Iterative Amortized Inference

Joseph Marino [1] **Yisong Yue** [1] **Stephan Mandt** [2]

Abstract

Inference models are a key component in scaling variational inference to deep latent variable models, most notably as encoder networks in variational auto-encoders (VAEs). By replacing conventional optimization-based inference with a learned model, inference is amortized over data examples and therefore more computationally efficient. However, standard inference models are restricted to direct mappings from data to approximate posterior estimates. The failure of these models to reach fully optimized approximate posterior estimates results in an *amortization gap*. We aim toward closing this gap by proposing *iterative inference models*, which learn to perform inference optimization through repeatedly encoding gradients. Our approach generalizes standard inference models in VAEs and provides insight into several empirical findings, including top-down inference techniques. We demonstrate the inference optimization capabilities of iterative inference models and show that they outperform standard inference models on several benchmark data sets of images and text.

1. Introduction

Variational inference (Jordan et al., 1998) has been essential in learning deep directed latent variable models on high-dimensional data, enabling extraction of complex, nonlinear relationships, such as object identities (Higgins et al., 2016) and dynamics (Xue et al., 2016; Karl et al., 2017) directly from observations. Variational inference reformulates inference as optimization (Neal & Hinton, 1998; Hoffman et al., 2013). However, the current trend has moved toward employing *inference models* (Dayan et al., 1995; Gregor et al., 2014; Kingma & Welling, 2014; Rezende et al., 2014), mappings from data to approximate posterior estimates that are amortized across examples. Intuitively, the inference model encodes observations into latent representations, and the generative model decodes these representations into reconstructions. Yet, this approach has notable limitations. For instance, in models with empirical priors, such as hierarchical latent variable models, "bottom-up" data-encoding inference models cannot account for "top-down" priors (Section 4.1). This has prompted the use of top-down inference techniques (Sønderby et al., 2016), which currently lack a rigorous theoretical justification. More generally, the inability of inference models to reach fully optimized approximate posterior estimates results in decreased modeling performance, referred to as an *amortization gap* (Krishnan et al., 2018; Cremer et al., 2017).

To combat this problem, our work offers a departure from previous approaches by re-examining inference from an optimization perspective. We utilize approximate posterior gradients to perform inference optimization. Yet, we improve computational efficiency over conventional optimizers by encoding these gradients with an inference model that learns how to iteratively update approximate posterior estimates. The resulting *iterative inference models* resemble learning to learn (Andrychowicz et al., 2016) applied to variational inference optimization. However, we refine and extend this method along several novel directions. Namely, (1) we show that learned optimization models can be applied to inference optimization of latent variables; (2) we show that non-recurrent optimization models work well in practice, breaking assumptions about the necessity of non-local curvature for outperforming conventional optimizers (Andrychowicz et al., 2016; Putzky & Welling, 2017); and (3) we provide a new form of optimization model that encodes errors rather than gradients to approximate higher order derivatives, empirically resulting in faster convergence.

Our main contributions are summarized as follows:

1. we introduce a family of iterative inference models, which generalize standard inference models,
2. we provide the first theoretical justification for top-down inference techniques,
3. we empirically evaluate iterative inference models, demonstrating that they outperform standard inference models on several data sets of images and text.

[1]California Institute of Technology (Caltech), Pasadena, CA, USA [2]Los Angeles, CA, USA. Correspondence to: Joseph Marino <jmarino@caltech.edu>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

2. Background

2.1. Latent Variable Models & Variational Inference

Latent variable models are generative probabilistic models that use local (per data example) latent variables, $\mathbf{z}$, to model observations, $\mathbf{x}$, using global (across data examples) parameters, θ. A model is defined by the joint distribution $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})$, composed of the conditional likelihood and the prior. Learning the model parameters and inferring the posterior, $p(\mathbf{z}|\mathbf{x})$, are intractable for all but the simplest models, as they require evaluating the marginal likelihood, $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z}$, which involves integrating the model over $\mathbf{z}$. For this reason, we often turn to approximate inference methods.

Variational inference reformulates this intractable integration as an optimization problem by introducing an approximate posterior[1], $q(\mathbf{z}|\mathbf{x})$, typically chosen from some tractable family of distributions, and minimizing the KL-divergence from the posterior, $D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x}))$. This quantity cannot be minimized directly, as it contains the posterior. Instead, KL-divergence can be decomposed into

$$D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) = \log p_\theta(\mathbf{x}) - \mathcal{L}, \tag{1}$$

where $\mathcal{L}$ is the evidence lower bound (ELBO), which is defined as:

$$\mathcal{L} \equiv \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}|\mathbf{x})\right] \tag{2}$$

$$= \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - D_{KL}(q(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})). \tag{3}$$

The first term in eq. 3 expresses how well the output reconstructs the data example. The second term quantifies the dissimilarity between the approximate posterior and the prior. Because $\log p_\theta(\mathbf{x})$ is not a function of $q(\mathbf{z}|\mathbf{x})$, we can minimize $D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x}))$ in eq. 1 by maximizing $\mathcal{L}$ w.r.t. $q(\mathbf{z}|\mathbf{x})$, thereby performing approximate *inference*. Likewise, because $D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x}))$ is non-negative, $\mathcal{L}$ is a lower bound on $\log p_\theta(\mathbf{x})$. Therefore, once we have inferred an optimal $q(\mathbf{z}|\mathbf{x})$, *learning* corresponds to maximizing $\mathcal{L}$ w.r.t. θ.

2.2. Variational Expectation Maximization (EM) via Gradient Ascent

The optimization procedures for variational inference and learning are respectively the expectation and maximization steps of the variational EM algorithm (Dempster et al., 1977; Neal & Hinton, 1998), which alternate until convergence. This is typically performed in the batched setting of stochastic variational inference (Hoffman et al., 2013). When $q(\mathbf{z}|\mathbf{x})$ takes a parametric form, the expectation step for data example $\mathbf{x}^{(i)}$ involves finding a set of distribution parameters, $\boldsymbol{\lambda}^{(i)}$, that are optimal w.r.t. $\mathcal{L}$. With a factorized Gaussian density over continuous latent variables, i.e. $\boldsymbol{\lambda}^{(i)} = \{\boldsymbol{\mu}_q^{(i)}, \boldsymbol{\sigma}_q^{2(i)}\}$ and $q(\mathbf{z}^{(i)}|\mathbf{x}^{(i)}) = \mathcal{N}(\mathbf{z}^{(i)}; \boldsymbol{\mu}_q^{(i)}, \text{diag}\,\boldsymbol{\sigma}_q^{2(i)})$, conventional optimization techniques repeatedly estimate the stochastic gradients $\nabla_{\boldsymbol{\lambda}}\mathcal{L}$ to optimize $\mathcal{L}$ w.r.t. $\boldsymbol{\lambda}^{(i)}$, e.g.:

$$\boldsymbol{\lambda}^{(i)} \leftarrow \boldsymbol{\lambda}^{(i)} + \alpha\nabla_{\boldsymbol{\lambda}}\mathcal{L}(\mathbf{x}^{(i)}, \boldsymbol{\lambda}^{(i)}; \theta), \tag{4}$$

where α is the step size. This procedure, which is repeated for each example, is computationally expensive and requires setting step-size hyper-parameters.

2.3. Amortized Inference Models

Due to the aforementioned issues, gradient updates of approximate posterior parameters are rarely performed in practice. Rather, inference models are often used to map observations to approximate posterior estimates. Optimization of each data example's approximate posterior parameters, $\boldsymbol{\lambda}^{(i)}$, is replaced with the optimization of a shared, i.e. amortized (Gershman & Goodman, 2014), set of parameters, ϕ, contained within an inference model, f, of the form:

$$\boldsymbol{\lambda}^{(i)} \leftarrow f(\mathbf{x}^{(i)}; \phi). \tag{5}$$

While inference models have a long history, e.g. (Dayan et al., 1995), the most notable recent example is the variational auto-encoder (VAE) (Kingma & Welling, 2014; Rezende et al., 2014), which employs the reparameterization trick to propagate stochastic gradients from the generative model to the inference model, both of which are parameterized by neural networks. We refer to inference models of this form as *standard inference models*. As discussed in Section 3, the aim of this paper is to move beyond the direct encoder paradigm of standard inference models to develop improved techniques for performing inference.

3. Iterative Amortized Inference

In Section 3.3, we introduce our contribution, iterative inference models. However, we first motivate our approach in Section 3.1 by discussing the limitations of standard inference models. We then draw inspiration from other techniques for learning to optimize (Section 3.2).

3.1. Standard Inference Models & Amortization Gaps

As described in Section 2.1, variational inference reformulates inference as the maximization of $\mathcal{L}$ w.r.t. $q(\mathbf{z}|\mathbf{x})$, constituting the expectation step of the variational EM algorithm. In general, this is a difficult non-convex optimization problem, typically requiring a lengthy iterative estimation procedure. Yet, standard inference models attempt to perform this optimization through a direct, discriminative mapping from data observations to approximate posterior

[1] We use $q(\mathbf{z}|\mathbf{x})$ to denote that the approximate posterior is conditioned on a data example (i.e. local), however this does not necessarily imply a direct functional mapping.

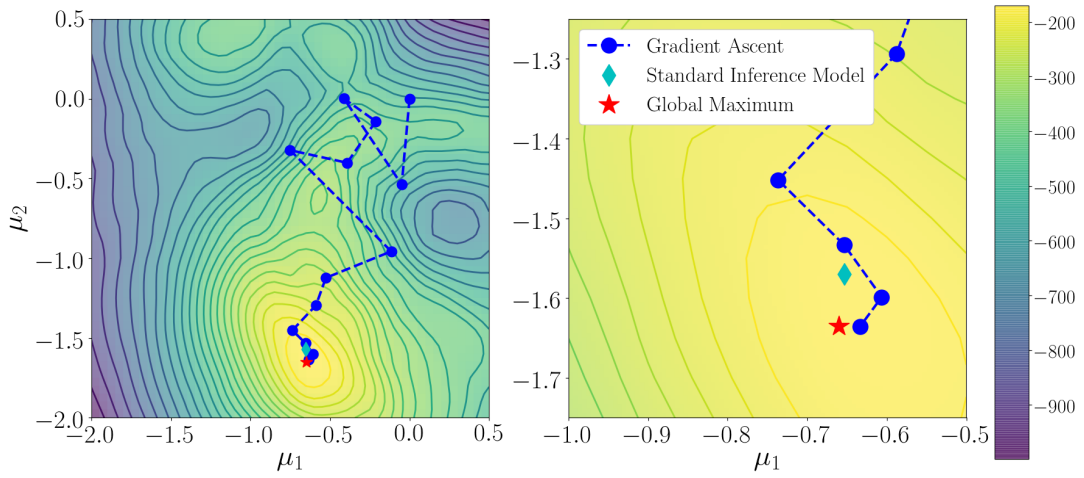


Figure 1. **Visualizing the amortization gap.** Optimization surface of $\mathcal{L}$ (in *nats*) for a 2-D latent Gaussian model and an MNIST data example. Shown on the plots are the optimal estimate (MAP), the output of a standard inference model, and an optimization trajectory of gradient ascent. The plot on the right shows an enlarged view near the optimum. Conventional optimization outperforms the standard inference model, exhibiting an amortization gap. With additional latent dimensions or more complex data, this gap could become larger.

parameters. Of course, generative models can adapt to accommodate sub-optimal approximate posteriors. Nevertheless, the possible limitations of a direct inference mapping applied to this difficult optimization procedure may result in a decrease in overall modeling performance.

We demonstrate this concept in Figure 1 by visualizing the optimization surface of $\mathcal{L}$ defined by a 2-D latent Gaussian model and a particular binarized MNIST (LeCun et al., 1998) data example. To visualize the approximate posterior, we use a point estimate, $q(\mathbf{z}|\mathbf{x}) = \delta(\boldsymbol{\mu}_q)$, where $\boldsymbol{\mu}_q = (\mu_1, \mu_2)$ is the estimate and δ is the Dirac delta function. See Appendix C.1 for details. Shown on the plot are the optimal (maximum a posteriori or MAP) estimate, the estimate from a standard inference model, and an optimization trajectory of gradient ascent. The inference model is unable to achieve the optimum, but manages to output a reasonable estimate in one pass. Gradient ascent requires many iterations and is sensitive to step-size, but through the iterative estimation procedure, ultimately arrives at a better final estimate. The inability of inference models to reach optimal approximate posterior estimates, as typically compared with gradient-based methods, creates an amortization gap (Krishnan et al., 2018; Cremer et al., 2017), which impairs modeling performance. Additional latent dimensions and more complex data could further exacerbate this problem.

3.2. Learning to Iteratively Optimize

While offering significant benefits in computational efficiency, standard inference models can suffer from sizable amortization gaps (Krishnan et al., 2018). Parameterizing inference models as direct, static mappings from $\mathbf{x}$ to $q(\mathbf{z}|\mathbf{x})$ may be overly restrictive, widening this gap. To improve upon this direct encoding paradigm, we pose the following question: *can we retain the computational efficiency of inference models while incorporating more powerful iterative estimation capabilities?* Our proposed solution is a new class of inference models, capable of learning how to update approximate posterior estimates by encoding gradients or errors. Due to the iterative nature of these models, we refer to them as *iterative inference models*. Through an analysis with latent Gaussian models, we show that iterative inference models generalize standard inference models (Section 4.3) and offer theoretical justification for top-down inference in hierarchical models (Section 4.1).

Our approach relates to learning to learn (Andrychowicz et al., 2016), where an *optimizer* model learns to optimize the parameters of an *optimizee* model. The optimizer receives the optimizee's parameter gradients and outputs updates to these parameters to improve the optimizee's loss. The optimizer itself can be learned due to the differentiable computation graph. Such models can adaptively adjust step sizes, potentially outperforming conventional optimizers. For inference optimization, previous works have combined standard inference models with gradient updates (Hjelm et al., 2016; Krishnan et al., 2018; Kim et al., 2018), however, these works do not *learn* to iteratively optimize. (Putzky & Welling, 2017) use recurrent inference models for MAP estimation of denoised images in linear models. We propose a unified method for learning to perform variational inference optimization, generally applicable to probabilistic latent variable models. Our work extends techniques for learning to optimize along several novel directions, discussed in Section 4.

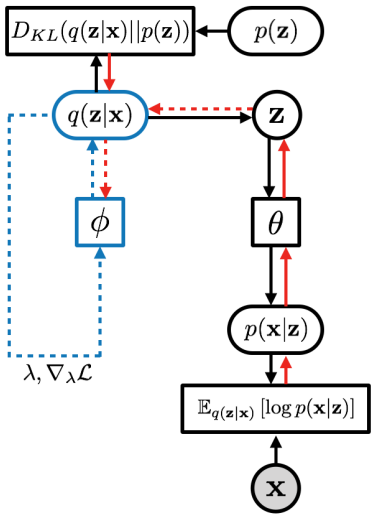


Figure 2. **Computation graph** for a single-level latent variable model with an iterative inference model. Black components evaluate the ELBO. Blue components are used during variational inference. Red corresponds to gradients. Solid arrows denote deterministic values. Dashed arrows denote stochastic values. During inference, $\boldsymbol{\lambda}$, the distribution parameters of $q(\mathbf{z}|\mathbf{x})$, are first initialized. $\mathbf{z}$ is sampled from $q(\mathbf{z}|\mathbf{x})$ to evaluate the ELBO. Stochastic gradients are then backpropagated to $\boldsymbol{\lambda}$. The iterative inference model uses these gradients to update the current estimate of $\boldsymbol{\lambda}$. The process is repeated iteratively. The inference model parameters, ϕ, are trained through accumulated estimates of $\nabla_\phi \mathcal{L}$.

3.3. Iterative Inference Models

We denote an iterative inference model as f with parameters ϕ. With $\mathcal{L}_t^{(i)} \equiv \mathcal{L}(\mathbf{x}^{(i)}, \boldsymbol{\lambda}_t^{(i)}; \theta)$ as the ELBO for data example $\mathbf{x}^{(i)}$ at inference iteration t, the model uses the approximate posterior gradients, denoted $\nabla_{\boldsymbol{\lambda}} \mathcal{L}_t^{(i)}$, to output updated estimates of $\boldsymbol{\lambda}^{(i)}$:

$$\boldsymbol{\lambda}_{t+1}^{(i)} \leftarrow f_t(\nabla_{\boldsymbol{\lambda}} \mathcal{L}_t^{(i)}, \boldsymbol{\lambda}_t^{(i)}; \phi), \tag{6}$$

where $\boldsymbol{\lambda}_t^{(i)}$ is the estimate of $\boldsymbol{\lambda}^{(i)}$ at inference iteration t. Eq. 6 is in a general form and contains, as special cases, the linear update in eq. 4, as well as the residual, non-linear update used in (Andrychowicz et al., 2016). Figure 2 displays a computation graph of the inference procedure, and Algorithm 1 in Appendix B describes the procedure in detail. As with standard inference models, the parameters of an iterative inference model can be updated using estimates of $\nabla_\phi \mathcal{L}$, obtained through the reparameterization trick (Kingma & Welling, 2014; Rezende et al., 2014) or through score function methods (Gregor et al., 2014; Ranganath et al., 2014). Model parameter updating is performed using stochastic gradient techniques with $\nabla_\theta \mathcal{L}$ and $\nabla_\phi \mathcal{L}$.

4. Iterative Inference in Latent Gaussian Models

We now describe an instantiation of iterative inference models for (single-level) latent Gaussian models, which have a Gaussian prior density over latent variables: $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_p, \operatorname{diag} \boldsymbol{\sigma}_p^2)$. Although the prior is typically a standard Normal density, we use this prior form for generality. Latent Gaussian models are often used in VAEs and are a common choice for continuous-valued latent variables. While the approximate posterior can be any probability density, it is typically also chosen as Gaussian: $q(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_q, \operatorname{diag} \boldsymbol{\sigma}_q^2)$. With this choice, $\boldsymbol{\lambda}^{(i)}$ corresponds to $\{\boldsymbol{\mu}_q^{(i)}, \boldsymbol{\sigma}_q^{2(i)}\}$ for example $\mathbf{x}^{(i)}$. Dropping the superscript (i) to simplify notation, we can express eq. 6 for this model as:

$$\boldsymbol{\mu}_{q,t+1} = f_t^{\boldsymbol{\mu}_q}(\nabla_{\boldsymbol{\mu}_q} \mathcal{L}_t, \boldsymbol{\mu}_{q,t}; \phi), \tag{7}$$

$$\boldsymbol{\sigma}_{q,t+1}^2 = f_t^{\boldsymbol{\sigma}_q^2}(\nabla_{\boldsymbol{\sigma}_q^2} \mathcal{L}_t, \boldsymbol{\sigma}_{q,t}^2; \phi), \tag{8}$$

where $f_t^{\boldsymbol{\mu}_q}$ and $f_t^{\boldsymbol{\sigma}_q^2}$ are the iterative inference models for updating $\boldsymbol{\mu}_q$ and $\boldsymbol{\sigma}_q^2$ respectively. In practice, these models can be combined, with shared inputs and model parameters but separate outputs to update each term.

In Appendix A, we derive the stochastic gradients $\nabla_{\boldsymbol{\mu}_q} \mathcal{L}$ and $\nabla_{\boldsymbol{\sigma}_q^2} \mathcal{L}$ for the cases where $p_\theta(\mathbf{x}|\mathbf{z})$ takes a Gaussian and Bernoulli form, though *any* output distribution can be used. Generally, these gradients are comprised of (1) errors, expressing the mismatch in distributions, and (2) Jacobian matrices, which invert the generative mappings. For instance, assuming a Gaussian output density, $p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_\mathbf{x}, \operatorname{diag} \boldsymbol{\sigma}_\mathbf{x}^2)$, the gradient for $\boldsymbol{\mu}_q$ is

$$\nabla_{\boldsymbol{\mu}_q} \mathcal{L} = \mathbf{J}^\intercal \boldsymbol{\varepsilon}_\mathbf{x} - \boldsymbol{\varepsilon}_\mathbf{z}, \tag{9}$$

where the Jacobian ($\mathbf{J}$), bottom-up errors ($\boldsymbol{\varepsilon}_\mathbf{x}$), and top-down errors ($\boldsymbol{\varepsilon}_\mathbf{z}$) are defined as

$$\mathbf{J} \equiv \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left[\frac{\partial \boldsymbol{\mu}_\mathbf{x}}{\partial \boldsymbol{\mu}_q} \right], \tag{10}$$

$$\boldsymbol{\varepsilon}_\mathbf{x} \equiv \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})}[(\mathbf{x} - \boldsymbol{\mu}_\mathbf{x})/\boldsymbol{\sigma}_\mathbf{x}^2], \tag{11}$$

$$\boldsymbol{\varepsilon}_\mathbf{z} \equiv \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})}[(\mathbf{z} - \boldsymbol{\mu}_p)/\boldsymbol{\sigma}_p^2]. \tag{12}$$

Here, we have assumed $\boldsymbol{\mu}_\mathbf{x}$ is a function of $\mathbf{z}$ and $\boldsymbol{\sigma}_\mathbf{x}^2$ is a global parameter. The gradient $\nabla_{\boldsymbol{\sigma}_q^2} \mathcal{L}$ is comprised of similar terms as well as an additional term penalizing approximate posterior entropy. Inspecting and understanding the composition of the gradients reveals the forces pushing the approximate posterior toward agreement with the data, through $\boldsymbol{\varepsilon}_\mathbf{x}$, and agreement with the prior, through $\boldsymbol{\varepsilon}_\mathbf{z}$. In other words, *inference is as much a top-down process as it is a bottom-up process*, and the optimal combination of these terms is given by the approximate posterior gradients. As discussed in Section 4.1, standard inference models have traditionally been purely bottom-up, only encoding the data.

4.1. Reinterpreting Top-Down Inference

To increase the model capacity of latent variable models, it is common to add higher-level latent variables, thereby providing flexible *empirical priors* on lower-level variables. Traditionally, corresponding standard inference models were parmeterized as purely bottom-up (e.g. Fig. 1 of (Rezende et al., 2014)). It was later found to be beneficial to incorporate top-down information from higher-level variables in the inference model, the given intuition being that "*a purely bottom-up inference process ... does not correspond well with real perception*" (Sønderby et al., 2016), however, a rigorous justification of this technique was lacking.

Iterative inference models, or rather, the gradients that they encode, provide a theoretical explanation for this previously empirical heuristic. As seen in eq. 9, the approximate posterior parameters are optimized to agree with the prior, while also fitting the conditional likelihood to the data. Analogous terms appear in the gradients for hierarchical models. For instance, in a chain-structured hierarchical model, the gradient of $\boldsymbol{\mu}_q^\ell$, the approximate posterior mean at layer ℓ, is

$$\nabla_{\boldsymbol{\mu}_q^\ell} \mathcal{L} = \mathbf{J}^{\ell\intercal} \boldsymbol{\varepsilon}_{\mathbf{z}}^{\ell-1} - \boldsymbol{\varepsilon}_{\mathbf{z}}^{\ell}, \tag{13}$$

where $\mathbf{J}^\ell$ is the Jacobian of the generative mapping at layer ℓ and $\boldsymbol{\varepsilon}_{\mathbf{z}}^\ell$ is defined similarly to eq. 12. $\boldsymbol{\varepsilon}_{\mathbf{z}}^\ell$ depends on the top-down prior at layer ℓ, which, unlike the single-level case, varies across data examples. Thus, a purely bottom-up inference procedure may struggle, as it must model both the bottom-up data dependence as well as the top-down prior. Top-down inference (Sønderby et al., 2016) explicitly uses the prior to perform inference. Iterative inference models instead rely on approximate posterior gradients, which naturally account for both bottom-up and top-down influences.

4.2. Approximating Approximate Posterior Derivatives

In the formulation of iterative inference models given in eq. 6, inference optimization is restricted to first-order approximate posterior derivatives. Thus, it may require many inference iterations to reach reasonable approximate posterior estimates. Rather than calculate costly higher-order derivatives, we can take a different approach.

Approximate posterior derivatives (e.g. eq. 9 and higher-order derivatives) are essentially defined by the errors at the current estimate, as the other factors, such as the Jacobian matrices, are internal to the model. Thus, the errors provide more general information about the curvature beyond the gradient. As iterative inference models already learn to perform approximate posterior updates, it is natural to ask whether the errors provide a sufficient signal for faster inference optimization. In other words, we may be able to offload approximate posterior derivative calculation onto the inference model, yielding a model that requires fewer inference iterations while maintaining or possibly improving computational efficiency.

Comparing with eqs. 7 and 8, the form of this new iterative inference model is

$$\boldsymbol{\mu}_{q,t+1} = f_t^{\boldsymbol{\mu}_q}(\boldsymbol{\varepsilon}_{\mathbf{x},t}, \boldsymbol{\varepsilon}_{\mathbf{z},t}, \boldsymbol{\mu}_{q,t}; \phi), \tag{14}$$

$$\boldsymbol{\sigma}^2_{q,t+1} = f_t^{\boldsymbol{\sigma}^2_q}(\boldsymbol{\varepsilon}_{\mathbf{x},t}, \boldsymbol{\varepsilon}_{\mathbf{z},t}, \boldsymbol{\sigma}^2_{q,t}; \phi), \tag{15}$$

where, again, these models can be shared, with separate outputs per parameter. In Section 5.2, we empirically find that models of this form converge to better solutions than gradient-encoding models when given fewer inference iterations. It is also worth noting that this error encoding scheme is similar to DRAW (Gregor et al., 2015). However, in addition to architectural differences in the generative model, DRAW and later extensions do not include top-down errors (Gregor et al., 2016), nor error precision-weighting.

4.3. Generalizing Standard Inference Models

Under certain assumptions on single-level latent Gaussian models, iterative inference models of the form in Section 4.2 *generalize* standard inference models. First, note that $\boldsymbol{\varepsilon}_{\mathbf{x}}$ (eq. 11) is a stochastic affine transformation of $\mathbf{x}$:

$$\boldsymbol{\varepsilon}_{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{b}, \tag{16}$$

where

$$\mathbf{A} \equiv \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[(\operatorname{diag} \boldsymbol{\sigma}^2_{\mathbf{x}})^{-1}\right], \tag{17}$$

$$\mathbf{b} \equiv -\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}\left[\frac{\boldsymbol{\mu}_{\mathbf{x}}}{\boldsymbol{\sigma}^2_{\mathbf{x}}}\right]. \tag{18}$$

Reasonably assuming that the initial approximate posterior and prior are both constant, then in expectation, $\mathbf{A}$, $\mathbf{b}$, and $\boldsymbol{\varepsilon}_{\mathbf{z}}$ are constant across all data examples at the first inference iteration. Using proper weight initialization and input normalization, it is equivalent to input $\mathbf{x}$ or an affine transformation of $\mathbf{x}$ into a fully-connected neural network. Therefore, *standard inference models are equivalent to the special case of a one-step iterative inference model*. Thus, we can interpret standard inference models as learning a map of local curvature around a fixed approximate posterior estimate. Iterative inference models, on the other hand, learn to traverse the optimization landscape more generally.

5. Experiments

Using latent Gaussian models, we performed an empirical evaluation of iterative inference models on both image and text data. For images, we used **MNIST** (LeCun et al., 1998), **Omniglot** (Lake et al., 2013), **Street View House Numbers (SVHN)** (Netzer et al., 2011), and **CIFAR-10** (Krizhevsky & Hinton, 2009). MNIST and Omniglot were dynamically binarized and modeled with Bernoulli output

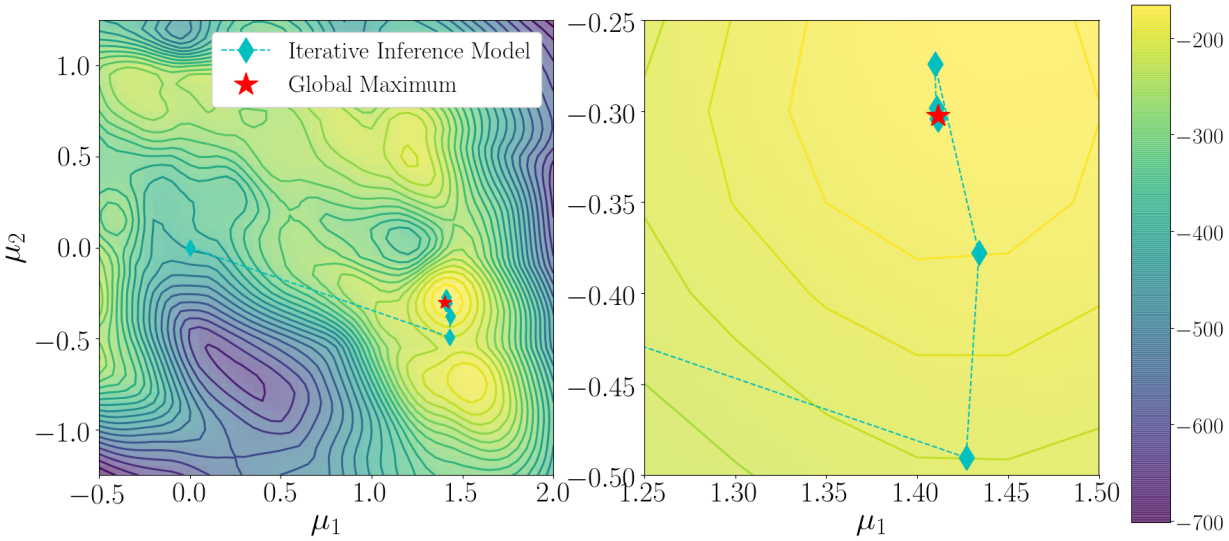


Figure 3. **Direct visualization of iterative amortized inference optimization.** Optimization trajectory on $\mathcal{L}$ (in *nats*) for an iterative inference model with a 2D latent Gaussian model for a particular MNIST example. The iterative inference model adaptively adjusts inference update step sizes to iteratively refine the approximate posterior estimate.

distributions, and SVHN and CIFAR-10 were modeled with Gaussian output densities, using the procedure from (Gregor et al., 2016). For text, we used **RCV1** (Lewis et al., 2004), with word count data modeled with a multinomial output.

Details on implementing iterative inference models are found in Appendix B. The primary difficulty of training iterative inference models comes from shifting gradient and error distributions during the course of inference and learning. In some cases, we found it necessary to normalize these inputs using layer normalization (Ba et al., 2016). We also found it beneficial, though never necessary, to additionally encode the data itself, particularly when given few inference iterations (see Figure 7a). For comparison, all experiments use feedforward networks, though we observed similar results with recurrent inference models. Reported values of $\mathcal{L}$ were estimated using 1 sample, and reported values of $\log p(\mathbf{x})$ and perplexity (Tables 1 & 2) were estimated using 5,000 importance weighted samples. Additional experiment details, including model architectures, can be found in Appendix C. Accompanying code can be found on GitHub at `joelouismarino/iterative_inference`.

Section 5.1 demonstrates the optimization capabilities of iterative inference models. Section 5.2 explores two methods by which to further improve the modeling performance of these models. Section 5.3 provides a quantitative comparison between standard and iterative inference models.

5.1. Approximate Inference Optimization

We begin with a series of experiments that demonstrate the inference optimization capabilities of iterative inference models. These experiments confirm that iterative inference models indeed learn to perform inference optimization through an adaptive iterative estimation procedure. These results highlight the qualitative differences between this inference optimization procedure and that of standard inference models. That is, iterative inference models are able to effectively utilize multiple inference iterations rather than collapsing to static, one-step encoders.

Direct Visualization As in Section 3.1, we directly visualize iterative inference optimization in a 2-D latent Gaussian model trained on MNIST with a point estimate approximate posterior. Model architectures are identical to those used in Section 3.1, with additional details found in Appendix C.1. Shown in Figure 3 is a 16-step inference optimization trajectory taken by the iterative inference model for a particular example. The model adaptively adjusts inference update step sizes to navigate the optimization surface, quickly arriving and remaining at a near-optimal estimate.

$\mathcal{L}$ During Inference We can quantify and compare optimization performance through the ELBO. In Figure 4, we plot the average ELBO on the MNIST validation set during inference, comparing iterative inference models with conventional optimizers. Details are in Appendix C.2. On average, the iterative inference model converges significantly *faster* to *better* estimates than the optimizers. The model actually has *less* derivative information than the optimizers; it only has access to the local gradient, whereas the optimizers use momentum and similar terms. The model's final estimates are also stable, despite only being trained using 16 inference iterations.

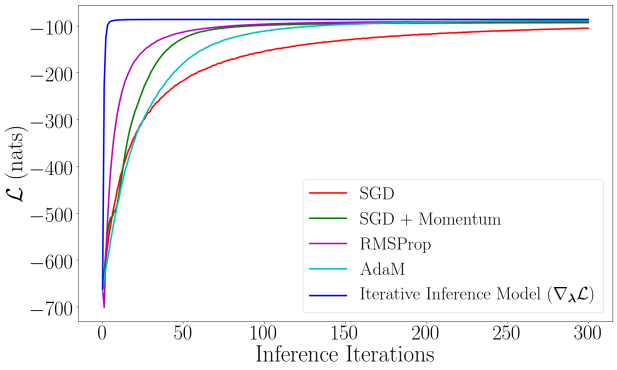


Figure 4. **Comparison of inference optimization performance between iterative inference models and conventional optimization techniques.** Plot shows ELBO, averaged over MNIST validation set. On average, the iterative inference model converges faster than conventional optimizers to better estimates. Note that the iterative inference model remains stable over hundreds of iterations, despite only being trained with 16 inference iterations.

Reconstructions Approximate inference optimization can also be visualized through image reconstructions. As the reconstruction term is typically the dominant term in $\mathcal{L}$, the output reconstructions should improve in terms of visual quality during inference optimization, resembling $\mathbf{x}$. We demonstrate this phenomenon with iterative inference models for several data sets in Figure 5. Additional reconstructions are shown in Appendix C.3.

Gradient Magnitudes During inference optimization, iterative inference models should ideally obtain approximate posterior estimates near local maxima. The approximate posterior gradient magnitudes should thus decrease during inference. Using a model trained on RCV1, we recorded average gradient magnitudes for the approximate posterior mean during inference. In Figure 6, we plot these values throughout training, finding that they do, indeed, decrease. See Appendix C.4 for more details.

5.2. Additional Inference Iterations & Latent Samples

We highlight two sources that allow iterative inference models to further improve modeling performance: additional inference iterations and samples. Additional inference iterations allow the model to further refine approximate posterior estimates. Using MNIST, we trained models by encoding approximate posterior gradients ($\nabla_{\boldsymbol{\lambda}}\mathcal{L}$) or errors ($\boldsymbol{\varepsilon}_{\mathbf{x}}, \boldsymbol{\varepsilon}_{\mathbf{z}}$), with or without the data ($\mathbf{x}$), for 2, 5, 10, and 16 inference iterations. While we kept the model architectures identical, the encoded terms affect the number of input parameters to each model. For instance, the small size of $\mathbf{z}$ relative to $\mathbf{x}$ gives the gradient encoding model *fewer* input parameters than a standard inference model. The other models have more input parameters. Results are shown in Figure 7a, where we observe improved performance with increasing inference iterations. All iterative inference models outperformed standard inference models. Note that encoding errors to approximate higher-order derivatives helps when training with fewer inference iterations.

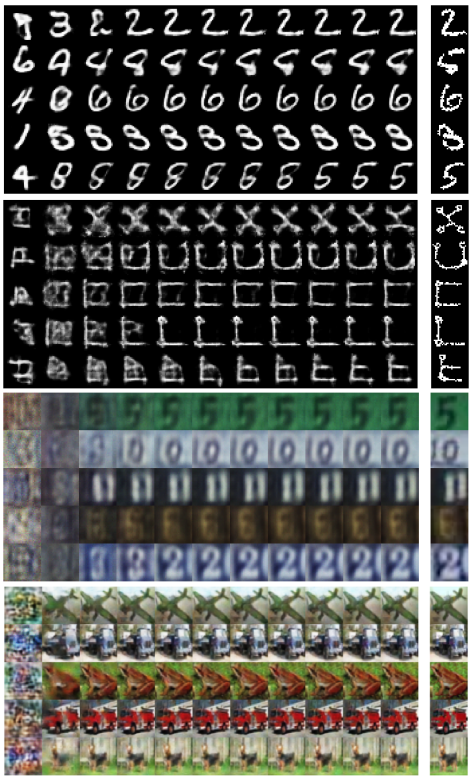


Figure 5. **Reconstructions over inference iterations** (left to right) for (top to bottom) MNIST, Omniglot, SVHN, and CIFAR-10. Data examples are shown on the right. Reconstructions become gradually sharper, remaining stable after many iterations.

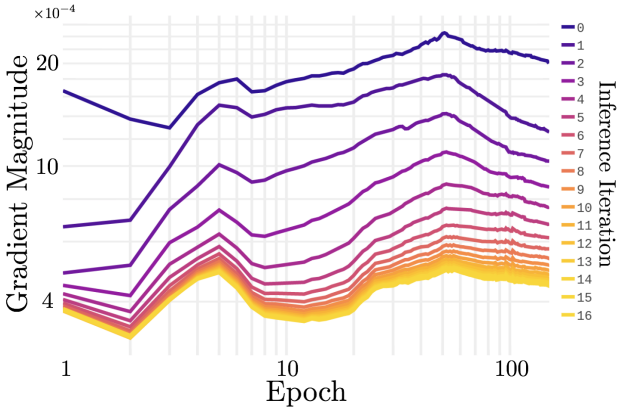
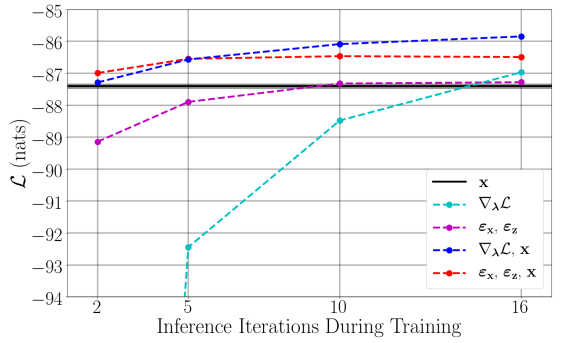
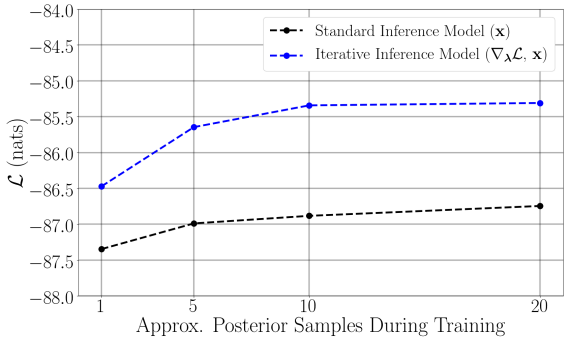


Figure 6. **Gradient magnitudes** (*vertical axis*) over inference iterations (indexed by color on right) during training (*horizontal axis*) on RCV1. Approx. posterior mean gradient magnitudes decrease over inference iterations as estimates approach local maxima.



(a)



(b)

Figure 7. ELBO for standard and iterative inference models on MNIST for **(a)** additional inference iterations during training and **(b)** additional samples. Iterative inference models improve significantly with both quantities. Lines do not imply interpolation.

Additional approximate posterior samples provide more precise gradient and error estimates, potentially allowing an iterative inference model to output improved updates. To verify this, we trained standard and iterative inference models on MNIST using 1, 5, 10, and 20 approximate posterior samples. Iterative inference models were trained by encoding the data ($\mathbf{x}$) and approximate posterior gradients ($\nabla_{\boldsymbol{\lambda}} \mathcal{L}$) for 5 iterations. Results are shown in Figure 7b. Iterative inference models improve by more than 1 nat with additional samples, further widening the improvement over similar standard inference models.

5.3. Comparison with Standard Inference Models

We now provide a quantitative performance comparison between standard and iterative inference models on MNIST, CIFAR-10, and RCV1. Inference model architectures are identical across each comparison, with the exception of input parameters. Details are found in Appendix C.7. Table 1 contains estimated marginal log-likelihood performance on MNIST and CIFAR-10. Table 2 contains estimated perplexity on RCV1[2]. In each case, iterative inference models outperform standard inference models. This holds for both single-level and hierarchical models. We observe larger improvements on the high-dimensional RCV1 data set, consistent with (Krishnan et al., 2018). Because the generative model architectures are kept fixed, performance improvements demonstrate improvements in inference optimization.

[2]Perplexity re-weights log-likelihood by document length.

Table 1. Negative log likelihood on MNIST (in *nats*) and CIFAR-10 (in *bits/input dim.*) for standard and iterative inference models.

	$-\log p(\mathbf{x})$
MNIST	
Single-Level	
Standard	84.14 ± 0.02
Iterative	$\mathbf{83.84 \pm 0.05}$
Hierarchical	
Standard	82.63 ± 0.01
Iterative	$\mathbf{82.457 \pm 0.001}$
CIFAR-10	
Single-Level	
Standard	5.823 ± 0.001
Iterative	$\mathbf{5.64 \pm 0.03}$
Hierarchical	
Standard	5.565 ± 0.002
Iterative	$\mathbf{5.456 \pm 0.005}$

Table 2. Perplexity on RCV1 for standard and iterative inference models.

	Perplexity	$\leq$
RCV1		
Krishnan et al. (2018)		331
Standard	323 ± 3	377.4 ± 0.5
Iterative	$\mathbf{285.0 \pm 0.1}$	$\mathbf{314 \pm 1}$

6. Conclusion

We have proposed iterative inference models, which learn to refine inference estimates by encoding approximate posterior gradients or errors. These models generalize and extend standard inference models, and by naturally accounting for priors during inference, these models provide insight and justification for top-down inference. Through empirical evaluations, we have demonstrated that iterative inference models learn to perform variational inference optimization, with advantages over current inference techniques shown on several benchmark data sets. However, this comes with the limitation of requiring additional computation over similar standard inference models. While we discussed the relevance of iterative inference models to hierarchical latent variable models, sequential latent variable models also contain empirical priors. In future work, we hope to apply iterative inference models to the online filtering setting, where fewer inference iterations, and thus less additional computation, may be required at each time step.

Acknowledgements

We would like to thank the reviewers as well as Peter Carr, Oisin Mac Aodha, Grant Van Horn, and Matteo Ruggero Ronchi for their insightful feedback. This research was supported in part by JPL PDF 1584398 and NSF 1564330.

References

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., and de Freitas, N. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 3981–3989, 2016.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Cremer, C., Li, X., and Duvenaud, D. Inference suboptimality in variational autoencoders. *NIPS Workshop on Advances in Approximate Bayesian Inference*, 2017.

Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.

Gershman, S. and Goodman, N. Amortized inference in probabilistic reasoning. In *Proceedings of the Cognitive Science Society*, volume 36, 2014.

Gregor, K., Danihelka, I., Mnih, A., Blundell, C., and Wierstra, D. Deep autoregressive networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 1242–1250, 2014.

Gregor, K., Danihelka, I., Graves, A., Rezende, D. J., and Wierstra, D. Draw: A recurrent neural network for image generation. *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 1462–1471, 2015.

Gregor, K., Besse, F., Rezende, D. J., Danihelka, I., and Wierstra, D. Towards conceptual compression. In *Advances In Neural Information Processing Systems (NIPS)*, pp. 3549–3557, 2016.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

Hjelm, D., Salakhutdinov, R. R., Cho, K., Jojic, N., Calhoun, V., and Chung, J. Iterative refinement of the approximate posterior for directed belief networks. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 4691–4699, 2016.

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

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. An introduction to variational methods for graphical models. *NATO ASI SERIES D BEHAVIOURAL AND SOCIAL SCIENCES*, 89:105–162, 1998.

Karl, M., Soelch, M., Bayer, J., and van der Smagt, P. Deep variational bayes filters: Unsupervised learning of state space models from raw data. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Kim, Y., Wiseman, S., Miller, A. C., Sontag, D., and Rush, A. M. Semi-amortized variational autoencoders. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

Kingma, D. P. and Welling, M. Stochastic gradient vb and the variational auto-encoder. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

Krishnan, R. G., Liang, D., and Hoffman, M. On the challenges of learning with inference networks on sparse, high-dimensional data. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 143–151, 2018.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. 2009.

Lake, B. M., Salakhutdinov, R. R., and Tenenbaum, J. One-shot learning by inverting a compositional causal process. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2526–2534, 2013.

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

Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. Rcv1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5(Apr):361–397, 2004.

Neal, R. M. and Hinton, G. E. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pp. 355–368. Springer, 1998.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, 2011.

Putzky, P. and Welling, M. Recurrent inference machines for solving inverse problems. *arXiv preprint arXiv:1706.04008*, 2017.

Ranganath, R., Gerrish, S., and Blei, D. Black box variational inference. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 814–822, 2014.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 1278–1286, 2014.

Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. Ladder variational autoencoders. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 3738–3746, 2016.

Xue, T., Wu, J., Bouman, K., and Freeman, B. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 91–99, 2016.

A. Approximate Posterior Gradients for Latent Gaussian Models

A.1. Model & Variational Objective

Consider a latent variable model, $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})$, where the prior on $\mathbf{z}$ is a factorized Gaussian density, $p_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_p, \operatorname{diag} \boldsymbol{\sigma}_\mathbf{x}^2)$, and the conditional likelihood, $p_\theta(\mathbf{x}|\mathbf{z})$, depends on the type of data (e.g. Bernoulli for binary observations or Gaussian for continuous observations). We introduce an approximate posterior distribution, $q(\mathbf{z}|\mathbf{x})$, which can be any parametric probability density defined over real values. Here, we assume that q also takes the form of a factorized Gaussian density, $q(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_q, \operatorname{diag} \boldsymbol{\sigma}_q^2)$. The objective during variational inference is to maximize $\mathcal{L}$ w.r.t. the parameters of $q(\mathbf{z}|\mathbf{x})$, i.e. $\boldsymbol{\mu}_q$ and $\boldsymbol{\sigma}_q^2$:

$$\boldsymbol{\mu}_q^*, \boldsymbol{\sigma}_q^{2*} = \underset{\boldsymbol{\mu}_q, \boldsymbol{\sigma}_q^2}{\arg\max}\, \mathcal{L}. \tag{1}$$

To solve this optimization problem, we will use the gradients $\nabla_{\boldsymbol{\mu}_q} \mathcal{L}$ and $\nabla_{\boldsymbol{\sigma}_q^2} \mathcal{L}$, which we now derive. The objective can be written as:

$$\begin{aligned} \mathcal{L} &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}|\mathbf{x}) \right] && (2) \\ &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} \left[\log p_\theta(\mathbf{x}|\mathbf{z}) + \log p_\theta(\mathbf{z}) - \log q(\mathbf{z}|\mathbf{x}) \right]. && (3) \end{aligned}$$

Plugging in $p_\theta(\mathbf{z})$ and $q(\mathbf{z}|\mathbf{x})$:

$$\mathcal{L} = \mathbb{E}_{\mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_q, \operatorname{diag} \boldsymbol{\sigma}_q^2)} \left[\log p_\theta(\mathbf{x}|\mathbf{z}) + \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_p, \operatorname{diag} \boldsymbol{\sigma}_p^2) - \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_q, \operatorname{diag} \boldsymbol{\sigma}_q^2) \right] \tag{4}$$

Since expectation and differentiation are linear operators, we can take the expectation and derivative of each term individually.

A.2. Gradient of the Log-Prior

We can write the log-prior as:

$$\log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_p, (\operatorname{diag} \boldsymbol{\sigma}_p^2) = -\frac{1}{2} \log \left((2\pi)^{n_\mathbf{z}} |\operatorname{diag} \boldsymbol{\sigma}_p^2| \right) - \frac{1}{2} (\mathbf{z} - \boldsymbol{\mu}_p)^\intercal (\operatorname{diag} \boldsymbol{\sigma}_p^2)^{-1} (\mathbf{z} - \boldsymbol{\mu}_p), \tag{5}$$

where $n_\mathbf{z}$ is the dimensionality of $\mathbf{z}$. We want to evaluate the following terms:

$$\nabla_{\boldsymbol{\mu}_q} \mathbb{E}_{\mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_q, \operatorname{diag} \boldsymbol{\sigma}_q^2)} \left[-\frac{1}{2} \log \left((2\pi)^{n_\mathbf{z}} |\operatorname{diag} \boldsymbol{\sigma}_p^2| \right) - \frac{1}{2} (\mathbf{z} - \boldsymbol{\mu}_p)^\intercal (\operatorname{diag} \boldsymbol{\sigma}_p^2)^{-1} (\mathbf{z} - \boldsymbol{\mu}_p) \right] \tag{6}$$

and

$$\nabla_{\boldsymbol{\sigma}_q^2} \mathbb{E}_{\mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_q, \operatorname{diag} \boldsymbol{\sigma}_q^2)} \left[-\frac{1}{2} \log \left((2\pi)^{n_\mathbf{z}} |\operatorname{diag} \boldsymbol{\sigma}_p^2| \right) - \frac{1}{2} (\mathbf{z} - \boldsymbol{\mu}_p)^\intercal (\operatorname{diag} \boldsymbol{\sigma}_p^2)^{-1} (\mathbf{z} - \boldsymbol{\mu}_p) \right]. \tag{7}$$

To take these derivatives, we will use the reparameterization trick (Kingma & Welling, 2014; Rezende et al., 2014) to re-express $\mathbf{z} = \boldsymbol{\mu}_q + \boldsymbol{\sigma}_q \odot \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is an auxiliary standard Gaussian variable, and $\odot$ denotes the element-wise product. We can now perform the expectations over $\boldsymbol{\epsilon}$, allowing us to bring the gradient operators inside the expectation brackets. The first term in eqs. 6 and 7 does not depend on $\boldsymbol{\mu}_q$ or $\boldsymbol{\sigma}_q^2$, so we can write:

$$\mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I})} \left[\nabla_{\boldsymbol{\mu}_q} \left(-\frac{1}{2} (\boldsymbol{\mu}_q + \boldsymbol{\sigma}_q \odot \boldsymbol{\epsilon} - \boldsymbol{\mu}_p)^\intercal (\operatorname{diag} \boldsymbol{\sigma}_p^2)^{-1} (\boldsymbol{\mu}_q + \boldsymbol{\sigma}_q \odot \boldsymbol{\epsilon} - \boldsymbol{\mu}_p) \right) \right] \tag{8}$$

and

$$\mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I})} \left[\nabla_{\boldsymbol{\sigma}_q^2} \left(-\frac{1}{2} (\boldsymbol{\mu}_q + \boldsymbol{\sigma}_q \odot \boldsymbol{\epsilon} - \boldsymbol{\mu}_p)^\intercal (\operatorname{diag} \boldsymbol{\sigma}_p^2)^{-1} (\boldsymbol{\mu}_q + \boldsymbol{\sigma}_q \odot \boldsymbol{\epsilon} - \boldsymbol{\mu}_p) \right) \right]. \tag{9}$$

To simplify notation, we define the following term:

$$\boldsymbol{\xi} \equiv (\operatorname{diag} \boldsymbol{\sigma}_p^2)^{-1/2} (\boldsymbol{\mu}_q + \boldsymbol{\sigma}_q \odot \boldsymbol{\epsilon} - \boldsymbol{\mu}_p), \tag{10}$$

allowing us to rewrite eqs. 8 and 9 as:

$$\mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I})} \left[\nabla_{\boldsymbol{\mu}_q} \left(-\frac{1}{2} \boldsymbol{\xi}^\intercal \boldsymbol{\xi} \right) \right] = \mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I})} \left[-\frac{\partial \boldsymbol{\xi}^\intercal}{\partial \boldsymbol{\mu}_q} \boldsymbol{\xi} \right] \tag{11}$$

and

$$\mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{I})}\left[\nabla_{\boldsymbol{\sigma}_q^2}\left(-\frac{1}{2}\boldsymbol{\xi}^\intercal\boldsymbol{\xi}\right)\right] = \mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{I})}\left[-\frac{\partial\boldsymbol{\xi}^\intercal}{\partial\boldsymbol{\sigma}_q^2}\boldsymbol{\xi}\right]. \tag{12}$$

We must now find $\frac{\partial\boldsymbol{\xi}}{\partial\boldsymbol{\mu}_q}$ and $\frac{\partial\boldsymbol{\xi}}{\partial\boldsymbol{\sigma}_q^2}$:

$$\frac{\partial\boldsymbol{\xi}}{\partial\boldsymbol{\mu}_q} = \frac{\partial}{\partial\boldsymbol{\mu}_q}\left((\operatorname{diag}\boldsymbol{\sigma}_p^2)^{-1/2}(\boldsymbol{\mu}_q + \boldsymbol{\sigma}_q\odot\boldsymbol{\epsilon} - \boldsymbol{\mu}_p)\right) = (\operatorname{diag}\boldsymbol{\sigma}_p^2)^{-1/2} \tag{13}$$

and

$$\frac{\partial\boldsymbol{\xi}}{\partial\boldsymbol{\sigma}_q^2} = \frac{\partial}{\partial\boldsymbol{\sigma}_q^2}\left((\operatorname{diag}\boldsymbol{\sigma}_p^2)^{-1/2}(\boldsymbol{\mu}_q + \boldsymbol{\sigma}_q\odot\boldsymbol{\epsilon} - \boldsymbol{\mu}_p)\right) = (\operatorname{diag}\boldsymbol{\sigma}_p^2)^{-1/2}\operatorname{diag}\frac{\boldsymbol{\epsilon}}{2\boldsymbol{\sigma}_q}, \tag{14}$$

where division is performed element-wise. Plugging eqs. 13 and 14 back into eqs. 11 and 12, we get:

$$\mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{I})}\left[-\left((\operatorname{diag}\boldsymbol{\sigma}_p^2)^{-1/2}\right)^\intercal(\operatorname{diag}\boldsymbol{\sigma}_p^2)^{-1/2}(\boldsymbol{\mu}_q + \boldsymbol{\sigma}_q\odot\boldsymbol{\epsilon} - \boldsymbol{\mu}_p)\right] \tag{15}$$

and

$$\mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{I})}\left[-\left(\operatorname{diag}\frac{\boldsymbol{\epsilon}}{2\boldsymbol{\sigma}_q}\right)^\intercal\left((\operatorname{diag}\boldsymbol{\sigma}_p^2)^{-1/2}\right)^\intercal(\operatorname{diag}\boldsymbol{\sigma}_p^2)^{-1/2}(\boldsymbol{\mu}_q + \boldsymbol{\sigma}_q\odot\boldsymbol{\epsilon} - \boldsymbol{\mu}_p)\right]. \tag{16}$$

Putting everything together, we can express the gradients as:

$$\nabla_{\boldsymbol{\mu}_q}\mathbb{E}_{\mathcal{N}(\mathbf{z};\boldsymbol{\mu}_q,\operatorname{diag}\boldsymbol{\sigma}_q^2)}\left[\log\mathcal{N}(\mathbf{z};\boldsymbol{\mu}_p,\operatorname{diag}\boldsymbol{\sigma}_p^2)\right] = \mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{I})}\left[-\frac{\boldsymbol{\mu}_q + \boldsymbol{\sigma}_q\odot\boldsymbol{\epsilon} - \boldsymbol{\mu}_p}{\boldsymbol{\sigma}_p^2}\right], \tag{17}$$

and

$$\nabla_{\boldsymbol{\sigma}_q^2}\mathbb{E}_{\mathcal{N}(\mathbf{z};\boldsymbol{\mu}_q,\operatorname{diag}\boldsymbol{\sigma}_q^2)}\left[\log\mathcal{N}(\mathbf{z};\boldsymbol{\mu}_p,\operatorname{diag}\boldsymbol{\sigma}_p^2)\right] = \mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{I})}\left[-\left(\operatorname{diag}\frac{\boldsymbol{\epsilon}}{2\boldsymbol{\sigma}_q}\right)^\intercal\frac{\boldsymbol{\mu}_q + \boldsymbol{\sigma}_q\odot\boldsymbol{\epsilon} - \boldsymbol{\mu}_p}{\boldsymbol{\sigma}_p^2}\right]. \tag{18}$$

A.3. Gradient of the Log-Approximate Posterior

We can write the log-approximate posterior as:

$$\log\mathcal{N}(\mathbf{z};\boldsymbol{\mu}_q,\operatorname{diag}\boldsymbol{\sigma}_q^2) = -\frac{1}{2}\log\left((2\pi)^{n_\mathbf{z}}|\operatorname{diag}\boldsymbol{\sigma}_q^2|\right) - \frac{1}{2}(\mathbf{z}-\boldsymbol{\mu}_q)^\intercal(\operatorname{diag}\boldsymbol{\sigma}_q^2)^{-1}(\mathbf{z}-\boldsymbol{\mu}_q), \tag{19}$$

where $n_\mathbf{z}$ is the dimensionality of $\mathbf{z}$. Again, we will use the reparameterization trick to re-express the gradients. However, notice what happens when plugging the reparameterized $\mathbf{z} = \boldsymbol{\mu}_q + \boldsymbol{\sigma}_q\odot\boldsymbol{\epsilon}$ into the second term of eq. 19:

$$-\frac{1}{2}(\boldsymbol{\mu}_q + \boldsymbol{\sigma}_q\odot\boldsymbol{\epsilon} - \boldsymbol{\mu}_q)^\intercal(\operatorname{diag}\boldsymbol{\sigma}_q^2)^{-1}(\boldsymbol{\mu}_q + \boldsymbol{\sigma}_q\odot\boldsymbol{\epsilon} - \boldsymbol{\mu}_q) = -\frac{1}{2}\frac{(\boldsymbol{\sigma}_q\odot\boldsymbol{\epsilon})^\intercal(\boldsymbol{\sigma}_q\odot\boldsymbol{\epsilon})}{\boldsymbol{\sigma}_q^2} = -\frac{1}{2}\boldsymbol{\epsilon}^\intercal\boldsymbol{\epsilon}. \tag{20}$$

This term does not depend on $\boldsymbol{\mu}_q$ or $\boldsymbol{\sigma}_q^2$. Also notice that the first term in eq. 19 depends only on $\boldsymbol{\sigma}_q^2$. Therefore, the gradient of the entire term w.r.t. $\boldsymbol{\mu}_q$ is zero:

$$\nabla_{\boldsymbol{\mu}_q}\mathbb{E}_{\mathcal{N}(\mathbf{z};\boldsymbol{\mu}_q,\operatorname{diag}\boldsymbol{\sigma}_q^2)}\left[\log\mathcal{N}(\mathbf{z};\boldsymbol{\mu}_q,\operatorname{diag}\boldsymbol{\sigma}_q^2)\right] = \mathbf{0}. \tag{21}$$

The gradient w.r.t. $\boldsymbol{\sigma}_q^2$ is

$$\nabla_{\boldsymbol{\sigma}_q^2}\left(-\frac{1}{2}\log\left((2\pi)^{n_\mathbf{z}}|\operatorname{diag}\boldsymbol{\sigma}_q^2|\right)\right) = -\frac{1}{2}\nabla_{\boldsymbol{\sigma}_q^2}\left(\log|\operatorname{diag}\boldsymbol{\sigma}_q^2|\right) = -\frac{1}{2}\nabla_{\boldsymbol{\sigma}_q^2}\sum_j\log\sigma_{q,j}^2 = -\frac{\mathbf{1}}{2\boldsymbol{\sigma}_q^2}. \tag{22}$$

Note that the expectation has been dropped, as the term does not depend on the value of the sampled $\mathbf{z}$. Thus, the gradient of the entire term w.r.t. $\boldsymbol{\sigma}_q^2$ is:

$$\nabla_{\boldsymbol{\sigma}_q^2}\mathbb{E}_{\mathcal{N}(\mathbf{z};\boldsymbol{\mu}_q,\operatorname{diag}\boldsymbol{\sigma}_q^2)}\left[\log\mathcal{N}(\mathbf{z};\boldsymbol{\mu}_q,\operatorname{diag}\boldsymbol{\sigma}_q^2)\right] = -\frac{\mathbf{1}}{2\boldsymbol{\sigma}_q^2}. \tag{23}$$

A.4. Gradient of the Log-Conditional Likelihood

The form of the conditional likelihood will depend on the data, e.g. binary, discrete, continuous, etc. Here, we derive the gradient for Bernoulli (binary) and Gaussian (continuous) conditional likelihoods.

Bernoulli Output Distribution The log of a Bernoulli output distribution takes the form:

$$\log \mathcal{B}(\mathbf{x}; \boldsymbol{\mu}_\mathbf{x}) = (\log \boldsymbol{\mu}_\mathbf{x})^\intercal \mathbf{x} + (\log(\mathbf{1} - \boldsymbol{\mu}_\mathbf{x}))^\intercal (\mathbf{1} - \mathbf{x}), \tag{24}$$

where $\boldsymbol{\mu}_\mathbf{x} = \boldsymbol{\mu}_\mathbf{x}(\mathbf{z}, \theta)$ is the mean of the output distribution. We drop the explicit dependence on $\mathbf{z}$ and θ to simplify notation. We want to compute the gradients

$$\nabla_{\boldsymbol{\mu}_q} \mathbb{E}_{\mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_q, \operatorname{diag} \boldsymbol{\sigma}_q^2)} \left[(\log \boldsymbol{\mu}_\mathbf{x})^\intercal \mathbf{x} + (\log(\mathbf{1} - \boldsymbol{\mu}_\mathbf{x}))^\intercal (\mathbf{1} - \mathbf{x}) \right] \tag{25}$$

and

$$\nabla_{\boldsymbol{\sigma}_q^2} \mathbb{E}_{\mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_q, \operatorname{diag} \boldsymbol{\sigma}_q^2)} \left[(\log \boldsymbol{\mu}_\mathbf{x})^\intercal \mathbf{x} + (\log(\mathbf{1} - \boldsymbol{\mu}_\mathbf{x}))^\intercal (\mathbf{1} - \mathbf{x}) \right]. \tag{26}$$

Again, we use the reparameterization trick to re-express the expectations, allowing us to bring the gradient operators inside the brackets. Using $\mathbf{z} = \boldsymbol{\mu}_q + \boldsymbol{\sigma}_q \odot \boldsymbol{\epsilon}$, eqs. 25 and 26 become:

$$\mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I})} \left[\nabla_{\boldsymbol{\mu}_q} \left((\log \boldsymbol{\mu}_\mathbf{x})^\intercal \mathbf{x} + (\log(\mathbf{1} - \boldsymbol{\mu}_\mathbf{x}))^\intercal (\mathbf{1} - \mathbf{x}) \right) \right] \tag{27}$$

and

$$\mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I})} \left[\nabla_{\boldsymbol{\sigma}_q^2} \left((\log \boldsymbol{\mu}_\mathbf{x})^\intercal \mathbf{x} + (\log(\mathbf{1} - \boldsymbol{\mu}_\mathbf{x}))^\intercal (\mathbf{1} - \mathbf{x}) \right) \right], \tag{28}$$

where $\boldsymbol{\mu}_\mathbf{x}$ is re-expressed as function of $\boldsymbol{\mu}_q, \boldsymbol{\sigma}_q^2, \boldsymbol{\epsilon}$, and θ. Distributing the gradient operators yields:

$$\mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I})} \left[\frac{\partial (\log \boldsymbol{\mu}_\mathbf{x})^\intercal}{\partial \boldsymbol{\mu}_q} \mathbf{x} + \frac{\partial (\log(\mathbf{1} - \boldsymbol{\mu}_\mathbf{x}))^\intercal}{\partial \boldsymbol{\mu}_q} (\mathbf{1} - \mathbf{x}) \right] \tag{29}$$

and

$$\mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I})} \left[\frac{\partial (\log \boldsymbol{\mu}_\mathbf{x})^\intercal}{\partial \boldsymbol{\sigma}_q^2} \mathbf{x} + \frac{\partial (\log(\mathbf{1} - \boldsymbol{\mu}_\mathbf{x}))^\intercal}{\partial \boldsymbol{\sigma}_q^2} (\mathbf{1} - \mathbf{x}) \right]. \tag{30}$$

Taking the partial derivatives and combining terms gives:

$$\mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I})} \left[\frac{\partial \boldsymbol{\mu}_\mathbf{x}}{\partial \boldsymbol{\mu}_q}^\intercal \frac{\mathbf{x}}{\boldsymbol{\mu}_\mathbf{x}} - \frac{\partial \boldsymbol{\mu}_\mathbf{x}}{\partial \boldsymbol{\mu}_q}^\intercal \frac{\mathbf{1} - \mathbf{x}}{\mathbf{1} - \boldsymbol{\mu}_\mathbf{x}} \right] = \mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I})} \left[\frac{\partial \boldsymbol{\mu}_\mathbf{x}}{\partial \boldsymbol{\mu}_q}^\intercal \frac{\mathbf{x} - \boldsymbol{\mu}_\mathbf{x}}{\boldsymbol{\mu}_\mathbf{x} \odot (\mathbf{1} - \boldsymbol{\mu}_\mathbf{x})} \right] \tag{31}$$

and

$$\mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I})} \left[\frac{\partial \boldsymbol{\mu}_\mathbf{x}}{\partial \boldsymbol{\sigma}_q^2}^\intercal \frac{\mathbf{x}}{\boldsymbol{\mu}_\mathbf{x}} - \frac{\partial \boldsymbol{\mu}_\mathbf{x}}{\partial \boldsymbol{\sigma}_q^2}^\intercal \frac{\mathbf{1} - \mathbf{x}}{\mathbf{1} - \boldsymbol{\mu}_\mathbf{x}} \right] = \mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon}; \mathbf{0}, \mathbf{I})} \left[\frac{\partial \boldsymbol{\mu}_\mathbf{x}}{\partial \boldsymbol{\sigma}_q^2}^\intercal \frac{\mathbf{x} - \boldsymbol{\mu}_\mathbf{x}}{\boldsymbol{\mu}_\mathbf{x} \odot (\mathbf{1} - \boldsymbol{\mu}_\mathbf{x})} \right]. \tag{32}$$

Gaussian Output Density The log of a Gaussian output density takes the form:

$$\log \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_\mathbf{x}, \operatorname{diag} \boldsymbol{\sigma}_\mathbf{x}^2) = -\frac{1}{2} \log \left((2\pi)^{n_\mathbf{x}} |\operatorname{diag} \boldsymbol{\sigma}_\mathbf{x}^2| \right) - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_\mathbf{x})^\intercal (\operatorname{diag} \boldsymbol{\sigma}_\mathbf{x}^2)^{-1} (\mathbf{x} - \boldsymbol{\mu}_\mathbf{x}), \tag{33}$$

where $\boldsymbol{\mu}_\mathbf{x} = \boldsymbol{\mu}_\mathbf{x}(\mathbf{z}, \theta)$ is the mean of the output distribution and $\boldsymbol{\sigma}_\mathbf{x}^2 = \boldsymbol{\sigma}_\mathbf{x}^2(\theta)$ is the variance. We assume $\boldsymbol{\sigma}_\mathbf{x}^2$ is not a function of $\mathbf{z}$ to simplify the derivation, however, using $\boldsymbol{\sigma}_\mathbf{x}^2 = \boldsymbol{\sigma}_\mathbf{x}^2(\mathbf{z}, \theta)$ is possible and would simply result in additional gradient terms in $\nabla_{\boldsymbol{\mu}_q} \mathcal{L}$ and $\nabla_{\boldsymbol{\sigma}_q^2} \mathcal{L}$. We want to compute the gradients

$$\nabla_{\boldsymbol{\mu}_q} \mathbb{E}_{\mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_q, \operatorname{diag} \boldsymbol{\sigma}_q^2)} \left[-\frac{1}{2} \log \left((2\pi)^{n_\mathbf{x}} |\operatorname{diag} \boldsymbol{\sigma}_\mathbf{x}^2| \right) - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_\mathbf{x})^\intercal (\operatorname{diag} \boldsymbol{\sigma}_\mathbf{x}^2)^{-1} (\mathbf{x} - \boldsymbol{\mu}_\mathbf{x}) \right] \tag{34}$$

and

$$\nabla_{\boldsymbol{\sigma}_q^2} \mathbb{E}_{\mathcal{N}(\mathbf{z}; \boldsymbol{\mu}_q, \operatorname{diag} \boldsymbol{\sigma}_q^2)} \left[-\frac{1}{2} \log \left((2\pi)^{n_\mathbf{x}} |\operatorname{diag} \boldsymbol{\sigma}_\mathbf{x}^2| \right) - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_\mathbf{x})^\intercal (\operatorname{diag} \boldsymbol{\sigma}_\mathbf{x}^2)^{-1} (\mathbf{x} - \boldsymbol{\mu}_\mathbf{x}) \right]. \tag{35}$$

The first term in eqs. 34 and 35 is zero, since $\boldsymbol{\sigma}_{\mathbf{x}}^2$ does not depend on $\boldsymbol{\mu}_q$ or $\boldsymbol{\sigma}_q^2$. To take the gradients, we will again use the reparameterization trick to re-express $\mathbf{z} = \boldsymbol{\mu}_q + \boldsymbol{\sigma}_q \odot \boldsymbol{\epsilon}$. We now implicitly express $\boldsymbol{\mu}_{\mathbf{x}}$ as $\boldsymbol{\mu}_{\mathbf{x}}(\boldsymbol{\mu}_q, \boldsymbol{\sigma}_q^2, \theta)$. We can then write:

$$\mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{I})}\left[\nabla_{\boldsymbol{\mu}_q}\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}})^\intercal(\operatorname{diag}\boldsymbol{\sigma}_{\mathbf{x}}^2)^{-1}(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}})\right)\right] \tag{36}$$

and

$$\mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{I})}\left[\nabla_{\boldsymbol{\sigma}_q^2}\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}})^\intercal(\operatorname{diag}\boldsymbol{\sigma}_{\mathbf{x}}^2)^{-1}(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}})\right)\right]. \tag{37}$$

To simplify notation, we define the following term:

$$\boldsymbol{\xi} \equiv (\operatorname{diag}\boldsymbol{\sigma}_{\mathbf{x}}^2)^{-1/2}(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}}), \tag{38}$$

allowing us to rewrite eqs. 36 and 37 as

$$\mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{I})}\left[\nabla_{\boldsymbol{\mu}_q}\left(-\frac{1}{2}\boldsymbol{\xi}^\intercal\boldsymbol{\xi}\right)\right] = \mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{I})}\left[-\frac{\partial\boldsymbol{\xi}^\intercal}{\partial\boldsymbol{\mu}_q}\boldsymbol{\xi}\right] \tag{39}$$

and

$$\mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{I})}\left[\nabla_{\boldsymbol{\sigma}_q^2}\left(-\frac{1}{2}\boldsymbol{\xi}^\intercal\boldsymbol{\xi}\right)\right] = \mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{I})}\left[-\frac{\partial\boldsymbol{\xi}^\intercal}{\partial\boldsymbol{\sigma}_q^2}\boldsymbol{\xi}\right]. \tag{40}$$

We must now find $\frac{\partial\boldsymbol{\xi}}{\partial\boldsymbol{\mu}_q}$ and $\frac{\partial\boldsymbol{\xi}}{\partial\boldsymbol{\sigma}_q^2}$:

$$\frac{\partial\boldsymbol{\xi}}{\partial\boldsymbol{\mu}_q} = \frac{\partial}{\partial\boldsymbol{\mu}_q}\left((\operatorname{diag}\boldsymbol{\sigma}_{\mathbf{x}}^2)^{-1/2}(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}})\right) = -(\operatorname{diag}\boldsymbol{\sigma}_{\mathbf{x}}^2)^{-1/2}\frac{\partial\boldsymbol{\mu}_{\mathbf{x}}}{\partial\boldsymbol{\mu}_q} \tag{41}$$

and

$$\frac{\partial\boldsymbol{\xi}}{\partial\boldsymbol{\sigma}_q^2} = \frac{\partial}{\partial\boldsymbol{\sigma}_q^2}\left((\operatorname{diag}\boldsymbol{\sigma}_{\mathbf{x}}^2)^{-1/2}(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}})\right) = -(\operatorname{diag}\boldsymbol{\sigma}_{\mathbf{x}}^2)^{-1/2}\frac{\partial\boldsymbol{\mu}_{\mathbf{x}}}{\partial\boldsymbol{\sigma}_q^2}. \tag{42}$$

Plugging these expressions back into eqs. 39 and 40 gives

$$\mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{I})}\left[\frac{\partial\boldsymbol{\mu}_{\mathbf{x}}}{\partial\boldsymbol{\mu}_q}^\intercal((\operatorname{diag}\boldsymbol{\sigma}_{\mathbf{x}}^2)^{-1/2})^\intercal(\operatorname{diag}\boldsymbol{\sigma}_{\mathbf{x}}^2)^{-1/2}(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}})\right] = \mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{I})}\left[\frac{\partial\boldsymbol{\mu}_{\mathbf{x}}}{\partial\boldsymbol{\mu}_q}^\intercal\frac{\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}}}{\boldsymbol{\sigma}_{\mathbf{x}}^2}\right] \tag{43}$$

and

$$\mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{I})}\left[\frac{\partial\boldsymbol{\mu}_{\mathbf{x}}}{\partial\boldsymbol{\sigma}_q^2}^\intercal((\operatorname{diag}\boldsymbol{\sigma}_{\mathbf{x}}^2)^{-1/2})^\intercal(\operatorname{diag}\boldsymbol{\sigma}_{\mathbf{x}}^2)^{-1/2}(\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}})\right] = \mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{I})}\left[\frac{\partial\boldsymbol{\mu}_{\mathbf{x}}}{\partial\boldsymbol{\sigma}_q^2}^\intercal\frac{\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}}}{\boldsymbol{\sigma}_{\mathbf{x}}^2}\right]. \tag{44}$$

Despite having different distribution forms, Bernoulli and Gaussian output distributions result in approximate posterior gradients of a similar form: the Jacobian of the output model multiplied by a weighted error term.

A.5. Summary

Putting the gradient terms from $\log p_\theta(\mathbf{x}|\mathbf{z})$, $\log p_\theta(\mathbf{z})$, and $\log q(\mathbf{z}|\mathbf{x})$ together, we arrive at

Bernoulli Output Distribution:

$$\nabla_{\boldsymbol{\mu}_q}\mathcal{L} = \mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{I})}\left[\frac{\partial\boldsymbol{\mu}_{\mathbf{x}}}{\partial\boldsymbol{\mu}_q}^\intercal\frac{\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}}}{\boldsymbol{\mu}_{\mathbf{x}}\odot(\mathbf{1}-\boldsymbol{\mu}_{\mathbf{x}})} - \frac{\boldsymbol{\mu}_q+\boldsymbol{\sigma}_q\odot\boldsymbol{\epsilon}-\boldsymbol{\mu}_p}{\boldsymbol{\sigma}_p^2}\right] \tag{45}$$

$$\nabla_{\boldsymbol{\sigma}_q^2}\mathcal{L} = \mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{I})}\left[\frac{\partial\boldsymbol{\mu}_{\mathbf{x}}}{\partial\boldsymbol{\sigma}_q^2}^\intercal\frac{\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}}}{\boldsymbol{\mu}_{\mathbf{x}}\odot(\mathbf{1}-\boldsymbol{\mu}_{\mathbf{x}})} - \left(\operatorname{diag}\frac{\boldsymbol{\epsilon}}{2\boldsymbol{\sigma}_q}\right)^\intercal\frac{\boldsymbol{\mu}_q+\boldsymbol{\sigma}_q\odot\boldsymbol{\epsilon}-\boldsymbol{\mu}_p}{\boldsymbol{\sigma}_p^2}\right] - \frac{\mathbf{1}}{2\boldsymbol{\sigma}_q^2} \tag{46}$$

Gaussian Output Distribution:

$$\nabla_{\boldsymbol{\mu}_q}\mathcal{L} = \mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{I})}\left[\frac{\partial\boldsymbol{\mu}_{\mathbf{x}}}{\partial\boldsymbol{\mu}_q}^\intercal\frac{\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}}}{\boldsymbol{\sigma}_{\mathbf{x}}^2} - \frac{\boldsymbol{\mu}_q+\boldsymbol{\sigma}_q\odot\boldsymbol{\epsilon}-\boldsymbol{\mu}_p}{\boldsymbol{\sigma}_p^2}\right] \tag{47}$$

$$\nabla_{\boldsymbol{\sigma}_q^2}\mathcal{L} = \mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{I})}\left[\frac{\partial\boldsymbol{\mu}_{\mathbf{x}}}{\partial\boldsymbol{\sigma}_q^2}^\intercal\frac{\mathbf{x}-\boldsymbol{\mu}_{\mathbf{x}}}{\boldsymbol{\sigma}_{\mathbf{x}}^2} - \left(\operatorname{diag}\frac{\boldsymbol{\epsilon}}{2\boldsymbol{\sigma}_q}\right)^\intercal\frac{\boldsymbol{\mu}_q+\boldsymbol{\sigma}_q\odot\boldsymbol{\epsilon}-\boldsymbol{\mu}_p}{\boldsymbol{\sigma}_p^2}\right] - \frac{\mathbf{1}}{2\boldsymbol{\sigma}_q^2} \tag{48}$$

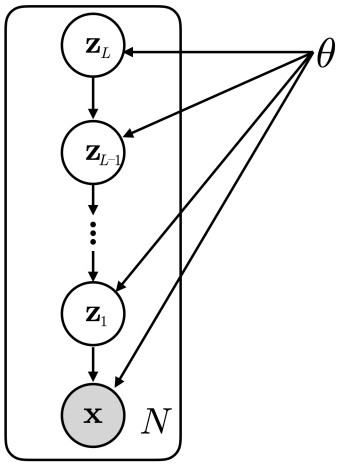


Figure 1. Plate notation for a hierarchical latent variable model consisting of L levels of latent variables. Variables at higher levels provide empirical priors on variables at lower levels. With data-dependent priors, the model has more flexibility.

A.6. Approximate Posterior Gradients in Hierarchical Latent Variable Models

Hierarchical latent variable models factorize the latent variables over multiple levels, $\mathbf{z} = \{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_L\}$. Latent variables at higher levels provide *empirical priors* on latent variables at lower levels. Here, we assume a first-order Markov graphical structure, as shown in Figure 1, though more general structures are possible. For an intermediate latent level, we use the notation $q(\mathbf{z}_\ell|\cdot) = \mathcal{N}(\mathbf{z}_\ell; \boldsymbol{\mu}_{\ell,q}, \operatorname{diag} \boldsymbol{\sigma}^2_{\ell,q})$ and $p(\mathbf{z}_\ell|\mathbf{z}_{\ell+1}) = \mathcal{N}(\mathbf{z}_\ell; \boldsymbol{\mu}_{\ell,p}, \operatorname{diag} \boldsymbol{\sigma}^2_{\ell,p})$ to denote the approximate posterior and prior respectively. Analogously to the case of a Gaussian output density in a one-level model, the approximate posterior gradients at an intermediate level ℓ are:

$$\nabla_{\boldsymbol{\mu}_{q,\ell}} \mathcal{L} = \mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{I})} \left[\frac{\partial \boldsymbol{\mu}_{\ell-1,p}}{\partial \boldsymbol{\mu}_{\ell,q}}^\intercal \frac{\boldsymbol{\mu}_{\ell-1,q} + \boldsymbol{\sigma}_{\ell-1,q} \odot \boldsymbol{\epsilon}_{\ell-1} - \boldsymbol{\mu}_{\ell-1,p}}{\boldsymbol{\sigma}^2_{\ell-1,p}} - \frac{\boldsymbol{\mu}_{\ell,q} + \boldsymbol{\sigma}_{\ell,q} \odot \boldsymbol{\epsilon}_\ell - \boldsymbol{\mu}_{\ell,p}}{\boldsymbol{\sigma}^2_{\ell,p}} \right], \tag{49}$$

$$\nabla_{\boldsymbol{\sigma}^2_q} \mathcal{L} = \mathbb{E}_{\mathcal{N}(\boldsymbol{\epsilon};\mathbf{0},\mathbf{I})} \left[\frac{\partial \boldsymbol{\mu}_{\ell-1,p}}{\partial \boldsymbol{\sigma}^2_{\ell,q}}^\intercal \frac{\boldsymbol{\mu}_{\ell-1,q} + \boldsymbol{\sigma}_{\ell-1,q} \odot \boldsymbol{\epsilon}_{\ell-1} - \boldsymbol{\mu}_{\ell-1,p}}{\boldsymbol{\sigma}^2_{\ell-1,p}} - \left(\operatorname{diag} \frac{\boldsymbol{\epsilon}_\ell}{2\boldsymbol{\sigma}_{\ell,q}} \right)^\intercal \frac{\boldsymbol{\mu}_{\ell,q} + \boldsymbol{\sigma}_{\ell,q} \odot \boldsymbol{\epsilon}_\ell - \boldsymbol{\mu}_{\ell,p}}{\boldsymbol{\sigma}^2_{\ell,p}} \right] - \frac{\mathbf{1}}{2\boldsymbol{\sigma}^2_{\ell,q}}. \tag{50}$$

The first terms inside each expectation are "bottom-up" gradients coming from reconstruction errors at the level below. The second terms inside the expectations are "top-down" gradients coming from priors generated by the level above. The last term in the variance gradient acts to reduce the entropy of the approximate posterior.

B. Implementing Iterative Inference Models

Here, we provide specific implementation details for these models. Code for reproducing the experiments will be released online.

B.1. Input Form

Approximate posterior gradients and errors experience distribution shift during inference and training. Using these terms as inputs to a neural network can slow down and prevent training. For experiments on MNIST, we found the $\log$ transformation method proposed by (Andrychowicz et al., 2016) to work reasonably well: replacing $\nabla_{\boldsymbol{\lambda}} \mathcal{L}$ with the concatenation of $[\alpha \log(|\nabla_{\boldsymbol{\lambda}} \mathcal{L}| + \epsilon), \operatorname{sign}(\nabla_{\boldsymbol{\lambda}} \mathcal{L})]$, where α is a scaling constant and ϵ is a small constant for numerical stability. We also encode the current estimates of $\boldsymbol{\mu}_q$ and $\log \boldsymbol{\sigma}^2_q$. For experiments on CIFAR-10, we instead used layer normalization (Ba et al., 2016) to normalize each input to the iterative inference model. This normalizes each input over the non-batch dimension.

Algorithm 1 Iterative Amortized Inference

Input: data $\mathbf{x}$, generative model $p_\theta(\mathbf{x}, \mathbf{z})$, inference model f
Initialize $t = 0$
Initialize $\nabla_\phi = 0$
Initialize $q(\mathbf{z}|\mathbf{x})$ with $\boldsymbol{\lambda}_0$
repeat
  Sample $\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})$
  Evaluate $\mathcal{L}_t = \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}_t; \theta)$
  Calculate $\nabla_{\boldsymbol{\lambda}} \mathcal{L}_t$ and $\nabla_\phi \mathcal{L}_t$
  Update $\boldsymbol{\lambda}_{t+1} = f_t(\nabla_{\boldsymbol{\lambda}} \mathcal{L}_t, \boldsymbol{\lambda}_t; \phi)$
  $t = t + 1$
  $\nabla_\phi = \nabla_\phi + \nabla_\phi \mathcal{L}_t$
until $\mathcal{L}$ converges
$\theta = \theta + \alpha_\theta \nabla_\theta \mathcal{L}$
$\phi = \phi + \alpha_\phi \nabla_\phi$

B.2. Output Form

For the output of these models, we use a gated updating scheme, where approximate posterior parameters are updated according to

$$\boldsymbol{\lambda}_{t+1} = \mathbf{g}_t \odot \boldsymbol{\lambda}_t + (\mathbf{1} - \mathbf{g}_t) \odot f_t(\nabla_{\boldsymbol{\lambda}} \mathcal{L}, \boldsymbol{\lambda}_t; \phi). \tag{51}$$

Here, $\odot$ represents element-wise multiplication and $\mathbf{g}_t = g_t(\nabla_{\boldsymbol{\lambda}} \mathcal{L}, \boldsymbol{\lambda}_t; \phi) \in [0, 1]$ is the gating function for $\boldsymbol{\lambda}$ at time t, which we combine with the iterative inference model f_t. We found that this yielded improved performance and stability over the additive updating scheme used in (Andrychowicz et al., 2016).

B.3. Training

To train iterative inference models for latent Gaussian models, we use stochastic estimates of $\nabla_\phi \mathcal{L}$ from the reparameterization trick. We accumulate these gradient estimates during inference, then update both ϕ and θ jointly. We train using a fixed number of inference iterations.

C. Experiment Details

Inference model and generative model parameters (ϕ and θ) were trained jointly using the adam optimizer (Kingma & Ba, 2014). The learning rate was set to 0.0002 for both sets of parameters and all other optimizer parameters were set to their default values. Learning rates were decayed exponentially by a factor of 0.999 each epoch. All models utilized exponential linear unit (ELU) activation functions (Clevert et al., 2015), although we found other non-linearities to work as well. Unless otherwise stated, all inference models were symmetric to their corresponding generative models. Iterative inference models for all experiments were implemented as feed-forward networks to make comparison with standard inference models easier.

C.1. Two-Dimensional Latent Gaussian Models

We trained models with 2 latent dimensions and a point estimate approximate posterior. That is, $q(\mathbf{z}|\mathbf{x}) = \delta(\mathbf{z} = \boldsymbol{\mu}_q)$ is a Dirac delta function at the point $\boldsymbol{\mu}_q = (\mu_1, \mu_2)$. We trained these models on binarized MNIST. The generative models consisted of a neural network with 2 hidden layers, each with 512 units. The output of the generative model was the mean of a Bernoulli distribution. The optimization surface of each model was evaluated on a grid of range [-5, 5] in increments of 0.05 for each latent variable. The iterative inference model shown in Figure 3 encodes $\mathbf{x}$, $\boldsymbol{\varepsilon}_\mathbf{x}$, and $\boldsymbol{\varepsilon}_\mathbf{z}$.

C.2. $\mathcal{L}$ During Inference

We trained one-level models on MNIST using iterative inference models that encode gradients ($\nabla_{\boldsymbol{\lambda}} \mathcal{L}$) for 16 iterations. We compared against stochastic gradient descent (SGD), SGD with momentum, RMSProp, and Adam, using learning rates in $\{0.5, 0.4, 0.3, 0.2, 0.1, 0.01, 0.001\}$ and taking the best result. In addition to performance over iterations, we also compared the optimization techniques on the basis of wall clock time. Despite requiring more time per inference iteration, we observed

that the iterative inference model still outperformed the conventional optimization techniques.

C.3. Reconstructions Over Inference Iterations

We trained iterative inference models on MNIST, Omniglot, and SVHN by encoding approximate posterior gradients ($\nabla_{\boldsymbol{\lambda}}\mathcal{L}$) for 16 iterations. For CIFAR-10, we trained an iterative inference model by encoding errors for 10 inference iterations. For MNIST and Omniglot, we used a generative model architecture with 2 hidden layers, each with 512 units, a latent space of size 64, and a symmetric iterative inference model. For SVHN and CIFAR-10, we used 3 hidden layers in the iterative inference and 1 in the generative model, with 2,048 units at each hidden layer and a latent space of size 1,024.

C.4. Gradient Magnitudes

While training iterative inference models, we recorded approximate posterior gradient magnitudes at each inference iteration. We observed that, on average, the magnitudes decreased during inference optimization. This decrease was more prevalent for the approximate posterior mean gradients. For Figure 6, we trained an iterative inference model on RCV1 by encoding gradients ($\nabla_{\boldsymbol{\lambda}}\mathcal{L}$) for 16 inference iterations. The generative model contained a latent variable of size 512 and 2 fully-connected layers of 512 units each. The inference model was symmetric.

C.5. Additional Inference Iterations

We used an architecture of 2 hidden layers, each with 512 units, for the output model and inference models. The latent variable contained 64 dimensions. We trained all models for 1,500 epochs. We were unable to run multiple trials for each experimental set-up, but on a subset of runs for standard and iterative inference models, we observed that final performance had a standard deviation less than 0.1 nats, below the difference in performance between models trained with different numbers of inference iterations.

C.6. Additional Latent Samples

We used an architecture of 2 hidden layers, each with 512 units, for the output model and inference models. The latent variable contained 64 dimensions. Each model was trained by drawing the corresponding number of samples from the approximate posterior distribution to obtain ELBO estimates and gradients. Iterative inference models were trained by encoding the data ($\mathbf{x}$) and the approximate posterior gradients ($\nabla_{\boldsymbol{\lambda}}\mathcal{L}$) for 5 inference iterations. All models were trained for 1,500 epochs.

C.7. Comparison with Standard Inference Models

C.7.1. MNIST

For MNIST, one-level models consisted of a latent variable of size 64, and the inference and generative networks both consisted of 2 hidden layers, each with 512 units. Hierarchical models consisted of 2 levels with latent variables of size 64 and 32 in hierarchically ascending order. At each level, the inference and generative networks consisted of 2 hidden layers, with 512 units at the first level and 256 units at the second level. At the first level of latent variables, we also used a set of deterministic units, also of size 64, in both the inference and generative networks. Hierarchical models included batch normalization layers at each hidden layer of the inference and generative networks; we found this beneficial for training both standard and iterative inference models. Both encoder and decoder networks in the hierarchical model utilized highway skip connections at each layer at both levels. Iterative models were trained by encoding data and errors for 5 inference iterations.

C.7.2. CIFAR-10

For CIFAR-10, one-level models consisted of a latent variable of size 1,024, an encoder network with 3 hidden layers of 2,048 units, and a decoder network with 1 hidden layer with 2,048 units. We found this set-up performed better than a symmetric encoder and decoder for both standard and iterative inference models. Hierarchical models were the same as the one-level model, adding another latent variable of size 512, with another 3 layer encoder of with 1,024 units and a 1 layer decoder with 1,024 units. Both encoder and decoder networks in the hierarchical model utilized highway skip connections at each layer at both levels. Models were all trained for 150 epochs. We annealed the KL-divergence term during the first 50 epochs when training hierarchical models. Iterative inference models were trained by encoding the data and gradients for 5

inference iterations.

C.7.3. RCV1

We followed the same processing procedure as (Krishnan et al., 2017), encoding data using normalized TF-IDF features. For encoder and decoder, we use 2-layer networks, each with 2,048 units and ELU non-linearities. We use a latent variable of size 1,024. The iterative inference model was trained by encoding gradients for 10 steps. Both models were trained using 5 approximate posterior samples at each iteration. We evaluate the models by reporting perplexity on the test set (Table 2). Perplexity, P, is defined as

$$P \equiv \exp(-\frac{1}{N}\sum_i \frac{1}{N_i}\log p(\mathbf{x}^{(i)})), \tag{52}$$

where N is the number of examples and N_i is the total number of word counts in example i. We evaluate perplexity by estimating each $\log p(\mathbf{x}^{(i)})$ with 5,000 importance weighted samples. We also report an upper bound on perplexity using $\mathcal{L}$.

References

Andrychowicz, Marcin, Denil, Misha, Gomez, Sergio, Hoffman, Matthew W, Pfau, David, Schaul, Tom, and de Freitas, Nando. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pp. 3981–3989, 2016.

Ba, Jimmy Lei, Kiros, Jamie Ryan, and Hinton, Geoffrey E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

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

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Krishnan, Rahul G, Liang, Dawen, and Hoffman, Matthew. On the challenges of learning with inference networks on sparse, high-dimensional data. *arXiv preprint arXiv:1710.06085*, 2017.

Rezende, Danilo Jimenez, Mohamed, Shakir, and Wierstra, Daan. Stochastic backpropagation and approximate inference in deep generative models. *Proceedings of the 31st International Conference on Machine Learning*, pp. 1278–1286, 2014.

Figure 2. Reconstructions over inference iterations (left to right) for examples from (top to bottom) MNIST, Omniglot, SVHN, and CIFAR-10. Corresponding data examples are shown on the right of each panel.