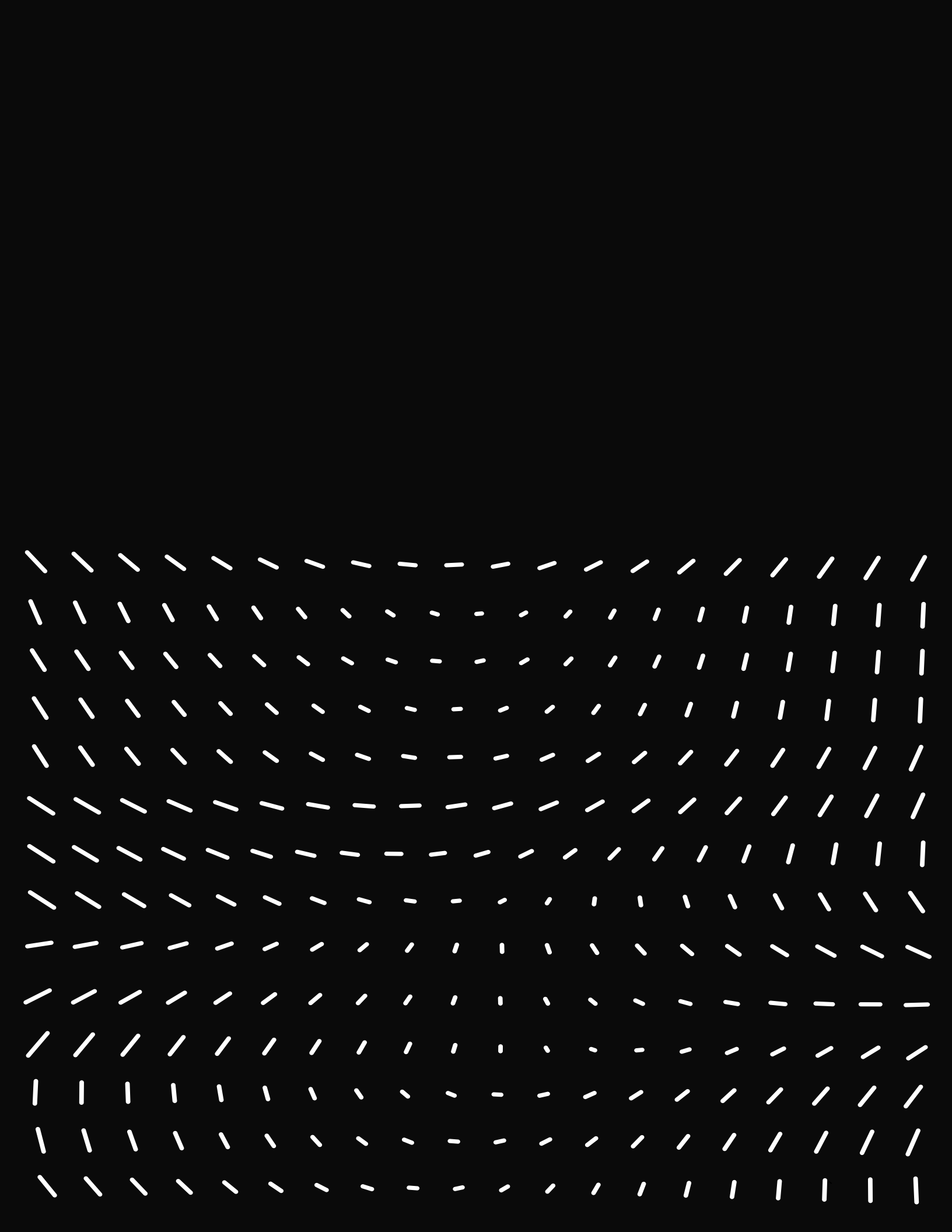
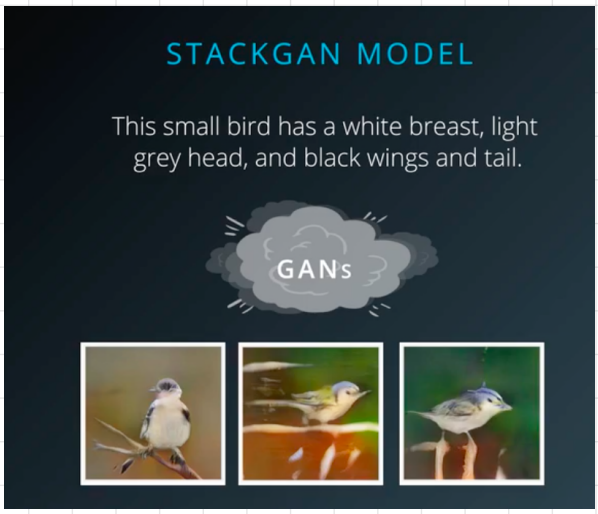


Application:

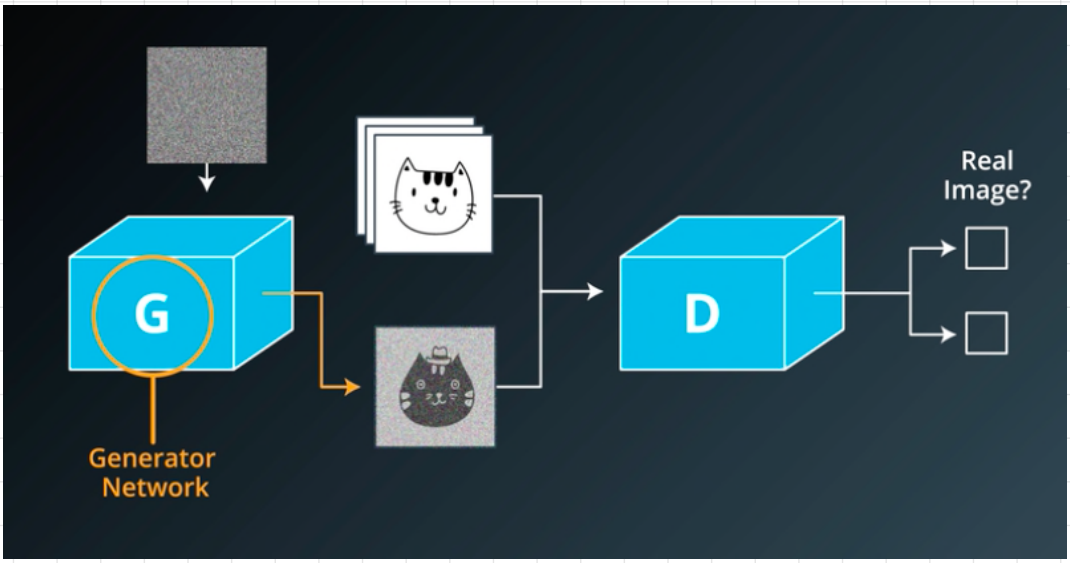
Generate realistic data (Mostly for images)



Generate images based on description of words.

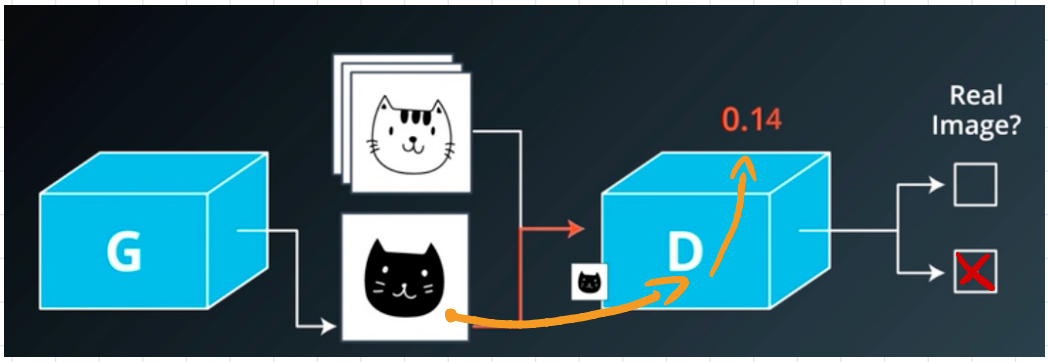
- Pix2Pix
  - Drawings → real images
  - Real images → cartoon characters.
- Simulated Training Set
- Imitation Learning

How GANs works?



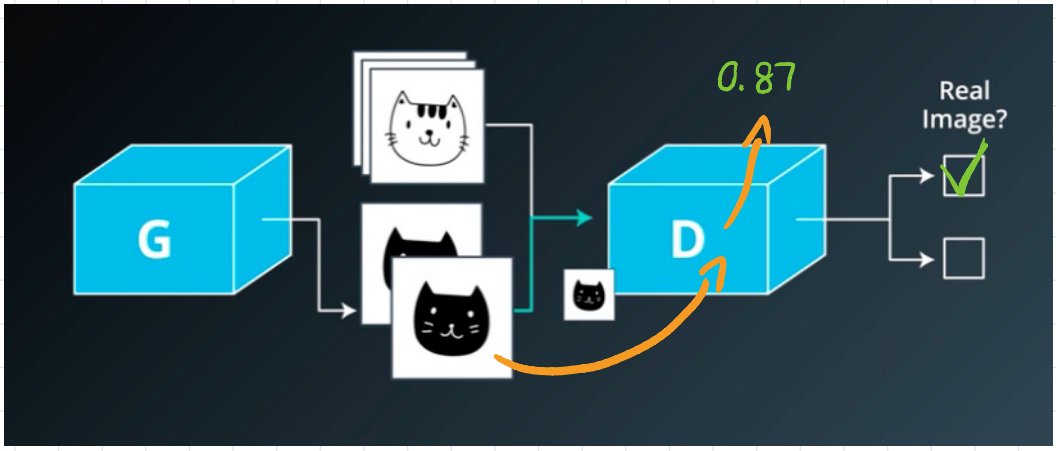
- Generator takes **Random noise** as input, runs it through differentiable functions & transform a noise into recognizable structure.
  - output **a realistic sample**
  - A choice of noise returns diverse samples
  - Goal: want these generated samples to be fair samples from the distribution over real data.
  - Doesn't start out generating real/right samples.
  - Unsupervised: doesn't have targets associated with labels. We show GANs a bunch of samples & ask it to create more samples by drawing from the same probability distribution.
- Discriminator is fed with real imgs from samples & fake ones from generator 50% of the time.

The discriminator tries to:



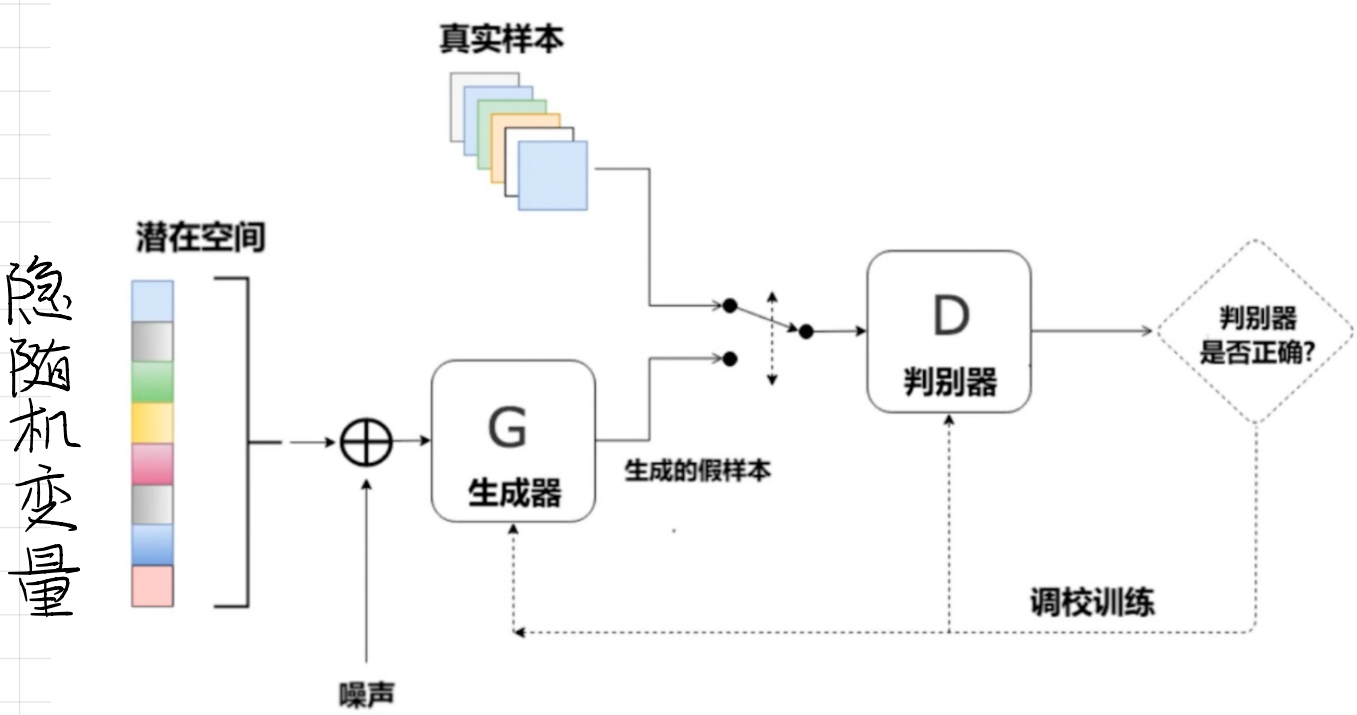
Assign a value near 0 for fake imgs.

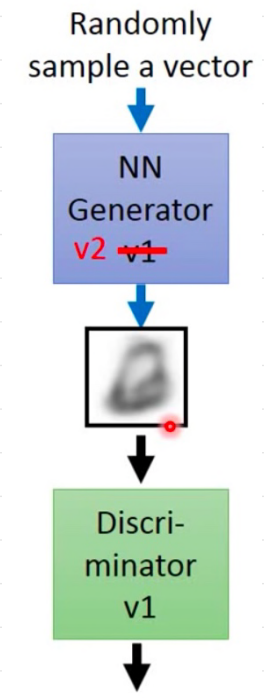
The generator tries to:



generate imgs that will be assigned a value near 1.

Overtime, the generator is forced to produce near perfect samples.





lower number

↓

Update $G(v_1)$ use gradient descent to update parameters of G & fix D

Some Math/Theory

Maximum Likelihood Estimation

- Given data distribution $P_{data}(x)$ → E.g: a joint of vectors of the pixels of an img
- We have a distribution $P_G(x;\theta)$, parameterized by θ

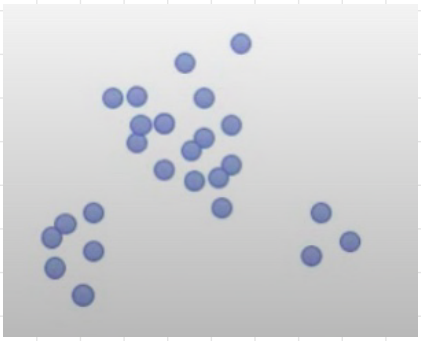
↓ (x & θ are similar)

E.g. A Gaussian Mixture Model, θ are means & variances of the Gaussians.

GOAL: Want to find θ such that $P_G(x;\theta) \approx P_{data}(x)$

1. Sample $\{x^1, x^2, \cdots x^m\}$ from $P_{data}(x)$
2. Compute $P_G(x^i;\theta)$
3. Likelihood of generating the models

Basically, prob of P_G generating a distribution as close to $\{x^1 \cdots x^m\}$ as possible.



$$L = \prod_{i=1}^{m} P_G(x^i;\theta)$$ → Calculate P_G of sample $\{x^1 \cdots x^m\}$

4. Find θ^* to maximize likelihood.

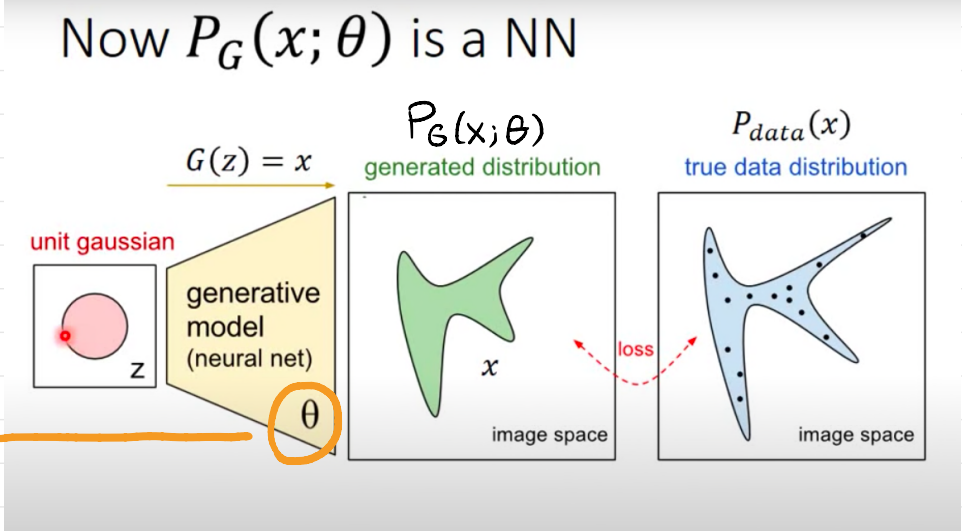
$$\theta^* = arg\max_\theta \prod_{i=1}^{m} P_G(x^i;\theta) = arg\max_\theta log \prod_{i=1}^{m} P_G(x^i;\theta)$$

$$= arg\max_\theta \sum_{i=1}^{m} log P_G(x^i;\theta)$$

Didn't quite understand the rest...

G is determined by θ

Find parameter θ to make G generate a distribution P_G close to $P_{data}(x)$



z = random sampled vector from a distribution

x = G(z), also a distribution.

$$P_G(x) = \int_z P_{prior}(z)\, I_{[G(z)=x]}\, dz$$

Prob of generating x from P_G

Generator G & Discriminator D

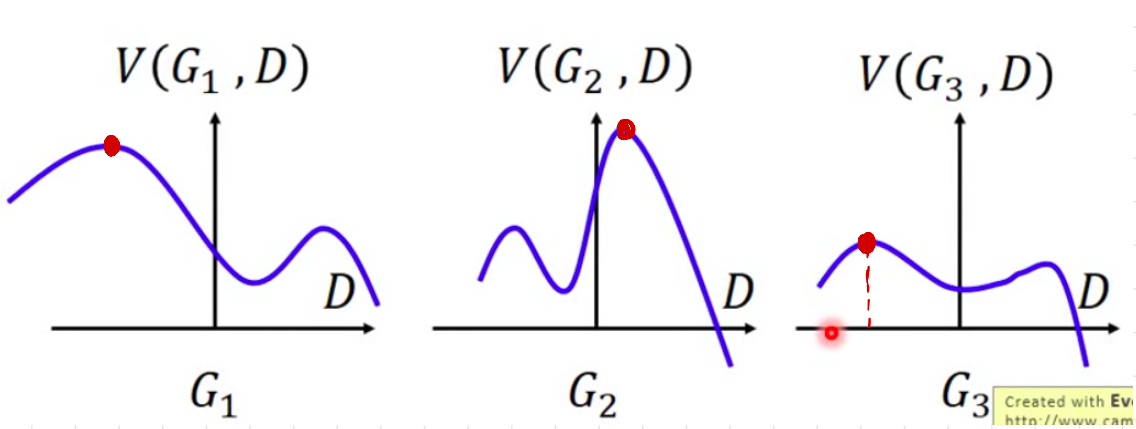
- G input z, output x
- D input x, output scalar
  - Evaluate difference bt. $P_G(x)$ & $P_{data}(x)$
- Define a function V(G, D)

$$G^* = \arg\min_G \max_D V(G,D)$$

G^* ≡ generator that generates samples near $P_{data}(x)$

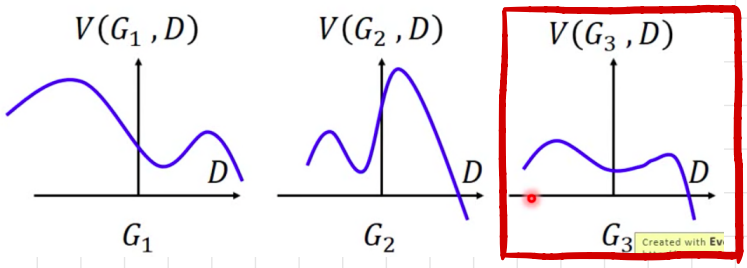
Regarding Equation G^*

Assume that there are 3 discrete choices of G



1. $\max_D$: $G^* = \arg\min_G \max_D V(G, D)$
   - Fix G, vary D
   - Find a point that maximizes $V(G, D)$ (aka the red pts ↑)

2. $\min_G$ →



$$V = E_{x \sim P_{data}}[\log D(x)] + E_{x \sim P_G}[\log(1 - D(x))]$$

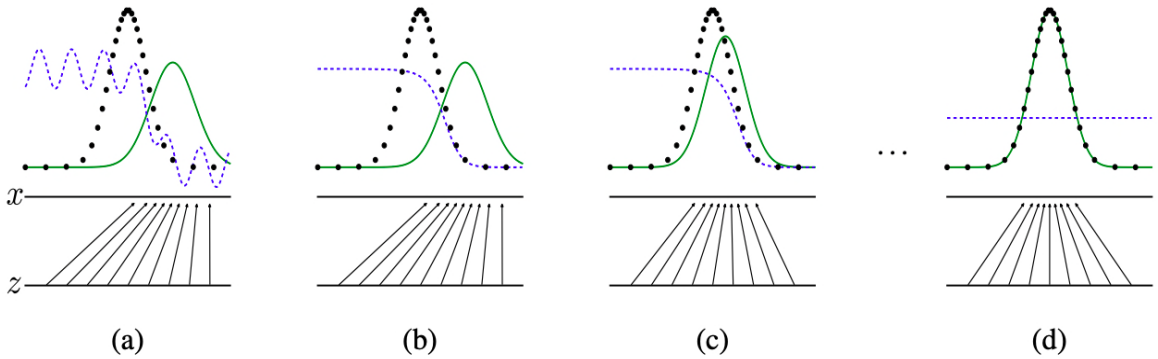
Given a generator G, $\max_D V(G, D)$ evaluate the "difference" between P_G and P_{data}

↓

We want G_3 because $V(G, D)$ is the smallest

Therefore, difference bt. P_G & P_{data} is the smallest.

Find a G that let D on the curve $V(G, D)$ be minimal



(a) (b) (c) (d)

D unable to tell the diff. bt P_{data} & P_G, $D(x) = \frac{1}{2}$

Games & Equilibria

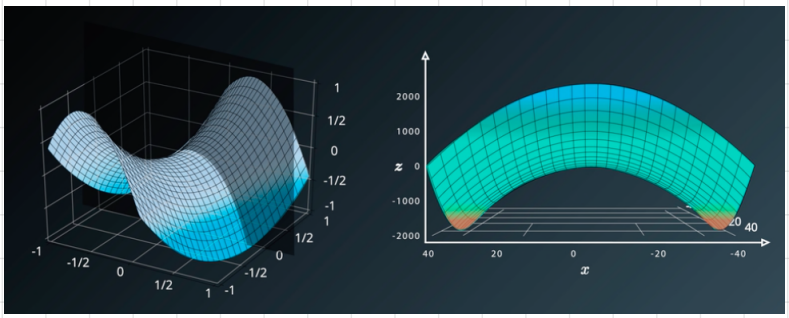
Game Theory:

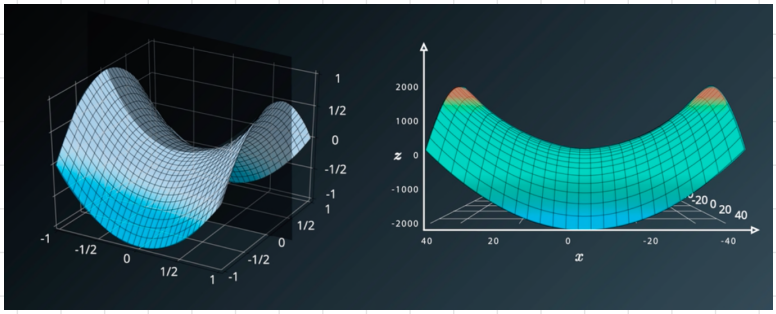
- **Equilibrium:** neither players can improve their payoff by changing their own strategy without changing the other player's strategy.
- Both sides choose their own actions uniformly at random

$$Cost_D = -Cost_G$$

Generator wants to minimize the value function, discriminator wants to maximize it.

∴ Equilibrium ≡ where value is max for D & min for G



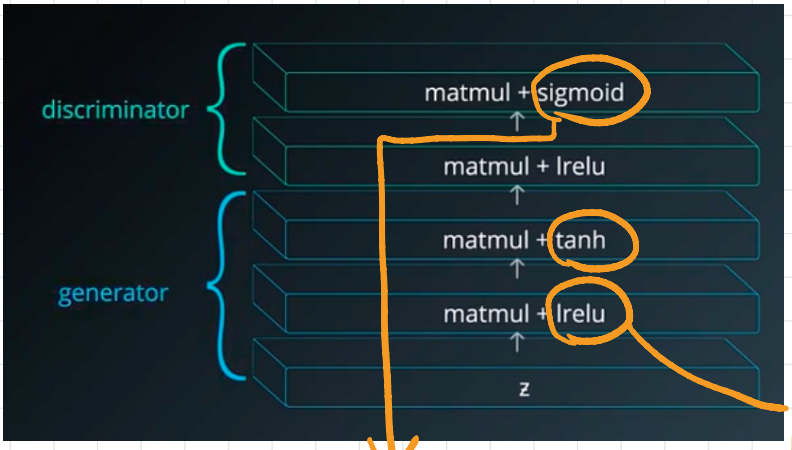


Run 2 optim. algorithms, each minimizing one player's cost with respect to that player's parameters.

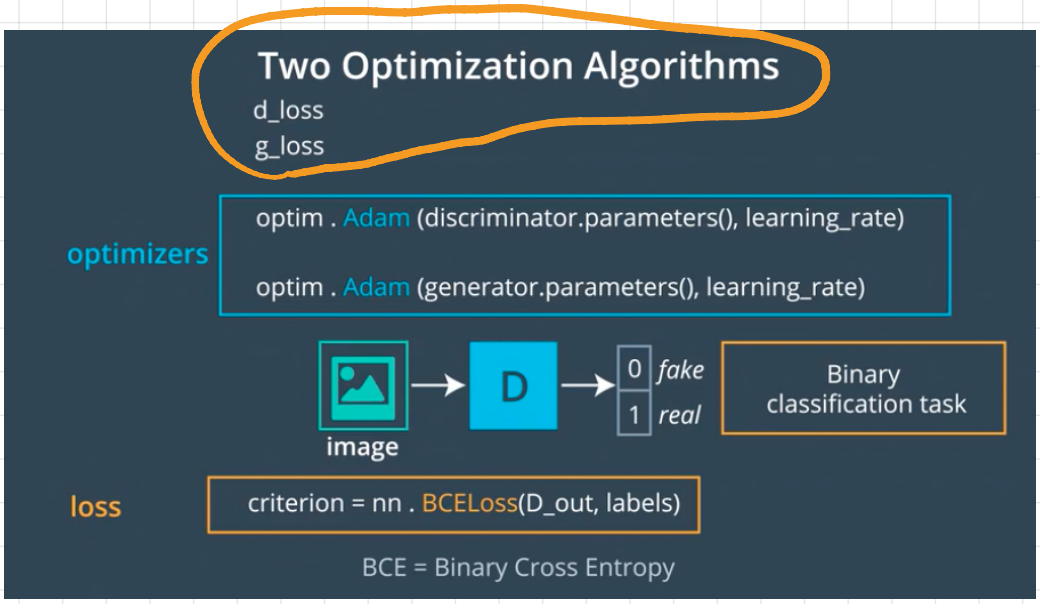
Tips for Training GANs

Fully Connected GANs

- Have at least 1 hidden layer for both D and G.
- leaky ReLU for hidden units activation function: ensure gradients to flow through the architecture
- Because the only way generator will learn is to receive a gradient from the discriminator

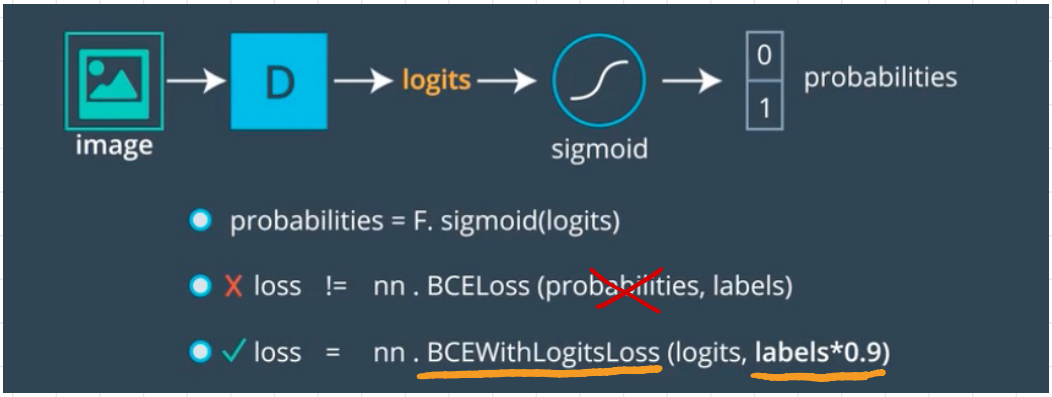


Need to output a prob.

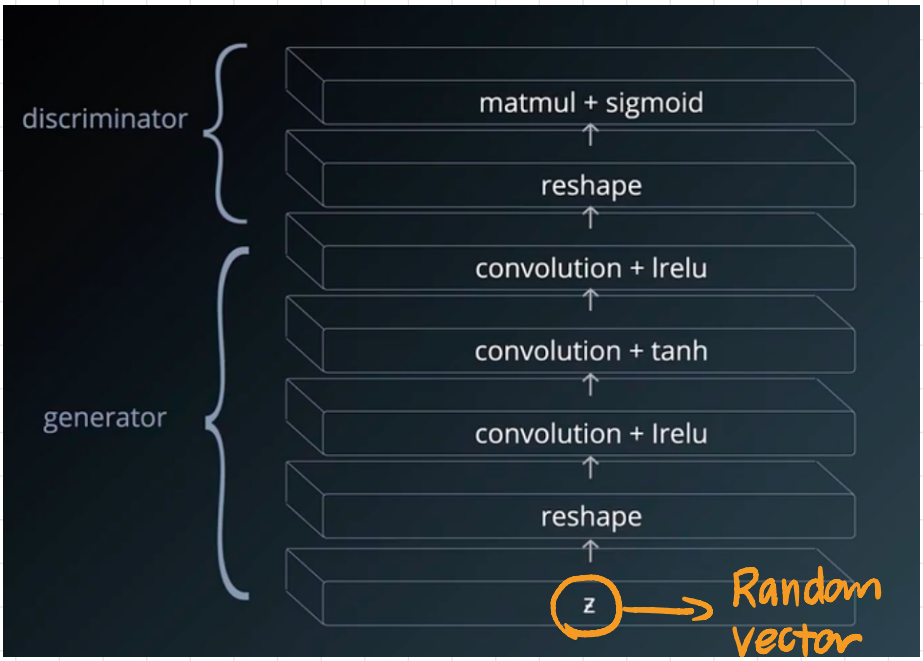


Discriminator Loss

Need to use numerically stable cross entropy loss [logits]

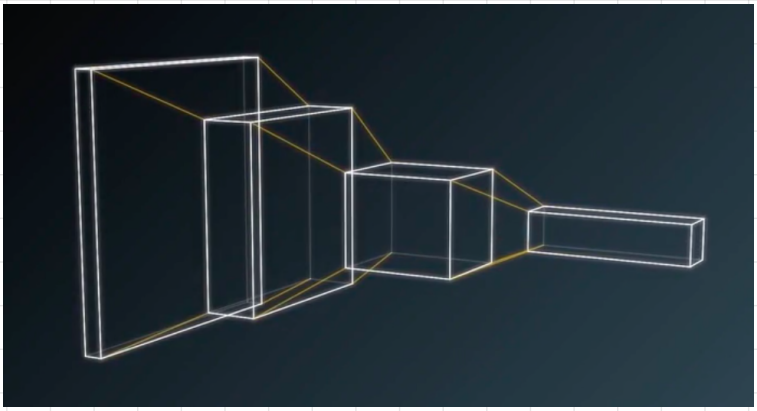


Convolutional GANs



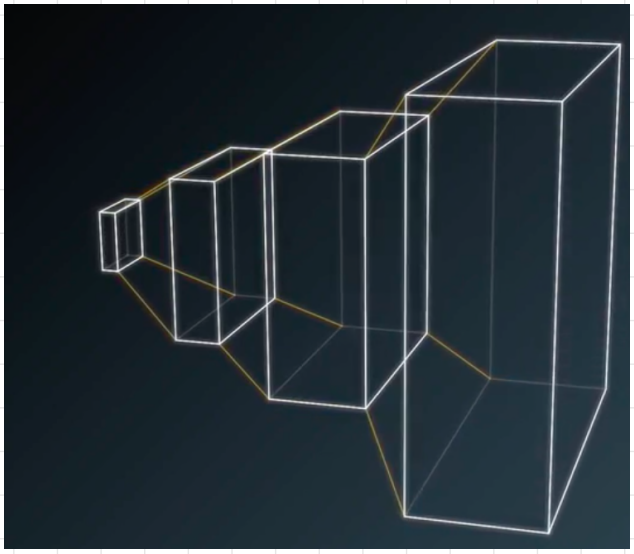
- A minibatch of z forms a matrix.
- A convolution expects a 4D tensor for input:
  - 1 axis for different examples in minibatch
  - 1 axis for diff. feature maps
  - 2 axis for width & height

Opposite to a classifier conv net:



Height ↓
width ↓
depth ↑

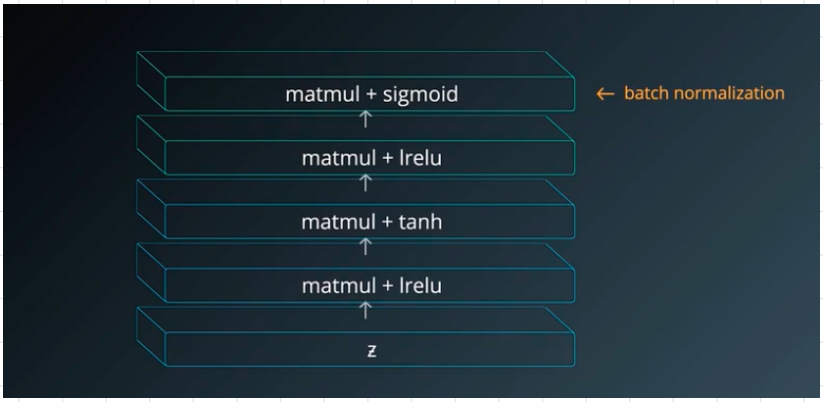
When using conv net as a generator net:



Do the opposite!

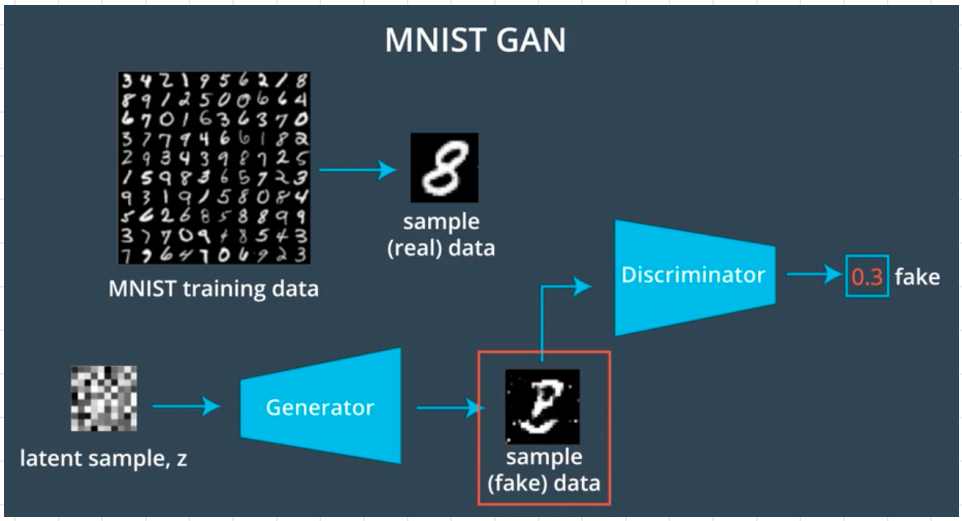
DCGANs.

Batch Norm



Batch norm on all layers but input & final classifier

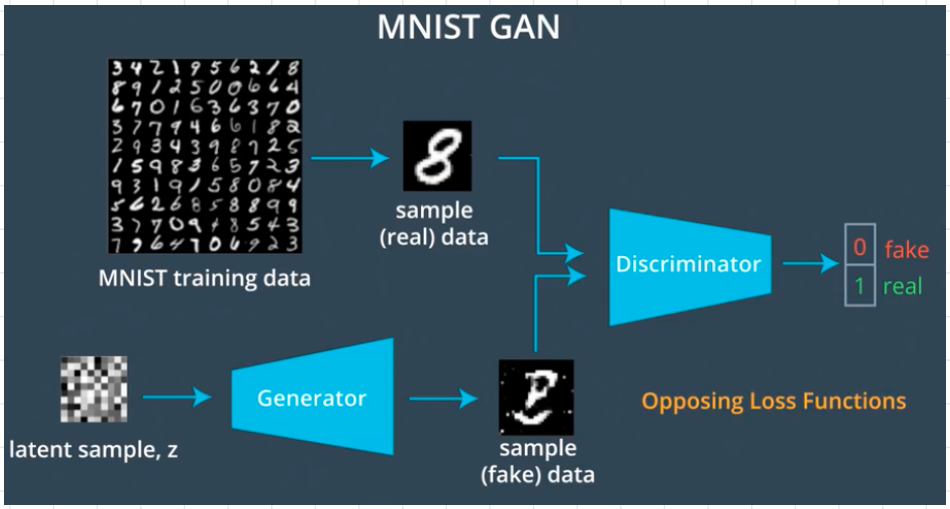
A Final Overview



Generator is learning & getting better at creating better samples & fool D to mis-labeled fake data as true.

Goal:

Maximize the error made by D.



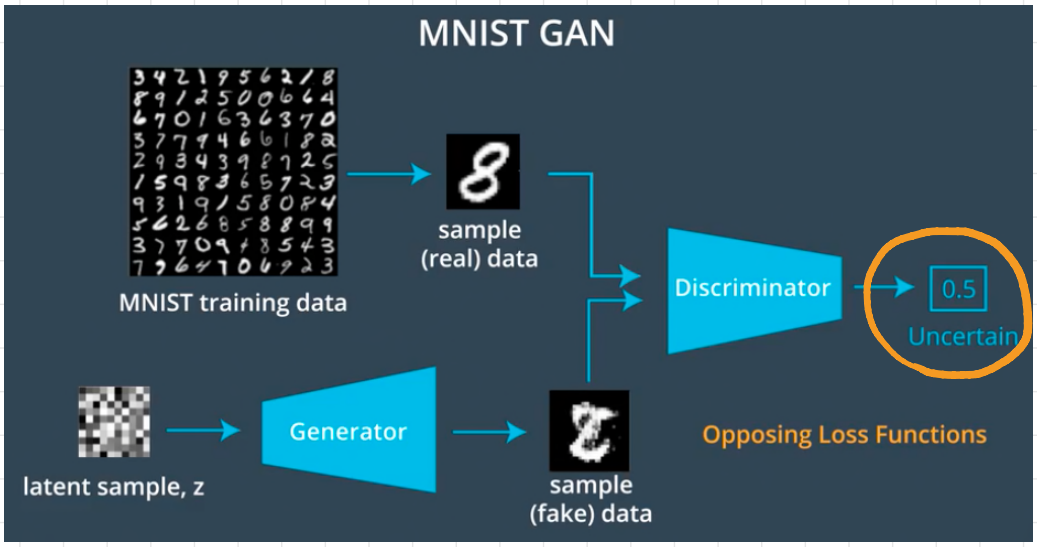
D is also training, by being fed w/ alternating real & fake imgs, getting better at identifying.

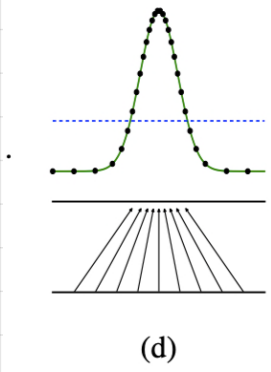
Goal:

Minimize error rate.

Recall equlibrium & Game Theory:

Two players at equilibrium, no move tradeoff





$D(x) = 1/2$