Generative Adversarial Networks (GAN)

Narada Warakagoda

Generative Models

- Given a set of data samples

$$X = \{x_1, x_2, x_3, \cdots, x_N\}$$

- Estimate the probability distribution $p_\theta(x)$ these samples are drawn from.
  - θ is the model parameters
- Typically unsupervised learning
- Applications:
  - Classification
  - Anomaly detection
  - Generation of similar samples

Maximum Likelihood with Parametric Probability model

- The most common approach
- Steps:
  - Assume a parametric probability distribution with a convenient mathematical form (eg: Gaussian)

$$\mathrm{p}_\theta(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp -\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2, \quad \text{where } \theta = \text{ parameters } (\mu, \sigma)$$

  - Find the likelihood

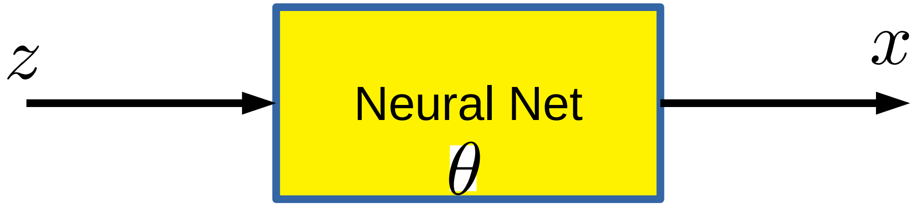
$$\mathrm{L}(\theta) = \sum_{i=1}^{N} \log p_\theta(x_i)$$

  - Maximize the likelihood with respect to model parameters

$$\theta^\star = \arg\max_\theta L(\theta)$$

- Problem
  - Difficult to model complex probability distributions

More capable approach



- z is a random variable drawn from a simple distribution such as Gaussian, i.e. $z \sim \mathcal{N}(0, 1)$
- By adjusting parameters θ, the distribution of x can be made to resemble the distribution of the given data set $X = \{x_1, x_2, \cdots, x_N\}$
- Problem:
  - How to change the parameters θ so that the "distance" between the distribution of x and data distribution $d(p_{\mathrm{model}}(x), p_{\mathrm{data}}(x))$

Distance between two probability distributions

- Kullback-Leibler (KL) Divergence

$$D_{KL}(p\|q) = \int_x p(x) \log \frac{p(x)}{q(x)} dx$$

- Jensen-Shannon Divergence

$$D_{JS}(p\|q) = \frac{1}{2} D_{KL}(p\|\frac{p+q}{2}) + \frac{1}{2} D_{KL}(q\|\frac{p+q}{2})$$

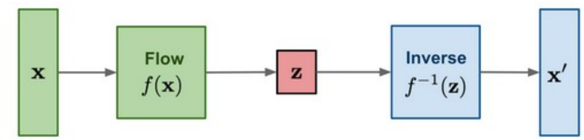
- Wasserstein Distance

$$D_W(p,q) = \inf_{\gamma \sim \Pi(p,q)} \mathbb{E}_{(x,y)\sim\gamma}[||x-y||]$$

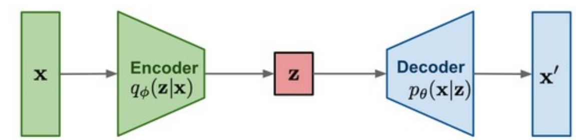
  - where $\int \gamma(x,y)dy = p(x) \quad \text{and} \quad \int \gamma(x,y)dx = q(y)$

Generative modelling approaches with Deep Learning

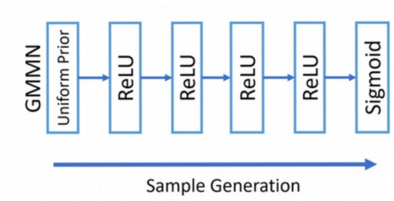
- Normalizing Flows



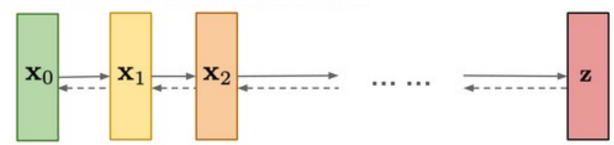
- Variational Auto Encoders (VAE)



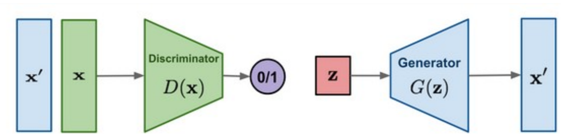
- Moment Matching Networks (MMN)



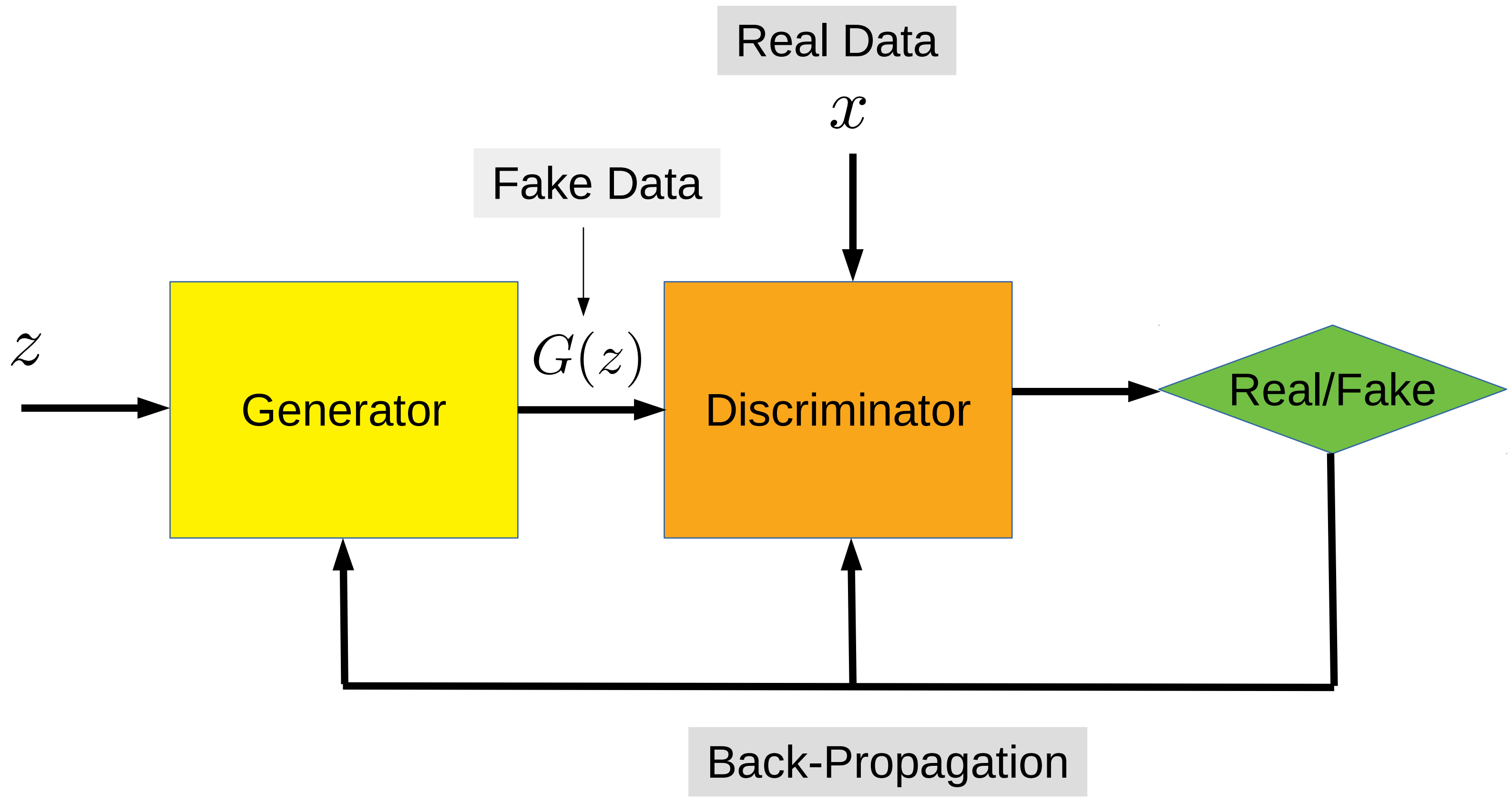
- Diffusion models



- Generative Adversarial Networks (GAN)



Generative Adversarial Networks (GAN)



GAN Operation

- Two networks: Generator G and Discriminator D
- Generator:
  - Generates data samples based on random input
  - Tries to fool the Discriminator
- Discriminator:
  - Tries to classify real data samples against fake data samples
  - Assign high probability to real data (1)
  - Assign low probability to fake data (0)
- They engage in a minimax game.

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

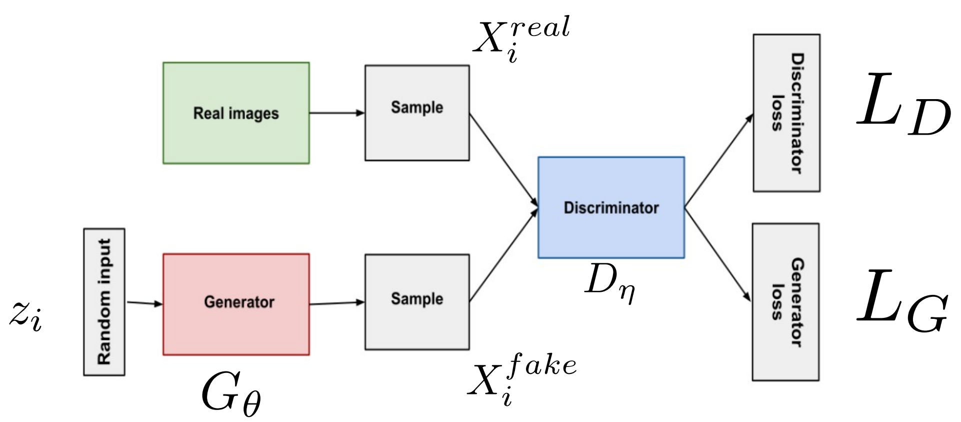
GAN Operation (II)

- Discriminator loss

$$L_D = -\Big(\frac{1}{m}\sum_{i=1}^{m}\log(D_\eta(x_i^{real})) + \frac{1}{m}\sum_{i=1}^{m}\log(1 - D_\eta(x_i^{fake}))\Big)$$

- Equivalently

$$L_D = -\Big(\frac{1}{m}\sum_{i=1}^{m}\log(D_\eta(x_i)) + \frac{1}{m}\sum_{i=1}^{m}\log(1 - D_\eta(G_\theta(z_i)))\Big)$$



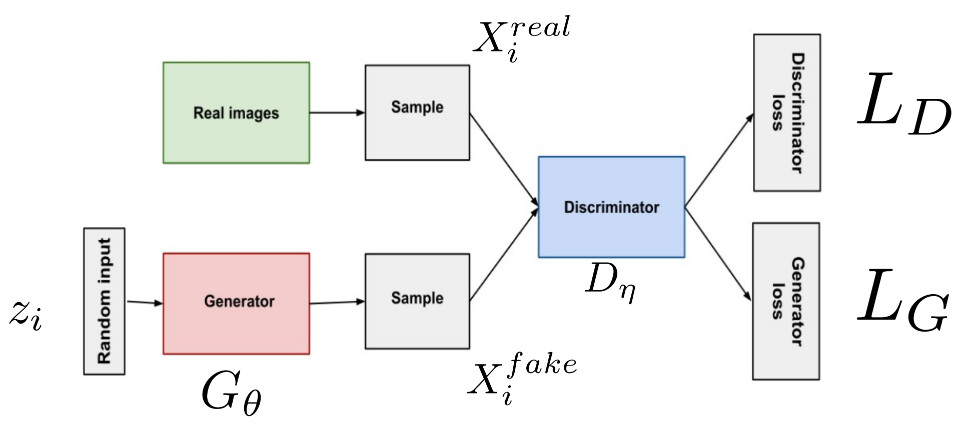
GAN Operation (III)

- Generator loss

$$L_G = -L_D = \frac{1}{m}\sum_{i=1}^{m}\log(D_\eta(x_i)) + \frac{1}{m}\sum_{i=1}^{m}\log(1 - D_\eta(G_\theta(z_i)))$$

- Equivalently (since the first term is independent of G)

$$L_G = \frac{1}{m}\sum_{i=1}^{m}\log(1 - D_\eta(G_\theta(z_i)))$$



MiniMax Solution

- Let
  - $p_r(x) = \text{probability distribution of real data samples}$
  - $p_g(x) = \text{probability distribution of generated (fake) data samples}$
- The minimax problem is

$$\begin{aligned}\min_G \max_D L(D,G) &= \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))] \\ &= \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{x\sim p_g(x)}[\log(1-D(x)]\end{aligned}$$

- The minimax objective can be written as

$$L(G,D) = \int_x \Big(p_r(x)\log(D(x)) + p_g(x)\log(1-D(x)) \Big) dx$$

- Differentiate $L(G,D)$ wrt $D(x)$ and equate the result to zero to find the optimum $D^\star(x)$
- This will give $D^\star(x) = \frac{p_r(x)}{p_r(x)+p_g(x)}$
- When G is also optimum $p_r(x) = p_g(x)$ and $D^\star(x) = \frac{1}{2}$

Generator optimization objective

$$
\begin{aligned}
D_{JS}(p_r \| p_g) =& \frac{1}{2} D_{KL}(p_r || \frac{p_r + p_g}{2}) + \frac{1}{2} D_{KL}(p_g || \frac{p_r + p_g}{2}) \\
=& \frac{1}{2}\left(\log 2 + \int_x p_r(x) \log \frac{p_r(x)}{p_r + p_g(x)} dx \right) + \\
& \frac{1}{2}\left(\log 2 + \int_x p_g(x) \log \frac{p_g(x)}{p_r + p_g(x)} dx \right) \\
=& \frac{1}{2}\left(\log 4 + L(G, D^*) \right)
\end{aligned}
$$

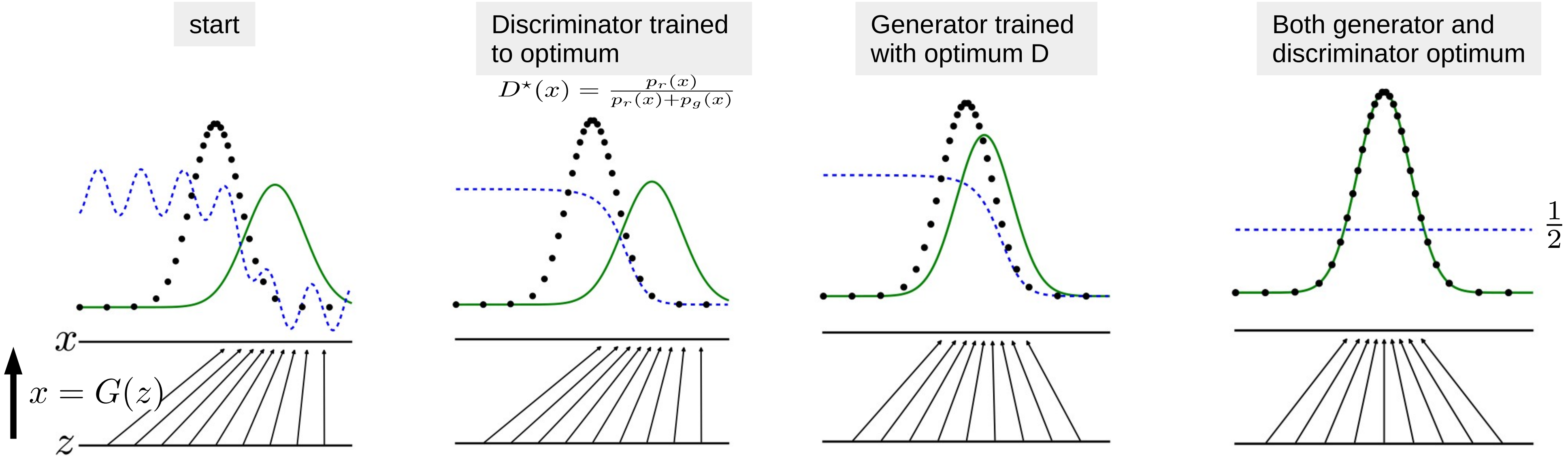
Therefore

$$
L(G, D^*) = 2D_{JS}(p_r \| p_g) - 2 \log 2
$$

- Minimization of L wrt generator G leads to minimization of JS distance between the real and fake data distributions.

Distribution change in training

- Fake data distribution gets closer to real data distribution
- "Distance" between fake and real data distributions is crucial for training.



Green: Fake data distribution p_g
Black: Real data distribution p_r
Blue: Discriminator output D

Problems of (regular) GANs

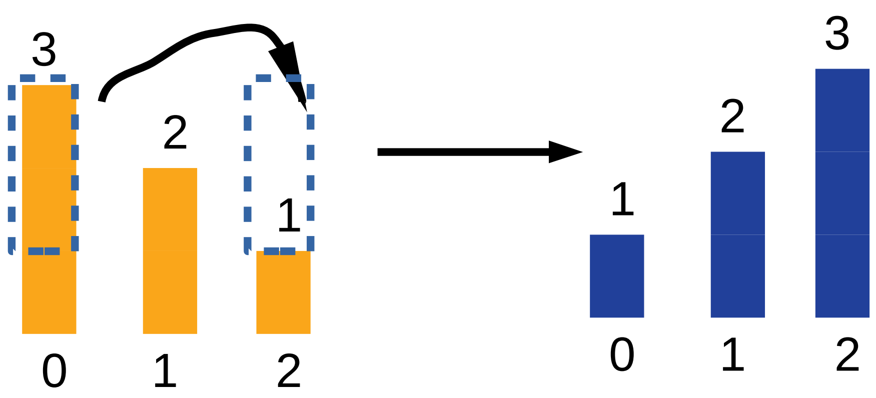
- Hard to achieve (Nash) equilibrium
  - Discriminator and Generator try to achieve contradictory goals.
  - Non-co-operative game
- Vanishing gradient
  - When the Discriminator is perfect (i.e. D=1), loss for the Generator is zero (No gradient to train Generator)
- Mode collapse
  - Generator produces the same output (or a small set of outputs) for any z and the Discriminator traps in a local minimum
- Lack of proper evaluation metric for training progress or performance
  - Evolution of loss does not indicate the reach of equilibrium.

Evaluation of GAN-performance

- Visual inspection
  - Subjective
  - Difficult to evaluate diversity
- Fréchet Inception Distance (FID)
  - Extract features using inception (for both generated and real test data)
  - Assuming features are normally distributed calculate mean and covariance (for generated and real test data)
  - Calculate the Fréchet Distance given by
  - $$d^2((m, C), (m_g, C_g)) = \|m - m_g\|_2^2 + trace(C + C_g - 2(CC_g)^{1/2})$$
  -
  - Where (m, C) and (m_g, C_g) are mean and covariance of the features of real test data and generated data.

Wasserstein GAN

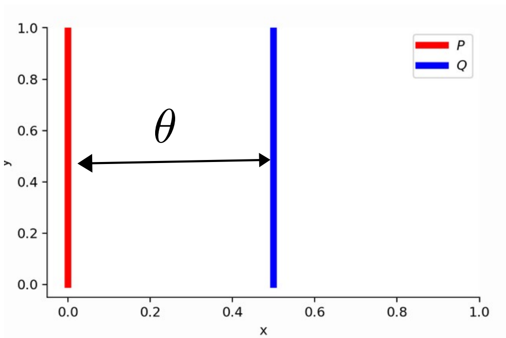
- Training of regular GAN is based on Jensen-Shannon (JS) distance
- JS-distance does not have well-behaved gradients for non overlapping distributions.
  - This is the root cause of difficulty in training the GAN
- Wasserstein (Earth Mover- EM) distance has well behaved gradients even if the distributions are non-overlapping
  - Consider the two probability distributions as two piles of dirt.
  - Transport dirt from one pile and form a pile which has the exact shape of the other pile
  - Define the EM distance as the minimum cost of transport



EM-distance=2 shovels x 2 meters
=4 units

Example: Non-overlapping distributions

- Two non-overlapping uniform distributions



for $\theta \neq 0$:

$$D_{JS}(P,Q) = \frac{1}{2}\left(\sum_{x=0, y \sim U(0,1)} 1 \cdot \log \frac{1}{1/2} + \sum_{x=0, y \sim U(0,1)} 1 \cdot \log \frac{1}{1/2} \right) = \log 2$$

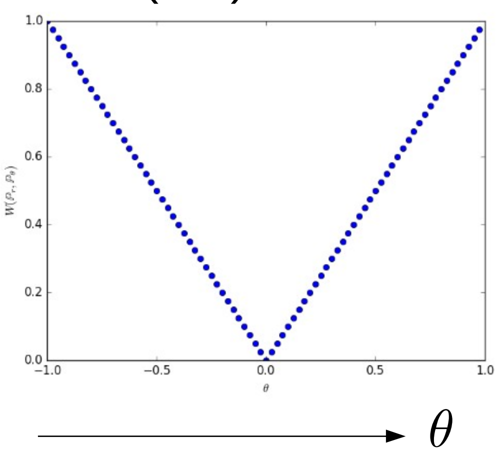
$$W(P,Q) = |\theta|$$

for $\theta = 0$:

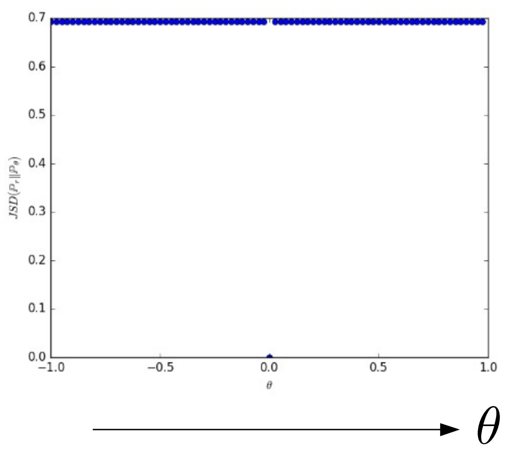
$$D_{JS}(P,Q) = 0$$

$$W(P,Q) = 0$$

W (EM) -distance



JS-distance



Wasserstein GAN (WGAN)

- WGAN Discriminator loss

$$L_D = -(\frac{1}{m}\sum_{i=1}^{m} D_\eta(x_i) - D_\eta(G_\theta(z_i))$$

- WGAN Generator loss

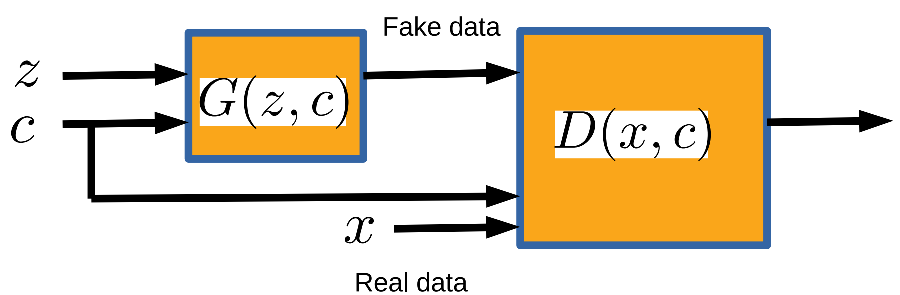
$$L_G = -\frac{1}{m}\sum_{i=1}^{m} D_\eta(G_\theta(z_i))$$

- D_η is no longer a probability, but any real number
- Assumes some conditions known as Lipschitz constraints on D_η
- G now tries to minimize Wasserstein distance between generated distribution and data distribution

Wasserstein GAN, M. Arjovsky, S. Chintala, L. Bottou 2017

Conditional GAN (c-GAN)

- Regular GAN
  - data samples $x_1, x_2, \cdots, x_N$
  - Generator $G(z)$
  - Discriminator $D(x)$
- Conditional GAN
  - Data samples $(x_1, c_1), (x_2, c_2), \cdots, (x_N, c_N)$
  - Generator $G(z, c)$
  - Discriminator $D(x, c)$



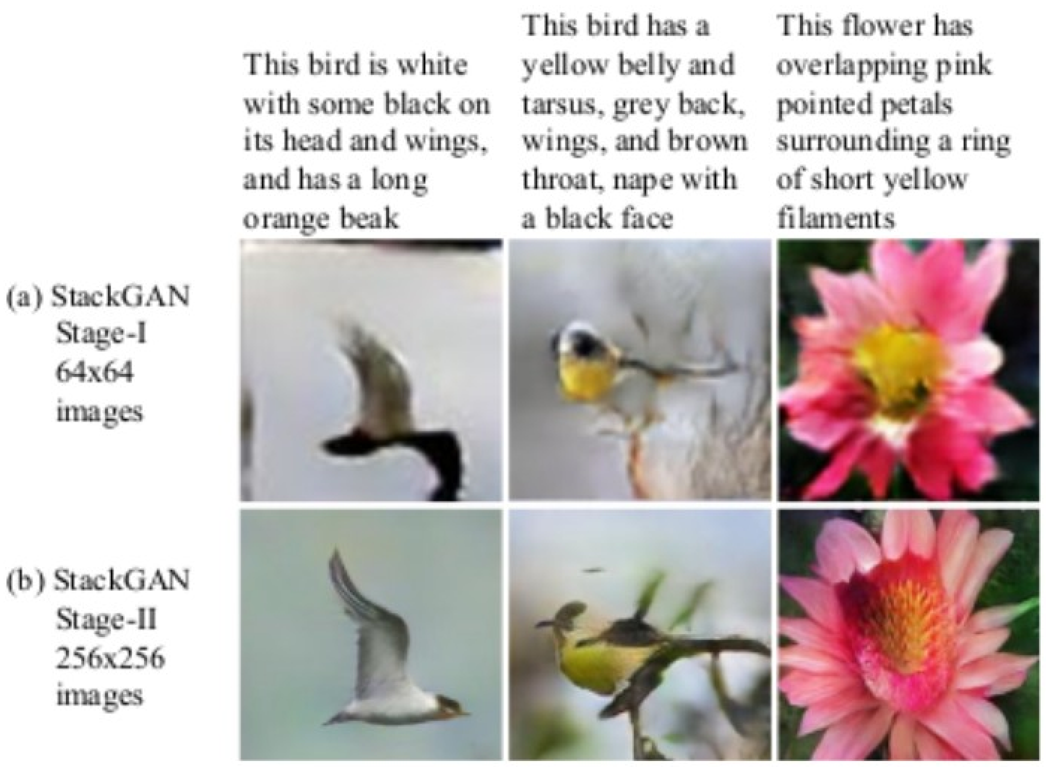
Conditional Generative Adversarial Nets, M. Mirza, S. Osindero, 2014

C-GAN Applications

- Text-to-image translation
- Image inpainting
- Image super-resolution

Text-to-image trasnlation

- (x,c) =(image, corresponding sentence)



StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks H. Zhang et.al 2017

Image Inpainting

- (x,c)=(image, image with missing data)

Conditional Image

Inpainting with L2 loss

Inpainting with CGAN

Context Encoders: Feature Learning by Inpainting, D.Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, A. Efros,2016

Super-resolution

- (x,c)=(image, lower resolution image)

Bicubic

Super Resolution GAN

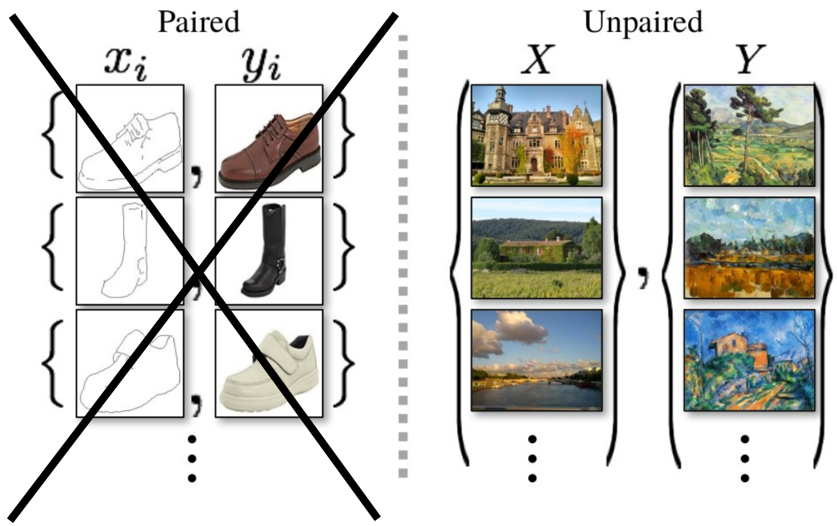
Original

Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network C. Ledig et.al, 2017

CycleGAN

- Considers the problem of image-to-image translation
  - Trained without paired data (i.e. with unpaired data)

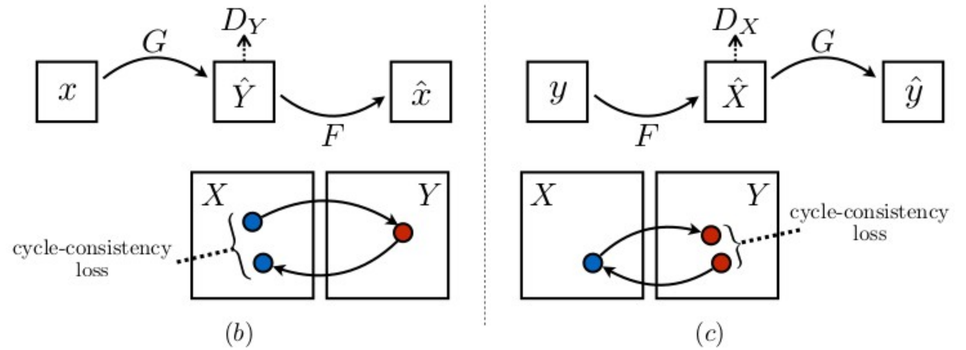
Training



Generation

Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks J. Zhu et.al , 2020

CycleGAN operation

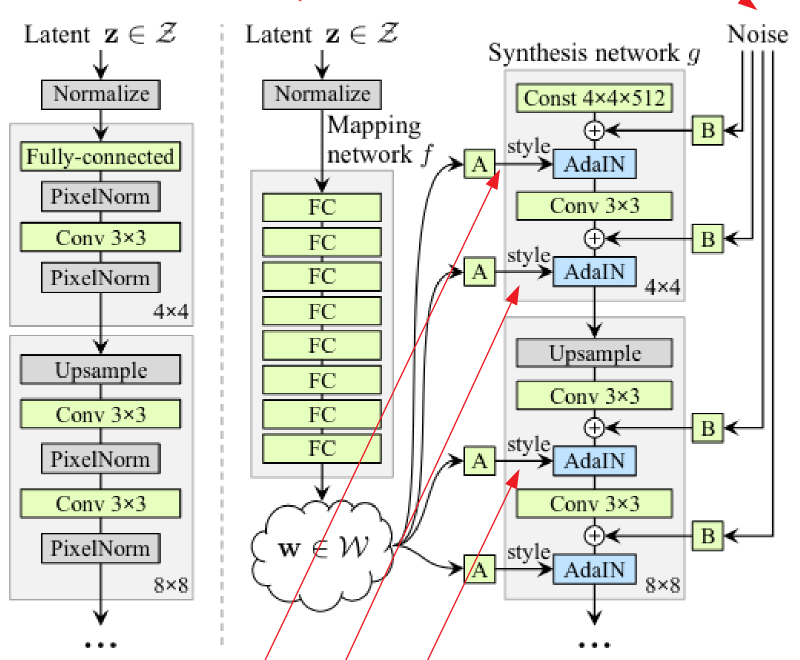


- Two generators G and F and associated discriminators D_Y and D_X
- Learn G such that G(x) should be distributed as Y
- Learn F such that F(y) should be distributed as X
- Additional constraints on cycle-consistency
  - F(G(x))-x $\approx$ 0 and G(F(y))-y $\approx$ 0

StyleGAN

Latent code
(defines main structure)

Random
features



(a) Traditional (b) Style-based generator

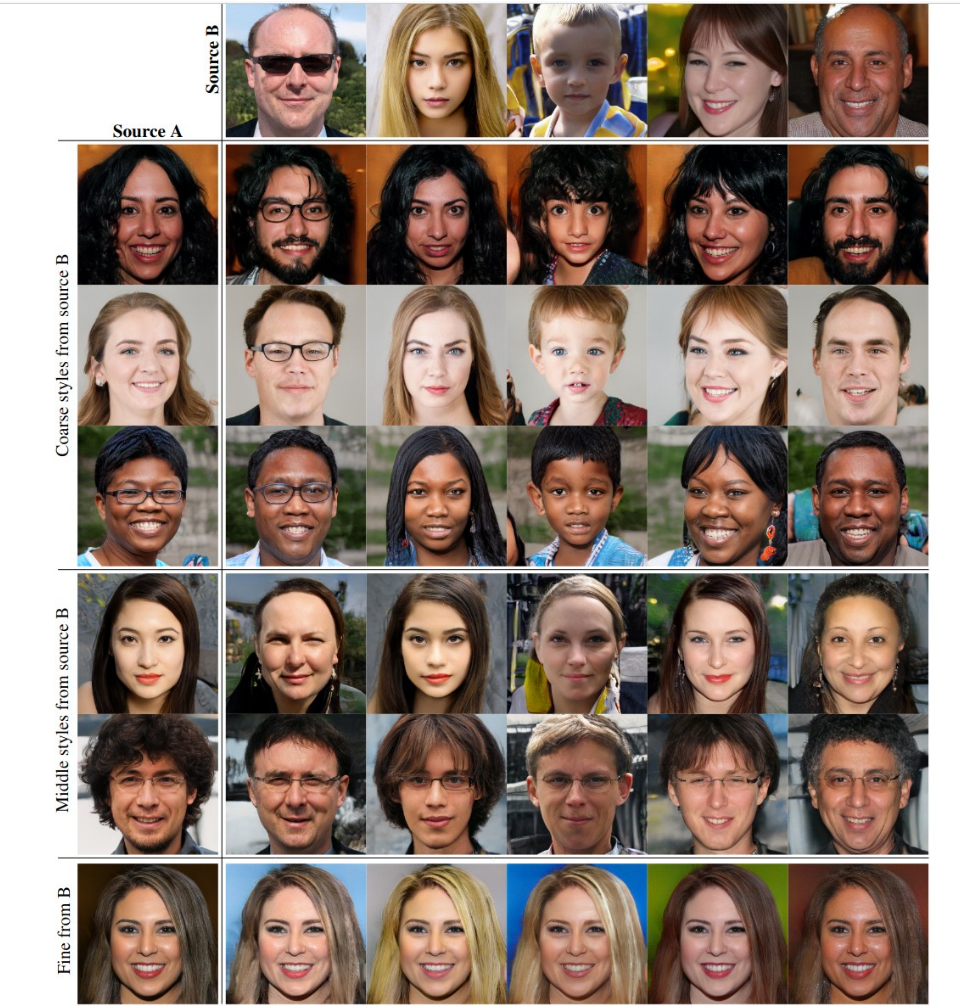
Styles at different levels

- Focuses only on the Generator
- Latent code to define the main structure
- W-space models the “styles” of different scales (eg: human faces)
  - Coarse styles (pose, face shape..)
  - Middle styles(Hair style, eyes)
  - Fine styles(color scheme, micro structure)
- Noise input to define random features (freckles, hair)

A Style-Based Generator Architecture for Generative Adversarial Networks Tero Karras, Samuli Laine, Timo Aila, 2019

StyleGAN (example images)

- Mixing styles from two source images



A Style-Based Generator Architecture for Generative Adversarial Networks Tero Karras, Samuli Laine, Timo Aila, 2019