APPENDIX

A KULLBACK-LEIBLER DIVERGENCE BETWEEN TWO DIRICHLET DISTRIBUTION

For two Dirichlet distributions $\mathcal{D}ir(\pi|\alpha_\theta(x))$ and $\mathcal{D}ir(\pi|1,\dots,1)$ over K-dimensional probability π, the following equality holds

$$\begin{aligned}&\mathcal{D}_{\text{KL}}(\mathcal{D}ir(\pi|\alpha_\theta(x))||\mathcal{D}ir(\pi|1,\dots,1))\\&= \log\left(\frac{\Gamma\big(\sum_k \alpha_\theta^k(x)\big)}{\Gamma(K)\prod_k \Gamma(\alpha_\theta^k(x))}\right) + \sum_{k=1}^{K}(\alpha_\theta^k(x)-1)\Big(\psi(\alpha_\theta^k(x)) - \psi\big(\sum_k \alpha_\theta^k(x)\big)\Big).\end{aligned}$$

where $\Gamma(.)$ and $\psi(\alpha) := \frac{\mathrm{d}}{\mathrm{d}\alpha}\log\Gamma(\alpha)$ are the *gamma* and *digamma* functions, respectively.

B DERIVATION DETAILS

As we define in Section 4, our forward model is:

$$p(\pi) = \mathcal{D}ir(\pi|1,\dots,1), \tag{9}$$
$$Pr(z|\pi) = \mathcal{C}at(z|\pi), \tag{10}$$
$$p(x|\mathcal{M}, z=k) = \mathcal{N}(x|\mathcal{D}_\phi(\mathcal{M}, z), \sigma^2 I). \tag{11}$$

and the approximate posterior is:

$$q(\pi, z|x) = \mathcal{C}at(z|\pi)\mathcal{D}ir(\pi|\alpha_\theta^1(x),\dots,\alpha_\theta^K(x)). \tag{12}$$

We derive the ELBO to be maximized during the training as:

$$\log p(x|\mathcal{M},\theta,\phi) = \log \mathbb{E}_{q(\pi,z|x)}\left[\frac{p(x|\mathcal{M},z)Pr(z|\pi)p(\pi)}{q(\pi,z|x)}\right] \tag{13}$$
$$= \log \mathbb{E}_{q(\pi,z|x)}\left[\frac{p(x|\mathcal{M},z)\cancel{Pr(z|\pi)}p(\pi)}{\cancel{Pr(z|\pi)}q(\pi|x)}\right] \tag{14}$$
$$\geq \mathbb{E}_{Pr(z|\pi)}\left[\mathbb{E}_{q(\pi|x)}[\log p(x|\mathcal{M},z)] - \mathbb{E}_{q(\pi|x)}[\log q(\pi|x)/p(\pi)]\right] \tag{15}$$
$$= \mathbb{E}_{Pr(z|\pi)}\left[\mathbb{E}_{q(\pi|x)}[\log p(x|\mathcal{M},z)]\right] - \mathcal{D}_{\text{KL}}(q(\pi|x)||p(\pi)). \tag{16}$$

The codebook can be viewed as $\mathcal{M} = [m_1,\dots,m_K]$ where m_ks are the codebook embeddings. The input of the decoder $z_q(x) = \mathcal{M}_k = z * \mathcal{M}$ consists of the codebook embeddings m_ks as shown in Figure 1. $\mathcal{M}_k$ is retrieved using the indices zs sampled as $z \sim \mathcal{C}at(z|\pi)$. Therefore, we can use $\mathcal{D}_\phi(\mathcal{M}_k)$ instead of $\mathcal{D}_\phi(\mathcal{M}, z)$ for the remaining of the derivations.

The first term in Equation 16 can be further derived as:

$$\mathbb{E}_{Pr(z|\pi)}\left[\mathbb{E}_{q(\pi|x)}[\log p(x|\mathcal{M},z)]\right] = -\frac{1}{2}\log\sigma^2 - \frac{1}{2\sigma^2}\sum_k q(\pi_k|x)(x-\mathcal{D}_\phi(\mathcal{M}_k))^2 \tag{17}$$

$$\propto \mathbb{E}_{q(\pi|x)}\left[\sum_k \pi_k (x-\mathcal{D}_\phi(\mathcal{M}_k))^2\right] \tag{18}$$

$$= \mathbb{E}_{q(\pi|x)}\left[\sum_k \pi_k (x^T x - 2x^T\mathcal{D}_\phi(\mathcal{M}_k) + \mathcal{D}_\phi^T(\mathcal{M}_k)\mathcal{D}_\phi(\mathcal{M}_k))\right] \tag{19}$$

$$= x^T x - 2x^T\mathbb{E}_{q(\pi|x)}\left[\sum_k \pi_k \mathcal{D}_\phi(\mathcal{M}_k)\right] + \mathbb{E}_{q(\pi|x)}\left[\sum_k \pi_k \mathcal{D}_\phi^T(\mathcal{M}_k)\mathcal{D}_\phi(\mathcal{M}_k)\right] \tag{20}$$

$$= x^T x - 2x^T\sum_k \mathbb{E}_{q(\pi|x)}\left[\pi_k\right]\mathcal{D}_\phi(\mathcal{M}_k) + \mathbb{E}_{q(\pi|x)}\left[\sum_k \pi_k \mathcal{D}_\phi^T(\mathcal{M}_k)\mathcal{D}_\phi(\mathcal{M}_k)\right] \tag{21}$$

$$= x^T x - 2x^T\sum_k \frac{\alpha_\theta^k(x)}{S}\mathcal{D}_\phi\left(\mathcal{M}_k\right) + \mathbb{E}_{q(\pi|x)}\left[\sum_k \pi_k \mathcal{D}_\phi^T(\mathcal{M}_k)\mathcal{D}_\phi(\mathcal{M}_k)\right] \tag{22}$$

$$= x^T x - 2x^T\sum_k \frac{\alpha_\theta^k(x)}{S}\mathcal{D}_\phi\left(\mathcal{M}_k\right) + \sum_k \mathbb{E}_{q(\pi|x)}\left[\pi_k\right]\mathcal{D}_\phi^T(\mathcal{M}_k)\mathcal{D}_\phi(\mathcal{M}_k) \tag{23}$$

$$= x^T x - 2x^T\sum_k \frac{\alpha_\theta^k(x)}{S}\mathcal{D}_\phi\left(\mathcal{M}_k\right) + \sum_k \frac{\alpha_\theta^k(x)}{S}\mathcal{D}_\phi^T(\mathcal{M}_k)\mathcal{D}_\phi(\mathcal{M}_k) \tag{24}$$

$$= \mathbb{E}_{z\sim\pi}\left[(x-\mathcal{D}_\phi(\mathcal{M}_k))^T(x-\mathcal{D}_\phi(\mathcal{M}_k))\right] \tag{25}$$

$$= \mathbb{E}_{z\sim\pi}\left[||x-\mathcal{D}_\phi(\mathcal{M}_k)||_2^2\right]. \tag{26}$$

We define $S = \sum_k \alpha_\theta^k(x)$, and $\pi = [\pi_1, \ldots, \pi_K]$ with $\pi_k = \alpha_\theta^k(x)/S$ since the mean of the concentration parameters of a Dirichlet distribution can be taken as an estimate of the probabilities (Sensoy et al., 2018). This approach is theoretically correct and practically proper since sequential sampling from the Dirichlet distribution and the Categorical distribution causes an instability in the training. Then, we can get samples $z \sim \mathcal{C}at(z|\pi)$ using Gumbel-Softmax trick in order to obtain the reconstruction loss.

C Training and Inference Procedures of Evidential dVAE

The training and the inference procedure of EdVAE are described in Algorithms 1 and 2, respectively. t denotes the index of the training iterations, and b denotes the batch index in the inference. RelaxedOneHotCategorical(.) distribution is differentiable, and the samples from this distribution are soft one-hot vectors. On the other hand, Categorical(.) distribution is not differentiable which is not required during the inference. The samples from this distribution are hard one-hot vectors which indicate one-to-one quantizations.

D Experimental Details

We use PyTorch framework in our implementation. We train all of the models using a single NVIDIA A100 GPU. We use some common choices for all of the models and all of the datasets in our experiments. We train all of the models for 150K iterations on CIFAR10, CelebA, and LSUN Church datasets, using the batch size of 128. We use the Adam optimizer Kingma & Ba (2014) with an initial learning rate $1e^{-3}$, and follow the cosine annealing schedule to anneal the learning rate from $1e^{-3}$ to $1.25e^{-6}$ over the first 50K iterations.

For GS-VQ-VAE, dVAE, and EdVAE, we follow the same cosine annealing schedule for the β coefficient of the KL divergence term as in Equation 8. We anneal the β coefficient starting from 0 over the first 5K iterations. Based on the model and the dataset, the upper bound for the β coefficient

Algorithm 1 Training algorithm of EdVAE

Input: Dataset $\mathbf{x}_{\text{train}}$
Output: Reconstructed $\mathbf{x}_{\text{train}}$
Initialize the encoder $\mathcal{E}_\theta^{[0]}$, the decoder $\mathcal{D}_\phi^{[0]}$, the codebook $\mathcal{M}^{[0]}$,
and the temperature parameter $\tau^{[0]} = 1.0$
for $t = 1, 2, \ldots, T$ **do**
  $x \leftarrow$ Random minibatch from $\mathbf{x}_{\text{train}}$
  $z_e(x) \leftarrow \mathcal{E}_\theta^{[t-1]}(x)$
  $\alpha_\theta \leftarrow e^{z_e(x)} + 1$
  $\pi \sim \mathcal{D}ir(\pi|\alpha_\theta)$
  $z \sim \text{RelaxedOneHotCategorical}(\text{temperature} = \tau^{[t-1]}, \text{probs} = \pi)$
  $\hat{x} \leftarrow \mathcal{D}_\phi^{[t-1]}(\mathcal{M}, z)$
  $g \leftarrow \nabla_{\mathcal{M},\theta,\phi}\mathcal{L}(\mathcal{M}^{[t-1]}, \theta^{[t-1]}, \phi^{[t-1]})$
    with sampled x and $\hat{x}$
  $\mathcal{M}^{[t]}, \theta^{[t]}, \phi^{[t]} \leftarrow$ Update parameters using g
  $\tau^{[t]} \leftarrow \text{CosineAnneal}(\tau^{[t-1]}, t)$
end for

Algorithm 2 Inference algorithm of EdVAE

Input: Dataset $\mathbf{x}_{\text{test}}$
Output: Reconstructed $\mathbf{x}_{\text{test}}$
Freeze the parameters of the trained encoder $\mathcal{E}_\theta$, the trained decoder $\mathcal{D}_\phi$,
and the trained codebook $\mathcal{M}$
for $b = 1, 2, \ldots, B$ **do**
  $x_b \leftarrow$ Minibatch from $\mathbf{x}_{\text{test}}$
  $z_e(x) \leftarrow \mathcal{E}_\theta(x)$
  $\alpha_\theta \leftarrow e^{z_e(x)} + 1$
  $\pi \sim \mathcal{D}ir(\pi|\alpha_\theta)$
  $z \sim \text{Categorical}(\text{probs} = \pi)$
  $\hat{x} \leftarrow \mathcal{D}_\phi(\mathcal{M}, z)$
end for

varies as the KL terms are different in different models, and the β coefficient decides the reconstruction vs KL term tradeoff. We also follow the temperature annealing schedule $\tau = \exp(-10^{-5}.t)$ for the Gumbel-Softmax where τ denotes the temperature, and t denotes the global training step. We initialize the codebook embeddings of these models using a Gaussian normal distribution. In EdVAE, we clamp the encoder's output $z_e(x)$ to be maximum 20 before converting it to the α_θ parameters for the training stability. We observe that after we obtain the training stability, we clamp too few variables which does not affect the integrity of the latent variables. We provide detailed analysis for the effects of logits clamping in E.3.

We rerun all of our experiments using the seed values of *42*, *1773*, and *1*. Based on the architecture and the loss function, we follow specific procedures and describe them below:

VQ-VAE-EMA: We use the same architecture and the hyperparameters as suggested in Oord et al. (2017). We set the β coefficient for VQ-VAE-EMA's loss to 0.25, and the weight decay parameter for the EMA to 0.99. We initialize the codebook embeddings using a uniform distribution as in Oord et al. (2017).

GS-VQ-VAE: We use the same architecture and hyperparameters as in VQ-VAE-EMA for a fair comparison. The difference between the GS-VQ-VAE and the VQ-VAE-EMA is the quantization where the GS-VQ-VAE uses the Gumbel-Softmax operation parameterized with the negative Euclidean distances between the encoder's output and the codebook. The upper bound for the annealed β coefficient of the KL divergence is set to $5e^{-6}$ for all datasets.

SQ-VAE: We use the official implementation (Takida et al., 2022a) of (Takida et al., 2022b). For all datasets, we use Gaussian SQ-VAE (I) architecture and hyperparameters as described in (Takida et al., 2022b).

VQ-STE++: We use the official implementation (Huh, 2022) of (Huh et al., 2023), and follow the same training procedure described in (Huh et al., 2023).

dVAE: The upper bound for the annealed β coefficient of the KL divergence is set to $5e^{-5}$ for all datasets.

EdVAE: For CIFAR10, we set the upper bound for the annealed β coefficient of the KL divergence to $5e^{-7}$ while we use $1e^{-7}$ for the remaining datasets.

D.1 Datasets

CIFAR10: CIFAR10 consists of 60,000 32×32 RGB images in 10 classes. Each class consists the same amount of images. We use the default train/test split of the dataset that the train split consists of 50,000 images and the test split consists of 10,000 images.

CelebA: CelebA dataset consists of more than 200,000 celebrity images, and each image is annotated with 40 different attribute such as gender, hair color, facial hair etc. We use the default train/val/test split of the dataset. As preprocessing, we perform center cropping of 140×140, and resize the cropped images to 64×64 using bilinear interpolation.

LSUN Church: LSUN Church consists of 126,000 256×256 RGB images of various churches. It is a part of LSUN dataset which includes different indoor and outdoor scene categories. We use the default train/test split of the dataset. We resize the images to 128×128 resolution using bilinear interpolation, and use the resized images.

D.2 Model Description

In this section, we describe the architecture of dVAE and EdVAE. The common building blocks used in the encoders and the decoders are given in Table 4. For the following architectures, w and h denote the *width* and the *height* of the images. For CIFAR10 $w = h = 32$, for CelebA $w = h = 64$, and for LSUN Church $w = h = 128$. We use a codebook $\mathcal{M} \in R^{512 \times 16}$ for all of our experiments.

Table 4: Notations of network layers used on all models.

Notation	Description
$\text{Conv}_n^{(7\times7)}$	2D Convolutional layer (out_channel= n, kernel= 7, stride= 1, padding= 3)
$\text{Conv}_n^{(4\times4)}$	2D Convolutional layer (out_channel= n, kernel= 4, stride= 2, padding= 1)
$\text{Conv}_n^{(3\times3)}$	2D Convolutional layer (out_channel= n, kernel= 3, stride= 1, padding= 1)
$\text{Conv}_n^{(1\times1)}$	2D Convolutional layer (out_channel= n, kernel= 1, stride= 1, padding= 1)
MaxPool	2D Max pooling layer (kernel_size= 2)
Upsample	2D upsampling layer (scale_factor= 2)
EncResBlock_n	$3 \times (\text{ReLU} \rightarrow \text{Conv}_n^{(3\times3)}) \rightarrow \text{ReLU} \rightarrow \text{Conv}_n^{(1\times1)}$ + identity mapping
DecResBlock_n	$\text{ReLU} \rightarrow \text{Conv}_n^{(1\times1)} \rightarrow 3 \times (\text{ReLU} \rightarrow \text{Conv}_n^{(3\times3)})$ + identity mapping

As the encoder of dVAE and EdVAE return a distribution over the codebook, the last dimensions of the encoders' outputs $z_e(x)$s are all equal to 512 for all datasets. After the quantization of $z_e(x)$s, the last dimensions of the decoders' inputs $z_q(x)$s are all equal to 16 for all datasets.

Encoder

$$
\begin{aligned}
x \in \mathbb{R}^{w\times h\times 3} &\rightarrow \text{Conv}_n^{(\text{kw}\times\text{kw})} && \text{shape: } (w \times h \times n)\\
&\rightarrow [\text{EncResBlock}_n]_2 && \text{shape: } (w \times h \times n)\\
&\rightarrow \text{MaxPool} && \text{shape: } (w/2 \times h/2 \times n)\\
&\rightarrow [\text{EncResBlock}_{2*n}]_2 && \text{shape: } (w/2 \times h/2 \times 2*n)\\
&\rightarrow \text{MaxPool} && \text{shape: } (w/4 \times h/4 \times 2*n)\\
&\rightarrow [\text{EncResBlock}_{4*n}]_2 && \text{shape: } (w/4 \times h/4 \times 4*n)\\
&\rightarrow \text{Conv}_{4*n}^{(1\times1)} && \text{shape: } (w/4 \times h/4 \times 4*n)
\end{aligned}
$$

Decoder

$$
\begin{aligned}
z_q(x) \in \mathbb{R}^{w/4\times h/4\times 16} &\rightarrow [\text{DecResBlock}_{4*n}]_2 && \text{shape: } (w/4 \times h/4 \times 4*n)\\
&\rightarrow \text{UpSample} && \text{shape: } (w/2 \times h/2 \times 4*n)\\
&\rightarrow [\text{DecResBlock}_{2*n}]_2 && \text{shape: } (w/2 \times h/2 \times 2*n)\\
&\rightarrow \text{UpSample} && \text{shape: } (w \times h \times 2*n)\\
&\rightarrow [\text{DecResBlock}_n]_2 && \text{shape: } (w \times h \times n)\\
&\rightarrow \text{ReLU} \rightarrow \text{Conv}_3^{(1\times1)} && \text{shape: } (w \times h \times 3)
\end{aligned}
$$

where n is equal to 128 for all datasets, and $4*n$ is equal to the number of the codebook embeddings. *kw* denotes the kernel size of the convolution layer. For CIFAR10 and CelebA kw $= 3$, for LSUN Church kw $= 7$.

E Additional Experiments and Results

E.1 Higher Temperature Usage with dVAE

We use temperature values of 2 and 5 in our current experiments, and observe that perplexity value obtained as 190 for CIFAR10 dataset slightly decreases to 170 and 180, respectively. Similarly, for CelebA dataset, perplexity value obtained as 255 decreases to 217 using temperature 2, and increases to 296 using temperature 5. The performance of dVAE is sensitive to temperature hyperparameter, and perplexity does not always increase with a high temperature. Therefore, using higher temperature is not an appropriate solution.

E.2 Uncertainty vs Perplexity

We anticipate that introducing uncertainty awareness will enhance codebook usage, addressing limitations in dVAE caused by the softmax operation. Our intuition is rooted in the definition of

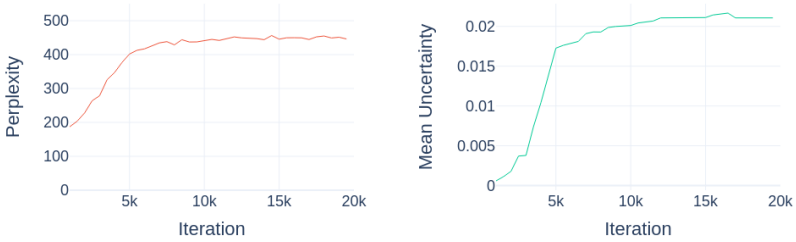


Figure 6: EdVAE training on CIFAR10: perplexity values increase during the training due to the increase in uncertainty values.

Table 5: Test perplexities using various max clamping values.

Max Clamping Value	CIFAR10	CelebA	LSUN Church
10	351	319	363
15	421	376	375
20	**425**	**386**	**393**
25	411	1	1
30	1	1	1

codebook collapse—new elements are introduced when existing ones fail to explain observations. In order to validate our intuition, we monitor the training of CIFAR10 and present an interval of the training until saturation in Figure 6.

We observe a correlation between the perplexity values and the uncertainty values during the training. The trend of perplexity values perfectly matches the trend of uncertainty values. This correlation emphasizes that our model dynamically adjusts codebook usage based on its uncertainty, preventing codebook collapse by utilizing embeddings effectively.

E.3 Effects of Logits Clamping

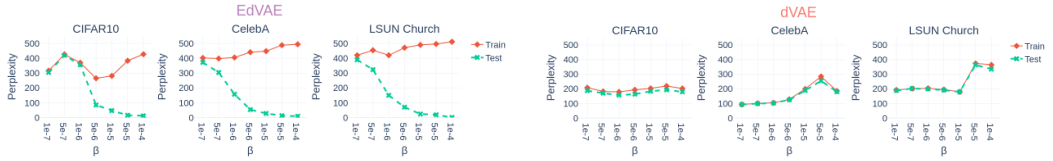
To obtain the parameters of the Dirichlet distribution, αs, we follow a common approach of logits clamping to stabilize the training since the exponential of logits might be really large. We conduct an ablation study to observe the effects of logits clamping, and present our findings in Table 5.

We observe that clamping the logits with smaller max values clamps some of the values in logits, and limits the range of positive values logits can have. This situation limits the representativeness of the logits, and leads to lower perplexities. On the other hand, using larger max values for clamping causes divergence in the training as the exponential of logits gets large, and the model cannot be trained. Therefore, the logits should be clamped eventually with proper values. If a proper max value can be selected, clamping acts as a regularizer at the beginning of the training, and the encoder naturally outputs logits with no values greater than the max clamping value after a few iterations. If the training is already stabilized, the max clamping value does not affect the performance dramatically as both 15 and 20 lead to similar results. Therefore, using 20 as the max value for all datasets can be a mutual design choice.

E.4 Effects of β Coefficient

As the β coefficient is an important parameter for the optimization, we conduct additional experiments to observe its effects on the performance. We perform several experiments by changing the β coefficient within [1e-7, 1e-4]. We repeat our experiments for dVAE and EdVAE using all of the datasets, and present our findings in Figure 7.

We observe that our method is more sensitive to β coefficient than dVAE, and EdVAE diverges when the β coefficient increases for all datasets. We think that the key factor to this sensitivity is



(a) EdVAE performance on training and testing.

(b) dVAE performance on training and testing.

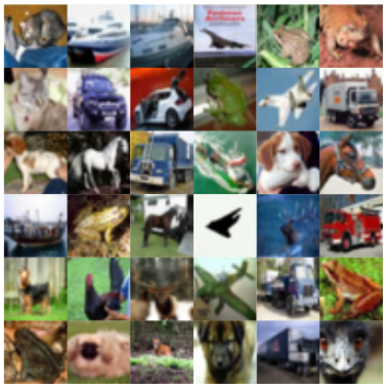
Figure 7: Effects of β coefficient to the performance.

the complexity introduced by the KL distance between our newly introduced posterior and prior, compared to the KL distance in dVAE. Therefore, fine-tuning the β coefficient emerges. Even though our original KL term brings some sensitivity to the training and it requires a hyper-parameter tuning like most of the AI models, its contribution to the performance is non-negligible and essential.

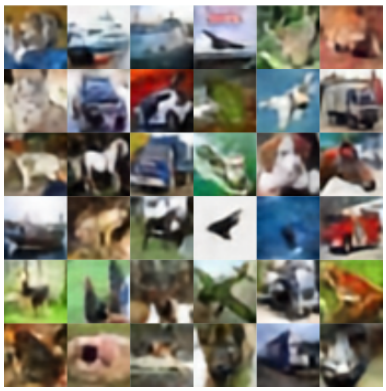
Besides, the best performing β coefficient for CIFAR10 dataset is slightly higher than the best performing β coefficient of CelebA and LSUN Church datasets. Our intuition for this difference is that, reconstructing images with lower resolution as in CIFAR10 is less challenging than reconstructing images with higher resolution as in CelebA and LSUN Church. Therefore, increasing the β coefficient from 1e-7 to 5e-7 improves the performance in CIFAR10 without hurting the reconstruction vs KL term tradeoff. On the other hand, 1e-7 to 5e-7 conversion slightly decreases the performance in CelebA and LSUN Church datasets since the reconstruction of the higher resolution images affects the reconstruction vs KL term tradeoff.

E.5 Additional Results

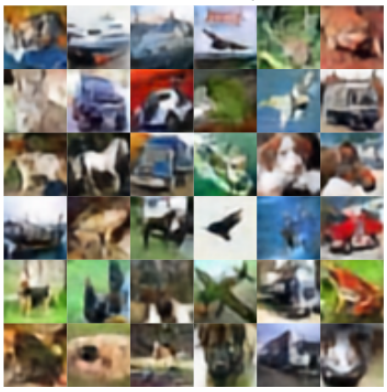
Further reconstructed samples from CIFAR10, CelebA, and LSUN Church datasets are given in Figure 8, Figure 9 and Figure 10, respectively. Moreover, the generated samples using the discrete latents of all models for CelebA and LSUN Church datasets are given in Figure 11 and Figure 12, respectively.



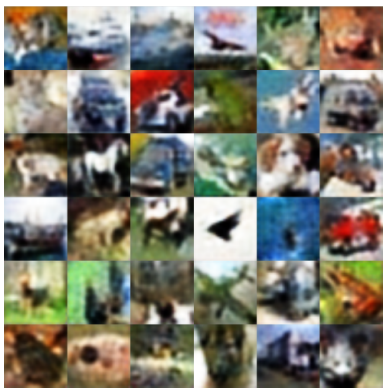
(a) Source images



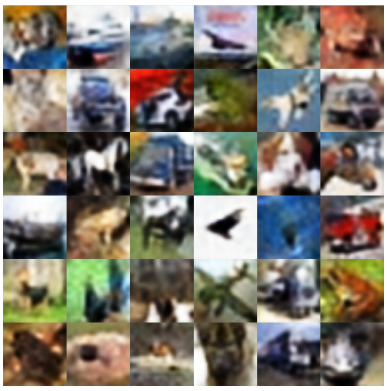
(b) VQ-VAE-EMA



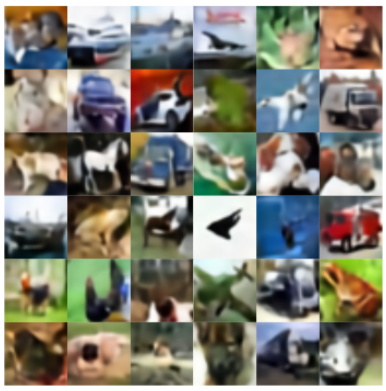
(c) GS-VQ-VAE



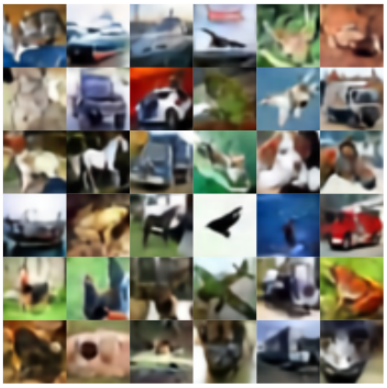
(d) SQ-VAE



(e) VQ-STE++



(f) dVAE



(g) EdVAE

Figure 8: Reconstructed samples from the CIFAR10 dataset.

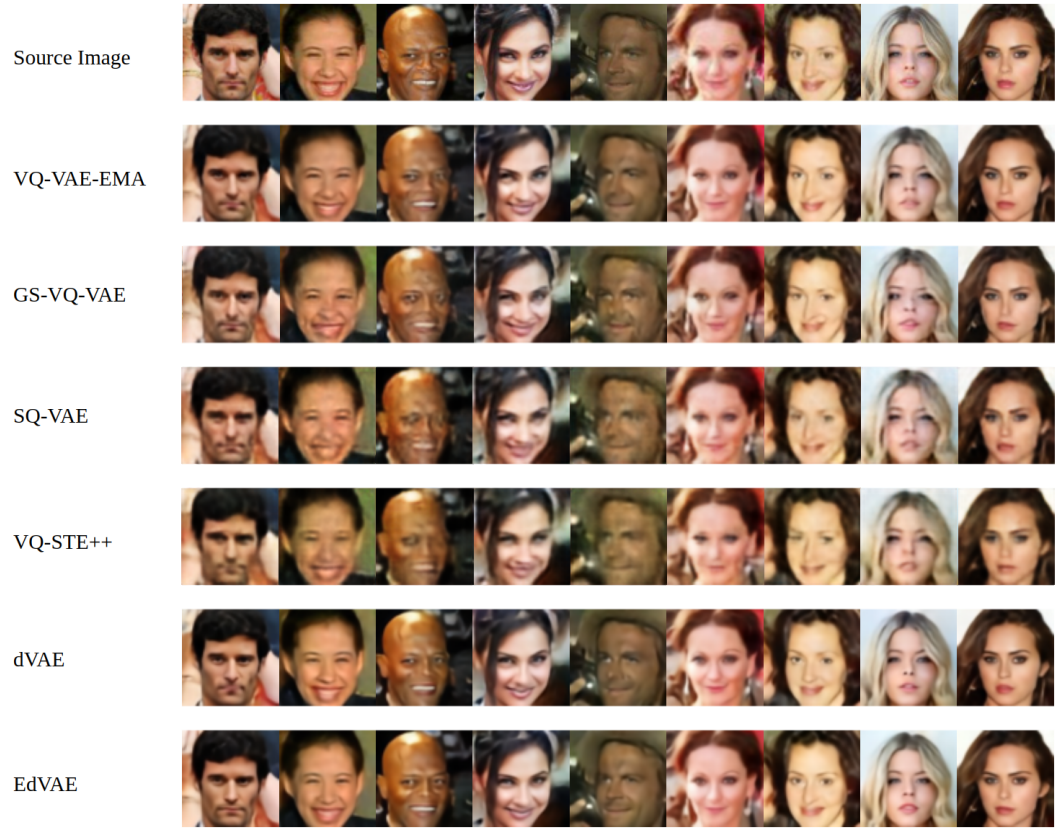


Figure 9: Reconstructed samples from CelebA dataset using EdVAE and other models.

Figure 10: Reconstructed samples from LSUN Church dataset using EdVAE and other models.

(a) VQ-VAE-EMA

(b) GS-VQ-VAE

(c) SQ-VAE

(d) VQ-STE++

(e) dVAE

(f) EdVAE

Figure 11: Generated samples from the CelebA 64×64 dataset.

(a) VQ-VAE-EMA

(b) GS-VQ-VAE

(c) SQ-VAE

(d) VQ-STE++

(e) dVAE

(f) EdVAE

Figure 12: Generated samples from the LSUN Church 128×128 dataset.