Amortized Population Gibbs Samplers with Neural Sufficient Statistics

Hao Wu
Northeastern University
haowu@ccs.neu.edu

Heiko Zimmermann
Northeastern University
hzimmermann@ccs.neu.edu

Eli Sennesh
Northeastern University
esennesh@ccs.neu.edu

Tuan Anh Le
Massachusetts Institute of Technology
tuananh@mit.edu

Jan-Willem van de Meent
Northeastern University
jwvdm@ccs.neu.edu

Abstract

We develop amortized population Gibbs (APG) samplers, a new class of autoencoding variational methods for deep probabilistic models. APG samplers construct high-dimensional proposals by iterating over updates to lower-dimensional blocks of variables. Each conditional update is a neural proposal, which we train by minimizing the inclusive KL divergence relative 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, resulting in quasi-conjugate variational approximations. Experiments demonstrate that learned proposals converge to the known analytical conditional posterior in conjugate models, and that APG samplers can learn inference networks for highly-structured deep generative models when the conditional posteriors are intractable. Here APG samplers offer a path toward scaling up stochastic variational methods to models in which standard autoencoding architectures fail to produce accurate samples.

1 Introduction

Deep probabilistic programming libraries such as Edward [1], Pyro [2], and Probabilistic Torch [3] extend deep learning frameworks with functionality for deep probabilistic models which combine a generative model with an inference model that approximates the Bayesian posterior. Both models are parameterized using neural networks, which are trained using stochastic gradient descent by optimize a lower or upper bound on the log marginal likelihood. Training an inference network to perform amortized inference can be equivalently understood as a form of variational inference or adaptive importance sampling.

At present, deep probabilistic models most commonly have the form of standard variational autoencoders (VAEs) [4, 5]. In these architectures, the generative model combines an unstructured prior (e.g. a spherical Gaussian) with a likelihood that is parameterized by an expressive neural network, often referred to as a decoder. The inference network, known as an encoder, maps input data (e.g. an image or sentence) onto an embedding vector, also known as the latent code.

Deep probabilistic programming aims to enable more general designs that incorporate structured priors for tasks such as multiple object detection [6], language modeling [7], or object tracking [8]. In these domains, a prior can incorporate useful inductive biases, such as the requirement that object trajectories are smooth. These biases in turn can help guide a model to uncover patterns in the data in an unsupervised manner, and aid generalization in complex domains where the training data may not contain exemplars for all possible combinations of latent features.

However, training structured models also poses challenges that are not encountered in unstructured problems. To optimize a lower or an upper bound, we need to approximate the gradient of an expectation with a Monte Carlo estimate (see [9] for a recent review). Standard VAEs rely on reparameterized estimators that can often approximate the gradient with a single sample. Unfortunately, these estimators can have a high variance in models where latent variables are high-dimensional and/or strongly correlated. Owing to these limitations, models that are trained using standard VAE objectives often consider relatively small-scale problems, such as tracking ≤ 2 objects over the course of 10 frames [8], or assigning ≤ 10 sentences in a review to distinct aspects [7].

In this paper, we develop methods for amortized inference that are designed to scale to structured models with 100s of latent variables. We are particularly interested in the frequently arising cases of models that are characterized by a combinations of *local variables*, such as the time-dependent position of an object, and *global variables*, such as the shape of the object. In this type of model, it is often the case that knowledge of the local variables can help us make predictions about global variables and vice versa; If we know the shape of an object, then it should be easier to identify its location in an image. Conversely, if we know the position of an object in each frame, then we can more readily infer its shape.

The methods that we develop in this paper are similar in spirit to work by Johnson et al. [10], who developed methods for conjugate-exponential models with a neural likelihood. In this setting, we can perform inference using variational expectation maximization (EM) algorithms [11–13] that exploit conjugacy and conditional independence to derive closed-form updates to blocks of variables. The advantage of these approaches is that they are highly computationally efficient; variational EM can often converge in a small number of iterations and easily scales to much larger number of variables. Unfortunately, variational EM is also model-specific, difficult to implement, and only applicable to a restricted class of conjugate-exponential models.

To overcome the limitations imposed by conjugate-exponential family models, we here develop a more general approach. Rather than requiring exact EM updates we develop an importance sampling method that employs conditional proposals to iterate between updates to blocks of variables. To train these proposals, we define a a variational method that minimizes the inclusive KL divergence between the proposal update and the exact conditional posterior. We refer to the resulting class of methods as *amortized Gibbs* samplers, since the proposals approximate Gibbs updates.

The variational objective that we derive is not computable, since the exact Gibbs updates are in general intractable. However, we can nonetheless derive a Monte Carlo estimator for its gradient. Building on a recent body of work that employs importance samplers to train variational distributions [14–17], we develop a sequential Monte Carlo sampler [18] that combines approximate Gibbs updates with resampling steps in order to construct high quality proposals, which serve both to compute gradient estimates at train time and to perform inference at test time. We demonstrate correctness of the proposed sampler by proving that samples are properly weighted relative to the generative model [19].

One of the challenges in designing networks that parameterize conditional proposals is network outputs need to appropriately account for the amount of data on which we are conditioning; The conditional posterior on the mean for a cluster with a large number of points is more tightly peaked than that of a cluster with a small number of points. To address this difficulty, we propose a class of networks that we refer to as *neural sufficient statistics*, which define parameterizations of proposals in a manner that is additive in the local variables, much like the sufficient statistics in conjugate-exponential families.

Our experiments show that learned proposals converge to the true conditional posteriors in Gaussian mixture models, where the Gibbs updates can be computed in closed form. Moreover we establish that amortized Gibbs methods serve can a basis for scalable inference in structured deep generative models, including mixtures with neural likelihoods and unsupervised tracking models. Both of these tasks are representative of the current state-of-the art in unsupervised approaches for learning structured deep generative models.

2 Amortized Population Gibbs Samplers

We are interested in the task of jointly training a generative model $p_\theta(x, z)$ by maximizing its marginal likelihood $p_\theta(x)$ and learning an inference model $q_\phi(z \mid x)$ that approximates the posterior $p_\theta(z \mid x)$. Like most amortized inference approaches, we assume that we can sample from a (possibly implicit) distribution $\hat{p}(x)$ that either takes the form of an empirical distribution over training data or a data simulator.

As a means of generating high-quality samples in an incremental manner, we develop methods that are inspired by expectation maximization and classic Gibbs sampling strategies, which perform iterative updates to blocks of variables. Concretely, we will assume that the latent variables in the generative model decompose into blocks $z = \{z_1, \ldots, z_B\}$ and train proposals $\log q_\phi(z_b \mid x, z_{-b})$ that update the variables in a each block z_b conditioned on the variables in the remaining blocks $z_{-b} = z \setminus \{z_b\}$.

Starting with an initial sample $q_\phi(z^1 \mid x)$ from a standard encoder we will generate a sequence of samples $\{z^1, \ldots, z^K\}$ by performing conditional updates to each block z_b, which we refer to as a *sweep*

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

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

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

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 strategy 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 [20].

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*

$$p_\theta(z^k \,|\, x) = \int p_\theta(z^{k-1} \,|\, x)\, q_\phi(z^k \,|\, x, z^{k-1}) dz^{k-1}. \quad (2)$$

When this condition is met, the proposal $q_\phi(z^1, \dots z^K \,|\, x)$ 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.

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

$$\begin{aligned} p_\theta(z'_b, z_{-b} \mid x) &= \int p_\theta(z_b, z_{-b} \mid x)\, q_\phi(z'_b, \mid x, z_{-b}) dz_b, \\ &= p_\theta(z_{-b} \mid x)\, q_\phi(z'_b \mid x, z_{-b}), \end{aligned} \quad (3)$$

Then we see that a block 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 3 is met, the proposal $q_\phi(z^1, \dots z^K \,|\, x)$ is a Gibbs sampler.

2.1 Variational Objective

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

$$\mathbb{E}_{\hat{p}(x) p_\theta(z_{-b}|x)} \left[\mathrm{KL} \left(p_\theta(z_b \mid x, z_{-b}) || q_\phi(z_b \mid x, z_{-b}) \right) \right]. \quad (4)$$

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 5, this has implications for the evaluation of learned proposals, since we cannot compute a lower or upper bound on the log marginal likelihood as in other variational methods. However, it nonetheless possible to approximate the gradient of the objective

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

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

$$-\nabla_\phi \mathcal{K}_b(\phi) \simeq \sum_{l=1}^{L} \frac{w^l}{\sum_{l'} w^{l'}} \nabla_\phi \log q_\phi(z_b^l \,|\, x, z_{-b}^l). \quad (5)$$

In problems where we would like to learn a deep generative model $p_\theta(x, z)$, we can apply a similar self-normalized gradient estimator of the form

$$\begin{aligned} \nabla_\theta \log p_\theta(x) &= \mathbb{E}_{p_\theta(z|x)} \left[\nabla_\theta \log p_\theta(x, z) \right] \\ &\simeq \sum_{l=1}^{L} \frac{w^l}{\sum_{l'} w^{l'}} \nabla_\theta \log p_\theta(x, z^l). \end{aligned} \quad (6)$$

This identity holds due to the standard property $\mathbb{E}_{p_\theta(z|x)}[\nabla_\theta \log p_\theta(z|x)] = 0$ (see Appendix A for details).

The estimator in Equation 5 is similar to the self-normalized estimator in reweighted wake-sleep methods [21], which also minimizes an inclusive KL divergence. This estimator has a number of advantages over the estimator that is commonly used to train standard VAEs, which minimize an exclusive KL divergence [22]. Standard VAE objectives rely on reparameterization to compute gradient estimates. For discrete variables, reparameterization is not possible. This means that we need to compute likelihood-ratio estimators (also known as REINFORCE-style estimators [23]), which can have very high variance. A range of approaches for variance reduction have been put forward, including continuous relaxations that are amenable to reparameterization [24, 25], credit assignment techniques (see [26] for a review), and other control variates [27–29].

The estimator in Equation 5 sidesteps the need for these variance reduction techniques. To compute this gradient, we only require 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, since this estimator minimizes the inclusive KL divergence, and not the exclusive KL divergence, there is smaller risk of learning a proposal that collapses to a single mode of a multi-modal posterior [22].

2.2 Generating High Quality Samples

Approximating the gradient presents a chicken-and-egg problem; we need samples from the posterior to compute a Monte Carlo estimate of the gradient, but generating these samples is precisely what we are hoping to use learned proposals for in the first place. Moreover, self-normalized importance samplers are consistent, but they are not unbiased. In the early stages of training, we will have poor quality proposals, which means that the bias of the gradient estimators in Equations 5 and 6 can be high.

Standard reweighted wake-sleep methods generate proposals from an encoder $z \sim q_\phi(z \mid x)$ and compute weights $w = p_\theta(x, z)/q_\phi(z \mid x)$. A well-known limitation of this type of naive importance sampling strategy is that the computed weights will have a very high variance in models with high-dimensional and/or correlated latent variables, which in turn implies a high bias of the estimator. There is a very

broad class of importance sampling strategies that can be employed to reduce the variance of importance weights. If we replace the naive importance sampler in reweighted wake-sleep with a more sophisticated sampling strategy, then this both improves the quality of gradient estimates at training, and the quality of inference results at test time.

To improve upon standard reweighted wake-sleep methods, we will we use the learned proposals to define a sequential Monte Carlo (SMC) sampler [18]. SMC methods [30] combine two basic ideas. The first is sequential importance sampling, which decomposes a proposal for a sequence of variables into a sequence of conditional proposals. The second is resampling, which selects partial proposals with probability proportional to their weights in order 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, which is sometimes also referred to as *resample-move* SMC.

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

Sequential Importance Sampling. To understand how approximate Gibbs proposals can be used in a SMC sampler, we will first explain how they can be used to define a sequential importance sampler, which decomposes the importance weight into a sequence of *incremental* weights. In general, 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)

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

We will now consider a specific sequence of intermediate densities that are defined using 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* such that

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

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

$$v^k = \frac{p_\theta(x, z^k)\, r(z^{k-1} \mid z^k)}{p_\theta(x, z^{k-1})\, q(z^k \mid z^{k-1})}. \tag{7}$$

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

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

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

We will now use the approximate Gibbs kernel from Equation 1 as both the forward and the reverse kernel

$$q(z^k \mid z^{k-1}) = r(z^{k-1} \mid z^k) = q_\phi(z^k \mid x, z^{k-1}). \tag{8}$$

When the approximate Gibbs kernel converges to the actual Gibbs kernel, this choice becomes optimal, since the kernel will satisfy detailed balance in this limit

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

Resampling. In general, the weights w^k in the sequential importance sampling scheme defined above will have a high variance. The weights w^1 are just normal importance sampling weights, which themselves will have a high variance

Algorithm 1 SMC sampler

for $l = 1$ **to** L **do**
  $z^{1,l} \sim q^1(\cdot)$ ▷ Propose
  $w^{1,l} = \frac{\gamma^1(z^{1,l})}{q^1(z^{1,l})}$ ▷ Weigh
for $k = 2$ **to** K **do**
  $z^{k-1,1:L}, w^{k-1,1:L} = \text{RESAMPLE}(z^{k-1,1:L}, w^{k-1,1:L})$
  for $l = 1$ **to** L **do**
    $z^{k,l} \sim q^k(\cdot \mid z^{k-1,l})$ ▷ Propose
    $w^{k,l} = \frac{\gamma^k(z^{k,l}) r^{k-1}(z^{k-1,l} \mid z^{k,l})}{\gamma^{k-1}(z^{k-1,l}) q^k(z^{k,l} \mid z^{k-1,l})} w^{k-1,l}$ ▷ Weigh

Algorithm 2 RESAMPLE

Input: $z^{1:L}, w^{1:L}$
Output: $z'^{1:L}, w'^{1:L}$
for $i = 1$ **to** L **do**
  $a^i \sim \text{Disc}(\{w^l / \sum_{l'=1}^{L} w^{l'}\}_{l=1}^{L})$ ▷ Index Selection
  Set $z'^{i} = z^{a^i}$
  Set $w'^{i} = \frac{1}{L} \sum_{l=1}^{L} w^l$ ▷ Re-Weigh
Return $z'^{1:L}, w'^{1:L}$

when z is high-dimensional, or there are correlations between variables. Moreover, we are now sampling these same 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 this limit), but during training variance of weights w^k will increase with k, since we are now jointly sampling an entire Markov chain. To overcome this problem, SMC samplers interleave application of the transition kernel with a *resampling* step. This step generates a new set of samples by selecting current samples with replacement, with probability proportional to their weight. Concretely, suppose that we have a set *incoming* samples $\{(w^{k,l}, z^{k,l})\}_{l=1}^{L}$, then the resampling procedure (see Algorithm 2) selects index a with probability $P(a{=}l) = w^{k,l} / \sum_{l'} w^{k,l'}$ and returns an outgoing sample $z'^{k,l} = z^{k,a}$ whose $w'^{k,l} = \frac{1}{L} \sum_l w^{k,l}$ is equal to the average weight. The reduces the variance of the importance weights at the expense of also reducing the diversity of the sample set; high-weight samples are selected frequently, whereas low-weight samples are selected infrequently or not at all.

When we perform resampling after each sweep, we reduce the variance of importance weights to an extent. However we will likely still have high-variance weights, since each sample from the approximate Gibbs kernel still constitutes a high-dimensional proposal over all variables in the model. To further reduce the variance, we will employ resampling after each block update, rather than after each sweep. Because the incoming weights are now equal at each block update, we can compute gradient estimates using incremental weights v of the form

$$v = \frac{p_\theta(x, z'_b, z_{-b})\, q_\phi(z_b \mid x, z_{-b})}{p_\theta(x, z_b, z_{-b})\, q_\phi(z'_b \mid x, z_{-b})}. \tag{9}$$

These incremental weights will have a much lower variance than the incremental weights for a full sweep, since we are now able to decompose a sampling problem for all the variables in a model into sampling problems for individual blocks. In models with many latent variables, such as the ones that we will consider in our experiments, this has the potential to greatly increase the tractability of the gradient estimation problem.

We refer to this implementation of a SMC sampler as an amortized population Gibbs (APG) sampler, and summarize all the steps of the computation in Algorithm 3. In Appendix D, we prove that this algorithm is correct using an argument based on proper weighting. More informally this property holds due to the fact that this sampler is a specific instance of a SMC sampler.

Algorithm 3 Amortized Population Gibbs Sampling

$g_\phi = 0, g_\theta = 0$ ▷ Initialize gradient to 0
for $l = 1$ **to** L **do** ▷ Initial proposal
  $z^{1,l} \sim q_\phi(\cdot \mid x)$ ▷ Propose
  $w^{1,l} = \frac{p_\theta(x, z^{1,l})}{q_\phi(z^{1,l} \mid x)}$ ▷ Weigh
$g_\phi = g_\phi + \sum_{l=1}^{L} \frac{w^{1,l}}{\sum_{l'=1}^{L} w^{1,l'}} \nabla_\phi \log q_\phi(z^{1,l} \mid x)$
$g_\theta = g_\theta + \sum_{l=1}^{L} \frac{w^{1,l}}{\sum_{l'=1}^{L} w^{1,l'}} \nabla_\theta \log p_\theta(x, z^{1,l})$
for $k = 2$ **to** K **do** ▷ Gibbs sweeps
  $\tilde{z}^{1:L}, \tilde{w}^{1:L} = z^{k-1,1:L}, w^{k-1,1:L}$
  for $b = 1$ **to** B **do** ▷ Block updates
    $\tilde{z}^{1:L}, \tilde{w}^{1:L} = \text{RESAMPLE}(\tilde{z}^{1:L}, \tilde{w}^{1:L})$
    for $l = 1$ **to** L **do**
      $\tilde{z}'^{\,l}_b \sim q_\phi(\cdot \mid x, \tilde{z}^l_{-b})$ ▷ Propose
      $\tilde{w}^l = \frac{p_\theta(x, \tilde{z}'^{\,l}_b, \tilde{z}^{\,l}_{-b}) q_\phi(\tilde{z}^{\,l}_b \mid x, \tilde{z}^{\,l}_{-b})}{p_\theta(x, \tilde{z}^l_b, \tilde{z}^{\,l}_{-b}) q_\phi(\tilde{z}'^{\,l}_b \mid x, \tilde{z}^{\,l}_{-b})} \tilde{w}^l$ ▷ Weigh
      $\tilde{z}^l_b = \tilde{z}'^{\,l}_b$ ▷ Reassign
    $g_\phi = g_\phi + \sum_{l=1}^{L} \frac{\tilde{w}^l}{\sum_{l'=1}^{L} \tilde{w}^{l'}} \nabla_\phi \log q_\phi(\tilde{z}^l_b \mid x, \tilde{z}^{\,l}_{-b})$
    $g_\theta = g_\theta + \sum_{l=1}^{L} \frac{\tilde{w}^l}{\sum_{l'=1}^{L} \tilde{w}^{l'}} \nabla_\theta \log p_\theta(x, \tilde{z}^l)$
  $z^{k,\,1:L}, w^{k,\,1:L} = \tilde{z}^{1:L}, \tilde{w}^{1:L}$
return g_ϕ, g_θ ▷ Output: gradient

3 Neural Sufficient Statistics

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

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

In these densities $h(\cdot)$ 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

$$p(z \mid x) \propto h(z) \exp\big\{(\lambda_1 + T(x))^\top T(z) - (\lambda_2 + 1) \log A(\eta(z))\big\}.$$

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 the more general setting we consider here, these conjugacy relationships will typically not hold. However, we can still take inspiration to design variational distributions that make use of conditional independencies in a model. We will 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 $T_\phi(x, z_{-b})$

$$q_\phi(z_b \mid x, z_{-b}) = p(z_b \mid \lambda + T_\phi(x, z_{-b})). \tag{10}$$

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. [13]), 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 particularly useful case arises in models where the data $x = \{x_1, \ldots, x_N\}$ are independent conditioned on z. In these models it is often possible to partition the latent variables $z = \{z^{\mathrm{G}}, z^{\mathrm{L}}\}$ into global and local 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.

4 Related Work

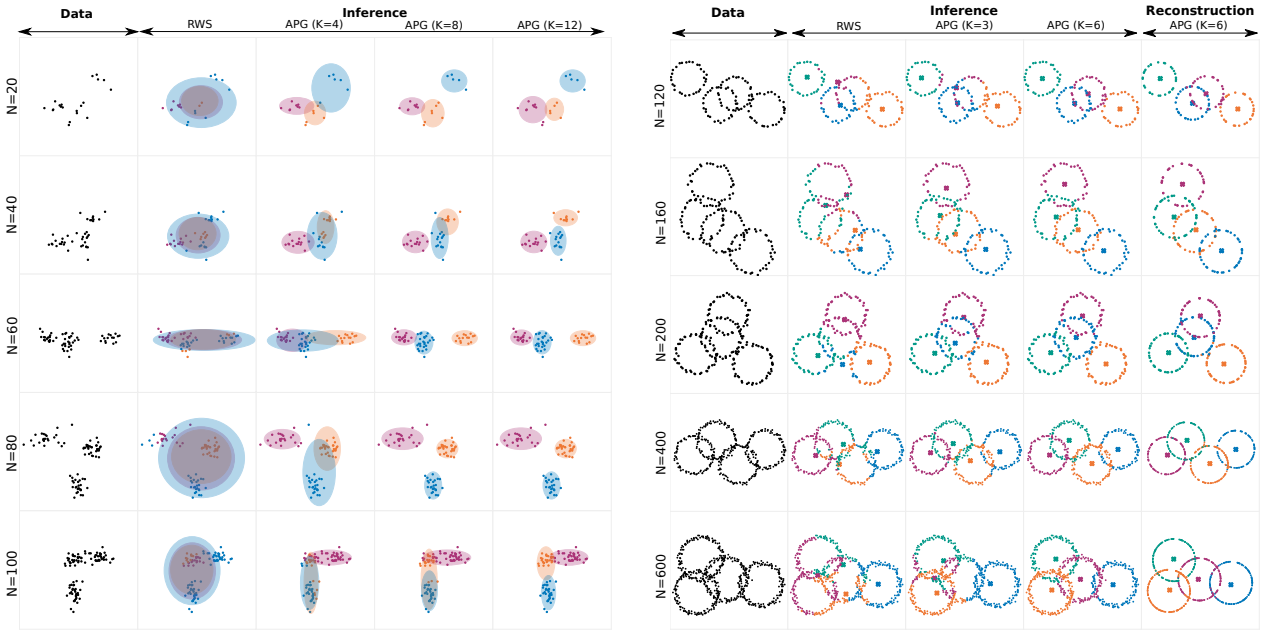
Our work fits into a line of recent methods for deep generative modeling that seek to improve inference quality, either by introducing auxiliary variables [32, 33], or by performing iterative updates [34]. Our specific approach to learning block proposals is related to a number of methods that, in some way or other, combine transition kernels with variational inference. Work by Hoffman [35] applies Hamiltonian Monte Carlo to samples that are generated from the encoder, which serves to improve the gradient estimate w.r.t. θ (Equation 6), while learning the inference network using a standard reparameterized ELBO objective. Li et al. [36] similarly use MCMC to improve the quality of samples that are generated by 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. therefore use an adversarial objective to minimize the inclusive KL. Neither of these two lines of work consider block decomposition of the latent variable space, nor do they learn transition kernels.

Work by Salimans et al. [37] uses transition kernels in variational inference. The authors use an importance weight to define (stochastic) lower bound, which is defined using a forward and reverse kernel in the same manner as in Equation 7. Huang et al. [20] extend the work by Salimans et al. 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, gradient estimates must be computed using reparameterization, which means that these methods are not applicable to models that contain discrete variables. Moreover, these methods perform a joint update on all variables at each iteration, and do not consider the task of learning conditional proposals as we do here.

Work by Wang et al. [38] develops a meta-learning approach to learning Gibbs block conditionals. This work assumes a setup in which it is possible to sample $x, z \sim p(x, z)$ from the true generative model $p(x, z)$, which means gradients can be estimated using sleep-phase Monte Carlo estimators. This circumvents the need for self-normalized estimators of the form in Equation 5, which are necessary when we additionally wish to learn the generative model. Like in our work, the approach by Wang et al. minimizes the inclusive KL, but uses the learned conditionals to directly define an (approximate) MCMC sampler, rather than using them as proposals in an 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.

5 Experiments

We evaluate APG methods in 3 tasks. We begin by considering a Gaussian mixture model (GMM) as an exemplar of a model in the conjugate-exponential family. Here we verify that the learned block updates converge the analytical conditional posteriors as predicted by our analysis in Section 2. We next consider a deep generative mixture model (DGMM) that incorporates a neural likelihood to parame-



(a) GMM (b) DGMM

Figure 1: Samples from the GMM and the DGMM. (**a**) GMM, the left column shows 5 test datasets with different number of data points. The subsequent columns show inference results by RWS, followed by results after 4, 8 and 12 APG updates. (**b**) DGMM, the left column shows 5 test datasets with different number of data points. The subsequent columns show the inference results by RWS, followed by results after 3 and 6 APG updates. The right column shows reconstructions from the learned generative model.

terize ring-shaped clusters. We show that we can train both the generative model and inference model in an end-to-end manner using APG methods, and that inference scales to datasets containing up to 600 points. For both models we quantify performance in terms of the effective sample size (ESS) and the relative magnitude of the log joint. In our third experiment, we consider an unsupervised model for multiple bouncing MNIST data. We extend the task proposed by Srivastava et al. [39] to consider up to 5 individual digits, and learn both a deep generative model for videos and an inference model that performs tracking.

Results on each of these tasks constitute a significant advance relative to the state of the art. Standard VAEs perform poorly at Gaussian mixture modeling tasks, and to our knowledge there are no existing methods that scale to a problem of the complexity of the DGMM for rings. In the context of the unsupervised tracking model, APG easily scales beyond previously reported results for a specialized recurrent architecture [8]. APG is not only is able to scale to models with higher complexity in these settings, 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.

5.1 Gaussian Mixture Model

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

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

In this model, the global variables $z^{\text{G}} = \{\mu_{1:I}, \tau_{1:I}\}$ are the mean an 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 $p(x_{1:N} \mid z_{1:N}, \mu_{1:I}, \tau_{1:I})$ is conjugate to a normal-gamma prior $p(\mu_{1:I}, \tau_{1:I})$ with sufficient statistics $T(x_n, c_n)$

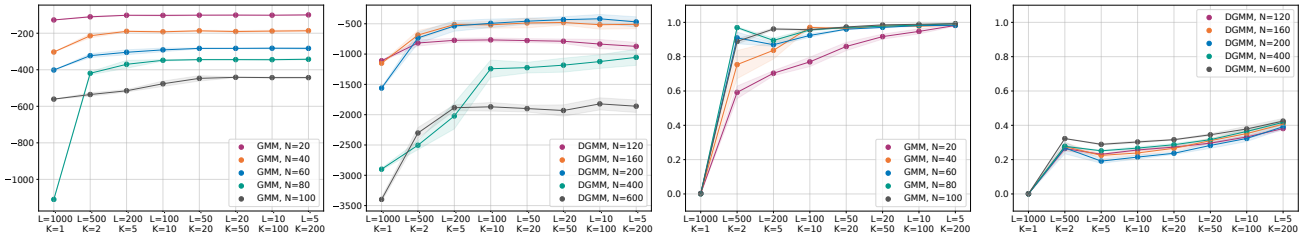
$$\Big\{ \text{I}[c_n = i],\ \text{I}[c_n = i]\, x_n,\ \text{I}[c_n = i]\, x_n^2 \ \Big|\ i = 1, 2, \ldots, I \Big\},$$

where $\text{I}[z_n = i]$ is an indicator function that evaluates to 1 if the equality holds, and 0 otherwise.

We employ a variational distribution that updates the global variables $q_\phi(\mu, \tau \mid x, c)$ and the local variables $q_\phi(c \mid x, \mu, \tau)$, using point-wise neural sufficient statistics modeled after the ones in the analytical updates (see Appendix E for architecture details).

We train our models on 20,000 datasets with $I = 3$ clusters and $N = 60$ data points with fixed hyperparameters ($\mu_0 = 0$, $\nu_0 = 0.3$, $\alpha_0 = 2$, $\beta_0 = 2$). We use 20 GMM datasets per batch, $K = 10$ sweeps, $L = 10$ particles, and Adam ($\text{lr} = 10^{-4}, \beta_1 = 0.9, \beta_2 = 0.99$) for 200,000 iterations.

We compare the APG sampler to samples from a standard encoder with MLP and LSTM architectures, which is trained



(a) $\log p_\theta(x, z)$ (b) ESS / L

Figure 2: APG sampler performance as a function of number of sweeps K for a constant sample budget $K \cdot L = 1000$.

Table 1: APG performance in the GMM and DGMM. The left column in each table shows the change in log joint distribution, i.e. the difference between the log joint in the baseline and the log joint in other models. We compute the ESS/L metric is computed w.r.t. different variable blocks. For the GMM we additionally report the inclusive KL (Equation 4) for each block.

(a) GMM

	$\Delta \log p_\theta(x, z)$	ESS/L			$\mathrm{KL}(p_\theta \| q_\phi)$	
		$\{\tau, \mu, c\}$	$\{\tau, \mu\}$	$\{c\}$	$\{\tau, \mu\}$	$\{c\}$
MLP-RWS	–	0.001	–	–	–	–
LSTM-RWS	202.2	0.104	–	–	–	–
APG (K=5)	198.5	0.261	0.980	0.631	0.005	0.005
APG (K=10)	211.9	0.398	0.981	0.760	0.004	0.004
APG (K=15)	215.2	0.416	0.983	0.780	0.003	0.004

(b) DGMM

$\Delta \log p_\theta(x, z)$	ESS/L		
	$\{\mu, c, \alpha\}$	$\{\mu\}$	$\{c, \alpha\}$
2538	0.001	–	–
–	0.001	–	–
6201	0.002	0.013	0.422
6293	0.002	0.019	0.454
6310	0.003	0.025	0.488

using reweighted wake-sleep (RWS). Both architectures are parameterized using the same neural sufficient statistics as the APG sampler.

Figure 1a shows sequences of single samples from the variational distribution, where the first sample is drawn using RWS. Even when using a parameterization that employs neural sufficient statistics, the RWS encoder fails to propose reasonable clusters, whereas the APG sampler typically converges within 12 iterations across a range of dataset sizes.

Furthermore, we would like to quantify how similar learned proposals $q_\phi(z_b \mid x, z_{-b})$ are to the conditional posteriors $p_\theta(z_b \mid x, z_{-b})$. With the case of GMM where the exact conditional posterior is tractable, we verify the convergence of the learned proposals by computing the inclusive KL divergence $\mathcal{K}_b(\phi)$ defined in equation 4 (see Table 1a). We can see that the APG samplers of the both $\{\tau, \mu\}$ and $\{c\}$ converge to the true conditional posterior.

5.2 Deep Generative Mixture Model

We next consider the task of training a deep generative model $p_\theta(x, z)$ is jointly with the APG sampler. Our dataset consists of ring-shaped clusters. The true generative model (which we assume is unknown) takes the form

$$\begin{aligned} \mu_i &\sim \mathrm{Norm}(0, \sigma_0^2 I), \qquad i = 1, 2, ..., I \\ c_n &\sim \mathrm{Disc}(\pi), \quad \alpha_n \sim \mathrm{Unif}[0, 2\pi], \\ x_n | c_n = i &\sim \mathrm{Norm}(g_\theta(\alpha_n) + \mu_i, \Sigma_\epsilon). \end{aligned}$$

Here μ_i is center of the ith ring. Given a cluster assignment c_n and an angle α_n we define a position on a ring, from which We sample a data point x_n with 2D Gaussian noise.

We train our model on 20,000 datasets with $N = 200$ data points and $I = 4$ clusters with fixed hyperparameters ($\sigma_0 = 3.5$, $\Sigma_\epsilon = 0.2$). We use 20 datasets per batch, $K = 10$ sweeps, $L = 10$ particles, and Adam ($\mathrm{lr} = 10^{-4}, \beta_1 = 0.9, \beta_2 = 0.99$) for 200,000 iterations (see Appendix E for architecture details).

Once again, we compare the APG sampler with the encoders using RWS. Figure 1b shows individual samples analogous to the ones in Figure 1a. The APG sampler scales to a large range of number of variables, whereas a standard encoder trained using RWS fails to produce reasonable proposals.

5.3 Sample Quality Evaluation

In both mixture models, we compute the log-joint distribution $\log p_\theta(x, z)$ (see Table 1) as a function of sweep iteration to measure the convergence and the effective sample size (see Table 1) to assess proposal quality

$$\frac{\mathrm{ESS}}{L} = \frac{(\sum_{l=1}^{L} w^{k,l})^2}{L \sum_{l=1}^{L} (w^{k,l})^2}. \tag{11}$$

Log joint $\log p_\theta(x, z)$. Because the marginal $q_\phi(z^k | x)$ is intractable, it is difficult to compute an lower bound or upper bound at each sweep. Here we compute the log joint in each test dataset for both the APG sampler with different number

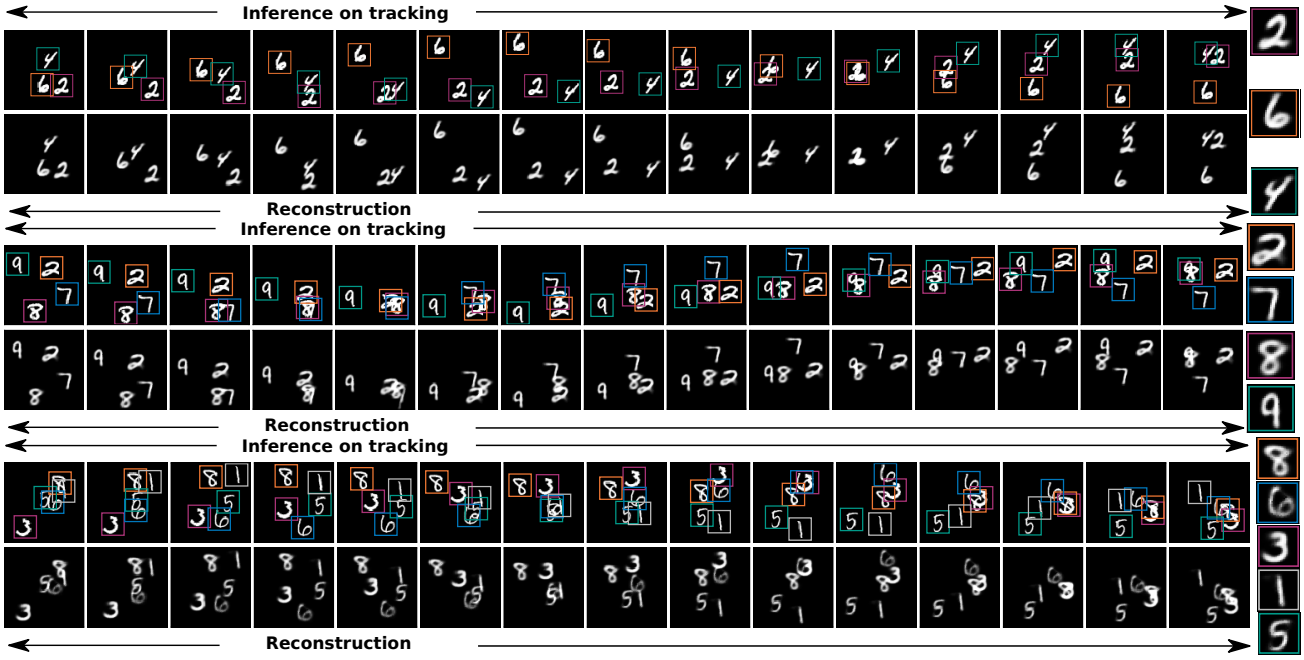


Figure 3: Inferred digit trajectories and reconstructions for (top) $D = 3$ (middle) $D = 4$ and (bottom) $D = 5$ digits for $T = 15$ for a model trained on $D = 3$ and $T = 10$.

of sweeps and the RWS baselines and report the differences on average to see how much more is achieved by the APG sampler. In both models, the APG sampler gains a higher log joint compared with the encoder trained by RWS.

ESS. One advantage of the APG sampler that it decomposes a high dimensional sampling problem into a sequence of lower dimensional sampling problems. To show that, we compute the ESS when 1) we resample only after one sweep and 2) we resample after each block update. WE can see that the granular sampling strategy significantly improves the ESS in both cases.

5.4 Fixed Computation Budget Analysis

As a mean of comparing the performance of APG samplers for varying numbers of sweeps K, we perform an experiment in which the computation budget is fixed at $K \cdot L = 1000$ samples. Figure 2 shows $\log p_\theta(x, z)$ and ESS/L. The shaded area denotes the standard deviation over 10 runs that each comprise 5 datasets that were chosen at random. We can see that it in general, it is more effective to perform more APG sweeps K with a smaller number of particles L, that it is to increase the particle budget.

5.5 Time Series Model – Bouncing MNIST

Finally, we apply the APG sampler to a time series model that is trained with short timescales, and evaluate its performance with longer timescales and larger numbers of latent variables. The data $x_{1:T}$ is a sequence of images of D moving MNIST digits. Our generative model consists of global variables $z^{\text{what}}_{1:D}$ corresponding to digit latent variables and local variables $z^{\text{where}}_{1:D,1:T}$ corresponding to the digit trajectories. The deep generative model is a state space model that factorizes across digits of the form

$$
\begin{aligned}
z^{\text{what}}_d &\sim \text{Norm}(0, I), \quad z^{\text{where}}_{d,1} \sim \text{Norm}(0, I), \\
z^{\text{where}}_{d,t} &\sim \text{Norm}(z^{\text{where}}_{d,t-1}, \sigma_0^2 I) \\
x_t &\sim \text{Bern}\Big(\sigma\Big(\sum_d \text{ST}\big(\mu_\theta(z^{\text{what}}_d), z^{\text{where}}_{d,t}\big)\Big)\Big)
\end{aligned}
$$

Here, ST is a spatial transformer [40] that maps the output of a feedforward decoder μ_θ that maps logits for a 28×28 MNIST image onto a 96×96 canvas based on the location variable $z^{\text{where}}_{d,t}$.

Our amortized Gibbs updates employ $T + 1$ blocks $(z^{\text{what}}_{1:D}, z^{\text{where}}_{1:D,1}, z^{\text{where}}_{1:D,2}, \ldots, z^{\text{where}}_{1:D,T})$. Empirically this works better than splitting the latent variables into global and local variables, since resampling at each time step t helps disentangle the digit locations if they overlap.

We train our model on 60000 bouncing MNIST sequences, each of which contains $D = 3$ digits and $T = 10$ frame images. We use 10 sequences per batch, $K = 5$ sweeps, $L = 10$ particles, and Adam (lr $= 10^{-4}, \beta_1 = 0.9, \beta_2 = 0.99$) for 200,000 iterations (see Appendix E for architecture details).

We show that APG sampler can scale to larger number of variables by testing the model on datasets with $T \in \{20, 100\}$ time steps and $D \in \{3, 4, 5\}$ digits. Figure 3

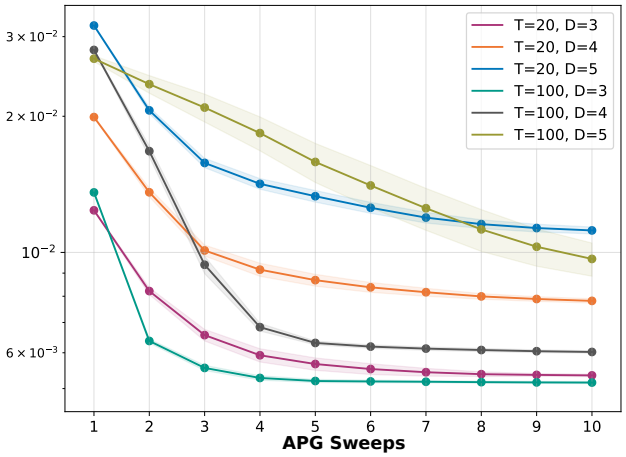


Figure 4: Mean squared error between video frames and reconstructions as a function of the number of APG sweeps.

shows the inference and reconstruction using single samples from the variational distribution. (plots are truncated by the first 15 time steps due to limited space, see Appendix F for more examples with full time steps). Qualitatively, we see that the digit trajectories $z^{\text{where}}_{1:D,1:T}$ and latent variables $z^{\text{what}}_{1:D}$ are inferred well. In Figure 4, we show the mean squared error between the video and its reconstruction for different T and D. The results confirm that performance improves with increasing number of Gibbs sweeps K. In certain cases, a larger number of time points T in fact improves convergence as a function of the number of sweeps K.

6 Conclusion

One of the challenges in amortized inference for deep generative models is learning high-quality proposals for models with a structured prior over a high-dimensional set of latent variables. These priors arise naturally when, rather than encoding a single data point (e.g. an image), we wish to encode a dataset (e.g. a sequence of images). Even for apparently simple problems, such as inferring the cluster parameters and assignments in a mixture model, standard encoders often fail to produce good samples. One of the reasons for this is that it is fundamentally difficult to jointly generate proposals for a high-dimensional set of latent variables.

APG samplers are very general, and offer a path towards the development of deep generative models that incorporate structured priors to provide meaningful inductive biases in settings where we have little or no supervision. These methods have particular strengths in problems with global variables, but more generally make it possible to design amortized approaches that exploit conditional independence. Moreover, our parameterization in terms of neural sufficient statistics makes it comparatively easy to design models that scale to much larger number of latent variables and thus generalize to datasets that vary in size.

Immediate lines of future work are to compare the approach in this paper, which learns kernels that leave the target density invariant, with approaches that perform annealing, in which the learned kernels are assymmetric in the sense that they gradually transform the initial encoder distribution to the target density.

7 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

[1] Dustin Tran, Alp Kucukelbir, Adji B. Dieng, Maja Rudolph, Dawen Liang, and David M. Blei. "Edward: A Library for Probabilistic Modeling, Inference, and Criticism". In: *arXiv:1610.09787 [cs, stat]* (Oct. 2016). arXiv: 1610.09787 [cs, stat].

[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". en. In: (Oct. 2018).

[3] N. Siddharth, Brooks Paige, Jan-Willem van de Meent, Alban Desmaison, Noah D. Goodman, Pushmeet Kohli, Frank Wood, and Philip Torr. "Learning Disentangled Representations with Semi-Supervised Deep Generative Models". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. 2017, pp. 5927–5937.

[4] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *International Conference on Learning Representations* (2013).

[5] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. "Stochastic Backpropagation and Approximate Inference in Deep Generative Models". In: *Proceedings of the 31st International Conference on Machine Learning*. Ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research 2. Bejing, China: PMLR, June 2014, pp. 1278–1286.

[6] 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 29*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett. Curran Associates, Inc., 2016, pp. 3225–3233.

[7] Babak Esmaeili, Hongyi Huang, Byron Wallace, and Jan-Willem van de Meent. "Structured Neural Topic Models for Reviews". en. In: *Artificial Intelligence and Statistics*. 00000. Apr. 2019, pp. 3429–3439.

[8] Adam Kosiorek, Hyunjik Kim, Yee Whye Teh, and Ingmar Posner. "Sequential Attend, Infer, Repeat: Generative Modelling of Moving Objects". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Curran Associates, Inc., 2018, pp. 8606–8616.

[9] Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. "Monte Carlo Gradient Estimation in Machine Learning". In: *arXiv:1906.10652 [cs, math, stat]* (June 2019). 00000. arXiv: 1906.10652 [cs, math, stat].

[10] Matthew Johnson, David K. Duvenaud, Alex Wiltschko, Ryan P. Adams, and Sandeep R. Datta. "Composing Graphical Models with Neural Networks for Structured Representations and Fast Inference". In: *Advances in Neural Information Processing Systems*. 2016, pp. 2946–2954.

[11] Matthew J Beal. "Variational Algorithms for Approximate Bayesian Inference". PhD thesis. 2003.

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

[13] Martin J Wainwright and Michael I Jordan. "Graphical Models, Exponential Families, and Variational Inference". In: *Foundations and Trends in Machine Learning* 1.1–2 (2008), pp. 1–305. DOI: 10.1561/2200000001.

[14] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. "Importance Weighted Autoencoders". In: *International Conference on Representations*. 2016. arXiv: 1509.00519.

[15] Tuan Anh Le, Maximilian Igl, Tom Rainforth, Tom Jin, and Frank Wood. "Auto-Encoding Sequential Monte Carlo". In: *International Conference on Learning Representations*. 2018. arXiv: 1705.10306.

[16] Chris J Maddison, John Lawson, George Tucker, Nicolas Heess, Mohammad Norouzi, Andriy Mnih, Arnaud Doucet, and Yee Teh. "Filtering Variational Objectives". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., 2017, pp. 6573–6583.

[17] Christian Naesseth, Scott Linderman, Rajesh Ranganath, and David Blei. "Variational Sequential Monte Carlo". en. In: *International Conference on Artificial Intelligence and Statistics*. Mar. 2018, pp. 968–977.

[18] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. "Sequential Monte Carlo Samplers". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 68.3 (June 2006), pp. 411–436. DOI: 10.1111/j.1467-9868.2006.00553.x.

[19] Christian Naesseth, Fredrik Lindsten, and Thomas Schon. "Nested sequential monte carlo methods". In: *International Conference on Machine Learning*. 2015, pp. 1292–1301.

[20] Chin-Wei Huang, Shawn Tan, Alexandre Lacoste, and Aaron C Courville. "Improving explorability in variational inference with annealed variational objectives". In: *Advances in Neural Information Processing Systems*. 2018, pp. 9701–9711.

[21] Jörg Bornschein and Yoshua Bengio. "Reweighted wake-sleep". In: *arXiv preprint arXiv:1406.2751* (2014).

[22] Tuan Anh Le*, Adam R. Kosiorek*, N. Siddharth, Yee Whye Teh, and Frank Wood. "Revisiting Reweighted Wake-Sleep for Models with Stochastic Control Flow". In: *Uncertainty in Artificial Intelligence*. Le and Kosiorek contributed equally. 2019.

[23] Ronald J. Williams. "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning". In: *Machine Learning* 8.3-4 (1992), pp. 229–256. ISSN: 0885-6125. DOI: 10.1007/BF00992696.

[24] 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).

[25] Eric Jang, Shixiang Gu, and Ben Poole. "Categorical Reparametrization with Gumble-Softmax". In: *International Conference on Learning Representations 2017*. OpenReviews. net. 2017.

[26] Théophane Weber, Nicolas Heess, Lars Buesing, and David Silver. "Credit Assignment Techniques in Stochastic Computation Graphs". en. In: *arXiv:1901.01761 [cs, stat]* (Jan. 2019). arXiv: 1901.01761 [cs, stat].

[27] Andriy Mnih and Danilo Rezende. "Variational Inference for Monte Carlo Objectives". en. In: *International Conference on Machine Learning*. June 2016, pp. 2188–2196.

[28] 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*. 2017, pp. 2624–2633.

[29] 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).

[30] Arnaud Doucet, Nando Freitas, and Neil Gordon, eds. *Sequential Monte Carlo Methods in Practice*. New York, NY: Springer New York, 2001. ISBN: 978-1-4419-2887-0 978-1-4757-3437-9. DOI: 10.1007/978-1-4757-3437-9.

[31] Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of markov chain monte carlo*. CRC press, 2011.

[32] Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. "Auxiliary Deep Generative Models". In: *International Conference on Machine Learning*. 2016, pp. 1445–1453.

[33] Rajesh Ranganath, Dustin Tran, and David Blei. "Hierarchical variational models". In: *International Conference on Machine Learning*. 2016, pp. 324–333.

[34] Joseph Marino, Yisong Yue, and Stephan Mandt. "Iterative amortized inference". In: *International Conference on Machine Learning* (2018).

[35] Matthew D Hoffman. "Learning deep latent Gaussian models with Markov chain Monte Carlo". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 1510–1519.

[36] Yingzhen Li, Richard E Turner, and Qiang Liu. "Approximate inference with amortised mcmc". In: *arXiv preprint arXiv:1702.08343* (2017).

[37] Tim Salimans, Diederik Kingma, and Max Welling. "Markov chain monte carlo and variational inference: Bridging the gap". In: *International Conference on Machine Learning*. 2015, pp. 1218–1226.

[38] Tongzhou Wang, Yi Wu, Dave Moore, and Stuart J Russell. "Meta-learning MCMC proposals". In: *Advances in Neural Information Processing Systems*. 2018, pp. 4146–4156.

[39] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. "Unsupervised learning of video representations using lstms". In: *International conference on machine learning*. 2015, pp. 843–852.

[40] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. "Spatial transformer networks". In: *Advances in Neural Information Processing Systems*. 2015, pp. 2017–2025.

A Gradient of the generative model

This is actually a known (although indeed not obvious) identity. Briefly, we can 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, e.g., 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

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

which is Equation 6. As a result, we can then use self-normalized importance sampling to approximate $\mathbb{E}_{p_\theta(z|x)}\left[\nabla_\theta \log p_\theta(x,z)\right]$.

B Importance weights 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

$$w^k = \frac{\gamma^k(z^{1:k})}{q^1(z^1)\prod_{k'=2}^k q^{k'}(z^{k'} \mid z^{1:k'-1})}. \tag{12}$$

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 $k \geq 2$, assume the weight has the form in Equation 12, 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 12 at each step $k > 2$ in sequential importance sampling.

C Derivation of Posterior Invariance

We can see that individual block updates leave the posterior invariant by proposing variables $z^k_{\preceq b}$ from a partial kernel $\kappa(z^k_{\preceq b} \mid x, z^{k-1})$ and then marginalize over the corresponding variables from the previous step $z^{k-1}_{\preceq b}$,

$$\begin{aligned}
\int dz^{k-1}_{\preceq b}\, p_\theta(z^{k-1} \mid x)\, \kappa(z^k_{\preceq b} \mid x, z^{k-1}) &= \int dz^{k-1}_{\preceq b}\, p_\theta(z^{k-1} \mid x) \int dz^k_{\succ b}\, \kappa(z^k \mid x, z^{k-1}) \\
&= \int dz^{k-1}_{\preceq b}\, p_\theta(z^{k-1} \mid x) \prod_{m=1}^{b} p_\theta(z^k_m \mid x, z^k_{\prec m}, z^{k-1}_{\succ m}) \\
&= \int dz^{k-1}_{\preceq b}\, p_\theta(z^{k-1} \mid x) p_\theta(z^k_{\preceq b} \mid x, z^{k-1}_{\succ 1}) \\
&= p_\theta(z^k_{\preceq b}\,,\, z^{k-1}_{\succ b} \mid x).
\end{aligned}$$

D Proof of the amortized population Gibbs algorithm

Here, we provide an alternative proof of correctness of the APG algorithm given in Algorithm 3, based on the construction of proper weights [19] which was introduced after SMC samplers [18]. 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.

D.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

$$\mathbb{E}_{P(z,w)}\left[wf(z)\right] = Z_p \mathbb{E}_{p(z)}[f(z)]. \tag{13}$$

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 (13). 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

$$\mathbb{E}_{p(z)}[f(z)] \approx \sum_{l=1}^{L} \bar{w}^l f(z^l), \tag{14}$$

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

D.2 Operations that preserve proper weights

Proposition 1 (Nested importance sampling)**.** *Adapted from [19, Algorithm 1]. 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*

$$z, w \overset{\text{p.w.}}{\sim} \tilde{q}, \tag{15}$$

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)**.** *Adapted from [19, Section 3.1]. Given an unnormalized density $\tilde{p}(z)$ (normalizing constant Z_p, normalized density $p(z)$), if we have a set of properly weighted samples*

$$z^l, w^l \overset{\text{p.w.}}{\sim} \tilde{p}, \quad l = 1, \dots, L \tag{16}$$

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, $\mathbb{E}[f(z^a)w'^l] = Z_p \mathbb{E}_{p(z)}[f(z)]$.

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

□

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 3).

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*

$$z^l, w^l \overset{\text{p.w.}}{\sim} \tilde{p}, \tag{17}$$

$$z'^l \sim q(z'^l | z^l), \tag{18}$$

$$w'^l = \frac{\tilde{p}(z'^l) r(z^l | z'^l)}{\tilde{p}(z^l) q(z'^l | z^l)} w^l, \qquad l = 1, \dots, L \tag{19}$$

then we have

$$z'^l, w'^l \overset{\text{p.w.}}{\sim} \tilde{p}, \qquad l = 1, \dots, L \tag{20}$$

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 (17), for any measurable $f(z)$, we have

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

To prove (20), 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:

$$\begin{aligned}
\mathbb{E}_{\hat{P}(z,w)q(z'|z)}[w'f(z')] &= \mathbb{E}_{\hat{P}(z,w)q(z'|z)}\left[\frac{\tilde{p}(z')r(z|z')}{\tilde{p}(z)q(z'|z)}wf(z')\right] \\
&= \int \hat{P}(z,w)q(z'|z)\frac{\tilde{p}(z')r(z|z')}{\tilde{p}(z)q(z'|z)}wf(z')\,\mathrm{d}z\,\mathrm{d}w\,\mathrm{d}z' \\
&= \int \hat{P}(z,w)\frac{\tilde{p}(z')r(z|z')}{\tilde{p}(z)}wf(z')\,\mathrm{d}z\,\mathrm{d}w\,\mathrm{d}z' \\
&= \int \tilde{p}(z')f(z')\left(\int \hat{P}(z,w)w\frac{r(z|z')}{\tilde{p}(z)}\,\mathrm{d}z\,\mathrm{d}w\right)\mathrm{d}z' \\
&= \int \tilde{p}(z')f(z')Z_p\mathbb{E}_{p(z)}\left[\frac{r(z|z')}{\tilde{p}(z)}\right]\mathrm{d}z'. \qquad (21)
\end{aligned}$$

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 21 simplifies to

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

□

D.3 Correctness of APG Sampler

We provide the proof by performing 2 steps in the APG sampler (Algorithm 3), i.e, we prove the correctness when we initialize samples at step $k = 1$ (line 2 - line 6) and then do one Gibbs sweep at step $k = 2$ (line 8 - line 17). 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 3) and train that encoder using the wake-ϕ phase objective in the standard reweighted wake-sleep[22] $\mathbb{E}_{p(x)}\left[\mathrm{KL}\left(p_\theta(z|x)||q_\phi(z|x)\right)\right]$. Then we estimate its gradient w.r.t. parameter ϕ (line 5) as

$$\begin{aligned}
g_\phi :&= -\nabla_\phi\,\mathbb{E}_{p(x)}\left[\mathrm{KL}\left(p_\theta(z|x)||q_\phi(z|x)\right)\right] &(22)\\
&= \mathbb{E}_{p(x)}\left[\mathbb{E}_{p_\theta(z|x)}\left[\nabla_\phi \log q_\phi(z \mid x)\right]\right]. &(23)
\end{aligned}$$

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

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

And we will prove that we correctly estimate the following gradient w.r.t. parameter ϕ at each block update (line 15)

$$\begin{aligned}
g_\phi^b :&= -\nabla_\phi\mathbb{E}_{p(x)}\left[\mathbb{E}_{p_\theta(z_{-b}|x)}\left[\mathrm{KL}\left(p_\theta(z_b|z_{-b},x)||q_\phi(z_b|z_{-b},x)\right)\right]\right] &(24)\\
&= \mathbb{E}_{p(x)}\left[\mathbb{E}_{p_\theta(z_{1:B}|x)}\left[\nabla_\phi \log q_\phi(z_b|z_{-b},x)\right]\right], \qquad b = 1,\ldots,B. &(25)
\end{aligned}$$

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

$$z_{1:B}^l, w^l \overset{\text{p.w.}}{\sim} p_\theta(z_{1:B}, x), \qquad l = 1,\ldots,L. \qquad (26)$$

Equation 14 will guarantee the validity of both gradient estimations (line 5 and line 15).

At step $k = 1$, samples are properly weighted because z^l and w^l are proposed using importance sampling (line 4) where $q_\phi(z|x)$ is the proposal density and $p_\theta(z^l, x)$ is the unnormalized target density. The resampling step (line 10) will preserve the proper weighting because of Proposition 2.

To prove that Gibbs sweep (line 8 - line 17) in the APG sampler also preserves proper weighting, we show that each block update satisfies all the 3 conditions (Equation 17, 19 and 26) 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 12), we already know that samples are properly weighted, i.e.

$$z, w \overset{\text{p.w.}}{\sim} p_\theta(z, x). \tag{27}$$

which corresponds Equation 17. 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

$$z' \sim q_\phi(z'_b|x, z_{-b})\delta_{z_{-b}}(z'_{-b}), \tag{28}$$

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 12) and let $z'_{-b} = z_{-b}$, which is exactly what the APG sampler assumes procedurally. This condition corresponds to Equation 18.

Finally, we define the weight w'

$$w' = \frac{p_\theta(x, z'_b, z'_{-b})r(z_b|x, z_{-b})\delta_{z_{-b}}(z_{-b})}{p_\theta(x, z_b, z_{-b})q_\phi(z'_b|x, z_{-b})\delta_{z_{-b}}(z'_{-b})}w, \tag{29}$$

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 13). This condition corresponds to Equation 19.

Now we can apply the conclusion (20) 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 14). 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 5 and line 15) are correct.

E Architecture of the Amortized Population Gibbs samplers

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)

DGMM

Layer	$q_\phi(\mu\|x)$	
Input	$x_n \in \mathbb{R}^2$	
1	FC 32 Tanh	FC 32 Tanh
2	FC 16 Tanh, $v_n \in \mathbb{R}$	FC 4 Softmax, $\gamma_n \in \mathbb{R}^3$
3	$T_n := \gamma_n \otimes v_n \in \mathbb{R}^{3\times 16}$	
4	$\text{Concat}[\sum_n^N T_n[i], \mu_0 \in \mathbb{R}^2, \text{Diag}(\sigma_0^2 I) \in \mathbb{R}^2], i = 1, 2, 3, 4$	
5	FC 2×32 Tanh	
6	FC 2×8 ($\mu_{1:I}$)	

Layer	$q_\phi(\mu\|z, c)$	
Input	$\text{Concat}[x_n \in \mathbb{R}^2, c_n \in \mathbb{R}^3]$	
1	FC 32 Tanh	FC 32 Tanh
2	FC 16, $v_n \in \mathbb{R}$	FC 4 Softmax, $\gamma_n \in \mathbb{R}^3$
3	$T_n := \gamma_n \otimes v_n \in \mathbb{R}^{3\times 16}$	
4	$\text{Concat}[\sum_n^N T_n[i], \mu_0 \in \mathbb{R}^2, \text{Diag}(\sigma_0^2 I) \in \mathbb{R}^2], i = 1, 2, 3, 4$	
5	FC 2×32 Tanh	
6	FC 2×2 (μ_i)	

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

Layer	$q_\phi(\alpha\|x, z, \mu)$
Input	$x_n - \mu_i \in \mathbb{R}^2 \| z_n = i$
1	FC 32 Tanh
2	FC 1 Tanh

Layer	$p_q(x\|\mu, c, \alpha)$
Input	$\text{Concat}[\alpha_n, c_n] \in \mathbb{R}^5$
1	FC 32 Tanh
2	FC 2 Tanh (μ_n, fixed σ_ϵ)

Bouncing MNIST

Layer	$p_\theta(x\|z^{\text{what}}, z^{\text{where}})$
Input	$z_i^{\text{what}} \in \mathbb{R}^{10}$
1	FC 200 ReLU
2	FC 400 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	$q_\phi(z^{\text{what}}\|z^{\text{where}})$
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 400 ReLU
3	FC 200 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	$q_\phi(z^{\text{where}}\|z^{\text{what}})$
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 400 Tanh
3	2$\times$ FC 200 Tanh
4	2×2 Tanh

F More Qualitative Results of Bouncing MNIST

The following are full reconstructions on test sets where time steps $T = 100$ and number of digits $D = 3, 4, 5, 6$, respectively. In each figure, the 1st, 3rd, 5th, 7th, 9th rows show the inference results, while the other rows show the reconstruction of the series above.

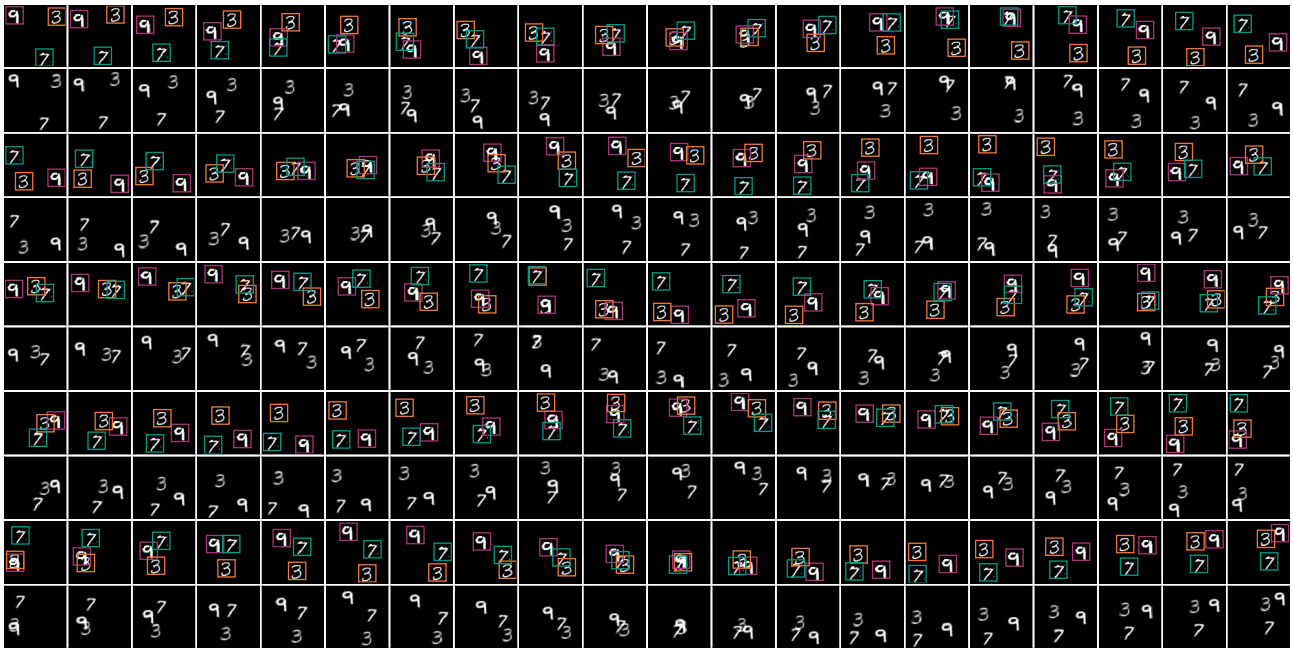


Figure 5: Full reconstruction for a video where $T = 100, D = 3$.

Figure 6: Full reconstruction for a video where $T = 100, D = 4$.

Figure 7: Full reconstruction for a video where $T = 100, D = 5$.

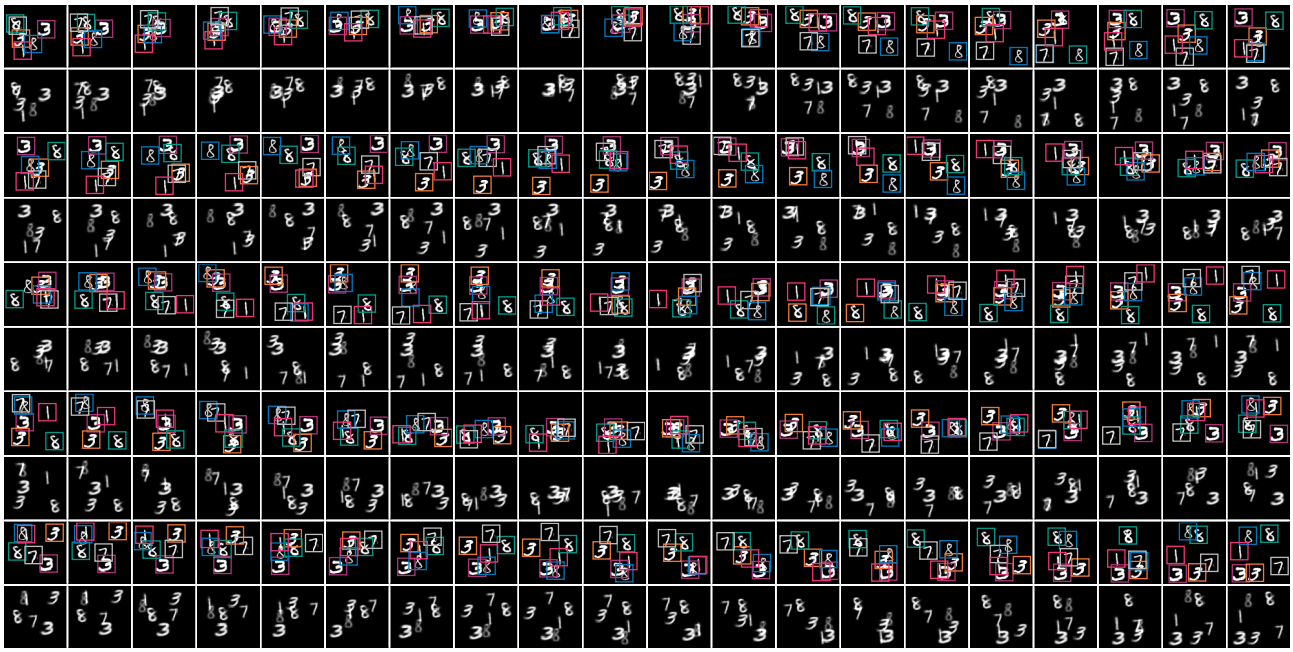


Figure 8: Full reconstruction for a video where $T = 100, D = 6$.