Amortized Population Gibbs Samplers with Neural Sufficient Statistics

Hao Wu [1] **Heiko Zimmermann** [1] **Eli Sennesh** [1] **Tuan Anh Le** [2] **Jan-Willem van de Meent** [1]

Abstract

Amortized variational methods have proven difficult to scale to structured problems, such as inferring positions of multiple objects from video images. We develop amortized population Gibbs (APG) samplers, a class of scalable methods that frames structured variational inference as adaptive importance sampling. APG samplers construct high-dimensional proposals by iterating over updates to lower-dimensional blocks of variables. We train each conditional proposal by minimizing the inclusive KL divergence with respect to the conditional posterior. To appropriately account for the size of the input data, we develop a new parameterization in terms of neural sufficient statistics. Experiments show that APG samplers can train highly structured deep generative models in an unsupervised manner, and achieve substantial improvements in inference accuracy relative to standard autoencoding variational methods.

1. Introduction

Many inference tasks involve hierarchical structure. For example, when modeling a corpus of videos that contain moving objects, there are three levels of representation. At the level of the corpus as a whole, we can reason about the distribution over objects and dynamics of motion. At the instance level, i.e. for each video, we can reason the visual features and trajectories of individual objects. Finally, at a data point level, i.e. for each video frame, we can reason about the visibility of objects and their position.

In the absence of supervision, uncovering structure from data requires inductive biases. Deep generative models let us incorporate biases in the form of priors that mirror the structure of the problem domain. By parameterizing conditional distributions with neural networks, we hope to learn models that use corpus-level characteristics (e.g. the appearance of objects and the motion dynamics) to make data-efficient inferences about instance-level variables (e.g. object appearance and positions).

[1]Northeastern University, USA [2]Massachusetts Institute of Technology, USA. Correspondence to: Hao Wu <haowu@ccs.neu.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 108, 2020.

In recent years we have seen applications of amortized variational methods to inference in hierarchical domains, such as object detection (Eslami et al., 2016), modeling of user reviews (Esmaeili et al., 2019), and object tracking (Kosiorek et al., 2018). These approaches build on the framework of variational autoencoders (VAEs) (Kingma & Welling, 2013; Rezende et al., 2014) to train structured deep generative models. However, scaling up these approaches to more complex domains still poses significant challenges. Whereas it is easy to train VAEs on large corpora of data, it is not easy to train structured encoders for models with a large number of (correlated) variables at the instance level.

In this paper, we develop methods for amortized inference that scale to structured models with hundreds of latent variables. Our approach takes inspiration from work by Johnson et al. (2016), which develops methods for efficient inference in deep generative models with conjugate-exponential family priors. In this class of models, we can perform inference using variational expectation maximization (EM) (Beal, 2003; Bishop, 2006; Wainwright & Jordan, 2008), which iterates between closed-form updates to blocks of variables. Variational EM is computationally efficient, often converges in a small number of iterations, and scales to a large number of variables. However, variational EM is also model-specific, difficult to implement, and only applicable to a restricted class of conjugate-exponential models.

To overcome these limitations, we develop a more general approach that frames variational inference as adaptive importance sampling. We propose *amortized population Gibbs* samplers, a class of methods that iterate between conditional proposals to blocks of variables to construct high-quality samples. To train these proposals, we minimize the inclusive KL divergence w.r.t. the conditional posterior to approximate Gibbs updates. We combine these proposals with a sequential Monte Carlo (SMC) sampling strategy to reduce the variance of importance weights, which improves efficiency during training and inference at test time.

Our experiments establish that APG samplers can capture the hierarchical structure in deep generative models in an unsupervised manner. In Gaussian mixture models, where Gibbs updates can be computed in closed form, the learned proposals converge to the conditional posteriors. To evaluate the performance on non-conjugate models, we trained a deep generative mixture model and an unsupervised tracking model and compare APG samplers to reweighted wake-sleep methods, a bootstrapped population sampler, and a HMC-augmented baseline. In all experiments APG samplers can capture the hierarchical characteristics of the problem and outperformed existing methods in terms of training efficiency and inference accuracy.

We summarize the contributions of this paper as follows:

1. We develop APG samplers, a new class of amortized inference methods that employ approximate Gibbs conditionals to iteratively improve samples.
2. We present a novel parameterization using neural sufficient statistics to learn proposals that generalize across input datasets that vary in size.
3. We show that APG samplers can be used to train structured deep generative models with 100s of instance-level variables in an unsupervised manner.
4. We demonstrate substantial improvements in terms of test-time inference efficiency relative to baseline methods.

2. Background

In probabilistic machine learning we commonly define generative models in terms of a joint distribution $p_\theta(x, z)$ over data x and latent variables z. Given a model specification, we are interested in reasoning about the posterior distribution $p_\theta(z \mid x)$ conditioned on data $x \sim p^{\text{DATA}}(x)$ sampled from an (unknown) data distribution. Amortized variational inference methods approximate the posterior $p_\theta(z \mid x)$ with a distribution $q_\phi(z \mid x)$ in some tractable variational family.

A widely used class of deep probabilistic models are variational autoencoders (Kingma & Welling, 2013; Rezende et al., 2014). These models are trained by maximizing the stochastic lower bound (ELBO) in Equation 1 on the log marginal likelihood, which is equivalent to minimizing the exclusive KL divergence $\mathrm{KL}(q_\phi(z|x)||p_\theta(z|x))$.

$$\mathcal{L}(\phi) = \mathbb{E}_{p^{\text{DATA}}(x)\, q_\phi(z|x)} \left[\log \frac{p_\theta(x, z)}{q_\phi(z \mid x)} \right] \tag{1}$$

When the distribution specified by the encoder is reparameterizable, we can compute Monte Carlo estimates of the gradient of this objective using pathwise derivatives. However, reparameterization is not always straightforward or even possible, e.g. in models with discrete variables. Non-reparameterizable cases require likelihood-ratio estimators (Williams, 1992) which can have a high variance. A range of approaches for variance reduction have been put forward, including reparameterized continuous relaxations (Maddison et al., 2017; Jang et al., 2017), credit assignment techniques (Weber et al., 2019), and other control variates (Mnih & Rezende, 2016; Tucker et al., 2017; Grathwohl et al., 2018).

Reweighted wake-sleep (RWS) methods (Bornschein & Bengio, 2014) sidestep the need for reparameterization by minimizing the stochastic upper bound in Equation 2, which is equivalent to minimizing an inclusive KL divergence:

$$\mathcal{U}(\phi) = \mathbb{E}_{p^{\text{DATA}}(x)\, p_\theta(z|x)} \left[\log \frac{p_\theta(x, z)}{q_\phi(z \mid x)} \right]. \tag{2}$$

Here we can compute a self-normalized gradient estimator using samples $z \sim q_\phi(z \mid x)$ and weights $w = \frac{p_\theta(x,z)}{q_\phi(z|x)}$

$$-\nabla_\phi \mathcal{U}(\phi) \simeq \sum_{l=1}^{L} \frac{w^l}{\sum_{l'} w^{l'}} \nabla_\phi \log q_\phi(z^l \mid x). \tag{3}$$

This gradient estimator has a number of advantages over the estimator used in standard VAEs (Le et al., 2019). Notably, it only requires that the *proposal density* is differentiable, whereas reparameterized estimators require that the *sample* itself is differentiable. This is a milder condition that holds for most distributions of interest, including those over discrete variables. Moreover, minimizing the inclusive KL reduces our risk of learning a proposal that collapses to a single mode of a multi-modal posterior (Le et al., 2019).

3. Amortized Population Gibbs Samplers

To generate high-quality samples from the posterior in an incremental manner, we develop methods that are inspired by EM and Gibbs sampling, both of which perform iterative updates to blocks of variables. Concretely, we assume that the latent variables in the generative model decompose into blocks $z = \{z_1, \dots, z_B\}$ and train proposals $q_\phi(z_b \mid x, z_{-b})$ that update the variables in a each block z_b conditioned on variables in the remaining blocks $z_{-b} = z \setminus \{z_b\}$.

Starting from a sample $q_\phi(z^1 \mid x)$ from a standard encoder, we generate a sequence of samples $\{z^1, \dots, z^K\}$ by performing a sweep of conditional updates to each block z_b

$$q_\phi(z^k \mid x, z^{k-1}) = \prod_{b=1}^{B} q_\phi(z_b^k \mid x, z_{\prec b}^k, z_{\succ b}^{k-1}), \tag{4}$$

where $z_{\prec b} = \{z_i \mid i < b\}$ and $z_{\succ b} = \{z_i \mid i > b\}$. Repeatedly applying sweep updates then yields a proposal

$$q_\phi(z^1, \dots, z^K \mid x) = q_\phi(z^1 \mid x) \prod_{k=2}^{K} q_\phi(z^k \mid x, z^{k-1}).$$

We want to train proposals that improve the quality of each sample z^k relative to that of the preceding sample z^{k-1}. There are two possible strategies for accomplishing this.

One is to define an objective that minimizes the discrepancy between the marginal $q_\phi(z^K \mid x)$ for the final sample and the posterior $p_\theta(z^K \mid x)$. This corresponds to learning a sweep update $q_\phi(z^k \mid x, z^{k-1})$ that transforms the initial proposal to the posterior in exactly K sweeps. An example of this type of approach, albeit one that does not employ block updates, is the recent work on annealing variational objectives (Huang et al., 2018).

In this paper, we will pursue a different approach. Instead of transforming the initial proposal in exactly K steps, we learn a sweep update that leaves the target density *invariant*

$$p_\theta(z^k \mid x) = \int p_\theta(z^{k-1} \mid x)\, q_\phi(z^k \mid x, z^{k-1}) dz^{k-1}.$$

When this condition is met, the proposal $q_\phi(z^1, \dots z^K \mid x)$ is a Markov Chain whose stationary distribution is the posterior. This means a sweep update learned at training time can be applied at test time to iteratively improve sample quality, without requiring a pre-specified number of updates K.

When we require that each block update $q_\phi(z'_b \mid x, z_{-b})$ leaves the target density invariant,

$$\begin{aligned} p_\theta(z'_b, z_{-b} \mid x) &= \int p_\theta(z_b, z_{-b} \mid x)\, q_\phi(z'_b, \mid x, z_{-b}) dz_b \\ &= p_\theta(z_{-b} \mid x)\, q_\phi(z'_b \mid x, z_{-b}), \end{aligned} \tag{5}$$

then the each update must equal the exact conditional posterior $q_\phi(z'_b \mid x, z_{-b}) = p_\theta(z'_b \mid x, z_{-b})$. In other words, when the condition in Equation 5 is met, the proposal $q_\phi(z^1, \dots z^K \mid x)$ is a Gibbs sampler.

3.1. Variational Objective

To learn each of the block proposals $q_\phi(z_b \mid x, z_{-b})$ we will minimize the inclusive KL divergence $\mathcal{K}_b(\phi)$.

$$\mathbb{E}_{\hat{p}(x) p_\theta(z_{-b} \mid x)} \Big[\mathrm{KL}\left(p_\theta(z_b \mid x, z_{-b}) \,||\, q_\phi(z_b \mid x, z_{-b})\right) \Big]$$

Unfortunately, this objective is intractable, since we are not able to evaluate the density of the true marginal $p_\theta(z_{-b} \mid x)$, nor that of the conditional $p_\theta(z_b \mid z_{-b}, x)$. As we will discuss in Section 6, this has implications for the evaluation of the learned proposals, since we cannot compute a lower or upper bound on the log marginal likelihood as in other variational methods. However, it is nonetheless possible to approximate the gradient of the objective

$$-\nabla_\phi \mathcal{K}_b(\phi) = \mathbb{E}_{\hat{p}(x)\, p_\theta(z_b, z_{-b} \mid x)} \left[\nabla_\phi \log q_\phi(z_b \mid x, z_{-b}) \right].$$

We can estimate this gradient using any Monte Carlo method that generates samples $z \sim p_\theta(x \mid z)$ from the posterior. In the next section, we will use the learned proposals to define an importance sampler, which we will use to obtain a set of weighted samples $\{(z_b^l, z_{-b}^l)\}_{l=1}^L$ to compute a self-normalized gradient estimate

$$-\nabla_\phi \mathcal{K}_b(\phi) \simeq \sum_{l=1}^{L} \frac{w^l}{\sum_{l'} w^{l'}} \nabla_\phi \log q_\phi(z_b^l \mid x, z_{-b}^l). \tag{6}$$

When we also want to learn a generative model $p_\theta(x, z)$, we can compute a similar self-normalized gradient estimate

$$\begin{aligned} \nabla_\theta \log p_\theta(x) &= \mathbb{E}_{p_\theta(z|x)} \left[\nabla_\theta \log p_\theta(x, z) \right] \\ &\simeq \sum_{l=1}^{L} \frac{w^l}{\sum_{l'} w^{l'}} \nabla_\theta \log p_\theta(x, z^l) \end{aligned} \tag{7}$$

to maximize the marginal likelihood. The identity in Equation 7 holds due to the identity $\mathbb{E}_{p_\theta(z|x)}[\nabla_\theta \log p_\theta(z|x)] = 0$ (see Appendix A for details).

In the following, we will describe an adaptive importance sampling scheme which generates high quality samples to accurately estimate gradients for the conditional proposal $q_\phi(z_b|x, z_{-b})$ and an initial one-shot encoder $q_\phi(z|x)$.

3.2. Generating High Quality Samples

A well-known limitation of importance sampling is that the weights have high variance for high-dimensional and/or correlated variables, which are typical in models with hierarchical structure. When weighted samples poorly approximate the posterior, the gradient estimates in Equations 6 and 7 will have high variance, leading to slow convergence.

To address these challenges, we propose to generate samples using a sequential Monte Carlo sampler (Del Moral et al., 2006). SMC methods (Doucet et al., 2001) reduce the variance of importance weights by decomposing a high-dimensional sampling problem into a sequence of lower-dimensional problems. They do so by combining two basic ideas: (1) sequential importance sampling, which decomposes a proposal for a sequence of variables into a sequence of conditional proposals, (2) resampling, which selects partial proposals with probability proportional to their weights to improve the overall sample quality. Most commonly, SMC methods are used in the context of state space models to generate proposals for a sequence of variables by proposing one variable at a time. SMC *samplers* (see Algorithm 1) are a subclass of SMC methods that interleave resampling with the application of a transition kernel.

The distinction between SMC methods for state space models and SMC samplers is subtle but important. Whereas the former generate proposals for a sequence of variables $z_{1:t}$ by proposing $z_t \sim q(z_t \mid z_{1:t-1})$ to *extend* the sample space at each iteration, SMC samplers can be understood as an importance sampling analogue to Markov chain Monte Carlo (MCMC) methods (Brooks et al., 2011), which construct a

Algorithm 1 Sequential Monte Carlo sampler

```
Input Model γ^{1:K}, Proposals q^{1:K}, Number of particles L
for l = 1 to L do
    z^{1,l} ~ q^1(·)                                   ▷ Propose
    w^{1,l} = γ^1(z^{1,l}) / q^1(z^{1,l})              ▷ Weigh
end for
for k = 2 to K do
    z^{k-1,1:L}, w^{k-1,1:L} = RESAMPLE(z^{k-1,1:L}, w^{k-1,1:L})
    for l = 1 to L do
        z^{k,l} ~ q^k(· | z^{k-1,l})                   ▷ Propose
        w^{k,l} = [γ^k(z^{k,l}) r^{k-1}(z^{k-1,l} | z^{k,l})] / [γ^{k-1}(z^{k-1,l}) q^k(z^{k,l} | z^{k-1,l})] w^{k-1,l}   ▷ Weigh
    end for
end for
Output Weighted Samples {z^{K,l}, w^{K,l}}_{l=1}^L
```

Algorithm 2 Amortized Population Gibbs Sampler

```
Input Data x, Model p_θ(x, z)
      Proposals q_φ(z|x), {q_φ(z_b|x, z_{-b})}_{b=1}^B
g_φ = 0, g_θ = 0                                       ▷ Initialize gradient to 0
for l = 1 to L do                                      ▷ Initial proposal
    z^{1,l} ~ q_φ(· | x)                               ▷ Propose
    w^{1,l} = p_θ(x, z^{1,l}) / q_φ(z^{1,l} | x)       ▷ Weigh
end for
g_φ = g_φ + Σ_{l=1}^L [w^{1,l} / Σ_{l'=1}^L w^{1,l'}] ∇_φ log q_φ(z^{1,l} | x)
g_θ = g_θ + Σ_{l=1}^L [w^{1,l} / Σ_{l'=1}^L w^{1,l'}] ∇_θ log p_θ(x, z^{1,l})
for k = 2 to K do                                      ▷ Gibbs sweeps
    z̃^{1:L}, w̃^{1:L} = z^{k-1,1:L}, w^{k-1,1:L}
    for b = 1 to B do                                  ▷ Block updates
        z̃^{1:L}, w̃^{1:L} = RESAMPLE(z̃^{1:L}, w̃^{1:L})
        for l = 1 to L do
            z̃'_b^l ~ q_φ(· | x, z̃_{-b}^l)              ▷ Propose
            w̃^l = [p_θ(x, z̃'_b^l, z̃_{-b}^l) q_φ(z̃_b^l | x, z̃_{-b}^l)] / [p_θ(x, z̃_b^l, z̃_{-b}^l) q_φ(z̃'_b^l | x, z̃_{-b}^l)] w̃^l   ▷ Weigh
            z̃_b^l = z̃'_b^l                             ▷ Reassign
        end for
        g_φ = g_φ + Σ_{l=1}^L [w̃^l / Σ_{l'=1}^L w̃^{l'}] ∇_φ log q_φ(z̃_b^l | x, z̃_{-b}^l)
        g_θ = g_θ + Σ_{l=1}^L [w̃^l / Σ_{l'=1}^L w̃^{l'}] ∇_θ log p_θ(x, z̃^l)
    end for
    z^{k, 1:L}, w^{k, 1:L} = z̃^{1:L}, w̃^{1:L}
end for
Output Gradient g_φ, g_θ, Weighted Samples {z^{K,l}, w^{K,l}}_{l=1}^L
```

Markov chain $z^{1:k}$ by generating a proposal $q(z^k \mid z^{k-1})$ from a transition kernel at each iteration.

To understand how approximate Gibbs proposals can be used within a SMC sampler, we will first define a sequential importance sampler (SIS), which decomposes the importance weight into a sequence of *incremental* weights. SIS considers a sequence of unnormalized target densities $\gamma^1(z^1), \gamma^2(z^{1:2}), \ldots, \gamma^K(z^{1:K})$. If we now consider an initial proposal $q^1(z^1)$, along with a sequence of conditional proposals $q^k(z^k \mid z^{1:k-1})$, then we can recursively construct a sequence of weights $w^k = v^k w^{k-1}$ by assuming $w^1 = \gamma^1(z^1)/q^1(z^1)$ and defining the incremental weight

$$v^k = \frac{\gamma^k(z^{1:k})}{\gamma^{k-1}(z^{1:k-1})q^k(z^k \mid z^{1:k-1})}.$$

This construction ensures that, at step k in the sequence, we have a weight w^k relative to the intermediate density $\gamma^k(z^{1:k})$ of the form (see Appendix B)

$$w^k = \frac{\gamma^k(z^{1:k})}{q^1(z^1)\prod_{k'=2}^{k} q^{k'}(z^{k'} \mid z^{1:k'-1})}.$$

In the following we will consider a sequence of intermediate target densities defined by a *reverse kernel* $r(z' \mid z)$:

$$\gamma^k(z^{1:k}) = p_\theta(x, z^k) \prod_{k'=2}^{k} r(z^{k'-1} \mid z^{k'}).$$

This defines a density on an *extended space* which admits the marginal density

$$\gamma^k(z^k) = \int \gamma^k(z^{1:k})\, dz^{1:k-1} = p_\theta(x, z^k).$$

This means that at each step k, we can treat the preceding samples z^{k-1} as *auxiliary variables*; if we generate a proposal $z^{1:k}$ and simply disregard $z^{1:k-1}$, then the pair (w^k, z^k) is a valid importance sample relative to $p_\theta(z^k \mid x)$. If we additionally condition proposals on x, the incremental weight for this particular choice of target densities is

$$v^k = \frac{p_\theta(x, z^k)\, r(z^{k-1} \mid x, z^k)}{p_\theta(x, z^{k-1})\, q(z^k \mid x, z^{k-1})}. \tag{8}$$

This construction is valid for any choice of *forward kernel* $q(z^k \mid z^{k-1})$ and reverse kernel $r(z^{k-1} \mid z^k)$. For a given forward kernel the *optimal* reverse kernel is given by

$$r(z^{k-1} \mid x, z^k) = \frac{p_\theta(x, z^{k-1})}{p_\theta(x, z^k)} q(z^k \mid x, z^{k-1}).$$

For this choice of kernel, the incremental weights are 1, which minimizes the variance of the weights w^k.

We propose to use the approximate Gibbs kernel $q_\phi(z^k | x, z^{k-1})$ as both the forward and the reverse kernel. When the approximate Gibbs kernel converges to the actual Gibbs kernel, this choice becomes optimal, since in limit the kernel satisfies detailed balance

$$p_\theta(x, z^k)\, q_\phi(z^{k-1} \,|\, x, z^k) = p_\theta(x, z^{k-1})\, q_\phi(z^k \,|\, x, z^{k-1}).$$

In general, the weights w^k in SIS will have a high variance when z is high-dimensional or correlated. Moreover, we sample these variables k times. When the approximate

Gibbs kernel converges to the true kernel, this will not increase the variance of weights (since $v^k = 1$ in limit). But during training variance of weights w^k will increase with k, since we are jointly sampling an entire Markov chain.

To overcome this problem, SMC samplers interleave applications of the transition kernel with a *resampling* steps (see Appendix D for Algorithm), which generates a new set of equally weighted samples by selecting samples from the current sample set with probability proportional to their weight. APG samplers employ resampling after each block update $z_b \sim q_\phi(z_b|x, z_{-b})$, which results in equal incoming weights in the subsequent block update. This allows us to compute the gradient estimate of block proposals based on the contribution of the incremental weight

$$v = \frac{p_\theta(x, z'_b, z_{-b})\, q_\phi(z_b \mid x, z_{-b})}{p_\theta(x, z_b, z_{-b})\, q_\phi(z'_b \mid x, z_{-b})}.$$

These incremental weights will generally have much lower variance and thus greatly improves the sample-efficiency for both, inference and gradient estimation, in models with high-dimensional correlated variables.

We refer to this implementation of a SMC sampler as an *amortized population Gibbs* sampler, and summarize all the steps of the computation in Algorithm 2. In Appendix E, we prove that this algorithm is correct using an argument based on proper weighting.

4. Neural Sufficient Statistics

Gibbs sampling strategies that sample from exact conditionals rely on conjugacy relationships. We assume a prior and likelihood that can both be expressed as exponential families

$$p(x \mid z) = h(x) \exp\{\eta(z)^\top T(x) - \log A(\eta(z))\},$$
$$p(z) = h(z) \exp\{\lambda^\top T(z) - \log A(\lambda)\}.$$

where $h(\cdot)$ is a base measure, $T(\cdot)$ is a vector of sufficient statistics, and $A(\cdot)$ is a log normalizer. The two densities are jointly conjugate when $T(z) = (\eta(z), -\log A(\eta(z)))$. In this case, the posterior distribution lies in the same exponential family as the prior. Typically, the prior $p(z \mid \lambda)$ and likelihood $p(x \mid z)$ are not jointly conjugate, but it is possible to identify conjugacy relationships at the level of individual blocks of variables,

$$p(z_b \mid z_{-b}, x) \propto h(z_b) \exp\big\{(\lambda_{b,1} + T(x, z_{-b}))^\top T(z_b) - (\lambda_{b,2}+1) \log A(\eta(z_b))\big\}.$$

In general, these conjugacy relationships will not hold. However, we can still take inspiration to design variational distributions that make use of conditional independencies in a model. We assume that each of the approximate Gibbs updates $q_\phi(z_b \mid x, z_{-b})$ is an exponential family, whose parameters are computed from a vector of prior parameters λ and a vector of neural sufficient statistics $T_\phi(x, z_{-b})$

$$q_\phi(z_b \mid x, z_{-b}) = p(z_b \mid \lambda + T_\phi(x, z_{-b})).$$

This parameterization has a number of desirable properties. Exponential families are the largest-entropy distributions that match the moments defined by the sufficient statistics (see e.g. Wainwright & Jordan (2008)), which is helpful when minimizing the inclusive KL divergence. In exponential families it is also more straightforward to control the entropy of the variational distribution. In particular, we can initialize $T_\phi(x, z_{-b})$ to output values close to zero in order to ensure that we initially propose from a prior and/or regularize $T_\phi(x, z_{-b})$ to help avoid local optima.

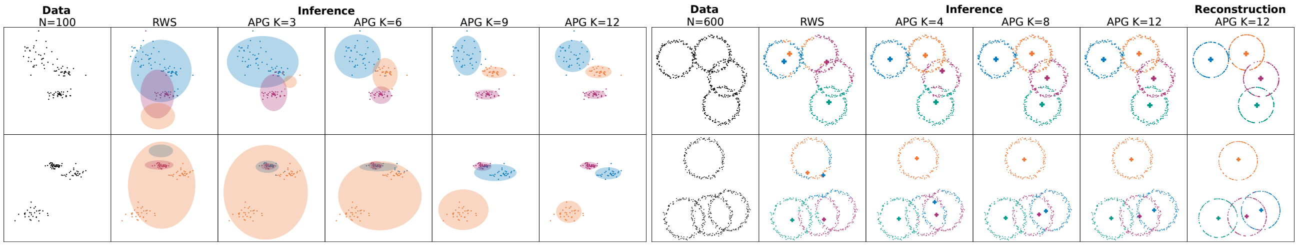
A useful case arises when the data $x = \{x_1, \ldots, x_N\}$ are independent conditioned on z. In this setting, it is often possible to partition the latent variables $z = \{z^{\mathrm{G}}, z^{\mathrm{L}}\}$ into global (instance-level) and local (datapoint-level) variables z^{G} and local variables z^{L}. The dimensionality of global variables is typically constant, whereas local variables $z^{\mathrm{L}} = \{z^{\mathrm{L}}_1, \ldots, z^{\mathrm{L}}_N\}$ have a dimensionality that increases with the data N. For models with this structure, the local variables are typically conditionally independent $z^{\mathrm{L}}_n \perp z^{\mathrm{L}}_{-n} \mid x, z^{\mathrm{G}}$, which means that we can parameterize the sufficient statistics as

$$\tilde{\lambda}^{\mathrm{G}} = \lambda^{\mathrm{G}} + \sum_{n=1}^{N} T^{\mathrm{G}}_\phi(x_n, z^{\mathrm{L}}_n), \quad \tilde{\lambda}^{\mathrm{L}}_n = \lambda^{\mathrm{L}}_n + T^{\mathrm{L}}_\phi(x_n, z^{\mathrm{G}}).$$

The advantage of this parameterization is it allows us to train approximate Gibbs updates for global variables in a manner that scales dynamically with the size of the dataset, and appropriately adjusts the posterior variance according to the amount of available data.

5. Related Work

Our work fits into a line of recent methods for deep generative modeling that seek to improve inference quality, either by introducing auxiliary variables (Maaløe et al., 2016; Ranganath et al., 2016), or by performing iterative updates (Marino et al., 2018). Work by Hoffman (2017) applies Hamiltonian Monte Carlo to samples that are generated from the encoder, which serves to improve the gradient estimate w.r.t. θ (Equation 7), while learning the inference network using a standard reparameterized ELBO objective. Li et al. (2017) also use MCMC to improve the quality of samples from an encoder, but additionally use these samples to train the encoder by minimizing the inclusive KL divergence relative to the filtering distribution of the Markov chain. As in our work, the filtering distribution after multiple MCMC steps is intractable. Li et al. (2017) therefore use an adversarial objective to minimize the inclusive KL.



(a) GMM

(b) DMM

Figure 1. Visualization of a single sample for (**a**) GMM instances with $N = 100$ data points and (**b**) DMM instances with $N = 600$ data points. The left columns show the data instance, the second column the inference result from RWS, and the subsequent columns the inference result after K APG sweeps. In addition the rightmost column in (**b**) shows the reconstruction from the generative model. Unlike RWS, APG successfully learns the model parameters and an inference model and its performance improves with increasing K.

Neither of these methods consider block decompositions of latent variables, nor do they learn kernels.

Salimans et al. (2015) derive a (stochastic) lower bound for variational inference which uses an importance weight defined in terms of forward and reverse kernels similar to Equation 8. Huang et al. (2018) extend the work by Salimans et al. (2015) by learning a sequence of transition kernels that performs annealing from the initial encoder to the posterior. Since both these methods minimize an exclusive KL, rather than an inclusive KL, the gradient estimates rely on reparameterization, which makes them inapplicable to models with discrete variables. Moreover, both methods perform a joint update on all variables, while we consider the task of learning conditional proposals.

Wang et al. (2018) develop a meta-learning approach to learn Gibbs block conditionals. This work assumes a setup in which it is possible to sample x, z from the true generative model $p(x, z)$. This circumvents the need for self-normalized estimators in Equation 6, which are necessary when we additionally wish to learn the generative model. Similar to our work, the approach by Wang et al. (2018) minimizes the inclusive KL, but uses the learned conditionals to define an (approximate) MCMC sampler, rather than using them as proposals in a SMC sampler. This work also has a somewhat different focus from ours, in that it primarily seeks to learn block conditionals that have the potential to generalize to previously unseen graphical models.

6. Experiments

We evaluate APG samplers on three different tasks. We begin by analysing APG samplers on a Gaussian mixture model (GMM) as an exemplar of a model in the conjugate-exponential family. This experiment allows us to analytically compute the conditional posterior to verify if the learned proposals indeed converge to the Gibbs kernels. Here APG samplers outperform other iterative inference methods even when using a smaller computational budget. Next we consider a deep generative mixture model (DMM) that incorporates a neural likelihood and use APG samplers to jointly train the generative and inference model. We demonstrate the scalability of APG samplers by performing accurate inference at test-time on instances with 600 points. In our third experiment, we consider an unsupervised tracking model for multiple bouncing MNIST digits. We extend the task proposed by Srivastava et al. (2015) to up to five digits, and learn both a deep generative model for videos and an inference model that performs tracking. At test time we show that APG easily scales beyond previously reported results for a specialized recurrent architecture (Kosiorek et al., 2018). The results for each of these tasks constitute significant advances relative to the state of the art. APG samplers are not only able to scale to models with higher complexity, but also provides a general framework for performing inference in models with global and local variables, which can be adapted to a variety of model classes with comparative ease.

6.1. Baselines

We compare our APG sampler with three baseline methods. The first is RWS. The second is a bootstrapped population Gibbs (BPG) sampler, which uses exactly the same sampling scheme as in Algorithm 2, but proposes from the prior rather than from approximate Gibbs kernels. This serves to evaluate whether learning proposals improves the quality of inference results. The third is a method that augments RWS with Hamiltonian Monte Carlo (HMC) updates, which is analogous to the method proposed by Hoffman (2017). This serves to compare Gibbs updates to those that can be obtained with state-of-the-art MCMC methods.

In the HMC-RWS baseline, we employ the standard RWS gradient estimators from Equations 3 and 7, but update samples from the standard encoder using HMC. In the bouncing MNIST model, all variables are continuous and are updated jointly. In the GMM, we use HMC to sample from the marginal for continuous variables (by marginalizing over cluster assignments via direct summation), and then sample discrete variables from enumerated Gibbs conditionals to evaluate the full log joint $\log p_\theta(x, z)$. For the DMM we perform the same procedure, but use HMC to sample from the conditional distribution given discrete variables, because direct summation is not possible in this model.

Unless otherwise stated, we compare methods under an equivalent computational budget. We compare an APG sampler that performs K sweeps with L particles to RWS with $K \cdot L$ particles. When comparing to HMC-RWS methods, we perform K updates for L particles with a number of leapfrog steps LF in {1,5,10}, which corresponds to a 1x, 5x, and 10x computational budget as measured in terms of the number of evaluations of the log joint. We resort to report the log joint $\log p_\theta(x, z)$ as an evaluation metric, because the marginal $q_\phi(z^k|x)$ is intractable. A summary of these results can be found in Table 1.

6.2. Gaussian Mixture Model

To evaluate whether APG samplers can learn the exact Gibbs updates in conditionally conjugate models, we consider a Gaussian mixture model

$$\mu_i, \tau_i \sim \text{Normal-Gamma}(\mu_0 I, \nu_0 I, \alpha_0 I, \beta_0 I),\ i = 1, 2.., I,$$
$$c_n \sim \text{Cat}(\pi),\ x_n | c_n = i \sim \text{N}(\mu_i, 1/\tau_i),\ n = 1, 2, .., N.$$

In this model, the global variables $z^{\text{G}} = \{\mu_{1:I}, \tau_{1:I}\}$ are the mean and precision for each mixture component, whereas the local variables are the cluster assignments $z^{\text{L}} = \{c_{1:N}\}$. Conditioned on cluster assignments, the Gaussian likelihood is conjugate to a normal-gamma prior with sufficient statistics $\left\{ \text{I}[c_n = i],\ \text{I}[c_n = i]\, x_n,\ \text{I}[c_n = i]\, x_n^2 \;\middle|\; i = 1, 2, \ldots, I \right\}$, where $\text{I}[z_n = i]$ evaluates to 1 if the equality holds, and 0 otherwise.

We employ a variational distribution that iterates over $q_\phi(c \mid x, \mu, \tau)$ and $q_\phi(\mu, \tau \mid x, c)$, using pointwise neural sufficient statistics modeled after the ones in the analytical updates. We train our model on 20,000 GMM instances, each containing $I = 3$ clusters and $N = 60$ data points, with $K = 5$ sweeps, $L = 10$ particles, 20 instances per batch, learning rate 2.5×10^{-4}, and 2×10^5 gradient steps.

Figure 1a shows single samples in four test instance, each containing $N = 100$ points. Even when using a parameterization that employs neural sufficient statistics, the RWS encoder fails to propose reasonable clusters, whereas the APG sampler converges within 12 sweeps in GMMs with more variables than training instances.

We compare the APG samplers with true Gibbs samplers and the baselines in section 6.1. Figure 2 shows that the APG sampler converges to the $p_\theta(x, z)$ achieved the true Gibbs sampler; it outperforms HMC-RWS even when we run it with 10 times the computation budget; and learned

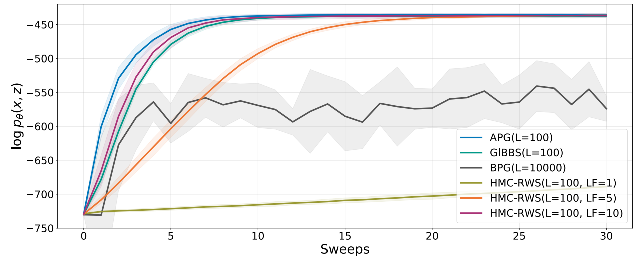


Figure 2. $\log p_\theta(x, z)$ as a function of number of sweeps for a GMM test instance. We perform each method with L particles. For HMC-RWS, we perform multiple leapfrog steps LF in $\{1, 5, 10\}$. To achieve results comparable to APG and Gibbs samplers, HMC-RWS needs 10 times the computational budget, while BPG underperforms even with 100 times the computational budget.

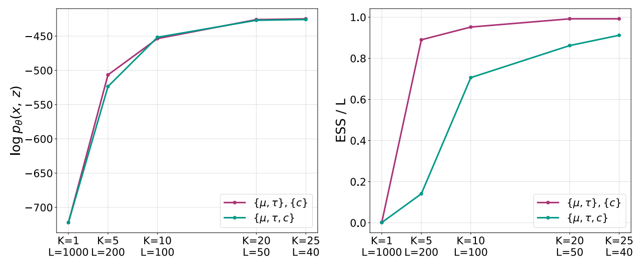


Figure 3. $\log p(x, z)$ (left) and ESS (right) as a function of number of sweeps for APG samplers with different block update strategies and same overall sampling budget $K \cdot L = 1000$. A higher ESS is achieved by performing more sweeps with fewer samples.

proposals substantially improve on inference since BPG does not converge even with 100x computational budget. In addition, we verify the approximate Gibbs kernel converges to the true Gibbs kernel (see Appendix G).

Moreover, we compare joint block updates $\{\mu, \tau, c\}$ with decomposed blocks $\{\mu, \tau\}, \{c\}$, for varying number of sweeps (see Figure 3). We additionally assess sample quality using the effective sample size (ESS)

$$\frac{\text{ESS}}{L} = \frac{(\sum_{l=1}^{L} w^{k,l})^2}{L \sum_{l=1}^{L} (w^{k,l})^2}.$$

We can see that it is more effective to perform more sweeps K with a smaller number of particles L for a fixed computation budget. We also see that the decomposed block updates result in higher sample quality.

6.3. Deep Generative Mixture Model

In the deep generative mixture model $p_\theta(x, z)$ our data is a corpus of ring-shaped mixtures. The generative model takes the form

$$\mu_i \sim \text{N}(0, \sigma_0^2 I), i = 1, 2, ..., I$$
$$c_n \sim \text{Cat}(\pi), \alpha_n \sim \text{Beta}(1, 1), n = 1, 2, ..., N$$
$$x_n | c_n = i \sim \text{N}(g_\theta(\alpha_n) + \mu_i, \sigma_\epsilon^2 I).$$

Table 1. $\log p_\theta(x, z)$ averaged over test corpus for RWS, BPG, HMC-RWS, and Gibbs Sampling for different numbers of sweeps K. We run HMC-RWS with 1, 5, and 10 leapfrog integration steps, which corresponds to 1, 5, and 10 times the computational budget used by APG with the same number of sweeps. The GMM test corpus contains $20,000$ instances, each of which has $N = 100$ data points. The DMM test corpus contains $20,000$ instances with $N = 600$ data points per instance. Bouncing MNIST's test corpus contains $10,000$ instances each containing $T = 100$ time steps and $D = 5$ digits. In all tasks APG with $K = 20$ performs the best.

	RWS	BPG K=20	HMC-RWS K=20, LF=1	HMC-RWS K=20, LF=5	HMC-RWS K=20, LF=10	APG K=5	APG K=10	APG K=20	GIBBS K=20
GMM ($\times 10^2$)	-7.651	-6.688	-6.594	-5.420	-4.673	-4.816	-4.556	**-4.469**	-4.573
DMM ($\times 10^3$)	-3.172	-4.204	-2.185	-2.184	-2.184	-2.050	-2.040	**-2.035**	–
BC-MNIST ($\times 10^5$)	-1.673	-1.884	-1.538	-1.405	-1.303	-0.706	-0.652	**-0.620**	–

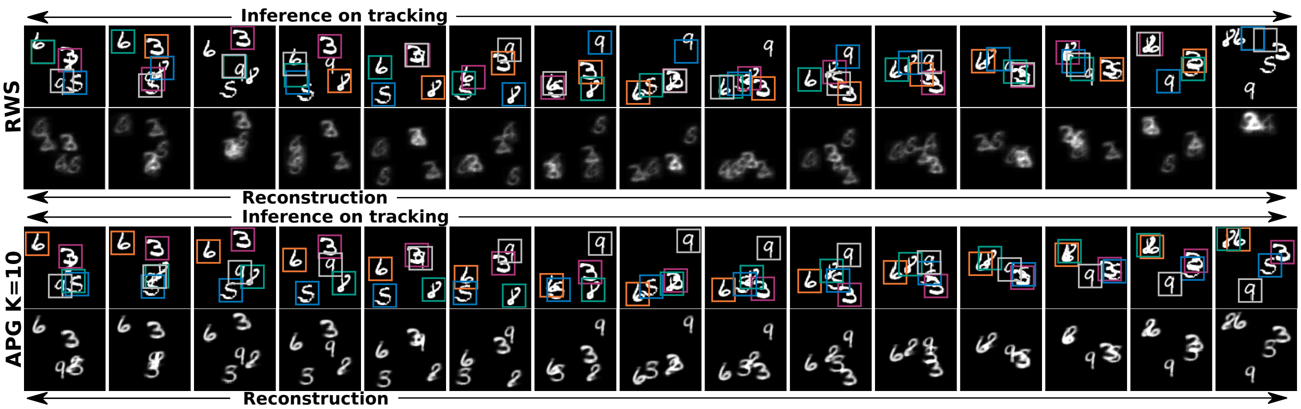


Figure 4. A single sample from variational distribution in one video instance with $T = 100$ time steps and $D = 5$ digits. The visualization is truncated to the first 15 time steps due to limited space (see Appendix I for full time series and Appendix H for more examples). The sample is initialized from RWS (top) and updated by $K = 10$ APG sweeps (bottom). The APG sampler improves the inference results in tracking and learns better MNIST digit embeddings that improve reconstruction.

Here μ_i is the center of ith ring. Given a cluster assignment c_n and an embedding α_n which goes into a MLP decoder g_θ, we sample a data point x_n with 2D Gaussian noise.

We employ a variational distribution that iterates over $q_\phi(c, \alpha|x, \mu)$ and $q_\phi(\mu|x, c, \alpha)$. We train on 20K DMM instances, each containing $I = 4$ clusters and $N = 200$ points, with $K = 8$ sweeps, $L = 10$ particles, 20 instances per batch, learning rate 10^{-4}, and 3×10^5 gradient steps.

Once again, we compare the APG sampler with the encoders using RWS. Figure 1b shows visualization of single samples in four test instances, each containing $N = 600$ data points. The APG sampler scales to a large range of numbers of variables, whereas a standard encoder trained using RWS fails to produce reasonable proposals.

6.4. Time Series Model – Bouncing MNIST

In the bouncing MNIST model, our data is a corpus of video frames that contain multiple moving MNIST digits. To demonstrate the scalability of APG samplers, we train both a deep generative model and an inference model with $600k$ video instances, each of which contains $T = 10$ time steps and $D = 3$ digits. We then evaluate the model on test instances that contain $T = 100$ time steps and $D = 5$ digits. Our generative model has the form

$$z_d^{\text{what}} \sim \mathrm{N}(0, I),\; z_{d,1}^{\text{where}} \sim \mathrm{N}(0, I),\; d = 1, 2, .., D,$$
$$z_{d,t}^{\text{where}} \sim \mathrm{Norm}(z_{d,t-1}^{\text{where}}, \sigma_0^2 I),\; t = 1, 2, .., T,$$
$$x_t \sim \mathrm{Bern}\Big(\sigma\Big(\sum_d \mathrm{ST}\big(\mu_\theta(z_d^{\text{what}}), z_{d,t}^{\text{where}}\big)\Big)\Big)$$

where $z_{1:D}^{\text{what}}$ are the latent code of digits, and $z_{1:D,1:T}^{\text{where}}$ are time-dependent positions. ST is a spatial transformer (Jaderberg et al., 2015) that maps the output of a MLP decoder μ_θ, i.e. logits for a 28×28 MNIST image, onto a $96{\times}96$ canvas based on $z_{d,t}^{\text{where}}$. We employ a block update that decomposes the variables into $T + 1$ blocks $(z_{1:D}^{\text{what}}, z_{1:D,1}^{\text{where}}, z_{1:D,2}^{\text{where}}, \ldots, z_{1:D,T}^{\text{where}})$. We train our model on 60000 bouncing MNIST instances, each containing $T = 10$ time steps and $D = 3$ digits, with $K = 5$ sweeps, $L = 10$ particles, $K = 5$ instances per batch, learning rate 10^{-4}, and 1.2×10^7 gradient steps (see Appendix F for architecture).

Figure 4 shows that the APG sampler substantially improves inference and reconstruction results over RWS encoders,

Table 2. Bouncing MNIST performance. Mean-squared error between test instance and its reconstruction across 5000 video instances with different number of time steps T and digits D. We use a fixed sampling budget of $K \cdot L = 1000$. APG samplers achieve substantial improvements relative to RWS and HMC-RWS.

	RWS	HMC-RWS K=20, LF=10	APG K=10	APG K=20
D=5, T=20	265.1	261.2	117.7	**103.8**
D=5, T=100	272.1	267.9	115.4	**102.3**
D=3, T=20	134.0	124.1	60.4	**58.9**
D=3, T=100	137.3	127.4	56.0	**55.7**

and scales to test instances with hundreds of latent variables. Moreover we compute the mean squared error between the video instances and their reconstructions. Table 2 shows that APG sampler can substantially improve the sample from RWS encoder, and scales up to 100 time steps and 5 digits.

7. Conclusion

We introduce APG samplers, a framework for amortized variational inference based on adaptive importance samplers that iterates between updates to blocks of variables. To appropriately account for the size of the input data we developed a novel parameterization in terms of neural sufficient statistics inspired by sufficient statistics in conjugate exponential families. We show that APG samplers can train structured deep generative models with hundreds of instance-level variables in an unsupervised manner, and compare favorably to existing amortized inference methods in terms of computational efficiency.

8. Acknowledgements

This work was supported by the Intel Corporation, NSF award 1835309, the DARPA LwLL program, and startup funds from Northeastern University. Tuan Anh Le was supported by AFOSR award FA9550-18-S-0003.

References

Beal, M. J. *Variational Algorithms for Approximate Bayesian Inference*. PhD thesis, 2003.

Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer, New York, 2006. ISBN 978-0-387-31073-2.

Bornschein, J. and Bengio, Y. Reweighted wake-sleep. *arXiv preprint arXiv:1406.2751*, 2014.

Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. *Handbook of markov chain monte carlo*. CRC press, 2011.

Del Moral, P., Doucet, A., and Jasra, A. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, June 2006. doi: 10.1111/j.1467-9868.2006.00553.x.

Doucet, A., Freitas, N., and Gordon, N. (eds.). *Sequential Monte Carlo Methods in Practice*. Springer New York, New York, NY, 2001. ISBN 978-1-4419-2887-0 978-1-4757-3437-9. doi: 10.1007/978-1-4757-3437-9.

Eslami, S. M. A., Heess, N., Weber, T., Tassa, Y., Szepesvari, D., kavukcuoglu, k., and Hinton, G. E. Attend, Infer, Repeat: Fast Scene Understanding with Generative Models. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29*, pp. 3225–3233. Curran Associates, Inc., 2016.

Esmaeili, B., Huang, H., Wallace, B., and van de Meent, J.-W. Structured Neural Topic Models for Reviews. In *Artificial Intelligence and Statistics*, pp. 3429–3439, April 2019. 00000.

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

Hoffman, M. D. Learning deep latent gaussian models with markov chain monte carlo. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1510–1519. JMLR. org, 2017.

Huang, C.-W., Tan, S., Lacoste, A., and Courville, A. C. Improving explorability in variational inference with annealed variational objectives. In *Advances in Neural Information Processing Systems*, pp. 9701–9711, 2018.

Jaderberg, M., Simonyan, K., Zisserman, A., et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pp. 2017–2025, 2015.

Jang, E., Gu, S., and Poole, B. Categorical reparametrization with gumble-softmax. In *International Conference on Learning Representations 2017*. OpenReviews. net, 2017.

Johnson, M., Duvenaud, D. K., Wiltschko, A., Adams, R. P., and Datta, S. R. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in Neural Information Processing Systems*, pp. 2946–2954, 2016.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *International Conference on Learning Representations*, 2013.

Kosiorek, A., Kim, H., Teh, Y. W., and Posner, I. Sequential Attend, Infer, Repeat: Generative Modelling of Moving Objects. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 8606–8616. Curran Associates, Inc., 2018.

Le, T. A., Kosiorek, A. R., Siddharth, N., Teh, Y. W., and Wood, F. Revisiting reweighted wake-sleep for models with stochastic control flow. In *Uncertainty in Artificial Intelligence*, 2019. Le and Kosiorek contributed equally.

Li, Y., Turner, R. E., and Liu, Q. Approximate inference with amortised mcmc. *arXiv preprint arXiv:1702.08343*, 2017.

Maaløe, L., Sønderby, C. K., Sønderby, S. K., and Winther, O. Auxiliary deep generative models. In *International Conference on Machine Learning*, pp. 1445–1453, 2016.

Maddison, C. J., Mnih, A., and Teh, Y. W. The concrete distribution: A continuous relaxation of discrete random variables. *International Conference on Learning Representations*, 2017.

Marino, J., Yue, Y., and Mandt, S. Iterative amortized inference. *International Conference on Machine Learning*, 2018.

Mnih, A. and Rezende, D. Variational Inference for Monte Carlo Objectives. In *International Conference on Machine Learning*, pp. 2188–2196, June 2016.

Naesseth, C., Lindsten, F., and Schon, T. Nested sequential monte carlo methods. In *International Conference on Machine Learning*, pp. 1292–1301, 2015.

Ranganath, R., Tran, D., and Blei, D. Hierarchical variational models. In *International Conference on Machine Learning*, pp. 324–333, 2016.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In Xing, E. P. and Jebara, T. (eds.), *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 1278–1286, Bejing, China, June 2014. PMLR.

Salimans, T., Kingma, D., and Welling, M. Markov chain monte carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, pp. 1218–1226, 2015.

Srivastava, N., Mansimov, E., and Salakhudinov, R. Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pp. 843–852, 2015.

Tucker, G., Mnih, A., Maddison, C. J., Lawson, J., and Sohl-Dickstein, J. REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models. In *Advances in Neural Information Processing Systems*, pp. 2624–2633, 2017.

Wainwright, M. J. and Jordan, M. I. Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305, 2008. doi: 10.1561/2200000001.

Wang, T., Wu, Y., Moore, D., and Russell, S. J. Meta-learning mcmc proposals. In *Advances in Neural Information Processing Systems*, pp. 4146–4156, 2018.

Weber, T., Heess, N., Buesing, L., and Silver, D. Credit Assignment Techniques in Stochastic Computation Graphs. *arXiv:1901.01761 [cs, stat]*, January 2019.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992. ISSN 0885-6125. doi: 10.1007/BF00992696.

Supplementary Material: Amortized Population Gibbs Samplers with Neural Sufficient Statistics

A. Gradient of the generative model

We show that the gradient of the marginal $\nabla_\theta \ \log \ p_\theta(x)$ can be estimated using self-normalized importance sampling. First of all, we express the expected gradient of the log joint as

$$\mathbb{E}_{p_\theta(z|x)}\left[\nabla_\theta \log p_\theta(x,z)\right] = \mathbb{E}_{p_\theta(z|x)}\left[\nabla_\theta \log p_\theta(x) + \nabla_\theta \log p_\theta(z|x)\right] = \mathbb{E}_{p_\theta(z|x)}\left[\nabla_\theta \log p_\theta(x)\right] = \nabla_\theta \log p_\theta(x)$$

Here we make use of a standard identity that is also used in likelihood-ratio estimators

$$\mathbb{E}_{p_\theta(z|x)}\left[\nabla_\theta \log p_\theta(z|x)\right] = \int p_\theta(z|x) \nabla_\theta \log p_\theta(z|x)\, dz = \int \nabla_\theta p_\theta(z|x)\, dz = \nabla_\theta \int p_\theta(z|x)\, dz = \nabla_\theta 1 = 0$$

Therefore, we have the the following equality

$$\nabla_\theta \log p_\theta(x) = \mathbb{E}_{p_\theta(z|x)}\left[\nabla_\theta \log p_\theta(x,z)\right] \simeq \sum_{l=1}^{L} \frac{w^l}{\sum_{l'} w^{l'}} \nabla_\theta \log p_\theta(x, z^l).$$

which is the self-normalized gradient estimator in Equation 7.

B. Importance weights in sequential importance sampling

We will prove the form of importance weight w^k in sequential importance sampling. At step $k = 1$, we use exactly the standard importance sampler, thus it is obvious that the following is a valid importance weight

$$w^1 = \frac{\gamma^1(z^1)}{q^1(z^1)}.$$

When step $k > 2$, we are going to prove that the importance weight relative to the intermediate densities has the form

$$w^k = \frac{\gamma^k(z^{1:k})}{q^1(z^1) \prod_{k'=2}^{k} q^{k'}(z^{k'} \mid z^{1:k'-1})}. \tag{9}$$

At step $k = 2$, the importance weight is defined as

$$w^k = v^2 \ w^1 \ = \frac{\gamma^2(z^{1:2})}{\gamma^1(z^1)\, q^2(z^2 \mid z^1)} \frac{\gamma^1(z^1)}{q^1(z^1)} \ = \frac{\gamma^2(z^{1:2})}{q^1(z^1)\, q^2(z^2 \mid z^1)}.$$

which is exactly that form. Now we prove weights in future steps by induction. At step $k \geq 2$, we assume that the weight has the form in Equation 9, i.e.

$$w^k = \frac{\gamma^k(z^{1:k})}{q^1(z^1) \prod_{k'=2}^{k} q^{k'}(z^{k'} \mid z^{1:k'-1})}.$$

then at step $k + 1$, the importance weight is the product of incremental weight and incoming weight

$$w^{k+1} = v^{k+1} \ w^k = \frac{\gamma^{k+1}(z^{1:k+1})}{\gamma^k(z^{1:k})\, q^{k+1}(z^{k+1} \mid z^{1:k})} \frac{\gamma^k(z^{1:k})}{q^1(z^1) \prod_{k'=2}^{k} q^{k'}(z^{k'} \mid z^{1:k'-1})} = \frac{\gamma^{k+1}(z^{1:k+1})}{q^1(z^1) \prod_{k'=2}^{k+1} q^{k'}(z^{k'} \mid z^{1:k'-1})}.$$

Thus the importance weight w^k has the form of Equation 9 at each step $k > 2$ in sequential importance sampling.

C. Derivation of Posterior Invariance

We consider a *sweep* of conditional proposals at step K as

$$p_\theta\big(z^k \mid x, z^{k-1}\big) = \prod_{b=1}^{B} p_\theta(z_b^k \mid x, z_{\prec b}^k, z_{\succ b}^{k-1}), \tag{10}$$

where $z_{\prec b} = \{z_i \mid i < b\}$ and $z_{\succ b} = \{z_i \mid i > b\}$. Additionally we define $z_{\preceq b} = \{z_i \mid i \leq b\}$.

We will show that any partial update within a sweep, i.e.

$$p_\theta\big(z_{\preceq b}^k \mid x, z^{k-1}\big) = \prod_{v=1}^{b} p_\theta(z_v^k \mid x, z_{\prec v}^k, z_{\succ v}^{k-1}), \quad \forall b \in \{1, 2, ..., B\} \tag{11}$$

will leave the posterior invariant. In fact, for any choice of b we have

$$\begin{aligned}\int p_\theta(z^{k-1} \mid x)\, p_\theta(z_{\preceq b}^k \mid x,\, z^{k-1})\, dz_{\preceq b}^{k-1} &= \int p_\theta(z_{\preceq b}^{k-1}, z_{\succ b}^{k-1} \mid x)\, dz_{\preceq b}^{k-1} \prod_{v=1}^{b} p_\theta(z_v^k \mid x, z_{\prec v}^k, z_{\succ v}^{k-1}) \\ &= p_\theta(z_{\succ b}^{k-1} \mid x)\, p_\theta(z_{\preceq b}^k \mid x, z_{\succ b}^{k-1}) \\ &= p_\theta(z_{\preceq b}^k, z_{\succ b}^{k-1} \mid x).\end{aligned}$$

When we require the APG proposal $q_\phi(z_b' \mid x, z_{-b})$ leaves the posterior invariant (by minimizing the inclusive KL divergence relative to the conditional posterior $p_\theta(z_b \mid x, z_{-b})$), then any sweep or part of one sweep will also leave the posterior invariant, as what we prove above. This means that at test time we can apply arbitrary number of APG sweeps, each of which will results in samples that approximate the posterior $p_\theta(z \mid x)$.

D. Resampling Algorithm

Algorithm 3 Multinomial Resampler

1: **Input** Weighted samples $\{z^l, w^l\}_{l=1}^L$
2: **for** $i = 1$ **to** L **do**
3: $\quad a^i \sim \text{Discrete}(\{w^l / \sum_{l'=1}^L w^{l'}\}_{l=1}^L)$ ▷ Index Selection
4: $\quad$ Set Set $\tilde{z}^{\,i} = z^{a^i}$
5: $\quad$ Set $\tilde{w}^{\,i} = \frac{1}{L}\sum_{l=1}^L w^l$ ▷ Re-weigh
6: **end for**
7: **Output** Equally weighted samples $\{\tilde{z}^l, \tilde{w}^l\}_{l=1}^L$

E. Proof of the amortized population Gibbs algorithm

Here, we provide an alternative proof of correctness of the APG algorithm given in Algorithm 2, based on the construction of proper weights (Naesseth et al., 2015) which was introduced after SMC samplers (Del Moral et al., 2006). We first introduce proper weights, and then present several operations that preserve the proper weighting property and finally we apply these properties in proving correctness of APG.

E.1. Proper weights

Definition 1 (Proper weights)**.** Given an unnormalized density $\tilde{p}(z)$, with corresponding normalizing constant $Z_p := \int \tilde{p}(z)\, \mathrm{d}z$ and normalized density $p \equiv \tilde{p}/Z_p$, the random variables $z, w \sim P(z, w)$ are properly weighted with respect to $\tilde{p}(z)$ if and only if for any measurable function f

$$\mathbb{E}_{P(z,w)}\left[w f(z)\right] = Z_p \mathbb{E}_{p(z)}[f(z)]. \tag{12}$$

We will also denote this as

$$z, w \overset{\text{p.w.}}{\sim} \tilde{p}.$$

Using proper weights. Given independent samples $z^l, w^l \sim P$, we can estimate Z_p by setting $f \equiv 1$:

$$Z_p \approx \frac{1}{L} \sum_{l=1}^{L} w^l.$$

This estimator is unbiased because it is a Monte Carlo estimator of the left hand side of (12). We can also estimate $\mathbb{E}_{p(z)}[f(z)]$ as

$$\mathbb{E}_{p(z)}[f(z)] \approx \frac{\frac{1}{L}\sum_{l=1}^{L} w^l f(z^l)}{\frac{1}{L}\sum_{l=1}^{L} w^l}.$$

While the numerator and the denominator are unbiased estimators of $Z_p \mathbb{E}_{p(z)}[f(z)]$ and Z_p respectively, their fraction is biased. We often write this estimator as

$$\mathbb{E}_{p(z)}[f(z)] \approx \sum_{l=1}^{L} \bar{w}^l f(z^l), \tag{13}$$

where $\bar{w}^l := w^l / \sum_{l'=1}^{L} w^{l'}$ is the normalized weight.

E.2. Operations that preserve proper weights

Proposition 1 (Nested importance sampling)**.** *This is similar to Algorithm 1 in (Naesseth et al., 2015). Given unnormalized densities $\tilde{q}(z), \tilde{p}(z)$ with the normalizing constants Z_q, Z_p and normalized densities $q(z), p(z)$, if*

$$z, w \overset{\text{p.w.}}{\sim} \tilde{q}, \tag{14}$$

then

$$z, \frac{w\tilde{p}(z)}{\tilde{q}(z)} \overset{\text{p.w.}}{\sim} \tilde{p}.$$

Proof. First define the distribution of z, w as Q. For measurable $f(z)$

$$\mathbb{E}_{Q(z,w)}\left[\frac{w\tilde{p}(z)}{\tilde{q}(z)} f(z)\right] = Z_q \mathbb{E}_{q(z)}\left[\frac{\tilde{p}(z) f(z)}{\tilde{q}(z)}\right] = Z_q \int q(z) \frac{\tilde{p}(z) f(z)}{\tilde{q}(z)} \,\mathrm{d}z = \int \tilde{p}(z) f(z) \,\mathrm{d}z = Z_p \mathbb{E}_{p(z)}[f(z)].$$

□

Proposition 2 (Resampling)**.** *This is similar to Section 3.1 in (Naesseth et al., 2015). Given an unnormalized density $\tilde{p}(z)$ (normalizing constant Z_p, normalized density $p(z)$), if we have a set of properly weighted samples*

$$z^l, w^l \overset{\text{p.w.}}{\sim} \tilde{p}, \quad l = 1, \dots, L \tag{15}$$

then the resampling operation preserves the proper weighting, i.e.

$$z'^{\,l}, w'^{\,l} \overset{\text{p.w.}}{\sim} \tilde{p}, \quad l = 1, \dots, L$$

where $z'^{\,l} = z^a$ with probability $P(a = i) = w^i / \sum_{l=1}^{L} w^l$ and $w'^{\,l} := \frac{1}{L}\sum_{l=1}^{L} w^l$.

Proof. Define the distribution of z^l, w^l as $\hat{P}$. We show that for any f, $\mathbb{E}[f(z^a)w'^{\,l}] = Z_p \mathbb{E}_{p(z)}[f(z)]$.

$$
\begin{aligned}
&\mathbb{E}_{(\prod_{l=1}^{L} \hat{P}(z^l,w^l))p(a|w^{1:L})}\left[f(z^a)w'^{l}\right]\\
&= \mathbb{E}_{\prod_{l=1}^{L} \hat{P}(z^l,w^l)}\left[\sum_{i=1}^{L} f(z^i)w' \, P(a=i)\right]\\
&= \mathbb{E}_{\prod_{l=1}^{L} \hat{P}(z^l,w^l)}\left[\sum_{i=1}^{L} f(z^i)w' \frac{w^i}{\sum_{l'=1}^{L} w^{l'}}\right]\\
&= \mathbb{E}_{\prod_{l=1}^{L} \hat{P}(z^l,w^l)}\left[\frac{1}{L}\sum_{i=1}^{L} f(z^i)w^i\right]\\
&= \frac{1}{L}\sum_{i=1}^{L} \mathbb{E}_{\hat{P}(z^i,w^i)}\left[f(z^i)w^i\right] = \frac{1}{L}\sum_{i=1}^{L} Z_p \mathbb{E}_{p(z)}[f(z)] = Z_p \, \mathbb{E}_{p(z)}[f(z)].
\end{aligned}
$$

□

Therefore, the resampling will return a new set of samples that are still properly weighted relative to the target distribution in the APG sampler (Algorithm 2).

Proposition 3 (Move)**.** *Given an unnormalized density $\tilde{p}(z)$ (normalizing constant Z_p, normalized density $p(z)$) and normalized conditional densities $q(z'|z)$ and $r(z|z')$, the proper weighting is preserved if we apply the transition kernel to a properly weighted sample, i.e. if we have*

$$z^l, w^l \overset{p.w.}{\sim} \tilde{p}, \tag{16}$$

$$z'^{\,l} \sim q(z'^{\,l}|z^l), \tag{17}$$

$$w'^{\,l} = \frac{\tilde{p}(z'^{\,l})r(z^l|z'^{\,l})}{\tilde{p}(z^l)q(z'^{\,l}|z^l)} w^l, \qquad l = 1, \ldots, L \tag{18}$$

then we have

$$z'^{\,l}, w'^{\,l} \overset{p.w.}{\sim} \tilde{p}, \qquad l = 1, \ldots, L \tag{19}$$

Proof. Firstly we simplify the notation by dropping the superscript l without loss of generality. Define the distribution of z, w as $\hat{P}$. Then, due to (16), for any measurable $f(z)$, we have

$$\mathbb{E}_P[wf(z)] = Z_p E_p[f(z)].$$

To prove (19), we show $\mathbb{E}_{\hat{P}(z,w)q(z'|z)}[w'f(z')] = Z_p \mathbb{E}_{p(z')}[f(z')]$ for any f as follows:

$$
\begin{aligned}
\mathbb{E}_{\hat{P}(z,w)q(z'|z)}[w'f(z')] &= \mathbb{E}_{\hat{P}(z,w)q(z'|z)}\left[\frac{\tilde{p}(z')r(z|z')}{\tilde{p}(z)q(z'|z)} wf(z')\right]\\
&= \int \hat{P}(z,w)q(z'|z)\frac{\tilde{p}(z')r(z|z')}{\tilde{p}(z)q(z'|z)} wf(z') \, \mathrm{d}z \, \mathrm{d}w \, \mathrm{d}z'\\
&= \int \hat{P}(z,w)\frac{\tilde{p}(z')r(z|z')}{\tilde{p}(z)} wf(z') \, \mathrm{d}z \, \mathrm{d}w \, \mathrm{d}z'\\
&= \int \tilde{p}(z')f(z') \left(\int \hat{P}(z,w)w\frac{r(z|z')}{\tilde{p}(z)} \, \mathrm{d}z \, \mathrm{d}w\right) \mathrm{d}z'\\
&= \int \tilde{p}(z')f(z')Z_p \mathbb{E}_{p(z)}\left[\frac{r(z|z')}{\tilde{p}(z)}\right] \mathrm{d}z'.
\end{aligned} \tag{20}
$$

Using the fact that $\mathbb{E}_{p(z)}\left[\frac{r(z|z')}{\tilde{p}(z)}\right] = \int p(z)\frac{r(z|z')}{\tilde{p}(z)}\,\mathrm{d}z = \int r(z|z')\,\mathrm{d}z/Z_p = 1/Z_p$. Equation 20 simplifies to

$$\int \tilde{p}(z')f(z')\,\mathrm{d}z' = Z_p\mathbb{E}_{p(z')}[f(z')].$$

□

E.3. Correctness of APG Sampler

We provide the proof by performing 2 steps in the APG sampler (Algorithm 2), i.e, we prove the correctness when we initialize samples at step $k = 1$ (line 4 - line 9) and then do one Gibbs sweep at step $k = 2$ (line 11 - line 22). In fact, its correctness still holds if we perform more Gibbs sweeps by induction.

Step $k = 1$. We initialize the proposal $z \sim q_\phi(z|x)$ (line 5) and train that encoder using the wake-ϕ phase objective in the standard reweighted wake-sleep(Le et al., 2019) $\mathbb{E}_{p(x)}\left[\mathrm{KL}\left(p_\theta(z|x)||q_\phi(z|x)\right)\right]$. Then we estimate its gradient w.r.t. parameter ϕ (line 8) as

$$\begin{aligned} g_\phi &:= -\nabla_\phi\, \mathbb{E}_{p(x)}\left[\mathrm{KL}\left(p_\theta(z|x)||q_\phi(z|x)\right)\right] && (21)\\ &= \mathbb{E}_{p(x)}\left[\mathbb{E}_{p_\theta(z|x)}\left[\nabla_\phi \log q_\phi(z\mid x)\right]\right]. && (22)\end{aligned}$$

Step $k = 2$. After one full sweep, we have the following objective

$$\mathbb{E}_{p(x)}\Big[\sum_{b=1}^{B}\mathbb{E}_{p_\theta(z_{-b}|x)}\left[\mathrm{KL}\left(p_\theta(z_b\mid z_{-b},x)||q_\phi(z_b\mid x,z_{-b})\right)\right]\Big]$$

And we will prove that we correctly estimate the following gradient w.r.t. parameter ϕ at each block update (line 19)

$$\begin{aligned} g_\phi^b &:= -\nabla_\phi\mathbb{E}_{p(x)}\left[\mathbb{E}_{p_\theta(z_{-b}|x)}\left[\mathrm{KL}\left(p_\theta(z_b|z_{-b},x)||q_\phi(z_b|z_{-b},x)\right)\right]\right] && (23)\\ &= \mathbb{E}_{p(x)}\left[\mathbb{E}_{p_\theta(z_{1:B}|x)}\left[\nabla_\phi \log q_\phi(z_b|z_{-b},x)\right]\right], \qquad b = 1,\ldots,B. && (24)\end{aligned}$$

At each step, as long as we show that samples are properly weighted

$$z_{1:B}^l, w^l \overset{\text{p.w.}}{\sim} p_\theta(z_{1:B},x), \qquad l = 1,\ldots,L. \tag{25}$$

Equation 13 will guarantee the validity of both gradient estimations (line 8 and line 19).

At step $k = 1$, samples are properly weighted because z^l and w^l are proposed using importance sampling (line 6) where $q_\phi(z|x)$ is the proposal density and $p_\theta(z^l, x)$ is the unnormalized target density. The resampling step (line 13) will preserve the proper weighting because of Proposition 2.

To prove that Gibbs sweep (line 11 - line 22) in the APG sampler also preserves proper weighting, we show that each block update satisfies all the 3 conditions (Equation 16, 18 and 25) in Proposition 3, by which we can conclude the samples are still properly weighted after each block update. Without loss of generality, we drop all l superscripts in the rest of the proof. Before we start any block update (before line 15), we already know that samples are properly weighted, i.e.

$$z, w \overset{\text{p.w.}}{\sim} p_\theta(z,x). \tag{26}$$

which corresponds to Equation 16. Next we define a conditional distribution $q(z'\mid z) := q_\phi(z_b'|x,z_{-b})\delta_{z_{-b}}(z_{-b}')$, from which we propose a new sample

$$z' \sim q_\phi(z_b'|x,z_{-b})\delta_{z_{-b}}(z_{-b}'), \tag{27}$$

where the density of z_{-b}' is a delta mass on z_{-b} defined as $\delta_{z_{-b}}(z_{-b}') = 1$ if $z_{-b} = z_{-b}'$ and 0 otherwise. In fact, this sampling step is equivalent to firstly sampling $z_b' \sim q_\phi(\cdot|x,z_{-b})$ (line 15) and let $z_{-b}' = z_{-b}$, which is exactly what the APG sampler assumes procedurally. This condition corresponds to Equation 17.

Finally, we define the weight w'

$$w' = \frac{p_\theta(x, z'_b, z'_{-b}) r(z_b | x, z_{-b}) \delta_{z_{-b}}(z_{-b})}{p_\theta(x, z_b, z_{-b}) q_\phi(z'_b | x, z_{-b}) \delta_{z_{-b}}(z'_{-b})} w, \tag{28}$$

where the terms in blue are treated as densities (normalized or unnormalized) of $z'_{1:B}$ and the terms in red are treated as densities of $z_{1:B}$. Since both delta mass densities evaluate to one, this weight is equal to the weight computed after each block update (line 16). This condition corresponds to Equation 18.

Now we can apply the conclusion (19) in Proposition 3 and claim

$$z'_{1:B}, w' \overset{\text{p.w.}}{\sim} p_\theta(z'_{1:B}, x).$$

since $z_{-b} = z'_{-b}$ and $z_b = z'_b$ due to the re-assignment (line 17). Based on the fact that proper weighting is preserves at both initial step $k = 1$ and the Gibbs sweep $k = 2$, we have proved that both gradient estimations (line 8 and line 19) are correct.

F. Architecture of the Amortized Population Gibbs samplers

Here are the architectures of the model in the 3 tasks: **GMM:**

Layer	$q_\phi(\mu, \tau \| x)$	
Input	Concat$[x_n \in \mathbb{R}^2]$	
1	FC 2	FC 3 Softmax

Layer	$q_\phi(\mu, \tau \| x, c)$	
Input	Concat$[x_n \in \mathbb{R}^2, c_n \in \mathbb{R}^3]$	
1	FC 2	FC 3 Softmax

Layer	$q_\phi(c \| x, \mu, \tau)$
Input	Concat$[x_n \in \mathbb{R}^2, \mu_i \in \mathbb{R}^2]$
1	FC 32 Tanh
2	FC 1, Intermediate Variable $o_i \in \mathbb{R}$
3	Concat$[o_i \in \mathbb{R}]$, Softmax (c_n)

DMM:

Layer	$q_\phi(\mu \| x)$	
Input	$x_n \in \mathbb{R}^2$	
1	FC 32 Tanh	FC 32 Tanh
2	FC 8, $v_n \in \mathbb{R}^8$	FC 4 Softmax, $\gamma_n \in \mathbb{R}^4$
3	$T^n_\phi := \gamma_n \otimes v_n \in \mathbb{R}^{4\times 8}$	
4	Concat$[\text{sum}(T^n_\phi, \text{dim} = 0), \mu_0 \in \mathbb{R}^2, \text{Diag}(\sigma_0^2 I) \in \mathbb{R}^2]$	
5	FC 2×32 Tanh	
6	FC 2×8 $(\mu_{1:4})$	

Layer	$q_\phi(\mu \vert x, c, \alpha)$	
Input	$\text{Concat}[x_n \in \mathbb{R}^2, c_n \in \mathbb{R}^4, \alpha_n \in \mathbb{R}^1]$	
1	FC 32 Tanh	FC 32 Tanh
2	FC 8, $v_n \in \mathbb{R}^8$	FC 4 Softmax, $\gamma_n \in \mathbb{R}^4$
3	$T_\phi^n := \gamma_n \otimes v_n \in \mathbb{R}^{4\times 8}$	
4	$\text{Concat}[\text{sum}(T_\phi^n, \text{dim}=0), \mu_0 \in \mathbb{R}^2, \text{Diag}(\sigma_0^2 I) \in \mathbb{R}^2]$	
5	FC 2×32 Tanh	
6	FC 2×8 ($\mu_{1:4}$)	

Layer	$q_\phi(c, \alpha \vert x, \mu)$
Input	$i = 1, 2, 3, 4,\ x_n - \mu_i \in \mathbb{R}^2$
1	$i = 1, 2, 3, 4$, FC 32 ReLU FC 1 Intermediate Variable $o_i \in \mathbb{R}^1$
2	$\text{Concat}[o_i \in \mathbb{R}]$, Softmax ($c_n$)
3	$x_n - \mu_{c_n} \in \mathbb{R}^2$
4	2×32 Tanh 2×1 (α_n)

Layer	$p_\theta(x \vert \mu, c, \alpha)$
Input	$\text{Concat}[\alpha_n] \in \mathbb{R}^1$
1	FC 32 Tanh FC 2, Let $g_\theta(\alpha_n)$ denote the output
2	Normal($x_n; g_\theta(\alpha_n) + \mu_{c_n}, \sigma_\epsilon^2 I$)

BOUNCING MNIST

Layer	$p_\theta(x \vert z^{\text{what}}, z^{\text{where}})$
Input	$z_i^{\text{what}} \in \mathbb{R}^{10}$
1	FC 100 ReLU
2	FC 200 ReLU
3	digit $d_i \in \mathbb{R}^{784}$
4	ST($d_i, z_{i,t}^{\text{where}}) \in \mathbb{R}^{9276}, i = 1..., i, t = 1..., T$

Layer	$q_\phi(z^{\text{what}} \vert z^{\text{where}})$
Input	$x_t \in \mathbb{R}^{9276}, z_{i,t}^{\text{where}} \in \mathbb{R}^2, i = 1..., I, t = 1..., T$
1	ST(x_t, $z_{i,t}^{\text{where}}) \in \mathbb{R}^{784}, i = 1, .., I, t = 1, ..., T$
2	FC 200 ReLU
3	FC 100 ReLU
4	$z_{i,t}^{\text{what}} \in \mathbb{R}^{10}, i = 1, .., I, t = 1, ..., T$
5	Mean($z_{1,t}^{\text{what}}, 1 : T) \in \mathbb{R}^{10}, i = 1, ..., I$

Layer	$q_\phi(z^{\text{where}} \mid z^{\text{what}})$
Input	$x_t \in \mathbb{R}^{9276}$, $z_{i,t}^{\text{what}}, i = 1, .., t = 1, .., T$
1	Conv2d(x_t, $z_{i,t}^{\text{what}}$)$\in \mathbb{R}^{4638}$, $i = 1, .., t = 1, .., T$
2	FC 200 Tanh
3	2× FC 100 Tanh
4	2×2 Tanh

G. Analytical inclusive KL divergence during Training in GMM

In GMM experiment, we train the model with different number of sweeps K under fixed computational budget $K \cdot L = 100$. Figure 5 shows that more number of sweeps results in slightly faster convergence.

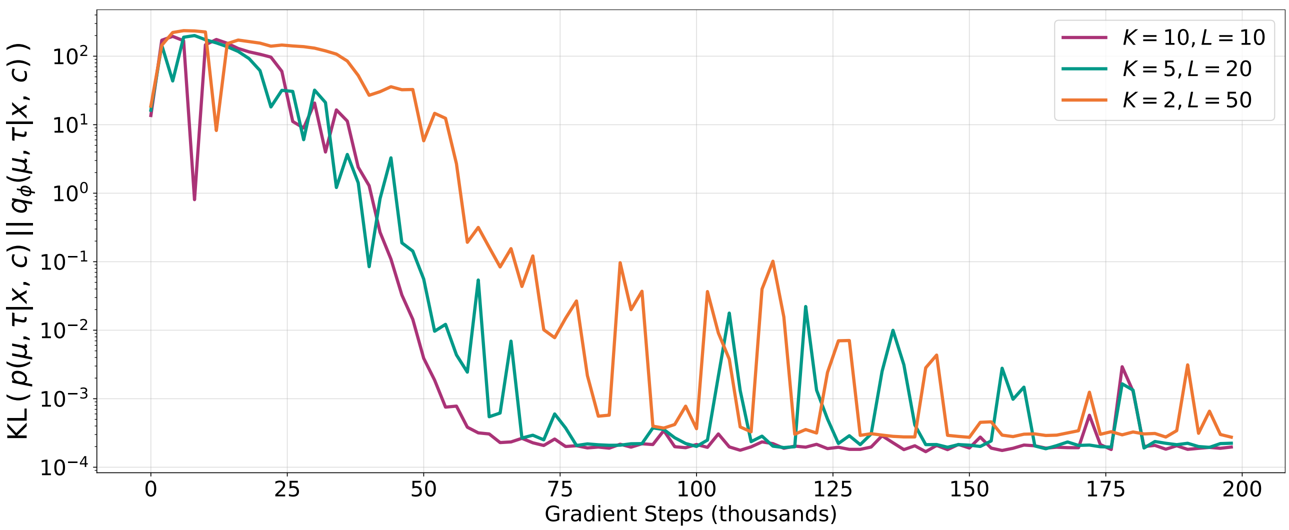


Figure 5. Inclusive KL divergence as a function of gradient steps. Each model is trained with 20000 gradient steps, $K \cdot L = 100$

H. Comparison between APG sampler and RWS method in Bouncing MNIST

We visualize the inference results and reconstruction from the APG sampler and reweighted wake-sleep method. We can see that APG sampler significantly improves both tracking inference results and the reconstruction on instances with $T = 20$ time steps and $D = 5$ digits. We can see that the APG sampler achieves substantial results while the RWS method does not make reasonable prediction at all.

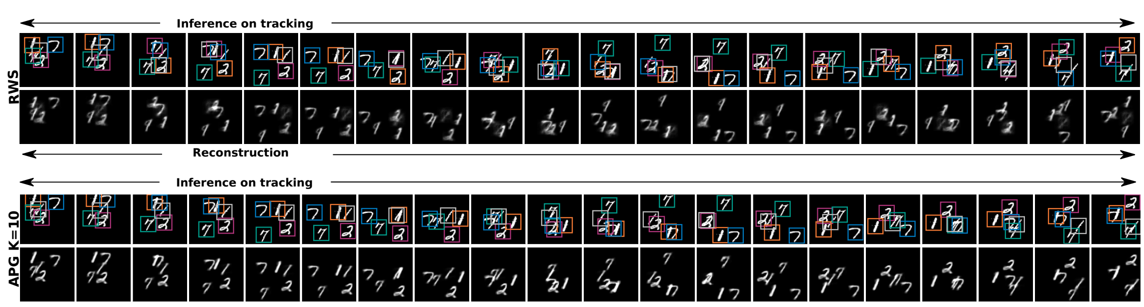


Figure 6. Example 1.

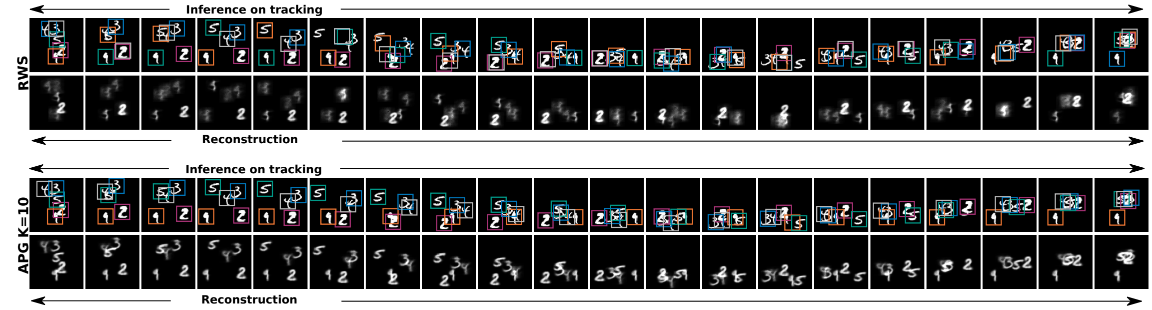


Figure 7. Example 2.

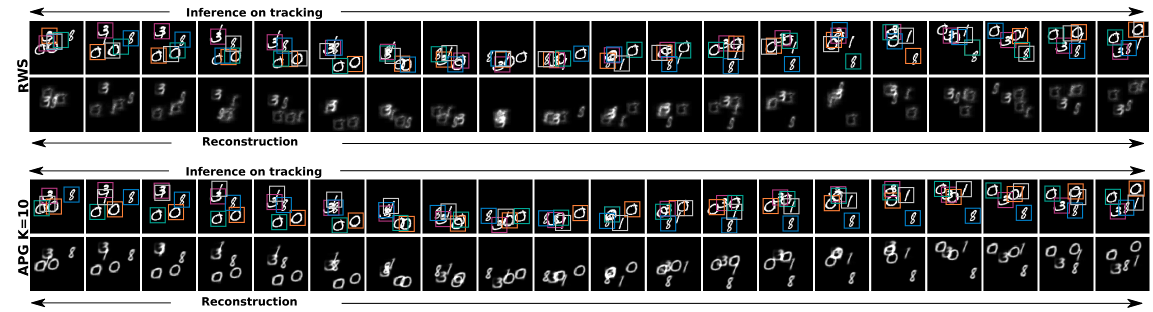


Figure 8. Example 3.

I. Inference Results and Reconstruction on large time steps Bouncing MNIST

We show the inference results on tracking and the reconstruction on test instances with $T = 100$ time steps and $D = 3, 4, 5$ MNIST digits, using models that is trained with instances containing only $T = 10$ time steps and $D = 3$ digits. In each figure below, the 1st, 3rd, 5th, 7th, 9th rows show the inference results, while the other rows show the reconstruction of the series above. We can see the APG sampler is scalable with much large number of latent variables, achieving accurate inference and making impressive reconstruction.

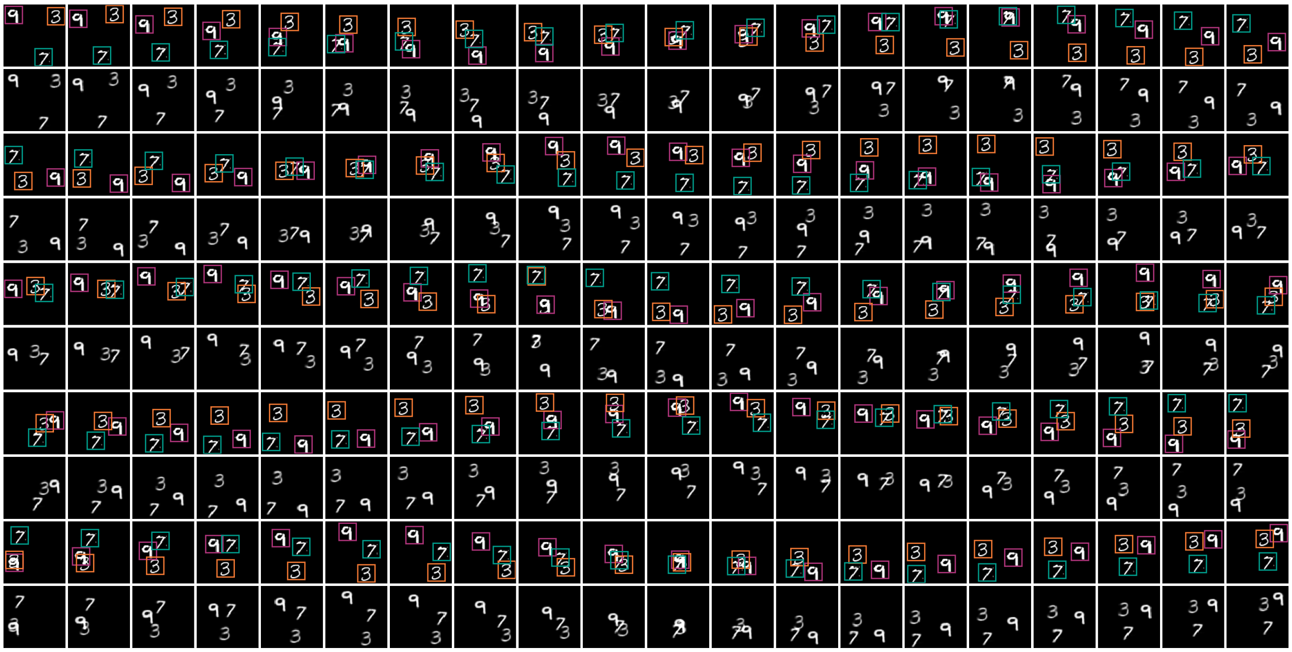


Figure 9. Full reconstruction for a video where $T = 100, D = 3$.

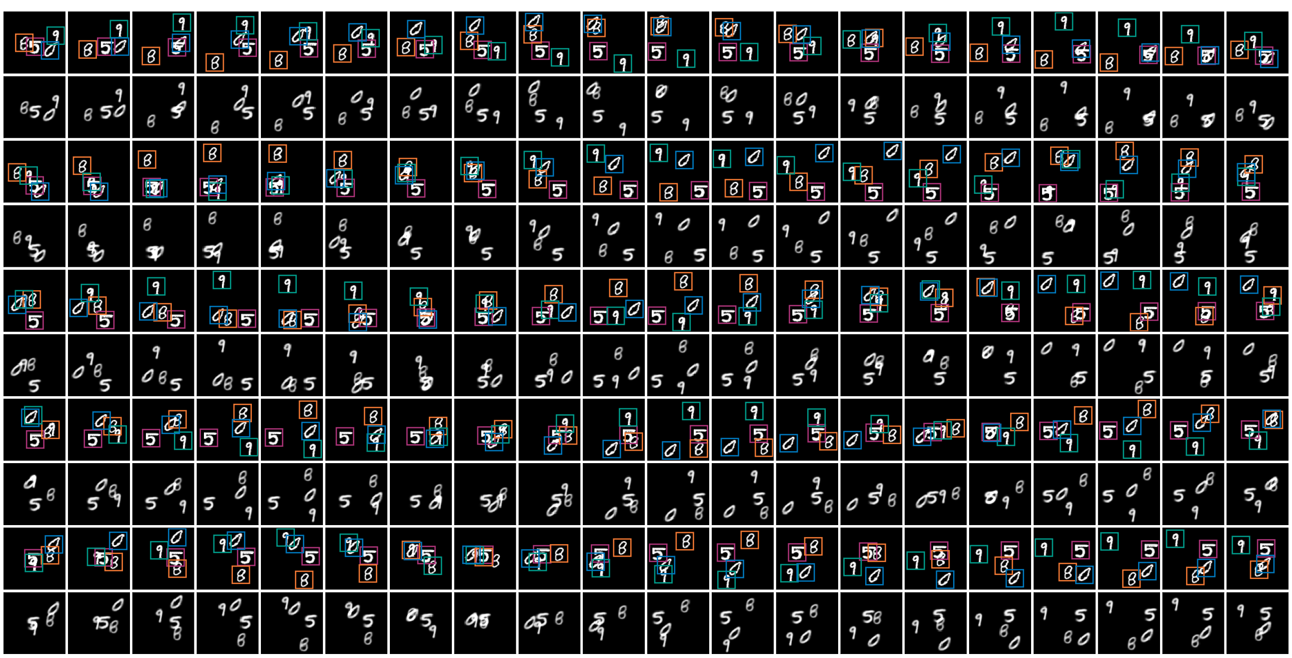


Figure 10. Full reconstruction for a video where $T = 100, D = 4$.

Figure 11. Full reconstruction for a video where $T = 100, D = 5$.

Figure 12. Full reconstruction for a video where $T = 100, D = 5$.