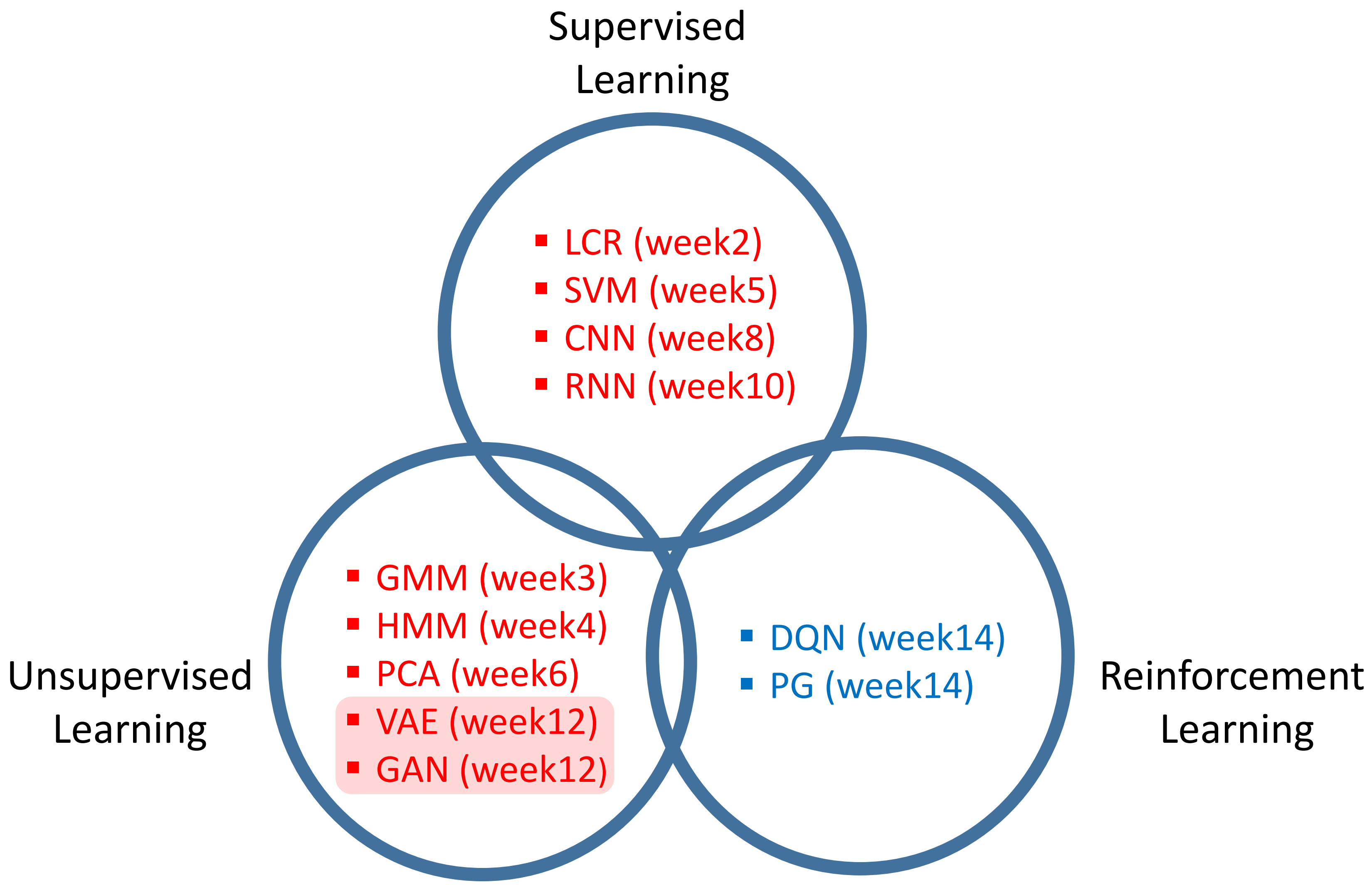
Practical Machine Learning

Lecture 12
Generative Models: Variational Auto Encoder (VAE) and Generative Adversarial Networks (GAN)

Dr. Suyong Eum

OSAKA UNIVERSITY

Where we are

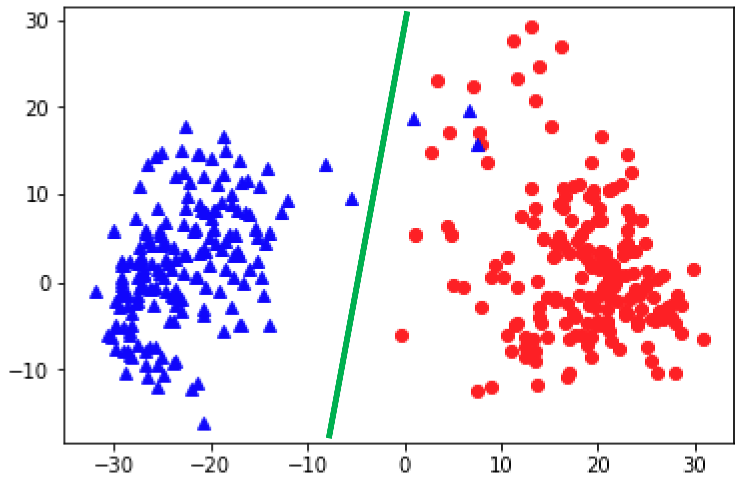

Supervised Learning
LCR (week2)
SVM (week5)
CNN (week8)
RNN (week10)
Unsupervised Learning
GMM (week3)
HMM (week4)
PCA (week6)
VAE (week12)
GAN (week12)
DQN (week14)
PG (week14)
Reinforcement Learning

You are going to learn

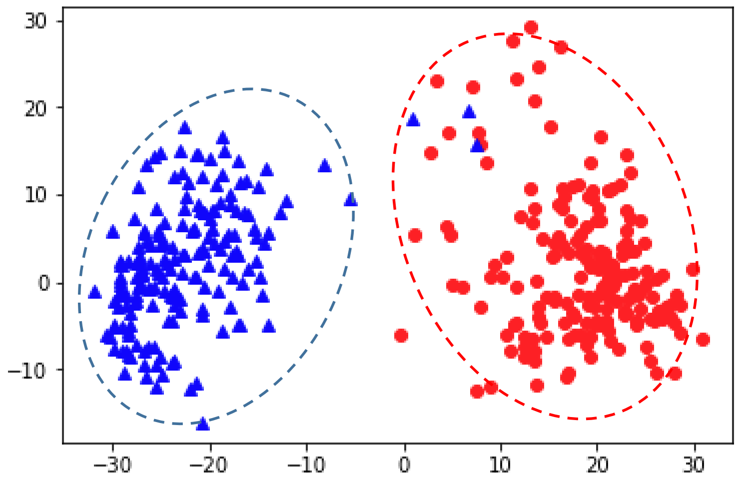
- The basic concept of generative models
- Two generative models:
  1) Variational Auto Encoder (VAE)
  2) Generative Adversarial Networks (GAN)
- Some applications you may be interested

What are Generative models?

- There are two types of models:
  1) Discriminative models
  2) Generative models



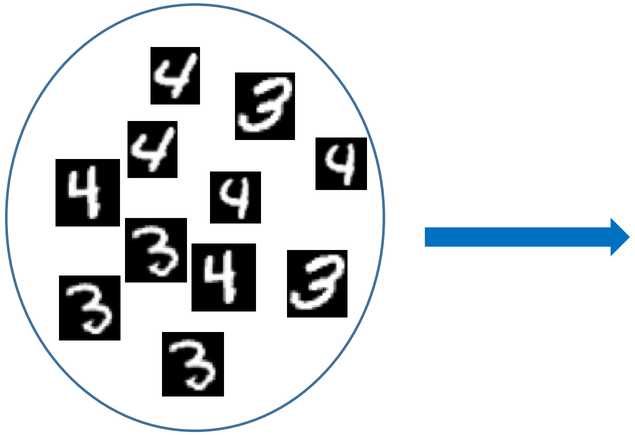
Determinative model, e.g., SVM



Generative model, e.g, GMM

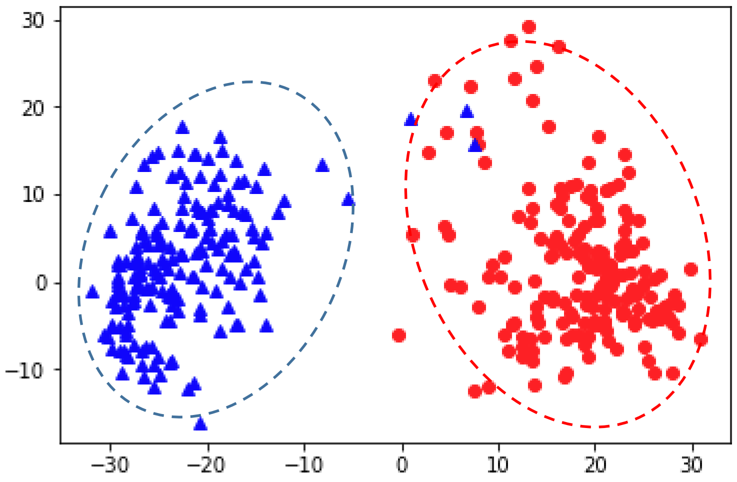
Basic operation

- Literally speaking, a sample can be generated from generative models.
  - Of course, the model needs to be trained in advance to generate such a sample which you are interested.



Each data point has 64 dimension

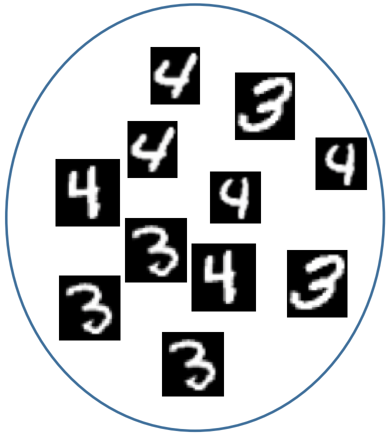
e.g., GMM



Project the data points in 2 dimension

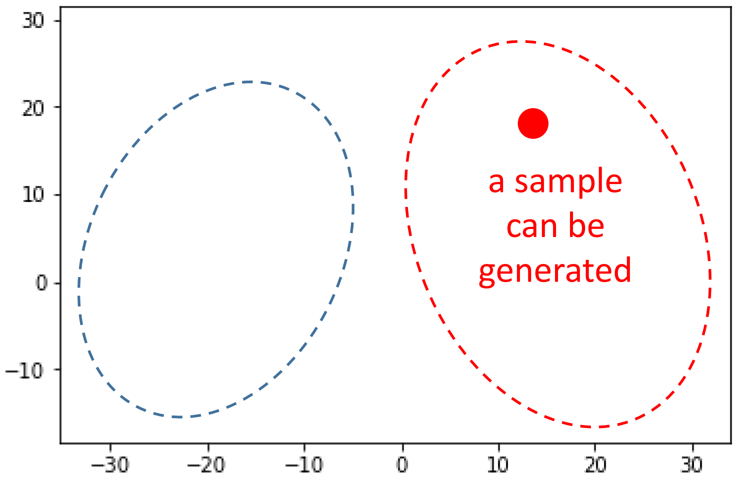
Basic operation

- Literally speaking, a sample can be generated from generative models.
  - Of course, the model needs to be trained in advance to generate such a sample which you are interested.

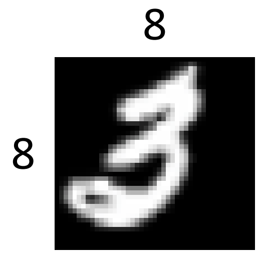


Each data point has 64 dimension

e.g., GMM



Project the data point in 2 dimension



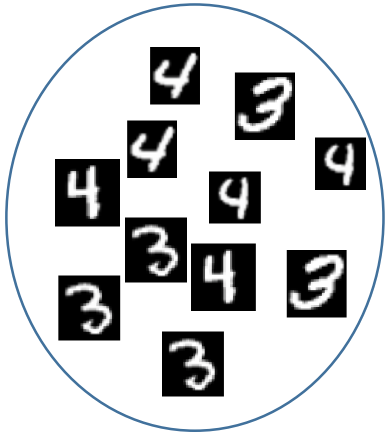
64 dimensions

- We can generate a new image which corresponds to the sample.
- It is NOT one of training data points!

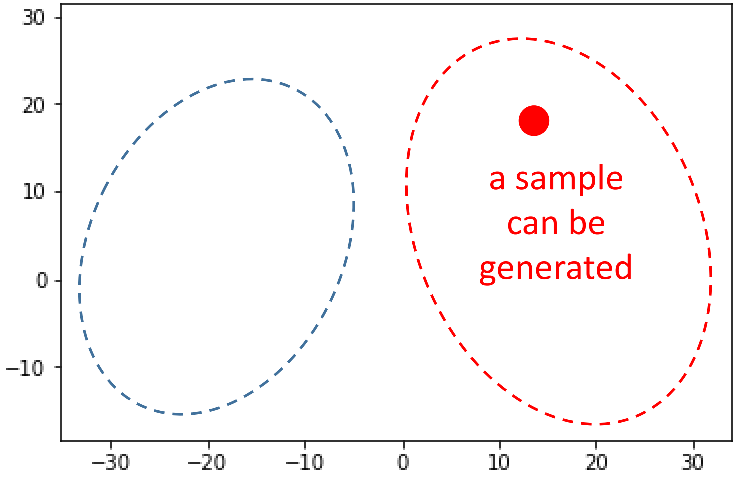
Variational AutoEncoder (VAE)

Idea of VAE

- To build a method which does the following procedure systematically.



Each data point
has 64 dimension



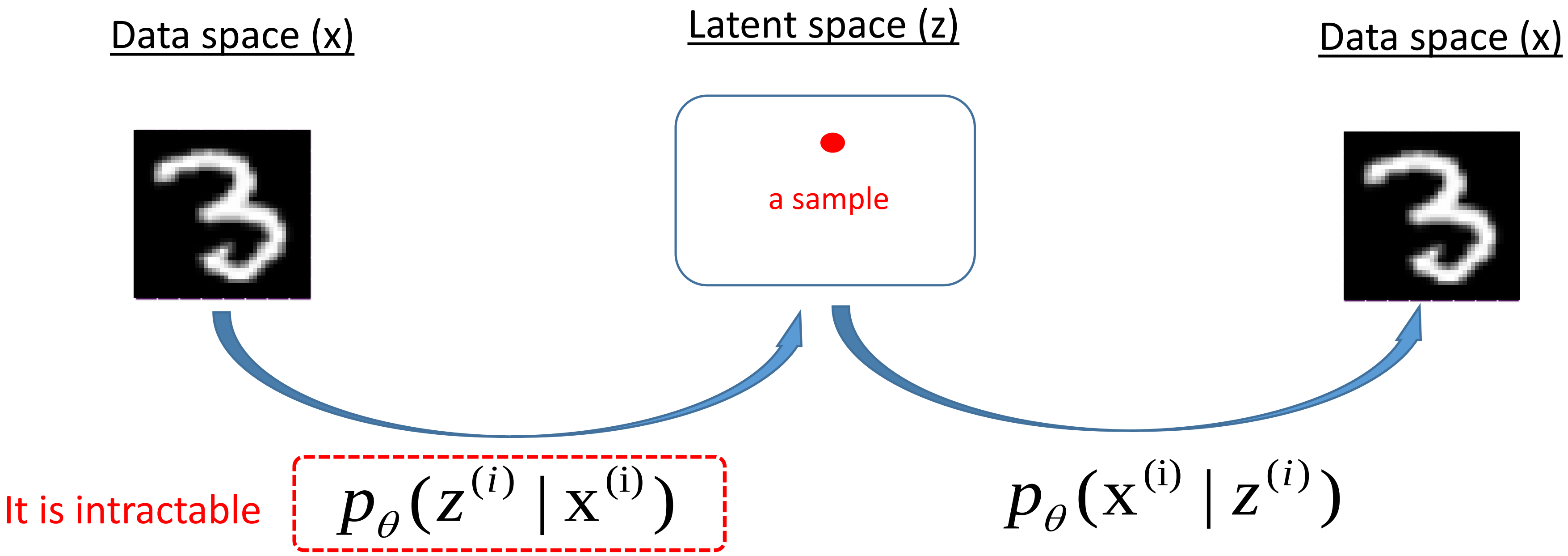
Project the data point in
2 dimension

64 dimensions

- We can generate a new image which corresponds to the sample.
- It is NOT one of training data points!

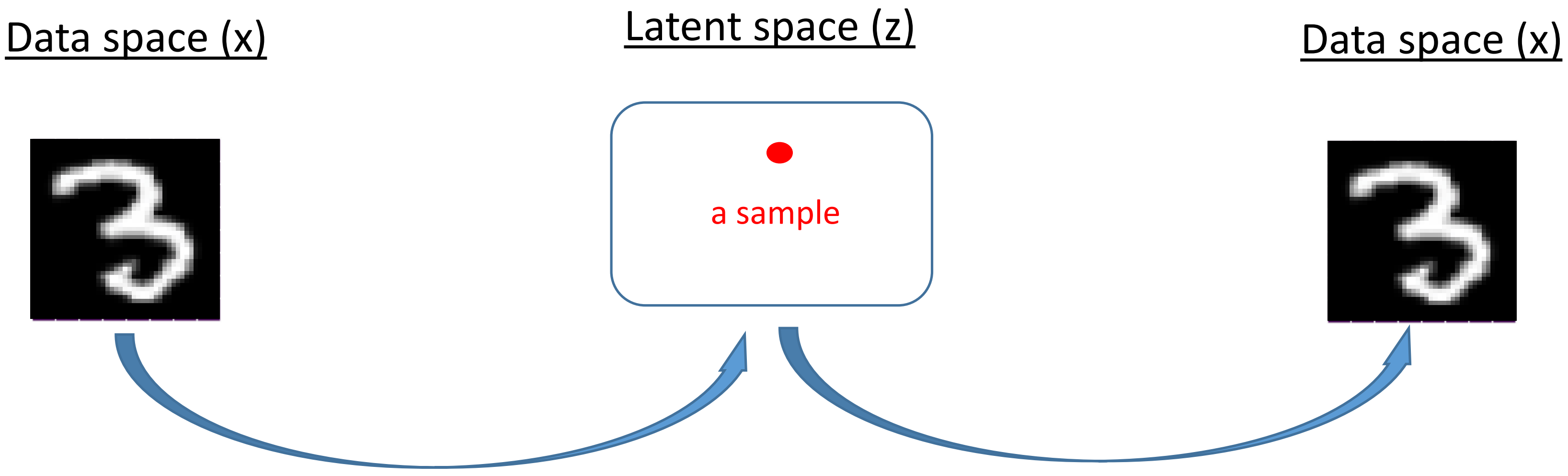
Idea of VAE

- Assuming that there is a complex model parameterized with "θ"
- The model generates data $\{x^{(1)}, x^{(2)}, \dots, x^{(N)}\}$ given a latent variable "z": $p_\theta(x|z)$
- Also, the model maps data set into the latent space: $p_\theta(z|x)$



Idea of VAE

- This Intractability is well known, which can be handled with 1) Markov Chain Monte Carlos (MCMC) and 2) Variational Inference (VI).
- VAE uses the idea of Variational Inference and so the term “Variational” is in the name.

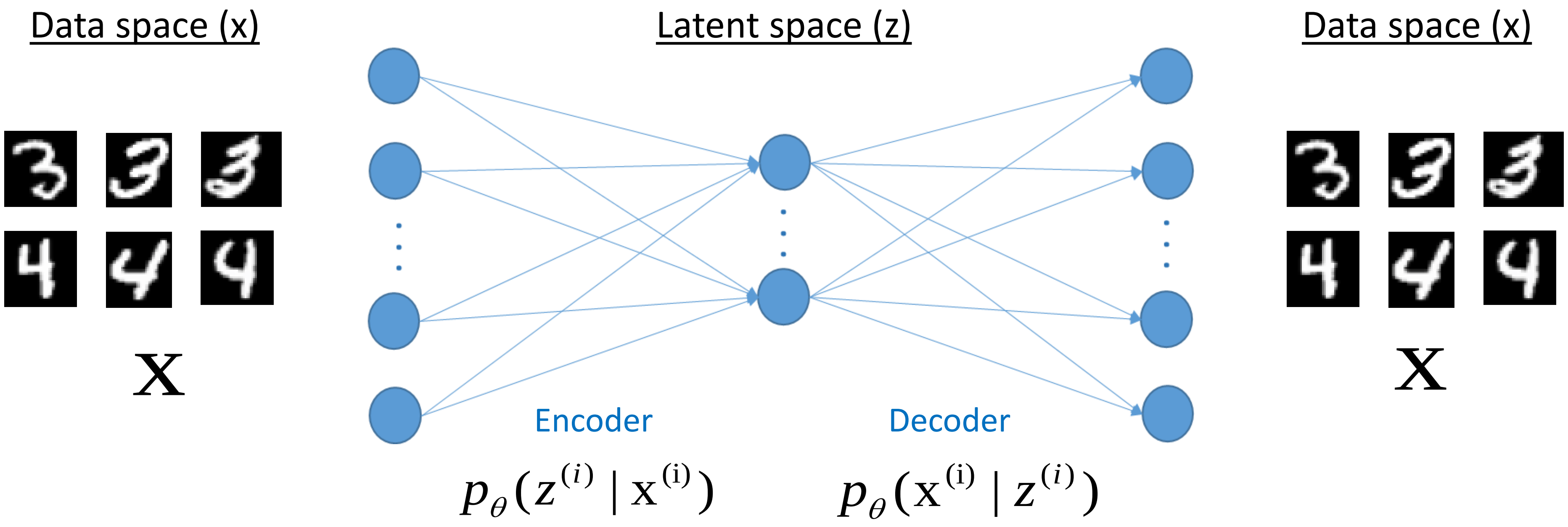


It is intractable $p_\theta(z^{(i)} \mid \mathrm{x}^{(\mathrm{i})})$

$p_\theta(\mathrm{x}^{(\mathrm{i})} \mid z^{(i)})$

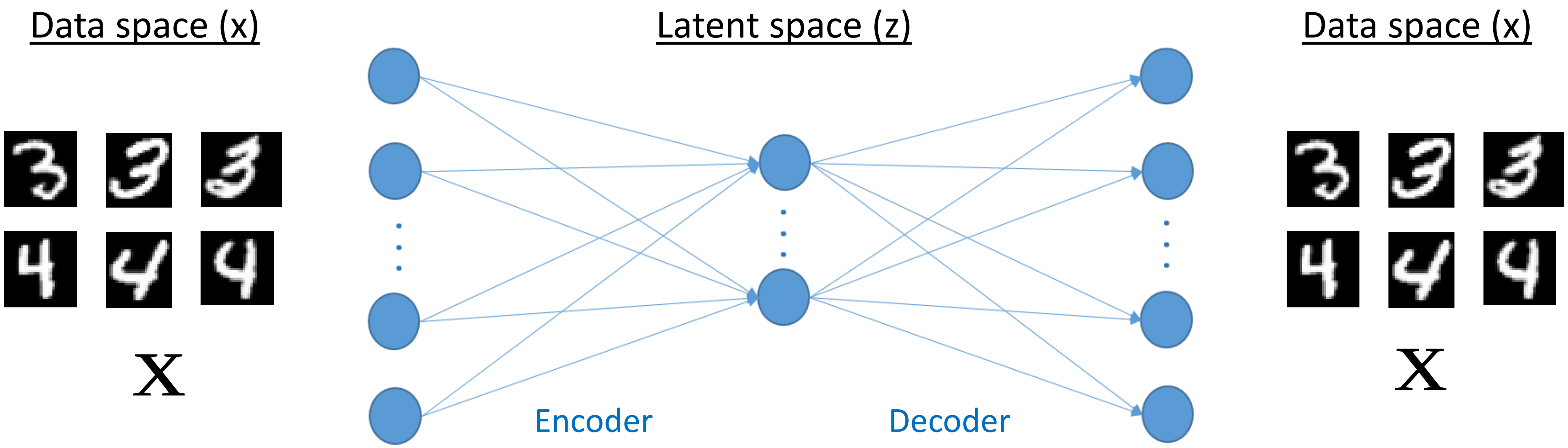
Approximate the function "p" using a neural network

- Auto Encoder is a neural network which reproduces its input



Approximate the function "p" using a neural network

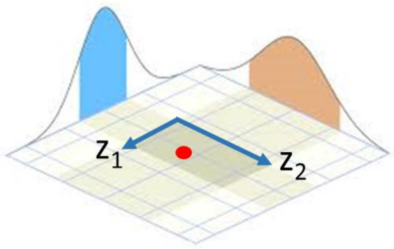
- Auto Encoder is a neural network which reproduces its input



$p_\theta(z^{(i)} \mid x^{(i)})$ $\quad$ $p_\theta(x^{(i)} \mid z^{(i)})$

approximation

$q_\varphi(z^{(i)} \mid x^{(i)})$



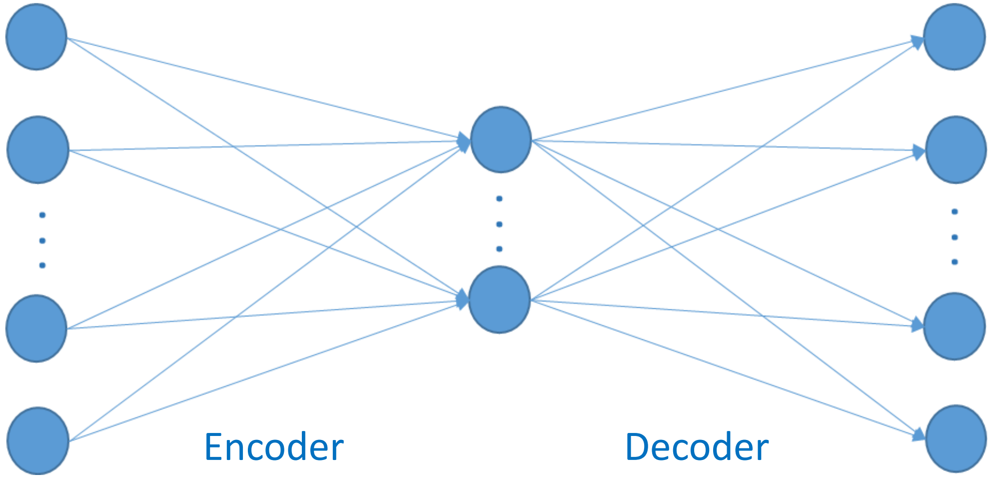
- Assuming $q_\varphi()$ as Gaussian distribution

Approximate the function "p" using a neural network

- Auto Encoder is a neural network which reproduces its input

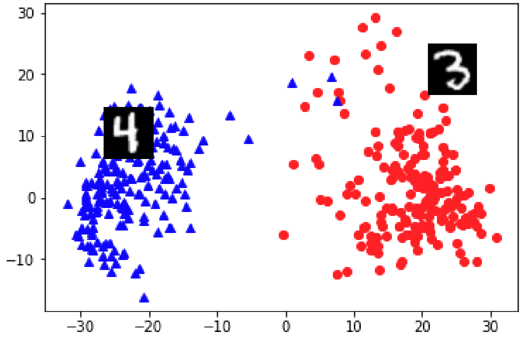
Data space (x) | Latent space (z) | Data space (x)

$\mathbf{x}$



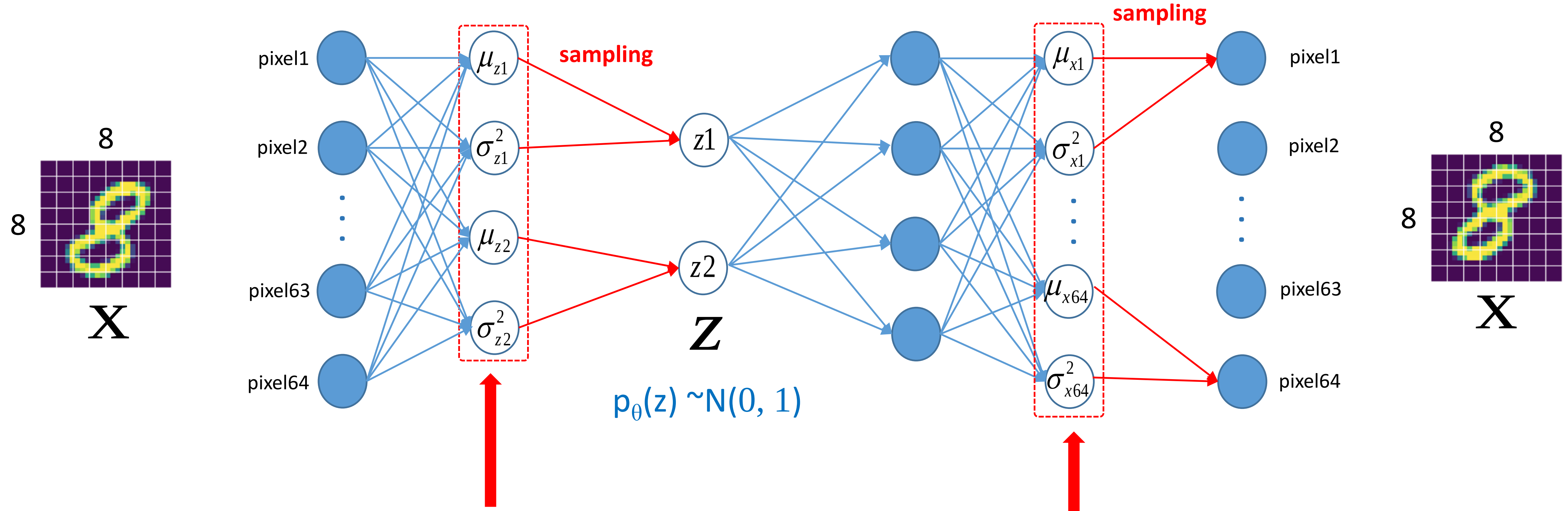
$\mathbf{x}$

$q_\varphi(z^{(i)} \mid \mathrm{x}^{(i)})$ $\quad$ $p_\theta(\mathrm{x}^{(i)} \mid z^{(i)})$



2 dimensions

VAE model



$q_\varphi(z|x)$ ~ $N(\mu, \sigma^2)$

$\{w, b\} \in \varphi$

$p_\theta(x|z)$ ~ $N(\mu, \sigma^2)$ or Bernoulli

$\{w, b\} \in \theta$

A loss function for AutoEncoder

- How can we train the network to obtain the parameter φ and θ ?
  - To train the neural network, a loss function is necessary. Then, the parameter "φ and θ" can be calculated through a backpropagation.
- Let's derive the loss function from the likelihood function $p_\theta(x)$

$$p_\theta(\mathrm{x}^{(1)},\cdots,\mathrm{x}^{(N)}) = \prod_{i=1}^{N} \log p_\theta(\mathrm{x}^{(\mathrm{i})})$$

→ Likelihood function showing the probability that given batch data set occur with the parameter θ in the neural network.

$$\log p_\theta(\mathrm{x}^{(1)},\cdots,\mathrm{x}^{(N)}) = \sum_{i=1}^{N} \log p_\theta(\mathrm{x}^{(\mathrm{i})})$$

→ (log) likelihood

$$\log p_\theta(\mathrm{x}^{(\mathrm{i})}) = L(\theta,\varphi;\mathrm{x}^{(\mathrm{i})}) + D_{KL}(q_\varphi(z\,|\,\mathrm{x}^{(\mathrm{i})})\,||\,p_\theta(z\,|\,\mathrm{x}^{(\mathrm{i})}))$$

- Since $D_{KL} \geq 0$, "L" is the lower bound of the likelihood function, which is called "ELBO" (Evidence Lower Bound)

- Kullback-Leibnitz divergence showing how difference between two posterior distributions: true posterior p(z|x) and its approximate posterior q(z|x)
- This term is intractable because of p(z|x).
- However, we know $D_{KL} \geq 0$.

$$\log p_\theta(\mathrm{x}^{(1)},\cdots,\mathrm{x}^{(N)}) = \sum_{i=1}^{N} \log p_\theta(\mathrm{x}^{(i)})$$

Proof: If you are interested…

$$\log p(\mathrm{x}) = \sum_z q(z|\mathrm{x}) \log p(\mathrm{x})$$

$$= \sum_z q(z|\mathrm{x}) \log\left(\frac{p(z,\mathrm{x})}{\mathrm{p}(z|\mathrm{x})}\right)$$

$$= \sum_z q(z|\mathrm{x}) \log\left(\frac{p(z,\mathrm{x})}{\mathrm{q}(z|\mathrm{x})} \frac{q(z|\mathrm{x})}{\mathrm{p}(z|\mathrm{x})}\right)$$

$$= \sum_z q(z|\mathrm{x}) \log\left(\frac{p(z,\mathrm{x})}{\mathrm{q}(z|\mathrm{x})}\right) + \sum_z q(z|\mathrm{x}) \log\left(\frac{q(z|\mathrm{x})}{\mathrm{p}(z|\mathrm{x})}\right)$$

$$p(x,z) = p(z,x)$$

$$p(x,z) = p(z|x)p(x)$$

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

$$\log p_\theta(\mathrm{x}^{(i)}) = L(\theta,\varphi;\mathrm{x}^{(i)}) + D_{KL}(q_\varphi(z|\mathrm{x}^{(i)}) \,||\, p_\theta(z|\mathrm{x}^{(i)}))$$

A loss function for AutoEncoder

- By maximizing "L", we can maximize the likelihood function as well

$$\log p_\theta(\mathrm{x}^{(\mathrm{i})}) \geq L(\theta, \varphi; \mathrm{x}^{(\mathrm{i})})$$

Lower bound of the likelihood function

$$L(\theta, \varphi; \mathrm{x}^{(\mathrm{i})}) = \sum_z q_\varphi(z \mid \mathrm{x}) \log\left(\frac{p_\theta(\mathrm{z}, \mathrm{x})}{\mathrm{q}_\varphi(\mathrm{z} \mid \mathrm{x})}\right)$$

$$= \sum_z q_\varphi(z \mid \mathrm{x}) \log\left(\frac{p_\theta(\mathrm{x} \mid \mathrm{z}) p_\theta(\mathrm{z})}{\mathrm{q}_\varphi(\mathrm{z} \mid \mathrm{x})}\right)$$

$$= \sum_z q_\varphi(z \mid \mathrm{x}) \log\left(\frac{p_\theta(\mathrm{z})}{\mathrm{q}_\varphi(\mathrm{z} \mid \mathrm{x})}\right) + \sum_z q_\varphi(z \mid \mathrm{x}) \log\left(p_\theta(\mathrm{x} \mid \mathrm{z})\right)$$

$$= -D_{KL}(q(z \mid \mathrm{x}^{(\mathrm{i})}) \,\|\, p(z)) + E_{q(z \mid \mathrm{x}^{(\mathrm{i})})}(\log p(\mathrm{x}^{(\mathrm{i})} \mid z))$$

A loss function for AutoEncoder

- By maximizing "L", we can maximize the likelihood function as well,
- In other words, the most likely model (p_θ and p_φ), which generates the observed data, can be obtained by maximizing the function below.

$$\log p_\theta(\mathrm{x}^{(i)}) \geq -D_{KL}(q_\varphi(z|\mathrm{x}^{(i)}) \| p_\theta(z)) + E_{q_\varphi(z|\mathrm{x}^{(i)})}(\log p_\theta(\mathrm{x}^{(i)}|z))$$

- It can be computed in a closed form

- $q_\varphi(z|x) \sim N(\mu, \sigma^2)$
- $p_\theta(z) \sim N(0, I)$
- J: dimension of z

$$= \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log((\sigma_{z_j}^{(i)})^2) - (\mu_{z_j}^{(i)})^2 - (\sigma_{z_j}^{(i)})^2\right)$$

- $p_\theta(x|z) \sim N(\mu, \sigma^2)$ or Bernoulli
- D: dimension of x

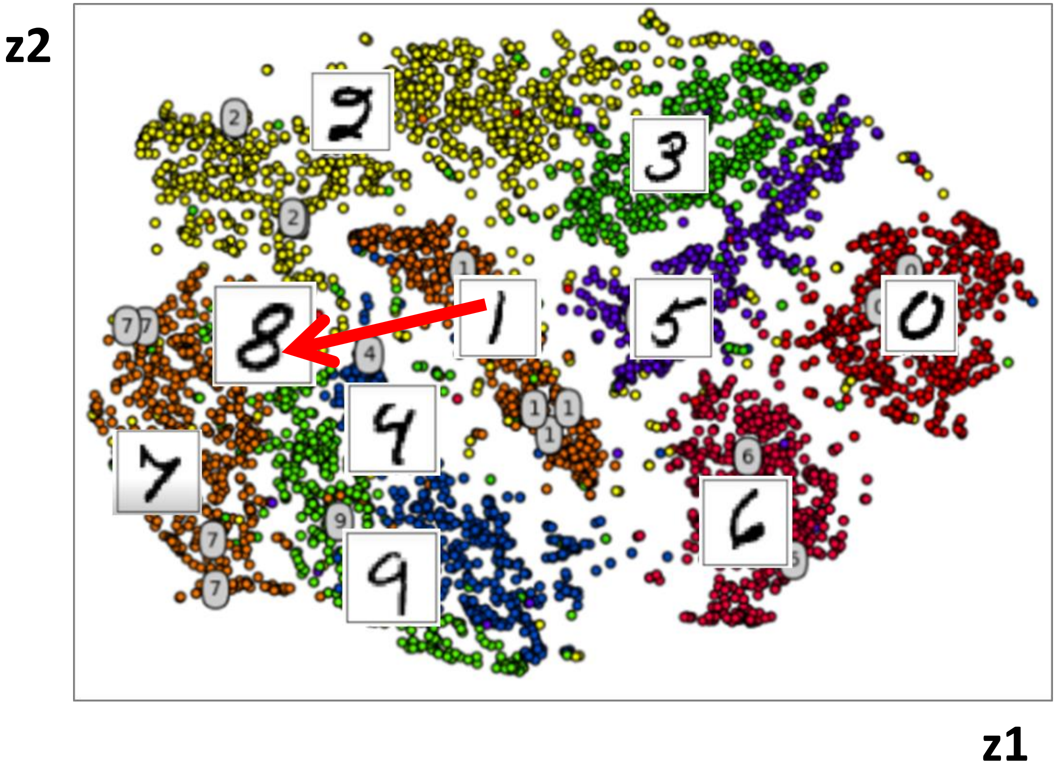
$$= \sum_{j=1}^{D}\left(\frac{1}{2}\log((\sigma_{x_j}^{(i)})^2) + \frac{\left(x_j^{(i)} - \mu_{x_j}\right)^2}{2\sigma_{x_j}^2}\right)$$

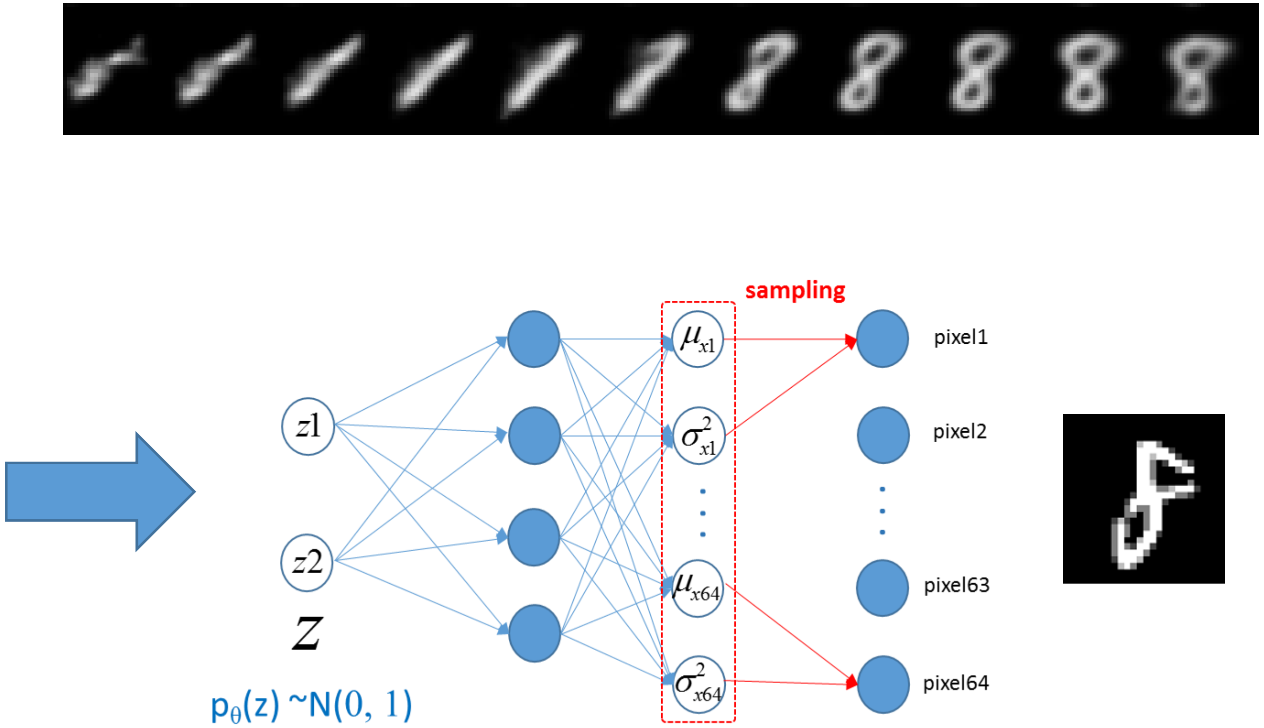
$$\min |x - \hat{x}|$$

Auto-Encoding Variational Bayes (appendix B: derivation) https://arxiv.org/pdf/1312.6114.pdf

Variational Auto Encoder (VAE): summary

- ❑ A generative model based on a neural network (AutoEncoder)
- ❑ Its loss function is derived based on variational inference approach (Variational)
- ❑ The loss function calculates the error used to train Auto Encoder through backpropagation
  - That is the reason why it is called “Variational AutoEncoder” (VAE).



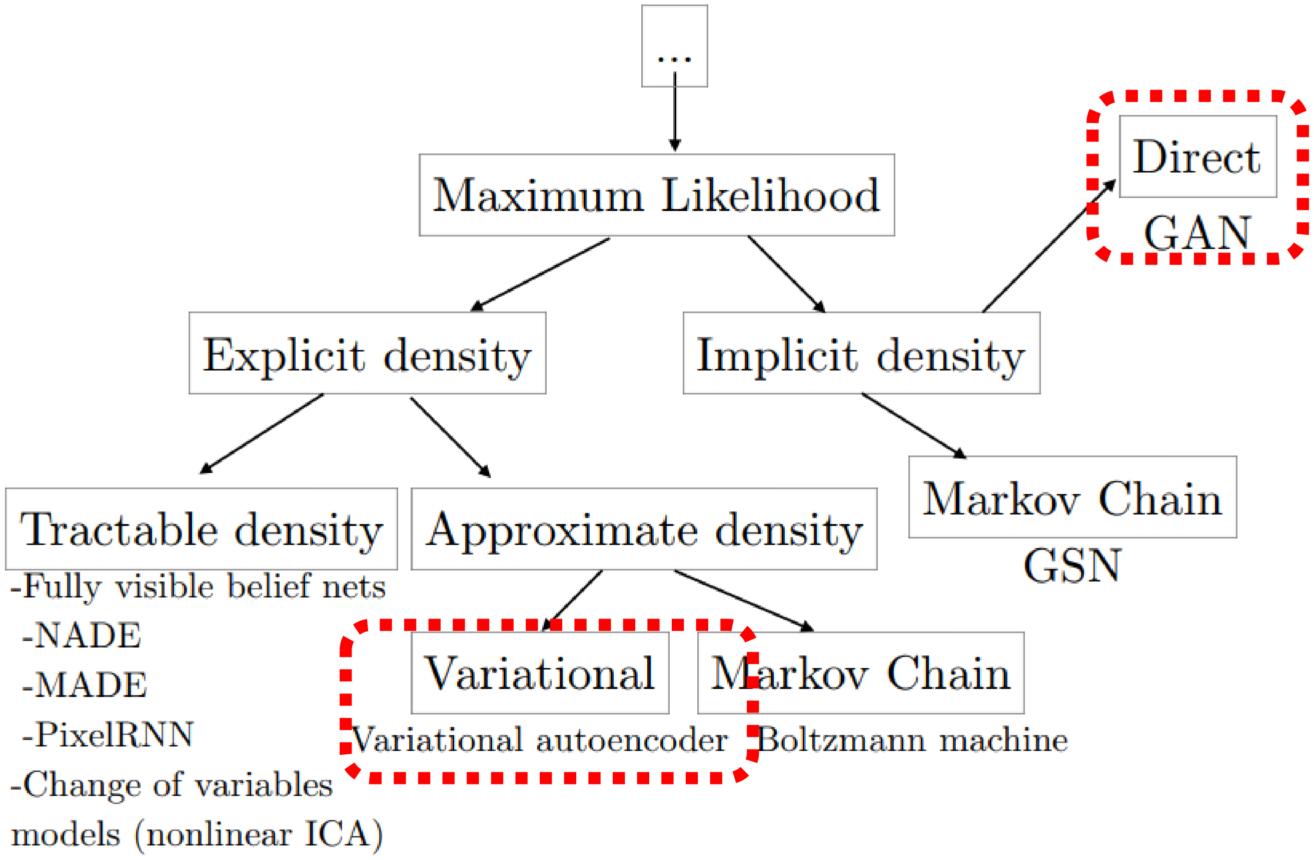


Generative Adversarial Networks (GAN)

https://arxiv.org/pdf/1701.00160.pdf

What is the motivation of GAN instead of VAE?

- In VAE, we design a latent space which maps to a data space.
- Then, a latent variable in the space is used to generate a data sample.
- However, actually we are interested in not the latent space but a sample itself.
- Then, why do we generate samples directly without the latent space estimation?

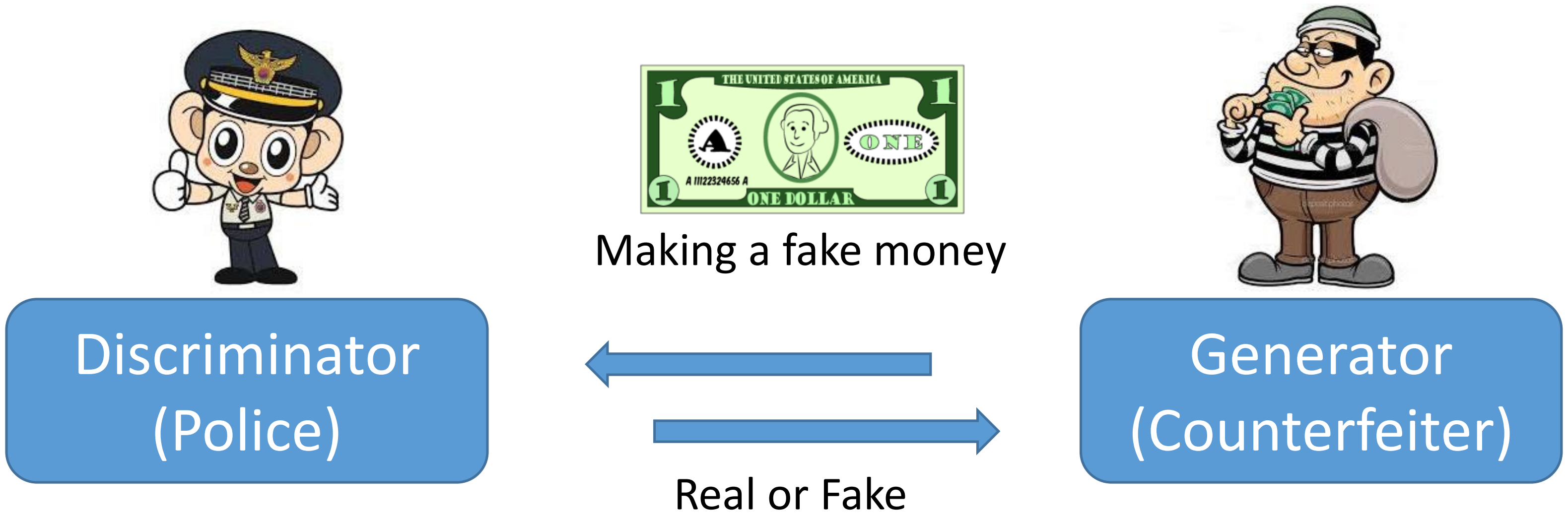


https://arxiv.org/pdf/1701.00160.pdf

How does GAN work?

- GAN: Generative Adversarial Network
- Based on game theory to train the system which directly generates a sample
- Adversarial:

  'GAN framework can naturally be analyzed with the tools of game theory, we call GANs "adversarial".' - Ian Goodfellow



https://arxiv.org/pdf/1701.00160.pdf

Theory: formulation of an optimization problem

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

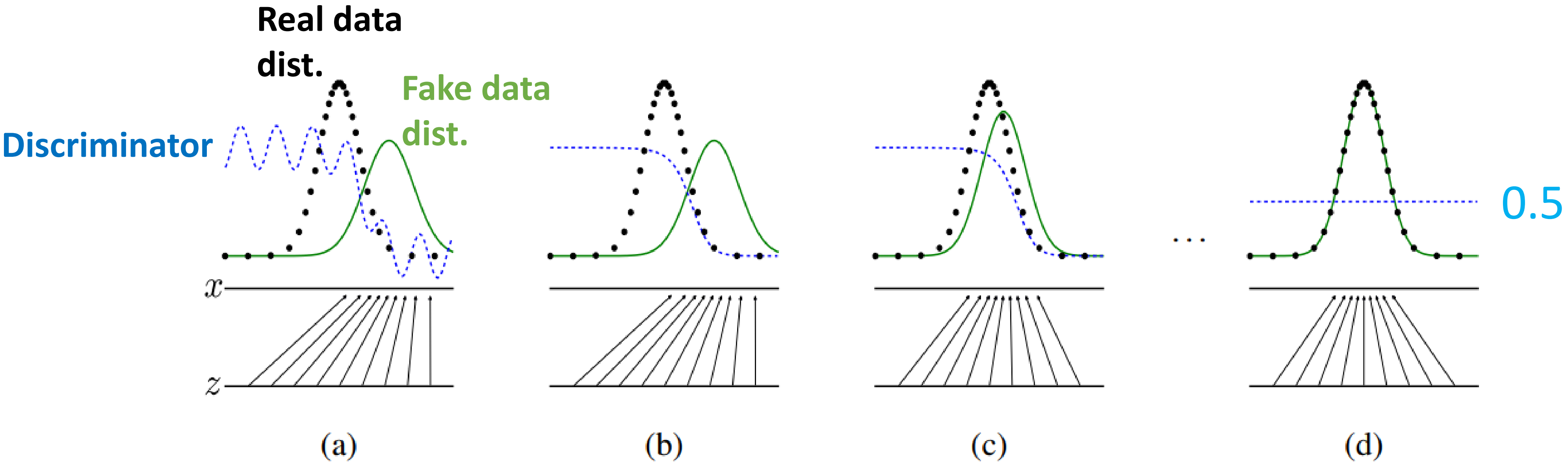
- Expectation that discriminator (D) tells **real is real** (D successes)
- Training discriminator to maximize it

- Expectation that D tells **fake is real** (D fails)
- Training generator to minimize a fake

notation	description
$x \sim p_{data}(x)$	Real data sample
$z \sim p_z(z)$	A random number from N(0, 1)
G(z)	Fake data sample
D(x)=1	Probability of discriminator (D) telling that given real data “x” is real
D(G(z))=0	Probability of discriminator (D) telling that given fake data “G(z)” is fake
1 – D(G(z))	Probability of discriminator (D) telling that given fake data “G(z)” is real

Theory: illustration

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



- ❑ Generator keeps trained to generate a fake one similar to real and so finally Discriminator cannot tell a fake from a real => its probability becomes 0.5.

https://arxiv.org/abs/1406.2661

Theory: its global optimal solution $p_g = p_{data}$

- For the fixed generator (G), the optimal discriminator value (D*) is

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

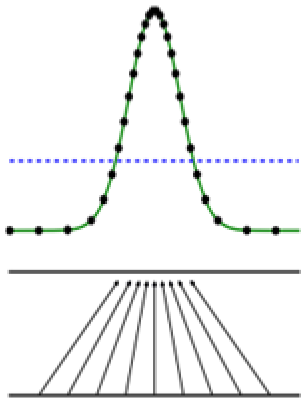
$$\max_D V(D) = \int_x p_{data}(x)\log(D(x))dx + \int_z p_z(z)\log(1-D(G(z)))dz$$

$$= \int_x p_{data}(x)\log(D(x)) + p_g(x)\log(1-D(x))dx$$

$$\frac{dV(D)}{dD} = \frac{p_{data}(x)}{D(x)} - \frac{p_g(x)}{1-D(x)} = 0$$

$$D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x)+p_g(x)}$$

$$= 0.5\ (p_g = p_{data})$$



Theory: its global optimal solution $p_g=p_{data}$

- With the optimum value of D*, lower bound of V(G) is

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

$$\min_G V(G) = E_{x\sim p_{data}(x)}[\log D^*(x)] + E_{z\sim p_g(x)}[\log(1-D^*(x)))]$$

Backup slide

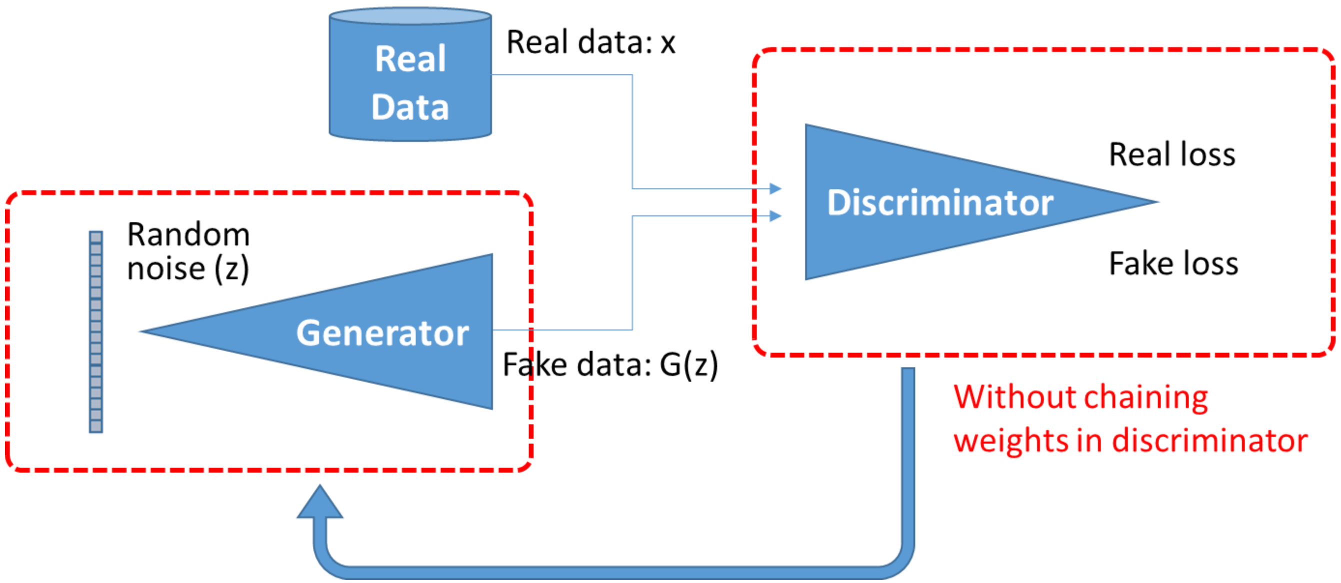
$$= -\log(4) + 2\cdot JSD(p_{data} \| p_g)$$

This is the minimum value of V(G) when JSD=0 ($p_g=p_{data}$)

- JSD: Jensen Shannon divergence
  - A method of measuring the similarity between two probability distribution.
  - $0 \le JSD(p|q) \le 1$

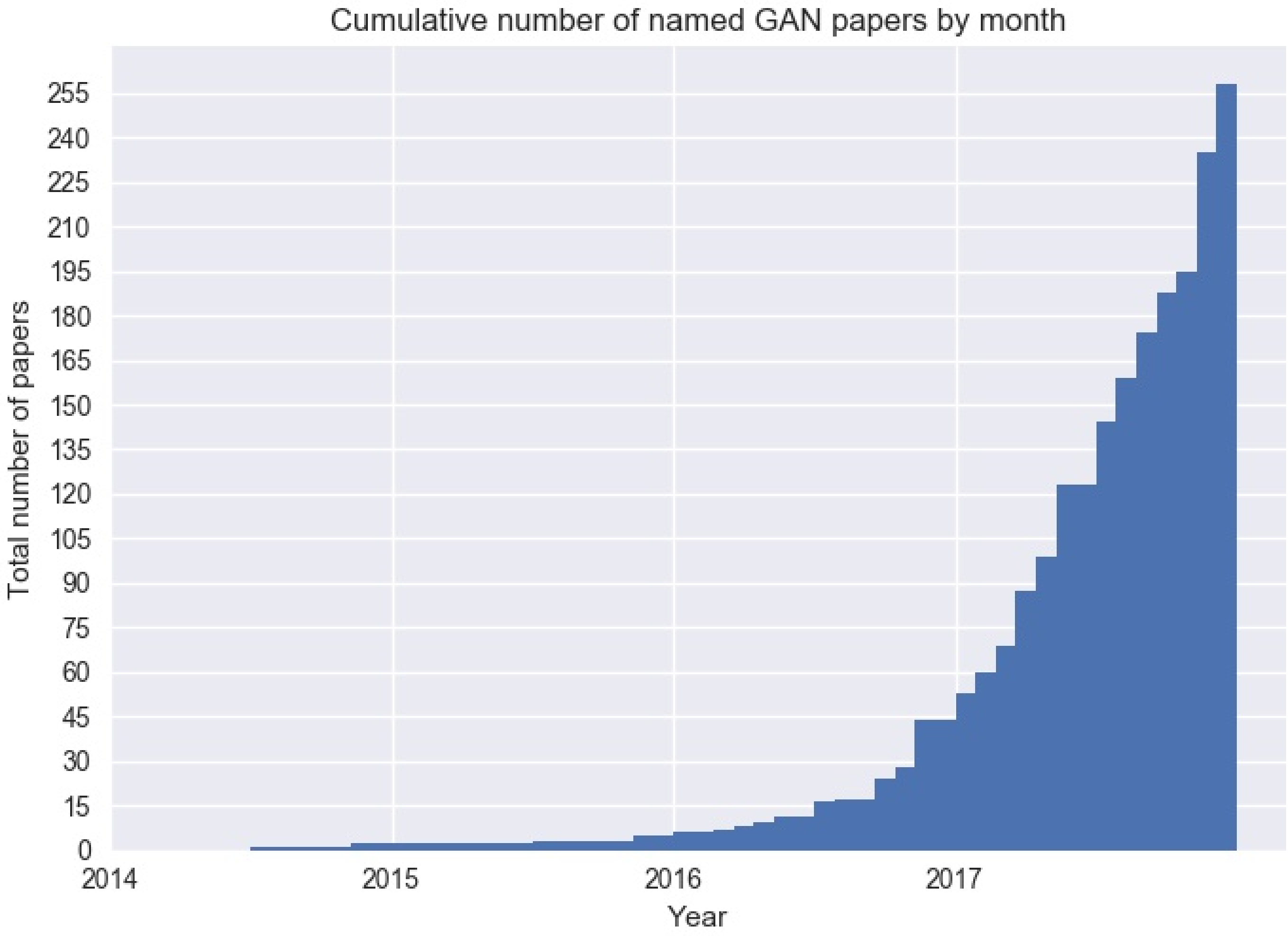
Generative Adversarial Networks (GAN): summary

- ❑ Given the system below, we train it based on the objective function.
- ❑ The objective function is derived based on game theory.
  - Generator tries to make a real like fake data to deceive the discriminator
  - Discriminator tries not to be deceived by the generator
- ❑ In this manner, generator learns how to make a sample close to real data.
- ❑ It is about how to define the objective function and whether it converges to an optimum solution.



Applications

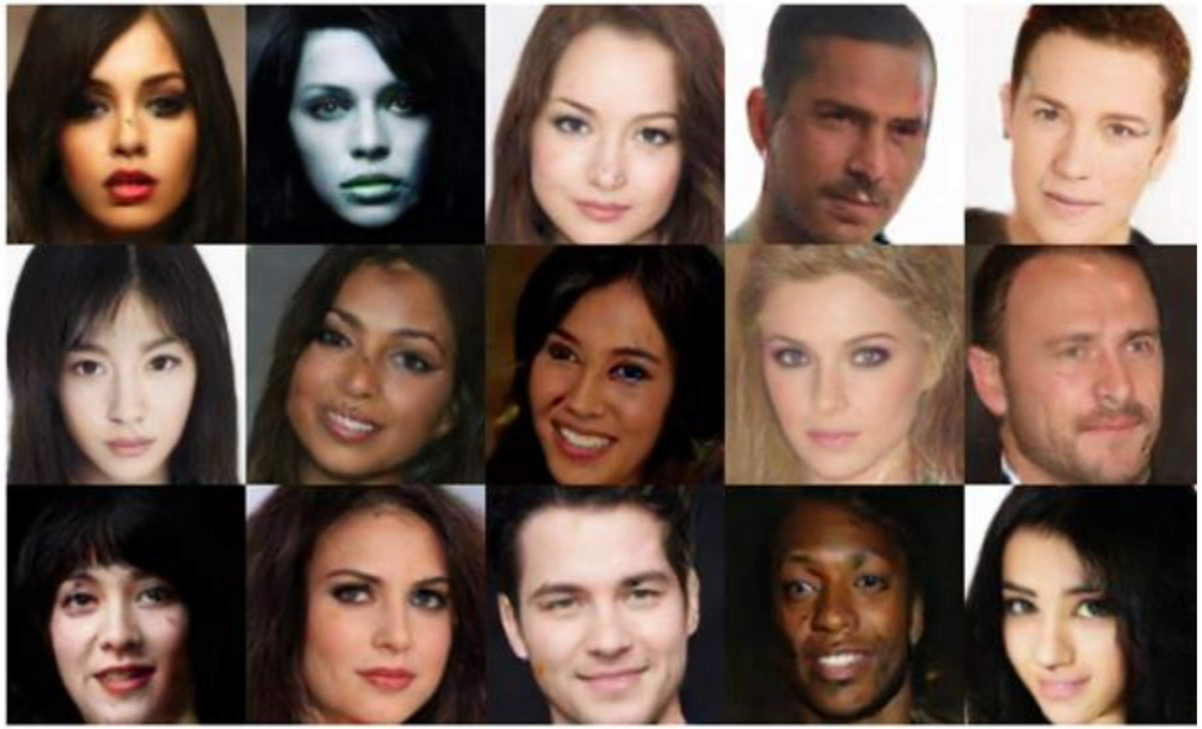
Explosive growth of the popularity of GAN



https://deephunt.in/the-gan-zoo-79597dc8c347

High resolution image generation

- [ ] https://arxiv.org/pdf/1703.10717.pdf (BGAN)



Text to image

- https://arxiv.org/pdf/1710.10916.pdf (StackGAN)

Style transfer

- https://github.com/junyanz/CycleGAN

winter Yosemite → summer Yosemite

summer Yosemite → winter Yosemite

Input Output

horse → zebra

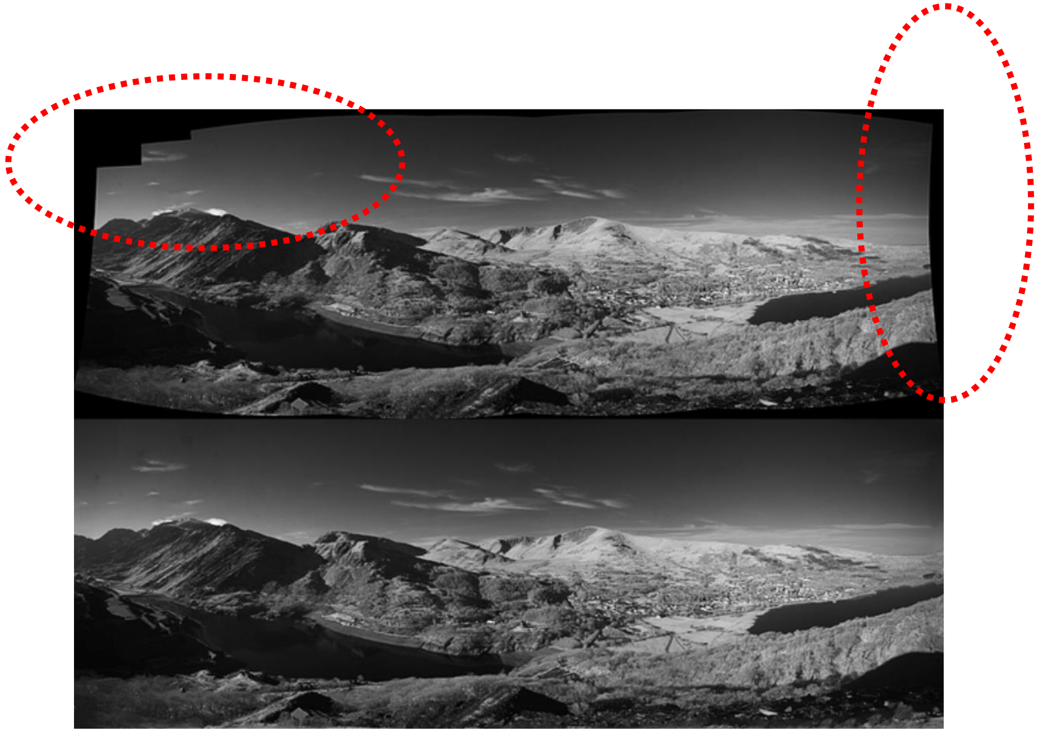
zebra → horse

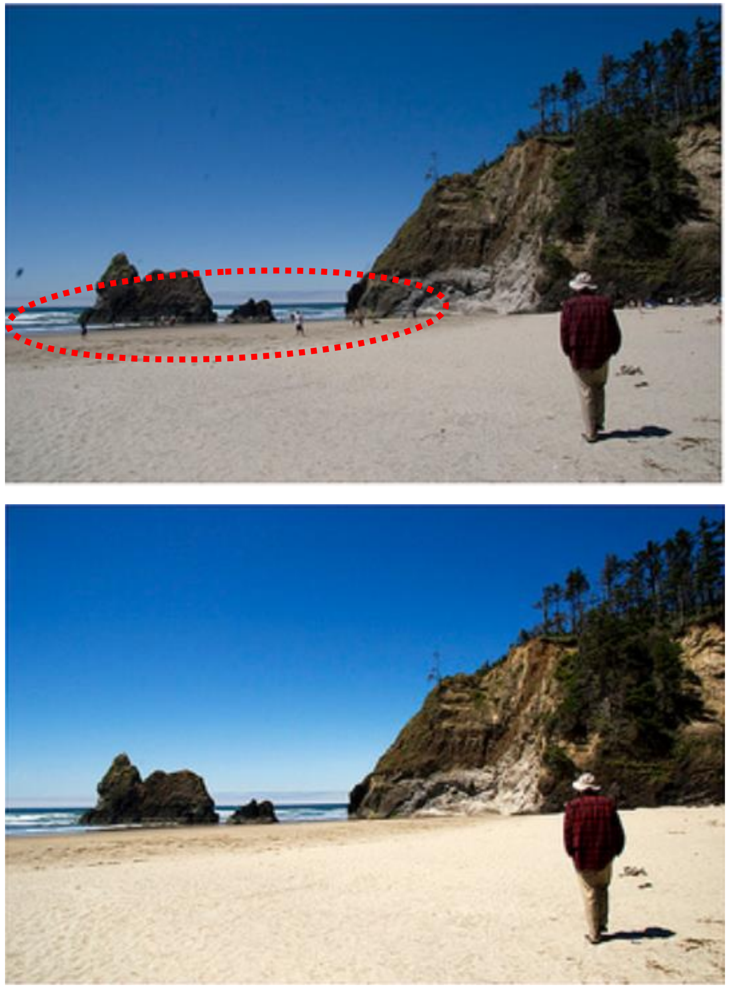
apple → orange

orange → apple

Image Completion with Deep Learning in TensorFlow

- http://bamos.github.io/2016/08/09/deep-completion/





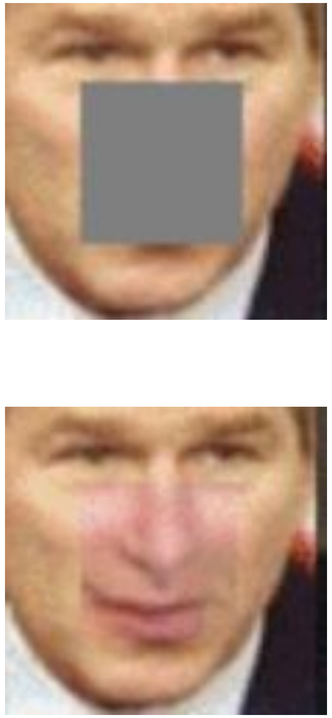
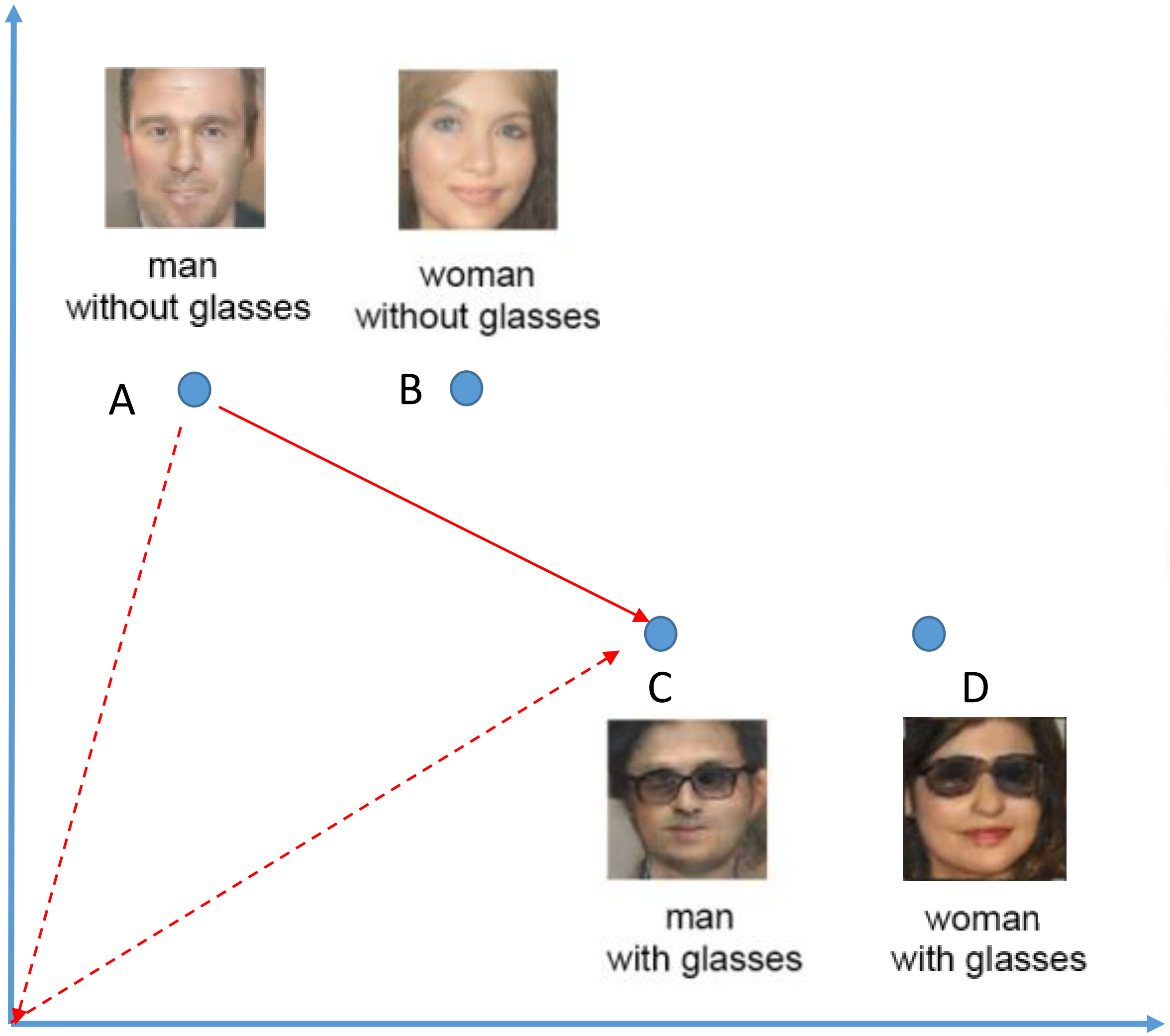


Image2Vec

- https://arxiv.org/pdf/1511.06434.pdf



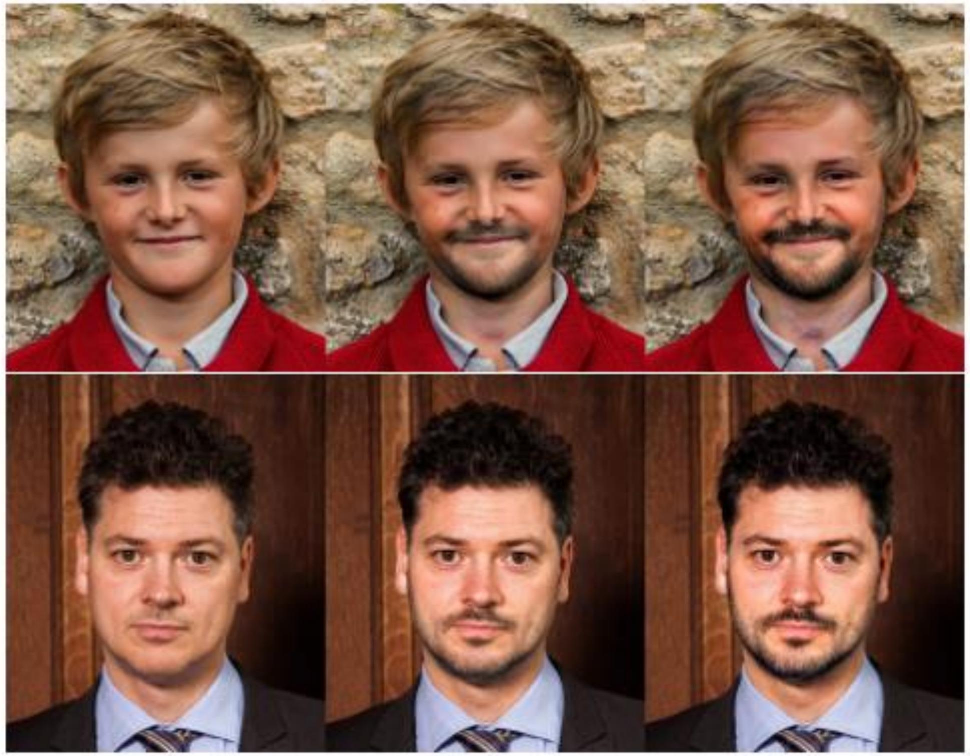
A->C = B->D

C-A = D-B

C-A+B=D

Deep Feature Interpolation

- https://arxiv.org/pdf/1611.05507.pdf



Backup Slides

V(G) when D is fixed

$$\min_G V(G) = E_{x\sim p_{data}(x)}\left[\log \frac{p_{data}(x)}{p_{data}(x)+p_g(x)}\right] + E_{x\sim p_g(x)}\left[\log \frac{p_g(x)}{p_{data}(x)+p_g(x)}\right]$$

$$KL(P \| Q) = \sum P \log \frac{P}{Q}$$

$$JSD(P \| Q) = \frac{1}{2} KL(P \| M) + \frac{1}{2} KL(Q \| M)$$

$$V(G) = V(G) + \log(4) - \log(4)$$

$$= -\log(4) + E_{x\sim p_{data}(x)}[\log D(x)] + E_{x\sim p_q(x)}[\log(1-D(x))] + \log(4)$$

$$= -\log(4) + \sum p_{data}(x)\log(D^*(x)) + \log(2) + \sum p_g(x)\log(1-D^*(x)) + \log(2)$$

$$= -\log(4) + \sum p_{data}(x)\log\frac{p_{data}(x)}{p_{data}(x)+p_g(x)} + \log(2) + \sum p_g(x)\log\frac{p_g(x)}{p_{data}(x)+p_g(x)} + \log(2)$$

$$= -\log(4) + KL\left(p_{data} \| \frac{p_{data}+p_g}{2}\right) + KL\left(p_g \| \frac{p_{data}+p_g}{2}\right)$$

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