotically exact posteriors if parses are ambiguous.

A PCFG is defined by sets of terminals (or words) $\{t_i\}$, non-terminals $\{n_i\}$, production rules $\{n_i \rightarrow \zeta_j\}$ with $\zeta_j$ a sequence of terminals and non-terminals, probabilities for each production rule such that $\sum_j P(n_i \rightarrow \zeta_j) = 1$ for each $n_i$, and a start symbol $n_1$. Consider the Astronomers PCFG given in Manning et al. [35] (c.f. Appendix A). A parse tree $z$ is obtained by recursively applying the production rules until there are no more non-terminals. For example, a parse tree (S (NP astronomers) (VP (V saw) (NP stars))) is obtained by applying the production rules as follows:

$$
S \xrightarrow{1.0} NP \xrightarrow{0.1} \text{astronomers} \xrightarrow{0.7} \text{astronomers} V NP \xrightarrow{1.0} \text{astronomers} \xrightarrow{0.18} \text{saw stars},
$$

where the probability $p(z)$ is obtained by multiplying the corresponding production probabilities as indicated on top of the arrows. The likelihood of a PCFG, $p(x|z)$, is 1 if the sentence $x$ matches the sentence produced by $z$ (in this case “astronomers saw stars”) and 0 otherwise. One can easily construct infinitely long $z$ by choosing productions which contain non-terminals, for example: $S \rightarrow NP \xrightarrow{0.5} NP \xrightarrow{0.5} NP \xrightarrow{0.5} ...$

We learn the production probabilities of the PCFG and an inference network computing the conditional distribution of a parse tree given a sentence. The architecture of the inference network is the same as described in [32 Section 3.3] except the input to the recurrent neural network (RNN) consists only of the sentence embedding, previous sample embedding, and an address embedding. Each word is represented as a one-hot vector and the sentence embedding is obtained through another RNN. Instead of a hard $\{0, 1\}$ likelihood which can make learning difficult, we use a relaxation, $p(x|z) = \exp(-L(x, s(z))^2)$.
where $L$ is the Levenshtein distance and $s(z)$ is the sentence produced by $z$. Using the Levenshtein distance can also be interpreted as an instance of approximate Bayesian computation \cite{50}. Training sentences are obtained by sampling from the \textit{astronomers} PCFG with the true production probabilities.

We run \textit{WW}, \textit{WS}, \textit{VIMCO} and \textit{REINFORCE} ten times for $K \in \{2, 5, 10, 20\}$, with batch size $B = 2$, using the Adam optimizer \cite{26} with default hyperparameters. We observe that the inference network can often end up sub-optimally sampling very long $z$ (by choosing production rules with many non-terminals), leading to slow and ineffective runs. We therefore cap the runtime to 100 hours—out of ten runs, \textit{WW}, \textit{WS}, \textit{VIMCO} and \textit{REINFORCE} retain on average 6, 6, 5.75 and 4 runs respectively. In Figure 3, we show both (i) the quality of the generative model as measured by the average KL between the true and the model production probabilities, and (ii) the quality of the inference network as measured by $\mathbb{E}_{p(x)}[D_{KL}(p(z|x), q_{\theta}(z|x))]$ which is estimated up to an additive constant (the conditional entropy $H(p(z|x))$) by the sleep-$\phi$ loss \cite{5} using samples from the true PCFG.

Quantitatively, \textit{WS} improves as $K$ increases and outperforms \textit{IWAE}-based algorithms both in terms of learning and inference amortization. While \textit{WW}'s inference amortization improves slightly as $K$ increases, it is significantly worse than \textit{WS}'s. This is because \textit{WS} proposals will rarely produce a parse tree $z$ for which $s(z)$ matches $x$, leading to extremely biased estimates of the wake-$\phi$ update. In this case, this bias is more significant than that of the data-distribution which can harm the sleep-$\phi$ update.

We inspect the quality of the inference network by sampling from it. Figure 4 shows samples from an inference network trained with \textit{WS}, conditioned on the sentence “astronomers saw stars with telescopes”, weighted according to the frequency of occurrence. Appendix A further includes samples from an inference network trained with \textit{VIMCO}, showing that none of them match the given sentence ($s(z) \neq x$), and whose production probabilities are poor, unlike with \textit{RWS}.

\subsection{4.2 ATTEND, INFER, REPEAT}

Next, we evaluate \textit{WW} and \textit{VIMCO} on \textit{AIR} \cite{14}, a structured deep generative model with both discrete and continuous latent variables. \textit{AIR} uses the discrete variable to decide how many continuous variables are necessary to explain an image. The sequential inference procedure of \textit{AIR} poses a difficult problem, since it implies a sequential decision process with possible branching. See \cite{14} and Appendix B for the model notation and details.

We set the maximum number of inference steps in \textit{AIR} to three and train on $50 \times 50$ images with zero, one or
two MNIST digits. The training and testing data sets consist of 60000 and 10000 images respectively, generated from the respective MNIST train/test datasets. Unlike AIR, which used Gaussian likelihood with fixed standard deviation and continuous inputs (i.e., input $x \in [0, 1]^{50 \times 50}$), we use a Bernoulli likelihood and binarized data; the stochastic binarization is similar to Burda et al. [6]. Training is performed over two million iterations by RmsProp [51] with the learning rate of $10^{-5}$, which is divided by three after 400k and 1000k training iterations. We set the glimpse size to $20 \times 20$.

We first evaluate the generative model via the average test log marginal where each log marginal is estimated by a one-sample, 5000-particle IWAE estimate. The inference network is then evaluated via the average test KL from the inference network to the posterior under the current model where each $D_{KL}(qφ(z|x), p_θ(z|x))$ is estimated as a difference between the log marginal estimate above and a 5000-sample, one-particle IWAE estimate. Note that this estimate is just a proxy to the desired KL from the inference network to the true model posterior.

This experiment confirms that increasing number of particles improves VIMCO only up to a point, whereas WW improves monotonically with increased $K$ (Figure 5). WW also results in significantly lower variance and better inference networks than VIMCO.

### 4.3 GAUSSIAN MIXTURE MODEL

In order to examine the differences between RWS and IWAE more closely, we study a GMM which branches on a discrete latent variable to select cluster assignments. The generative model and inference network are defined as

$$p_θ(z) = \text{Cat}(z|\text{softmax}(θ)),$$

$$p(x|z) = \mathcal{N}(x|μ_z, σ_z^2),$$

$$qφ(z|x) = \text{Cat}(z|\text{softmax}(θφ(x))),$$

where $z \in \{0, \ldots, C - 1\}$, $C$ is the number of clusters and $μ_z, σ_z^2$ are fixed to $μ_z = 10c$ and $σ_z^2 = 5^2$. The generative model parameters are $θ \in \mathbb{R}^C$. The inference network consists of a multilayer perceptron $θφ : \mathbb{R} \rightarrow \mathbb{R}^C$, with the 1-16-C architecture and the tanh nonlinearity, parameterized by $φ$. The chosen family of inference networks is empirically expressive enough to capture the posterior under the true model. The true model is set to $p_θ(x)$ where $\text{softmax}(θ_{true})_{c} = (c + 5)/\sum_{c=1}^{C} (i + 5)$ ($c = 0, \ldots, C - 1$), i.e., the mixture probabilities are linearly increasing with the $z$. We fix the mixture parameters in order to study the important features of the problem at hand in isolation.

We train using WS, WW, as well as using IWAE with REINFORCE, RELAX, VIMCO and the Concrete distribution. We attempted different variants of relaxations [48, 56] in this setting, but they performed considerably worse than any of the alternatives (c.f. Appendix F). We fix $C = 20$ and increase number of particles from $K = 2$ to 20. We use the Adam optimizer with default parameters. Each training iteration samples a batch of 100 data points from the true model. Having searched over several temperature schedules for the Concrete distribution, we use the one with the lowest trainable terminal temperature (linearly annealing from 3 to 0.5). We found that using the control variate $c_φ(g_{1:K}) = \frac{1}{K} \sum_{k=1}^{K} \text{MLP}_φ([x, g_k])$, with multilayer perceptron (MLP) architecture $(1 + C)$-16-16-1 (tanh) led to most stable training (c.f. Appendix C). The generative model is evaluated via the $L_2$ distance between the probability mass functions (PMFs) of its prior and true prior as $||\text{softmax}(θ) - \text{softmax}(θ_{true})||$. The inference network is evaluated via the $L_2$ distance between PMFs of the current and true posteriors, averaged over a fixed set ($M = 100$) of observations $(x_{test}^{(m)})_{m=1}^M$ from the true model: $\frac{1}{M} \sum_{m=1}^{M} ||qφ(z|x_{test}^{(m)}) - p_{true}(z|x_{test}^{(m)})||$.

We demonstrate that using WS and WW with larger particle budgets leads to better inference networks whereas this is not the case for IWAE methods (Figure 6, bottom). Recall that the former is because using more samples to estimate the gradient of the sleep $φ$ objective [5] for WS reduces variance at a standard Monte Carlo rate and that
using more particles in (7) to estimate the gradient of the wake $\phi$ objective results in a lower bias. The latter is because using more particles results in the signal-to-noise of IWAE’s $\phi$ gradient estimator to drop at the rate $O(1/\sqrt{K})$.

Learning of the generative model, through inference-network learning, also monotonically improves with increasing $K$ for WS and WW, but worsens for all IWAE methods except VIMCO, since the $\theta$ gradient estimator (common to all methods), $\nabla_{\theta} \text{ELBO}^K_{\phi}(\theta, \phi, x)$ can be seen as an importance sampling estimator whose quality is tied to the proposal distribution (inference network).

To highlight the difference between WW and WS, we study the performance of the generative model and the inference network for different initializations of $\theta$. In Figure 6, $\theta$ is initialized such that the mixture probabilities are exponentially decreasing with $z$ which results in the data distribution $p_{\theta}(x)$ being far from $p_{\theta^*}(x)$. Consequently, the sleep-phase $\phi$ update is highly biased which is supported by WS being worse than WW. On the other hand, if $\theta$ is initialized such that the mixture probabilities are equal, $p_{\theta}(x)$ is closer to $p_{\theta^*}(x)$, which is supported by WS outperforming WW (see Appendix C.2).

We now describe a failure mode affecting WS, WW, VIMCO, RELAX and REINFORCE due the adverse initialization of $\theta$ which we call branch-pruning. It is best illustrated by inspecting the generative model and the inference network as training progresses, focusing on the low-particle ($K=2$) regime (Figure 7). For WS, the generative model $p_{\theta}(z)$ peaks at $z = 9$ and puts zero mass for $z > 9$; the inference network $q_{\phi}(z|x)$ becomes the posterior for this model which, here, has support at most $\{0, \ldots, 9\}$ for all $x$. This is a local optimum for WS as (i) the inference network already approximates the posterior of the model $p_{\theta}(z|x)$ well, and (ii) the generative model $p_{\theta}(z)$, trained using samples from $q_{\phi}(z|x)$, has no samples outside of its current support. Similar failures occur for WW and VIMCO/RELAX/REINFORCE although the support of the locally optimal $p_{\theta}(z)$ is larger ($\{0, \ldots, 14\}$ and $\{0, \ldots, 17\}$ respectively).

While this failure mode is a particular feature of the adverse initialization of $\theta$, we hypothesize that WS and WW suffer from it more, as they alternate between two different objectives for optimizing $\theta$ and $\phi$. WS attempts to amortize inference for the current model distribution $p_{\theta}(x)$ which reinforces the coupling between the generative model and the inference network, making it easier to get stuck in a local optimum. WW with few particles (say $K=1$) on the other hand, results in a highly-biased gradient estimator (7) that samples $z$ from $q_{\phi}(z|x)$ and evaluates $\nabla_{\phi} \log q_{\phi}(z|x)$; this encourages the inference network to concentrate mass. This behavior is not seen in WW with many particles where it is the best algorithm at learning both a good generative model and inference network.

We propose a simple extension of WW, denoted $\delta$-WW, that mitigates this shortcoming by changing the proposal of the self-normalized importance sampling estimator in (7) to $q_{\phi,\delta}(z|x) = (1-\delta)q_{\phi}(z|x) + \delta \text{Uniform}(z)$. We use $\delta = 0.2$, noting that the method is robust to a range of values. Using a different proposal than the inference network $q_{\phi}(z|x)$ means that using the low-particle es-
timator in (7) no longer leads to branch-pruning. This is known as defensive importance sampling [21], and is used to better estimate integrands that have long tails using short-tailed proposals. Using δ-WW outperforms all other algorithms in learning both the generative model and the inference network in the low-K regime and performs similarly as WW in the high-K regime.

5 DISCUSSION

The central argument here is that where one needs both amortization and model learning for SCFMs, the RWS family of methods is preferable to IWAE with either continuous relaxations or control-variates. The PCFG experiment (§4.1) demonstrates a setting where continuous relaxations are inapplicable due to potentially infinite recursion, but where RWS applies and WS outperforms all other methods. The AIR experiment (§4.2) highlights a case where more particles, performance of VIMCO degrades for the inference network [44] and consequently the generative model as well, but where RWS’s performance on both increases monotonically. Finally, the analysis on GMMs (§4.3) focuses on a simple model to understand nuances in the performances of different methods. Beyond implications from prior experiments, it indicates that for the few-particle regime, the WW gradient estimator can be biased, leading to poor learning. For this, we design an alternative involving defensive sampling that ameliorates the issue. The precise choice of which variant of RWS to employ depends on which of the two kinds of gradient bias described in §3.3 dominates. Where the data distribution bias dominates, as with the AIR experiment, WW is preferable, and where the self-normalized IS bias dominates, as in the PCFG experiment, WS is preferable. In the GMM experiment, we verify this empirically by studying two optimization procedures with low and high data distribution biases.

Acknowledgments

TAL’s research leading to these results is supported by EPSRC DTA and Google (project code DF6700) studentships. AK’s and YWT’s research leading to these results are supported by funding from the European Research Council under the European Unions Seventh Framework Programme (FP7/2007-2013) ERC grant agreement no. 617071. NS is supported by EPSRC/MURI grant EP/N019474/1. FW’s research leading is supported by The Alan Turing Institute under the EPSRC grant EP/N510129/1; DARPA PPAML through the U.S. AFRL under Cooperative Agreement FA8750-14-2-0006; Intel and DARPA D3M, under Cooperative Agreement FA8750-17-2-0093.
References


A PROBABILISTIC CONTEXT-FREE GRAMMAR

We show the astronomers PCFG in Figure 8. Figure 9 shows samples from an inference network trained with VIMCO with $K = 20$, conditioned on the sentence $x = \text{"astronomers saw stars with telescopes"}$. Figure 10 shows production probabilities of the non-terminal NP learned by VIMCO and WS with $K = 20$.

$$S \rightarrow \text{NP VP (1.0)}$$
$$\text{NP} \rightarrow \text{NP PP (0.4)}|\text{astronomers (0.1)}|\text{ears (0.18)}|\text{saw (0.04)}|\text{stars (0.18)}|\text{telescopes (0.1)}$$
$$VP \rightarrow \text{V NP (0.7)}|\text{VP PP (0.3)}$$
$$\text{PP} \rightarrow \text{P NP (1.0)}$$
$$\text{P} \rightarrow \text{with (1.0)}$$
$$V \rightarrow \text{saw (1.0)}.$$

Figure 8: The astronomers PCFG from Manning et al. [35, Table 11.2]. The terminals are \{astronomers, ears, saw, stars, telescopes, with\}, the non-terminals are \{S, NP, VP, PP, P, V\} and the start symbol is S. Each row above lists production rules \{$n_i \rightarrow \zeta_j$\} with the corresponding probabilities $p_{ij}$ in the format $n_i \rightarrow \zeta_1 (p_{i1})|\zeta_2 (p_{i2})|\cdots|\zeta_J (p_{ij})$.

$\begin{array}{cccc}
0.029 & 0.021 & 0.016 & 0.015 \\
0.029 & 0.021 & 0.016 & 0.015 \\
\end{array}$

Figure 9: Samples from the inference network which was trained with VIMCO with $K = 20$.

B ATTEND, INFER, REPEAT

AIR is a model with many components and might be difficult to understand if not described explicitly. Here, we outline details of our implementation and provide pseudo-code for the inference (Algorithm 1) and generative models (Algorithm 2) in the case of continuous data and Gaussian data likelihood.

Algorithm 1: Inference in AIR

**Input:** Image $x$, maximum number of inference steps $N$ (0.029 0.021 0.016 0.015 $\cdots$)

for $n \in [1, \ldots, N]$ do

\[ w_n, h_n = R_\theta (x, z_n^{\text{what}}, z_n^{\text{where}}, h_{n-1}) \]

\[ p_n \sim \text{Bernoulli} (p | w_n) \]

if $p_n = 0$ then

break

end if

\[ z_n^{\text{where}} \sim q_n^{\text{where}} (z_n^{\text{where}} | w_n) \]

\[ g_n = \text{STN} (x, z_n^{\text{where}}) \]

\[ z_n^{\text{what}} \sim q_n^{\text{what}} (z_n^{\text{what}} | g_n) \]

end for

**Output:** $z_1^{\text{what}}, z_1^{\text{where}}, \ldots, z_N^{\text{what}}, z_N^{\text{where}}$

Algorithm 2: Generation in AIR

**Input:** $z_1^{\text{what}}, z_1^{\text{where}}, \ldots, z_N^{\text{what}}, z_N^{\text{where}}$

$y_0 = 0$

for $t \in [1, \ldots, n]$ do

\[ \hat{y}_t = h_\theta (z_t^{\text{what}}) \]

\[ y_t = y_{t-1} + \text{STN}^{-1} (\hat{y}_t, z_t^{\text{where}}) \]

$\hat{x} \sim \text{Normal} (x | y_n, \sigma_x^2 I)$

end for

**Output:** $\hat{x}$

C GAUSSIAN MIXTURE MODEL

C.1 CONTROL VARIATES

Here, we present the architectures for the REBAR/RELAX control variate used in the GMM experiment.

The reparameterized sampling of Gumbels and conditional Gumbels is described by Tucker et al. [53, Appendix C] and Grathwohl et al. [16, Appendix B]. In the following, we describe architectures used for the GMM experiment (§ 4.3).

REBAR proposes the following architecture for the control variate $c_\rho (g; k)$:

$$c_\rho^{\text{REBAR}} (g; k) = \rho_1 \log \left( \frac{1}{K} \sum_{k=1}^{K} \frac{p_{\theta} (\sin (g_k / e^{p_2}), x)}{q_\phi (\sin (g_k / e^{p_2}) | x)} \right)^{1 / \rho}$$

where $\rho = (\rho_1, \rho_2)$, and $\sin$ refers to the softmax function. While the functional form of $\rho$ suggests that it will be highly correlated with $\log (1 / \sum_{k=1}^{K} w_k)$, the terms PMFs in the fraction are undefined due to the softmax.
C.2 ADDITIONAL RESULTS

A straightforward fix is to evaluate “a soft PMF” instead:

\[
p_\theta(s(g_k / e^{r_2}), x) = p_\theta(s(g_k / e^{r_2})) p(x | s(g_k / e^{r_2}))
\]

\[
= \text{Categorical}(s(g_k / e^{r_2}) | s(\theta)).
\]

\[
\text{Normal}(x | \mu s(g_k / e^{r_2}), \sigma^2 s(g_k / e^{r_2}))
\]

\[
\approx \text{sm}(g_k / e^{r_2})^T \text{sm}(\theta).
\]

\[
q_\phi(s(g_k / e^{r_2}) | x)
\]

\[
= \text{Categorical}(s(g_k / e^{r_2}) | q_\phi(x))
\]

\[
\approx \text{sm}(g_k / e^{r_2})^T \text{sm}(q_\phi(x)).
\]

Optimization of the log-temperature \(\rho_2\) is highly sensitive to low values can make the training unstable.

RELAX proposes using an arbitrary neural network for \(c_\theta(g_{1:K})\). Due to the symmetry in the arguments, we pick the following for the GMM experiment:

\[
\epsilon^\text{RELAX}_\rho(g_{1:K}) = \frac{1}{K} \sum_{k=1}^K \text{MLP}_\rho([x, g_k]),
\]

where the architecture of the MLP is \((1 + C)\)-16-16-1 (with the tanh nonlinearity between layers) and \(\rho\) are the weights is its weights. This architecture is—unlike the one in (8)—well-defined for all inputs. A drawback of using such control variate is that it can start out not being very correlated with \(\log \left( \frac{\epsilon_{\rho}}{K \sum_{k=1}^K w_k} \right)\).

RELAX also proposes using a summation of the free-form control variate like the one in (9) and a more correlated control variate in (8).

We have tried all architectures and found that (9) leads to the most stable and best training.

Using REBAR/RELAX for more complicated models is possible, however designing an architecture that is highly correlated with the high-variance term and stable to train still remains a challenge.

D SIGMOID BELIEF NETWORKS

In Figure 13, we show training of sigmoid belief networks with three stochastic layers with the same architecture as in Mnih and Rezende [38]. We additionally drive number of particles up to \(K = 5000\) and include KL plots. We find that in high particle regimes, model learning is virtually the same for WW and VIMCO. However, WW outperforms VIMCO in terms of inference network learning.

E DISCRETE VAES

Rolfe 2016 [48] introduces discrete VAEs (DVAEs). It combines a prior over binary latent variables with an element-wise spike-and-Smoothing transformation, allowing approximate marginalization of the discrete variables. This results in a continuous relaxation of discrete variables and a low-variance gradient estimator. Vahdat et al. [56] replaced the original transformation with an overlapping exponential transformation, leading to a yet lower-variance gradient estimator. While both approaches produce relaxed binary variables, the relaxation is significantly less tight [56]. Appendix C, Figure 5) then the CONCRETE of [23, 34] Both approaches require analytical inverse CDFs of the smoothing transformations, a shortcoming addressed by Vahdat et al. [55] — it also leads to a tighter relaxation than its predecessors, however no comparison to CONCRETE is available.

DVAE was designed for undirected binary priors, e.g. restricted Boltzmann machines (RBM), and it does not account for the case of categorical latent variables. It is possible to construct a \(d\)-dimensional categorical variable from \(d-1\) binary variables via stick-breaking construction. This process is slow, however, as it requires \(O(d)\) sequential operations and cannot be parallelized. Moreover, in the case of relaxed variables, the tightness of the derived relaxed categorical variable decreases exponentially with the number of dimensions. This is a major issue in control flows: not only we have to evaluate all branches of the control flow, but the indicator variables that we multiply with outcomes of different branches become exponentially loose with the depth of the flow.
Figure 11: Standard deviation of gradient estimator of $\phi$ for GMM. Median and interquartile ranges from 10 repeats shown. WW and WS have lower-variance gradient estimators of $\phi$ than IWAE except VIMCO, as they avoid the high-variance term $\delta$ in equation 2. This is a necessary, but not sufficient, condition for efficient learning, with other factors being gradient direction and the ability to escape local optima. The standard deviation of $\phi$’s gradient estimator is given by $\frac{1}{D_\phi} \sum_{d=1}^{D_\phi} \text{std}(g_d)$ where $g_d$ is the $d$th (out of $D_\phi$) element of one of $\phi$’s gradient estimators (e.g. (2) for REINFORCE) and $\text{std}(\cdot)$ is estimated using 10 samples.

Figure 12: GMM training when $p_\theta(x)$ is close to $p_{\theta^*}(x)$. WS outperforms other methods including WW in generative model (top) and inference network (middle) learning. VIMCO has the lowest gradient variance (bottom) but still performs worse than WS and results in worsening of the inference network as number of particles is increased.

Figure 13: Training of sigmoid belief nets. (Left) Training curves: WW learns faster than VIMCO but results in equal or slightly worse end test log likelihood. (Middle) Log evidence values at the end of training: VIMCO is slightly better than WW in low-particle regimes but virtually the same in high-particle regimes. (Right) KL divergence at the end of training: WW results in much lower KL divergence than VIMCO.