# 16824: Visual Learning & Recognition
# Homework 2

## TASK 1: GAN

### Question 1.1: GAN Network Architecture:

**Answer:**
Networks.py has been filled.

### Question 1.2: GAN Training Code:

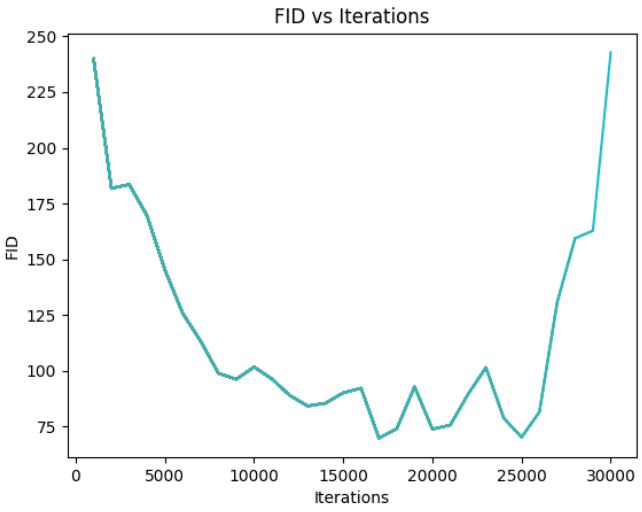
**Answer:**
Train.py has been filled with relevant files

### Question 1.3: Implement GAN loss:

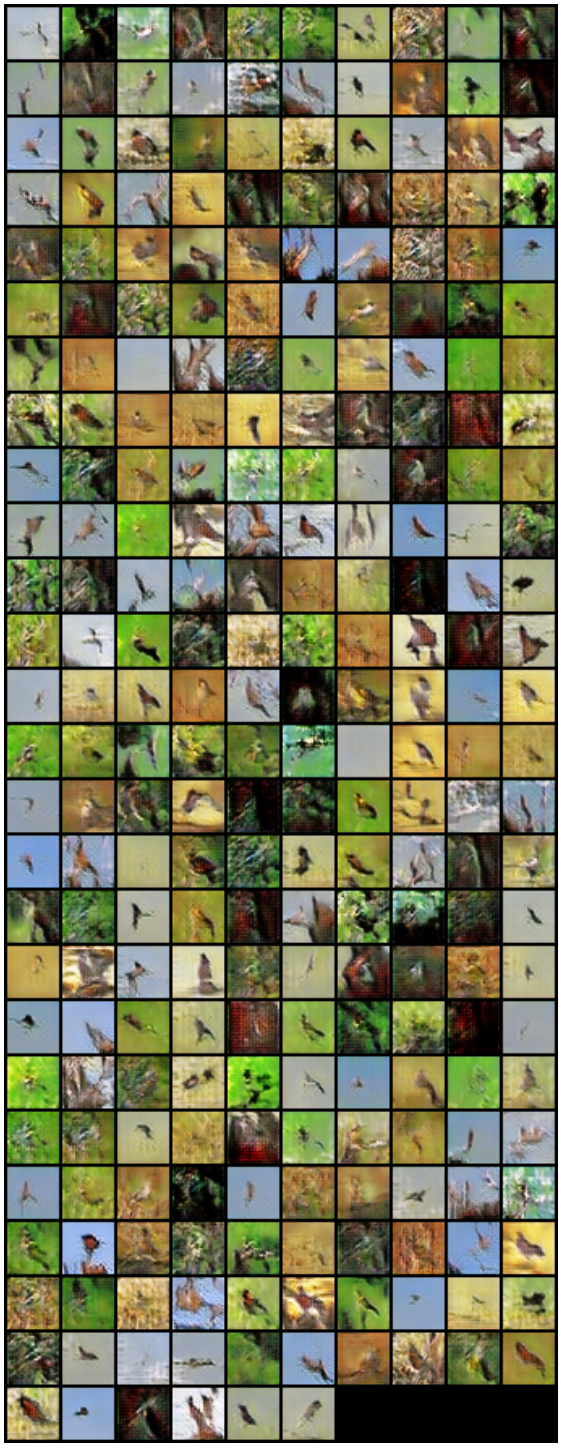
**Answer:**
Gan Loss is implemented in q1_3.py.

**Final FID Attained is: 243.1099**

**FID vs Iterations for GAN:**



Sample Plot:



Latent Space:

Latent Space for normal GANs is not disentangled.

As per FID and as mentioned in the paper (https://arxiv.org/pdf/2108.02353.pdf), lower values of FID are signs of high quality and diversity in generated data. GANs have a failure type called mode collapse where the generator finds itself producing single type or small sets of outputs. This may happen due to the generator finding a type of data which can easily fool the discriminator. This can result in the entire system over optimizing on that similar output. Due to which we see an increase in FID score vs iteration plot. Thus we see unstable FID values.

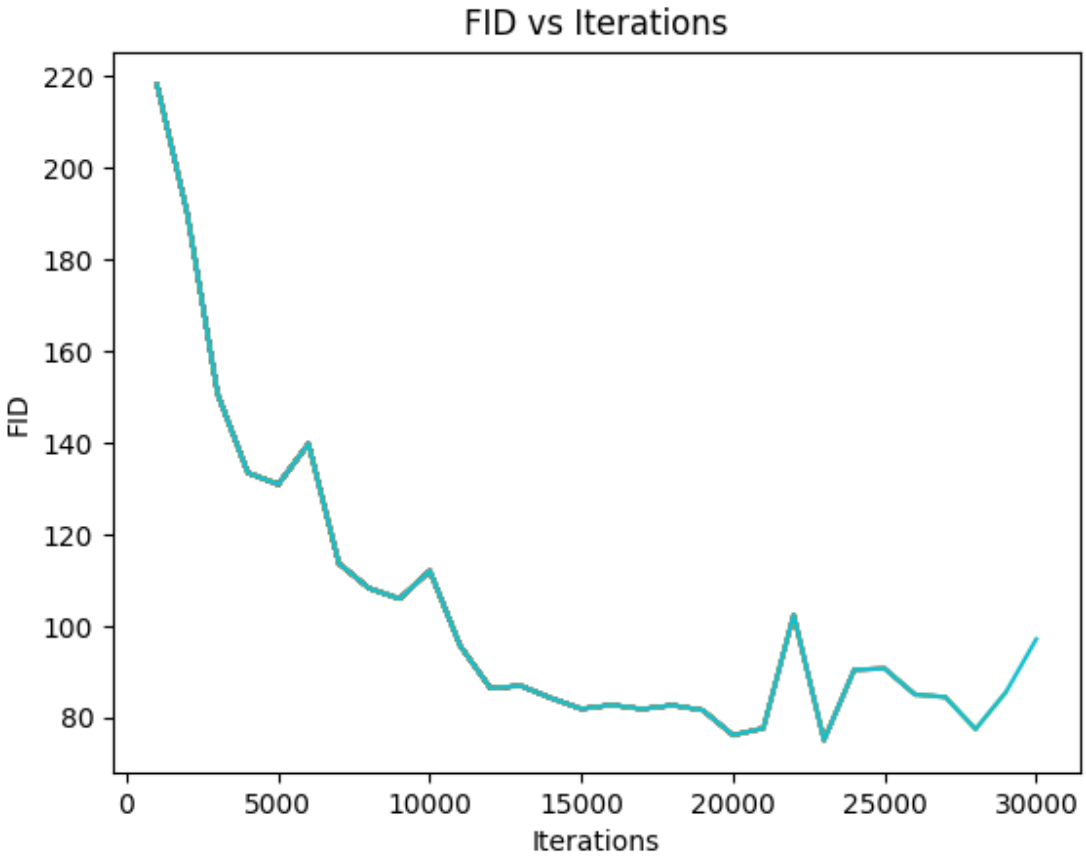
**Question 1.4: Implement LSGAN loss:**

**Answer:**

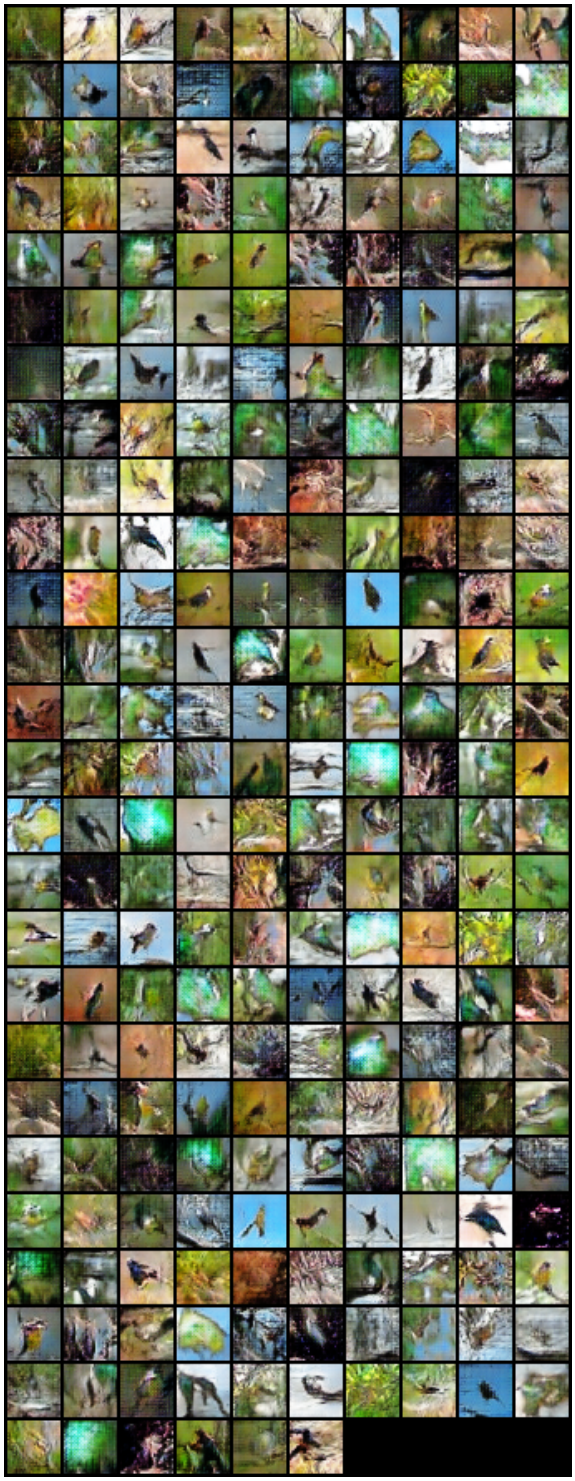
LSGAN Loss has been implemented in q1_4.py.

**Final FID attained is: 90.7404**

**FID Score VS Iterations:**



**Sample Plot:**



**Latent Space:**

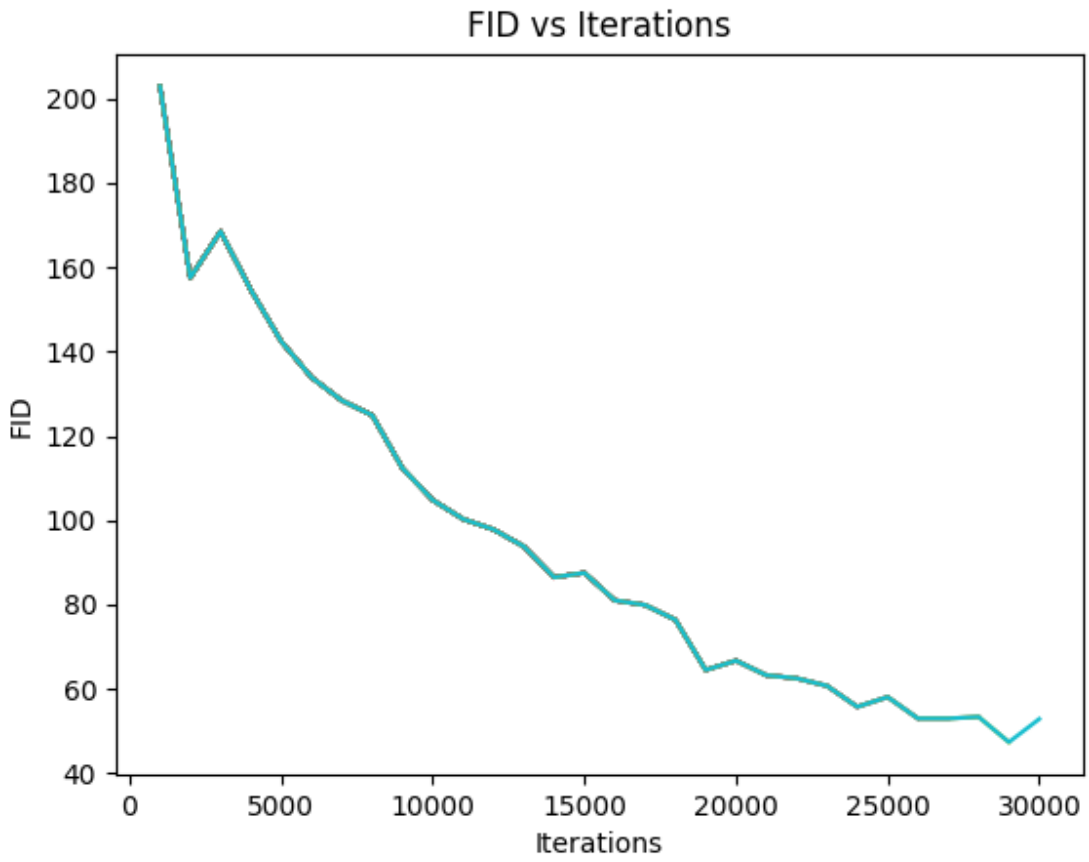
Latent Space for LSGANs is disentangled.

Regular GANs adopt a sigmoid cross entropy loss function which faces a problem of vanishing gradients. This problem is solved by LSGANs. The loss function of LSGANs works by encouraging the movement of generated samples towards the decision boundary and penalizing the samples lying a long way to the decision boundary which can generate more gradients when updating the generator. This allows LSGANs to perform in a more stable way than regular GANs during the learning process.

**Question 1.5: Implement WGAN-GP loss**

WGAN loss has been implemented in q1_5.py.

**Final FID Score attained: 46.78155**
**FID Score VS Iterations Plot:**



**Sample Plot:**

**Latent Space:**

Latent Space for WGANs is disentangled.

Instead of using a discriminator, WGAN uses a critic which predicts the realness and fakeness of the given image. The WGAN loss function reflects the image quality which is more desirable. WGAN also has no signs of Mode Collapse as seen from its experiments and the generator can still learn when the critic performs well. These points make WGAN more stable as compared to regular GANs and LSGANs.

# 16824: Visual Learning & Recognition
# Homework 2

## Task 2: VAE

### Question 2.1: AutoEncoder:

### Question 2.1.1: Architecture

In model.py, fill in the TODOs where 2.1.1 is mentioned. This includes the encoder and decoder network architectures, and the forward passes through each.

**Answer:**
Model architecture has been filled in model.py file.

### Question 2.1.2: Loss function

In train.py, fill in the TODOs where 2.1.2 is mentioned. This includes the loss function for the autoencoder, which is the MSE loss between the input data and the reconstruction. Important - remember to only average across the batch dimension.

**Answer:**

Loss function has been written in train.py file.

Reconstruction Loss for Latent sizes: 16, 128, 1024:

Reconstruction Plots:

Latent Size 16:

Latent Size 128:

Latent Size 1024:


Latent Size 1024 performs better which can be clearly seen from the reconstruction plots. This is because greater the latent size, more information is stored for a better reconstruction result.

**Question 2.2: Variational Auto-Encoder:**

**Question 2.2.1: Architecture**

In model.py. fill in the TODOs where 2.2.1 is mentioned. This only includes the fc layer of the VAEEncoder, and the forward pass through the network.

**Answer:**

Model.py file has been updated to solve VAE.

**Question 2.2.2: Loss function**

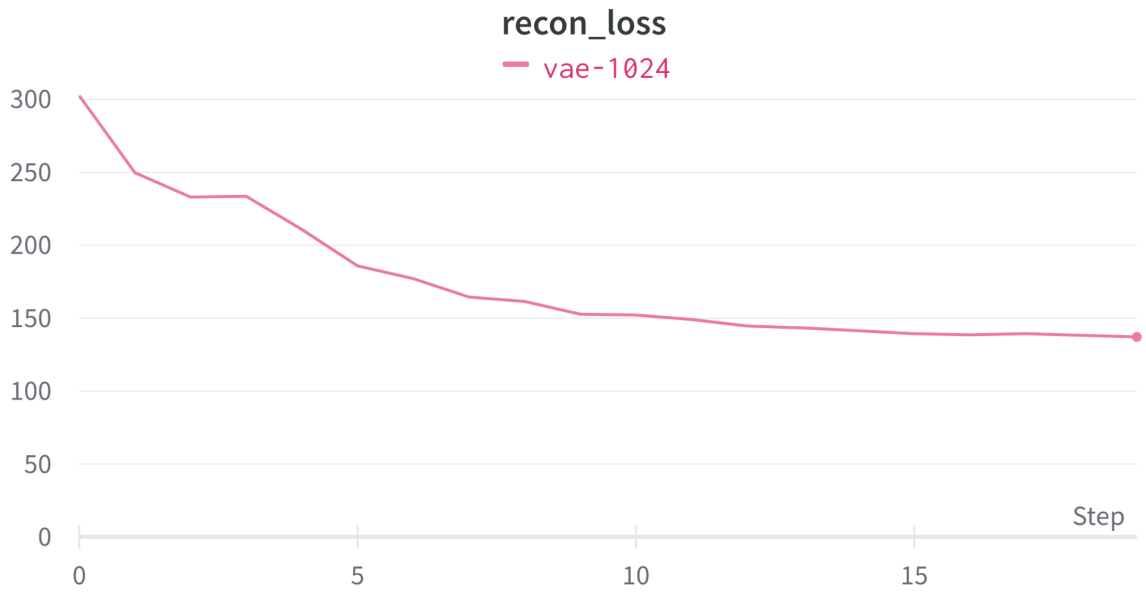
Fill in the recon_loss and kl_loss that make up the total loss for the VAE, under the TODO where 2.2.2 is mentioned. Important - remember to only average across the batch dimension.

**Answer:**

Recon Loss and KL Loss has been updated in vae_loss function in train.py file.

**Losses for latent size = 1024:**

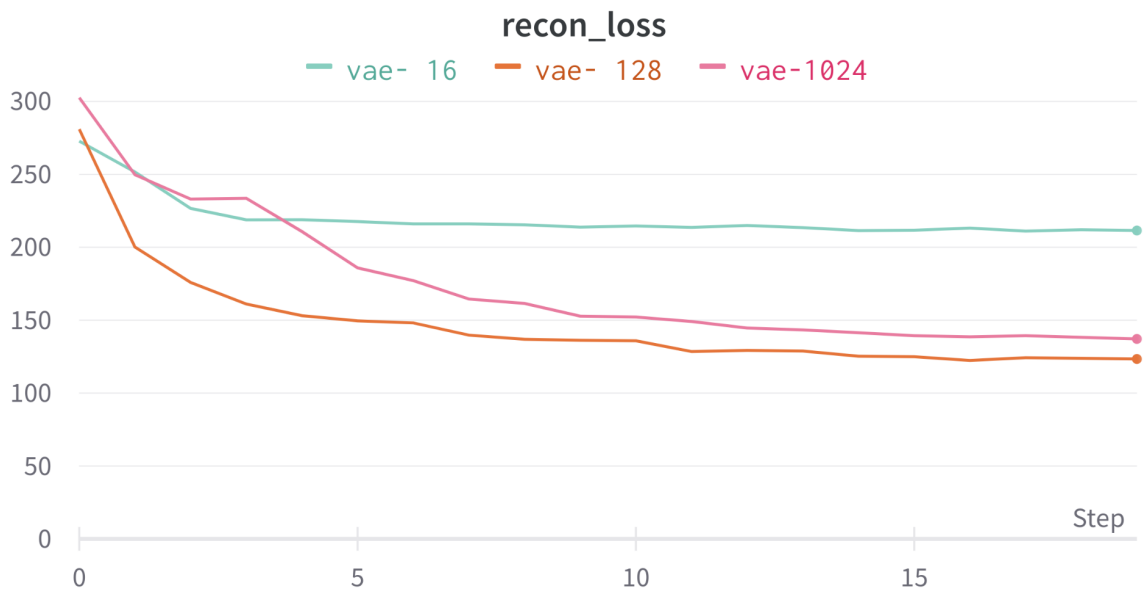
**Reconstruction Loss:**



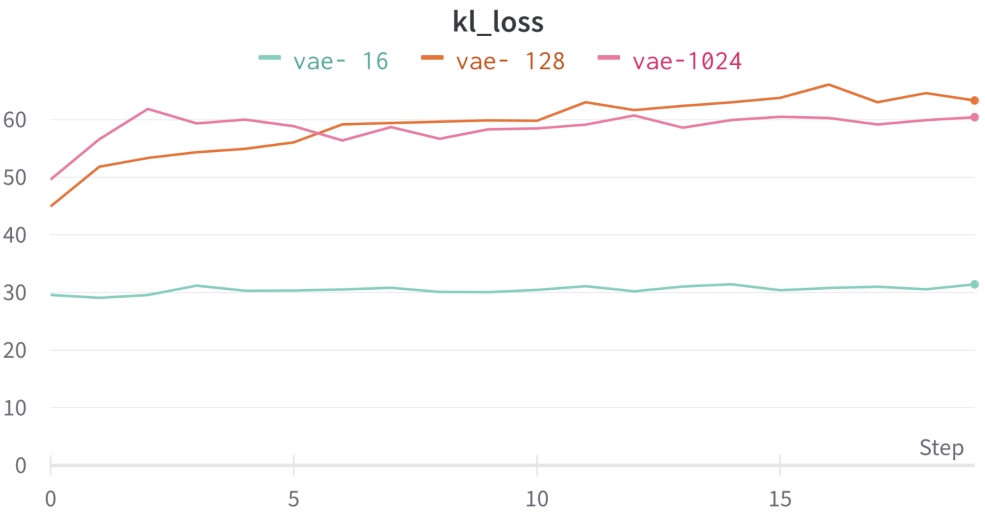
**KL Loss:**

**Losses for Latent Size: 16, 128, 1024:**

**Reconstruction Loss:**



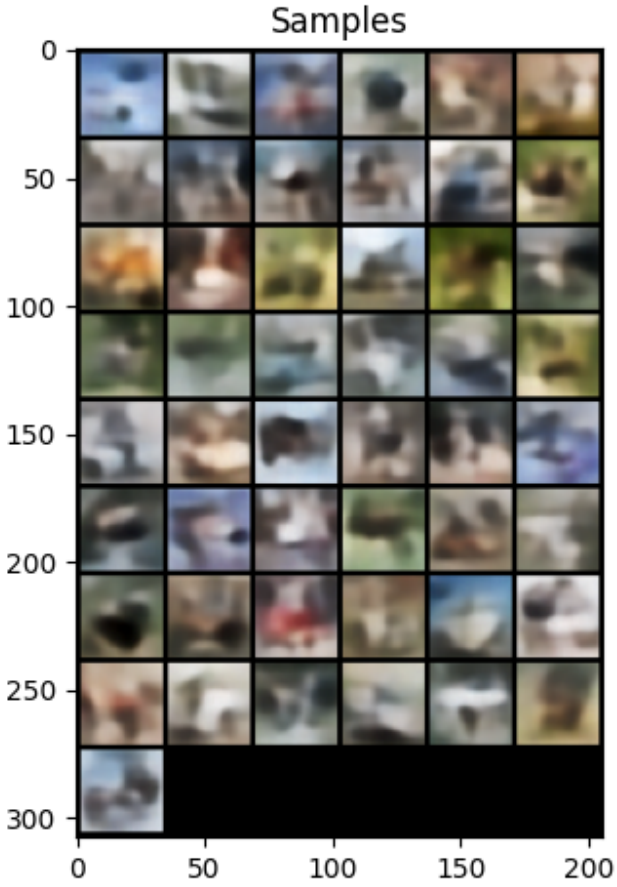

**KL Loss:**



**Plots: (For Latent Size of 1024):**

**Reconstruction plots:**

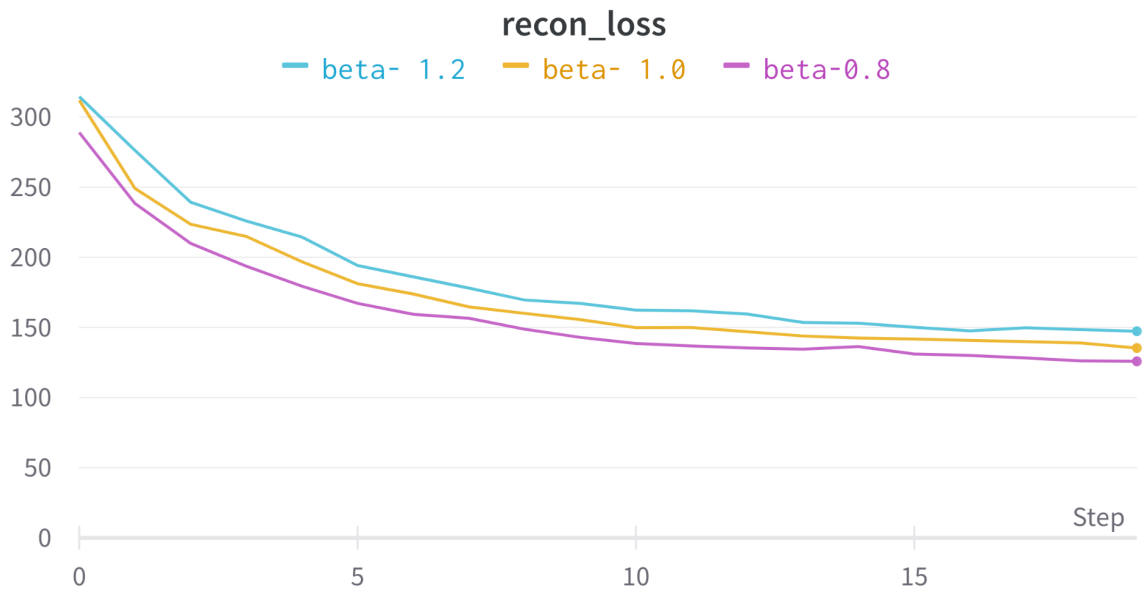
**Sample Plots:**



## Question 2.3: Beta Variational Auto-Encoder:

## Question 2.3.1: Tuning beta:

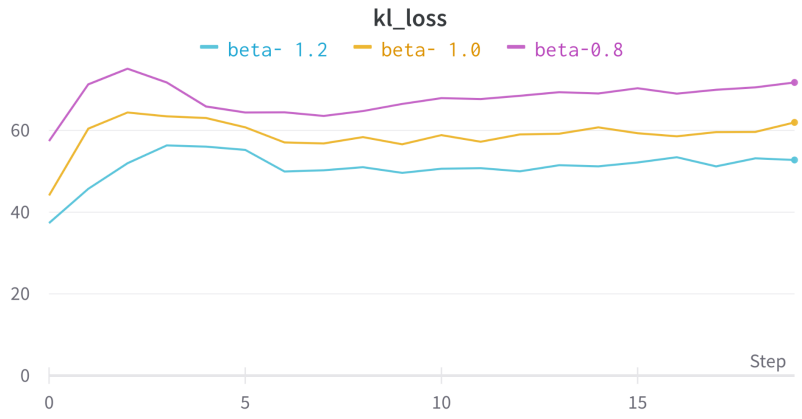
**Reconstruction Loss:**




Recon_loss for:
Beta = 0.8: 125.86
Beta = 1.0: 135.35
Beta = 1.2: 147.322

**Recon Loss increases with increase in beta values which makes it directly proportional to beta.**

**KL Loss:**

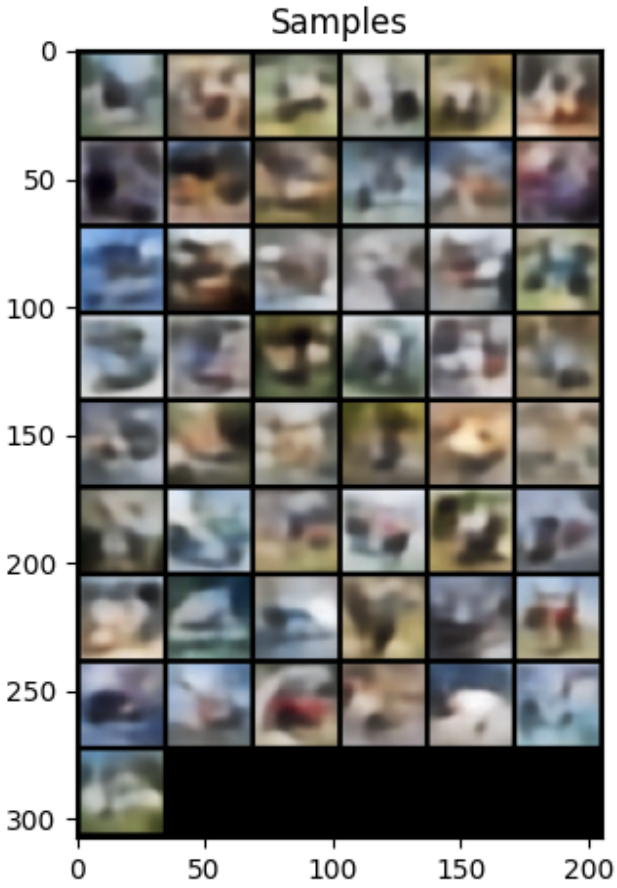



**From the plot, it is seen that higher the beta value, lower is the KL Loss making it inversely proportional to beta.**

**Plots:**

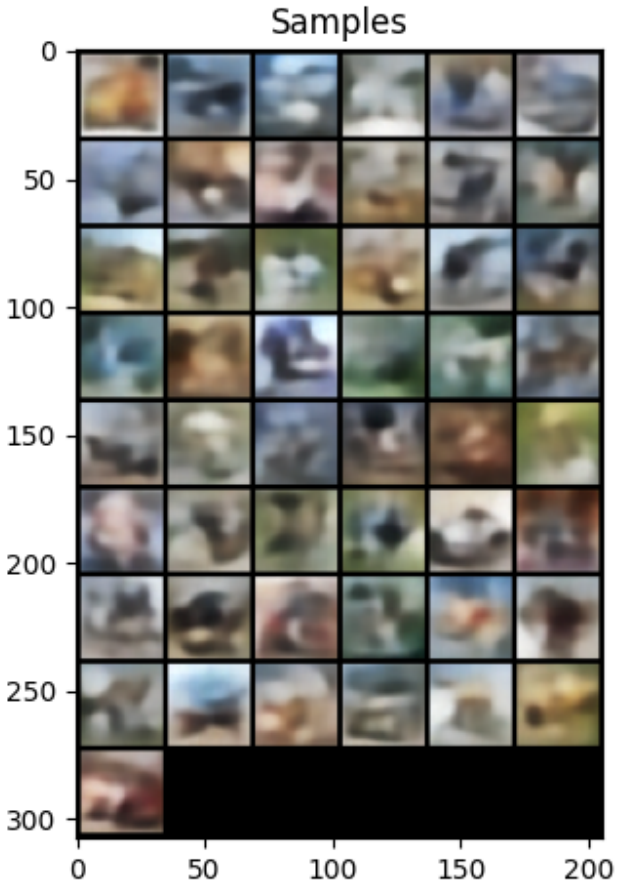
**Reconstruction Plot (Beta 0.8)**

**Sample Plot (Beta 0.8)**



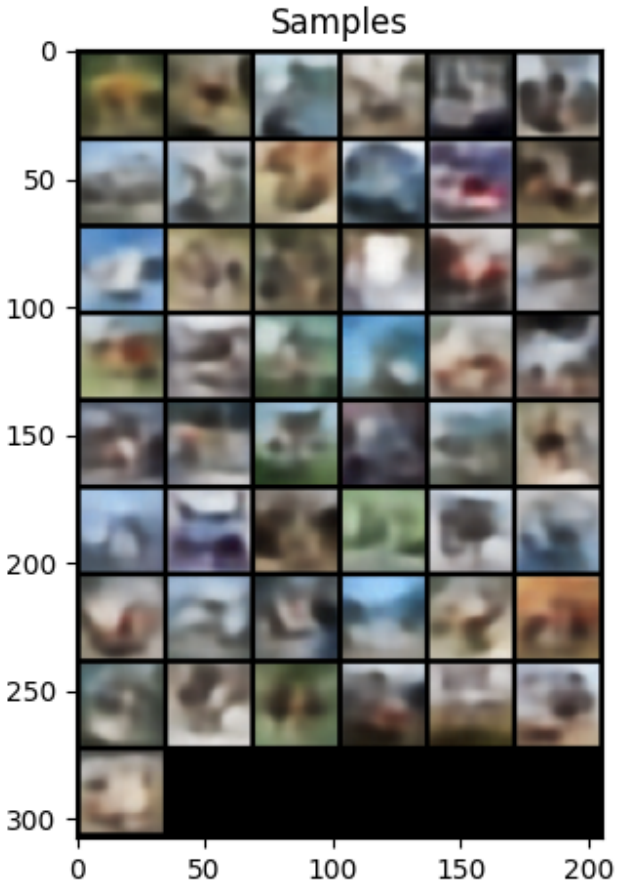
**Reconstruction Plot (Beta 1.0)**

**Sample Plot (Beta 1.0)**



**Reconstruction Plot (Beta 1.2)**

**Sample Plot (Beta 1.2)**



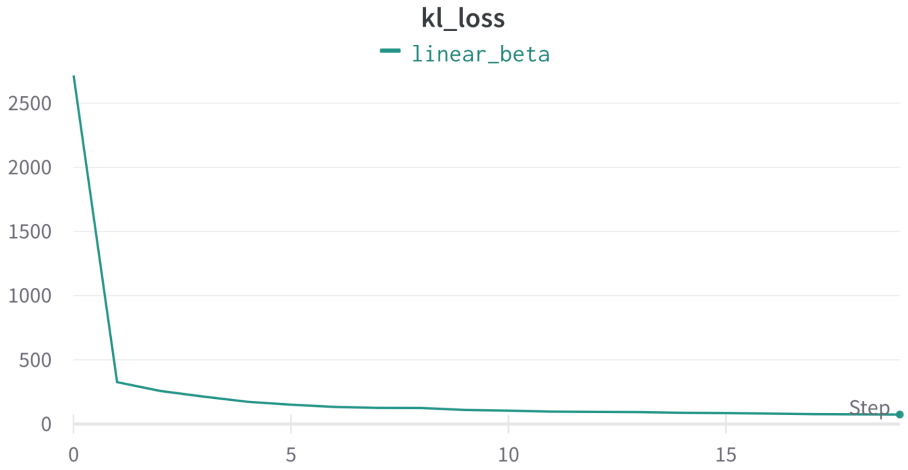

**The samples from beta tuning still stays blurry after we tune beta.**

For Beta = 0, the equation for VAE loss would be recon_loss + 0*kl_loss = recon_loss

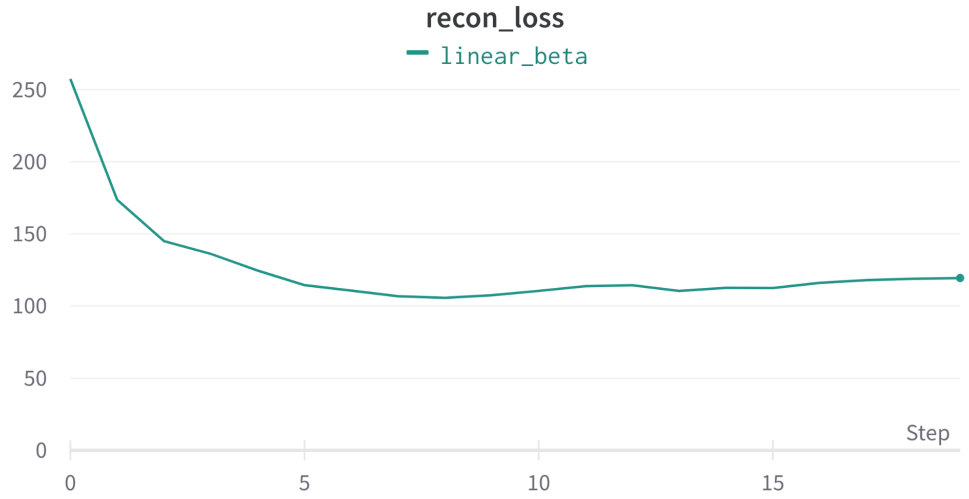
**This means that VAE would reduce to AE for a beta value of 0.**

## Question 2.3.2: Linear schedule for beta:

### KL Loss:




### Reconstruction Loss:

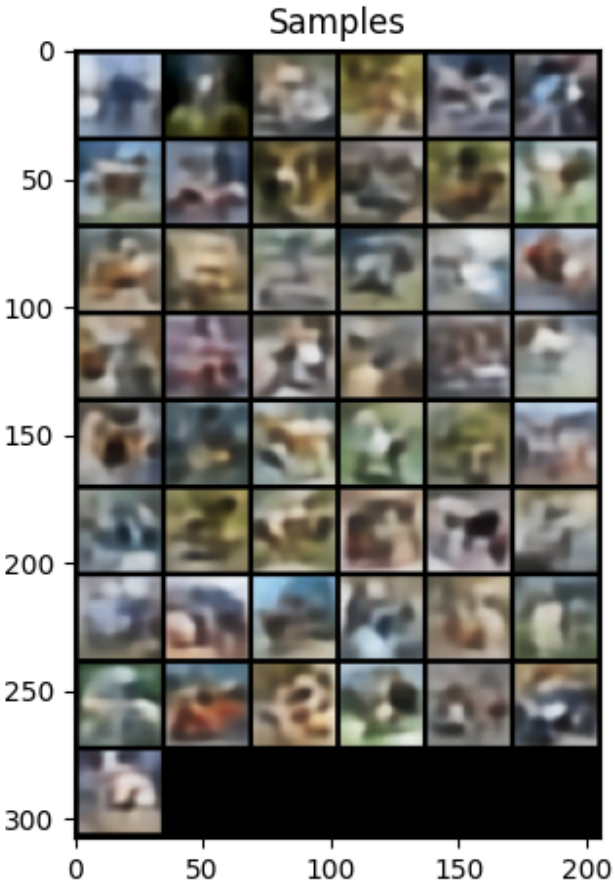



### Recon_loss at epoch 19 = 119.43

**Plots:**

**Reconstruction Plot:**

**Sample Plot:**



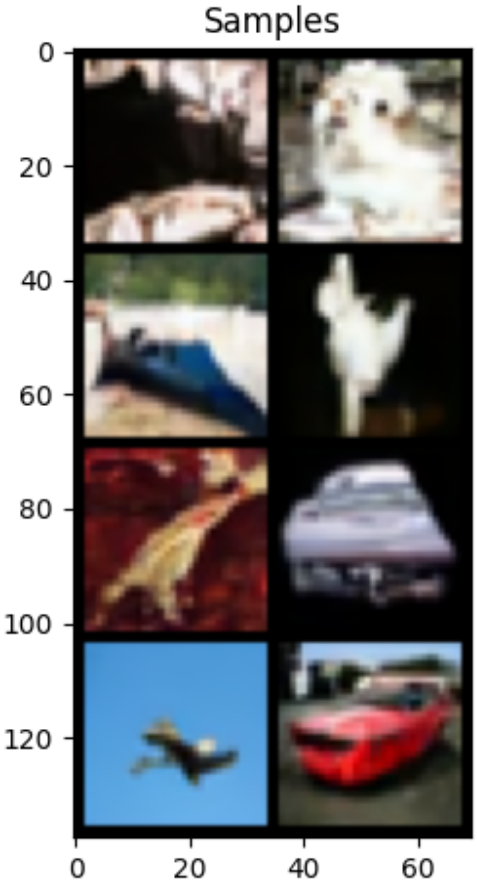

After comparison with Vanilla VAE, it is seen that recon_loss for linear schedule for beta in VAE is lower on last epoch as compared to Vanilla VAE. It is also evident that linear scheduling is causing a smoother reduction in recon_loss as compared to vanilla VAE which stays on higher values during initial epochs. KL_loss for linear beta although higher on 20th Epoch as compared to Vanilla VAE, is clearly seen reducing through the plots.

# 16824: Visual Learning & Recognition
# Homework 2

## TASK 3: DIFFUSION

### 3.1 Denoising Diffusion Probabilistic Models:

**DDPM has been implemented in model.py**




**COLLABORATORS:**

- **Hemant Sharma and I brainstormed about DownsampleConv2D while implementing. He also pointed out that I should use reduction as None and manually implement the reduction when I was not getting the right results after writing the losses for VAE and AE.**
- **Tanmay Chinchanikar and I brainstormed while debugging our errors for DDPM.**