Question #1

(1) If we apply standard gradient decent,

$$\frac{\partial Loss}{\partial x} = \begin{cases} -1 & (x < 1) \\ 1 & (1 < x < 1 + h) \\ -1 & (1 + h < x < 1 + 2h) \end{cases}$$

Assume initial point o (0,0), learning rate a= 0.3.
The point moves from
(0,0) -> (0.3, -0.3)-> (0.6, -0.6)->(0.9,-0.9)->(1.2,-1.2)-> (0.9,-0.9)->(1.2,-1.2)

When comes to (1.2,-1.2), the slope becomes 1. If we don't stop, it will repeat the slope of -1,1,-1,1….Therefore, the gradient decent stuck at 0.9 - 1.2, which is around the point X. That because the learning rate is too large, we cannot reach the local minimum point X.

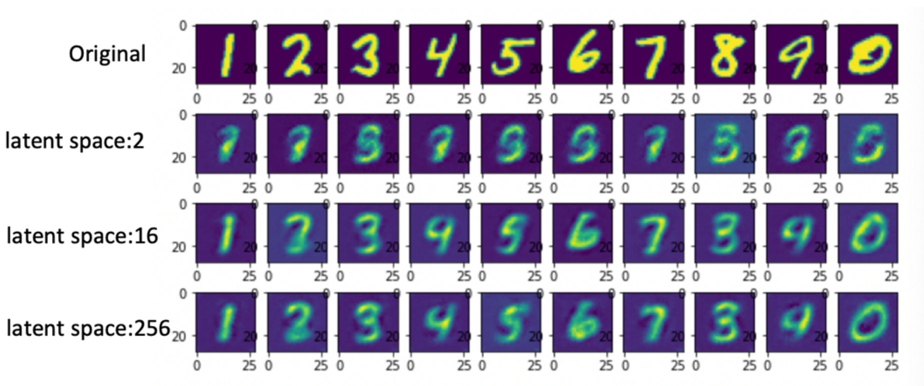
(2) Adam Optimizer

```
m = 0
v = 0
x = 0
for t in range(1,n):
    g = grad(x,h)
    m = beta1 * m + (1-beta1)*g
    v = beta2 * v + (1-beta2)*g**2
    mhat = m/(1-beta1**t)
    vhat = v/(1-beta2**t)
    x = x-learning_rate*mhat / (vhat**0.5+epsilon)
```

```
Maximum hight= 0.4110000000000001
```

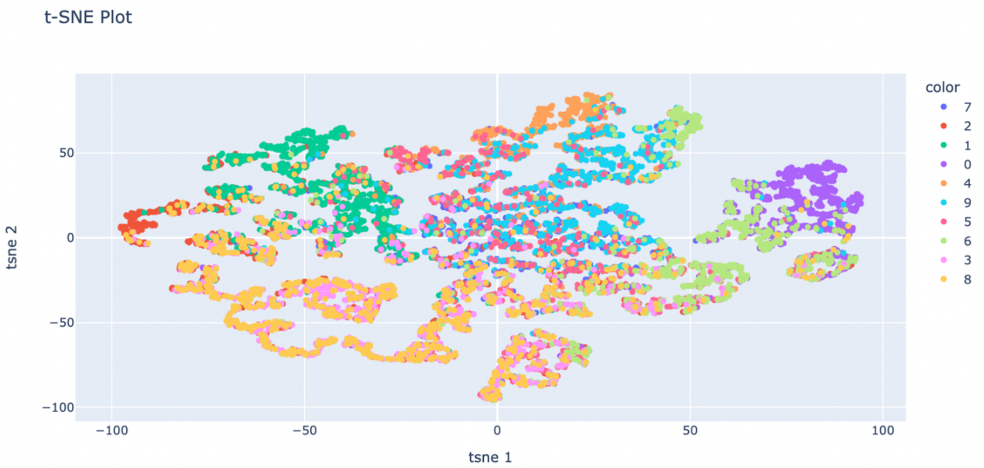
Question #2a

The reconstructed images of question 2a are shown below. We can see that the result for 2D latent space is the worst, with most of the numbers being identified as 3. The result of latent space 16 and 256 are similar, both having difficulty distinguishing between 4 and 9, 3and 8.



2D plot of latent space for latent space of 2D:

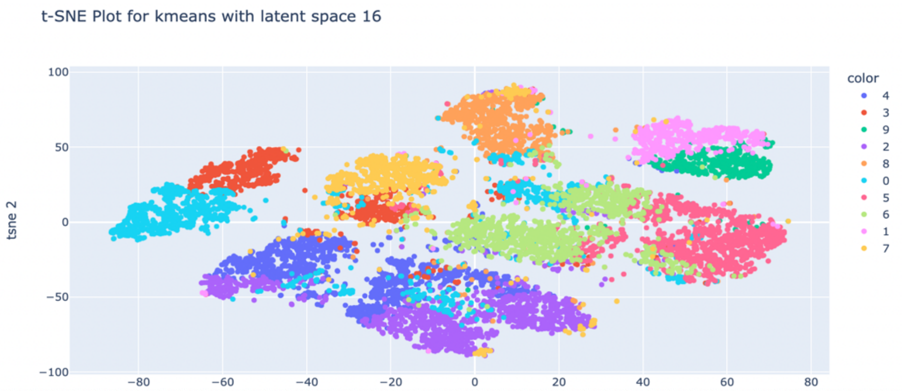
Features are clearly clustered. Since only two features are packed in the latent space representation of the data, the differences between similar samples are removed as redundant. Only two features are retained. So, we can see that the numbers are not well clustered. There are many similar numbers such as 3 and 8 are overlapped.

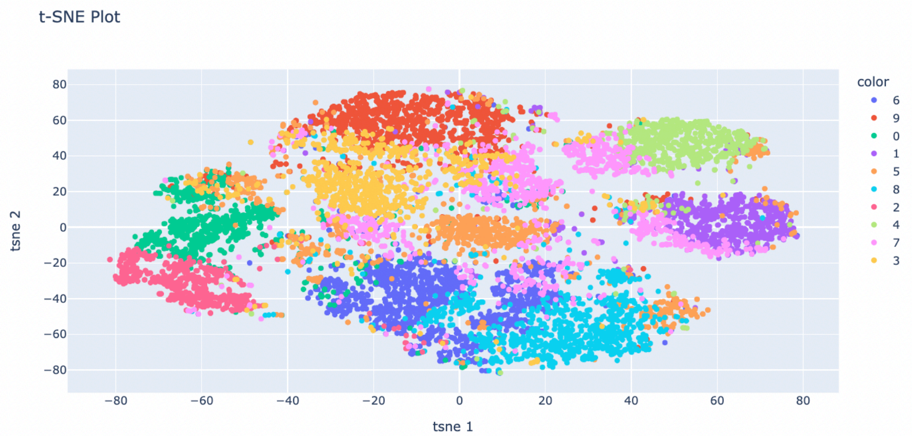


K-means clustering result.

Latent space of 16D and 256D

K-Means result shows that 16D and 256D are well clustered, since they retained more features.





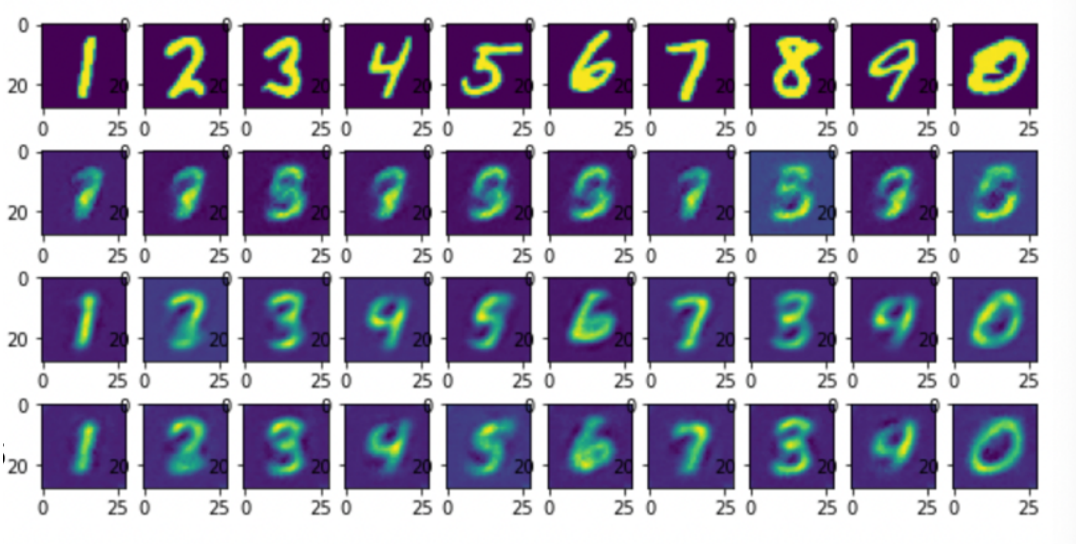
Question #2 b

L1 (MSE) loss and discriminate loss are coded as below.

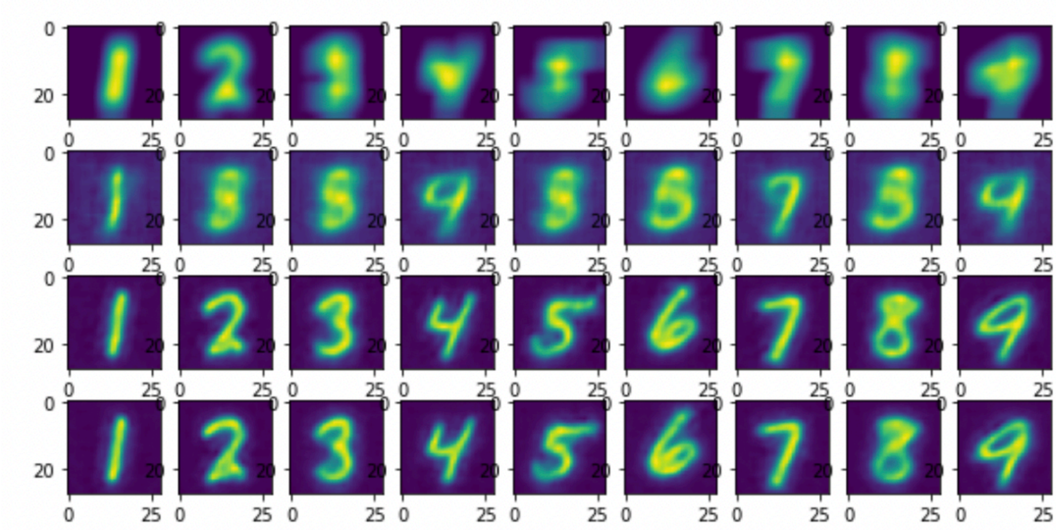
```
loss_latent_2 = distance(output_latent_2, img) + distance(output_latent_2, og_img)
loss_latent_16 = distance(output_latent_16, img)+ distance(output_latent_16, og_img)
loss_latent_256 = distance(output_latent_256, img)+ distance(output_latent_256, og_img)
```

Comparing result of a and b:

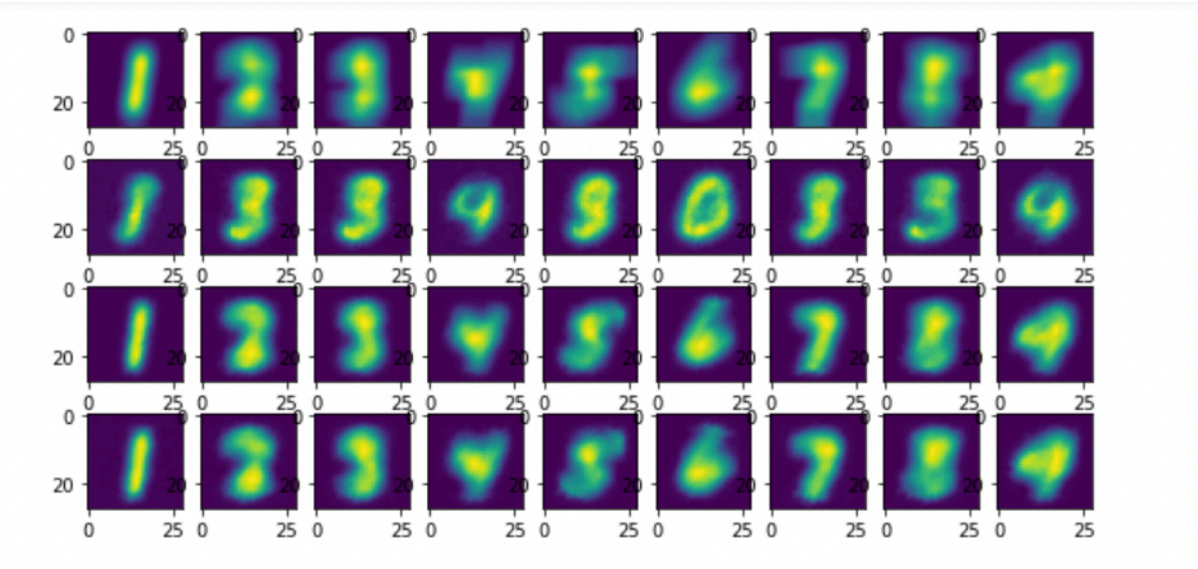
a. L1



b. MSE reconstruction loss + discriminator loss



c. L1 reconstruction loss + discriminator loss



I have tried two loss algorithms for question 2b; one is L1(MAE) another is MSE. The result of MSE is better than L1. The reconstructed result of MSE is clearer. Comparing with result a, both results of latent 2D are unsatisfactory. a(2D) results look more structured but lose elements in the numbers, b(2D) result looks blurred but clear elements of number. However, c shows the worst result of reconstruction.