A General Method for Amortizing Variational Filtering

Joseph Marino, Milan Cvitkovic, Yisong Yue
California Institute of Technology
{jmarino, mcvitkovic, yyue}@caltech.edu

Abstract

We introduce the *variational filtering EM algorithm*, a simple, general-purpose method for performing variational inference in dynamical latent variable models using information from only past and present variables, i.e. filtering. The algorithm is derived from the variational objective in the filtering setting and consists of an optimization procedure at each time step. By performing each inference optimization procedure with an iterative amortized inference model, we obtain a computationally efficient implementation of the algorithm, which we call *amortized variational filtering*. We present experiments demonstrating that this general-purpose method improves performance across several deep dynamical latent variable models.

1 Introduction

Complex tasks with time-series data, like audio comprehension or robotic manipulation, must often be performed online, where the model can only consider past and present information. Models for such tasks, e.g. Hidden Markov Models, frequently operate by inferring the hidden state of the world at each time-step. This type of online inference procedure is known as *filtering*. Learning filtering models purely through supervised labels or rewards can be impractical, requiring massive collections of labeled data or significant efforts at reward shaping. In contrast, generative models can learn and infer hidden structure and states directly from data. Deep latent variable models [19, 28, 38], in particular, offer a promising direction; they infer latent representations using expressive deep networks, commonly using variational methods to perform inference [25]. Recent works have extended deep latent variable models to the time-series setting, e.g. [7, 13]. However, inference procedures for these dynamical models have been proposed on the basis of intuition rather than from a rigorous inference optimization perspective, potentially limiting performance.

We introduce *variational filtering EM*, an algorithm for performing filtering variational inference and learning that is rigorously derived from the variational objective. As detailed below, the variational objective in the filtering setting results in a sequence of inference optimization objectives, with one at each time-step. By initializing each of these inference optimization procedures from the corresponding prior distribution, a classic Bayesian prediction-update loop naturally emerges. This contrasts with existing filtering approaches for deep dynamical models, which use inference models that do not explicitly account for prior predictions during inference. However, using iterative inference models [33], which overcome this limitation, we develop a computationally efficient implementation of the variational filtering EM algorithm, which we refer to as *amortized variational filtering* (AVF).

The main contributions of this paper are the variational filtering EM algorithm and its amortized implementation, AVF. This general-purpose filtering algorithm is widely applicable to dynamical latent variable models, as we demonstrate in our experiments. Moreover, the variational filtering EM algorithm is derived from the filtering variational objective, providing a solid theoretical framework for filtering inference. By precisely specifying the inference optimization procedure, this method takes a simple form compared to previous hand-designed methods. Using several deep dynamical

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

latent variable models, we demonstrate that this filtering approach compares favorably against current methods across a variety of benchmark sequence data sets.

2 Background

Section 2.1 provides the general form of a dynamical latent variable model. Section 2.2 covers variational inference. Deep latent variable models are often trained efficiently by amortizing inference optimization (Section 2.3). Applying this technique to dynamical models is non-trivial, leading many prior works to use hand-designed amortized inference methods (Section 2.4).

2.1 Dynamical latent variable models

A sequence of T observations, $\mathbf{x}_{\leq T}$, can be modeled using a dynamical latent variable model, $p_\theta(\mathbf{x}_{\leq T}, \mathbf{z}_{\leq T})$, which models the joint distribution between $\mathbf{x}_{\leq T}$ and a sequence of latent variables, $\mathbf{z}_{\leq T}$, with parameters θ. It is typically assumed that $p_\theta(\mathbf{x}_{\leq T}, \mathbf{z}_{\leq T})$ can be factorized into conditional joint distributions at each step, $p_\theta(\mathbf{x}_t, \mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})$, which are conditioned on preceding variables. This results in the following auto-regressive formulation:

$$p_\theta(\mathbf{x}_{\leq T}, \mathbf{z}_{\leq T}) = \prod_{t=1}^{T} p_\theta(\mathbf{x}_t, \mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t}) = \prod_{t=1}^{T} p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}, \mathbf{z}_{\leq t}) p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t}). \tag{1}$$

$p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}, \mathbf{z}_{\leq t})$ is the *observation* model, and $p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})$ is the *dynamics* model, both of which can be arbitrary functions of their conditioning variables. However, while Eq. 1 provides the general form of a dynamical latent variable model, further assumptions about the dependency structure, e.g. Markov, or functional forms, e.g. linear, are often necessary for tractability.

2.2 Variational inference

Given a model and a set of observations, we typically want to *infer* the posterior for each sequence, $p(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})$, and *learn* the model parameters, θ. Inference can be performed online or offline through Bayesian filtering or smoothing respectively [39], and learning can be performed through maximum likelihood estimation. Unfortunately, inference and learning are intractable for all but the simplest model classes. For non-linear functions, which are present in *deep* latent variable models, we must resort to approximate inference. Variational inference [25] reformulates inference as optimization by introducing an approximate posterior, $q(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})$, then minimizing the KL-divergence to the true posterior, $p(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})$. To avoid evaluating $p(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})$, one can express the KL-divergence as

$$D_{KL}(q(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})||p(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})) = \log p_\theta(\mathbf{x}_{\leq T}) + \mathcal{F}, \tag{2}$$

where $\mathcal{F}$ is the *variational free energy*, also referred to as the (negative) *evidence lower bound* or ELBO, defined as

$$\mathcal{F} \equiv -\mathbb{E}_{q(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})}\left[\log \frac{p_\theta(\mathbf{x}_{\leq T}, \mathbf{z}_{\leq T})}{q(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})}\right]. \tag{3}$$

In Eq. 2, $\log p_\theta(\mathbf{x}_{\leq T})$ is independent of $q(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})$, so one can minimize the KL-divergence to the true posterior, thereby performing approximate inference, by minimizing $\mathcal{F}$ w.r.t. $q(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})$. Further, as KL-divergence is non-negative, Eq. 2 implies that free energy upper bounds the negative log likelihood. Therefore, upon minimizing $\mathcal{F}$ w.r.t. $q(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})$, one can use the gradient $\nabla_\theta \mathcal{F}$ to learn the model parameters. These two optimization procedures are respectively the expectation and maximization steps of the variational EM algorithm [35], which alternate until convergence. To scale this algorithm, stochastic gradients can be used for both inference [36] and learning [22].

2.3 Amortized variational inference

Performing inference optimization using conventional stochastic gradient descent techniques can be computationally demanding, potentially requiring many inference iterations. To increase efficiency, a separate inference model can learn to map data examples to approximate posterior estimates [9, 19, 28, 38], thereby amortizing inference across examples [17]. Denoting the distribution parameters of q as $\boldsymbol{\lambda}^q$ (e.g. Gaussian mean and variance), standard inference models take the form

$$\boldsymbol{\lambda}^q \leftarrow f_\phi(\mathbf{x}), \tag{4}$$

where the inference model is denoted as f with parameters ϕ. These models, though efficient, have limitations. Notably, because these models only receive the data as input, they are unable to account for empirical priors, which occur from one latent variable to another. Such priors arise in the dynamics of dynamical models, forming priors across time steps, as well as in hierarchical models, forming priors across levels. Previous works have neglected to include empirical priors during inference, attempting to overcome this limitation through heuristics, like "top-down" inference in hierarchical models [41] and recurrent inference models in dynamical models, e.g. [7].

Iterative inference models [33] directly account for these priors, instead performing inference optimization by iteratively encoding approximate posterior estimates and gradients:

$$\boldsymbol{\lambda}^q \leftarrow f_\phi(\boldsymbol{\lambda}^q, \nabla_{\boldsymbol{\lambda}^q}\mathcal{F}). \tag{5}$$

The gradients, $\nabla_{\boldsymbol{\lambda}^q}\mathcal{F}$, can be estimated through black box methods [36] or the reparameterization trick [28, 38] when applicable. Analogously to learning to learn [1], iterative inference models learn to perform inference optimization, thereby *learning to infer*. Eq. 5 provides a viable encoding form for an iterative inference model, but other forms, such as additionally encoding the data, $\mathbf{x}$, can potentially lead to faster inference convergence. Empirically, iterative inference models have also been shown to yield improved modeling performance over comparable standard models [33].

2.4 Related work

Many deterministic deep dynamical latent variable models have been proposed for sequential data [6, 42, 31, 11]. While these models often capture many aspects of the data, they cannot account for the uncertainty inherent in many domains, typically arising from partial observability of the environment. By averaging over multi-modal distributions, these models often produce samples in regions of low probability, e.g. blurry video frames. This inadequacy necessitates moving to probabilistic models, which can explicitly model uncertainty to accurately capture the distribution of possible sequences.

Amortized variational inference [28, 38] has enabled many recently proposed probabilistic deep dynamical latent variable models, with applications to video [44, 27, 45, 24, 16, 12, 3, 10, 30, 20], speech [7, 13, 18, 23, 30], handwriting [7], music [13], etc. While these models differ in their functional mappings, most fall within the general form of Eq. 1. Crucially, simply encoding the observation at each step is insufficient to accurately perform approximate inference, as the prior can vary across steps. Thus, with each model, a hand-crafted amortized inference procedure has been proposed. For instance, many filtering inference methods re-use various components of the generative model [7, 13, 16, 10], while some methods introduce separate recurrent neural networks into the filtering procedure [4, 10] or encode the previous latent sample [27]. Specifying a filtering method has been an engineering effort, as we have lacked a theoretical framework.

The variational filtering EM algorithm precisely specifies the inference optimization procedure implied by the filtering variational objective. The main insight from this analysis is that, having drawn approximate posterior samples at previous steps, inference becomes a local optimization, depending only on the current prior and observation. This suggests one unified approach that explicitly performs inference optimization at each step, replacing the current collection of custom filtering methods. When the approximate posterior at each step is initialized at the corresponding prior, this approach entails a Bayesian prediction-update loop, with the update composed of a gradient (error) signal.

Perhaps the closest technique in the probabilistic modeling literature is the "residual" inference method from Fraccaro *et al.* [13], which updates the approximate posterior mean from the prior. Similar ideas have been proposed on an empirical basis for deterministic models [31, 21]. PredNet [31] is a deterministic model that encodes prediction errors to perform inference. This approach is inspired by predictive coding [37, 14], a theory from neuroscience that postulates that feedforward pathways in sensory processing areas of the brain use prediction errors to update state estimates from prior predictions. In turn, this theory is motivated by classical Bayesian filtering [39], which updates the posterior from the prior using the likelihood of the prediction. For linear Gaussian models, this manifests as the Kalman filter [26], which uses prediction errors to perform exact inference.

Finally, several recent works have used particle filtering in conjunction with amortized inference to provide a tighter lower bound on the log likelihood for sequential data [32, 34, 29]. The techniques developed here can also be applied to this tighter bound.

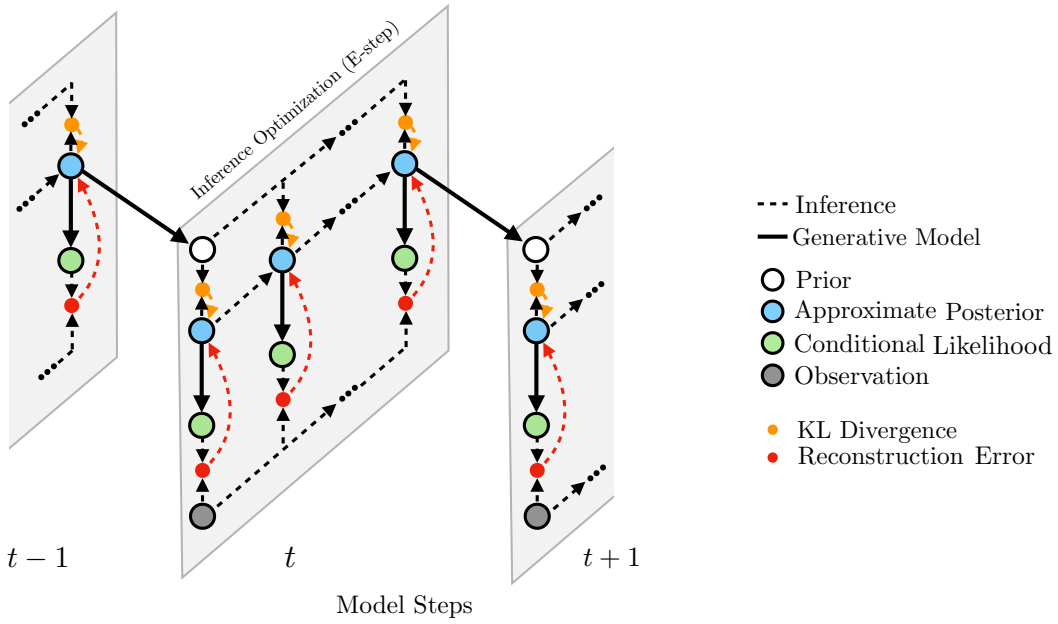


Figure 1: **Variational filtering EM.** The diagram shows filtering inference within a dynamical latent variable model, as outlined in Algorithm 1. The central gray region depicts inference optimization of the approximate posterior, $q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})$, at step t, which can be initialized at or near the corresponding prior, $p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})$. Sampling from the approximate posterior generates the conditional likelihood, $p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}, \mathbf{z}_{\leq t})$, which is evaluated at the observation, $\mathbf{x}_t$, to calculate the reconstruction error. This term is combined with the KL divergence between the approximate posterior and prior, yielding the step free energy, $\mathcal{F}_t$ (Eq. 9). Inference optimization (E-step) involves finding the approximate posterior that minimizes the step free energy terms. Learning (M-step), which is not shown, corresponds to updating the model parameters, θ, to minimize the total free energy, $\mathcal{F}$.

3 Variational filtering

Section 3.1 describes variational filtering EM (Algorithm 1), a general algorithm for performing filtering variational inference in dynamical latent variable models. In Section 3.2, we introduce a method for amortizing inference optimization using iterative inference models.

3.1 Variational filtering expectation maximization (EM)

In the filtering setting, the approximate posterior at each step is conditioned only on information from past and present variables, enabling *online* approximate inference. This implies a structured approximate posterior, in which $q(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})$ factorizes across steps as

$$q(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T}) = \prod_{t=1}^{T} q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t}). \tag{6}$$

Note that the conditioning variables in each term of q denote an *indirect* dependence that arises through free energy minimization and does not necessarily constitute a direct functional mapping. Under a filtering approximate posterior, the free energy (Eq. 3) can be expressed as

$$\mathcal{F} = \sum_{t=1}^{T} \mathbb{E}_{\prod_{\tau=1}^{t-1} q(\mathbf{z}_\tau|\mathbf{x}_{\leq \tau}, \mathbf{z}_{<\tau})} [\mathcal{F}_t] = \sum_{t=1}^{T} \tilde{\mathcal{F}}_t, \tag{7}$$

(see Appendix A for the derivation) where $\mathcal{F}_t$ is the *step free energy*, defined as

$$\mathcal{F}_t \equiv -\mathbb{E}_{q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})} \left[\log \frac{p_\theta(\mathbf{x}_t, \mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})}{q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})} \right], \tag{8}$$

Algorithm 1 Variational Filtering Expectation Maximization

Input: observation sequence $\mathbf{x}_{1:T}$, model $p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T})$
$\nabla_\theta \mathcal{F} = 0$ ▷ parameter gradient
for $t = 1$ **to** T **do**
  initialize $q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})$ ▷ at/near $p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})$
  $\tilde{\mathcal{F}}_t := \mathbb{E}_{q(\mathbf{z}_{<t}|\mathbf{x}_{<t}, \mathbf{z}_{<t-1})}[\mathcal{F}_t]$
  $q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t}) = \arg\min_q \tilde{\mathcal{F}}_t$ ▷ inference (E-Step)
  $\nabla_\theta \mathcal{F} = \nabla_\theta \mathcal{F} + \nabla_\theta \tilde{\mathcal{F}}_t$
end for
$\theta = \theta - \alpha \nabla_\theta \mathcal{F}$ ▷ learning (M-Step)

and we have also defined $\tilde{\mathcal{F}}_t$ as the t^{th} term in the summation. Note that with a single step, the filtering free energy reduces to the first step free energy, thereby recovering the static case. As in this setting, the step free energy can be re-expressed as a reconstruction term and a KL-divergence term:

$$\mathcal{F}_t = -\mathbb{E}_{q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})}[\log p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}, \mathbf{z}_{\leq t})] + D_{KL}(q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})||p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})). \quad (9)$$

The filtering free energy in Eq. 7 is the sum of these step free energy terms, each of which is evaluated according to expectations over past latent sequences. To perform filtering variational inference, we must find the set of T terms in $q(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})$ that minimize the filtering free energy summation.

We now describe the variational filtering EM algorithm, given in Algorithm 1 and depicted in Figure 1, which optimizes Eq. 7. This algorithm sequentially optimizes each of the approximate posterior terms to perform filtering inference. Consider the approximate posterior at step t, $q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})$. This term appears in $\mathcal{F}$, either directly or in expectations, in terms t through T of the summation:

$$\mathcal{F} = \underbrace{\tilde{\mathcal{F}}_1 + \tilde{\mathcal{F}}_2 + \cdots + \tilde{\mathcal{F}}_{t-1} + \overbrace{\tilde{\mathcal{F}}_t}^{\text{terms in which } q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t}) \text{ appears}}}_{\text{steps on which } q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t}) \text{ depends}} + \tilde{\mathcal{F}}_{t+1} + \cdots + \tilde{\mathcal{F}}_{T-1} + \tilde{\mathcal{F}}_T. \quad (10)$$

However, the filtering setting dictates that the optimization of the approximate posterior at each step can only condition on past and present variables, i.e. steps 1 through t. Therefore, of the T terms in $\mathcal{F}$, the *only* term through which we can optimize $q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})$ is the t^{th} term:

$$q^*(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t}) = \underset{q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})}{\arg\min} \tilde{\mathcal{F}}_t. \quad (11)$$

Optimizing $\tilde{\mathcal{F}}_t$ requires evaluating expectations over previous approximate posteriors. Again, because approximate posterior estimates cannot be influenced by future variables, these past expectations remain *fixed* through the future. Thus, variational filtering (the variational E-step) can be performed by sequentially minimizing each $\mathcal{F}_t$ w.r.t. $q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})$, holding the expectations over past variables fixed. Conveniently, once the past expectations have been evaluated, inference optimization is entirely defined by the free energy at that step.

For simple models, such as linear Gaussian models, these expectations may be computed exactly. However, in general, the expectations must be estimated through Monte Carlo samples from q, with inference optimization carried out using stochastic gradients [36]. As in the static setting, we can initialize $q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})$ at (or near) the prior, $p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})$. This yields a simple interpretation: starting with q at the prior, we generate a *prediction* of the data through the likelihood, $p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}, \mathbf{z}_{\leq t})$, to evaluate the current step free energy. Using the approximate posterior gradient, we then perform an inference *update* to the estimate of q. This resembles classical Bayesian filtering, where the posterior is updated from the prior prediction according to the likelihood of observations. Unlike the classical setting, reconstruction and update steps are repeated until inference convergence.

After inferring an optimal approximate posterior, learning (the variational M-step) can be performed by minimizing the total filtering free energy w.r.t. the model parameters, θ. As Eq. 7 is a summation and differentiation is a linear operation, $\nabla_\theta \mathcal{F}$ is the sum of contributions from each of these terms:

$$\nabla_\theta \mathcal{F} = \sum_{t=1}^{T} \nabla_\theta \left[\mathbb{E}_{\prod_{\tau=1}^{t-1} q(\mathbf{z}_\tau|\mathbf{x}_{\leq \tau}, \mathbf{z}_{<\tau})}[\mathcal{F}_t] \right]. \quad (12)$$

Parameter gradients can be estimated online by accumulating the result from each term in the filtering free energy. The parameters are then updated at the end of the sequence. For large data sets, stochastic estimates of parameter gradients can be obtained from a mini-batch of data examples [22].

3.2 Amortized variational filtering

Performing approximate inference optimization (Algorithm 1, Line 6) with traditional techniques can be computationally costly, requiring many iterations of gradient updates and hand-tuning of optimizer hyper-parameters. In online settings, with large models and data sets, this may be impractical. An alternative approach is to employ an amortized inference model, which can learn to minimize $\mathcal{F}_t$ w.r.t. $q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})$ more efficiently at each step. Note that $\mathcal{F}_t$ (Eq. 8) contains $p_\theta(\mathbf{x}_t, \mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t}) = p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}, \mathbf{z}_{\leq t})p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})$. The prior, $p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})$, varies across steps, constituting the latent dynamics. Standard inference models, which only encode $\mathbf{x}_t$, do not have access to the prior and therefore cannot properly optimize $q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})$. Many inference models in the sequential setting attempt to account for this information by including hidden states, e.g. [7, 13, 10]. However, given the complexities of many generative models, it can be difficult to determine how to properly route the necessary prior information into the inference model. As a result, each dynamical latent latent variable model has been proposed with an accompanying custom inference model set-up.

We propose a simple and general alternative method for amortizing filtering inference that is agnostic to the particular form of the generative model. Iterative inference models [33] naturally account for the changing prior through the approximate posterior gradients. These models are thus a natural candidate for performing inference at each step. Similar to Eq. 5, when $q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})$ is a parametric distribution with parameters $\boldsymbol{\lambda}_t^q$, the inference update takes the form:

$$\boldsymbol{\lambda}_t^q \leftarrow f_\phi(\boldsymbol{\lambda}_t^q, \nabla_{\boldsymbol{\lambda}_t^q}\tilde{\mathcal{F}}_t). \tag{13}$$

We refer to this set-up as *amortized variational filtering* (AVF). As in Eq. 5, we note that Eq. 13 offers just one particular encoding form for an iterative inference model. For instance, $\mathbf{x}_t$ could be additionally encoded at each step. Marino *et al.* also note that in latent Gaussian models, precision-weighted errors provide an alternative inference optimization signal [33]. There are two main benefits to using iterative inference models in the filtering setting:

- The approximate posterior is updated from the prior, so model capacity is utilized for inference *corrections* rather than re-estimating the approximate posterior at each step.
- These inference models contain all of the terms necessary to perform inference optimization, providing a simple model form that does not require any additional hidden states or inputs.

In practice, these advantages permit the use of relatively simple iterative inference models that can perform filtering inference efficiently and accurately. We demonstrate this in the following section.

4 Experiments

We empirically evaluate amortized variational filtering using multiple deep dynamical latent Gaussian model architectures on a variety of sequence data sets. Specifically, we use AVF to train VRNN [7], SRNN [13], and SVG [10] on speech [15], music [5], and video [40] data. In each setting, we compare AVF against the originally proposed filtering method for the model. Diagrams of the filtering methods are shown in Figure 2. Implementations of the models are based on code provided by the respective authors of VRNN[1], SRNN[2], and SVG[3]. Accompanying code can be found online at `github.com/joelouismarino/amortized-variational-filtering`.

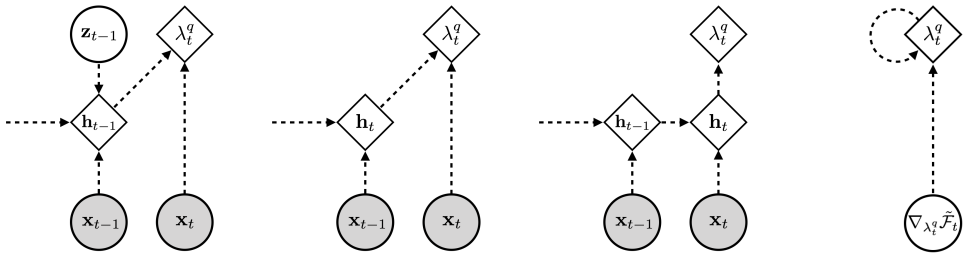
4.1 Experiment set-up

Iterative inference models are implemented as specified in Eq. 13, encoding the approximate posterior parameters and their gradients at each inference iteration at each step. Following [33], we normalize the inputs to the inference model using layer normalization [2]. The generative models that we

[1] `https://github.com/jych/nips2015_vrnn`

[2] `https://github.com/marcofraccaro/srnn`

[3] `https://github.com/edenton/svg`



(a) VRNN (b) SRNN (c) SVG (d) AVF

Figure 2: Filtering inference models for VRNN, SRNN, SVG, and AVF. Each diagram shows the computational graph for inferring the approximate posterior parameters, $\boldsymbol{\lambda}^q$, at step t. Previously proposed methods rely on hand-crafted architectures of observations, hidden states, and latent variables. AVF is a simple, general filtering procedure that only requires the local inference gradient.

evaluate contain non-spatial latent variables, thus, we use fully-connected layers to parameterize the inference models. Importantly, minimal effort went into engineering the inference model architectures: across all models and data sets, we utilize the *same* inference model architecture for AVF. Further details are found in Appendix B.

4.1.1 Speech modeling

Models For speech modeling, we use VRNN and SRNN, attempting to keep the model architectures consistent with the original implementations. The most notable difference in our implementation occurs in SRNN, where we use an LSTM rather than a GRU as the recurrent module. As in [13], we anneal the KL divergence initially during training. In both models, we use a Gaussian output density. Unlike [7, 13, 18], which evaluate log *densities*, we evaluate and report log *probabilities* by integrating the output density over the data discretization window, as in modeling image pixels. This permits comparison across different output distributions.

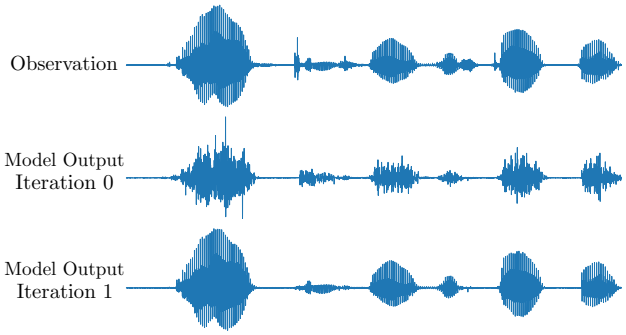


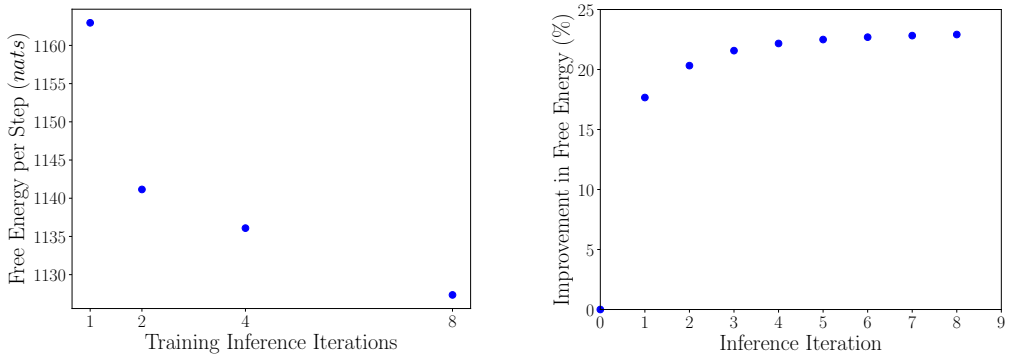
Figure 3: Test data (top), output predictions (middle), and reconstructions (bottom) for TIMIT using SRNN with AVF. Sequences run from left to right. The predictions made by the model already contain the general structure of the data. AVF explicitly updates the approximate posterior from the prior prediction, focusing on inference *corrections* rather than re-estimation.

Data We train and evaluate on TIMIT [15], which consists of audio recordings of 6,300 sentences spoken by 630 individuals. As performed in [7], we sample the audio waveforms at 16 kHz, split the training and validation sets into half second clips, and group each sequence into bins of 200 consecutive samples. Thus, each training and validation sequence consists of 40 model steps. Evaluation is performed on the full duration of each test sequence, averaging roughly 3 seconds.

4.1.2 Music modeling

Model We model polyphonic music using SRNN. The generative model architecture is the same as in the speech modeling experiments, with changes in the number of layers and units to match [13]. To model the binary music notes, we use a Bernoulli output distribution. Again, we anneal the KL divergence initially during training.

Data We use four data sets of polyphonic (MIDI) music [5]: Piano-midi.de, MuseData, JSB Chorales, and Nottingham. Each data set contains between 100 and 1,000 songs, with each song



(a) (b)

Figure 4: **Improvement with inference iterations.** Results are shown on the TIMIT validation set using VRNN with AVF. (a) Average free energy per step with varying numbers of inference iterations during training. Additional iterations tend to result in improved performance. (b) Average relative improvement in free energy from the initial (prior) estimate at each inference iteration for a single model. Empirically, each successive iteration provides further, smaller improvements.

between 100 to 4,000 steps. For training and validation, we break the sequences into clips of length 25, and we test on the entire test sequences.

4.1.3 Video modeling

Model Our implementation of SVG differs from the original model in that we evaluate conditional log-likelihood under a Gaussian output density rather than mean squared output error. All other architecture details are identical to the original model. However, [10] down-weight the KL-divergence by a factor of 1e-6 at all steps. We instead remove this factor to use the free energy during training and evaluation. As to be expected, this results in the model using the latent variables to a lesser extent. We train and evaluate SVG using filtering inference at all steps, rather than predicting multiple steps into the future, as in [10].

Data We train and evaluate SVG on KTH Actions [40], which contains 760 train / 768 val / 863 test videos of people performing various actions, each of which is between roughly 50 – 150 frames. Frames are re-sized to 64×64. For training and validation, we split the data into clips of 20 frames.

4.2 Results

4.2.1 Additional Inference Iterations

The variational filtering EM algorithm involves inference optimization at each step (Algorithm 1, Line 6). AVF optimizes each approximate posterior through a model that learns to perform iterative updates (Eq. 13). Additional inference iterations may lead to further improvement in performance [33]. We explore this aspect on TIMIT using VRNN. In Figure 4a, we plot the average free energy per step on validation sequences for models trained with varying numbers of inference iterations. Figure 4b shows average relative improvement over the prior estimate for a single model trained with 8 inference iterations. We observe that training with additional inference iterations empirically leads to improved performance (Figure 4a), with each iteration providing diminishing improvement during inference (Figure 4b). This aspect is distinct from many baseline filtering methods, which directly output the approximate posterior at each step.

We can also directly visualize inference improvement through the model output. Figure 3 illustrates example reconstructions over inference iterations, using SRNN on TIMIT. At the initial inference iteration, the approximate posterior is initialized from the prior, resulting in an output prediction. The iterative inference model then uses the approximate posterior gradients to update the estimate, improving the output reconstruction.

Table 1: Average free energy per step (in *nats*) on the TIMIT speech data set for SRNN and VRNN with the respective originally proposed filtering procedures (baselines) and with AVF.

	TIMIT
VRNN	
baseline	**1,082**
AVF	1,105
SRNN	
baseline	1,026
AVF	**1,024**

Table 2: Average free energy per step (in *nats*) on the KTH Actions video data set for SVG with the originally proposed filtering procedure (baseline) and with AVF.

	KTH Actions
SVG	
baseline	15,097
AVF	**11, 714**

Table 3: Average free energy per step (in *nats*) on polyphonic music data sets for SRNN with and without AVF. Results from Fraccaro *et al.* [13] are provided for comparison, however, our model implementation differs in several aspects (see Appendix B).

	Piano-midi.de	MuseData	JSB Chorales	Nottingham
SRNN				
baseline [13]	8.20	6.28	4.74	2.94
baseline	8.19	6.27	**6.92**	3.19
AVF	**8.12**	**5.99**	6.97	**3.13**

4.2.2 Quantitative Comparison

Tables 1, 2, and 3 present quantitative comparisons of average filtering free energy per step between AVF (with 1 inference iteration per step) and baseline filtering methods for TIMIT, KTH Actions, and the polyphonic music data sets respectively. On TIMIT, training with AVF performs comparably to the baseline methods for both VRNN and SRNN. We note that VRNN with AVF using 2 inference iterations resulted in a final test performance of 1,071 *nats* per step, outperforming the baseline method. Similar results are also observed on each of the polyphonic music data sets. Again, increasing the number of inference iterations to 5 for AVF on JSB Chorales resulted in a final test performance of 6.77 *nats* per step. AVF significantly improves the performance of SVG on KTH Actions. We attribute this, likely, to the absence of the KL down-weighting factor in our training objective as compared with [10]. The baseline filtering procedure seems to struggle to a greater degree than AVF. From comparing the results above, we see that AVF is a general filtering procedure that performs well across multiple models and data sets, despite using a relatively simple inference model structure.

5 Conclusion

We introduced the variational filtering EM algorithm for filtering in dynamical latent variable models. Variational filtering inference can be expressed as a sequence of optimization objectives, linked across steps through previous latent samples. Using iterative inference models to perform inference optimization, we arrived at an efficient implementation of the algorithm: amortized variational filtering. This general filtering algorithm scales to large models and data sets. Numerous methods have been proposed for filtering in deep dynamical latent variable models, with each method hand-designed for each model. The variational filtering EM algorithm provides a single framework for analyzing and constructing these methods. Amortized variational filtering is a simple, theoretically-motivated, and general filtering method that we have shown performs on-par with or better than multiple existing state-of-the-art methods.

Acknowledgments

We would like to thank Matteo Ruggero Ronchi for helpful discussions. This work was supported by the following grants: JPL PDF 1584398, NSF 1564330, and NSF 1637598.

References

[1] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, 2016.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. In *NIPS Deep Learning Symposium*, 2016.

[3] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H Campbell, and Sergey Levine. Stochastic variational video prediction. In *International Conference on Learning Representations*, 2018.

[4] Justin Bayer and Christian Osendorfer. Learning stochastic recurrent networks. In *NIPS 2014 Workshop on Advances in Variational Inference*, 2014.

[5] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *International Conference on Machine Learning*, 2012.

[6] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[7] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in Neural Information Processing Systems*, 2015.

[8] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[9] Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

[10] Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. In *International Conference on Machine Learning*, 2018.

[11] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in Neural Information Processing Systems*, 2016.

[12] Marco Fraccaro, Simon Kamronn, Ulrich Paquet, and Ole Winther. A disentangled recognition and nonlinear dynamics model for unsupervised learning. In *Advances in Neural Information Processing Systems*, 2017.

[13] Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. In *Advances in Neural Information Processing Systems*, 2016.

[14] Karl Friston. A theory of cortical responses. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 360(1456):815–836, 2005.

[15] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, and N. L. Dahlgren. Darpa timit acoustic phonetic continuous speech corpus, 1993.

[16] Mevlana Gemici, Chia-Chun Hung, Adam Santoro, Greg Wayne, Shakir Mohamed, Danilo J Rezende, David Amos, and Timothy Lillicrap. Generative temporal models with memory. *arXiv preprint arXiv:1702.04649*, 2017.

[17] Samuel Gershman and Noah Goodman. Amortized inference in probabilistic reasoning. In *Cognitive Science Society*, volume 36, 2014.

[18] Anirudh Goyal, Alessandro Sordoni, Marc-Alexandre Côté, Nan Ke, and Yoshua Bengio. Z-forcing: Training stochastic recurrent networks. In *Advances in Neural Information Processing Systems*, 2017.

[19] Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra. Deep autoregressive networks. In *International Conference on Machine Learning*, 2014.

[20] Jiawei He, Andreas Lehrmann, Joseph Marino, Greg Mori, and Leonid Sigal. Probabilistic video generation using holistic attribute control. In *European Conference on Computer Vision*, 2018.

[21] Mikael Henaff, Junbo Zhao, and Yann LeCun. Prediction under uncertainty with error-encoding networks. *arXiv preprint arXiv:1711.04994*, 2017.

[22] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

[23] Wei-Ning Hsu, Yu Zhang, and James Glass. Unsupervised learning of disentangled and interpretable representations from sequential data. In *Advances in Neural Information Processing Systems*, 2017.

[24] Matthew Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and Sandeep R Datta. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in Neural Information Processing Systems*, 2016.

[25] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *NATO ASI SERIES D BEHAVIOURAL AND SOCIAL SCIENCES*, 89:105–162, 1998.

[26] Rudolph Emil Kalman et al. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[27] Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. In *International Conference on Learning Representations*, 2017.

[28] Diederik P Kingma and Max Welling. Stochastic gradient vb and the variational auto-encoder. In *International Conference on Learning Representations*, 2014.

[29] Tuan Anh Le, Maximilian Igl, Tom Jin, Tom Rainforth, and Frank Wood. Auto-encoding sequential monte carlo. In *International Conference on Learning Representations*, 2018.

[30] Yingzhen Li and Stephan Mandt. A deep generative model for disentangled representations of sequential data. In *International Conference on Machine Learning*, 2018.

[31] William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. In *International Conference on Learning Representations*, 2017.

[32] Chris J Maddison, John Lawson, George Tucker, Nicolas Heess, Mohammad Norouzi, Andriy Mnih, Arnaud Doucet, and Yee Teh. Filtering variational objectives. In *Advances in Neural Information Processing Systems*, 2017.

[33] Joseph Marino, Yisong Yue, and Stephan Mandt. Iterative amortized inference. In *International Conference on Machine Learning*, 2018.

[34] Christian Naesseth, Scott Linderman, Rajesh Ranganath, and David Blei. Variational sequential monte carlo. In *International Conference on Artificial Intelligence and Statistics*, 2018.

[35] Radford M Neal and Geoffrey E Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.

[36] Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *Artificial Intelligence and Statistics*, 2014.

[37] Rajesh PN Rao and Dana H Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1), 1999.

[38] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, 2014.

[39] Simo Särkkä. *Bayesian filtering and smoothing*, volume 3. Cambridge University Press, 2013.

[40] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: a local svm approach. In *International Conference on Pattern Recognition*, 2004.

[41] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *Advances in Neural Information Processing Systems*, 2016.

[42] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *International Conference on Machine Learning*, 2015.

[43] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[44] Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In *European Conference on Computer Vision*, 2016.

[45] Tianfan Xue, Jiajun Wu, Katherine Bouman, and Bill Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *Advances in Neural Information Processing Systems*, 2016.

A Filtering variational free energy

This derivation largely follows that of [16] and is valid for factorized filtering approximate posteriors. From Eq. 3, we have the definition of variational free-energy:

$$\mathcal{F} \equiv -\mathbb{E}_{q(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})}\left[\log \frac{p_\theta(\mathbf{x}_{\leq T}, \mathbf{z}_{\leq T})}{q(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})}\right]. \tag{14}$$

Plugging in the forms of the joint distribution (Eq. 1) and approximate posterior (Eq. 6), we can write the term within the expectation as a sum:

$$\mathcal{F} = -\mathbb{E}_{q(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})}\left[\log\left(\prod_{t=1}^{T} \frac{p(\mathbf{x}_t, \mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})}{q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})}\right)\right] \tag{15}$$

$$\mathcal{F} = -\mathbb{E}_{q(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})}\left[\sum_{t=1}^{T} \log \frac{p(\mathbf{x}_t, \mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})}{q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})}\right] \tag{16}$$

$$\mathcal{F} = -\mathbb{E}_{q(\mathbf{z}_{\leq T}|\mathbf{x}_{\leq T})}\left[\sum_{t=1}^{T} C_t\right] \tag{17}$$

where the term C_t is defined to simplify notation. We then expand the expectation:

$$\mathcal{F} = -\mathbb{E}_{q(\mathbf{z}_1|\mathbf{x}_1)} \cdots \mathbb{E}_{q(\mathbf{z}_T|\mathbf{x}_{\leq T}, \mathbf{z}_{<T})}\left[\sum_{t=1}^{T} C_t\right] \tag{18}$$

There are T terms within the sum, but each C_t only depends on the expectations up to time t because we only condition on past and present variables. This allows us to write:

$$\begin{aligned}\mathcal{F} = &-\mathbb{E}_{q(\mathbf{z}_1|\mathbf{x}_1)}\left[C_1\right] \\ &-\mathbb{E}_{q(\mathbf{z}_1|\mathbf{x}_1)}\mathbb{E}_{q(\mathbf{z}_2|\mathbf{x}_{\leq 2}, \mathbf{z}_1)}\left[C_2\right] \\ &-\cdots \\ &-\mathbb{E}_{q(\mathbf{z}_1|\mathbf{x}_1)}\mathbb{E}_{q(\mathbf{z}_2|\mathbf{x}_{\leq 2}, \mathbf{z}_1)} \cdots \mathbb{E}_{q(\mathbf{z}_T|, \mathbf{x}_{\leq T}, \mathbf{z}_{<T})}\left[C_T\right]\end{aligned} \tag{19}$$

$$\mathcal{F} = -\sum_{t=1}^{T} \mathbb{E}_{q(\mathbf{z}_{\leq t}|\mathbf{x}_{\leq t})}\left[C_t\right] \tag{20}$$

$$\mathcal{F} = -\sum_{t=1}^{T} \mathbb{E}_{\prod_{\tau=1}^{t} q(\mathbf{z}_\tau|\mathbf{x}_{\leq \tau}, \mathbf{z}_{<\tau})}\left[C_t\right] \tag{21}$$

$$\mathcal{F} = -\sum_{t=1}^{T} \mathbb{E}_{\prod_{\tau=1}^{t-1} q(\mathbf{z}_\tau|\mathbf{x}_{\leq \tau}, \mathbf{z}_{<\tau})}\left[\mathbb{E}_{q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})}\left[C_t\right]\right] \tag{22}$$

As in Section 3, we define $\mathcal{F}_t$ as

$$\mathcal{F}_t \equiv -\mathbb{E}_{q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})}\left[C_t\right] \tag{23}$$

$$\mathcal{F}_t = -\mathbb{E}_{q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})}\left[\log \frac{p_\theta(\mathbf{x}_t, \mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})}{q(\mathbf{z}_t|\mathbf{x}_{\leq t}, \mathbf{z}_{<t})}\right]. \tag{24}$$

This allows us to write Eq. 22 as

$$\mathcal{F} = \sum_{t=1}^{T} \mathbb{E}_{\prod_{\tau=1}^{t-1} q(\mathbf{z}_\tau|\mathbf{x}_{\leq \tau}, \mathbf{z}_{<\tau})}\left[\mathcal{F}_t\right], \tag{25}$$

which agrees with Eq. 7.

B Implementation details

For all iterative inference models, we followed [33], using two layer fully-connected networks with 1,024 units per layer, "highway" gating connections [43], and ELU non-linearities [8]. Unless otherwise noted, these models received a concatenation of the current estimate of the approximate posterior distribution parameters and the corresponding gradients (4 terms in total), normalizing each term separately using layer normalization [2]. We used the output gating update employed in [33]. We also found that applying layer normalization to the approximate posterior mean estimates resulted in improved training stability. AVF is comparable in runtime to the baseline filtering methods. For example, with our implementation of SRNN on TIMIT, AVF requires 13.1 ms per time step, whereas the baseline method requires 15.6 ms per time step. AVF requires an additional decoding step, but inference is local to the current step, meaning that backpropagation is more efficient. Accompanying code can be found online at `github.com/joelouismarino/amortized-variational-filtering`.

B.1 Speech modeling

The VRNN architecture is implemented as in [7], matching the number of layers and units in each component of the model, as well as the non-linearities. We trained on TIMIT with sequences of length 40, using a batch size of 64. For the baseline method, we used a learning rate of 0.001, as specified in [7]. For AVF, we used a learning rate of 0.0001. We annealed the learning rates by a factor of 0.999 after each epoch. Both models were trained for 700 epochs.

We implement SRNN following [13], with the exception of an LSTM in place of the GRU. All other architecture details, are kept consistent, including the use of clipped (± 3) leaky ReLU non-linearities. The sequence length and batch size are the same as above. We use a learning rate of 0.001 for the baseline method, following [13]. We use a learning rate of 0.0001. We use the same learning rate annealing strategy as above. Following [13], we anneal the KL-divergence of the baseline linearly over the first 20 epochs. We increase this duration to 50 epochs for AVF to account for the lower learning rate. The iterative inference model additionally encodes the data observation at each step, which we found necessary to overcome the local minima from the KL-divergence. Models were trained for 1,000 epochs.

B.2 Music modeling

Our SRNN implementation is the same as in the speech modeling setting, with the appropriate changes in the number of units and layers to match [13]. We used a sequence length of 25 and a batch size of 16. All models were trained with a learning rate of 0.0001, with a decay factor of 0.999 per epoch. We annealed the KL-divergence linearly over the first 50 epochs. Models were trained for 800 epochs.

B.3 Video modeling

The SVG model architecture is implemented identically to [10], with an additional log-variance term in the observation model's output to provide a Gaussian output density. Unlike [10], we do not down-weight the KL-divergence term in the free energy in order to achieve a proper bound. We note that this is likely the reason that our SVG baseline performs poorly, as the original model was not intended to be trained with the true variational bound objective. Indeed, we observed that the model struggles to make use of the latent variables, even when annealing the KL-divergence weight in the objective. We trained on sequences of length 20 using a batch size of 20. For both methods, we used a learning rate of 0.0001, with decay of 0.999 after each epoch. Models were trained for 100 epochs.