SUPPLEMENTARY MATERIAL FOR THE PAPER: DEEP ATTENTIVE VARIATIONAL INFERENCE

A MODEL ARCHITECTURE

A.1 INFERENCE NETWORK

In this section, we provide the formal description of the attention-enhanced inference network presented in Section 2.5. Equivalently, we present the procedure for the construction of the conditioning factor of the posterior distribution $q(\boldsymbol{z}_l \mid \boldsymbol{x},\ \boldsymbol{z}_{<l})$.

The inference network is first traversed from the bottom (stochastic layer closer to the data $\boldsymbol{x}$) to the top, to compute successive hidden data representations $\boldsymbol{h}_l \in \mathbb{R}^{H\times W\times C}$ and their keys: $\boldsymbol{k}_l^q \in \mathbb{R}^{H\times W\times Q}$, with $Q \ll C$:

$$[\boldsymbol{h}_l,\ \boldsymbol{k}_l^q] \leftarrow T_l^q(\boldsymbol{h}_{l+1} \oplus \boldsymbol{k}_{l+1}^q), \quad \text{for} \quad l = L, L-1, \ldots, 1. \tag{12}$$

L is the number of stochastic layers. We assume $\boldsymbol{h}_{L+1} \equiv \boldsymbol{x}$. T_l^q is a non-linear transformation, typically implemented as a cascade of ResNet cells (He et al., 2016a;b).

Afterwards, samples from the posterior are drawn in top-down order following the topological ordering of the generative model:

$$\begin{aligned} &\boldsymbol{z}_1 \sim q(\boldsymbol{z}_1 \mid \boldsymbol{x}), \\ &\boldsymbol{z}_l \sim q(\boldsymbol{z}_l \mid \boldsymbol{x},\ \boldsymbol{z}_{<l}), \quad \text{for} \quad l = 2, \ldots, L. \end{aligned} \tag{13}$$

The inference network receives the prior context $\boldsymbol{c}_l^p \in \mathbb{R}^{H\times W\times C}$ and a query feature $\boldsymbol{s}_l^q \in \mathbb{R}^{H\times W\times Q}$ from the generative model, see Equation 7. $\boldsymbol{c}_l^p$ and $\boldsymbol{s}_l^q$ contain latent information of previous layers $\boldsymbol{z}_{<l}$. The observed, data-dependent feature $\boldsymbol{h}_l$ and the latent feature $\boldsymbol{c}_l^p$ are combined via an operator $\oplus$ to give the context of the posterior $\boldsymbol{c}_l^q \in \mathbb{R}^{H\times W\times C}$ along with its key $\boldsymbol{k}_l^{p,q} \in \mathbb{R}^{H\times W\times Q}$:

$$[\boldsymbol{c}_l^q,\ \boldsymbol{k}_l^{p,q}] \leftarrow \boldsymbol{h}_l \oplus \boldsymbol{c}_l^p. \tag{14}$$

As already pointed out, $\boldsymbol{k}_l^{p,q}$ and $\boldsymbol{c}_l^q$ contain stochastic, latent information, while $\boldsymbol{k}_l^q$ and $\boldsymbol{h}_l$ of Equation 12 contain only deterministic, observed information.

We allow variational layer l to have access to $\boldsymbol{h}_l$ and all subsequent data representations $\boldsymbol{h}_{>l}$. During inference, the variational layer decides parts of the deterministic features $\boldsymbol{h}_{\geq l}$ and the stochastic feature $\boldsymbol{c}_l^q$ to pay attention to. For pixel (i, j), with $1 \leq i \leq H$, $1 \leq j \leq W$, the generative model queries according to $\boldsymbol{s}_l^q(i, j) \in \mathbb{R}^Q$ and computes an attention distribution, as follows:

$$\boldsymbol{\alpha}_{m\to l}^q(i,j) = \frac{\exp(\boldsymbol{k}_m^q(i,j)^T \boldsymbol{s}_l^q(i,j))}{\mathcal{Z}^q(i,j)}, \tag{15}$$

$$\boldsymbol{\alpha}_{p\to l}^q(i,j) = \frac{\exp(\boldsymbol{k}_l^{p,q}(i,j)^T \boldsymbol{s}_l^q(i,j))}{\mathcal{Z}^q(i,j)}, \tag{16}$$

$$\mathcal{Z}^q(i,j) = \exp(\boldsymbol{k}_l^{p,q}(i,j)^T \boldsymbol{s}_l^q(i,j)) + \sum_{m\geq l} \exp(\boldsymbol{k}_m^q(i,j)^T \boldsymbol{s}_l^q(i,j)). \tag{17}$$

$\boldsymbol{\alpha}_{p\to l}^q(i,j) \in \mathbb{R}$ reflects the importance of stochastic context $\boldsymbol{c}_l^q(i,j) \in \mathbb{R}^C$ to pixel (i,j). Similarly, $\boldsymbol{\alpha}_{m\to l}^q(i,j) \in \mathbb{R}$ reflects the importance of deterministic feature $\boldsymbol{h}_m(i,j) \in \mathbb{R}^C$ of layer m to pixel (i,j). $\mathcal{Z}^q(i,j) \in \mathbb{R}$ is a normalization constant.

The posterior context for pixel (i, j), $\hat{\boldsymbol{c}}_l^q(i,j) \in \mathbb{R}^C$, is the weighted sum:

$$\hat{\boldsymbol{c}}_l^q(i,j) = \boldsymbol{\alpha}_{p\to l}^q(i,j)\boldsymbol{c}_l^q(i,j) + \sum_{m\geq l} \boldsymbol{\alpha}_{m\to l}^q(i,j)\boldsymbol{h}_m(i,j). \tag{18}$$

To scale the deep variational model with ease, we use the following normalization:

$$\hat{\boldsymbol{c}}_l^q \leftarrow \hat{\boldsymbol{c}}_l^q + \mathrm{Gelu}(\mathrm{LayerNorm}(\hat{\boldsymbol{c}}_l^q)), \tag{19}$$

where Gelu is the GELU non-linearity (Hendrycks & Gimpel, 2016) and $\mathrm{LayerNorm}$ a layer normalization operation (Ba et al., 2016).

At the beginning of the training, we let the inference model rely on nearby latent and observed information represented by $\boldsymbol{c}_l^q$, by introducing the residual connection:

$$\hat{\boldsymbol{c}}_l^q \leftarrow \boldsymbol{c}_l^q + \gamma_l^q \hat{\boldsymbol{c}}_l^q, \tag{20}$$

where γ_l^q is a learnable scalar parameter initialized to zero.

We adopt the Gaussian residual parametrization between the prior and the posterior proposed by Vahdat & Kautz (2020). The prior is given by:

$$p(\boldsymbol{z}_l \mid \boldsymbol{z}_{<l}) = \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{c}_l^p), \boldsymbol{\sigma}(\boldsymbol{c}_l^p)). \tag{21}$$

The posterior is then given by:

$$q(\boldsymbol{z}_l \mid \boldsymbol{x},\ \boldsymbol{z}_{<l}) = \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{c}_l^p) + \Delta\boldsymbol{\mu}(\hat{\boldsymbol{c}}_l^q), \boldsymbol{\sigma}(\boldsymbol{c}_l^p)\Delta\boldsymbol{\sigma}(\hat{\boldsymbol{c}}_l^q)), \tag{22}$$

where the summation and multiplication are pointwise, and $\boldsymbol{c}_l^q$ is defined in Equation 20. $\boldsymbol{\mu}(\cdot)$, $\boldsymbol{\sigma}(\cdot)$, $\Delta\boldsymbol{\mu}(\cdot)$, and $\Delta\boldsymbol{\sigma}(\cdot)$ are transformations implemented as convolutions.

The inference procedure is also described in detail in Algorithm 3.

A.2 Spatially Non-Local Variational Layers

In this section, we provide the formal description of the non-local blocks (Wang et al., 2018) in the recidual cells of encoder and decoder. As shown in Figure 3 and explained below, the output of each block is scaled and added back to the input feature.

Let $\boldsymbol{c} \in \mathbb{R}^{H\times W\times C}$ a context feature. We treat $\boldsymbol{c}$ as a sequence of N, where $N = H \times W$, $C-$ dimensional features, such that $\boldsymbol{c} \in \mathbb{R}^{N\times C}$. The non-local block maps $\boldsymbol{c}$ to a feature $\boldsymbol{v} \in \mathbb{R}^{N\times C}$ of the same dimension by taking into account the response of all the locations in the sequence.

The context feature $\boldsymbol{c}$ is first mapped to two feature spaces of a lower dimension Q, with $Q << C$, through a key $\boldsymbol{\alpha}$ and a query $\boldsymbol{\beta}$ linear transformation:

$$\boldsymbol{\alpha}(\boldsymbol{c}) = \boldsymbol{c}\boldsymbol{W}_\alpha, \tag{23}$$

$$\boldsymbol{\beta}(\boldsymbol{c}) = \boldsymbol{c}\boldsymbol{W}_\beta, \tag{24}$$

where the linear transformations $\boldsymbol{W}_{\boldsymbol{\alpha}} \in \mathbb{R}^{C\times Q}$, $\boldsymbol{W}_{\boldsymbol{\beta}} \in \mathbb{R}^{C\times Q}$ are implemented as 1×1 convolutions, and $\boldsymbol{\alpha}(\boldsymbol{c}) \in \mathbb{R}^{N\times Q}$, $\boldsymbol{\beta}(\boldsymbol{c}) \in \mathbb{R}^{N\times Q}$.

The energy matrix $\boldsymbol{E}(\boldsymbol{c}) \in \mathbb{R}^{N\times N}$ is computed as:

$$\boldsymbol{E}(\boldsymbol{c}) = \boldsymbol{\beta}(\boldsymbol{c})\boldsymbol{\alpha}^T(\boldsymbol{c}). \tag{25}$$

Each element $\boldsymbol{E}_{i,j}(\boldsymbol{c}) = \boldsymbol{\beta}_i(\boldsymbol{c}) \times \boldsymbol{\alpha}_j^T(\boldsymbol{c})$, where $\boldsymbol{\beta}_i$ is the $i-$th row of $\boldsymbol{\beta}$, reflects the relevance of pixel j to pixel i. The attention distribution is computed by a scaled softmax row normalization, such that the corresponding attention score $\boldsymbol{A}_{i,j}(\boldsymbol{c})$ is :

$$\boldsymbol{A}_{i,j}(\boldsymbol{c}) = \frac{\exp(\omega\boldsymbol{\beta}_i(\boldsymbol{c})\boldsymbol{\alpha}_j^T(\boldsymbol{c}))}{\mathcal{Z}(i)}, \tag{26}$$

$$\mathcal{Z}(i) = \sum_{j=1}^{N} \exp(\omega\boldsymbol{\beta}_i(\boldsymbol{c})\boldsymbol{\alpha}_j^T(\boldsymbol{c})). \tag{27}$$

$\omega \in \mathbb{R}$ is learnable scalar parameter initialized to one. $\boldsymbol{c}$ is also linearly projected to an embedding space via a transformation $\boldsymbol{\eta}$:

$$\boldsymbol{\eta}(\boldsymbol{c}) = \boldsymbol{c}\boldsymbol{W}_{\boldsymbol{\eta}}, \tag{28}$$

where $\boldsymbol{W}_{\boldsymbol{\eta}} \in \mathbb{R}^{C \times C}$ and $\boldsymbol{\eta}(\boldsymbol{c}) \in \mathbb{R}^{N \times C}$. $\boldsymbol{W}_{\boldsymbol{\eta}}$ is implemented as a 1×1 convolution.

The final output $\boldsymbol{v} \in \mathbb{R}^{N \times C}$ is:

$$\boldsymbol{v} = \boldsymbol{A}(\boldsymbol{c})\boldsymbol{\eta}(\boldsymbol{c}). \tag{29}$$

$\boldsymbol{A}(\boldsymbol{c}) \in \mathbb{R}^{N \times N}$ is the matrix of the attention scores defined in Equation 27. Finally, $\boldsymbol{v}$ represents a scaled, residual mapping of $\boldsymbol{c}$ such that the final output is:

$$\boldsymbol{c} \leftarrow \boldsymbol{c} + \gamma \boldsymbol{v}, \tag{30}$$

where $\gamma \in \mathbb{R}$ is a learnable scalar parameter initialized to zero.

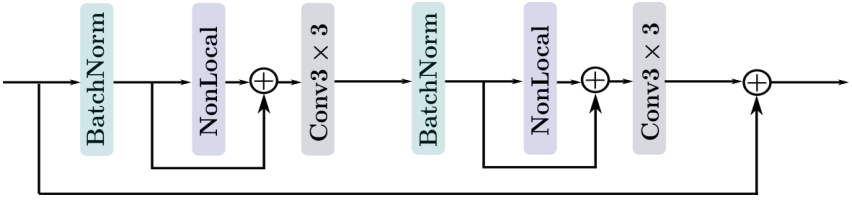


Figure 3: **The spatially non-local ResNet cell.** Non-local operations (Wang et al., 2018) interleaved with the convolutions are responsible for capturing inter-pixel (equivalently, intra-layer) long-range interactions between stochastic features in the same layer of the hierarchy.

A.3 ALGORITHMS

Algorithms 1, 2, and 3 summarize the inference in our model. T_l^p and T_l^q refer to a block of ResNet cells (Figure 3). F^d refers to a single convolution computing the parameters of the data distribution, while convolutions F_l^p and F_l^q give the parameters of the prior and posterior distribution of layer l respectively. $\oplus$ is a merging operator unless one of its operands is zero, in which case it corresponds to an identity mapping. Algorithm 1 computes the optimization objective of Equation 2. It consists

Algorithm 1 Infer

`This algorithm computes a lower bound on the marginal data log-likelihood (Equation 2).`

Inputs
  $\boldsymbol{x}$: The data.
$\boldsymbol{h},\ \boldsymbol{k}^q \leftarrow \text{Bottom} - \text{Up Pass}\ (\boldsymbol{x})$ (Algo. 2).
$\boldsymbol{c}^p,\ \boldsymbol{z},\ p(\boldsymbol{z}),\ q(\boldsymbol{z} \mid \boldsymbol{x}) \leftarrow \text{Top} - \text{Down Pass}\ \left(\boldsymbol{h},\ \boldsymbol{k}^q\right)$ (Algo. 3).
$\boldsymbol{c}_{L+1}^p \leftarrow T_{L+1}^p(\boldsymbol{c}_L^p \oplus \boldsymbol{z}_L)$.
$\boldsymbol{\theta} \leftarrow F^d(\boldsymbol{c}_{L+1}^p)$.
$p(\boldsymbol{x} \mid \boldsymbol{z}) \triangleq p(\boldsymbol{x}; \boldsymbol{\theta})$.
ELBO $\approx \log p(\boldsymbol{x} \mid \boldsymbol{z}) - D_{KL}(q(\boldsymbol{z} \mid \boldsymbol{x}) \parallel p(\boldsymbol{z}))$.
Return
  ELBO: The evidence lower bound.

of three main steps:

1. The bottom-up pass that is described described in Algorithm 2.
2. The top-down pass that is described described in Algorithm 3.
3. The formation of the conditional data distribution $p(\boldsymbol{x} \mid \boldsymbol{z})$.

Algorithm 2 Bottom-Up Pass

`This algorithm computes a sequence of deterministic, hidden features and keys of the data.`

1: **Inputs**
2: $\boldsymbol{x}$: The data.
3: **Set** $\boldsymbol{h}_{L+1} \equiv \boldsymbol{x}$ and $\boldsymbol{k}^q_{L+1} \equiv 0$.
4: **for** $l = L, L-1, \dots, 1$ **do**
5: $[\boldsymbol{h}_l,\ \boldsymbol{k}^q_l] \leftarrow T^q_l(\boldsymbol{h}_{l+1} \oplus \boldsymbol{k}^q_{l+1})$.
6: **end for**
7: **Return**
8: $\boldsymbol{h} \triangleq \{\boldsymbol{h}_l\}_{l=1}^L$: The hidden data features.
9: $\boldsymbol{k}^q \triangleq \{\boldsymbol{k}^q_l\}_{l=1}^L$: The data keys.

During the bottom-up pass described in Algorithm 2 (see also left part of Figures 1a, 1b), the data $\boldsymbol{x}$ are passed through a sequence of ResNet transformations T^q_l to give a sequence of features $\boldsymbol{h}$ and keys $\boldsymbol{k}^q$. $\boldsymbol{h}$ and $\boldsymbol{k}^q$ will be used by the stochastic layers of the VAE hierarchy when forming the posterior distribution, in a reverse order. Note that in Algorithm 2, $\boldsymbol{k}^q_{L+1}$ is set to zero since layer L has access only to its own $\boldsymbol{h}_L$ and it is not connected with another layer from the bottom. Algorithms 1 and 2 are very similar to that of typical deep VAEs. The only difference is that the ResNet transformations are also producing the data keys $\boldsymbol{k}^q$ along with the data features $\boldsymbol{h}$.

The second, top-down phase of Algorithm 1 infers samples of latent variables $\boldsymbol{z}$ along with the prior and posterior distributions. It is described in Algorithm 3 and explained in detail below. In Algorithm 1, $\boldsymbol{c}^p_L$ encapsulates latent information of $\boldsymbol{z}_{<L}$. Along with $\boldsymbol{z}_L$, they feed the ResNet block T^p_{L+1} succeeded by the convolution F^d to compute the conditional data likelihood $p(\boldsymbol{x} \mid \boldsymbol{z})$ that is parameterized by $\boldsymbol{\theta}$.

Algorithm 3 describes the top-down pass that is also visualized in the right part of Figures 1a, 1b. The layers are traversed from the top ($l = 0$) to the bottom-up ($l = L$), sampling latent variables $\boldsymbol{z}_l$ from the posterior distribution and computing context features $\boldsymbol{c}^p_l$ that contain processed latent information of previous layers. The prior distribution is also computed in order to obtain the prior regularization term $D_{KL}(q(\boldsymbol{z} \mid \boldsymbol{x}) \parallel p(\boldsymbol{z}))$ of the objective function in Equation 2.

The first layer does not receive latent information or context from a previous layer, therefore it sets $\boldsymbol{z}_0$ and $\boldsymbol{c}^p_0$ to zero. Similarly, since there is no other layer in the generative network to connect with, it does not produce a query $\boldsymbol{s}^p_1$ to attend with. The last layer does not pass any prior information to a subsequent layer, therefore it does not compute a key $\boldsymbol{k}^p_L$. Similarly, the last layer can use only its own data features $\boldsymbol{h}_L$. Therefore, the generative portion of the layer does not compute a query for the inference portion setting $\boldsymbol{s}^q_L \equiv 0$.

Each layer l, unless in a marginal case described above, computes four types of features $\boldsymbol{k}^p_l$, $\boldsymbol{s}^p_l$, $\boldsymbol{s}^q_l$ and $\boldsymbol{c}^p_l$ (Line 6). Key features $\boldsymbol{k}^p_l$ are used at the depth-wise Attend step of next layers $> l$ (Line 10). $\boldsymbol{s}^p_l$ is a query to the generative network of previous layers $< l$. $\boldsymbol{s}^q_l$ is a query to the inference network of current and next layers $\geq l$. Finally, $\boldsymbol{c}^p_l$ are stochastic features that, along with $\boldsymbol{s}^q_l$, will be used to form the condition of the prior distribution. We note here that $\boldsymbol{s}^q_l$ is used both to form the condition of the prior $\hat{\boldsymbol{c}}^p_l$ (Line 7) and to query the inference network when forming the condition of the posterior $\hat{\boldsymbol{c}}^q_l$ (Line 21). A layer in a deep VAE that is not attention-guided does not produce $\boldsymbol{k}^p_l, \boldsymbol{s}^q_l, \boldsymbol{s}^p_l$ and the computation falls into the "else" cases of Algorithm 3 (Lines 15, 26).

In Line 10, the generative network queries previous layers via $\boldsymbol{s}^p_l$ to identify latent context features $\boldsymbol{c}^p_{l'}$ represented by keys $\boldsymbol{k}^p_{l'}$ that are most critical when forming its prior beliefs. The resulting feature $\hat{\boldsymbol{c}}^p_l$, after being normalized in Line 12, rectifies the initial context $\boldsymbol{c}^p_l$ by the residual connection of Line 13. The initial prior context $\boldsymbol{c}^p_l$ is being used by next layers while the attention-guided context $\hat{\boldsymbol{c}}^p_l$ is the conditioning factor of the prior distribution formed in Lines 17, 18.

A similar procedure takes place to form the posterior distribution. $\boldsymbol{c}^q_l$ is the initial posterior context that merges deterministic information $\boldsymbol{h}_l$ and stochastic, latent information $\hat{\boldsymbol{c}}^p_l$, as seen in Lines 20, 26. $\boldsymbol{c}^q_l$ is updated with an attention-driven feature computed in Line 21 by the residual connection of

Algorithm 3 Top-Down Pass

`This algorithm infers a sequence of stochastic, latent representations of the data.`

Inputs
  $\boldsymbol{h} \triangleq \{\boldsymbol{h}_l\}_{l=1}^L$: The hidden data features.
  $\boldsymbol{k}^q \triangleq \{\boldsymbol{k}_l^q\}_{l=1}^L$: The data keys.
Set $\boldsymbol{z}_0 \equiv 0$, $\boldsymbol{c}_0^p \equiv 0$, $\boldsymbol{s}_1^p \equiv 0$, $\boldsymbol{s}_L^q \equiv 0$, and $\boldsymbol{k}_L^p \equiv 0$.
for $l = 1, 2, \ldots, L$ **do**
  $[\boldsymbol{s}_l^q,\ \boldsymbol{c}_l^p, \boldsymbol{s}_l^p,\ \boldsymbol{k}_l^p] \leftarrow T_l^p(\boldsymbol{z}_{l-1} \oplus \boldsymbol{c}_{l-1}^p)$.
  $\boldsymbol{c}_l^p \leftarrow \boldsymbol{c}_l^p \oplus \boldsymbol{s}_l^q$.

  `Form the prior distribution.`
  if $l > 1$ **then**
    $\boldsymbol{c}_{l-1}^p \leftarrow \mathrm{LayerNorm}\big(\boldsymbol{c}_{l-1}^p\big)$.
    $\hat{\boldsymbol{c}}_l^p \leftarrow \mathrm{Attend}\big(\{\boldsymbol{c}_{l'}^p\}_{l'=1}^{l-1},\ \boldsymbol{s}_l^p,\ \{\boldsymbol{k}_{l'}^p\}_{l'=1}^{l-1}\big)$.
    $\hat{\boldsymbol{c}}_l^p \leftarrow \hat{\boldsymbol{c}}_l^p + \mathrm{Gelu}(\mathrm{LayerNorm}(\hat{\boldsymbol{c}}_l^p))$.
    $\boldsymbol{c}_l^p \leftarrow \boldsymbol{c}_l^p + \mathrm{Gelu}(\mathrm{LayerNorm}(\boldsymbol{c}_l^p))$.
    $\hat{\boldsymbol{c}}_l^p \leftarrow \boldsymbol{c}_l^p + \gamma_l^p \hat{\boldsymbol{c}}_l^p$.
  else
    $\hat{\boldsymbol{c}}_l^p \leftarrow \boldsymbol{c}_l^p$.
  end if
  $[\boldsymbol{\mu}_l^p,\ \boldsymbol{\sigma}_l^p] \leftarrow F_l^p(\hat{\boldsymbol{c}}_l^p)$.
  $p(\boldsymbol{z}_l \mid \boldsymbol{z}_{<l}) = \mathcal{N}(\boldsymbol{\mu}_l^p, \boldsymbol{\sigma}_l^p)$.

  `Form and sample from the posterior distribution.`
  if $l < L$ **then**
    $[\boldsymbol{c}_l^q,\ \boldsymbol{k}_l^{p,q}] \leftarrow \boldsymbol{h}_l \oplus \hat{\boldsymbol{c}}_l^p$.
    $\hat{\boldsymbol{c}}_l^q \leftarrow \mathrm{Attend}\big(\{\boldsymbol{h}\}_{l'=l}^L \cup \{\boldsymbol{c}_l^q\},\ \boldsymbol{s}_l^q,\ \{\boldsymbol{k}_{l'}^q\}_{l'=l}^L \cup \{\boldsymbol{k}_l^{p,q}\}\big)$.
    $\hat{\boldsymbol{c}}_l^q \leftarrow \hat{\boldsymbol{c}}_l^q + \mathrm{Gelu}(\mathrm{LayerNorm}(\hat{\boldsymbol{c}}_l^q))$.
    $\boldsymbol{c}_l^q \leftarrow \boldsymbol{c}_l^q + \mathrm{Gelu}(\mathrm{LayerNorm}(\boldsymbol{c}_l^q))$.
    $\hat{\boldsymbol{c}}_l^q \leftarrow \boldsymbol{c}_l^q + \gamma_l^q \hat{\boldsymbol{c}}_l^q$.
  else
    $\hat{\boldsymbol{c}}_l^q \leftarrow \boldsymbol{h}_l \oplus \hat{\boldsymbol{c}}_l^p$.
  end if
  $[\Delta\boldsymbol{\mu}_l^q,\ \Delta\boldsymbol{\sigma}_l^q] \leftarrow F_l^q(\hat{\boldsymbol{c}}_l^q)$.
  $q(\boldsymbol{z}_l \mid \boldsymbol{x},\ \boldsymbol{z}_{<l}) = \mathcal{N}(\boldsymbol{\mu}_l^p + \Delta\boldsymbol{\mu}_l^q, \boldsymbol{\sigma}_l^p \Delta\boldsymbol{\sigma}_l^q)$.
  $\boldsymbol{z}_l \sim q(\boldsymbol{z}_l \mid \boldsymbol{x},\ \boldsymbol{z}_{<l})$.
end for
Return
  $\boldsymbol{c}^p \triangleq \{\boldsymbol{c}_l^p\}_{l=1}^L$: The latent context features.
  $\boldsymbol{z} \triangleq \{\boldsymbol{z}_l\}_{l=1}^L$: The inferred latent variables.
  $p(\boldsymbol{z}) = \prod_l p(\boldsymbol{z}_l \mid \boldsymbol{z}_{<l})$: The prior distribution of $\boldsymbol{z}$.
  $q(\boldsymbol{z} \mid \boldsymbol{x}) = \prod_l q(\boldsymbol{z}_l \mid \boldsymbol{x},\ \boldsymbol{z}_{<l})$: The approximate posterior distribution of $\boldsymbol{z}$.

Line 24. In Line 29, the residual parametrization of the posterior, as proposed by (Vahdat & Kautz, 2020) and also described in Equation 22, is derived.

B IMPLEMENTATION DETAILS

For all the experiments, we use the AdaMax (Kingma & Ba, 2014) optimizer with the default parameters for training. The learning rate is decayed according to the cosine schedule with warm-up during the first 5 epochs. When we train for 900 epochs, we use a cosine decay schedule with restarts with $T_0 = 300$ and $T_{mult} = 2$ following the notation of Loshchilov & Hutter (2017). We apply weight normalization (Salimans & Kingma, 2016) in all our models. We use the Swish (Ramachandran et al., 2017) activation function in the ResNet cells of the architecture.

Table 6 summarizes the main training and architectural hyperparameters for the experiments presented in Section 3. However, it should be noted that due to the heavy computing requirements, we do not exhaustively optimize the hyperparameters. Therefore, the results reported could potentially be further improved.

Table 6: **Summary of key hyperparamaters for experiments.** The first set pertains to training hyperparameters. The second set refers to statistical hyperparameters specific to variational models. The last set includes architectural hyperparameters.

Hyperparameter	OMNIGLOT	MNIST	CIFAR-10
# epochs	400	400	900
# GPUs	2	2	4
GPU type	GeForce RTX 2080 Ti	GeForce RTX 2080 Ti	32GB V100-SXM2
batch size per GPU	16	50	40
initial learning rate	0.01	0.01	0.01
l_2-regularization λ	$1.5e-4$	$1.5e-4$	$1.5e-4$
Batch Normalization			
momentum	0.9	0.9	0.9
epsilon	$1e-5$	$1e-5$	$1e-5$
KL annealing	$0.1 \rightarrow 1$	$0.1 \rightarrow 1$	$0.1 \rightarrow 1$
# KL annealing epochs	16	16	16
σ_{noise}	✗	✗	0.001
$\log \sigma_p \geq$	-5.0	-1.0	-1.0
$\log \sigma_p \leq$	5.0	5.0	5.0
$\log \sigma_q \geq$	-5.0	-5.0	-5.0
$\log \sigma_q \leq$	5.0	5.0	5.0
number of layers (L)	15	15	16
spatial dims of latent space $\boldsymbol{z}$	16×16	8×8	16×16
# channels in $\boldsymbol{z}$	20	20	20
Encoder			
# pre-proc blocks	1	2	1
# ResNet cells in pre-proc blocks	3	3	2
# initial channels in pre-proc blocks	32	32	128
# channels in encoder	64	128	256
# ResNet cells in encoder	2	2	2
Decoder			
# channels in decoder	64	128	256
# ResNet cells in decoder	2	2	2
# post-proc blocks	1	2	1
# ResNet cells in post-proc blocks	3	3	2
# initial channels in post-proc blocks	64	128	256
Attention			
key dim in top-down attention path	8	8	20
layer norm in top-down attention path	✓	✓	✓
key dim in bottom-up attention path	8	8	20
layer norm in bottom-up attention path	✗	✓	✓
key dim in spatially non-local blocks	8	16	32

KL Annealing. We apply KL annealing (Sønderby et al., 2016) for the first 16 training epochs starting from 0.1 to 1. This technique introduces a scheduled regularization coefficient β for the KL-divergence. Formally, the regularized ELBO objective becomes:

$$\mathbb{E}_q[\log p(\boldsymbol{x} \mid \boldsymbol{z})] - \beta D_{KL}(q(\boldsymbol{z} \mid \boldsymbol{x}) \parallel p(\boldsymbol{z})), \quad \beta_0 \leq \beta \leq 1,\ \beta_0 < 1, \tag{31}$$

where β linearly increases from β_0 to 1 for a number of epochs at the beginning of the training.

Variational Distributions. We use Gaussian prior and posterior distributions connected via a residual parametrization, see Equation 22. To ease training stability, we bound the log-scale of the prior $\log \boldsymbol{\sigma}_p$ and the posterior $\log \boldsymbol{\sigma}_q$ as reported in Table 6. In case of CIFAR-10, we also apply additive zero mean, Gaussian noise to the prior and posterior scale. The mean $\boldsymbol{\mu}$ and the log-scale $\log \boldsymbol{\sigma}$ are jointly learned by using a shared convolution. In particular, a 1×1 convolution with an ELU activation is used for the prior parameters. For the posterior parameters of all but the first layer, two 3×3 convolutions with an ELU non-linearity are used. For the posterior of the first layer, a single 1×1 convoltion with no activation is used.

Data Distributions. For MNIST and OMNIGLOT, we assume a logit-based parametrization of a Bernoulli distribution. For CIFAR-10, we use a mixture of discretized Logistic distributions (Salimans et al., 2017) with five mixture components. The parameters are jointly learned by a single 3×3 convolution with an ELU activation.

Pre-Processing and Post-Processing Blocks. The data $\boldsymbol{x}$, before being passed to the stochastic hierarchy of L layers, are first pre-processed by a cascade of ResNet cells. Each block is responsible for downsampling the spatial dimension by a factor of 2. For MNIST, the image is downsampled twice by two blocks, leading to latent variables with spatial dimension 8×8. For OMNIGLOT and CIFAR-10, one pre-processing block is being used leading to latent spaces of dimension 16×16. No further downsampling is performed by encoders in the main hierarchy. Each block consists of a series of ResNet cells. The last cell of the block is responsible for downsampling while doubling the number of the output channels. Symmetrically, the context of the last layer L is first being post-processed by a cascade of post-processing blocks before being passed to the data distribution of the decoder. Each such block is responsible for upsampling the context by a factor of 2 in order to match the initial spatial dimension of the image. The upsampling operation is performed by the first ResNet cell while halving the number of the output channels.

Spatially Non-Local Blocks. For CIFAR-10, we interleave non-local blocks as shown in Figure 3 in the pre-processing blocks of the encoder, as well as the encoder and decoder of each layer. We interleave non-local blocks with the convolutions of the ResNet cells as shown in Figure 3. For OMNIGLOT and MNIST, we interleave non-local blocks as shown in Figure 3 in the post-processing blocks of the encoder. We noticed that it suffices to add a single non-local block right before the first ResNet cell of the encoder and decoder of each layer.

C Training Plots

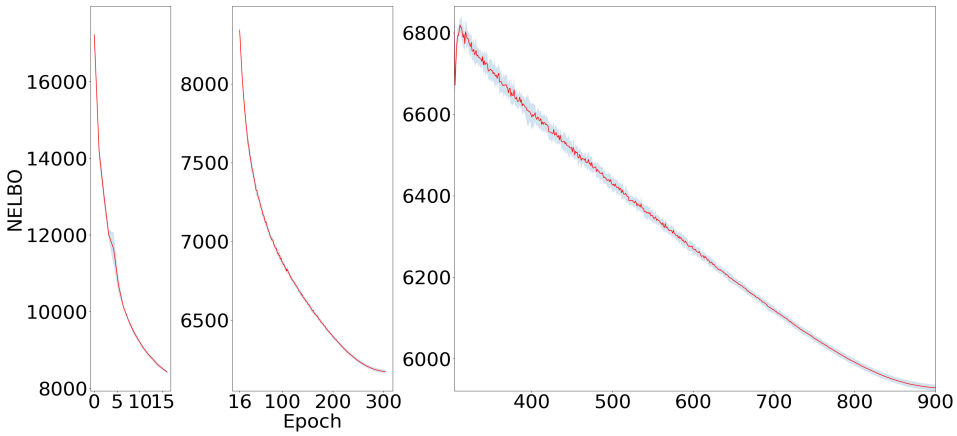


Figure 4: **Training NELBO of Attentive VAE.** NELBO for the CIFAR-10 training dataset across 900 training iterations. The shaded areas show the standard deviation across 10 network initializations and optimizations.

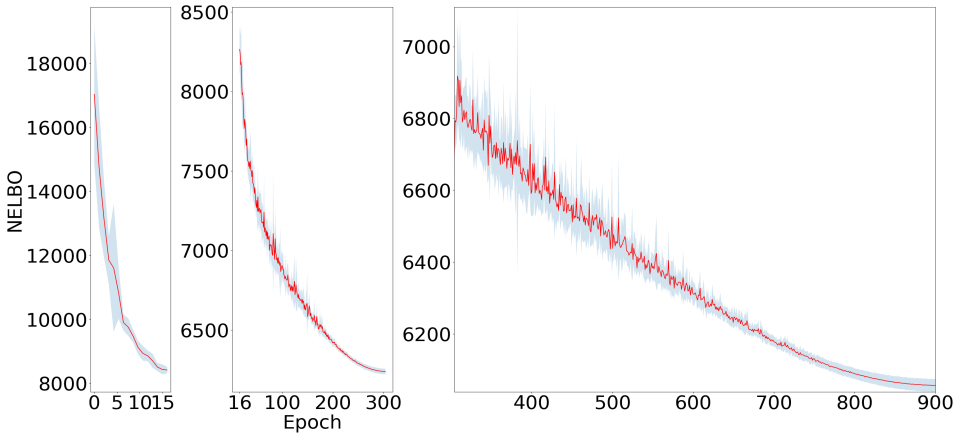


Figure 5: **Validation NELBO of Attentive VAE.** NELBO for the CIFAR-10 validation dataset across 900 training iterations. The shaded areas show the standard deviation across 10 network initializations and optimizations.

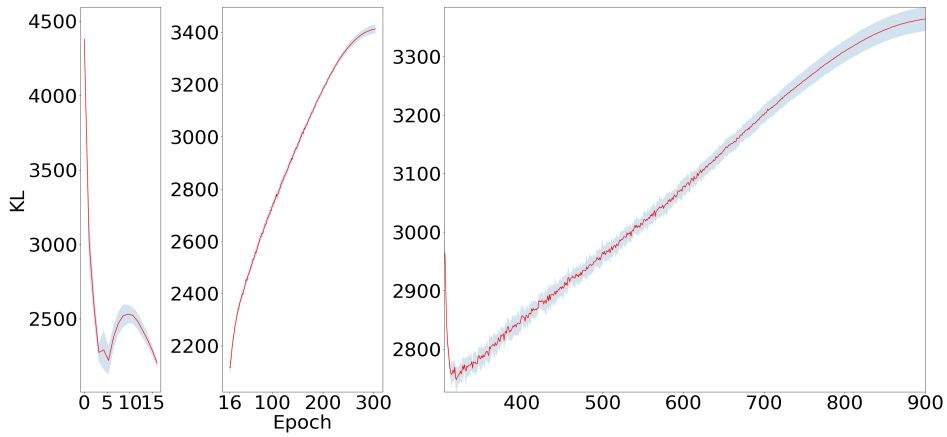


Figure 6: **Training KL loss of Attentive VAE.** KL loss for the CIFAR-10 training dataset across 900 training iterations. The shaded areas show the standard deviation across 10 network initializations and optimizations.

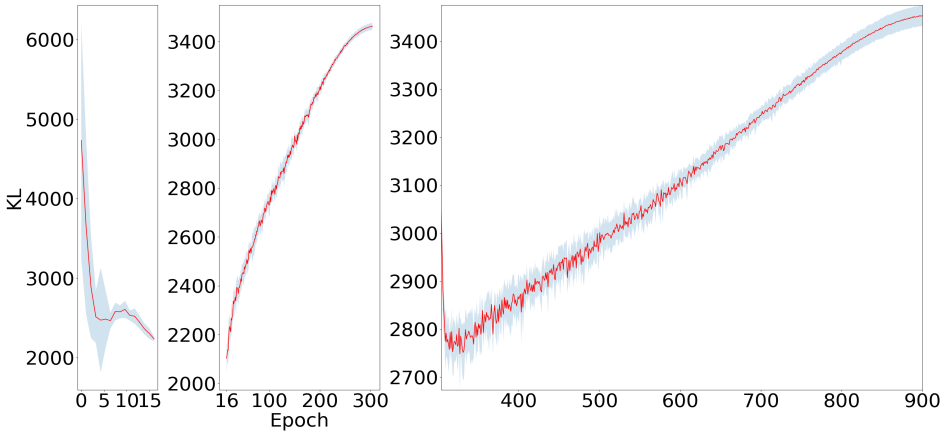


Figure 7: **Validation KL loss of Attentive VAE.** KL loss for the CIFAR-10 validation dataset across 900 training iterations. The shaded areas show the standard deviation across 10 network initializations and optimizations.

Figures 4, 5 show the minimization of the negative ELBO (NELBO) loss in Equation 2 on the training and the validation CIFAR-10 images across 900 training epochs. The first training phase (left part) refers to the KL annealing period. For the next two training phases, the learning rate is annealed according to the cosine schedule. At epoch 300, the learning rate is restarted (right part). Figures 6, 7 focus on the KL loss of the NELBO.

D Ablation studies

In this section, we perform ablation experiments to investigate the effect of different architectural choices on model performance.

For the experiments in Tables 7, 8 and 9, the models were trained on CIFAR-10 for 500 epochs. We used shallower attentive VAE models of 8 layers for these experiments. We considered model instances of a varying size, with 64 and 128 channels at the beginning of the bottom-up network and at the end of the top-down network. As already noted in Section B, the number of channels is halved when a residual cell upsamples the spatial dimension and vice versa. In all cases, model instances with more channels in the ResNet modules in encoder and decoder perform significantly better.

We can also see that both weight normalization (Salimans & Kingma, 2016), Table 7, and batch normalization layers interleaved with the convolutions of both the bottom-up and the top-down model, Table 8, improve performance. These results are consistent with the observations made by Vahdat & Kautz (2020). Moreover, their effect is greater on the larger models.

Table 7: **Weight Normalization.** Predictive performance with and without weight normalization for two model sizes. $-\log p(\boldsymbol{x}) \leq$ (bits/dim) $\downarrow$ is reported.

# initial channels	w/ Weight Norm.	w/o Weight Norm.
64	3.07	3.09
128	**2.94**	2.99

Table 8: **Batch Normalization.** Predictive performance with and without batch normalization for two model sizes. $-\log p(\boldsymbol{x}) \leq$ (bits/dim) $\downarrow$ is reported.

# initial channels	w/ Batch Norm.	w/o Batch Norm.
64	3.07	3.09
128	**2.94**	2.97

In Table 9, we report the performance of different activation functions. The Swish (Ramachandran et al., 2017) and GELU (Hendrycks & Gimpel, 2016) activations yield the best performance. This is in accordance with similar choices in current deep VAE architectures (Vahdat & Kautz, 2020; Child, 2020). The rest of the hyper-parameters are identical to that of Section B.

Table 9: **Activation Function.** Predictive performance for various activation functions for two model sizes. $-\log p(\boldsymbol{x}) \leq$ (bits/dim) $\downarrow$ is reported.

# initial channels	Activation			
	ReLU	ELU	Swish	GELU
64	3.12	3.09	3.07	3.07
128	2.99	2.99	**2.94**	**2.94**

In Table 10, we take a closer look at the normalized depth-wise attention proposed in Section 2.4. The gap in the performance is more noticeable for deep VAE models of 16 layers. We train all model instances for 400 epochs. We omit the application of the normalization scheme presented in Equations 5 and 6 in the generative and/or inference network. We confirm that usage of depth-wise normalization facilitates training and helps scale the depth of the proposed architecture.

Table 10: **Depth-wise Attention Normalization.** Predictive performance for normalized and unnormalized depth-wise attention in decoder and encoder in a deep, 16-layer, VAE.
$-\log p(\boldsymbol{x}) \leq$ (bits/dim) $\downarrow$ is reported.

Normalized Depth-Wise Attention		
Generative Network	Inference Network	$-\log p(\boldsymbol{x}) \leq$ (bits/dim) $\downarrow$
✓	✓	**2.82**
✗	✓	2.89
✓	✗	2.86
✗	✗	2.90

E EMPIRICAL EVIDENCE OF BETTER LATENT VARIABLE UTILIZATION BY ATTENTIVE VAE

In this section, we probe the capacity of attentive VAE to utilize the latent information inserted in the model. We first examine the Kullback–Leibler divergence $D_{KL}(q \parallel p)$ observed on the test data across the hierarchy during training. Unless it explodes, a high $D_{KL}(q \parallel p)$ loss term is often desirable since it leads to a smaller negative ELBO (NELBO) loss in Equation 2. This is because higher $D_{KL}(q \parallel p)$ indicates more informative latent factors $\boldsymbol{z}$, and in turn, higher conditional likelihood $p(\boldsymbol{x} \mid \boldsymbol{z})$. For example, Kingma et al. (2016) use a modified objective function that lower bounds $D_{KL}(q \parallel p)$ by some free bits. Similarly, Vahdat & Kautz (2020) use a residual parametrization between the prior and the posterior to keep more latent variables active.

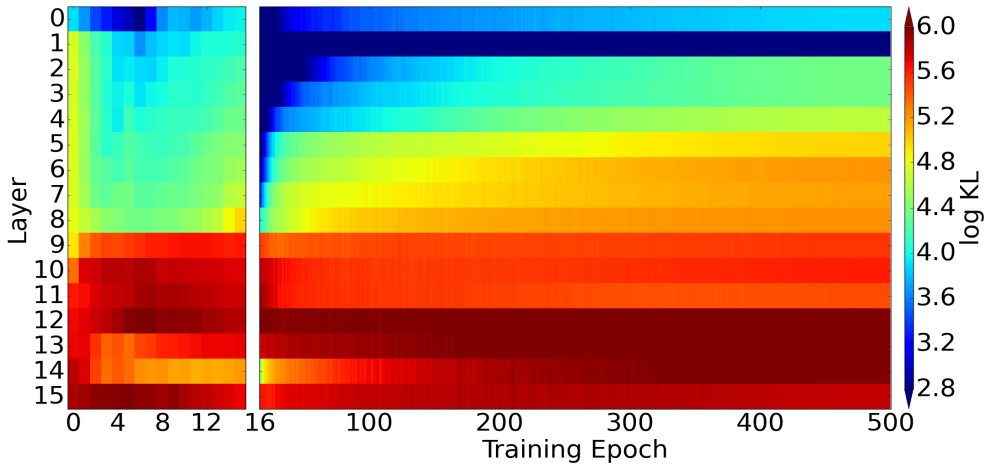


Figure 8: **Latent space activation map for NVAE.** The $\log D_{KL}(q \parallel p)$ for each latent layer $\boldsymbol{z}_l$ as a function of the training epoch on CIFAR-10 for the baseline NVAE.

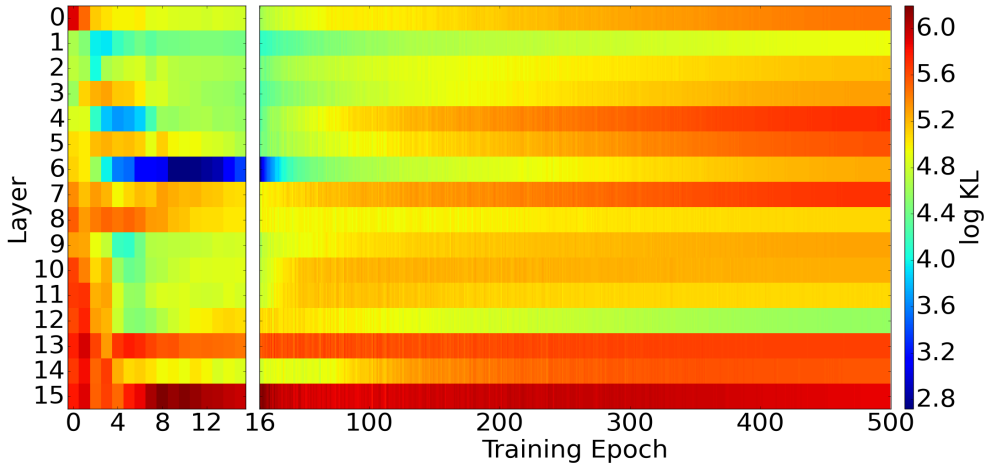


Figure 9: **Latent space activation map for Attentive VAE (ours).** The $\log D_{KL}(q \parallel p)$ for each latent layer $\boldsymbol{z}_l$ as a function of the training epoch on CIFAR-10 for Attentive VAE.

In this work, we tackle posterior collapse by imposing attention-driven connections between the layers. The proposed depth-wise attention naturally realizes long-range couplings between the layers at a pixel level. Therefore, it selectively preserves the most informative latent information which may lie far beyond in the hierarchy. In Figure 8, we show $\log D_{KL}(q \parallel p)$ per layer $\boldsymbol{z}_l$ for the baseline NVAE during the first 16 KL annealing epochs (left part) and for the rest of the training (right part). Similarly, in Figure 9, we show $\log D_{KL}(q \parallel p)$ for the proposed model.

Given the above plots, we make the following observations:

- Although in both models, $\log D_{KL}(q \parallel p)$ remains large (always positive), our model presents a more balanced activation map in the latent space. In contrast, in Figure 8, we see that the first two layers are significantly much less active compared to the layers closer to $p(\boldsymbol{x} \mid \boldsymbol{z})$. This phenomenon is exacerbated in deeper hierarchies.
- In Figure 8, $\log D_{KL}(q \parallel p)$ increases with depth. In contrast, in Figure 9 of our model, the first layer converges to a much higher $D_{KL}(q \parallel p)$ compared to layer 12 although it is much farther from $p(\boldsymbol{x} \mid \boldsymbol{z})$.
- In Figure 9, we notice that as the learning progresses, the attention scores can help activate some latent units. For example, layer 6 is relatively dormant at the beginning of the training, while it converges at a highly informative posterior distribution at the end.

Finally, in Table 11, we show the bits per dimension as the depth varies for our model. All models in Table 11 were trained for 500 epochs. As we can see the predictive improvement gained when doubling the size of the model from 8 to 16 layers is on par with the improvement accomplished when transitioning from a model of 2 layers to a model of 4 layers. In contrast, deep VAEs without attention experience half the amount of relative improvement, as seen in Table 1. We also note that the first two smallest models have converged during the last 100 training epochs which is not the case for the last two model instances. Therefore, the gap in the relative decrease for the models of 8 and 16 layers could be further closed by longer training.

Table 11: $-\log p(x)$ **for varying depth** L **(bits/dim) for attentive VAE (ours).**

Depth (L)	bits/dim $\downarrow$	$\Delta(\cdot)\%$
2	3.31	–
4	3.12	-5.7
8	2.96	-5.1
16	2.81	-5.1

F VISUALIZATION OF ATTENTION PATTERNS

In order to get further insights into the attention-driven inference, we visualize the attention maps proposed in this work. In Figure 10, we show the layer connectivity in the generative network as explained in Section 2.4 for some test images that belong to different classes. For the same images, in Figure 11 we illustrate the depth-wise attention scores in the inference network we describe in Section 2.5, and explain in greater detail in Section A.1. For both cases, we chose to visualize the layer which has the largest receptive field, i.e, the last layer for $L = 16$ which is right before the conditional likelihood $p(\boldsymbol{x} \mid \boldsymbol{z}_L)$ in the decoder, and the first layer which has access to all hidden, data-dependent feature maps of the bottom-up pass in the encoder.

This analysis validates the hypothesis made in this work and suggests the following conclusions:

- For all pixels, the layer relies greatly on the immediately above layer when forming the prior distribution, as shown in Figure 10.
- Different latent pixels of the same image can discover related latent clues that lie in different layers.
- Similarly, the connectivity patterns in the generative network vary significantly across images.
- Although the context $\boldsymbol{c}_{14}^{p}$ already carries latent information from all previous layers, the decoder does not depend solely on that. Instead, it chooses to activate direct connections with layers far beyond in the hierarchy. This reinforces our hypothesis that the effect of latent variables in early layers may be obfuscated by more recent latent information leading in practice to non fully factorized prior distributions: $p(\boldsymbol{z}) = \prod_l p(\boldsymbol{z}_l \mid \boldsymbol{z}_{<l})$.
- The inference attention scores in Figure 11 are more balanced across layers compared to the generative attention scores in Figure 10.
- Mitigation of posterior collapse as described in E is accomplished in a dual fashion. First, all layers during the bottom-up pass of the inference can directly interact with the initial data feature maps of the last layers in the hierarchy, as shown in Figure 11. Second, early latent variables of the top layers condition the prior distribution of the latent variables in layers with closer affinity to the conditional data distribution in the decoder, as shown in Figure 10.

In Figures 12, 13, 14, we plot the attention scores among pixels that lie in the same layer as discussed in Sections 2.6 and A.2. The attention scores pertain to the non-local block of the last convolution of the last residual cell, as depicted in Figure 3, of the ResNet module in the generative portion of each layer.

We derive the following observations:

- The connectivity patterns in all layers are highly structured. This implies that the model could be benefited by sparse attention factorizations such as those proposed by Child et al. (2019), that rely on a similar assumption.
- The attention schemes vary widely across layers. For example, the activation map is more scattered in the first layer of Figure 12 compared to that of the last layer of Figure 14 which is sparser and gathered in periodic spatial locations. This may align with the functionality of the latent information as discussed in current research (Child, 2020), according to which top layers capture global, high-level features while lower layers are responsible for capturing finer details.
- Different images exhibit different spatial attention maps. The differences are more pronounced in the middle layer, as depicted in Figure 13.

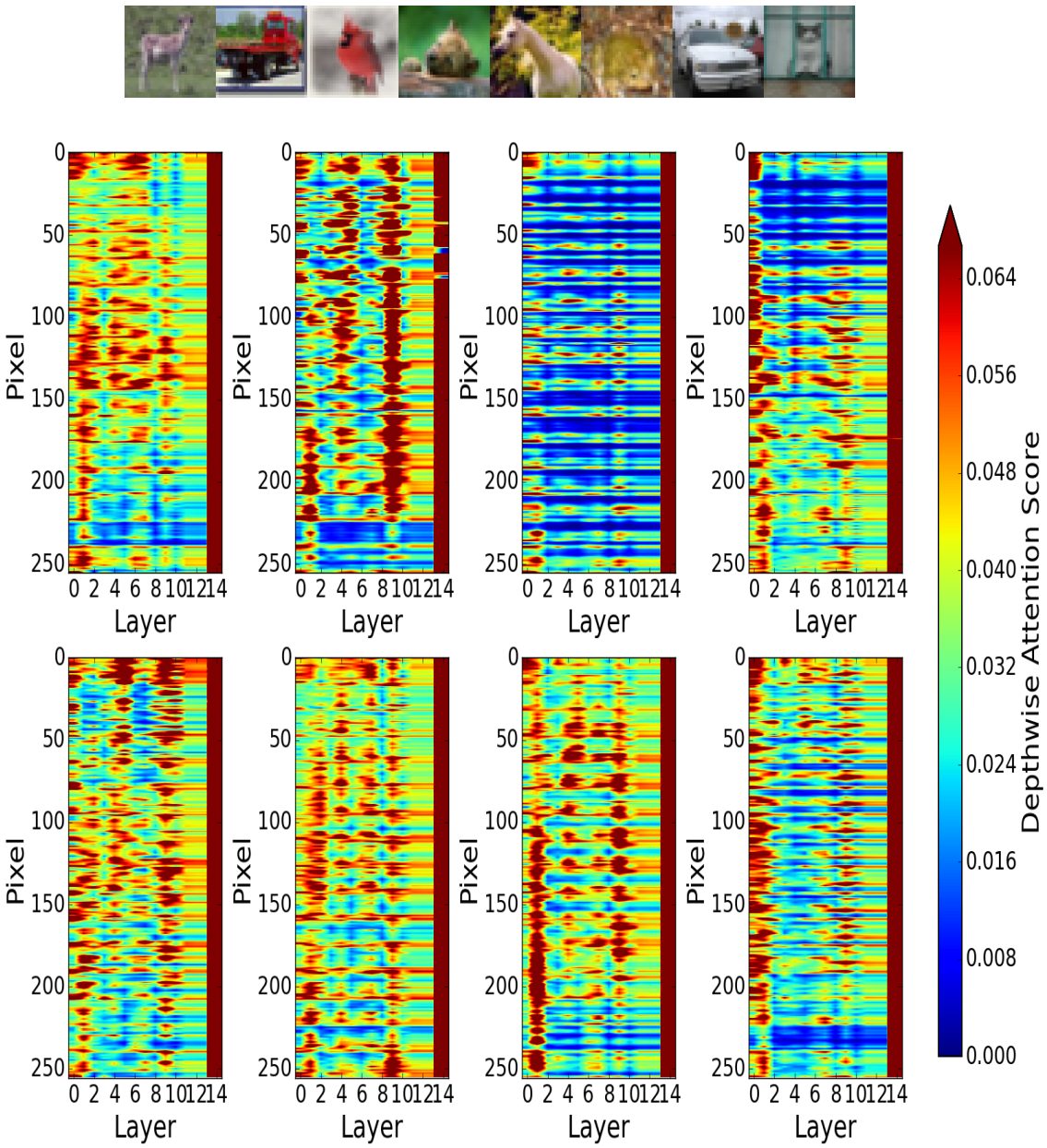


Figure 10: **Depth-wise generative attention scores**. Connectivity map at pixel granularity between the last (16th) layer and the preceding layers in the hierarchy in the generative network using the proposed depth-wise attention for eight test images (top).

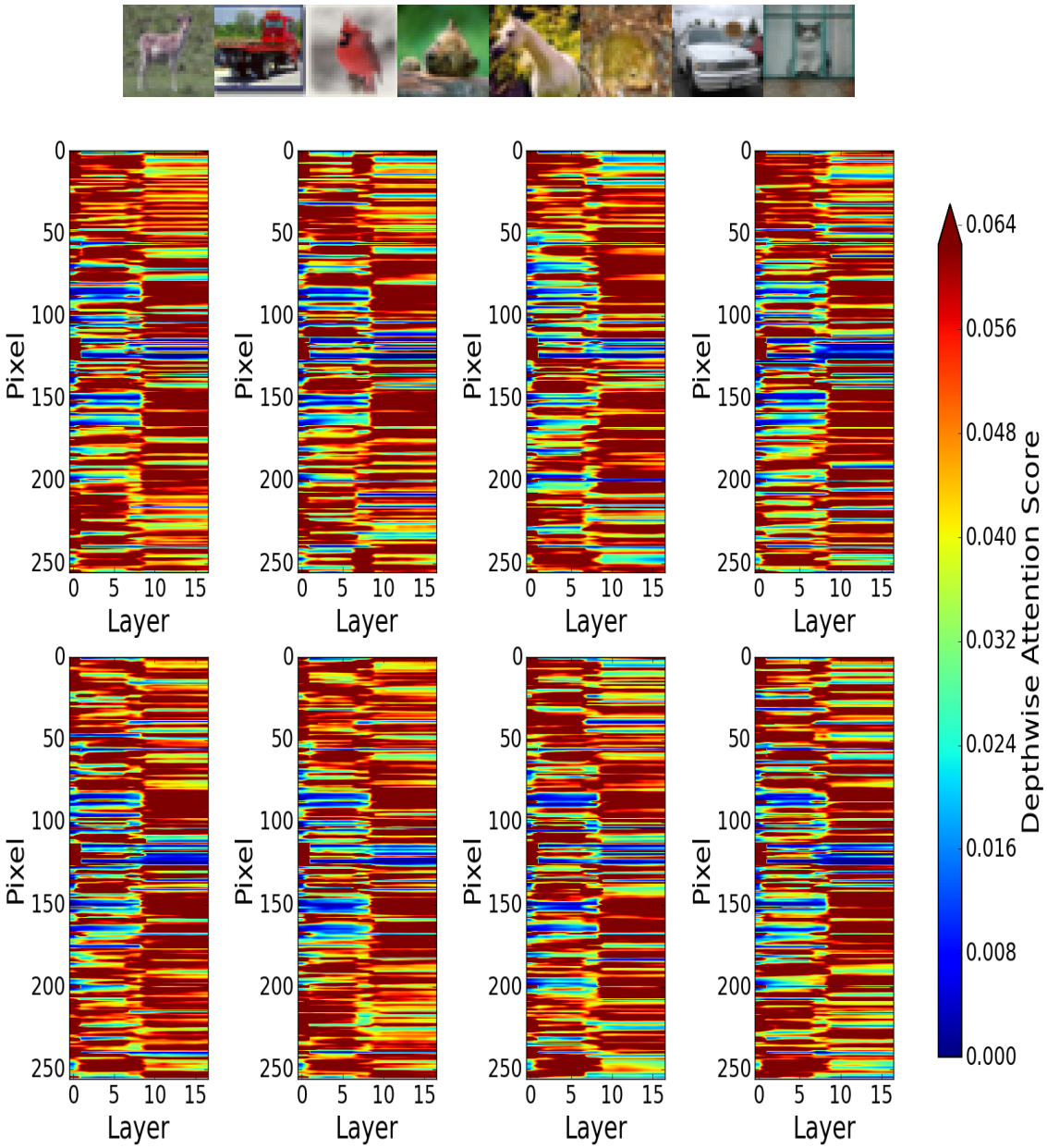


Figure 11: **Depth-wise inference attention scores**. Connectivity map at pixel granularity between the first layer and the subsequent layers in the inference network as dictated by the generative network using the proposed depth-wise attention for eight test images (top).

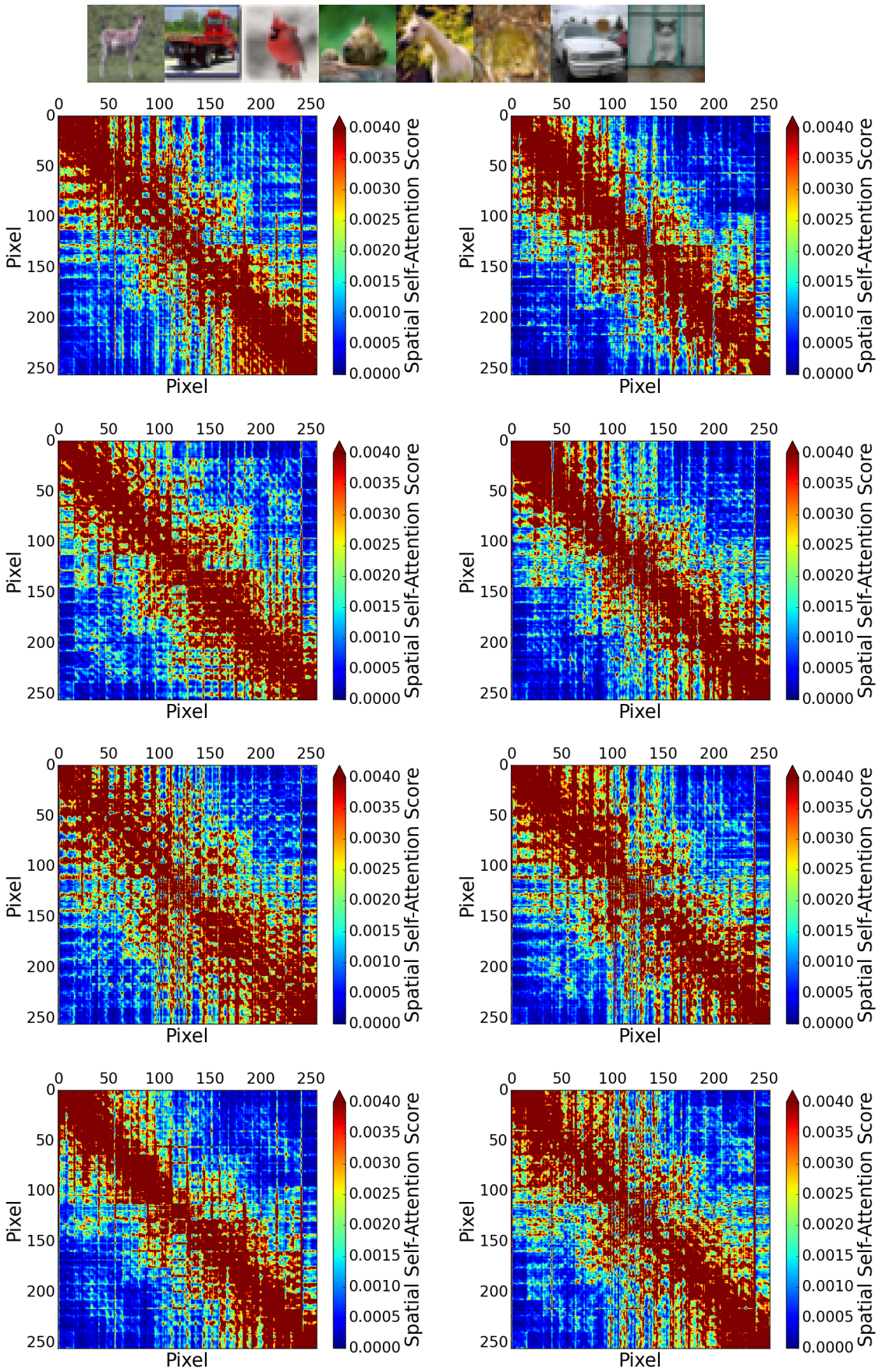


Figure 12: **Spatial attention maps.** Pixel connectivity at the final latent feature map of the first layer in the generative network for eight test images (top).

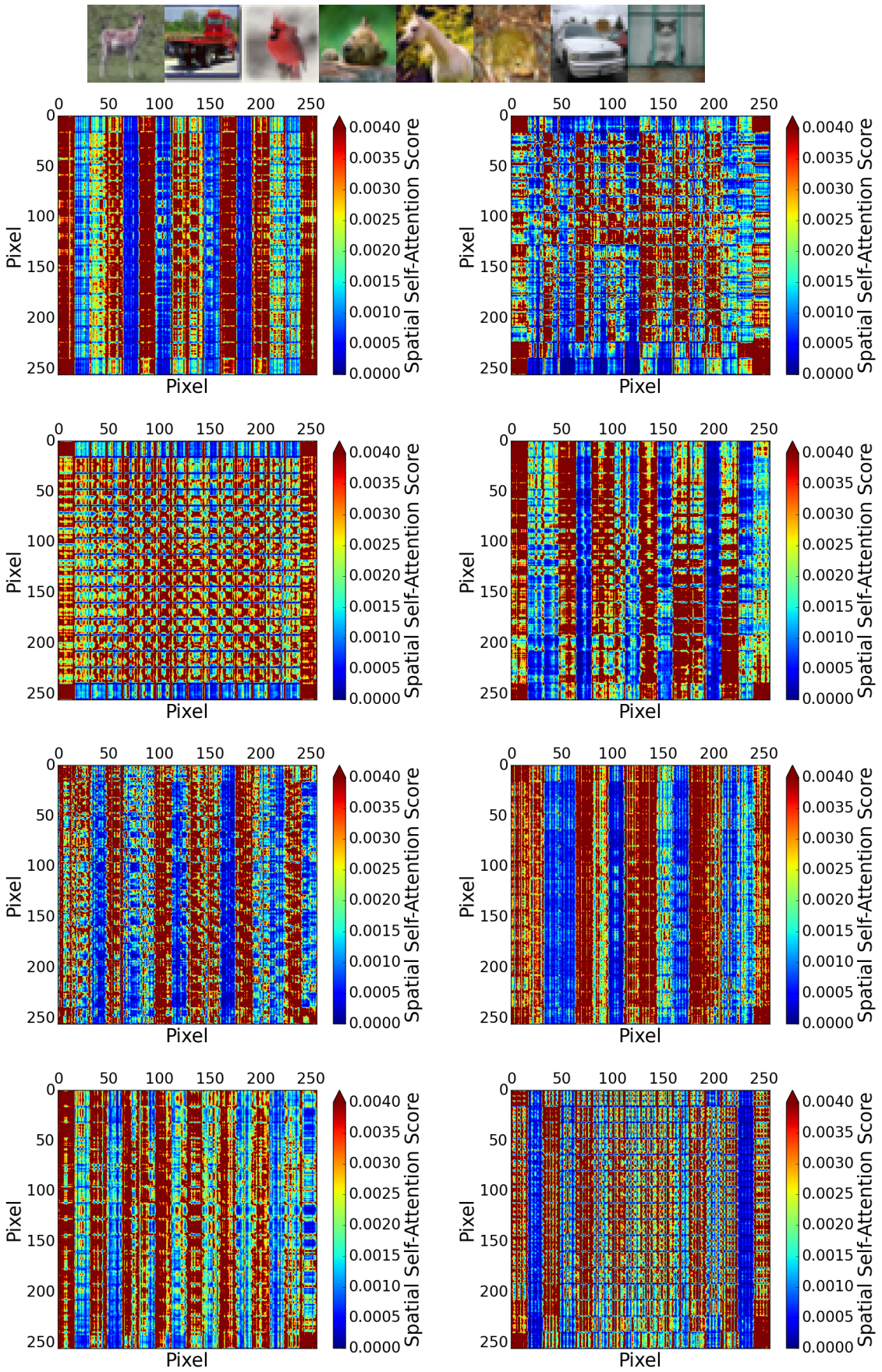


Figure 13: **Spatial attention maps.** Pixel connectivity at the final latent feature map of the middle (8th) layer in the generative network for eight test images (top).

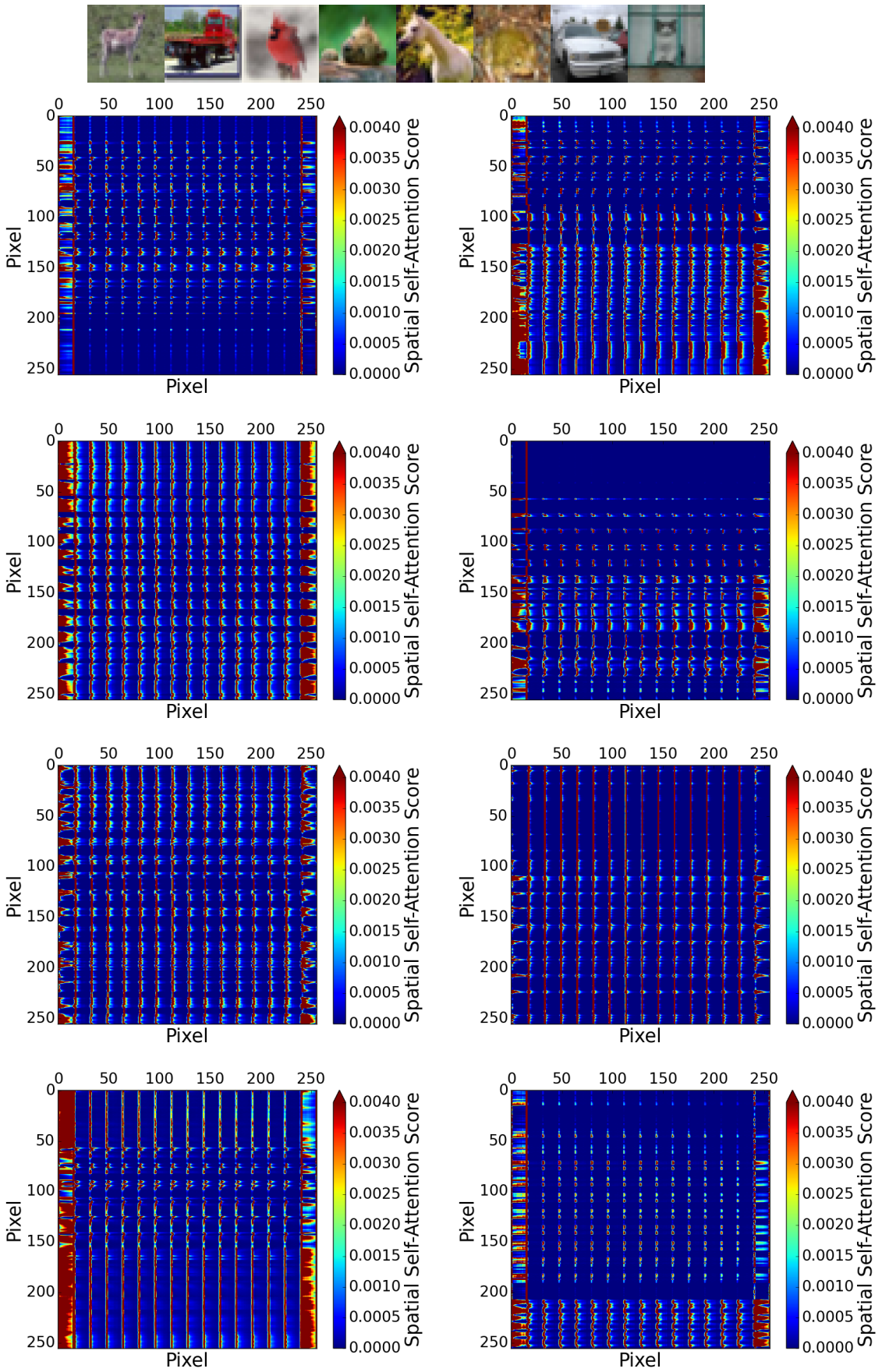
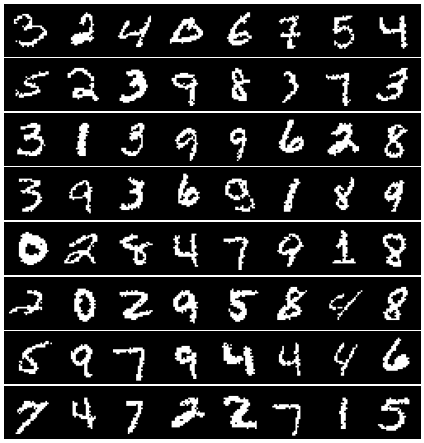


Figure 14: **Spatial attention maps.** Pixel connectivity at the final latent feature map of the last (16th) layer in the generative network for eight test images (top).

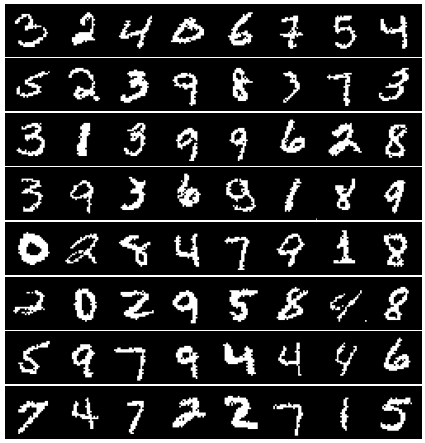
G Reconstructions and Fantasy Samples Generated from our Model

In this section, we qualitatively assess the Attentive VAE presented in the paper. Figures 15b and 16b illustrate reconstructed MNIST and CIFAR-10 test images from the inferred latent variables.

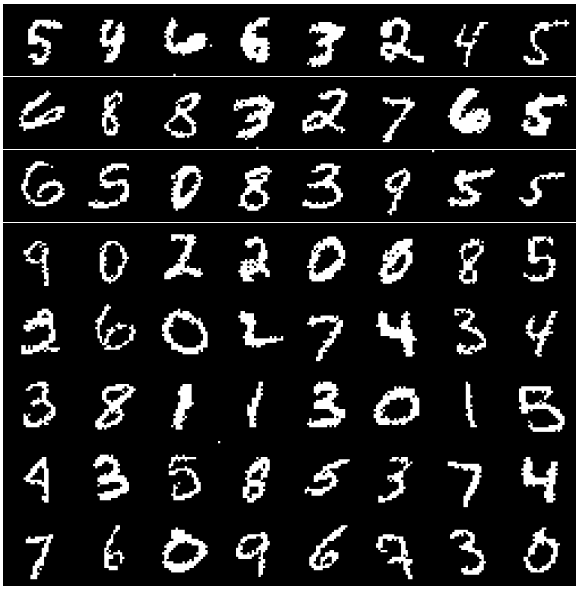
Figures 15c and 16c show novel MNIST and CIFAR-10 images sampled by the generative model.



(a) Original MNIST images.

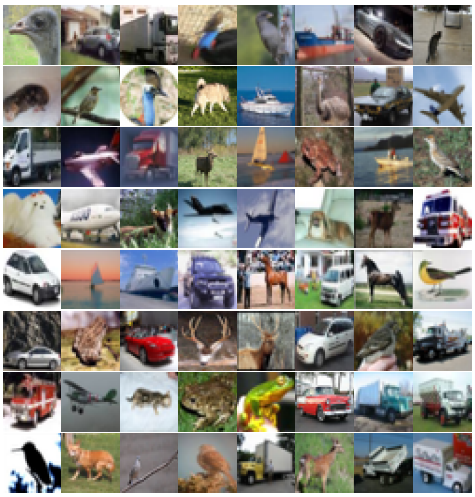


(b) Reconstructed MNIST images.

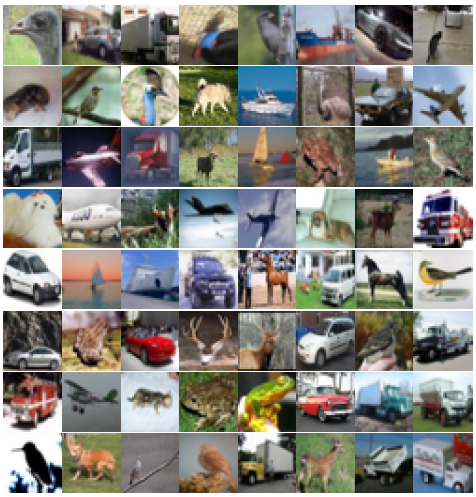


(c) Uncurated generated MNIST images.

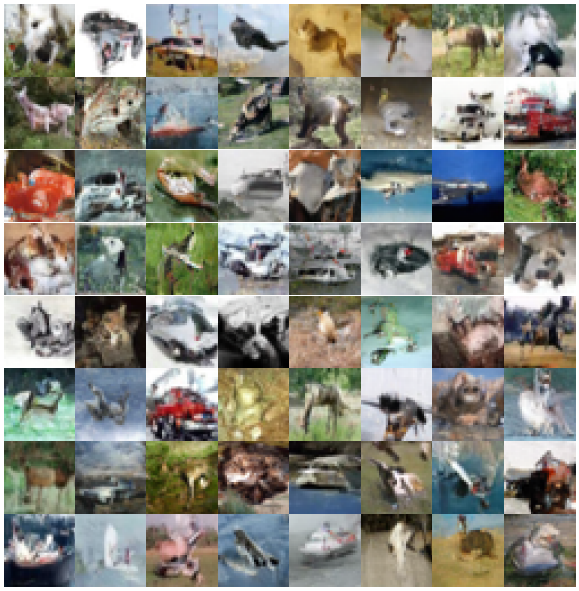
Figure 15: Qualitative evaluation on MNIST images.



(a) Original CIFAR-10 images.



(b) Reconstructed CIFAR-10 images.



(c) Uncurated generated CIFAR-10 images.

Figure 16: Qualitative evaluation on CIFAR-10 images.