# Amortized Population Gibbs Samplers with Neural Sufficient Statistics


## Abstract

We develop amortized population Gibbs samplers, a new class of autoencoding variational methods for deep probabilistic models. These methods construct high-quality proposals by iterating between updates to blocks of variables, which each approximate a conditional posterior by minimizing an inclusive Kullback-Leibler divergence. We develop a new parameterization in terms of neural sufficient statistics, resulting in quasi-conjugate variational approximations that appropriately account for the size of the input data. 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 $\leq 2$ objects over the course of 10 frames [8], or assigning $\leq 10$ 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 computation-


Preliminary work. Under review by AISTATS 2020. Do not distribute.

ally efficient; variational EM can often converge in a small number of iterations and easily scales to 100s 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 [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 $\log q_\phi(z_b \mid z_{-b}, x)$ 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*

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

When this condition is met, the proposal $q_\phi(z^1, \ldots z^K \,|\, x)$ 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-specfied number of updates $K$.

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

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

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 $q_\phi(z^1, \dots z^K \mid x)$ 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 $\mathcal{K}_b(\phi)$

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

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 $p_\theta(z_b \mid x, z_{-b})$. As we will discuss in Section 5, this has implications for the evaluation of learned proposals. However, it is possible to approximate the gradient of the objective

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

with any Monte Carlo method for approximate inference that generates samples $z \sim p_\theta(z \mid x)$ from the posterior.

Approximating the gradient presents a chicken-and-egg problem; we need samples from the posterior to compute a Monte Carlo estimate of the gradient, which is needed to learn a proposals for in the first place. To work around this problem, we will use the proposals to define an importance sampler compute a self-normalized estimator of the gradinet from weighted samples $\{(w^l, z^l)\}_{l=1}^L$,

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

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

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

where the first equality holds by due to the standard identity $\mathbb{E}_{p_\theta(z|x)}[\nabla_\theta \log p_\theta(z|x)] = 0$ (see Appendix B).

The estimator in Equation 5 is similar to the self-normalized estimator in reweighted wake-sleep methods [21], which also minimize an inclusive KL divergence. These estimators have a number of advantages over standard VAE objectives, which minimize an exclusive KL divergence [22]. Standard VAE objectives rely on reparameterization to compute gradient estimates. For discrete variables, we typically 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

**Algorithm 1** SMC sampler

**for** $l = 1, \dots, L$ **do**
  $y_0^l \sim q_0(y_0)$. ▷ propose
  $w_0^l = \frac{\gamma_0(y_0^l)}{q_0(y_0^l)}$. ▷ weigh
**for** $b$ **in** $1, \dots, B$ **do**
  $y_{b-1}^{1:L}, w_{b-1}^{1:L} = \textsc{resample}(y_{b-1}^{1:L}, w_{b-1}^{1:L})$. ▷ resample
  **for** $l = 1, \dots, L$ **do**
    $y_b^l \sim q_b(\cdot \mid y_{b-1}^l)$. ▷ propose
    $w_b^l = \frac{\gamma_b(y_b^l) r_{b-1}(y_{b-1}^l \mid y_b^l)}{\gamma_{b-1}(y_{b-1}^l) q_b(y_b^l \mid y_{b-1}^l)} w_{b-1}^l$. ▷ weigh

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 only requires that the gradient of the proposal density is computable, which is a milder condition that holds for most distributions of interest, including those over discrete variables. Moreover, since this gradient miminizes the inclusive KL divergence, and not the exclusive KL divergence, there is less of a risk of learning a proposal that collapses to a single mode of a multi-modal posterior [22].

### 2.2 Generating High Quality Samples

Using importance sampling to compute gradient estimators has one fundamental limitation. Self-normalized importance samplers are consistent, but they are not unbiased. A standard reweighted wake-sleep method proposes from an encoder $z \sim q_\phi(z \mid x)$ and computes weights $w = p_\theta(x, z)/q_\phi(z \mid x)$. In models with high-dimensional and/or correlated latent variables, these weights can have a high variance, resulting in a high bias of the estimator. This is particularly true during early stages of training, when the randomly initialized encoder will be a very poor approximation of the posterior.

In order to generate sufficiently high-quality samples, we will use a sequential Monte Carlo (SMC) sampler [18]. SMC methods reduce the variance of importance weights by decomposing a high-dimensional sampling problem into a sequence of lower-dimensional problems. SMC *samplers* (Algorithm 1) are a subclass of SMC methods that iteratively improve a batch of proposals by applying a transition kernel. By defining a SMC sampler in which each kernel $q_\phi(z_b \mid x, z_{-b})$ is an approximate Gibbs update, we break the problem of generating a high-quality joint proposal into a sequence of sampling problems for individual blocks of variables, which each have a much lower dimensionality. We refer to this implementation of an SMC sampler as an amortized population Gibbs sampler.

The mechanism by which SMC methods reduce weight variance is the resampling step, which constructs a new set of equally weighted samples by selecting from the current samples with probability proportional to the weight. In the

**Algorithm 2** Amortized Population Gibbs Sampling

1: $g_\phi = 0, g_\theta = 0$. ▷ initialize gradient estimators.
2: **for** $l = 1, \ldots, L$ **do**
3: $\quad z^l_{1:B} \sim q_\phi(\cdot \mid x)$. ▷ Propose
4: $\quad w_l = \frac{p_\theta(z^l_{1:B}, x)}{q_\phi(z^l_{1:B} \mid x)}$. ▷ Weigh
5: $g_\phi = g_\phi + \sum_{l=1}^L \frac{w_l}{\sum_{i=1}^L w_i} \nabla_\phi \log q_\phi(z^l_{1:B} \mid x)$.
6: $g_\theta = g_\theta + \sum_{l=1}^L \frac{w_l}{\sum_{i=1}^L w_i} \nabla_\theta \log p_\theta(x, z^l_{1:B})$
7: **for** $b = 1, \ldots, B$ **do**
8: $\quad z^{1:L}_{1:B}, w_{1:L} = \text{RESAMPLE}(z^{1:L}_{1:B}, w_{1:L})$ ▷ Resample
9: $\quad$ **for** $l = 1, \ldots, L$ **do**
10: $\quad\quad z'^l_b \sim q_\phi(\cdot \mid z^l_{-b}, x)$. ▷ Propose
11: $\quad\quad w_l = \frac{p_\theta(z'^l_b, z^l_{-b}, x) q_\phi(z^l_b \mid z^l_{-b}, x)}{p_\theta(z^l_b, z^l_{-b}, x) q_\phi(z'^l_b \mid z^l_{-b}, x)} w_l$. ▷ Weigh
12: $\quad\quad z^l_b = z'^l_b$. ▷ Reassign
13: $\quad g_\phi = g_\phi + \sum_{l=1}^L \frac{w_l}{\sum_{i=1}^L w_i} \nabla_\phi \log q_\phi(z^l_b \mid z^l_{-b}, x)$
14: $\quad g_\theta = g_\theta + \sum_{l=1}^L \frac{w_l}{\sum_{i=1}^L w_i} \nabla_\theta \log p_\theta(x, z^l_{1:B})$
**return** $g_\phi$, $g_\theta$

context of APG methods, the resampling step ensures that each block update is strongly correlated with the sampled value $z_b$. In general, the importance weight of a sample will depend on both $z_b$ and the values $z_{-b}$ that were sampled using previous updates. This implies that a bad proposal for $z_{-b}$ can negatively influence the overall importance weight.

To mitigate this effect, we the APG sampler incorporates a resampling step (line 8) after every block update $q_\phi(z_b \mid x, z_{-b})$. This ensures that incoming samples $z^l_{-b}$ are equally weighted and that the importance weight is proportional to the incremental weight (line 11). We empirically observed that this procedure results in much better gradient signals for the individual block proposals than a sequential importance sampling procedure in which the weights depend on the entire sample history.

We can interpret APG as an instance of the SMC sampler algorithm [18] (Algorithm 1) which justifies using $\{(\tilde{w}^l, \tilde{z}^l_{1:B})\}_{l=1}^L$ (Algorithm 2) to approximate the posterior $p_\theta(z_{1:B} \mid x)$ as is required to minimize the variational objective $\mathcal{K}_b(\phi)$ in (4). SMC samplers approximate a sequence of target distributions $\pi_b(y_b)$ $(b = 0, \ldots, B)$ for which we can only evaluate unnormalized densities $\gamma_b(y_b) \propto \pi_b(y_b)$. Given a set of forward kernels $q_0(y_0)$, $q_b(y_b \mid y_{b-1})$ $(b = 1, \ldots, B)$ and reverse kernels $r_{b-1}(y_{b-1} \mid y_b)$ $(b = 1, \ldots, B)$, the algorithm proceeds by iterating over a sequence of propose-weigh-resample steps. At each step $b$, the weighted set $\{(w^l_b, y^l_b)\}_{l=1}^L$ can be used to estimate an expectation of test functions $f$ under $\pi_b$.

In APG, we define $y_b$ as the sequence of $z_{1:B}$ at every iteration of the for loop in line 7 (Algorithm 2). We define the unnormalized target densities $\gamma_b(y_b) = p_\theta(z_{1:B}, x)$ which normalize as $\pi_b(y_b) = p_\theta(z_{1:B} \mid x)$. The forward and reverse kernels are defined as

$$q_0(y_0) = q_\phi(z_{1:B} \mid x),$$
$$q_b(y_b \mid y_{b-1}) = q_\phi(z'_b \mid x, z_{-b}) \delta_{z_{-b}}(z'_{-b}), \quad (7)$$
$$r_{b-1}(y_{b-1} \mid y_b) = q_\phi(z_b \mid z_{-b}, x) \delta_{z_{-b}}(z_{-b}), \quad (8)$$

where $\delta_{z_{-b}}(z'_{-b})$ is a delta mass on $z_{-b}$ and is treated as a density of $z'_{-b}$ which evaluates to 1 if $z_{-b} = z'_{-b}$ and 0 otherwise. Similarly, $\delta_{z_{-b}}(z_{-b})$ is a density of $z_{-b}$. In APG, we propose $y_b = z'_{1:B}$ from the forward kernel $q_b$ by sampling $z'_b$ from $q_\phi(z'_b \mid x, z_{-b})$ and setting $z'_{-b} = z_{-b}$ (line 7, Algorithm 2). The weigh step of APG directly corresponds to one in the SMC sampler since the delta mass densities in (7) and (8) evaluate to one. The reassignment step in line 12 of the APG algorithm recovers $y_b = z_{1:B}$. Our final gradient estimator also has a term for learning the initial proposal $q_\phi(z_{1:B} \mid x)$ in line 5 which is identical to the wake-$\phi$ estimator of RWS. In practice, the APG algorithm is run as an inner loop within a loop that runs for $K$ Gibbs sweeps for a given batch of $x \sim \hat{p}(x)$. In Appendix C, we provide an alternative proof of correctness of the APG algorithm using proper weights [19].

## 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 invididual 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 $\lambda$ and a vector of neural sufficient statistics $T_\phi(x, z_{-b})$

$$q_\phi(z_b \mid x, z_{-b}) = p(z_b \mid \lambda + T_\phi(x, z_{-b})). \tag{9}$$

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 $z = \{z^{\text{G}}, z^{\text{L}}\}$ into global and local variables $z^{\text{G}}$ and local variables $z^{\text{L}}$. The dimensionality of global variables is typically constant, whereas local variables $z^{\text{L}} = \{z^{\text{L}}_1, \ldots, z^{\text{L}}_N\}$ have a dimensionality that increases with the data $N$. For models with this structure, the local variables are typically conditionally independent $z^{\text{L}}_n \perp z^{\text{L}}_{-n} \mid x, z^{\text{G}}$, which means that we can parameterize the sufficient statistics as

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

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

There are many works that consider the combination of ideas from MCMC and amortized inference. Li et al. [30] consider learning an inference network which is used for initializing an MCMC chain. They focus on a setting where the density of the inference network cannot be evaluated. Learning is also performed by targeting the inclusive KL. However, the posterior expectation is approximated using MCMC and, due to the inability to compute the density, the gradient is estimated adversarially. Like us, Wang et al. [31] considered the problem of learning block conditionals for performing Gibbs updates. In their case, the model parameters are not learned and hence the gradient of the inclusive KL divergence can be estimated by directly sampling from the generative model, akin to the sleep-phase of wake-sleep. They consider this in the context of meta-learning in which they learn a family of Gibbs conditionals amortizing inference in a family of models.

Our work can also be viewed as combining MCMC and variational inference. Salimans et al. [32] define a variational lower bound to learn the marginal $q_\phi(z^K \mid x)$ to approximate the posterior by introducing auxiliary variables $z^{1:K-1}$ in the generative model. This requires amortizing inference in an extended space as well as learning the auxiliary model distribution. Learning relies on the reparameterization trick which further limits the model family. This idea has also been studied in the context of deep generative modeling [33] and variational inference [34] where the goal is to increase the expressivity of the inference network. Like in our setting, it is hard to evaluate the quality of $q_\phi(z^K \mid x)$ due to high-dimensional marginalization. Hoffman [35] uses MCMC as means for better estimation of the $\theta$ gradient given in (6) while learning inference using the typical ELBO objective.

Our work can also be viewed as learning to refine a one-shot inference network. In this sense, it is similar to [36] which learns deterministic transformations of the inference network parameters by taking gradients through the optimization process. Huang et al. [20] also learn a sequence of transitions, however their goal is to improve explorability in the inference network and during test time, their inference network must be run for a fixed number of steps.

## 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 parameterize 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. [37] 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

Table 1: APG performance in the GMM and DGMM. For the GMM we report the inclusive and exclusive KL relative to the ground-truth conditional posteriors. We additionally report the increase in log joint $\Delta \log p_\theta(x, z)$ relative to baseline methods (MLP for GMM, LSTM for DGMM) and the ESS/L for joint sweep updates and individual block updates.

(a) **GMM**

| | $\Delta \log p_\theta(x,z)$ | ESS/L | | | 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 | | |
|---|---|---|---|---|
| | | $\{\mu, c, \alpha\}$ | $\{\mu\}$ | $\{c, \alpha\}$ |
| MLP-RWS | 2538 | 0.001 | – | – |
| LSTM-RWS | – | 0.001 | – | – |
| APG (K=5) | 6201 | 0.002 | 0.013 | 0.422 |
| APG (K=10) | 6293 | 0.002 | 0.019 | 0.454 |
| APG (K=15) | 6310 | 0.003 | 0.025 | 0.488 |

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 variable,s which can be adapted to a variety of model classes with comparative ease.

### 5.1 Evaluation Metrics

When evaluating APG samplers we would like to quantify:

1. How similar learned proposals $q_\phi(z_b \mid x, z_{-b})$ are to the conditional posteriors $p_\theta(z_b \mid x, z_{-b})$.
2. How quickly the marginal $q_\phi(z^K \mid x)$ for the final sample converges to the posterior $p_\theta(z^K \mid x)$ as a function of the number of sweeps $K$.

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 (see in Table 1) between the posterior and the learned proposal. In both mixture models, we additional compute the effective sample size as a means of assesing proposal quality

$$\frac{\text{ESS}}{L} = \frac{(\sum_{l=1}^{L} w^{k,l})^2}{L \sum_{l=1}^{L} (w^{k,l})^2} \tag{10}$$

and the log-joint distritbution $\log p_\theta(x, z)$ as a function of sweep iteration as an indirect measure of convergence.

### 5.2 Gaussian Mixture Model

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

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

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 pointwise neural sufficient statistics modeled after the ones in the analytical updates (for details on these architecture see Appendix D).

We compare the APG sampler to samples from a standard encoder with MLP and LSTM architectures, which is trained using reweighted wake-sleep (RWS). Both architectures are parameterized using the same neural sufficient statistics as the APG sampler.

We train our models on 20,000 unique datasets with $I = 3$ clusters and $M = 60$ data points with fixed hyperparameters ($\mu_0 = 0$, $\nu_0 = 0.3$, $\alpha_0 = 2$, $\beta_0 = 2$). We use 20 GMM

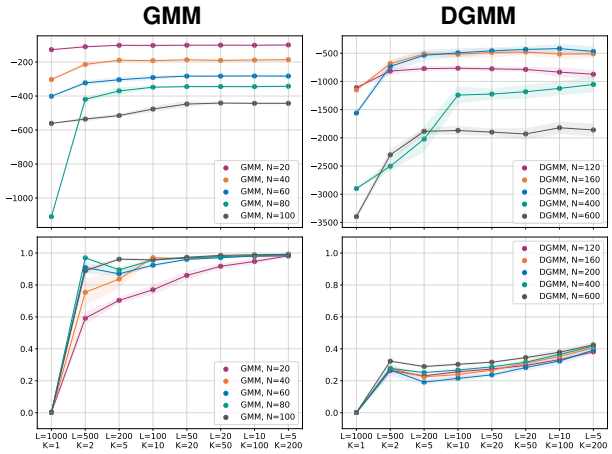
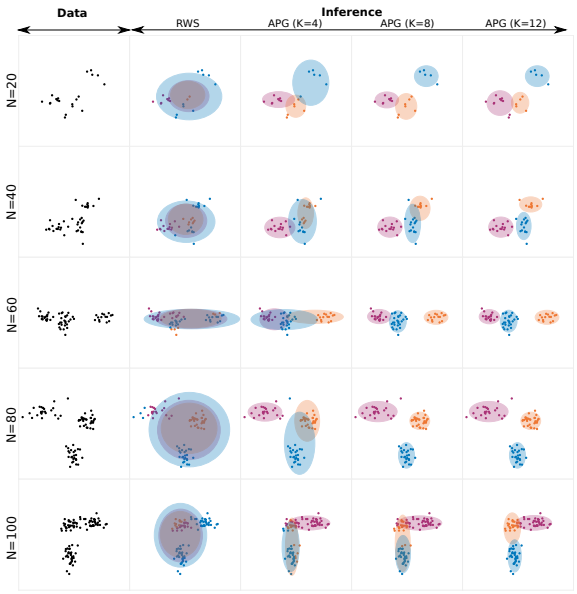
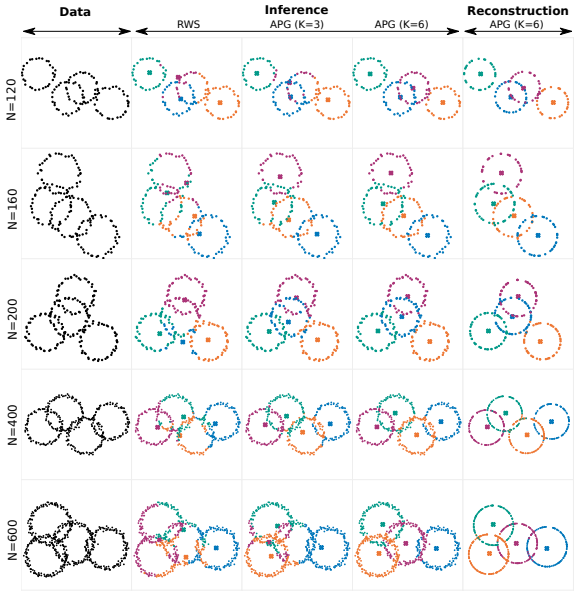



Figure 1: APG sampler performance as a function of number of sweeps $K$ for a constant sample budget $K \cdot L = 1000$. **Top**: Log joint $\log p_\theta(x, z)$. **Bottom**: ESS / L.




(a) GMM




(b) DGMM

Figure 2: 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. Middle 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.

datasets per batch, $K = 10$ sweeps for each dataset and $L = 10$ particles. We train using Adam (lr $= 10^{-4}, \beta_1 = 0.9, \beta_2 = 0.99$) for 200,000 iterations.

Figure 2a 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.

### 5.3 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

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

Here $\mu_i$ is center of the $i$th ring. Given a cluster assignment $c_n$ and an angle $\alpha_n$ we define a position on a ring, from which We sample the datapoint $x_n$ with 2D Gaussian noise.

We generate 20,000 datasets with $N = 200$ data points and $I = 4$ clusters with fixed hyperparameters ($\sigma_0 = 3.5$, $\Sigma_\epsilon = 0.2$). Once again, we compare the APG sampler with the RWSs (see Appendix D for encoder architecture).

Figure 2b shows individual samples analogous to the ones in Figure 2a. Once again the APG sampler scales to a large range of number of variables, whereas a standard encoder trained using RWS fails to produce reasonable proposals.

### 5.4 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 1 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 effect 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 $z_{1:D}^{\text{what}}$ corresponding to digit latent variables and local variables $z_{1:D,1:T}^{\text{where}}$ corresponding to the digit trajectories. The deep generative model is a state space model that

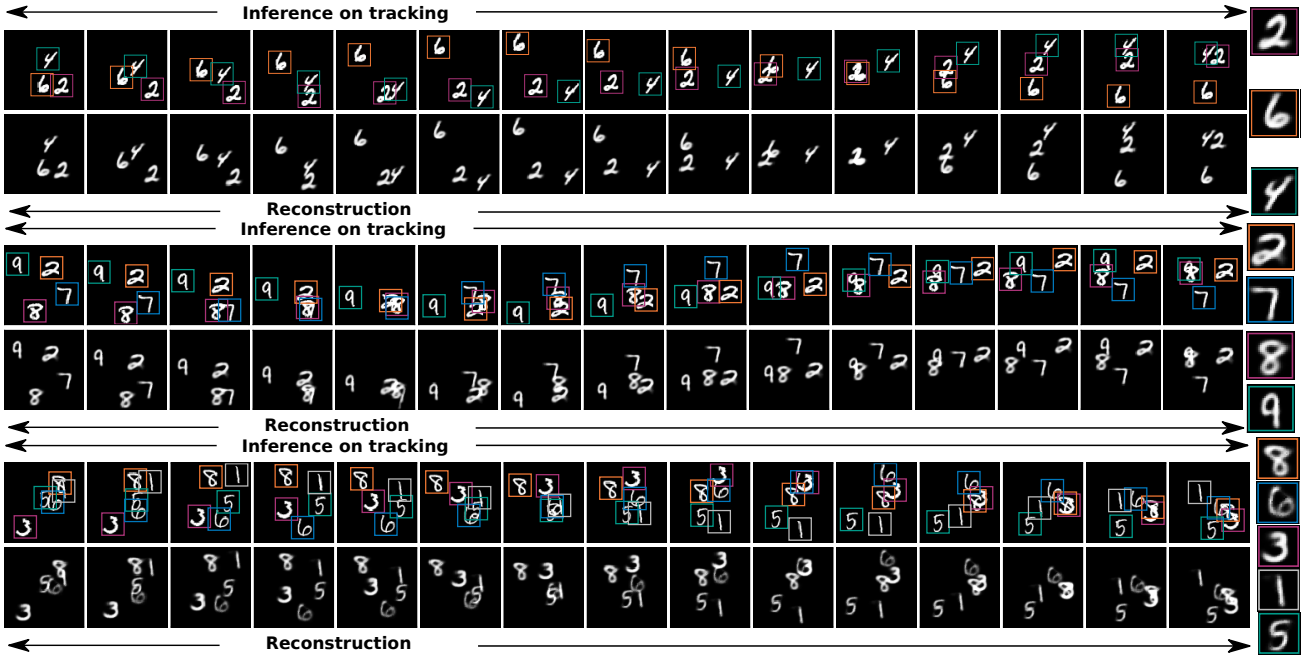



Figure 3: Inferred digit trajectories and reconstructions for (top) $D = 3$ (middle) $D = 4$ and (bottom) $D = 5$ digits for $T = 15$ for a model trained on $D = 3$ and $T = 10$.

factorizes across digits of the form

$$z_d^{\text{what}} \sim \text{Norm}(0, I), \quad z_{d,1}^{\text{where}} \sim \text{Norm}(0, I),$$
$$z_{d,t}^{\text{where}} \sim \text{Norm}(z_{d,t-1}^{\text{where}}, \sigma^2 I)$$
$$x_t \sim \text{Bern}\Big(\sigma\Big(\sum_d \text{ST}\big(\mu_\theta(z_d^{\text{what}}), z_{d,t}^{\text{where}}\big)\Big)\Big)$$

Here, ST is a spatial transformer [38] that maps the output of a feedforward decoder $\mu_\theta$ that maps logits for a 28×28 MNIST image onto a 96×96 canvas based on the location variable $z_{d,t}^{\text{where}}$. Our amortized Gibbs updates employ $T + 1$ blocks $(z_{1:D}^{\text{what}}, z_{1:D,1}^{\text{where}}, z_{1:D,2}^{\text{where}}, \ldots, z_{1:D,T}^{\text{where}})$. 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 show that APG can generalize to larger number of timesteps $T$ and digits $D$. We train the model on a dataset with $T = 10$ and $D = 3$ while we test on $T \in \{20, 100\}$ and $D \in \{3, 4, 5\}$. In Figure 3, we show inference and reconstruction based on a sequence of $T = 15$ images (cut due to space) for varying number of digits $D \in \{3, 4, 5\}$. Qualitatively, we see that the digit trajectories $z_{1:D,1:T}^{\text{where}}$ and latent variables $z_{1:D}^{\text{what}}$ 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 $K$.

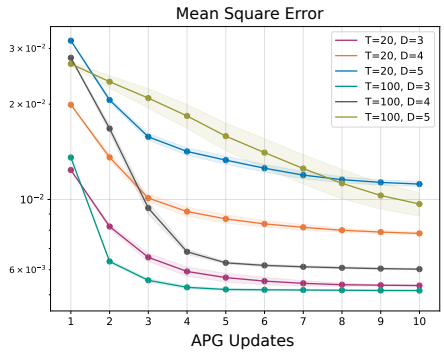



Figure 4: Mean squared error between video frames and reconstructions as a function of the number of APG sweeps.

## 6 Conclusion

We developed amortized Gibbs samplers that iterate between updates to global and local variables using neural proposals. These methods offer a path towards designing variational approximations to intractable posteriors in structured deep generative models. APG samplers have particular strengths in problems with global variables, but more generally make it possible to design amortized approaches that exploit conditional independence. This decomposes high-dimesional sampling problems into a sequence of lower-dimensional problems, hereby greatly reducing estimator variance. Moreover, our parameterization in terms of neural sufficient statistics makes it comparatively easy to design models that will generalize to datasets that vary in size.

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

[31] 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.

[32] 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.

[33] 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.

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

[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] Joseph Marino, Yisong Yue, and Stephan Mandt. "Iterative amortized inference". In: *arXiv preprint arXiv:1807.09356* (2018).

[37] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. "Unsupervised Learning of Video Representations Using Lstms". In: *International Conference on Machine Learning*. 2015, pp. 843–852.

[38] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. "Spatial transformer networks". In: *Advances in neural information processing systems*. 2015, pp. 2017–2025.

## A 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 $z^{k-1}_{\preceq b}$,

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

## B 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

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

Equation B then follows if we use self-normalized importance sampling to approximate $\mathbb{E}_{p_\theta(z|x)}\left[\nabla_\theta \log p_\theta(x, z)\right]$.

## C 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 [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.

### C.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 $f$

$$\mathbb{E}_{P(z,w)}\left[w f(z)\right] = Z_p \mathbb{E}_{p(z)}[f(z)]. \tag{11}$$

We will also denote this as

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

**Using proper weights.** Given independent samples $z_\ell, w_\ell \sim P$, we can estimate $Z_p$ by setting $f \equiv 1$:

$$Z_p \approx \frac{1}{L} \sum_{\ell=1}^{L} w_\ell.$$

This estimator is unbiased because it is a Monte Carlo estimator of the left hand side of (11). We can also estimate $\mathbb{E}_{p(z)}[f(z)]$ as

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

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

$$\mathbb{E}_{p(z)}[f(z)] \approx \sum_{\ell=1}^{L} \bar{w}_\ell f(z_\ell), \tag{12}$$

where $\bar{w}_\ell := w_\ell / \sum_{\ell=1}^{L} w_\ell$ is the normalized weight.

### C.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*

$$z, w \overset{\text{p.w.}}{\sim} \tilde{q}, \tag{13}$$

*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*

$$z_\ell, w_\ell \overset{\text{p.w.}}{\sim} \tilde{p}, \tag{14}$$

*then*

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

*where $\mathbb{P}(b = i) = w_i / \sum_\ell w_\ell$ and $w' := \frac{1}{L} \sum_{\ell=1}^{L} w_\ell$.*

*Proof.* Define the distribution of $z_\ell, w_\ell$ as $P$. We show that for any $f$, $\mathbb{E}[f(z_b)w'] = Z_p \mathbb{E}_{p(z)}[f(z)]$.

$$\begin{aligned}
\mathbb{E}_{(\prod_\ell P(z_\ell, w_\ell)) p(b|w_{1:L})}\left[f(z_b)w'\right] &= \mathbb{E}_{\prod_\ell P(z_\ell, w_\ell)}\left[\sum_{i=1}^{L} f(z_i) w' \mathbb{P}(b = i)\right] \\
&= \mathbb{E}_{\prod_\ell P(z_\ell, w_\ell)}\left[\sum_{i=1}^{L} f(z_i) w' \frac{w_i}{\sum_\ell w_\ell}\right] \\
&= \mathbb{E}_{\prod_\ell P(z_\ell, w_\ell)}\left[\frac{1}{L} \sum_{i=1}^{L} f(z_i) w_i\right] \\
&= \frac{1}{L} \sum_{i=1}^{L} \mathbb{E}_{P(z_i, w_i)}\left[f(z_i) w_i\right] = \frac{1}{L} \sum_{i=1}^{L} Z_p \mathbb{E}_{p(z)}[f(z)] = Z_p \mathbb{E}_{p(z)}[f(z)].
\end{aligned}$$

□

We define the following resampling operation which we will use in the APG algorithm. Due to Proposition 2, if $z_\ell, w_\ell \overset{\text{p.w.}}{\sim} \tilde{p}(z)$ then $z'_\ell, w'_\ell \overset{\text{p.w.}}{\sim} \tilde{p}(z')$.

**Algorithm 3** RESAMPLE

1: **Input:** $z_{1:L}, w_{1:L}$
2: **Output:** $z'_{1:L}, w'_{1:L}$
3: Sample ancestral indices $a_{1:L}$ such that $\mathbb{P}(a_\ell = i) = w_i / \sum_{\ell=1}^{L} w_\ell$.
4: Set $z'_\ell = z_{a_\ell}$.
5: Set $w'_\ell = \frac{1}{L} \sum_{\ell=1} w_\ell$
6: **Return** $z'_{1:L}, w'_{1:L}$.

**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 $r(z|z')$, if*

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

$$z' \sim q(z'|z), \tag{15b}$$

$$w' = \frac{\tilde{p}(z')r(z|z')}{\tilde{p}(z)q(z'|z)} w, \tag{15c}$$

then

$$z', w' \overset{\text{p.w.}}{\sim} \tilde{p}. \tag{16}$$

*Proof.* Define the distribution of $z, w$ as $P$. Then, due to (15a), for any measurable $f(z)$, we have

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

To prove (16), we show $\mathbb{E}_{P(z,w)q(z'|z)}[w'f(z')] = Z_p \mathbb{E}_{p(z')}[f(z')]$ for any $f$ as follows:

$$\begin{aligned}
\mathbb{E}_{P(z,w)q(z'|z)}[w'f(z')] &= \mathbb{E}_{P(z,w)q(z'|z)} \left[ \frac{\tilde{p}(z')r(z|z')}{\tilde{p}(z)q(z'|z)} wf(z') \right] \\
&= \int P(z,w)q(z'|z) \frac{\tilde{p}(z')r(z|z')}{\tilde{p}(z)q(z'|z)} wf(z') \,\mathrm{d}z \,\mathrm{d}w \,\mathrm{d}z' \\
&= \int P(z,w) \frac{\tilde{p}(z')r(z|z')}{\tilde{p}(z)} wf(z') \,\mathrm{d}z \,\mathrm{d}w \,\mathrm{d}z' \\
&= \int \tilde{p}(z')f(z') \left( \int P(z,w) w \frac{r(z|z')}{\tilde{p}(z)} \,\mathrm{d}z \,\mathrm{d}w \right) \mathrm{d}z' \\
&= \int \tilde{p}(z')f(z') Z_p \mathbb{E}_{p(z)} \left[ \frac{r(z|z')}{\tilde{p}(z)} \right] \mathrm{d}z'.
\end{aligned} \tag{17}$$

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 (17) simplifies to

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

□

### C.3 Correctness of APG

The point of the APG algorithm (Algorithm 2) is to estimate the gradient of the loss

$$\mathcal{L}(\phi) := \mathbb{E}_{p(x)} \left[ \mathrm{KL} \left( p_\theta(z_{1:B}|x) || q_\phi(z_{1:B}|x) \right) + \sum_{b=1}^{B} \mathbb{E}_{p_\theta(z_{-b}|x)} \left[ \mathrm{KL} \left( p_\theta(z_b|z_{-b}, x) || q_\phi(z_b|z_{-b}, x) \right) \right] \right] \tag{18}$$

in order to minimize it using SGD. The first term is for learning the initial proposal $q_\phi(z_{1:B}|x)$. The second term (sum over $B$ terms) is for learning the $B$ conditionals $q_\phi(z_b|z_{-b}, x)$.

Lines 2–5 accumulate the standard wake-$\phi$ estimator in reweighted wake-sleep. After line 5, $g_\phi$ estimates $\nabla_\phi \mathbb{E}_{p(x)} \left[ \mathrm{KL} \left( p_\theta(z_{1:B}|x) || q_\phi(z_{1:B}|x) \right) \right]$.

Now, we wish to show that line 13 of every iteration $b$ of the for loop (lines 7–13) accumulates an estimate of

$$g_\phi^b := \nabla_\phi \mathbb{E}_{p_\theta(z_{-b}|x)} \left[ \mathrm{KL} \left( p_\theta(z_b|z_{-b}, x) || q_\phi(z_b|z_{-b}, x) \right) \right] = \mathbb{E}_{p_\theta(z_{1:B}|x)} \left[ -\nabla_\phi \log q_\phi(z_b|z_{-b}, x) \right]. \tag{19}$$

Our strategy will be to show that we maintain the loop invariant

$$z_{1:B}^\ell, w_\ell \overset{\text{p.w.}}{\sim} p_\theta(z_{1:B}, x) \qquad \text{for } \ell \in 1 : L. \tag{20}$$

Following (12), this property allows us to estimate (19) as the sum in line 13.

Initially (start of line 8), (20) is true because $z_{1:B}^\ell$ and $w_\ell$ are obtained through importance sampling where $q_\phi(z_{1:B}|x)$ is the proposal density and $p_\theta(z_{1:B}^\ell, x)$ is the unnormalized target density. After line 8, (20) remains true because resampling preserves proper weights (see Proposition 2).

To prove that lines 9–12 preserve (20), we use Proposition 3 in the following way. Dropping all $\ell$ superscripts to avoid clutter, at the start of line 9, we have

$$z_{1:B}, w \overset{\text{p.w.}}{\sim} p_\theta(z_{1:B}, x). \tag{21}$$

This corresponds to (15a).

Next, we define the conditional distribution corresponding to $q(z'|z)$ in (15b) as

$$z'_{1:B} \sim q_\phi(z'_b|z_{-b}, x)\delta_{z_{-b}}(z'_{-b}), \tag{22}$$

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. Procedurally, we obtain a sample by just sampling $z'_b$ as in line 10 and treating $z_{-b}$ as $z'_{-b}$.

Next, we define the weight $w'$ corresponding to the weight in (15c) as

$$w' = \frac{p_\theta(z'_b, z'_{-b}, x) r(z_b|z_{-b}, x) \delta_{z_{-b}}(z_{-b})}{p_\theta(z_b, z_{-b}, x) q_\phi(z'_b|z_{-b}, x) \delta_{z_{-b}}(z'_{-b})} w, \tag{23}$$

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, line 11 evaluates this weight and reassigns it to $w_\ell$.

Finally, applying the (16) from Proposition 3, $z'_{1:B}, w' \overset{\text{p.w.}}{\sim} p_\theta(z'_{1:B}, x)$. Since $z_{-b} = z'_{-b}$ and $z_b = z'_b$ due to the assignment in line 12, we recover the invariant (20).

## D 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 |

**DGMM**

| Layer | $q_\phi(c\|x,\mu,\tau)$ |
|---|---|
| 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** | $q_\phi(\mu\|x)$ | |
|---|---|---|
| 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 ($\mu_i$) | |

| **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 ($\mu_n$, fixed $\sigma_\epsilon$) |

**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 | $2 \times 2$ Tanh |

## E More Qualitative Results of Bouncing MNIST

The following are full reconstructions on test sets where timesteps $T = 100$ and number of digits $D = 5, 6, 7$, 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 $T = 100, D = 5$.


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

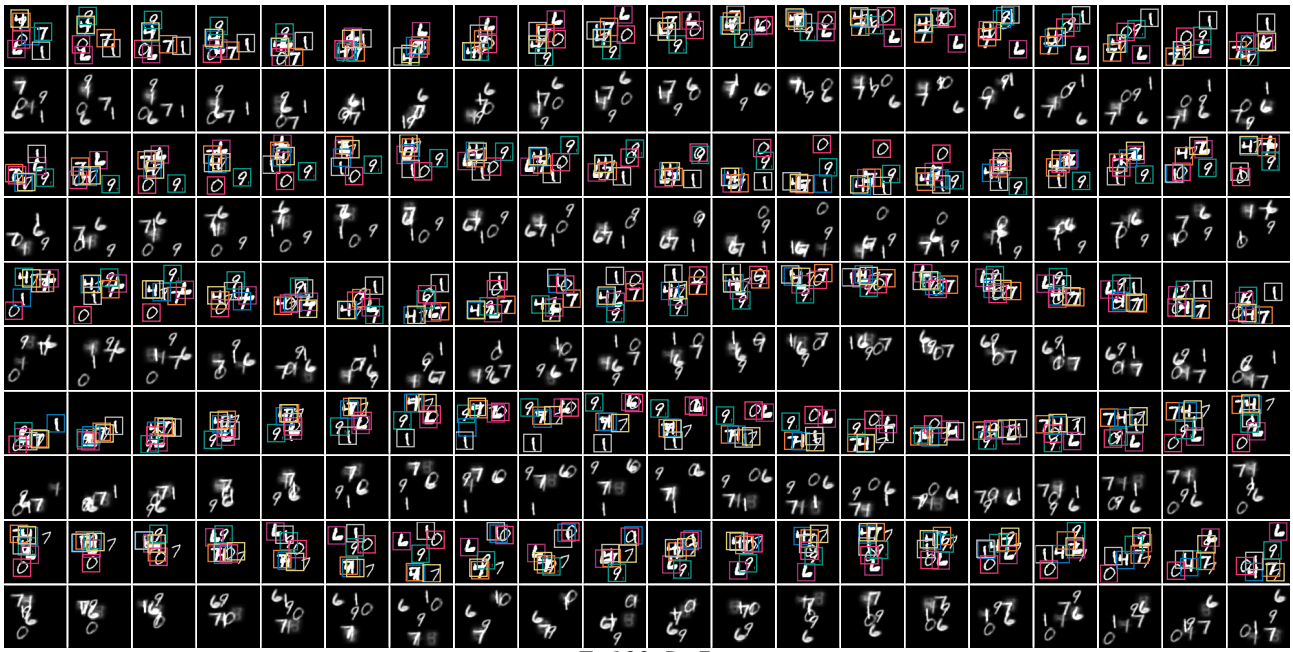



Figure 7: Full reconstruction for a video where $T = 100, D = 7$. The APG samplers start to lose track of at least 1 digit in some frames, when the video contains more than 7 digits.