Supplementary Material for AdaVAE: Bayesian Structural Adaptation for Variational Autoencoders

Paribesh Regmi **Rui Li***
Rochester Institute of Technology
{pr8537, rxlics}@rit.edu

1 Proof of Lemma 1

We follow the notations defined in the main text. First, the LHS of Eqn. (12) can be re-written as:

$$\mathbb{E}_{q(\mathbf{Z},\boldsymbol{\nu})}\mathbb{E}_{q_\phi(\mathbf{h}|\mathbf{x},\mathbf{Z})}\log\frac{p_\theta(\mathbf{x},\mathbf{h}|\mathbf{Z})}{q_\phi(\mathbf{h}|\mathbf{x},\mathbf{Z})} = -\mathbb{E}_{q(\mathbf{Z},\boldsymbol{\nu})}\mathrm{KL}(q_\phi(\mathbf{h}|\mathbf{x},\mathbf{Z})||p_\theta(\mathbf{h}|\mathbf{x},\mathbf{Z})) + \log p_\theta(\mathbf{x}) \quad (1)$$

Since KL divergence $\mathrm{KL}(q||p)$ is convex in the pair of distributions (q, p) [1], using the first order condition of convexity [2], we have: [3] (the conditioning on $\mathbf{x}$ is dropped for notational convenience):

$$\begin{aligned}\mathrm{KL}(q_\phi(\mathbf{h}|\mathbf{Z})||p_\theta(\mathbf{h}|\mathbf{Z})) \geq &\mathrm{KL}(Q(\mathbf{h})||p_\theta(\mathbf{h}|\mathbf{Z}))+\\ &(q_\phi(\mathbf{h}|\mathbf{Z}) - Q(\mathbf{h}) \cdot (\nabla_{q_\phi(\mathbf{h}|\mathbf{Z})}\mathrm{KL}(q_\phi(\mathbf{h}|\mathbf{Z})||p_\theta(\mathbf{h}|\mathbf{Z}))|_{Q(\mathbf{h})})\end{aligned}$$

Taking expectation with respect to $q(\mathbf{Z}, \boldsymbol{\nu})$,

$$\begin{aligned}\mathbb{E}_{q(\mathbf{Z},\boldsymbol{\nu})}\mathrm{KL}(q_\phi(\mathbf{h}|\mathbf{Z})||p_\theta(\mathbf{h}|\mathbf{Z})) \geq &\mathbb{E}_{q(\mathbf{Z},\boldsymbol{\nu})}\mathrm{KL}(Q(\mathbf{h})||p_\theta(\mathbf{h}|\mathbf{Z}))+\\ &\mathbb{E}_{q(\mathbf{Z},\boldsymbol{\nu})}[(q_\phi(\mathbf{h}|\mathbf{Z}) - Q(\mathbf{h}) \cdot (\nabla_{q_\phi(\mathbf{h}|\mathbf{Z})}\mathrm{KL}(q_\phi(\mathbf{h}|\mathbf{Z})||p_\theta(\mathbf{h}|\mathbf{Z}))|_{Q(\mathbf{h})})]\end{aligned}$$

Since, $\mathbb{E}_{q(\mathbf{Z},\boldsymbol{\nu})}q_\phi(\mathbf{h}|\mathbf{Z}) - Q(\mathbf{h}) = 0$

$$\mathbb{E}_{q(\mathbf{Z},\boldsymbol{\nu})}\mathrm{KL}(q_\phi(\mathbf{h}|\mathbf{Z})||p_\theta(\mathbf{h}|,\mathbf{Z})) \geq \mathrm{KL}(Q(\mathbf{h})||p_\theta(\mathbf{h}|\mathbf{Z})) \quad (2)$$

Combining Eqn.(1) and (2), we have

$$\begin{aligned}&\mathbb{E}_{q(\mathbf{Z},\boldsymbol{\nu})}\mathbb{E}_{q_\phi(\mathbf{h}|\mathbf{x},\mathbf{Z})}\log\frac{p_\theta(\mathbf{x},\mathbf{h}|\mathbf{Z})}{q_\phi(\mathbf{h}|\mathbf{x},\mathbf{Z})}\\ &\leq -\mathrm{KL}(Q(\mathbf{h}|\mathbf{x})||p_\theta(\mathbf{h}|\mathbf{x},\mathbf{Z})) + \log p_\theta(\mathbf{x})\\ &= \mathbb{E}_{Q(\mathbf{h}|\mathbf{x})}\log\frac{p_\theta(\mathbf{x},\mathbf{h}|\mathbf{Z})}{Q(\mathbf{h}|\mathbf{x})}\end{aligned}$$

2 Proof of Theorem 1

We have the lower bound for log marginal likelihood as:

$$\mathbb{E}_{q(\mathbf{Z},\boldsymbol{\nu})}\mathbb{E}_{q_\phi(\mathbf{h}|\mathbf{x},\mathbf{Z})}\log\frac{p_\theta(\mathbf{x},\mathbf{h}|\mathbf{Z})}{q_\phi(\mathbf{h}|\mathbf{x},\mathbf{Z})} \leq \log p_{\alpha,\beta}(\mathbf{x})$$

Following the proof of Burda et al. [4], let I be a subset of $Q = \{1, 2, 3..q\}$ with $|I| = k$, $|Q| = q$ and $k < q$. For any sequence of numbers $\{a_1, a_2, ..., a_k\}$, $\mathbb{E}_I = \{i_1, .., i_k\}(\frac{a_{i1}+...+a_{ik}}{k}) = \frac{a_1+...+a_q}{q}$.

*Corresponding author

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

Then we get,

$$\begin{aligned}
\mathcal{L}_q =& \mathbb{E}_{\{\mathbf{Z}_1,..,\mathbf{Z}_q\}}\mathbb{E}_{q_\phi(\mathbf{h}|\mathbf{x},\mathbf{Z})}\log[\frac{1}{q}\sum_q \frac{p_\theta(\mathbf{x},\mathbf{h}|\mathbf{Z}_q)}{q_\phi(\mathbf{h}|\mathbf{x},\mathbf{Z}_q)}] \\
=& \mathbb{E}_{\{\mathbf{Z}_1,..,\mathbf{Z}_q\}}\mathbb{E}_{q_\phi(\mathbf{h}|\mathbf{x},\mathbf{Z})}\log[\mathbb{E}_I\{i_1,..,i_k\}[\frac{1}{k}\sum_k \frac{p_\theta(\mathbf{x},\mathbf{h}|\mathbf{Z}_{i_k})}{q_\phi(\mathbf{h}|\mathbf{x},\mathbf{Z}_{i_k})}]] \\
\geq& \mathbb{E}_{\{\mathbf{Z}_1,..,\mathbf{Z}_q\}}\mathbb{E}_{q_\phi(\mathbf{h}|\mathbf{x},\mathbf{Z})}[\mathbb{E}_I\{i_1,..,i_k\}\log[\frac{1}{k}\sum_k \frac{p_\theta(\mathbf{x},\mathbf{h}|\mathbf{Z}_{i_k})}{q_\phi(\mathbf{h}|\mathbf{x},\mathbf{Z}_{i_k})}]] \\
=& \mathbb{E}_{\{\mathbf{Z}_1,..,\mathbf{Z}_k\}}\mathbb{E}_{q_\phi(\mathbf{h}|\mathbf{x},\mathbf{Z})}\log[\frac{1}{k}\sum_k \frac{p_\theta(\mathbf{x},\mathbf{h}|\mathbf{Z}_k)}{q_\phi(\mathbf{h}|\mathbf{x},\mathbf{Z}_k)}] = \mathcal{L}_k
\end{aligned}$$

Thus,

$$\begin{aligned}
&\mathcal{L}_S \leq \mathcal{L}_{S+1} \leq \log p_{\alpha,\beta}(\mathbf{x}) \\
&\text{where, } \mathcal{L}_S = \mathbb{E}_{q(\mathbf{Z},\boldsymbol{\nu})}\mathbb{E}_{q_\phi(\mathbf{h}|\mathbf{x},\mathbf{Z})}\log[\frac{1}{S}\sum_s \frac{p_\theta(\mathbf{x},\mathbf{h}|\mathbf{Z_s})}{q_\phi(\mathbf{h}|\mathbf{x},\mathbf{Z_s})}], \quad \mathbf{Z}_s \sim q(\mathbf{Z},\boldsymbol{\nu})
\end{aligned}$$

3 Structural diagram of the proposed framework

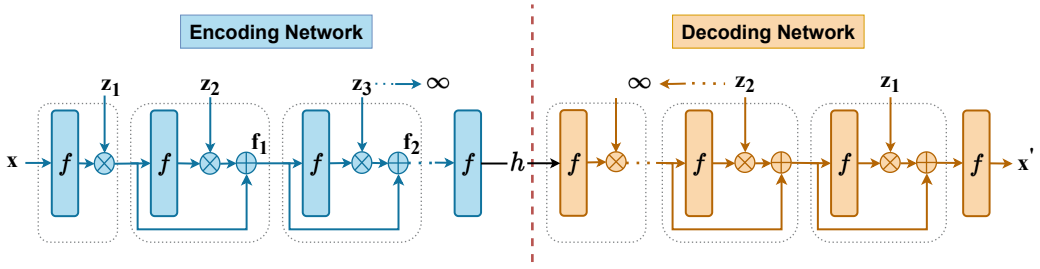

Figure 1: The structural diagram of our proposed framework, as detailed in Section 4 of the paper. At each layer, the output is multiplied by a layerwise binary vector $\mathbf{z}_l$, and the result is added to the output from the previous layer through a skip connection. The binary vector $\mathbf{z}_l$ is generated from the conjugate Bernoulli process. We employ two separate beta processes to model the depths of the encoding and decoding networks.

4 Algorithmic Description

Our proposed method first draws S samples of the network structures $\{\mathbf{Z}_s\}_s^S = 1$ (network depth with layer-wise activation probabilities) from the variational distribution $q(\mathbf{Z}, v)$. The stick-breaking construction of beta-Bernoulli process induces that the probability of seeing activated neurons in hidden layers decreases, and we only need to retain the maximum number of layers with activated neurons up to the current iteration. The details of the algorithm of our method is in Algorithm 1.

Algorithm 1 Training of our proposed method

Input $\{D_i\}_{i=1}^{B}$: B mini batches of data
Input S, M, K

for i $= 1, \ldots, B$ **do**
  Draw S samples of network structures $\{\mathbf{Z}_s\}_{s=1}^{S}$ from $q(\mathbf{Z}, \boldsymbol{\nu})$
  for s $= 1, \ldots, S$ **do**
    Compute the μ_s and σ_s of the variational distribution $q_\phi(\mathbf{h}|\mathbf{x}, \mathbf{Z}_s)$ by performing forward propagation of the encoding network.
    Draw (MK/S) samples of latent variable $\boldsymbol{h}$ from q_ϕ.
    Feed $\boldsymbol{h}$ to the decoding network and obtain $p_\theta(\mathbf{x}|\mathbf{h}, \mathbf{Z}_s)$.
  end for
  Estimate $\mathcal{L}_{\theta,\phi}(\mathbf{x}|\mathbf{Z})$ and $\mathcal{L}_{\{a_t\},\{b_t\}}(\mathbf{x})$ in Eqn. 12 and 13 with Monte Carlo samples of $\boldsymbol{h}$ and $\boldsymbol{Z}$.
  Update $\{a_t, b_t\}_{t=1}^{T}$ and encoding/decoding network weights $\{\mathbf{W}\}_{k=1}^{l^c}$ using backpropagation.
end for

5 Experimental Details

5.1 Datasets

For comparison, we use the binarized MNIST and Omniglot datasets same as in [4], the Caltech101 Silhouettes dataset [5] and the FashionMNIST [6] dataset. All the datasets have an image size of 28×28. For MNIST, we follow the standard split of training and test sets with 60,000 and 10,000 examples respectively. The omniglot dataset is split into 24, 345 training and 8, 070 test examples. For the Caltech101 dataset we have 6,364 examples for training and 2,307 for testing. We also binarize the FashionMNIST dataset and it has a similar train/test split as the MNIST dataset. The details are organized in Table 1. The details of the two citation graph datasets, Cora and Citeseer are presented in Table 2.

Table 1: Image Dataset details

	Training size	Test size	Image size	Classes	Binarized?
MNIST	60,000	10,000	28*28	10	Yes
Omniglot	24,345	8,070	28*28	1623	Yes
Caltech101	6,364	2,307	28*28	101	Yes
FashionMNIST	60,000	10,000	28*28	10	Yes

Table 2: Graph Dataset details

Dataset	Nodes	Edges	Classes	Features
Cora	2708	5429	7	1433
Citeseer	3327	4732	6	3703

5.2 Evaluation Metrics

We evaluate the performance of baselines and our method using three metrics: negative log-likelihood ($-LL$), mutual information (MI) and KL divergence (KL).

For negative log-likelihood $-LL$, we compute the Evidence Lower Bound (ELBO) i.e the IWAE estimator in Eqn. (2) in the main text using importance samples $K = 5000$.

Mutual information (MI) metric quantifies the quality of learned representations by assessing the dependency between the latent representation $\mathbf{h}$ and the input data $\mathbf{x}$. As suggested in [7], we compute

MI using the following equation:

$$\text{MI}(\mathbf{x}, \mathbf{h}) = \mathbb{E}_{p_d(\mathbf{x})}[\text{KL}(q_\phi(\mathbf{h}|\mathbf{x})||p_\theta(\mathbf{h})) - \text{KL}(q_\phi(\mathbf{h})||p_\theta(\mathbf{h}))]$$
$$\text{where, } q_\phi(\mathbf{h}) = \int q_\phi(\mathbf{h}|\mathbf{x})p_d(\mathbf{x})d\mathbf{x}$$

The first term in Eqn. (5.2) corresponds to the KL divergence between the variational distribution and the prior distribution of the latent variable $\mathbf{h}$. The second term represents the KL divergence between the aggregated posterior distribution $q_\phi(\mathbf{h})$ and the prior distribution of $\mathbf{h}$. MI is calculated as the expectation of the terms inside the brackets over the empirical data distribution $p_d(\mathbf{x})$.

We compute the KL divergence between the variational distribution $q_\phi(\mathbf{h}|\mathbf{x})$ and the prior distribution $p_\theta(\mathbf{h})$ of the latent variable $\mathbf{h}$. This metric serves as an indicator of posterior collapse, which occurs when the variational distribution collapses towards the prior distribution, becoming uninformative about the input that generated it. A higher value of KL indicates a lower degree of posterior collapse.

5.3 Implementation of the VAE regularization methods and our framework

The general setup for the experiments (unless mentioned otherwise) including the width of hidden layers ($\mathcal{O}$), the size of latent variables (D_h), learning rate (lr), activation function (act) are detailed in Table 3. The baseline methods are trained to decrease the MIWAE objective (Eqn. (3) in the paper).

Table 3: General hyperparameter setup for baseline methods and our method.

General hyperparameter setup	
$\mathcal{O}$	200
D_h	50
batch size	50
epochs	1000
lr	1e-3
act	$tanh$
optimizer	AdamW (weight decay = 1e-6)

For DVAE, we perturb the input images with a Gaussian noise ϵ and optimize the following ELBO:

$$\mathbb{E}_{q_\phi(\mathbf{h}|\mathbf{x})}[\log p_\theta(\mathbf{x}_\epsilon|\mathbf{h})] - \text{KL}[q_\phi(\mathbf{h}|\mathbf{x}_\epsilon)||p_\theta(\mathbf{h})]$$
$$\text{where }, \mathbf{x}_\epsilon = \mathbf{x} + \epsilon, \quad \epsilon \sim \mathcal{N}(\mu_\epsilon, \sigma_\epsilon^2)$$

For CR-VAE, we augment the input $\mathbf{x}$ with additional transformed samples $\tilde{\mathbf{x}} \sim t(\tilde{\mathbf{x}}|\mathbf{x})$ generated by semantic preserving transformation of the input [7] and add an extra regularization term in the training objective:

$$\begin{aligned}\mathcal{L}_{\text{CR-VAE}}(\mathbf{x}) =& \mathcal{L}_{\theta,\phi}(\mathbf{x}) + \mathbb{E}_{t(\tilde{\mathbf{x}}|\mathbf{x})}[\mathcal{L}_{\theta,\phi}(\tilde{\mathbf{x}})] - \lambda \cdot \mathcal{R}(\mathbf{x}, \phi) \\ &\text{where, } \mathcal{R}(\mathbf{x}, \phi) = \mathbb{E}_{t(\tilde{\mathbf{x}}|\mathbf{x})}\text{KL}(q_\phi(\mathbf{h}|\tilde{\mathbf{x}})||q_\phi(\mathbf{h}|\mathbf{x})) \\ &\text{and, } \mathcal{L}_{\theta,\phi}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{h}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{h})] - \text{KL}[q_\phi(\mathbf{h}|\mathbf{x})||p_\theta(\mathbf{h})]\end{aligned}$$

In Eqn. (5.3), $\mathcal{R}$ represents a regularization term that encourages an image and its transformed sample to share similar latent representations and t denotes the distribution of transformed samples derived from the input $\mathbf{x}$. We augment the training set by including a sample obtained through the simultaneous application of random rotation and translation.

Regarding IBP-DGM and BB-VAE, the truncation level for the dimension of the latent variable is set to 100, as suggested in [8]. To infer the number of latent variable dimensions, a stick-breaking beta-bernoulli process is utilized. The prior for the stick-breaking process is characterized by α and β. Variational inference is carried out using reparametrization trick as in [8].

For our method, we optimize the architecture variables $\{a_t, b_t\}$ using the Adam optimizer with a learning rate of 1e-3. Due to the limited number of training examples per class in the Caltech101 dataset, we impose an additional constraint on the encoding/decoding network. This constraint enforces both the encoding and decoding networks to share the same structure, thereby achieving a symmetric architecture for encoding and decoding networks.

The training was conducted utilizing NVIDIA A100-PCIE-40GB and NVIDIA RTX A5000 GPUs.

Table 4: Implementation details of baselines and our method with MLP backbone.

Methods	Hyperparameter details
MIWAE	M=8, K=8,
MIWAE + DO	M=8, K=8, dropout rate = 0.3
DVAE	M= 8, K=8, $\mu_\epsilon = 0$, $\sigma_\epsilon = 0.18$
IBP-DGM	M=8, K=8, $\alpha = 10$, $\beta = 1$, $\tau = 1$
BB-VAE	M=8, K=8, $\alpha = 10$, $\beta = 1$, $\tau = 1$
CR-VAE	M= 8, K=8, $\lambda = 0.1$, rotation $\sim [-15°, 15°]$, translation $\sim [-7\%, 7\%]$
Ours	S=8, M=8, K=8, T=25, $\alpha = 2$, $\beta = 2$, $\tau = 3$

5.4 Implementation for application to VAE backbone networks

In section 5.5, we assess the efficacy of our framework for VAE with different encoding/decoding backbone networks. For cVAE, the first encoding layer downsamples the input image with a convolutional (conv(5x5)) layer followed by a pooling layer (pool(2x2)). The subsequent L convolutional layers (conv(3x3)) maintain the dimensionality of the features. The output from the convolutional layer block is flattened and fed into a Multi-Layer Perceptron (MLP()) layer that outputs a 50-dimensional latent representation. In the decoder, the latent layer is first unflattened to restore a proper structure and then fed into L transpose convolution (convT(3x3)) layers. Finally, the features are upscaled in the final two layers to reconstruct the original image. The layer details are provided in Table 5. For calculating the $-LL$ metric, we reduce the importance sample size to $K = 3000$ for this experiment to avoid an out-of-memory error.

Table 5: Implementation details of baselines and our method with convolutional layers in the backbone networks. Other hyperparameter details is the same as in Table 3

.

Methods	Details
cVAE	**encoder layers**: conv(5x5) $\rightarrow$ pool(2x2) $\rightarrow$ [conv(3x3)]$_L$ $\rightarrow$ Flatten() $\rightarrow$ MLP() **decoder layers**: UnFlatten() $\rightarrow$ [convT(3x3)]$_L$ $\rightarrow$ Upsample(scale=2) $\rightarrow$ convT(3x3) **others**: number of channels (O) = 30,
Ours + cVAE	**encoder layers**: conv(5x5) $\rightarrow$ pool(2x2) $\rightarrow$ [conv(3x3)]$_T$ $\rightarrow$ Flatten() $\rightarrow$ MLP() **decoder layers**: UnFlatten() $\rightarrow$ [convT(3x3)]$_T$ $\rightarrow$ Upsample(scale=2) $\rightarrow$ convT(3x3) **others**: number of channels (O) = 30,

The Variational Graph Autoencoder (VGAE) introduced by Kipf et al. (2016) [9] incorporates an encoding network composed of graph convolutional (GC) layers, along with an inner product decoder. In a graph convolutional network, the l^{th} hidden layer is defined as follows:

$$\mathbf{f}_l = f(\mathbf{A}\mathbf{f}_{l-1}\mathbf{W}_l)$$

In Eqn. (5.4), $\mathbf{A}$ is the normalized adjacency matrix for the input graph and $\mathbf{W}_l$ is the weight matrix for layer l. We combine our framework with VGAE by masking the GC channels f with the binary mask $\mathbf{z}_l$ generated from the beta-Bernoulli process and adding skip connections between the layers:

$$\mathbf{f}_l = f(\mathbf{A}\mathbf{f}_{l-1}\mathbf{W}_l) \odot \mathbf{z}_{\cdot l} + \mathbf{f}_{l-1}$$

To evaluate the performance of the Variational Graph Autoencoder (VGAE) with and without applying our framework, we conduct experiments on two citation graph datasets: Cora and Citeseer. Following the experimental settings recommended in [9], we split 5% and 10% of the citation links as validation and test sets respectively.

We evaluate VGAE with and without combining our framework on two citation graph datasets: Cora [10] and Citeseer [11]. We follow the experimental settings suggested by [9] and split 5% and 10% of the citation links as validation and test sets respectively. Additional experimental details are given in Table 6. Graph Autoencoder (GAE) shares the same hyperparameters and network structure as VGAE, with the only distinction being that the latent representation layer in GAE is deterministic, whereas in VGAE, it is stochastic.

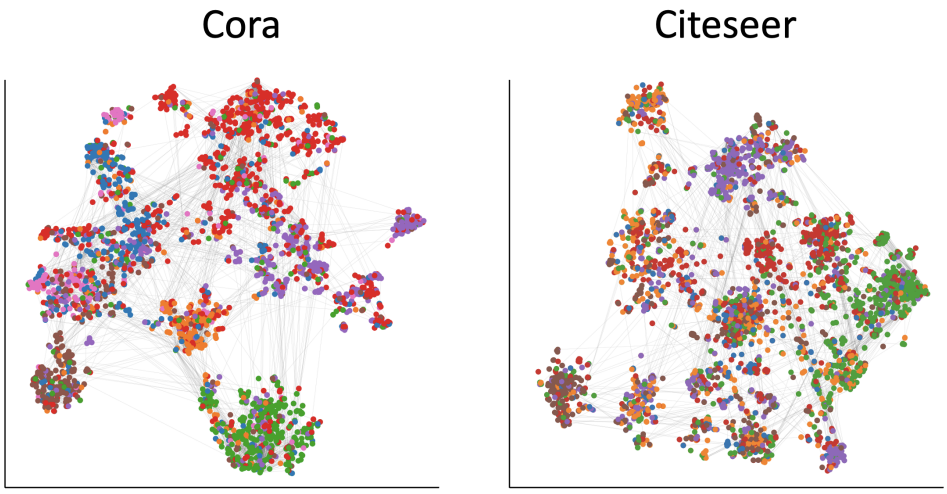

Figure 2: Visualizations of the latent representations obtained by combining our model with VGAE for the Cora and Citeseer citation graphs. To visualize the latent representations, we generate a 2-dimensional t-SNE embedding from the original 16-dimensional latent space. In the visualization, each point represents a node in the graph, and the color of the point corresponds to the class to which it belongs.

Table 6: Implementation details of VGAE with and without combining with our framework. 3

Methods	Details
VGAE	O = 32, D_h = 16, epochs = 200, lr = 1e-2, act = $relu$, optimizer = Adam
Ours+VGAE	S = 40, O = 32, D_h = 16, epochs = 200, lr = 1e-2, act = $relu$, optimizer = Adam

5.5 Implementation for application to VAE variants

We implement three VAE variants with and without combining with our framework: β-VAE [12], SkipVAE [13], and ladder-VAE (LVAE) [14].

β-VAE: In this variant, the KL divergence term in the ELBO of a regular VAE objective is scaled by a factor β. Combining our framework with β-VAE is straightforward and can be done by scaling the KL-term in Equation (11) in the paper, which is the log-likelihood of input $\mathbf{x}$ conditioned on architecture variable $\mathbf{Z}$:

$$\mathcal{L}_{\theta,\phi}(\mathbf{x}|\mathbf{Z}) = \mathbb{E}_{q_\phi(\mathbf{h}|\mathbf{x},\mathbf{Z})}[\log p_\theta(\mathbf{x}|\mathbf{h},\mathbf{Z})] - \beta \cdot \mathrm{KL}[q_\phi(\mathbf{h}|\mathbf{x},\mathbf{Z})||p_\theta(\mathbf{h}|\mathbf{Z})]$$

We implement β-VAE with a convolutional layer backbone with $\beta = 2$ and evaluate on FashionMNIST [6] dataset. We use sample sizes $M = K = 1$ for both methods and $S = 1$ for our method. The rest of the experimental setup for the baseline and our method is the same as in Table 5. We pick the best setting of layer depth i.e. $L = 10$ from section 5.5 (in the paper) for β-VAE. For our method, we set $T = 10$.

SkipVAE: In SkipVAE, the decoding network is characterized by a connection from the latent variable to each of its layers. On the other hand, the encoding network in SkipVAE follows a similar structure to a standard VAE. The l^{th} layer in the encoding and the decoding network of SkipVAE is defined as:

$$\text{Encoding network} : \mathbf{f}_l = f(\mathbf{W}_l \mathbf{f}_{l-1})$$

$$\text{Decoding network} : \mathbf{f}_l = f(\mathbf{W}_l \mathbf{f}_{l-1} + \mathbf{W}_l^h \mathbf{h})$$

We combine our framework with SkipVAE by masking f with a layerwise binary mask $\mathbf{z}_{\cdot l}$ and adding an skip connection between the layers as:

$$\text{Encoding network} : \mathbf{f}_l = f(\mathbf{W}_l \mathbf{f}_{l-1}) \odot \mathbf{z}_{\cdot l} + \mathbf{f}_{l-1}$$

$$\text{Decoding network} : \mathbf{f}_l = f(\mathbf{W}_l \mathbf{f}_{l-1} + \mathbf{W}_l^h \mathbf{h}) \odot \mathbf{z}_{\cdot l} + \mathbf{f}_{l-1}$$

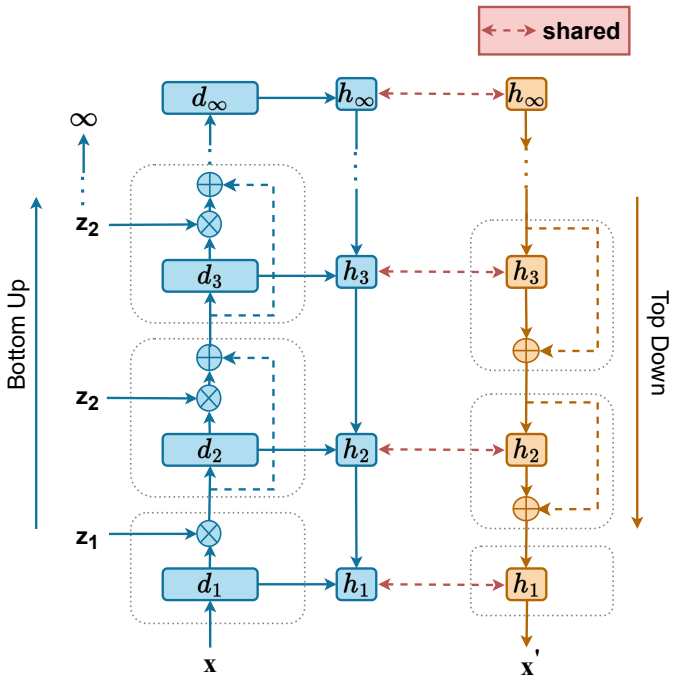

Figure 3: Structural diagram of leveraging our framework to ladder-VAE (LVAE).

We carry experiment on both the methods with sample sizes $M = K = 1$ and our method with $S = 1$ on the FashionMNIST dataset. For the baselines, the experimental setup is the same as in Table 3. The layer depth is set to $L = 3$. For our method combined with SkipVAE, the hyperparameter settings are $\alpha = 2$, $\beta = 2$, $\tau = 3$ and $T = 3$.

Table 7: Negative log-likelihood performance of VAE variants with and without combining with our method on the MNIST dataset.

VAE Variants	$-LL$
β-cVAE [12]	83.56$\pm$0.06
Ours+β-cVAE	**82.67**$\pm$**0.02**
LVAE [14]	116.07$\pm$2.21
Ours+LVAE	**86.07**$\pm$**0.07**
SkipVAE [13]	89.91$\pm$0.08
Ours+SkipVAE	**89.36**$\pm$**0.12**
NVAE [15]	**80.54**$\pm$**0.05**
Ours+NVAE	81.22$\pm$0.15

LVAE: LVAE employs a top-down dependency structure in the inference model, in addition to the bottom-up structure. The top-down structure is shared with the generative model (Figure 1 in [14]), resulting in a ladder-like structure. We combine our framework with LVAE as shown in Figure 3. In the figure, the layers denoted as $d_1, d_2, \ldots$ represent deterministic layers, while $h_1, h_2, \ldots$ represent stochastic layers. Keeping the dependency structure as it is, we infer the number of layers in the bottom-up and top-down networks by modeling the number of deterministic layers in the bottom-up network as a beta process. Then, we multiply the deterministic layers with a binary mask $\mathbf{z}_{\cdot l}$ generated from the conjugate Bernoulli process. We also incorporate skip connections between the deterministic layers in the bottom-up network and stochastic layers in the top-down network as shown in the figure.

We evaluate LVAE and the combination of our method with LVAE on the FashionMNIST dataset. For LVAE, we set the sample sizes $M = K = 1$ and $L = 3$. The details of the rest of the hyperparameters

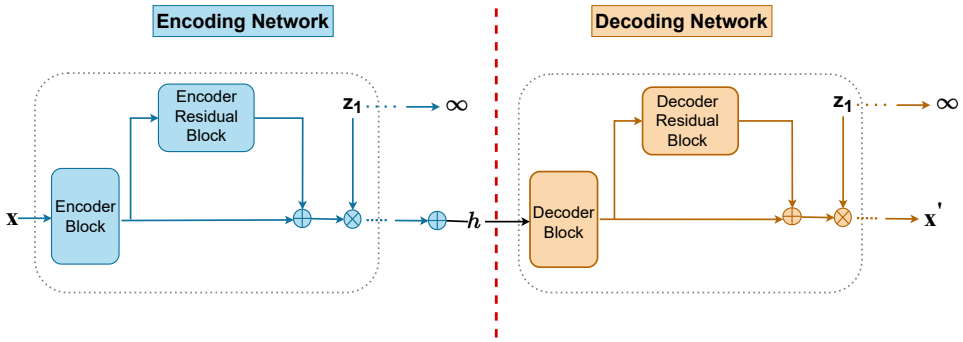

Figure 4: Structural diagram of leveraging our framework to NVAE*.

for LVAE is same as in Table 3. For our method, we set $S = M = K = 1$ and $\alpha = 2$, $\beta = 2$, $\tau = 3$ and $T = 3$.

NVAE: We test our framework on VAE with sophisticated encoding and decoding network layers. For this, we implement a light NVAE network with stacking NVAE [15] blocks in the encoding and decoding networks. The schematic diagram of NVAE is presented in Figure 4. The different blocks used have the following compositions:

Encoder Block: [conv(3x3)→ conv(3x3)→ BN()→ swish()→ conv(3x3)→ BN()→ swish()]$_3$
Encoder Residual Block: [conv(3x3)→ conv(3x3)→ BN()→ swish()→ conv(3x3)→ BN()→ SELayer()]$_3$

Decoder Block: [convT(3x3)→ BN()]$_3$
Decoder Residual Block: [conv(3x3)→ BN()→ swish()→ conv(3x3)→ BN()→ SELayer()]$_3$

SELayer: Pool()→ Linear()→ ReLU()→ Linear()→ Sigmoid()

We conducted additional experiments on the VAE variants using the MNIST dataset, and the results are presented in Table 7. These results are consistent with the findings in Table 3 of the paper, demonstrating that when our method is combined with the VAE variants significantly improves their performance.

6 Training Time Comparison

We compare the training time of the baseline methods and our method in Table 2 in the paper. The time taken for an epoch of training with L/T = 25 for methods is reported in Table 8. The number of structure samples $S = 8$ for our method. The results are consistent with the time complexity analysis in Section 4.3 of the paper.

Table 8: Training time comparison of our method with the VAE regularization methods. The reported value is the time (in seconds) taken to complete an epoch of training. The value of T/L is set to 25.

Methods	MNIST	Omniglot	Caltech
MIWAE	25.41	10.12	2.96
MIWAE+DO	27.98	13.50	3.39
DVAE	34.45	14.26	3.37
CR-VAE	48.69	20.68	5.65
BB-VAE	34.70	12.31	3.38
Ours	235.50	94.05	21.20
Ours+DVAE	262.80	107.6	20.48
Ours+CR-VAE	424.80	182.73	38.40

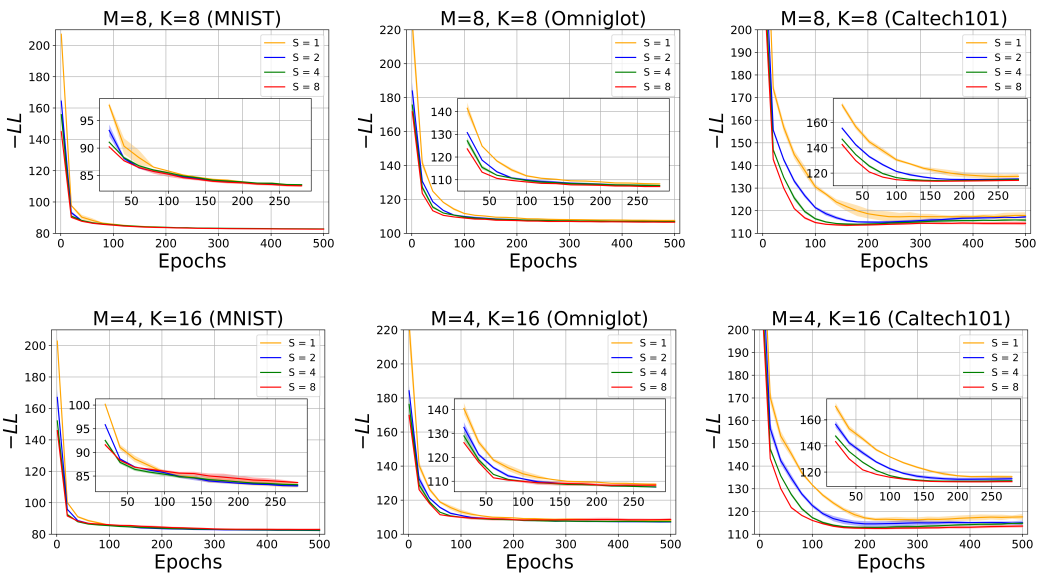

Figure 5: Evaluation of our estimator's convergence on the test sets of MNIST (left column), Omniglot (middle column) and Caltech101 (right column) over training epochs. All lines show mean ± 1 standard deviation over 4 runs with random initializations. The evaluation metric is negative log-likelihood ($-LL$), and smaller values are preferable. By zooming into the "knee" sections of each configuration of S and (M, K), it shows increasing the architecture sample size S leads to more efficient convergence.

7 Comparison of Number of Network Parameters

For the experiment in Figure 3 in the main text, the baselines activate the whole network structures, so the total number of parameters for encoding and decoding networks are 2, where O denotes the maximum number of neurons per layer (i.e., width) and L is the number of layers. For layers $L = 25$ and width of $O = 200$, the baseline methods have 2 million parameters. With the same size of truncation, AdaVAE only activates a part of it in general and fits the activated structures to data. Thus, the activated number of parameters (neuron activation percentage) for $T = 25$ are reported in Table 7. As the table shows AdaVAE uses a comparatively smaller number of parameters.

Table 9: Number of active parameters in millions (neuron activation percentage) of our model for the MNIST, Omniglot and Caltech101 dataset.

Methods	MNIST	Omniglot	Caltech
Ours	0.32(16%)	0.36(18%)	0.21(10.5%)

8 Convergence of our Estimator

We examine the convergence of our estimator during the training of our model four settings of the VAE architecture sample size $S = \{1, 2, 4, 8\}$ for the latent variable sample sizes $(M, K) = \{(8, 8), (4, 16)\}$ as suggested in [16]. The final result of these settings is reported in Table 1 in the paper. As we zoom into the "knee" sections in Figure 5, it shows that larger S leads to faster convergence. This is consistent with Theorem 1 in the paper. In particular, for the smaller dataset Caltech101 this advantage becomes more significant.

9 Evolution of the Encoding/Decoding Network Structure during Training

In Figure 6, we show the evolution of activated layers in the VAE networks along with the percentage neuron activation in the truncation over the number of epochs as discussed in section 5.1 in the paper.

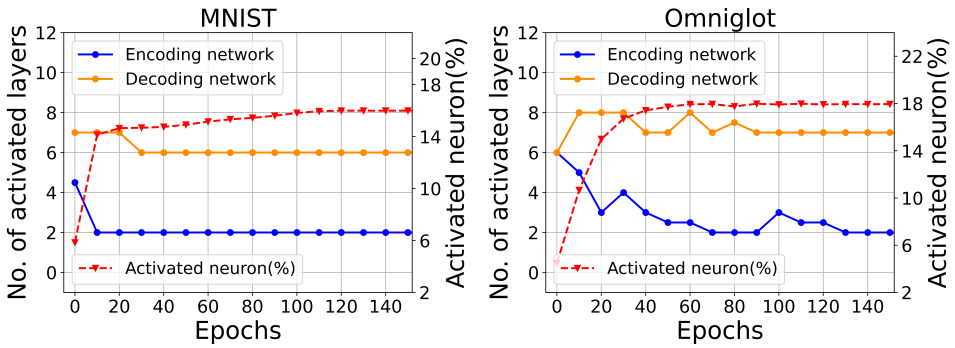

Figure 6: The median change of the number of layers over training epochs and the percentage of activated neurons in the truncation for the MNIST and Omniglot datasets.

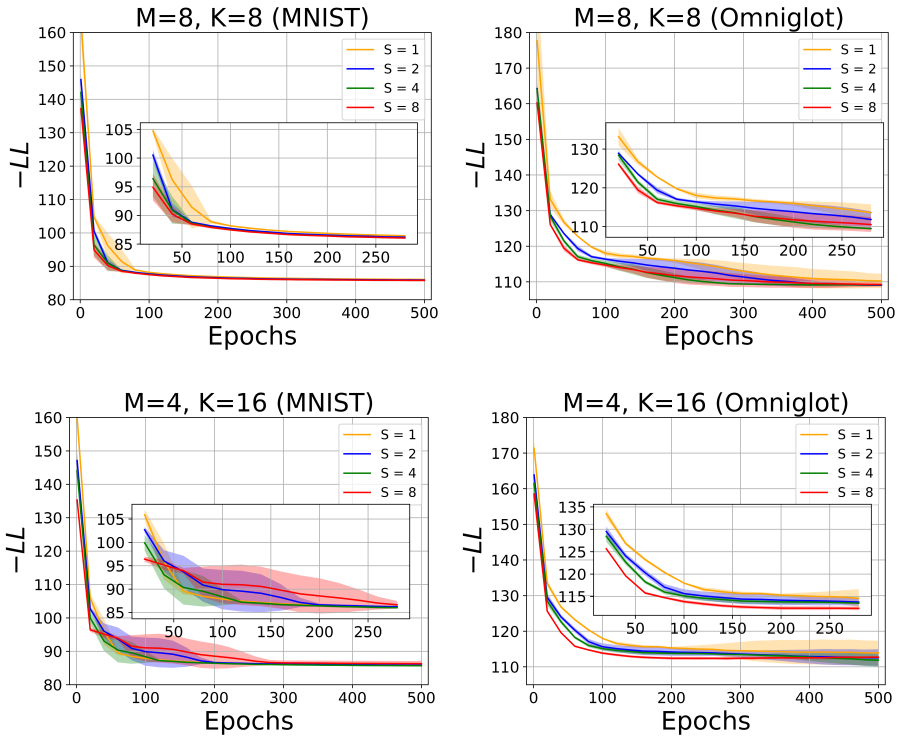

Figure 7: Evaluation of our estimator's convergence on the test sets of MNIST (left column) and Omniglot (right column) over training epochs when the encoding and decoding networks share the same network structure.

10 Additional Results

10.1 Analysis of symmetric encoding/decoding network structures

We analyze the case when both the encoding networks and the decoding networks share the same network structure, creating a symmetric VAE architecture. Figure 7 shows the convergence of our estimator across the architecture sample sizes $S = \{1, 2, 4, 8\}$ for each $(M, K) = \{(8, 8), (4, 16)\}$ setting similar to section 8. We find that increasing architecture sample size S leads to faster convergence which is consistent with section 8.

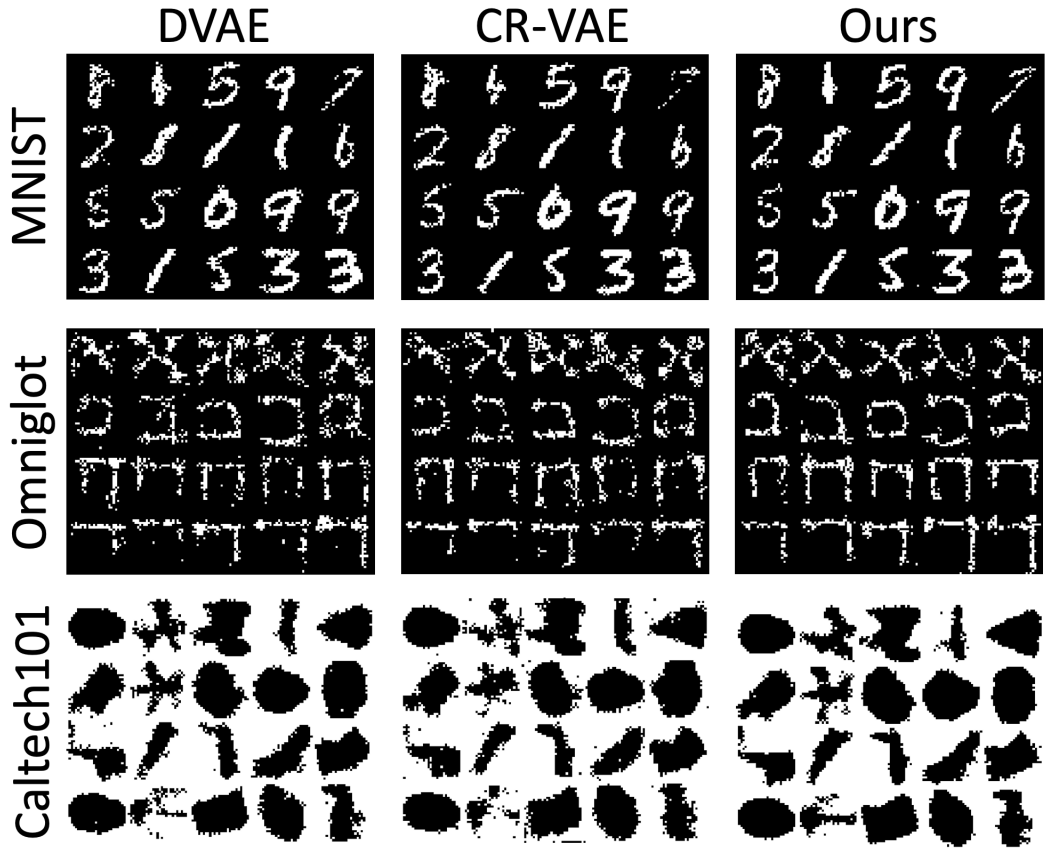

Figure 8: The reconstructions of the original images for the MNIST, Omniglot and Caltech101 datasets for DVAE, CR-VAE, and our method. The overall reconstruction quality of our method is better.

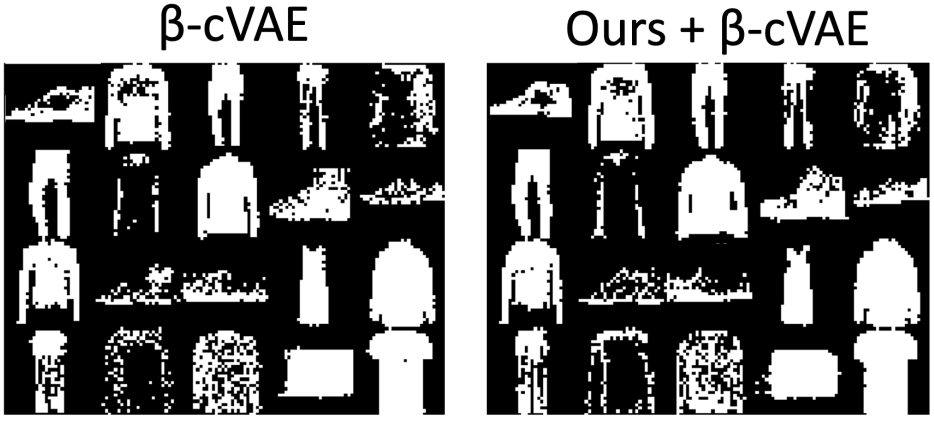

Figure 9: The FashionMNIST reconstructed samples by β-cVAE with and without our method.

10.2 Visualization of reconstructed samples

We visualize reconstructed image samples in Figure 8 and 9. Our method yield the best quality images, consistent with the quantitative analysis in the papers.

10.3 Downstream classification

We perform downstream classification tasks to evaluate the quality of the latent representation $\mathbf{h}$ obtained by our VAE architecture inference framework. We evaluate the baseline regularization methods and our method from Table 2 in the paper. For the baseline methods, we select the network depth that has the best negative log-likelihood ($-LL$) performance. For our method, we set the truncation $T = 25$. We train the VAE models with MNIST and Caltech101 datasets so that each one is capable of reconstructing the inputs. We then evaluate the latent representations obtained from these models by training a support vector machine (SVM) with the means of the learned latent variable distributions over the latent space for all the training examples. We use RBF kernels for the SVM. To train and test the classifier, we follow the same training/test splits for MNIST and Caltech101 as described in Table 1.

Table 10: Performance for the downstream classification tasks using the learned latent representations. We report the classification accuracy in terms of percentage. On both MNIST and Caltech101, our method combined with CR-VAE achieves the best performance.

	MNIST	Caltech101
MIWAE	97.26±0.10	70.89±0.06
DVAE	97.54±0.08	71.99±0.19
CR-VAE	98.18±0.04	72.28±0.35
BB-VAE	96.34±0.35	63.76±1.11
Ours	98.73±0.07	71.98±0.51
Ours+DVAE	98.76±0.05	72.52±0.27
Ours+CR-VAE	**99.00±0.04**	**73.31±0.29**

Table 10 shows that first by comparing each single method, the latent representations obtained from our method achieves comparable or better performance than the VAE regularization methods. Note that our method achieves the performance without any computation overhead to fine-tune or pre-determine the network architectures as other methods. Instead, we jointly infer both the VAE architecture and the latent variables. As Lemma 1 (in the paper) suggests our variational distribution over the latent variable $\mathbf{h}$ becomes a highly flexible semi-implicit distribution by marginalizing the architecture variable $\mathbf{Z}$ out and leads to expressive latent representations. Although CR-VAE achieves the best performance on Caltech101, it uses a larger number of training examples by augmenting the training datasets with additional transformed examples. Furthermore, when we combine our architecture inference framework with CR-VAE, we achieve the overall best performance.

10.4 Experiment with CIFAR10

We evaluated NVAE on the real-world CIFAR10 dataset, where the output from the decoder is a discrete logistic distribution. Table 11 and Figure 10 shows the metrics and reconstructions of the two methods.

Table 11: Reconstruction loss and KL divergence of NVAE with and withour our framework on the CIFAR10 dataset.

Methods	Reconstruction Loss	KL
NVAE	**13313±09**	150.12±03
Ours+NVAE	13600±30	**171.50±00**

11 Concrete Bernoulli Distribution

$$\text{ConBer}(z_{ot}|\pi_t) = \tau \frac{\pi_t (z_{ot})^{-\tau-1}(1-\pi_t)(1-z_{ot})^{-\tau-1}}{(\pi_t (z_{ot})^{-\tau} + (1-\pi_t)(1-z_{ot})^{-\tau})^2}$$

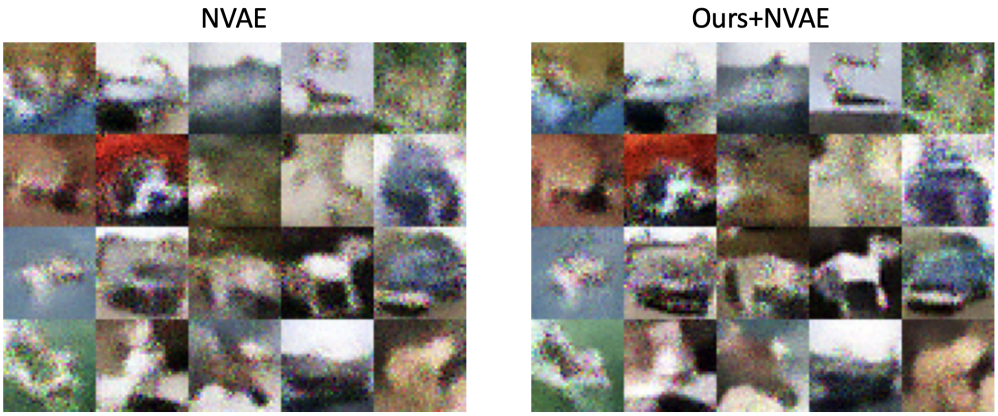

Figure 10: Reconstructions of the original CIFAR10 images from NVAE with and without our framework.

where τ controls the distribution smoothness. We thus generate the VAE architecture samples $\mathbf{Z}_s$ by first sampling from a logistic distribution, and then putting the samples through a logistic function:

$$z_{ot} = \frac{1}{1 + \exp(-\tau^{-1}(\log \pi_t - \log(1 - \pi_t) + \epsilon))}$$
$$\text{where, } \epsilon \sim \text{Logistic}(0, 1)$$

References

[1] Thomas M Cover and Joy A Thomas. Elements of information theory. 2012. *Google Scholar Google Scholar Digital Library Digital Library*.

[2] Andrew J Kurdila and Michael Zabarankin. *Convex functional analysis*. Springer Science & Business Media, 2005.

[3] Mingzhang Yin and Mingyuan Zhou. Semi-implicit variational inference. In *International Conference on Machine Learning (ICML)*, pages 5660–5669. PMLR, 2018.

[4] Yuri Burda, Roger B Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[5] Benjamin Marlin, Kevin Swersky, Bo Chen, and Nando Freitas. Inductive principles for restricted boltzmann machine learning. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 9, pages 509–516. PMLR, 13–15 May 2010.

[6] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[7] Samarth Sinha and Adji Bousso Dieng. Consistency regularization for variational auto-encoders. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

[8] Rachit Singh, Jeffrey Ling, and Finale Doshi-Velez. Structured variational autoencoders for the beta-bernoulli process. In *NIPS 2017 Workshop on Advances in Approximate Bayesian Inference*, 2017.

[9] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[10] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163, 2000.

[11] C Lee Giles, Kurt D Bollacker, and Steve Lawrence. CiteSeer: An automatic citation indexing system. In *Proceedings of the third ACM Conference on Digital Libraries*, pages 89–98, 1998.

[12] Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[13] Adji B. Dieng, Yoon Kim, Alexander M. Rush, and David M. Blei. Avoiding latent variable collapse with generative skip models. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics (AISTATS)*, Proceedings of Machine Learning Research, pages 2397–2405, 2019.

[14] Casper Kaae S, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 29, 2016.

[15] Arash Vahdat and Jan Kautz. NVAE: A deep hierarchical variational autoencoder. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 19667–19679, 2020.

[16] Tom Rainforth, Adam Kosiorek, Tuan Anh Le, Chris Maddison, Maximilian Igl, Frank Wood, and Yee Whye Teh. Tighter variational bounds are not necessarily better. In *International Conference on Machine Learning (ICML)*, pages 4277–4285. PMLR, 2018.