Report

Task 1: Deep Convolutional Generative Adversarial Networks (DCGAN)

Task 1-1

We trained a DCGAN model on the FashionMNIST training data using the following architecture:

Generator:

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
   ConvTranspose2d-1            [-1, 128, 7, 7]         627,200
       BatchNorm2d-2            [-1, 128, 7, 7]             256
         LeakyReLU-3            [-1, 128, 7, 7]               0
   ConvTranspose2d-4           [-1, 64, 14, 14]         131,072
       BatchNorm2d-5           [-1, 64, 14, 14]             128
         LeakyReLU-6           [-1, 64, 14, 14]               0
   ConvTranspose2d-7           [-1, 32, 28, 28]          32,768
       BatchNorm2d-8           [-1, 32, 28, 28]              64
         LeakyReLU-9           [-1, 32, 28, 28]               0
           Conv2d-10            [-1, 1, 28, 28]             288
             Tanh-11            [-1, 1, 28, 28]               0
================================================================
Total params: 791,776
Trainable params: 791,776
Non-trainable params: 0
----------------------------------------------------------------
```

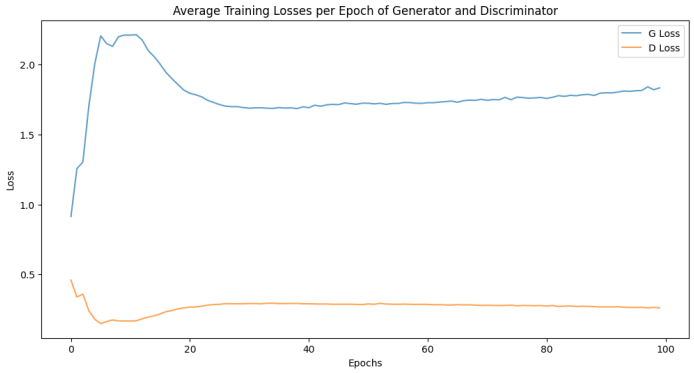
Discriminator:

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
           Conv2d-1           [-1, 32, 28, 28]             288
      BatchNorm2d-2           [-1, 32, 28, 28]              64
        LeakyReLU-3           [-1, 32, 28, 28]               0
           Conv2d-4           [-1, 64, 14, 14]          32,768
      BatchNorm2d-5           [-1, 64, 14, 14]             128
        LeakyReLU-6           [-1, 64, 14, 14]               0
           Conv2d-7            [-1, 128, 7, 7]         131,072
      BatchNorm2d-8            [-1, 128, 7, 7]             256
        LeakyReLU-9            [-1, 128, 7, 7]               0
          Conv2d-10            [-1, 256, 3, 3]         524,288
     BatchNorm2d-11            [-1, 256, 3, 3]             512
       LeakyReLU-12            [-1, 256, 3, 3]               0
          Conv2d-13              [-1, 1, 1, 1]           2,304
================================================================
Total params: 691,680
Trainable params: 691,680
Non-trainable params: 0
----------------------------------------------------------------
```

We made the discriminator to have a bit less number of parameters as its task is a bit easier than that of the generator.

Task 1-2

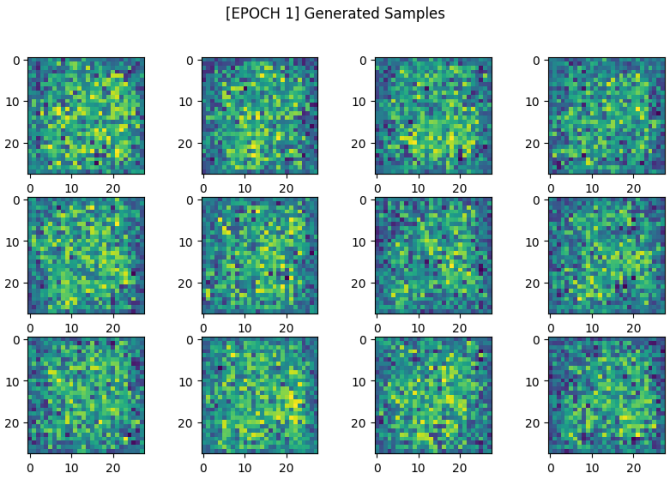
Here are the learning curves of Generator and Discriminator:



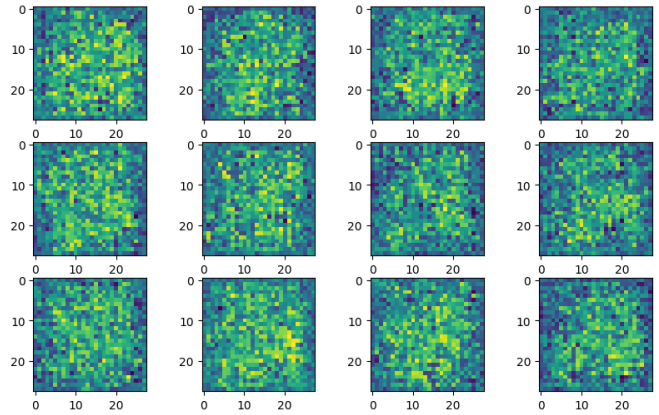
We can see that after about 5 epochs, generator and discriminator start to improve in their tasks, however the generator performance degrades after around 40 epochs, while the discriminator gets a little bit better. This may be due to the limited architecture of the generator.

Task 1-3

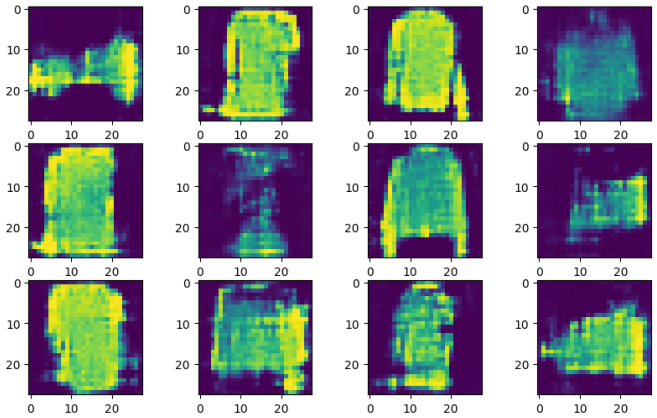
We sampled 12 random noise vectors and plotted the corresponding results of the Generator model over the 5 epochs [1,5,10,50,100] during training. Here are the results:



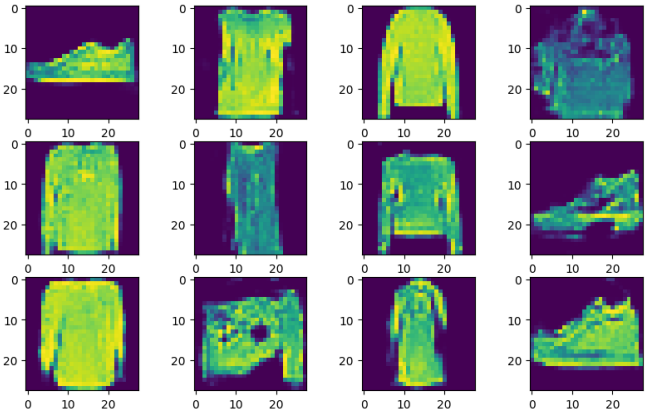
[EPOCH 1] Generated Samples



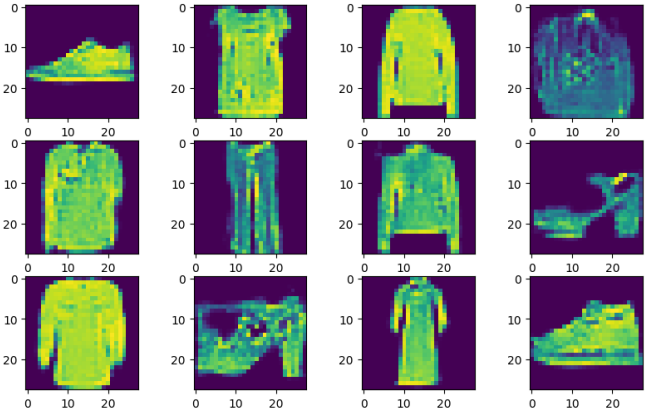
[EPOCH 10] Generated Samples



[EPOCH 50] Generated Samples



[EPOCH 100] Generated Samples



We can see that in the beginning, the generator cannot generate images similar to the training set. But as we train for more epochs, the generated images become similar to the real images.

Task 1: Variational Autoencoders

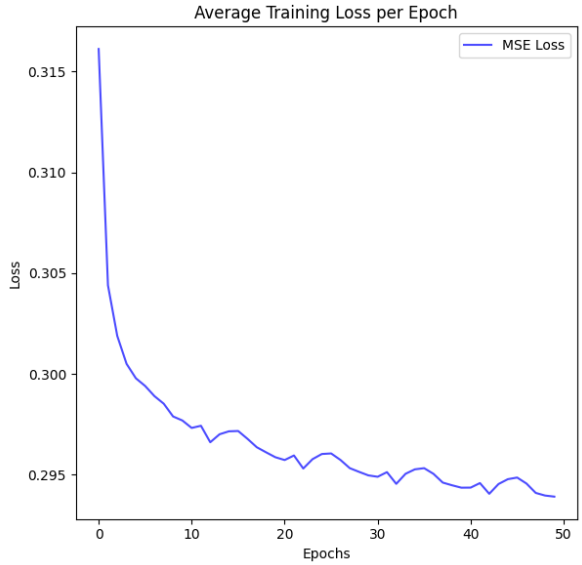
Task 2-1

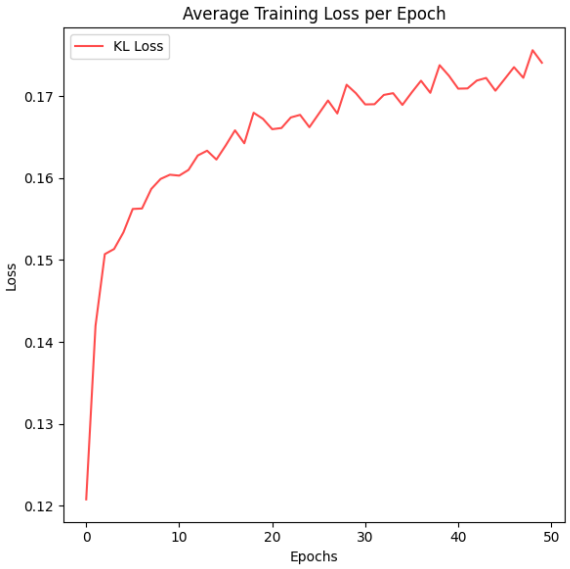
Here is our VAE model architecture:

```
VAEMLP(
  (encoder): Encoder(
    (layers): Sequential(
      (0): Linear(in_features=784, out_features=500, bias=True)
      (1): LeakyReLU(negative_slope=0.01, inplace=True)
    )
    (z_mean): Linear(in_features=500, out_features=2, bias=True)
    (z_logvar): Linear(in_features=500, out_features=2, bias=True)
  )
  (decoder): Decoder(
    (layers): Sequential(
      (0): Linear(in_features=2, out_features=500, bias=True)
      (1): LeakyReLU(negative_slope=0.01, inplace=True)
      (2): Linear(in_features=500, out_features=784, bias=True)
    )
  )
)
```

Task 2-2

Here is the training loss trend over epochs for VAE:





When doing an initial training using the training loss criterion with **a coefficient equal to 1** for the KL term, we got almost the same generated images for all epochs and MSE loss stopped decreasing after only a few epochs. So we tried to change that coefficient to different values and see the results.

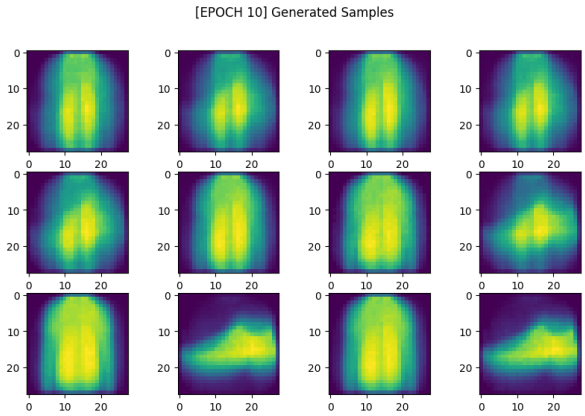
We tried a variety of values such as 0.7, 0.5, 0.4, 0.3, but for all of these values, we had almost no change in the generated images.

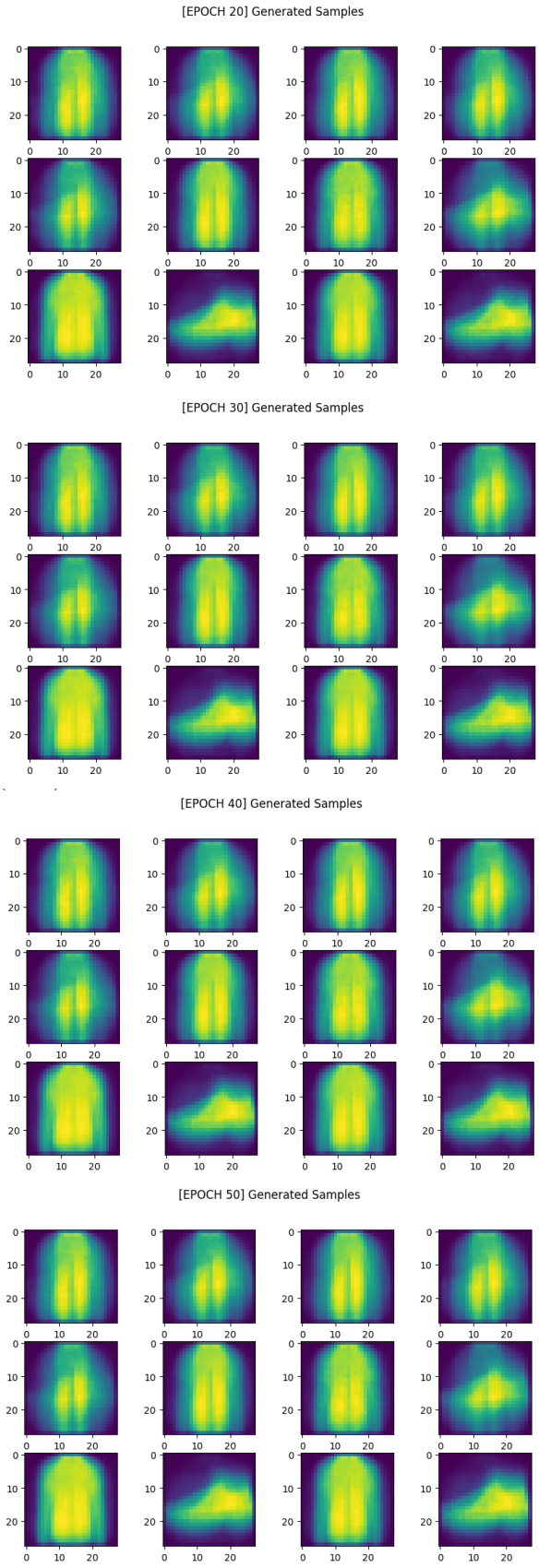
When we tried **0.25**, the images similar to the training set images started to merge (thus we decided to use that coefficient for the final experiment), but when we plotted the loss curves, the KL loss started to increase as the training progressed. So, we encountered a tradeoff MSE loss and KL loss; as we gave more importance to the MSE loss, the KL loss curve got worse and when we gave more importance to KL loss, the MSE loss stagnated.

All of those problems were solved, we increased the dimension of the hidden vector z from 2 to a higher number such as 64. So the real problem can be that the hidden dimension of 2 is too small to accurately model the distribution and generate real-looking images.

Task 2-3

We sampled 12 random noise vectors and plotted the corresponding results of the VAE decoder model over the 5 epochs [1,5,10,50,100] during training. Here are the results:





Task 2-4

Here is the MLP classifier model architecture based on VAE encoder:

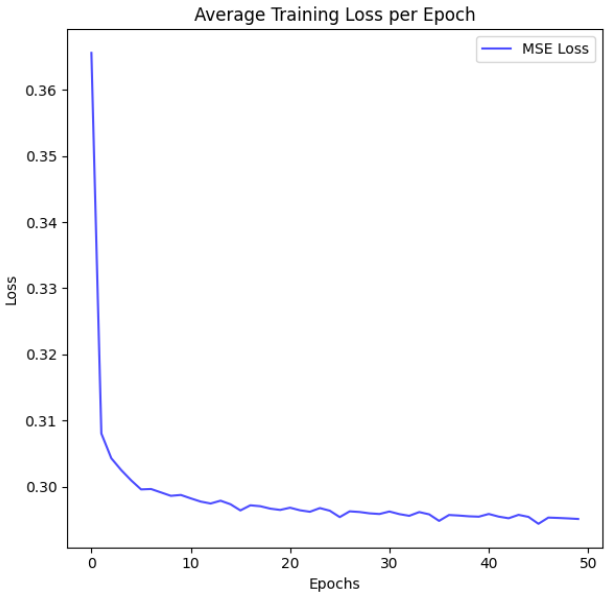
```
MLP_Classifier(
  (layers): Sequential(
    (0): Linear(in_features=784, out_features=500, bias=True)
    (1): LeakyReLU(negative_slope=0.01, inplace=True)
  )
  (output_projection): Linear(in_features=500, out_features=10, bias=True)
)
```

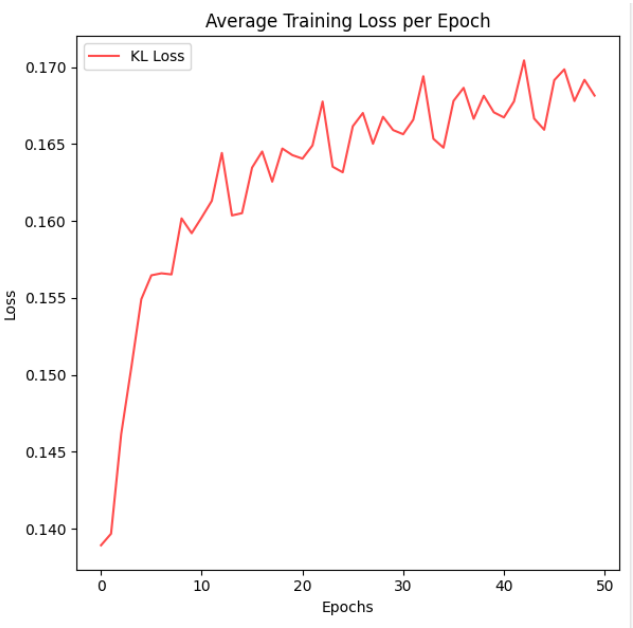
We trained the classifier model for 30 epochs and got 88.6% accuracy on the test set. Below, we show the progress bar of the last 10 epochs:

```
[EPOCH 20] TRAIN LOSS : 0.254, TEST LOSS :0.330, TRAIN ACCU: 0.909, TEST ACCU :0.881
Epoch 21/30: 100% 469/469 [00:14<00:00, 34.88it/s]
[EPOCH 21] TRAIN LOSS : 0.251, TEST LOSS :0.330, TRAIN ACCU: 0.911, TEST ACCU :0.880
Epoch 22/30: 100% 469/469 [00:15<00:00, 24.80it/s]
[EPOCH 22] TRAIN LOSS : 0.246, TEST LOSS :0.326, TRAIN ACCU: 0.913, TEST ACCU :0.883
Epoch 23/30: 100% 469/469 [00:14<00:00, 33.89it/s]
[EPOCH 23] TRAIN LOSS : 0.241, TEST LOSS :0.325, TRAIN ACCU: 0.914, TEST ACCU :0.882
Epoch 24/30: 100% 469/469 [00:15<00:00, 35.28it/s]
[EPOCH 24] TRAIN LOSS : 0.237, TEST LOSS :0.327, TRAIN ACCU: 0.915, TEST ACCU :0.882
Epoch 25/30: 100% 469/469 [00:15<00:00, 25.54it/s]
[EPOCH 25] TRAIN LOSS : 0.232, TEST LOSS :0.320, TRAIN ACCU: 0.918, TEST ACCU :0.883
Epoch 26/30: 100% 469/469 [00:14<00:00, 34.28it/s]
[EPOCH 26] TRAIN LOSS : 0.229, TEST LOSS :0.323, TRAIN ACCU: 0.919, TEST ACCU :0.885
Epoch 27/30: 100% 469/469 [00:14<00:00, 34.37it/s]
[EPOCH 27] TRAIN LOSS : 0.225, TEST LOSS :0.318, TRAIN ACCU: 0.920, TEST ACCU :0.885
Epoch 28/30: 100% 469/469 [00:15<00:00, 26.50it/s]
[EPOCH 28] TRAIN LOSS : 0.223, TEST LOSS :0.317, TRAIN ACCU: 0.921, TEST ACCU :0.883
Epoch 29/30: 100% 469/469 [00:14<00:00, 34.98it/s]
[EPOCH 29] TRAIN LOSS : 0.218, TEST LOSS :0.316, TRAIN ACCU: 0.923, TEST ACCU :0.887
Epoch 30/30: 100% 469/469 [00:14<00:00, 34.94it/s]
[EPOCH 30] TRAIN LOSS : 0.215, TEST LOSS :0.316, TRAIN ACCU: 0.924, TEST ACCU :0.886
```

Task 2-5

Here we plot the training loss trend of our VAE model, which has an encoder initialized by the classifier weights:





Using the pretrained weights from the classifier slightly helped to improve the model training performance as the overall training loss was 0.3374 after 50 epochs when we did not use the pretrained weights and it became 0.3371 after 50 epochs when we utilized the pretrained weights. Also, the KL divergence loss was slightly lower after 50 epochs when we used the pretrained weights.

Here are the generated samples across epochs: