॥ त्वं ज्ञानमयो विज्ञानमयोऽसि ॥

Indian Institute of Technology Jodhpur

Course :

CSL7590 Deep Learning

Assignment-5

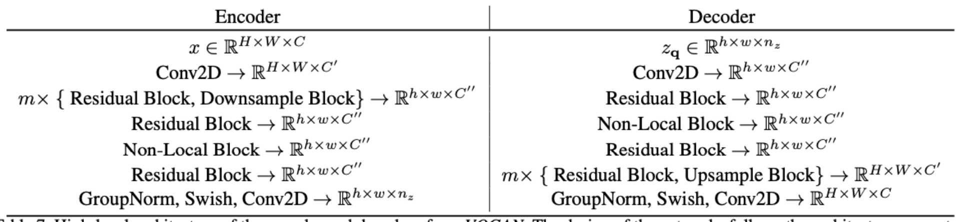
Group Members:

Anil Kumar Vishnoi(M21MA202)
Mitesh Kumar(M23MAC004)
Sougata Moi (M23MAC008)

1 Dataset Information:

The ISIC dataset comprises training and test images for dermatology-related classification tasks, aiding in the development and evaluation of machine learning algorithms for skin lesion analysis. It serves as a valuable resource for researchers and practitioners in the field of medical image analysis and healthcare technology.

2 VQ-GAN Architecture-



Encoder	Decoder
$x \in \mathbb{R}^{H\times W\times C}$	$z_{\mathbf{q}} \in \mathbb{R}^{h\times w\times n_z}$
Conv2D $\rightarrow \mathbb{R}^{H\times W\times C'}$	Conv2D $\rightarrow \mathbb{R}^{h\times w\times C''}$
$m\times$ { Residual Block, Downsample Block} $\rightarrow \mathbb{R}^{h\times w\times C''}$	Residual Block $\rightarrow \mathbb{R}^{h\times w\times C''}$
Residual Block $\rightarrow \mathbb{R}^{h\times w\times C''}$	Non-Local Block $\rightarrow \mathbb{R}^{h\times w\times C''}$
Non-Local Block $\rightarrow \mathbb{R}^{h\times w\times C''}$	Residual Block $\rightarrow \mathbb{R}^{h\times w\times C''}$
Residual Block $\rightarrow \mathbb{R}^{h\times w\times C''}$	$m\times$ { Residual Block, Upsample Block} $\rightarrow \mathbb{R}^{H\times W\times C'}$
GroupNorm, Swish, Conv2D $\rightarrow \mathbb{R}^{h\times w\times n_z}$	GroupNorm, Swish, Conv2D $\rightarrow \mathbb{R}^{H\times W\times C}$

Table 7. High-level architecture of the encoder and decoder of our *VQGAN*. The design of the networks follows the architecture presented in [25] with no skip-connections. For the discriminator, we use a patch-based model as in [28]. Note that $h = \frac{H}{2^m}$, $w = \frac{W}{2^m}$ and $f = 2^m$.

The VQ-VAE model comprises an encoder and a decoder network, along with a vector quantization module. The encoder network maps the input image to a compressed latent representation, which is quantized using a codebook. The decoder network reconstructs the image from the quantized latent representation. The architecture is designed to leverage self-attention mechanisms and residual connections to improve the model's performance. The overall architecture can be broken down into the following components:

2.1 Encoder

- The encoder consists of a series of DownBlock and MidBlock modules.
- The DownBlock module contains two residual blocks, each with two convolutional layers, followed by a downsampling convolutional layer.
- The MidBlock module incorporates self-attention mechanisms using the MultiheadAttention module from PyTorch. It also includes residual connections and group normalization layers.
- The encoder layers progressively downsample the input image while increasing the number of channels.

2.2 Quantization Module

- The quantization module uses a learnable codebook (an Embedding layer) to quantize the latent representation obtained from the encoder.

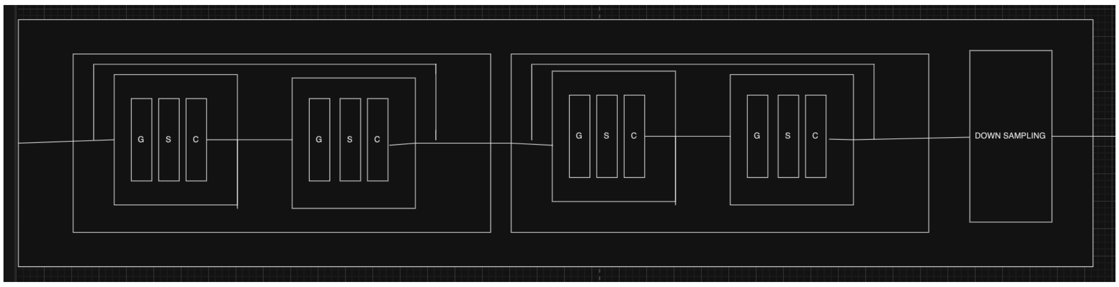
- The quantization process involves finding the nearest codebook vector for each latent vector using the torch.cdist function, which computes the distance between the latent vectors and the codebook vectors.
- The quantization module also computes the commitment loss and the codebook loss, which are used during training to encourage the encoder to produce latent representations close to the codebook vectors.

2.3 Decoder

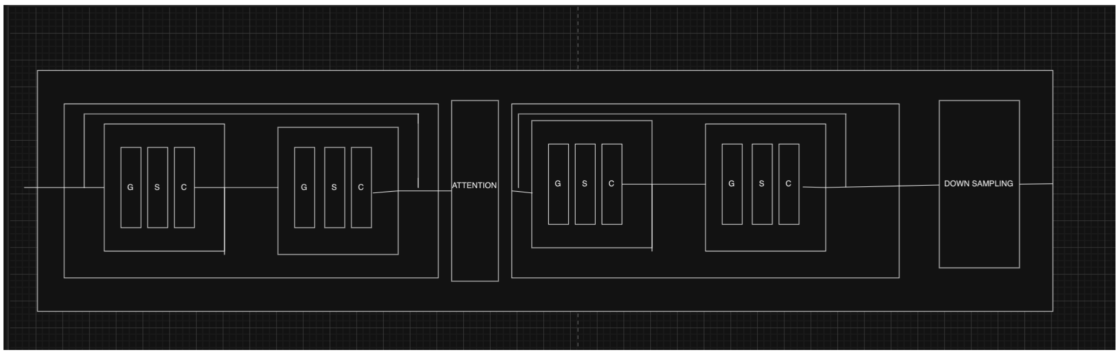
- The decoder mirrors the structure of the encoder but in reverse order.
- It consists of MidBlock and UpBlock modules.
- The UpBlock module contains two residual blocks, each with two convolutional layers, preceded by a upsampling transposed convolutional layer.
- The MidBlock modules in the decoder also incorporate self-attention mechanisms and residual connections.
- The decoder layers progressively upsample the quantized latent representation while decreasing the number of channels, eventually reconstructing the input image.

The details of the DownBlock, MidBlock, and UpBlock modules used in this architecture.

2.4 DownBlock



2.5 MidBlock



2.6 UpBlock

- The UpBlock module is used in the decoder part of the VQ-VAE model.
- It mirrors the structure of the DownBlock but in reverse order. The module starts with a transposed convolutional layer (also known as a deconvolutional layer) with a stride of 2 to upsample the spatial dimensions of the input.

3 Latent Diffusion Model Architecture-

The only difference between VQ-GAN and Latent Diffusion model are

- We are adding time embanding in between 2-conv layer of is rasnet block.
- The output of downblock is concatnate with the upblock after upsampling.

4 Methodology-

4.1 VQGAN

First we train VQGAN based on this loss function-

$\mathcal{L}_{VQ}(E, G, Z) = \|x - \hat{x}\|^2 + \|sg[E(x)] - z_{\mathbf{q}}\|_2^2 + \|sg\left[z_{\mathbf{q}}\right] - E(x)\|_2^2 + perceptual_loss(x, \hat{x})$

- The first term, $\|x - \hat{x}\|^2$, is the reconstruction loss, it checks how well our network was able to approximate (via $\hat{x}$) our input x when given only its quantized version $z_{\mathbf{q}}$.
- The second term, $\|sg[E(x)] - z_{\mathbf{q}}\|_2^2$, optimizes our embeddings. The operation sg stands for "stop-gradient." It is an identity function during forward pass, and has zero partial derivatives, thus constraining it to be constant.
- The third term, $\|sg\left[z_{\mathbf{q}}\right] - E(x)\|_2^2$, is called the commitment loss. It ensures that the encoder E commits to a particular representation of the image.
- The last term is the perceptual loss

Finally, training VQGAN to obtain the optimal compression model $Q^\star$ becomes a matter of combining the two losses from the autoencoder, $\mathcal{L}_{VQ}$, and the GAN, $\mathcal{L}_{GAN}$:

$$Q^\star = \arg \min_{E,G,Z} \max_{D} \mathbb{E}_{xp(x)} \left[\mathcal{L}_{VQ}(E, G, Z) + \lambda \mathcal{L}_{GAN}(N, D)\right]$$

where Z is the codebook and λ is the adaptive weight.
Generator starts train after 8 epochs.

4.2 Stable Diffusion

```
Algorithm 1 Training
1: repeat
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \ldots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
   $\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta \left( \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon}, t \right) \right\|^2$
```

```
Algorithm 2 Sampling
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: for $t = T, \ldots, 1$ do
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t \boldsymbol{\epsilon}_\theta}} (\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: end for
6: return $\mathbf{x}_0$ height
```

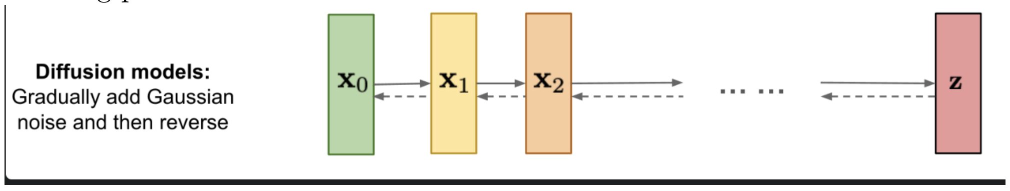
- Forward diffusion process- Given a data point sampled from a real data distribution $\mathbf{x}_0 \sim q(\mathbf{x})$, let us define a forward diffusion process in which we add small amount of Gaussian noise to the sample in T steps, producing a sequence of noisy samples $\mathbf{x}_1, \ldots, \mathbf{x}_T$. The step sizes are controlled by a variance schedule $\{\beta_t \in (0,1)\}_{t=1}^T$.

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}\right) \quad q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$$

- Reverse diffusion process- If we can reverse the above process and sample from $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$, we will be able to recreate the true sample from a Gaussian noise input, $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Note that if β_t is small enough, $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ will also be Gaussian. Unfortunately, we cannot easily estimate $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ because it needs to use the entire dataset and therefore we need to learn a model p_θ to approximate these conditional probabilities in order to run the reverse diffusion process.

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \quad p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}\left(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)\right)$$

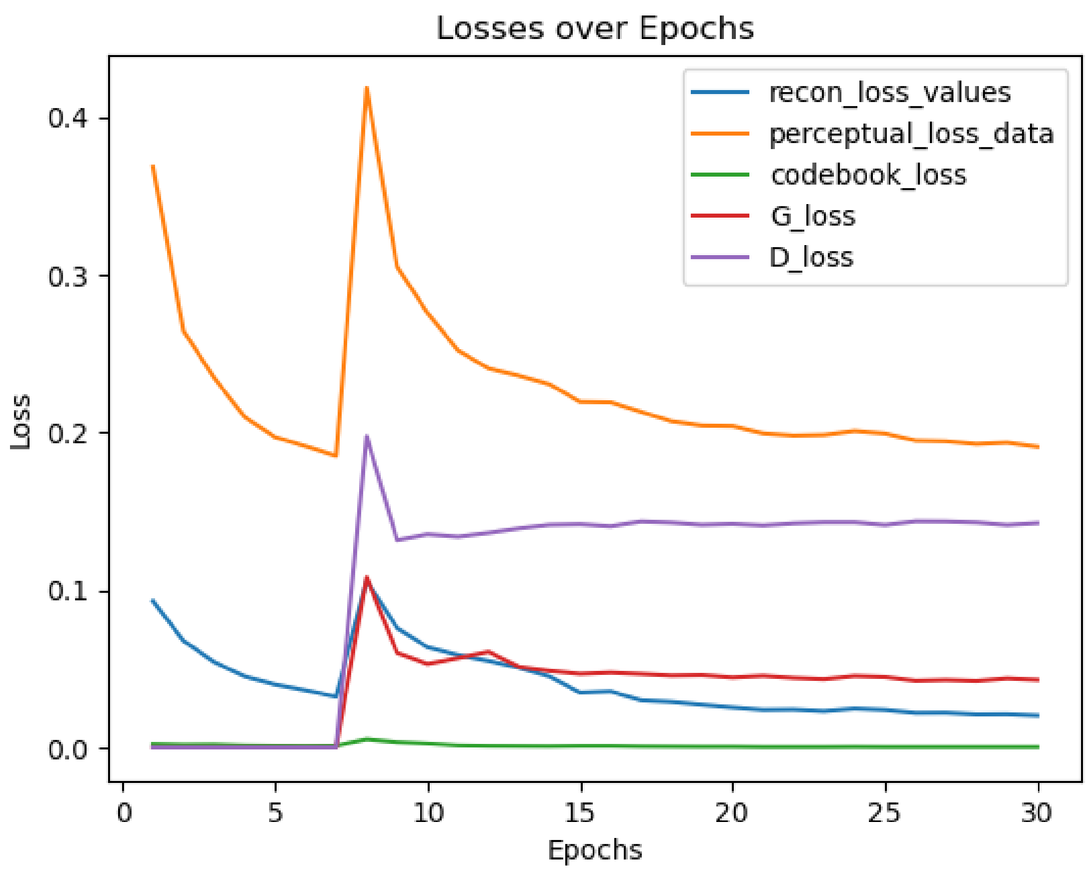
- Training process for Diffusion Model-



5 Results

5.1 VQ-VAE

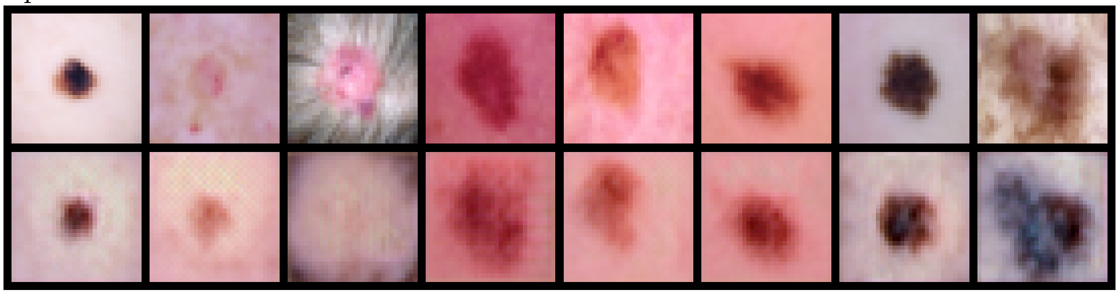
VQ-VAE Training Graph-



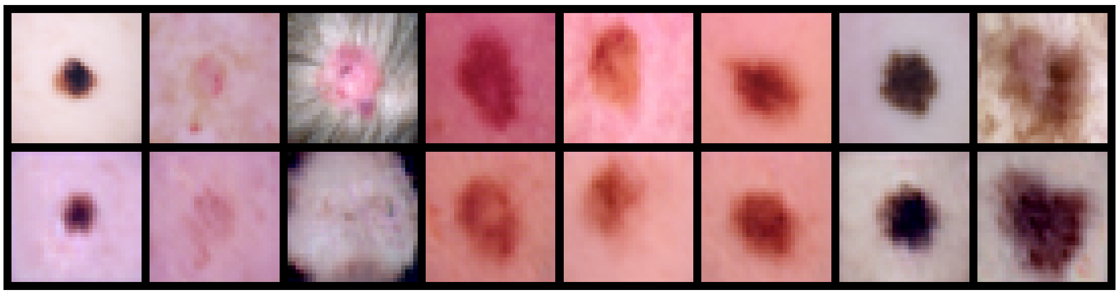
Wandb link-https://wandb.ai/m23mac004/Variatational%20Encoder/reports/VAGAN-TRAINING--
accessToken=1bb6wp0spsqairv6magq9jypnmjrkge7ughp8950zlxnqwa3z6kw9mcj7f6ytmjq

First row has original image and second has corresponding model output-

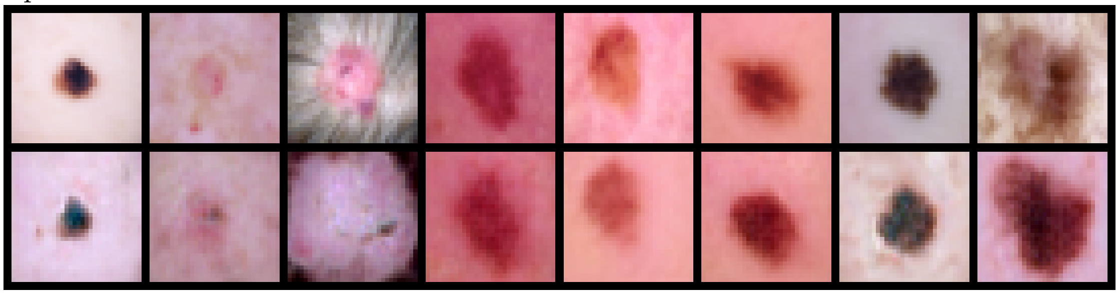
Epoch-10



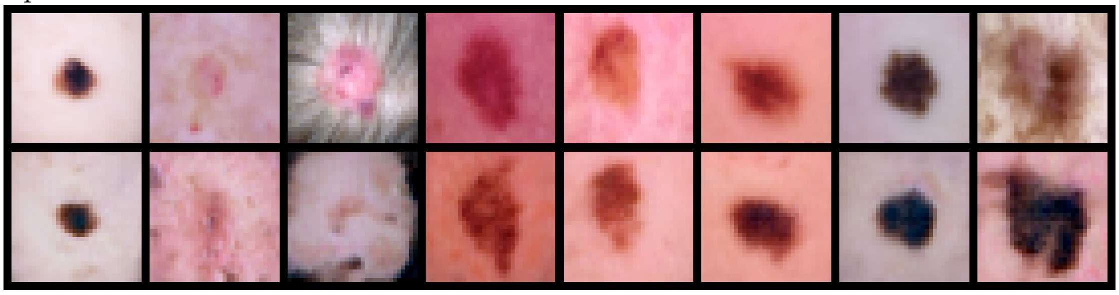
Epoch-15



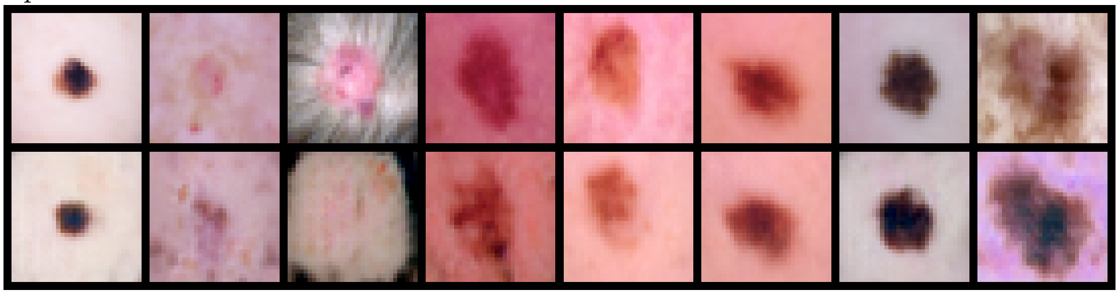
Epoch-20



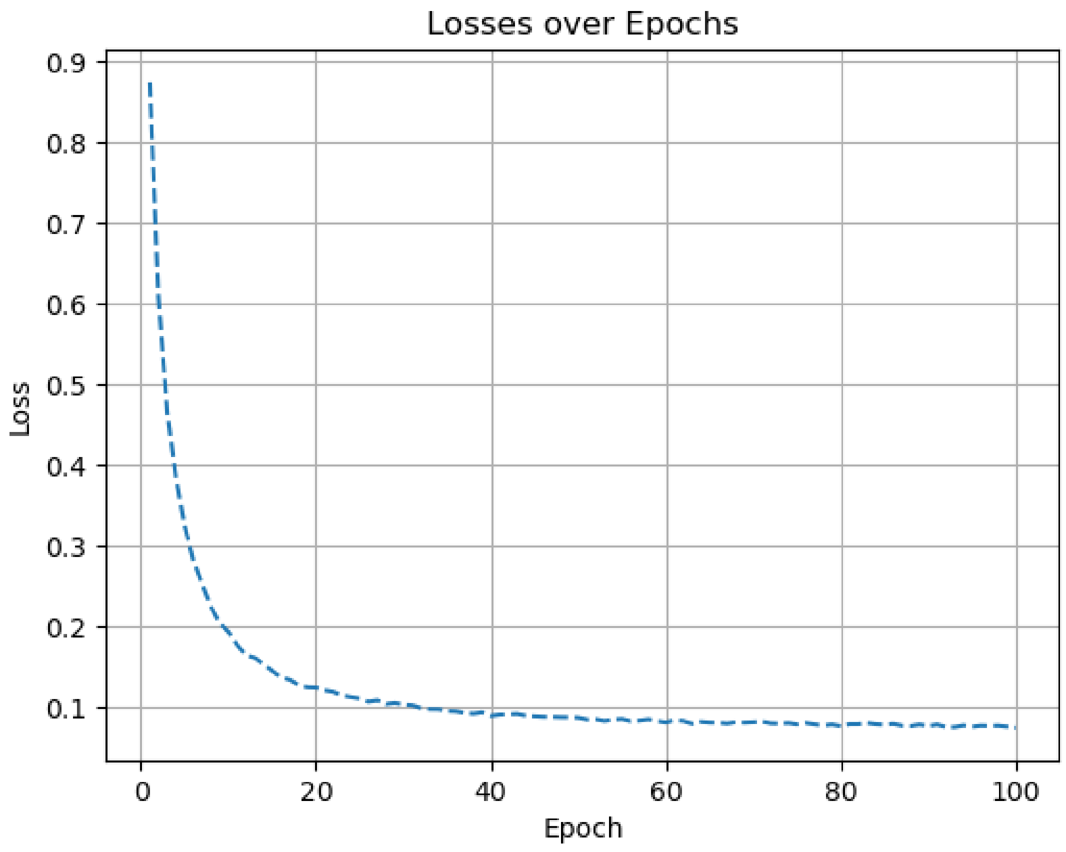
Epoch-25



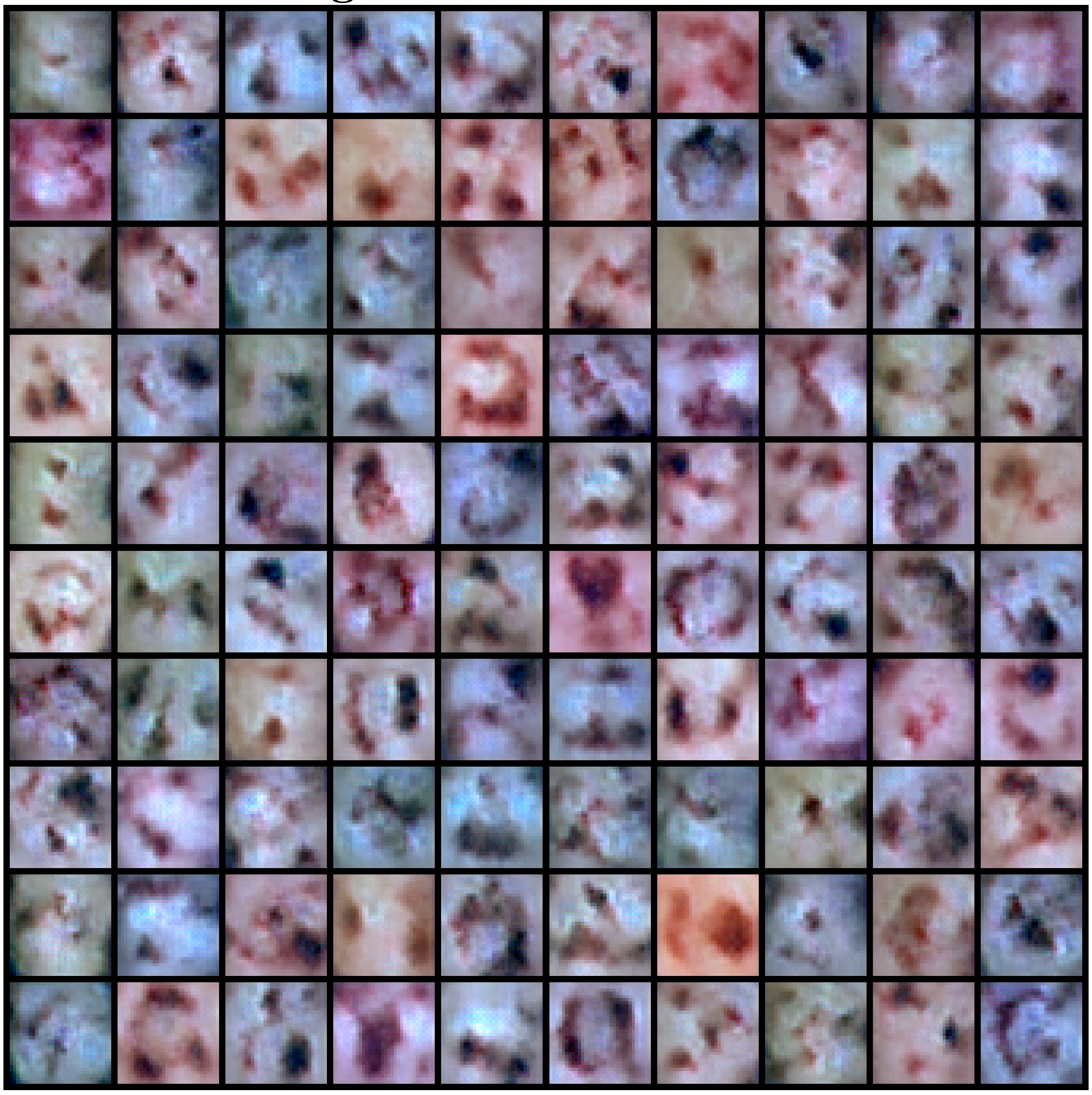
Epoch-30



5.2 Diffusion



Generated image-



References

[1] Patrick Esser, Robin Rombach, and Björn Ommer, *Taming Transformers for High-Resolution Image Synthesis*, *CoRR*, vol. abs/2012.09841, 2020, https://arxiv.org/abs/2012.09841.

[2] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang, *The Unreasonable Effectiveness of Deep Features as a Perceptual Metric*, *CoRR*, vol. abs/1801.03924, 2018, http://arxiv.org/abs/1801.03924.

[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel, *Denoising Diffusion Probabilistic Models*, *CoRR*, vol. abs/2006.11239, 2020, https://arxiv.org/abs/2006.11239.

[4] Lester James Miranda, *The Illustrated VQGAN*, *ljvmiranda921.github.io*, 2021, https://ljvmiranda921.github.io/notebook/2021/08/08/clip-vqgan/.

[5] Lilian Weng, *What are diffusion models?*, *lilianweng.github.io*, Jul 2021, https://lilianweng.github.io/posts/2021-07-11-diffusion-models/.