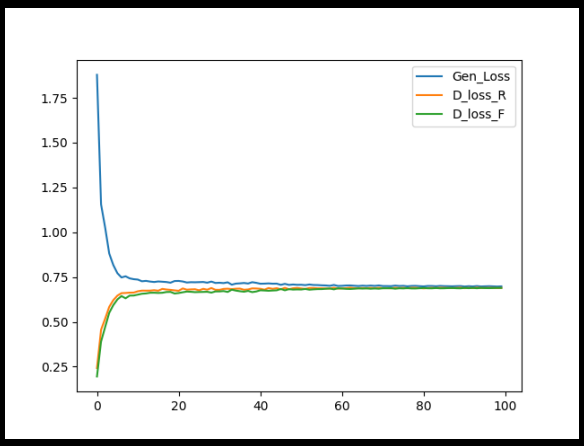
Generative Adverserial Networks

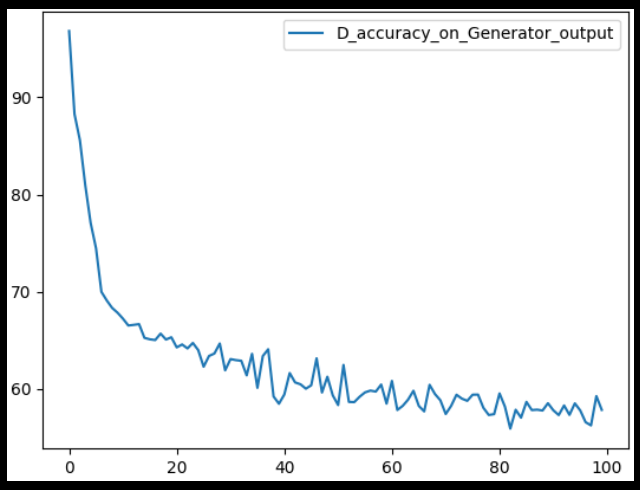
① Training Loss / Discriminator Accuracy



Observation :-

Initially discriminator loss was low and generator loss was high. Then as the min-max game proceeded in epochs, both Generator and Discriminator sort of reach a nashe equillibrium state.

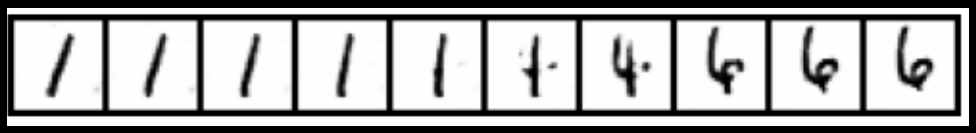
Discriminator accuracy on generated images.



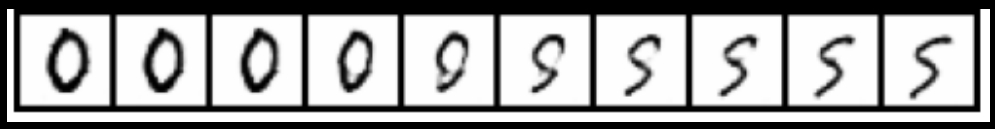
Theoratically it should be 50%

Here it goes to 58.03% at 100th Epoch.

② "Exploring" Latent space : linearly interpolating in latent space [10 steps]



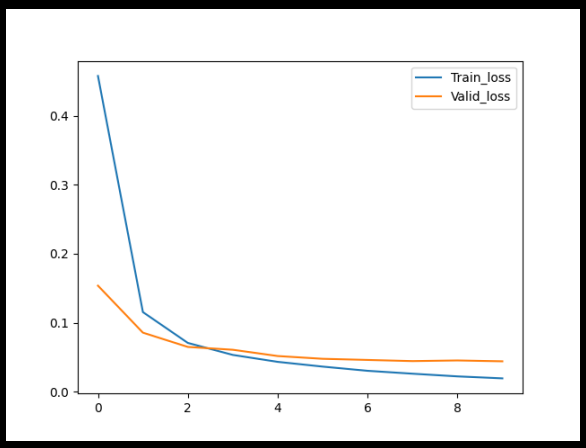
Traversing in latent space



Observation :- Notice how a digit "transforms" to another digit as we traverse in latent space [latent_dim = 100]

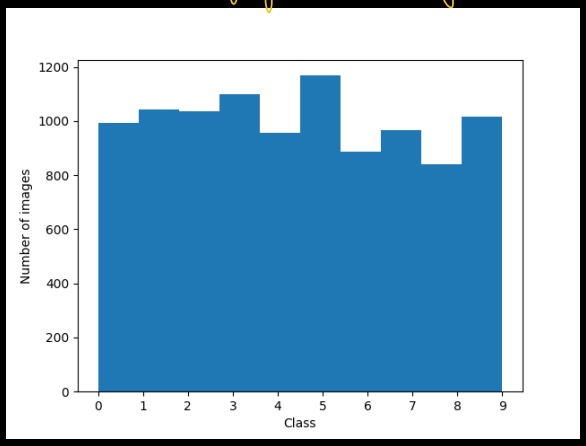
③ MNIST classifier

Trained a classifier on MNIST data. As you can see, it trained well.



Test data loss : 0.04285

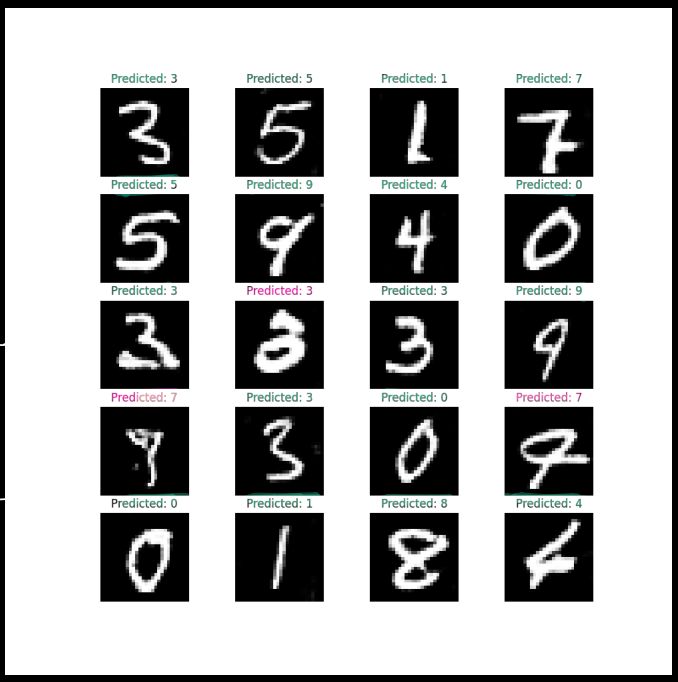
Distribution of generated images



Observation :-

The model produces images which are almost uniformly distributed over classes [0-9].

Analysis of 20 Randomly generated Images and their corresponding predictions by the classifier.



Predictions

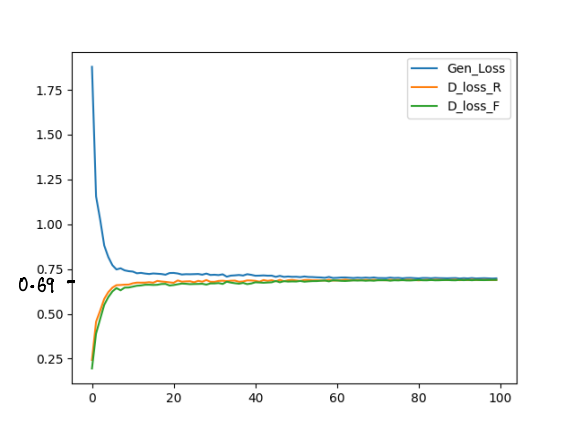
- Wrong
- Correct

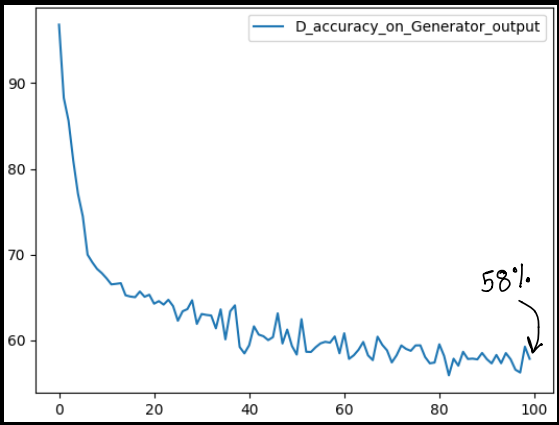
Observations :

1) Model generates almost readable digits
2) Predictions match with the generated digits for most of the cases.

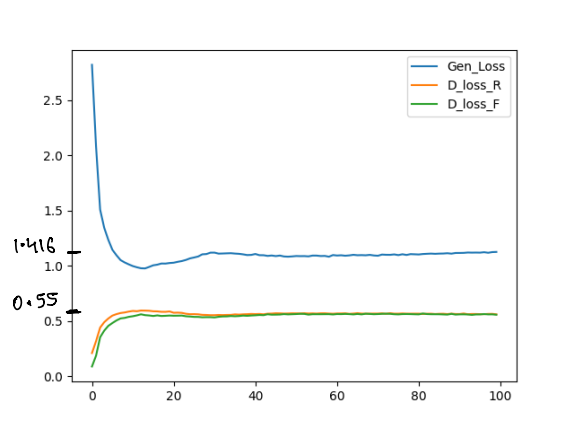
(4) Training GANs With / Without Spectral Norm

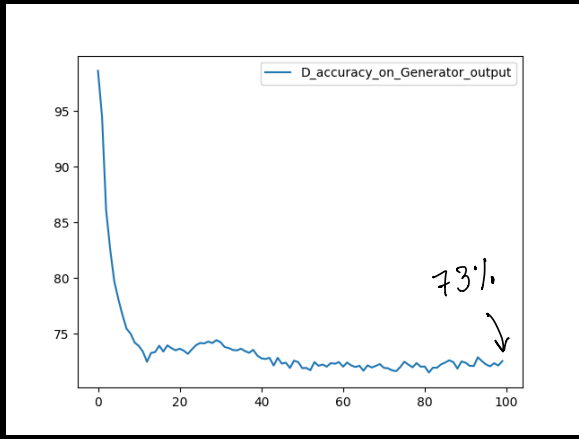
With Spectral Norm in both Generator and Discriminator





Without Spectral Norm in Generator and Discriminator



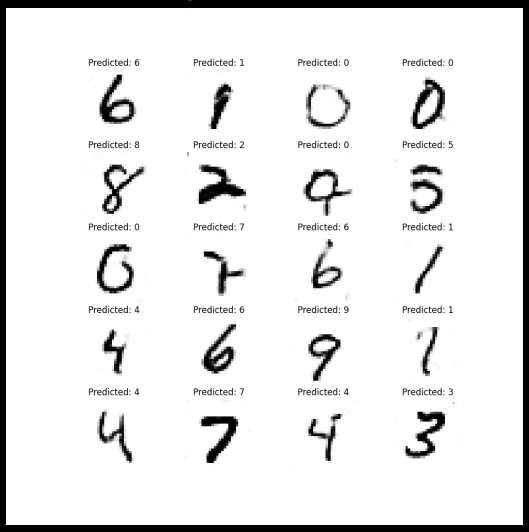


Observations :- Discriminator accuracy is higher [Not good] without SN
Also D_loss is lower but G_loss is higher. Applying SN gave better results!

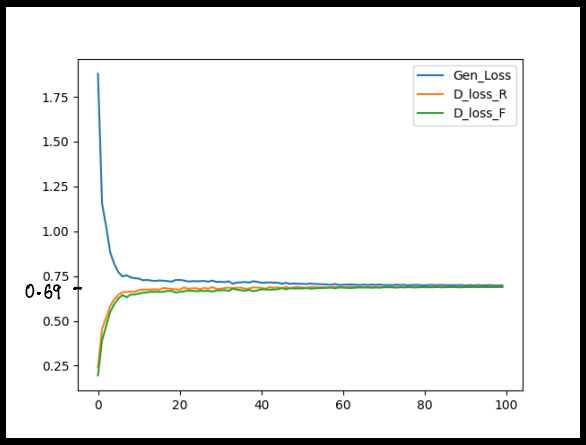
4 Analysis on Hyperparameters

① Spectral Norm of parameter tensors.

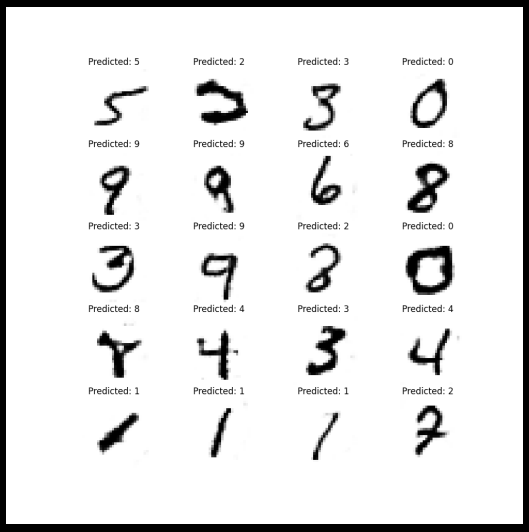
Spectral Norm in both Generator and Discriminator



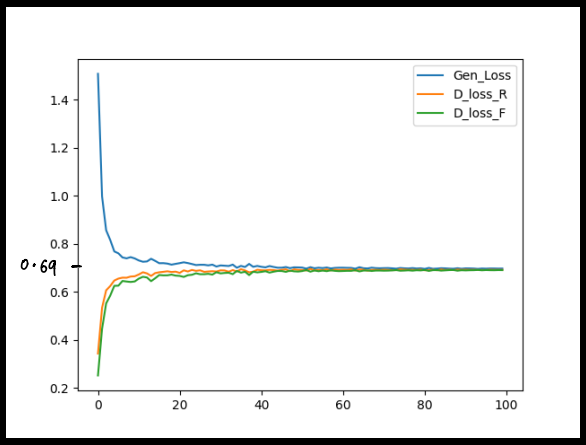
Generated Samples with their predicted classes



Spectral Norm in Discriminator only



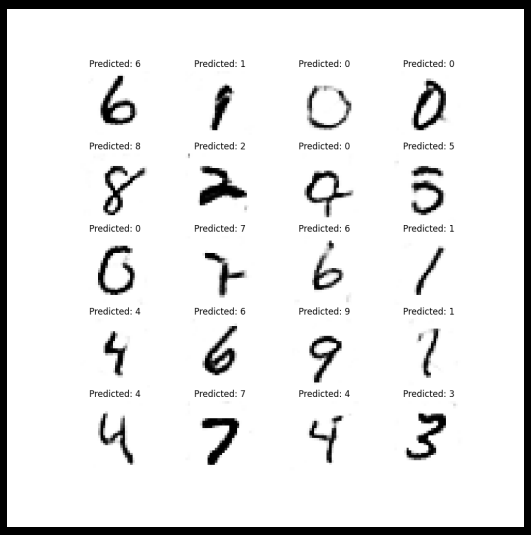
Generated Samples with their predicted classes



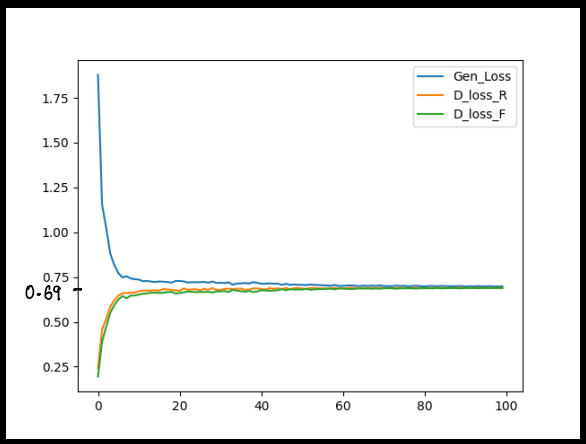
Observation :- Outputs generated with SN on discriminator parameters only are more crisp / visually good than the former case.

(2) Number of Generator / Discriminator updates per iteration

D_steps_per_iter = 5 and G_steps_per_iter = 5



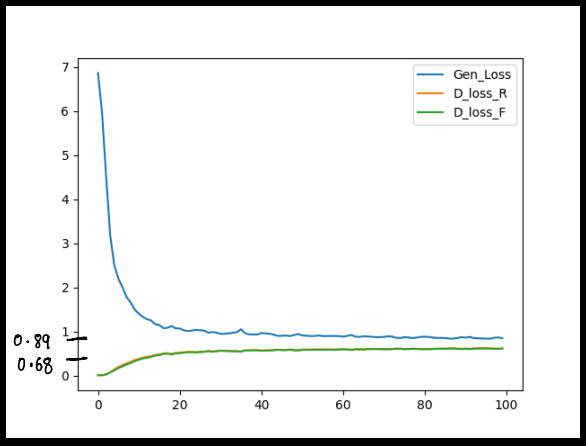
Generated samples with their predicted classes



D_steps_per_iter = 5 and G_steps_per_iter = 1

Generated Samples with their predicted classes

loss



Epochs

Observation :-

1) Gen_steps_per_iter = 5
   Dis_steps_per_iter = 5

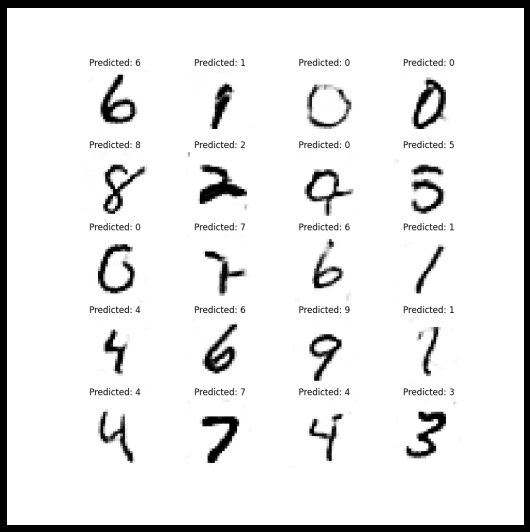
   Loss decreases more and generated images are visually good.

2) Gen_steps_per_iter = 1
   Dis_steps_per_iter = 5

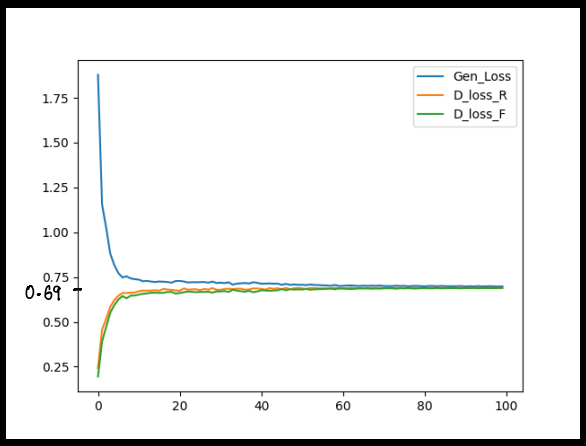
   Gen loss converge to a relatively high value and generated images are not that good.

③ Dropout rate in Discriminator

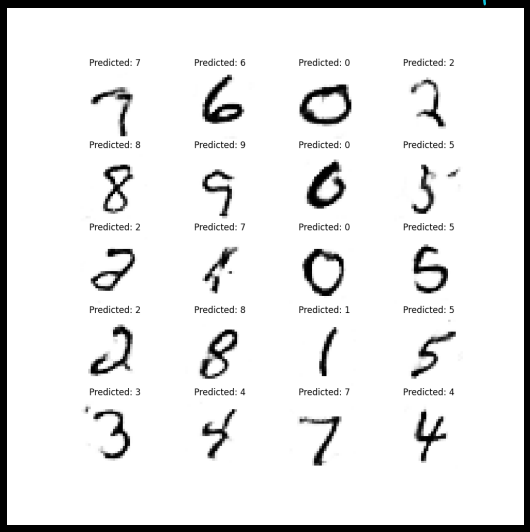
Dropout = 0.3

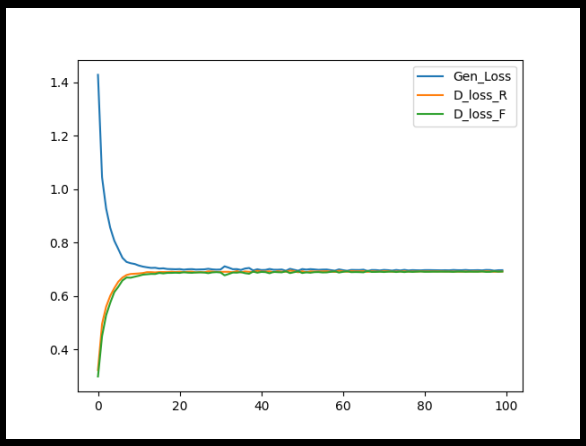


Generated Samples with their predicted classes



Dropout = 0.7



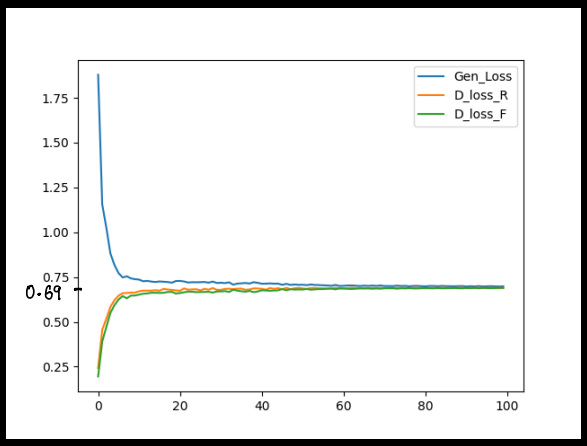


Observation :- Increasing dropout rate to 0.7 made convergence faster!

⑤ Implementing Self-Attention in GANs

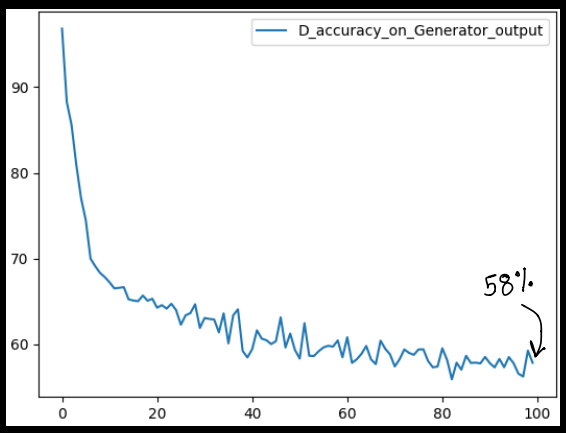
I implemented Self-Attention before the last conv. layers in both Generator and discriminator. [ref:- SAGANs by Zhang, Goodfellow et al.]

1) Without Self-Attention



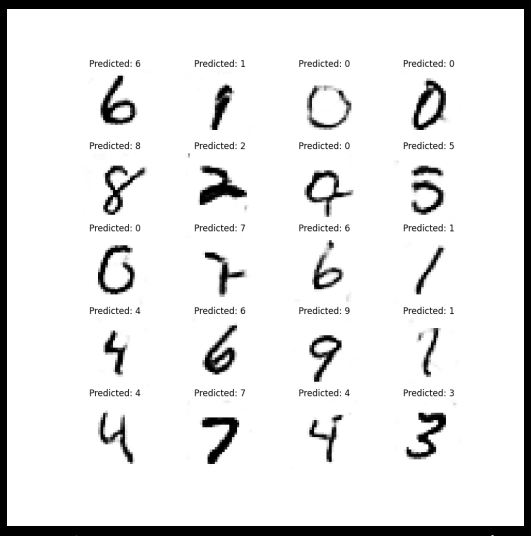
loss

Epochs



Accuracy

Epochs



Generated Outputs with predicted labels

SN in Generator = True

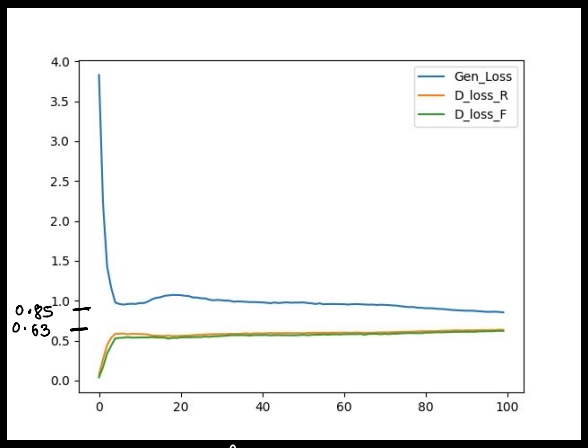
SN in Discriminator = True

D_steps_per_itr = 5

G_steps_per_itr = 5

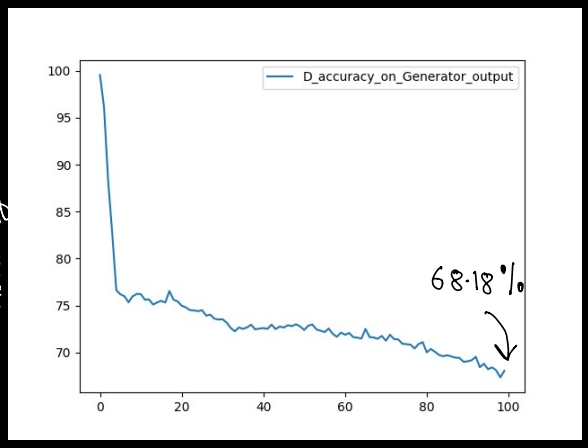
LR_G , LR_D = $3e^{-4}$

2) With Self-Attention



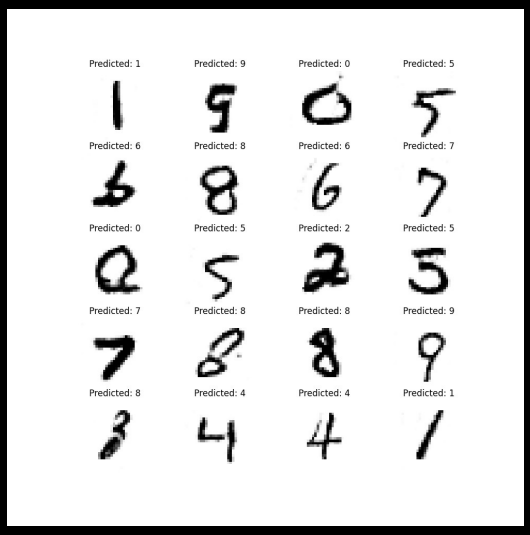
Loss

Accuracy



Accuracy

Epochs



Generated Images with predicted labels

SN in Generator = True

SN in Discriminator = True

D_steps_per_iter = 5

G_steps_per_iter = 5

LR_G = $1e^{-4}$

LR_D = $4e^{-4}$

Observation : ① Discriminator accuracy decreases consistently for self attention case while it goes up-down in without self attention case

② Generator loss is also consistantly decreasing for self-attention

case. It seems like running this for more epochs will give better results (loss, accuracy)