## A. Overlapping Transformation with the Mixture of Exponential Distributions

The CDF for each conditional distribution is given by

$$F_{r(\zeta|z=0)}(\zeta) = \frac{1-e^{-\beta\zeta}}{1-e^{-\beta}}$$
$$F_{r(\zeta|z=1)}(\zeta) = \frac{e^{\beta(\zeta-1)}-e^{-\beta}}{1-e^{-\beta}}.$$

To simplify notation, the mean of the Bernoulli distribution $q(z=1|x)$ is denoted by $q$. The CDF for the mixture $q(\zeta|x)=\sum_z r(\zeta|z)q(z|x)$ is

$$F_{q(\zeta|x)}(\zeta) = \frac{1-q}{1-e^{-\beta}}\left[1-e^{-\beta\zeta}\right] + \frac{q}{1-e^{-\beta}}\left[e^{\beta(\zeta-1)}-e^{-\beta}\right].$$

Defining $m \equiv e^{-\beta\zeta}$ and $d \equiv e^{-\beta}$, the inverse CDF is found by solving $F_{q(\zeta)}(\zeta)-\rho=0$ which gives rise to the quadratic equation

$$m^2 + \underbrace{\left[-1+\frac{\rho+d(q-\rho)}{1-q}\right]}_{b} m \underbrace{-\frac{qd}{1-q}}_{c} = 0,$$

which has solutions $m=(-b\pm\sqrt{b^2-4c})/2$. Since $-4c \geq 0$ (as $q\geq 0$ and $d>0$), there are two real solutions. Further, $m=(-b+\sqrt{b^2-4c})/2$ is the valid solution since $m$ must be positive (recall $m=e^{-\beta\zeta}$) and $\sqrt{b^2-4c} \geq |b|$. Lastly, the inverse CDF is obtained using $\zeta=-\log m/\beta$. The inverse CDF is a differentiable mapping from uniform samples $\rho\sim U(0,1)$ to samples from $q(\zeta|x)$.

## B. Overlapping Transformation with the Mixture of Logistic Distributions

The Dirac $\delta$ distribution can be approximated by a normal distribution whose variance approaches to zero. We use this observation to define a smoothing transformation where each $r(\zeta|z)$ is modeled with a Normal distribution (with $\zeta\in\mathbb{R}$):

$$r(\zeta|z=0) = \mathcal{N}(\zeta|0,\sigma^2)$$
$$r(\zeta|z=1) = \mathcal{N}(\zeta|1,\sigma^2).$$

The resulting mixture $q(\zeta|x)=\sum_z r(\zeta|z)q(z|x)$ converges to a Bernoulli distribution as $\sigma$ goes to 0, but its CDF (which is a mixture of error functions) cannot be inverted in closed form.

To derive a Normal-like distribution with an invertible CDF and support $\zeta\in\mathbb{R}$, we define a smoothing transformation using the logistic distribution

$$r(\zeta|z=0) = \mathcal{L}(\zeta|\mu_0,s)$$
$$r(\zeta|z=1) = \mathcal{L}(\zeta|\mu_1,s)$$

where

$$\mathcal{L}(\zeta|\mu,s) = \frac{e^{-\frac{\zeta-\mu}{s}}}{s(1+e^{-\frac{\zeta-\mu}{s}})^2}.$$

For the mixture distribution[1]

$$q(\zeta|x) = (1-q)\mathcal{L}(\zeta,\mu_0,s) + q\mathcal{L}(\zeta,\mu_1,s),$$

the inverse CDF is derived by solving

$$F_{q(\zeta)}(\zeta) = \frac{1-q}{1+e^{-\frac{\zeta-\mu_0}{s}}} + \frac{q}{1+e^{-\frac{\zeta-\mu_1}{s}}} = \rho$$
$$\frac{(1-q)(1+e^{-\frac{\zeta-\mu_1}{s}})+q(1+e^{-\frac{\zeta-\mu_0}{s}})}{(1+e^{-\frac{\zeta-\mu_1}{s}})(1+e^{-\frac{\zeta-\mu_0}{s}})} = \rho.$$

Defining $m\equiv e^{-\zeta/s}$, $d_0\equiv e^{\mu_0/s}$, and $d_1=e^{\mu_1/s}$ yields a quadratic in $m$

$$\underbrace{\rho d_0 d_1}_{a} m^2 + \underbrace{[\rho(d_0+d_1)-d_0 q - d_1(1-q)]}_{b} m + \underbrace{\rho-1}_{c} = 0$$

which has the valid solution

$$m^* = \frac{-b+\sqrt{b^2-4ac}}{2a}.$$

This gives the inverse CDF as $F^{-1}_{q(\zeta|x)}(\rho) = -s\log m^*$. When $s$ is very small and $\mu_1 > \mu_0$, $d_1$ is suceptible to overflow. A numerically stable solution can be obtained by applying the change of variable $m'=\sqrt{d_0 d_1}m$.

## C. Visualization of Inverse CDFs

To provide insight into the differences between overlapping transformations, Concrete (Maddison et al., 2016; Jang et al., 2016), and spike-and-exponential (Rolfe, 2016) smoothing, Fig.5 visualizes the inverse CDFs at different temperatures. In cases where Concrete and spike-and-exponential have small gradients with respect to $q(z=1|x)$ (thereby slowing learning), overlapping transformations provide a larger gradient signal (for faster learning).

## D. Joint versus Marginal ELBOs

We have presented two alternative ELBO bounds, one based on a joint inference model $q(\boldsymbol{\zeta}_1,\boldsymbol{\zeta}_2,\boldsymbol{z}_1,\boldsymbol{z}_2|\boldsymbol{x})$ and the other based on $q(\boldsymbol{\zeta}_1,\boldsymbol{\zeta}_2|\boldsymbol{x})$ obtained by marginalizing the discrete variables. Here, we show that variational bound obtained with the marginal model is tighter.

For simplicity we consider a model with only one latent variable. Figs. 6(a) and (b) visualize the generative and inference models. Figs. 6(c) and (d) show the joint models, and Figs. 6(e) and (f) show the marginal models. The

[1] $q$ is again shorthand for $q(z=1|x)$.

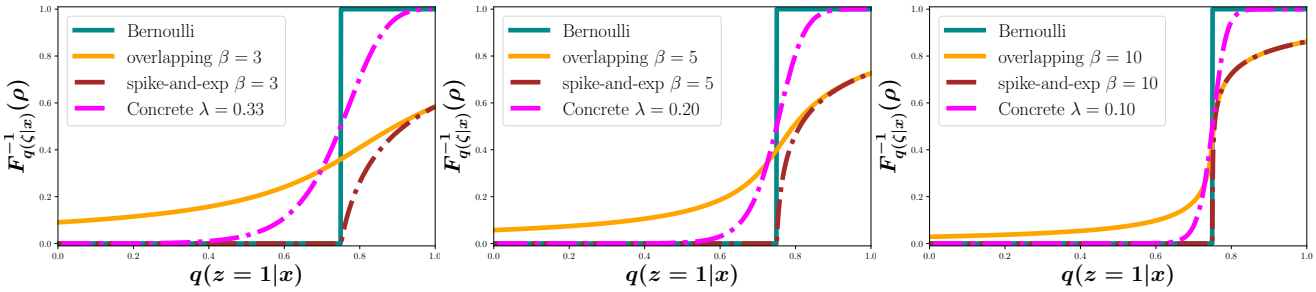


Figure 5: Visualization of inverse CDF as a function of $q(z = 1|x)$ at $\rho = 0.25$ for different smoothing transformations with three different temperatures ($\lambda$) and inverse temperatures ($\beta$). We have selected temperature values that are often used in practice.

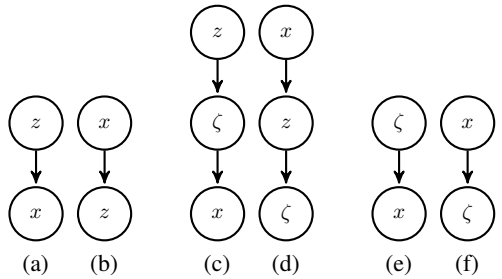


Figure 6: (a) A generative model with binary latent variable $z$. (b) The corresponding inference model. In (c) and (d), the continuous $\zeta$ is introduced and dependency on $z$ is transferred to dependency on $\zeta$. In (e) and (f) the binary latent variable $z$ is marginalized out.

respective variational bounds are

$$L_1(\boldsymbol{x}) = \log p(x) - \mathbb{E}_{q(\boldsymbol{z},\boldsymbol{\zeta}|\boldsymbol{x})}\left[\frac{q(\boldsymbol{z},\boldsymbol{\zeta}|\boldsymbol{x})}{p(\boldsymbol{z},\boldsymbol{\zeta}|\boldsymbol{x})}\right]$$

and

$$L_2(\boldsymbol{x}) = \log p(\boldsymbol{x}) - \mathbb{E}_{q(\boldsymbol{\zeta}|\boldsymbol{x})}\left[\log \frac{q(\boldsymbol{\zeta}|\boldsymbol{x})}{p(\boldsymbol{\zeta}|\boldsymbol{x})}\right].$$

Subtracting we find

$$\begin{aligned} L_2(\boldsymbol{x}) - L_1(\boldsymbol{x}) &= \mathbb{E}_{q(\boldsymbol{\zeta},\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{q(\boldsymbol{\zeta},\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{\zeta},\boldsymbol{z}|\boldsymbol{x})}\right] - \\ &\quad \mathbb{E}_{q(\boldsymbol{\zeta}|\boldsymbol{x})}\left[\log \frac{q(\boldsymbol{\zeta}|\boldsymbol{x})}{p(\boldsymbol{\zeta}|\boldsymbol{x})}\right] \\ &= \mathbb{E}_{q(\boldsymbol{\zeta},\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{q(\boldsymbol{\zeta},\boldsymbol{z}|\boldsymbol{x})}{p(\boldsymbol{\zeta},\boldsymbol{z}|\boldsymbol{x})} - \log \frac{q(\boldsymbol{\zeta}|\boldsymbol{x})}{p(\boldsymbol{\zeta}|\boldsymbol{x})}\right] \\ &= \mathbb{E}_{q(\boldsymbol{\zeta},\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{q(\boldsymbol{z}|\boldsymbol{\zeta},\boldsymbol{x})}{p(\boldsymbol{z}|\boldsymbol{\zeta},\boldsymbol{x})}\right] \\ &= \mathbb{E}_{q(\boldsymbol{\zeta}|\boldsymbol{x})}\big[\mathrm{KL}\big(q(\boldsymbol{z}|\boldsymbol{\zeta},\boldsymbol{x})\|p(\boldsymbol{z}|\boldsymbol{\zeta},\boldsymbol{x})\big)\big], \end{aligned}$$

which is clearly positive since $\mathrm{KL}(\cdot\|\cdot) \geq 0$. Thus, $L_2(\boldsymbol{x})$, the marginal ELBO, provides a tighter bound.

## E. Adding the Gradient of $\log Z$ to the Objective Function

For training the DVAE++ model with an RBM prior, the gradient of $\log Z$ is needed for each parameter update. Since $\log Z$ only depends on the prior parameters $\theta = \{\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{W}\}$, its gradient is

$$\begin{aligned} \frac{\partial \log Z}{\partial \theta} &= \frac{\partial}{\partial \theta} \log \sum_{\boldsymbol{z}_1,\boldsymbol{z}_2} e^{-E_\theta(\boldsymbol{z}_1,\boldsymbol{z}_2)} \\ &= -\frac{\sum_{\boldsymbol{z}_1,\boldsymbol{z}_2} e^{-E_\theta(\boldsymbol{z}_1,\boldsymbol{z}_2)} \frac{\partial E_\theta(\boldsymbol{z}_1,\boldsymbol{z}_2)}{\partial \theta}}{\sum_{\boldsymbol{z}'_1,\boldsymbol{z}'_2} e^{-E_\theta(\boldsymbol{z}'_1,\boldsymbol{z}'_2)}} \\ &= -\sum_{\boldsymbol{z}_1,\boldsymbol{z}_2} p_\theta(\boldsymbol{z}_1,\boldsymbol{z}_2) \frac{\partial E_\theta(\boldsymbol{z}_1,\boldsymbol{z}_2)}{\partial \theta} \\ &= -\mathbb{E}_{p_\theta(\boldsymbol{z}_1,\boldsymbol{z}_2)}\left[\frac{\partial E_\theta(\boldsymbol{z}_1,\boldsymbol{z}_2)}{\partial \theta}\right]. \end{aligned}$$

This expectation is estimated using Monte Carlo samples from the RBM. We maintain persistence chains and run block Gibbs updates for a fixed number of iterations (40) to update the samples after each parameter update. This approach is known as persistent contrastive divergence (PCD) (Younes, 1989; Tieleman, 2008).

Instead of manually coding the gradient of the negative energy function for each sample and modifying the gradient of whole objective function, we compute the negative average energy on $L$ samples (indexed by $l$) generated from PCD chains ($-\sum_{l=1}^{L} E_\theta(\boldsymbol{z}_1^{(l)}, \boldsymbol{z}_2^{(l)})/L$ where $\boldsymbol{z}_1^{(l)}, \boldsymbol{z}_2^{(l)} \sim p_\theta(\boldsymbol{z}_1, \boldsymbol{z}_2)$). This gives a scalar tensor whose gradient is the sample-based approximation to $\partial Z/\partial\theta$. By adding this tensor to the objective function, an automatic differentiation (AD) library backpropagates through this tensor and computes the appropriate gradient estimate. Note that

an AD library cannot backpropagate the gradients through the samples generated from the PCD computation graph because of the discrete nature of the process. However, one can use stop gradient commands on the samples to prevent unnecessary AD operations.

## F. Implementation Details for the RBM Prior Experiments

In this section, we summarize the implementation details for the experiments reported in Sec.7.3 on the RBM prior VAE. During training, the KL term is annealed linearly from 0 to 1 in 300K iterations. The learning rate starts at $3 \cdot 10^{-3}$ and is multiplied by 0.3 at iterations 600K, 750K, and 950K. Batch normalization is used for the nonlinear deterministic hidden layers. Training runs for 1M iterations with batch size of 100 and the performance is measured using 4000-sample importance weight estimation of log-likelihood. $\beta$ is fixed during training for all methods and is cross-validated from the set $\{5, 6, 8, 10\}$. Nonlinear models use two hidden layers of size 200 with tanh activations. There are 200 stochastic latent variables in each layer.

## G. Implementation Details for the DVAE++ Experiments

The network architecture visualized in Fig. 7 is divided into three parts: i) the inference network or encoder that represents $q(\boldsymbol{z}, \boldsymbol{\zeta}, \boldsymbol{h}|\boldsymbol{x})$, ii) the prior network that models $p(\boldsymbol{z}, \boldsymbol{\zeta}, \boldsymbol{h})$, and iii) the decoder network that implements $p(\boldsymbol{x}|\boldsymbol{\zeta}, \boldsymbol{h})$. Our architecture consists of many modules that may differ for different datasets. The details here describe the network architecture used for CIFAR10 data. Table 4 lists the specifics of the networks for all the datasets.

For the encoder side, "down 1" denotes a series of downsampling residual blocks that extract convolutional features from the input image. The output of this network is a feature of size $8 \times 8 \times 256$ (expressed by height $\times$ width $\times$ depth). The module "down 2" denotes another residual network that takes the output of "down 1" and progressively reduces the spatial dimensions of the feature maps until it reaches to a feature map of size $1 \times 1 \times 1024$. This feature map is flattened and is iteratively fed to a series of fully connected networks that model $q(\boldsymbol{z}_i|\boldsymbol{x}, \boldsymbol{\zeta}_{<i})$. In addition to the feature map, each network accepts the concatenation of samples drawn from the smoothed variables. These networks have identical architecture with no parameter sharing and all model factorial Bernoulli distributions in their output.

The concatenation of samples from smoothed variables $\boldsymbol{\zeta}_i$ is fed to "upsample", a residual network that upsamples its input to $8 \times 8 \times \times 32$ dimensions using transposed convolutions. These features are concatenated with the feature map generated by "down 1". The concatenated feature is then iteratively fed to a series of residual network that defines $q(\boldsymbol{h}_j|\boldsymbol{x}, \boldsymbol{\zeta}, \boldsymbol{h}_{<j})$. Each network accepts the concatenation samples from local variables in the previous group ($\boldsymbol{h}_{<j}$) in addition to the concatenated feature. When the local latent variables are modeled by normal distributions, these networks return mean and logarithm of the standard deviation of the elements in $\boldsymbol{h}_j$ similar to the original VAE (Kingma & Welling, 2014).

In the prior network, the same "upsample" network defined above is used to scale-up the global latent variables to the intermediate scale ($8 \times 8 \times 32$). Then, the output of this network is fed to a set of residual networks that defines $p(\boldsymbol{h}_j|\boldsymbol{\zeta}, \boldsymbol{h}_{<j})$ one at a time in the same scale. Similar to the encoder, these network accept the concatenation of all the local latent variables and they generate the parameters of the same type of distribution.

In the decoder network, the "context" residual network first maps the concatenation of all the local latent variables and upsampled global latent variables to a feature space. The output of this network is fed to a convolutional layer that generated parameters of a distribution on $\boldsymbol{x}_0$, which is subsampled $\boldsymbol{x}$ at scale $4 \times 4$. In the case of CIFAR10, the output of this layer correspond to the mixture of discretized logistic distribution (Salimans et al., 2017) with 10 mixtures. In the binary datasets, it is the parameters of a factorial Bernoulli distribution. The residual network input $4 \times 4$ is applied to the sample from this scale and its output is concatenated with the output of "context". The distribution on the next scale is formed similarly using another upsampling residual network. This process is repeated until we generate the image in the full scale.

The residual blocks in our work consist of two convolutional layers with an skip connection. Resizing the dimensions is always handled in the first convolutional layer. Downsampling is done using stride of 2 and upsampling is implemented using transposed convolution. The squeeze and excitation unit is applied with the reduction ratio $r = 4$ (Hu et al., 2017).

The SELU (Klambauer et al., 2017) and ELU (Clevert et al., 2015) activation functions are used in all the fully connected and convolutional layers respectively. No batch normalization was used except in the input of the encoder, the output of "down 1", and "down 2". AdaMax (Kingma & Ba, 2014) is used for training all the models. The learning rate is set to 0.001 and is decreased when the value of the variational bound on the validation set plateaus. The batch size is 100 for all the experiments. In all experiments, the $\beta$ smoothing parameter is set to 8.

We use parallel tempering (Hukushima & Nemoto, 1996; Iba, 2001) to approximate the partition function of the RBM which is required to evaluate generative log-likelihoods.

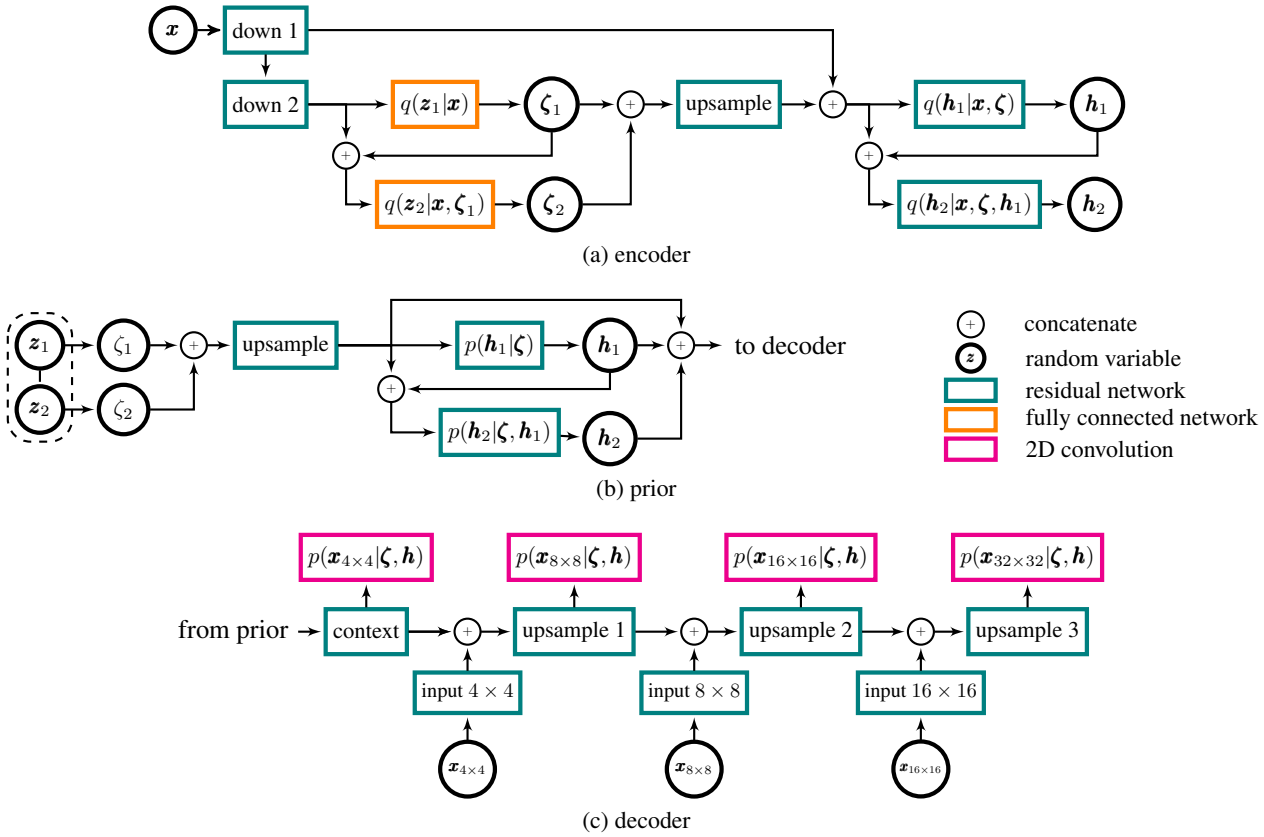


Figure 7: The DVAE++ architecture is divided into three parts: a) the inference network or encoder that represents $q(\boldsymbol{z}, \boldsymbol{\zeta}, \boldsymbol{h}|\boldsymbol{x})$, b) the prior network that models $p(\boldsymbol{z}, \boldsymbol{\zeta}, \boldsymbol{h})$, and c) the decoder network that implements $p(\boldsymbol{x}|\boldsymbol{\zeta}, \boldsymbol{h})$. Each part consists of different modules colored differently based on their type. The specific detail for each module is listed in Table 4.

Table 4: The architecture specifics for each module in DVAE++ for different datasets. The numbers correspond to the number of filters used in residual and convolutional blocks or the number of units used in the fully connected layers starting from the first hidden layer up to the last layer in each module. The arrows ↓ and ↑ in front of each number indicate that the corresponding block is downsampling or upsampling its input. For Binarized MNIST and MNIST, we used an identical architecture to OMNIGLOT except that the autoregressive connections are disabled when forming $p(\boldsymbol{x}|\boldsymbol{\zeta}, \boldsymbol{h})$ in the decoder.

| Module | Type | OMNIGLOT | Caltech-101 | CIFAR10 |
|---|---|---|---|---|
| down 1 | residual | 32↓, 32, 64↓, 64, 128↓ | 8↓, 8↓ | 64↓, 64, 256↓, 256 |
| down 2 | residual | 256↓, 512↓ | 8↓, 16↓, 16↓ | 512↓, 512, 1024↓, 1024↓ |
| $q(\boldsymbol{z}_i|\boldsymbol{x}, \boldsymbol{\zeta}_{<i})$ | fully connected | 4 | 4 | 16 |
| upsample | residual | 128↑, 128↑ | 32↑, 32↑, 32↑ | 128↑, 64↑, 32↑ |
| $q(\boldsymbol{h}_i|\boldsymbol{x}, \boldsymbol{\zeta}, \boldsymbol{h}_{<i})$ | residual | 32, 32, 32, 32, 64 | 32, 32, 32, 32, 64 | 32, 64 |
| $p(\boldsymbol{h}_i|\boldsymbol{\zeta}, \boldsymbol{h}_{<i})$ | residual | 32, 32, 64 | 32, 32, 32, 32, 64 | 32, 32, 32, 32, 64 |
| context | residual | 16 | 16 | 256 |
| upsample 1 | residual | 16↑, 16, | 8, 8 | 128, 128, |
| upsample 2 | residual | 16↑, 8 | 8↑, 4 | 64↑, 64 |
| upsample 3 | residual | 8↑ | 4↑ | 32↑ |
| $p(\boldsymbol{x}_i|\boldsymbol{\zeta}, \boldsymbol{h}, \boldsymbol{x}_{<i})$ | convolutional | 1 | 1 | 100 |
| input $h \times w$ | residual | 32 | 32 | 32 |

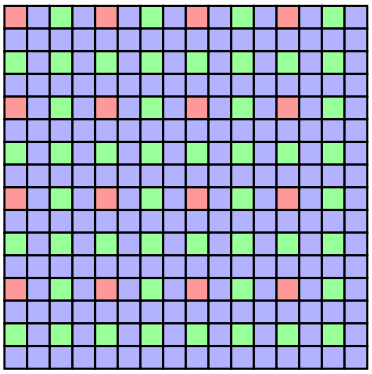

Figure 8: In the conditional decoder we decompose an image into several scales. The generative process starts from the subset of pixels, $\boldsymbol{x}_0$, at the lowest scale ($4 \times 4$). Conditioned on $\boldsymbol{x}_0$, we generate the $8 \times 8$ subset, $\boldsymbol{x}_1$, at the next larger scale. Conditioned on $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$, we generate the $16 \times 16$ subset, $\boldsymbol{x}_2$, at the next larger scale. This process is repeated until all pixels are covered. In this figure, pixels in $\boldsymbol{x}_0$/$\boldsymbol{x}_1$/$\boldsymbol{x}_2$ are indicated by their corresponding color. In order to have a consistent probabilistic model, given $\boldsymbol{x}_{<i}$, we only generate the remaining pixels, $\boldsymbol{x}_i$, at scale $i$.

We use chains at an adaptive number of temperatures, and perform 100,000 sweeps over all variables to ensure that the $\log Z$ estimate is reliable.

### G.1. Conditional Decoder

The conditional decoder for a $16 \times 16$-pixel image is illustrated in Fig. 8.

## H. Balancing the KL Term

With many layers of latent variable, the VAEs tend to disable many stochastic variables (Bowman et al., 2016; Sønderby et al., 2016). Common mitigations to this problem include KL annealing (Sønderby et al., 2016), free bits (Kingma et al., 2016), and soft free bit (Chen et al., 2016).

In our experiments, we observe that annealing the KL term is more effective in maintaining active latent variables than the free bits method. Nevertheless, at the end of training, the units tend to be disabled unevenly across different groups. Some are completely inactive while other groups have many active variables. To address this, we modify the VAE objective function to

$$\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log p(\boldsymbol{x}|\boldsymbol{z})\right] - \gamma \sum_i \alpha_i \mathrm{KL}(q(\boldsymbol{z}_i|\boldsymbol{x})||p(\boldsymbol{z}_i)).$$

$\gamma$ is annealed from zero to one during training, and $\alpha_i$ is introduced to balance the KL term across variable groups. As in soft free bits, we reduce $\alpha_i$ if the $i^{th}$ group has a lower KL value in comparison to other groups and increase it if the KL value is higher for the group. In each parameter update $\alpha_i$ is determined as

$$\alpha_i = \frac{N\hat{\alpha}_i}{\sum_j \hat{\alpha}_j} \text{ where } \hat{\alpha}_i = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{M}}[\mathrm{KL}(q(\boldsymbol{z}_i|\boldsymbol{x})||p(\boldsymbol{z}_i))] + \epsilon.$$

$N$ is the number of latent groups, $\mathcal{M}$ is the current mini-batch, and $\epsilon = 0.1$ is a small value that softens the coefficients for very small values of $KL$. In this way, a group is penalized less in the KL term if it has smaller a KL value, thereby encouraging the group to use more latent variables. We apply a stop gradient operation on $\alpha_i$ to prevent the AD from backpropagating through these coefficients. The $\alpha_i$ are included in the objective only while $\gamma$ is annealed. After $\gamma$ saturates at one, we set all $\alpha_i = 1$ to allow the model to maximize the variational lower bound.

Table 5: The generative performance of DVAE++ improves with the number of local variable groups. Performance is measured by test set log-likelihood in bits per dim.

| # groups | 8 | 12 | 16 | 20 | 24 |
|---|---|---|---|---|---|
| Bits per dim. | 3.45 | 3.41 | 3.40 | 3.40 | 3.39 |

Table 6: The performance of DVAE++ improves with the number of global variable groups in the inference model (i.e. $q(\boldsymbol{z}_i|\boldsymbol{x}, \boldsymbol{\zeta}_{<i})$). Performance is measured by test set log-likelihood in bits per dim.

| # groups | 1 | 2 | 4 |
|---|---|---|---|
| Bits per dim. | 3.39 | 3.38 | 3.37 |

## I. Additional Ablation Experiments

In this section, we provide additional ablation experiments that target individual aspects of DVAE++. The test-set evaluations reported in this section do not use the binary model ($\beta = \infty$), but instead use the same $\beta$ that was used during training.

### I.1. Hierarchical Models Help

As expected, we observe that increasing the number of local variable groups (the number of hierarchical levels) improves the performance of the generative model. Table 5 summarizes the performance of the DVAE++ for CIFAR10 as the hierarchy of continuous local variables is increased. Similarly, when global latent variables are modeled by an

Table 7: DVAE++ with and without conditional decoder

| Baseline | MNIST (static) | MNIST (dynamic) | OMNI-GLOT | Caltech-101 | CIFAR10 |
|---|---|---|---|---|---|
| Conditional | -79.37 | -78.62 | **-92.36** | **-81.85** | **3.37** |
| Unconditional | **-79.12** | **-78.47** | -92.94 | -82.40 | 3.91 |

Table 8: DVAE++ is compared against baselines with either no global latent variables or no KL balancing coefficients.

| Baseline | MNIST (static) | OMNIGLOT | CIFAR10 |
|---|---|---|---|
| DVAE++ | -79.12 | **-92.36** | **3.37** |
| DVAE++ w/o global latent | **-78.96** | -92.60 | 3.41 |
| DVAE++ w/o kl balancing | -79.72 | -92.74 | 3.42 |

RBM, dependencies between discrete latent variables can develop. Modeling of these dependencies can require a deeper hierarchical inference model. Table 6 summarizes the performance of DVAE++ on CIFAR10. In this experiment the number of local groups is fixed to 16. The RBM consists of 128 binary variables and the number of hierarchical levels in the inference model is varied from 1 to 4. Deeper inference models generate high log-likelihoods (low bits per dimmension).

### I.2. Conditional vs. Unconditional Decoder

In Table 7, the performance of DVAE++ with and without conditional decoder is reported. Multi-scale conditional decoders improve generative performance in all the datasets but MNIST.

### I.3. Global Latent Variables and KL Balancing

Table 8 compares the performance of DVAE++ with global latent variables trained with KL balancing with two baselines on three datasets. In the first baseline, the global latent variables are completely removed from the model. In the second baseline, the KL balancing coefficient ($\alpha_i$ in Sec. H) is removed from original DVAE++. Removing either the global latent variables or the balancing coefficients typically decreases the performance of our generative model. However, on binarized MNIST, DVAE++ without a global prior attains a new state-of-the-art result at -78.96 nats.