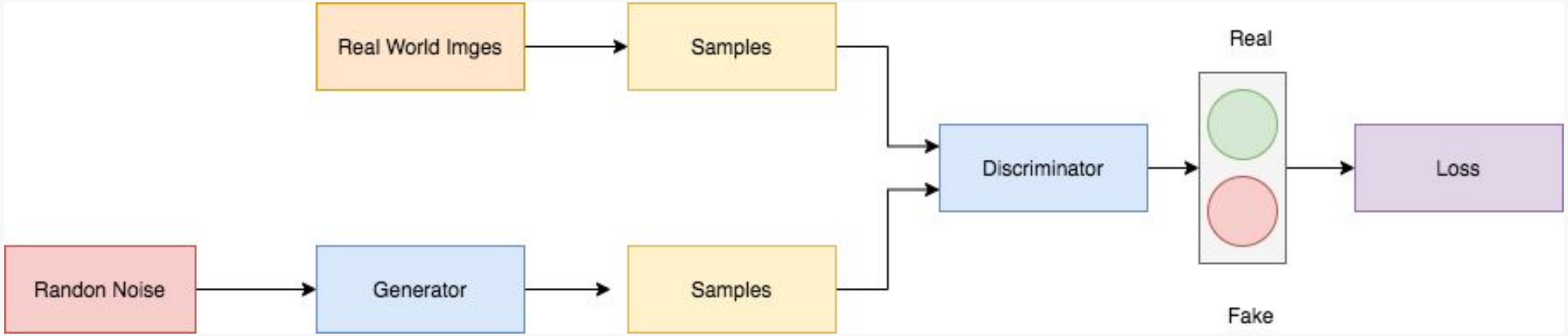
Generative Adversarial Networks and Image Generation

Farid Jamshid, Noah Robertson

What is a GAN?

- Game-Theoretic framework
- generator tries to produce data while a discriminator tries to differentiate the generated data from real data
- The overall goal of the modeling is for the generator to learn to create data indistinguishable from real data



Math

Generator

$$G : \mathcal{Z} \to \mathcal{X}$$

Discriminator

$$D : \mathcal{X} \to [0, 1].$$

Objective

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))].$$

Math

Non-Saturating Generator Loss

$$\mathcal{L}_G = -\mathbb{E}_z[\log D(G(z))]$$

Discriminator Loss

$$\mathcal{L}_D = -\mathbb{E}_x[\log D(x)] \; - \; \mathbb{E}_z[\log(1 - D(G(z)))]$$

Why Non-Saturing Generator Loss?

Originally supposed to minimize

$$\mathbb{E}_{z\sim p_z}\left[\log(1-D(G(z)))\right]$$

But initial generation leads to near 0 score

$$D(G(z)) \approx 0.$$

Leads to non-existent gradient

$$\log(1-D(G(z))) \approx \log(1-0) = \log(1) = 0.$$

So instead of minimizing

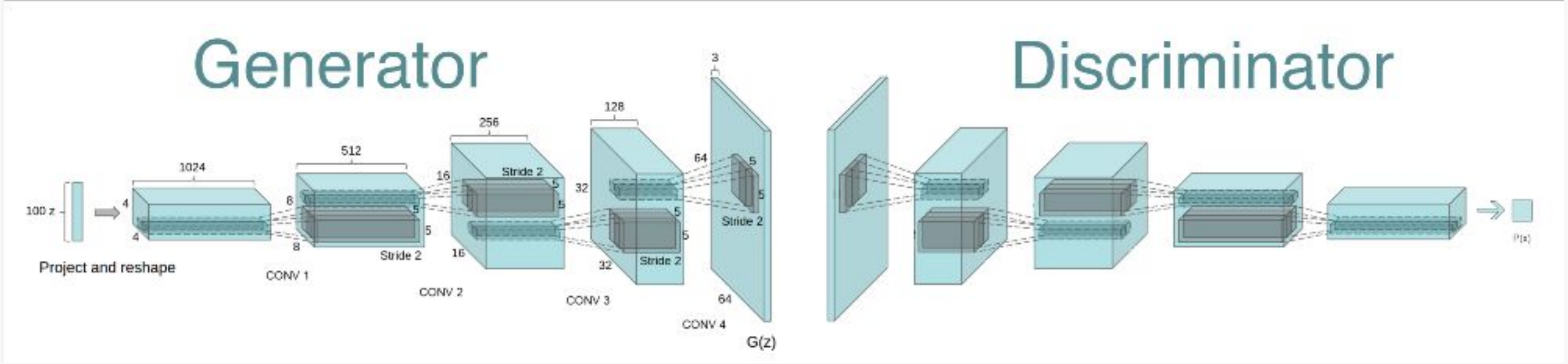
$$\log\big(1-D(G(z))\big),$$

We maximize

$$\log\big(D(G(z))\big).$$

What is a DCGAN?

- Deep Convolutional GANs
- Discriminators work like regular deep convolutional neural networks
- Generators work in reverse – use feature maps to create images from noise



ConvTranspose2d

- Function parameters:
  (in_channels, out_channels, kernel_size, stride, padding)
- ConvTranspose2d uses stride + kernel based learning to learn patterns such as edges, textures and shapes
- Other than the first layer ConvTranspose2d follows a 4,2,1 pattern for kernel, stride, padding to double the spatial resolution at each layer: 2x2 - 4x4 - 8x8 - 16x16 ect

Example of geometry:

2x2: $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ becomes 4x4: $\begin{bmatrix} a & 0 & b & 0 \\ 0 & 0 & 0 & 0 \\ c & 0 & d & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

BatchNorm2d

- BatchNorm2d normalizes activations ensuring stabilization. It calculates the mean and variance then uses the following formula to calculate the norm:

$$\hat{x} = \frac{x-\mu}{\sqrt{\sigma^2+\epsilon}} \quad \text{then is used for} \quad y = \gamma\hat{x} + \beta$$

- GAN is famously unstable, BatchNorm2d helps to solve this problem by keeping activations well scaled and stabilizing gradients