# ECS795P Deep Learning and Computer Vision, 2023

by Abhishek Kumar Singh(220982845), Saurabh Sanjay Shrivastav (220447230)

## Coursework 2:
## Unsupervised Learning by Generative Adversarial Network

1. **What is the difference between an auto-encoder and a generative adversarial network considering (1) model structure; (2) optimized objective function; (3) training procedure on different components? (10% of CW2 Report)**

Ans. Autoencoders and GANs are popular neural network architectures used mostly for unsupervised learning. The key differences between these two models lie in the way they are trained, the model architecture, and the loss function.

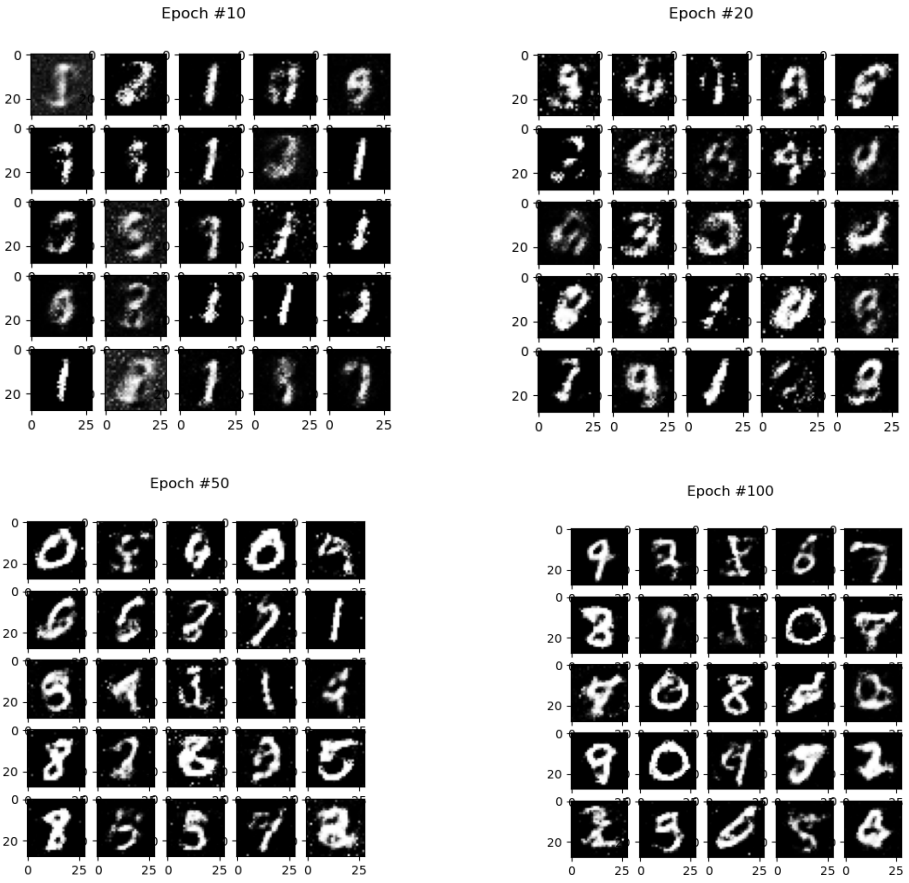
**Model Architecture**:
Auto-encoders model consists of two major parts, an encoder, and a decoder. The purpose of the encoder is to reduce the input data to a low-dimension space in a compressed manner, while the decoder reconstructs the data from this low-dimension representation.
The network learns the encoding/decoding because of the loss function. The loss increases with much disparity between the input and the output image, until after many epochs the network is good at finding an efficient compressed form of the input, while the decoder reconstructs the input from the encoded form.

Generative Adversarial Networks: GANs consist of a generator and discriminator. The generator takes noise as input and generates a fake sample that resembles the training data. The discriminator's job is to evaluate and distinguish between the generator's generated samples and the real data. Both generator and discriminator are trained in an adversarial manner with different losses.

**Loss Function:**
Autoencoders: The loss function used in the autoencoder is for minimizing the reconstruction error between the original and the reconstructed output by the decoder. The most commonly used objective functions in autoencoders are Mean Squared Error(MSE) and binary cross-entropy loss.
GANs: For GANs, there is a zero-sum game between the generator and the discriminator, hence that's the objective function or we can say that the objective function in this case maximizes the probability of the discriminator classifying the real and the fake data correctly at the same time minimizing the probability of the generator creating a fake that is easily identified as a fake by the discriminator.

**Training procedure:** Autoencoders: The training process of autoencoders is relatively straightforward, we train the encoder and the decoders together using backpropagation. Minimizing the reconstruction error is the aim and the error reduces means that the reconstructed input data is more similar to the original data.
GANs: The training process for GANs is a bit complicated and can lead to unstable training if not done properly. The generator and discriminators are training one after another, for example for every k epoch discriminator is training and then the generator is trained for 1 epoch. In every iteration, the generator-generated samples are evaluated by the discriminator. The loss is calculated and then gradients are backpropagated to update the parameters of both the generator and discriminator. This is repeated until equilibrium is reached that is the fakes generated by the generator are indistinguishable from the original data.

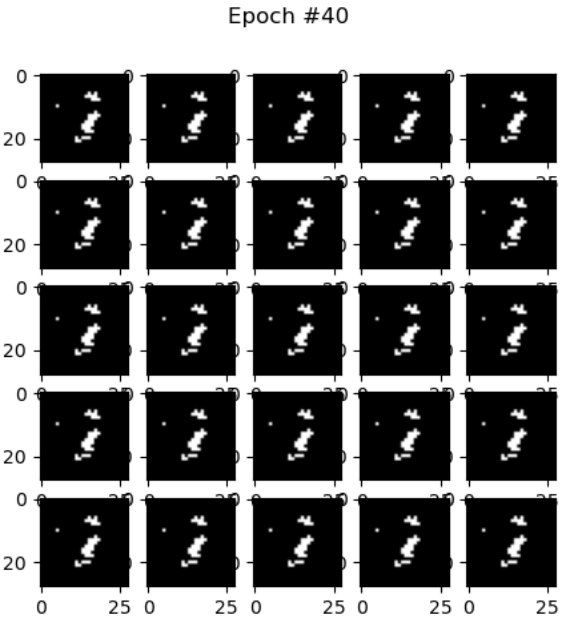
2. **Show the generated images at the 10th epoch, the 20th epoch, the 50th epoch, the 100th epoch by using the architecture required in Guideline (10% of CW2 Report)**

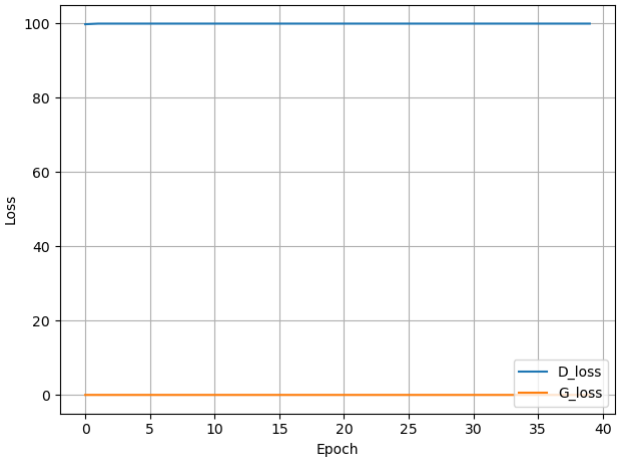



3. **(a). Change the learning rate to 0.01 and train for a few epochs to understand how the learning rate will influence the model outcome.**
**(b). Change the batch size to 256 and discuss how the batch size influences the model performance regarding its training speed and test accuracy. (10% of CW2 Report)**

a) The effect of changing the learning rate to 0.01 on the network is that the generator never learned to generate good-quality fakes which are indistinguishable from the original data and the discriminator becomes ineffective in successfully classifying them as fake. The generator kept generating repetitive low-quality images

```
100%|
Epoch 39 of 40 with 16.25 s
Generator loss: 0.00000000, Discriminator loss: 100.00000000
100%|
Epoch 40 of 40 with 16.31 s
Generator loss: 0.00000000, Discriminator loss: 100.00000000
Avg per epoch ptime: 16.23, total 40 epochs ptime: 680.04
```




The decision to select an optimal learning rate is really crucial for GANs to balance the convergence speed, and stability and generate good-quality images. There are some techniques like LR scheduling that reduces the learning rate over time to help in more stable training.

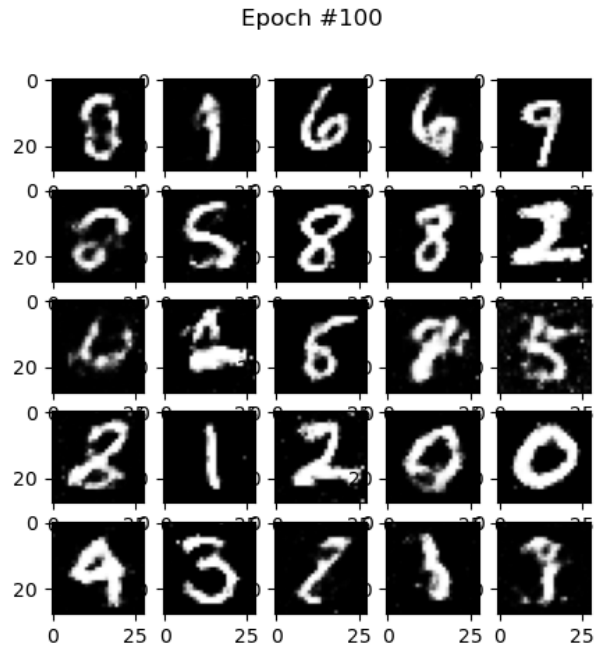
b) Model with batch size = 100

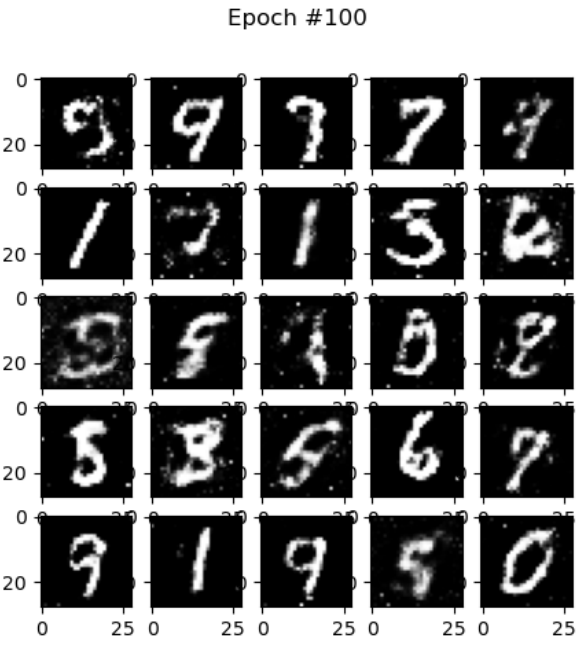
```
Epoch 100 of 100 with 16.40 s
Generator loss: 0.95533300, Discriminator loss: 1.24167691
Avg per epoch ptime: 15.86, total 100 epochs ptime: 1662.66
Training finish!... save training results
jovyan@jupyter-ec22395:~/deeplearning_CV/Coursework2$
```

Model with batch size = 256

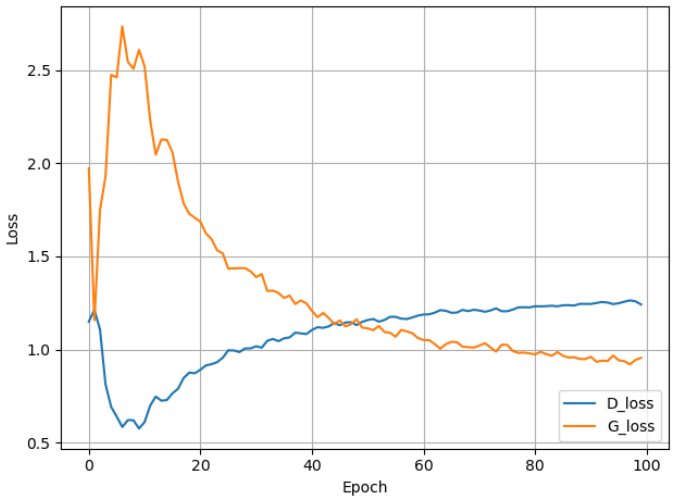
```
Epoch 100 of 100 with 11.65 s
Generator loss: 1.25981444, Discriminator loss: 1.07280142
Avg per epoch ptime: 10.89, total 100 epochs ptime: 1158.09
Training finish!... save training results
jovyan@jupyter-ec22395:~/deeplearning_CV/Coursework2$
```

Here is the side-by-side comparison of the 100th epoch results of the two models:

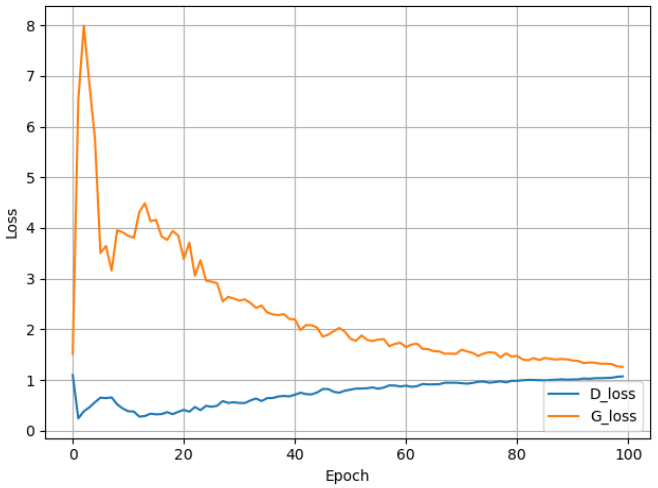



Model with batch size = 100




Model with batch size = 256




From the above data, we can conclude that increasing the batch size to 256 made the process of training faster, as it took more samples in each epoch, hence the model had faster convergence. But, we can also observe that the model performance was inferior to our old model with a 100 batch size. This can be attributed to a few factors, The model is probably performed poorly because of mode collapse, where the generator produces only a small subset of the total possible outputs. This is because of large batch size might have limited diversity of each example in each batch. Here we can observe the generator only produced a few good-quality sets of images rather than exploring the full space of possible images

4. **List some typical optimizers in deep learning. What optimizer was used for training in this case.(10% of CW2 Report)**

   Here are a few typical deep-learning optimizers:

   1. Stochastic Gradient Descent (SGD):
   2. Adam (Adaptive Moment Estimation):
   3. Adagrad (Adaptive Gradient):
   4. RMSprop (Root Mean Square Propagation):
   5. Adadelta (Adaptive Delta):
   6. Nadam (Nesterov-accelerated Adaptive Moment Estimation):

   In our case, we have used Adam(Adaptive Moment Estimation) optimizer

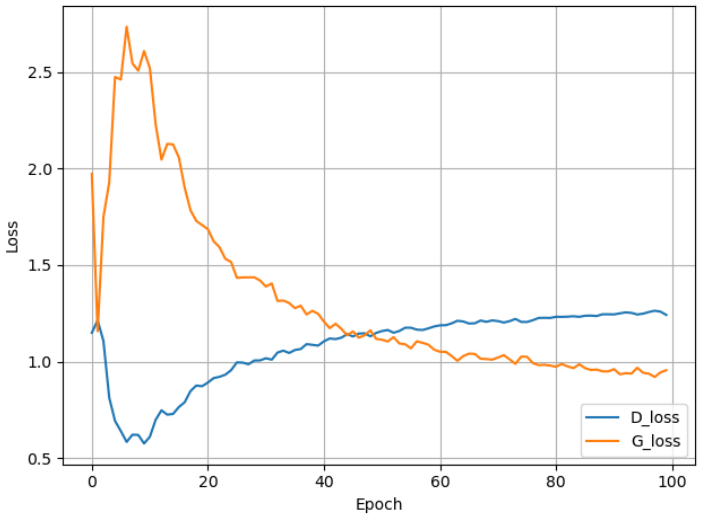
   Deep learning often uses the well-liked optimization approach known as Adam (Adaptive Moment Estimation). It combines the benefits of stochastic gradient descent (SGD) and RMSprop, two additional optimization techniques.

   Adam's goal is to monitor the first and second gradient moments, which are utilized to modify the learning rate for each network parameter. The average gradient is the first moment, and the average of the squared gradients is the second moment. These moments are used to calculate the gradient's variance and mean, respectively.

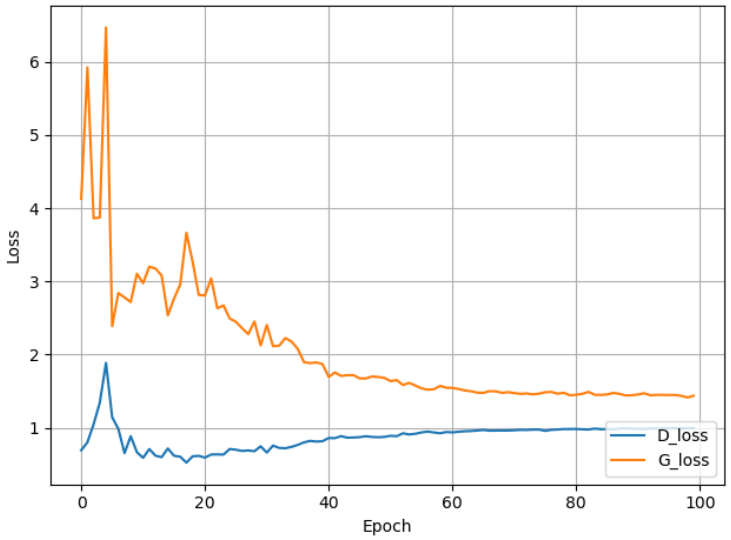
   The first and second-moment estimations are initialized to 0 at the beginning of the method. The gradients of the loss function are then calculated for each iteration with respect to each network parameter.

5. **Plot the loss curve during training. Remove dropout function for this architecture and plot its training loss (10% of CW2 Report)**

Loss curve with drop-out in discriminator:




Loss curve with dropout removed from discriminator:




With the removal of dropout, the discriminator is being overfitted and hence it is affecting the learning process of the generator. For a GAN network, it is important for the generator and the discriminator to be a good fit in tandem with each other. A discriminator directly impacts the performance of the generator, hence an overfit discriminator has not learned the complete representation of the data and thus it results in poor generated samples by the generator.

6. **Assume $p_g(x)$ is the distribution learned by the generator G, and $p_{data}(x)$ is real data distribution. $p_g(x)=p_{data}(x)$ when the discriminator D cannot tell the difference between these two distributions. Please Prove at this point, with a fixed G, the optimal Discriminator $D(x) = \frac{p_{data}(x)}{p_{data}(x)+p_g(x)}$ (25% of CW2 Report)**

Recall: A GAN is defined by the following min-max game:

$$min_G\, max_D\, V(G,D) = E_x(log(D(x))) + E_z(log(1 - D(g(z)))),$$

Hint: You can start by computing the partial derivative $\partial \frac{V(G,D)}{D(x)}$

Proof:

$$D(x) = \frac{P_{data}(x)}{P_{data}(x) + P_g(x)}$$

for fixed G.

$$\min_G \max_D V(G,D) = E_x[\log(D(x))] + E_z[\log(1-D(g(z)))]$$

We want to maximize $V(G,D)$

for that we will partially differentiate w.r.t $D(x)$ & equate to 0.

$$V(G,D) = \int_x P_{data}(x)\log(D(x))dx + \int_z P_z(z)\log(1-D(g(z)))dz$$

$$= \int_x (P_{data}(x)\log(D(x)) + P_g(x)\log(1-D(x)))dx$$

$$\frac{\partial V(G,D)}{\partial D(x)} = P_{data}(x)\cdot\frac{1}{D(x)} - P_g(x)\times\frac{1}{1-D(x)}$$

Equating to zero.

$$\frac{P_{data}(x)}{D(x)} - \frac{P_g(x)}{1-D(x)} = 0$$

$$\Rightarrow P_{data}(x)(1-D(x)) - P_g(x)D(x) = 0$$

$$\Rightarrow D(x) = \frac{P_{data}(x)}{P_{data}(x)+P_g(x)}$$

Hence Proved.

7. **Instability is one of the particular problems in the training of GAN. Answer why GANs suffer from the instability problem from the perspective of zero-sum game theory. Define the Wasserstein distance and how it improves the stability of GANs. (25% of CW2 Report)**

The generator and discriminator networks are basically engaged in a zero-sum game, GANs (Generative Adversarial Networks) hence they have an instability issue. In this game, the discriminator must properly distinguish between real images and fake ones created by the generator while the generator attempts to create realistic images that will deceive the player.

The GAN training process can occasionally become unstable, where one network dominates the other, resulting in subpar generated images or no convergence at all. The generator and the discriminator are trained simultaneously, with the objective of finding the Nash equilibrium. But it can be a really hard task in reality because in GANs, we deal with a high-dimensional parameter space and the training can get stuck in sub-optimal points or local minima, or it may fail to converge to a stable equilibrium point leading to oscillations or divergences.

The distance between two probability distributions is measured using the Wasserstein distance, also called the Earth-Mover's distance. It is applied as a loss function in the setting of GANs to swap out the discriminator's conventional binary cross-entropy loss. Between the distribution of actual images and the distribution of fake images produced by the generator, the Wasserstein distance gauges the difference. The quality of the generated images is improved by the generator learning to make images that are closer to the actual distribution by minimizing this distance.

One benefit of using the Wasserstein distance is that it gives the generator a more steady gradient, assisting in the prevention of the instability issue. In contrast to the binary cross-entropy loss, which saturates when the discriminator is sure in its classification, the Wasserstein distance is a smooth function that offers meaningful gradients even when the discriminator is confident. This stability promotes better convergence and higher picture quality by enabling the generator to learn more efficiently from the gradients calculated by the discriminator.