# ECS767P Emerging Topics in Machine Learning and Computer Vision, 2018

**Name : Tanmaiyii Rao**
**Student ID : 140361229**

## 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)**

When you use supervised learning for image classification, you will need large amounts of labelled training data. In supervised learning, the system is presented with numerous examples of images that must be manually labelled. Using this training data, a learned model is then generated and used to predict the features of unknown images. Such traditional supervised learning techniques can use either generative or discriminative models to perform this task. Unsupervised learning systems find patterns of values in the image data and build a probabilistic model or models from multiple unknown images. "Unsupervised techniques are useful to identify sub-structures of the problem such as sub-types of defects or product varieties. By interpreting the clustering results, an expert may understand the problem better and develop a more accurate supervised classifier. In some cases, however, the data clusters generated by unsupervised learning systems may not be accurate. In such cases, semi-supervised learning techniques can be used that incorporate labelled data to reduce the amount of images that need to be trained by an operator while improving the accuracy of unsupervised clustering. GANs are unsupervised learning algorithms that use a supervised loss as part of the training. The GAN sets up a supervised learning problem in order to do unsupervised learning, generates fake / random looking data, and tries to determine if a sample is generated fake data or real data. This is a supervised component, yes. But it is not the *goal* of the GAN, and the labels are trivial.

### Supervised Learning

- **Data:** $\{(x_i, y_i)\}_{i=1,2,\ldots,n}$, label $y_i$ is given during training
- **Goal:** Learn a mapping function: $f(): x \rightarrow y$
- **Examples:** Image classification

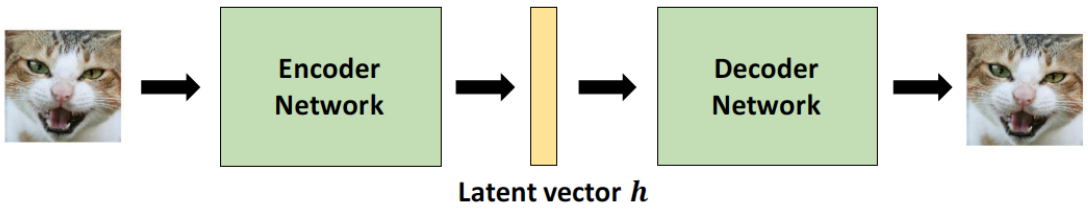
### Unsupervised Learning

- **Data:** $\{(x_i)\}_{i=1,2,\ldots,n}$, no label $y_i$ is given during training
- **Goal:** Learn some underlying structure of data $x$
- **Examples:** clustering

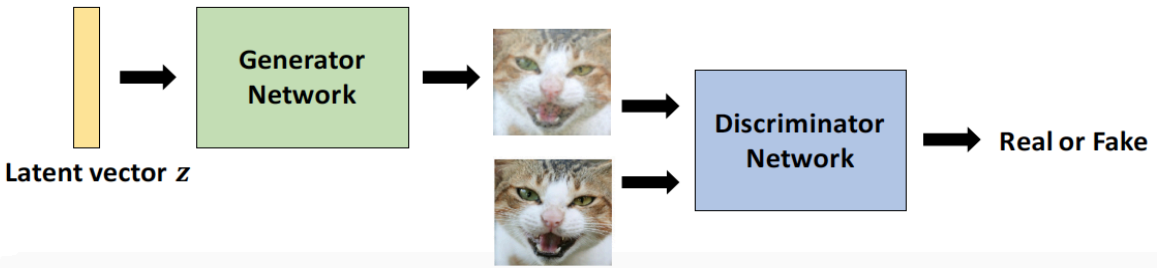
2. **What is the difference between an auto-encoder and a generative adversarial network considering (1) model structure; (2) optimized objective function; (3) training procedure on different components. (10% of CW2)**

The difference between an auto-encoder and a generative adversarial network (GAN) in terms of:

1) **Model Structure** : Auto-Encoder is a type of neural network that learns to generate the output data $r$ as close as to the input $x$ as possible. Auto-Encoder consists of an encoder network (map input x to latent code h) and a decoder network (map h to reconstructed input r).

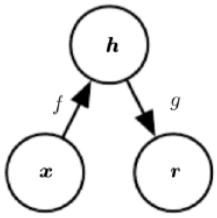



Contrary to Auto-Encoder, GANs aim to generate samples from a simple distribution, i.e. use Gaussian random noise as input $z$. A type of generative models that learn to generate samples through a two-player minimax game. GAN consists of a generator network (learn to fool the discriminator by generating real looking images (as real as possible to the ground-truth)) and a discriminator network learn to differentiate between the generated images (fake) and the ground-truth images (real).




2) **optimized objective function :**

**Auto-Encoder =** Learn encoder $f$ and decoder $(f, g)$: $x \rightarrow h \rightarrow r$




- Encoder: Project the higher-dimensional input data $x$ into a lower dimensional latent space
- Decoder: Reconstruct the lower-dimensional latent code $h$ into an output $r$ that resemble input $x$.
- Objective: Minimizing reconstruction L2 loss:

$$L(x, y; \theta) = -\frac{1}{M}\sum_{i=1}^{M} ||x_i - r_i||^2$$

**GAN**s = Whereas in the case of GANs the objective is a two-player minimax game.

- Discriminator Network: maximise the objective such that D(x) is close to 1, i.e. when given real images x; while D(G(z)) is close to 0, i.e. when given fake images G(z).
- Generator Network: minimise the objective such that D(G(z)) is close to 1, i.e. try to generate image G(z) look as genuine as the real image x
- Objective : Minimax game

$$\min_{\theta_g}\max_{\theta_d}\left[\mathbb{E}_{x\sim p_{data}}\log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z\sim p(z)}\log(1-\underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}})\right]$$

3) **training procedure on different components:**
Auto-encoder trains both encoder and decoder to make the output same with input image minimizing loss function. While for GAN, it also tries to train both discriminator and generator. For the number of steps, updating the discriminator by ascending its stochastic gradient. After update the discriminator, update the generator by descending its stochastic gradient.

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)**

The convergence is reached when the discriminator fails to discriminate between the real images and the generated images by the generator. That is when
D(x) = D(G(z)) = 1/2. In other words, the optimal discriminator given the generator is $D^*_G = ½$.
Only when p(g) = p(data), the virtual training criterion of the generator is reached. The distribution $p_g(x)$ is learned in the following way:

**Theorem 1.** *The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{data}$. At that point, $C(G)$ achieves the value $-\log 4$.*

*Proof.* For $p_g = p_{\text{data}}$, $D^*_G(x) = \frac{1}{2}$, (consider Eq. 2). Hence, by inspecting Eq. 4 at $D^*_G(x) = \frac{1}{2}$, we find $C(G) = \log\frac{1}{2} + \log\frac{1}{2} = -\log 4$. To see that this is the best possible value of $C(G)$, reached only for $p_g = p_{\text{data}}$, observe that

$$\mathbb{E}_{x\sim p_{\text{data}}}[-\log 2] + \mathbb{E}_{x\sim p_g}[-\log 2] = -\log 4$$

and that by subtracting this expression from $C(G) = V(D^*_G, G)$, we obtain:

$$C(G) = -\log(4) + KL\left(p_{\text{data}}\,\middle\|\,\frac{p_{\text{data}}+p_g}{2}\right) + KL\left(p_g\,\middle\|\,\frac{p_{\text{data}}+p_g}{2}\right) \quad (5)$$

Where KL is the Kullback–Leibler divergence.

The Jensen– Shannon divergence (JSD) between the model's distribution and the data generating process for the previous equation :

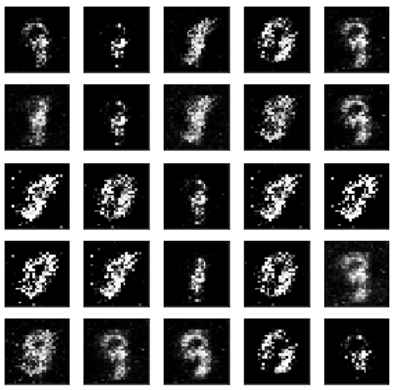
$$C(G) = -\log(4) + 2\cdot JSD\left(p_{\text{data}}\,\|\,p_g\right)$$

As JSD = 0, (and non-negative if they are equal)

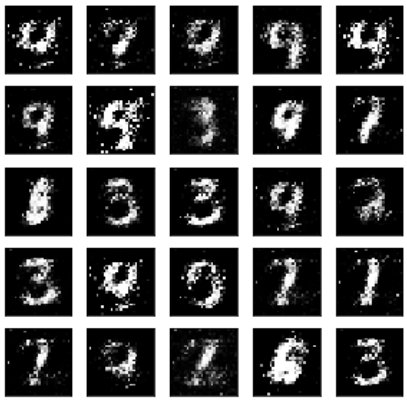
C* = - log(4) is the global minimum of C(G) and that the only solution is $p_g = p_{data}$

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

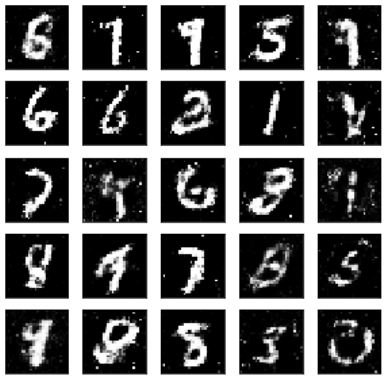
The images generated using normal architecture with dropout :



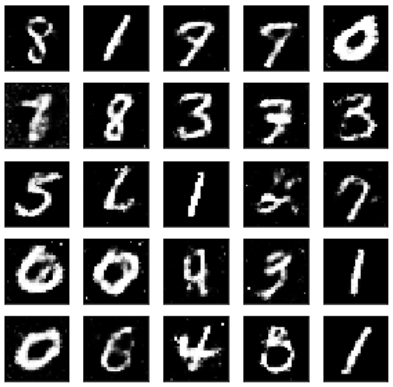
Epoch 10



Epoch 20

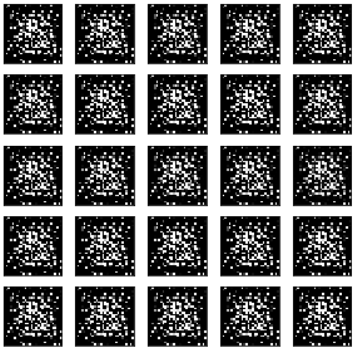


Epoch 50

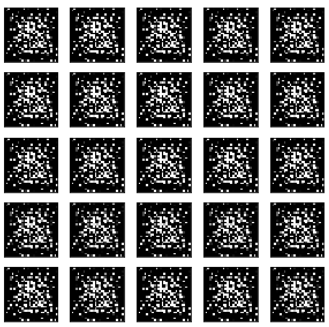


Epoch 100

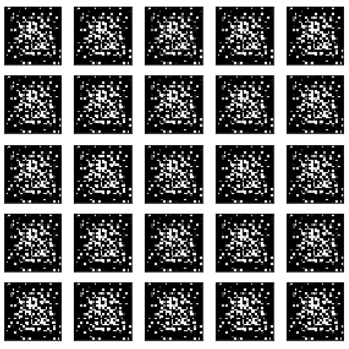
Images without dropout :



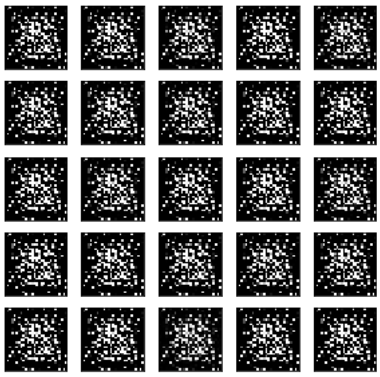
Epoch 10



Epoch 20



Epoch 50



Epoch 100

As you can see, the images without dropout are not accurate at all. This is because the loss functions of the discriminator and generator never converge when dropout is removed or set to zero.

**References**


- Unsupervised Learning by Generative Adversarial Networks - Yanbei Chen, Shaogang Gong - Queen Mary University of London
- Generative Adversarial Nets - Ian J. Goodfellow_ , Jean Pouget-Abadiey , Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozairz , Aaron Courville, Yoshua Bengiox
- https://www.vision-systems.com/articles/print/volume-20/issue-2/features/machine-learning-leverages-image-classification-techniques.html