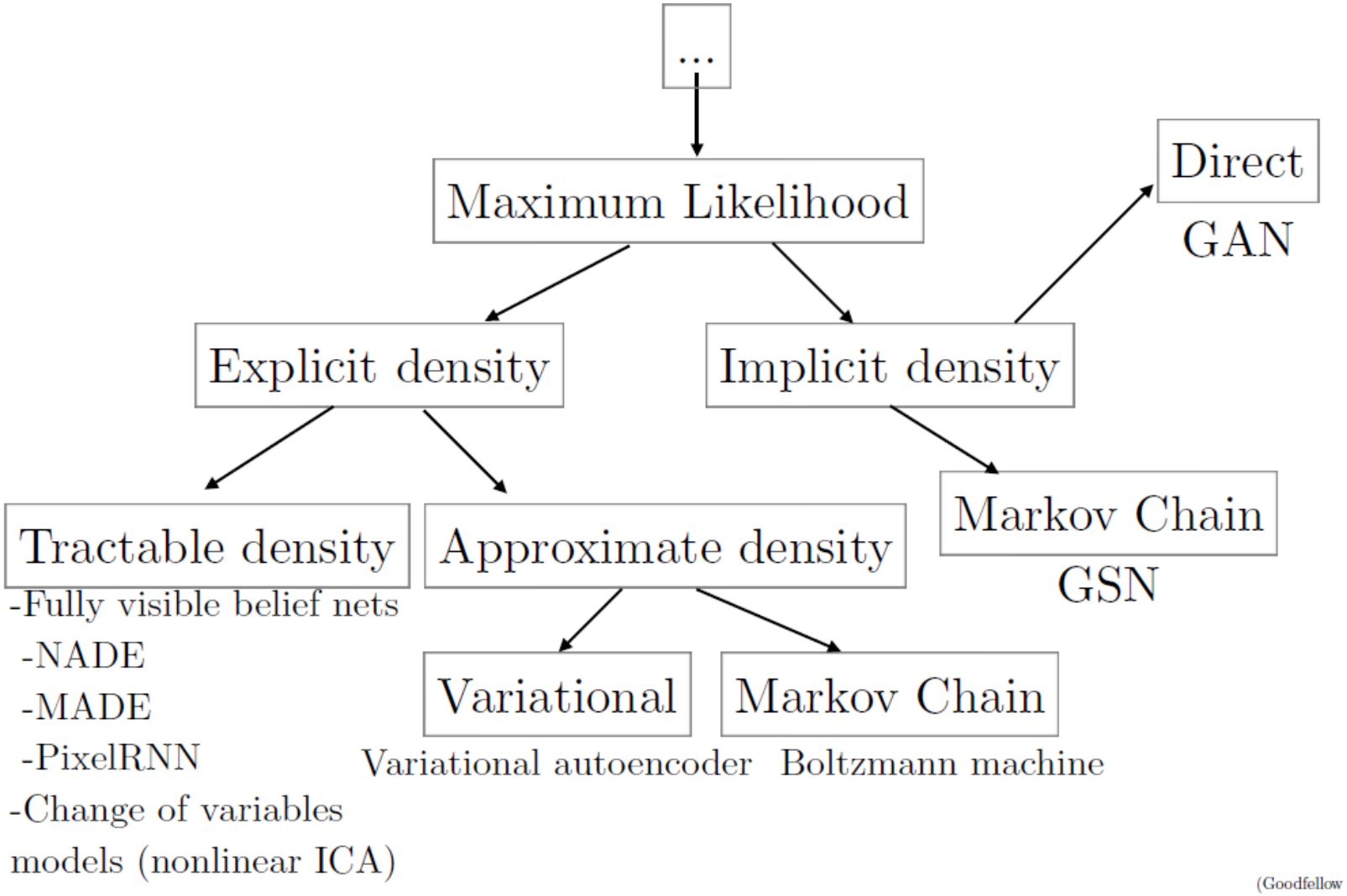
DSE 3121 DEEP LEARNING

Generative Adversarial Networks

Rohini Rao & Abhilash K Pai
Department of Data Science and Computer Applications
MIT Manipal

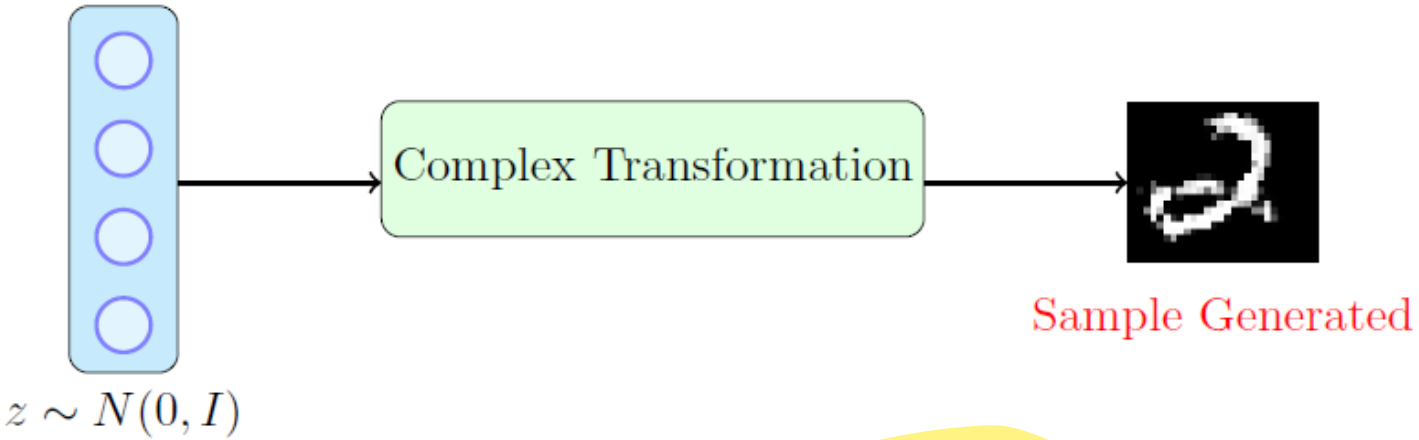
Credits: Most of the slides are adapted from CS7015 Deep Learning, Dept. of CSE, IIT Madras

Taxonomy of Generative Models



(Goodfellow 2016)

Generative Adversarial Networks: Overview



yt video link -> https://www.youtube.com/watch?v=MKedB9qOHi4&t=994s

- What can we use for such a complex transformation?
  - A Neural Network
- How do you train such a neural network?
  - Using a two-player game.

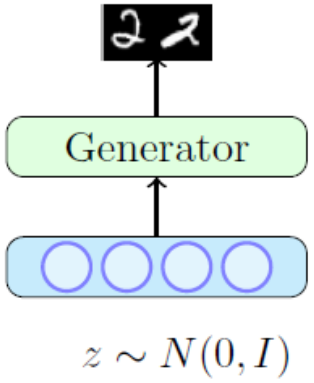
- Basic ideas of GANs is similar to Variational autoencoders (VAEs) where we sample from a simple tractable distribution and then learn a complex transformation on it so that the o/p looks as if it came from the training distribution.
- However, GANs start with a d-dimensional vector and generate a n-dimensional vector (usually, d<n) as compared to VAEs which start from n-dimensional i/p and produce o/p of same dimension.

Generative Adversarial Networks: Overview

GANs consists of two networks that involve in an adversarial game:

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

2. **Discriminator:** A neural network that distinguishes between output data point (Fake) from the Generator and training data samples (Real)

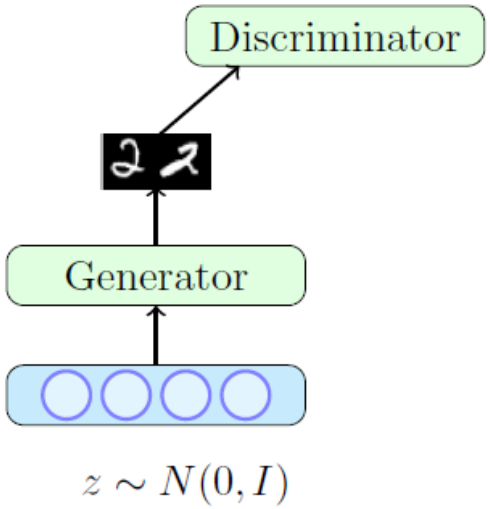


Generative Adversarial Networks: Overview

GANs consists of two networks that involve in an adversarial game:

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

2. **Discriminator:** A neural network that distinguishes between output data point (Fake) from the Generator and training data samples (Real)



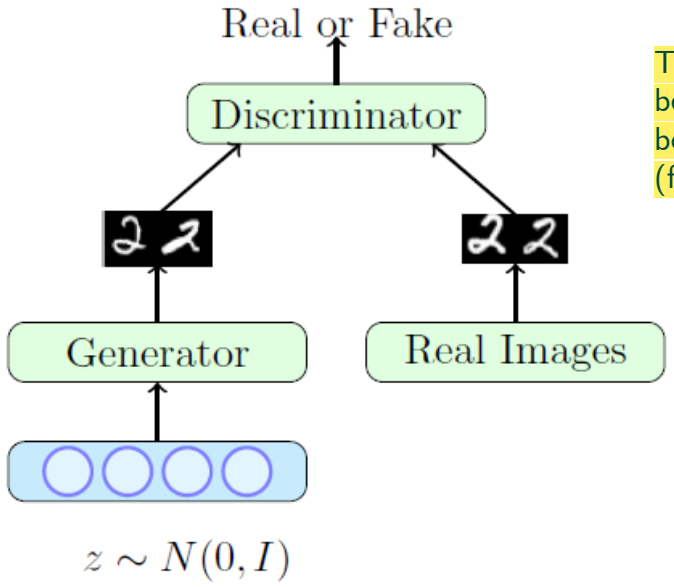
Generative Adversarial Networks: Overview

GANs consists of two networks that involve in an adversarial game:

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

2. **Discriminator:** A neural network that distinguishes between output data point (Fake) from the Generator and training data samples (Real)

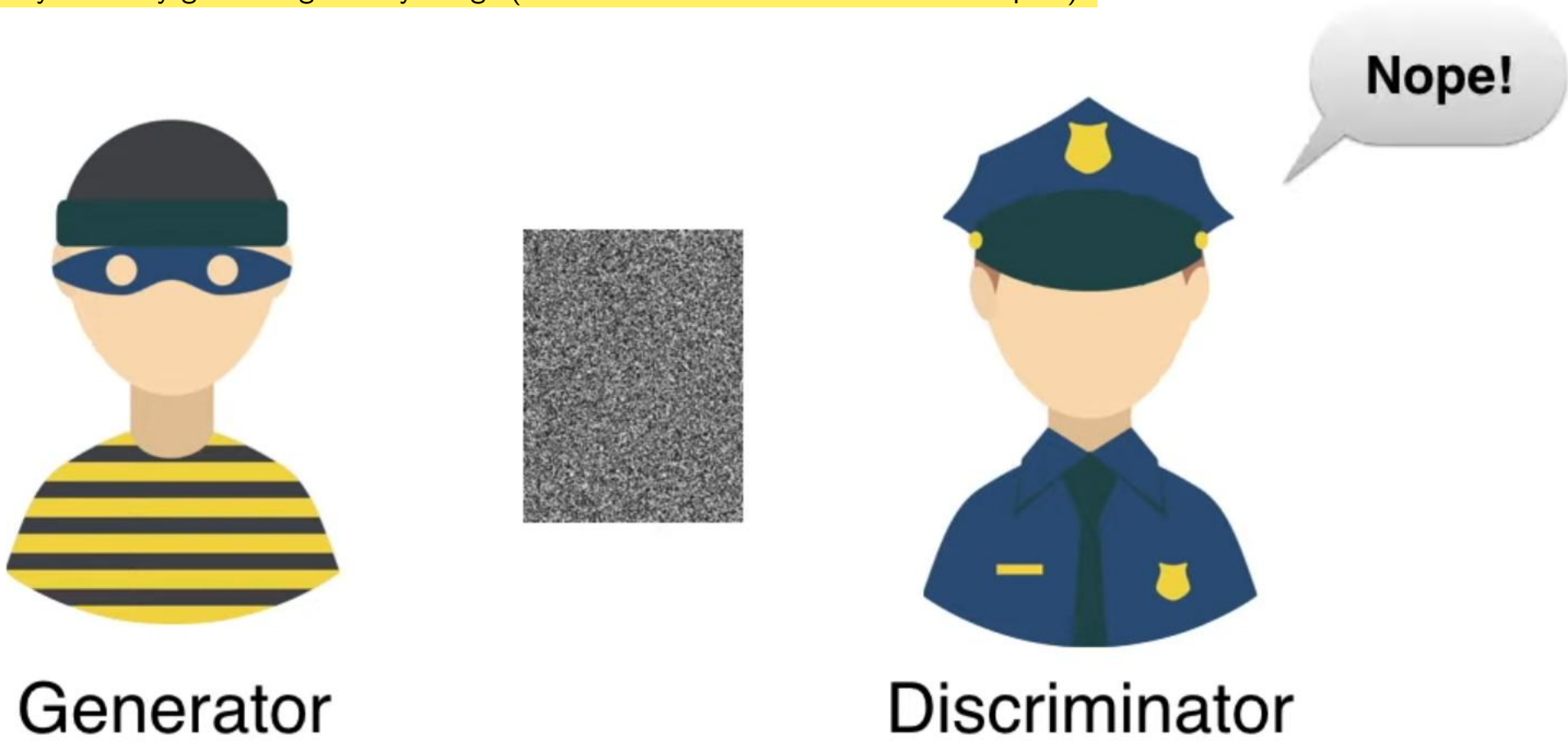
The job of the generator is to produce images which look so natural that the discriminator thinks that the images came from the real data distribution.



The job of the discriminator is to get better and better at distinguishing between true images and generated (fake) images

Generative Adversarial Networks: Overview

The generator mostly starts by generating a noisy image (as it takes random latent vectors as inputs).



A Friendly Introduction to Generative Adversarial Networks (GANs) - YouTube

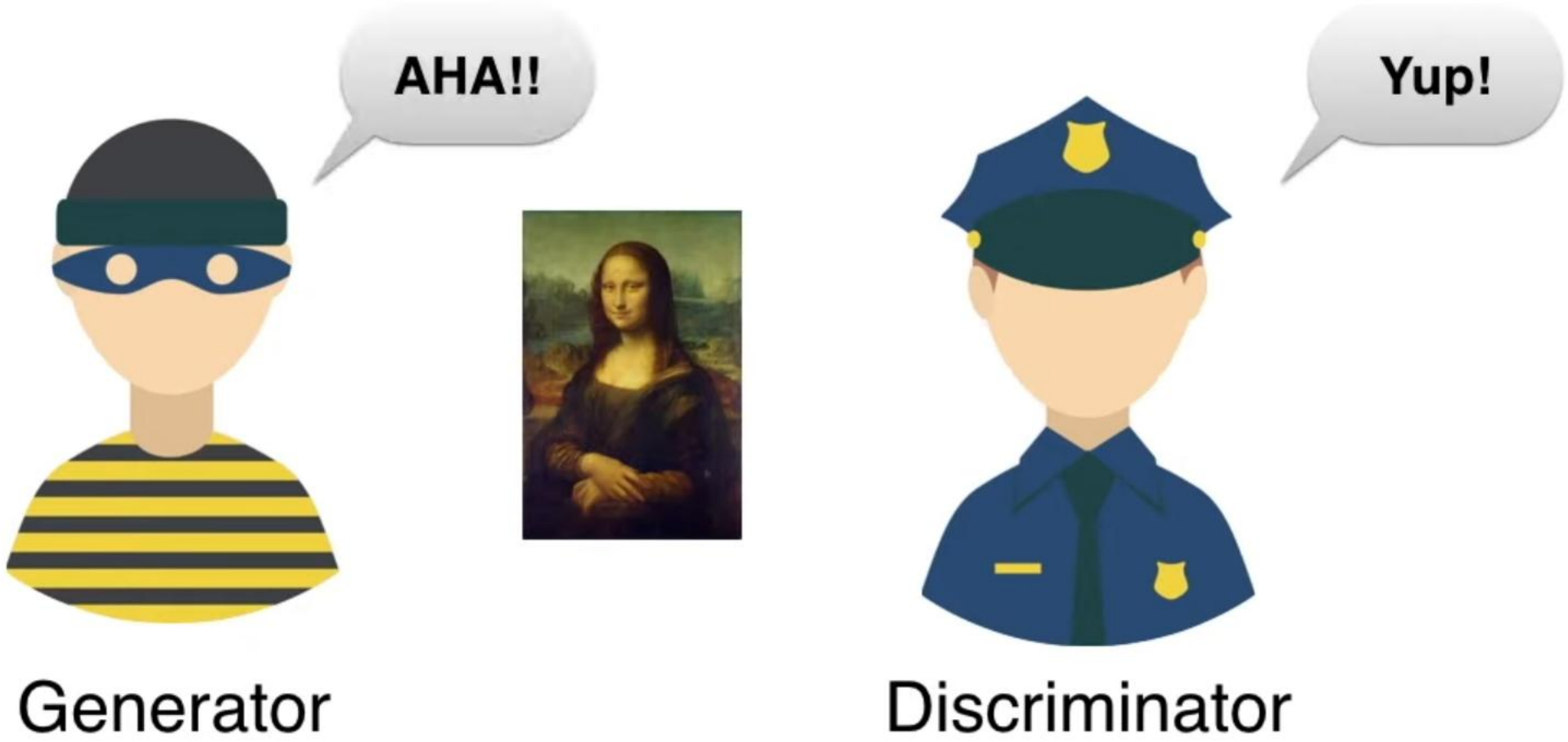
Generative Adversarial Networks: Overview

The generator output keeps improving during training.

A Friendly Introduction to Generative Adversarial Networks (GANs) - YouTube

Generative Adversarial Networks: Overview

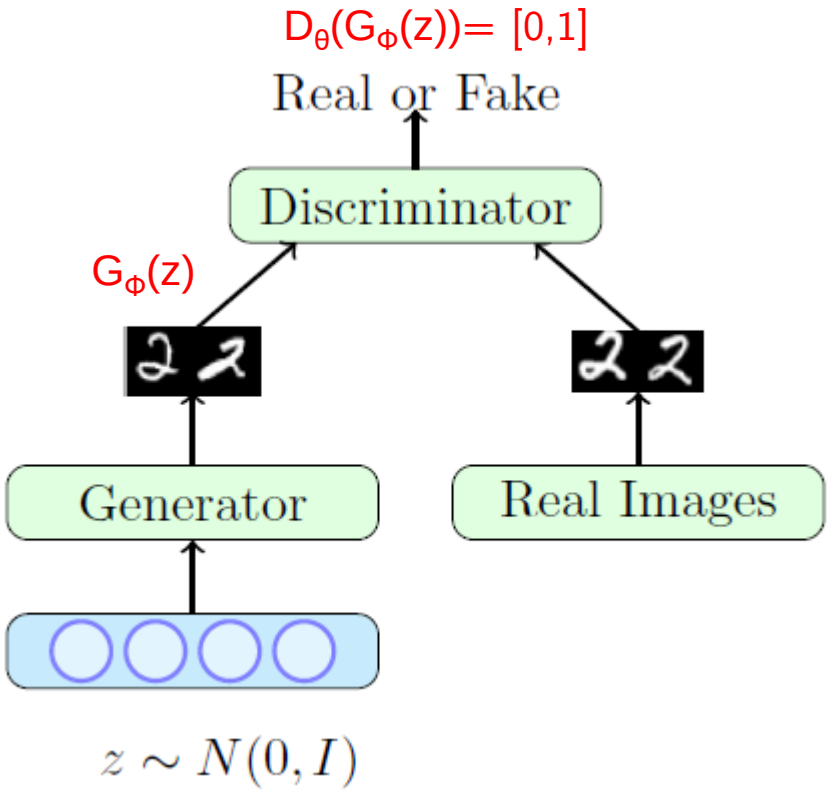
Equilibrium is reached when the generator finally succeeds to fool the discriminator.



A Friendly Introduction to Generative Adversarial Networks (GANs) - YouTube

Objective function of Generator

- Let G_Φ be the generator and D_θ be the discriminator (Φ and θ are the parameters of G and D, respectively)
- We have a neural network-based generator which takes as input a noise vector z N(0; I) and produces $G_\Phi(z) = X$
- We have a neural network-based discriminator which could take as input a real X or a generated $X = G_\Phi(z)$ and classify the input as real/fake
- Given an image generated by the generator as $G_\Phi(z)$the discriminator assigns a score $D_\theta(G_\Phi(z))$to it
- This score will be between 0 and 1 and will tell us the probability of the image being real or fake
- For a given z, the generator would want to maximize log $D_\theta(G_\Phi(z))$ (log likelihood) or minimize $\log(1 - D_\theta(G_\Phi(z)))$



Objective function of Generator

This is just for a single z and the generator would like to do this for all possible values of z,

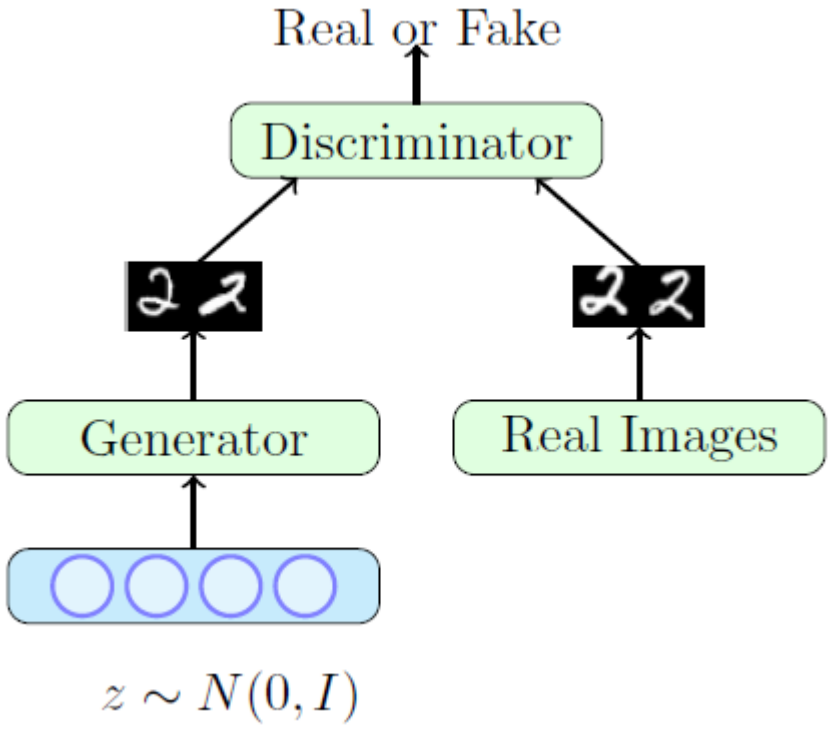
For example, if z was discrete and drawn from a uniform distribution (*i.e.*, $p(z) = \frac{1}{N}\ \forall z$) then the generator's objective function would be

$$\min_{\phi} \sum_{i=1}^{N} \frac{1}{N} \log(1 - D_\theta(G_\phi(z)))$$

However, in our case, z is continuous and not uniform ($z \sim N(0, I)$) so the equivalent objective function would be

$$\min_{\phi} \int p(z) \log(1 - D_\theta(G_\phi(z)))$$

$$\min_{\phi} E_{z \sim p(z)}[\log(1 - D_\theta(G_\phi(z)))]$$



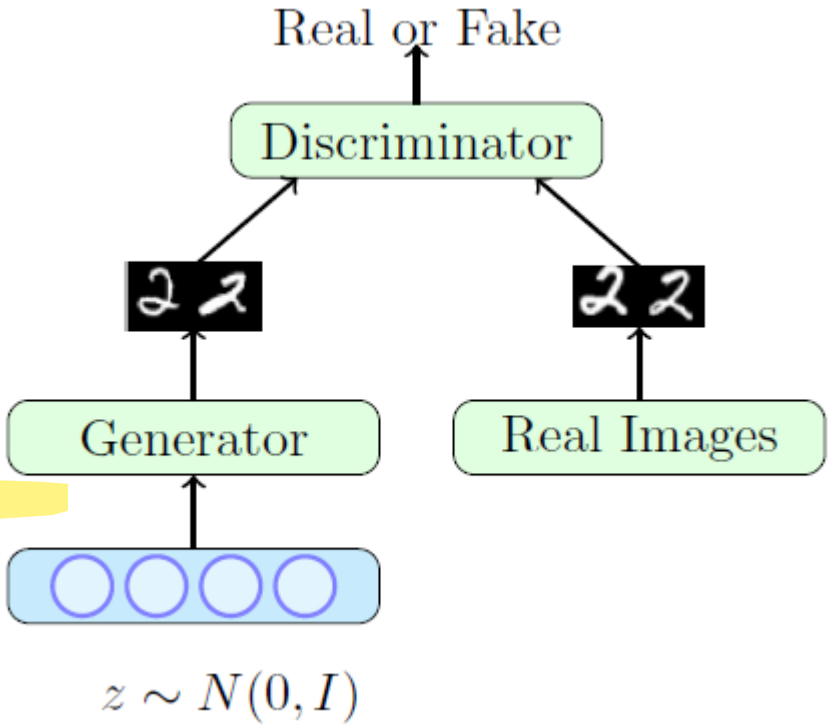
Objective function of Discriminator

- The task of the discriminator is to assign a high score to real images and a low score to fake images

- And it should do this for all possible real images and all possible fake images

- In other words, it should try to maximize the following objective function:

$$\max_{\theta} E_{x \sim p_{data}}[\log D_\theta(x)] + E_{z \sim p(z)}[\log(1 - D_\theta(G_\phi(z)))]$$



Generative Adversarial Networks: MiniMax formulation

Objective function:

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

Maximizes the obj. fn. w.r.t to the Discriminator network parameters

Minimizes the obj. fn. w.r.t to the Generator network parameters

The discriminator wants to maximize the second term whereas the generator wants to minimize it (hence it is a two-player game)

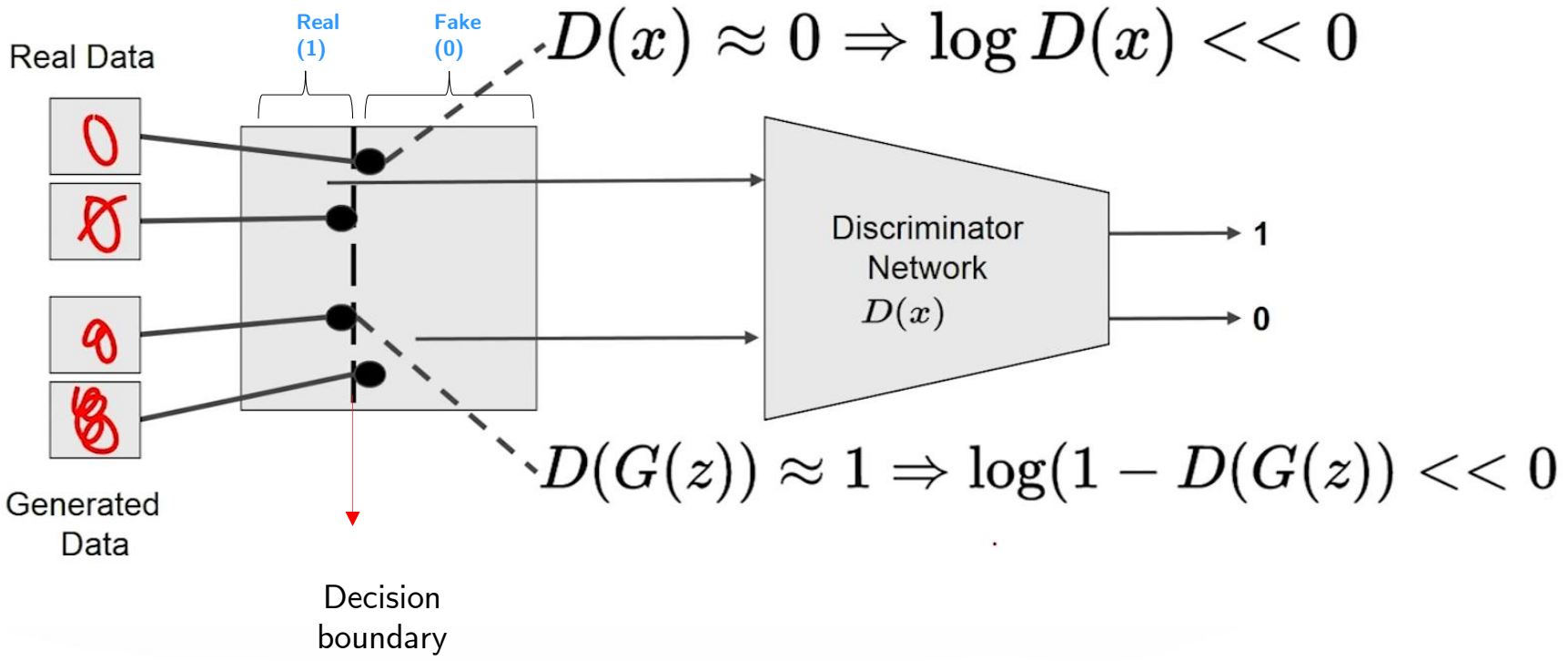
Generative Adversarial Networks (GAN) - YouTube

Training the Discriminator Network

- Before training (when discriminator is not performing optimally- cannot clearly distinguish real and fake data)

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

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



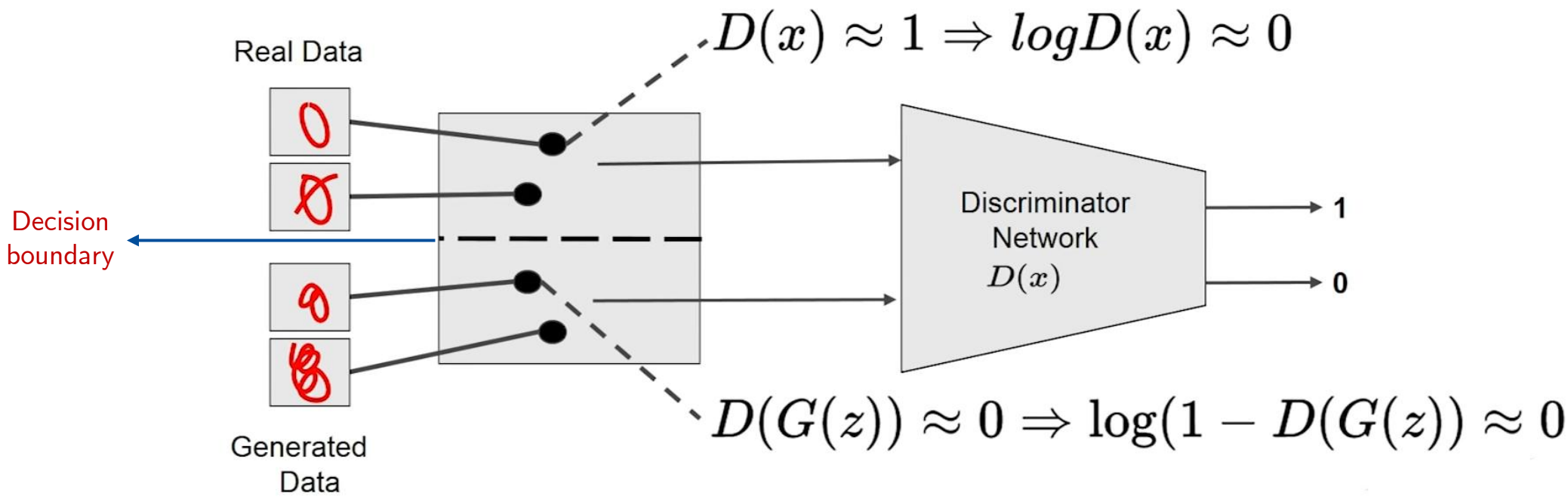
Generative Adversarial Networks (GAN) - YouTube

Training the Discriminator Network

- After training (when discriminator is performing optimally – clearly distinguishes real and fake data)

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

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



Generative Adversarial Networks (GAN) - YouTube

Training the Generator Network

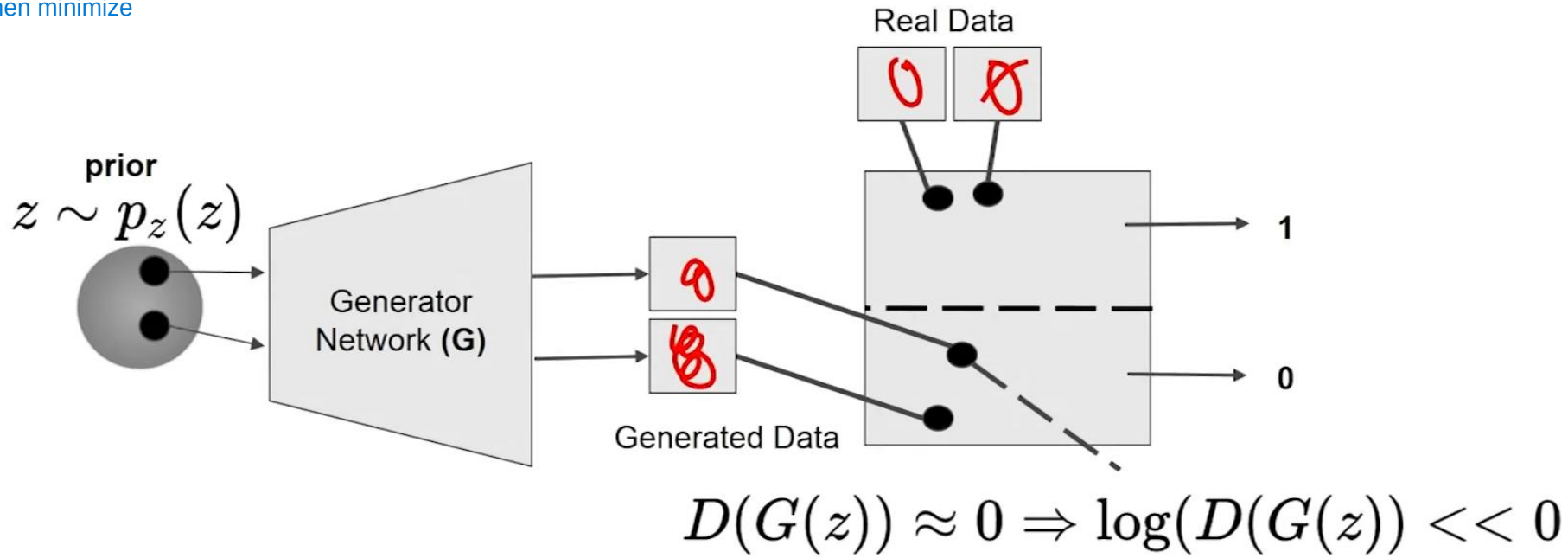
- Before training

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

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

Tries to maximize the error in D.
That is, it incorrectly classifies D(G(z)) as 1 instead of 0

if it was 1 - then minimize



$$D(G(z)) \approx 0 \Rightarrow \log(D(G(z)) << 0$$

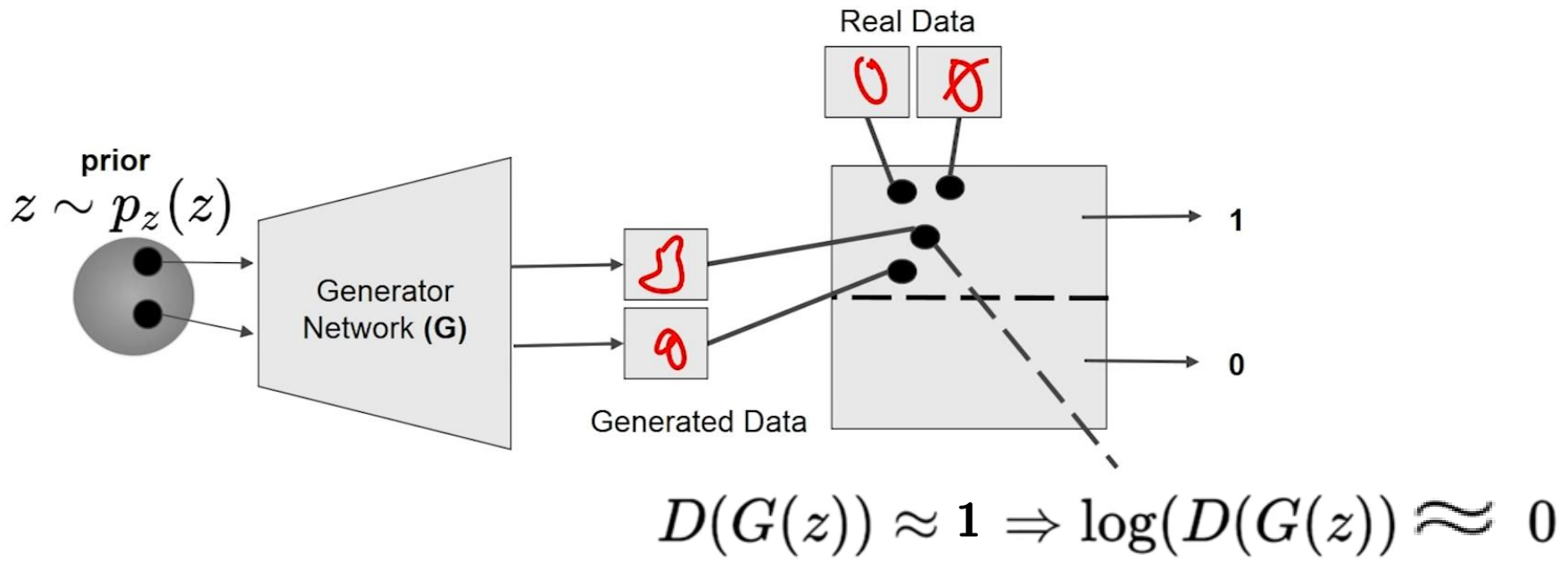
Generative Adversarial Networks (GAN) - YouTube

Training the Generator Network

- After training

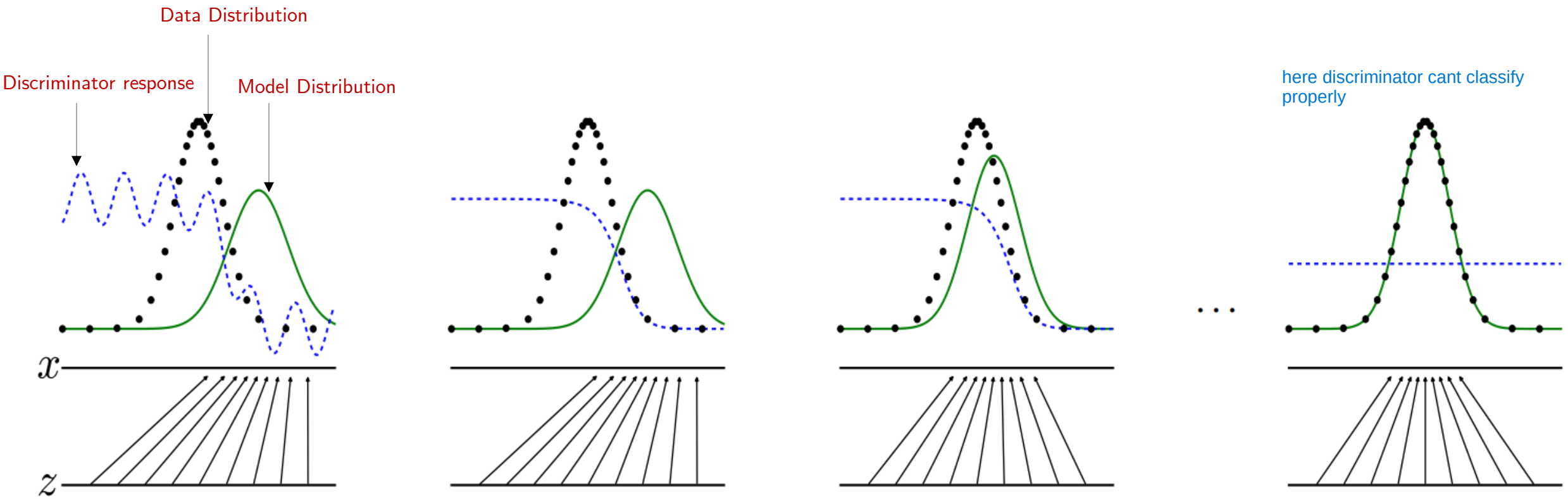
$$\max_G [\mathbb{E}_{z \sim P_z(z)} log(D(G(z)))]$$

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



$$D(G(z)) \approx 1 \Rightarrow \log(D(G(z)) \approx 0$$

Training the Generator Network



Poorly fit model

After updating D

After updating G

Mixed strategy equilibrium

Generative Adversarial Networks (GAN) - YouTube

Training the GANs

procedure GAN TRAINING

  for number of training iterations **do**

    for k steps **do**

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, .., \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$
- Sample minibatch of m examples $\{\mathbf{x}^{(1)}, .., \mathbf{x}^{(m)}\}$ from data generating distribution $p_{data}(\mathbf{x})$
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} \left[\log D_\theta \left(x^{(i)} \right) + \log \left(1 - D_\theta \left(G_\phi \left(z^{(i)} \right) \right) \right) \right]$$

    end for

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, .., \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$
- Update the generator by ascending its stochastic gradient

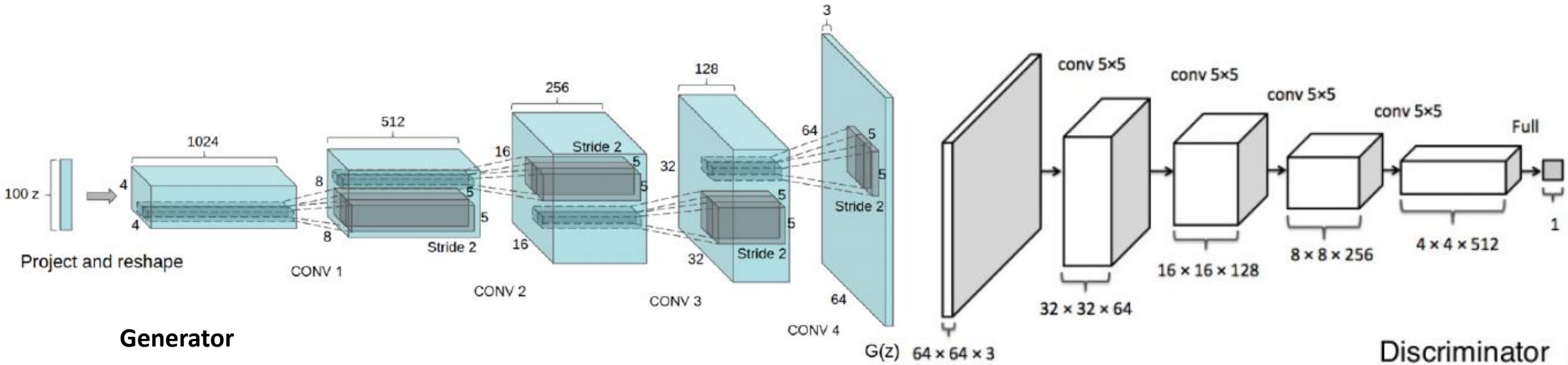
$$\nabla_\phi \frac{1}{m} \sum_{i=1}^{m} \left[\log \left(D_\theta \left(G_\phi \left(z^{(i)} \right) \right) \right) \right]$$

  end for

end procedure

Deep Convolutional GANs

For discriminator, any CNN based classifier with 1 class (real) at the output can be used (e.g. VGG, ResNet, etc.)



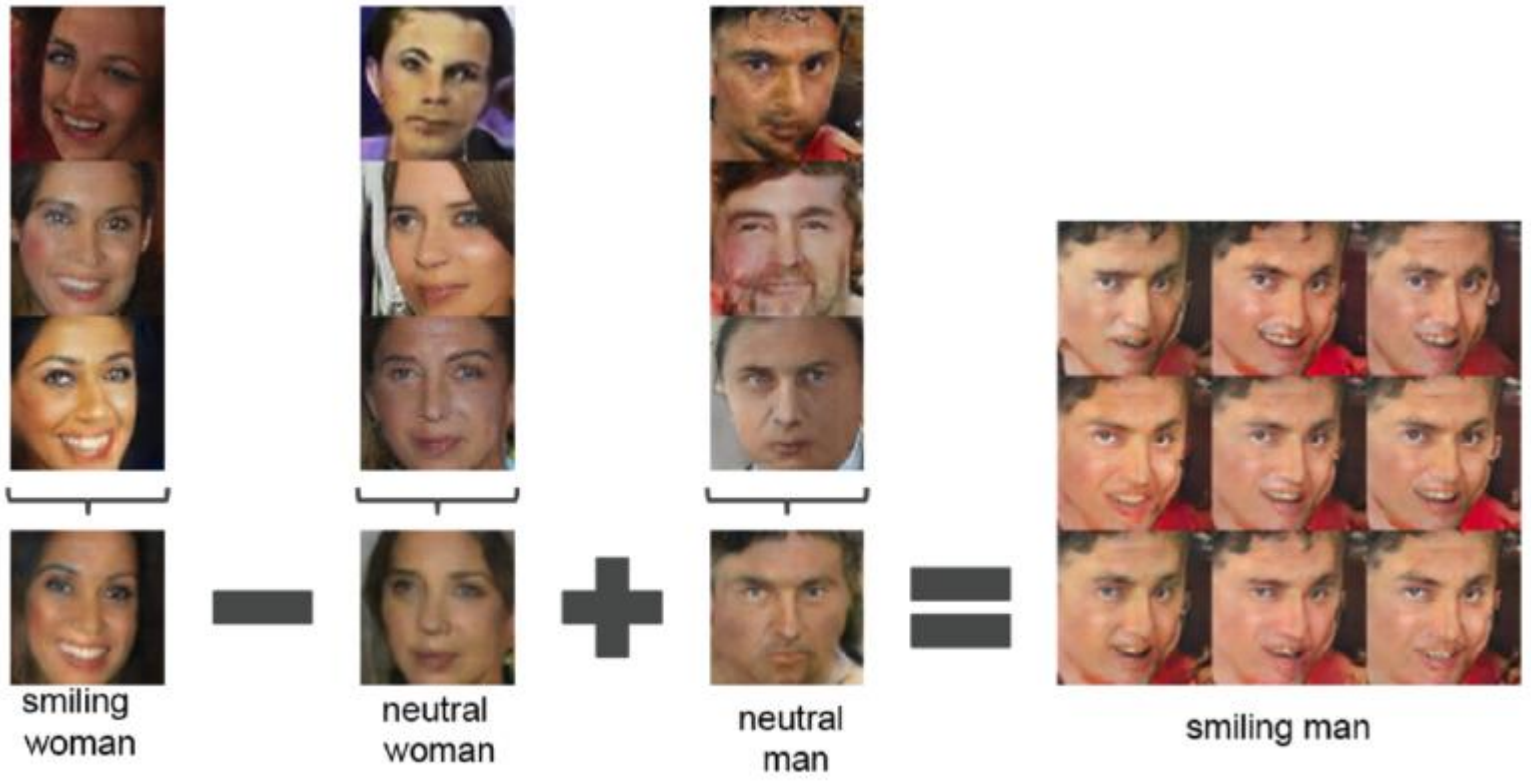
Radford et al. Unsupervised Representational Learning with Deep Convolutional GANs, ICLR 2016

Deep Convolutional GANs

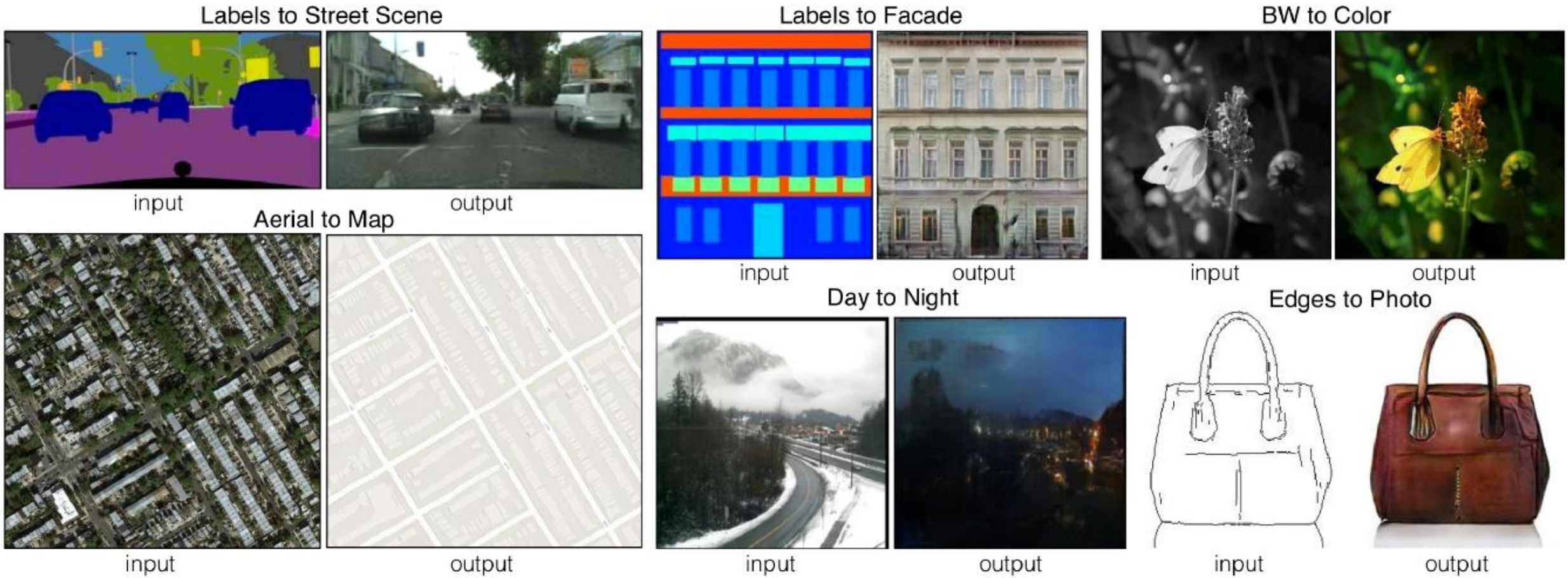
Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses tanh.
- Use LeakyReLU activation in the discriminator for all layers

GANs: Applications



GANs: Applications



Philip et al. Image-to-Image Translation with Conditional Adversarial Networks, CVPR 2017