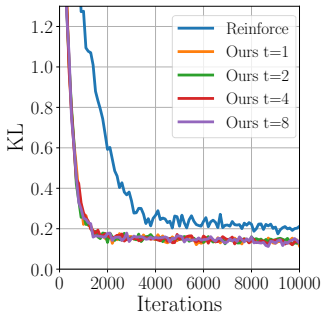

Figure 5: Optimization of KL divergence from an RBM to a mixture of Bernoulli distributions using the proposed gradient estimator. Increasing the number of MCMC updates t does not improve the gradient estimation.

A. Toy Example: RBM Posterior

In this section, we examine the dependence of the gradient estimator the number of MCMC updates t on a toy example. Here, a 4+4 bit RBM is learned to minimize the KL divergence to a mixture of Bernoulli distributions:

$$\mathcal{L} = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z})}\left[\log \frac{q_{\boldsymbol{\phi}}(\boldsymbol{z})}{p_{\text{mix}}(\boldsymbol{z})}\right] \tag{17}$$

$$p_{\text{mix}}(\boldsymbol{z}) = \sum_{i=1}^{C} \alpha_i e^{\sum_a (\nu_i^a z^a - \text{softplus}(\nu_i^a))}. \tag{18}$$

We choose $C = 3$ mixture components with weights α_i chosen uniformly and Bernoulli components having variance 0.09. The training of parameters $\boldsymbol{\phi}$ is done with the Adam optimizer (Kingma & Ba, 2014), using learning rate 0.003 for 10000 iterations. We show the results for several values of t in Fig 5. In all cases the optimization locates the global minimizer. We note that the effectiveness of optimization does not depend on t.

B. Examining the Shared Context

For the directed posterior baselines, we initially experimented with the structure introduced in DVAE#[6] in which each factor $q(\boldsymbol{z}_i|\boldsymbol{x}, \boldsymbol{z}_{<i})$ is represented using two tanh nonlinearities as shown in Fig. 3(b). However, initial experiments indicated that when the number of latent variables is held constant, increasing the number of subgroups, L, does not improve the generative performance.

In undirected posteriors, the parameters of the posterior for each training example are predicted using a single shared context feature ($\boldsymbol{c}(\boldsymbol{x})$ in Fig. 3(a)). However, in DVAE#, parallel neural networks generate the parameters of the posterior for each level of the hierarchy. Inspired by this observation, we define a new structure for the directed posteriors by introducing a shared (200 dimensional) context feature that is computed using two tanh nonlinearities. This shared feature $\boldsymbol{c}(\boldsymbol{x})$ is fed to the subsequent conditional each represented with a linear layer, i.e., $q(\boldsymbol{z}|\boldsymbol{x}) = \prod_{i=1}^{L} q(\boldsymbol{z}_i|\boldsymbol{c}(\boldsymbol{x}), \boldsymbol{z}_{<i})$ (see Fig. 3(c)).

In Table 5, we compare the original structure used in DVAE# to the structure with a shared context. In almost all cases the shared context feature improves the performance of the original structure. Moreover, increasing the number of hierarchical layers often improves the performance of the new structure.

Note that in Table 5, we report the average negative log-likelihood measured on the test set for 5 runs. In the $L = 1$ case, both structures are identical and they achieve statistically similar performance.

C. Annealing the Importance Weighted Bound

KL annealing (Sønderby et al., 2016) is used to prevent latent variables from turning off in VAE training. A scalar λ is introduced which weights KL contribution to the ELBO and λ is annealed from zero to one during training:

$$\mathcal{L} = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z})] - \lambda \text{KL}\big(q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\big). \tag{19}$$

Since the KL contribution is small early in training, the VAE model initially optimizes the reconstruction term which prevents the approximate posterior from matching to the prior.

Here, we apply the same approach to the importance weighted bound by rewriting the IW bound by:

$$\mathcal{L}_{IW} = \mathbb{E}_{\boldsymbol{z}_{1:K}}\left[\log\left(\frac{1}{K}\sum_{i=1}^{K}\frac{p(\boldsymbol{z}_i)^\lambda p(\boldsymbol{x}|\boldsymbol{z}_i)}{q(\boldsymbol{z}_i|\boldsymbol{x})^\lambda}\right)\right], \tag{20}$$

where $\boldsymbol{z}_{1:K} \sim \prod_{i=1}^{K} q(\boldsymbol{z}_i|\boldsymbol{x})$. Similar to the KL annealing idea, we anneal the scalar λ during training from zero to one. When λ is small, the IW bound emphasizes reconstruction of the data which inhibits the latent variables from turning off. When $K = 1$, Eq. (20) reduces to Eq. (19).

The gradient of Eq. (20) with respect to $\boldsymbol{\phi}$, the parameters of $q(\boldsymbol{z}|\boldsymbol{x})$ and $\boldsymbol{\theta}$, the parameters of the generative model is:

$$\partial_{\boldsymbol{\phi},\boldsymbol{\theta}}\mathcal{L}_{IW} = \mathbb{E}_{\boldsymbol{\epsilon}_{1:K}}\left[\sum_{i=1}^{K}\frac{w_i}{\sum_j w_j}\partial_{\boldsymbol{\phi},\boldsymbol{\theta}}\log w_i\right], \tag{21}$$

where $w_i = p_{\boldsymbol{\theta}}(\boldsymbol{z}_i)^\lambda p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}_i)/q_{\boldsymbol{\phi}}(\boldsymbol{z}_i|\boldsymbol{x})^\lambda$. The gradient $\partial_{\boldsymbol{\phi}}\log w_i$ is evaluated as:

$$\partial_{\boldsymbol{\phi}}\log w_i = -\lambda\partial_{\boldsymbol{\phi}}\log q_{\boldsymbol{\phi}}(\boldsymbol{z}_i|\boldsymbol{x}) + (\partial_{\boldsymbol{z}_i}\log w_i)(\partial_{\boldsymbol{\phi}}\boldsymbol{z}_i). \tag{22}$$

The second term in the right hand side of Eq. (22) is the pathwise gradient that can be computed easily using our biased

[6] These models are also used in DVAE++ and GumBolt.

Table 5: The performance of structure with a shared context feature (Fig. 3(c)) is compared against the original structure used in DVAE# (Fig. 3(b)). The shared context feature often improves the performance.

			Prior RBM Size: 100+100						Prior RBM Size: 200+200					
			DVAE#		GumBolt		PWL		DVAE#		GumBolt		PWL	
context			✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓
MNIST	#layers	1	**84.95**	**84.97**	**84.65**	**84.66**	**84.57**	**84.62**	**83.26**	**83.21**	**83.21**	**83.19**	**83.23**	**83.22**
		2	**84.81**	84.96	**84.75**	**84.71**	84.70	**84.50**	83.26	**83.13**	83.24	**83.04**	83.38	**82.99**
		4	**84.61**	**84.58**	84.90	**84.39**	85.00	**84.07**	83.13	**82.93**	83.69	**83.14**	83.85	**82.90**
OMNI.	#layers	1	**101.69**	**101.64**	**101.41**	**101.41**	**101.21**	**101.24**	**99.53**	**99.51**	**99.39**	**99.39**	**99.37**	**99.32**
		2	101.84	**101.75**	102.45	**101.39**	101.64	**101.14**	99.66	**99.40**	100.52	**99.12**	100.07	**99.10**
		4	101.93	**101.74**	102.76	**102.04**	101.97	**101.14**	99.63	**99.47**	100.99	**99.97**	100.45	**99.30**

gradient estimator. This term does not have any dependence on the partition function $Z_{\boldsymbol{\phi}}$ since

$$\partial_{\boldsymbol{z}_i} \log w_i = \partial_{\boldsymbol{z}_i} \left(\log p_{\boldsymbol{\theta}}(\boldsymbol{z}_i)^{\lambda} p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}_i) - \lambda \log q_{\boldsymbol{\phi}}(\boldsymbol{z}_i|\boldsymbol{x}) \right)$$

and $\partial_{\boldsymbol{z}_i} \log q_{\boldsymbol{\phi}}(\boldsymbol{z}_i|\boldsymbol{x}) = -\partial_{\boldsymbol{z}_i} E_{\boldsymbol{\phi}}(\boldsymbol{z}_i)$ depends only on the energy function.

However, the first term in the right hand side of Eq. (22) does contain $\partial_{\boldsymbol{\phi}} Z_{\boldsymbol{\phi}}$ which can be high-variance. We apply the doubly reparameterized method proposed by (Tucker et al., 2018) to remove this term. Following a derivation similar to (Tucker et al., 2018), it is easy to show that:

$$\partial_{\boldsymbol{\phi}} \mathcal{L}_{IW} = \mathbb{E}_{\boldsymbol{\epsilon}_{1:K}} \left[\sum_{i=1}^{K} \left(\lambda \tilde{w_i}^2 + (1-\lambda)\tilde{w_i} \right) (\partial_{\boldsymbol{z}_i} \log w_i)(\partial_{\boldsymbol{\phi}} \boldsymbol{z}_i) \right]$$

where $\tilde{w_i} = w_i / \sum_j w_j$.

D. The Effect of Gibbs Chain Lengths s and t

In Table. 6, we study the effect of changing the number of Gibbs sampling steps on the final performance of DVAE##. We vary the number of discrete Gibbs steps s and the number of relaxed Gibbs steps t in training the DVAE## model on MNIST with the variational bound for an RBM size of $100 + 100$. We observe that the final performance does not change for $s \geq 5$, however, increasing t decreases the final log likelihood. This degradation with increasing t likely arises as sampling from the relaxed chain diverges from the exact discrete chain resulting in increasingly biased gradient estimates.

E. MCMC Sampling from True Posterior

A natural question to ask is how our method compares against MCMC approaches, which approximately sample from the true posterior, instead of sampling from an amortized undirected approximate posterior. In this section, we provide a small experiment to compare these two approaches.

Training a generative model using MCMC is similar to training a VAE, with the main difference that, instead of using an encoder to sample from an approximate posterior, we use MCMC to approximately sample from the true posterior. Given the samples from true posterior, we use the following objective to train the parameters of the generative model ($\boldsymbol{\theta}$):

Table 6: The performance of DVAE## trained on MNIST with the variational bound is compared for varying numbers of discrete Gibbs steps s and relaxed Gibbs steps t. The mean±standard deviation of the negative log-likelihood over 4 runs is reported. Final performance does not change for $s \geq 5$, but increasing t degrades performance.

RBM Size: 100+100			
	$s = 1$	$s = 5$	$s = 10$
$t = 1$	**84.11**$\pm$**0.04**	**84.06**$\pm$**0.04**	**84.09**$\pm$**0.03**
$t = 2$	84.20$\pm$0.05	**84.12**$\pm$**0.04**	**84.12**$\pm$**0.06**
$t = 4$	84.46$\pm$0.04	84.32$\pm$0.05	84.34$\pm$0.02

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})} \left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z}) \right]. \tag{23}$$

There are several pitfalls in using MCMC to sample from the true posterior in generative models. First, we mainly consider binary latent variables in this paper. Thus, gradient-based MCMC techniques such as Hamiltonian Monte Carlo are not available. To overcome this, we sample one bit at a time from the true posterior, and in each sweep, we iterate over all the bits. Second, MCMC may take many iterations to reach equilibrium. To address this, we use persistent sampling for each data point in the training set. We start the MCMC chains from the state of the chains from the previous epoch, and we use 10 sweeps before each parameter update. Finally, in the absence of encoders. computing the log-likelihood of the MCMC approaches on the test dataset is not trivial. The best known approach is AIS (Wu et al., 2017) which is challenging to implement. To address this, we limit the latent space to only 16 bits, and we compute log-likelihood exactly by enumerating all the states.

In Table 7, we report the negative log-likelihood on the training and test datasets and training time per iteration. We make two observations. First, the training log-likelihood for both RBM posteriors and the true posteriors are not

Table 7: The performance of DVAE## trained on MNIST with the variational bound is compared against using MCMC to sample from the true posterior. Here, we limit the latent space to 16 bits. The negative log-likelihood on the training and test dataset and training time per iteration in milliseconds are reported.

Prior Size: 16 bits		
	DVAE## RBM approx. posterior	MCMC True posterior
train neg. LL	**105.7**$\pm$**0.3**	**106.7**$\pm$**1.3**
test neg. LL	**110.9**$\pm$**0.3**	113.4$\pm$1.5
Time (ms)	**22**	103

significantly different. However, the test log-likelihood is inferior to the MCMC baseline. When we examine training the MCMC baseline, we notice that the training dynamic for the MCMC method is very different than DVAE##. MCMC training is sensitive to the initialization and is prone to overfitting. We hypothesize that MCMC methods require their own special treatment (beyond the scope of this paper), making a direct comparison against VAE models unfair.

Second, we note that training using MCMC sampling from the posterior is $\sim 5\times$ slower than training using the RBM approximate posteriors. This is mostly because, in the RBM approximate posteriors, each conditional in the transition kernel is a linear function. However, in the MCMC sampling from the true posterior, we require computing the likelihood $p(\boldsymbol{x}|\boldsymbol{z})$ in each sampling step, which is computationally expensive as $p(\boldsymbol{x}|\boldsymbol{z})$ is implemented by a neural network. We expect MCMC approaches to be even slower as the generative model becomes deeper and more complex while the computational complexity of sampling from the RBM approximate posteriors does not depend on the generative model.