Automatic structured variational inference

Luca Ambrogioni [1] **Max Hinne** [1] **Marcel van Gerven** [1]

Abstract

The aim of probabilistic programming is to automatize every aspect of probabilistic inference in arbitrary probabilistic models (programs) so that the user can focus her attention on modeling, without dealing with *ad-hoc* inference methods. Gradient based automatic differentiation stochastic variational inference offers an attractive option as the default method for (differentiable) probabilistic programming as it combines high performance with high computational efficiency. However, the performance of any (parametric) variational approach depends on the choice of an appropriate variational family. Here, we introduced a fully automatic method for constructing structured variational families inspired to the closed-form update in conjugate models. These pseudo-conjugate families incorporate the forward pass of the input probabilistic program and can capture complex statistical dependencies. Pseudo-conjugate families have the same space and time complexity of the input probabilistic program and are therefore tractable in a very large class of models. We validate our automatic variational method on a wide range of high dimensional inference problems including deep learning components.

1. Introduction

Probabilistic programming is concerned with the symbolic specification of probabilistic models in which inference can be performed automatically. Stochastic gradient-based variational methods are gradually replacing MCMC as the default inference technique in (differentiable) probabilistic programming languages (Wingate & Weber, 2013; Kucukelbir et al., 2017; Tran et al., 2016; Kucukelbir et al., 2015; Bingham et al., 2019). This trend is a consequence of the increasing automatization of variational inference (VI) techniques, which passed from being highly mathematically sophisticated and model-specific tools to generic algorithms that can be applied to a broad class of problems without model-specific derivations (Hoffman et al., 2013; Kingma & Welling, 2013; Hernández-Lobato & Adams, 2015; Ranganath et al., 2014). However, applications of VI relies on the choice of a parameterized variational family and this reliance on user input arguably violates the spirit of probabilistic programming. In general, it is relatively easy to automatize the construction of the variational family under the mean-field approximation, where the approximate posterior distribution factorizes as a product of univariate distributions. While there is a substantial amount of model-specific research on structured variational families, few existing methods can be used for automatically constructing an appropriate scalable structured variational approximation for an arbitrary chosen probabilistic model. Furthermore, these existing methods either ignore most of the prior structure of the model (e.g. ADVI with multivariate Gaussian distribution (Kucukelbir et al., 2017)) or require strict assumptions such as local conjugacy (e.g. structured stochastic VI (Hoffman & Blei, 2015)). Furthermore, several of these methods require the use of *ad hoc* gradient estimators or variational lower bounds (Tran et al., 2015; Ranganath et al., 2016).

In this paper, we introduce an automatic procedure for constructing variariational approximations that incorporate the structure (forward pass) of the probabilistic model while being flexible enough to capture the distribution of the observed data. The construction of these variational approximations is fully automatic and the resulting variational distribution has the same time/memory complexity of the input probabilistic program. The new family of variational models, which we call *pseudo-conjugate variational families*, interpolate the evidence coming from the observed data with the probabilistic structure of the prior model. Specifically, the parameters of the posterior distribution of each latent variable is a convex combination of the parameters induced by the probabilistic program and a term reflecting the influence of the data. This mimics the evidence update in the expectation parameters of conjugate exponential family models, where this posterior form is exact. Pseudo-conjugate variational families can be trained using used standard inference techniques and gradient estimators and can therefore be used as drop-in replacement of the mean-field approach in automatic differentiation stochastic VI. We call this new form of fully automatic inference as *automatic structured variational inference* (ASVI).

2. Background on variational inference and exact parameter update

VI is used to approximate the posterior over the latent variables of a probabilistic program $p(\boldsymbol{x})$ with a member of a parameterized family of probability distributions $q(\boldsymbol{x};\boldsymbol{\psi})$. The vector of variational parameters $\boldsymbol{\psi}$ is obtained by maximizing the ELBO:

$$\mathcal{L}\left[\boldsymbol{\psi}\right] = -\mathbb{E}_{q(\boldsymbol{x};\boldsymbol{\psi})}\left[\log \frac{q(\boldsymbol{x};\boldsymbol{\psi})}{L\left(\boldsymbol{y} \mid \boldsymbol{x}\right) p(\boldsymbol{x})}\right] \tag{1}$$

where $L\left(\boldsymbol{y} \mid \boldsymbol{x}\right)$ is a likelihood function. The resulting variational posterior (i.e. the maximum of this optimization problem) depends on the choice of the parameterized family $q(\boldsymbol{x};\boldsymbol{\psi})$ and it is equal to the exact posterior only when the latter is included in the family. In this paper, we restrict our attention to probabilistic programs that are specified in terms of conditional probabilities and densities chained together by deterministic functions:

$$p(\boldsymbol{x}) = \prod_j \rho_j\left(x_j \mid \boldsymbol{\theta}(X_j)\right) \tag{2}$$

where $\rho_j\left(x \mid \boldsymbol{\gamma}\right)$ is a family of probability distributions and $X_j = \{x_j, \ldots, x_M\}$ is a subset parent variables such that the resulting graphical model is a directed acyclic graph (DAG). The vector-valued functions $\boldsymbol{\theta}_j(X_j)$ specifies the value of the parameters of the j-th of the distribution of the latent variable x_j given the values of all its parents.

An automatic variational family is determined by a algorithm that takes as input a probabilistic program $p(\boldsymbol{x})$ and outputs a parameterized variational family $q(\boldsymbol{x};\boldsymbol{\psi})$. The most commonly used algorithm consists in creating a random variable for each latent variable in the program which follows the same distribution but with uncoupled variational parameters (i.e. the mean-field (MF) approximation). This approach is somehow reminiscent of parameter update in conjugate models, where the posterior in in the same family as the prior. In this paper, we go further by introducing an explicit parameterization of the variational family that mimics the update rule of (expectation) parameters in exactly solvable conjugate models. This approach leads to a flexible structured family that includes the prior probabilistic program as special case.

2.1. Parameter update in conjugate models

Exponential family distributions have a central role in Bayesian statistics as they are the only to admit conjugate priors, where inference can be performed in closed form. An exponential family distribution $p(y)$ can be parameterized by a vector of expectation parameters $\boldsymbol{\mu} = \mathbb{E}_{p(y)}[\boldsymbol{T}(y)]$, where $\boldsymbol{T}(y)$ is the vector of sufficient statistics of the data. We can assign to these parameters a conjugate prior distribution $p(\boldsymbol{\mu})$, which in turn is parameterized by the prior expectations $\bar{\boldsymbol{\mu}}_0 = \mathbb{E}_{p(\boldsymbol{\mu})}[\boldsymbol{\mu}]$. Upon observing N independently sampled datapoints, it can be shown that the posterior expectation parameters are convex combination of the prior parameters and the maximum likelihood estimators:

$$\bar{\boldsymbol{\mu}} = \boldsymbol{\lambda} \odot \bar{\boldsymbol{\mu}}_0 + (1 - \boldsymbol{\lambda}) \odot \boldsymbol{\mu}_{\text{ML}} \,. \tag{3}$$

where $\boldsymbol{\lambda}$ is a vector of convex combination coefficients, $\odot$ denotes the element-wise product, $\boldsymbol{\mu}_{\text{ML}}$ is the maximal likelihood estimator. For example, in a Gaussian model with known likelihood precision τ and Gaussian prior over the mean, the mean parameter updates as

$$\bar{\mu} = \frac{\tau_0}{\tau_0 + N\tau}\bar{\mu}_0 + \frac{N\tau}{\tau_0 + N\tau}\left(\frac{1}{N}\sum_{n=1}^{N} y_n\right) \,, \tag{4}$$

where τ_0 is the precision of the prior. This formula shows that the posterior parameters are a trade-off between the prior hyper-parameters and the value induced by the data.

3. Pseudo-conjugate variational families

We are finally ready to introduce the central innovation of the paper. Consider the following probabilistic model:

$$p(x, y) = L(y \mid x)\rho(x \mid \boldsymbol{\theta}) \tag{5}$$

where $L\left(y \mid x\right)$ is a likelihood function and $\rho\left(x \mid \boldsymbol{\theta}\right)$ is a prior distribution parameterized by a vector of parameters $\boldsymbol{\theta}$. We do not assume that the likelihood or the prior to be in the exponential family. Nevertheless, we can construct a pseudo-conjugate parameterized variational family by copying the form of the parameter update rule in conjugate models:

$$q(x; \boldsymbol{\lambda}, \boldsymbol{\alpha}) = \rho(x \mid \boldsymbol{\lambda} \odot \boldsymbol{\theta} + (1 - \boldsymbol{\lambda}) \odot \boldsymbol{\alpha}) \,, \tag{6}$$

where $\boldsymbol{\lambda}$ is now a vector of learnable parameters with entries ranging from 0 to 1 and $\boldsymbol{\alpha}$ is a vector of learnable parameters that have the same domain of definition of the parameters $\boldsymbol{\theta}$.

In a model with a single latent variable, the pseudo-conjugate parameterizations is overparameterized. However, the power of this approach becomes evident in multivariate models constructed by chaining basic probability distributions. Consider a probabilistic program specified in the form of Eq. 2. We can construct a structured variational family by applying the pseudo-conjugate form to each latent conditional distribution in the model:

$$\begin{aligned} &q(\boldsymbol{x}; \boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_J, \boldsymbol{\alpha}_1., \ldots, \boldsymbol{\alpha}_J) \\ &= \prod_j \rho_j\Big(x_j \mid \mathcal{U}^{\boldsymbol{\alpha}_j}_{\boldsymbol{\lambda}_j}\left[\boldsymbol{\theta}_j(X_j)\right]\Big) \,. \end{aligned} \tag{7}$$

Where we rewrite the formula succinctly using the *convex update operator*:

$$\mathcal{U}^{\boldsymbol{\alpha}}_{\boldsymbol{\lambda}}\left[\boldsymbol{\theta}\right] = \boldsymbol{\lambda} \odot \boldsymbol{\theta} + (1 - \boldsymbol{\lambda}) \odot \boldsymbol{\alpha} \tag{8}$$

.

3.1. Theoretical and practical justifications of the structured pseudo-conjugate families

The multivariate structured distributions induced by this family have several appealing theoretical properties that justify their usage in structured inference problems:

- The family always contains the original probabilistic program (i.e. the prior distribution). This is trivial to see as we can obtain the prior by setting all the lambdas equal to 1. On the other hand, setting all the lambdas equal to 0 leaves us with the standard mean-field approximation. Note that none of the commonly used automatic structured variational approaches share this simple property.

- In pseudo-conjugate stochastic VI, the gradient estimator can backpropagate through forward pass of the probabilistic program. Consequently, all the variational variables can be updated from the very first stochastic gradient update. Conversely, MF stochastic VI can update variables that are not directly connected to observations only by updating all the intermediary variables through multiple gradient updates. This phenomenon is formally analogous of the bootstrap of policy updates in model-free reinforcement learning (Sutton & Barto, 2018). The phenomenon is visualized in Fig. 1 which shows the magnitude of the gradients during the first 100 updates of stochastic VI training of a timeseries experiment (described in Appendix A).

- The family includes both the filtering and the smoothing exact posterior of univariate linear Gaussian time-series models such as

$$x_t \sim \mathcal{N}\left(a x_{t-1}, \sigma^2\right), \quad y_t \sim \mathcal{N}\left(x_t, \xi^2\right) . \tag{9}$$

  In this case, the filtering conditional posterior is given by Kalman filter update:

$$\begin{aligned} \bar{\mu}_{t+1} &= (1 - K_t)\mu_{t+1}(x_t) + K_t y_t \\ &= (1 - K_t) a x_{t-1} + K_t y_t \, . \end{aligned} \tag{10}$$

  where $0 \leq K_t \leq 1$ is the Kalman gain which in our case corresponds to λ_t. The smoothing update has a similar form where the data term is augmented with a estimate integrating all observations of future time-points.

- The pseudo-conjugate family has a very parsimoniously parameterization compared with other structured families. The number of parameters is $2P$, where P is the total number of parameters of the conditional distributions. Conversely, the multivariate normal approach scales quadratically with the number of latent variables. However, this parsimonious parameterization implies that the pseudo-conjugate family cannot capture dependencies that are not already present in the prior probabilistic program. Specifically, pseudo-conjugate family cannot model correlations originating from colliding arrows in the DAG ("explaining away" dependencies).

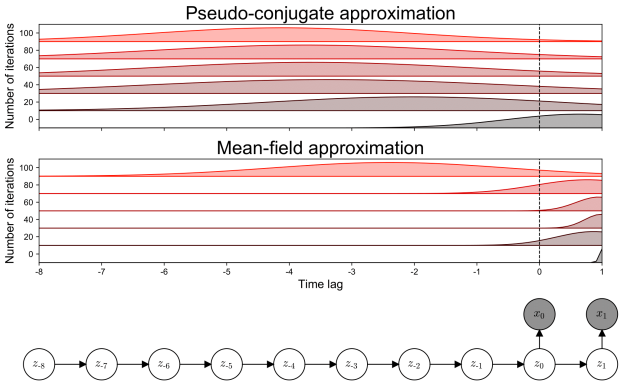


Figure 1. Propagation of the gradient to earlier latent variables. The different plots visualize the magnitude of the normalized gradients over the past latent variables. The corresponding graphical model is also shown, which indicates that only from time point $t = 0$ and onward anything is observed. Any information for the earlier latent variables must therefore come via the propagation of the gradient.

3.2. Pseudo-conjugate families for stochastic processes

The pseudo-conjugate family can be extended to discrete-time and continuous-time stochastic processes. Stochastic processes can be seen as probabilistic programs with a potentially infinite number of variables. As a example, consider a discrete-time Markov process defined marginals of the following form:

$$p(x_1, \ldots, x_t, \ldots, x_N) = \prod_{j=0}^{N-1} \rho_j(x_{j+1} \mid \boldsymbol{\theta}_j(x_j)) \tag{11}$$

for all sets of ordered contiguous time points starting from 0. Assume that we collected noisy observations of the process at the arbitrary ordered set of time points $\{s_1, \ldots, s_M\}$. We can construct a pseudo-conjugate variational process by applying the convex update operator to the active set of all time points prior to the last observation:

$$\begin{aligned} &q(x_1, \ldots, x_t, \ldots, x_N) = \\ &\prod_{j=0}^{s_M - 1} \rho_j(x_{j+1} \mid \mathcal{U}_{\boldsymbol{\lambda}}^{\boldsymbol{\alpha}}\left[\boldsymbol{\theta}_k(x_k)\right]) \prod_{k=s_M}^{N} \rho_k(x_{k+1} \mid \boldsymbol{\theta}_k(x_k)) \, . \end{aligned} \tag{12}$$

It is straightforward to dynamically expand the active set simply by adding the appropriate update operators. This suggests the use of pseudo-conjugate families in Bayesian

	BR (Full)	BR (Bridge)	OS (Full)	OS (Bridge)	LZ (Full)	LZ (Bridge)
ASVI	0.06 ± 0.01	$\mathbf{0.08} \pm 0.01$	$\mathbf{0.10} \pm 0.01$	$\mathbf{0.15} \pm 0.01$	$\mathbf{0.48} \pm 0.06$	$\mathbf{0.56} \pm 0.05$
ADVI (MF)	0.06 ± 0.01	0.13 ± 0.02	0.12 ± 0.01	0.21 ± 0.02	1.39 ± 0.24	1.90 ± 0.313
ADVI (MN)	$\mathbf{0.05} \pm 0.01$	0.12 ± 0.02	$0.11 + -0.01$	0.16 ± 0.01	3.04 ± 0.45	4.01 ± 0.53
NN	0.06 ± 0.01	0.11 ± 0.01	$\mathbf{0.10} \pm 0.01$	0.17 ± 0.01	0.71 ± 0.08	1.07 ± 0.20

Table 1. Average and standard error of the root mean squared errors between the posterior means and the ground truth curves computed on 15 simulations.

nonparametrics that combine sampling of the DAG structure with VI in the DAG parameters (Wang & Blei, 2012). This can be particularly useful when combined with nonparametric models that can learn the graphical structure of the DAG (Patrick et al., 2020).

We can also define pseudo-conjugate variational families for diffusion processes defined as solutions of stochastic differential equations (SDE). Consider the distribution induced by the following SDE:

$$\mathrm{d}x(t) = f\left(x\left(t\right), t\right)\mathrm{d}t + g\left(x\left(t\right), t\right)\mathrm{d}B(t)\ . \tag{13}$$

where f is the drift function, g is the volatility function and $B(t)$ is a standard Brownian motion process. The corresponding variational SDE is obtained by applying the convex update operator to the drift function:

$$\mathrm{d}x(t) = \mathcal{U}_{\lambda(t)}^{\alpha(t)}\left[f\left(x\left(t\right), t\right)\right]\mathrm{d}t + g\left(x\left(t\right), t\right)\mathrm{d}B(t)\ , \tag{14}$$

where $\lambda(t)$, $\alpha(t)$, $\eta(t)$ and $\beta(t)$ are now functions. Note that in this context $\lambda(t)$ and $\alpha(t)$ can be interpreted as control variables which can redirect the paths of the SDE towards the datapoints. Variational inference in these SDEs model can be performed either by discretizing the time-axis or using more sophisticated continuous stochastic backpropagation methods (Li et al., 2020).

4. Automatic structured variational inference

ASVI is a form of automatic differentiation variational inference in which the variational family is the pseudo-conjugate family constructed from the input probabilistic program. The family is constructed by copying the input probabilistic program and applying the convex update operator to each function that specify the parameters of a node given the values of its parents (Eq. 8). We denote the conditional distributions obtained in this way as $\mathcal{U}\rho_j$. The lambda variables are constrained to be between 0 and 1. This constraint is implemented by passing a unconstrained variable through a sigmoid function. The alpha and lambda parameters are trained by minimizing the ELBO, which in our case has the following form:

$$\begin{aligned}\mathcal{L} = & \\ &\mathbb{E}_{\boldsymbol{x}}[\log L(\boldsymbol{y} \mid \boldsymbol{x})] \\ &- \sum_j \mathbb{E}_{X_j}[D_{KL}(\rho_j(x_j \mid X_j) \| \mathcal{U}\rho_j(x_j \mid X_j))]\ ,\end{aligned} \tag{15}$$

where the expectations are taken with respect to the variational family. Note that, if ρ_j is a tractable exponential family distribution, we can automatically compute the KL divergence analytically, thereby reducing the variance of the gradient estimator.

5. Related work

Structured VI is commonly applied in time series models such as hidden Markov models and autoregressive models. In these models, the posterior distributions inherit strong statistical dependencies from the sequential nature of the prior. Structured VI for timeseries usually use structured variational families that capture the temporal dependencies while being fully-factorized in the non-temporal variables (Eddy, 1996; Foti et al., 2014; Johnson & Willsky, 2014; Karl et al., 2016; Fortunato et al., 2017). This differs from the pseudo-conjugate families preserve where both temporal and non-temporal dependencies are preserved. Furthermore, these approaches typically require model specific derivations and variational bounds.

Several forms of model agnostic structured variational distributions have been introduced. Hierarchical VI accounts for dependencies between latent variables by coupling the parameters of their factorized distributions through a joint *variational prior* (Ranganath et al., 2016). While this method is very general, it requires user input in order to define the variational prior and the use of a modified variational lower bound. Copula VI models the dependencies between latent variables using a vine copula function (Tran et al., 2015). In the context of probabilistic programming, copula VI shares the some of the same limitations of hierarchical VI: It requires the appropriate specification of bivariate copulas and it needs a specialized inference technique. The approach that is closest to our current work is perhaps structured stochastic VI (Hoffman et al., 2013). Similarly to our model, its variational posteriors have the same conditional independence structure as the input probabilistic program. However, this

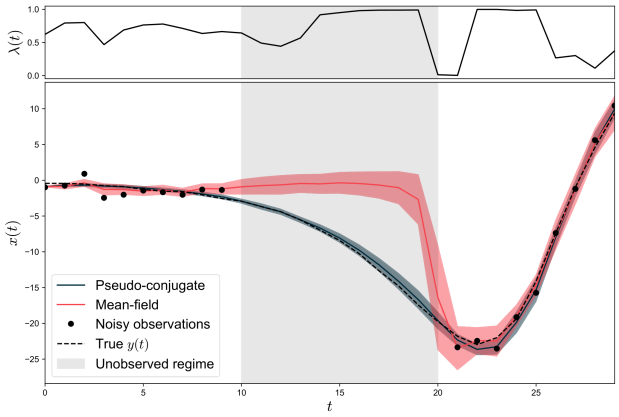


Figure 2. Example of inference on a stochastic Lorenz bridge.

method is limited to conditionally conjugate models with exponential family distributions. Furthermore, the resulting ELBO is intractable and it needs to be estimated using specialized techniques. Finally, automatic differentiation VI (Kucukelbir et al., 2017) maps the values of all latent variables to a unbounded coordinate space based on the support of each distribution. The variational distribution in this new space is then parameterized as a multivariate Gaussian. While this approach is very generic, it exploits very little information information from the original probabilistic model and has scalability problems due to the cubic complexity of Bayesian inference with multivariate Gaussian distributions.

6. Applications

We evaluate the performance of ASVI and relevant baselines in a range of probabilistic inference problems. Since the main goal of this paper is to introduce a new form of fully automatic VI that works in arbitrary probabilistic programs, we will only compare with general purpose variational families. Therefore, we will not include model-tailored variational families among our baselines. Furthermore, since our approach is meant to be generally applicable, we will only compere with existing methods that do not requires special mathematical tractability assumptions such as local conjugacy (Hoffman & Blei, 2015). Finally, we will exclude from our comparisons methods that require the use of *ad hoc* loss functionals and gradient estimators since our pseudo-conjugate family is meant as a drop-in replacement in standard stochastic VI settings where path-derivative and reinforce estimators are used.

6.1. Time series analysis

As first application, we focus on timeseries models and SDEs. We used three SDE models. The first model (BR) is a Brownian motion without drift. The second model (OS) is a linear second-order Langevin equation with oscillatory dynamics. Finally, the third model (LZ) is a stochastic Lorenz dynamical system. The details of these SDEs are given in Appendix A. The form of our pseudo-conjugate family is given in Eq. 11 where the conditional densities are Gaussian distributions obtained by discretizing the SDE:

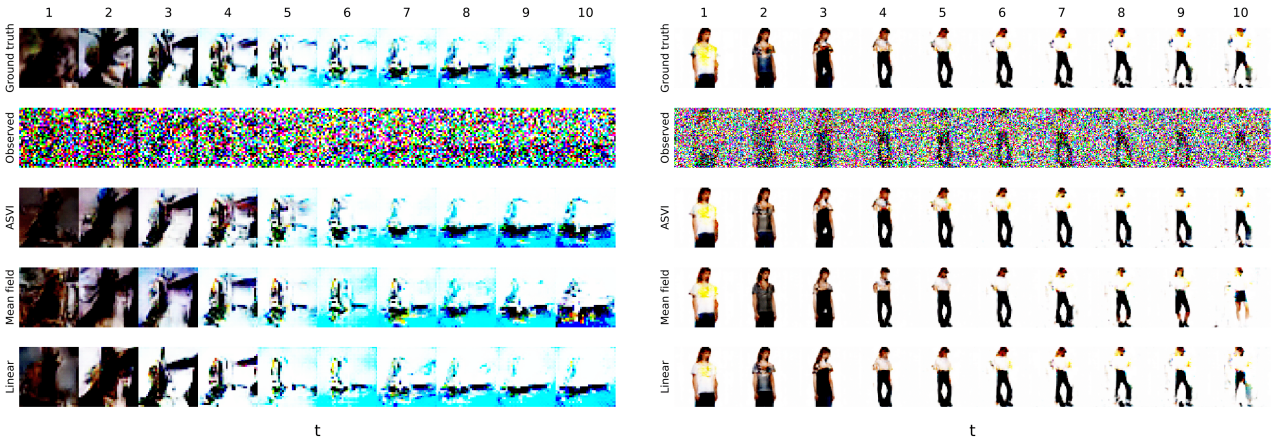
$$\mathcal{N}\Big(x_{t+1} \mid x_t + f(x_t, t)\,\mathrm{d}t, g(x_t, t)\sqrt{\mathrm{d}t}\Big) \qquad (16)$$

where f is the drift function while g is the volatility function. As baselines, we implemented mean field ADVI, multivariate normal ADVI and a more expressive hierarchical structured variational approach where the dependencies are learned using a fully connected neural network (Ranganath et al., 2016). The details of this baseline is given in Appendix A. For each timeseries model, we performed inference in three experimental situations. In the first case (Full) the processes were observed at all time points with a Gaussian likelihood (BR: sd = 0.15, OS: sd = 0.2, LZ: sd = 1). In the second case (Bridge), the processes were observed only in the first 10 and last 10 timepoints. Finally, in the third case (Past) the processes were observed only in the last 10 timepoints. In the case of the Lorenz system, only the x timeseries was observed while z and y were latent variables.

Table 1 reports the performance of ASDI and baselines quantified as root mean squared deviation of the posterior mean (rMSE) from the generated ground truth curves. ASDI achieves the highest performance in almost all comparisons with the gain being more pronounced in the bridge experiments. Figure 2 shows the analysis in a example trial of the LZ Bridge experiment. Figure 2 (Top panel) shows the variational parameter $\lambda(t)$ as function of the time-step. In a pseudo-conjugate distribution, λ can be interpreted as a surprise detector with low values being associated with high surprise. In this example, the dynamic of $\lambda(t)$ is particularly interpretable. In the first observed regime, $\lambda(t)$ stays at a intermediary value as the trajectory needs to be corrected in order to keep track of the data. Subsequently, in the unobserved regime, $\lambda(t)$ increases as the extrapolation has to rely on the prior dynamics. Eventually the second observed regime is reached and lambda suddenly spikes down in order to correct the trajectory to fit the new datapoints. Interestingly, after this first correction $\lambda(t)$ shoots back to very high values as in this new high gain regime the Lorenz dynamics can fit the data without the need of much interference. This situation exemplifies how the pseudo-conjugate optimization works in general: If the prior model is structured, the variational parameters α and λ only need to nudge the prior dynamics towards the observations at selected points.

6.2. Deep Bayesian smoothing with neural SDEs

So far, we performed inference in simple timeseries models with low-dimensional state spaces. We will now test the



(a) CIFAR10 (b) FashionGEN

Figure 3. Qualitative results of the deep Bayesian smothers using ASVI, mean-field, and linear coupling. The top row shows the ground truth images, the second row the noisy-corrupted observations.

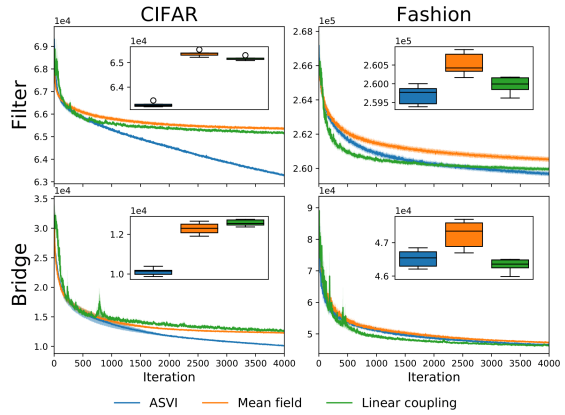


Figure 4. Quantitative results of the deep Bayesian smoothing experiment. Negative ELBO of ASVI, mean field and linear coupling as function of training iteration.

performance of ASVI on high dimensional problems with complex nonlinearities parameterized by deep networks. As latent model, we used neural stochastic differential equations (SDE) (Chen et al., 2018; Li et al., 2020):

$$\mathrm{d}\boldsymbol{x} = \boldsymbol{F}(\boldsymbol{x})\mathrm{d}t + \mathrm{d}\boldsymbol{B}(t) \tag{17}$$

where $\boldsymbol{F}(\boldsymbol{x})$ is a nonlinear function parameterized by a neural network and $\boldsymbol{B}(t)$ is a standard multivariate Brownian motion (see Appendix B for the details of the architecture). This latent processes generate noise-corrupted observations through a deep network $\boldsymbol{G}$:

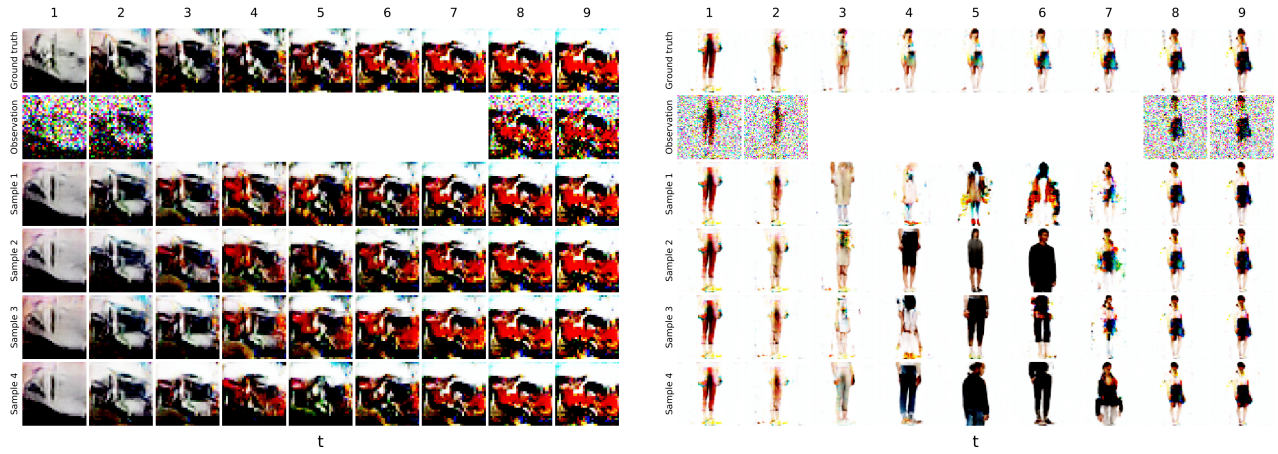
$$\boldsymbol{y}_t \sim \mathcal{N}\big(\boldsymbol{G}(\boldsymbol{x}(t)), \sigma^2\big) \tag{18}$$

In our examples, $\boldsymbol{G}$ is a generator which converts latent vectors into RGB images. The details of the networks and generators are given in Appendix B. We tested on two kinds of pre-trained generator: A DCGAN trained on CIFAR10 and a DCGAN trained on FashionGEN (Radford et al., 2015) (see Appendix B for the details of the architecture). In the former case, the latent space is 100-dimensional while in the latter it is 120-dimensional. We considered two kinds of inference problems. In smoothing problems, we aim to remove the noise from a series of images generated by a trajectory in the latent process. On the other hand, in bridge problems we reconstruct a series of intermediate images given the beginning (first three time points) and the end (last two time points) of a trajectory. For both problems,we assumed to know the dynamical and generative model and we discretize the neural SDE using a EulerMaruyama scheme and we backpropagate through the integrator (Chen et al., 2018).

Figure 3 shows the filtering performance of ASVI and two baselines (mean field, and linear Gaussian model (see Appendix B for the details) in a filtering problem. The quantitative results (negative ELBOs) are shown in Figure 4. As you can see, ASVI always reaches tighter lower bounds except in the Fashion bridge experiment where it has slightly lower performance than the linear coupling baseline. This tighter variational bound results is discernibly higher quality filtered images in both CIFAR10 and Fashon, as shown in Figure 3. Figure 5a shows several samples from the ASVI bridge posterior. As expected, the generated images diverge in the unobserved period and reconverge at the end.

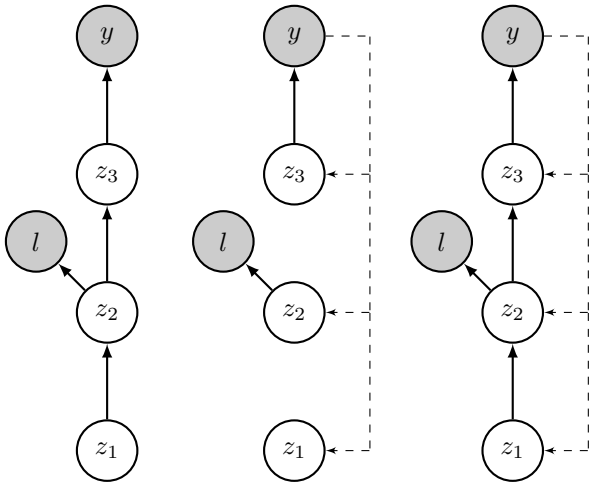
6.3. Deep amortized generative modeling

Finally, we apply a amortized form of ASVI to deep generative modeling problem. The goal is to model the joint distribution of a set of binary images $\boldsymbol{y}$ paired with class labels l. To this aim, we use a deep variational autoencoder



(a) CIFAR10 (b) FashionGEN

Figure 5. Interpolation results for ASVI. Once again, the top row shows the ground truth images. The second row shows the noisy observations, which are missing for frames 3–8. The bottom four rows show posterior samples for the model estimated with ASVI.



(a) Prior (b) Mean-field (c) ASVI

Figure 6. Graphical representation of the amortized pseudo-conjugate family associated with a deep autoencoder. Solid arrows denote probabilistic dependencies while dashed arrows denote the flow of the inference network. In the mean-field family (b) the latent variables are not joined by arrows. Conversely, the pseudo-conjugate family (d) has the same graph structure of the input probabilistic program (a).

with three layers of latent variables $\boldsymbol{z}_1$, $\boldsymbol{z}_2$ and $\boldsymbol{z}_3$ coupled in a feed forward fashion through ReLu fully connected networks:

$$\begin{aligned} \boldsymbol{z}_1 &\sim \mathcal{N}(0, 1.5) \\ \boldsymbol{z}_2 &\sim \mathcal{N}(\text{ReLu}\left(\boldsymbol{f}_1(\boldsymbol{z}_1)\right), 0.25) \\ \boldsymbol{z}_3 &\sim \mathcal{N}(\text{ReLu}\left(\boldsymbol{f}_2(\boldsymbol{z}_2)\right), 0.1) \\ \boldsymbol{y} &\sim \mathcal{N}(\boldsymbol{g}_1(\boldsymbol{z}_3), 0.1) \\ l &\sim \text{Categorical}(\text{Softmax}(\boldsymbol{g}_2(\boldsymbol{z}_2))) \end{aligned} \tag{19}$$

where $\boldsymbol{f}_1$, $\boldsymbol{f}_2$ are fully connected two-layers networks with ReLU activations and linear output units and 25 and 75 hidden units respectively while $\boldsymbol{g}_1$, $\boldsymbol{g}_2$ are linear layers. Figure 6c shows the graphical model associated to this probabilistic model. The amortized pseudo-conjugate distribution has the following form:

$$\begin{aligned} \boldsymbol{z}_1 &\sim \mathcal{N}(\boldsymbol{\alpha}_1(\boldsymbol{y}), \boldsymbol{\xi}_1(\boldsymbol{y})) \\ \boldsymbol{z}_2 &\sim \mathcal{N}(\boldsymbol{\lambda}_2 \boldsymbol{f}_1(\boldsymbol{z}_1) + (1 - \boldsymbol{\lambda}_2)\boldsymbol{\alpha}_2(\boldsymbol{y}), \boldsymbol{\xi}_2(\boldsymbol{y})) \\ \boldsymbol{z}_3 &\sim \mathcal{N}(\boldsymbol{\lambda}_3 \boldsymbol{f}_2(\boldsymbol{z}_1) + (1 - \boldsymbol{\lambda}_3)\boldsymbol{\alpha}_3(\boldsymbol{y}), \boldsymbol{\xi}_3(\boldsymbol{y})) \end{aligned} \tag{20}$$

where the mean vectors $\boldsymbol{\alpha}_k(\boldsymbol{y})$ is the activation of the $(8 - 2k)$-th layer (post ReLu) of a fully-connected 6-layers ReLu inference network taking the image $\boldsymbol{y}$ as input and with sizes $(120, 100, 70, 50, 70, 2)$. On the other hand, the scale parameter vectors $\boldsymbol{\xi}_k(\boldsymbol{y})$ were obtained by applying a linear layer to the $(8 - 2k)$-th layer followed by a softplus transformation. The details of all architectures are given in Appendix C. The amortized family was parameterized by the lambdas and by the weights and biases of the inference network. The mean-field baseline had the same form given in Eq. 20 but with $\boldsymbol{\alpha}_k(\boldsymbol{y})$ fully determining the expectation of the distribution. We did not include comparison with the

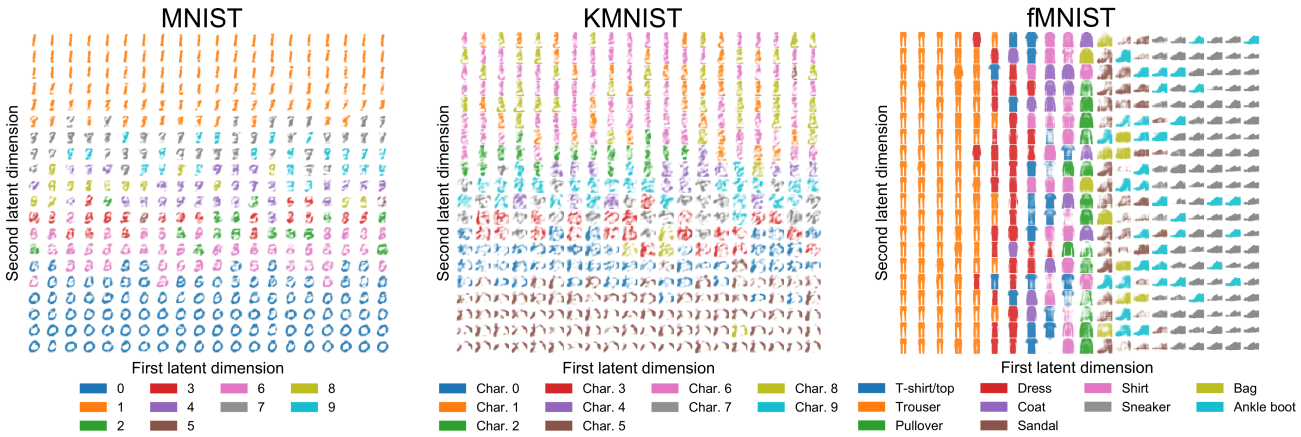


Figure 7. The latent embeddings for the different MNIST variants. Each element is a predicted sample together with its corresponding label.

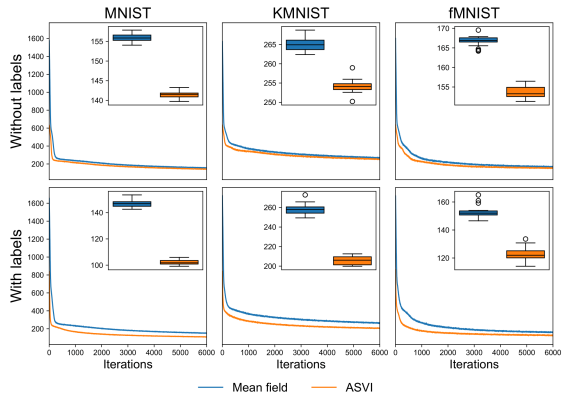


Figure 8. The losses for the different MNIST variants. Shaded interval indicates one standard deviation around the mean, over 5 runs. Inset boxes show the distribution of the results at the 6000-th iteration.

other baselines as they are computationally unfeasible in this larger scale experiment.

We tested the performance of these deep variational generative models in three computer vision datasets: MNIST, FashinMNIST and KMNIST (LeCun et al., 1998; Xiao et al., 2017; Clanuwat et al., 2018). Furthermore, we performed two types of experiment: I) Images and labels were generated jointly and II) only images were generated. Figure 8 shows the performance of ASVI and mean field baseline quantified as the negative ELBO. As you can see, ASVI achieves tighter bounds in all experiments for all datasets. Figure 7 shows a randomized selection of images generated by the ASVI model together with the corresponding label.

7. Discussion

In this paper we introduced a automatic algorithm for constructing an appropriate structured variational family given a input probabilistic program. The resulting method can be used on any probabilistic program specified by a directed Bayesian network and always preserves the forward-pass structure of the input program. The main limitation of the pseudo-conjugate family is that it cannot capture dependencies induced by colliding arrows om the input graphical model. Consequently, in a model such a standard Bayesian neural network, where the prior over the weights is decoupled, the pseudo-conjugate family is a mean-field family.

References

Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P., Horsfall, P., and Goodman, N. D. Pyro: Deep universal probabilistic programming. *The Journal of Machine Learning Research*, 20(1):973–978, 2019.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, 2018.

Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., and Ha, D. Deep learning for classical japanese literature. *arXiv preprint arXiv:1812.01718*, 2018.

Eddy, S. R. Hidden markov models. *Current Opinion in Structural Biology*, 6(3):361–365, 1996.

Fortunato, M., Blundell, C., and Vinyals, O. Bayesian recurrent neural networks. *arXiv preprint arXiv:1704.02798*, 2017.

Foti, N., Xu, J., Laird, D., and Fox, E. Stochastic variational inference for hidden markov models. In *Advances in Neural Information Processing Systems*, 2014.

Hernández-Lobato, J. M. and Adams, R. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pp. 1861–1869, 2015.

Hoffman, M. and Blei, D. Stochastic structured variational inference. In *Artificial Intelligence and Statistics*, pp. 361–369, 2015.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

Johnson, M. and Willsky, A. Stochastic variational inference for bayesian time series models. In *International Conference on Machine Learning*, 2014.

Karl, M., Soelch, M., Bayer, J., and Van der Smagt, P. Deep variational bayes filters: Unsupervised learning of state space models from raw data. *arXiv preprint arXiv:1605.06432*, 2016.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kucukelbir, A., Ranganath, R., Gelman, A., and Blei, D. Automatic variational inference in stan. In *Advances in Neural Information Processing Systems*, pp. 568–576, 2015.

Kucukelbir, A., Tran, D., Ranganath, R., Gelman, A., and Blei, D. M. Automatic differentiation variational inference. *The Journal of Machine Learning Research*, 18(1): 430–474, 2017.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, X., Wong, T. L., Chen, R. T., and Duvenaud, D. Scalable gradients for stochastic differential equations. *arXiv preprint arXiv:2001.01328*, 2020.

Patrick, D., Ambrogioni, L., Trottier, L., Gl, U., Hinne, M., Gigure, P., Chaib-Draa, B., van Gerven, M., and Laviolette, F. The indian chefs process. *arXiv preprint arXiv:2001.10657*, 2020.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Ranganath, R., Gerrish, S., and Blei, D. Black box variational inference. In *Artificial Intelligence and Statistics*, pp. 814–822, 2014.

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

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Tran, D., Blei, D., and Airoldi, E. M. Copula variational inference. In *Advances in Neural Information Processing Systems*, pp. 3564–3572, 2015.

Tran, D., Kucukelbir, A., Dieng, A. B., Rudolph, M., Liang, D., and Blei, D. M. Edward: A library for probabilistic modeling, inference, and criticism. *arXiv preprint arXiv:1610.09787*, 2016.

Wang, C. and Blei, D. M. Truncation-free online variational inference for bayesian nonparametric models. In *Advances in Neural Information Processing Systems*, 2012.

Wingate, D. and Weber, T. Automated variational inference in probabilistic programming. *arXiv preprint arXiv:1301.1299*, 2013.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

A. Details of the timeseries experiments

A.0.1. MODELS

As first application, we focus on timeseries models and SDEs. We used the following three models. The first model (BR) is a Brownian motion without drift and with innovation standard deviation equal to 0.1. The second model (OS) is a linear Langevin equation with oscillatory dynamics:

$$x''(t) = -\omega_0^2 x(t) - \beta x'(t) + w(t) \tag{21}$$

where $\omega_0 = 2\pi 8$, $\beta = 20$ and $w(t)$ is a Gaussian white noise process with standard deviation equal to 0.5. Finally, the third model (LZ) is a stochastic Lorenz system (nonlinear SDE):

$$\begin{aligned} x'(t) &= 10(y(t) - x(t)) + w_x(t) \\ y'(t) &= x(t)(28 - z(t)) - y(t) + w_y(t) \\ z'(t) &= x(t)y(t) - (8/3)z(t) + w_z(t) \end{aligned} \tag{22}$$

where $w_x(t)$, $w_y(t)$ and $w_z(t)$ are Gaussian white noise processes with standard deviation equal to 0.1.

A.0.2. BASELINES

The ADVI (MF) baseline was obtained by replacing all the conditional Gaussian distributions in the probabilistic program with Gaussian distributions with uncoupled trainable mean and standard deviation parameters. The optimization on the positive-valued standard deviations was performed by transforming real-valued trainable parameters using a sofplus function.

The ADVI (MF) baseline, the distribution over all the variables was modeled as a multivariate Gaussian parameterized by its mean vector and the lower-triangular factor of covariance (with positive-valued diagonal).

In the NN baseline was a hierarchical variational distribution (Ranganath et al., 2016). The mean parameters of all the Gaussian variables in the mean field model were obtained by transforming a standard 10-dimensional (30-dimensional for the LZ experiment) noise vector $\boldsymbol{\epsilon}$ with a trainable fully connected perceptron:

$$q_{\mathrm{S}}(\boldsymbol{x}, \nu) = \mathcal{N}(x_0 \mid \chi_0(\boldsymbol{\epsilon}), s_0) \prod_n \mathcal{N}(x_t \mid \chi_t(\boldsymbol{\epsilon}), s_t) \quad (23)$$

where χ_n is the n-th component of the linear output of a fully connected two layers perceptron with 10 (30 for the LZ experiment) hidden units, without biases and with sigmoid activations in the hidden units.

A.0.3. EXPERIMENT DETAILS

All processes were discretized with a EulerMaruyama method ($\mathrm{d}t = 0.01$ for BR and OS and $dt = 0.02$ for LZ) and the transition probability were approximated as Gaussian distributions (this approximation is exact for dt tending to 0). The total number of time points was 40 for BR and OS and 30 for LZ. Each experiment was repeated 15 times with different synthetic data generated from the joint model. The gradients of the ELBOs were estimated using path-derivative gradient estimators (20 samples, with entropy term integrated analytically). The models were optimized using Adam (Kingma & Ba, 2014) with parameters: lr=0.05 (0.015 for MN), betas=(0.9, 0.999), eps=1e-08. In the OS and BR experiment, the models were trained for 200 iterations. This number was chosen as all the model showed convergence within this iterations range. Conversely, the more challenging LZ problem required 400 iterations for ASDI and NN, 2000 iterations for ADVI (MF) and 4000 iterations for ADVI (MN). The training of this latter model was unstable, leading to some sub-optimal local optima.

B. Details of the neural SDE experiment

B.0.1. MODELS

The function $\boldsymbol{F}(\boldsymbol{x})$ had the following form

$$\boldsymbol{F}(\boldsymbol{x}) = W_2 \tanh\left(\boldsymbol{x} + \tanh(W_1 \boldsymbol{x})\right) \quad (24)$$

where W_2 and W_1 were $d \times d$ matrices whose entries were sampled in each of the 5 repetitions from a centered normal with SD equal to 0.2. Those matrices encodes the forward dynamical model and they were assumed to be known during the experiment. This is a Kalman filter-like setting where the form of the forward model is known and the inference is performed in the latent units. The neural SDE was integrated using EulerMaruyama integration with step size equal to 1 from $t = 0$ to $t = 9$. We trained the model by back-propagating though the integrator.

We used two DCGAN generators as emission models. The networks were the DCGAN implemented in PyTorch. In the CIFAR experiment, we used the following architecture:

```
ConvTranspose2d(100,64*8,4,1,0,
                bias=False),
BatchNorm2d(64*8),
ReLU(True)
ConvTranspose2d(64*8, 64*4,4,2,1,
                bias=False),
BatchNorm2d(ngf*4),
ReLU(True),
ConvTranspose2d(64*4,64*2,4,2,1,
                bias=False),
BatchNorm2d(64*2),
ReLU(True),
ConvTranspose2d(64*2,64,4,2,1,
                bias=False),
BatchNorm2d(ngf),
ReLU(True),
ConvTranspose2(64,4,kernel_size=1,
               stride=1,
               padding=0,
               bias=False),
Tanh()
```

Network pretrained on CFAR was obtained from the GitHub repository: csinva/gan-pretrained-pytorch. The Fashion-GEN network was downloaded from the pytorch GAN zoo repository. The architectural details are given in (Radford et al., 2015).

B.0.2. BASELINES

The ADVI (MF) baseline was obtained by replacing all the conditional Gaussian distributions in the probabilistic program with Gaussian distributions with uncoupled trainable mean and standard deviation parameters. ADVI (MN) was

not computationally feasible in this larger scale experiment. Therefore, we implemented a a linear Gaussian model whith conditional densities:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \sim \mathcal{N}\left(W\boldsymbol{x}_{t-1} + \boldsymbol{\alpha}_t, \boldsymbol{\sigma}_t^2\right) , \qquad (25)$$

where the matrix W, and the vectors $\boldsymbol{\alpha}_t$ and $\boldsymbol{\sigma}_t^2$ are learnable parameters.

C. Details of the autoencoder experiment

C.0.1. MODELS

Decoder 1 ($\boldsymbol{f}_1(\boldsymbol{z}_1)$)

```
hidden_size=25
Linear(latent_size1, hidden_size)
ReLU()
Linear(hidden_size, latent_size2)
```

Decoder 2 ($\boldsymbol{f}_2(\boldsymbol{z}_2)$)

```
hidden_size = 75
Linear(latent_size2, hidden_size)
ReLU()
Linear(hidden_size, latent_size3)
```

Decoder 3 ($\boldsymbol{g}_1(\boldsymbol{z}_3)$)

```
Linear(latent_size3, image_size)
```

Decoder 4 ($\boldsymbol{\alpha}(\boldsymbol{y})$)

```
Linear(latent_size3, image_size)
```

Inference network ($\boldsymbol{f}_1(\boldsymbol{z}_1)$)

```
hidden_size1=120
Linear(image_size, hidden_size)
ReLU() # For hidden units
# Latent mean output
# Latent log sd output
Linear(hidden_size1, latent_size3)
Softplus() # For standard deviation output
hidden_size2=70
Linear(latent_size3, hidden_size2)
ReLU() # For hidden units
# Latent mean output
Linear(hidden_size2, latent_size2)
# Latent log sd output
Linear(hidden_size2, latent_size2)
Softplus() # For standard deviation
hidden_size3=70
Linear(latent_size2, hidden_size3)
ReLU() # For hidden units
# Latent mean output
Linear(hidden_size3, latent_size1)
# Latent log sd output
Linear(hidden_size3, latent_size1)
Softplus() # For standard deviation
```