Generative Adversarial Networks (GANS).

- Basics of GAN
- Training
- Cost Function Derivative
- Drawbacks
- Implementation in Tensorflow

Generative Adversarial Network (GAN)
– Jun 10, 2014 Ian Goodfellow

Basics of GAN

- Adversarial → conflict / opposition
- 2 neural nets competing against each other. (Discriminator, Generator)

Objective: GANs are neural networks that are trained in an adversarial manner to generate data mimicking some distribution.

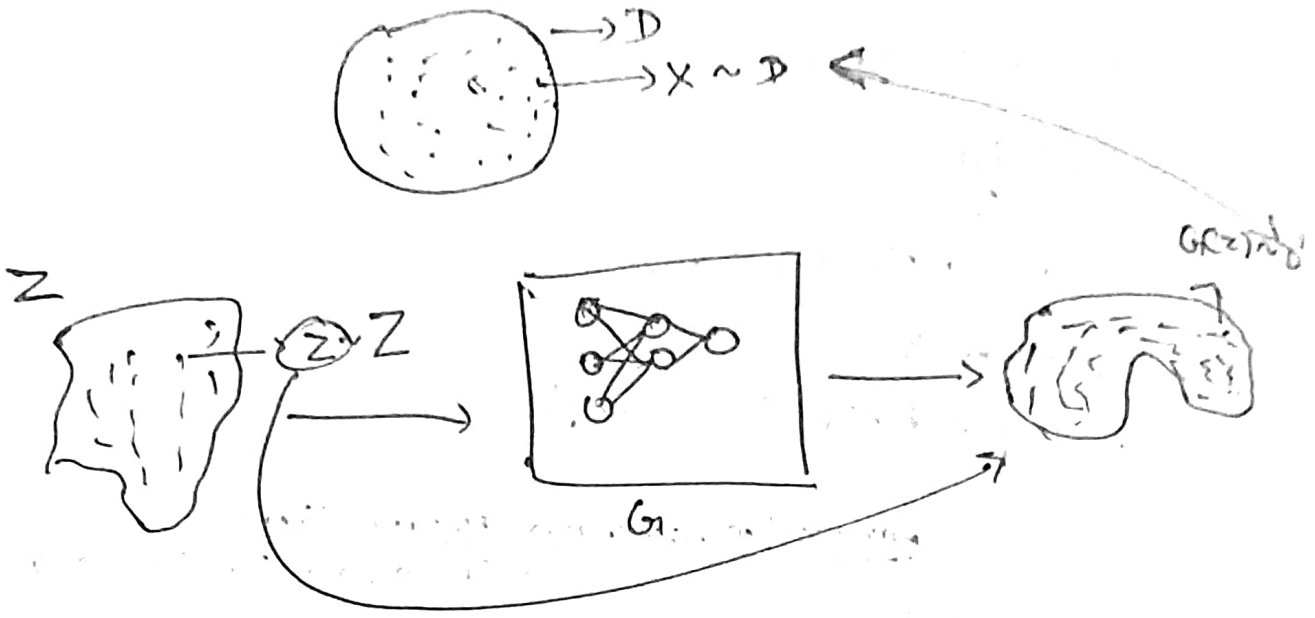
Discriminative Model

- Discriminates between 2 different classes of data (fake / not fake clf).

Generative Model

- A generative model G is trained on a sample data x sampled from true distribution D is the one which given a random distribution z produces D' which is close to D in some metric.

L_1, L_2 norm.



? what is it really doing.

- Take a random distribution z' from Z (true dist)

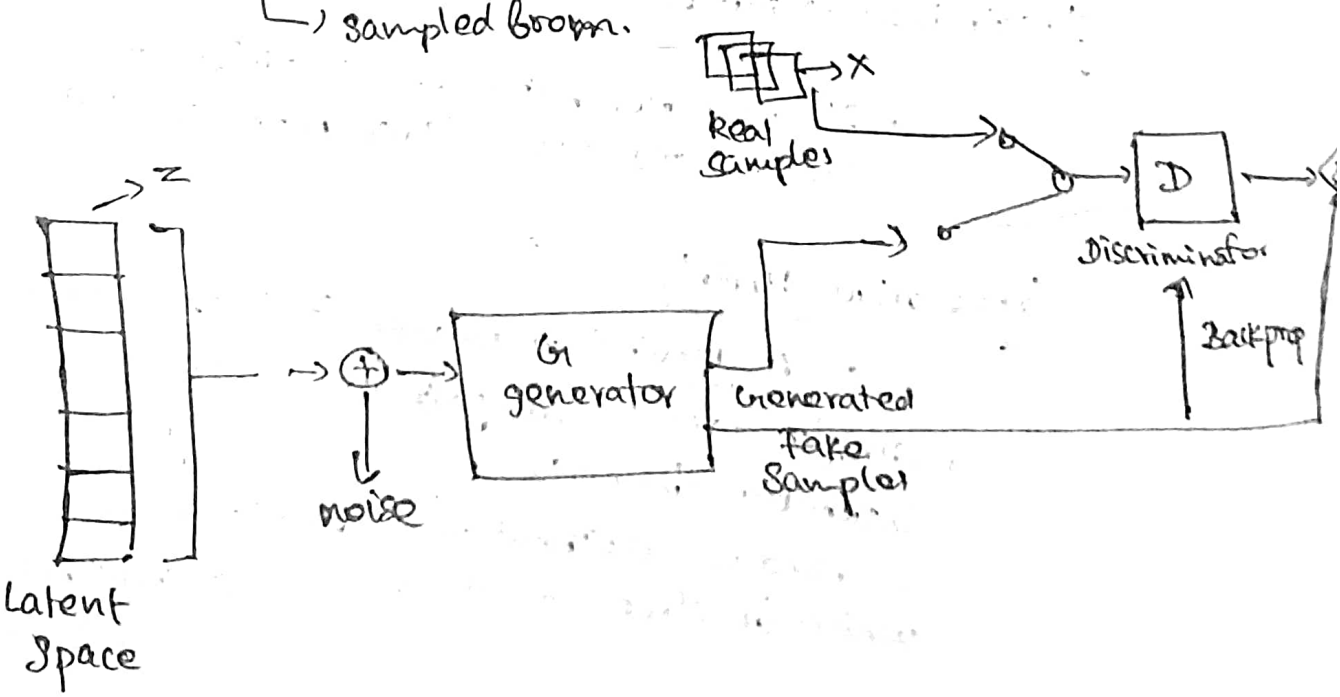
  and try converting z' to D' which mimicks D (X's distribution)

- now we have generated similar but not same data resembling X's distribution D.

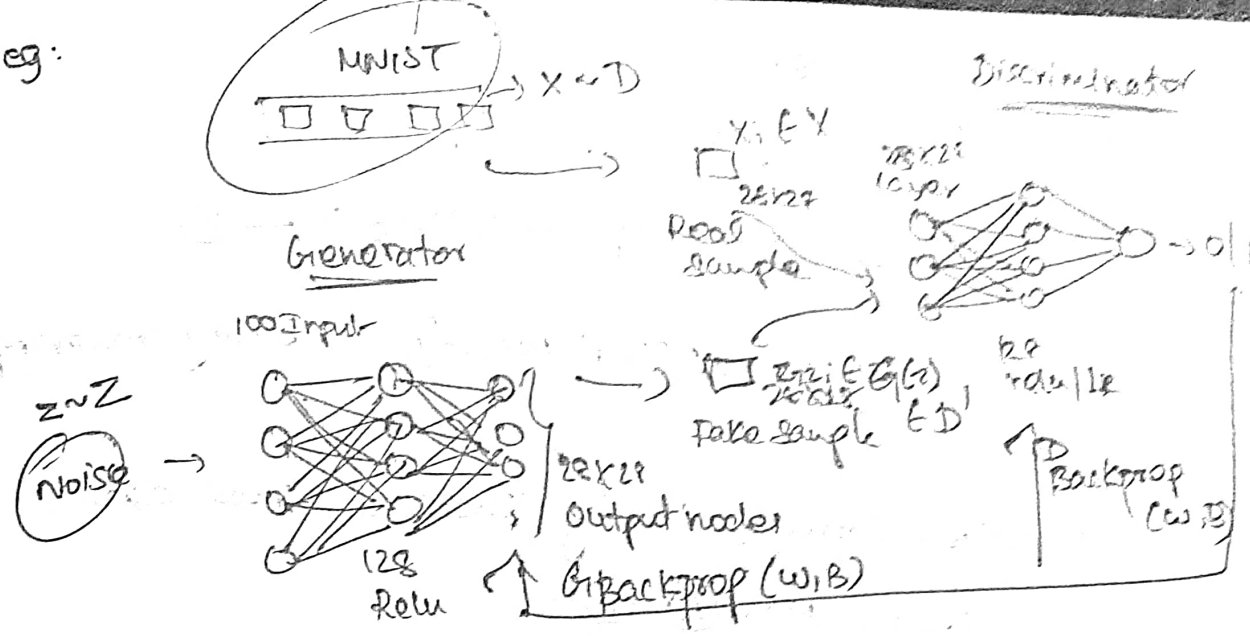
  G(z) ⟷ D'

$z \sim Z$ maps $G(z) \sim D'$ $[D' \approx D]$.

↳ sampled from.



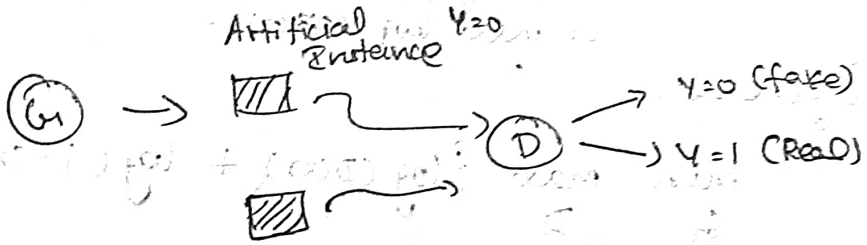
eg:



Generative Adversarial network

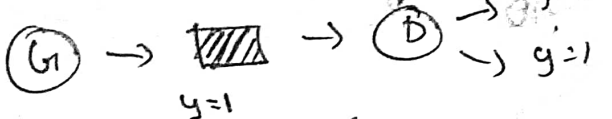
Learning mechanism (w, B)

Training the discriminator - $D_{Backprop}$



- (D) is a simple classifier only. The weights are updated like a classifier neural network.

Training the Generator



- Artificial Instance is explicitly named y=1 for the (D) to think its real.
- Then backprop is done to generator based on discriminator output.
- Convergence condition: discriminator outputs 0.5

Logs

Discriminator

$L(D(x), 1) = \log(D(x))$ — (A)

$L(D(G(z)), 0) = -\log(1 - D(G(z)))$ — (B)

$L = y \log \hat{y} + (1-y) \log(1-\hat{y})$ — Binary cross entropy

Objective: correctly classify generated, real data

i.e: max (A), (B)

$$\max \{\log D(x) + \log(1 - G(z))\} \quad \text{— (D)}$$

Generator

$$\min \{\log(D(G(z))) + \log(1 - D(G(z)))\} \quad \text{— (G)}$$

$\Rightarrow D(G(z)) = 1$ ((G) → give real samples)

or max $\log(D(G(z)))$.

In general,

$$\min_G \max_D \{\log(D(x)) + \log(1 - D(G(z)))\}$$

Gradients

$$\nabla G_D = \frac{\partial}{\partial \theta}\left(\log D(x) + \log(1 - D(G(z)))\right)$$

$$\nabla G_G = \frac{\partial}{\partial \theta} \log(1 - D(G(z)))$$

compute gradients based θ which are parameters of respective neural net architectures.

Limitations

- Vanishing Gradients.
  - use wasserstein loss
- Mode Collapse.
  - Generator is set to fool the discriminator and repeatedly produces same outputs.
- Failure of convergence
  - add extra noise to disc. ips.
  - penalizing disc. wts.