GAN Specialization (deeplearning.AI)

Sharon Zhou

$\boxed{\text{Artforger}}$ $\boxed{\text{Art Inspector}}$

Generator Classifier

GANs are unsupervised technique

Week 1: Fundamental Components of GANs

Week 2: Deep Conv GANs

Week 3: Wasserstein GANs with Gradient Penalty

Week 4: Conditional GAN and Controllable Generation

Week 1:

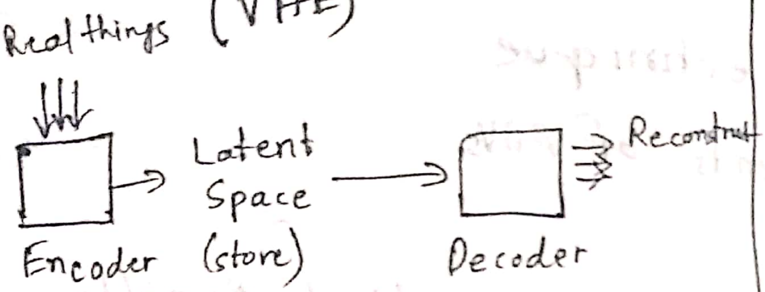
Generative Models:

Generative vs Discriminative models

Discriminative models	Generative Models:
Features Class $X \rightarrow Y$ $P(Y/X)$	Try to make a realistic Representation Noise Class Features $\xi, Y \longrightarrow X$ $P(X\|Y)$ Random set of values adding noise would generate diverse representation

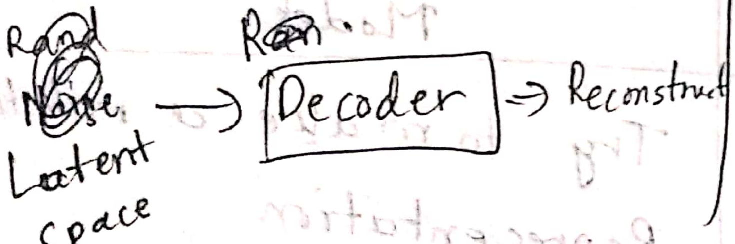
Generative Models: ~~(Variational Encoders, Ge~~

Variational Encoders (VAE)

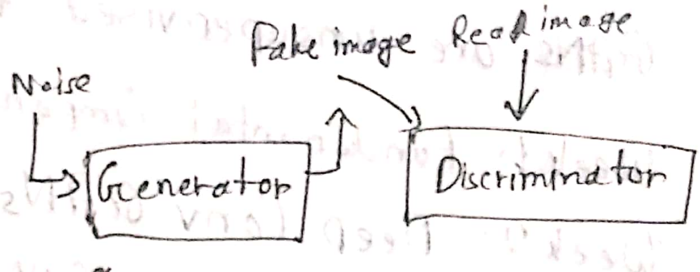


This inject some noise into this whole model and training process

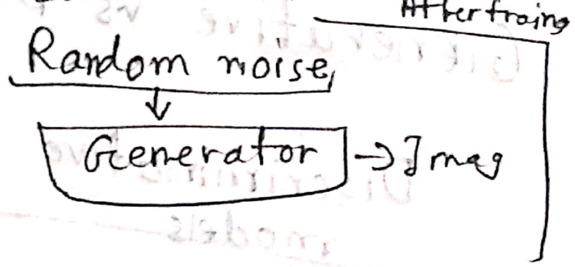
After training



Generative Adversarial Networks



Generator takes some random noise input & decoder, These two models compete with each other.



- Generative models learn to produce realistic examples
- Discriminative models distinguish between classes

Real life GANs

StyleGAN →
Cycle GAN (Image translation)
Generative Design

Intuition Behind GANs (Works by ~~computatio~~ competing between generator and discriminator)

Generator	Discriminator
learns to make fakes that look real	learns to distinguish real ~~an~~ from fake

⇒ Generator tries to fool the discriminator.
Discriminator is an ~~expert~~ inspector. It learns how to not get fooled by ~~is~~ the generator.

→ The generator isn't allowed to see the real images. Initially it generates a masterpiece of scribbles

→ The discriminator is allowed to look at the real images and distinguish between the real and fakes.

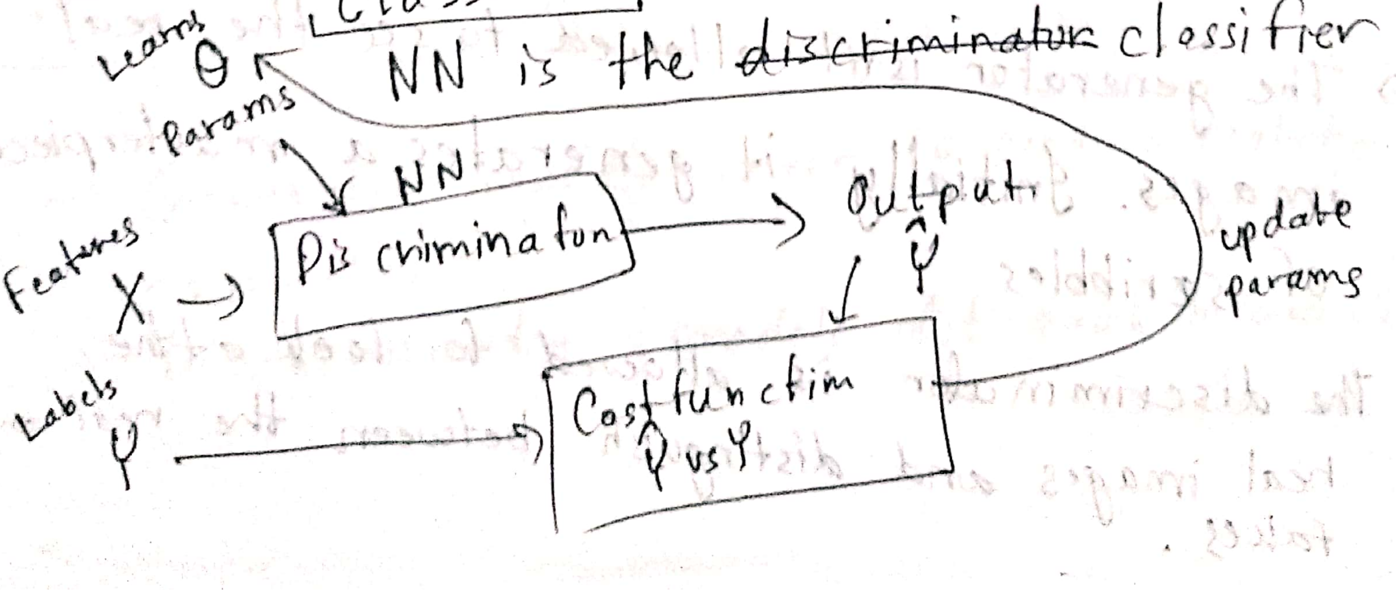
⇒ To start the competition, we train the the discriminator using the real artwork so that it knows which images are real. During training, we let the discriminator know which ones are real and which ones are fake so that it can turn out to be a good discriminator.

→ Generator ~~know~~ will know in what direction to go on and improve, by looking at the scores assigned to it's work by the discriminator.

Discriminator: (goal is to distinguish between classes)

Classifier

NN is the ~~discriminator~~ classifier

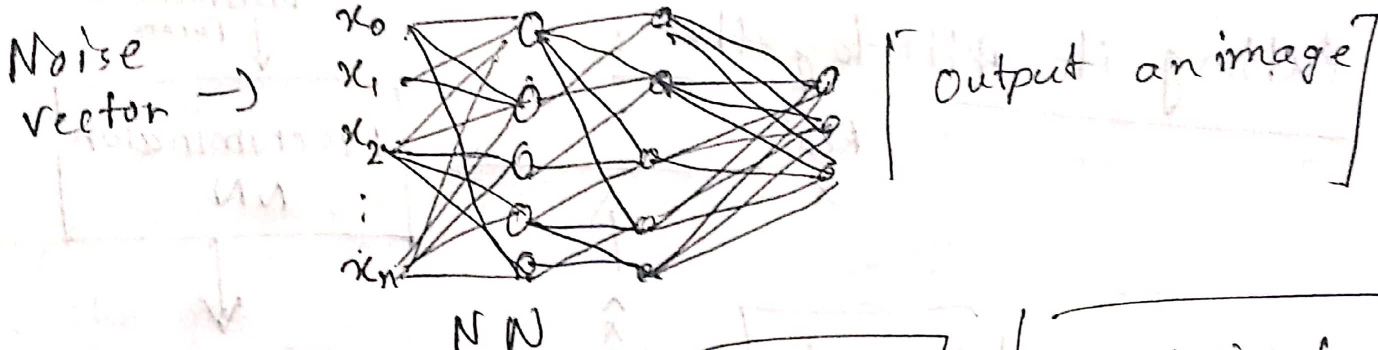


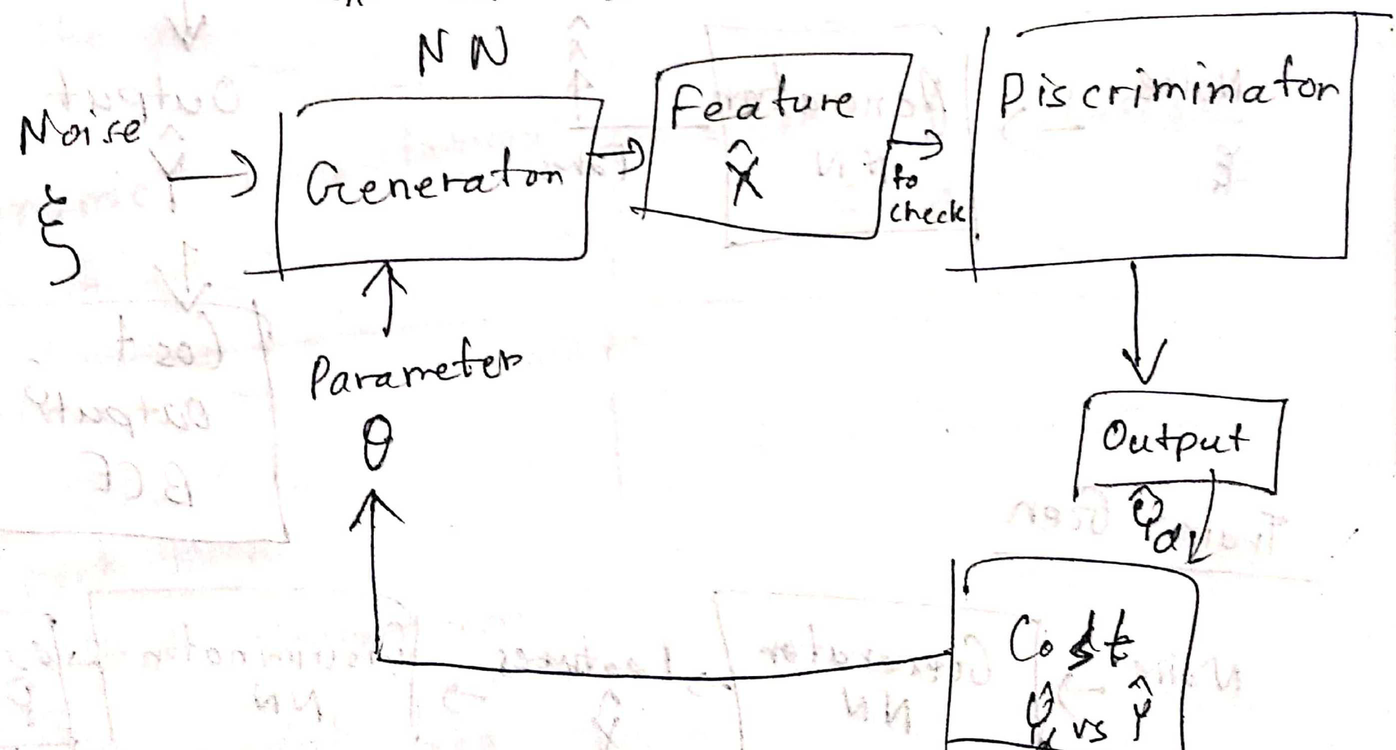
$\underbrace{P(Y|X)}$ → how fake the image is

⇓

This will be given to the generator so that it can generate better next time

Generator ⇒





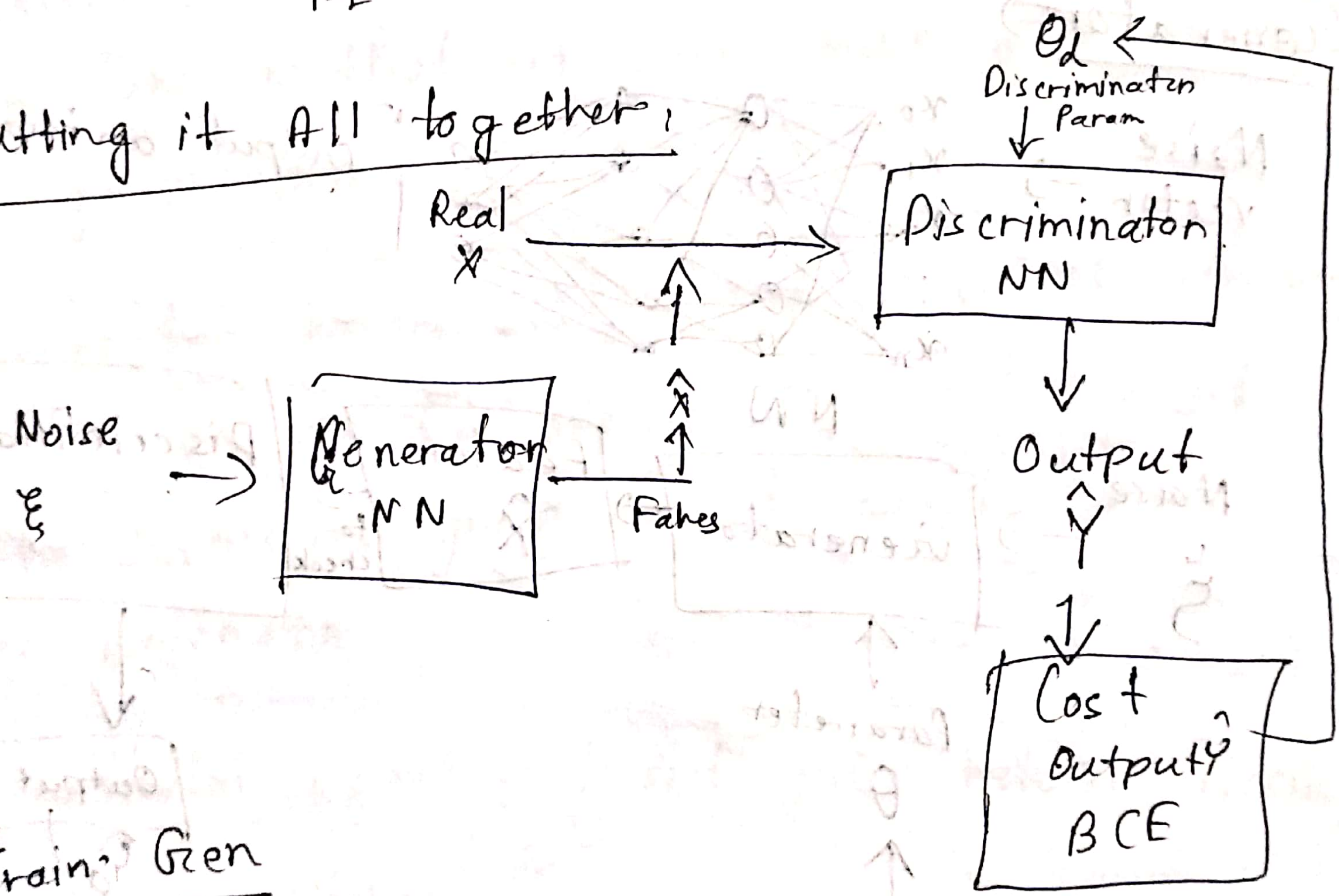
$\underbrace{P(X|Y)}$ → Generate

BCE Cost Function:

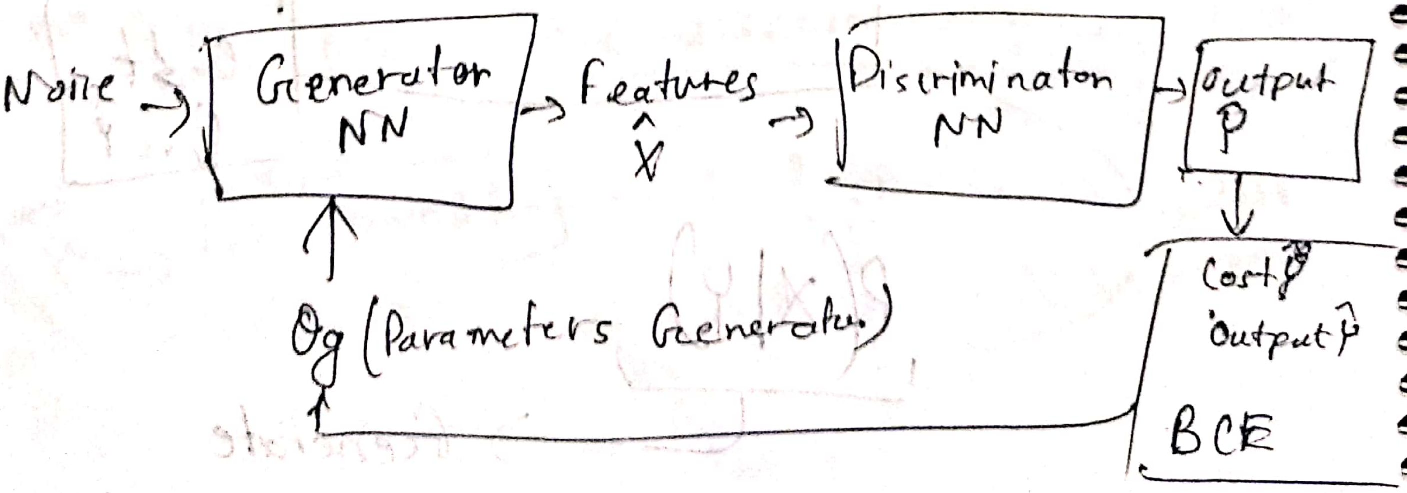
Binary Crosss Entropy

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log h(x^{(i)},\theta) + (1-y^{(i)})\log(1-h(x^{(i)},\theta))\right]$$

Putting it All together,



Train Gen



Both model should be kept at similar skill level

Discriminator output

Superior Discriminator → Fake as 100% fake → no way to improve

Pytorch:

Pytorch	Tensorflow
Imperative, computations on the go	Symbolic, first define and them compile.
Dynamic Computational Graphs	Static Computational Graphs

```
import torch
from torch import nn
define model as classes
    def __init__(self, in):
        super().init()                  Initialization method
                                        with parameters
        sef. log.reg = nn.Sequential(   Def of
            nn.linear (in,1),           arch
            nn.sigmoid()
        )
```