بسم الله الرحمن الرحیم

Deep Learning

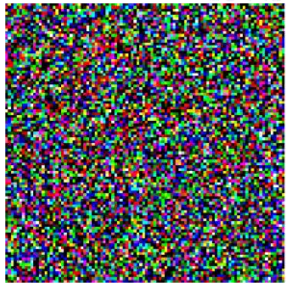
Mohammad Reza Mohammadi

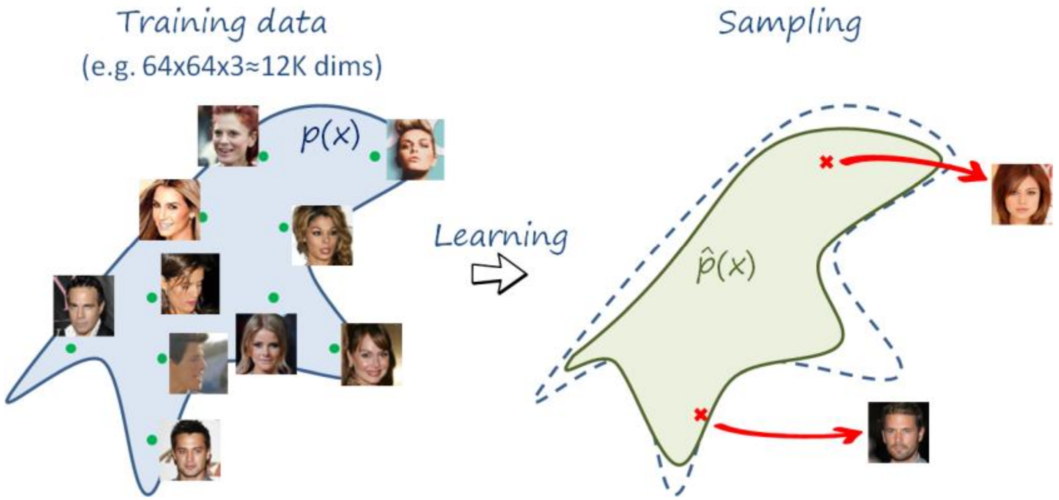
2021

Generative Models

Generative models

- Given training data, generate new samples from same distribution

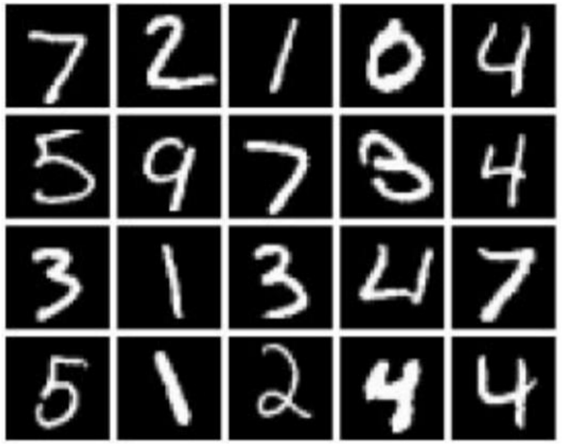




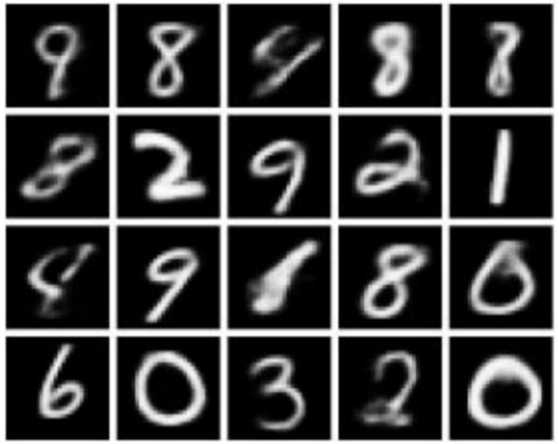
- Training data $\sim p_{data}(x)$
- Want to learn $p_{model}(x)$ similar to $p_{data}(x)$
- Generated samples $\sim p_{model}(x)$

Generative adversarial networks

- Generative Adversarial Networks (GAN) don't work with any explicit density function!
- Instead, take game-theoretic approach:
  - Learn to generate from training distribution through 2-player game



$\Rightarrow p_{\text{model}}(x) \Rightarrow$

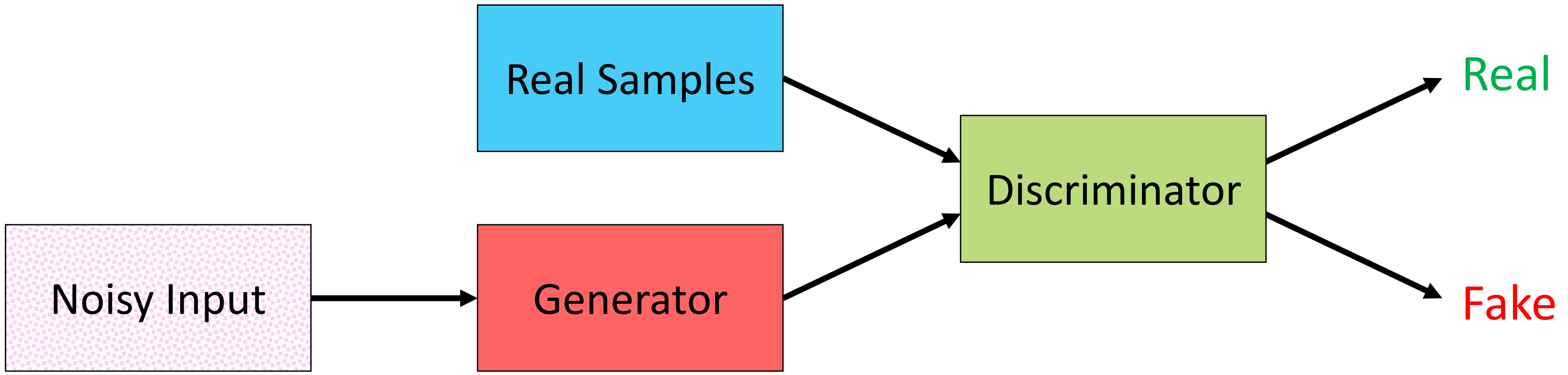


$training\ data \sim p_{\text{data}}(x)$ $\qquad$ $generated\ samples \sim p_{\text{model}}(x)$

Goodfellow, Ian, et al. "Generative adversarial nets." *Advances in neural information processing systems* (2014).

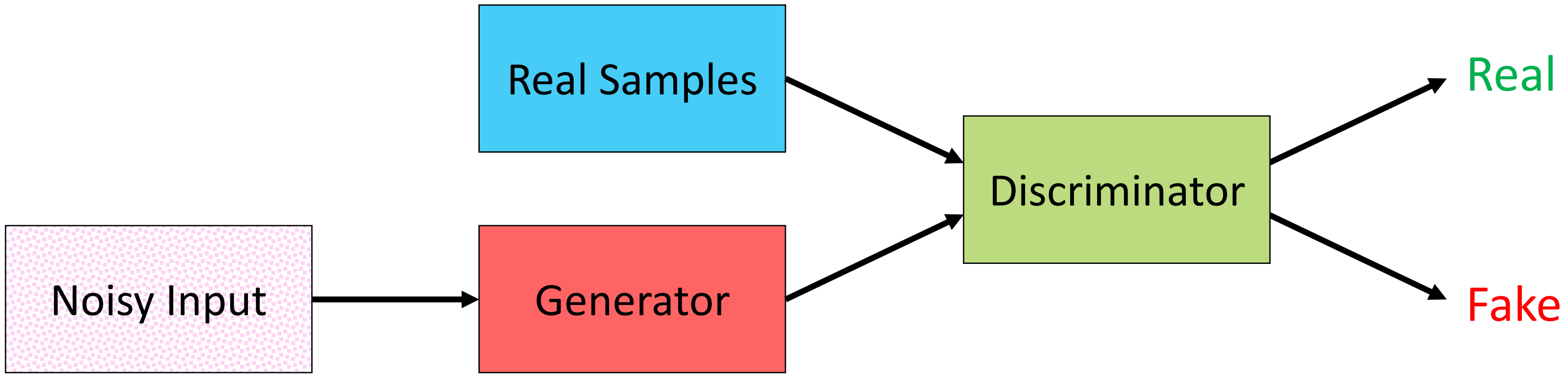
GAN

- The GAN model architecture involves two sub-models
  - Generator: Model that is used to generate new plausible examples from the problem domain
  - Discriminator: Model that is used to classify examples as real (from the domain) or fake (generated)

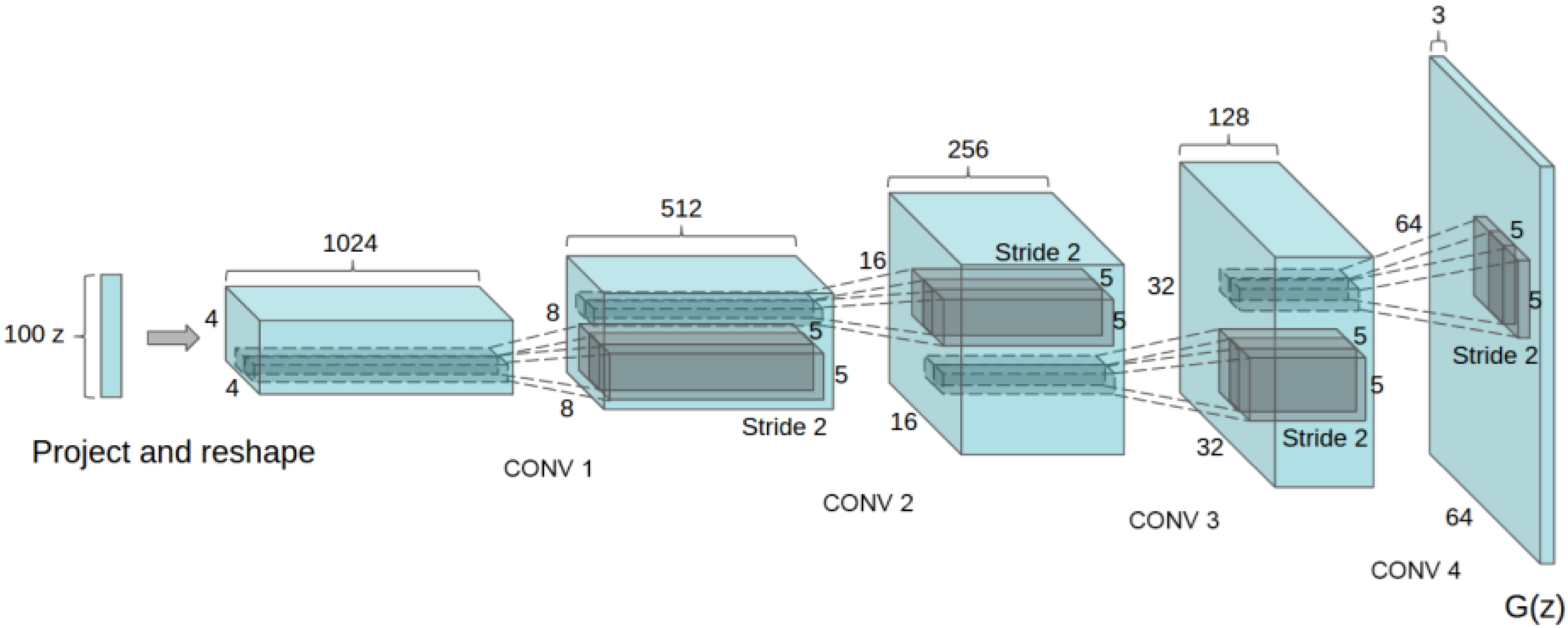


Generator network

- Want to sample from complex, high-dimensional training distribution
  - No direct way to do this!
  - Solution:
    - Sample from a simple distribution, e.g. random noise
    - Learn transformation to training distribution
    - We use a deep neural network to represent this complex transformation

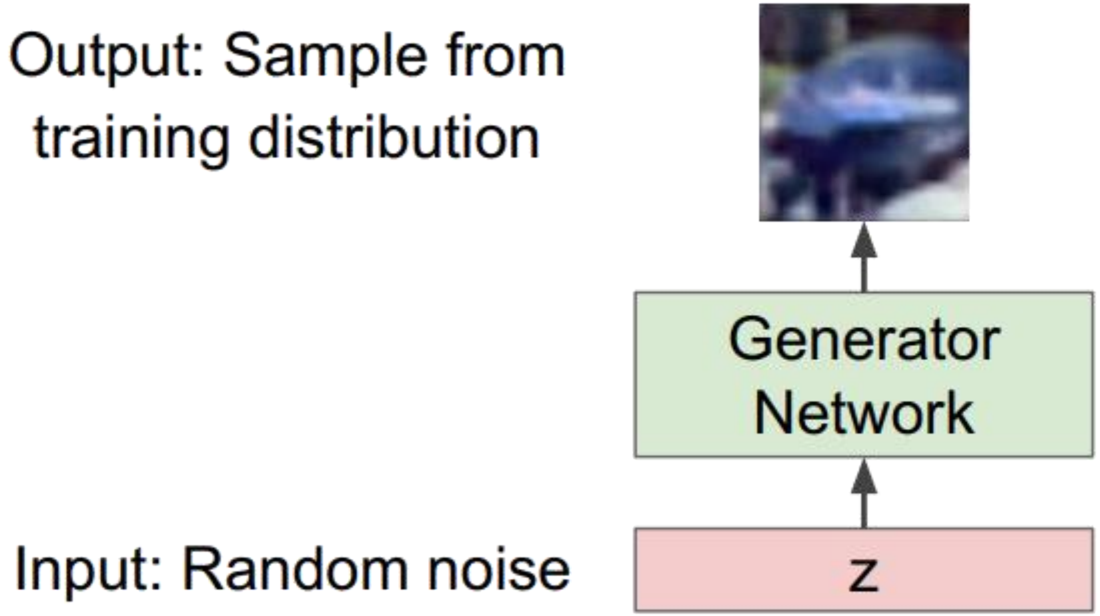


Generator network

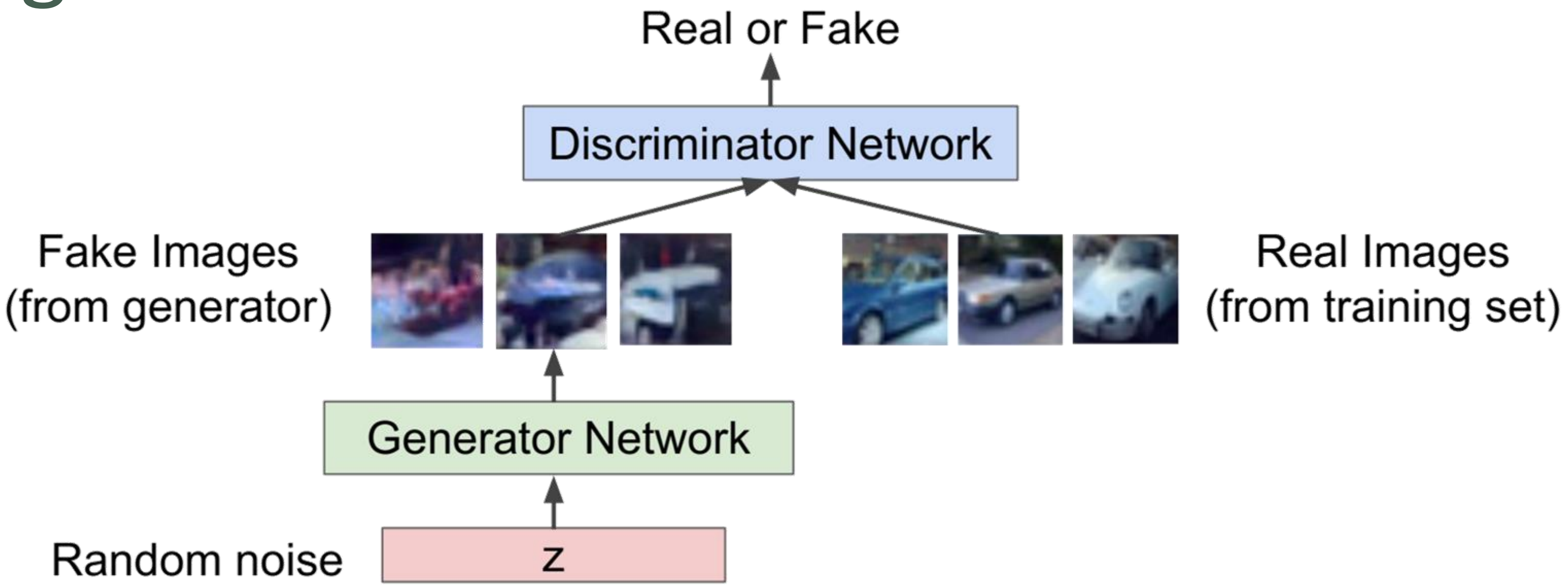


Training GANs

- How to optimize parameters?
- We need to evaluate the quality of the images produced
  - We use a discriminator network!



Training GANs

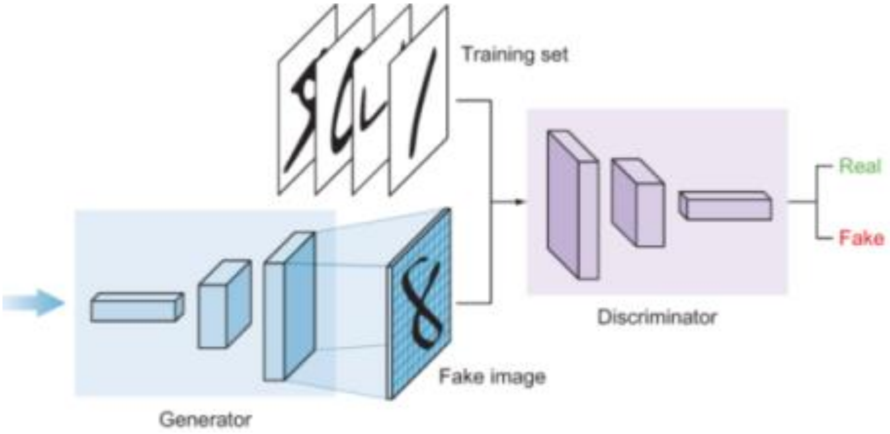


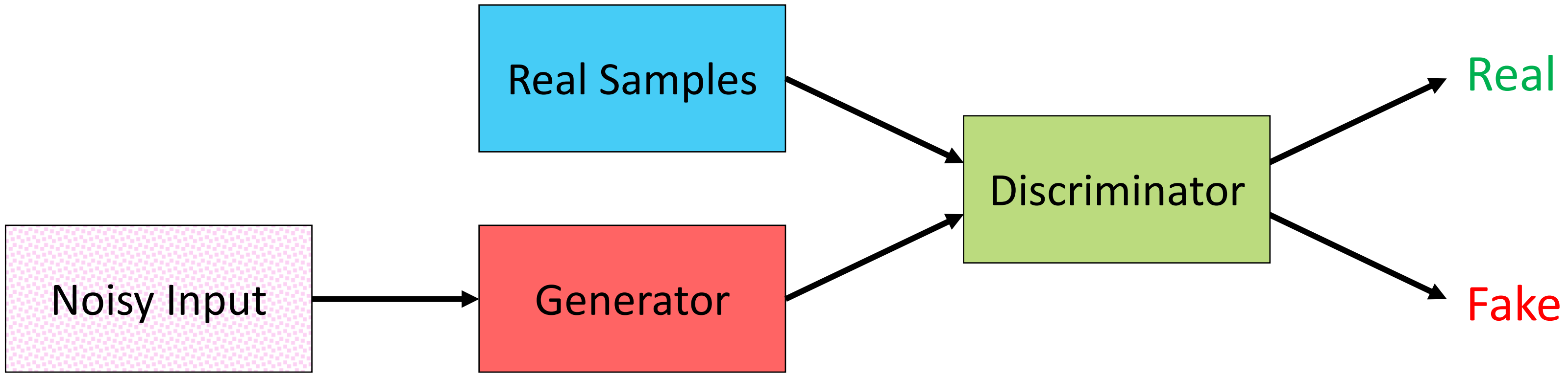
$$\min_{\theta_g}\max_{\theta_d}\left[\mathbb{E}_{x\sim p_{data}}\log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z\sim p(z)}\log(1-\underbrace{D_{\theta_d}(G_{\theta_g}(z)))}_{\text{Discriminator output for generated fake data G(z)}}\right]$$

Training GANs

- Discriminator (θ_d) wants to maximize objective such that $D(x)$ is close to 1 (real) and $D\big(G(z)\big)$ is close to 0 (fake)
- Generator (θ_g) wants to minimize objective such that $D\big(G(z)\big)$ is close to 1

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x\sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z\sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))\right]$$





Training GANs

- Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

- Alternate between:
  - Gradient ascent on discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

  - Gradient descent on generator

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

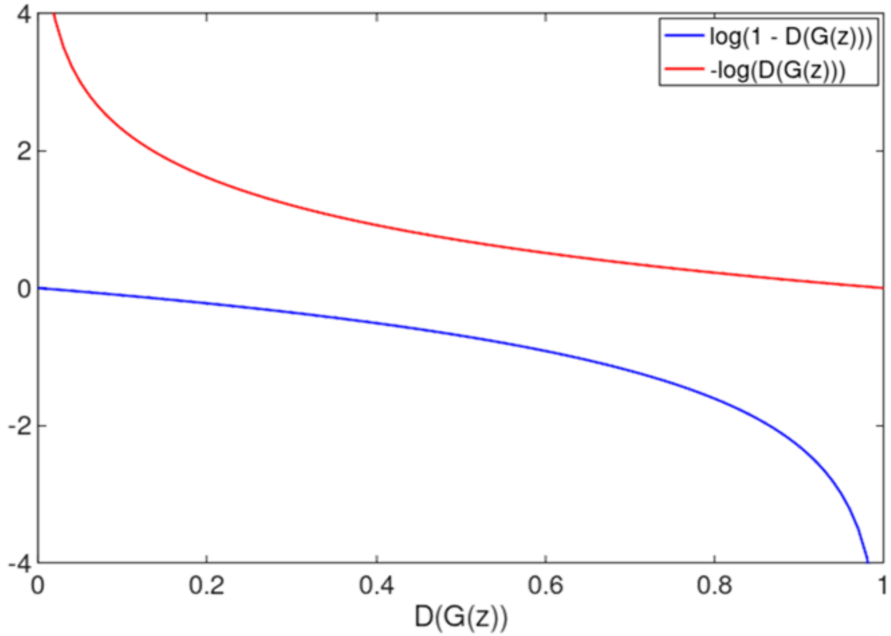
Training GANs

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x\sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z\sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

$$\max_{\theta_d} \left[\mathbb{E}_{x\sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z\sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

$$\min_{\theta_g} \mathbb{E}_{z\sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

$$\max_{\theta_g} \mathbb{E}_{z\sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$



GAN pseudocode

for number of training iterations **do**

for k steps **do**

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

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

end for

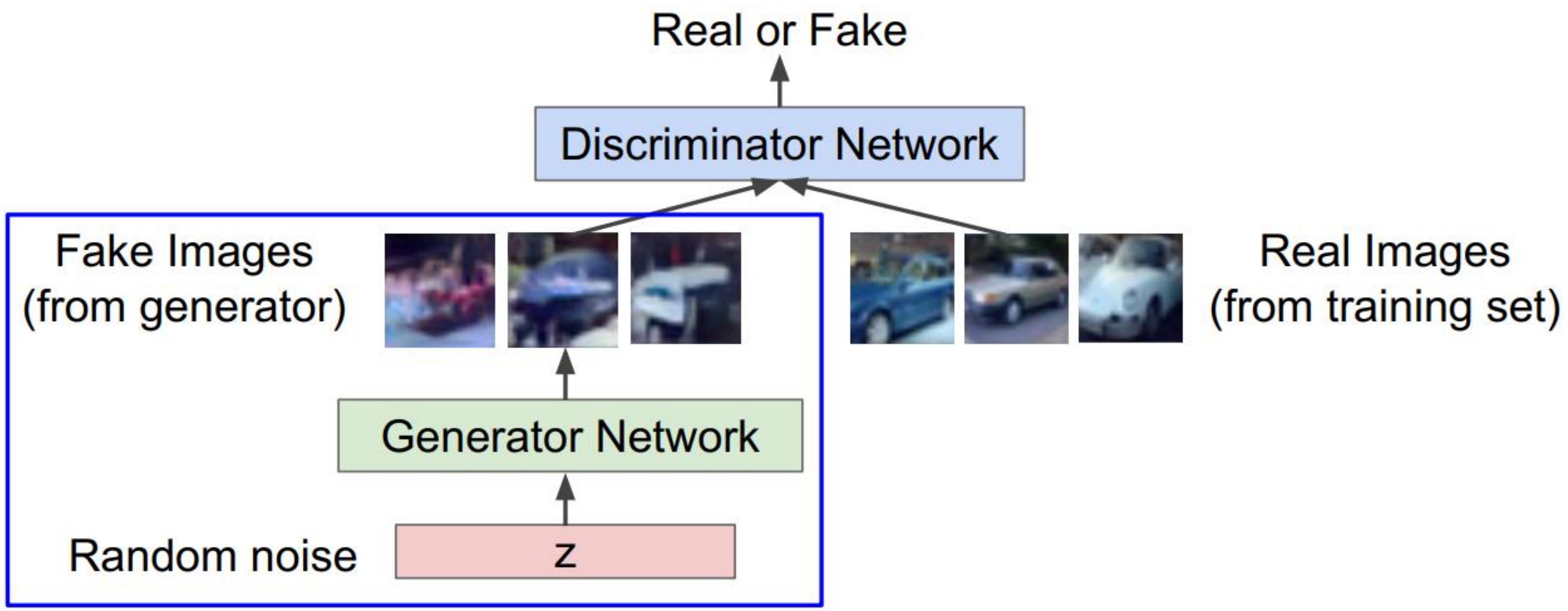
- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by ascending its stochastic gradient (improved objective):

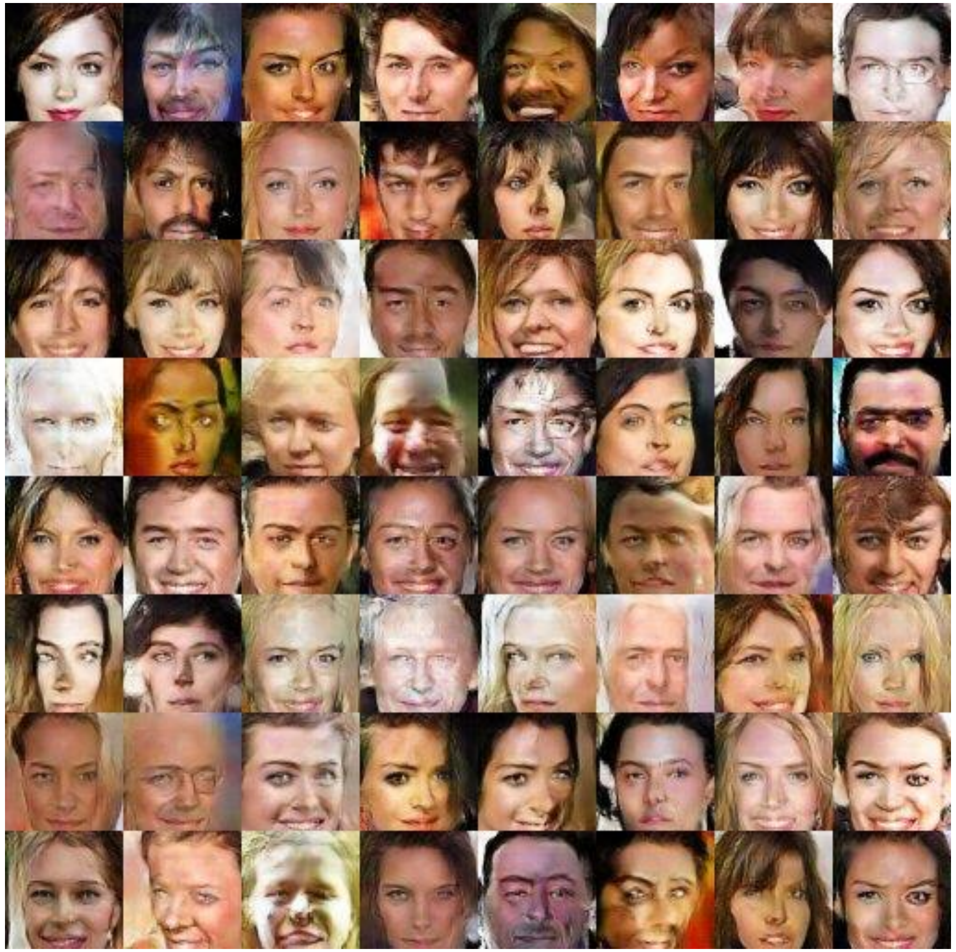
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

end for

Generator

- After training, use generator network to generate new images

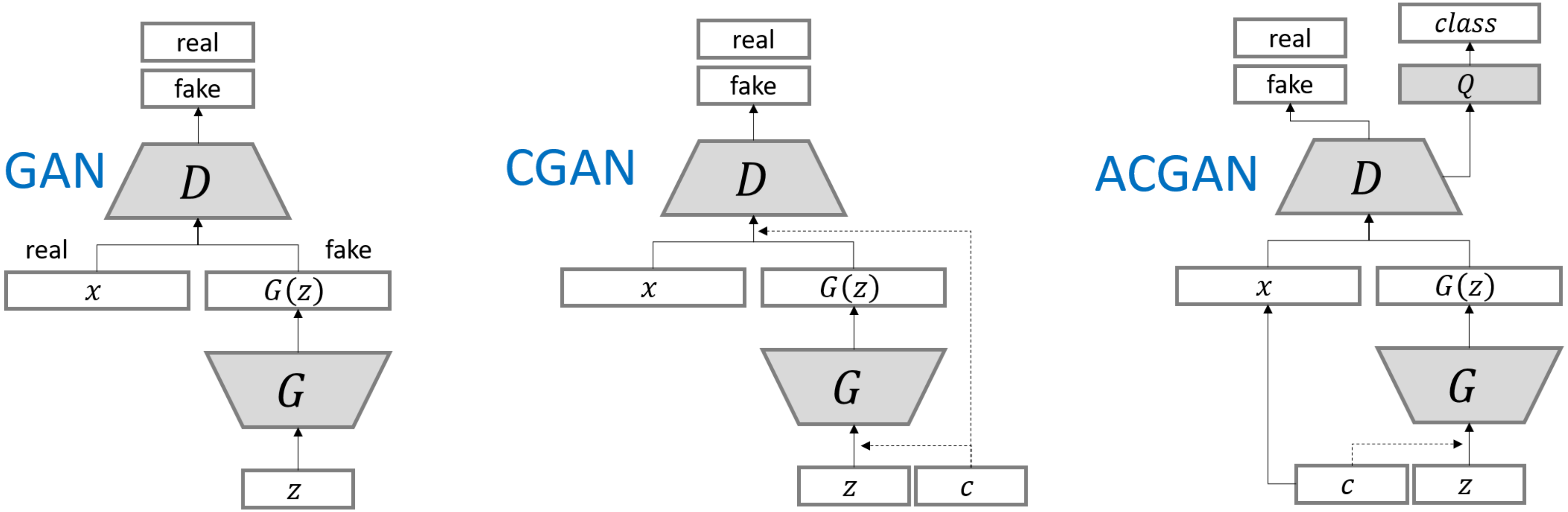




https://github.com/eriklindernoren/Keras-GAN/blob/master/gan/gan.py

Conditional GAN

- In a CGAN, the generator and the discriminator are conditioned on c, which could be a class label or some data from another modality
- In Auxiliary Classifier GANs, the discriminator is forced to identify fake and real images, as well as the class of the image, irrespective of whether it is fake or real



A. Odena, C. Olah, and J. Shlens. "Conditional image synthesis with auxiliary classifier gans." *International conference on machine learning*, 2017.