GENERATIVE ADVERSARIAL NETWORKS

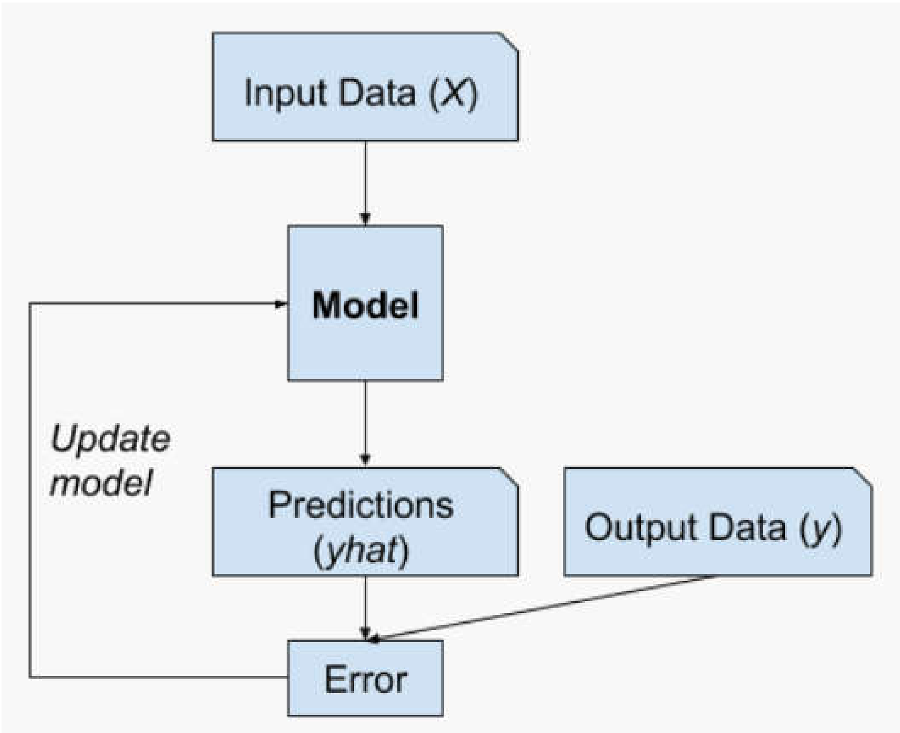
References:

- Generative Adversarial Networks, Goodfellow et al.
- Conditional Generative Adversarial Nets
  Mehdi Mirza et al.
- Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, Zhu et al.
- Generative Adversarial Networks lecture (by Hongyang Zhang)

Generative Models

Supervised vs. Unsupervised Learning

In the predictive methods, the goal is to learn a mapping from inputs x to outputs y, given a labeled set of input-output pairs

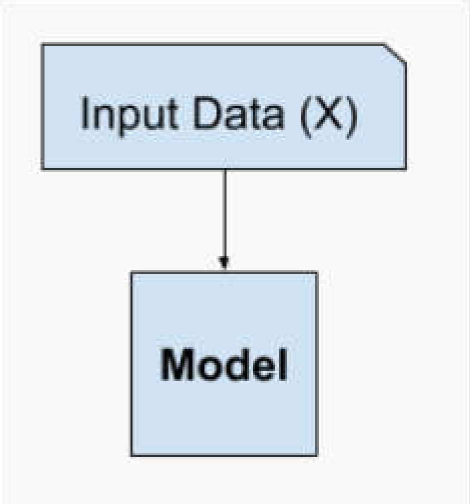


Generative Models

Supervised vs. Unsupervised Learning

unsupervised learning: only given inputs, and the goal is to find "interesting patterns" in the data.

clustering and generative modeling



Example of Unsupervised Learning

Generative Models

- Generative modeling is an unsupervised learning task in machine learning
- automatically learning the patterns in input data
- model used to generate new examples that plausibly drawn from the original dataset.
- In generative modeling, we'd like to train a distribution:
- such a way that the model generate new examples that plausibly could have been drawn from the original dataset.
- good generative model :generate new examples not just plausible, but indistinguishable from real examples

Generative Models

- Naïve Bayes
- GMM
- Deep learning methods can be used as generative models
- A modern example of deep learning generative modeling algorithms
- we'd like to train a network that models a distribution
- For example, LLMs want to model the distribution of natural language
- We want to design a generative model to generate images.

Example of the Progression in the Capabilities of GANs From 2014 to 2017. Taken from The Malicious Use of Artificial Intelligence: Forecasting, Prevention, and Mitigation, 2018.

- Given training data $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \sim p_{data}(\mathbf{x})$, the true data density
- Parameterize $p_{\boldsymbol{\theta}}(\mathbf{x})$, the data density estimated by model
- Estimate $\boldsymbol{\theta}$ by minimizing some "distance" between p_{data} (the unknown data density) and $p_{\boldsymbol{\theta}}$:

$$\min_{\boldsymbol{\theta}} \ \mathsf{dist}(p_{data} \| p_{\boldsymbol{\theta}})$$

- After training, can generate new data $\mathbf{x} \sim p_{\boldsymbol{\theta}}(\mathbf{x})$
- Need a training set from p_{data} and an explicit form of $p_{\boldsymbol{\theta}}$

Push-forward

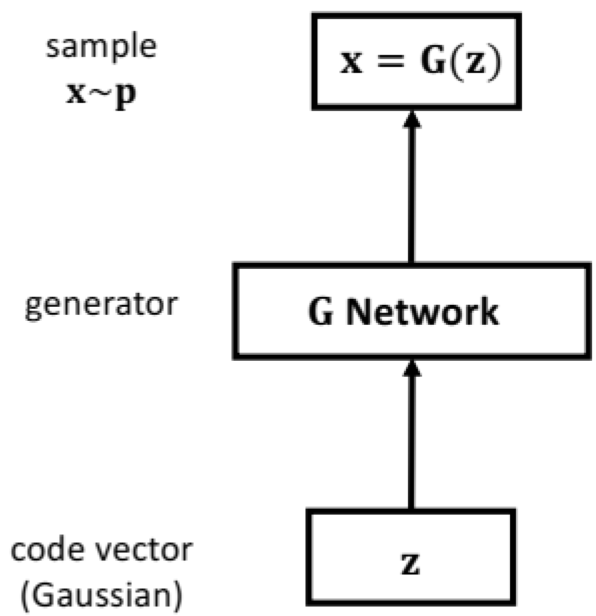
Theorem: Representation through push-forward

Let r be any continuous distribution on $\mathbb{R}^h$. For any distribution p on $\mathbb{R}^d$, there exist push-forward maps $\mathbf{G} : \mathbb{R}^h \to \mathbb{R}^d$ such that

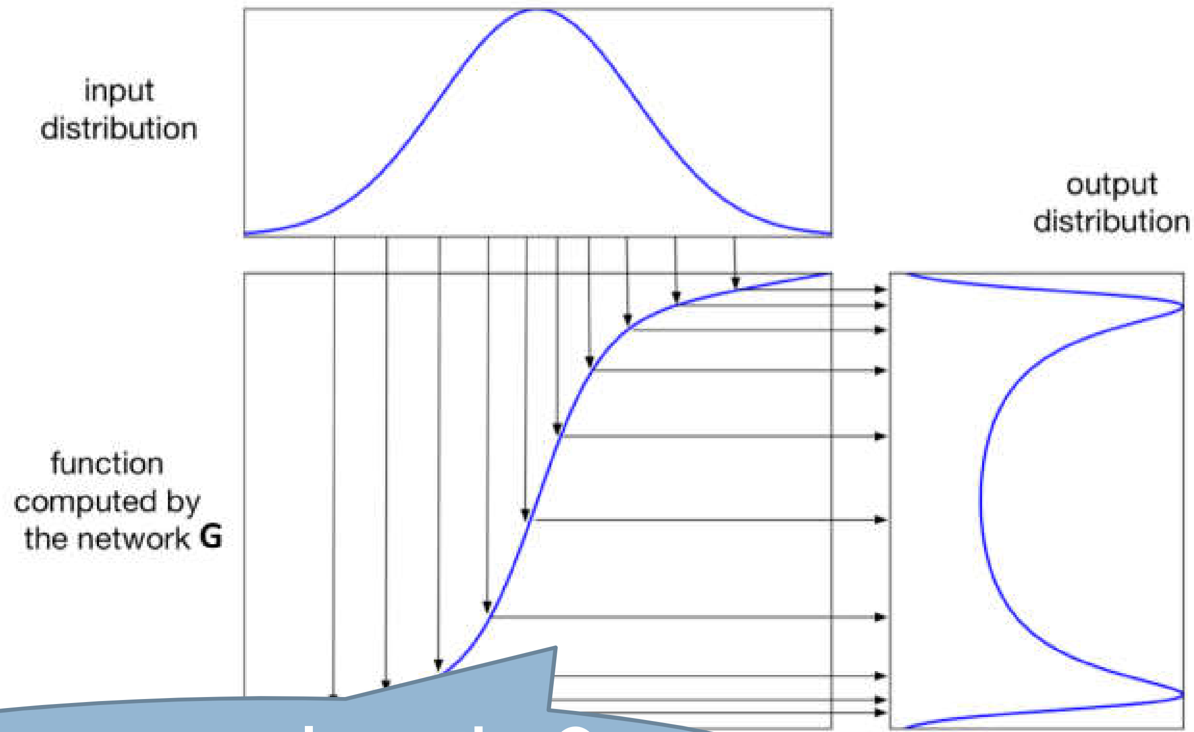
$$\mathbf{z} \sim r \implies \mathbf{G}(\mathbf{z}) \sim p$$

simply take *r* to be standard Gaussian noise

- Start by sampling the code vector z from a simple distribution (e.g., Gaussian)
- GAN computes a differentiable function **G** mapping **z** to an **x** in data space
- **G** maps one distribution to another

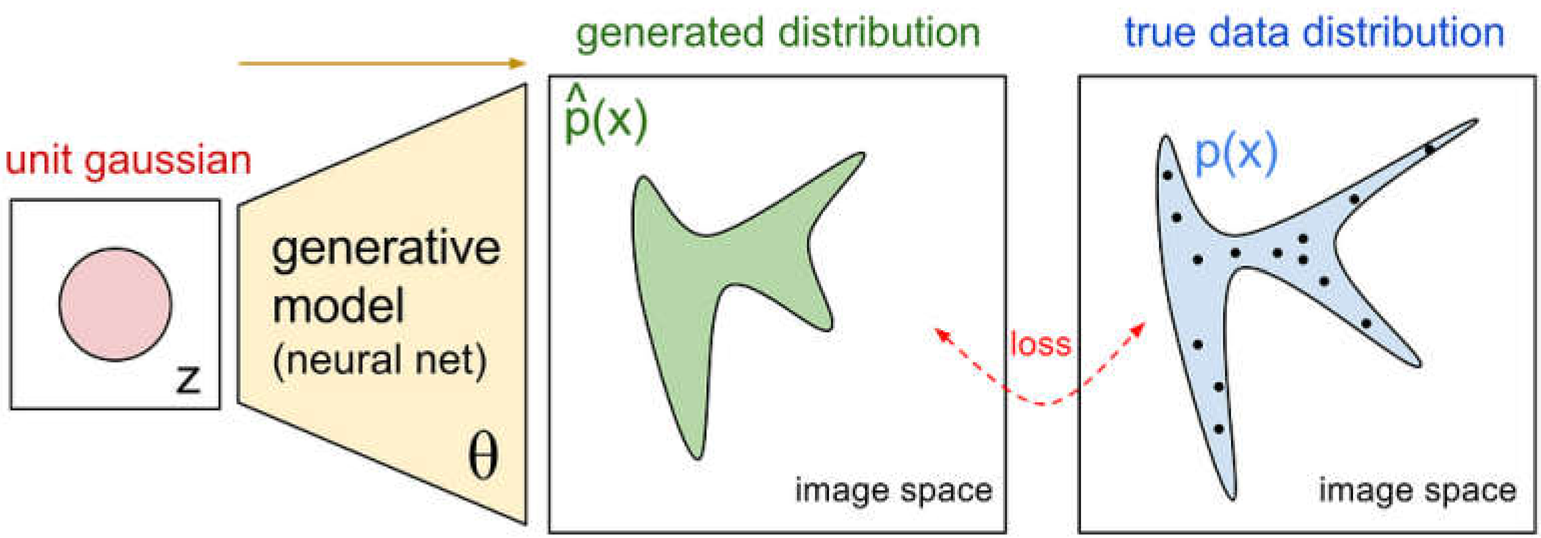


example



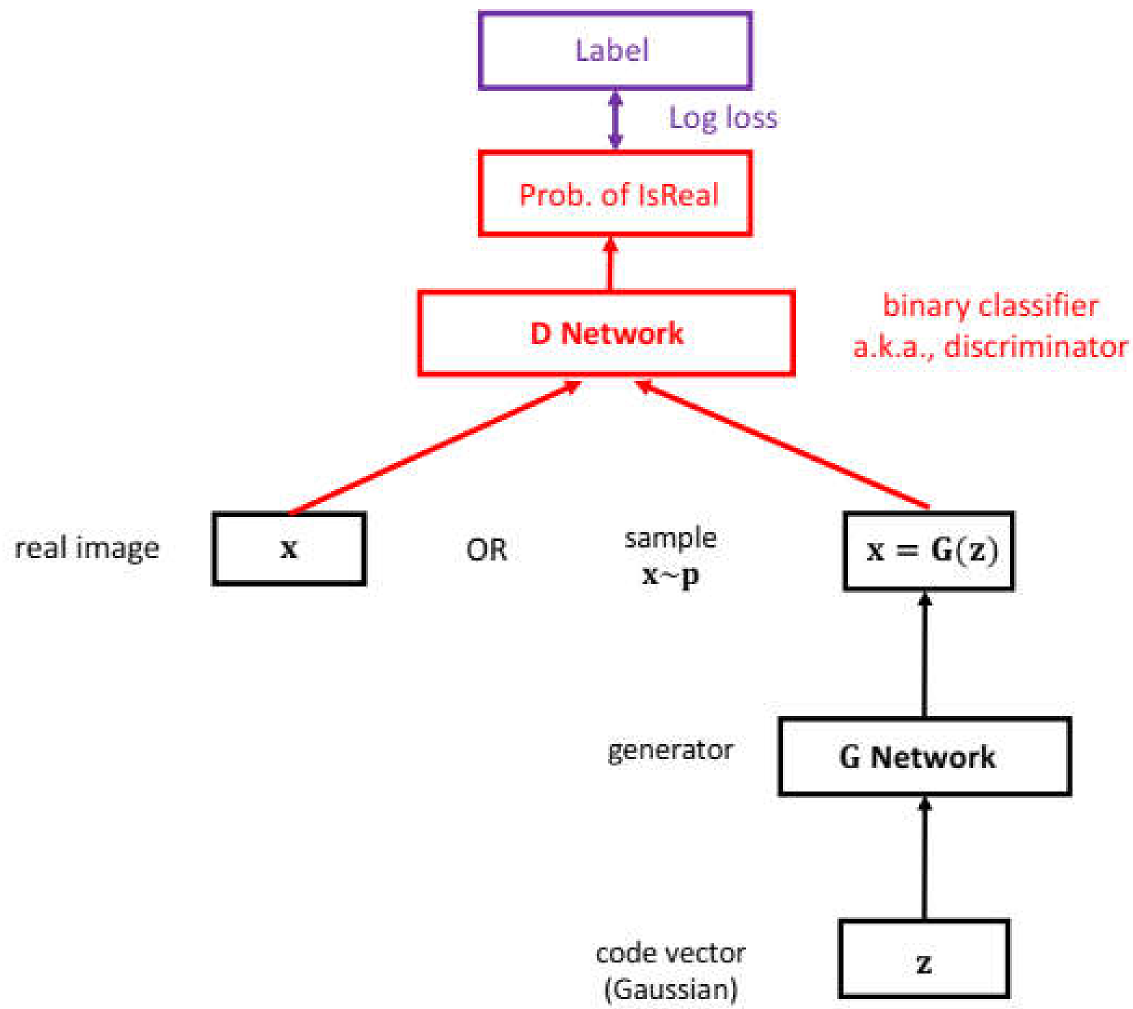
how we can learn the G network

Learn Generator



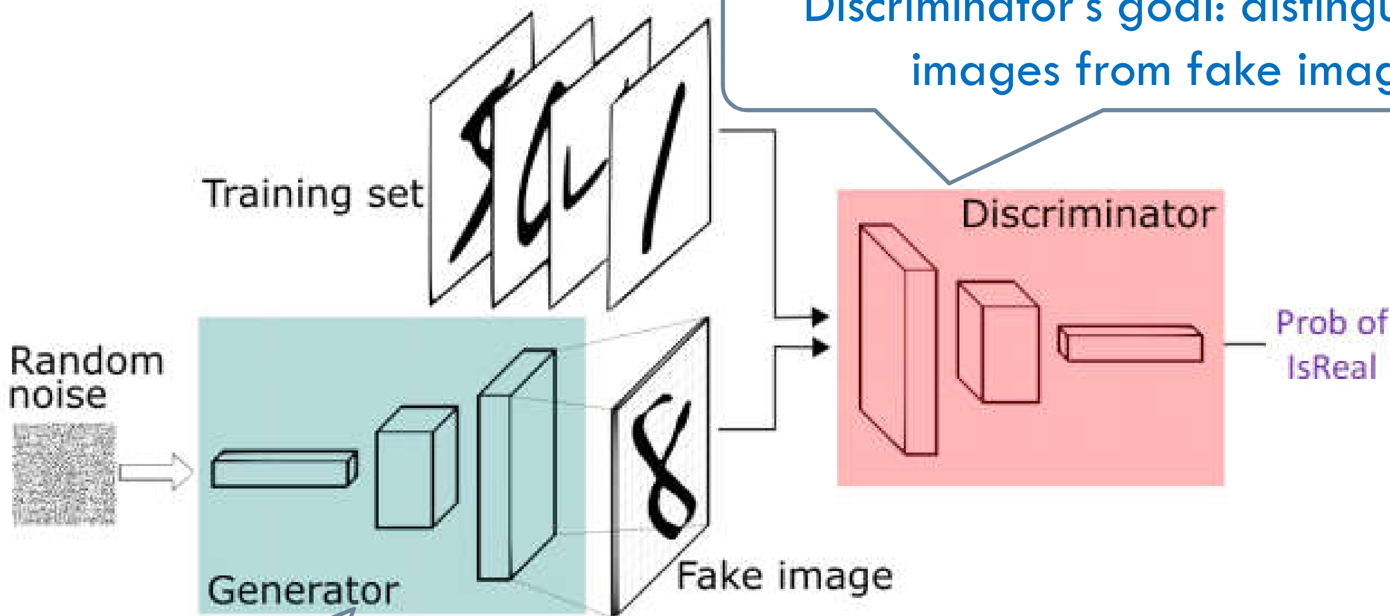
- How can we define the loss to distinguish two distributions?
  - Use a discriminator

Generative Adversarial Networks structure



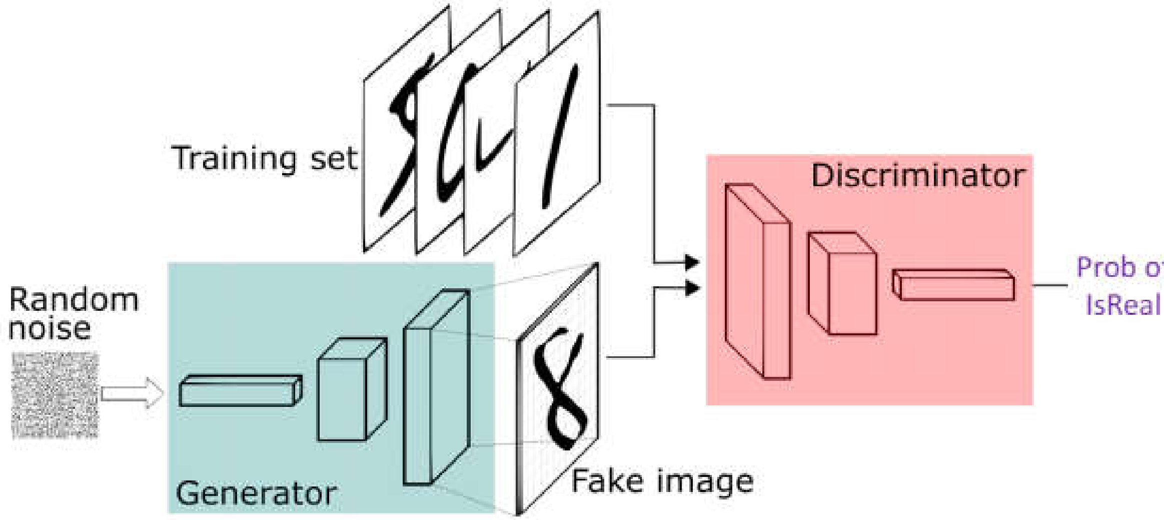
Generative Adversarial Networks

Discriminator’s goal: distinguish real images from fake images



Generator’s goal: generate images that look like real one to confuse discriminator

Discriminator's Loss



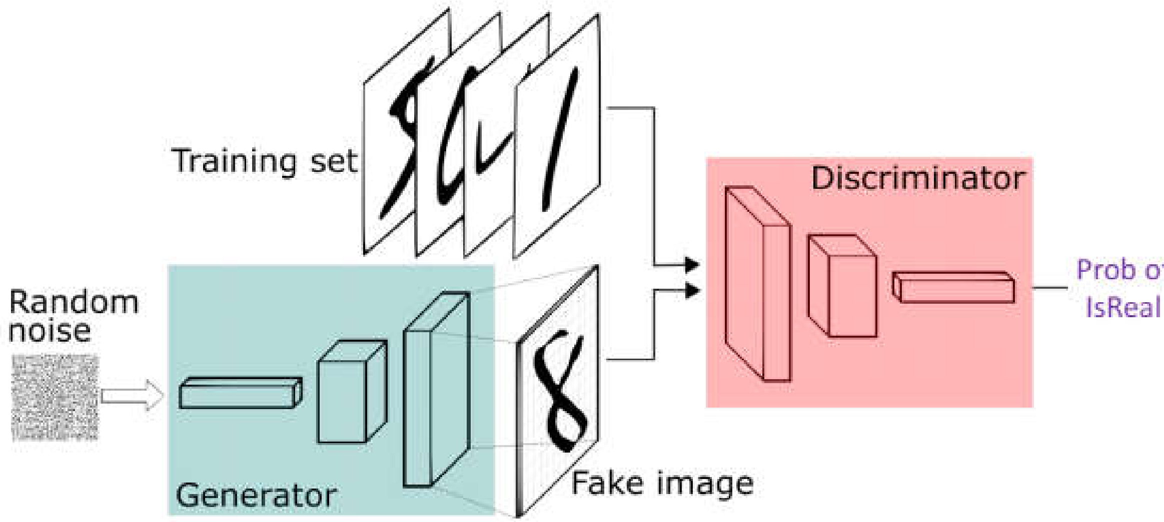
For a fixed generator *G*

- If $\mathbf{x}$ is real, minimize $-\log D(\mathbf{x})$; if $\mathbf{x}$ is fake, minimize $-\log(1 - D(\mathbf{x}))$

- Assume that $\mathbf{x}$ being from real/fake distribution is with equal chance:

$$\min_{D} \underbrace{-\frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_{data}}[\log D(\mathbf{x})]}_{\mathbf{x} \text{ is real with one half prob}} \underbrace{-\frac{1}{2}\mathbb{E}_{\mathbf{z}\sim\mathcal{N}(0,\mathbf{I})}[\log(1 - D(G(\mathbf{z})))]}_{\mathbf{x} \text{ is fake with another half prob}}$$

Generator's Loss

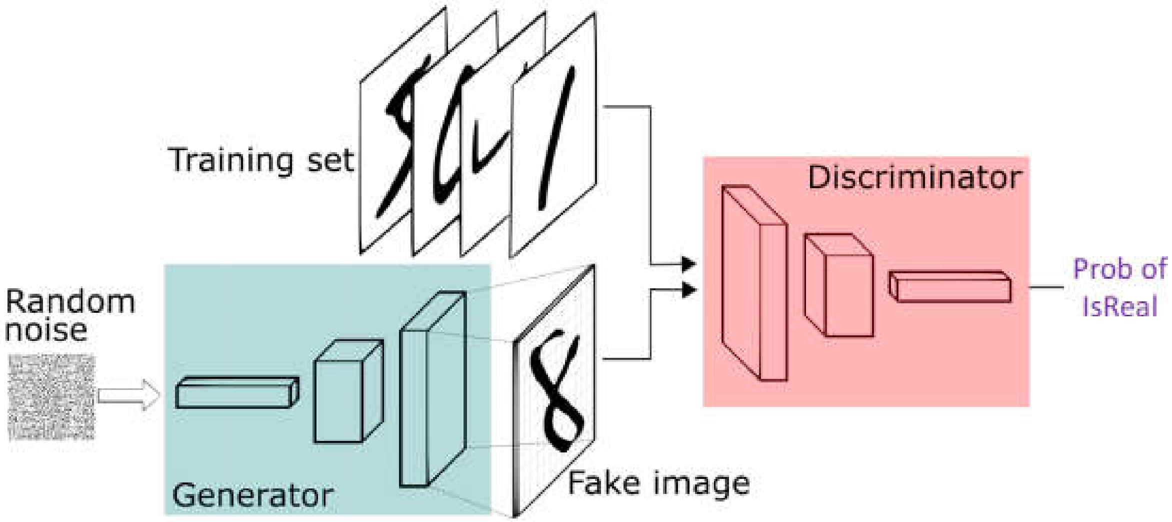


For a fixed Discriminator D

if $\mathbf{x}$ is fake, maximize $-\log(1 - D(\mathbf{x}))$

$$\max_G -\frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_{data}}[\log D(\mathbf{x})] - \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim\mathcal{N}(0,\mathbf{I})}[\log(1 - D(G(\mathbf{z})))]$$

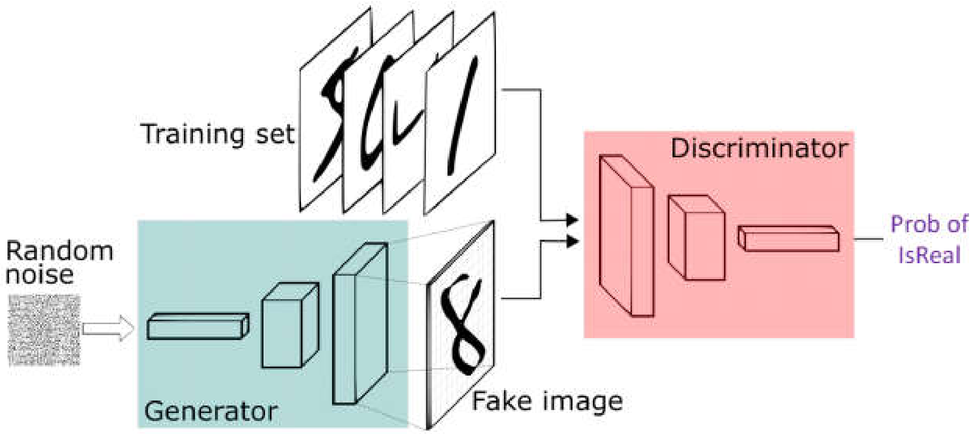
GAN



$$\max_{G} \min_{D} -\frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_{data}}[\log D(\mathbf{x})] - \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim\mathcal{N}(0,\mathbf{I})}[\log(1 - D(G(\mathbf{z})))]$$

$$\min_{G} \max_{D} \hat{\mathbb{E}}_{\mathbf{x}\sim p_{data}}[\log D(\mathbf{x})] + \hat{\mathbb{E}}_{\mathbf{z}\sim\mathcal{N}(0,\mathbf{I})}[\log(1 - D(G(\mathbf{z})))]$$

How to solve



$$\underset{G}{\min}\ \underset{D}{\max}\ V(G, D) := \mathbb{E}_{\mathbf{x} \sim p_{data}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim \mathcal{N}(0,\mathbf{I})}[\log(1 - D(G(\mathbf{z})))]$$

- Solved by alternative minimization-maximization:
  - G step: Fix D and update G by one-step gradient descent
  - D step: Fix G and update D by one-step gradient ascent
  - Repeat until the algorithm reaches an approximate equilibrium

Why does GAN work?

Let $p_g(\mathbf{x})$ be the density of $\mathbf{x}$ estimated by the generator G. For G fixed, the optimal discriminator D is $D^*_G(\mathbf{x}) = \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x})+p_g(\mathbf{x})}$.

$$V(G,D) := \mathbb{E}_{\mathbf{x}\sim p_{data}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(0,\mathbf{I})}[\log(1-D(G(\mathbf{z})))]$$

$$= \int_{\mathbf{x}} p_{data}(\mathbf{x}) \log D(\mathbf{x}) d\mathbf{x} + \int_{\mathbf{z}} p_{\mathbf{z}}(\mathbf{z}) \log(1-D(G(\mathbf{z})))d\mathbf{z}$$

$$= \int_{\mathbf{x}} \underbrace{p_{data}(\mathbf{x}) \log(D(\mathbf{x})) + p_g(\mathbf{x}) \log(1-D(\mathbf{x}))}_{f(D(\mathbf{x}))} d\mathbf{x}.$$

$$D^*_G(\mathbf{x}) := \mathrm{argmax}_{D(\mathbf{x})} f(D(\mathbf{x})) = \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x})+p_g(\mathbf{x})}.$$

Why does GAN work?

Theorem 1: solution of G^*

The global minimum of $\min_G \max_D V(G, D)$ is achieved if and only if $p_g = p_{data}$. The optimal objective value is $-\log 4$.

Therefore, the generator is able to learn the data distribution p_{data} exactly if we can solve $\min_G \max_D V(G, D)$ exactly.

Why does GAN work?

$$V(G, D_G^*) = \mathbb{E}_{\mathbf{x}\sim p_{data}}[\log D_G^*(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim\mathcal{N}(0,\mathbf{I})}[\log(1 - D_G^*(G(\mathbf{z})))]$$

$$= \mathbb{E}_{\mathbf{x}\sim p_{data}}[\log D_G^*(\mathbf{x})] + \mathbb{E}_{\mathbf{x}\sim p_g}[\log(1 - D_G^*(\mathbf{x}))]$$

$$= \mathbb{E}_{\mathbf{x}\sim p_{data}}\left[\log \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x}) + p_g(\mathbf{x})}\right] + \mathbb{E}_{\mathbf{x}\sim p_g}\left[\log \frac{p_g(\mathbf{x})}{p_{data}(\mathbf{x}) + p_g(\mathbf{x})}\right].$$

Recall that $KL(P||Q) = \mathbb{E}_{\mathbf{x}\sim P}\left[\log \frac{p(\mathbf{x})}{q(\mathbf{x})}\right]$

$$V(G, D_G^*) = -\log 4 + KL\left(p_{data}\middle|\middle|\frac{p_{data} + p_g}{2}\right) + KL\left(p_g\middle|\middle|\frac{p_{data} + p_g}{2}\right)$$

Why does GAN work?

Jensen–Shannon divergence

$$\mathrm{JSD}(P \parallel Q) = \frac{1}{2}D(P \parallel M) + \frac{1}{2}D(Q \parallel M),$$

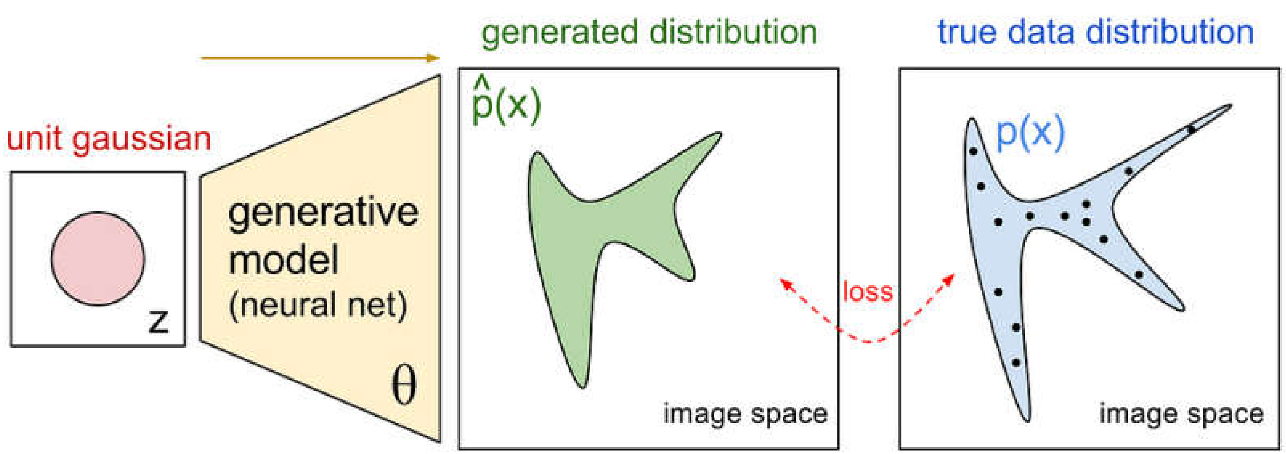
where $M = \frac{1}{2}(P + Q)$ is a mixture distribution of P and Q.

$$V(G, D_G^*) = -\log 4 + KL\left(p_{data} \middle\| \frac{p_{data} + p_g}{2}\right) + KL\left(p_g \middle\| \frac{p_{data} + p_g}{2}\right)$$

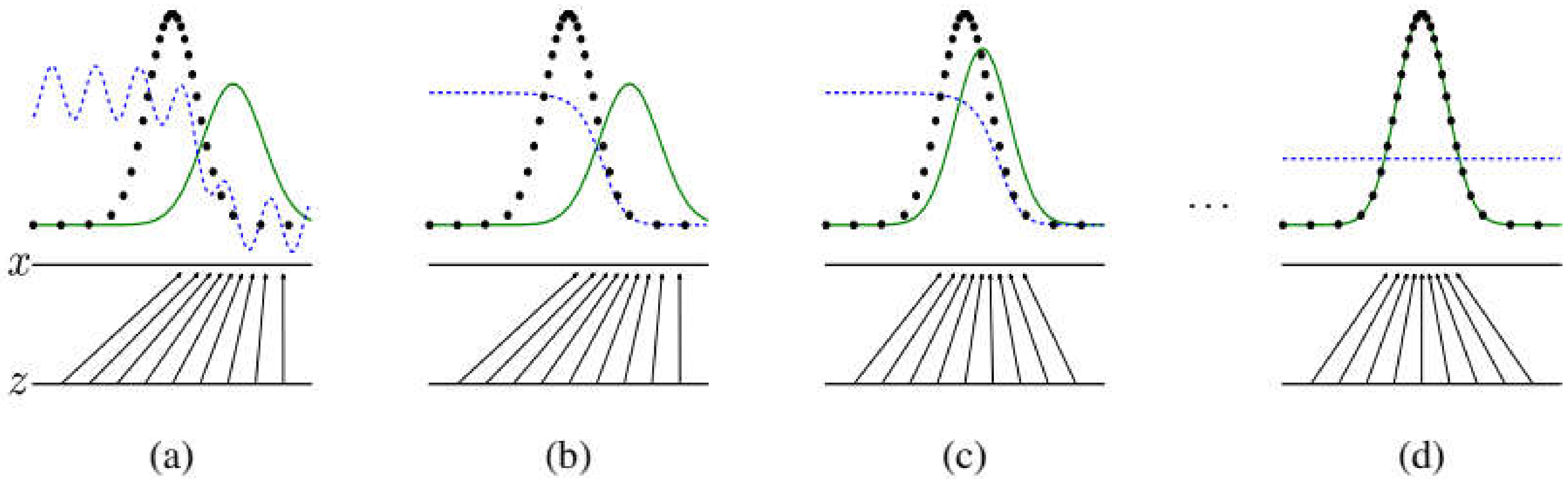
$$= -\log 4 + 2 \cdot JSD(p_{data} || p_g) \geq -\log 4,$$

$$p_{data} = p_g.$$

Why does GAN work?



- Thus GAN is minimizing the Jensen–Shannon divergence between generated and real data distributions.



x
z
(a)
(b)
(c)
(d)

Conditional GAN

- Generative adversarial nets can be extended to a conditional model
- generator and discriminator are conditioned on some extra information y

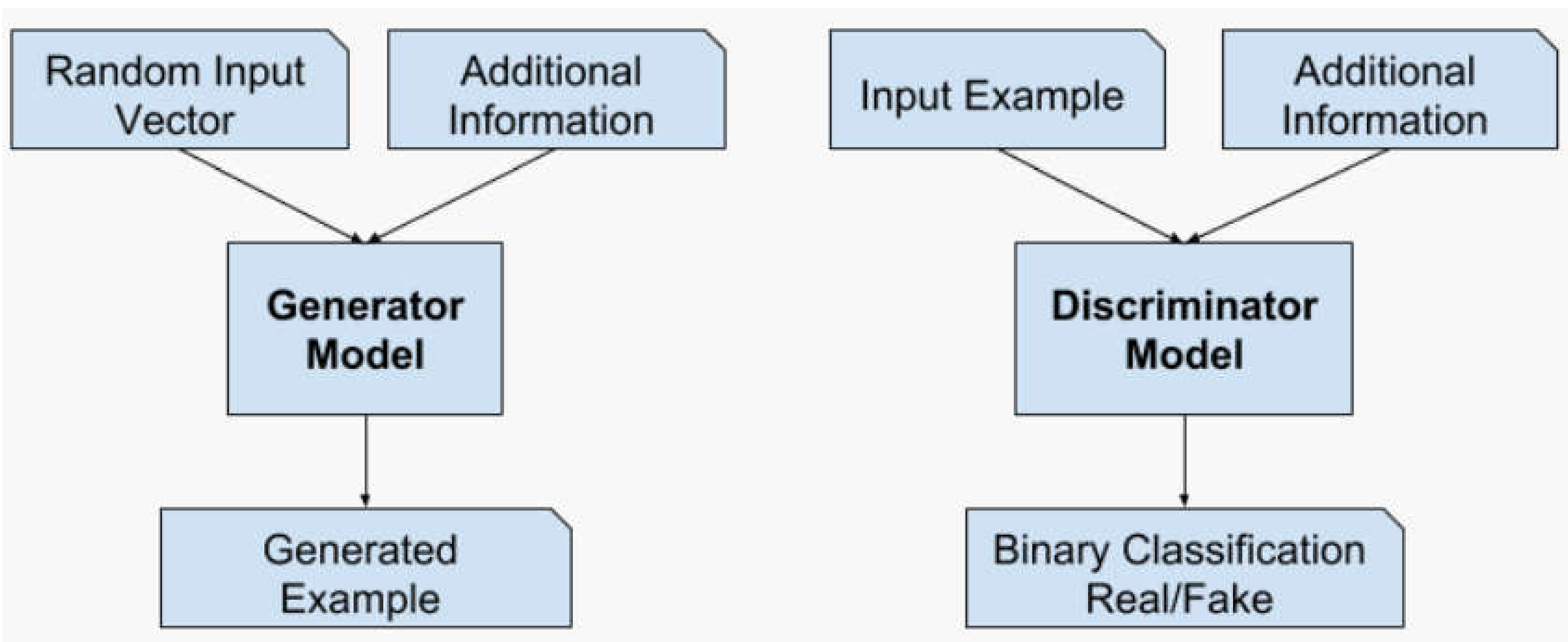
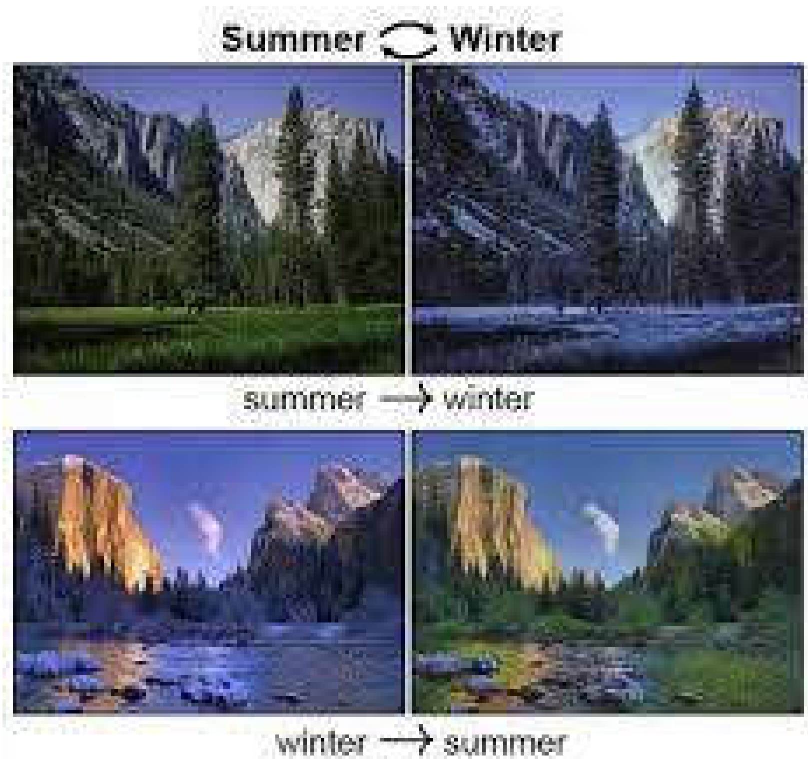


Image to Image Translation

- ❖ generating a new version of a given image with a specific modification, such as translating a summer landscape to winter.
- ❖ Training a model, a large dataset of paired examples



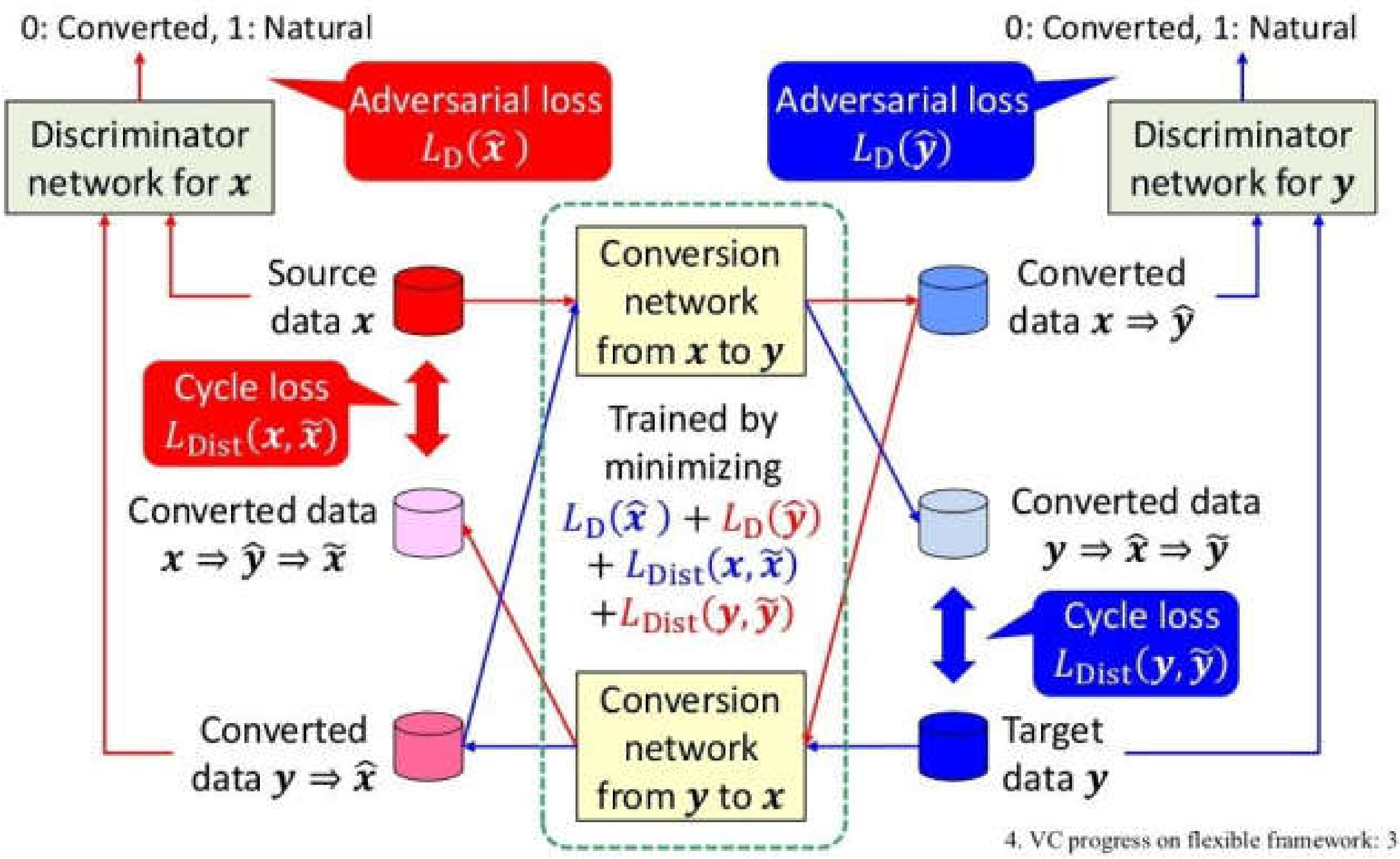
CycleGAN: Image to Image Translation

- CycleGAN : automatic training of image-to-image translation models without paired examples
- Unsupervised
- A collection of images from the source and target domain
- not need to be related
- Extension of GAN
- Two generators, Two discriminators
- translation should be "cycle consistent"

Cycle Consistency

if we translate a sentence from English to Persian, and then translate it back from Persian to English, we should arrive back at the original sentence

CycleGAN Model Architecture



CycleGAN Model Architecture

- translating images from summer to winter and winter to summer.
- two databases of images, unpaired
- different locations at different times;

- **Database 1**: summer
- **Database 2:** winter

- two GANs
- each GAN: a discriminator and a generator

CycleGAN Model Architecture

- **Generator Model 1:**
  - **Input**: Images of summer **Output** Images of winter
- **Discriminator Model 1**:
  - **Input**: Images of winter, output from Generator Model 1.
  - **Output**: Probability of image is from database2

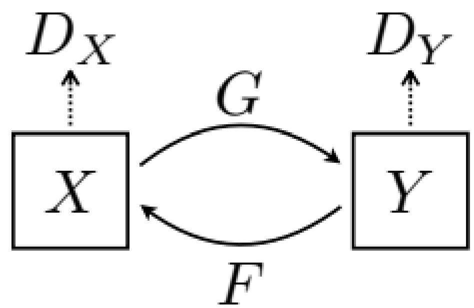
- **Generator Model 2**:
  - **Input**: Images of winter **Output**: Images of summer
- **Discriminator Model 2**:
  - **Input**: Images of summer(database 1), output from Generator Model 2.
  - **Output**: Probability of image is from database 1.

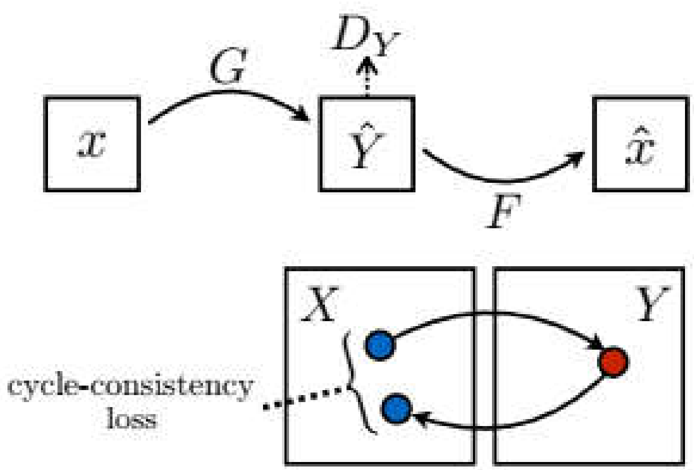
CycleGAN Model Architecture

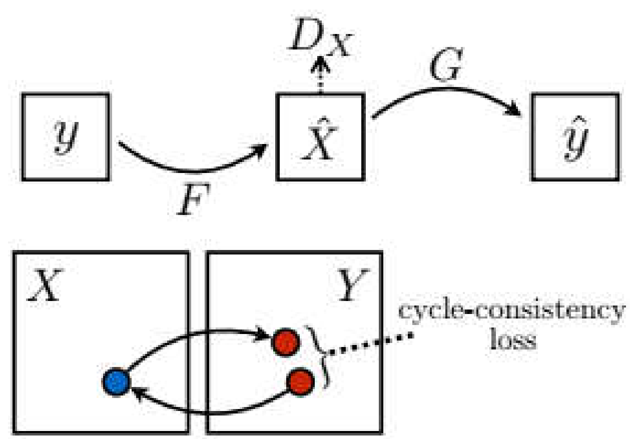
- **Forward Cycle Consistency Loss:**
  - Input summer images (DB1) to GAN 1
  - Output winter images from GAN 1
  - Input winter images from GAN 1 to GAN 2
  - Output summer images from GAN 2
  - Compare summer images (DB1) to summer images from GAN 2
- **Backward Cycle Consistency Loss:**
  - Input winter images (DB 2) to GAN 2
  - Output summer images from GAN 2
  - Input summer images from GAN 2 to GAN 1
  - Output winter images from GAN 1
  - Compare winter images (DB 2) to winter images from GAN 1

CycleGAN Model Architecture

$G : X \rightarrow Y$ and $F : Y \rightarrow X$







$x \rightarrow G(x) \rightarrow F(G(x)) \approx x$

$y \rightarrow F(y) \rightarrow G(F(y)) \approx y$

CycleGAN Model Loss

$$G : X \to Y$$

$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))]$$

$$\min_G \max_{D_Y} \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y)$$

$$F : Y \to X$$

$$\min_F \max_{D_X} \mathcal{L}_{\text{GAN}}(F, D_X, Y, X)$$

CycleGAN Model Loss

$$\begin{aligned}\mathcal{L}_{\text{cyc}}(G, F) &= \mathbb{E}_{x\sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] \\ &+ \mathbb{E}_{y\sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].\end{aligned}$$

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) =& \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\ &+ \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ &+ \lambda\mathcal{L}_{\text{cyc}}(G, F),\end{aligned}$$

$$G^*, F^* = \arg\min_{G,F}\max_{D_x, D_Y} \mathcal{L}(G, F, D_X, D_Y)$$