entropy

MDPI

Article

Variationally Inferred Sampling through a Refined Bound

Víctor Gallego [1,2,*] and David Ríos Insua [1,3]

[1] Institute of Mathematical Sciences (ICMAT), 28049 Madrid, Spain; david.rios@icmat.es
[2] Statistical and Applied Mathematical Sciences Institute, Durham, NC 7333, USA
[3] School of Management, University of Shanghai for Science and Technology, Shanghai 201206, China
[*] Correspondence: victor.gallego@icmat.es

Abstract: In this work, a framework to boost the efficiency of Bayesian inference in probabilistic models is introduced by embedding a Markov chain sampler within a variational posterior approximation. We call this framework "refined variational approximation". Its strengths are its ease of implementation and the automatic tuning of sampler parameters, leading to a faster mixing time through automatic differentiation. Several strategies to approximate evidence lower bound (ELBO) computation are also introduced. Its efficient performance is showcased experimentally using state-space models for time-series data, a variational encoder for density estimation and a conditional variational autoencoder as a deep Bayes classifier.

Keywords: variational inference; MCMC; stochastic gradients; neural networks

check for updates

Citation: Gallego, V.; Ríos Insua, D. Variationally Inferred Sampling through a Refined Bound. *Entropy* **2021**, *23*, 123. https://doi.org/10.3390/e23010123

Received: 24 December 2020
Accepted: 13 January 2021
Published: 19 January 2021

1. Introduction

Bayesian inference and prediction in large, complex models, such as in deep neural networks or stochastic processes, remains an elusive problem [1–3]. Variational approximations (e.g., automatic differentiation variational inference (ADVI) [4]) tend to be biased and underestimate uncertainty [5]. On the other hand, depending on the target distribution, Markov Chain Monte Carlo (MCMC) [6] methods, such as Hamiltonian Monte Carlo (HMC) [7]), tend to be exceedingly slow [8] in large scale settings with large amounts of data points and/or parameters. For this reason, in recent years, there has been increasing interest in developing more efficient posterior approximations [9–11] and inference techniques that aim to be as general and flexible as possible so that they can be easily used with any probabilistic model [12,13].

It is well known that the performance of a sampling method depends heavily on the parameterization used [14]. This work proposes a framework to automatically tune the parameters of a MCMC sampler with the aim of adapting the shape of the posterior, thus boosting the Bayesian inference efficiency. We deal with a case in which the latent variables or parameters are continuous. Our framework can also be regarded as a principled way to enhance the flexibility of variational posterior approximation in search of an optimally tuned MCMC sampler; thus the proposed name of our framework is the variationally inferred sampler (VIS).

The idea of preconditioning the posterior distribution to speed up the mixing time of a MCMC sampler has been explored recently in [15,16], where a parameterization was learned before sampling via HMC. Both papers extend seminal work in [17] by learning an efficient and expressive deep, non-linear transformation instead of a polynomial regression. However, they do not account for tuning the parameters of the sampler, as introduced in Section 3, where a fully, end-to-end differentiable sampling scheme is proposed.

The work presented in [18] introduced a general framework for constructing more flexible variational distributions, called normalizing flows. These transformations are one of the main techniques used to improve the flexibility of current variational inference (VI) approaches and have recently pervaded the approximate Bayesian inference literature with developments such as continuous-time normalizing flows [19] (which extend an initial simple variational posterior with a discretization of Langevin dynamics) or householder flow for mixtures of Gaussian distributions [20]. However, they require a generative adversarial network (GAN) [21] to learn the posterior, which can be unstable in high-dimensional spaces. We overcome this problem with our novel formulation; moreover, our framework is also compatible with different optimizers, rather than only those derived from Langevin dynamics [22]. Other recent proposals create more flexible variational posteriors based on implicit approaches typically requiring a GAN, as presented in [23] and including unbiased implicit variational inference (UIVI) [24] or semi-implicit variational inference (SIVI) [25]. Our variational approximation is also implicit but uses a sampling algorithm to drive the evolution of the density, combined with a Dirac delta approximation to derive an efficient variational approximation, as reported through extensive experiments in Section 5.

Closely related to our framework is the work presented in [26], where a variational autoencoder (VAE) is learned using HMC. We use a similar compound distribution as the variational approximation, yet our approach allows any stochastic gradient MCMC to be embedded, as well as facilitating the tuning of sampler parameters via gradient descent. Our work also relates to the recent idea of sampler amortization [27]. A common problem with these approaches is that they incur in an additional error—the amortization gap [28]—which we alleviate by evolving a set of particles through a stochastic process in the latent space after learning a good initial distribution, meaning that the initial approximation bias can be significantly reduced. A recent related article was presented in [29], which also defined a compound distribution. However, our focus is on efficient approximation using the reverse KL divergence, which allows sampler parameters to be tuned and achieves superior results. Apart from optimizing this kind of divergence, the main point is that we can compute the gradients of sampler parameters (Section 3.3), whereas in [29] the authors only consider a parameterless sampler: thus, our framework allows for greater flexibility, helping the user to tune sampler hyperparameters. In the Coupled Variational Bayes (CVB) [30] approach, optimization is in the dual space, whereas we optimize the standard evidence lower bound (ELBO). Note that even if the optimization was exact, the solutions would coincide, and it is not clear yet what happens in the truncated optimization case,other than performing empirical experiments on given datasets. We thus feel that there is room for implicit methods that perform optimization in the primal space (besides this, they are easier to implement). Moreover, the previous dual optimization approach requires the use of an additional neural network (see the paper on the Coupled Variational Bayes (CVB) approach or [31]). This adds a large number of parameters and requires another architecture decision. With VIS, we do not need to introduce an auxiliary network, since we perform a "non-parametric" approach by back-propagating instead through several iterations of SGLD. Moreover, the lack of an auxiliary network simplifies the design choices.

Thus, our contributions include a flexible and consistent variational approximation to the posterior, embedding an initial variational approximation within a stochastic process; an analysis of its key properties; the provision of several strategies for ELBO optimization using the previous approximation; and finally, an illustration of its power through relevant complex examples.

2. Background

Consider a probabilistic model $p(x|z)$ and a prior distribution $p(z)$, where x denotes the observations and $z \in \mathbb{R}^d$ the unobserved latent variables or parameters, depending on the context. Whenever necessary for disambiguation purposes, we shall distinguish between z for latent variables and θ for parameters. Our interest is in performing inference regarding the unobserved z by approximating its posterior distribution

$$p(z|x) = \frac{p(z)p(x|z)}{\int p(z)p(x|z)dz} = \frac{p(x,z)}{p(x)}.$$

The integral assessing the evidence $p(x) = \int p(z)p(x|z)dz$ is typically intractable. Thus, several techniques have been proposed to perform approximate posterior inference [3].

2.1. Inference as Optimization

Variational inference (VI) [4] tackles the problem of approximating the posterior $p(z|x)$ with a tractable parameterized distribution $q_\phi(z|x)$. The goal is to find the parameters ϕ so that the variational distribution $q_\phi(z|x)$ (also referred to as variational guide or variational approximation) can be as close as possible to the actual posterior. Closeness is typically measured through the Kullback–Leibler divergence $KL(q_\phi||p)$, reformulated into the ELBO as follows:

$$\text{ELBO}(q) = \mathbb{E}_{q_\phi(z|x)}\left[\log p(x,z) - \log q_\phi(z|x)\right], \tag{1}$$

This is the objective to be optimized, typically through stochastic gradient descent techniques. To enhance flexibility, a standard choice for $q_\phi(z|x)$ is a Gaussian distribution $\mathcal{N}(\mu_\phi(x), \sigma_\phi(x))$, with the mean and covariance matrix defined through a deep, non-linear model conditioned on observation x.

2.2. Inference as Sampling

HMC [7] is an effective sampling method for models whose probability is pointwise computable and differentiable. When scalability is an issue, as proposed by the authors in [32], a formulation of a continuous-time Markov process that converges to the target distribution $p(z|x)$ can be used, which is based on the Euler–Maruyama discretization of Langevin dynamics

$$z_{t+1} \leftarrow z_t + \eta_t \nabla_z \log p(x, z_t) + \mathcal{N}(0, 2\eta_t I), \tag{2}$$

where η_t is the step size at time period t, and I is the identity matrix. The required gradient $\nabla \log p(z_t, x)$ can be estimated using mini-batches of data. Several extensions of the original Langevin sampler have been proposed to increase its mixing speed, such as in [33–36]. We refer to these extensions as stochastic gradient MCMC samplers (SG-MCMC) [37].

3. A Variationally Inferred Sampling Framework

In standard VI, the variational approximation is analytically tractable and typically chosen as a factorized Gaussian, as mentioned above. However, it is important to note that other distributions can be adopted as long as they are easily sampled and their log-density and entropy values evaluated. However, in the rest of this paper, we focus on the Gaussian case, as the usual choice in the Bayesian deep learning community. Stemming from this variational approximation, we introduce several elements to construct the VIS.

Our first major modification of standard VI proposes the use of a more flexible distribution, approximating the posterior by embedding a sampler through

$$q_{\phi,\eta}(z|x) = \int Q_{\eta,T}(z|z_0) q_{0,\phi}(z_0|x) dz_0, \tag{3}$$

where $q_{0,\phi}(z|x)$ is the initial and tractable density $q_\phi(z|x)$ (i.e., the starting state for the sampler). We designate this as refined variational approximation. The conditional distribution $Q_{\eta,T}(z|z_0)$ refers to a stochastic process parameterized by η and used to evolve the original density $q_{0,\phi}(z|x)$ for T periods, so as to achieve greater flexibility. Specific forms for $Q_{\eta,T}(z|z_0)$ are described in Section 3.1. Observe that when $T = 0$, no refinement steps are performed and the refined variational approximation coincides with the original one; on the other hand, as T increases, the approximation will be closer to the exact posterior, assuming that $Q_{\eta,T}$ is a valid MCMC sampler in the sense of [37].

We next maximize a refined ELBO objective, replacing in Equation (1) the original q_ϕ by $q_{\phi,\eta}$:

$$\text{ELBO}(q_{\phi,\eta}) = \mathbb{E}_{q_{\phi,\eta}(z|x)}\left[\log p(x,z) - \log q_{\phi,\eta}(z|x)\right] \tag{4}$$

This is done to optimize the divergence $KL(q_{\phi,\eta}(z|x)||p(z|x))$. The first term of Equation (4) requires only being able to sample from $q_{\phi,\eta}(z|x)$; however, the second term, the entropy $-\mathbb{E}_{q_{\phi,\eta}(z|x)}[\log q_{\phi,\eta}(z|x)]$, also requires the evaluation of the evolving, implicit density. Section 3.2 describes efficient methods to approximate this evaluation. As a consequence, performing variational inference with the refined variational approximation can be regarded as using the original variational guide while optimizing an alternative, tighter ELBO, as Section 4.2 shows.

The above facilitates a framework for learning the sampler parameters ϕ, η using gradient-based optimization, with the help of automatic differentiation [38]. For this, the approach operates in two phases. First, in a refinement phase, the sampler parameters are learned in an optimization loop that maximizes the ELBO with the new posterior. After several iterations, the second phase, focused on inference, starts. We allow the tuned sampler to run for sufficient iterations, as in SG-MCMC samplers. This is expressed algorithmically as follows.

Refinement phase:

Repeat the following until convergence:

1. Sample an initial set of particles, $z_0 \sim q_{0,\phi}(z|x)$.
2. Refine the particles through the sampler, $z_T \sim Q_{\eta,T}(z|z_0)$.
3. Compute the ELBO objective from Equation (4).
4. Perform automatic differentiation on the objective wrt parameters ϕ, η to update them.

Inference phase:

Once good sampler parameters ϕ^*, η^* are learned,

1. Sample an initial set of particles, $z_0 \sim q_{0,\phi^*}(z|x)$.
2. Use the MCMC sampler $z_T \sim Q_{\eta^*,T}(z|z_0)$ as $T \to \infty$.

Since the sampler can be run for a different number of steps depending on the phase, we use the following notation when necessary: VIS-X-Y denotes $T = X$ iterations during the refining phase and $T = Y$ iterations during the inference phase.

Let us specify now the key elements.

3.1. *The Sampler $Q_{\eta,T}(Z|Z_0)$*

As the latent variables z are continuous, we evolve the original density $q_{0,\phi}(z|x)$ through a stochastic diffusion process [39]. To make it tractable, we discretize the Langevin dynamics using the Euler–Maruyama scheme, arriving at the stochastic gradient Langevin dynamics (SGLD) sampler (2). We then follow the process $Q_{\eta,T}(z|z_0)$, which represents T iterations of the MCMC sampler.

As an example, for the SGLD sampler $z_t = z_{t-1} + \eta\nabla \log p(x, z_{t-1}) + \xi_t$, where t iterates from 1 to T. In this case, the only parameter is the learning rate η and the noise is $\xi_t \sim \mathcal{N}(0, 2\eta I)$. The initial variational distribution $q_{0,\phi}(z|x)$ is a Gaussian parameterized by a deep neural network (NN). Then, after T iterations of the sampler Q are parameterized by η, we arrive at $q_{\phi,\eta}$.

An alternative arises by ignoring the noise ξ [22], thus refining the initial variational approximation using only the stochastic gradient descent (SGD). Moreover, we can use Stein variational gradient descent (SVGD) [40] or a stochastic version [36] to apply repulsion between particles and promote more extensive explorations of the latent space.

3.2. *Approximating the Entropy Term*

We propose four approaches for the ELBO optimization which take structural advantage of the refined variational approximation.

3.2.1. Particle Approximation (VIS-P)

In this approach, we approximate the posterior $q_{\phi,\eta}(z|x)$ by a mixture of Dirac deltas (i.e., we approximate it with a finite set of particles), by sampling $z^{(1)}, \dots, z^{(M)} \sim q_{\phi,\eta}(z|x)$ and setting

$$q_{\phi,\eta}(z|x) = \frac{1}{M}\sum_{m=1}^{M} \delta(z - z^{(m)}).$$

In this approximation, the entropy term in (4) is set to zero. Consequently, the sample converges to the maximum posterior (MAP). This may be undesirable when training generative models, as the generated samples usually have little diversity. Thus, in subsequent computations, we add to the refined ELBO the entropy of the initial variational approximation, $\mathbb{E}_{q_{0,\phi}(z|x)}\left[\log q_{0,\phi}(z|x)\right]$, which serves as a regularizer alleviating the previous problem. When using SGD as the sampler, the resulting ELBO is tighter than that without refinement, as shown in Section 4.2.

3.2.2. MC Approximation (VIS-MC)

Instead of performing the full marginalization in Equation (3), we approximate it with $q_{\phi,\eta}(z_T, \dots, z_0|x) = \prod_{t=1}^{T} q_\eta(z_t|z_{t-1}) q_{0,\phi}(z_0|x)$; i.e., we consider the joint distribution for the refinement. However, in inference we only keep the z_T values. The entropy for each factor in this approximation is straightforward to compute. For example, for the SGLD case, we have

$$z_t = z_{t-1} + \eta \nabla \log p(x, z_{t-1}) + \mathcal{N}(0, 2\eta I), \qquad t = 1, ..., T.$$

This approximation tracks a better estimate of the entropy than VIS-P, as we are not completely discarding it; rather, for each t, we marginalize out the corresponding z_t using one sample.

3.2.3. Gaussian Approximation (VIS-G)

This approach is targeted at settings in which it could be helpful to have a posterior approximation that places density over the whole z space. In the specific case of using SGD as the inner kernel, we have

$$\begin{aligned} z_0 &\sim q_{0,\phi}(z_0|x) = \mathcal{N}(z_0|\mu_\phi(x), \sigma_\phi(x)) \\ z_t &= z_{t-1} + \eta \nabla \log p(x, z_{t-1}), \qquad t = 1, \dots, T. \end{aligned}$$

By treating the gradient terms as points, the refined variational approximation can be computed as $q_{\phi,\eta}(z|x) = \mathcal{N}(z|z_T, \sigma_\phi(x))$. Observe that there is an implicit dependence on η through z_T.

3.2.4. Fokker–Planck Approximation (VIS-FP)

Using the Fokker–Planck equation, we derive a deterministic sampler via iterations of the form

$$z_t = z_{t-1} + \eta(\nabla \log p(x, z_{t-1}) - \nabla \log q_t(z_{t-1})), \qquad t = 1, ..., T.$$

Then, we approximate the density $q_{\phi,\eta}(z|x)$ using a mixture of Dirac deltas. A detailed derivation of this approximation is given in Appendix A.

3.3. *Back-Propagating through the Sampler*

In standard VI, the variational approximation $q(z|x;\phi)$ is parameterized by ϕ. The parameters are learned employing SGD, or variants such as Adam [41], using the gradient $\nabla_\phi \text{ELBO}(q)$. We have shown how to embed a sampler inside the variational guide. It is therefore also possible to compute a gradient of the objective with respect to the sampler parameters η (see Section 3.1). For instance, we can compute a gradient $\nabla_\eta \text{ELBO}(q)$ with respect to the learning rate η from the SGLD or SGD processes to search for an optimal step size at every VI iteration. This is an additional step apart from using the gradient $\nabla_{\phi}\text{ELBO}(q)$ which is used to learn a good initial sampling distribution.

4. Analysis of Vis

Below, we highlight key properties of the proposed framework.

4.1. Consistency

The VIS framework is geared towards SG-MCMC samplers, where we can compute the gradients of sampler hyperparameters to speed up mixing time (a common major drawback in MCMC [42]). After back-propagating for a few iterations through the SG-MCMC sampler and learning a good initial distribution, one can resort to the learned sampler in the second phase, so standard consistency results from SG-MCMC apply as $T \to \infty$ [43].

4.2. Refinement of ELBO

Note that, for a refined guide using the VIS-P approximation and $M = 1$ samples, the refined objective function can be written as

$$\mathbb{E}_{q(z_0|x)}[\log p(x, z_0 + \eta\nabla \log p(x, z_0)) - \log q(z_0|x)]$$

noting that $z = z_0 + \eta\nabla \log p(x, z_0)$ when using SGD for $T = 1$ iterations. This is equivalent to the refined ELBO in (4). Since we are perturbing the latent variables in the steepest direction, we show easily that, for a moderate η, the previous bound is tighter than $\mathbb{E}_{q(z_0|x)}[\log p(x, z_0) - \log q(z_0|x)]$, the one for the original variational guide $q(z_0|x)$. This reformulation of ELBO is also convenient since it provides a clear way of implementing our refined variational inference framework in any probabilistic programming language (PPL) supporting algorithmic differentiation.

Respectively, for the VIS-FP case, we find that its deterministic flow follows the same trajectories as SGLD: based on standard results of MCMC samplers [44], we have

$$KL(q_{\phi,\eta}(z|x)||p(z|x)) \leq KL(q_{0,\phi}(z|x)||p(z|x)).$$

A similar reasoning applies to the VIS-MC approximation; however, it does not hold for VIS-G since it assumes that the posterior is Gaussian.

4.3. Taylor Expansion

This analysis applies only to VIS-P and VIS-FP. As stated in Section 4.2, within the VIS framework, optimizing the ELBO resorts to the performance of $\max_z \log p(x, z + \Delta z)$, where Δz is one iteration of the sampler; i.e., $\Delta z = \eta\nabla \log p(x, z)$ in the SGD case (VIS-P), or $\Delta z = \eta\nabla(\log p(x, z) - \log q(z))$ in the VIS-FP case. For notational clarity, we consider the case $T = 1$, although a similar analysis follows in a straightforward manner if more refinement steps are performed.

Consider a first-order Taylor expansion of the refined objective

$$\log p(x, z + \Delta z) \approx \log p(x, z) + (\Delta z)^{\intercal}\nabla \log p(x, z).$$

Taking gradients with respect to the latent variables z, we arrive at

$$\nabla_z \log p(x, z + \Delta z) \approx \nabla_z \log p(x, z) + \eta\nabla_z \log p(x, z)^{\intercal}\nabla_z^2 \log p(x, z),$$

where we have not computed the gradient through the Δz term (i.e., we treated it as a constant for simplification). Then, the refined gradient can be deemed to be the original gradient plus a second order correction. Instead of being modulated by a constant learning rate, this correction is adapted by the chosen sampler. The experiments in Section 5.4 show that this is beneficial for the optimization as it typically takes fewer iterations than the original variant to achieve lower losses.

By further taking gradients through the Δz term, we may tune the sampler parameters such as the learning rate as presented in Section 3.3. Consequently, the next subsection describes two differentiation modes.

4.4. Two Automatic Differentiation Modes for Refined ELBO Optimization

For the first variant, remember that the original variant can be rewritten (which we term Full AD) as

$$\mathbb{E}_q[\log p(x, z + \Delta z) - \log q(z + \Delta z|x)]. \tag{5}$$

We now define a stop gradient operator $\perp$ (which corresponds to `detach` in Pytorch or `stop_gradient` in tensorflow) that sets the gradient of its operand to zero—i.e., $\nabla_x \perp(x) = 0$—whereas in a forward pass, it acts as the identity function—that is, $\perp(x) = x$. With this, a variant of the ELBO objective (which we term Fast AD) is

$$\mathbb{E}_q[\log p(x, z + \perp(\Delta z)) - \log q(z + \perp(\Delta z)|x)]. \tag{6}$$

Full AD ELBO enables a gradient to be computed with respect to the sampler parameters inside Δz at the cost of a slight increase in computational burden. On the other hand, the Fast AD variant may be useful in numerous scenarios, as illustrated in the experiments.

Complexity

Since we need to back propagate through T iterations of an SG-MCMC scheme, using standard results of meta-learning and automatic differentiation [45], the time complexity of our more intensive approach (Full-AD) is $\mathcal{O}(mT)$, where m is the dimension of the hyperparameters (the learning rate of SG-MCMC and the latent dimension). Since for most use cases, the hyperparameters lie in a low-dimensional space, the approach is therefore scalable.

5. Experiments

The following experiments showcase the power of our approach as well as illustrating the the impact of various parameters on its performance, guiding their choice in practice. We also present a comparison with standard VIS and other recent variants, showing that the increased computational complexity of computing gradients through sampling steps is worth the gains in flexibility. Moreover, the proposed framework is compatible with other structured inference techniques, such as the sum–product algorithm, as well as serving to support other tasks such as classification.

Within the spirit of reproducible research, the code for VIS has been released at https://github.com/vicgalle/vis. The VIS framework is implemented with Pytorch [46], although we have also released a notebook for the first experiment using Jax to highlight the simple implementation of VIS. In any case, we emphasize that the approach facilitates rapid iterations over a large class of models.

5.1. Funnel Density

We first tested the framework on a synthetic yet complex target distribution. This experiment assessed whether VIS is suitable for modeling complex distributions. The target bi-dimensional density was defined through

$$\begin{aligned} z_1 &\sim \mathcal{N}(0, 1.35) \\ z_2 &\sim \mathcal{N}(0, \exp(z_1)). \end{aligned}$$

We adopted the usual diagonal Gaussian distribution as the variational approximation. For VIS, we used the VIS-P approximation and refined it for $T = 1$ steps using SGLD. Figure 1 top shows the trajectories of the lower bound for up to 50 iterations of variational optimization with Adam: our refined version achieved a tighter bound. The bottom figures present contour curves of the learned variational approximations. Observe that the VIS variant was placed closer to the mean of the true distribution and was more disperse than the original variational approximation, illustrating the fact that the refinement step helps in attaining more flexible posterior approximations.

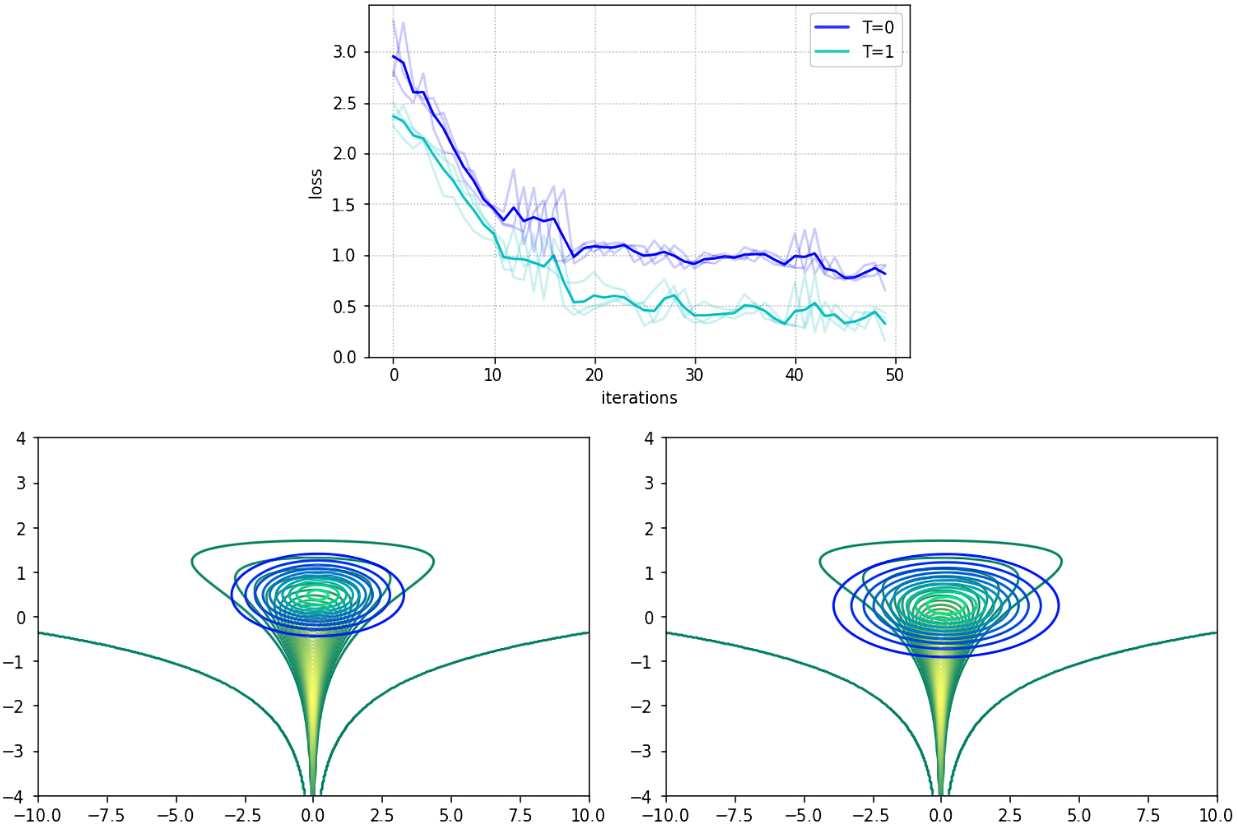

Figure 1. Top: Evolution of the negative evidence lower bound (ELBO) loss objective over 50 iterations. Darker lines depict means along different seeds (lighter lines). **Bottom left**: Contour curves (blue–turquoise) of the variational approximation with no refinement ($T = 0$) at iteration 30 (loss of 1.011). **Bottom right**: Contour curves (blue–turquoise) of refined variational approximation ($T = 1$) at iteration 30 (loss of 0.667). Green–yellow curves denote target density.

5.2. State-Space Markov Models

We tested our variational approximation on two state-space models: one for discrete data and another for continuous observations. These experiments also demonstrated that the framework is compatible with standard inference techniques such as the sum–product scheme from the Baum–Welch algorithm or Kalman filter. In both models, we performed inference on their parameters θ. All the experiments in this subsection used the Fast AD version (Section 4.4) as it was not necessary to further tune the sampler parameters to obtain competitive results. Full model implementations can be found in Appendix B.1, based on `funsor` (https://github.com/pyro-ppl/funsor/), a PPL on top of the `Pytorch` autodiff framework.

Hidden Markov Model (HMM): The model equations are

$$p(x_{1:\tau}, z_{1:\tau}, \theta) = \prod_{t=1}^{\tau} p(x_t | z_t, \theta_{em}) p(z_t | z_{t-1}, \theta_{tr}) p(\theta), \tag{7}$$

where each conditional is a categorical distribution taking five different classes. The prior is $p(\theta) = p(\theta_{em})p(\theta_{tr})$ based on two Dirichlet distributions that sample the observation and state transition probabilities, respectively.

Dynamic Linear Model (DLM): The model equations are as in (7), although the conditional distributions are now Gaussian and the parameters θ refer to the observation and transition variances.

For each model, we generated a synthetic dataset and used the refined variational approximation with $T = 0, 1, 2$. For the original variational approximation to the parameters θ, we used a Dirac delta. Performing VI with this approximation corresponded to MAP estimation using the Baum–Welch algorithm in the HMM case [47] and the Kalman filter in the DLM case [48], as we marginalized out the latent variables $z_{1:\tau}$. We used the VIS-P variant since it was sufficient to show performance gains in this case.

Figure 2 shows the results. The first row reports the experiments related to the HMM, the second row those for the DLM. We report the evolution of the log-likelihood during inference in all graphs; the first column reports the number of ELBO iterations, and the second column portrays clock times as the optimization takes place. They confirm that VIS ($T > 0$) achieved better results than standard VI ($T = 0$) for a comparable amount of time. Note also that there was not as much gain when changing from $T = 1$ to $T = 2$ as there is from $T = 0$ to $T = 1$, suggesting the need to carefully monitor this parameter. Finally, the top-right graph for the case $T = 0$ is shorter as it requires less clock time.

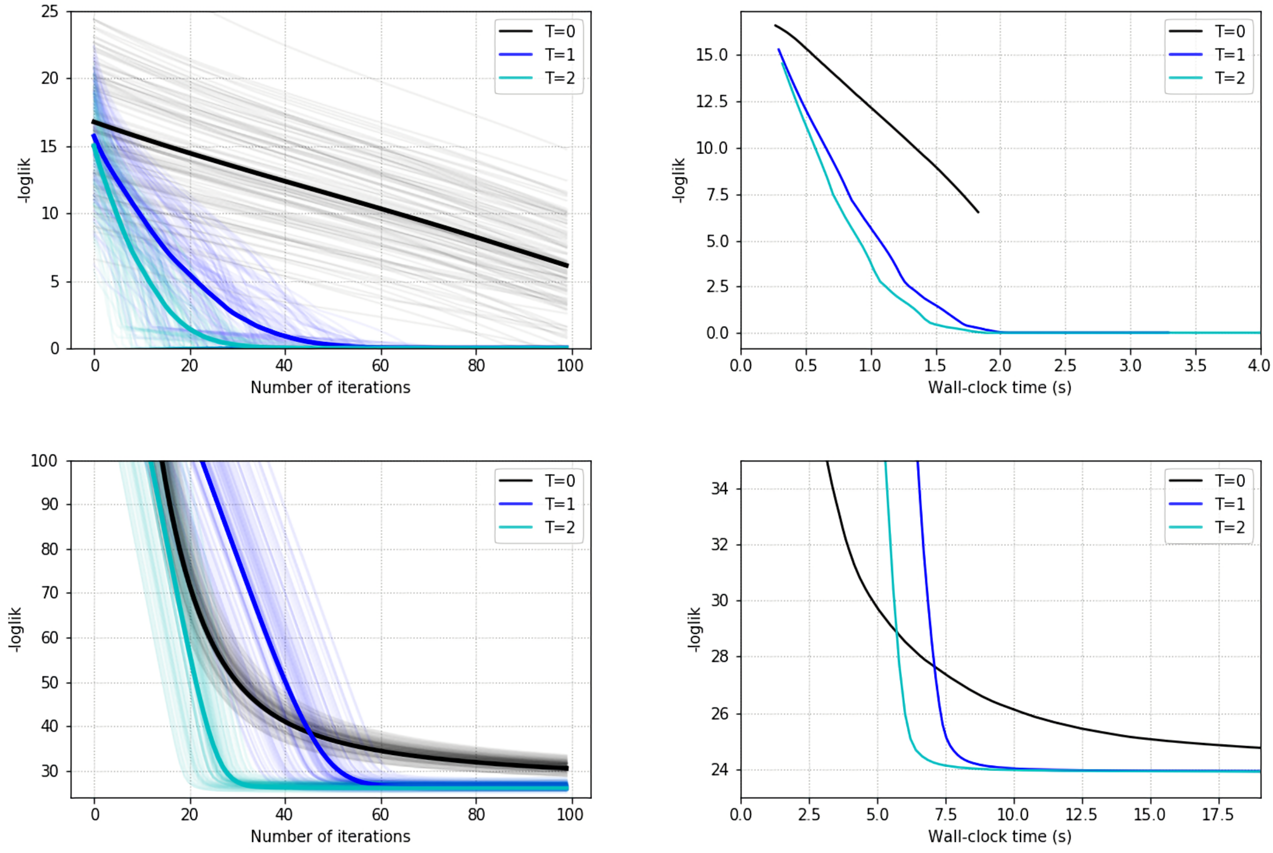

Figure 2. Results of ELBO optimization for state-space models. **Top-left** (Hidden Markov Model (HMM)): Log-likelihood against the number of ELBO gradient iterations. **Top-right** (HMM): Log-likelihood against clock time. **Bottom-left** (Dynamic Linear Model (DLM)): Log-likelihood against number of ELBO gradient iterations. **Bottom-right** (DLM): Log-likelihood against against clock time.

5.2.1. Prediction with an HMM

With the aim of assessing whether ELBO optimization helps in attaining better auxiliary scores, results in a prediction task are also reported. We generated a synthetic time series of alternating values of 0 and 1 for $\tau = 105$ timesteps. We trained the previous HMM model on the first 100 points and report in Table 1 the accuracy of the predictive distribution $p(y_t)$ averaged over the final five time-steps. We also report the predictive entropy as it helps in assessing the confidence of the model in its predictions, as a strictly proper scoring rule [49]. To guarantee the same computational budget time and a fair comparison, the model without refinement was run for 50 epochs (an epoch was a full iteration over the training dataset), whereas the model with refinement was run for 20 epochs. It can be observed that the refined model achieved higher accuracy than its counterpart. In addition, it was more correctly confident in its predictions.

Table 1. Prediction metrics for the HMM.

	$T = 0$	$T = 1$
accuracy	0.40	0.84
predictive entropy	1.414	1.056
logarithmic score	−1.044	−0.682

5.2.2. Prediction with a DLM

We tested the VIS framework on Mauna Loa monthly CO_2 time-series data [50]. We used the first 10 years as a training set, and we tested over the next 2 years. We used a DLM composed of a local linear trend plus a seasonal block of periodicity 12. Data were standardized to a mean of zero and standard deviation of one. To guarantee the same computational budget time, the model without refining was run for 10 epochs, whereas the model with refinement was run for 4 epochs. Table 2 reports the mean absolute error (MAE) and predictive entropy. In addition, we computed the interval score [49], as a strictly proper scoring rule. As can be seen, for similar clock times, the refined model not only achieved a lower MAE, but also its predictive intervals were narrower than the non-refined counterpart.

Table 2. Prediction metrics for the DLM.

	$T = 0$	$T = 1$
MAE	0.270	0.239
predictive entropy	2.537	2.401
interval score ($\alpha = 0.05$)	15.247	13.461

5.3. *Variational Autoencoder*

The third batch of experiments showed that VIS was competitive with respect to other algorithms from the recent literature, including unbiased implicit variational inference (UIVI [24]), semi-implicit variational inference (SIVI [25]), variational contrastive divergence (VCD [29]), and the HMC variant from [26], showing that our framework can outperform those approaches in similar experimental settings.

To this end, we tested the approach with a variational autoencoder (VAE) model [51]. The VAE defines a conditional distribution $p_\theta(x|z)$, generating an observation x from a latent variable z using parameters θ. For this task, our interest was in modeling the 28×28 image distributions underlying the MNIST [52] and the fashion-MNIST [53] datasets. To perform inference (i.e., to learn the parameters θ) the VAE introduces a variational approximation $q_\phi(z|x)$. In the standard setting, this distribution is Gaussian; we instead used the refined variational approximation comparing various values of T. We used the VIS-MC approximation (although we achieved similar results with VIS-G) with the Full AD variant given in Section 4.4.

For the experimental setup, we reproduced the setting in [24]. For $p_\theta(x|z)$, we used a factorized Bernoulli distribution parameterized by a two layer feed-forward network with 200 units in each layer and relu activations, except for a final sigmoid activation. As a variational approximation $q_\phi(z|x)$, we used a Gaussian with mean and (diagonal) covariance matrix parameterized by two distinct neural networks with the same structure as previously used, except for sigmoid activation for the mean and a softplus activation for the covariance matrix.

Results are reported in Table 3. To guarantee fair comparison, we trained the VIS-5-10 variant for 10 epochs, whereas all the other variants were trained for 15 epochs (fMNIST) or 20 epochs (MNIST), so that the VAE's performance was comparable to that reported in [24]. Although VIS was trained for fewer epochs, by increasing the number T of MCMC iterations, we dramatically improved the test log-likelihood. In terms of computational complexity, the average time per epoch using $T = 5$ was 10.46 s, whereas with no refinement ($T = 0$), the time was 6.10 s (which was the reason behind our decision to train the refined variant for fewer epochs): a moderate increase in computing time may be worth the dramatic increase in log-likelihood while not introducing new parameters into the model, except for the learning rate η.

Table 3. Test log-likelihood on binarized MNIST and fMNIST. Bold numbers indicate the best results. UIVI: unbiased implicit variational inference; SIVI: semi-implicit variational inference; VAE: variational autoencoder; VCD: variational contrastive divergence; HMC-DLGM: Hamiltonian Monte Carlo for Deep Latent Gaussian Models; VIS: variationally inferred sampler.

Method	MNIST	fMNIST
	Results from [24]	
UIVI	-94.09	-110.72
SIVI	-97.77	-121.53
VAE	-98.29	-126.73
	Results from [29]	
VCD	-95.86	-117.65
HMC-DLGM	-96.23	-117.74
	This paper	
VIS-5-10	$\mathbf{-82.74 \pm 0.19}$	$\mathbf{-105.08 \pm 0.34}$
VIS-0-10	-96.16 ± 0.17	-120.53 ± 0.59
VAE (VIS-0-0)	-100.91 ± 0.16	-125.57 ± 0.63

Finally, as a visual inspection of the VAE reconstruction quality trained with VIS, Figures 3 and 4, respectively, display 10 random samples of each dataset.

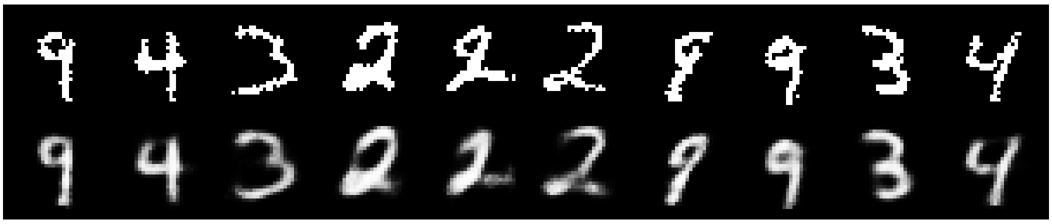

Figure 3. Top: original images from MNIST. Bottom: reconstructed images using VIS-5-10 at 10 epochs.

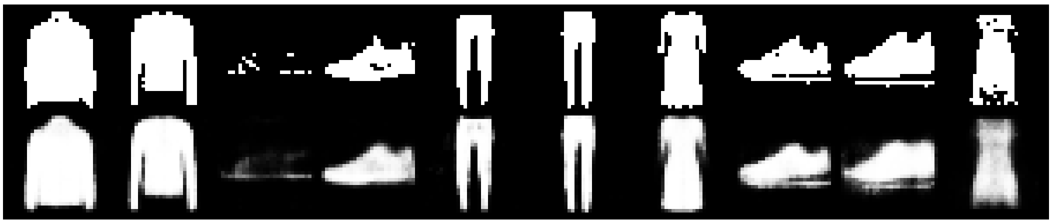

Figure 4. Top: original images from fMNIST. Bottom: reconstructed images using VIS-5-10 at 10 epochs.

5.4. Variational Autoencoder as a Deep Bayes Classifier

In the final experiments, we investigated whether VIS can deal with more general probabilistic graphical models and also perform well in other inference tasks such as classification. We explored the flexibility of the proposed scheme to solve inference problems in an experiment with a classification task in a high-dimensional setting with the MNIST dataset. More concretely, we extended the VAE model, conditioning it on a discrete variable $y \in \mathcal{Y} = \{0, 1, \ldots, 9\}$, leading to a conditional VAE (cVAE). The cVAE defined a decoder distribution $p_\theta(x|z, y)$ on an input space $x \in \mathbb{R}^D$ given a class label $y \in \mathcal{Y}$, latent variables $z \in \mathbb{R}^d$ and parameters θ. Figure 5 depicts the corresponding probabilistic graphic model. Additional details regarding the model architecture and hyperparameters are given in Appendix B.

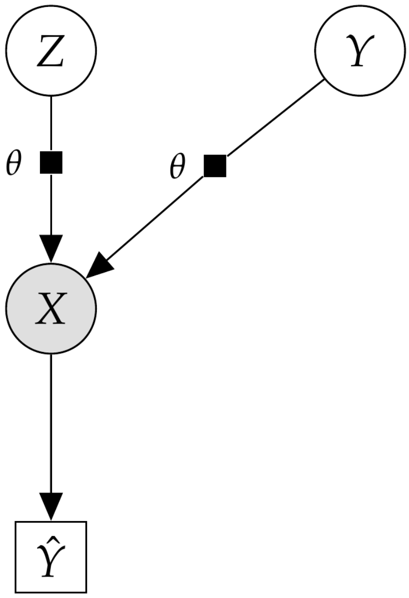

Figure 5. Probabilistic graphical model for the deep Bayes classifier.

To perform inference, a variational posterior was learned as an encoder $q_\phi(z|x, y)$ from a prior $p(z) \sim \mathcal{N}(0, I)$. Leveraging the conditional structure on y, we used the generative model as a classifier using the Bayes rule,

$$p(y|x) \propto p(y)p(x|y) = p(y) \int p_\theta(x|z, y) q_\phi(z|x, y) dz \approx \frac{1}{M} \sum_{m=1}^{M} p_\theta(x|z^{(m)}, y)p(y), \quad (8)$$

where we used M Monte Carlo samples $z^{(m)} \sim q_\phi(z|x, y)$. In the experiments, we set $M = 5$. Given a test sample x, the label $\hat{y}$ with the highest probability $p(y|x)$ is predicted.

For comparison, we performed several experiments changing T in the transition distribution $Q_{\eta,T}$ of the refined variational approximation. The results are given in Table 4, which reports the test accuracy at end of the refinement phase. Note that we are comparing different values of T depending on their use in refinement or inference phases (in the latter, the model and variational parameters were kept frozen). The model with $T_{ref} = 5$ was trained for 10 epochs, whereas the other settings were for 15 epochs, to give all settings a similar training time. Results were averaged over three runs with different random seeds. In all settings, we used the VIS-MC approximation for the entropy term. From the results, it is clear that the effect of using the refined variational approximation (the cases when $T > 0$) is crucially beneficial to achieve higher accuracy. The effect of learning a good initial distribution and inner learning rate by using the gradients $\nabla_\phi \text{ELBO}(q)$ and $\nabla_\eta \text{ELBO}(q)$ has a highly positive impact in the accuracy obtained.

On a final note, we have not included the case of only using an SGD or an SGLD sampler (i.e., without learning an initial distribution $q_{0,\phi}(z|x)$) since the results were much worse than those in Table 4 for a comparable computational budget. This strongly suggests that, for inference in high-dimensional, continuous latent spaces, learning a good initial distribution through VIS may accelerate mixing time dramatically.

Table 4. Results on digit classification task using a deep Bayes classifier.

T_{ref}	T_{inf}	Acc. (Test)
0	0	96.5 ± 0.5 %
0	10	97.7 ± 0.7 %
5	10	**99.8** ± **0.2** %

6. Conclusions

In this work, we have proposed a flexible and efficient framework to perform large-scale Bayesian inference in probabilistic models. The scheme benefits from useful properties and can be employed to efficiently perform inference with a wide class of models such as state-space time series, variational autoencoders and variants such as the conditioned VAE for classification tasks, defined through continuous, high-dimensional distributions.

The framework can be seen as a general approach to tuning MCMC sampler parameters, adapting the initial distributions and learning rate. Key to the success and applicability of the VIS framework are the ELBO approximations based on the introduced refined variational approximation, which are computationally cheap but convenient.

Better estimates of the refined density and its gradient may be a fruitful line of research, such as the spectral estimator used in [54]. Another alternative is to use a deterministic flow (such as SGD or SVGD), keeping track of the change in entropy at each iteration using the change of the variable formula, as in [55]. However, this requires a costly Jacobian computation, making it unfeasible to combine with our approach of back-propagation through the sampler (Section 3.3) for moderately complex problems. We leave this for future exploration. Another interesting and useful line of further research would be to tackle the case in which the latent variables z are discrete. This would entail adapting the automatic differentiation techniques to be able to back-propagate the gradients through the sequences of acceptance steps necessary in Metropolis–Hastings samplers.

In order to deal with the implicit variational density, it may be worthwhile to consider optimizing the Fenchel dual of the KL divergence, as in [31]. However, this requires the use of an auxiliary neural network, which may entail a large computational price compared with our simpler particle approximation.

Lastly, probabilistic programming offers powerful tools for Bayesian modeling. A PPL can be viewed as a programming language extended with random sampling and Bayesian conditioning capabilities, complemented with an inference engine that produces answers to inference, prediction and decision-making queries. Examples include WinBUGS [56], Stan [57] or the recent Edward [58] and Pyro [59] languages. We plan to adapt VIS into several PPLs to facilitate the adoption of the framework.

Author Contributions: Conceptualization, V.G. and D.R.I.; methodology, V.G.; software, V.G.; investigation, V.G.; writing—original draft preparation, V.G. and D.R.I.; writing—review and editing, V.G. and D.R.I. All authors have read and agreed to the published version of the manuscript.

Funding: VG acknowledges support from grant FPU16-05034. DRI is grateful to the MINECO MTM2017-86875-C3-1-R project and the AXA-ICMAT Chair in Adversarial Risk Analysis. Both authors acknowledge support from the Severo Ochoa Excellence Program CEX2019-000904-S. This material was based upon work partially supported by the National Science Foundation under Grant DMS-1638521 to the Statistical and Applied Mathematical Sciences Institute as well as a BBVA Foundation project.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Fokker-Planck Approximation (Vis-Fp)

The Fokker–Planck equation is a PDE that describes the temporal evolution of the density of a random variable under a (stochastic) gradient flow [39]. For a given SDE

$$dz = \mu(z,t)dt + \sigma(z,t)dB_t,$$

the corresponding Fokker–Planck equation is

$$\frac{\partial}{\partial t}q_t(z) = -\frac{\partial}{\partial z}[\mu(z,t)q_t(z)] + \frac{\partial^2}{\partial z^2}\left[\frac{\sigma^2(z,t)}{2}q_t(z)\right].$$

We are interested in converting the SGLD dynamics to a deterministic gradient flow.

Proposition A1. *The SGLD dynamics, given by the SDE*

$$dz = \nabla \log p(z)dt + \sqrt{2}dB_t,$$

have an equivalent deterministic flow, written as the ODE

$$dz = (\nabla \log p(z) - \nabla \log q_t(z))dt.$$

Proof. Let us write the Fokker–Planck equation for the respective flows. For the Langevin SDE, it is

$$\frac{\partial}{\partial t}q_t(z) = -\frac{\partial}{\partial z}\left[\nabla \log p(z)q_t(z)\right] + \frac{\partial^2}{\partial z^2}\left[q_t(z)\right].$$

On the other hand, the Fokker–Planck equation for the deterministic gradient flow is given by

$$\frac{\partial}{\partial t}q_t(z) = -\frac{\partial}{\partial z}\left[\nabla \log p(z)q_t(z)\right] + \frac{\partial}{\partial z}\left[\nabla \log q_t(z)q_t(z)\right].$$

The result immediately follows since $\frac{\partial}{\partial z}[\nabla \log q_t(z)q_t(z)] = \frac{\partial^2}{\partial z^2}[q_t(z)]$. □

Given that both flows are equivalent, we restrict our attention to the deterministic flow. Its discretization leads to iterations of the form

$$z_t = z_{t-1} + \eta(\nabla \log p(z_{t-1}) - \nabla \log q_{t-1}(z_{t-1})). \quad \text{(A1)}$$

In order to tackle the last term, we make the following particle approximation. Using a variational formulation, we have

$$-\nabla \log q(z) = \nabla\left(-\frac{\delta}{\delta q}\mathbb{E}_q[\log q]\right).$$

Then, we smooth the true density q convolving it with a kernel K, typically the rbf kernel, $K(z,z') = \exp\{-\gamma\|z-z'\|^2\}$, where γ is the bandwidth hyperparameter, leading to

$$\begin{aligned}\nabla\left(-\frac{\delta}{\delta q}\mathbb{E}_q[\log q]\right) &\approx \nabla\left(-\frac{\delta}{\delta q}\mathbb{E}_q[\log(q*K)]\right)\\ &= \nabla\log(q*K) - \nabla\left(\frac{q}{(q*K)}*K\right).\end{aligned}$$

If we consider a mixture of Dirac deltas, $q(z) = \frac{1}{M}\sum_{m=1}^{M}\delta(z-z_m)$, then the approximation is given by

$$-\nabla \log q(z) \approx -\frac{\sum_k \nabla_{z_m}K(z_m,z_n)}{\sum_n K(z_m,z_n)} - \sum_l \frac{\nabla_{z_m}K(z_m,z_l)}{\sum_n K(z_n,z_l)},$$

which can be inserted into Equation (A1). Finally, note that it is possible to back-propagate through this equation; i.e., the gradients of K can be explicitly computed.

Appendix B. Experiment Details

Appendix B.1. State-Space Models

Appendix B.1.1. Initial Experiments

For the HMM, both the observation and transition probabilities are categorical distributions, taking values in the domain $\{0,1,2,3,4\}$.

The equations of the DLM are

$$\begin{aligned}z_{t+1} &\sim \mathcal{N}(0.5z_t + 1.0, \sigma_{tr})\\ x_t &\sim \mathcal{N}(3.0z_t + 0.5, \sigma_{em}).\end{aligned}$$

with $z_0 = 0.0$.

Appendix B.1.2. Prediction Task in a DLM

The DLM model comprises a linear trend component plus a seasonal block with a period of 12. The trend is specified as

$$\begin{aligned}x_t &= z_{level,t} + \epsilon_t \qquad \epsilon_t \sim \mathcal{N}(0,\sigma_{obs})\\ z_{level,t} &= z_{level,t-1} + z_{slope,t-1} + \epsilon'_t \qquad \epsilon'_t \sim \mathcal{N}(0,\sigma_{level})\\ z_{slope,t} &= z_{slope,t-1} + \epsilon''_t \qquad \epsilon''_t \sim \mathcal{N}(0,\sigma_{slope}).\end{aligned}$$

With respect to the seasonal component, we specify it through

$$\begin{aligned}x_t &= Fz_t + v_t \qquad v_t \sim \mathcal{N}(0,\sigma_{obs})\\ z_t &= Gz_{t-1} + w_t \qquad w_t \sim \mathcal{N}(0,\sigma_{seas})\end{aligned}$$

where F is a 12-dimensional vector $(1,0,\dots,0,0)$ and G is the 12×12 matrix

$$G = \begin{bmatrix} 0 & 0 & \dots & 0 & 1\\ 1 & 0 & & 0 & 0\\ 0 & 1 & & 0 & 0\\ & & \ddots & & \\ 0 & 0 & & 1 & 0\end{bmatrix}.$$

Further details are in [60].

Appendix B.2. Vae

Appendix B.2.1. Model Details

The prior distribution $p(z)$ for the latent variables $z \in \mathbb{R}^{10}$ is a standard factorized Gaussian. The decoder distribution $p_\theta(x|z)$ and the encoder distribution (initial variational approximation) $q_{0,\phi}(z|x)$ are parameterized by two feed-forward neural networks, as detailed in Figure A1.

Appendix B.2.2. Hyperparameter Settings

The optimizer Adam is used in all experiments, with la earning rate of $\lambda = 0.001$. We also set $\eta = 0.001$. We train for 15 epochs (fMNIST) and 20 epochs (MNIST) to achieve a performance similar to the VAE in [24]. For the VIS-5-10 setting, we train only for 10 epochs to allow a fair computational comparison in terms of similar computing times.

Appendix B.3. cVAE

Appendix B.3.1. Model Details

The prior distribution $p(z)$ for the latent variables $z \in \mathbb{R}^{10}$ is a standard factorized Gaussian. The decoder distribution $p_\theta(x|y,z)$ and the encoder distribution (initial variational approximation) $q_{0,\phi}(z|x,y)$ are parameterized by two feed-forward neural networks whose details can be found in Figure A2. Equation (8) is approximated with one MC sample from the variational approximation in all experimental settings, as it allowed fast inference times while offering better results.

```
class VAE(nn.Module):
    def __init__(self):
        super(VAE, self).__init__()

        self.z_d = 10
        self.h_d = 200
        self.x_d = 28*28

        self.fc1_mu = nn.Linear(self.x_d, self.h_d)
        self.fc1_cov = nn.Linear(self.x_d, self.h_d)
        self.fc12_mu = nn.Linear(self.h_d, self.h_d)
        self.fc12_cov = nn.Linear(self.h_d, self.h_d)
        self.fc2_mu = nn.Linear(self.h_d, self.z_d)
        self.fc2_cov = nn.Linear(self.h_d, self.z_d)

        self.fc3 = nn.Linear(self.z_d, self.h_d)
        self.fc32 = nn.Linear(self.h_d, self.h_d)
        self.fc4 = nn.Linear(self.h_d, self.x_d)

    def encode(self, x):
        h1_mu = F.relu(self.fc1_mu(x))
        h1_cov = F.relu(self.fc1_cov(x))
        h1_mu = F.relu(self.fc12_mu(h1_mu))
        h1_cov = F.relu(self.fc12_cov(h1_cov))
        # we work in the logvar-domain
        return self.fc2_mu(h1_mu),
        torch.log(F.softplus(self.fc2_cov(h1_cov)))

    def decode(self, z):
        h3 = F.relu(self.fc3(z))
        h3 = F.relu(self.fc32(h3))
        return torch.sigmoid(self.fc4(h3))
```

Figure A1. Model architecture for the VAE.

```
class cVAE(nn.Module):
    def __init__(self):
        super(cVAE, self).__init__()

        self.z_d = 10
        self.h_d = 200
        self.x_d = 28*28
        num_classes = 10

        self.fc1_mu = nn.Linear(self.x_d + num_classes, self.h_d)
        self.fc1_cov = nn.Linear(self.x_d + num_classes, self.h_d)
        self.fc12_mu = nn.Linear(self.h_d, self.h_d)
        self.fc12_cov = nn.Linear(self.h_d, self.h_d)
        self.fc2_mu = nn.Linear(self.h_d, self.z_d)
        self.fc2_cov = nn.Linear(self.h_d, self.z_d)

        self.fc3 = nn.Linear(self.z_d + num_classes, self.h_d)
        self.fc32 = nn.Linear(self.h_d, self.h_d)
        self.fc4 = nn.Linear(self.h_d, self.x_d)

    def encode(self, x, y):
        h1_mu = F.relu(self.fc1_mu(torch.cat([x, y], dim=-1)))
        h1_cov = F.relu(self.fc1_cov(torch.cat([x, y], dim=-1)))
        h1_mu = F.relu(self.fc12_mu(h1_mu))
        h1_cov = F.relu(self.fc12_cov(h1_cov))
        # we work in the logvar-domain
        return self.fc2_mu(h1_mu),
        torch.log(F.softplus(self.fc2_cov(h1_cov)))

    def decode(self, z, y):
        h3 = F.relu(self.fc3(torch.cat([z, y], dim=-1)))
        h3 = F.relu(self.fc32(h3))
        return torch.sigmoid(self.fc4(h3))
```

Figure A2. Model architecture for the cVAE.

Appendix B.3.2. Hyperparameter Settings

The optimizer Adam was used in all experiments, with a learning rate of $\lambda = 0.01$. We set the initial $\eta = 5 \times 10^{-5}$.

References

1. Blei, D.M.; Kucukelbir, A.; McAuliffe, J.D. Variational inference: A review for statisticians. *J. Am. Stat. Assoc.* **2017**, *112*, 859–877. [CrossRef]
2. Insua, D.; Ruggeri, F.; Wiper, M. *Bayesian Analysis of Stochastic Process Models*; John Wiley & Sons: New York, NY, USA, 2012; Volume 978.
3. Alquier, P. Approximate Bayesian Inference. *Entropy* **2020**, *22*, 1272. [CrossRef] [PubMed]
4. Kucukelbir, A.; Tran, D.; Ranganath, R.; Gelman, A.; Blei, D.M. Automatic differentiation variational inference. *J. Mach. Learn. Res.* **2017**, *18*, 430–474.
5. Riquelme, C.; Johnson, M.; Hoffman, M. Failure modes of variational inference for decision making. In Proceedings of the Prediction and Generative Modeling in RL Workshop (AAMAS, ICML, IJCAI), Stockholm, Sweden, 15 July 2018.
6. Andrieu, C.; Doucet, A.; Holenstein, R. Particle Markov chain Monte Carlo methods. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **2010**, *72*, 269–342. [CrossRef]
7. Neal, R.M. MCMC using Hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*; CRC Press: Boca Raton, FL, USA, 2011; Volume 2, p. 2.
8. Van Ravenzwaaij, D.; Cassey, P.; Brown, S.D. A simple introduction to Markov Chain Monte–Carlo sampling. *Psychon. Bull. Rev.* **2018**, *25*, 143–154. [CrossRef] [PubMed]
9. Nalisnick, E.; Hertel, L.; Smyth, P. Approximate inference for deep latent gaussian mixtures. In Proceedings of the NIPS Workshop on Bayesian Deep Learning, Barcelona, Spain, 10 December 2016.
10. Salimans, T.; Kingma, D.; Welling, M. Markov chain Monte Carlo and variational inference: Bridging the gap. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 1218–1226.
11. Tran, D.; Ranganath, R.; Blei, D.M. The variational Gaussian process. In Proceedings of the 4th International Conference on Learning Representations, San Juan, Puerto Rico, 2–4 May 2016.
12. Wood, F.; Meent, J.W.; Mansinghka, V. A new approach to probabilistic programming inference. In Proceedings of the Artificial Intelligence and Statistics, Reykjavik, Iceland, 22–25 April 2014; pp. 1024–1032.
13. Ge, H.; Xu, K.; Ghahramani, Z. Turing: a language for flexible probabilistic inference. In Proceedings of the International Conference on Artificial Intelligence and Statistics, Lanzarote, Spain, 9–11 April 2018; pp. 1682–1690.
14. Papaspiliopoulos, O.; Roberts, G.O.; Sköld, M. A general framework for the parametrization of hierarchical models. *Stat. Sci.* **2007**, *22*, 59–73. [CrossRef]

15. Hoffman, M.; Sountsov, P.; Dillon, J.V.; Langmore, I.; Tran, D.; Vasudevan, S. Neutra-lizing bad geometry in hamiltonian Monte Carlo using neural transport. *arXiv* **2019**, arXiv:1903.03704.
16. Li, S.H.; Wang, L. Neural Network Renormalization Group. *Phys. Rev. Lett.* **2018**, *121*, 260601. [CrossRef]
17. Parno, M.; Marzouk, Y. Transport map accelerated markov chain monte carlo. *arXiv* **2014**, arXiv:1412.5492.
18. Rezende, D.; Mohamed, S. Variational Inference with Normalizing Flows. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 1530–1538.
19. Chen, C.; Li, C.; Chen, L.; Wang, W.; Pu, Y.; Carin, L. Continuous-Time Flows for Efficient Inference and Density Estimation. In Proceedings of the International Conference on Machine Learning, Vienna, Austria, 25–31 July 2018.
20. Liu, G.; Liu, Y.; Guo, M.; Li, P.; Li, M. Variational inference with Gaussian mixture model and householder flow. *Neural Netw.* **2019**, *109*, 43–55. [CrossRef]
21. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial nets. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, USA, 8–13 December 2014; pp. 2672–2680.
22. Mandt, S.; Hoffman, M.D.; Blei, D.M. Stochastic Gradient Descent as Approximate Bayesian Inference. *J. Mach. Learn. Res.* **2017**, *18*, 4873–4907.
23. Huszár, F. Variational inference using implicit distributions. *arXiv* **2017**, arXiv:1702.08235.
24. Titsias, M.K.; Ruiz, F. Unbiased Implicit Variational Inference. In Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics, Naha, Japan, 16–18 April 2019; pp. 167–176.
25. Yin, M.; Zhou, M. Semi-Implicit Variational Inference. *arXiv* **2018**, arXiv:1805.11183.
26. Hoffman, M.D. Learning deep latent Gaussian models with Markov chain Monte Carlo. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 22–31 July 2017; pp. 1510–1519.
27. Feng, Y.; Wang, D.; Liu, Q. Learning to draw samples with amortized stein variational gradient descent. *arXiv* **2017**, arXiv:1707.06626.
28. Cremer, C.; Li, X.; Duvenaud, D. Inference suboptimality in variational autoencoders. *arXiv* **2018**, arXiv:1801.03558.
29. Ruiz, F.; Titsias, M. A Contrastive Divergence for Combining Variational Inference and MCMC. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 10–15 June 2019; pp. 5537–5545.
30. Dai, B.; Dai, H.; He, N.; Liu, W.; Liu, Z.; Chen, J.; Xiao, L.; Song, L. Coupled variational bayes via optimization embedding. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, USA, 3–8 December 2018; pp. 9690–9700.
31. Fang, L.; Li, C.; Gao, J.; Dong, W.; Chen, C. Implicit Deep Latent Variable Models for Text Generation. *arXiv* **2019**, arXiv:1908.11527.
32. Welling, M.; Teh, Y.W. Bayesian learning via stochastic gradient Langevin dynamics. In Proceedings of the 28th International Conference on Machine Learning (ICML-11), Montreal, QC, USA, 11–13 June 2014; pp. 681–688.
33. Li, C.; Chen, C.; Carlson, D.; Carin, L. Preconditioned stochastic gradient Langevin dynamics for deep neural networks. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016.
34. Li, C.; Chen, C.; Fan, K.; Carin, L. High-order stochastic gradient thermostats for Bayesian learning of deep models. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016.
35. Abbati, G.; Tosi, A.; Osborne, M.; Flaxman, S. Adageo: Adaptive geometric learning for optimization and sampling. In Proceedings of the International Conference on Artificial Intelligence and Statistics, Canary Islands, Spain, 9–11 April 2018; pp. 226–234.
36. Gallego, V.; Insua, D.R. Stochastic Gradient MCMC with Repulsive Forces. *arXiv* **2018**, arXiv:1812.00071.
37. Ma, Y.A.; Chen, T.; Fox, E. A complete recipe for stochastic gradient MCMC. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 2917–2925.
38. Baydin, A.G.; Pearlmutter, B.A.; Radul, A.A.; Siskind, J.M. Automatic differentiation in machine learning: A survey. *J. Mach. Learn. Res.* **2017**, *18*, 5595–5637.
39. Pavliotis, G. Stochastic Processes and Applications: Diffusion Processes, the Fokker-Planck and Langevin Equations. In *Texts in Applied Mathematics*; Springer: New York, NY, USA, 2014.
40. Liu, Q.; Wang, D. Stein variational gradient descent: A general purpose Bayesian inference algorithm. In Proceedings of the Advances In Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 2378–2386.
41. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
42. Graves, T.L. Automatic step size selection in random walk Metropolis algorithms. *arXiv* **2011**, arXiv:1103.5986.
43. Brooks, S.; Gelman, A.; Jones, G.; Meng, X.L. *Handbook of Markov Chain Monte Carlo*; CRC Press: Boca Raton, FL, USA, 2011.
44. Murray, I.; Salakhutdinov, R. Notes on the KL-Divergence between a Markov Chain and Its Equilibrium Distribution; 2008. Available online: http://www.cs.toronto.edu/~rsalakhu/papers/mckl.pdf (accessed on 12 June 2020).
45. Franceschi, L.; Donini, M.; Frasconi, P.; Pontil, M. Forward and reverse gradient-based hyperparameter optimization. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 22–31 July 2017; pp. 1165–1173.
46. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*; Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: Granada, Spain, 2019; pp. 8024–8035.
47. Rabiner, L.R. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* **1989**, *77*, 257–286. [CrossRef]

48. Zarchan, P.; Musoff, H. *Fundamentals of Kalman filtering: A Practical Approach*; American Institute of Aeronautics and Astronautics, Inc.: Washington, DC, USA, 2013.
49. Gneiting, T.; Raftery, A.E. Strictly proper scoring rules, prediction, and estimation. *J. Am. Stat. Assoc.* **2007**, *102*, 359–378. [CrossRef]
50. Keeling, C.D. *Atmospheric Carbon Dioxide Record from Mauna Loa*; Scripps Institution of Oceanography, The University of California: La Jolla, CA, USA, 2005.
51. Kingma, D.P.; Welling, M. Auto-encoding variational bayes. *arXiv* **2013**, arXiv:1312.6114.
52. LeCun, Y.; Cortes, C. MNIST handwritten Digit Database. Available online: http://yann.lecun.com/exdb/mnist/ (accessed on 12 May 2020).
53. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv* **2017**, arXiv:1708.07747.
54. Shi, J.; Sun, S.; Zhu, J. A Spectral Approach to Gradient Estimation for Implicit Distributions. In Proceedings of the International Conference on Machine Learning, Vienna, Austria, 25–31 July 2018; pp. 4651–4660.
55. Duvenaud, D.; Maclaurin, D.; Adams, R. Early stopping as nonparametric variational inference. In Proceedings of the Artificial Intelligence and Statistics, Cadiz, Spain, 9–11 May 2016; pp. 1070–1077.
56. Lunn, D.J.; Thomas, A.; Best, N.; Spiegelhalter, D. WinBUGS-a Bayesian modelling framework: Concepts, structure, and extensibility. *Stat. Comput.* **2000**, *10*, 325–337. [CrossRef]
57. Carpenter, B.; Gelman, A.; Hoffman, M.D.; Lee, D.; Goodrich, B.; Betancourt, M.; Brubaker, M.; Guo, J.; Li, P.; Riddell, A. Stan: A probabilistic programming language. *J. Stat. Softw.* **2017**, *76*. [CrossRef]
58. Tran, D.; Hoffman, M.W.; Moore, D.; Suter, C.; Vasudevan, S.; Radul, A. Simple, distributed, and accelerated probabilistic programming. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 3–8 December 2018; pp. 7609–7620.
59. Bingham, E.; Chen, J.P.; Jankowiak, M.; Obermeyer, F.; Pradhan, N.; Karaletsos, T.; Singh, R.; Szerlip, P.; Horsfall, P.; Goodman, N.D. Pyro: Deep Universal Probabilistic Programming. *arXiv* **2018**, arXiv:1810.09538.
60. West, M.; Harrison, J. *Bayesian Forecasting and Dynamic Models*; Springer: New York, NY, USA, 2006.