ECS795P Deep Learning and Computer Vision, 2020

Course Work 2:
Unsupervised Learning by Generative Adversarial Network

1. **What is the difference between supervised learning & unsupervised learning in image classification task? (10% of CW2)**

   ANSWER:

   In **supervised learning** we are given a set of images that are labelled or the training data. We train our classifier on these labelled images and use an algorithm to learn the mapping function for each input and its label. Then given a new image without a label our classifier predicts the label of the new image. For example we will be given a data set which consists of images that have numbers written in it and each image will be labelled for example the image with number 8 will be labelled as 8, now given a new image which has the number 8 but in different font and handwriting our classifier should be able to classify it. Examples of supervised learning are linear regression, logistic regression.

   In **unsupervised learning** we are not given any training data. Here our classifier has to learn the underlying structure of the dataset and predict the classes of the labels. For example, if we consider a data set which has images which consist of numerical digits in it in different fonts, size and handwritings our classifier should be able to distinguish all the images that have number 8 together. Examples of unsupervised learning are clustering, nearest neighbor classification.

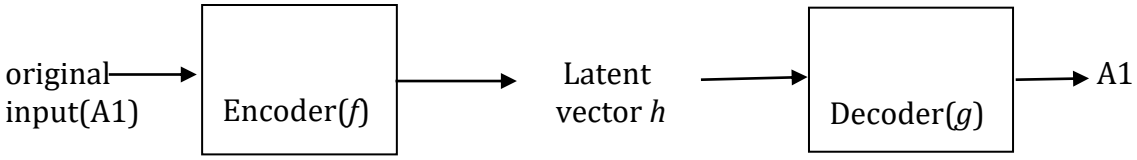
2. **What is the difference between an auto-encoder and a generative adversarial network considering (1) model structure; (2) optimized objective function; (3) training procedure on different components. (10% of CW2)**

   ANSWER:

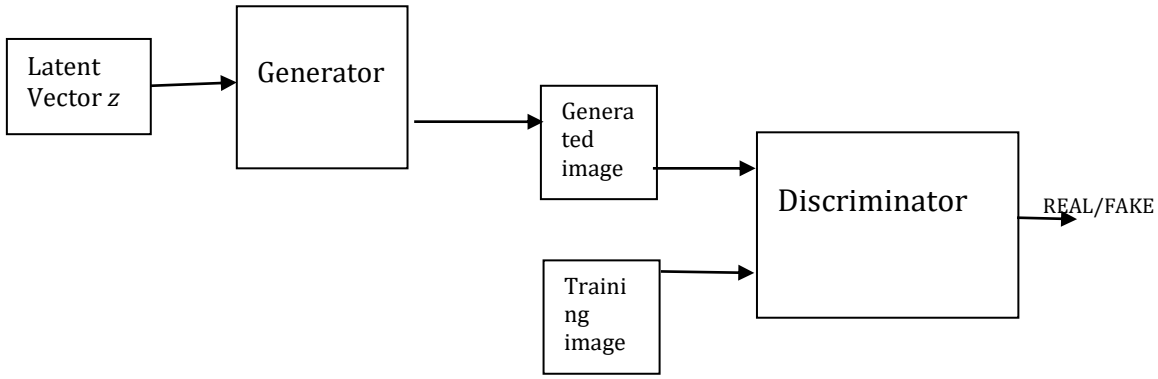
Auto-encoders and GANS both belong to the family of deep generative models that is they learn the data distribution rather than the density of the data. Auto-encoder learns to generate output data as close as to the input data.

Model structure:

auto-encoders comprise of an encoder and a decoder. The encoder is used to compress the image into a latent vector and the decoder is again used to reconstruct the compressed image from vector back to original input.



GANS consist of a generator and a discriminator, where the generator tries to mimic the actual data samples and the discriminator tells whether the sample produced is real or fake. GANS use gaussian random noise as input *Z* to generate samples



Optimized objective function:

Auto-encoders learn encoder *f* and decoder *g*. their objective is to minimize the reconstruction L2 loss.

$$L(x,y;theta) = -(1/M)\sum i=1 (||xi - ri||)^2$$

GAN consists of a generator and a discriminator; the generator generates real looking images, and the discriminator learns to tell the generated images from

the ground-truth images. The objective function is that of a two player min-max zero sum game-

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Training:

In generative auto-encoders training is done by Enforcing the tradeoff between mixing and power of reconstruction generation. Whereas in a GAN we synchronize the discriminator along with the generator. rather than fitting a separate discriminative model, the generative model itself is used to discriminate generated data from samples a fixed noise distribution. In the autoencoders we train our encoder and decoder while in the GANS we train generator and discriminator simultaneously.

3. **How is the distribution $p_g(x)$ learned by the generator compared to the real data distribution $p(x)$ when the discriminator cannot tell the difference between these two distributions? (15% of CW2)**

ANSWER:

In the beginning the generator implicitly produces a random sample distribution Z and then it is passed to the deep network of the Generator *G(z,theta)* which is then passed to the Discriminator. The objective of the Generator here is to achieve global optimum where $P_g = P_{data}$ in other words to fool the discriminator. To achieve global optimum it has to keep updating its bias and weights *'theta'.* Now how can it update its weight? According to Goodfellows algorithm as explained in his paper the *GANS* use mini batch stochastic gradient descent. Where K number of steps are applied and K is the hyper parameter:
For K steps:

1) Sample mini-batch (z(1)……z(m)) from $P_g(z)$

2) From data generating distribution P_{data} we sample mini batch of m examples as above step (x(1)……x(m))

3) By ascending its stochastic gradient we update the discriminator:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

4) After training discriminator we the train our generator. we sample minibatch of m noise samples {z(1).....z(m)} from $P_g(z)$.

5) The update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

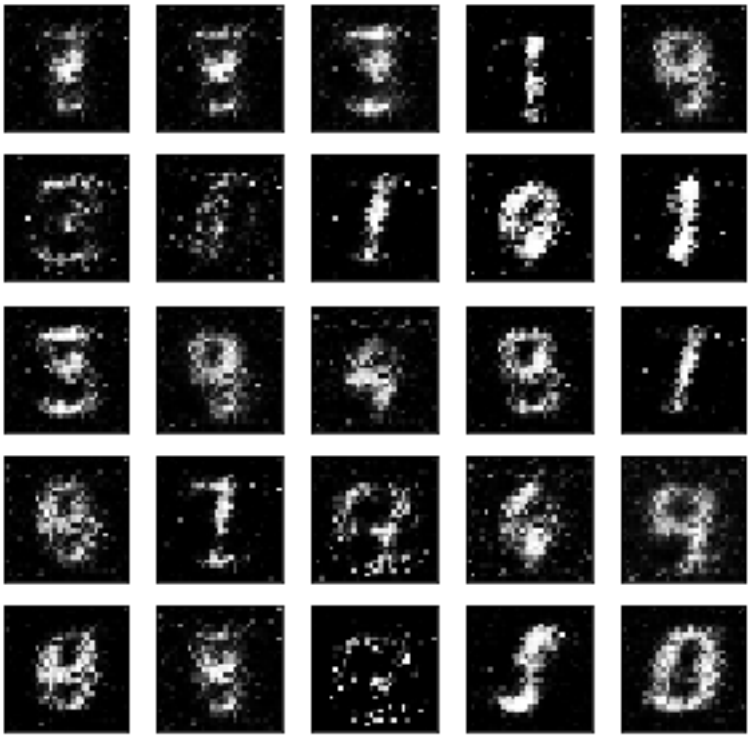
In simple words after the discriminator is trained, Every time the discriminator gets the right output that is *D(G(z))*==0 the loss is calculated and weights are updated as D and G both are differentiable functions. With every update made to the weights of the deep network of Generator *G(z)* come close to $P_{data}(x)$. hence when $P_{data}(x) == G(z)$ or *D(G(z))==1* the output of the discriminator will be 0.5 which means that now the discriminator has failed to tell the difference or failed to identify the input fed from generator was fake. To reach to that solution we iteratively update the loss function and want to achieve the $\min_G \max_D$ as proposed by Goodfellow in the given equation:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

When the output of the discriminator is 0.5, which is our goal we say that there is very less difference between the real data distribution and the data distribution $P_{data}(x)$ by Generator G(z) i.e. they are very similar to each other.

4. **Show the generated images at 10 epochs, 20 epochs,50 epochs,100 epochs by using the architecture required in Guidance. (15% of CW2)**
ANSWER:

1) 10 epochs-

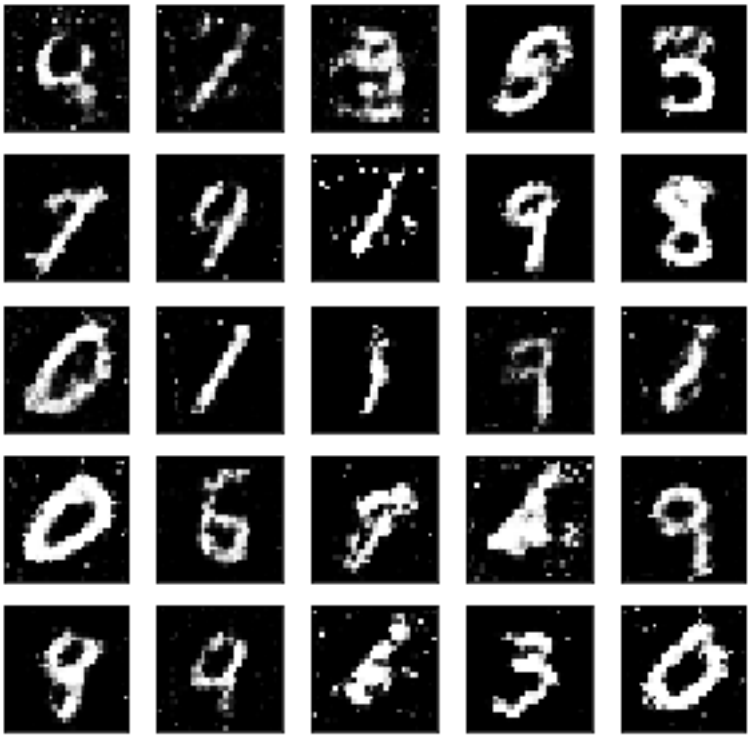


Epoch 10

2) 20 epochs-

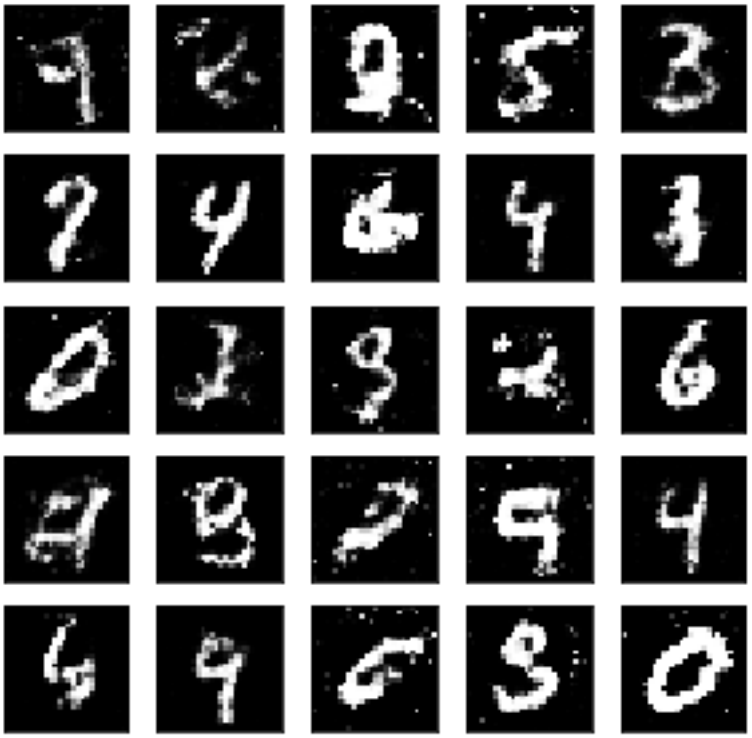
Epoch 20

3) 50 epochs-



Epoch 50

4) 100 epochs-



Epoch 100

Loss-