

Deep Learning: Implementation of DCGAN

Objective: Train a DCGAN to generate images from noise. Use the MNIST database to learn the GAN network and use VGG16 as a discriminator.

Perform the following tasks:

Google Colab Link:

https://colab.research.google.com/drive/1nP_5AT7Ty24QHnRRdP1ePwWtz1VOENLf?usp=sharing

Steps Involved:

1. Uniformly generated 10 noise vectors and used them as latent vectors to the generator for generating images.
2. Used my own Generator, and since my roll no%2 ==0, I used VGG16 as the discriminator.
3. Fine tuned the discriminator: removed the classifier layer and reduced the sequential layer from 30 to 16.
4. Trained the model for 20 epochs with batch size as 128.
5. As mentioned in the questions, save the generated sample image after the 1st, 9th and 19th epoch and plot the generator and discriminator loss of each iteration.

Hyperparameters Used:

Number of epochs	20
Batch Size	128
Generator Optimizer	AdamW
Discriminator Optimizer	AdamW
Optimizer's Learning rate	0.0002
Optimizer's beta	(0.5,0.999)
Loss	Binary Cross Entropy

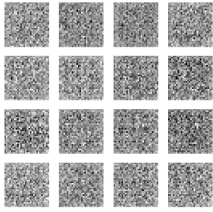
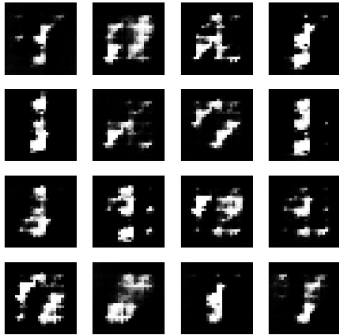
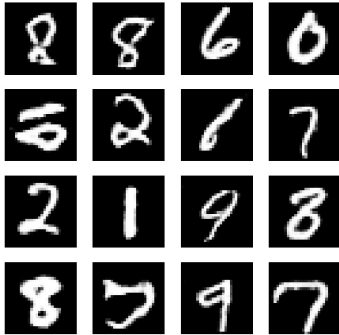
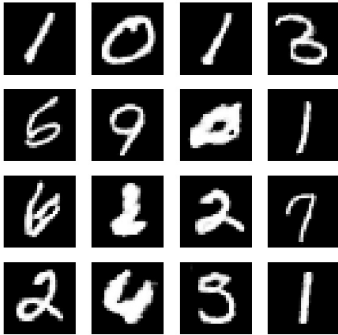
1.1. Uniformly generate ten noise vectors that act as latent representation vectors, and generate the images for these noise vectors, and visualize them at [5 + 5 + 5 marks]

i. After the first epoch.

ii. After $n/2$ th epoch.

iii. After your last epoch. (say n epochs in total) and comment on the image interpretation at (i), (ii) and (iii) and can you identify the images? [5 marks]

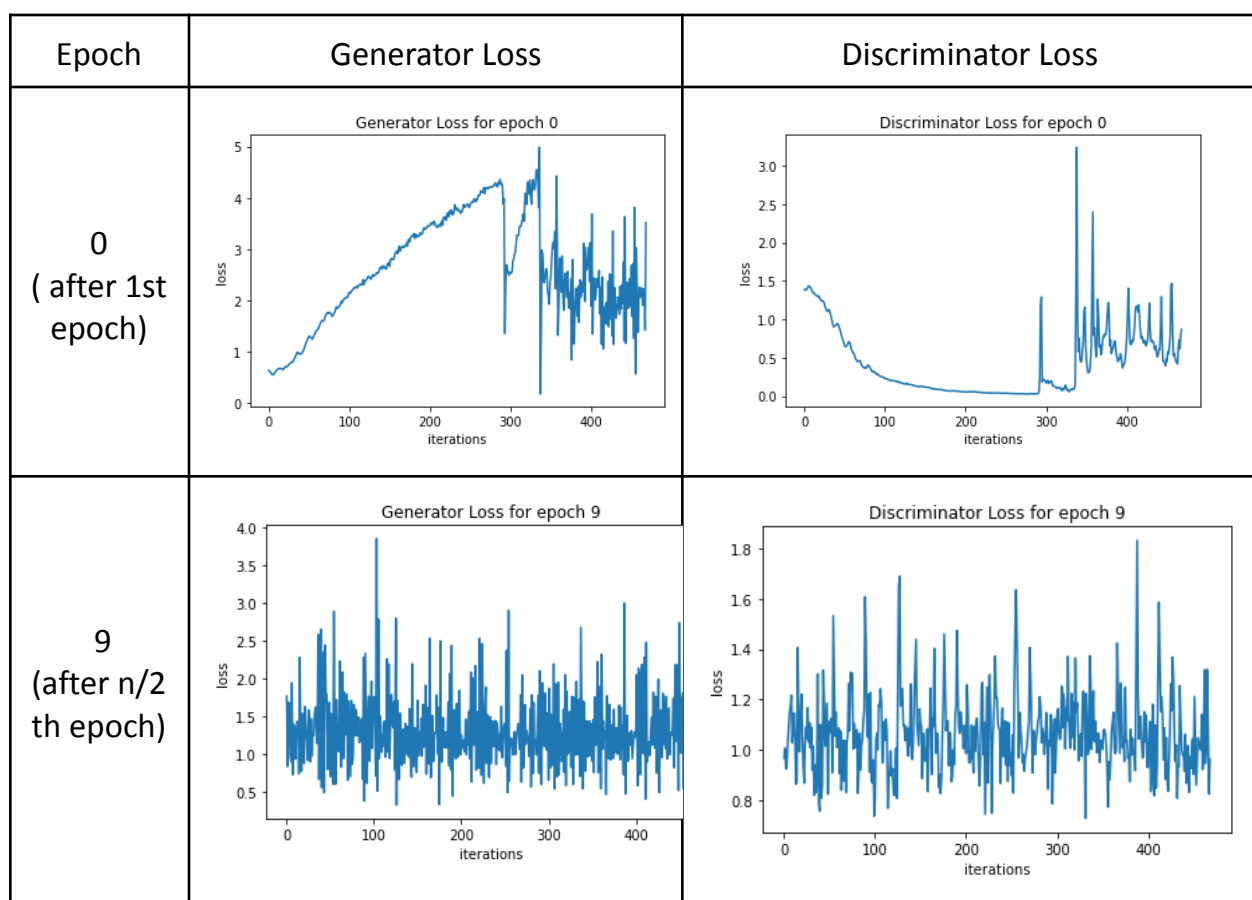
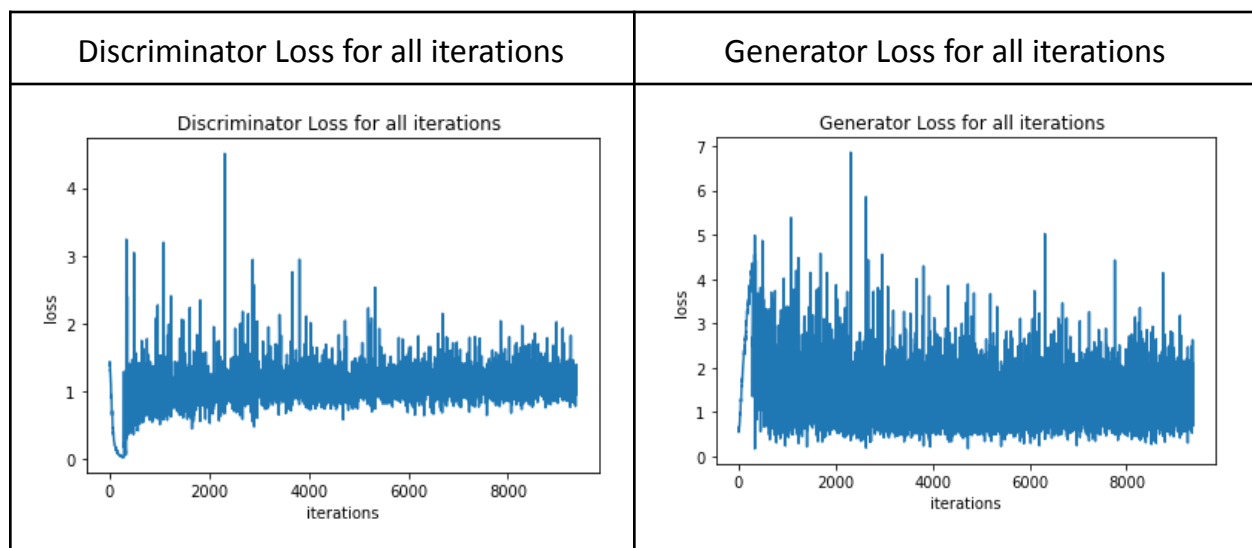
Generated 10 noise vectors and used them as a latent representation vector for the generator to generate the images from scratch. Below fig shows the image generated at epoch 0 (1st epoch) after 1st iteration. As we can clearly see that the generated image is full of noises, after each corresponding iteration in the epoch the generator starts learning and generating much better images.

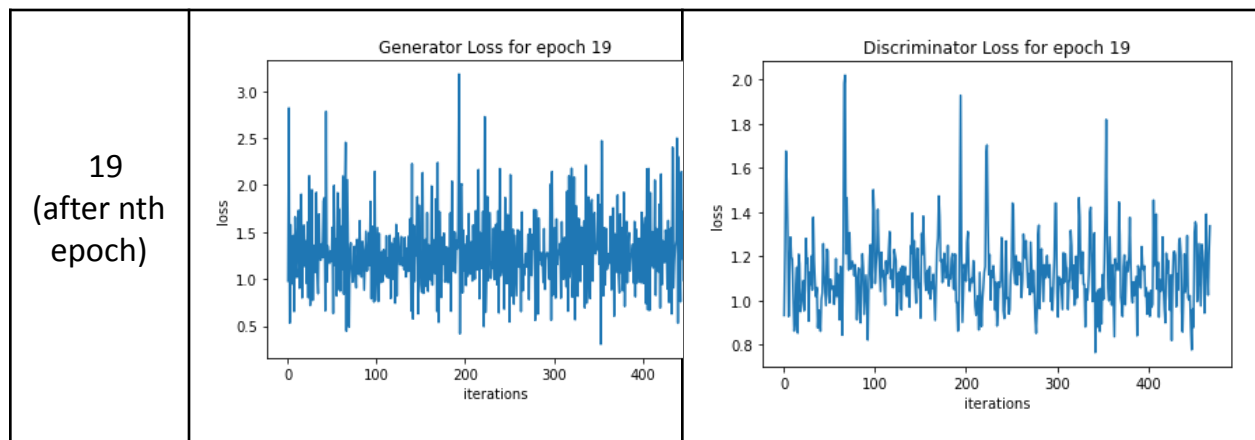
Image generated at epoch 0 (1st epoch) after 1st iteration.			
Image generated at 1st epoch (epoch 0)	Image generated at 9th epoch (n/2) epoch	Image generated at nth (19) epoch	
			

As we can see that with each epoch the quality of the generated image is improving. Initially the image generated has so much noise in it, but gradually when the generator and discriminator are learning the quality of the image is improving.

1.2. Plot generator and discriminator losses for all the iterations. [One iteration = forward pass of a mini-batch] [10 marks]

There are total 20 epochs, each epoch consisting of around 469 iterations (around 9380 iterations):





1.3. Do we have control of what class image the generator will generate, given a noise vector? Suppose, we are interested in generating only “4” images, will the GAN you trained in (1.1) can do that? Explain why. If not, modify the GAN trained above to do this. [10 + 40 marks]

In the previous implementation, *“no we cannot control what class image the generator will generate”* when noise is given as a latent representation to the Generator as the images generated using DC GAN are totally random and we cannot have a control over it.

If we are interested only in generating any particular image *“we need to use Conditional GANs”* (which focuses on targeted image generation) instead of DC GAN as we trained both the generator and discriminator model with conditioned class labels. In this way instead of generating a totally random sample, we have control over what to generate and are able to generate images of a given type or class label.

References:

[1]<https://github.com/znxlwm/pytorch-generative-model-collections/blob/master/CGAN.py>

[2]<https://machinelearningmastery.com/how-to-develop-a-conditional-generative-adversarial-network-from-scratch/>