Revisiting Reweighted Wake-Sleep
for Models with Stochastic Control Flow

Tuan Anh Le[1*] **Adam R. Kosiorek**[1,2*] **N. Siddharth**[1] **Yee Whye Teh**[2] **Frank Wood**[3]
[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 [3], model structure learning [1], counting [14], attention
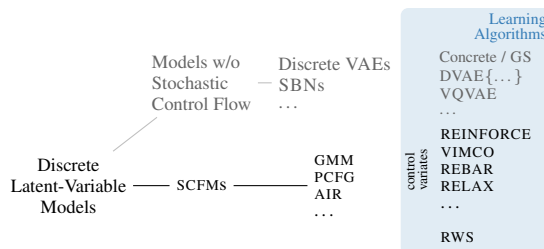
---
* Equal contribution.

Figure 1: An overview of learning algorithms for discrete latent-variable models, with focus on SCFMs.

[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.
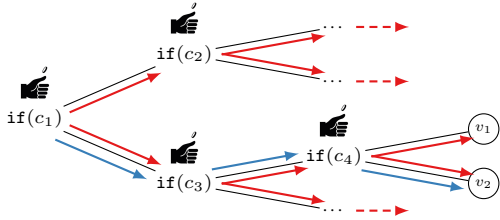
Figure 2: The challenge faced by continuous-relaxation methods on SCFMs—requiring exploration of all branches, in contrast to exploring only one branch at a time. Stochastic control flow proceeds through discrete choices (c_i) yielding values (v_i).

Alternatives to continuous-relaxation methods mainly involve the use of the importance weighted autoencoder (IWAE) [6] framework, employing the REIN-FORCE [59] gradient estimator, combined with control-variate schemes [16, 20, 37, 38, 53] 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 [44]. Secondly, its practical efficacy can still be limited due to high variance and the requirement to design and optimize a separate neural network (c.f. § 4.3).

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) [5]. Comparing extensively with 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 review state-of-the-art methods for learning deep generative models with discrete latent variables (§ 2). We then revisit RWS (§ 3) and present an extensive evaluation of these methods (§ 4) on i) a probabilistic context free grammar (PCFG) model on sentences, ii) the Attend, Infer, Repeat (AIR) model [14] 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 θ and a family of inference networks $q_\phi(z|x)$ parameterized by ϕ. 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 [15, 39] or inference network [32, 43] in isolation.

We begin by reviewing IWAEs [6] 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. [6] introduce the IWAE, maximizing the mean ELBOs over data, $\frac{1}{N} \sum_{n=1}^N \text{ELBO}_{\text{IS}}^K(\theta, \phi, x^{(n)})$, where, for K particles,

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

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

When $K = 1$, this reduces to the variational autoencoder (VAE) [27, 46]. Burda et al. [6] show that $\text{ELBO}_{\text{IS}}^K(\theta, \phi, x)$ 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 [44], as the signal-to-noise ratio of θ's gradient estimator is $O(\sqrt{K})$ whereas ϕ's is $O(1/\sqrt{K})$. Note that although Tucker et al. [54] 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 $\text{ELBO}_{\text{IS}}^K(\theta, \phi, x)$ with respect to θ and ϕ [47]. $\nabla_\theta \text{ELBO}_{\text{IS}}^K(\theta, \phi, x)$ is estimated by evaluating $\nabla_\theta \log \hat{Z}_K$ using samples $z_{1:K} \sim Q_\phi(\cdot|x)$, where $\hat{Z}_K = \frac{1}{K}\sum_{k=1}^K w_k$. $\nabla_\phi \text{ELBO}_{\text{IS}}^K(\theta, \phi, x)$

is estimated similarly for models with reparameterizable latents, discrete (and other non-reparameterizable) latents require the REINFORCE gradient estimator [59]

$$g_{\text{REINFORCE}} = \underbrace{\log \hat{Z}_K \nabla_\phi \log Q_\phi(z_{1:K}|x)}_{①} + \underbrace{\nabla_\phi \log \hat{Z}_K}_{②}. \quad (2)$$

2.2 CONTINUOUS RELAXATIONS AND CONTROL VARIATES

Since the gradient estimator in (2) typically suffers from high variance, mainly due to the effect of 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 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 ① with

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

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

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

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

$$g_{\text{RELAX}}^{①} = \left(\log \hat{Z}_K - c_\rho(\tilde{g}_{1:K}) \right) \nabla_\phi \log Q_\phi(z_{1:K}|x)$$
$$+ \nabla_\phi c_\rho(g_{1:K}) - \nabla_\phi c_\rho(\tilde{g}_{1:K}), \quad (4)$$

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 $\tilde{g}_k$ 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_ρ, parameterized by ρ, 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_ρ, 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 θ and ϕ 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 θ using a *wake-phase θ update* and the inference network parameters ϕ using either a *sleep-* or a *wake-phase ϕ update* (or both).

Wake-phase θ update. Given ϕ, θ is updated using an unbiased estimate of $\nabla_\theta - \left(\frac{1}{N} \sum_{n=1}^{N} \text{ELBO}_{\text{IS}}^K(\theta, \phi, x^{(n)}) \right)$, obtained without reparameterization or control variates, as the sampling distribution $Q_\phi(\cdot|x)$ is independent of θ.[2]

---

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

[2]We assume that the deterministic mappings induced by the parameters θ, ϕ are themselves differentiable, such that they are amenable to gradient-based learning.

Sleep-phase ϕ update. Here, ϕ is updated to minimize the Kullback-Leibler (KL) divergence between the posteriors under the generative model and the inference network, averaged over the data distribution of the current generative model

$$\mathbb{E}_{p_\theta(x)}[D_{\mathrm{KL}}(p_\theta(z|x), q_\phi(z|x))]$$
$$= \mathbb{E}_{p_\theta(z,x)}[\log p_\theta(z|x) - \log q_\phi(z|x)]. \quad (5)$$

Its gradient, $\mathbb{E}_{p_\theta(z,x)}[-\nabla_\phi \log q_\phi(z|x)]$, 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 ϕ update. Here, ϕ is updated to minimize the KL divergence between the posteriors under the generative model and the inference network, averaged over the true data distribution

$$\mathbb{E}_{p(x)}[D_{\mathrm{KL}}(p_\theta(z|x), q_\phi(z|x))]$$
$$= \mathbb{E}_{p(x)}[\mathbb{E}_{p_\theta(z|x)}[\log p_\theta(z|x) - \log q_\phi(z|x)]]. \quad (6)$$

The outer expectation $\mathbb{E}_{p(x)}[\mathbb{E}_{p_\theta(z|x)}[-\nabla_\phi \log q_\phi(z|x)]]$ 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

$$\sum_{k=1}^{K} \frac{w_k}{\sum_{\ell=1}^{K} w_\ell} \left(-\nabla_\phi \log q_\phi(z_k|x)\right), \quad (7)$$

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 ϕ update* and the *sleep-phase ϕ 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 θ targets the same objective as IWAE, the gradient update of ϕ 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 ϕ 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 ϕ update* case and more particles K in

the *wake-phase ϕ update*, we obtain a better estimator of these expected KL divergences. This is in contrast to IWAE, where optimizing $\mathrm{ELBO}_{\mathrm{IS}}^K$ 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 [44].

Moreover, targeting $D_{\mathrm{KL}}(p, q)$ as in RWS can be preferable to targeting $D_{\mathrm{KL}}(q, p)$ 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 [36]. Using the inference network as an importance sampling (IS) proposal requires mean-seeking behavior [42, Theorem 9.2]. Moreover, Chatterjee et al. [8] show that the number of particles required for IS to accurately approximate expectations of the form $\mathbb{E}_{p(z|x)}[f(z)]$ is directly related to $\exp(D_{\mathrm{KL}}(p, q))$.

3.3 CONS OF REWEIGHTED WAKE-SLEEP

While a common criticism of the wake-sleep family of algorithms is the lack of a unifying objective, we have not found any empirical evidence where this is a problem. Perhaps a more relevant criticism is that both the sleep and wake-phase ϕ gradient estimators are biased with respect to $\nabla_\phi \mathbb{E}_{p(x)}[D_{\mathrm{KL}}(p_\theta(z|x), q_\phi(z|x))]$. The bias in the sleep-phase ϕ gradient estimator arises from targeting the expectation under the model rather than the true data distribution, and the bias in the wake-phase ϕ gradient estimator results from estimating the KL divergence using self-normalized IS.

In theory, these biases should not affect the fixed point of optimization (θ^*, ϕ^*) where $p_{\theta^*}(x) = p(x)$ and $q_{\phi^*}(z|x) = p_{\theta^*}(z|x)$. First, if $\theta \to \theta^*$ through the wake-phase θ update, the data distribution bias reduces to zero. Second, although the wake-phase ϕ 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 [10]. Further, this gradient estimator follows the central limit theorem, so its asymptotic variance decreases linearly with K [42, Eq. (9.8)]. Thus, using larger K improves learning of the inference network.

In practice, the families of generative models, inference networks, and the data distributions determine which of the biases are more significant. In most of our findings, the bias of the data distribution appears to be the most detrimental. This is due to the fact that initially $p_\theta(x)$ is quite different from $p(x)$, and hence using sleep-phase

ϕ updates performs worse than using wake-phase ϕ updates. An exception to this is the PCFG experiment (c.f. § 4.1) where the data distribution bias is not as large and inference using self-normalized IS is extremely difficult.

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 outperform 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. [3]

Notationally, the different variants of RWS will be referred to as wake-sleep (WS) and wake-wake (WW). The *wake-phase θ update* is always used. We refer to using it in conjunction with the *sleep-phase ϕ update* as WS and using it in conjunction with the *wake-phase ϕ update* as WW. Using both *wake-* and *sleep-phase ϕ 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 θ* and *ϕ updates* is always specified, and computation between them is matched so a *wake-phase ϕ update* with batch size B implies a *sleep phase ϕ 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.

---

[3] In Appendix D, we include additional experiments on sigmoid belief networks which, however, are not SCFMs.

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 [13, 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 ζ_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, Table 11.2] (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:

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

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: $S \rightarrow NP\,VP \rightarrow NP\,PP\,VP \rightarrow NP\,PP\,PP\,VP \rightarrow \cdots$.

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, $p(x|z) = \exp(-L(x, s(z))^2)$,
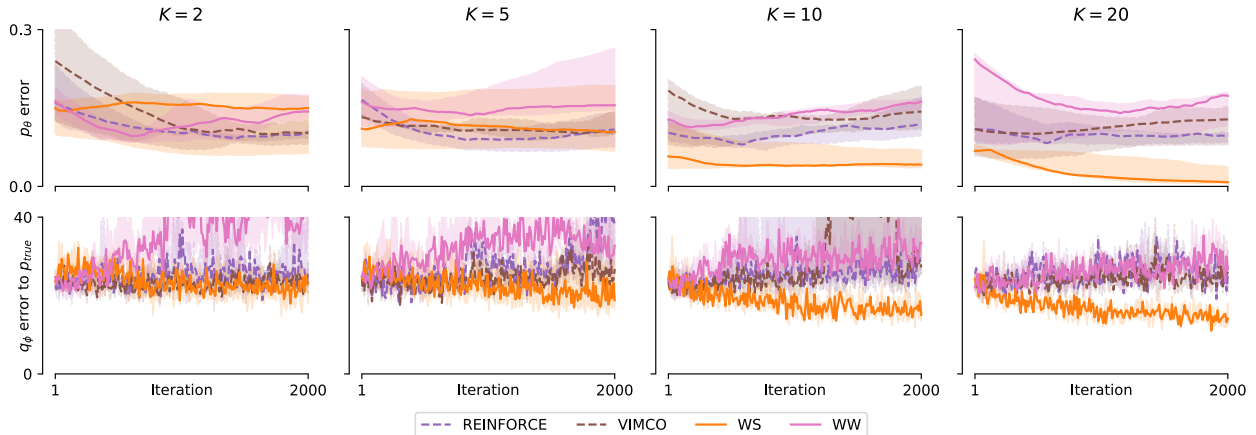
Figure 3: PCFG training. *(Top)* Quality of the generative model: While all methods have the same gradient update for θ, the performance of WS improves and is the best as K is increased. Other methods, including WW, do not yield significantly better model learning as K is increased, since WS's inference network learns the fastest. *(Bottom)* Quality of the inference network: VIMCO and REINFORCE do not improve with increasing K. WS performs best as K is increased, and while WW's performance improves, the improvement is not as significant. This can be attributed to the data-distribution bias being less significant than the bias coming from self-normalized IS (c.f. § 3.3). Median and interquartile ranges from up to 10 repeats shown (see text).

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 [50]. Training sentences are obtained by sampling from the *astronomers* PCFG with the true production probabilities.

We run WW, WS, VIMCO and REINFORCE ten times for $K \in \{2, 5, 10, 20\}$, with batch size $B = 2$, using the Adam optimizer [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, WW, WS, VIMCO and 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 $\mathbb{E}_{p(x)}[D_{\mathrm{KL}}(p(z|x), q_\phi(z|x))]$ which is estimated up to an additive constant (the conditional entropy $H(p(z|x))$) by the sleep-ϕ loss (5) using samples from the true PCFG.

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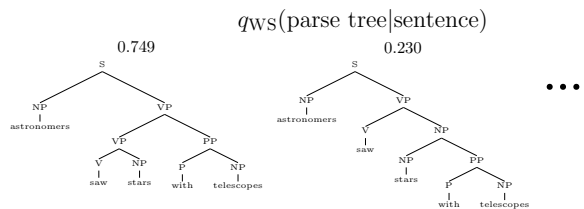

Figure 4: Samples from the inference network trained with WS ($K = 20$). Highest probability samples correspond to correct sentences ($s(z) = x$).

We inspect the quality of the inference network by sampling from it. Figure 4, shows samples from an inference network trained with 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 VIMCO, showing that none of them match the given sentence ($s(z) \neq x$), and whose production probabilities are poor, unlike with RWS.

4.2 ATTEND, INFER, REPEAT

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

We set the maximum number of inference steps in AIR to three and train on 50×50 images with zero, one or
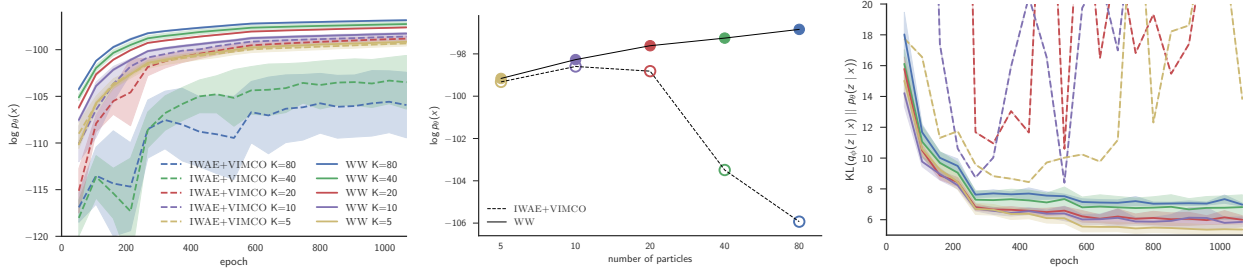
Figure 5: Training of AIR. *(Left)* Training curves: training with VIMCO leads to larger variance in training than WW. *(Middle)* Log evidence values at the end of training: increasing number of particles improves WW monotonically but improves VIMCO only up to a point ($K = 10$ is the best). *(Right)* WW results in significantly lower variance and better inference networks than VIMCO. Note that KL is between the inference network and the *current* generative model.

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 $\mathbf{x} \in [0, 1]^{50 \times 50}$), 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 20×20.

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 $D_{\text{KL}}(q_\phi(z|x), p_\theta(z|x))$ 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

$$p_\theta(z) = \text{Cat}(z|\text{softmax}(\theta)),\ p(x|z) = \mathcal{N}(x|\mu_z, \sigma_z^2),$$
$$q_\phi(z|x) = \text{Cat}(z|\text{softmax}(\eta_\phi(x))),$$

where $z \in \{0, \ldots, C-1\}$, C is the number of clusters and μ_c, σ_c^2 are fixed to $\mu_c = 10c$ and $\sigma_c^2 = 5^2$. The generative model parameters are $\theta \in \mathbb{R}^C$. The inference net-

work consists of a multilayer perceptron $\eta_\phi : \mathbb{R} \to \mathbb{R}^C$, 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 $p_{\theta_{\text{true}}}(x)$ where $\text{softmax}(\theta_{\text{true}})_c = (c+5)/\sum_{i=1}^C (i+5)$ ($c = 0, \ldots, C-1$), 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 RE-INFORCE, 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 E). 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 $c_\rho(g_{1:K}) = \frac{1}{K} \sum_{k=1}^K \text{MLP}_\rho([x, g_k])$, 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 $\|\text{softmax}(\theta) - \text{softmax}(\theta_{\text{true}})\|$. 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 $(x_{\text{test}}^{(m)})_{m=1}^M$ from the true model: $\frac{1}{M} \sum_{m=1}^M \|q_\phi(z|x_{\text{test}}^{(m)}) - p_{\theta_{\text{true}}}(z|x_{\text{test}}^{(m)})\|$.

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
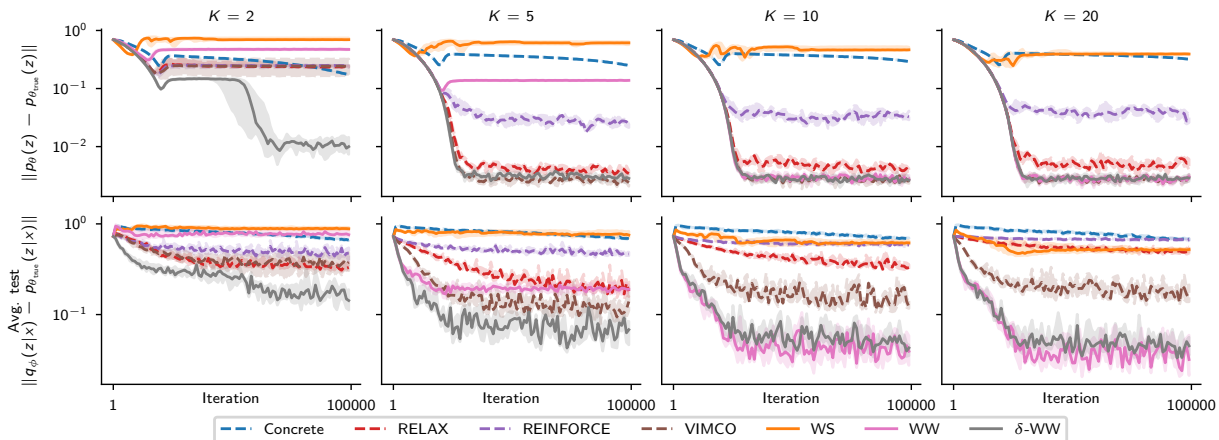
Figure 6: GMM training. Median and interquartile ranges from 10 repeats shown. *(Top)* Quality of the generative model: WS and WW improve with more particles thanks to lower variance and lower bias estimators of the gradient respectively. IWAE methods suffer with a larger particle budget [44]. WS performs the worst as a consequence of computing the expected KL under the model distribution $p_\theta(x)$ (5) instead of the true data distribution $p(x)$ as with WW (6). WW suffers from branch-pruning (see text) in low-particle regimes, but learns the best model fastest in the many-particle regime; δ-WW additionally learns well in the low-particle regime. *(Bottom)* Both inference network and generative model quality develop identically.

using more particles in (7) to estimate the gradient of the wake ϕ objective results in a lower bias. The latter is because using more particles results in the signal-to-noise of IWAE's ϕ gradient estimator to drop at the rate $O(1/\sqrt{K})$ [44].

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 θ gradient estimator (common to all methods), $\nabla_\theta \text{ELBO}_{\text{IS}}^K(\theta, \phi, x)$ 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 θ. In Figure 6, θ 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 ϕ update is highly biased which is supported by WS being worse than WW. On the other hand, if θ 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 θ 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 $\{0, \dots, 9\}$ 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 ($\{0, \dots, 14\}$ and $\{0, \dots, 17\}$ respectively).

While this failure mode is a particular feature of the adverse initialization of θ, we hypothesize that WS and WW suffer from it more, as they alternate between two different objectives for optimizing θ and ϕ. 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 $q_\phi(\cdot|x)$ 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 (Figure 6; Figure 7, right).

We propose a simple extension of WW, denoted δ-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-
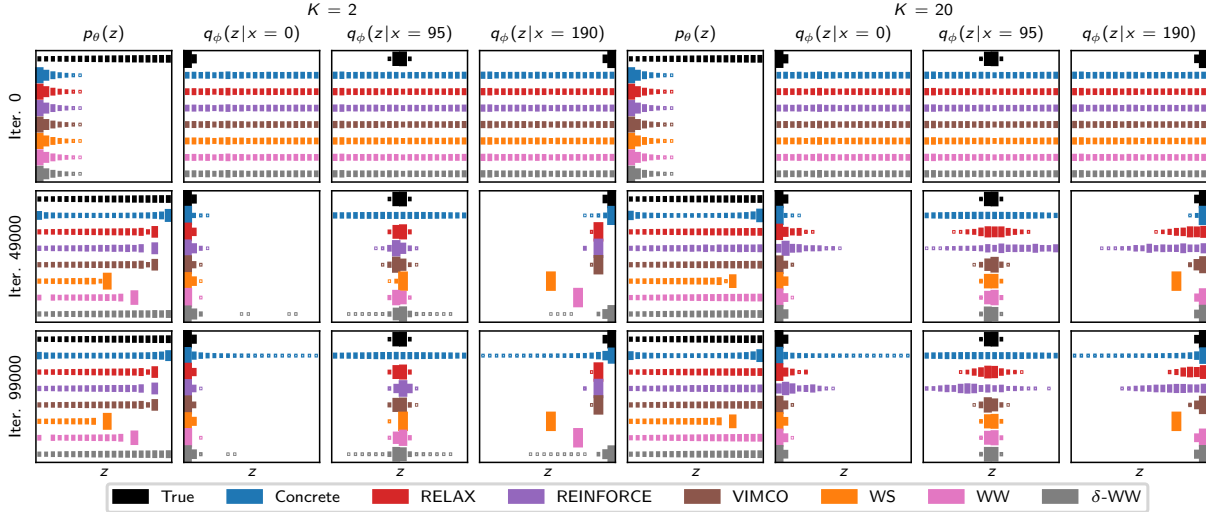
Figure 7: Generative model and inference network during GMM training shown as Hinton diagrams where areas are proportional to probability. Rows correspond to start, middle and end of optimization. *(Left half)* Learning with few particles leads to the branch-pruning (described in text) of the inference network (shown as conditional PMF given different x) and the generative model (first column of each half) for all methods except δ-WW. Concrete distribution fails. *(Right half)* Learning with many particles leads to branch-pruning only for WS; WW and δ-WW succeed where IWAE fails, learning a suboptimal final generative model.

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 with 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.

References

[1] Ryan Adams, Hanna Wallach, and Zoubin Ghahramani. Learning the structure of deep sparse graphical models. In *International Conference on Artificial Intelligence and Statistics*, 2010.

[2] Eli Bingham, Jonathan P Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D Goodman. Pyro: Deep universal probabilistic programming. *The Journal of Machine Learning Research*, 20(1):973–978, 2019.

[3] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[4] Taylor L Booth and Richard A Thompson. Applying probability measures to abstract languages. *IEEE transactions on Computers*, 100(5):442–450, 1973.

[5] Jörg Bornschein and Yoshua Bengio. Reweighted wake-sleep. In *International Conference on Learning Representations*, 2015.

[6] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In *International Conference on Learning Representations*, 2016.

[7] Nick Chater and Christopher D Manning. Probabilistic models of language processing and acquisition. *Trends in cognitive sciences*, 10(7):335–344, 2006.

[8] Sourav Chatterjee, Persi Diaconis, et al. The sample size required in importance sampling. *The Annals of Applied Probability*, 28(2):1099–1135, 2018.

[9] Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750, 2014.

[10] Jie Chen and Ronny Luss. Stochastic gradient descent with biased but consistent gradient estimators. *arXiv preprint arXiv:1807.11880*, 2018.

[11] Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel. The Helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

[12] 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.

[13] Jay Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102, 1970.

[14] S. M. Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Koray Kavukcuoglu, and Geoffrey E. Hinton. Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems*, 2016.

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[16] Will Grathwohl, Dami Choi, Yuhuai Wu, Geoff Roeder, and David Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *International Conference on Learning Representations*, 2018.

[17] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural Turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[18] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471, 2016.

[19] Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. Learning to transduce with unbounded memory. In *Advances in Neural Information Processing Systems*, pages 1828–1836, 2015.

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

[21] Tim Hesterberg. Weighted average importance sampling and defensive mixture distributions. *Technometrics*, 37 (2):185–194, 1995.

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

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

[24] Biing Hwang Juang and Laurence R Rabiner. Hidden markov models for speech recognition. *Technometrics*, 33(3):251–272, 1991.

[25] Charles Kemp, Joshua B Tenenbaum, Thomas L Griffiths, Takeshi Yamada, and Naonori Ueda. Learning systems of concepts with an infinite relational model. 2006.

[26] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

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

[28] Dan Klein and Christopher D Manning. A parsing: fast exact viterbi parse selection. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 40–47. Association for Computational Linguistics, 2003.

[29] Adam R. Kosiorek, Hyunjik Kim, Ingmar Posner, and Yee Whye Teh. Sequential attend, infer, repeat: Generative modelling of moving objects. In *Advances in Neural Information Processing Systems*, 2018.

[30] Brenden M Lake, Neil D Lawrence, and Joshua B Tenenbaum. The emergence of organizing structure in conceptual representation. *Cognitive science*, 2018.

[31] Karim Lari and Steve J Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer speech & language*, 4(1):35–56, 1990.

[32] Tuan Anh Le, Atilim Gunes Baydin, and Frank Wood. Inference compilation and universal probabilistic programming. In *International Conference on Artificial Intelligence and Statistics*, 2017.

[33] Tuan Anh Le, Maximilian Igl, Tom Rainforth, Tom Jin, and Frank Wood. Auto-encoding sequential Monte Carlo. In *International Conference on Learning Representations*, 2018.

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

[35] Christopher D Manning, Christopher D Manning, and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.

[36] Tom Minka. Divergence measures and message passing. Technical report, Technical report, Microsoft Research, 2005.

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

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

[39] Shakir Mohamed and Balaji Lakshminarayanan. Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*, 2016.

[40] Radford M Neal. Connectionist learning of belief networks. *Artificial intelligence*, 56(1):71–113, 1992.

[41] Willie Neiswanger, Frank Wood, and Eric Xing. The dependent Dirichlet process mixture of objects for detection-free tracking and object modeling. In *Artificial Intelligence and Statistics*, pages 660–668, 2014.

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

[43] Brooks Paige and Frank Wood. Inference networks for sequential monte carlo in graphical models. In *International Conference on Machine Learning*, pages 3040–3049, 2016.

[44] 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.

[45] Carl Edward Rasmussen. The infinite Gaussian mixture model. In *Advances in neural information processing systems*, pages 554–560, 2000.

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

[47] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[48] Jason Tyler Rolfe. Discrete variational autoencoders. In *International Conference on Learning Representations*, 2017.

[49] N. Siddharth, Brooks Paige, Jan-Willem van de Meent, Alban Desmaison, Noah D. Goodman, Pushmeet Kohli, Frank Wood, and Philip H. S. Torr. Learning disentangled representations with semi-supervised deep generative models. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NIPS)*, pages 5927–5937. Curran Associates, Inc., December 2017.

[50] Scott A Sisson, Yanan Fan, and Mark Beaumont. *Handbook of Approximate Bayesian Computation*. Chapman and Hall/CRC, 2018.

[51] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[52] Dustin Tran, Matthew D. Hoffman, Rif A. Saurous, Eugene Brevdo, Kevin Murphy, and David M. Blei. Deep probabilistic programming. In *International Conference on Learning Representations*, 2017.

[53] 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*, pages 2624–2633, 2017.

[54] George Tucker, Dieterich Lawson, Shixiang Gu, and Chris J. Maddison. Doubly reparameterized gradient estimators for monte carlo objectives. In *International Conference on Learning Representations*, 2019.

[55] Arash Vahdat, Evgeny Andriyash, and William G. Macready. Dvae#: Discrete variational autoencoders with relaxed boltzmann priors. In *Advances in Neural Information Processing Systems*, 2018.

[56] Arash Vahdat, William G. Macready, Zhengbing Bian, and Amir Khoshaman. Dvae++: Discrete variational autoencoders with overlapping transformations. In *International Conference on Machine Learning*, 2018.

[57] Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, and Frank Wood. An Introduction to Probabilistic Programming. *arXiv e-prints*, art. arXiv:1809.10756, Sep 2018.

[58] Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu. Neural discrete representation learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6306–6315. Curran Associates, Inc., 2017.

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

[60] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

[61] Daniel H Younger. Recognition and parsing of context-free languages in time n3. *Information and control*, 10 (2):189–208, 1967.