Who am I?

- Ali Mahdavi-Amiri
- Assistant prof at CS (GrUVi)
- Director of MPCS

GrUVi

Largest vision and graphics lab in Canada

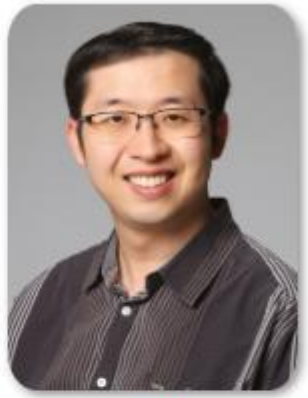
Faculty


Yağız Aksoy


Angel Chang


Eugene Fiume


Yasutaka Furukawa


Ke Li


Ali Mahdavi-Amiri


Jason Peng


Manolis Savva


Andrea Tagliasacchi


KangKang Yin


Richard (Hao) Zhang


SFU
SIMON FRASER
UNIVERSITY
Master's in Professional Computer Science
Master of Science in Big Data
Master of Visual Computing
Master of Cybersecurity

Master of Science in Big Data

The Master of Science in Big Data develops data architects who apply a deep knowledge of computer science to create new tools that find value in the vast amounts of information generated today.

#1 comprehensive university in Canada

Maclean's University Ranking (2023)

1st big data program of its kind in Canada

#2 in Canada for database research #29 in the world

CSrankings.org (2012-2022)

Our Curriculum:

- Machine Learning
- Distributed and Cloud Systems
- Big Data programming models
- Cloud computing
- Data Mining
- Visualization & Interactive data exploration

Career Prospects:

- Machine Learning Engineer
- Big Data Engineer
- Data Scientist
- AI Engineer
- Software Engineer
- Data Solutions Architect

Master of Visual Computing

The Master of Visual Computing trains visual data scientists and engineers who apply specialized knowledge in computer science to develop cutting-edge tools, stimulate product innovation, and explore new technology fronts in all commercial, engineering, and creative professions.

1st
and only visual computing program of its kind in Canada

#1
in Canada for computer vision publications

#2
in Canada for computer graphics publications; #10 in the world

CSrankings.org (2012-2022)

Core courses:

- Computer Vision
- Machine Learning
- Distributed and Cloud Systems
- Human-computer interfaces
- Multimedia System
- Computational Design

Career Prospects:

- Computer Vision Engineer
- Machine Learning Engineer
- Computer Graphics Engineer
- Deep Learning Engineer
- Game Developer

Master of Cybersecurity

The Master of Cybersecurity program engages students in developing specialized knowledge and practical skills in the area of cybersecurity. The program trains students to build and maintain safe systems and infrastructure that can withstand digital attacks.

#1 comprehensive university in Canada

Maclean's University Ranking (2023)

#3 in Canada for computer security

100 % employment rate after graduation

CSrankings.org (2012-2022)

Core courses:

- Applied Cryptography
- Distributed and Cloud Systems
- Secure Software Design
- Cloud and Network Security
- Digital Forensics

Career Prospects:

- Security Engineer/Analyst
- Chief Information Security Officer
- Cyber Forensic Analyst
- Security Compliance Analyst
- Cryptologist

Program Structure

hands-on lab courses

paid co-op internship

16-20 month duration

4 months: Lab I + Core course I → 4 months: Lab II + Core course II → 4-8 months: Co-op → 4 months: Elective courses → Graduation

Admission Requirements

- Ability to program in Java, Python, and C++ (or Matlab for graphics/vision courses).
- Bachelor's degree or equivalent in computer science or a related fields with a CGPA of 3.00/4.33 (B).

Learn more information about the Programs

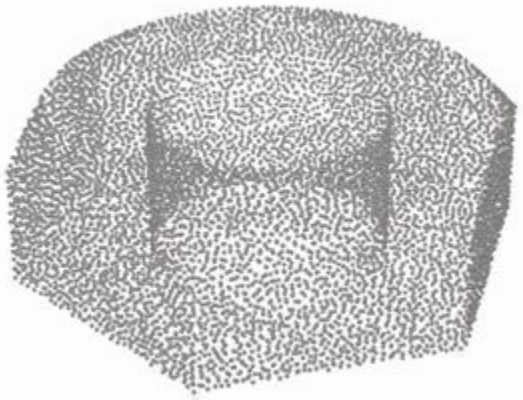
The program begins in September of each year.

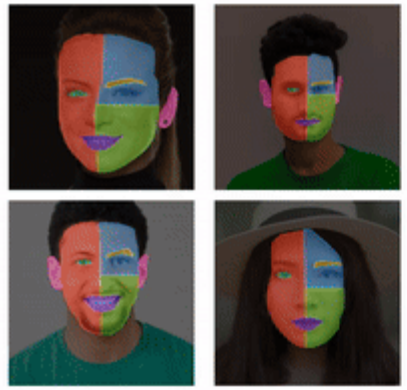
Applications open on October 1, 2024, and close on January 18, 2025!

SCAN ME

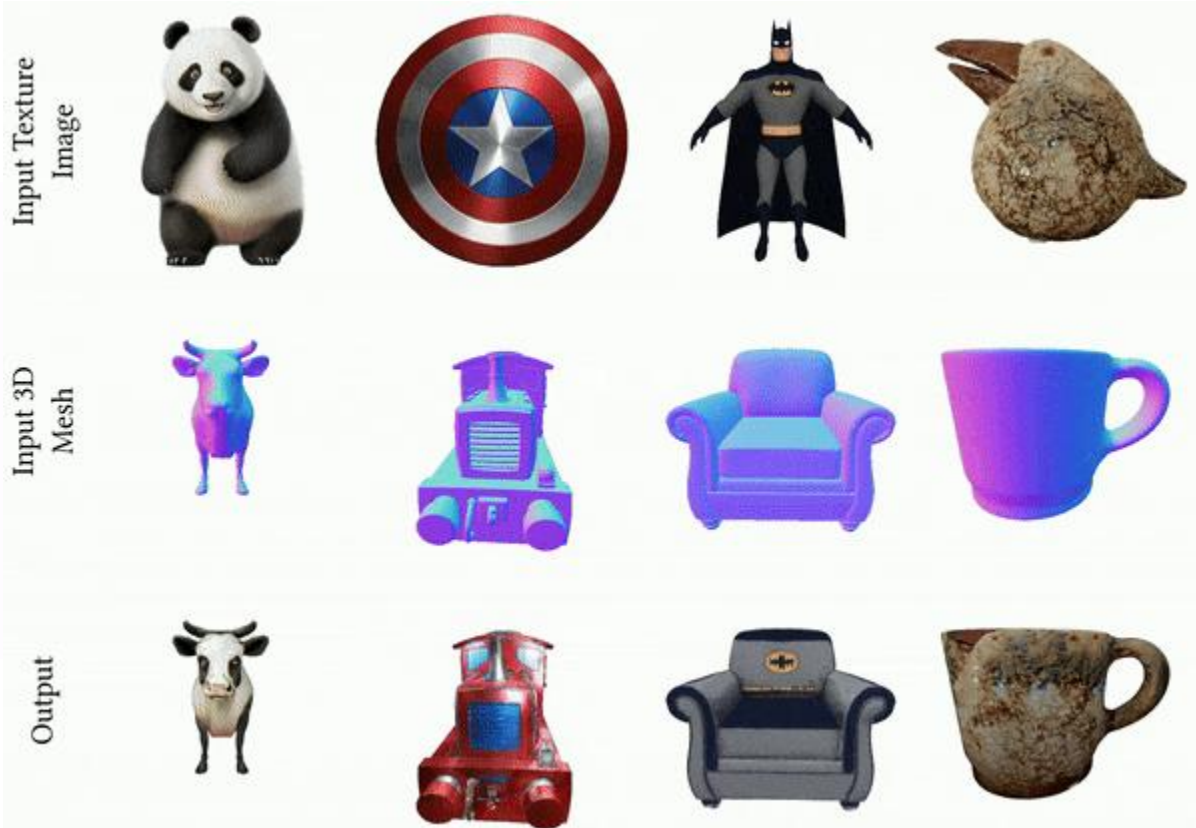
My research

- Computer Graphics
- Geometric Modelling





Reference Images



Generative models: a quick overview

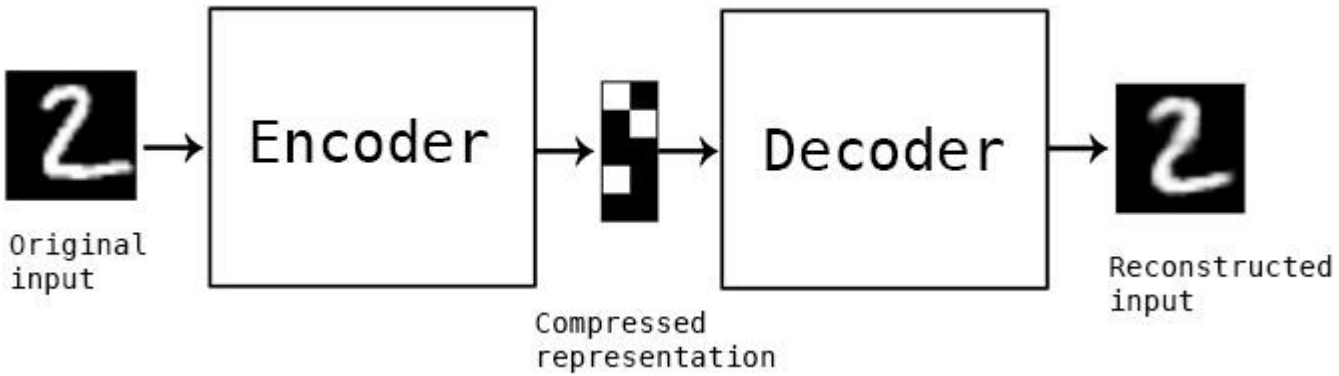
Ali Mahdavi Amiri

Overview

- Roughly 3× 45min lecture
- VAE
- GAN
- Diffusion
- No proof (you can check my course’s slides for proofs)

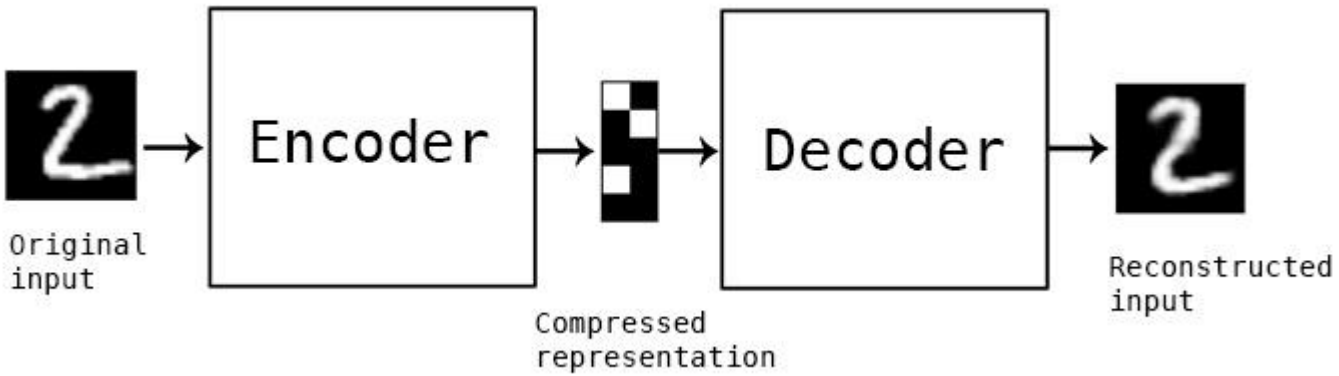
Autoencoders

- Autoencoding is a data specific compression technique.



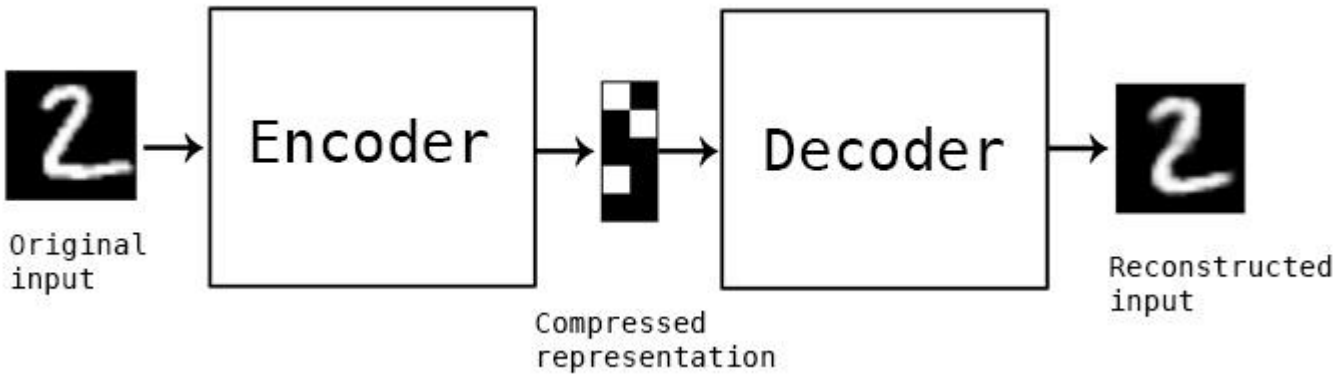
Autoencoders

- Autoencoding is a data specific compression technique.
  - Learning is not hand crafted.



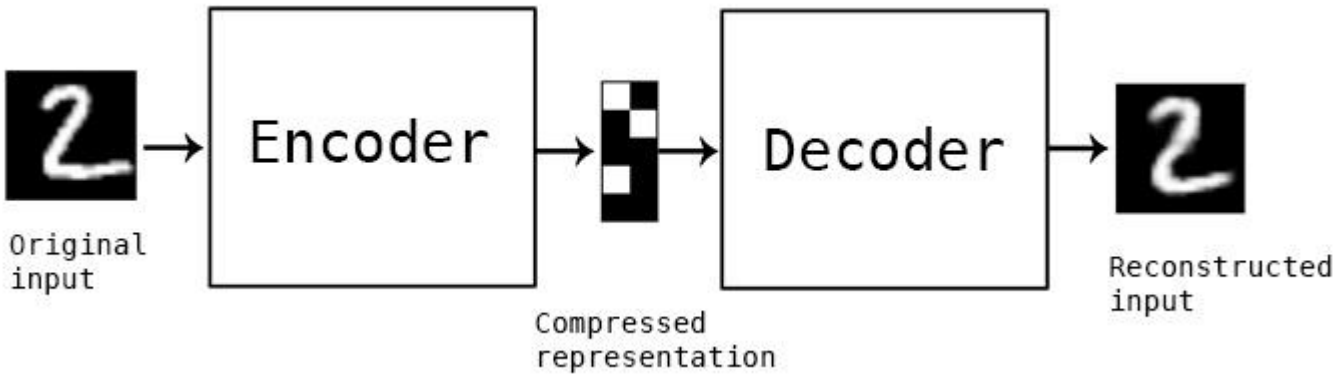
Autoencoders

- Encoder: Learns to map data to a compressed domain which is also called latent space.



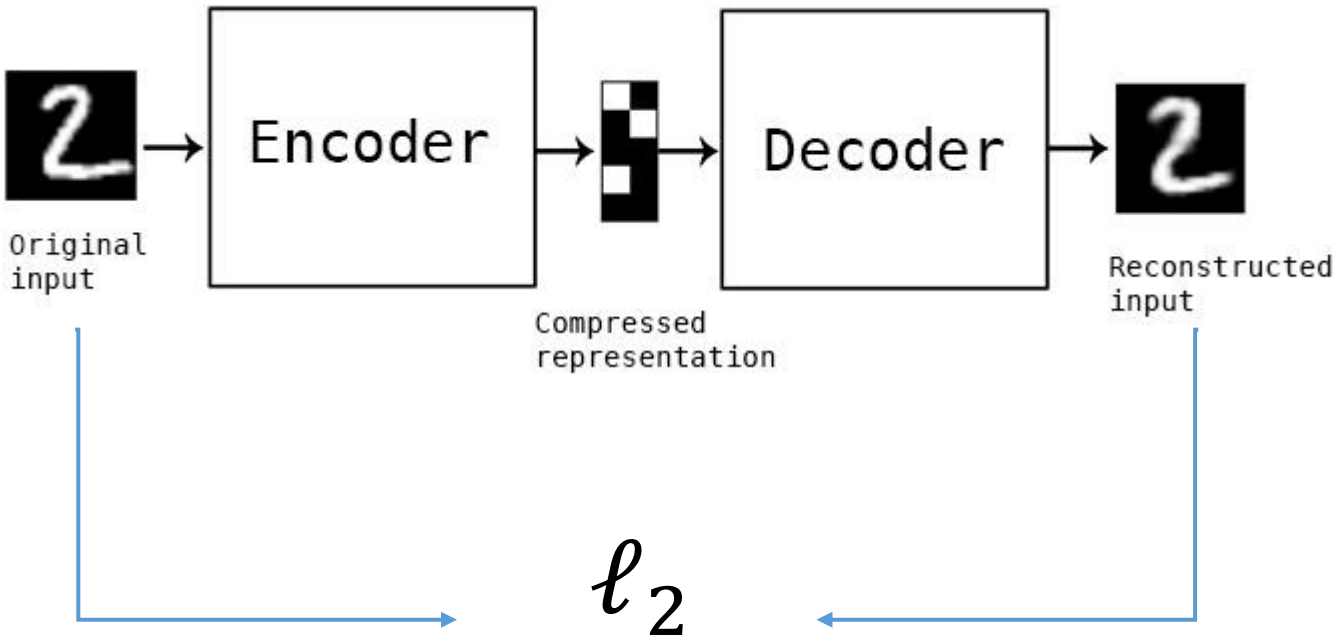
Autoencoders

- Encoder: Learns to map data to a compressed domain which is also called latent space.
- Decoder: Learns to map the compressed data to its initial form



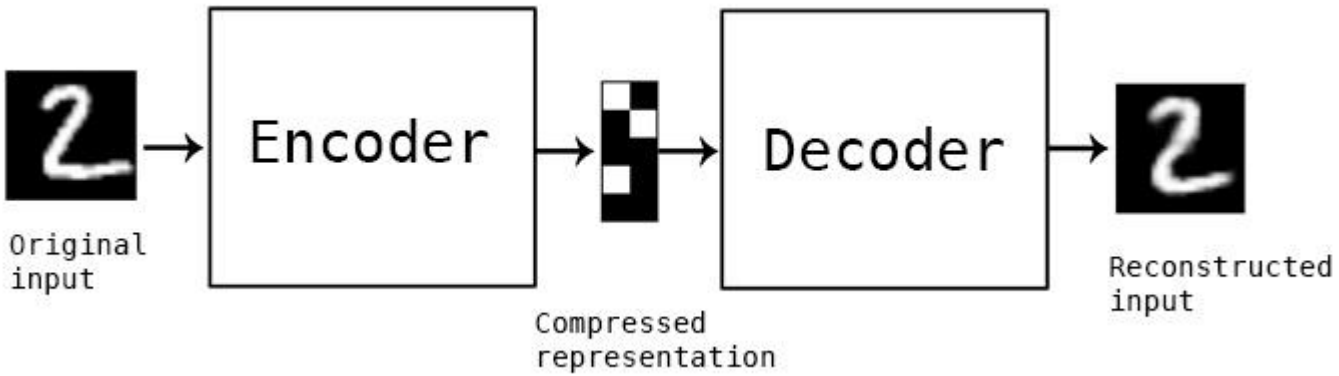
Autoencoders

- Through a loss function like ℓ_2 norm, we try to minimize the distance between actual data and the reconstructed data.



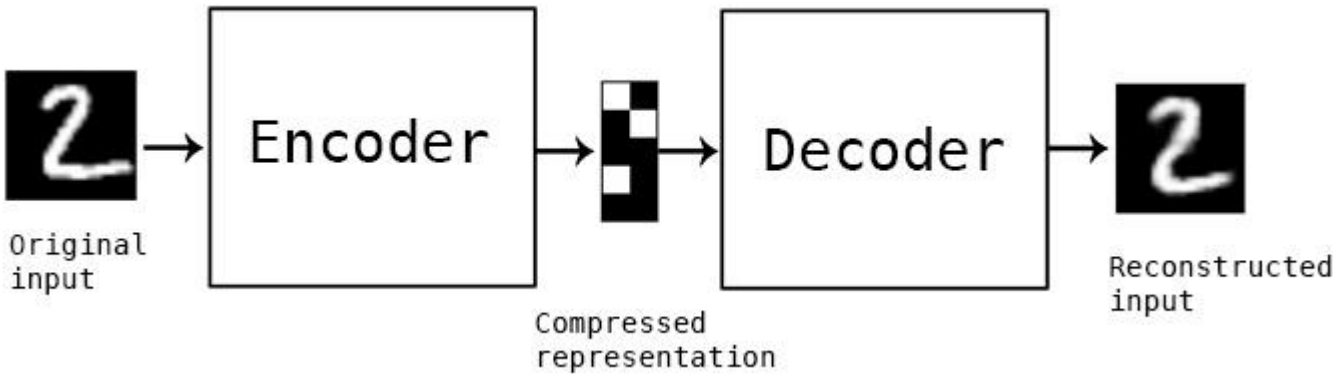
Autoencoders

- Decoder and encoder are usually two neural networks.



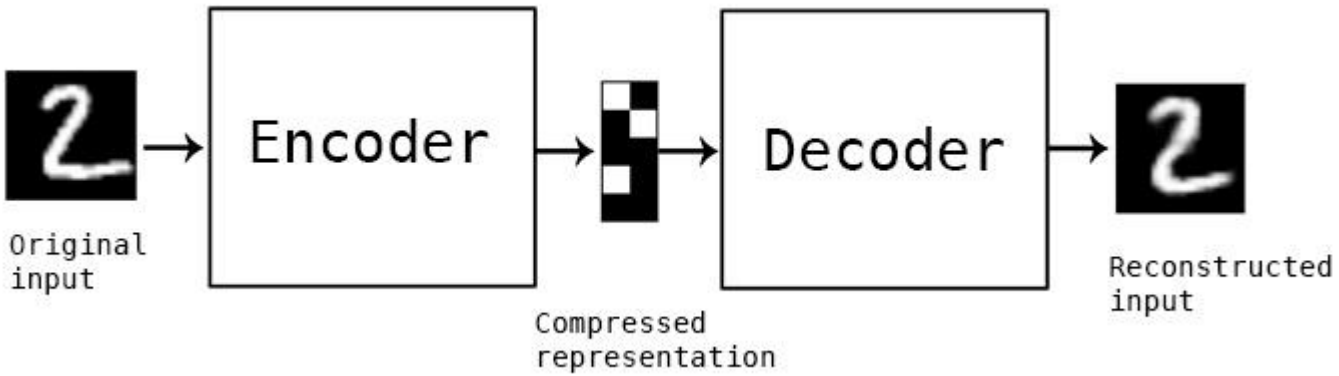
Autoencoders

- Since autoencoders are trained on specific data sets, they do not do a good job on other types of data sets
  - E.g., faces instead of digits



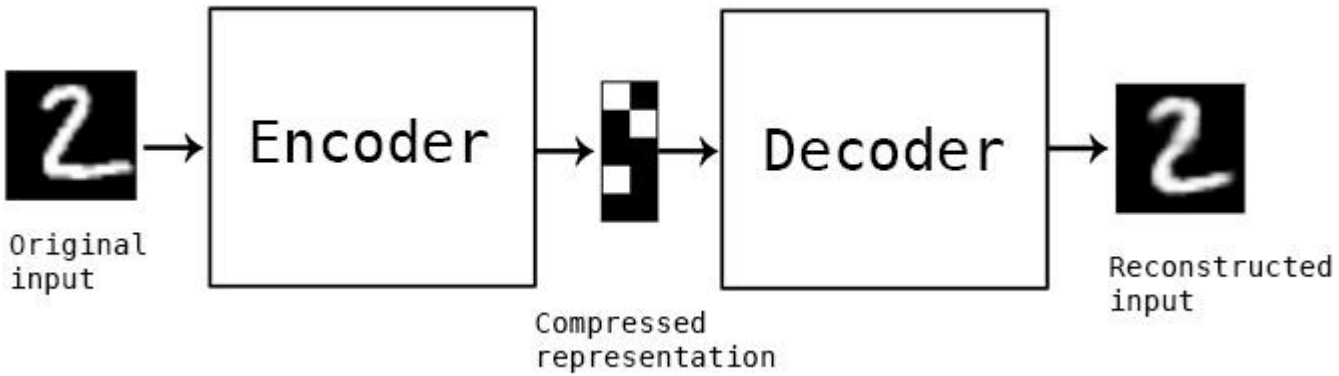
Autoencoders

- As opposed to some other techniques like wavelets that can be implemented in a loss-less form, autoencoders are lossy.



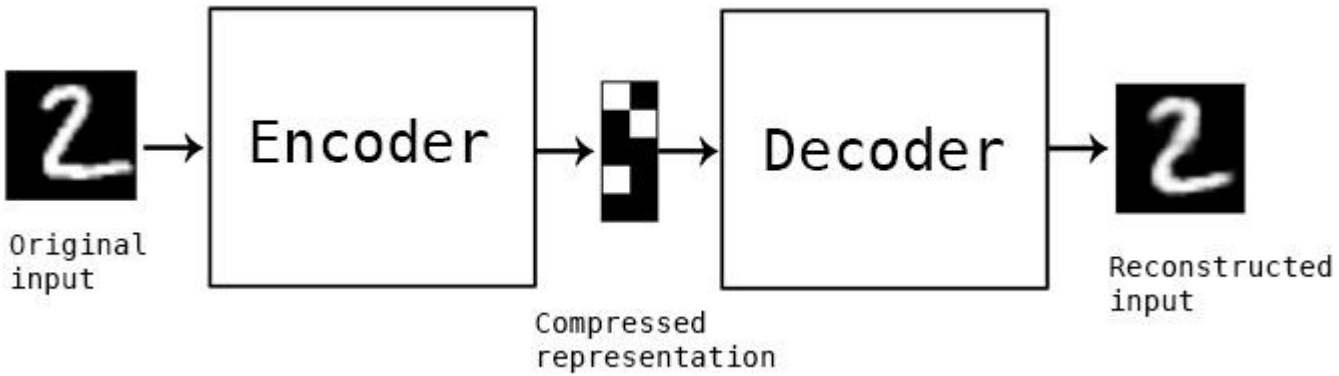
Autoencoders

- As opposed to some other techniques like wavelets that can be implemented in a loss-less form, autoencoders are lossy.
- However, we hope that they do a better job on compression (less information loss)



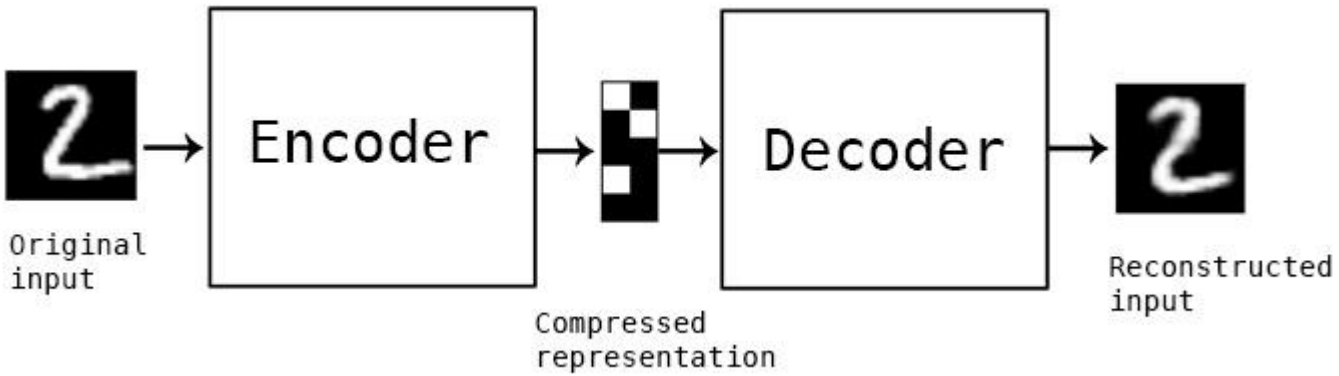
Autoencoders

- Plus since they are data specific, they are not as practical as general compression techniques like JPEG.



Autoencoders

- Plus since they are data specific, they are not as practical as general compression techniques like JPEG.
- However, they provide other benefits like denoising and dimensionality reduction.



Simplest Autoencoder

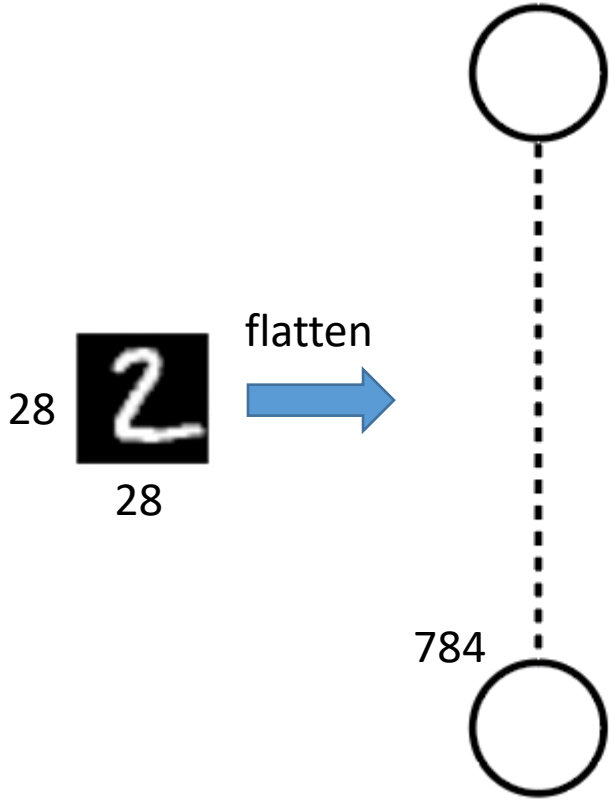
- Start with an image

28

28

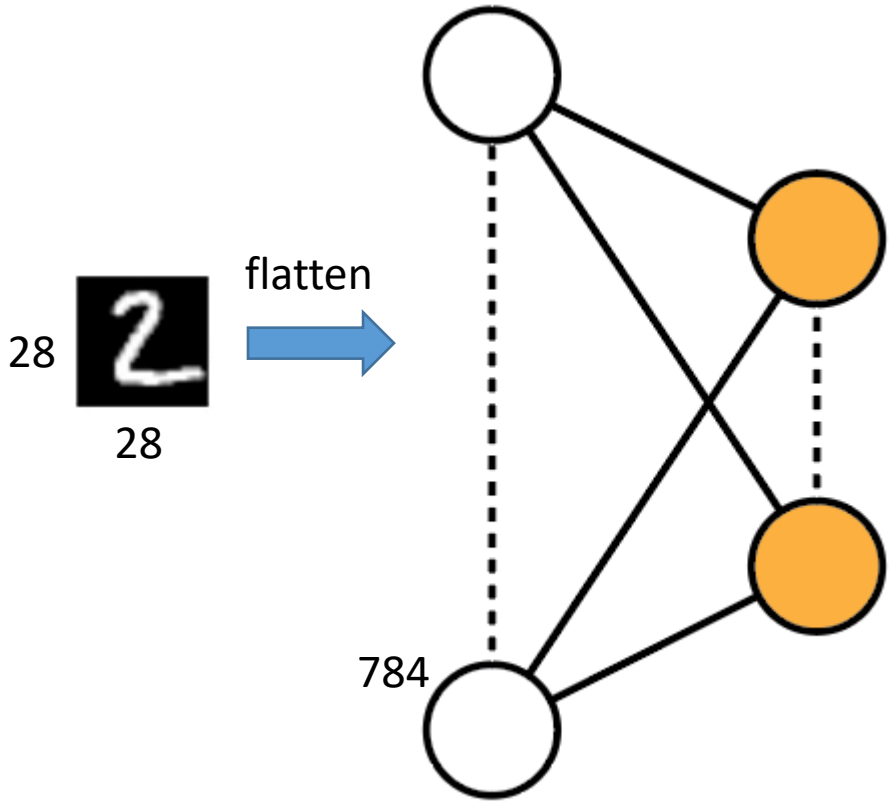
Simplest Autoencoder

- Flatten it to a vector



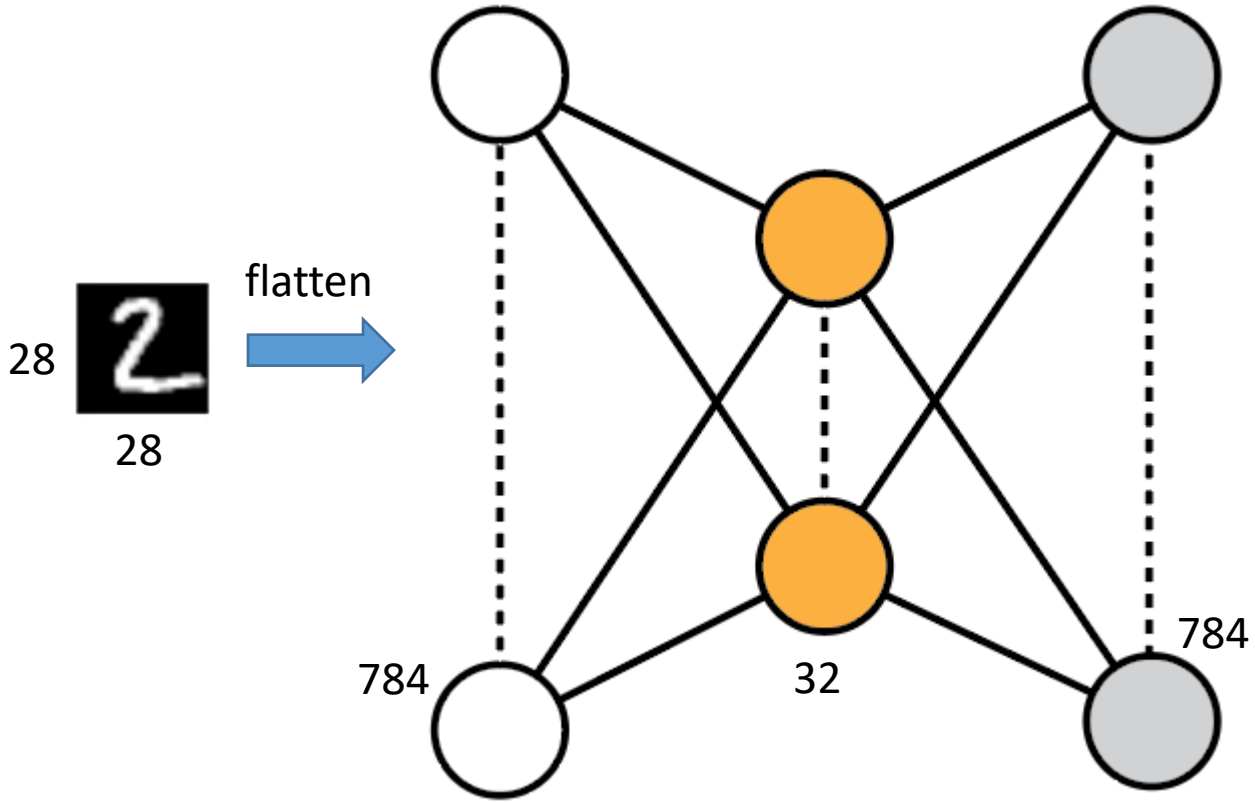
Simplest Autoencoder

- Linearly map it to a smaller domain (e.g.,32) via a linear dense network (no nonlinear activation.)



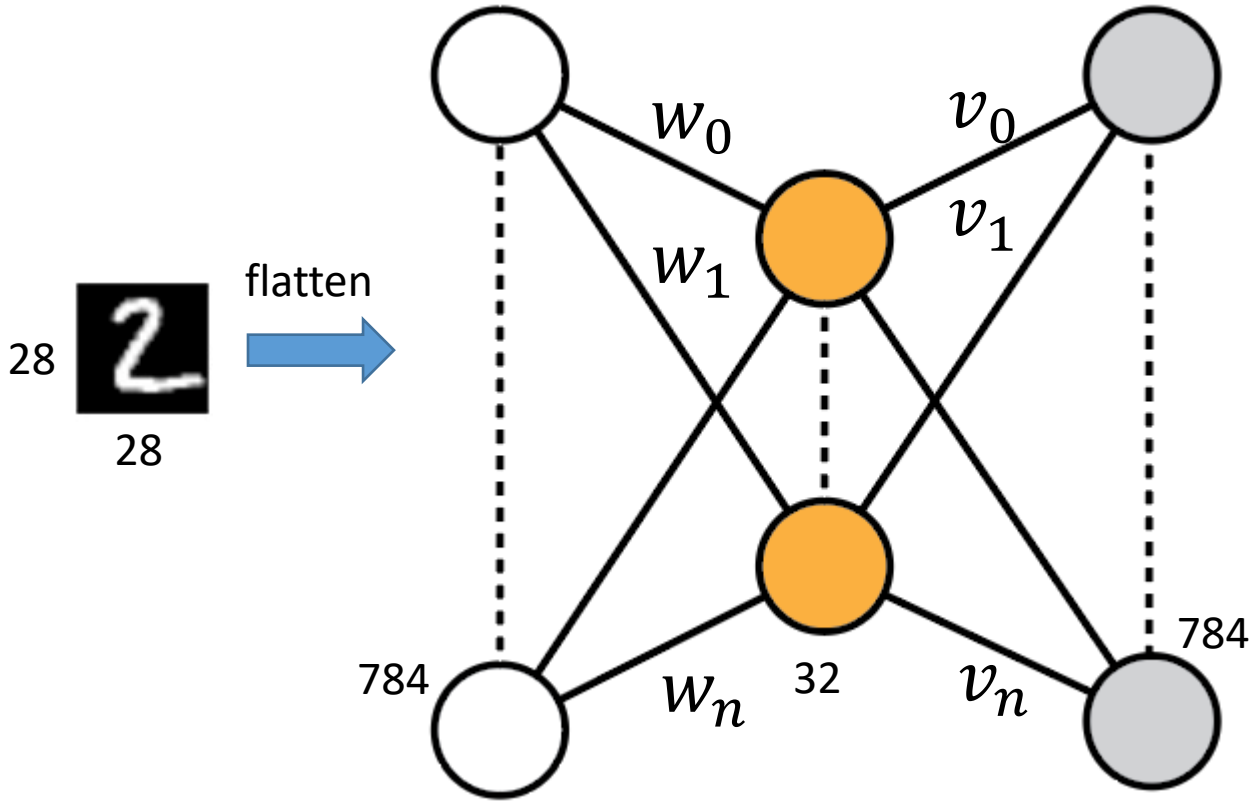
Simplest Autoencoder

- Map it back to the same dimension.



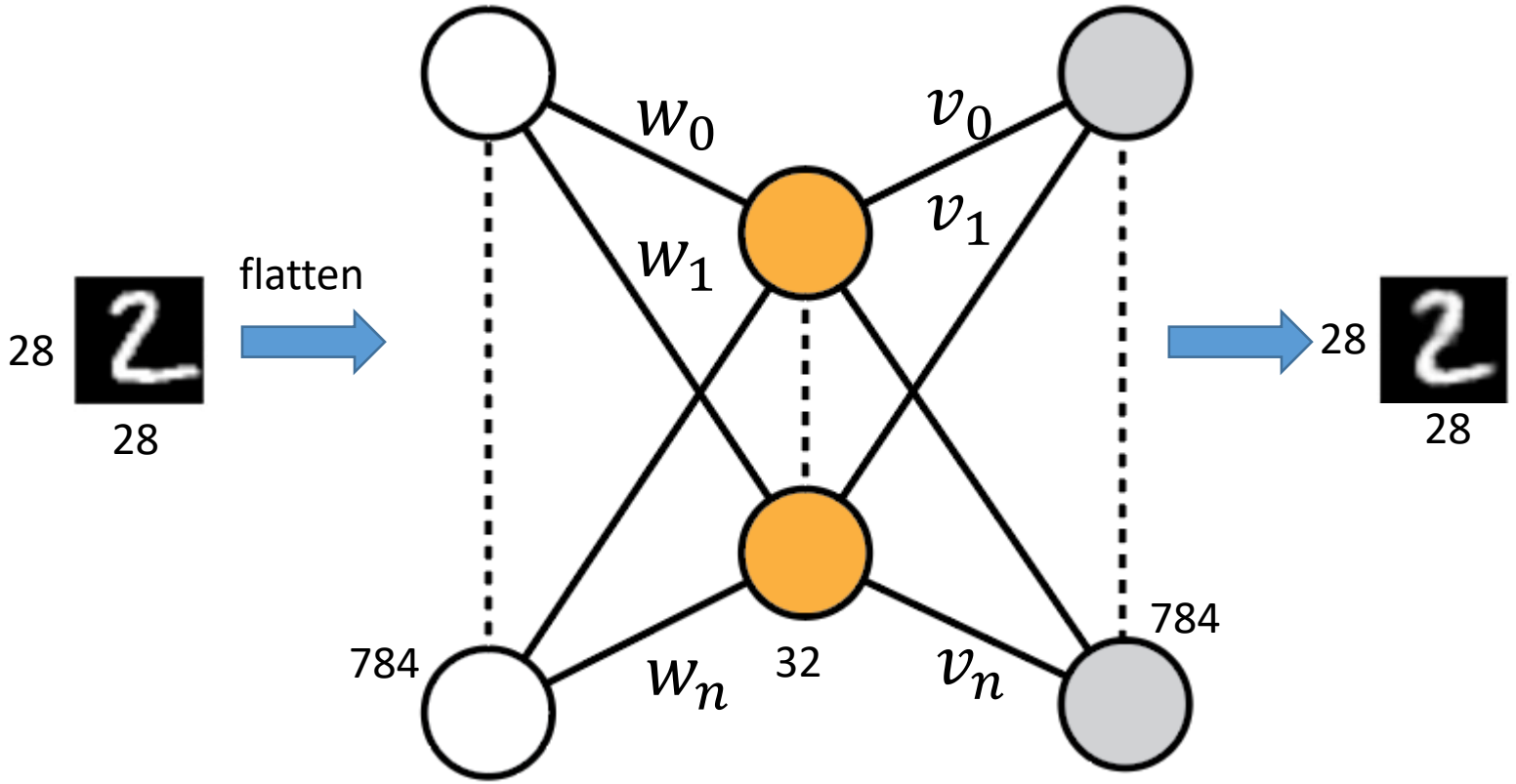
Simplest Autoencoder

- You only have integer values that are multiplied in the input.



Simplest Autoencoder

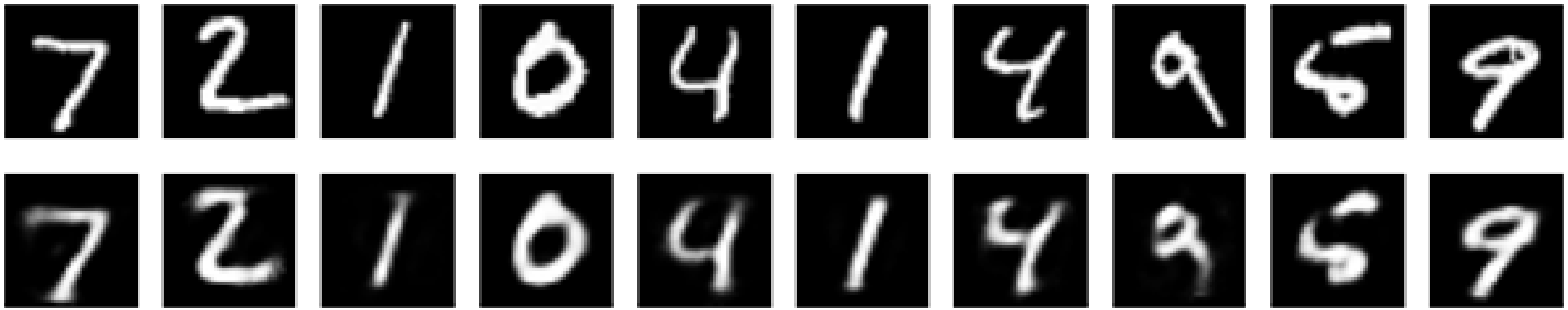
- Output is arranged back to the same format as the input.



Dense Linear Autoencoders

- Results

Data

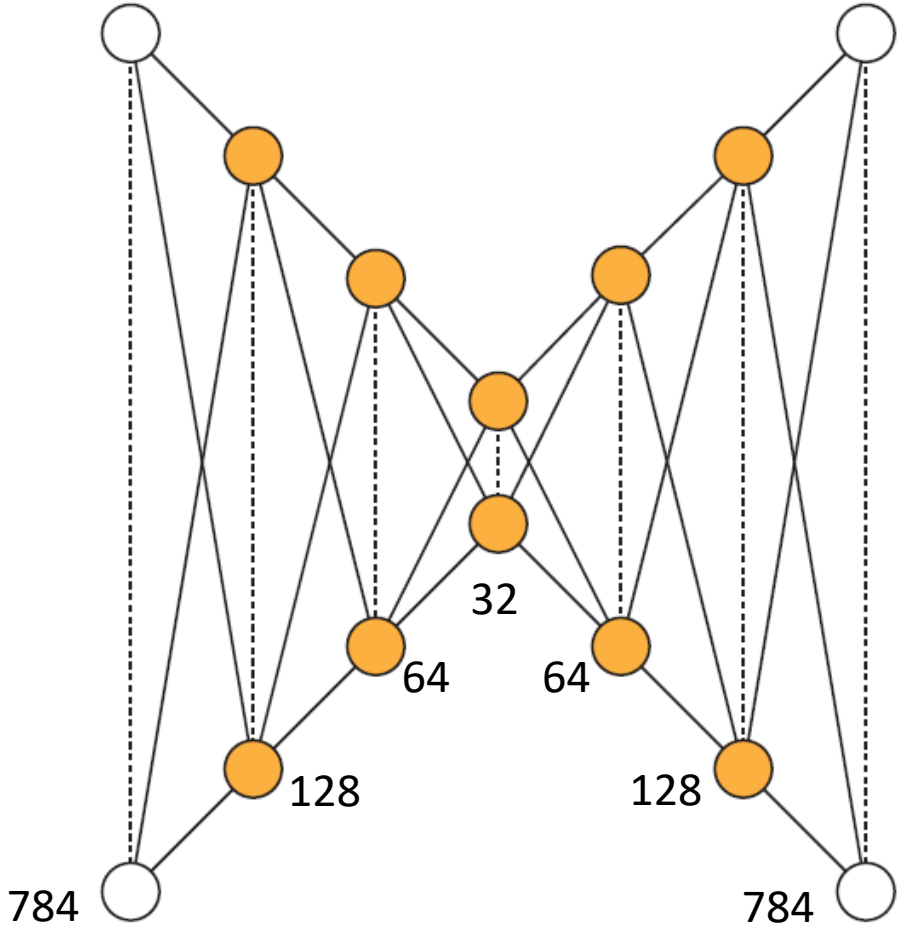


Reconstructed Data

https://blog.keras.io/building-autoencoders-in-keras.html

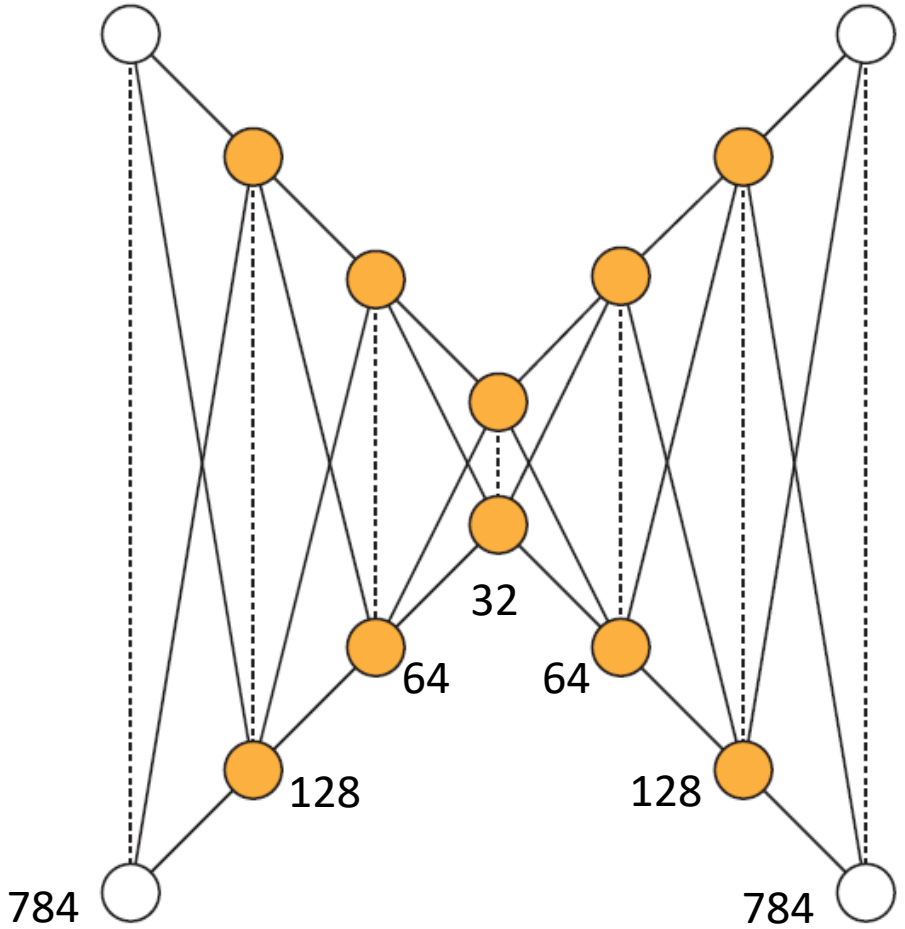
Deep Autoencoders

- We have multiple hidden layers



Deep Autoencoders

- We can use any non-linear function (e.g., Relu)



Deep Autoencoders

- Results

Data

Reconstructed Data

Convolutional Deep Autoencoders

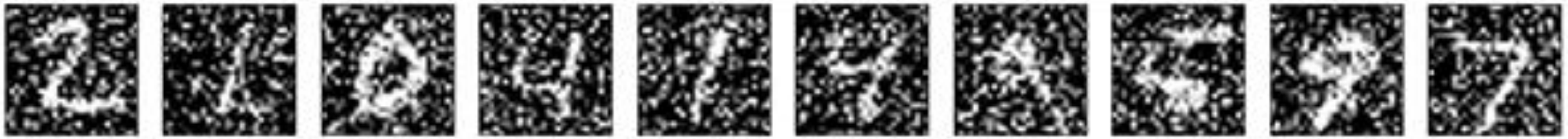
- Encoder: Conv-Relu-MaxPool
- Decode: Conv-Relu-Upsampling

Data

Reconstructed Data

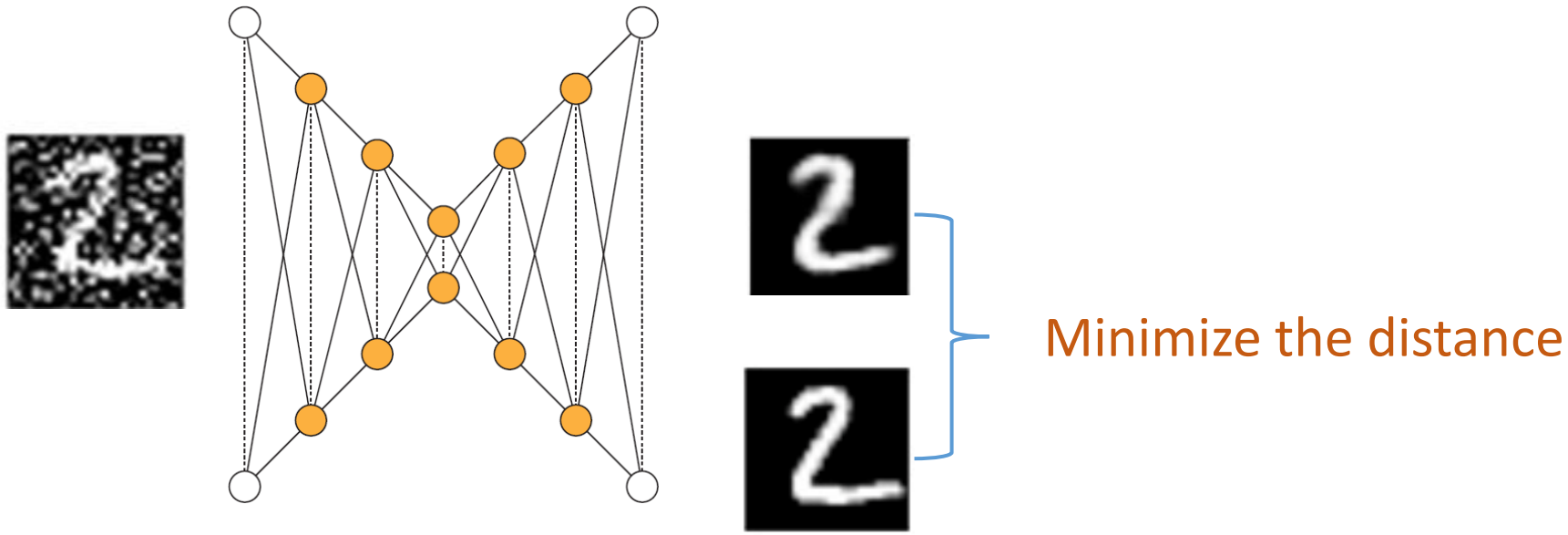
Application

- Denoising: remove unwanted noise from data



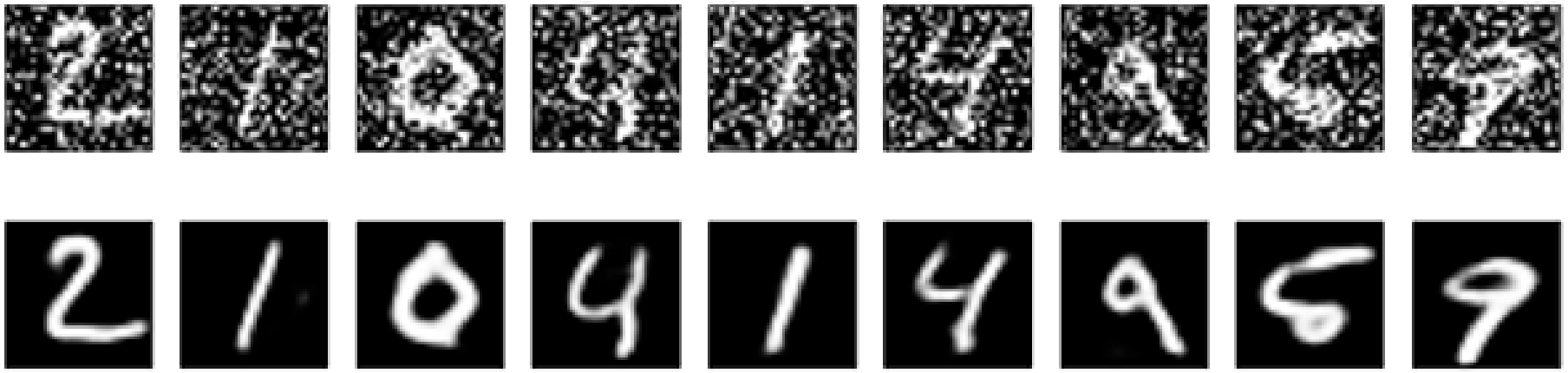
Application

- Train the convolutional autoencoder to remove noises from the image by inputting noisy data and minimizing the output against clean data.



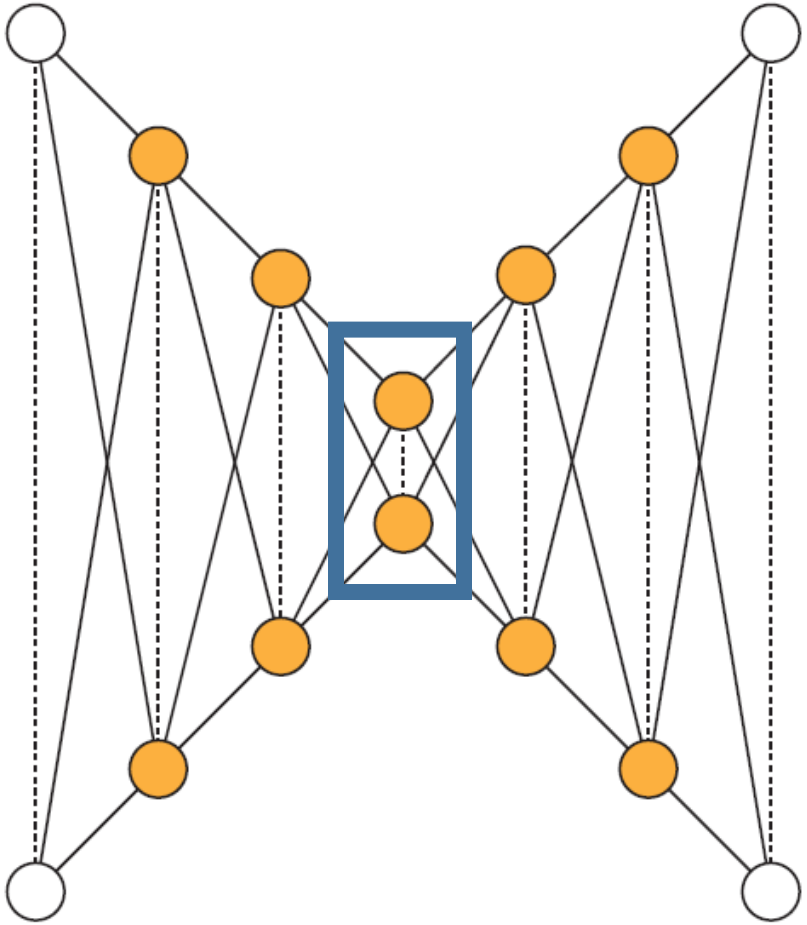
Application

- Results



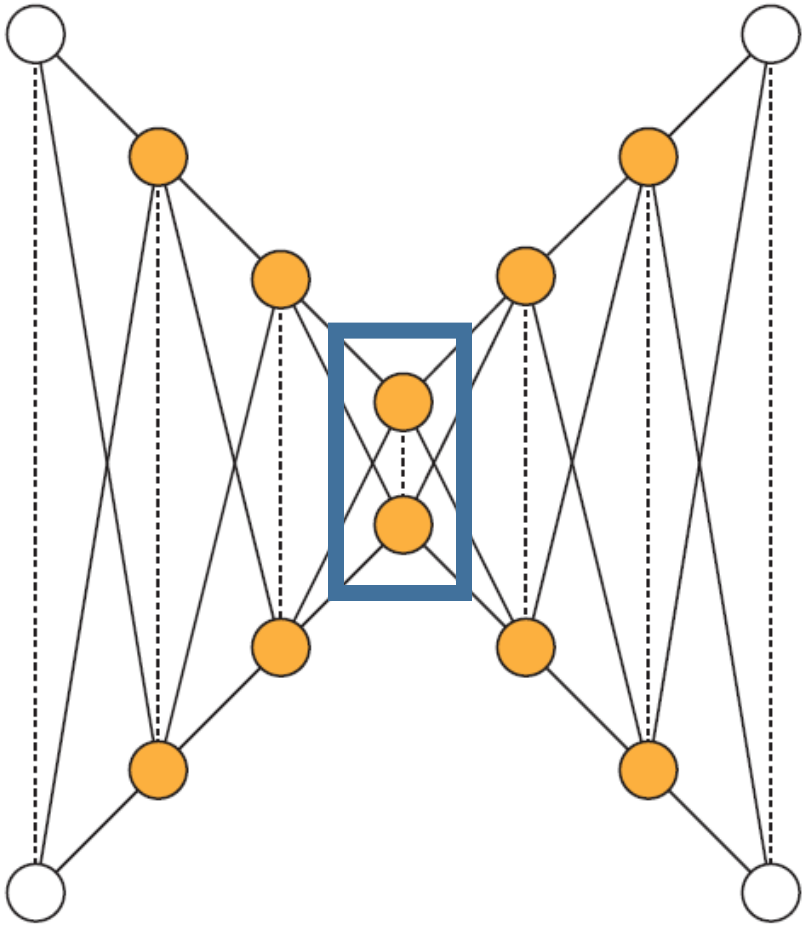
Autoencoders

- Latent space in autoencoders is unknown.



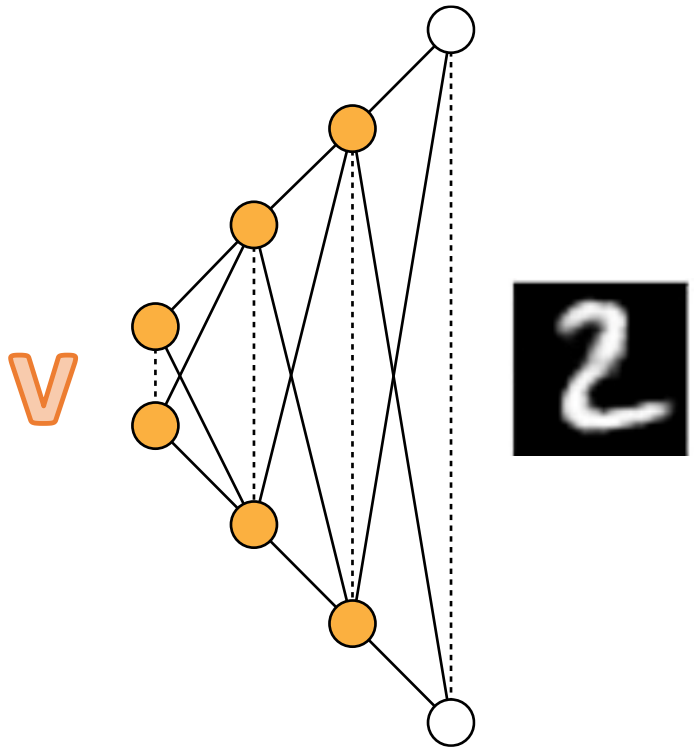
Variational Autoencoders

- What if we want to map our data to a known data distribution like Gaussian?



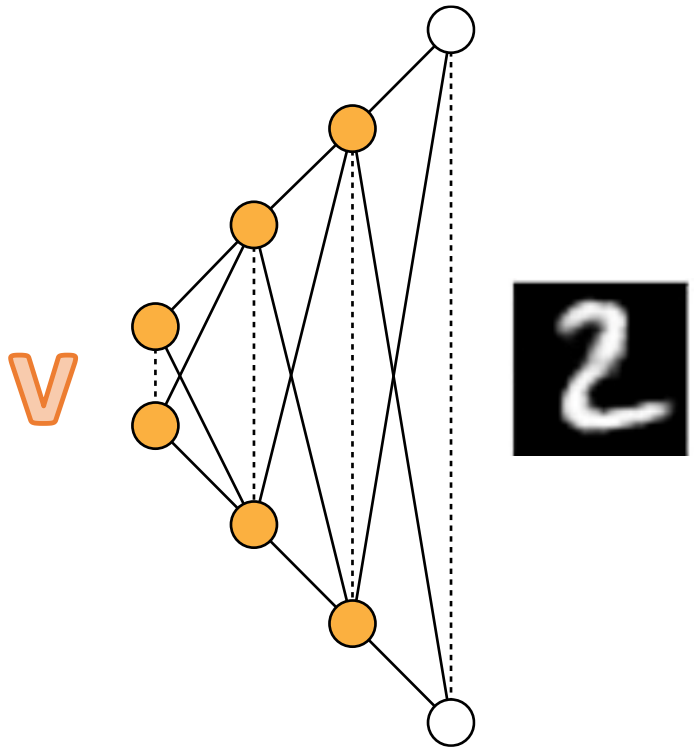
Variational Autoencoders

- Then we can sample the latent space and generate new data using decoder.



Variational Autoencoders

- In fact decoder performs as a generative model.



Background

- We can quantify the amount of information that an expression carries.

Background

- We can quantify the amount of information that an expression carries.

$$I = -logp(x)$$

Background

- We can quantify the amount of information that an expression carries.

$$I = -logp(x)$$

- Entropy is the average (expectation) of information

$$H_p = -\sum p(x)logp(x)$$

Background

- Entropy is the average (expectation) of information

$$H_p = -\sum p(x) log p(x)$$

Background

- Entropy is the average (expectation) of information

$$H_p = -\sum p(x) log p(x)$$

- KL-divergence is similar to entropy when we have two different distributions.

Background

- Entropy is the average (expectation) of information

$$H_p = -\sum p(x) log p(x)$$

- KL-divergence is similar to entropy when we have two different distributions. It measure dissimilarity of two distributions.

Background

- Entropy is the average (expectation) of information

$$H_p = -\sum p(x) log p(x)$$

- KL-divergence is similar to entropy when we have two different distributions. It measure dissimilarity of two distributions.

$$KL(p \parallel q) \approx H_p - H_q$$

Background

- KL-divergence is similar to entropy when we have two different distributions. It measure dissimilarity of two distributions.

$$KL(p \parallel q) \approx H_p - H_q$$

- KL-divergence is with respect to a distribution (e.g., p), we modify it to:

$$KL(p \parallel q) = -\sum p(x) log q(x) + \sum p(x) log p(x)$$

Background

- KL-divergence is similar to entropy when we have two different distributions. It measure dissimilarity of two distributions.

$$KL(p \parallel q) \approx H_p - H_q$$

- KL-divergence is with respect to a distribution (e.g., p), we modify it to:

$$KL(p \parallel q) = -\sum p(x) log q(x) + \sum p(x) log p(x)$$

$$= \sum p(x) log \frac{p(x)}{q(x)} = -\sum p(x) log \frac{q(x)}{p(x)}$$

Background

- KL-divergence is always non-negative

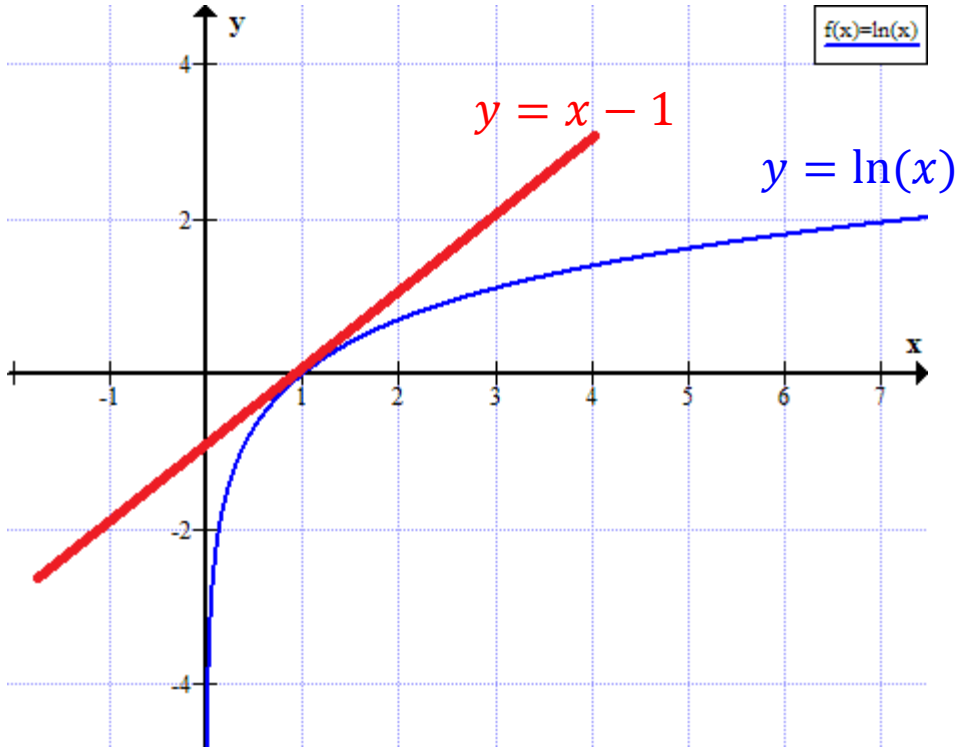
Background

- KL-divergence is always non-negative

why?

Background

- KL-divergence is always non-negative



$$\ln(x) \leq x - 1, x > 0$$

We are assuming that the base in KL Divergence is e

Background

- KL-divergence is always non-negative

$$\ln(x) \leq x - 1, x > 0$$

$$-KL(p \parallel q) = \sum p(x) ln \frac{q(x)}{p(x)}$$

Background

- KL-divergence is always non-negative

$$\ln(x) \leq x - 1, x > 0$$

$$-KL(p \parallel q) = \sum p(x) ln \frac{q(x)}{p(x)} \leq \sum p(x) \left(\frac{q(x)}{p(x)} - 1 \right)$$

Background

- KL-divergence is always non-negative

$$\ln(x) \leq x - 1, x > 0$$

$$-KL(p \parallel q) = \sum p(x) ln \frac{q(x)}{p(x)} \leq \sum p(x) \left(\frac{q(x)}{p(x)} - 1 \right) = \sum q(x) - \sum p(x)$$

Background

- KL-divergence is always non-negative

$$\ln(x) \leq x - 1, x > 0$$

$$-KL(p \parallel q) = \sum p(x) ln \frac{q(x)}{p(x)} \leq \sum p(x) \left(\frac{q(x)}{p(x)} - 1 \right) = \sum q(x) - \sum p(x) = 1 - 1 = 0$$

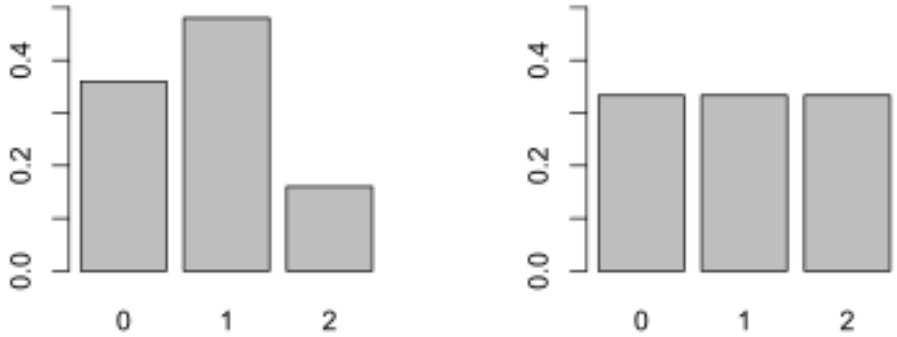
Background

- $KL(p \parallel q) \neq KL(q \parallel p)$

Background

- $KL(p \parallel q) \neq KL(q \parallel p)$

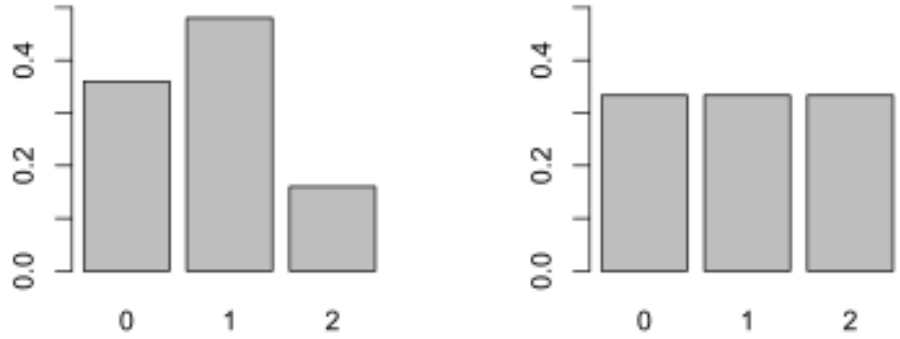
- Let's see an example:



Background

- $KL(p \parallel q) \neq KL(q \parallel p)$

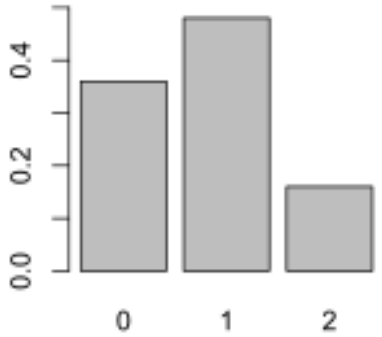
- Let's see an example:

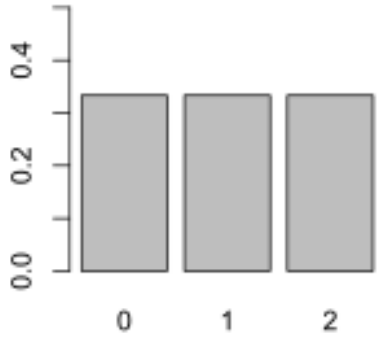


x	0	1	2
p(x)	0.36	0.48	0.16
q(x)	0.333	0.333	0.333

Background

- $KL(p \parallel q) \neq KL(q \parallel p)$





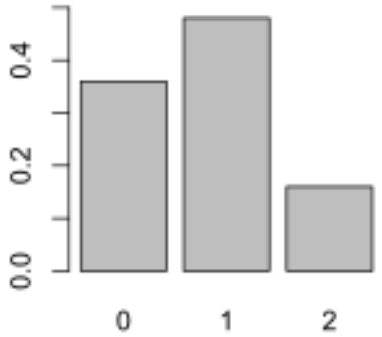
- Let's see an example:

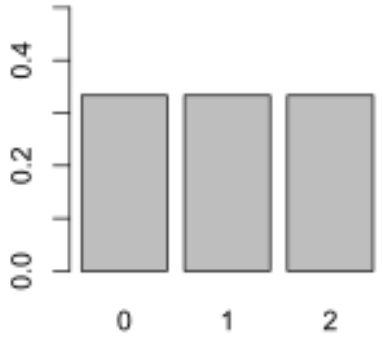
x	0	1	2
p(x)	0.36	0.48	0.16
q(x)	0.333	0.333	0.333

$$KL(p \parallel q) = \sum p(x) \ln\left(\frac{p(x)}{q(x)}\right) =$$

Background

- $KL(p \parallel q) \neq KL(q \parallel p)$





- Let's see an example:

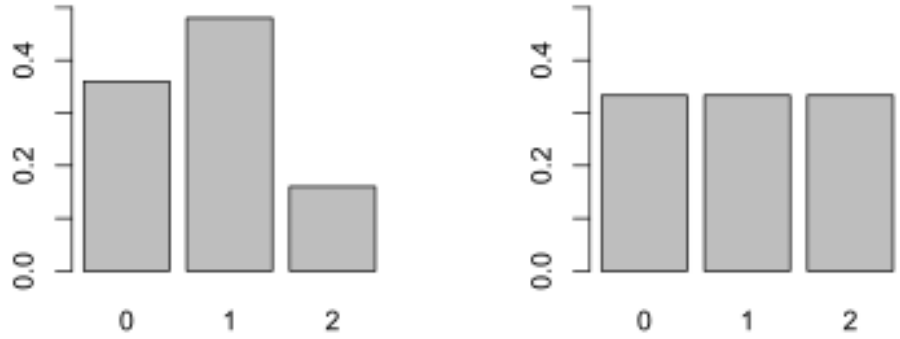
x	0	1	2
p(x)	0.36	0.48	0.16
q(x)	0.333	0.333	0.333

$$KL(p \parallel q) = \sum p(x) \ln\left(\frac{p(x)}{q(x)}\right) = 0.36 \ln\left(\frac{0.36}{0.33}\right) + 0.48 \ln\left(\frac{0.48}{0.33}\right) + 0.16 \ln\left(\frac{0.16}{0.33}\right)$$

Background

- $KL(p \parallel q) \neq KL(q \parallel p)$

- Let's see an example:



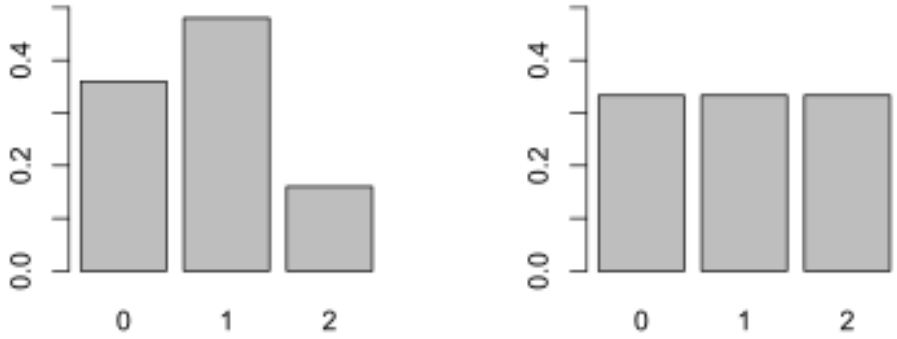
x	0	1	2
p(x)	0.36	0.48	0.16
q(x)	0.333	0.333	0.333

$$KL(p \parallel q) = \sum p(x) \ln\left(\frac{p(x)}{q(x)}\right) = 0.36 \ln\left(\frac{0.36}{0.33}\right) + 0.48 \ln\left(\frac{0.48}{0.33}\right) + 0.16 \ln\left(\frac{0.16}{0.33}\right) = 0.0852996$$

Background

- $KL(p \parallel q) \neq KL(q \parallel p)$

- Let's see an example:



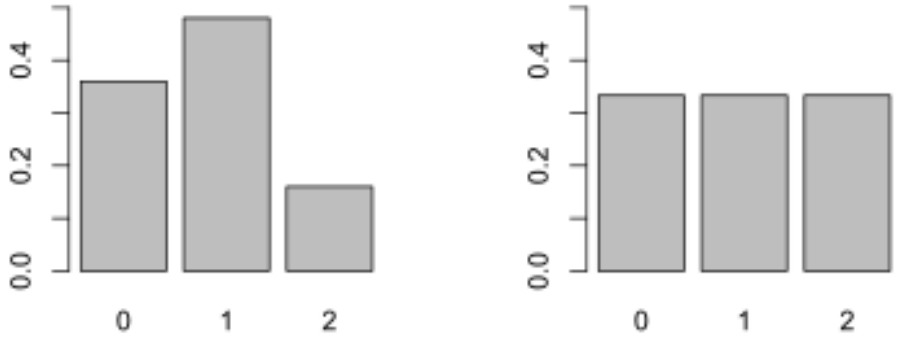
x	0	1	2
p(x)	0.36	0.48	0.16
q(x)	0.333	0.333	0.333

$$KL(p \parallel q) = \sum p(x) \ln\left(\frac{p(x)}{q(x)}\right) = 0.36 \ln\left(\frac{0.36}{0.33}\right) + 0.48 \ln\left(\frac{0.48}{0.33}\right) + 0.16 \ln\left(\frac{0.16}{0.33}\right) = 0.0852996$$

$$KL(q \parallel p) = \sum q(x) \ln\left(\frac{q(x)}{p(x)}\right) = 0.333 \ln\left(\frac{0.333}{0.36}\right) + 0.333 \ln\left(\frac{0.333}{0.48}\right) + 0.333 \ln\left(\frac{0.333}{0.26}\right) = 0.097455$$

Background

- $KL(p \parallel q) \neq KL(q \parallel p)$
- Let's see an example:



x	0	1	2
p(x)	0.36	0.48	0.16
q(x)	0.333	0.333	0.333

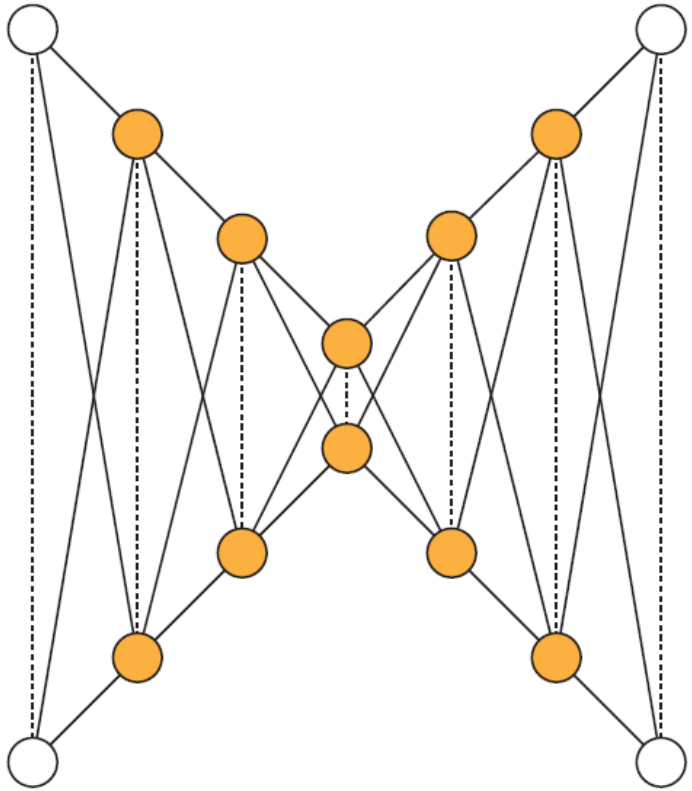
$$KL(p \parallel q) = \sum p(x) \ln\left(\frac{p(x)}{q(x)}\right) = 0.36 \ln\left(\frac{0.36}{0.33}\right) + 0.48 \ln\left(\frac{0.48}{0.33}\right) + 0.16 \ln\left(\frac{0.16}{0.33}\right) = 0.0852996$$

$$\neq$$

$$KL(q \parallel p) = \sum q(x) \ln\left(\frac{q(x)}{p(x)}\right) = 0.333 \ln\left(\frac{0.333}{0.36}\right) + 0.48 \ln\left(\frac{0.333}{0.48}\right) + 0.16 \ln\left(\frac{0.333}{0.26}\right) = 0.097455$$

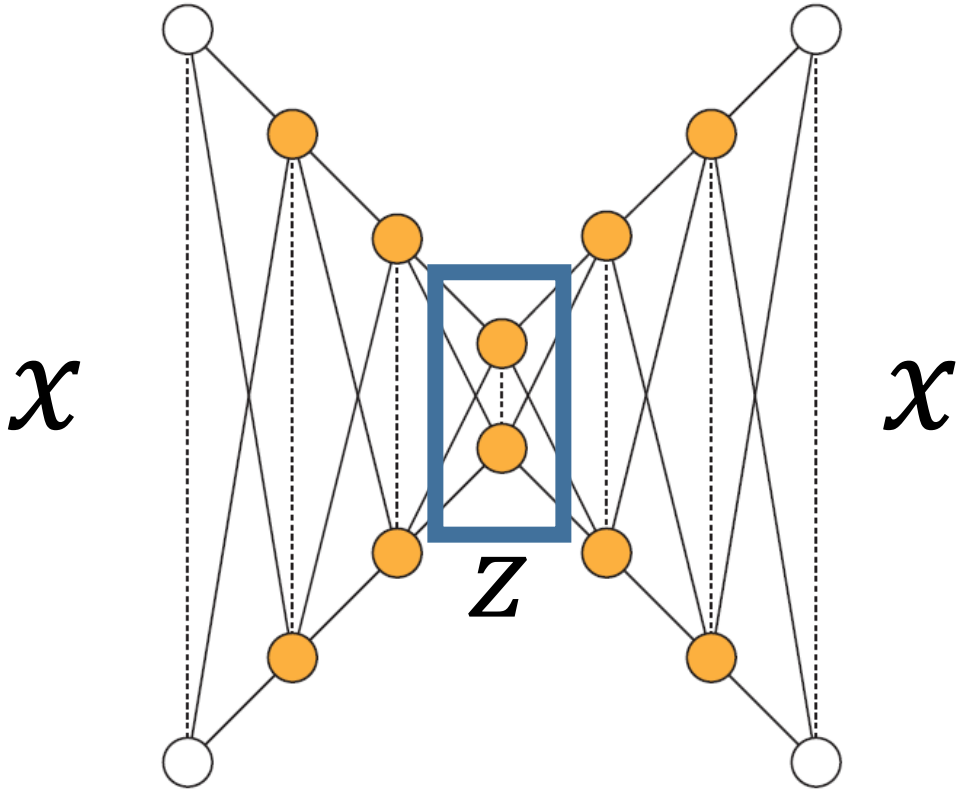
Variational Autoencoders

- The idea of VAE is to use deep autoencoders for generation.



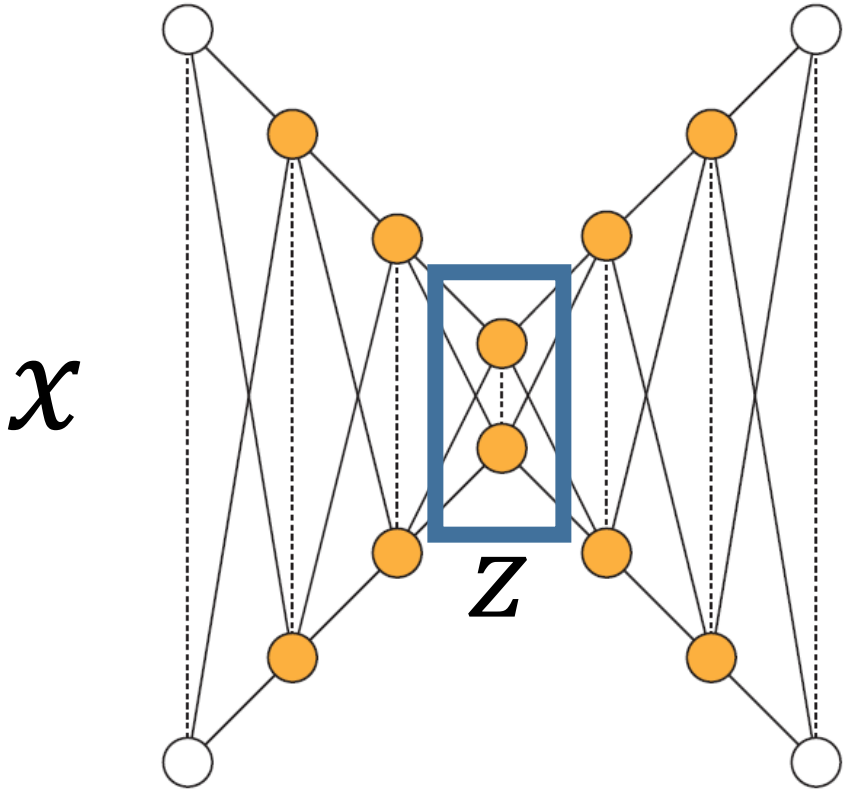
Variational Autoencoders

- The problem with autoencoders is that we don't know the distribution of the latent space for sampling.



Variational Autoencoders

- We want to know the distribution of $p(z|x)$ so that we can sample z and produce x

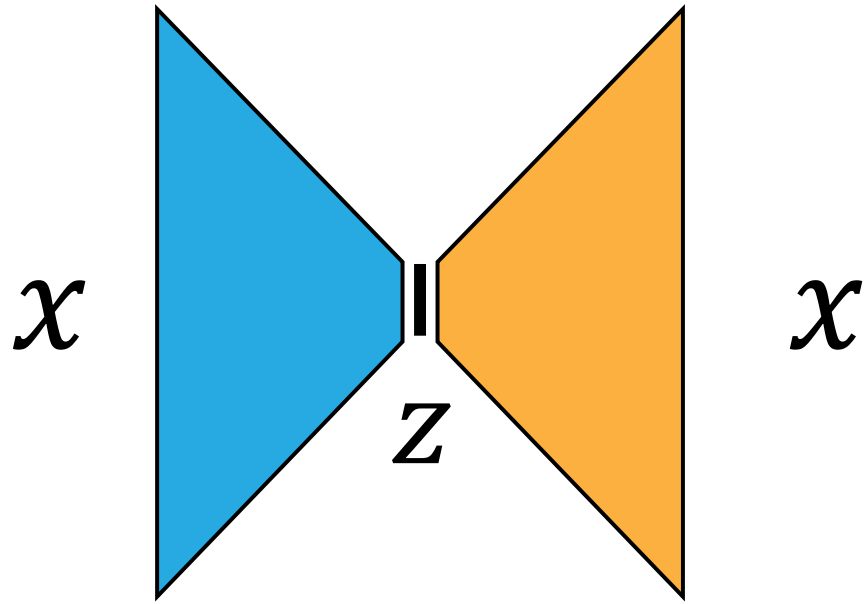


Variational Autoencoders

- It is hard to model $p(z|x)$ directly

$$p(z|x) = \frac{p(x,z)}{p(x)}$$

Unknown data distribution

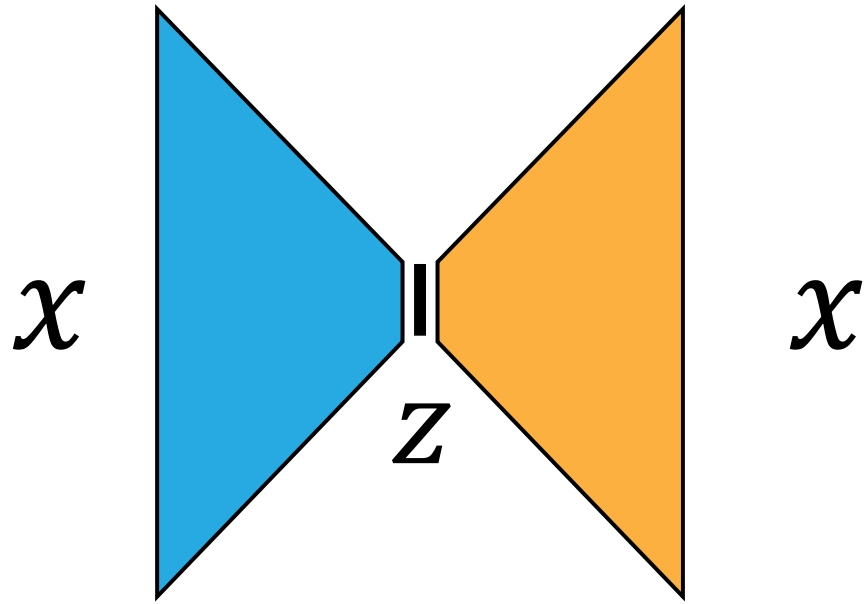


Variational Autoencoders

- We try to make $p(z|x)$ similar to a known distribution: $q(z)$

$$p(z|x) = \frac{p(x,z)}{p(x)}$$

$q(z)$: Gaussian



Variational Autoencoders

- We try to make $p(z|x)$ similar to a known distribution: $q(z)$
- $q(z)$ can be anything in the general form.

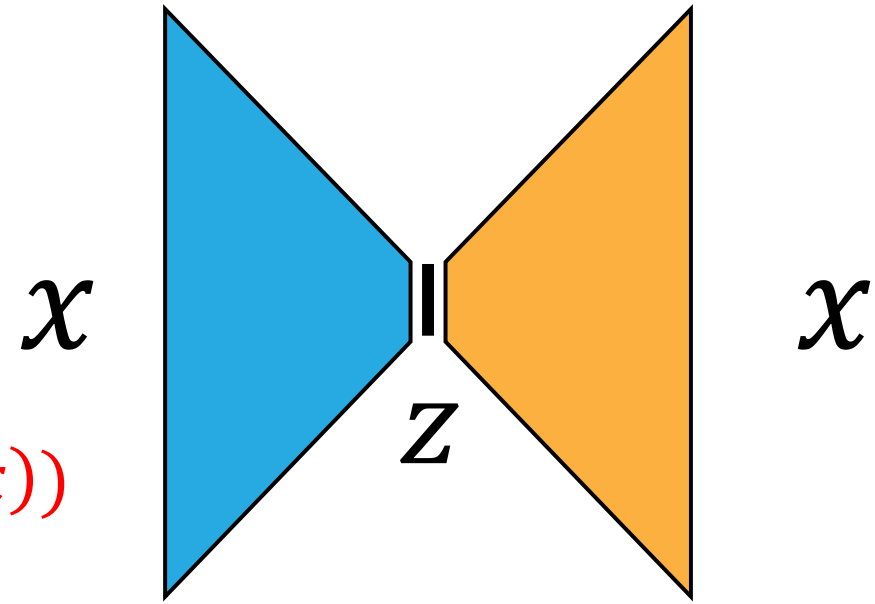
$$p(z|x) = \frac{p(x,z)}{p(x)}$$

Variational Autoencoders

- We try to minimize the KL-Divergence of $p(z|x)$ and $q(z)$

$$p(z|x) = \frac{p(x,z)}{p(x)}$$

$$\min KL(q(z) \parallel p(z|x))$$



Variational Autoencoders

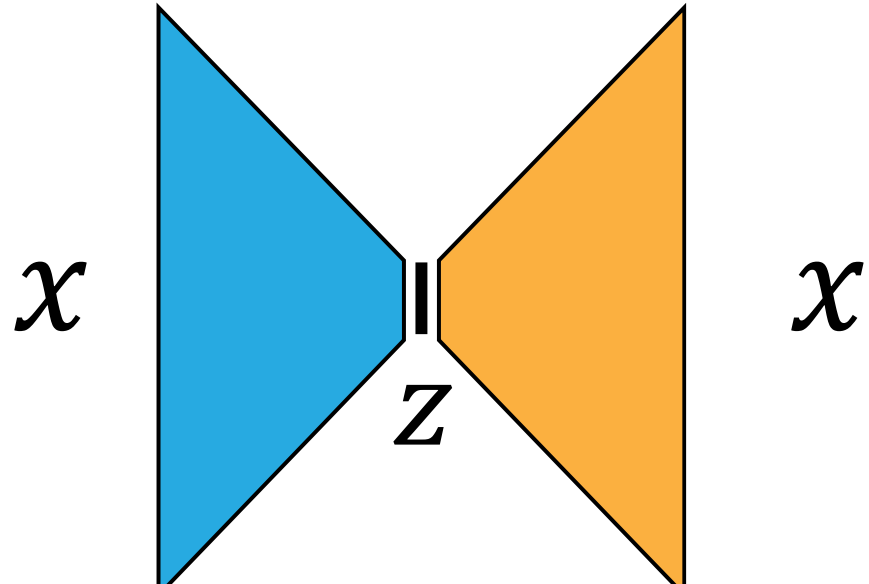
- The sketch of the proof:
  1. Try to $\min KL(q(z) \parallel p(z|x))$

Variational Autoencoders

- The sketch of the proof:
  1. Try to $\min KL(q(z) \parallel p(z|x))$
  2. The above minimization turns out to be equivalent to maximizing lower bound $\mathcal{L}$ for $p(x)$

Variational Autoencoders

- The sketch of the proof:
  1. Try to $\min KL(q(z) \parallel p(z|x))$
  2. The above minimization turns out to be equivalent to maximizing lower bound $\mathcal{L}$ for $p(x)$
  3. Maximizing lower bound is actually equivalent to making the latent space close to a known distribution like Gaussian and also minimizing the reconstruction error.

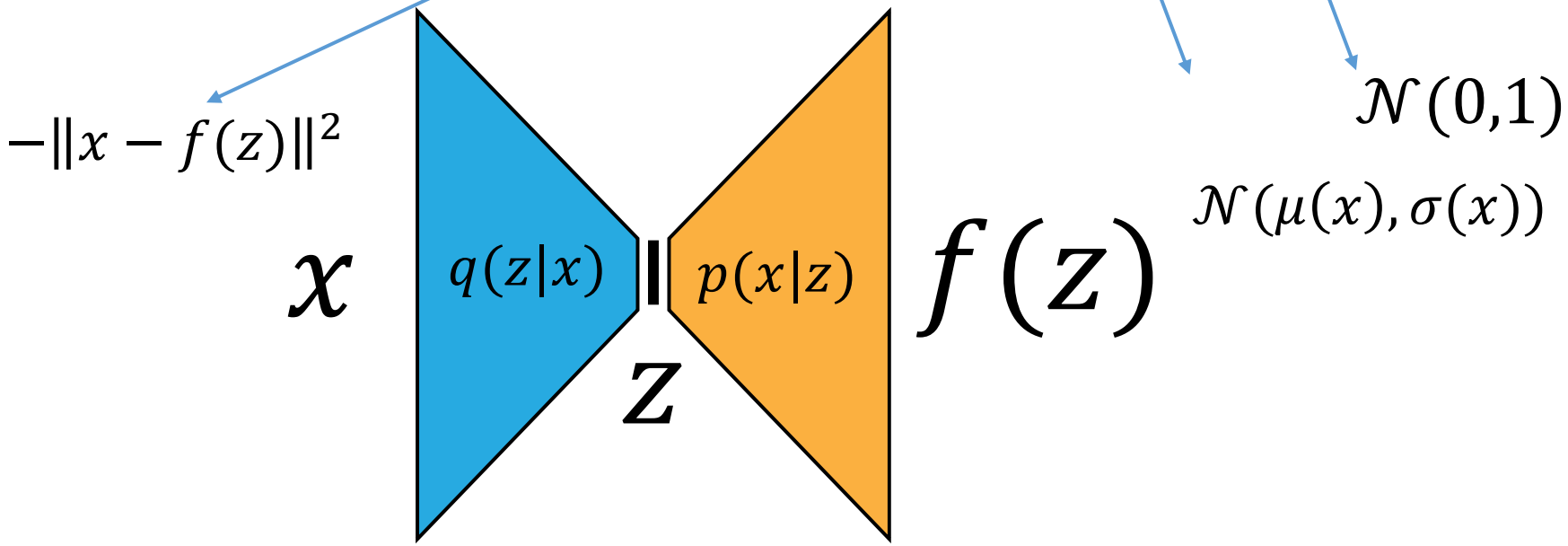


https://arash-mham.github.io/visual-computing-II/

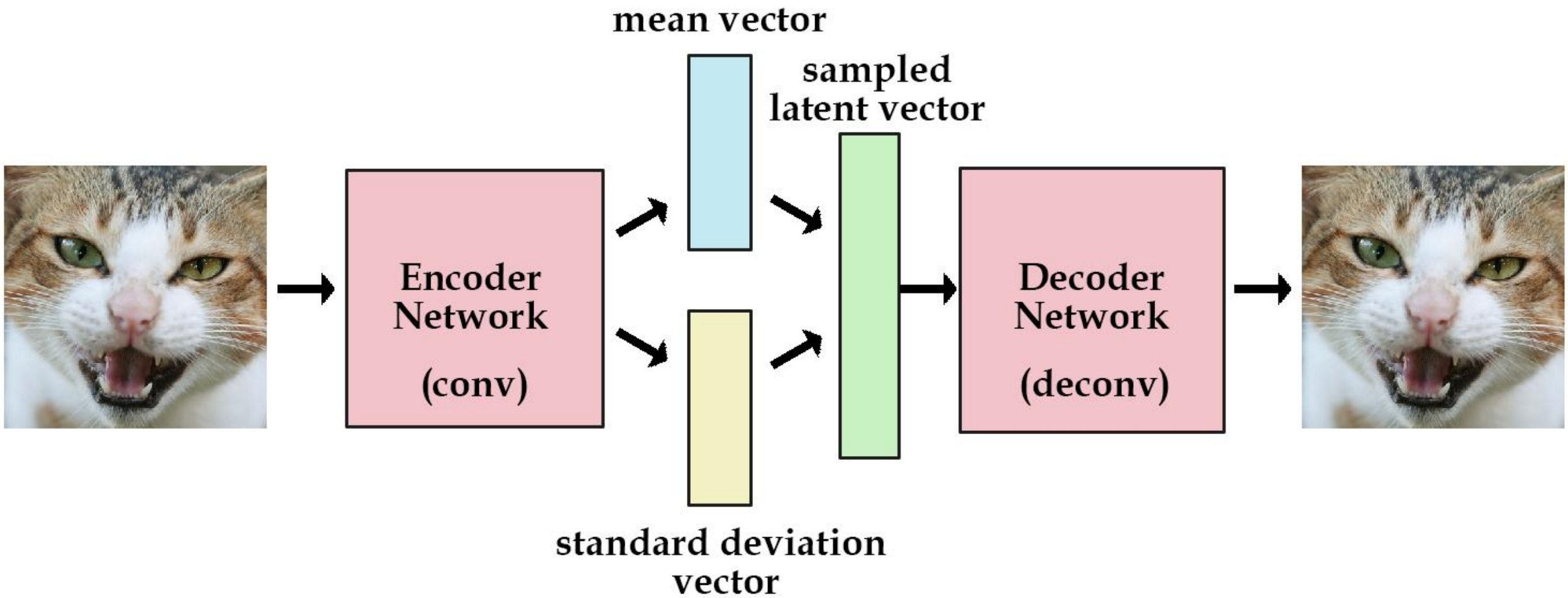
Variational Autoencoders

- If we choose Gaussian, first term will be a reconstruction error

$$\max \sum q(z|x) \log\left(\frac{p(x,z)}{q(z|x)}\right) = E_q(\log p(x|z)) - KL(q(z|x) \| p(z))$$

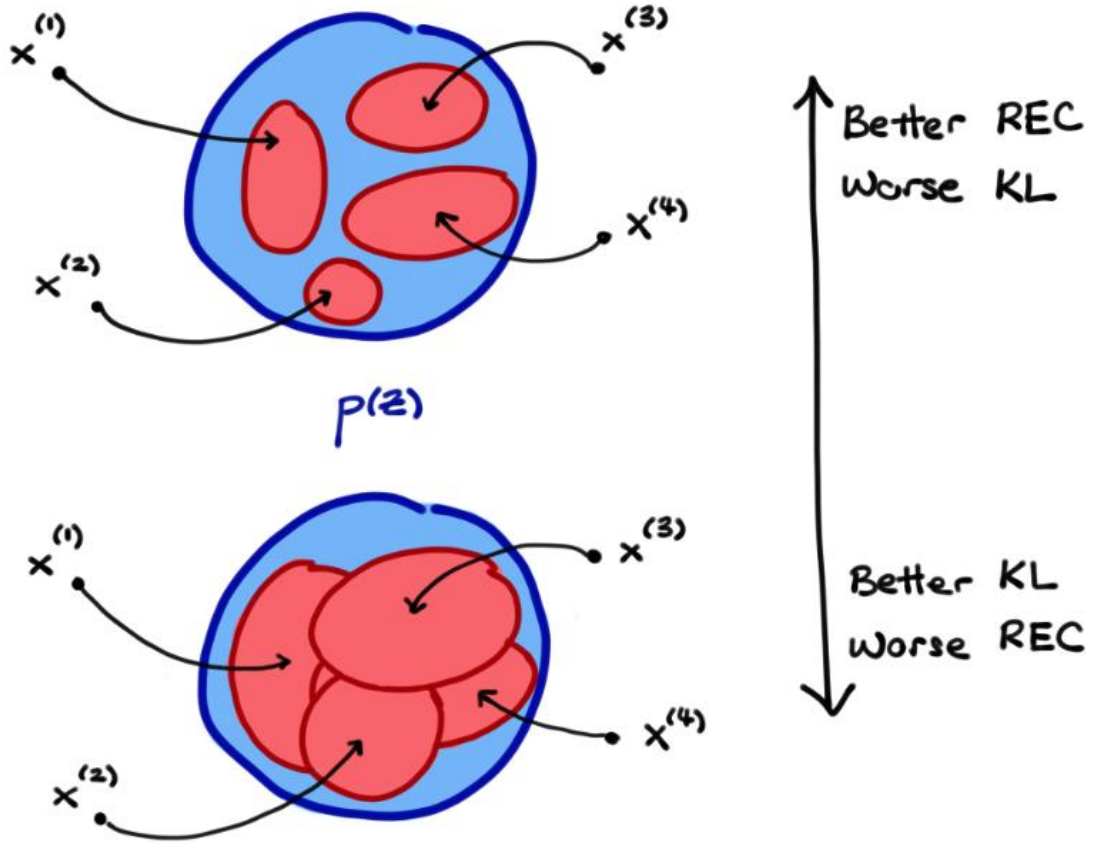


Summary


mean vector
sampled
latent vector
Encoder
Network
(conv)
Decoder
Network
(deconv)
standard deviation
vector

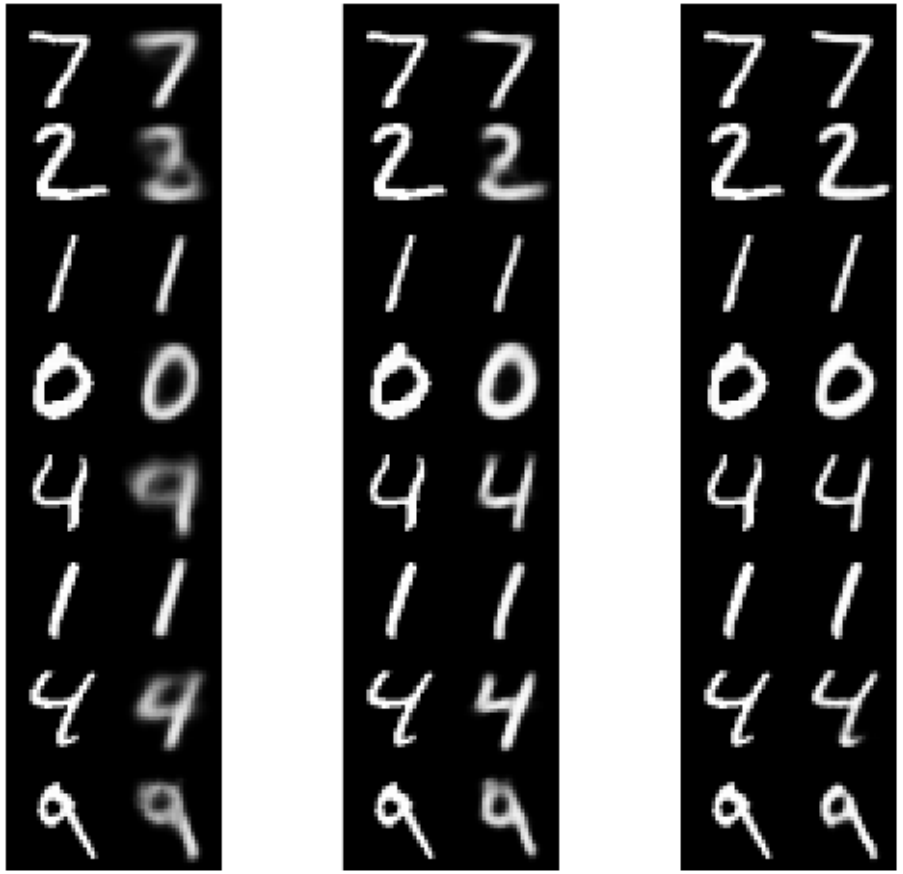
Variational Autoencoders

- KL-Divergence and reconstruction error compete with each other.



Results

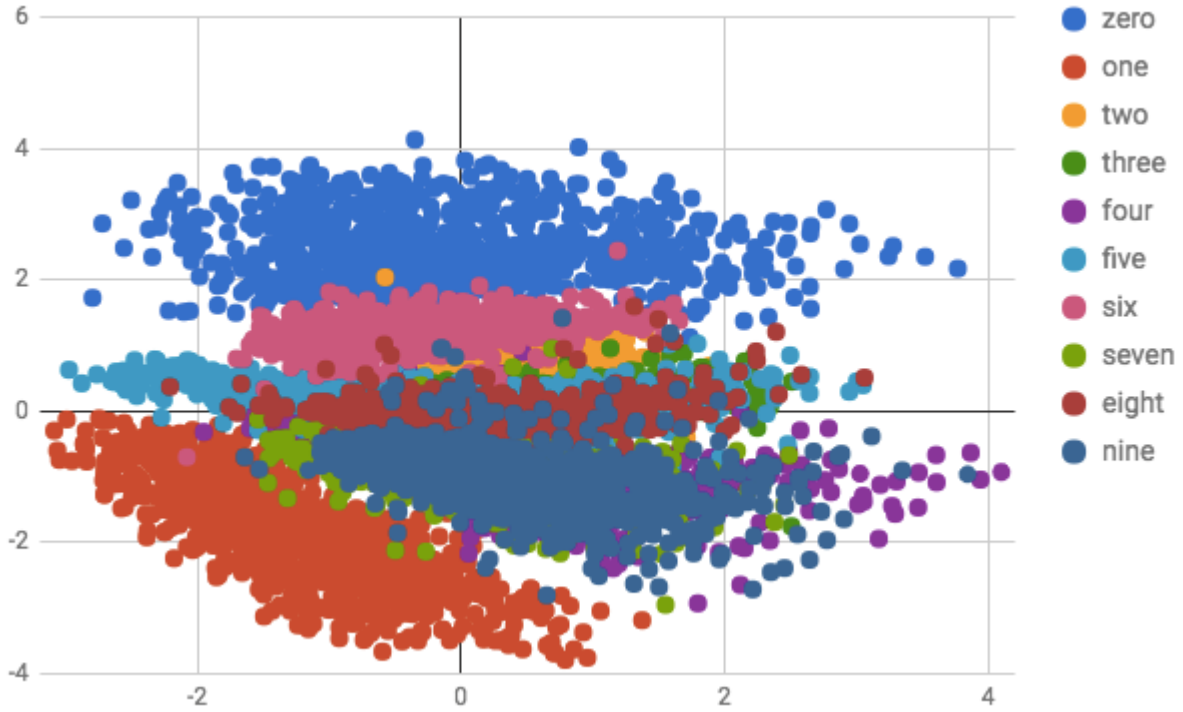
- Higher dimension for latent space usually better captures the data.



2-D 5-D 20-D

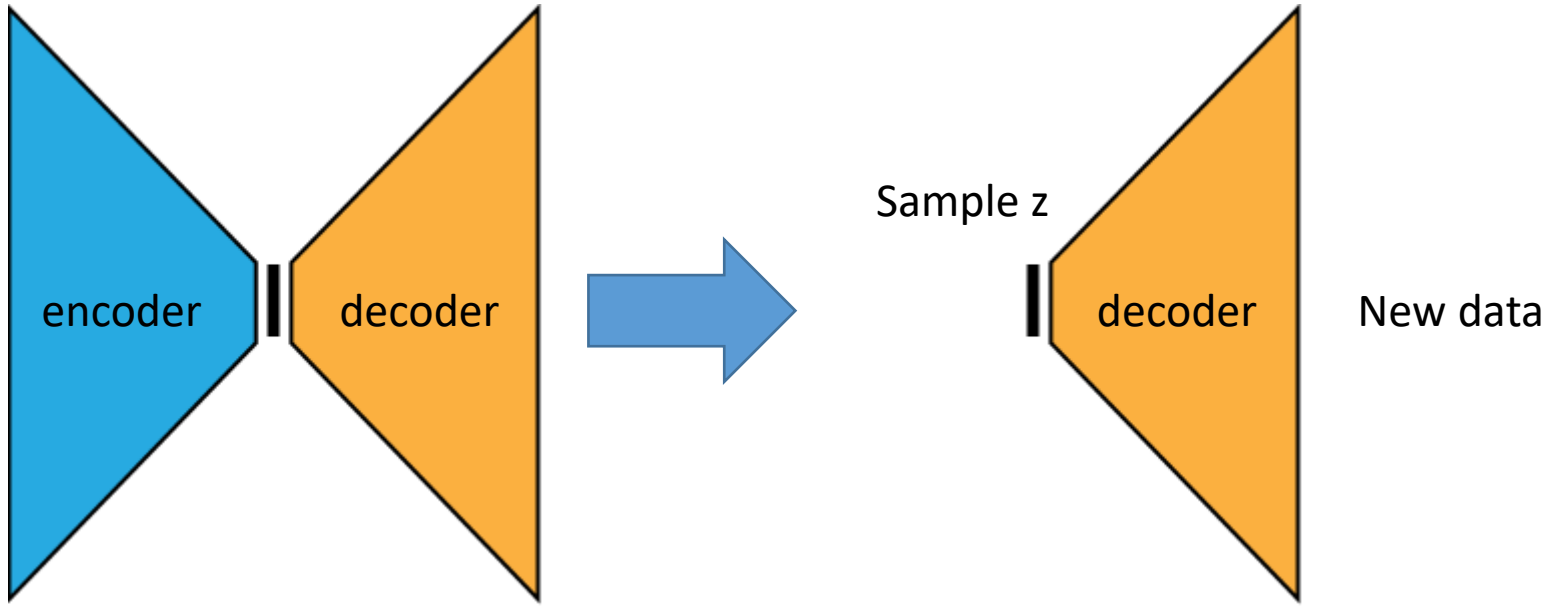
Results

- This is due to the fact that more overlaps appear on the data when dimension is small.



VAE as generative model

- We can just use the decoder, sample Gaussian and generate new data.



VAE as generative model

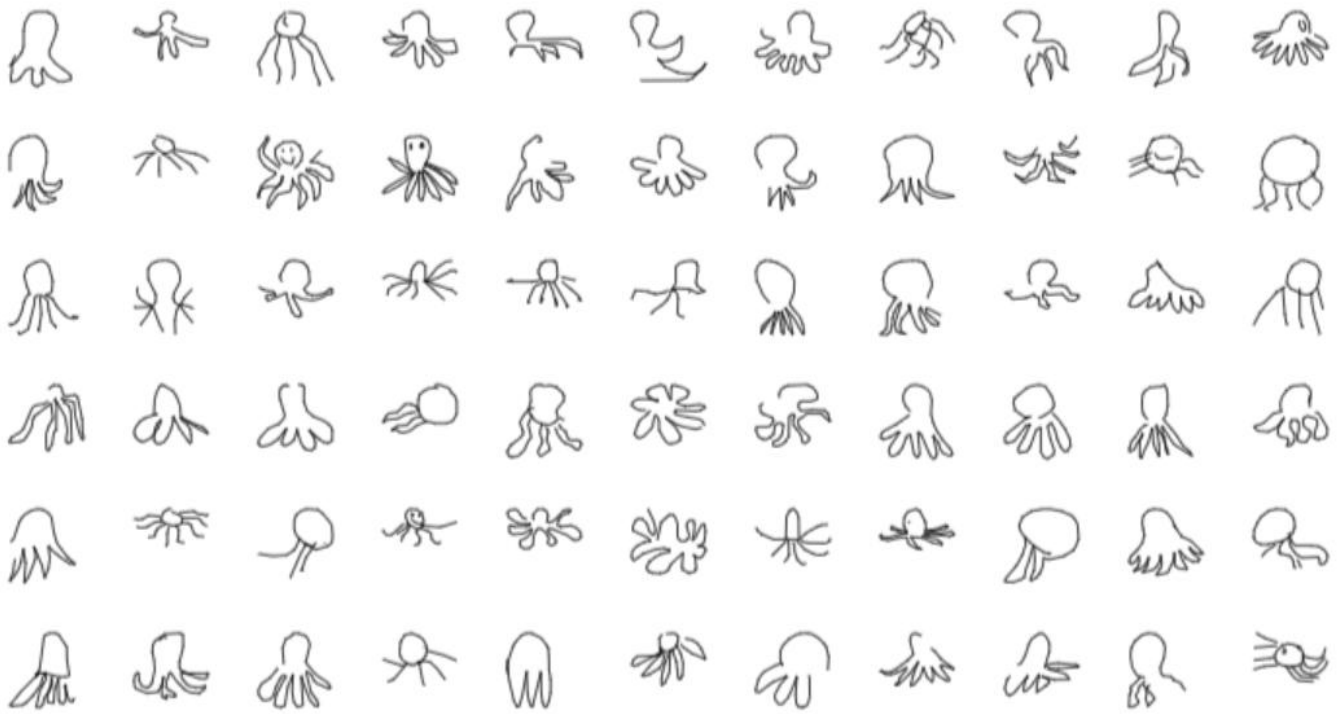
- QuickDraw Data
  - 15 M people played the game

https://quickdraw.withgoogle.com/data

VAE as generative model

- QuickDraw Data
  - 15 M people played the game
  - 345 categories

octopus



VAE as generative model

- QuickDraw Data
  - 15 M people played the game
  - 345 categories

cakes

VAE as generative model

- QuickDraw Data

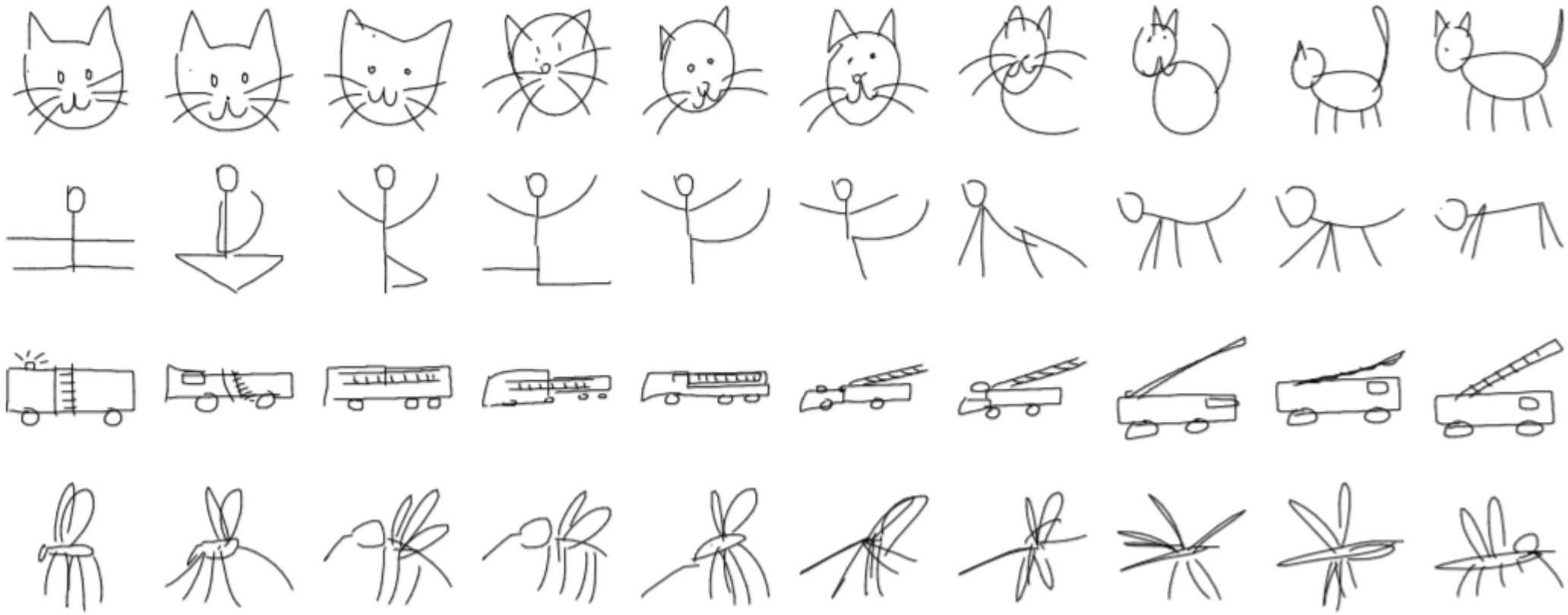
A Neural Representation of Sketch Drawings

David Ha
Google Brain
hadavid@google.com

Douglas Eck
Google Brain
deck@google.com

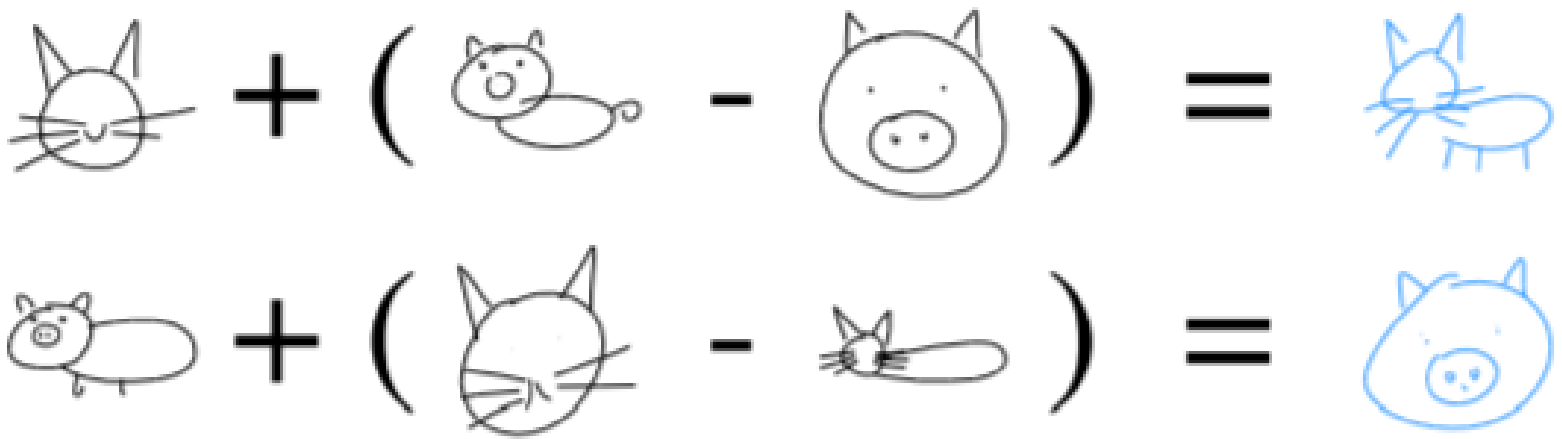
VAE as generative model

- QuickDraw Data



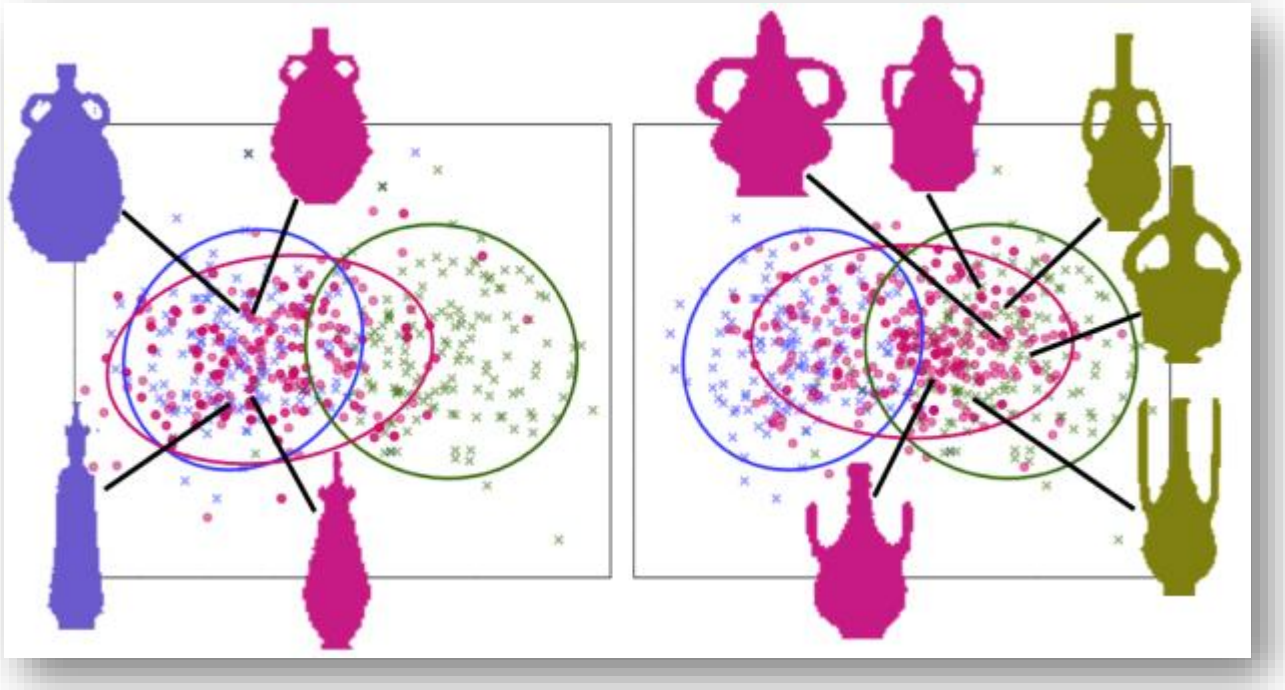
VAE as generative model

- You can explore the latent space and define some arithmetic (not that easy)



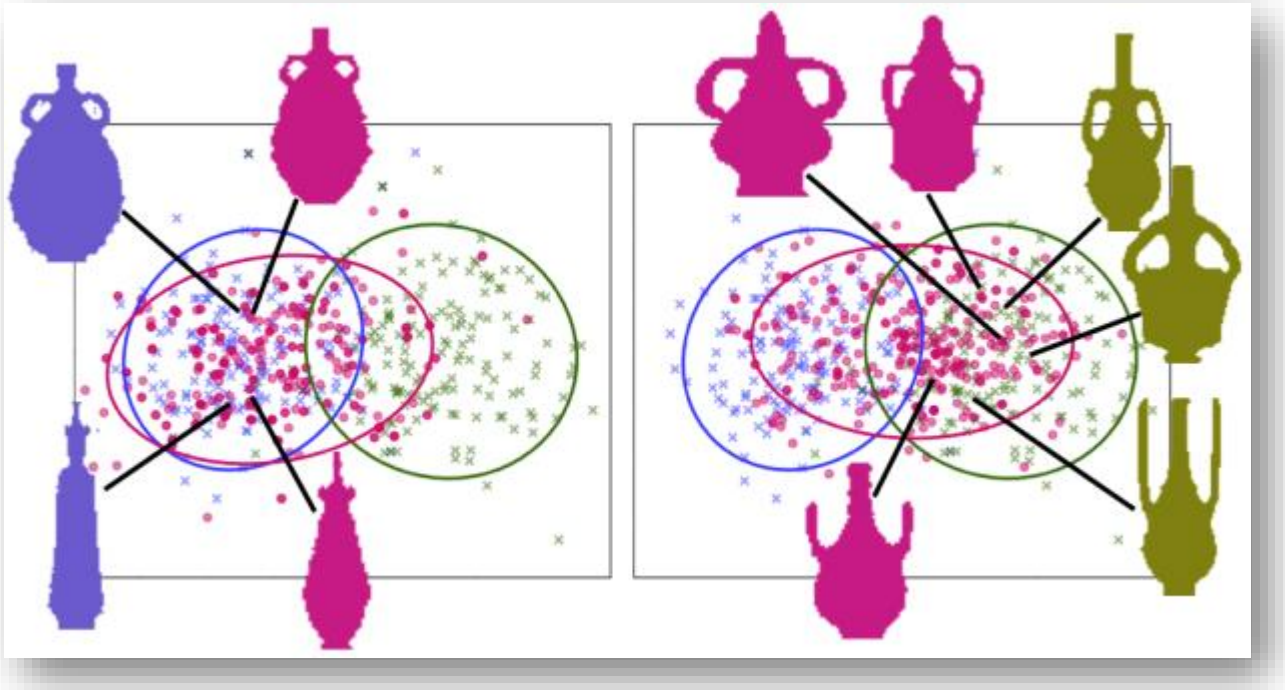
Generative Models

- We discussed VAE as a generative model



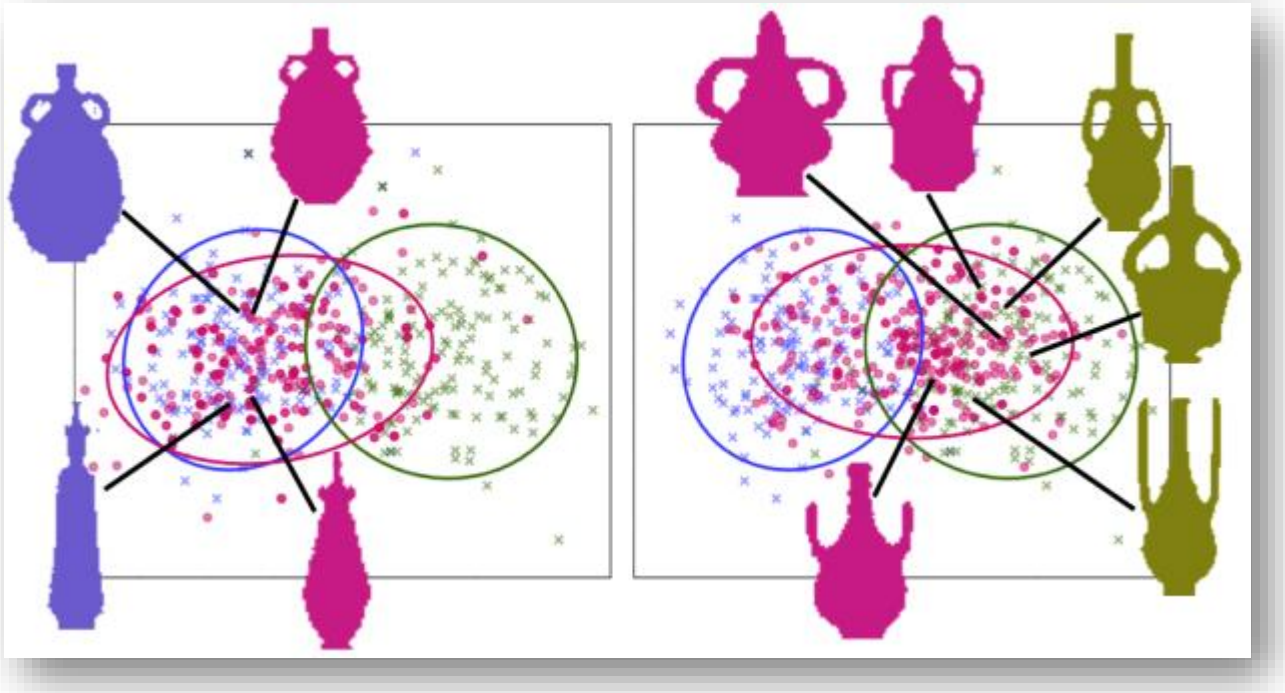
Generative Models

- We learn the distribution of our data somehow.
  - Fit a known distribution to the data distribution.
  - Find a mapping.



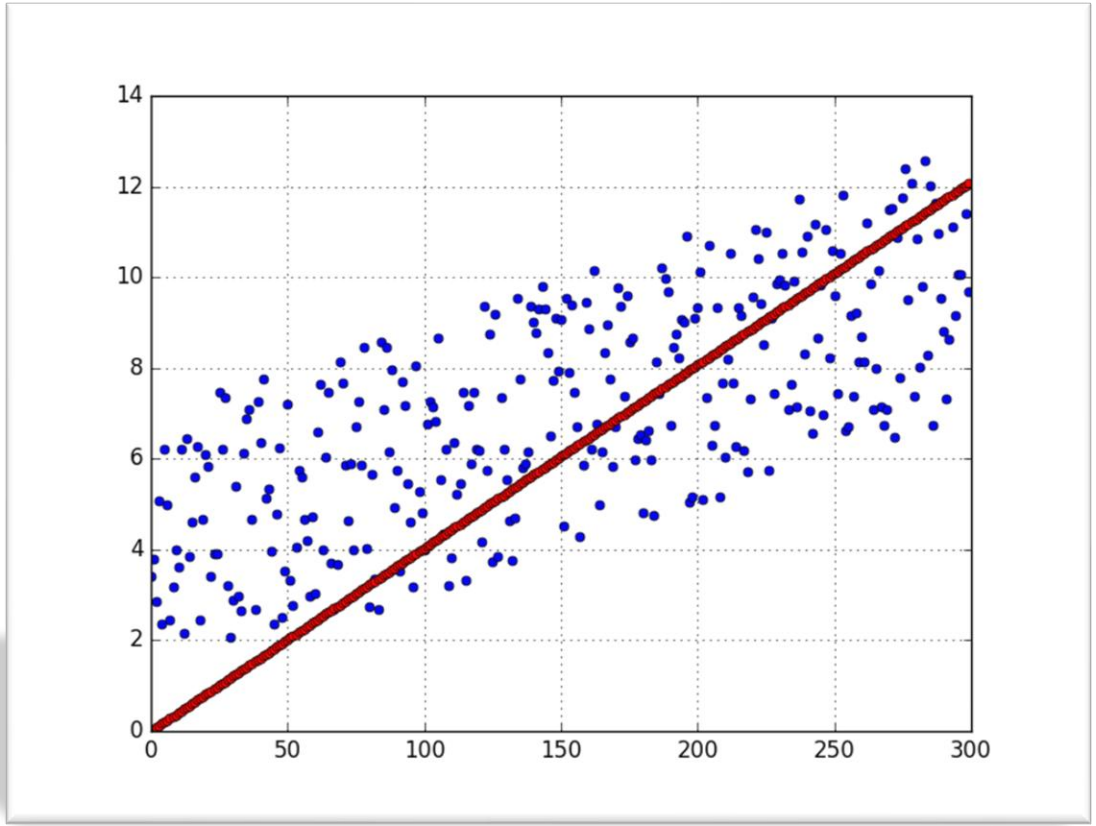
Generative Models

- We then generate a shape/image/data that is not part of our training set.



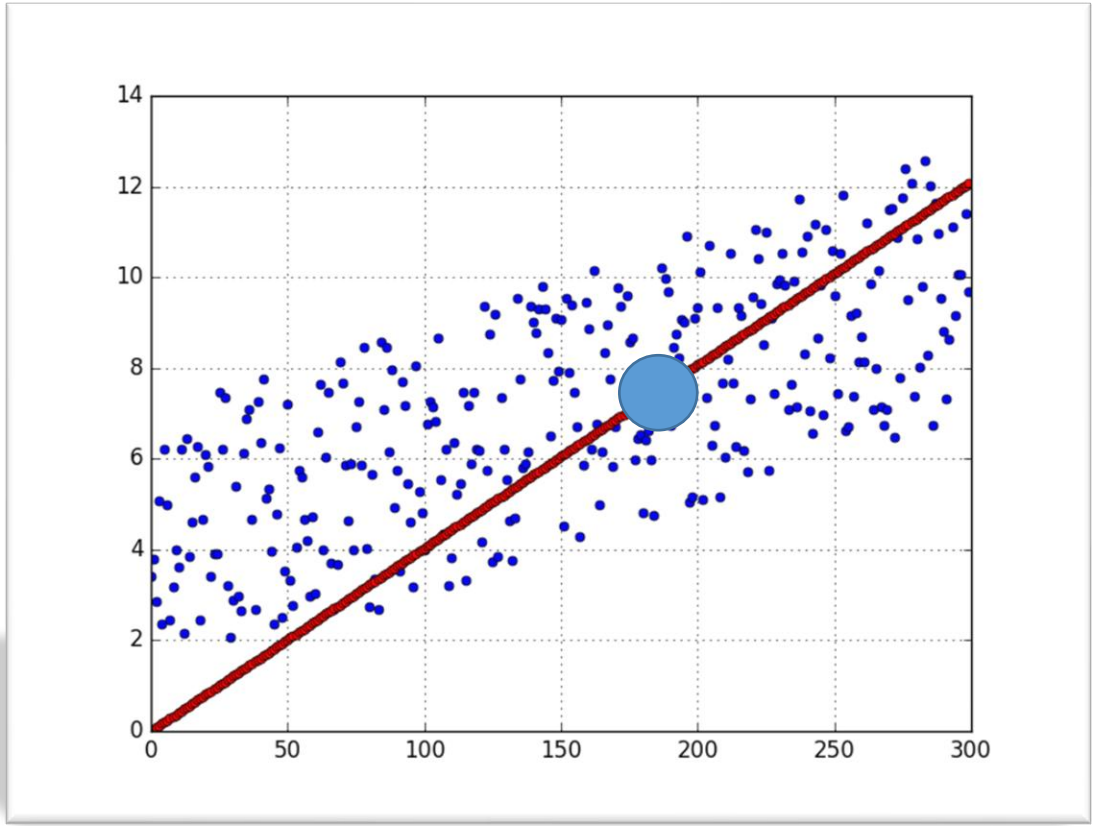
Generative Models

- One simple idea
  - Use a regression model, fit the data,



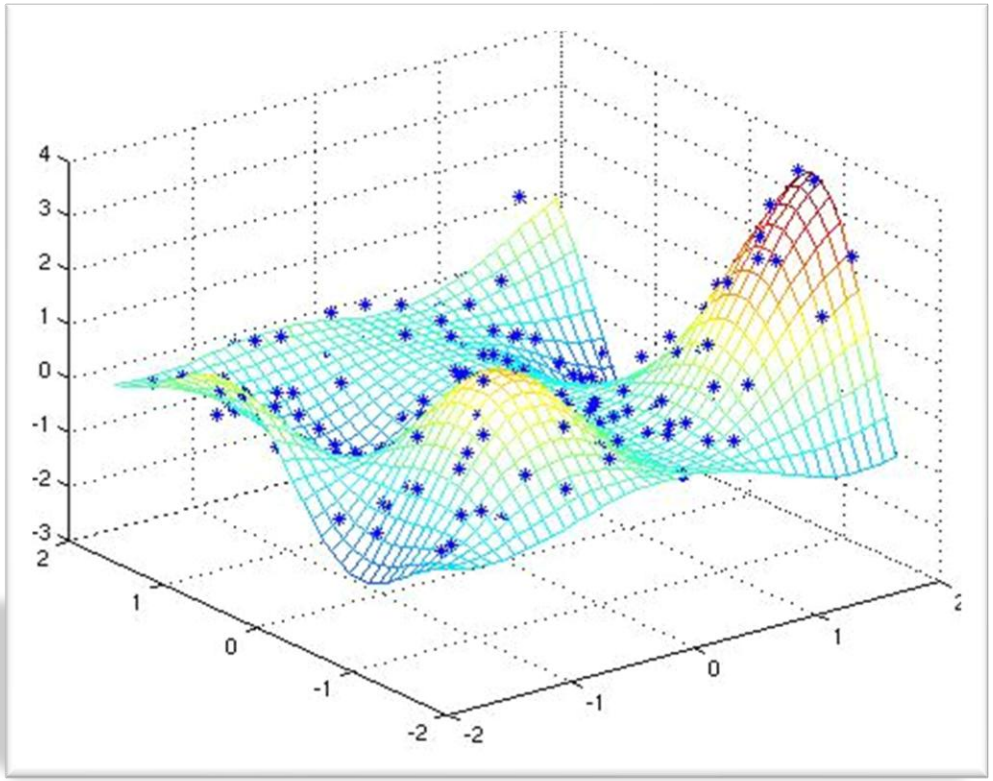
Generative Models

- One simple idea
  - Use a regression model, fit the data, sample model, generate new data.



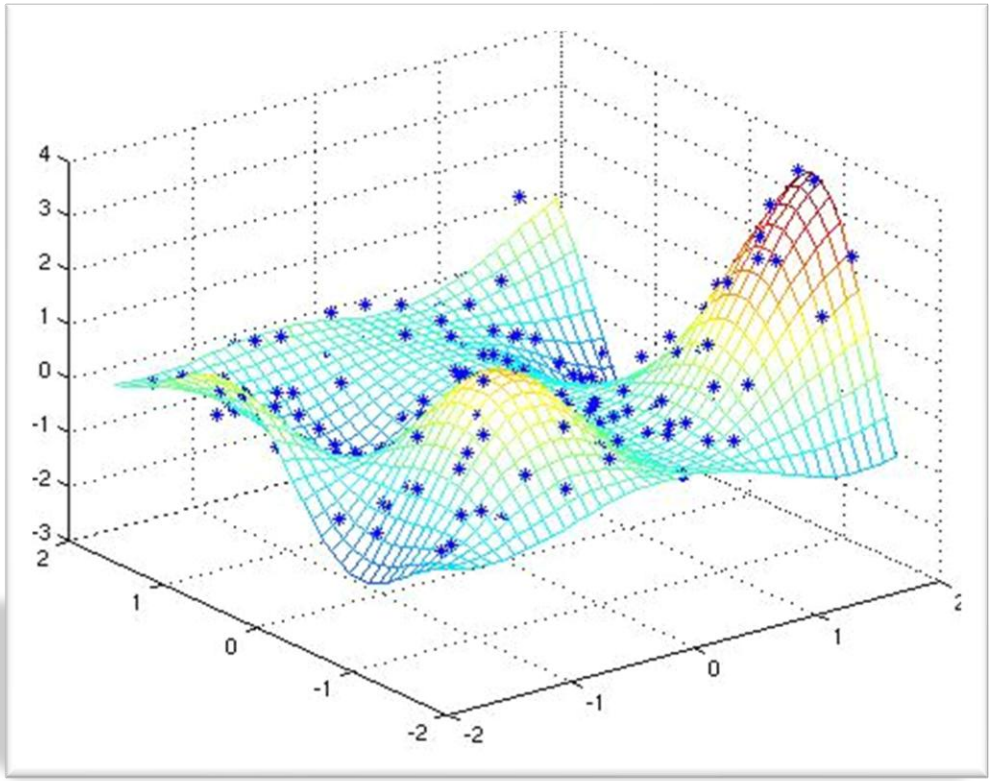
Generative Models

- Higher degree manifolds



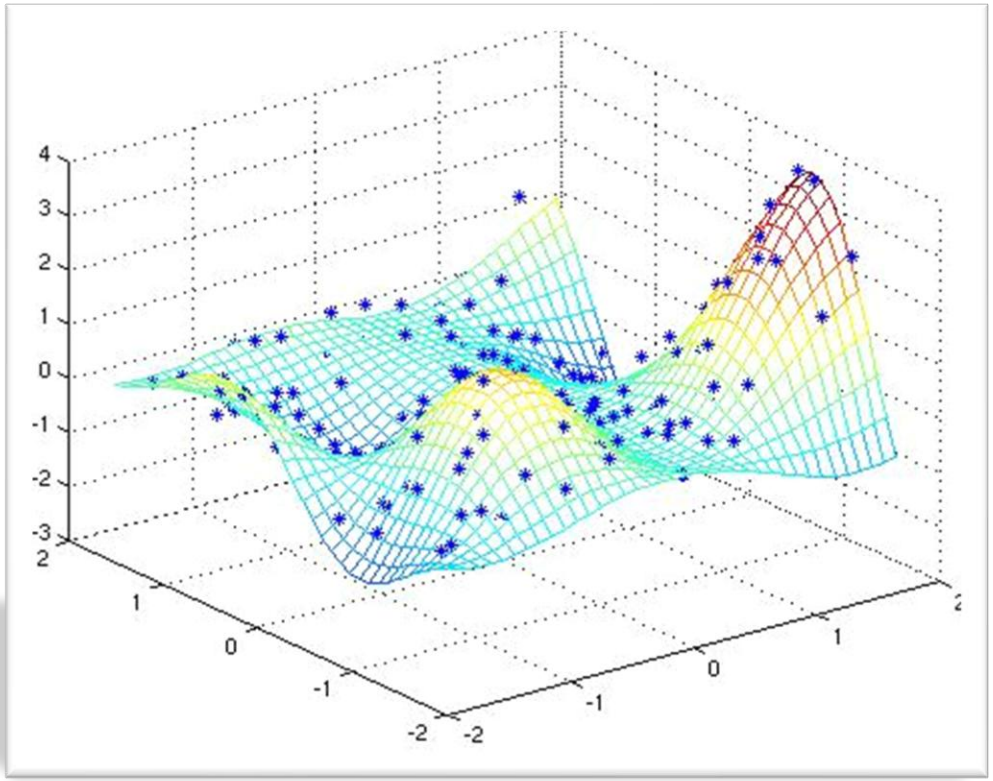
Generative Models

- Not very simple and not possible all the time



Generative Models

- Data is very complex and we cannot fit a predetermined model to it easily.



GAN

- Generative Adversarial Networks
  - Provide a mechanism to generate complex data sets.

GAN

- Ian Goodfellow was the inventor and the person who popularized GAN.

GAN

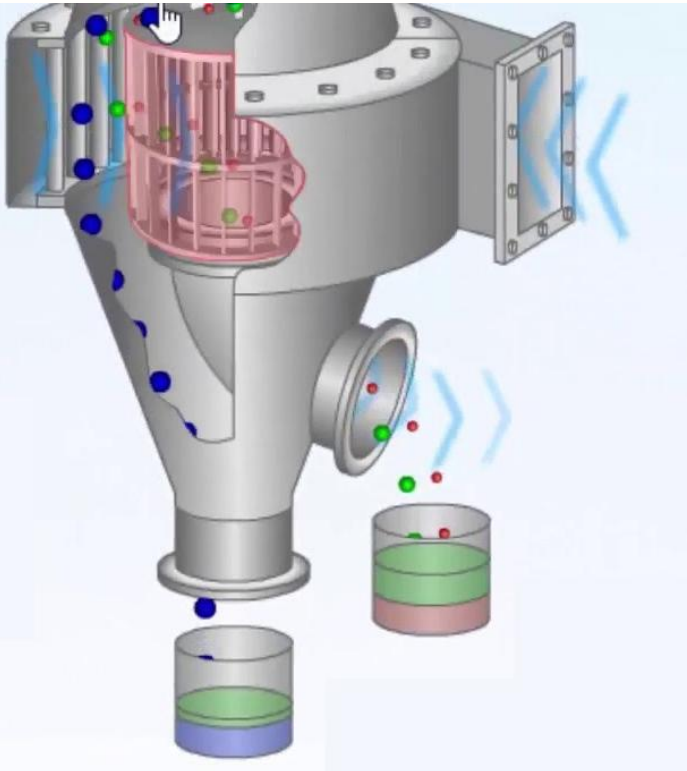
- Bengio: This may hold the key to making computers a lot more intelligent.

GAN

- Lecun: The most important breakthrough, in my opinion, is adversarial training (also called GAN). This is the most interesting idea in the last 10 years in ML, in my opinion.

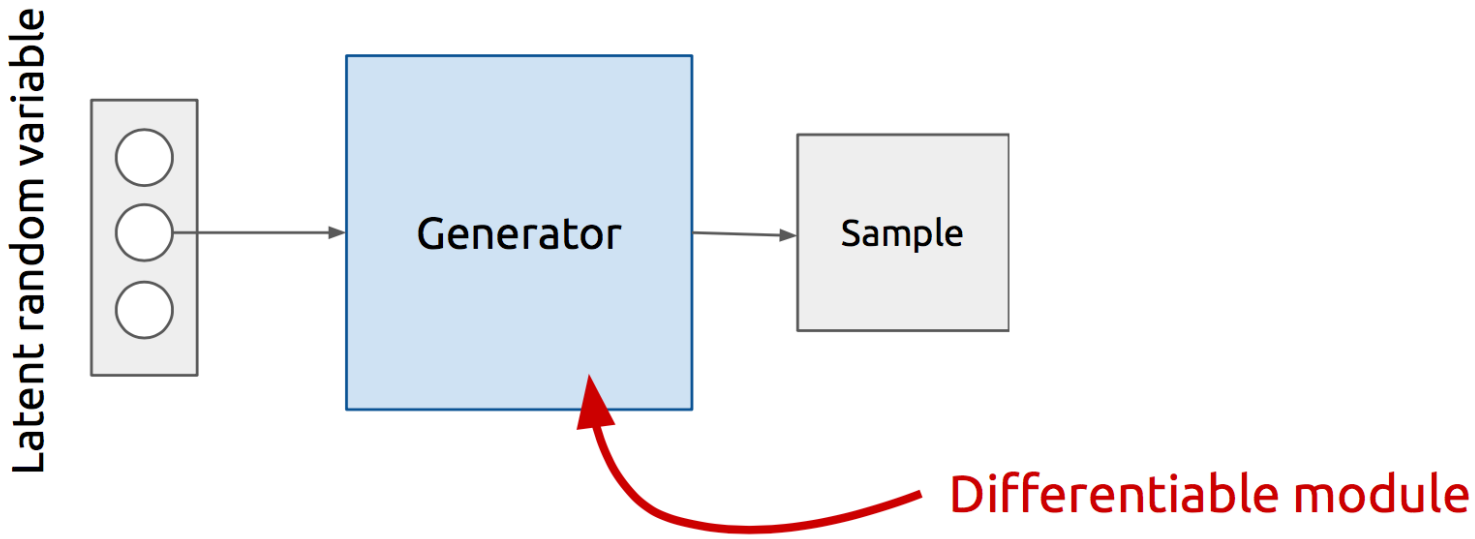
GAN

- Generative Adversarial Networks
  - Generator
  - Discriminator



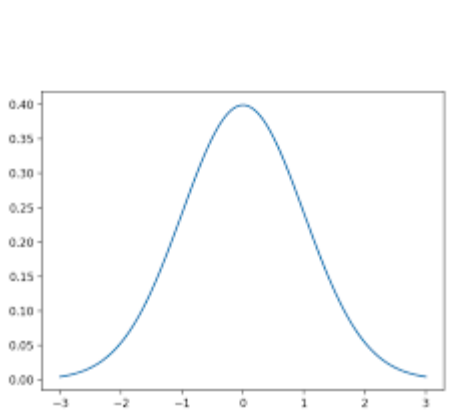
GAN

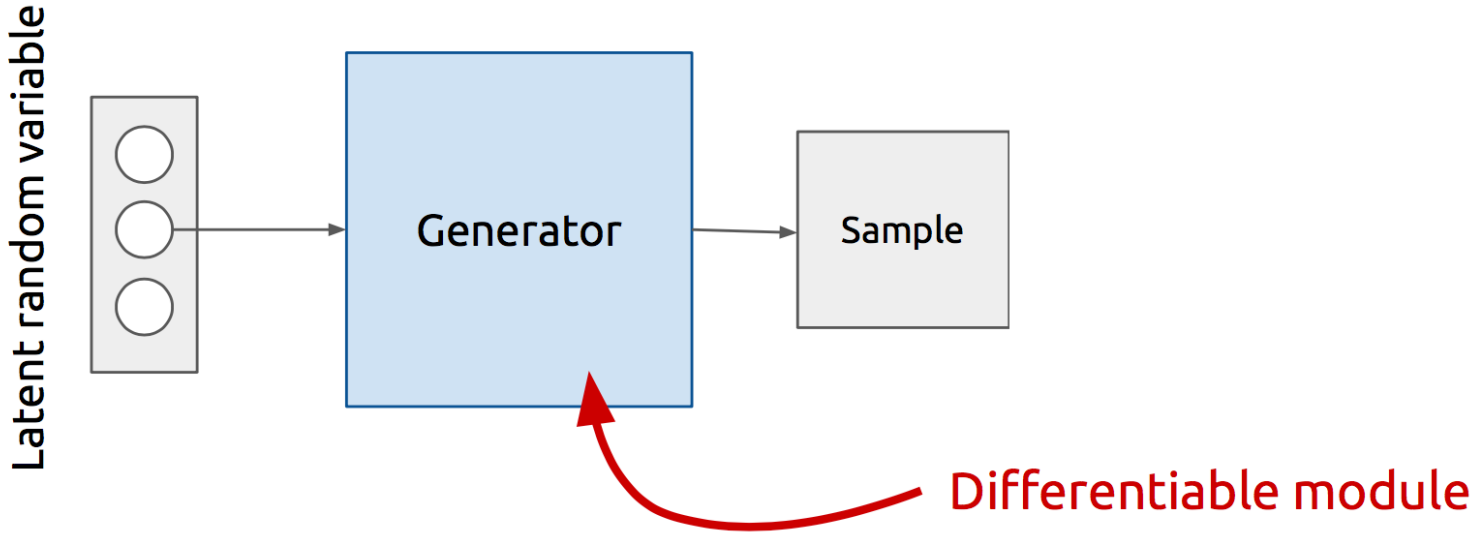
- Generator
  - Deep Network



GAN

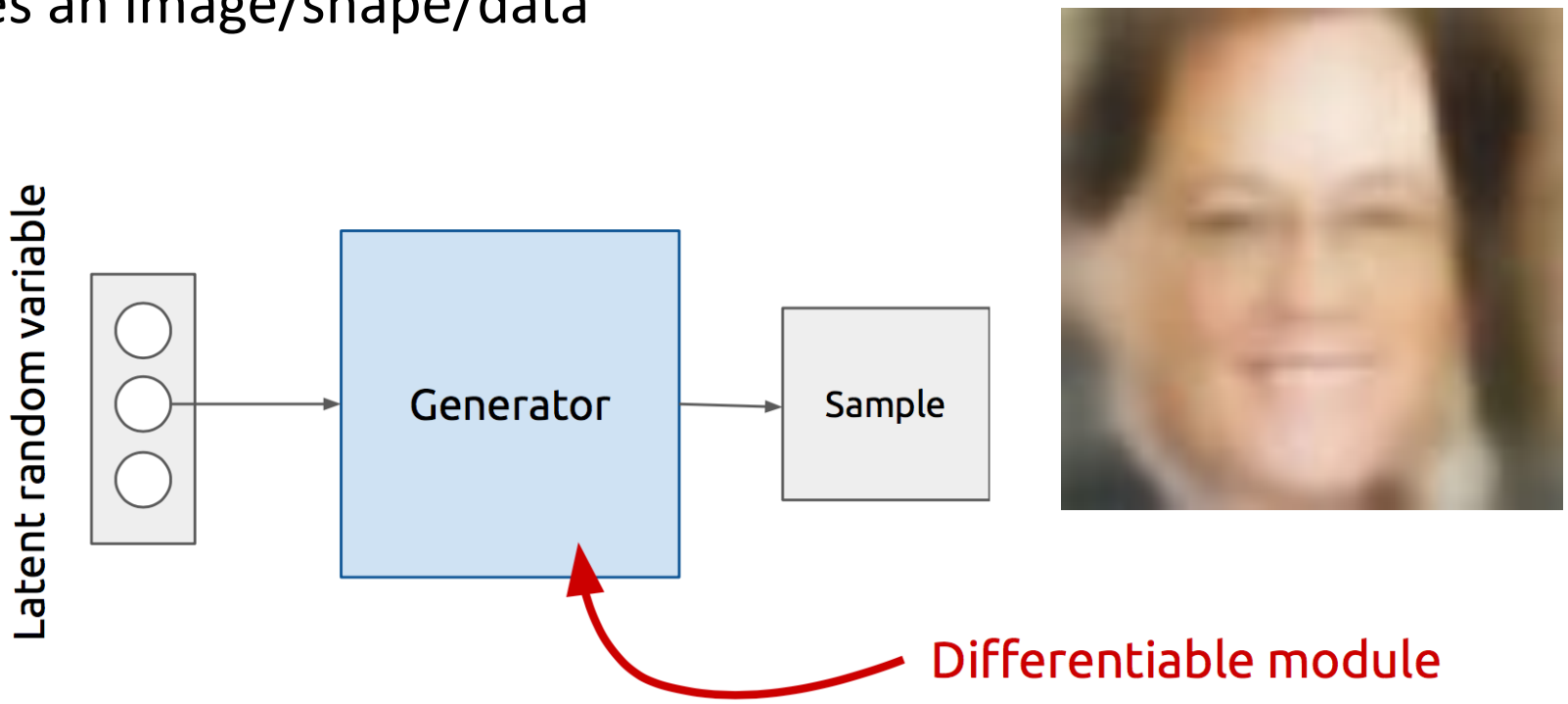
- Generator
  - Deep Network
  - Receives a sample from a Gaussian distribution





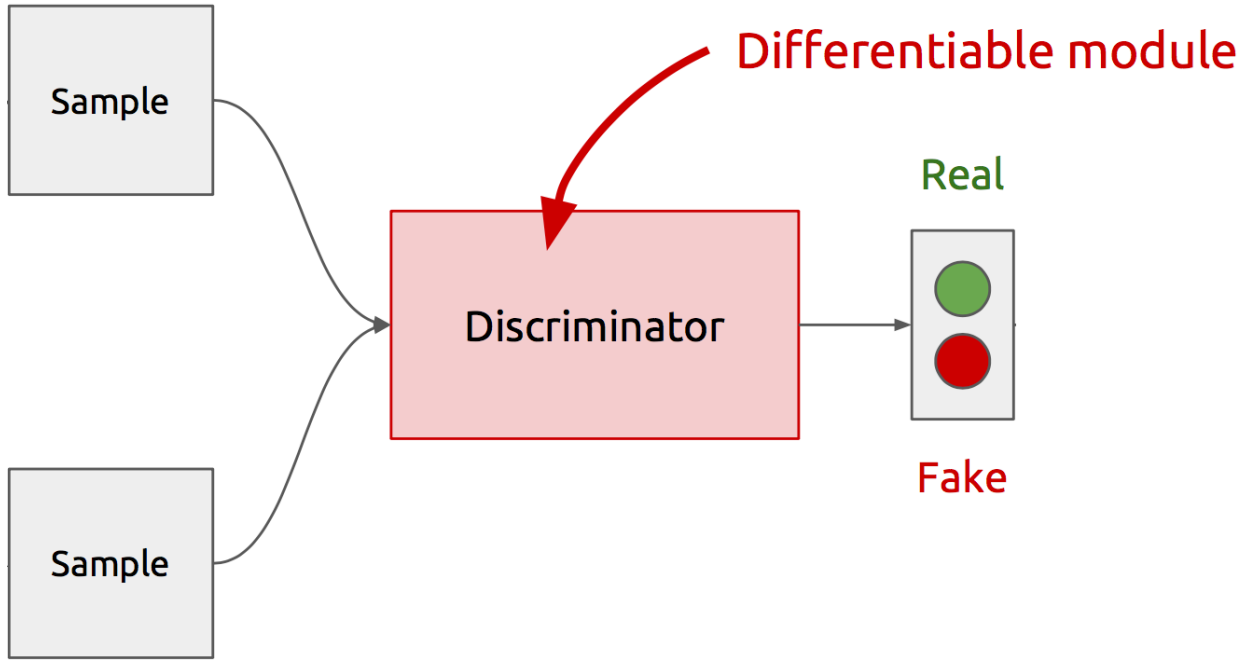
GAN

- Generator
  - Deep Network
  - Receives a sample from a Gaussian distribution
  - Produces an image/shape/data



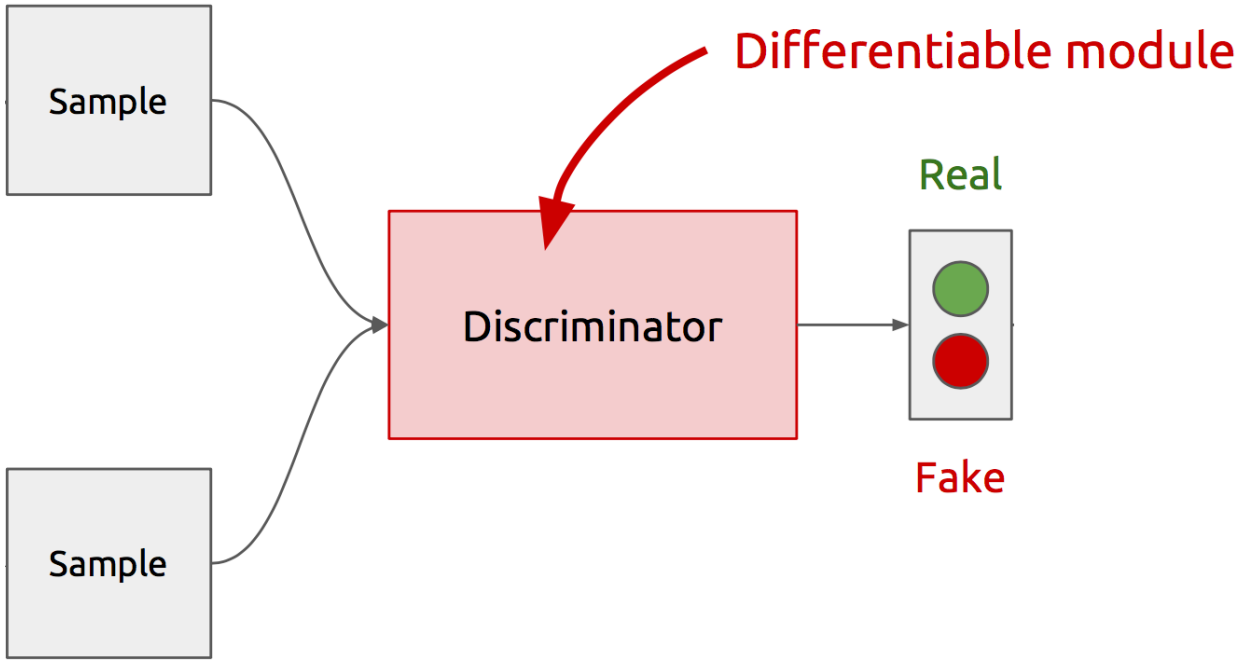
GAN

- Discriminator
  - It is a classifier (fake/real)



GAN

- Discriminator
  - It is a classifier
  - Receives real and fake images and learns to classifies them as fake or real

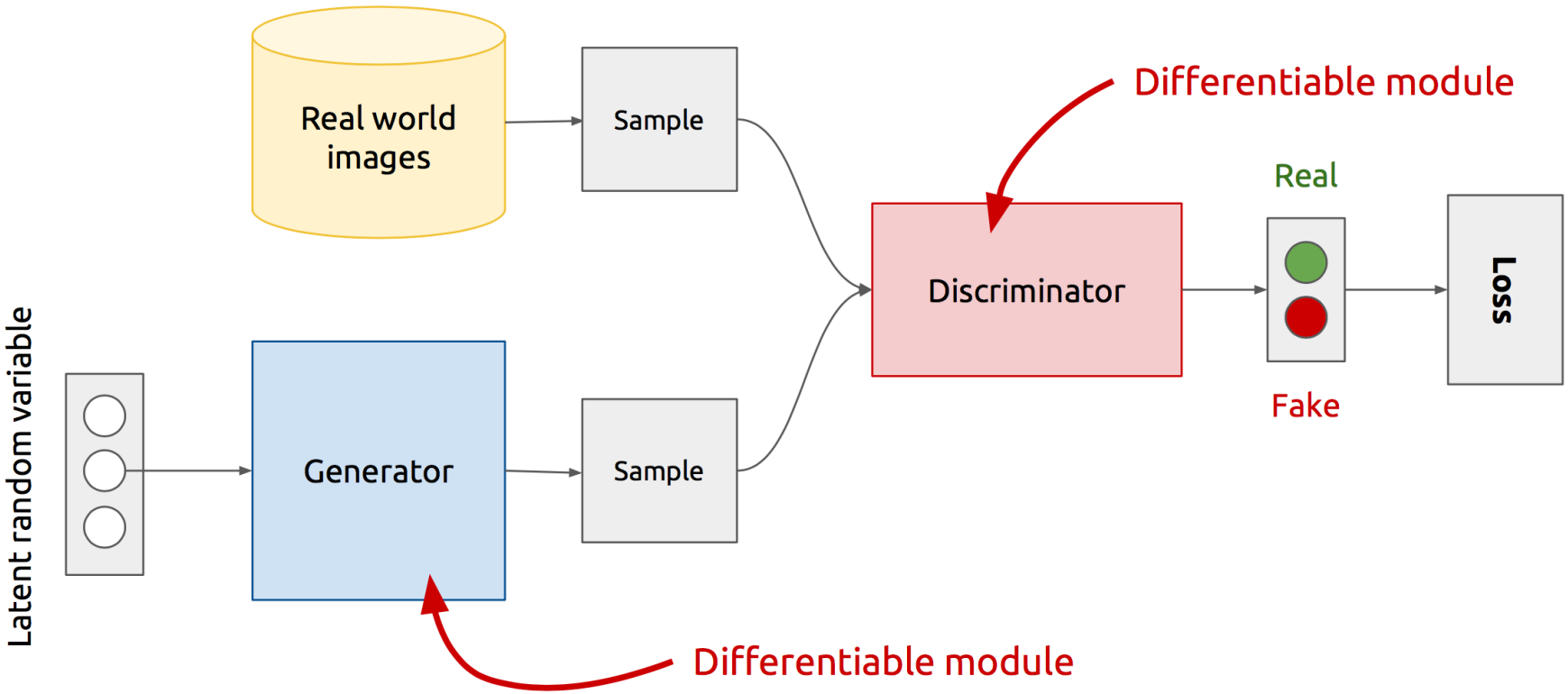


GAN

- Discriminator
  - Quality Control

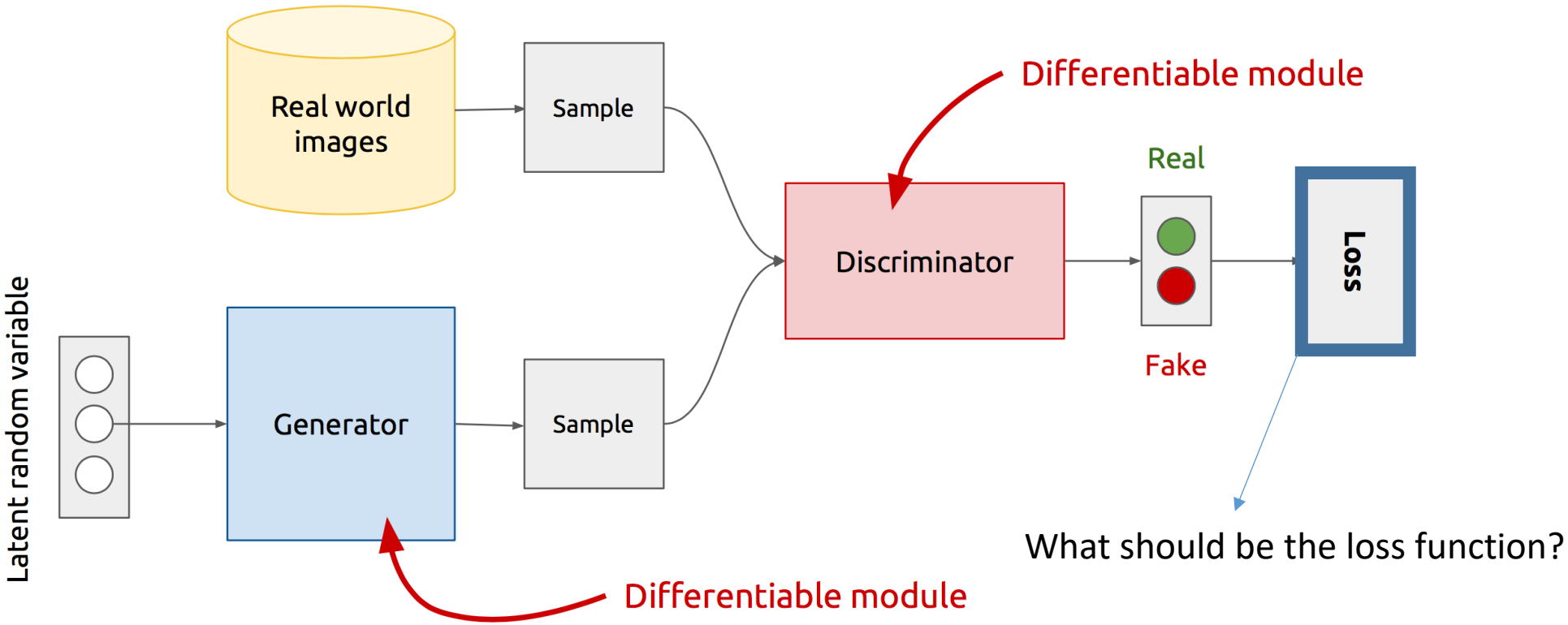
GAN

- Model all together



GAN

- Model all together



GAN

- Loss Function

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

GAN Loss Function

- Loss Function

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

Discriminator Generator

GAN Loss Function

- Loss Function

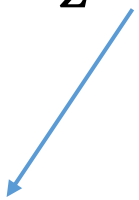
$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$

This means x is sampled from the actual data (real)

GAN Loss Function

- Loss Function

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

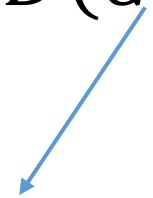


This means z is sampled from the noise

GAN Loss Function

- Loss Function

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



This means $G(z)$ is a fake image

GAN Loss Function

- D tries to maximize this equation, what does it mean?

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

GAN Loss Function

- D tries to maximize this equation, what does it mean?

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

- This means $\log D(x)$ and $\log(1 - D(G(z)))$ must be high and therefore $D(x)$ must be one and $D(G(z))$ must be zero.

GAN Loss Function

- D tries to maximize this equation, what does it mean?

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

- This means $\log D(x)$ and $\log(1 - D(G(z)))$ must be high and therefore $D(x)$ must be one and $D(G(z))$ must be zero.
- Classifier needs to assign true labels to the fed data.

GAN Loss Function

- G tries to minimize this equation, what does it mean?

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

GAN Loss Function

- G tries to minimize this equation, what does it mean?

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

Has no G term

GAN Loss Function

- G tries to minimize this equation, what does it mean?

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

- $\log\left(1 - D\left(G(z)\right)\right) = -\infty$

GAN Loss Function

- G tries to minimize this equation, what does it mean?

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

- $\log\big(1 - D\big(G(z)\big)\big) = -\infty \rightarrow 1 - D\big(G(z)\big) = 0$

GAN Loss Function

- G tries to minimize this equation, what does it mean?

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

- $\log\big(1 - D\big(G(z)\big)\big) = -\infty \rightarrow 1 - D\big(G(z)\big) = 0 \rightarrow D\big(G(z)\big) = 1$

GAN Loss Function

- G tries to minimize this equation, what does it mean?

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

- $\log\left(1 - D\left(G(z)\right)\right) = -\infty \rightarrow 1 - D\left(G(z)\right) = 0 \rightarrow D\left(G(z)\right) = 1$
- Generator needs to fool the discriminator

GAN Training

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:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

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 descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

GAN Training

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:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

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 descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

First train the discriminator, fix generator

GAN Training

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:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

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 descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

Fix the discriminator, train generator

GAN Training

Start over

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:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

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 descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

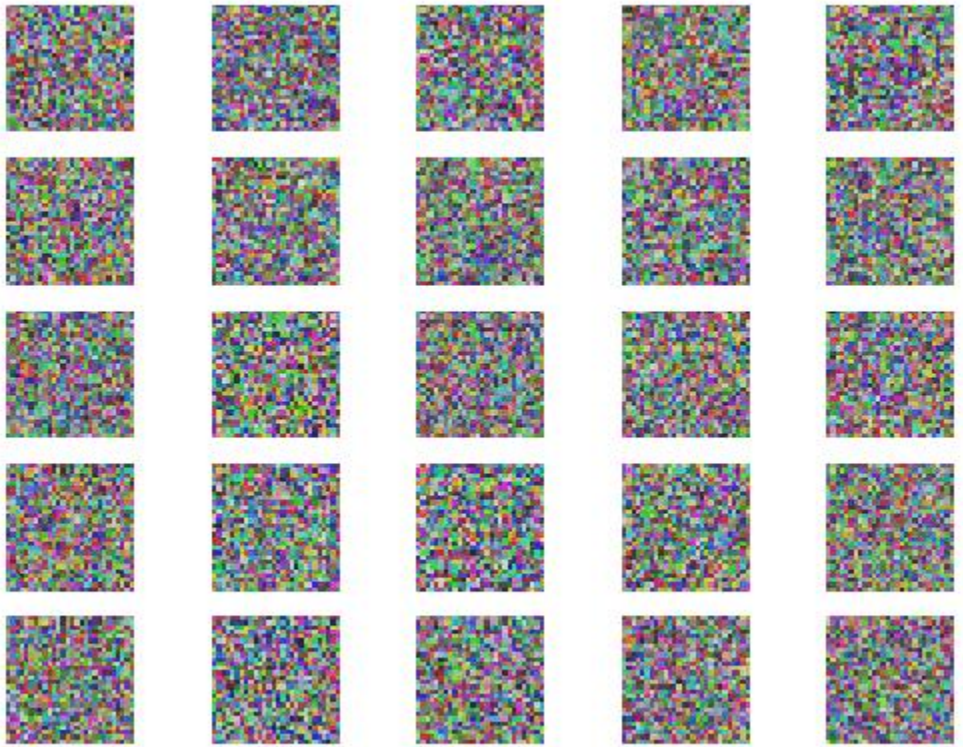
Distribution Losses Review

- Jensen-Shannon Divergence Loss is a tweaked version of KL-Divergence to make the distribution differences symmetric:

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

- GAN optimizes Jenson-Shannon loss.

GAN Results

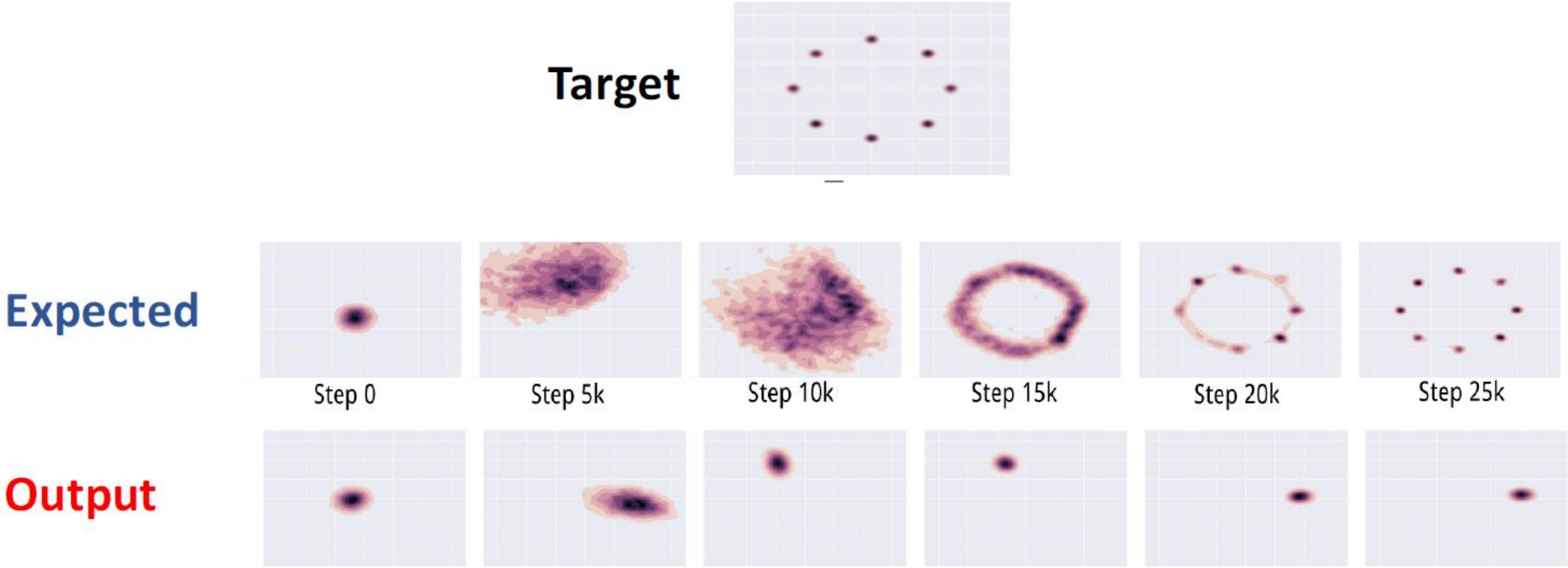


GAN Disadvantages

- Mode collapse.
- Discriminator is not useful after training.
- Very hard to train.

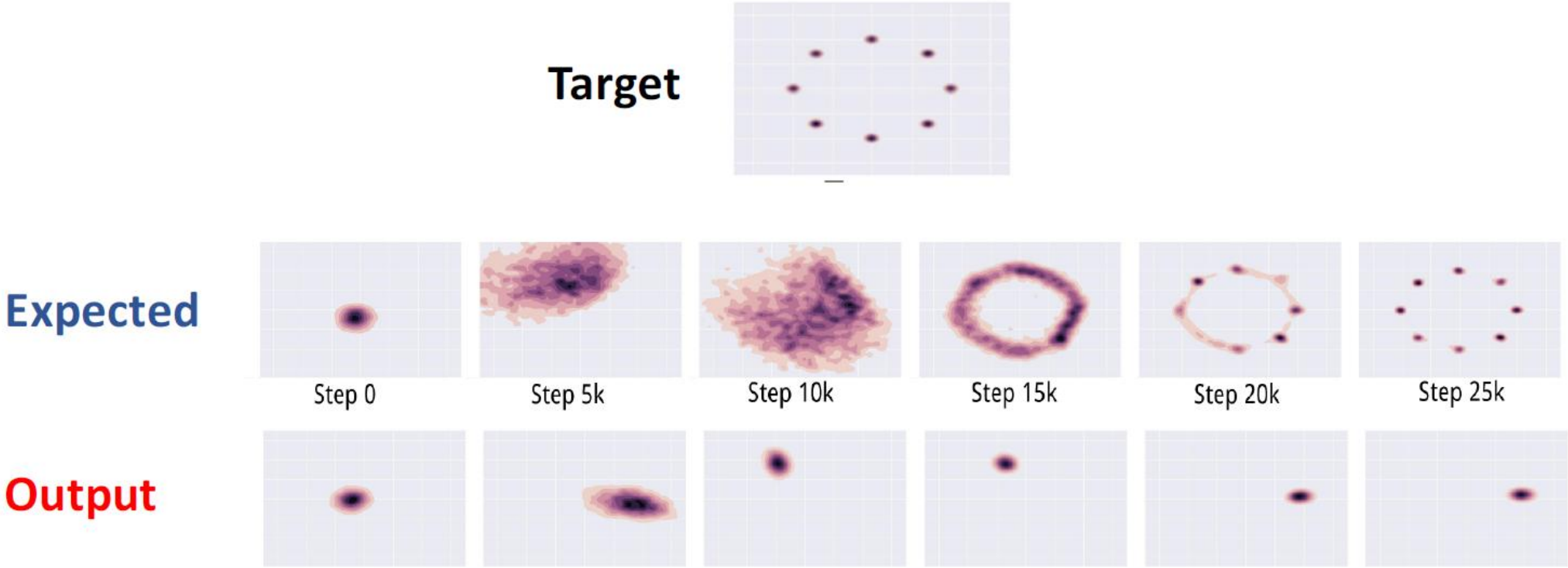
Mode Collapse

- Generator fails to output diverse samples



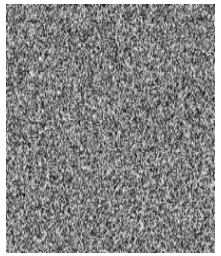
Mode Collapse

- Generator produces good samples but very few of them and discriminator cannot assign a fake label to it.



Discriminator After Training

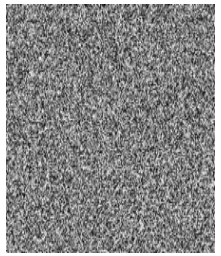
- Discriminator is usually useless after training in our experiments



~ 0.5

Discriminator After Training

- Discriminator is usually useless after training in our experiments



~ 0.5

- It somehow learns that for whatever input, ~0.5 results the best.

Training

- Since the given prior is a sample from Gaussian noise results of the original GAN are noisy.

GAN Disadvantages

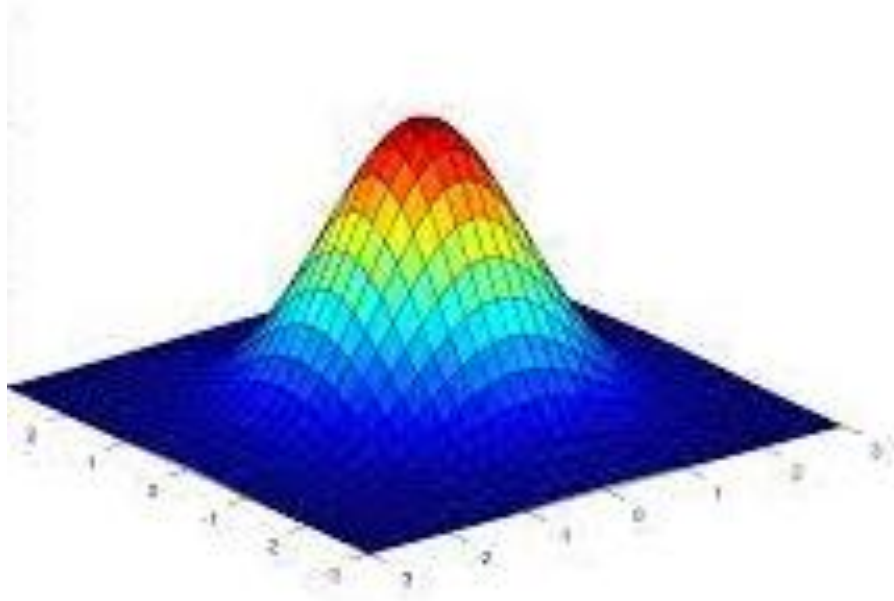
- Very hard to train.

GAN Disadvantages

- Very hard to train.

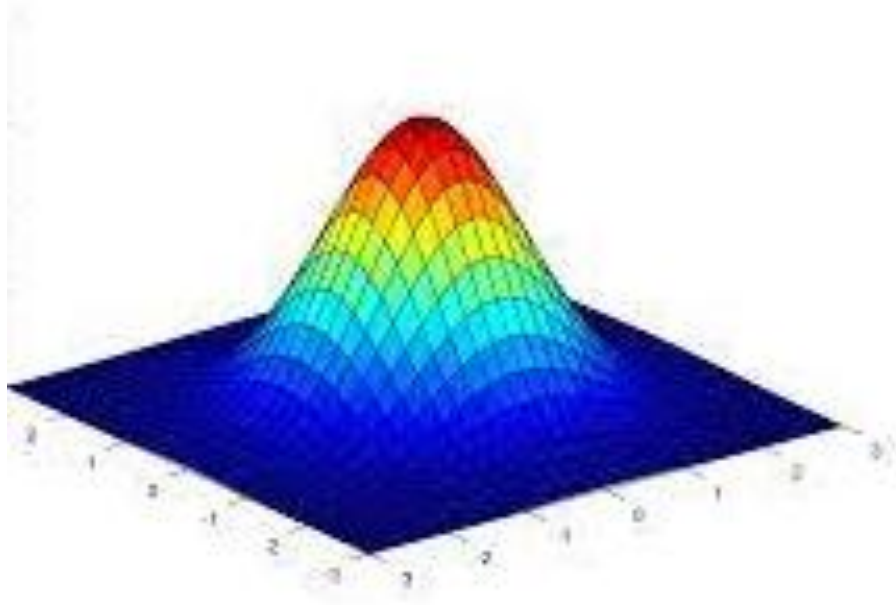
StyleGAN

- What is the input to a GAN?
  - Gaussian noise



StyleGAN

- What is the input to a GAN?
  - Gaussian noise
  - It’s a smooth distribution



StyleGAN

- What is the input to a GAN?
  - Gaussian noise
  - It’s a smooth distribution

- What is data distribution?

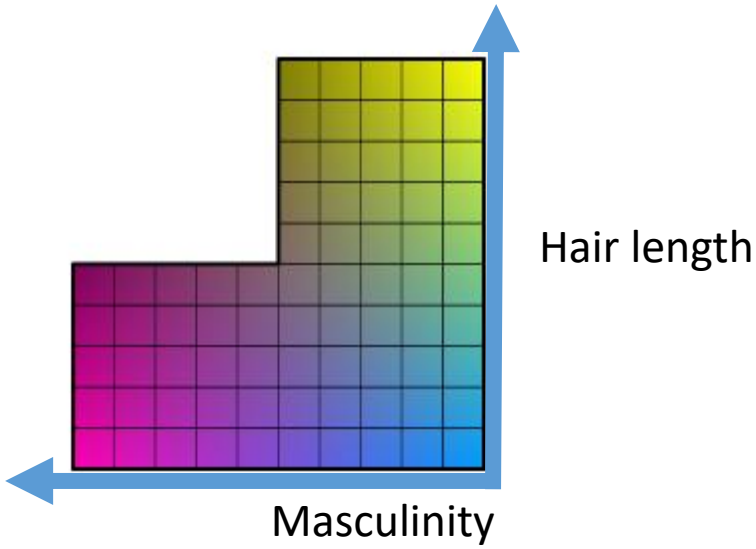
StyleGAN

- What is the input to a GAN?
  - Gaussian noise
  - It’s a smooth distribution

- What is data distribution?
  - An unknown and possibly highly skewed, maybe fragmented data distribution

StyleGAN

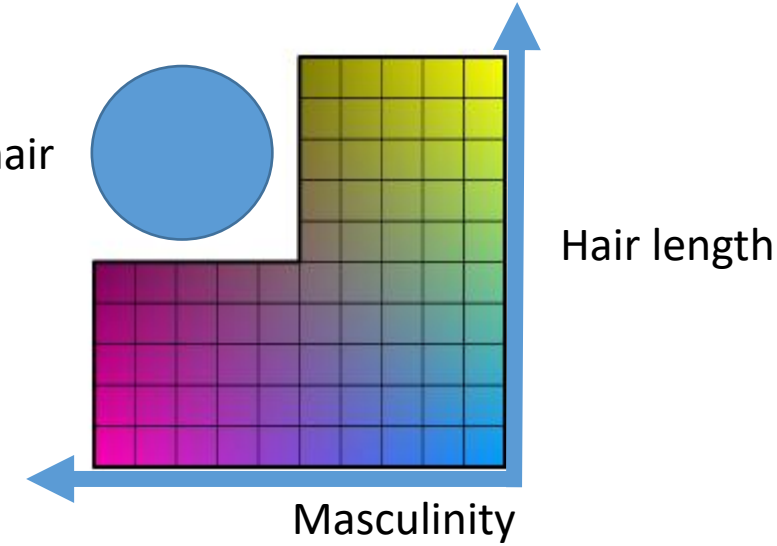
- Example: Masculinity vs hair length



StyleGAN

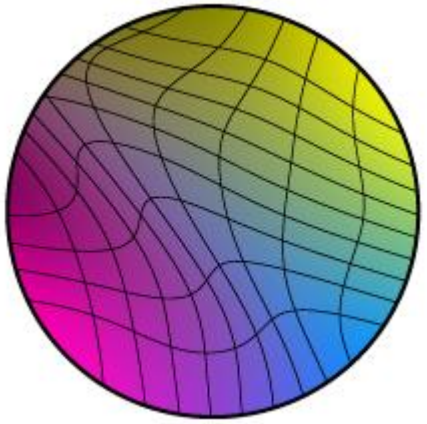
- Example: Masculinity vs hair length

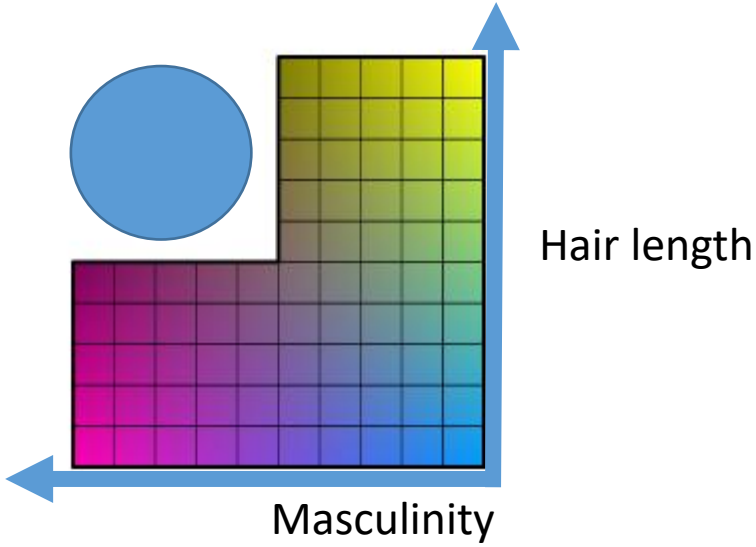
Very few data points for men with long hair



StyleGAN

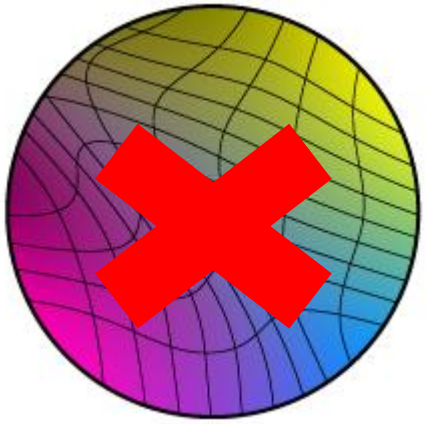
- If you force a Gaussian distribution to represent this distribution, you will get a skewed and distorted distribution.

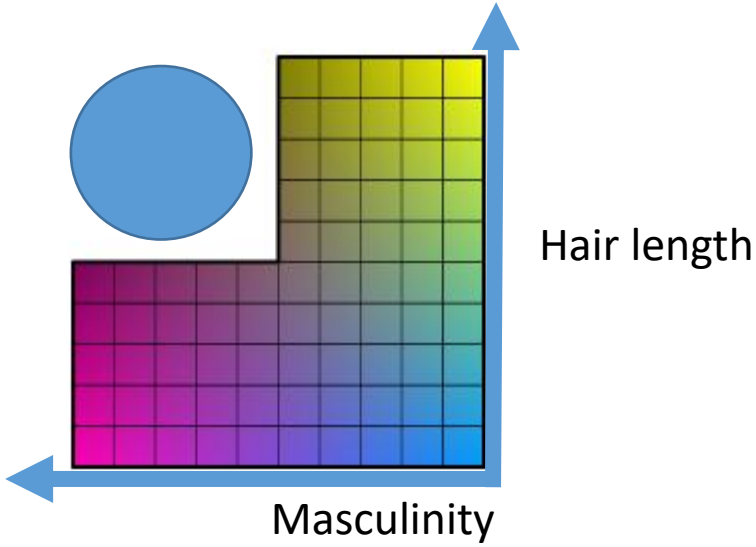




StyleGAN

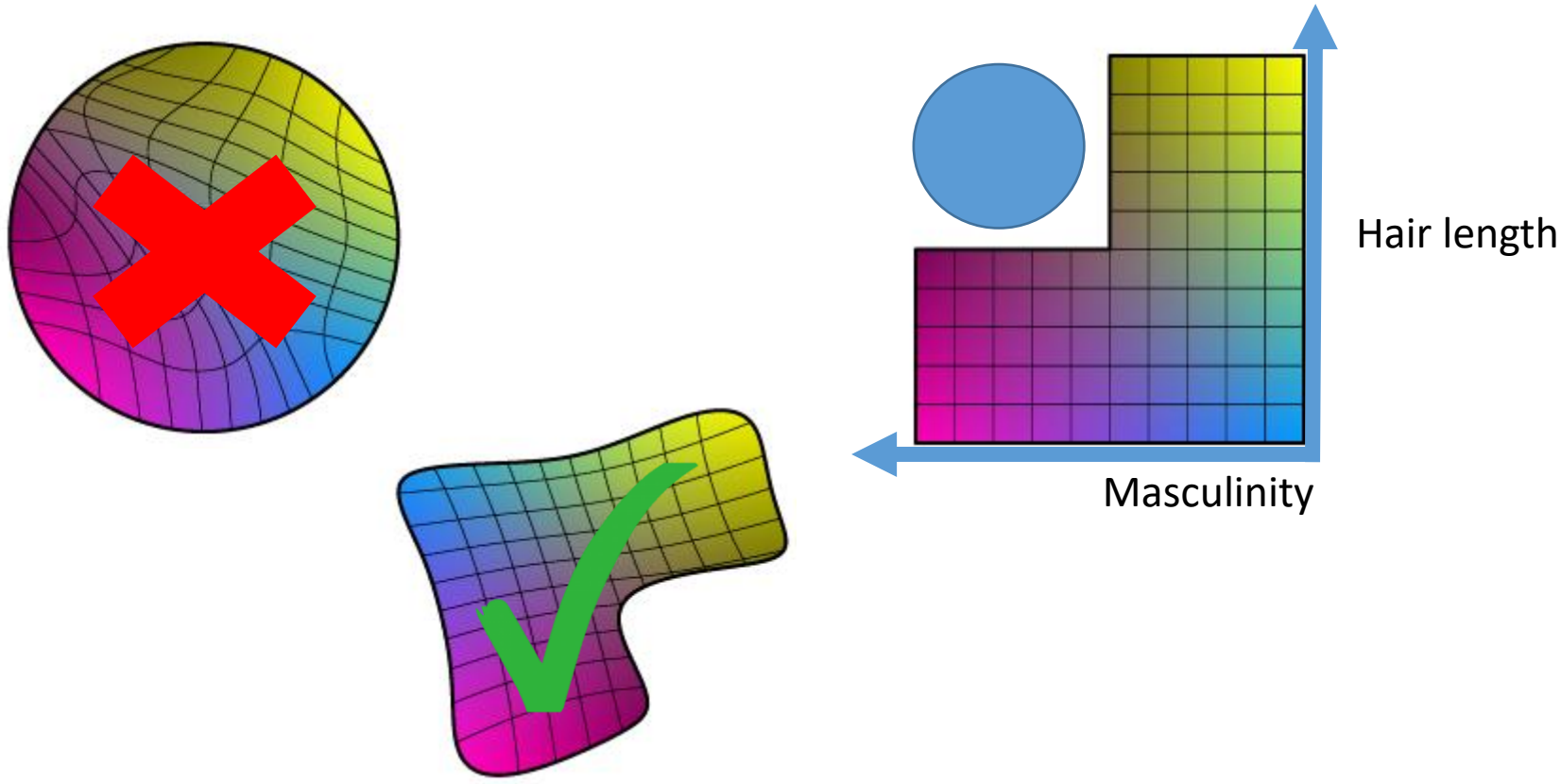
- We don’t want this





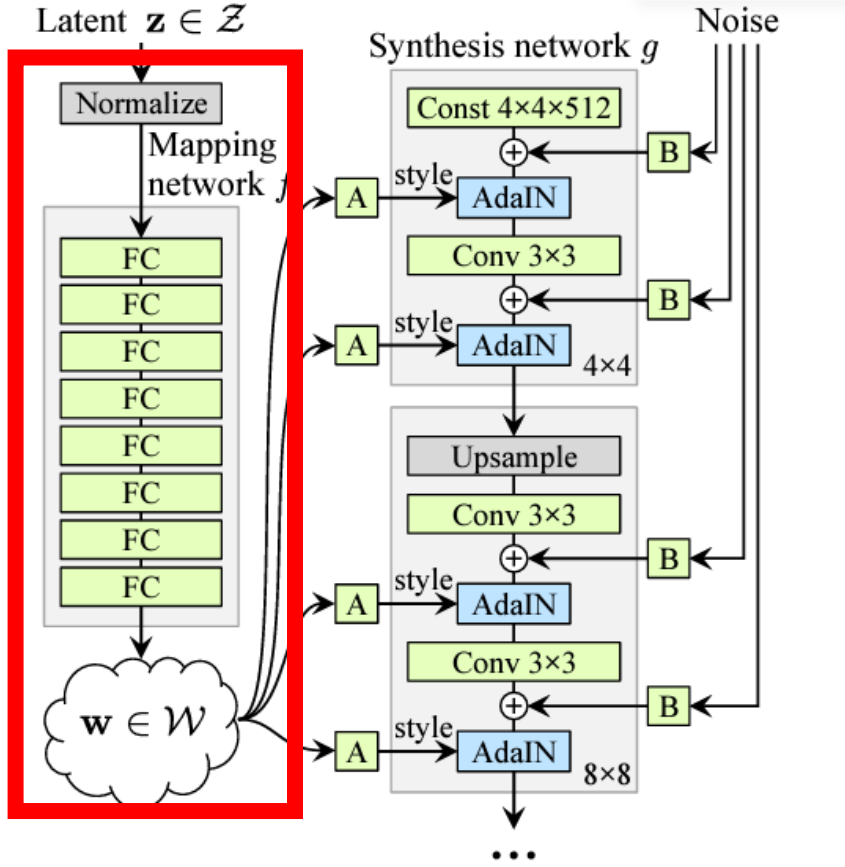
StyleGAN

- Instead we want to learn a mapping that better represent the data distribution.



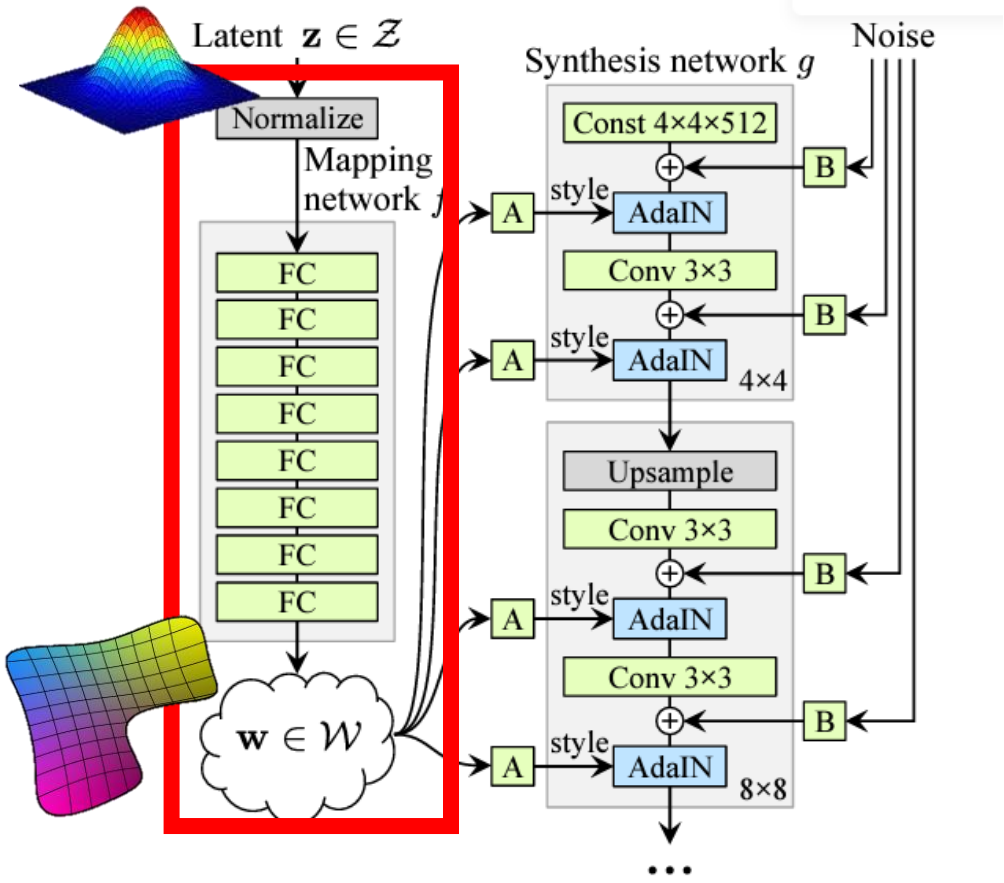
StyleGAN

- StyleGAN uses an intermediate mapper network is used to avoid mapping Gaussian noise to the data distribution directly.



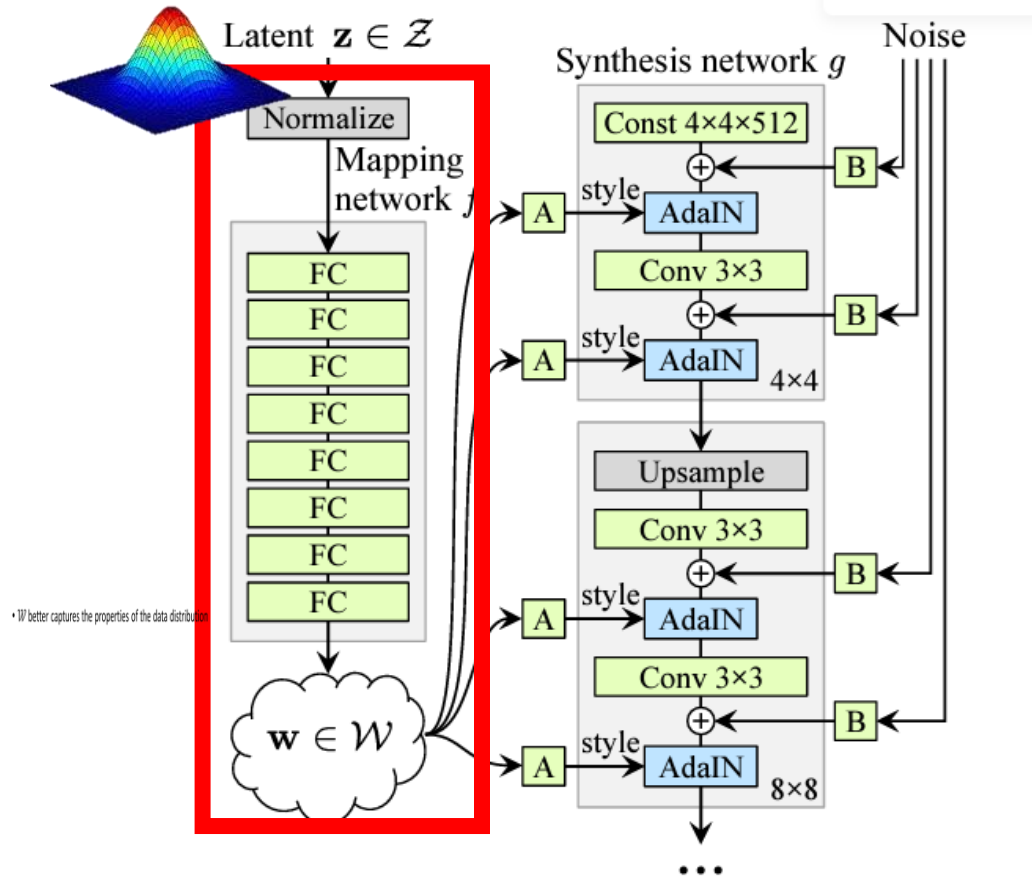
StyleGAN

- Our distribution example



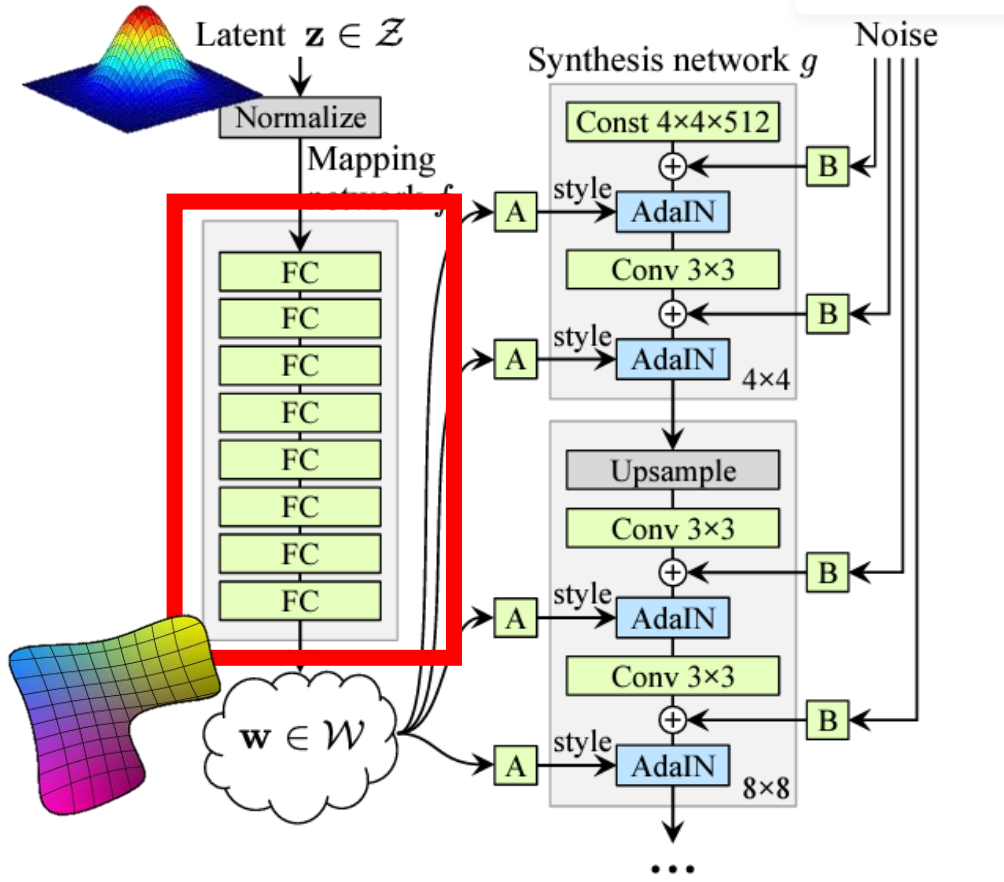
StyleGAN

- $\mathcal{W}$ better captures the properties of the data distribution



StyleGAN

- Mapping network is an 8-layer MLP.

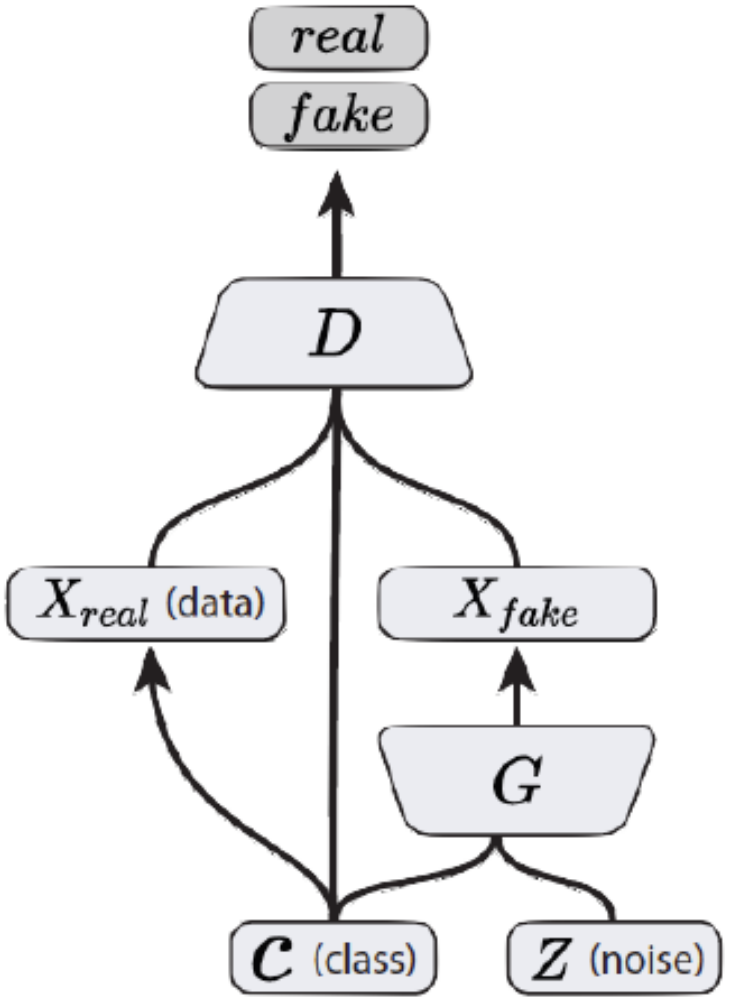


StyleGAN

- $\mathcal{W}$ is very disentangled.

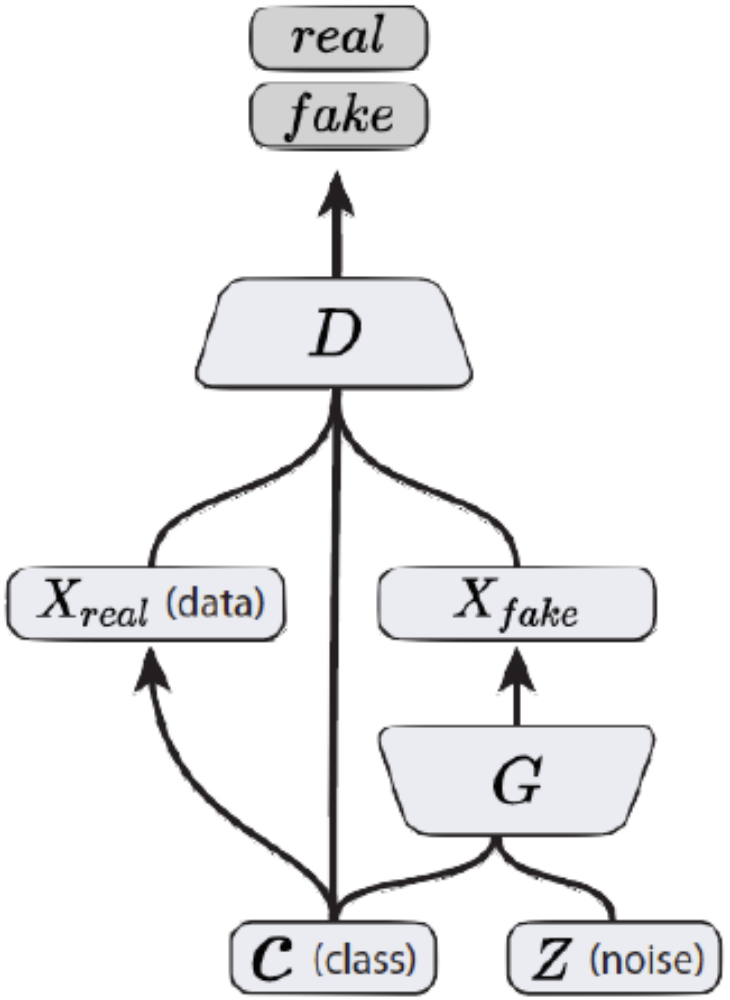
Conditional GAN

- Starting from noise with no condition, it can’t generate all the numbers from 0-9 in good quality.



Conditional GAN

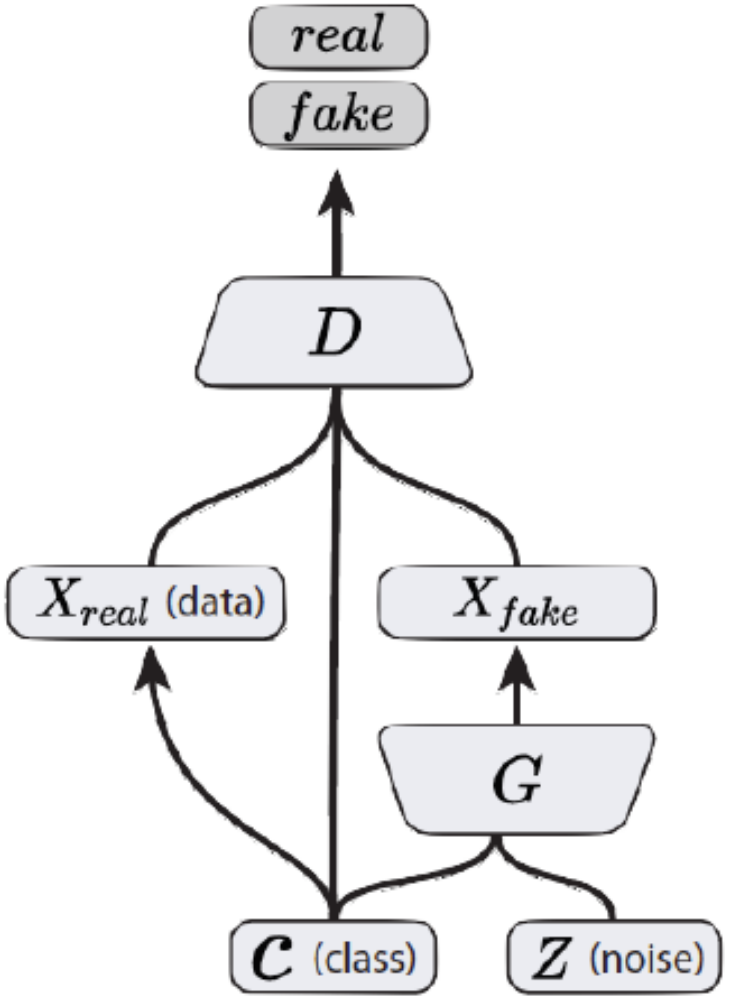
- Simple modification to the original GAN that conditions the model on additional information for better multi-modal learning.



Conditional GAN

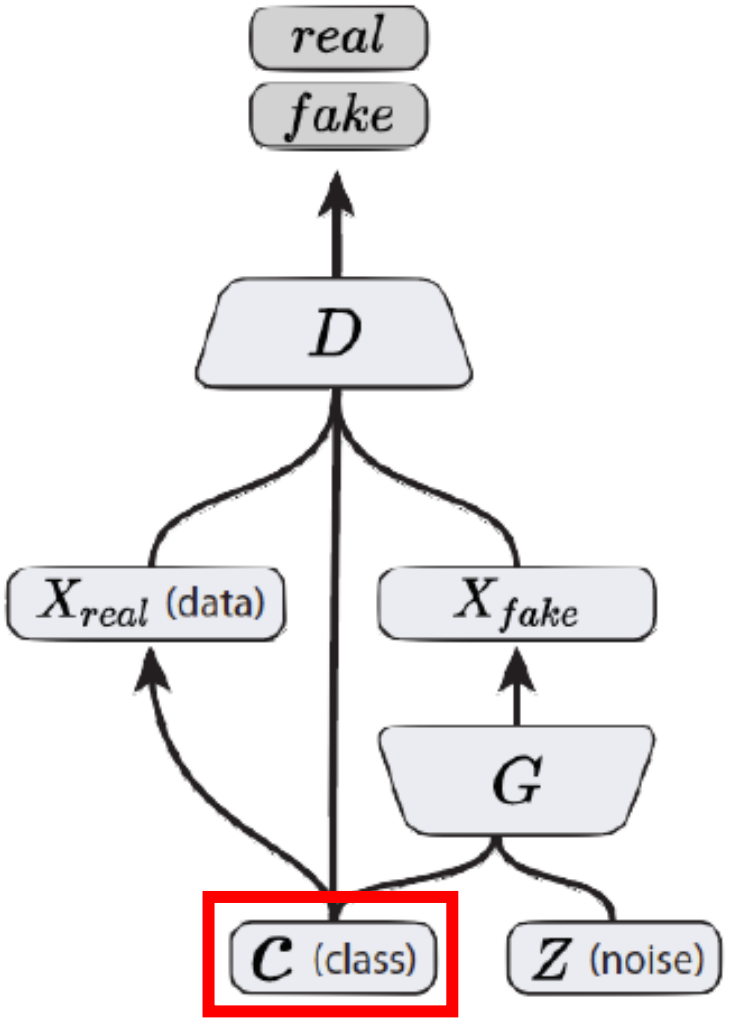
- Simple modification to the original GAN that conditions the model on additional information for better multi-modal learning.

- Very useful when we have explicit classes of data.



Conditional GAN

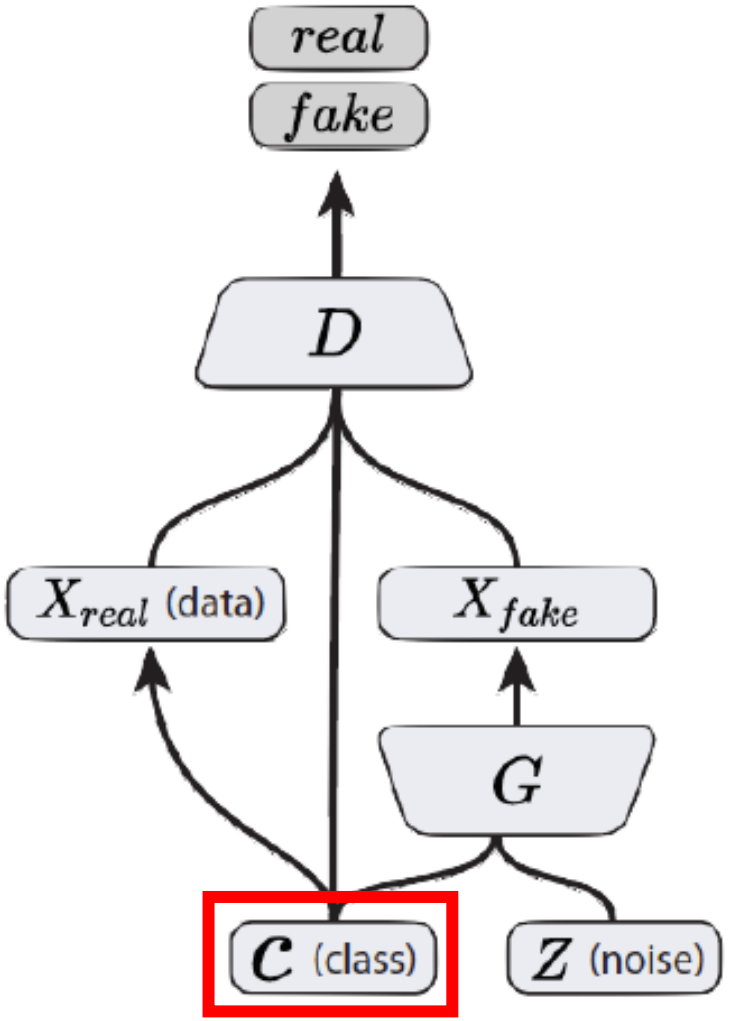
- As well as sample from the noise, a vector indicating the class of data is provided to both Generator and Discriminator.



Conditional GAN

- As well as sample from the noise, a vector indicating the class of data is provided to both Generator and Discriminator.

- It is expected that when this vector refers to a specific class, the Generator generates data from that same class.



Conditional GAN

$[1, 0, 0, 0, 0, 0, 0, 0, 0, 0] \rightarrow$

$[0, 1, 0, 0, 0, 0, 0, 0, 0, 0] \rightarrow$

$[0, 0, 1, 0, 0, 0, 0, 0, 0, 0] \rightarrow$

$[0, 0, 0, 1, 0, 0, 0, 0, 0, 0] \rightarrow$

$[0, 0, 0, 0, 1, 0, 0, 0, 0, 0] \rightarrow$

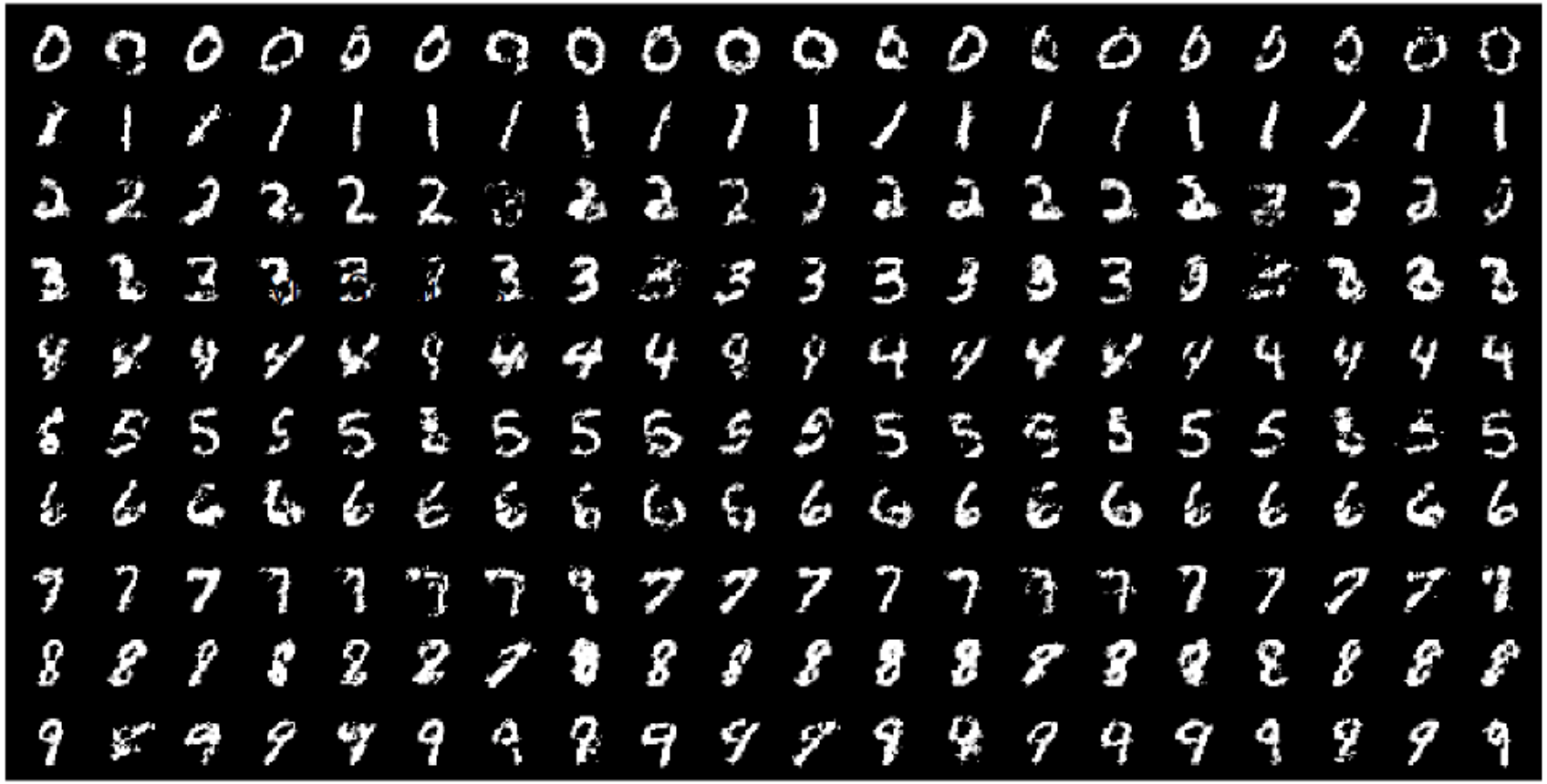
$[0, 0, 0, 0, 0, 1, 0, 0, 0, 0] \rightarrow$

$[0, 0, 0, 0, 0, 0, 1, 0, 0, 0] \rightarrow$

$[0, 0, 0, 0, 0, 0, 0, 1, 0, 0] \rightarrow$

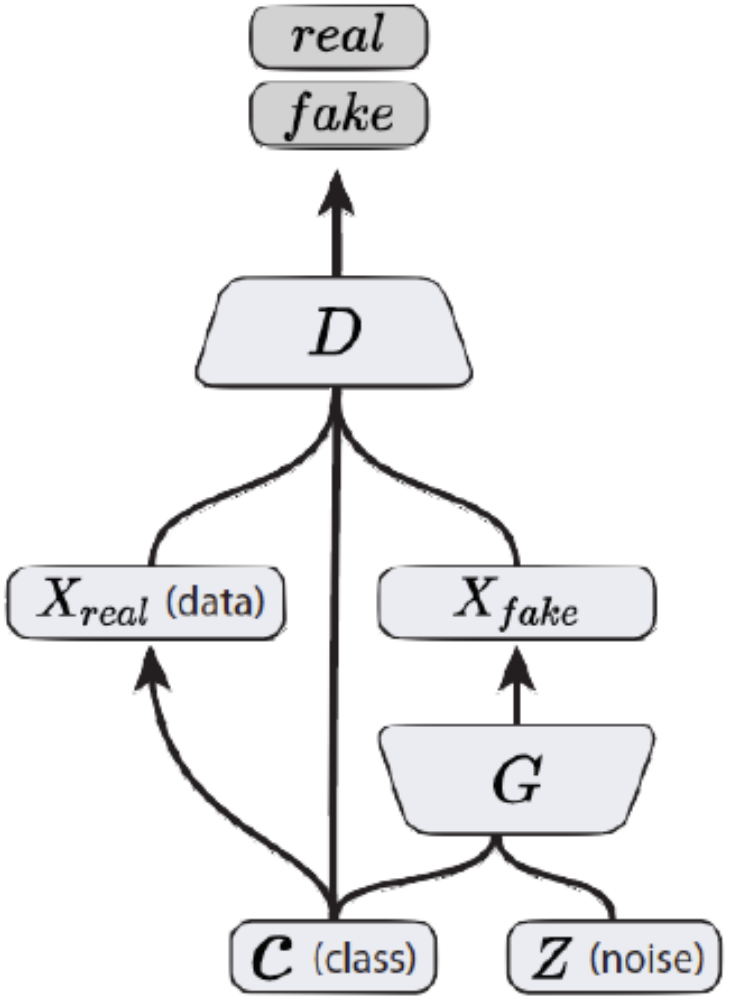
$[0, 0, 0, 0, 0, 0, 0, 0, 1, 0] \rightarrow$

$[0, 0, 0, 0, 0, 0, 0, 0, 0, 1] \rightarrow$



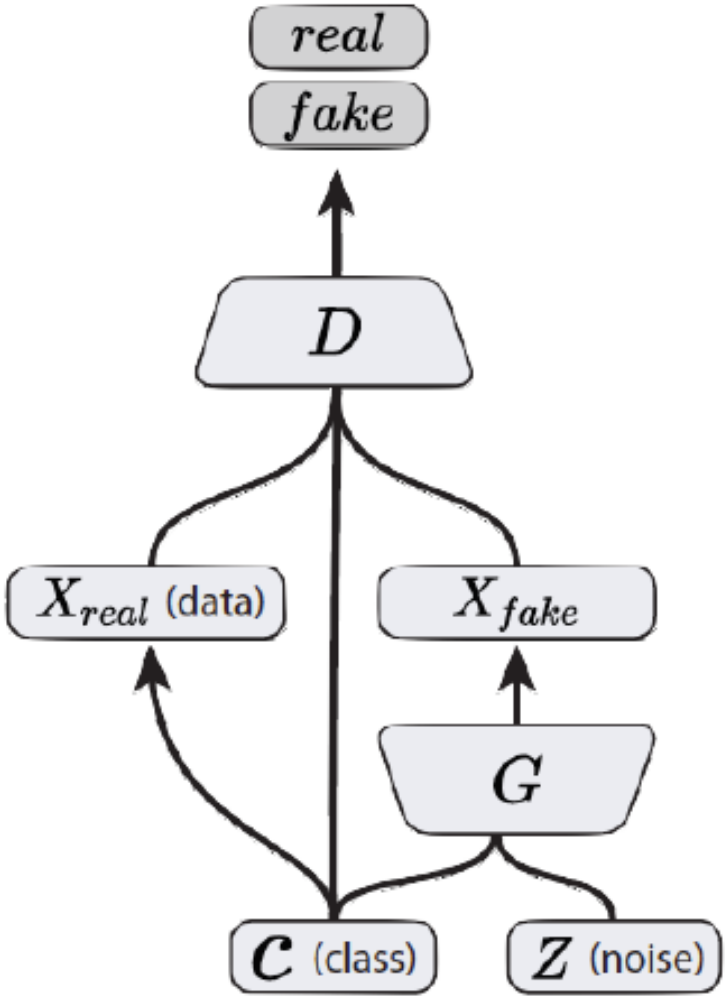
Pix2Pix

- Image as condition instead of a class.



Pix2Pix

- Image as condition



Pix2Pix

- Image as condition
- Image to image translation

Image-to-Image Translation with Conditional Adversarial Networks

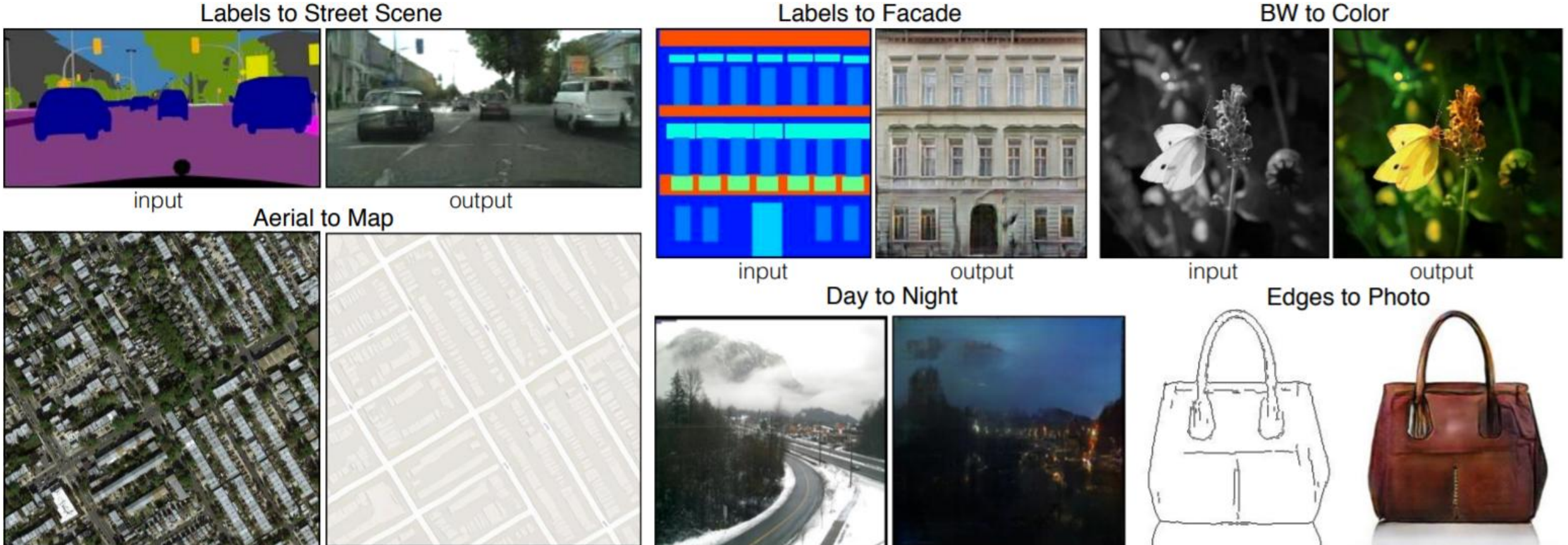
Phillip Isola Jun-Yan Zhu Tinghui Zhou Alexei A. Efros

Berkeley AI Research (BAIR) Laboratory, UC Berkeley

{isola,junyanz,tinghuiz,efros}@eecs.berkeley.edu

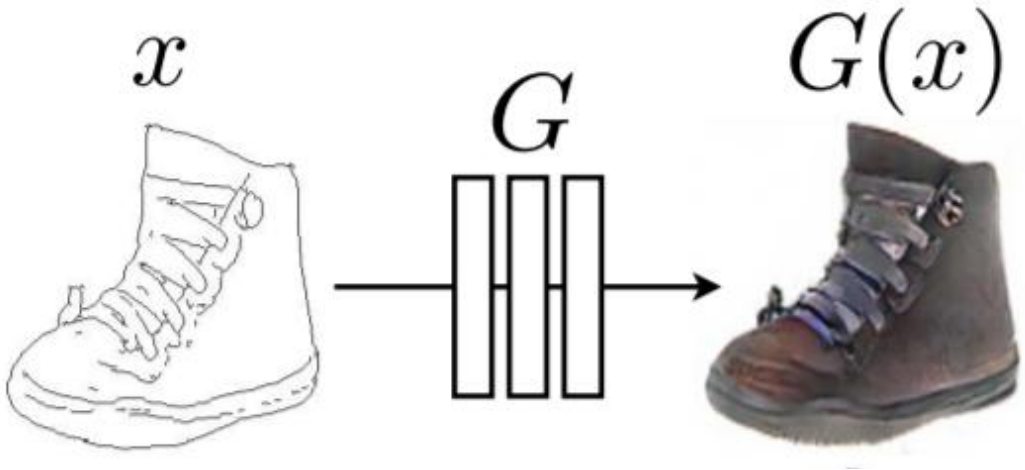
Pix2Pix

- Image as condition
- Image to image translation



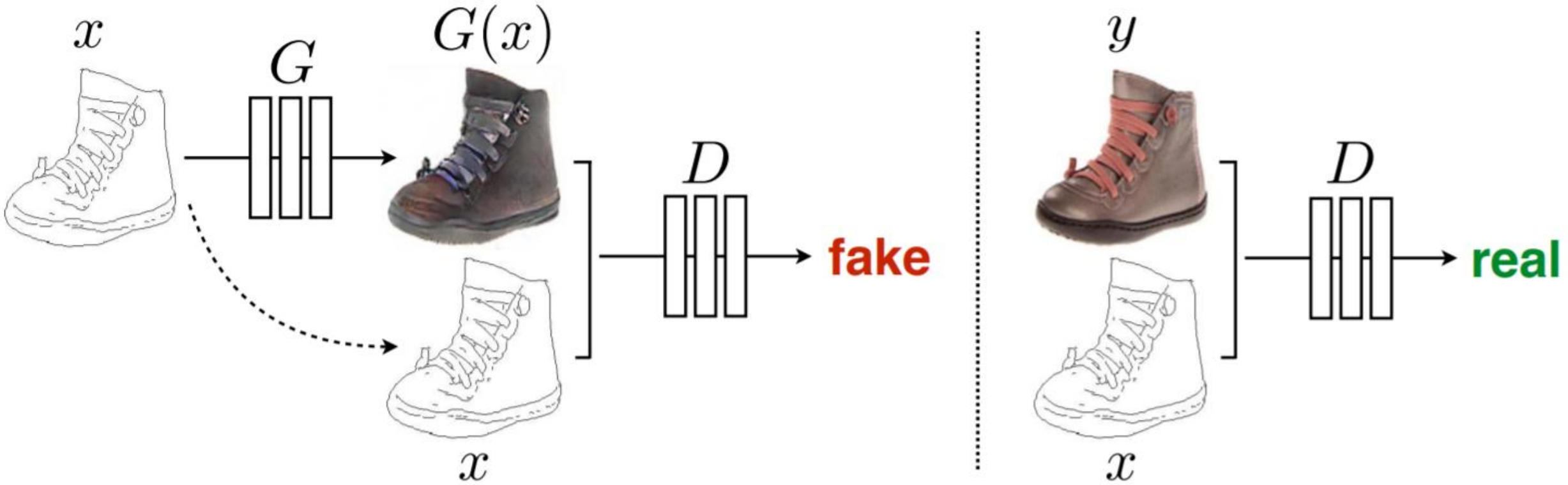
Pix2Pix

- Generator learns to map one image to another.



Pix2Pix

- Generator learns to map one image to another.
- Discriminator learns to distinguish between real models and fake ones



Pix2Pix Loss function

- Loss function is also the conditional version of GAN:

$$\mathcal{L}_{cGAN}(G, D) = E_{x,y}[\log D(x, y)] + E_{x,z}[\log(1 - D(x, G(x, z)))]$$

Pix2Pix Loss function

- Loss function is also the conditional version of GAN:

$$\mathcal{L}_{cGAN}(G, D) = E_{x,y}[\log D(x, y)] + E_{x,z}[\log(1 - D(x, G(x, z)))]$$

- Min-max game is also happening here:

$$argmin_G max_D \mathcal{L}_{cGAN}(G, D)$$

Pix2Pix Loss function

- Loss function for D is the same but loss function of G is slightly changed.

Pix2Pix Loss function

- Loss function for D is the same but loss function of G is slightly changed.

- An additional term is added to minimize the difference between the generated image and the ground truth

$$\mathcal{L}_{L1}(G) = E_{x,y,z}[\|y - G(x,z)\|_1]$$

Pix2Pix Loss function

- Loss function for D is the same but loss function of G is slightly changed.

- An additional term is added to minimize the difference between the generated image and the ground truth

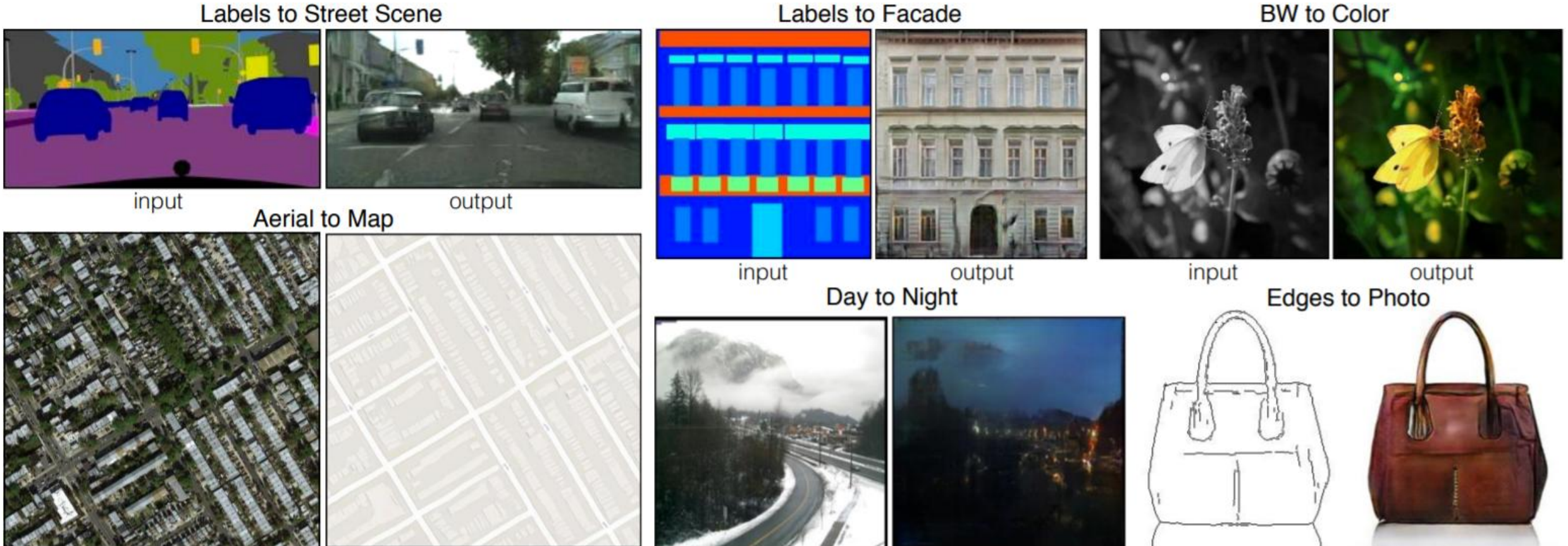
$$\mathcal{L}_{L1}(G) = E_{x,y,z}[\|y - G(x,z)\|_1]$$

- Therefore loss function for G is defined as:

$$G^* = argmin_G max_D \mathcal{L}_{cGAN}(G,D) + \lambda \mathcal{L}_{L1}(G)$$

Training

- A pair of data is available to train the model.



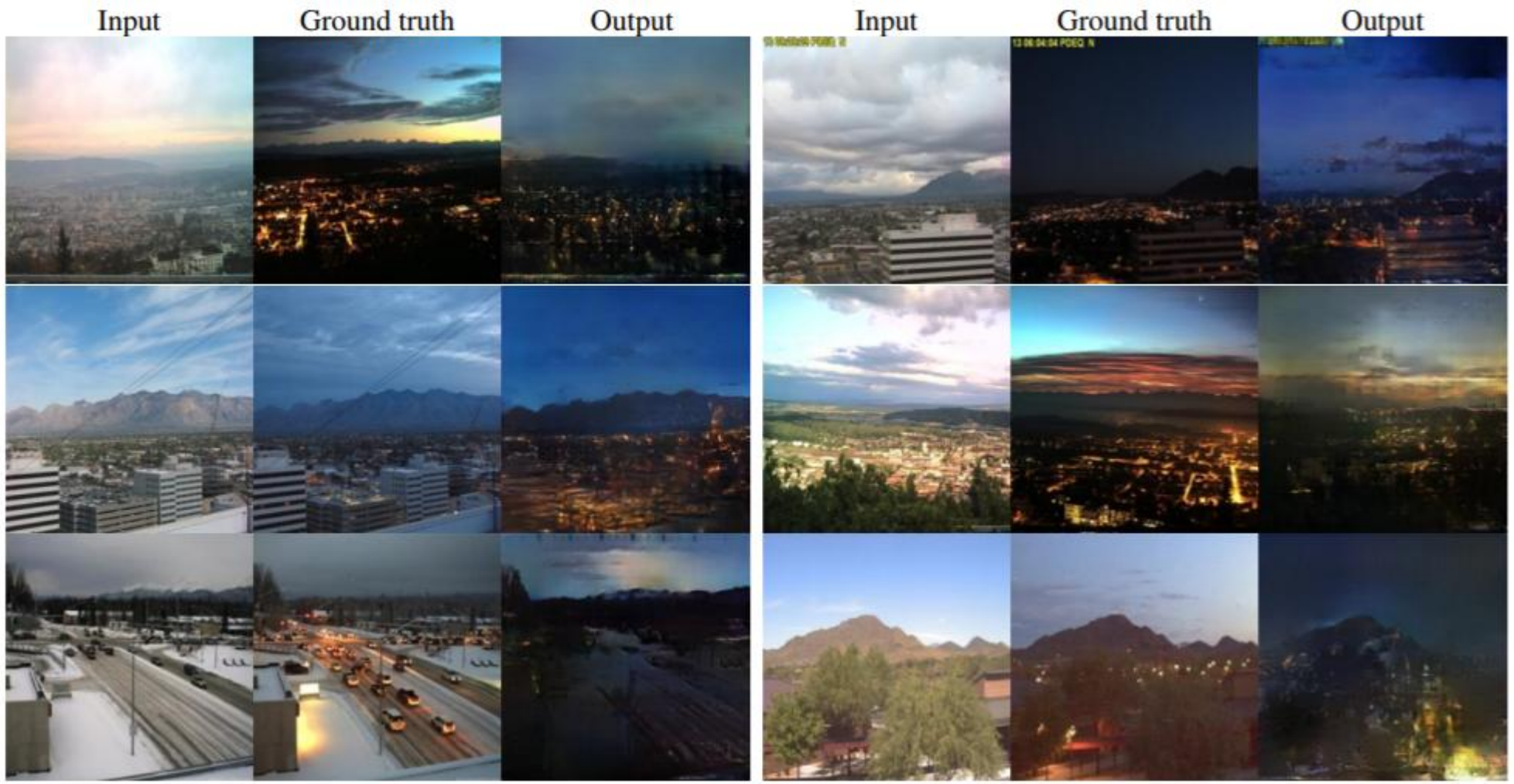
Training

- A pair of data is available to train the model.
- Such pair data is either available by nature.

Training

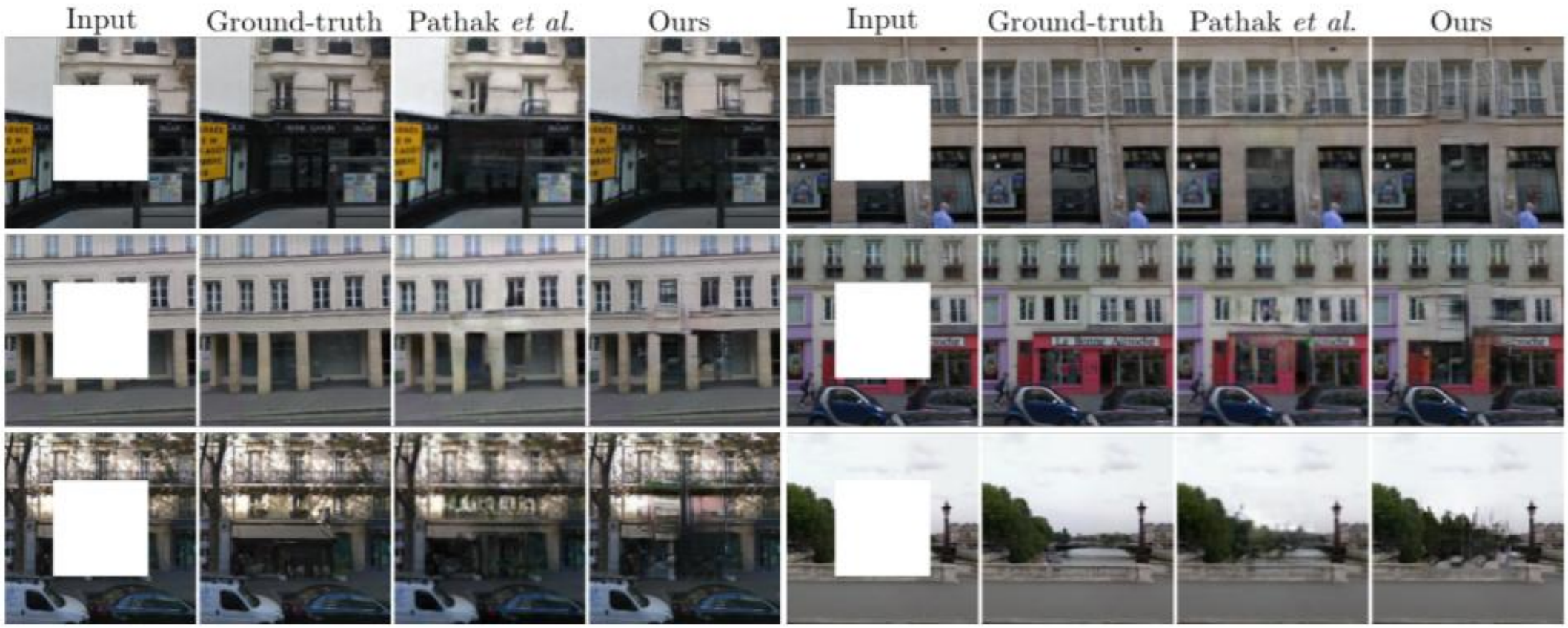
- A pair of data is available to train the model.
- Such pair data is either available by nature.
- Or it can be synthesized quite easily.

Results



Results

Results


Input
Ground-truth
Pathak *et al.*
Ours
Input
Ground-truth
Pathak *et al.*
Ours

Dual GAN, Cycle GAN

- In Pix2Pix, we need to have paired data.

Dual GAN, Cycle GAN

- In Pix2Pix, we need to have paired data.
- The idea is to transfer an image from one domain to another without needing to have paired data.

Dual GAN, Cycle GAN

- This idea was initially proposed in an NLP paper trying to translate from one language to another without the need of word for word translation. (French to English translation)

Dual Learning for Machine Translation

Di He[1,*], Yingce Xia[2,*], Tao Qin[3], Liwei Wang[1], Nenghai Yu[2], Tie-Yan Liu[3], Wei-Ying Ma[3]
[1]Key Laboratory of Machine Perception (MOE), School of EECS, Peking University
[2]University of Science and Technology of China [3]Microsoft Research
[1]{dih,wanglw}@cis.pku.edu.cn; [2]xiayingc@mail.ustc.edu.cn; [2]ynh@ustc.edu.cn
[3]{taoqin,tie-yan.liu,wyma}@microsoft.com

Dual GAN, Cycle GAN

- Dual GAN and similarly Cycle GAN use the same idea to transfer images from one domain to another without the need of paired training data.

DualGAN: Unsupervised Dual Learning for Image-to-Image Translation

Zili Yi[1,2], Hao Zhang[2], Ping Tan[2], and Minglun Gong[1]

[1]Memorial University of Newfoundland, Canada
[2]Simon Fraser University, Canada

Dual GAN, Cycle GAN

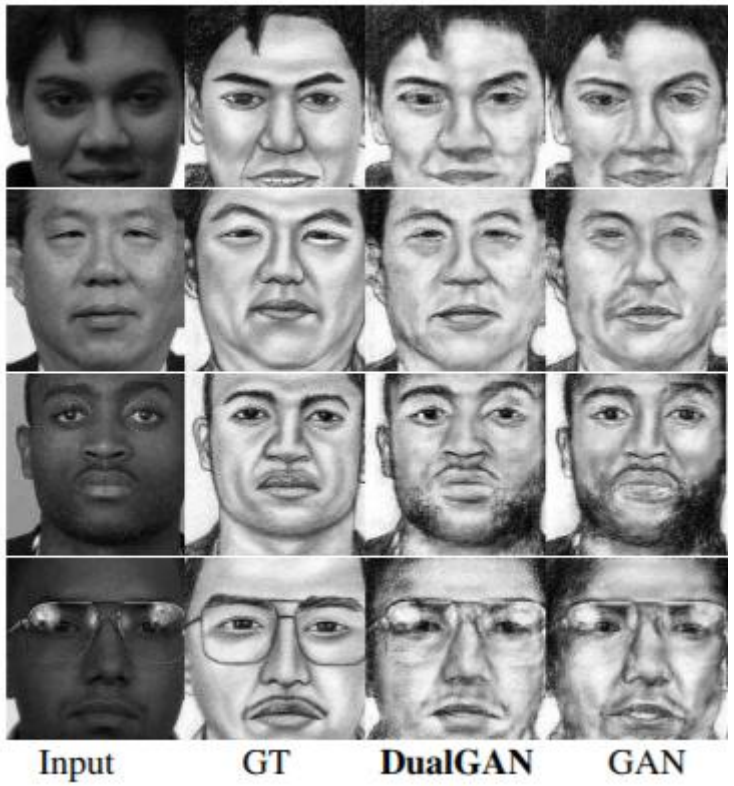
- CycleGAN

Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

Jun-Yan Zhu* Taesung Park* Phillip Isola Alexei A. Efros
Berkeley AI Research (BAIR) laboratory, UC Berkeley

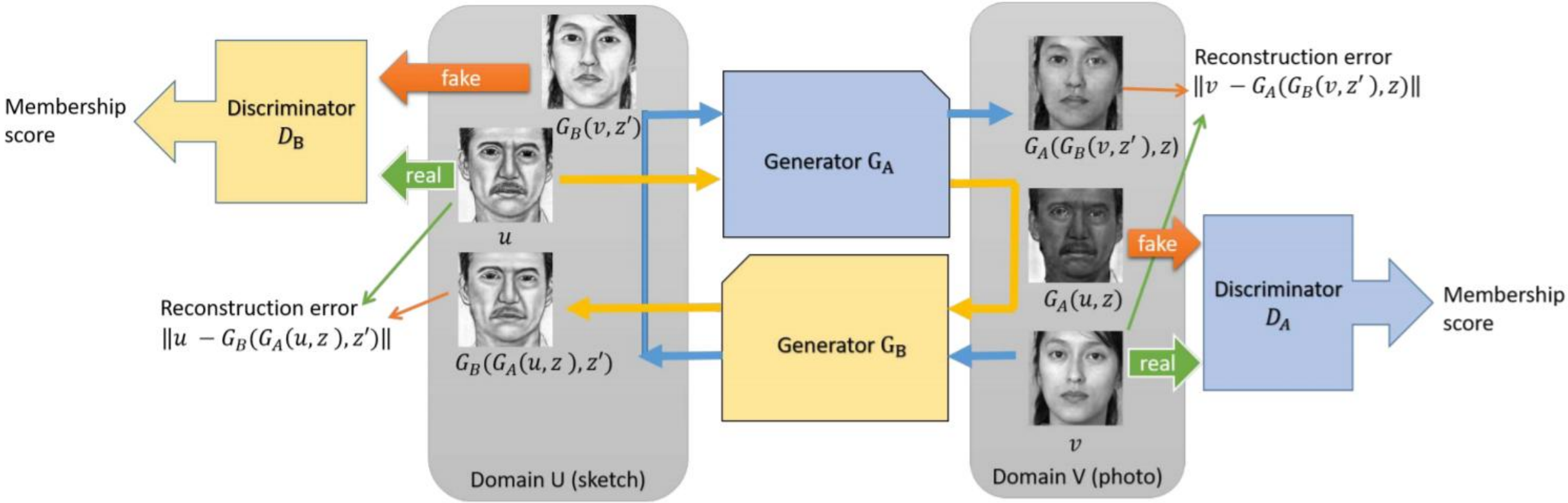
Dual GAN, Cycle GAN

- Here, we use the notation of Dual GAN.



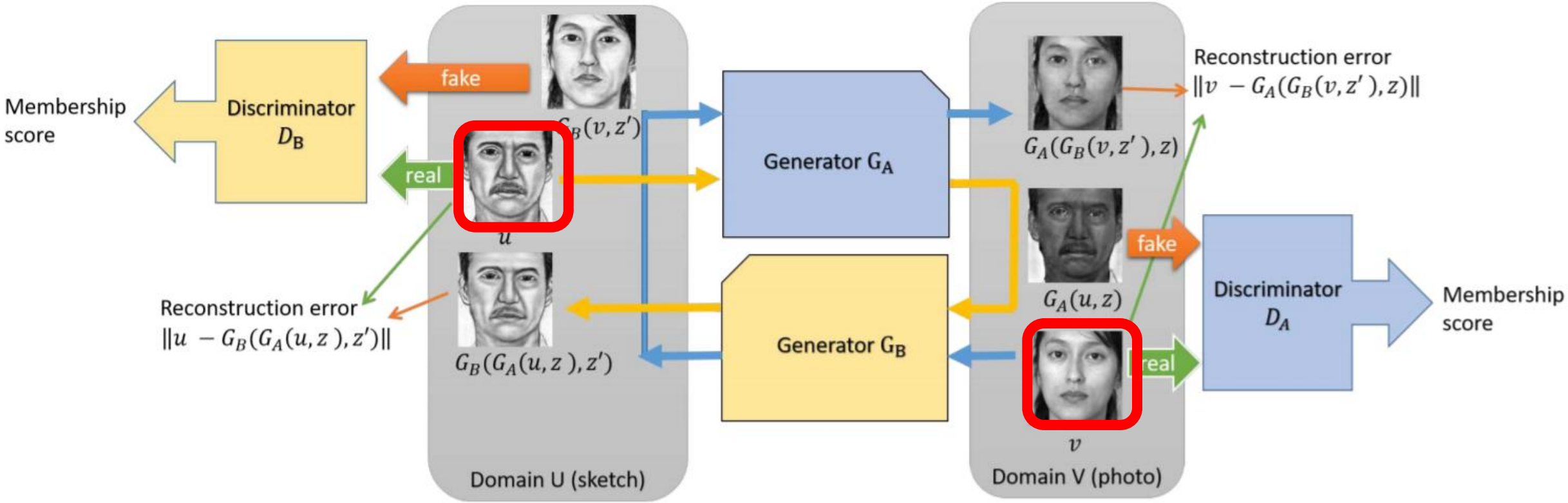
Dual GAN

- The idea is to form a loop between one domain to another



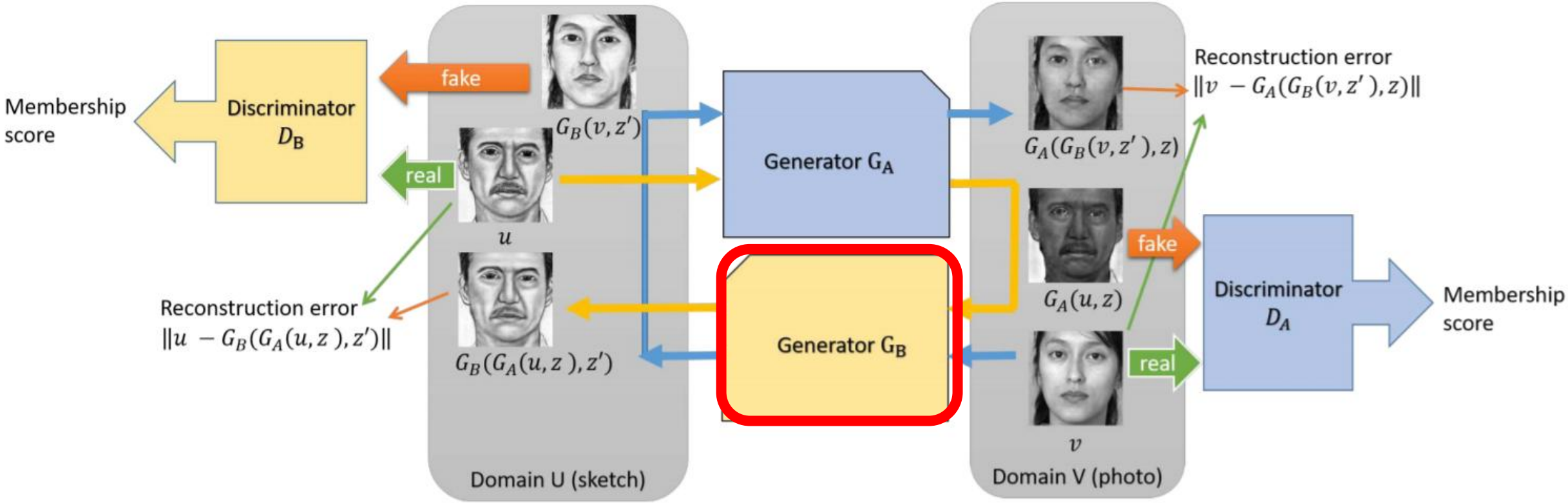
Dual GAN

- We have ground truth data u and v in two different domains U and V that are not paired necessarily.



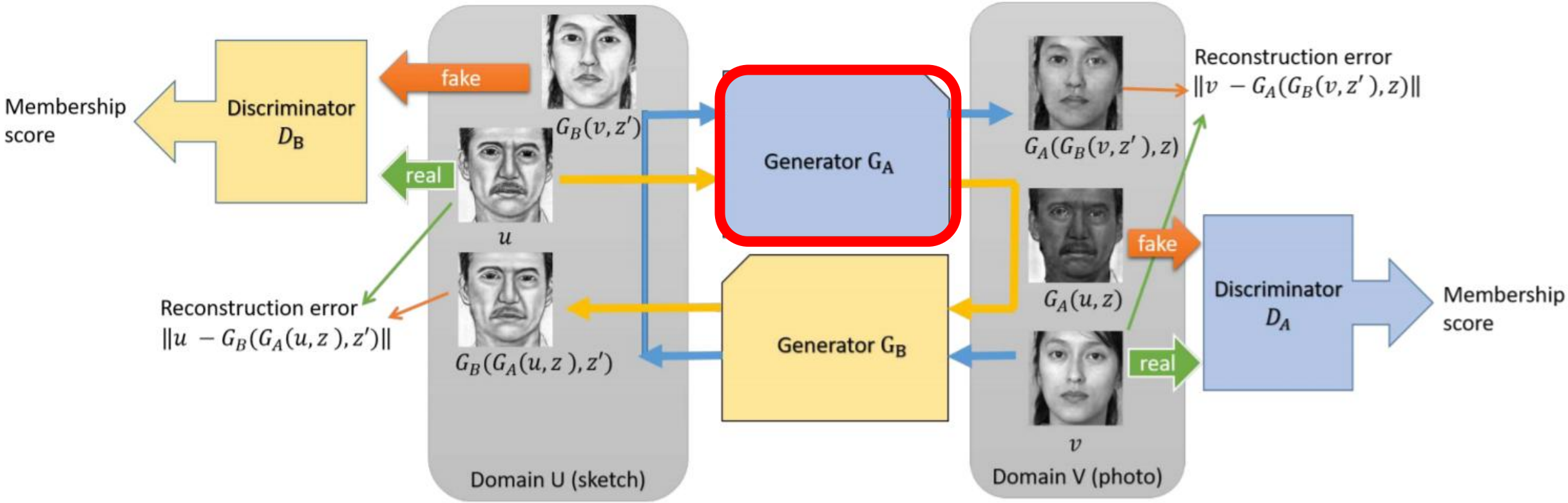
Dual GAN

- We have two generators G_B that maps images from domain V to domain U (essentially a pix2pix)



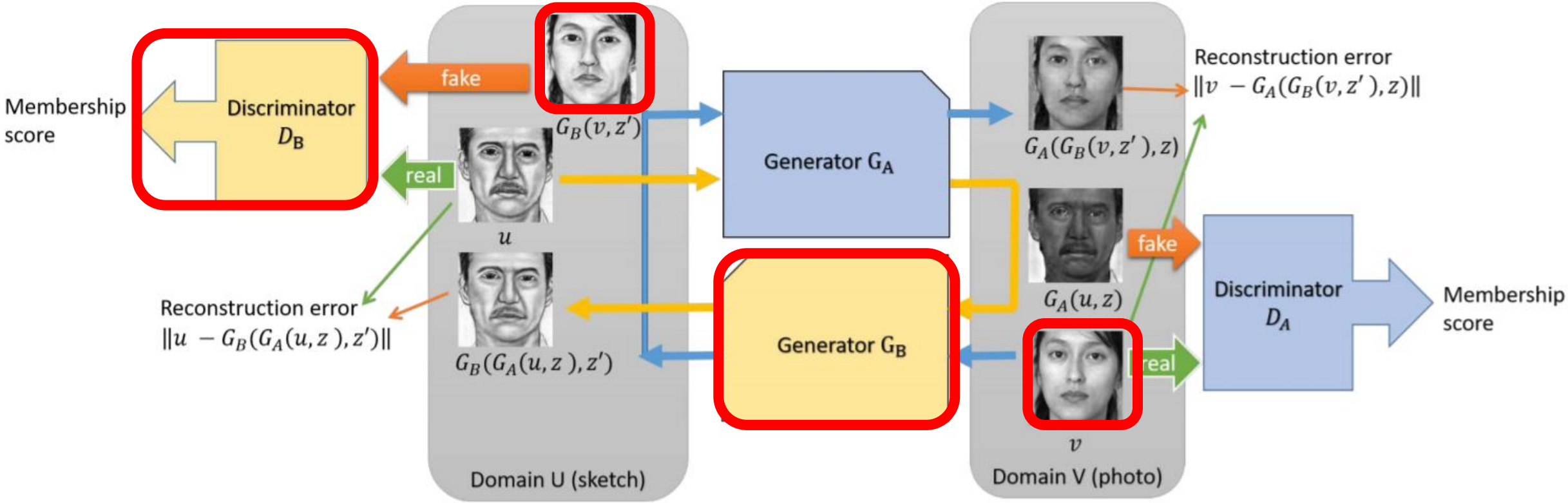
Dual GAN

- and G_A that maps images from domain U to domain V



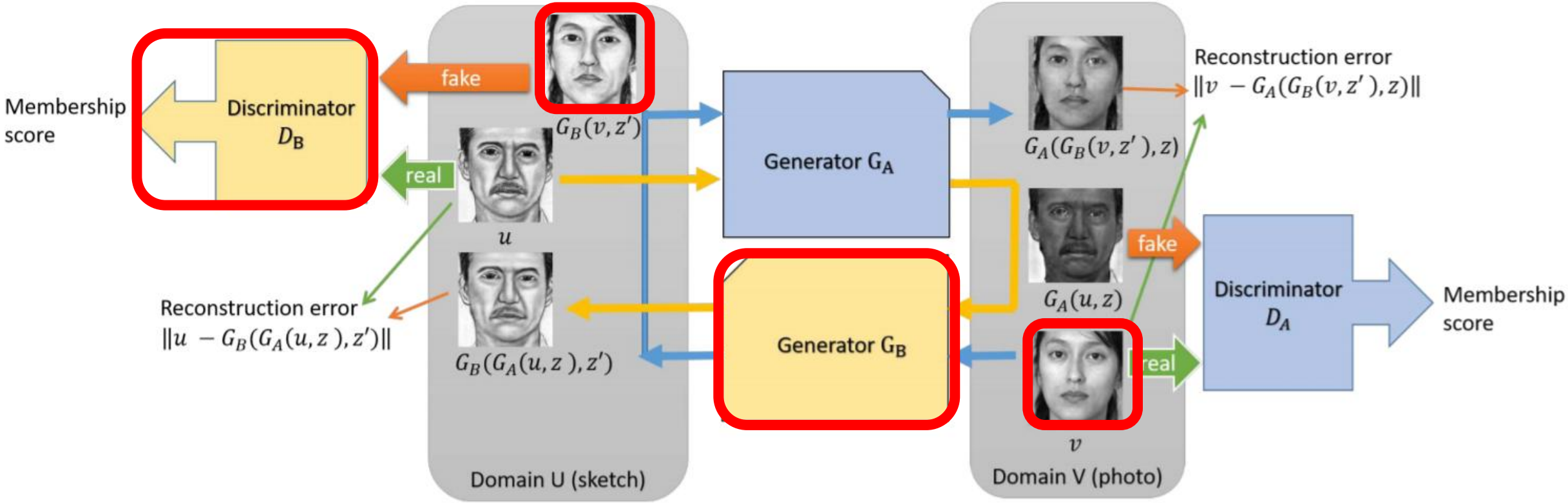
Dual GAN

- Let's focus on $V \rightarrow U$ mapping.



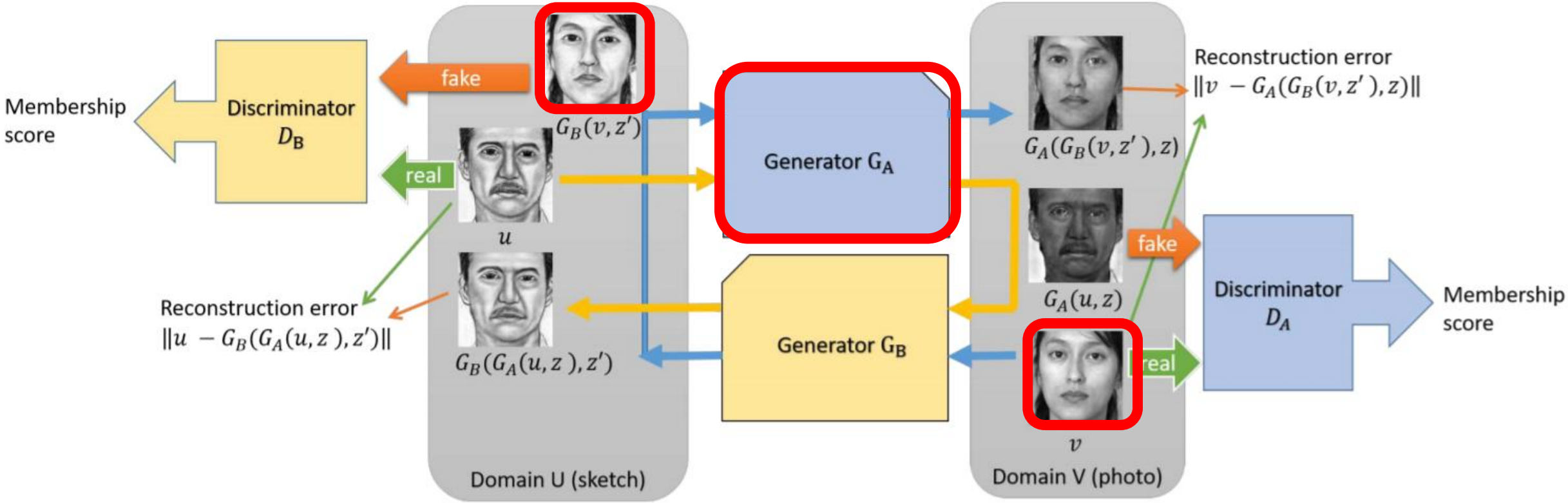
Dual GAN

- Discriminator D_B receives $G_B(v)$ as fake sample and u as real example



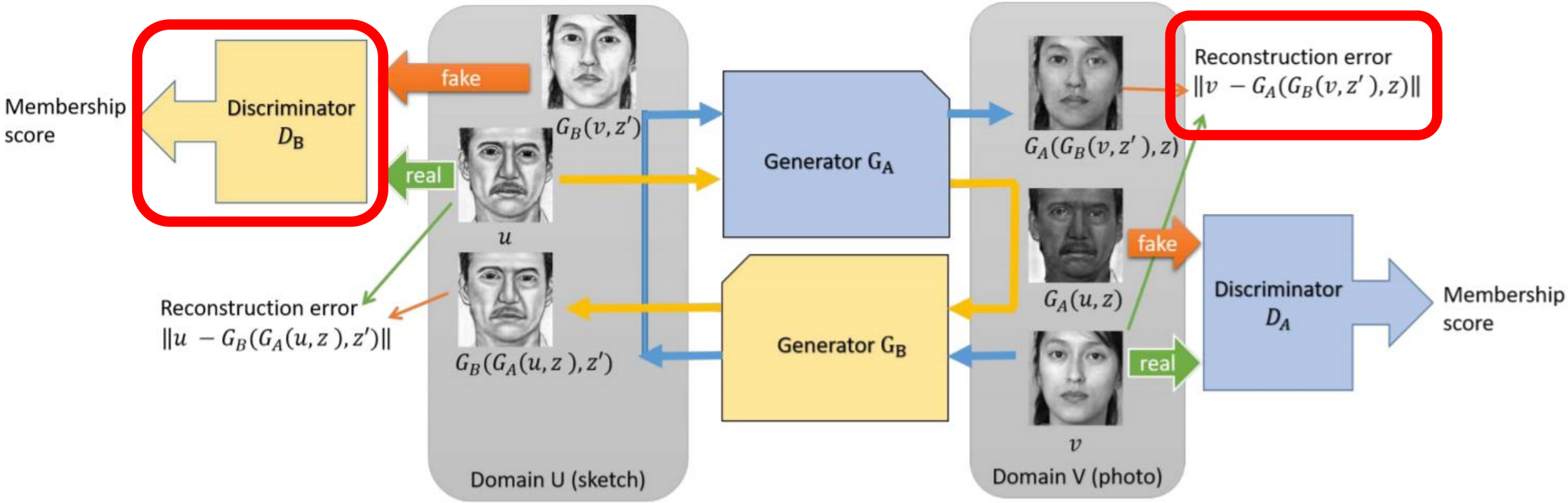
Dual GAN

- Cycle loss: $G_B(v)$ is also given to G_A as input. So the expectation is that $G_A(G_B(v))$ would be similar to v.



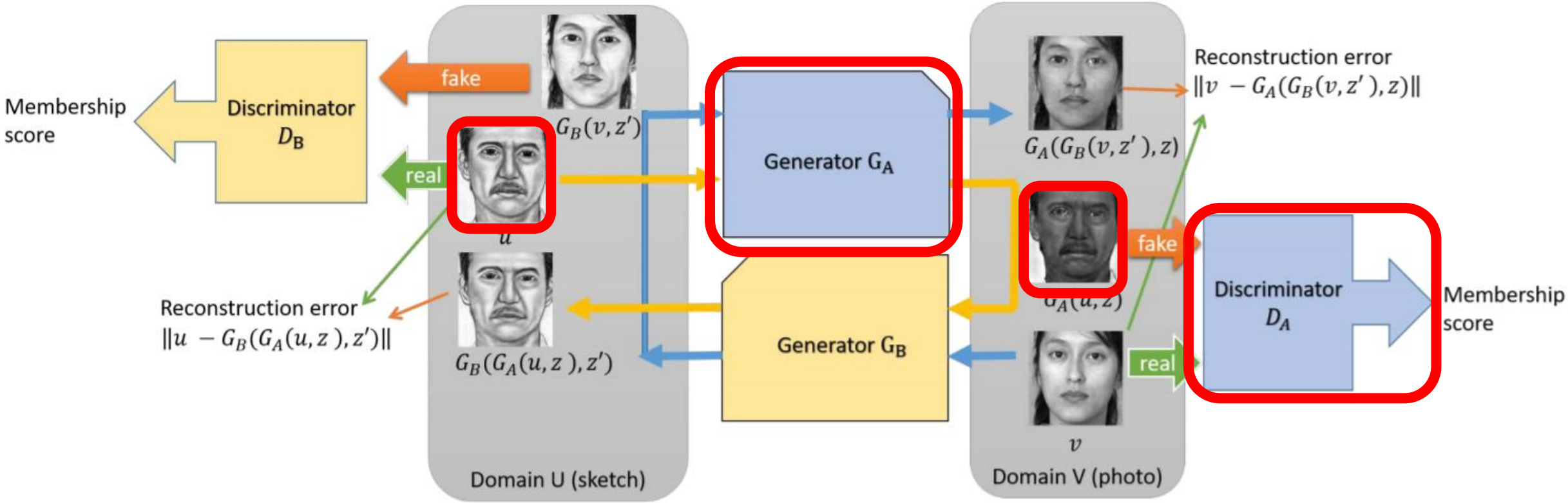
Dual GAN

- Therefore, we have two types of losses. An adversarial and a reconstruction loss.



Dual GAN

- The other cycle is similar



Dual GAN Loss

- Generator Loss

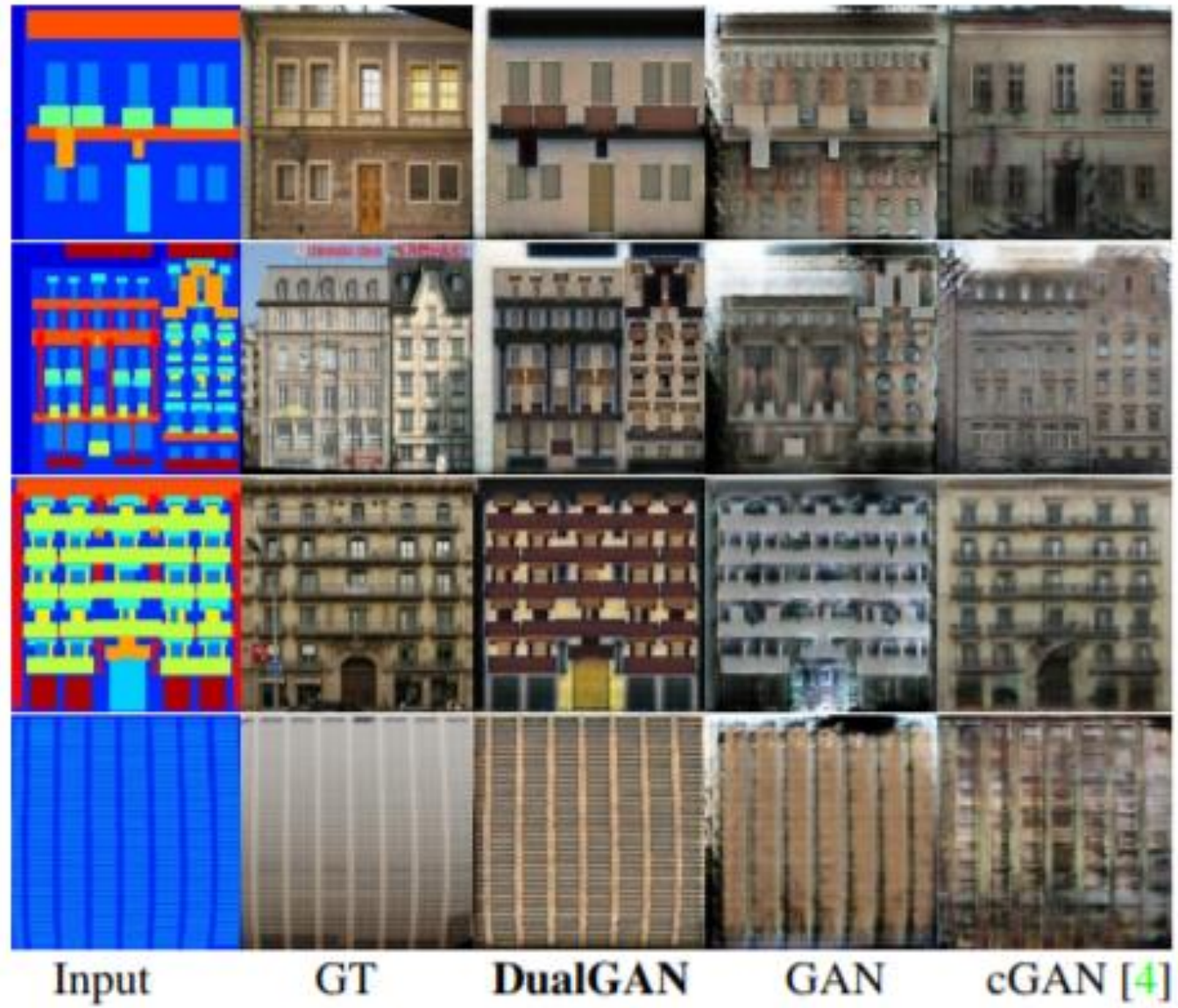
$$l^g(u,v) = \lambda_U \|u - G_B(G_A(u,z),\acute{z})\| + \lambda_V \|v - G_A(G_B(v,\acute{z}),z)\| - D_B\left(G_B(v,\acute{z})\right) - D_A\left(G_A(v,\acute{z})\right)$$

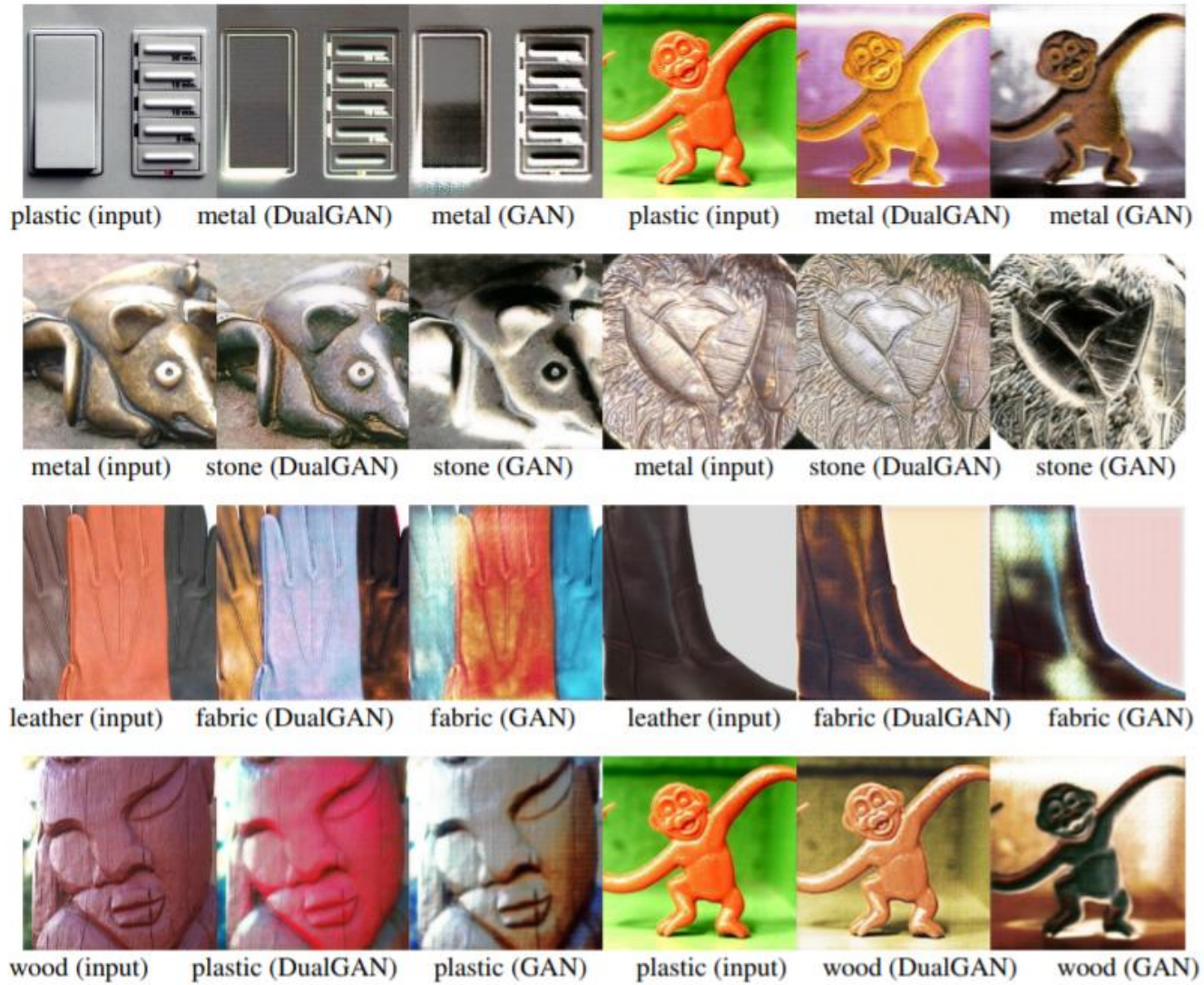
Dual GAN Loss

- Discriminator Loss

$$l_A^d(u, v) = D_A\big(G_A(u, z)\big) - D_A(v)$$

$$l_B^d(u, v) = D_B\big(G_B(v, \acute{z})\big) - D_B(u)$$


Input
GT
DualGAN
GAN
cGAN [4]


plastic (input)
metal (DualGAN)
metal (GAN)
plastic (input)
metal (DualGAN)
metal (GAN)
metal (input)
stone (DualGAN)
stone (GAN)
metal (input)
stone (DualGAN)
stone (GAN)
leather (input)
fabric (DualGAN)
fabric (GAN)
leather (input)
fabric (DualGAN)
fabric (GAN)
wood (input)
plastic (DualGAN)
plastic (GAN)
plastic (input)
wood (DualGAN)
wood (GAN)

Diffusion

- Diffusion models are generative models that can produce high-quality images and 3D models.

Diffusion

- Diffusion models are generative models that can produce high-quality images and 3D models.

- Their principle has some commonalities with VAE as both map a Gaussian noise to a data point.

Applications

- Diffusion is the technique behind recent famous text to image generation models:
  - Dall E
  - Imagen
  - Etc.

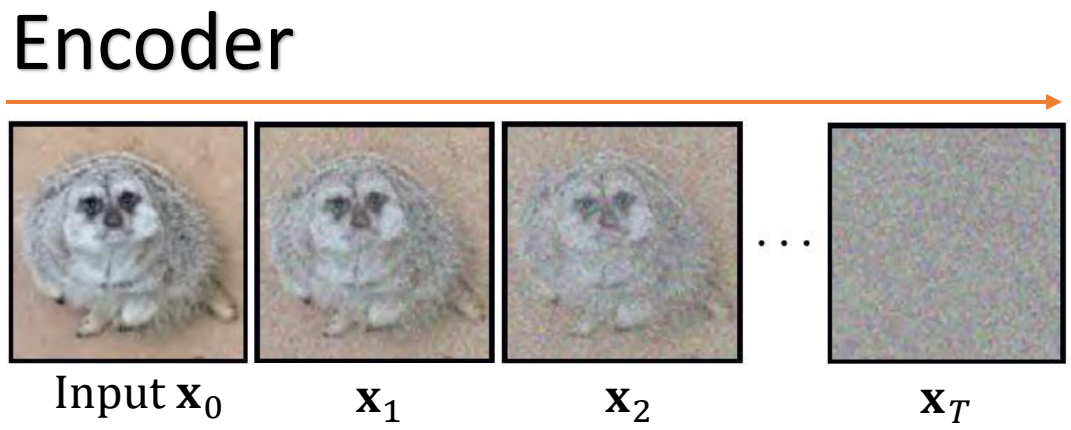
"A teddy bear on a skateboard in Times Square"

Diffusion architecture

- Similar to VAE, diffusion models consist of an encoder and a decoder.

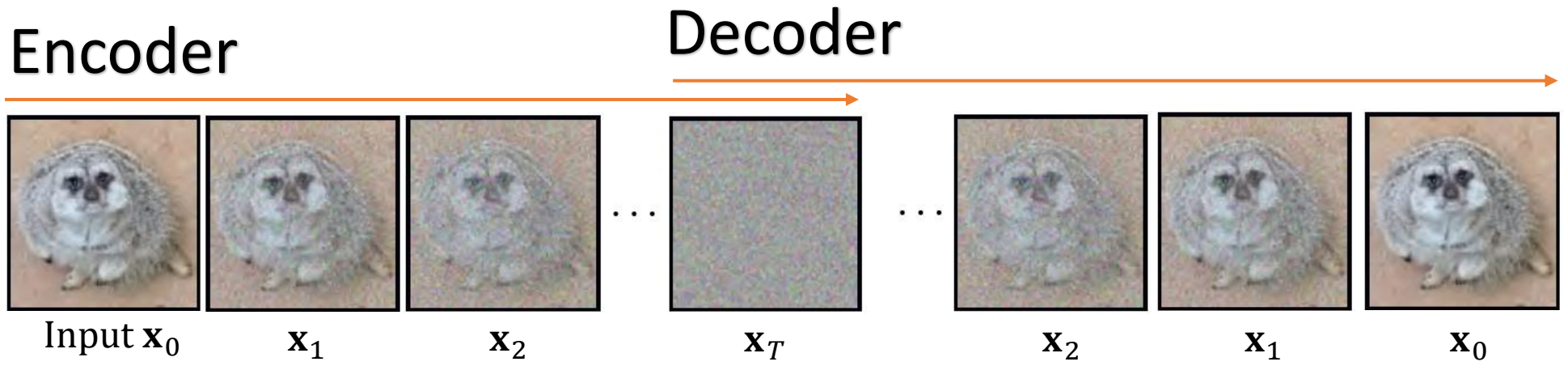
Diffusion architecture

- Similar to VAE, diffusion models consist of an encoder and a decoder.
- Encoder maps an input $\mathbf{x}_0$ to a noise $\mathbf{x}_T$ through an iterative process.



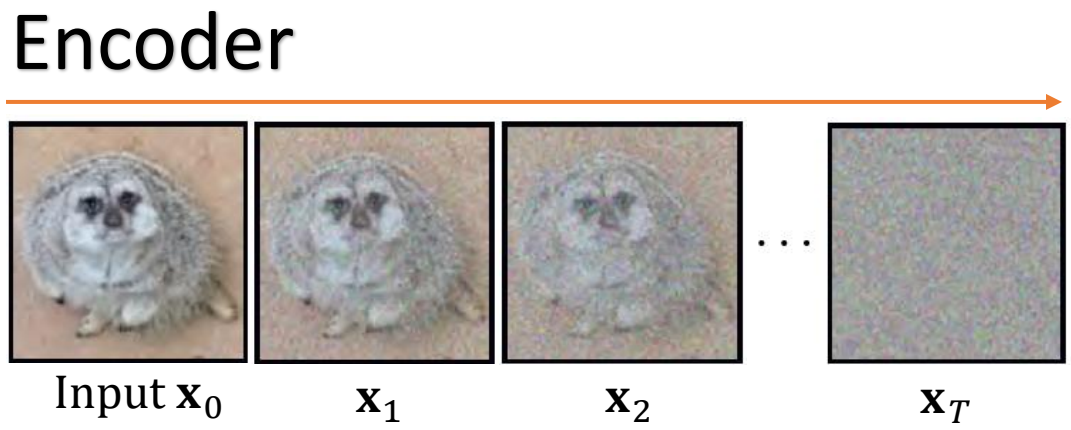
Diffusion architecture

- Similar to VAE, diffusion models consist of an encoder and a decoder.
- Encoder maps an input $\mathbf{x}_0$ to a noise $\mathbf{x}_T$ through an iterative process.
- Decoder removes the noise and generates an output.



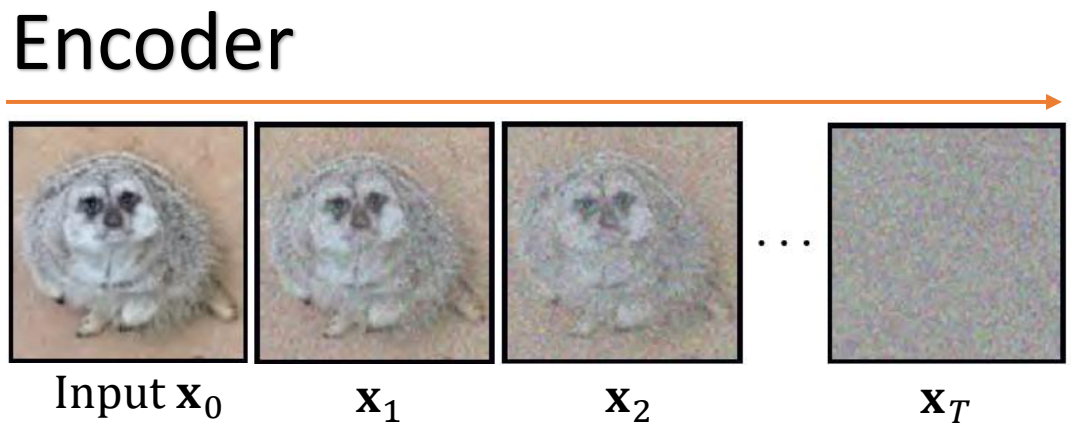
Encoder

- Encoder is pre-determined. It gradually adds noise to input $\mathbf{x}_0$.



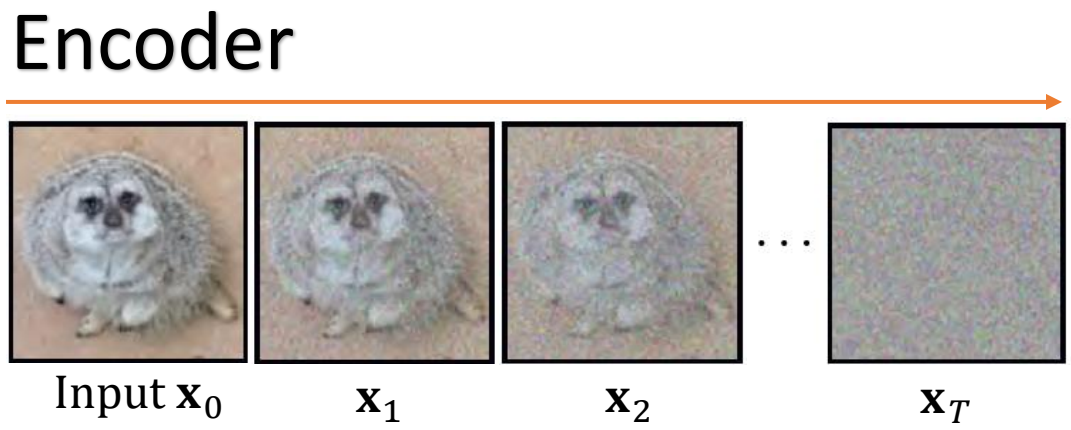
Encoder

- Encoder is pre-determined. It gradually adds noise to input $\mathbf{x}_0$.
- After several steps $q(\mathbf{x}_T)$ is equivalent to a Gaussian noise.



Encoder

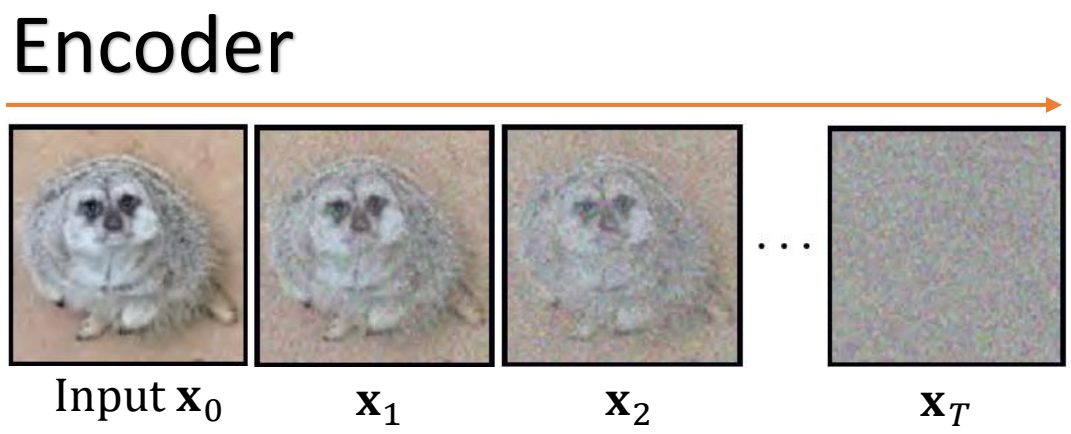
- Encoder is pre-determined. It gradually adds noise to input $\mathbf{x}_0$.
- After several steps $q(\mathbf{x}_T)$ is equivalent to a Gaussian noise.
- Nothing is learned in this process.



Encoder

- The diffusion or forward pass maps input $\mathbf{x}_0$ to intermediate variables $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$ as:

$$\mathbf{x}_1 = \sqrt{1-\alpha_1}\mathbf{x}_0 + \sqrt{\alpha_1}\epsilon_1$$

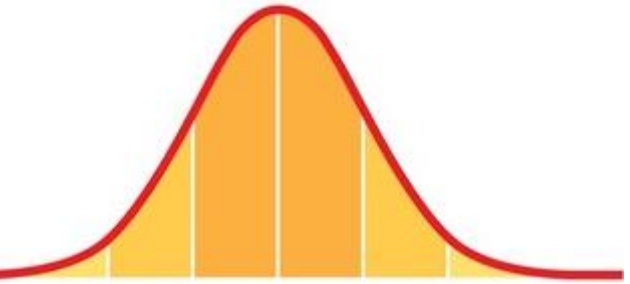


Encoder

- The diffusion or forward pass maps input $\mathbf{x}_0$ to intermediate variables $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$ as:

$$\mathbf{x}_1 = \sqrt{1-\alpha_1}\mathbf{x}_0 + \sqrt{\alpha_1}\epsilon_1$$

Gaussian



Encoder

Encoder

- The diffusion or forward pass maps input $\mathbf{x}_0$ to intermediate variables $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$ as:

$$\mathbf{x}_1 = \sqrt{1-\alpha_1}\mathbf{x}_0 + \sqrt{\alpha_1}\epsilon_1$$

Controls the deterioration of the image $\alpha_t \in [0,1]$

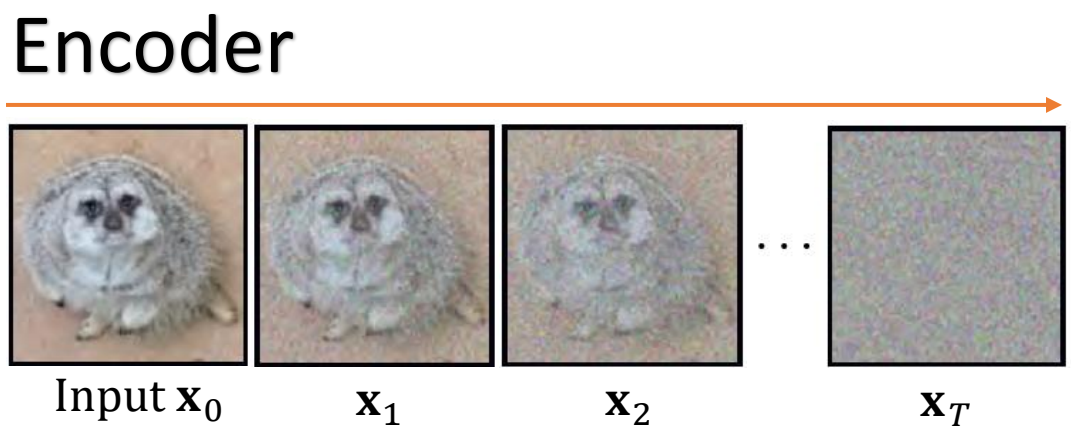
Encoder

Encoder

- The diffusion or forward pass maps input $\mathbf{x}_0$ to intermediate variables $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$ as:

$$\mathbf{x}_1 = \sqrt{1-\alpha_1}\mathbf{x}_0 + \sqrt{\alpha_1}\epsilon_1$$

$$\mathbf{x}_2 = \sqrt{1-\alpha_1}\mathbf{x}_1 + \sqrt{\alpha_1}\epsilon_2$$

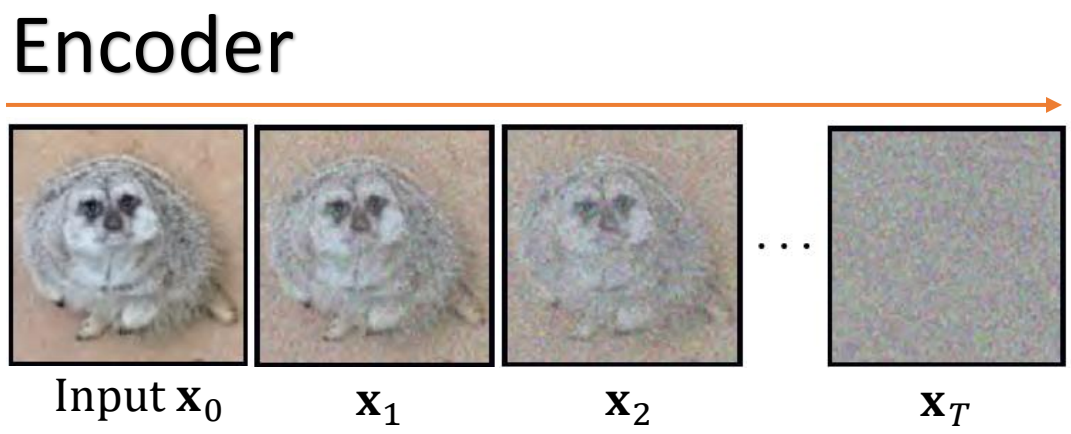


Encoder

- The diffusion or forward pass maps input $\mathbf{x}_0$ to intermediate variables $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$ as:

$$\mathbf{x}_1 = \sqrt{1-\alpha_1}\mathbf{x}_0 + \sqrt{\alpha_1}\epsilon_1$$

$$\mathbf{x}_t = \sqrt{1-\alpha_t}\mathbf{x}_{t-1} + \sqrt{\alpha_t}\epsilon_t$$



Encoder

- We can calculate $\mathbf{x}_t$ in one pass:

$$\mathbf{x}_1 = \sqrt{1-\alpha_1}\mathbf{x}_0 + \sqrt{\alpha_1}\epsilon_1$$

$$\mathbf{x}_t = \sqrt{1-\alpha_t}\mathbf{x}_{t-1} + \sqrt{\alpha_t}\epsilon_t$$

Encoder

- Having $\mathbf{x}_1$ as:

$$\mathbf{x}_1 = \sqrt{1-\alpha_1}\mathbf{x}_0 + \sqrt{\alpha_1}\epsilon_1$$

Encoder

- And $\mathbf{x}_2$ as :

$$\mathbf{x}_1 = \sqrt{1-\alpha_1}\mathbf{x}_0 + \sqrt{\alpha_1}\epsilon_1$$

$$\mathbf{x}_2 = \sqrt{1-\alpha_2}\mathbf{x}_1 + \sqrt{\boldsymbol{\alpha_2}}\epsilon_2$$

Encoder

- Replace $\mathbf{x}_1$ in $\mathbf{x}_2$:

$$\mathbf{x}_1 = \sqrt{1-\alpha_1}\mathbf{x}_0 + \sqrt{\alpha_1}\epsilon_1$$

$$\mathbf{x}_2 = \sqrt{1-\alpha_2}\mathbf{x}_1 + \sqrt{\alpha_2}\epsilon_2$$

Encoder

- We will have:

$$\mathbf{x}_1 = \sqrt{1-\alpha_1}\mathbf{x}_0 + \sqrt{\alpha_1}\epsilon_1$$

$$\mathbf{x}_2 = \sqrt{1-\alpha_2}\mathbf{x}_1 + \sqrt{\alpha_2}\epsilon_2$$

$$\mathbf{x}_2 = \sqrt{1-\alpha_2}(\sqrt{1-\alpha_1}\mathbf{x}_0 + \sqrt{\alpha_1}\epsilon_1) + \sqrt{\alpha_2}\epsilon_2$$

Encoder

- Open up $\mathbf{x}_2$, we will have:

$$\mathbf{x}_1 = \sqrt{1-\alpha_1}\mathbf{x}_0 + \sqrt{\alpha_1}\epsilon_1$$

$$\mathbf{x}_2 = \sqrt{1-\alpha_2}\mathbf{x}_1 + \sqrt{\alpha_2}\epsilon_2$$

$$\mathbf{x}_2 = \sqrt{1-\alpha_2}(\sqrt{1-\alpha_1}\mathbf{x}_0 + \sqrt{\alpha_1}\epsilon_1) + \sqrt{\alpha_2}\epsilon_2$$

$$\mathbf{x}_2 = \sqrt{(1-\alpha_2)(1-\alpha_1)}\mathbf{x}_0 + \sqrt{1-\alpha_2-(1-\alpha_2)(1-\alpha_1)}\epsilon_1 + \sqrt{\alpha_2}\epsilon_2$$

Encoder

- Open up $\mathbf{x}_2$, we will have:

$$\mathbf{x}_1 = \sqrt{1-\alpha_1}\mathbf{x}_0 + \sqrt{\alpha_1}\epsilon_1$$

$$\mathbf{x}_2 = \sqrt{1-\alpha_2}\mathbf{x}_1 + \sqrt{\alpha_2}\epsilon_2$$

$$\mathbf{x}_2 = \sqrt{1-\alpha_2}(\sqrt{1-\alpha_1}\mathbf{x}_0 + \sqrt{\alpha_1}\epsilon_1) + \sqrt{\alpha_2}\epsilon_2$$

$$\mathbf{x}_2 = \sqrt{(1-\alpha_2)(1-\alpha_1)}\mathbf{x}_0 + \sqrt{1-\alpha_2-(1-\alpha_2)(1-\alpha_1)}\epsilon_1 + \sqrt{\alpha_2}\epsilon_2$$

Encoder

- The last two items are the summation of zero-mean Gaussian distributions:

$$\mathbf{x}_1 = \sqrt{1-\alpha_1}\mathbf{x_0} + \sqrt{\alpha_1}\epsilon_1$$

$$\mathbf{x}_2 = \sqrt{1-\alpha_2}\mathbf{x}_1 + \sqrt{\alpha_2}\epsilon_2$$

$$\mathbf{x}_2 = \sqrt{1-\alpha_2}(\sqrt{1-\alpha_1}\mathbf{x_0} + \sqrt{\alpha_1}\epsilon_1) + \sqrt{\alpha_2}\epsilon_2$$

$$\mathbf{x}_2 = \sqrt{(1-\alpha_2)(1-\alpha_1)}\mathbf{x}_0 + \sqrt{1-\alpha_2-(1-\alpha_2)(1-\alpha_1)}\epsilon_1 + \sqrt{\alpha_2}\epsilon_2$$

Encoder

- Simplify:

$$\mathbf{x}_1 = \sqrt{1-\alpha_1}\mathbf{x_0} + \sqrt{\alpha_1}\epsilon_1$$

$$\mathbf{x}_2 = \sqrt{1-\alpha_2}\mathbf{x}_1 + \sqrt{\alpha_2}\epsilon_2$$

$$\mathbf{x}_2 = \sqrt{1-\alpha_2}(\sqrt{1-\alpha_1}\mathbf{x_0} + \sqrt{\alpha_1}\epsilon_1) + \sqrt{\alpha_2}\epsilon_2$$

$$\mathbf{x}_2 = \sqrt{(1-\alpha_2)(1-\alpha_1)}\mathbf{x}_0 + \sqrt{1-\cancel{\alpha_2}-(1-\alpha_2)(1-\alpha_1)}\epsilon_1 + \cancel{\sqrt{\alpha_2}}\epsilon_2$$

$$\mathbf{x}_2 = \sqrt{(1-\alpha_2)(1-\alpha_1)}\mathbf{x}_0 + \sqrt{1-(1-\alpha_2)(1-\alpha_1)}\epsilon$$

Encoder

- If we keep doing the same process, we can directly find $\mathbf{x}_t$:

$$\mathbf{x}_2 = \sqrt{(1-\alpha_2)(1-\alpha_1)}\mathbf{x}_0 + \sqrt{1-(1-\alpha_2)(1-\alpha_1)}\epsilon$$

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon \text{ where } \bar{\alpha}_t = \prod_{s=1}^{t} 1-\alpha_s$$

Encoder

- If we keep doing the same process, we can directly find $\mathbf{x}_t$:

$$\mathbf{x}_2 = \sqrt{(1-\alpha_2)(1-\alpha_1)}\mathbf{x}_0 + \sqrt{1-(1-\alpha_2)(1-\alpha_1)}\epsilon$$

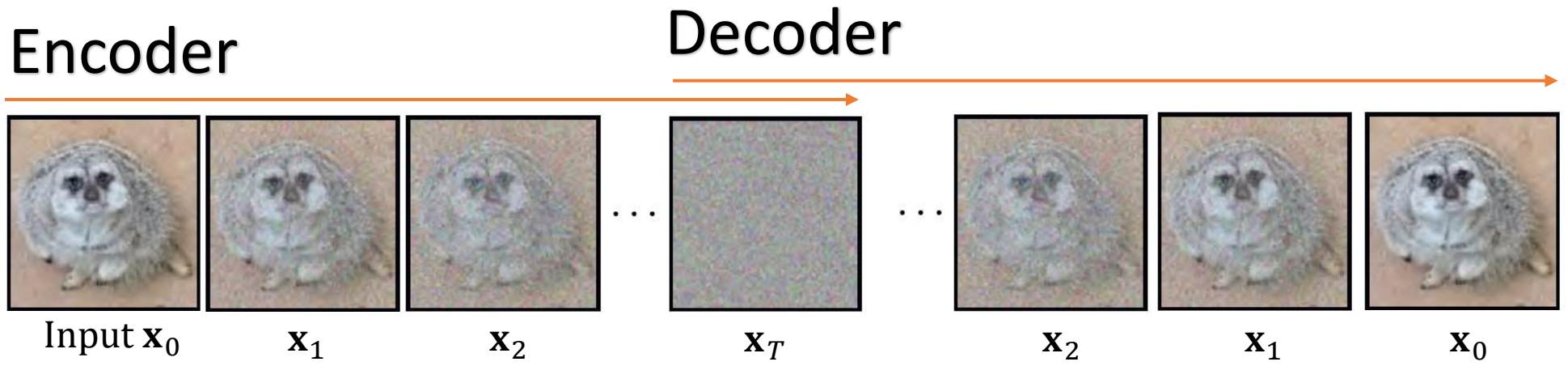
$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon \text{ where } \bar{\alpha}_t = \prod_{s=1}^{t} 1-\alpha_s$$

- Therefore, the probability density distribution is:

$$\mathbf{q}(\mathbf{x}_t|\mathbf{x_0}) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\mathbf{I})$$

Decoder

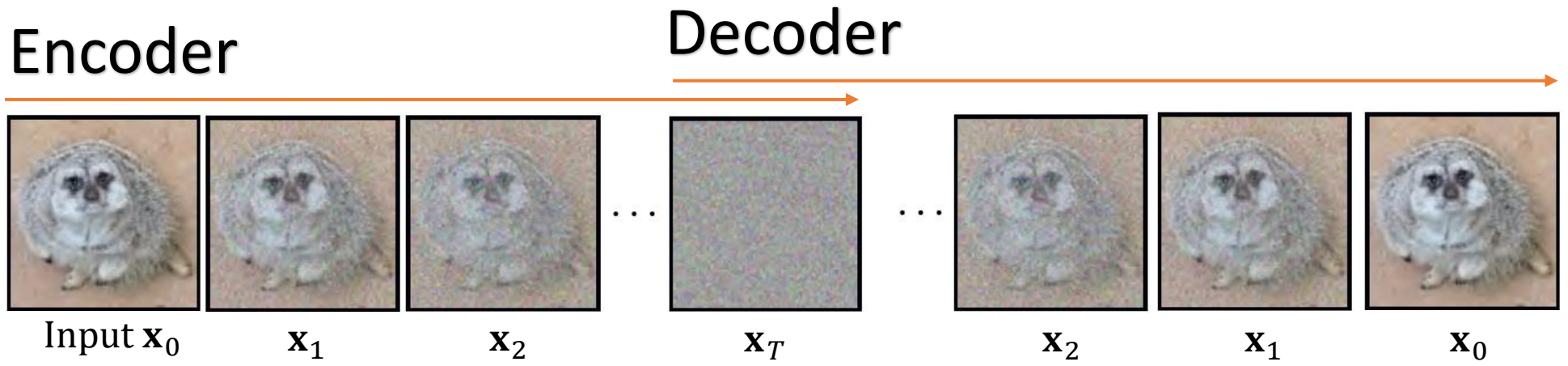
- Decoder is learned. It gradually removes noise from Gaussian noise $\mathbf{x}_T$.



Decoder

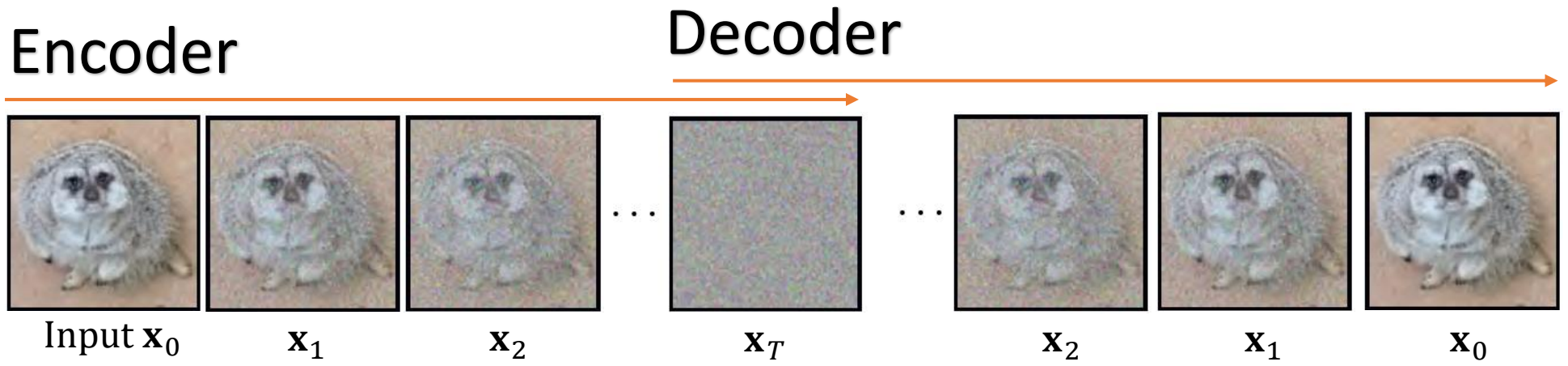
- Decoder is learned. It gradually removes noise from Gaussian noise $\mathbf{x}_T$.

- After several steps a new image produced.



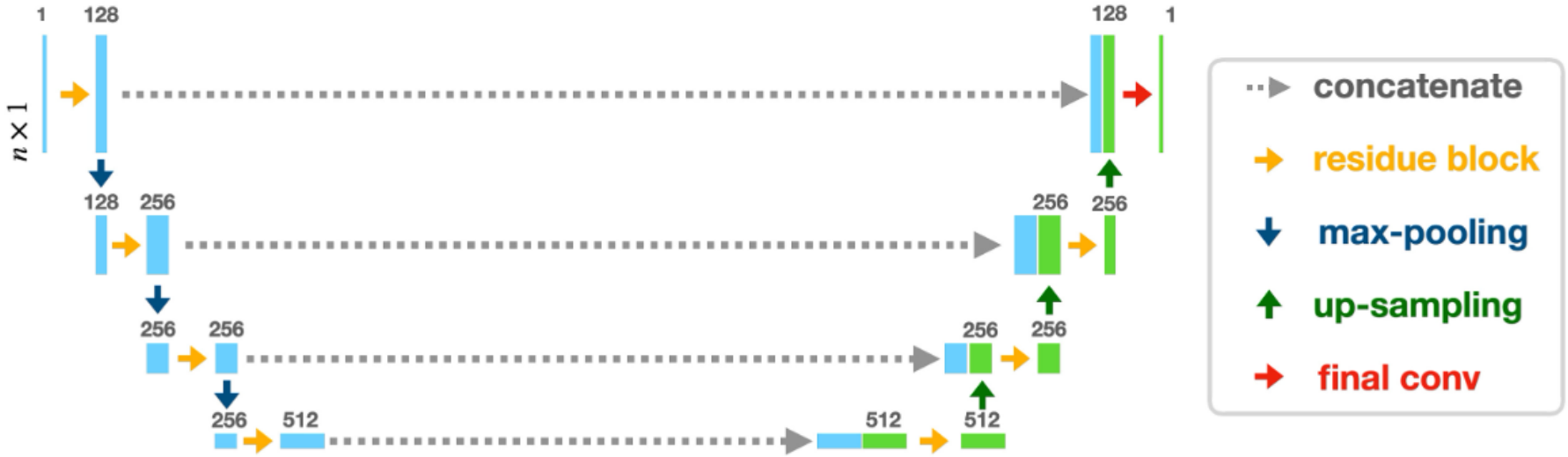
Decoder

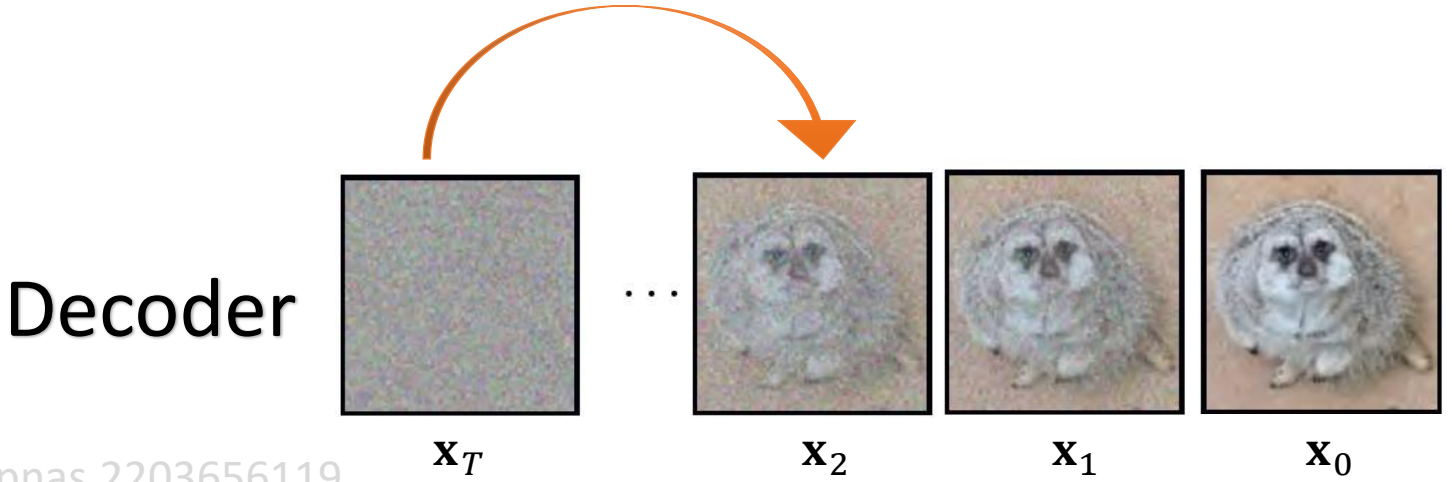
- Since we are learning, a network is needed along with a loss function.



Decoder

- Network: Unet

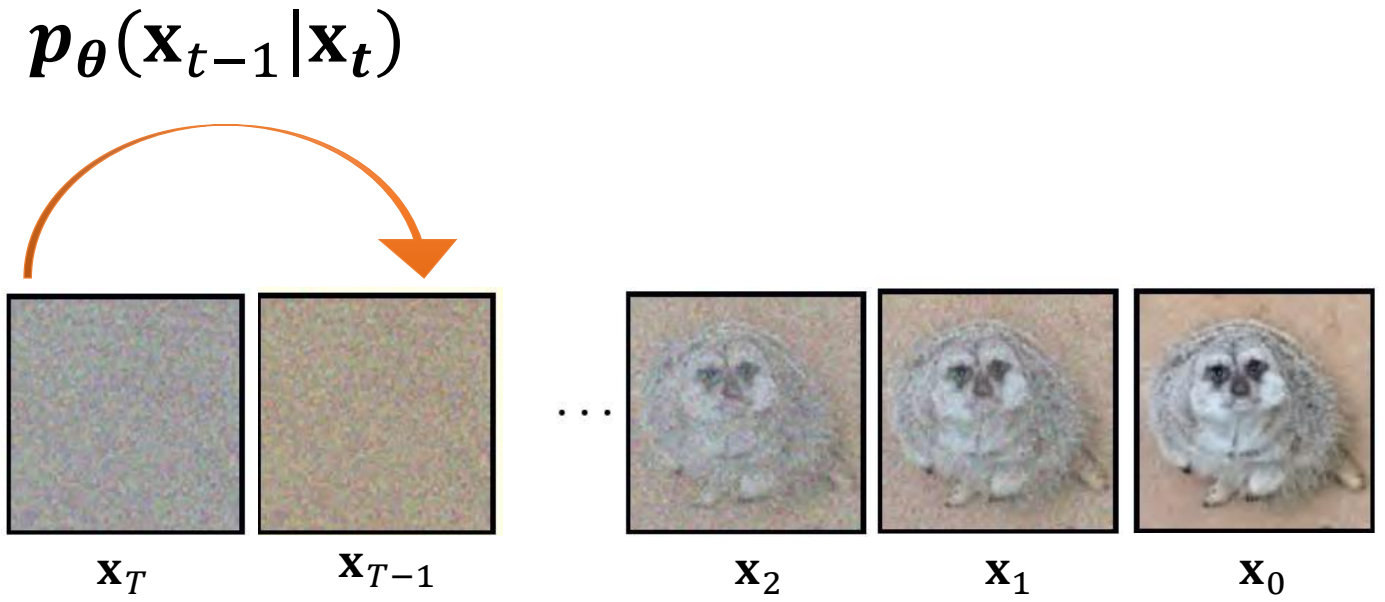




https://www.pnas.org/doi/10.1073/pnas.2203656119

Decoder-Loss

- Loss function: what should we maximize?



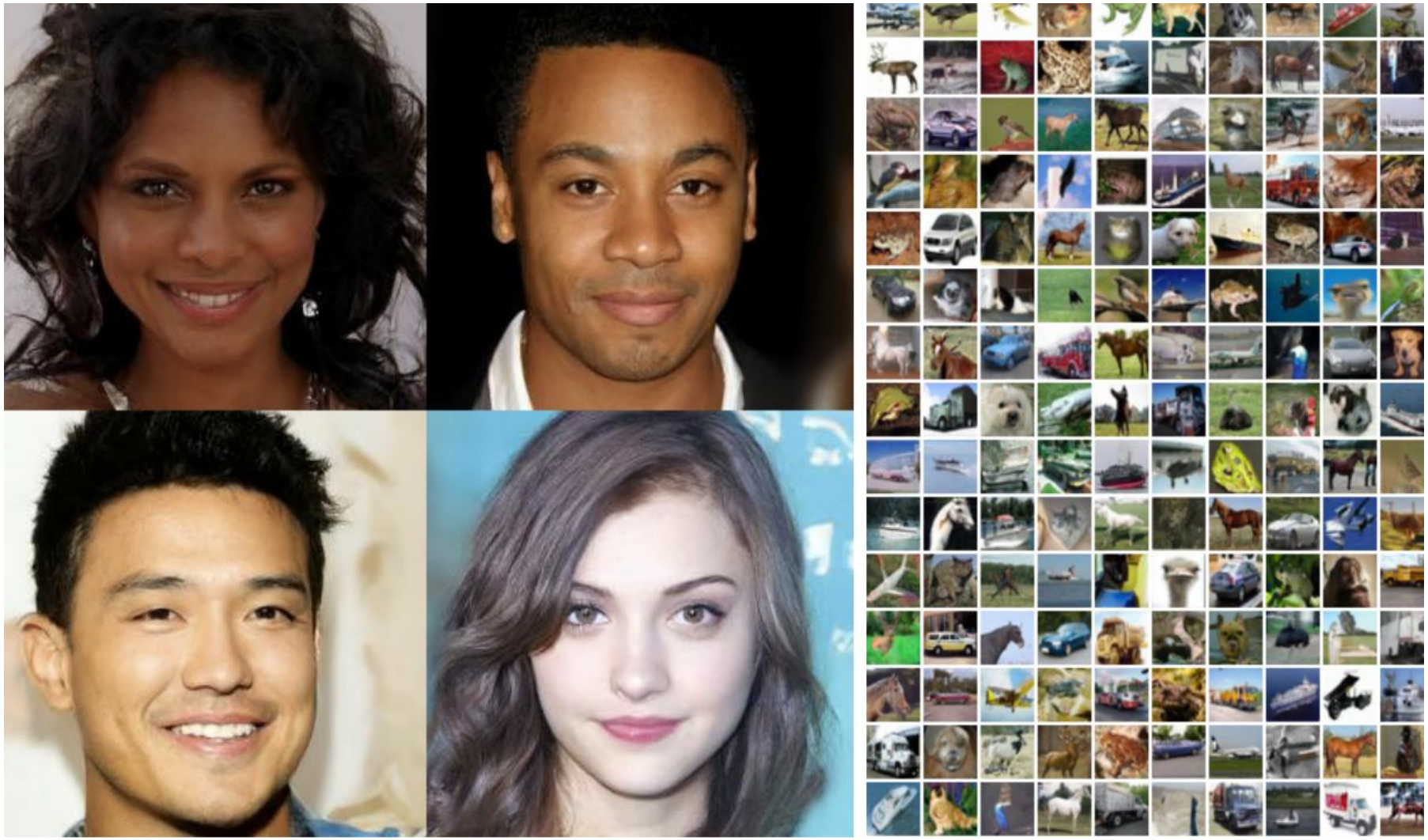
Decoder-Loss

- If you simplify the loss, you will get

$$E\left[\left\|\epsilon_t - \epsilon_\theta(\bar{\alpha}_t x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t)\right\|^2\right]$$

- You basically predict the noise that should be removed from the image.

Denoising Diffusion Probabilistic Models DDPM



https://arxiv.org/pdf/2006.11239.pdf

Dall E

Hierarchical Text-Conditional Image Generation with CLIP Latents

Aditya Ramesh*
OpenAI
aramesh@openai.com

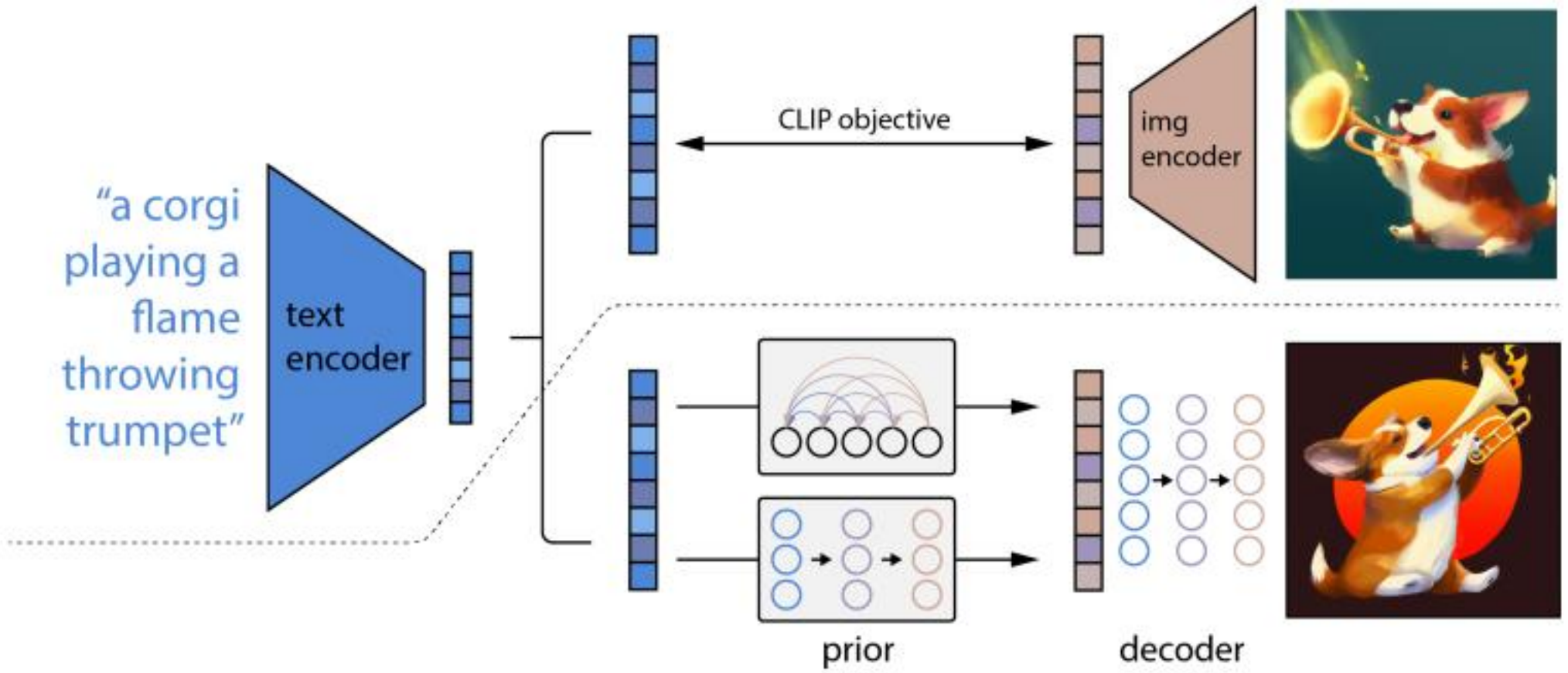
Prafulla Dhariwal*
OpenAI
prafulla@openai.com

Alex Nichol*
OpenAI
alex@openai.com

Casey Chu*
OpenAI
casey@openai.com

Mark Chen
OpenAI
mark@openai.com

Dall E


"a corgi playing a flame throwing trumpet"
text encoder
CLIP objective
img encoder
prior
decoder

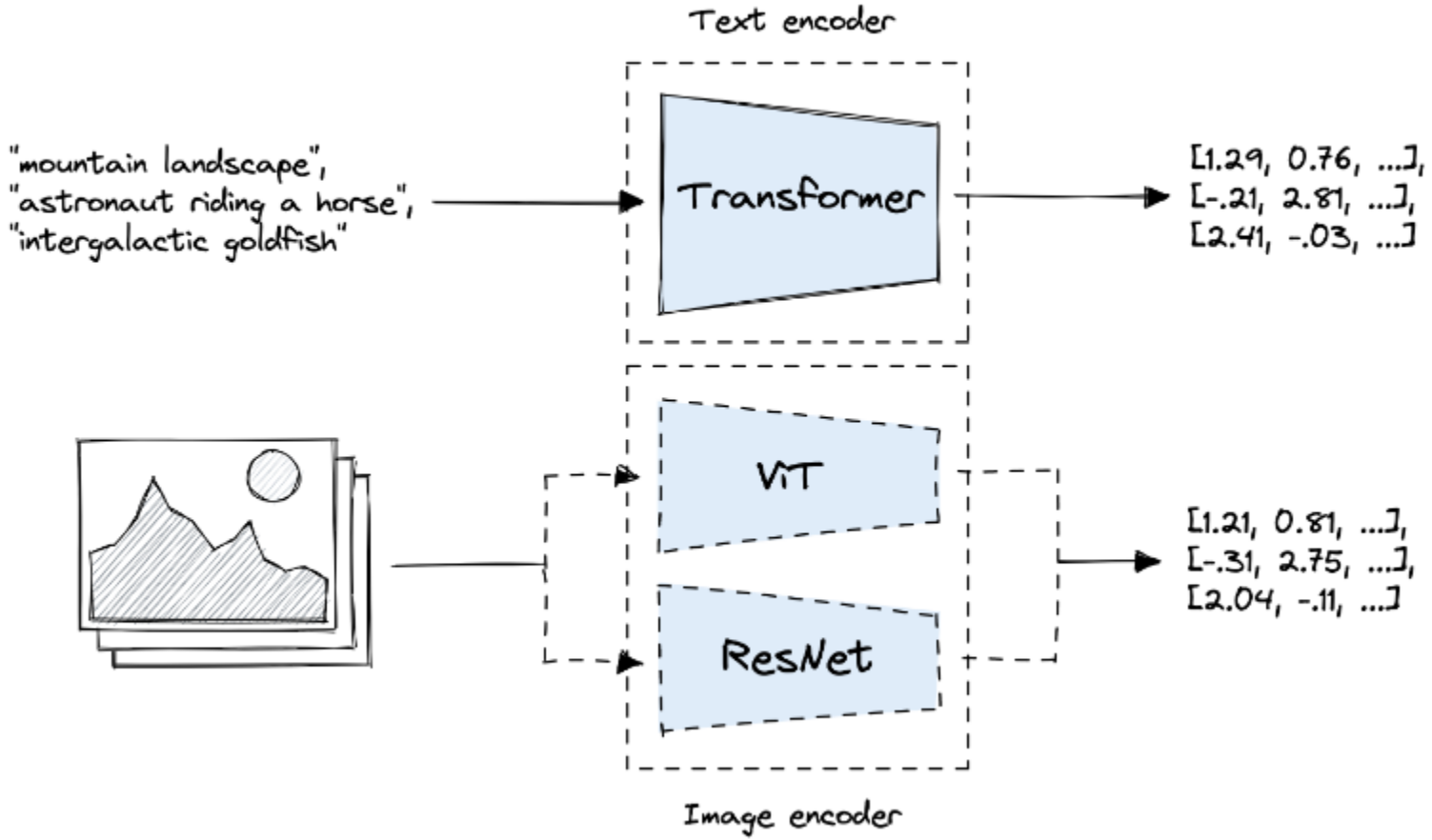
CLIP

- Contrastive Language-Image Pre-Training. It is a deep learning model developed by OpenAI that can understand both images and text in a unified way.

- CLIP is trained using a contrastive approach, where it learns to associate images and text that refer to the same concepts or objects in the world.

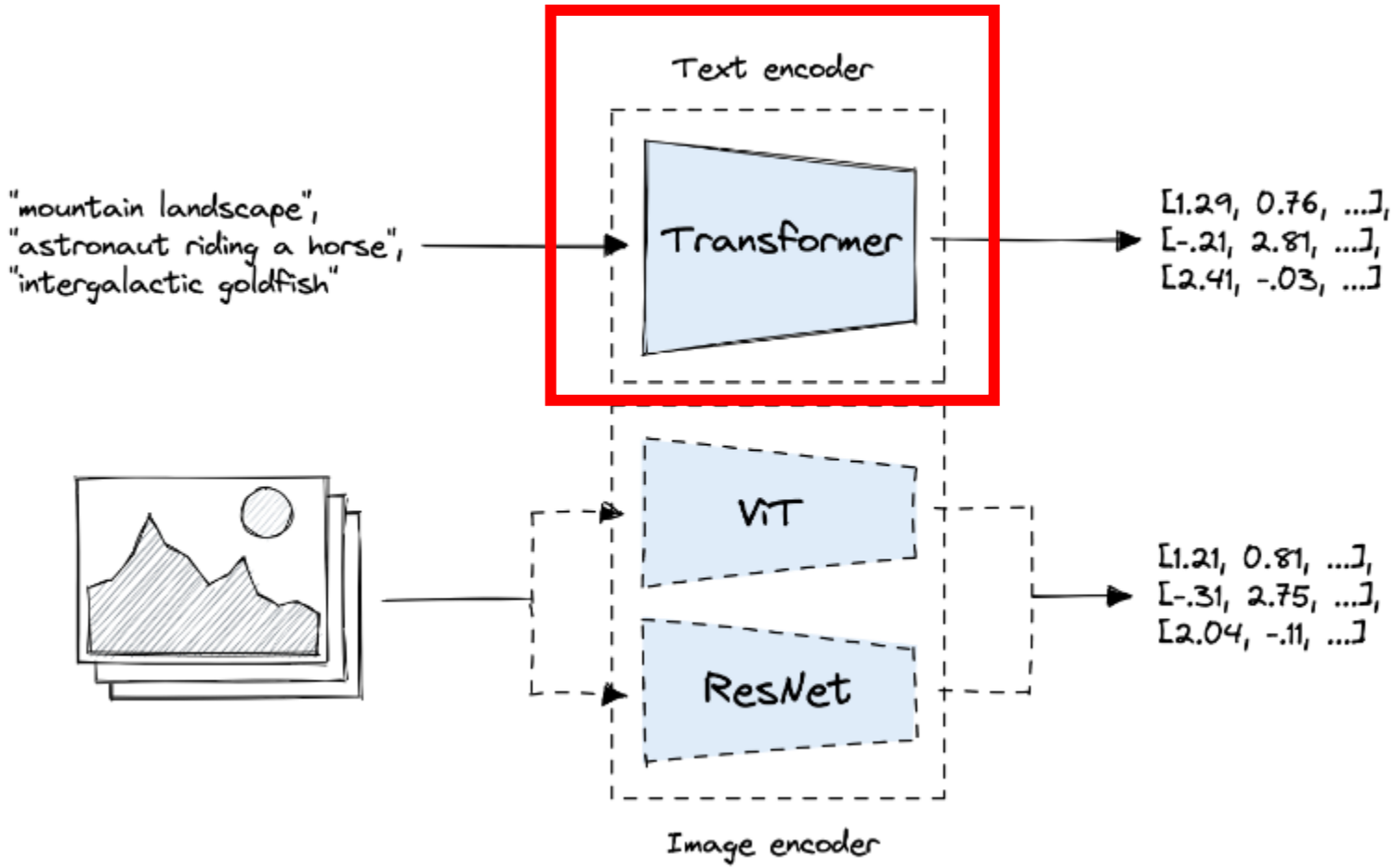
CLIP

- CLIP consists of two models trained in parallel.



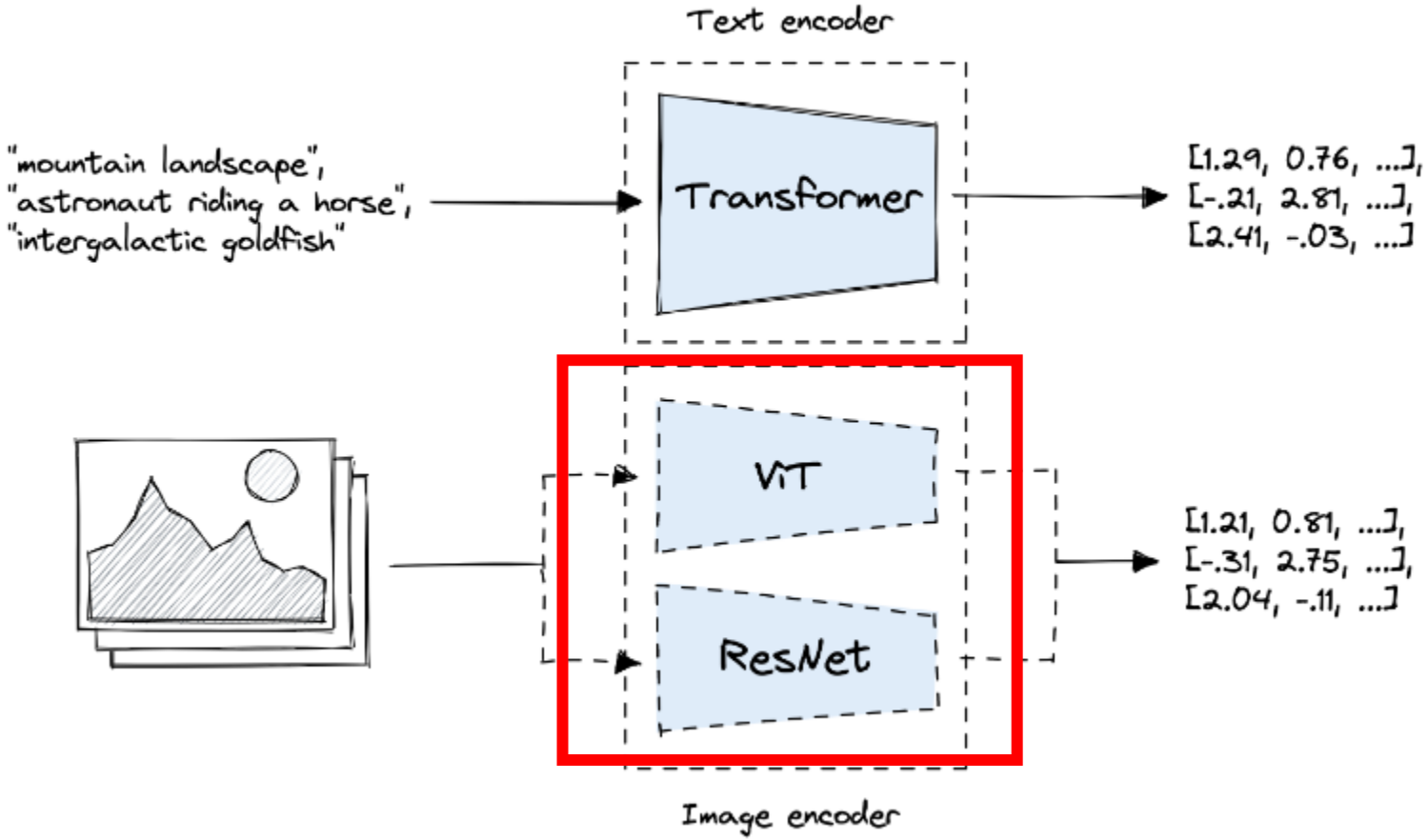
CLIP

- CLIP consists of two models trained in parallel.
- Text transformer for building text embeddings.



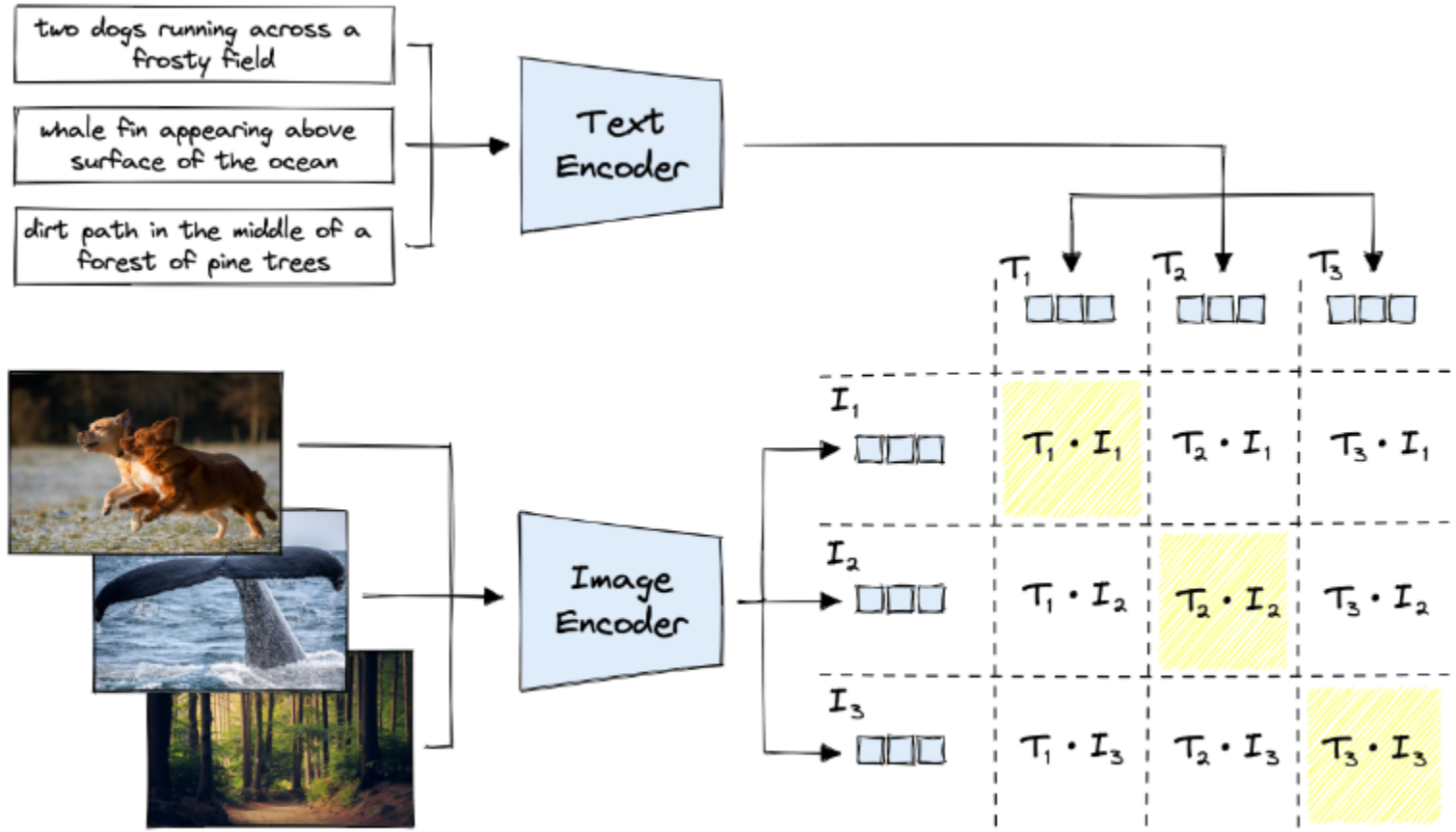
CLIP

- CLIP consists of two models trained in parallel.
- Text transformer for building text embeddings.
- ResNet or vision transformer (ViT) for building image embeddings.



CLIP

- A loss function maximizes the similarity between (T1,I1) and (T2,I2)
- and minimizes the similarity between (T1,I2) and (T2, I1) (contrastive learning)



Contrastive pretraining with CLIP.

Results

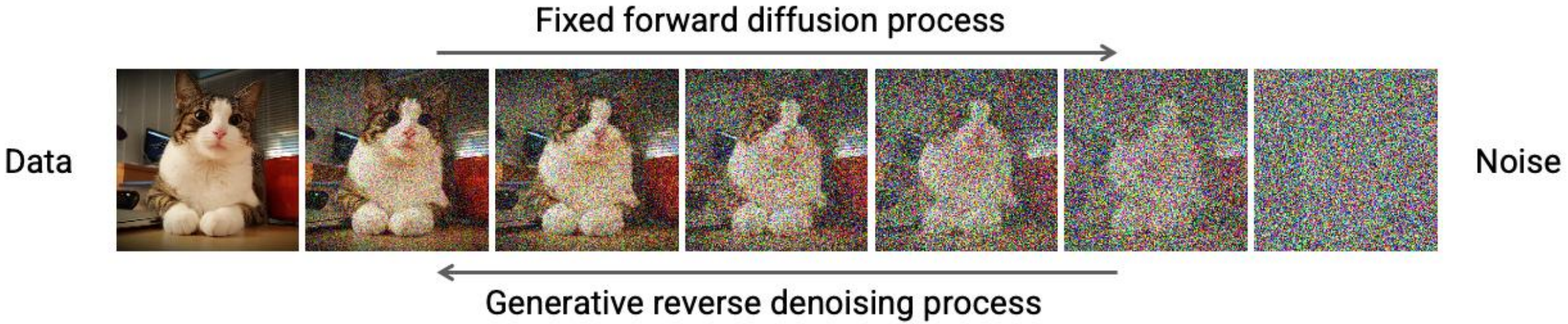
vibrant portrait painting of Salvador Dalí with a robotic half face

a shiba inu wearing a beret and black turtleneck

a close up of a handpalm with leaves growing from it

Stable Diffusion

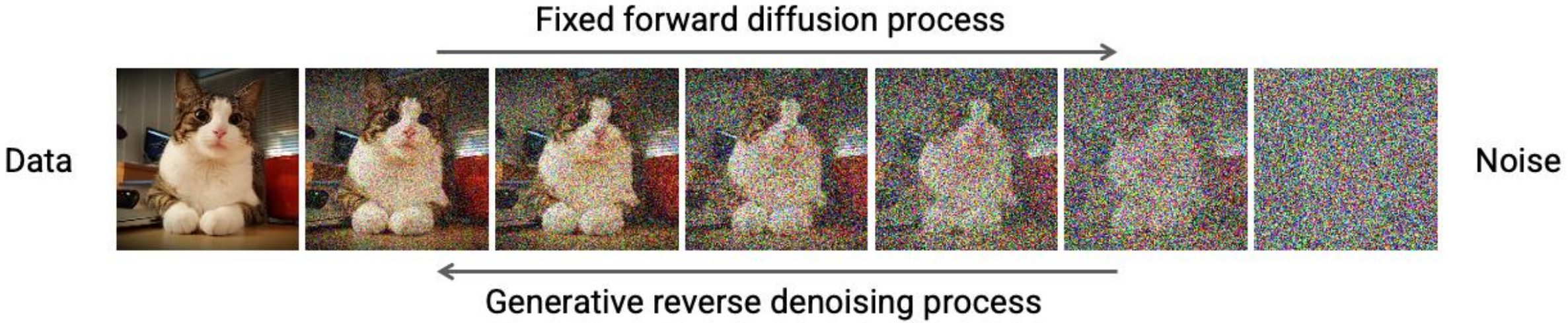
- Noising and denoising take place on the image space.



Stable Diffusion

- Noising and denoising take place on the image space.

- Time consuming and memory hungry.



Stable Diffusion

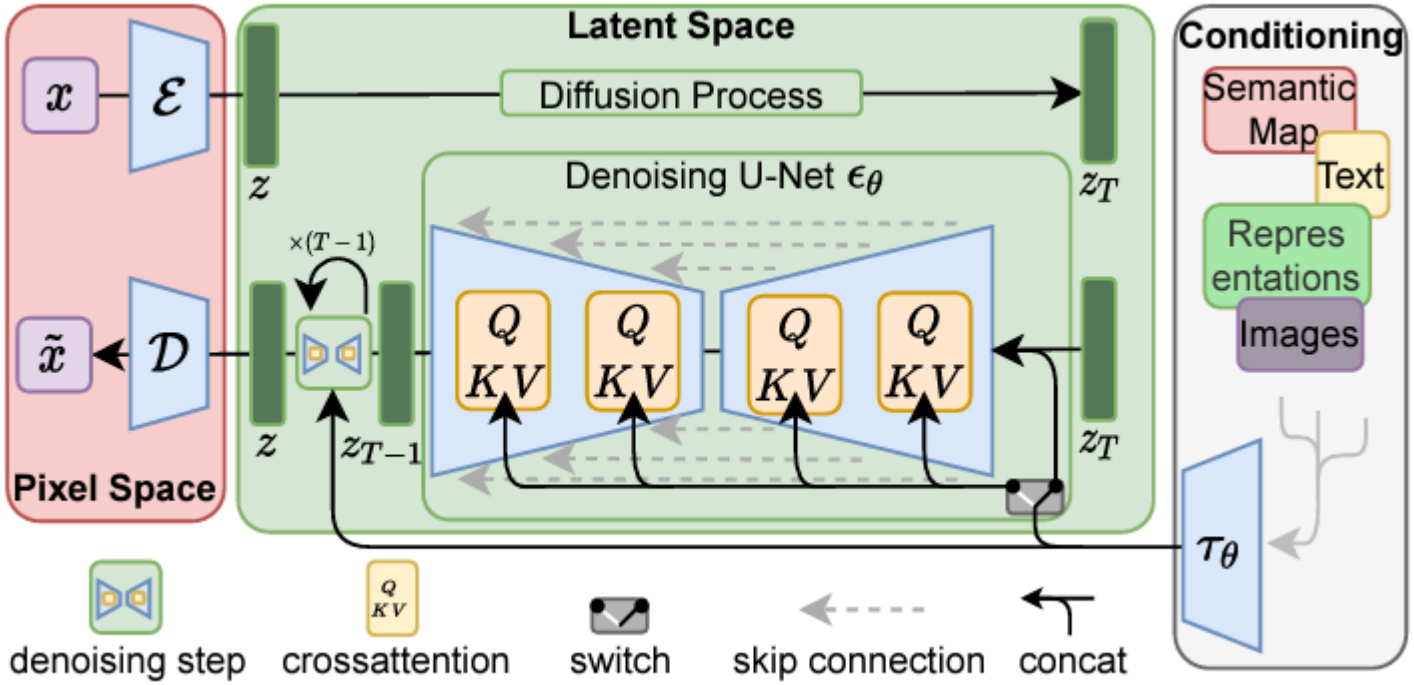
High-Resolution Image Synthesis with Latent Diffusion Models

Robin Rombach[1] * Andreas Blattmann[1] * Dominik Lorenz[1] Patrick Esser[®] Björn Ommer[1]
[1]Ludwig Maximilian University of Munich & IWR, Heidelberg University, Germany [®]Runway ML

https://github.com/CompVis/latent-diffusion

Stable Diffusion

- Encoder and decoder are fixed and noising denoising happen on the latent space.



Stable Diffusion

- Results