

Generative Adversarial Network (GAN)

Outline

- Introduction
- Sample the model distribtuino
- Training
- Cost function derivation
- Drawbacks of GAN
- Implementation in PyTorch

Introduction

- **Generative Adversarial** Network are deep neural network architecture comprised of two neural networks, computing one against the other
- Gan are neural networks that are trained in an adversarial manner to generate data mimicking some distribution

Machine Learning Models

- Discriminative model (classification problems): it is the one that discriminate between two different classes of data.
- Generative model: A generative model G to be trained on training data X sampled from some true distribution D is the one which, given some standard random distribution Z produces a distribution D' which is close to D according to some closeness metric.

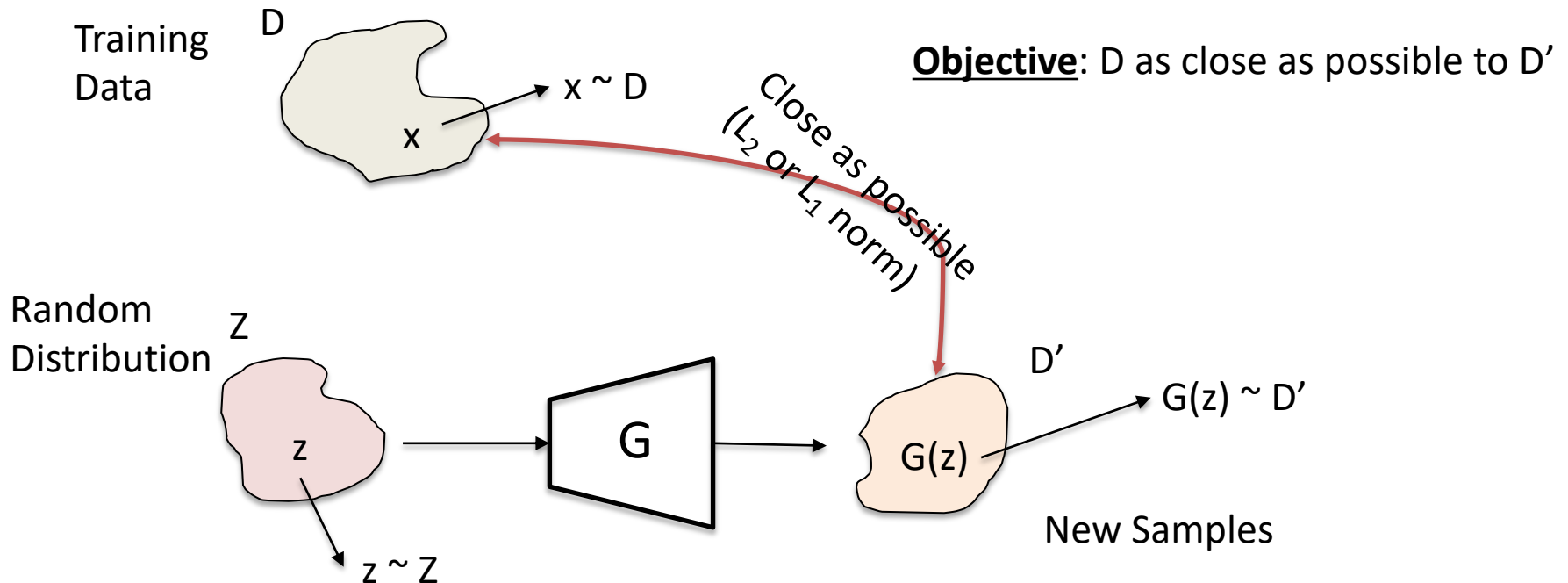
Generative Adversarial Networks(GAN)

Discriminator: A neural network that distinguishes between output data point (fake) from Generator and training data sample (real).

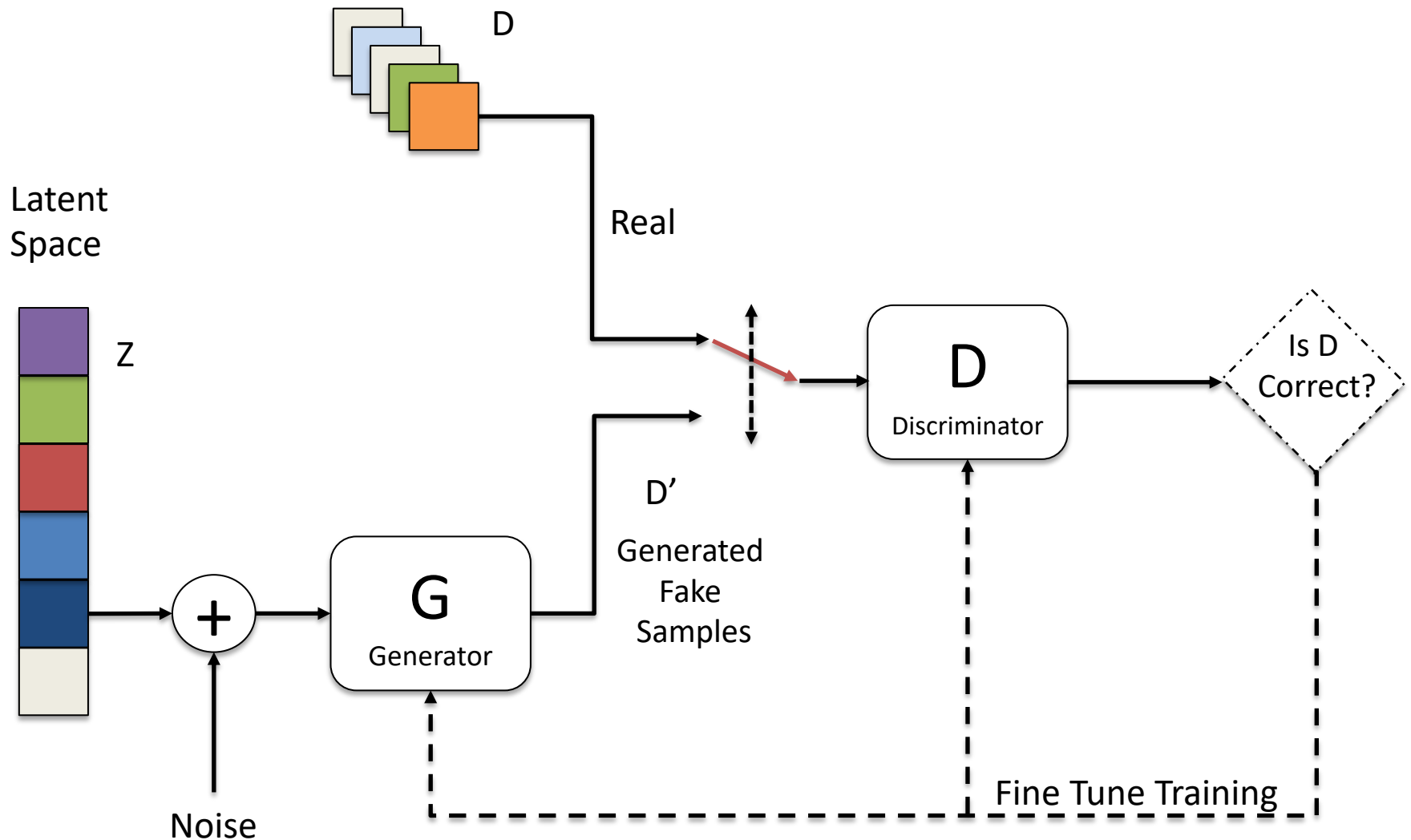
Generator: A neural network that takes as input random noise and transforms it into a sample from the model distribution.

Generative model

A generative model G to be trained on training data X sampled from some true distribution D is the one which, given some standard random distribution Z produces a distribution D' which is close to D (original samples) according to some closeness metric.



Generative Adversarial Network




MiniMax

Zero sum game function:

Cost function of Generator (J_G) = - Cost function of Discriminator (J_D)

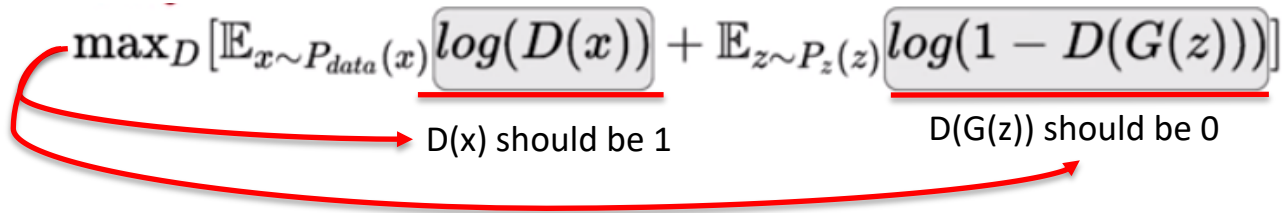
$$\min_G \max_D V(\theta^{(D)}, \theta^{(G)}) = \mathbb{E}_{x \sim P_{data}} \log D(x) + \mathbb{E}_{z \sim P(z)} \log(1 - D(G(z)))$$


Learnable parameters

This cost function: Expectation of log D of X plus the expectation over Z log of 1 minus D.

Training Discriminator Network

$$\max_D [\mathbb{E}_{x \sim P_{data}(x)} \log(D(x)) + \mathbb{E}_{z \sim P_z(z)} \log(1 - D(G(z)))]$$



$D(x)$ should be 1 $D(G(z))$ should be 0

Training Generator Network

$$\min_G [\mathbb{E}_{x \sim P_{data}} (\log D(x)) + \mathbb{E}_{z \sim P(z)} (\log(1 - D(G(z))))]$$

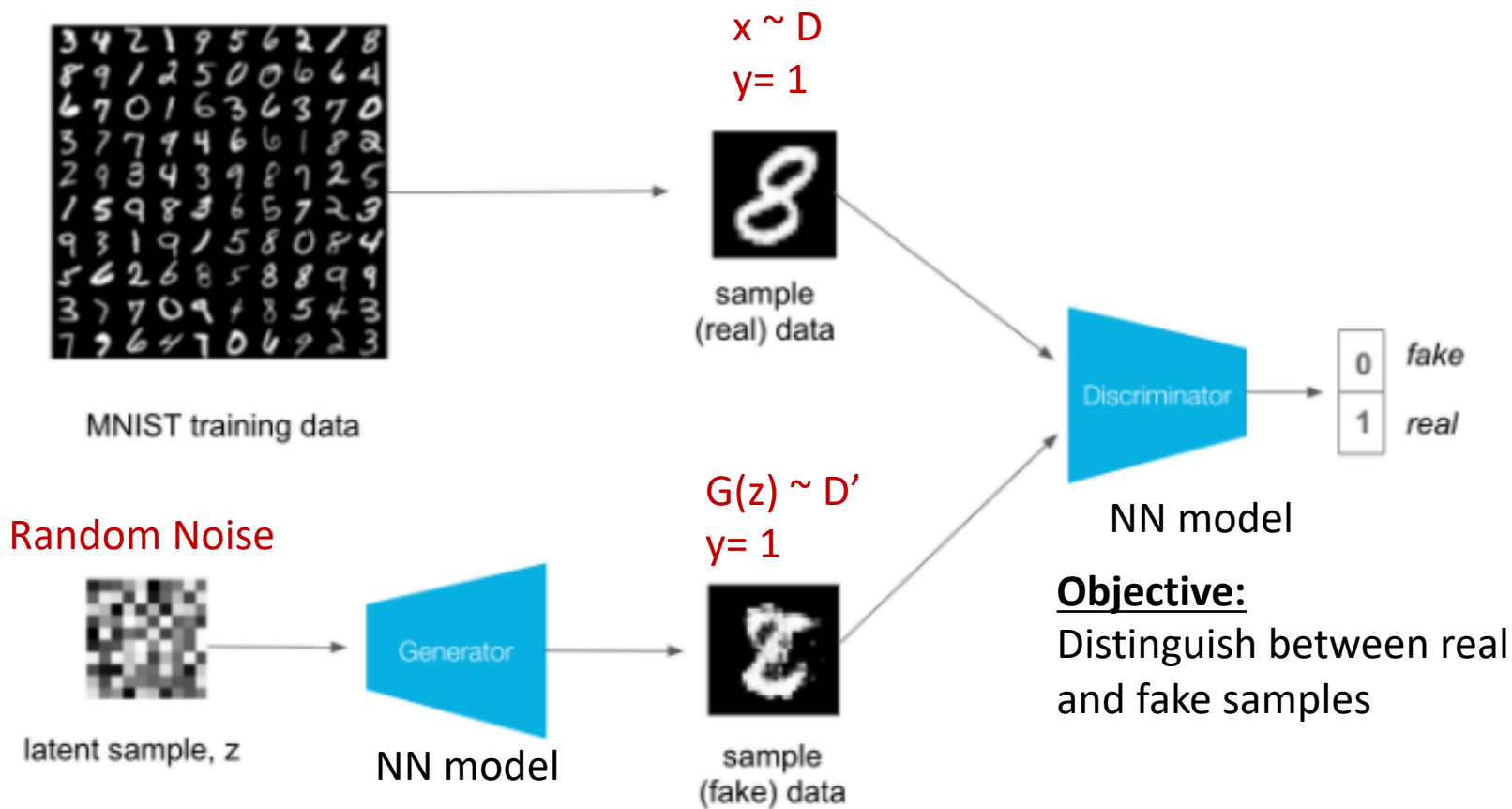
Constant with respect to G $D(G(z))$ should be Maximized

$$\max_G [\mathbb{E}_{z \sim P_z(z)} \log(D(G(z)))]$$

$D(G(z))$ should be 1

GAN based on MNIST

form a distribution D



Objective:

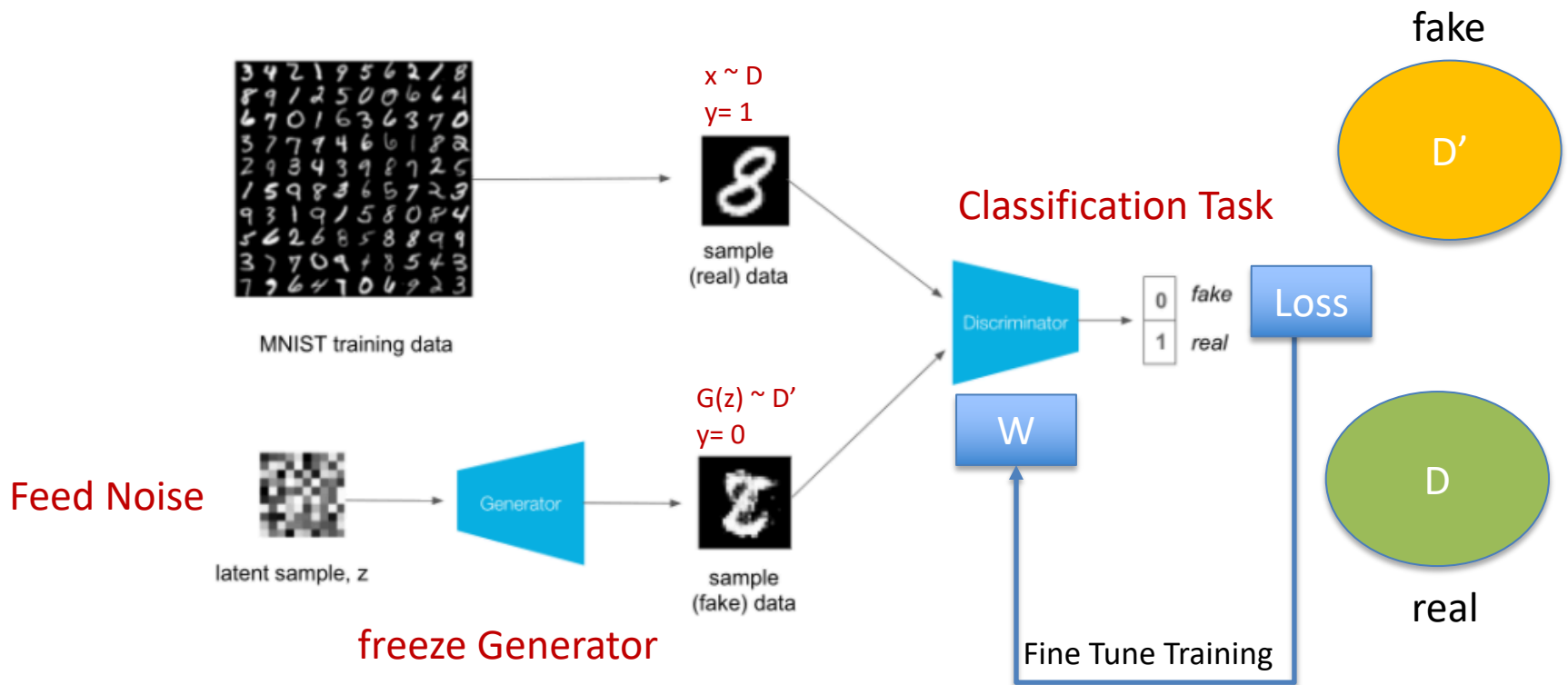
Distinguish between real and fake samples

Objective:

To produce fake samples that are as close as possible to real samples

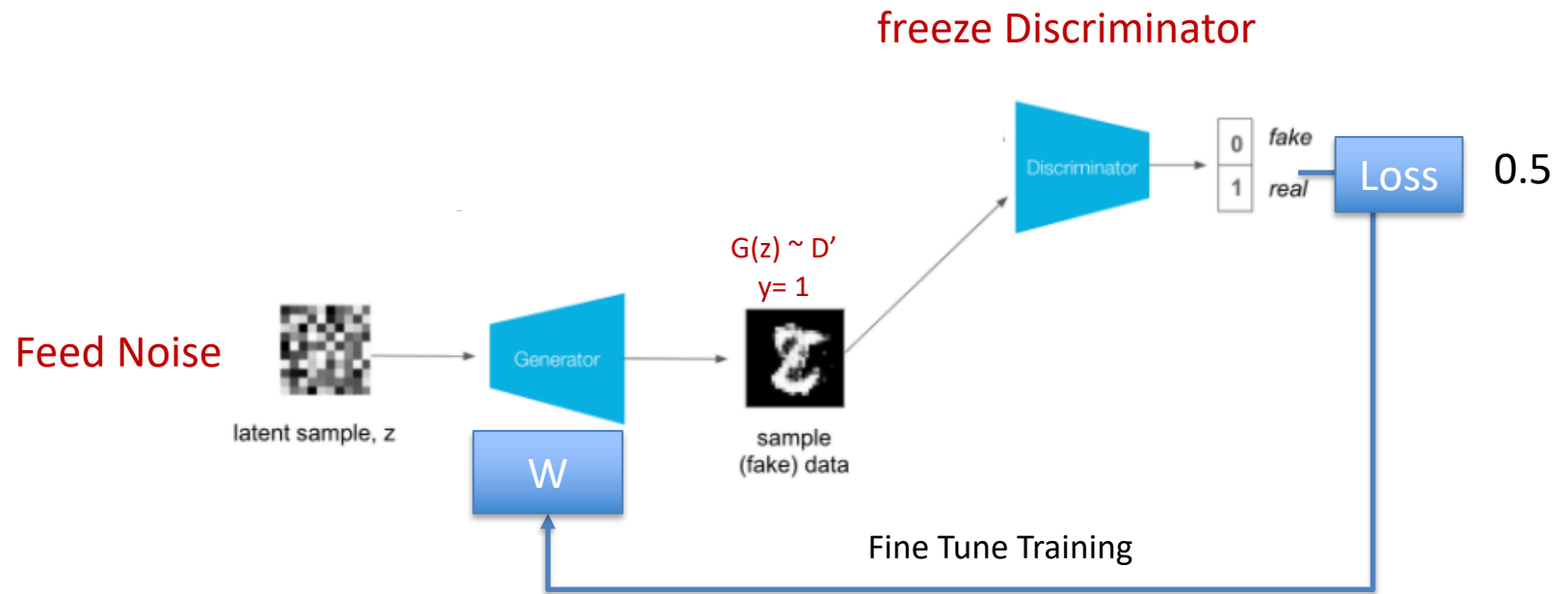
GAN Training

Step 1) Train the Discriminator network while freeze Generator.



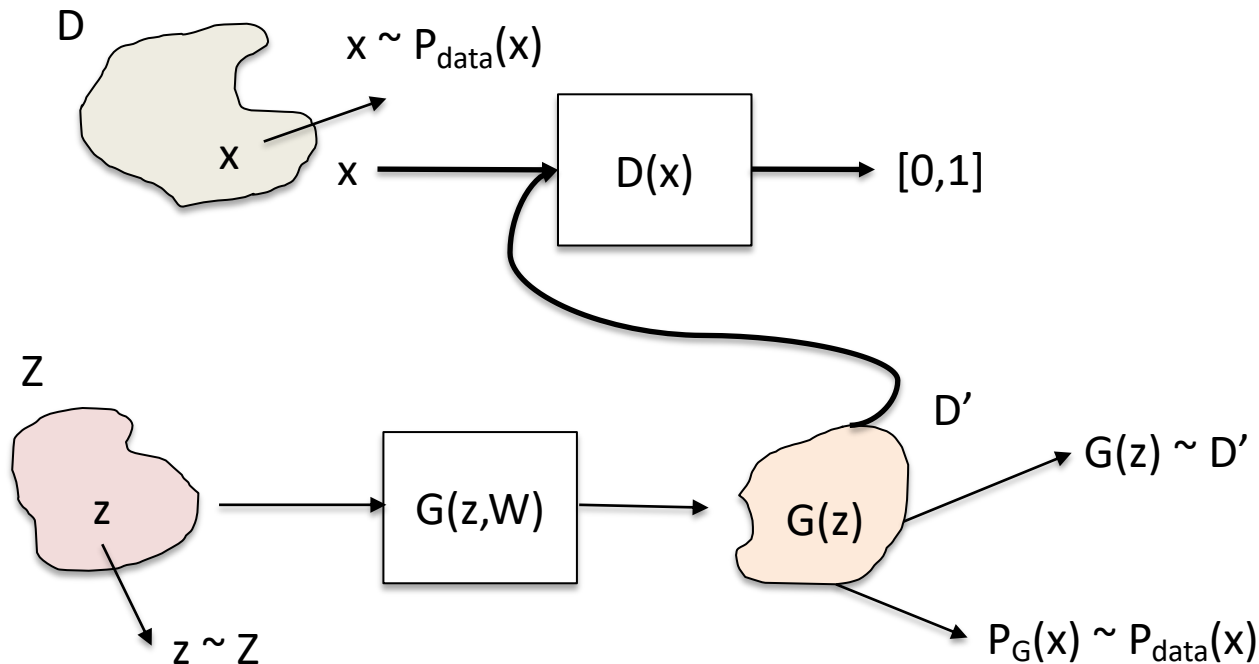
GAN Training

Step 2) Train the Generator network while freeze Discriminator.



Loss Function of GAN

- Discriminator role is to distinguish between actual data and fake data
- Generator role to create data is such as way that is can fool the discriminator.



Loss Function (Binary Cross-Entropy)

Discriminator

$$L(y', y) = [y \log y' + (1-y) \log(1-y')]$$

The label for the data coming from $P_{\text{data}}(x)$ is $y=1$

so $L(D(x), 1) = \log(D(x))$

and

for data coming from generator the label is $y=0$ so

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

$\text{Max} (\log(D(x)) + \log(1-DG(z)))$

Loss Function (Binary Cross-Entropy)

Generator

$$L(y', y) = [y \log y' + (1-y) \log(1-y')]$$

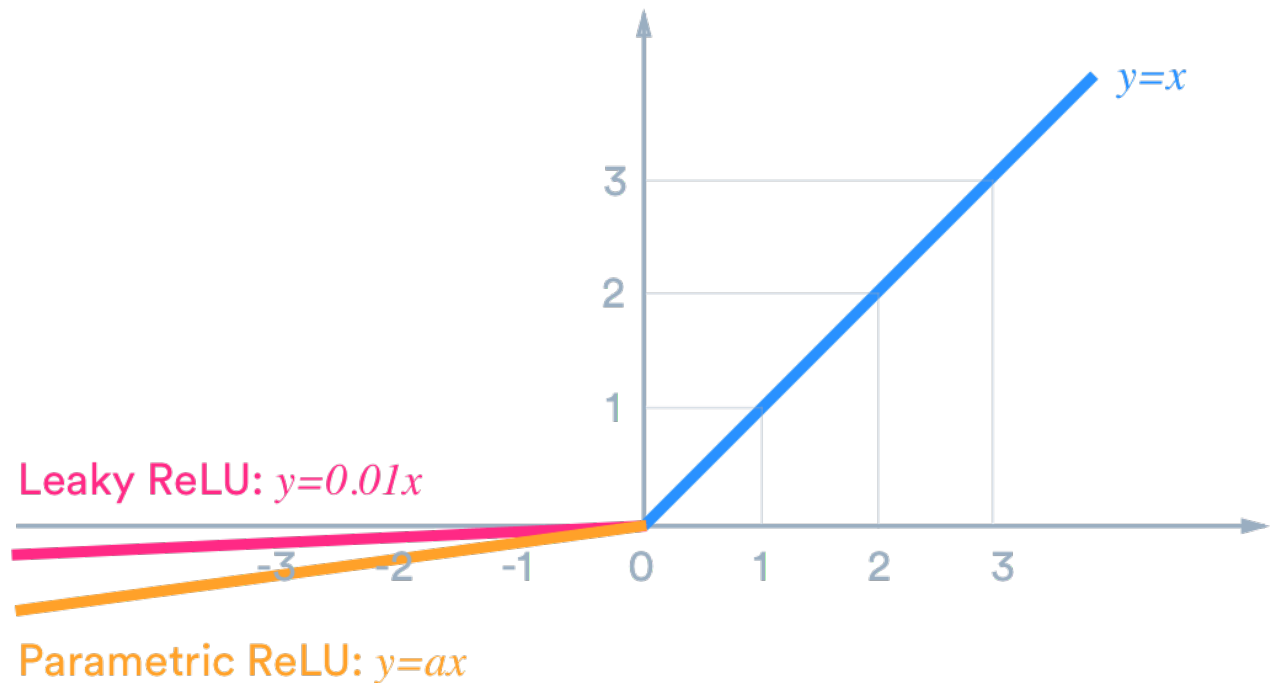
For data coming from generator the label is $y=1$
so $L(D(G(z)), 1) = \text{Log}(D(G(z)))$ to max this we can

$$\text{Min Log } (1-D(G(z)))$$

$$\text{Min}_G \text{Max}_D V(D, G) = \{E[\log D(x)] + E[\log (1-D(G(z)))]\}$$

Leaky Relu

$$f(x) = \begin{cases} x & \text{if } x > 0, \\ 0.01x & \text{otherwise.} \end{cases}$$



GAN based on MNIST

form a distribution D

