# Math Related to GANs

What is GAN?

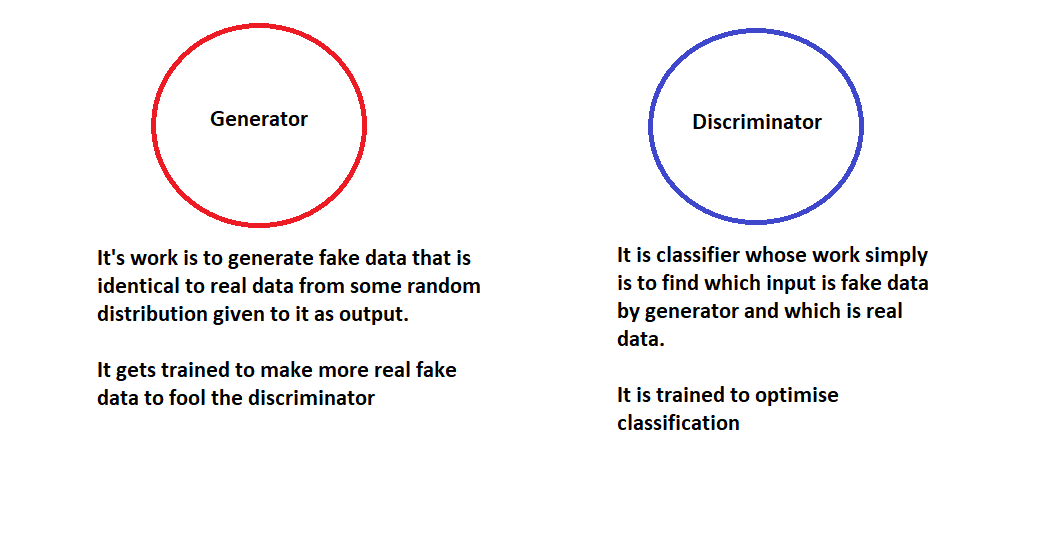
Generative Adversarial Network is an unsupervised learning model . It approaches to generative modelling using deep learning methods, such as convolutional neural networks.

What it contains?

The Generative Adversarial Network (GAN) comprises of a generative model **G** and a discriminative model **D**. The generative model can be considered as a counterfeiter who is trying to generate fake currency and use it without being caught, whereas the discriminative model is similar to police, trying to catch the fake currency. This competition goes on till the counterfeiter becomes smart enough to successfully fool the police.

How it works

It contains 2 models:



The generator **G** tries to fool the discriminator into believing that the input sent by generator is real

→While the discriminator **D** catches the generator by identifying that this is fake.

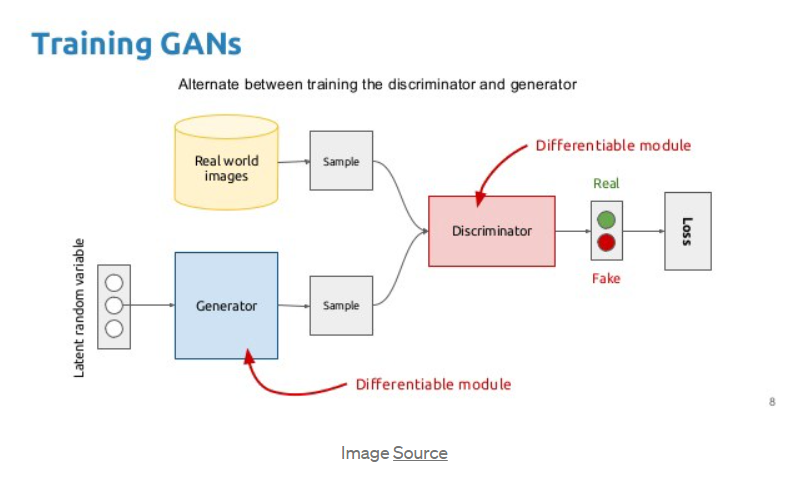
→Then after cached by the discriminator D , the generator G learns to produce similar real type of training data inputs.

→ And this process is repeated for a while or until Nash equilibrium found.

This process is called **Adversarial Training.**

Basically generator improve its output to be more like real data so that it can fool discriminator

**Training**

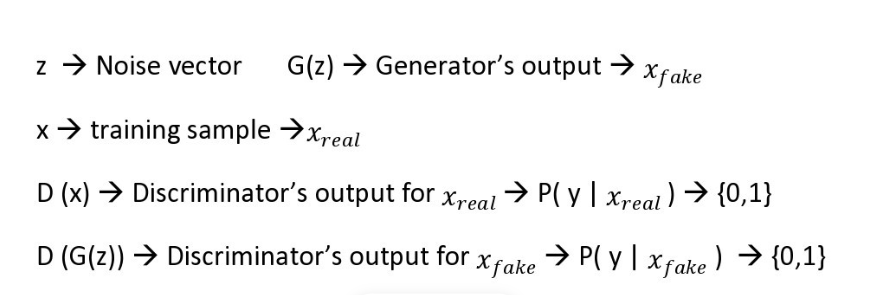


**GAN’s training process step by step.**

1. We take some noise from random distribution, then we feed it to the **Generator G** to produce the fake x (label y=0) → (x, y) input-label pair.
2. We take this **fake pair** and the **real pair** x (label y =1) and feed it to the **Discriminator D**alternatively.
3. The **discriminator D** is a binary classification neural network so it calculates the loss for both **fake x** and **real x** and combine them as the final loss as **D loss**.
4. The **generator G** also calculates the loss from it’s noise as **G loss**since each network has a different objective function**.**
5. The two losses go back to their respective networks to learn from the loss (adjusting the parameters w r t the loss).
6. Apply any optimization algorithm (Grad descent, ADAM, RMS prop, etc.) Repeat this process for certain no of epochs or as long as you wish.

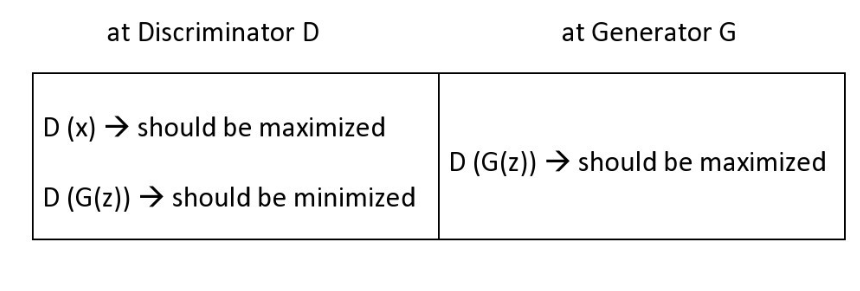
Each network has goals so these two networks pit against each other during the training.**The generator G**gets stronger and stronger at generating the real type of results and the **discriminator D** also gets stronger and stronger at identifying which one is real , which one is fake

Mathematics

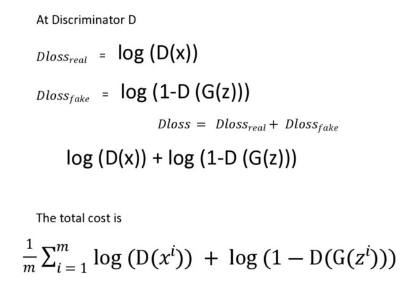
Let :

Discriminator – It is a binary classifier so when we feed the real data , the model should produce high probability for the real data and low probability for fake data( generator’s output)

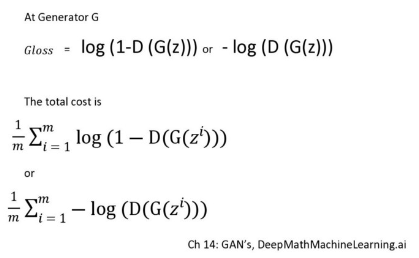
Generator – It is a model who wants to make fake picture as much close to real as possible so that discriminator consider as real input.



Now we will calculate loss function which is something models are penalized by for not getting optimized.



The discriminator wants to max this eq.

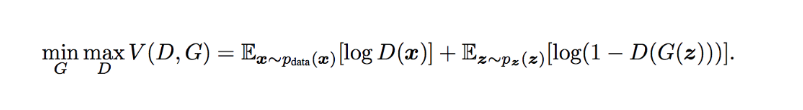


The generator will try to min this eq.

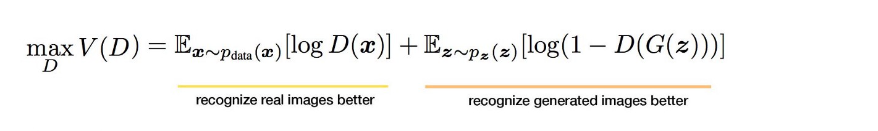
the discriminator network runs *twice*(one for real , one for fake) before it calculates the final loss while generator runs only *once*.

In pratical application these loss function are used by GANs.

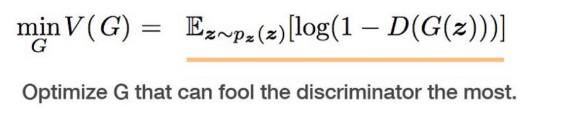
In Ian’s papers this this is represented in the expectation form in one formula only.



Here this formula shows the min-max game between discriminator and generator, where the discriminator seeks to maximize the given quantity whereas the generator seeks to achieve the reverse. In other words, the generator wants to max the below function(derived above)



And the generator wants to min the below function



**Training**

Once we got these two losses, we calculate the gradients w.r.t their parameters and back propagate through their networks independently .Like while keeping generator parameters constant we will try to train discriminator by maximizing its loss function . Whereas when we train generator we keep discriminator parameters constant and optimize generator by reducing its loss function.

Reference:

<https://jaketae.github.io/study/gan-math/>

<https://medium.com/deep-math-machine-learning-ai/ch-14-general-adversarial-networks-gans-with-math-1318faf46b43>

<https://youtu.be/8L11aMN5KY8>

<https://youtu.be/TpMIssRdhco>