

CS492D: Diffusion Models and Their Applications

Introduction to Generative Models: GAN and VAE

LECTURE 2
MINHYUK SUNG

Fall 2024
KAIST

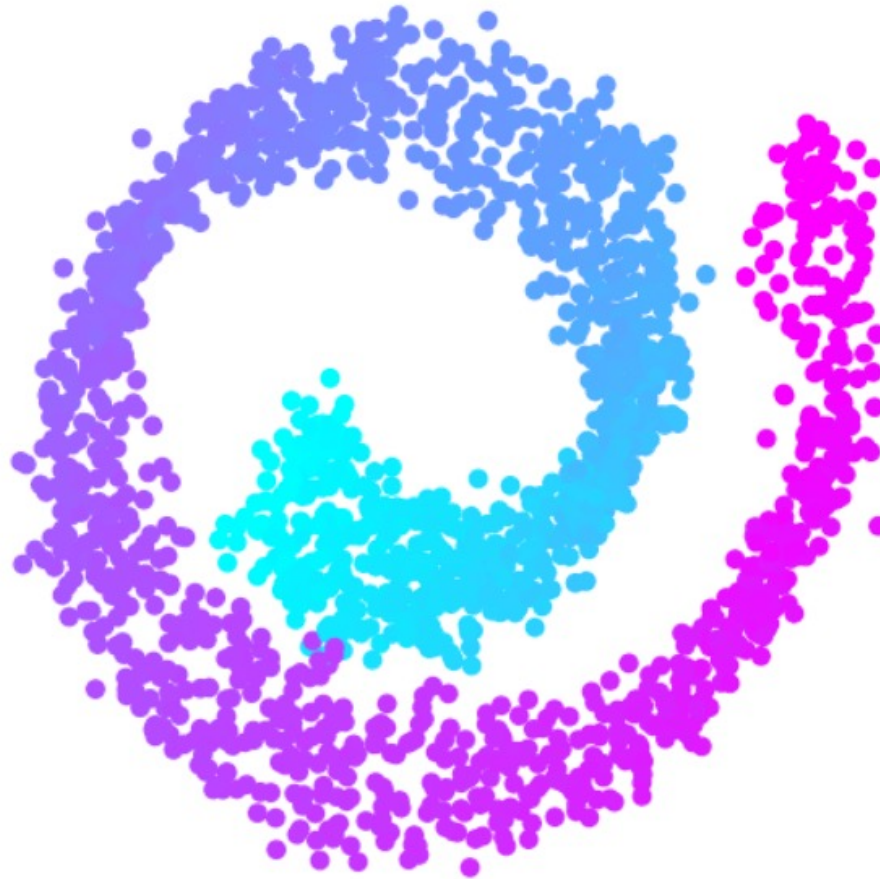
Let's consider a collection of real photos.



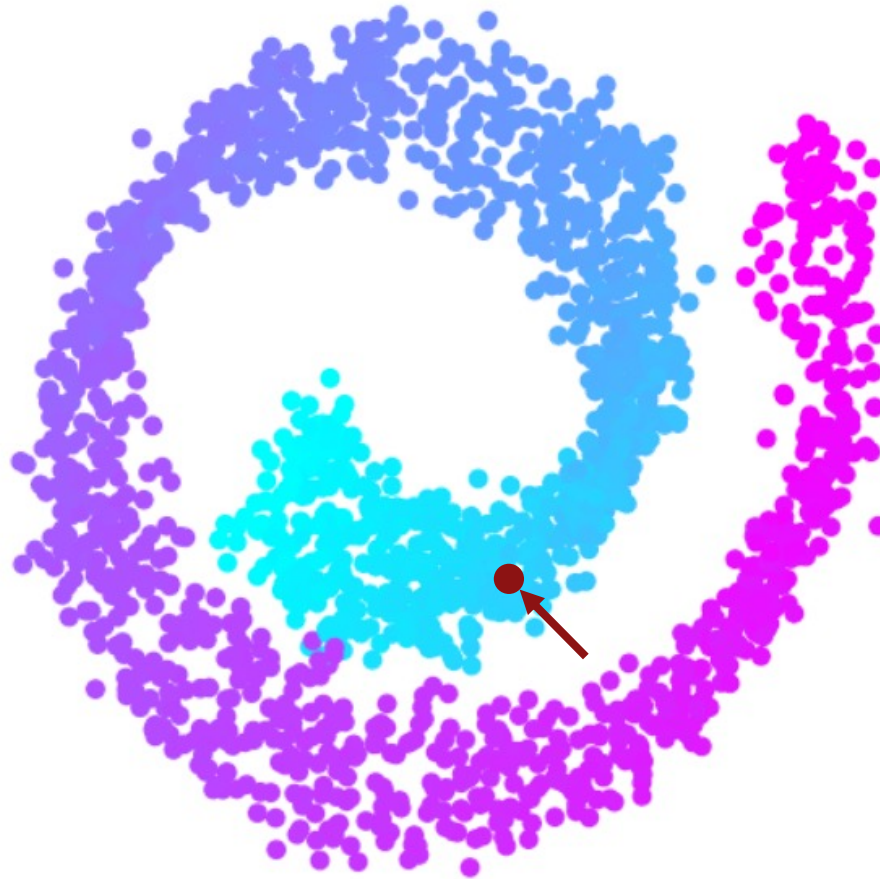
How can we generate a **new** realistic photo?



Let's consider a **simpler** case:
a collection of **2D** points.



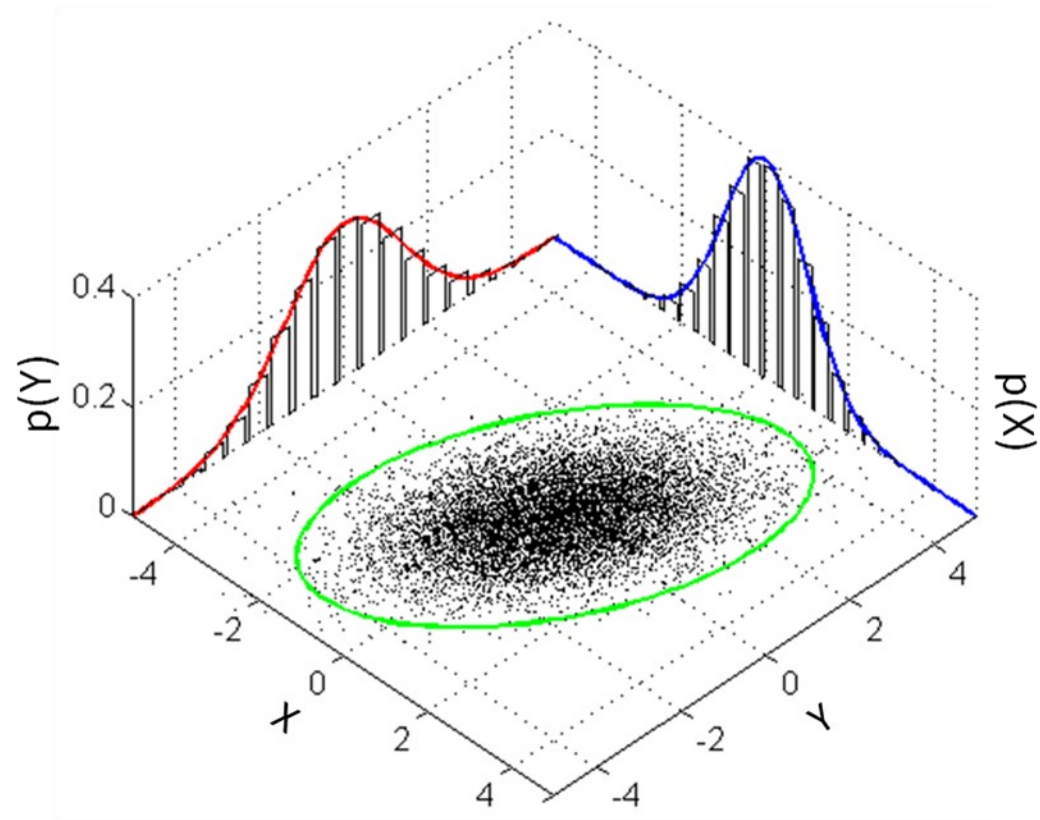
How can we **sample** a new plausible 2D point?



Statistical Perspective

From a statistical perspective, we will view this as

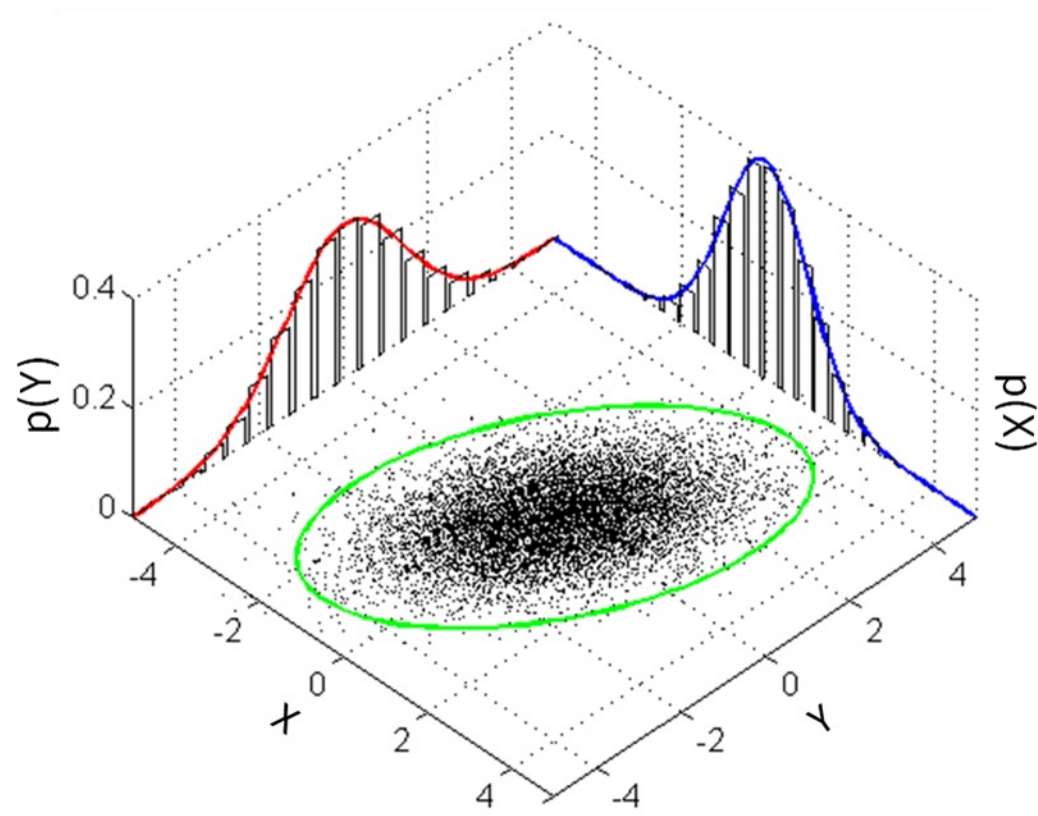
- there being a **probability distribution** of the data, and
- the given points are **samples** from the probability distribution.



Statistical Perspective

If the **probability density function (PDF)** of the **distribution** is known, we can sample from it directly.

How?



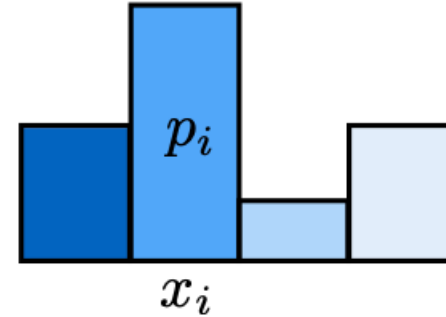
Discrete Probability Distributions

- n discrete values x_i with probability p_i .
- Requirements of a PDF:

$$p_i \geq 0$$

$$\sum_{i=1}^n p_i = 1$$

Q. How can we sample based on this PDF?



Cumulative Distribution Function (CDF)

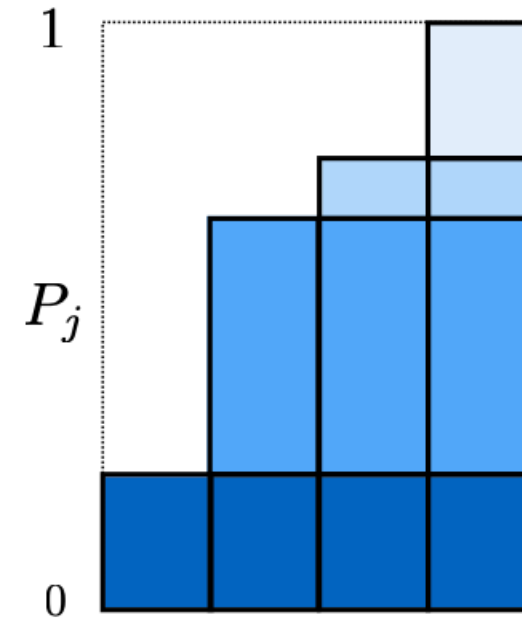
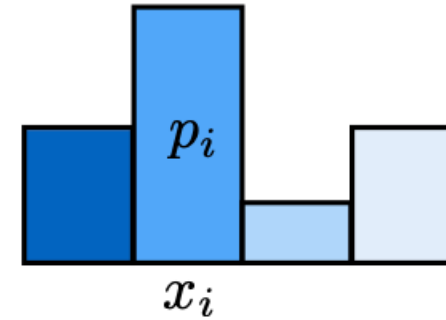
Cumulative distribution function:

$$P_i = \sum_{j=1}^i p_j$$

where

$$0 \leq P_i \leq 1$$

$$P_n = 1$$

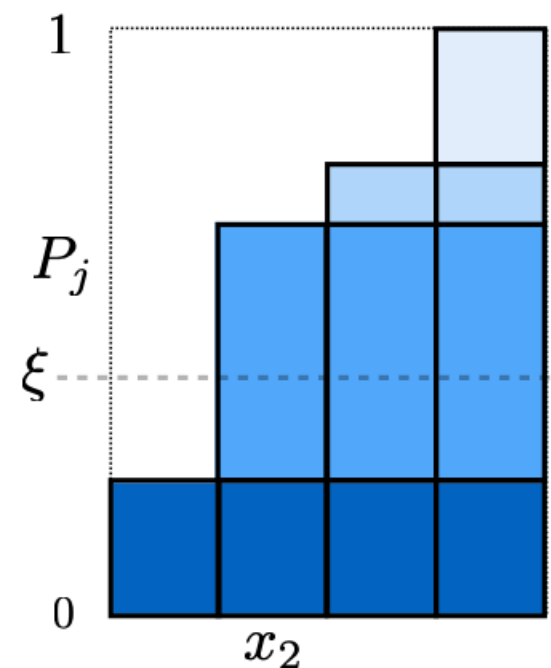
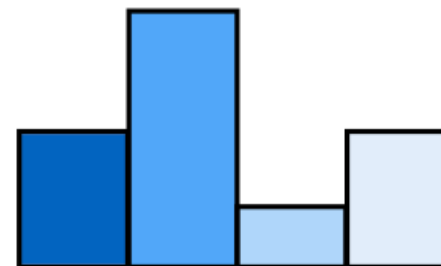


Inverse Transform Sampling

To randomly select an event,

- Draw $u \sim \mathcal{U}(0, 1)$.
- Take x_i if

$$P_{i-1} \leq u \leq P_i$$



Continuous Probability Distributions

Given a PDF $p(x)$,

- CDF $F_X(x)$

$$= \Pr(X \leq x) = \int_0^x p(t)dt$$

- $\Pr(a < X \leq b)$

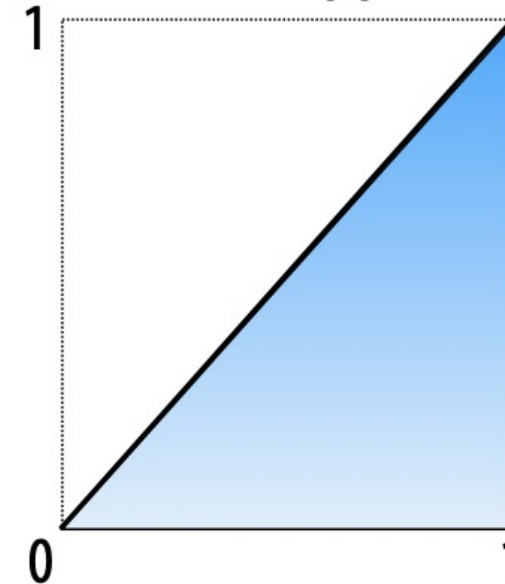
$$= \int_a^b p(t)dt = F_X(b) - F_X(a)$$

- $F_X(1) = 1$

Uniform distribution: $p(x) = c$
(for random variable X defined on $[0,1]$ domain)



CDF $P(x)$

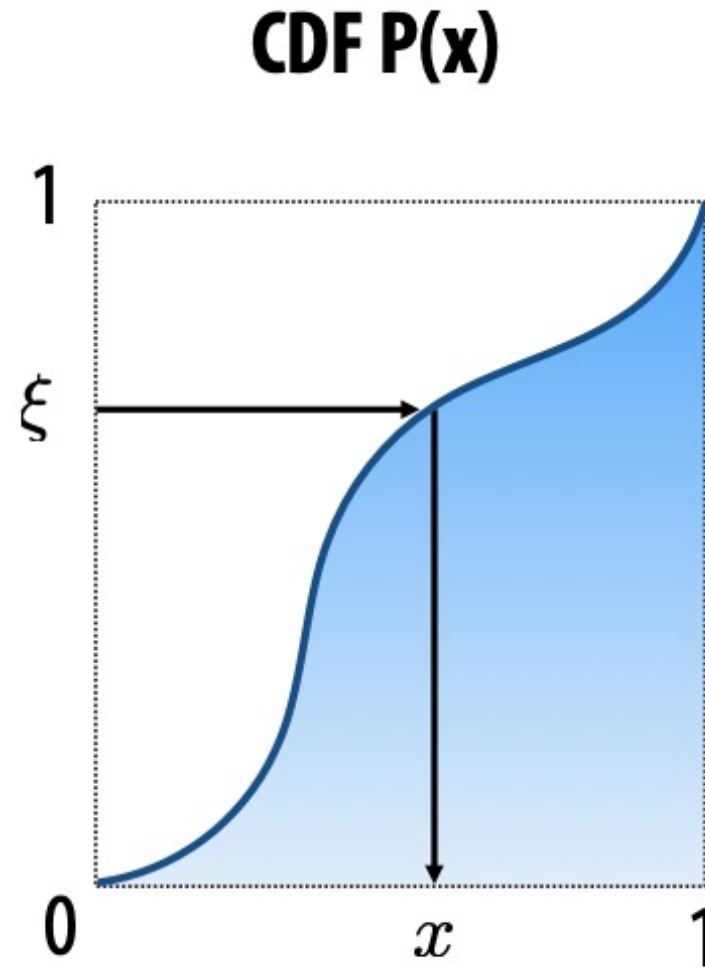


Inverse Transform Sampling

To randomly sample based on the given PDF,

- Compute **CDF** $F_X(x)$
- Draw $u \sim \mathcal{U}(0, 1)$.
- Take $x = F_X^{-1}(u)$.

*Need to know
the inverse function $F_X^{-1}(x)$.*

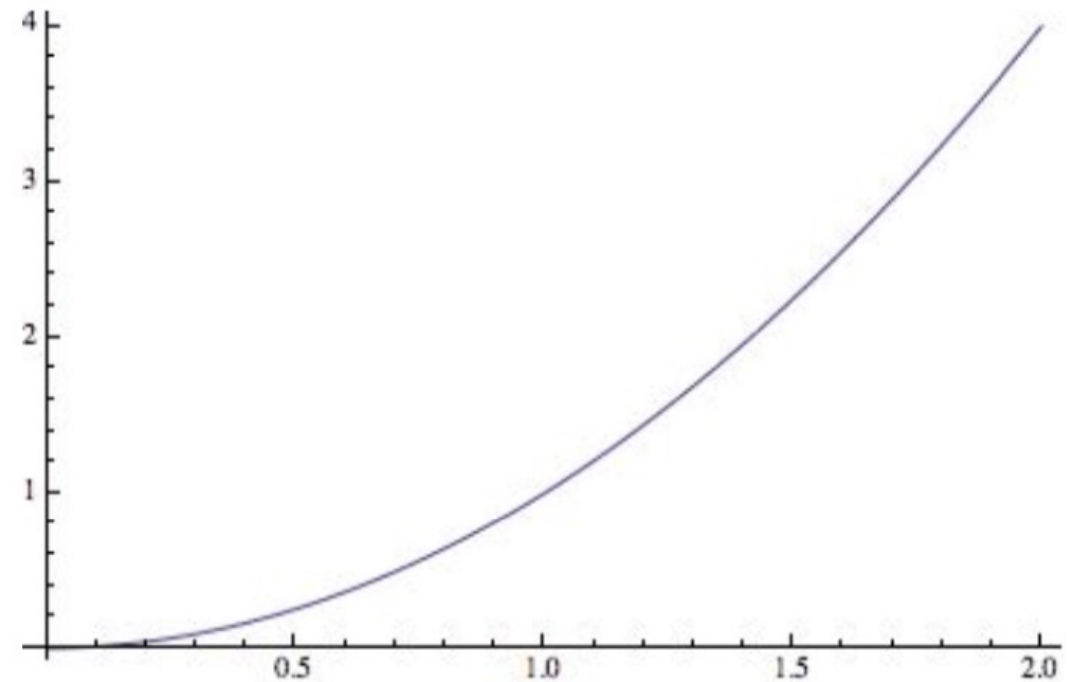


Inverse Transform Sampling – Example

PDF

$$p(x) = \frac{3}{8}x^2 \quad x \in [0, 2]$$

Q. What is the inverse of CDF?



Inverse Transform Sampling – Example

PDF

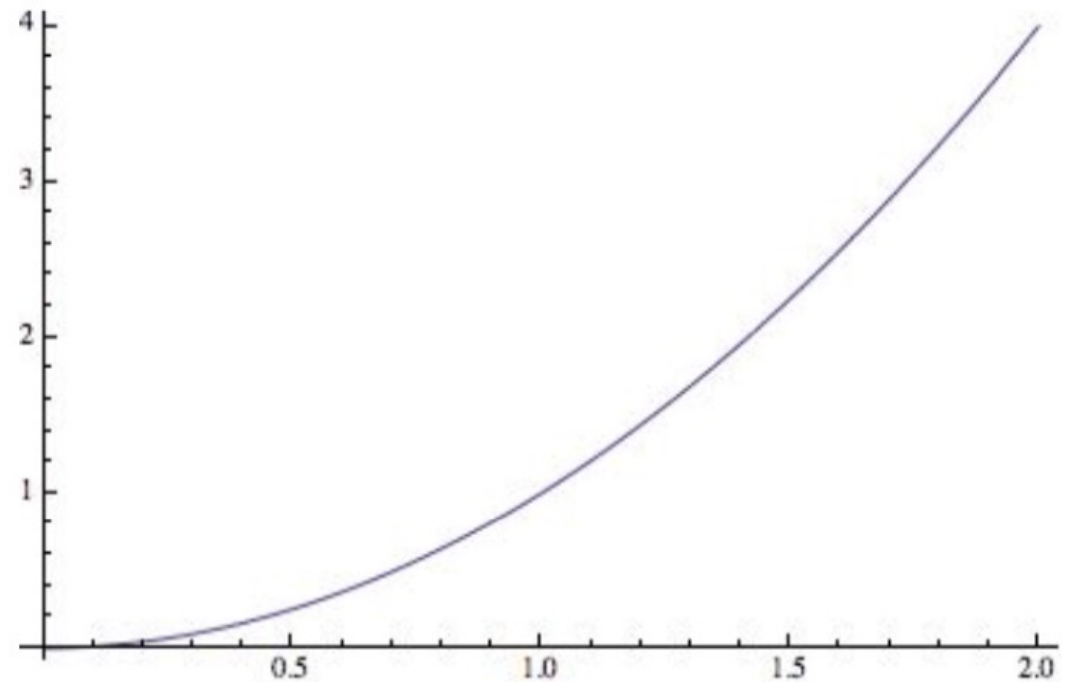
$$p(x) = \frac{3}{8}x^2 \quad x \in [0, 2]$$

CDF

$$F_X(x) = \int_0^x p(x)dx = \frac{1}{8}x^3$$

Inverse of CDF

$$F_X^{-1}(x) = 2\sqrt[3]{x}$$



Inverse Transform Sampling – Example

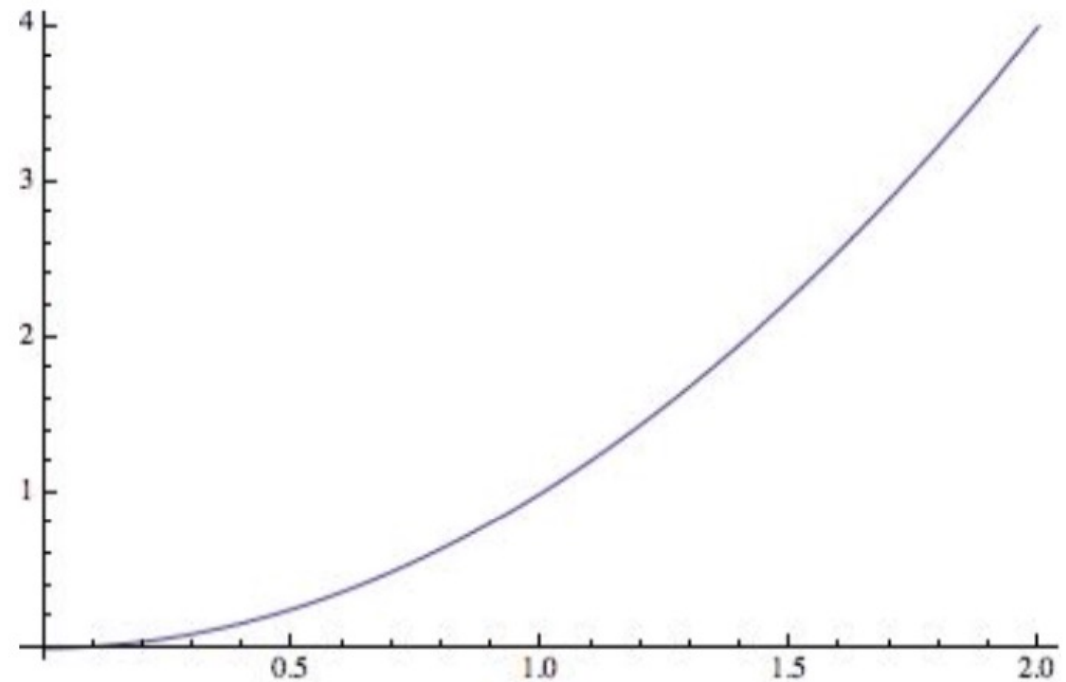
Sampling from

$$p(x) = \frac{3}{8}x^2$$

1. Draw a sample

$$u \sim \mathcal{U}(0, 1).$$

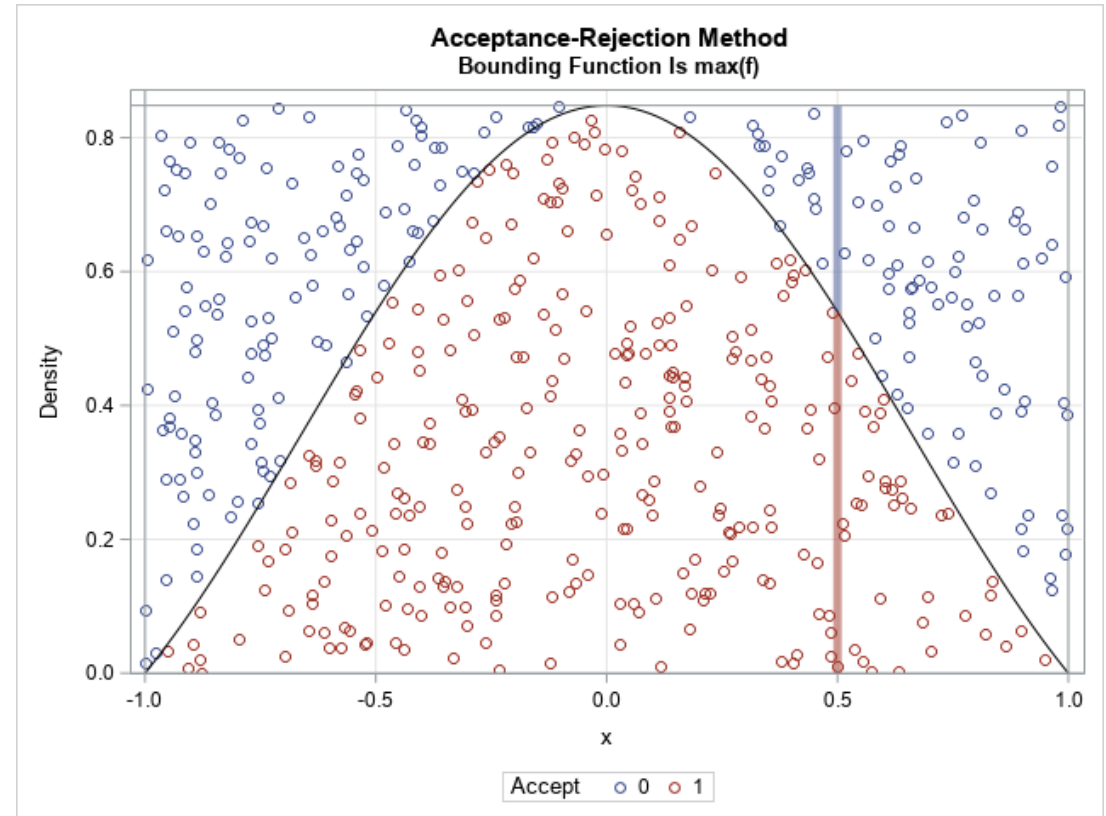
2. Take $2\sqrt[3]{u}$.



Rejection Sampling

If the inverse of the CDF cannot be computed:

1. Let $q(x)$ be an upper bound distribution: $\forall x \ q(x) \geq p(x)$.
2. Draw $x \sim q(x)$.
3. Draw a $h \sim \mathcal{U}(0, q(x))$.
4. Accept the sample x if $h \leq p(x)$; otherwise reject it.



Reparameterization Trick

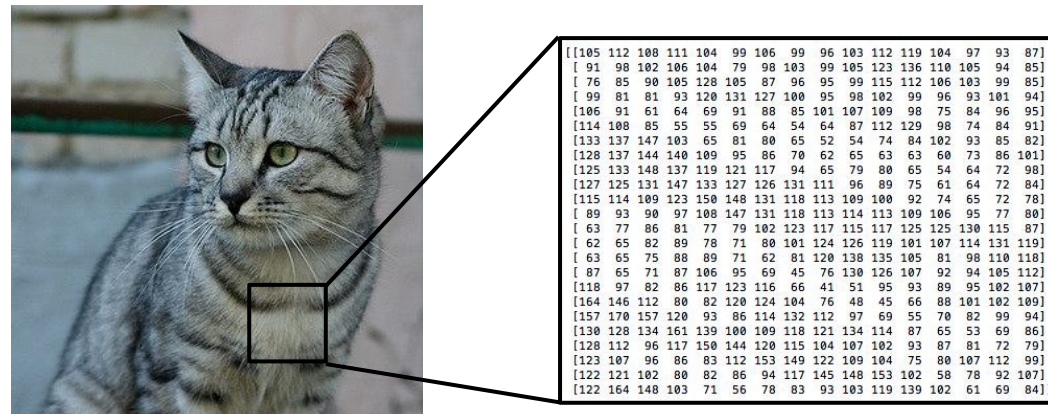
- A sample from a normal distribution $z \sim N(\mu, \Sigma)$ can be rewritten as follows:

$$z = \mu + \Sigma^{\frac{1}{2}} \epsilon \text{ where } \epsilon \sim N(0, I).$$

- We just need a standard normal sampler to sample from an arbitrary normal distribution.

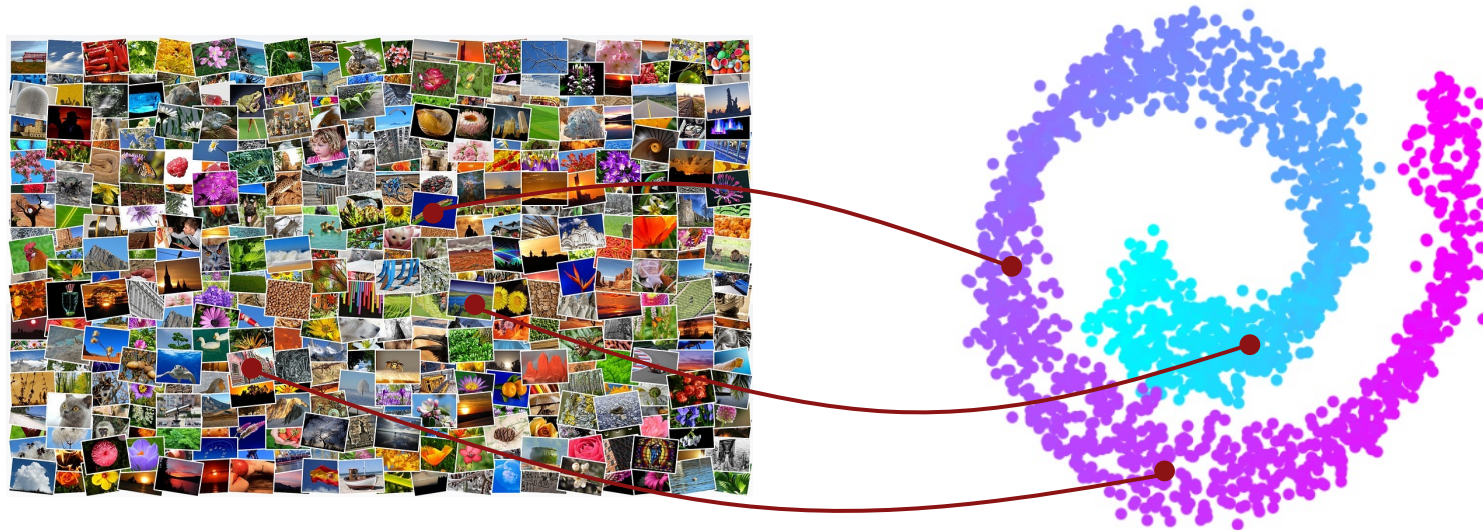
Statistical Perspective for Real Images

- Let's consider RGB images with a resolution of 256×256 .
- An image can be represented by a $256 \times 256 \times 3$ vector.
- This means that **an image is a point** in a $256 \times 256 \times 3$ -dimensional space.



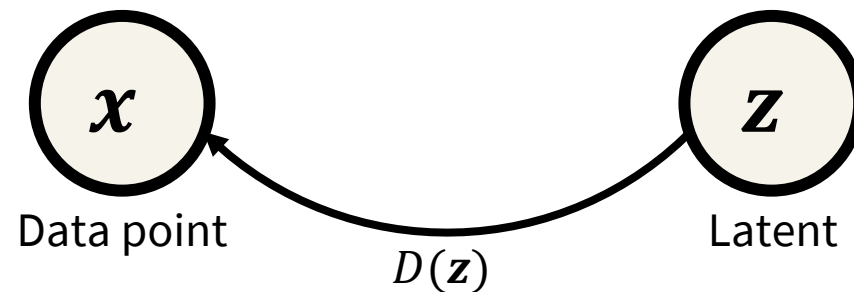
Statistical Perspective for Real Images

- Let us consider real images $\{x_1, x_2, \dots, x_n\}$ as **samples from a data distribution $p(x)$** that measures how likely it is for an image to be a real photo.
- Can we derive the PDF of the data distribution...?



The Basic Idea

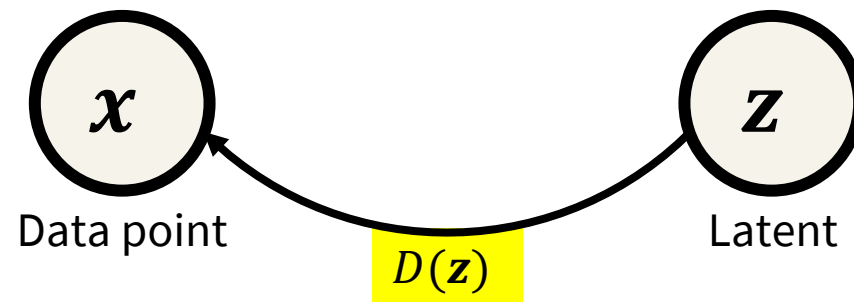
- **Map** a *simple* distribution $p(\mathbf{z})$ (e.g., a standard normal distribution $\mathcal{N}(\mathbf{x}; \mathbf{0}, \mathbf{I})$) to the **data distribution** $p(\mathbf{x})$.
 - \mathbf{z} : **Latent** variable
 - $p(\mathbf{z})$: **Latent** distribution
- Sample from $p(\mathbf{z})$ and map it to a data point.



The Basic Idea

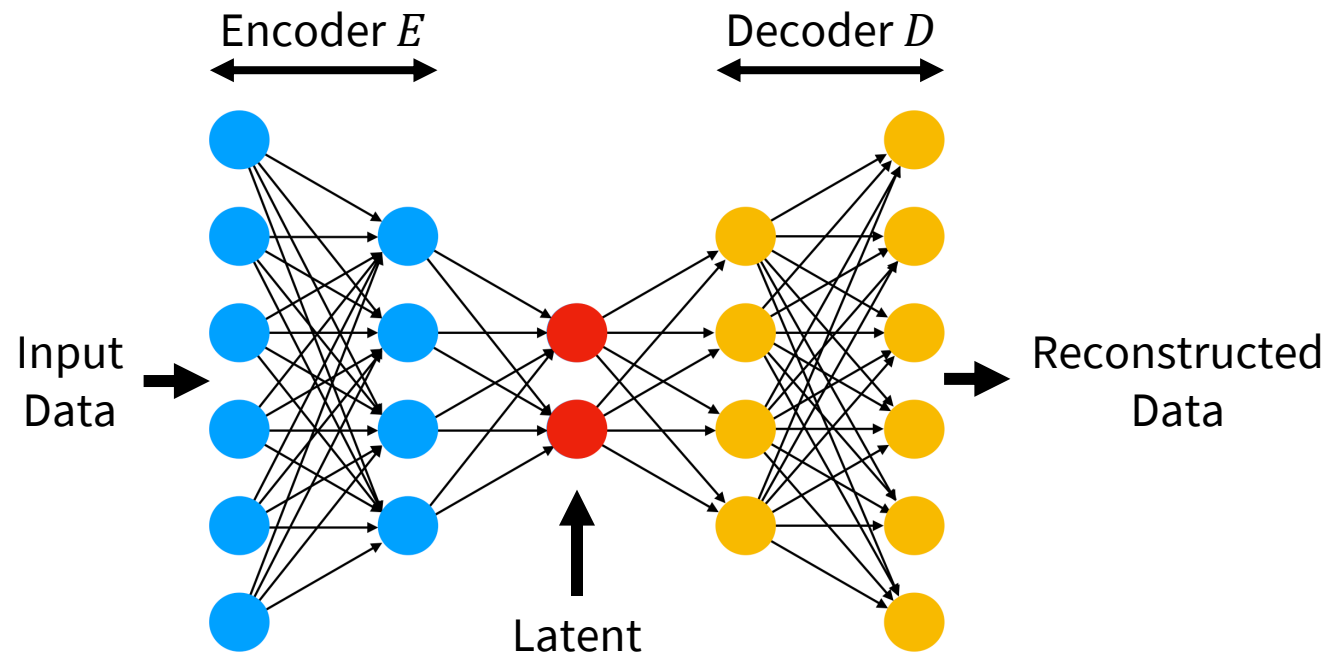
How to map a latent distribution $p(\mathbf{z})$ to the data distribution $p(\mathbf{x})$?

Let's use a **neural network**!



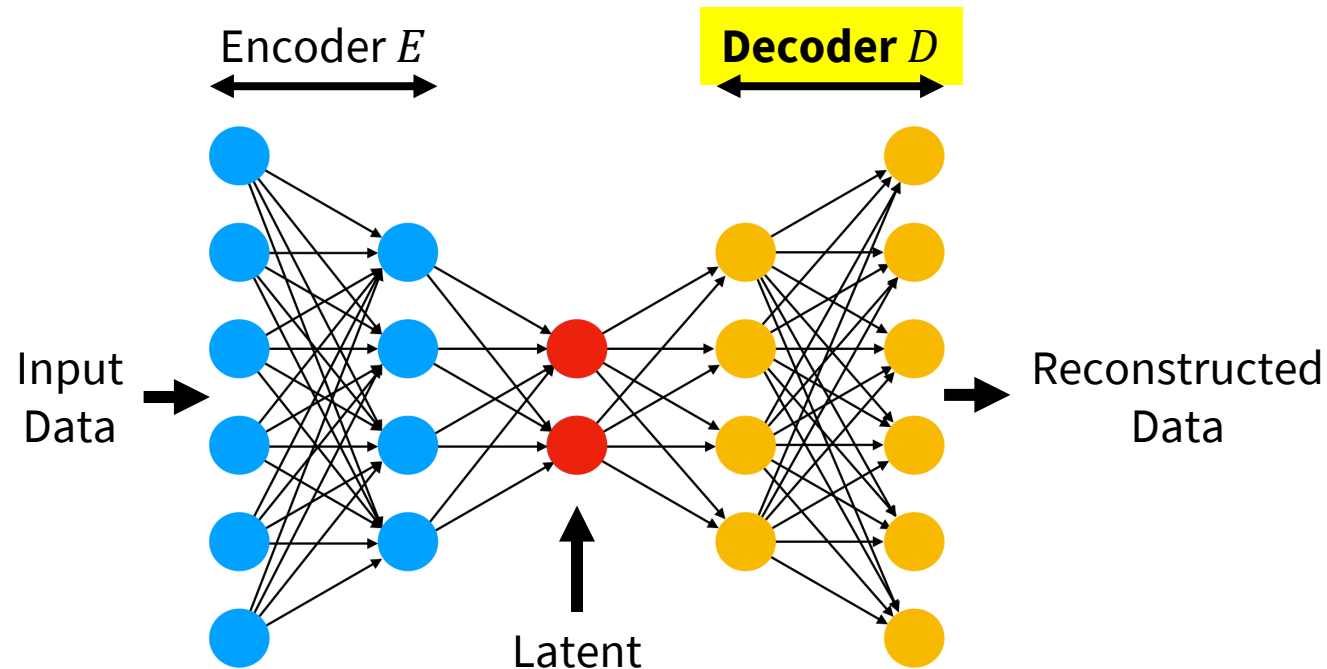
Autoencoder

An autoencoder is a neural network designed to **reconstruct** the input data while encoding it into a **lower-dimensional latent vector**.



Autoencoder

- What we need is the **decoder** (latent \rightarrow input data).
- But how do we **guarantee** that a latent is mapped to a data point of the data distribution?



Two Methods

1. Generative Adversarial Network (GAN)
2. Variational Autoencoder (VAE)

Generative Adversarial Network (GAN)

Goodfellow et al., Generative Adversarial Networks, NeurIPS 2014.

Off-Topic: Adversarial Examples

Neural networks work well, but how **vulnerable** is a neural network?



Images that are *correctly* classified

Off-Topic: Adversarial Examples

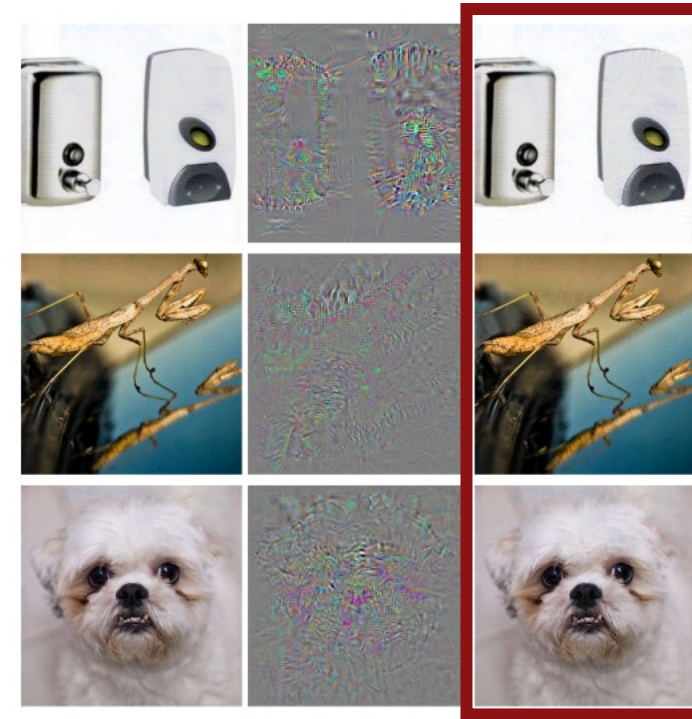
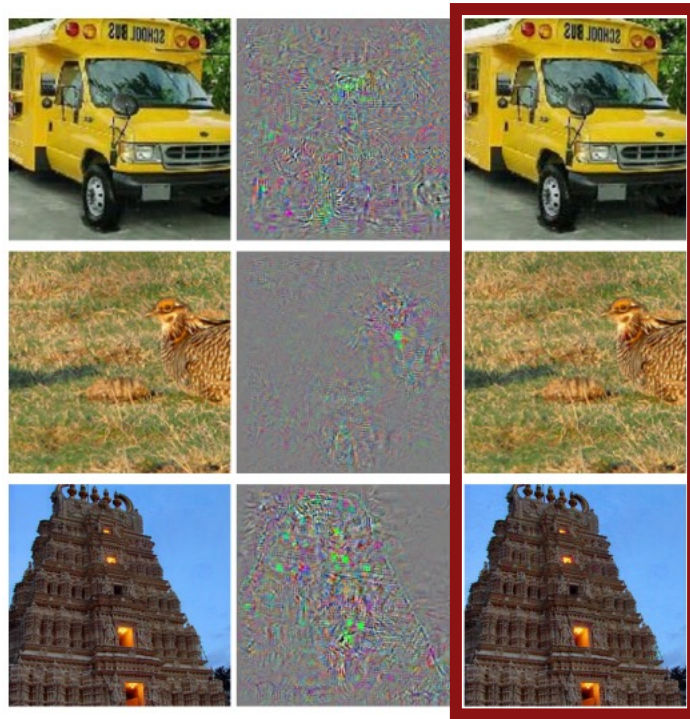
Neural networks work well, but how **vulnerable** is a neural network?



Difference between the two multiplied by $10\times$

Off-Topic: Adversarial Examples

Neural networks work well, but how **vulnerable** is a neural network?



Images that are *incorrectly* classified as ostrich

Wikipedia

Off-Topic: Adversarial Examples

- Let's train a network to predict an image that causes the classifier to fail. **Adversarial attack!**
- Can we also **finetune the classifier** to prevent it from failing due to adversarial attacks?
- What happens if we make them **compete** against each other?

Generative Adversarial Network (GAN)

Think about

- a real/fake image **classifier** → **Discriminator**
- an **adversarial attack network** → **Generator (Decoder)**
that tries to make a real-like generated image.

Generative Adversarial Network (GAN)

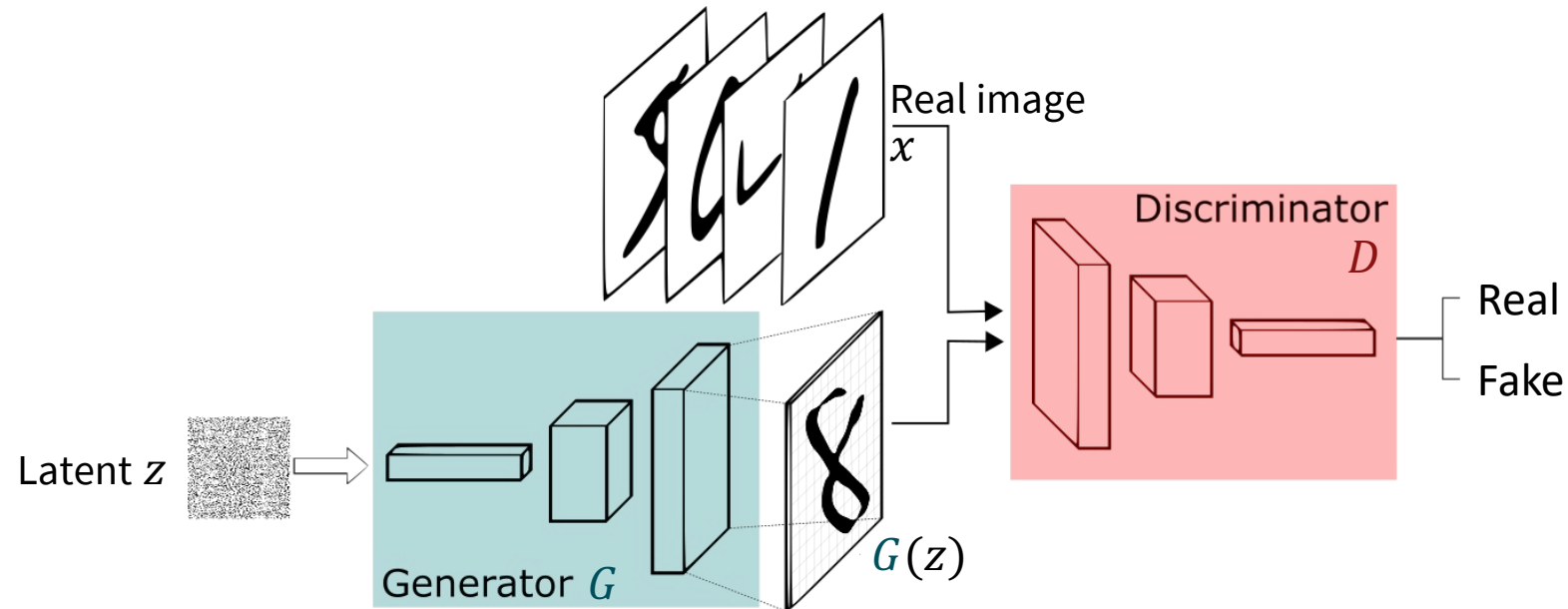
- **Generator** (or decoder) G takes a **latent** sampled from a unit Gaussian as input and generates a **synthetic** image.
- **Discriminator** D takes an image as input and **classifies** it as either **real or fake** (generated).

Make them compete against each other!

Generative Adversarial Network (GAN)

Loss function:

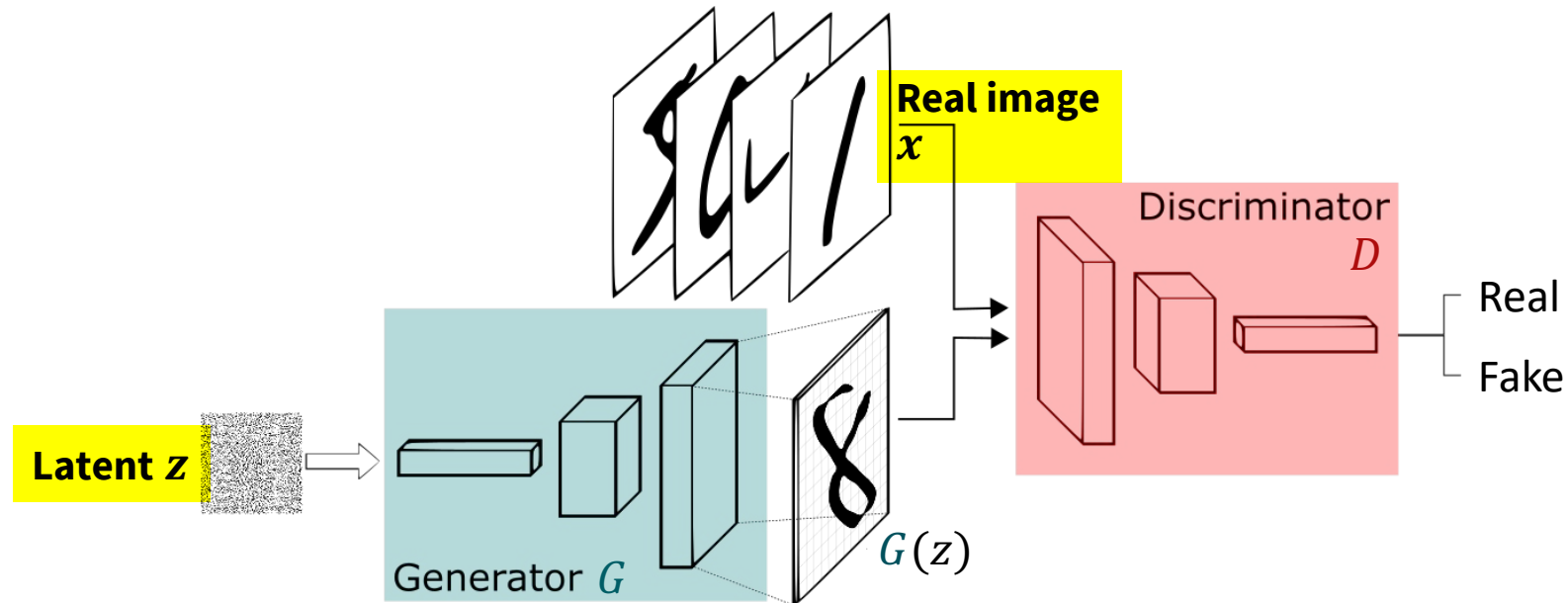
$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p(x)} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))]$$



Generative Adversarial Network (GAN)

Loss function:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\underset{\substack{\text{Real image} \\ \text{sampled from} \\ \text{the data distribution}}}{x \sim p(x)}} [\log D(x)] + \mathbb{E}_{\underset{\substack{\text{Latent} \\ \text{sampled from} \\ \text{the latent distribution}}}{z \sim p(z)}} [\log(1 - D(G(z)))]$$

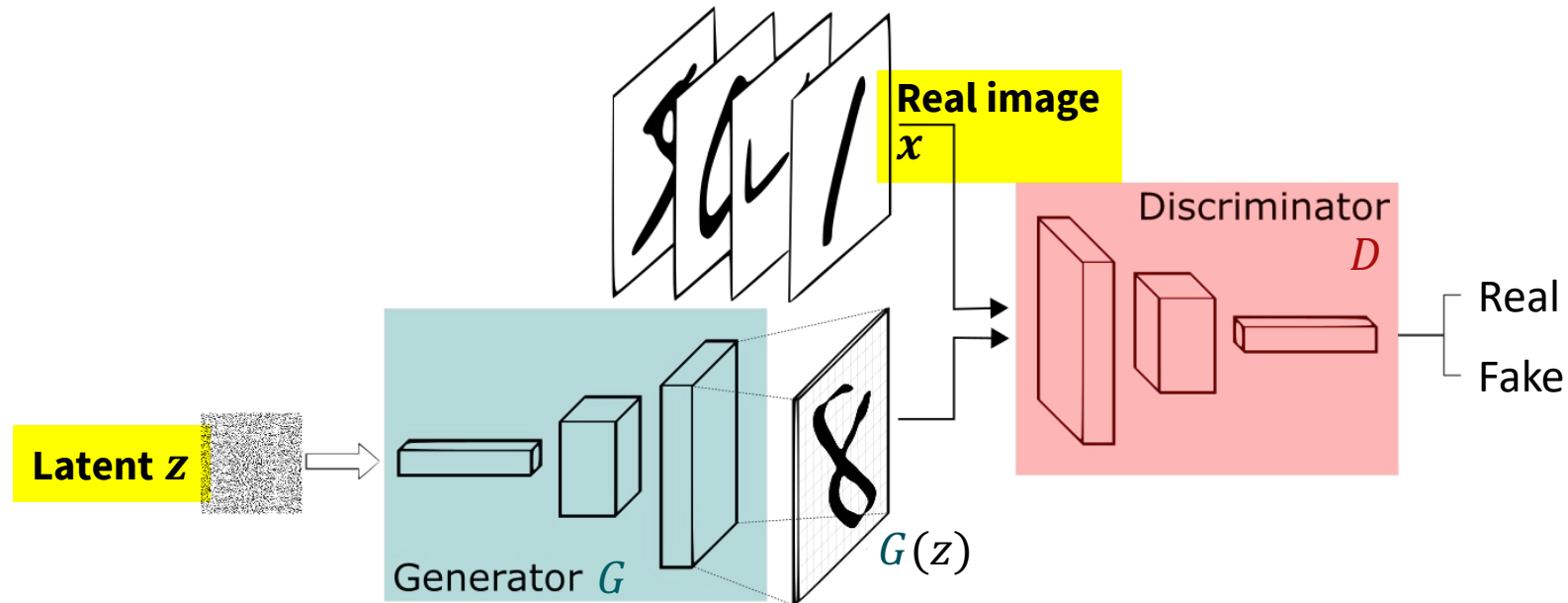


Generative Adversarial Network (GAN)

Loss function:

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

The *fake* image synthesized from the latent z .



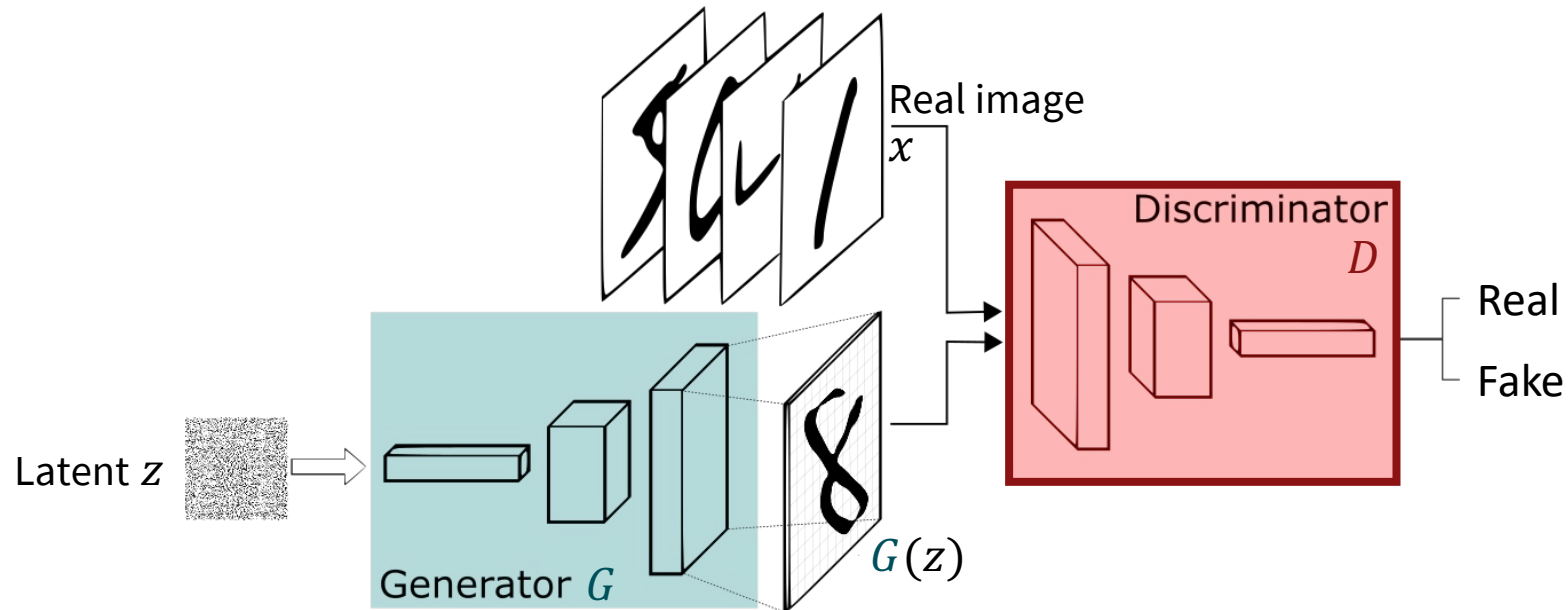
Generative Adversarial Network (GAN)

Loss function:

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

The probability of the *real* image x being a real image.

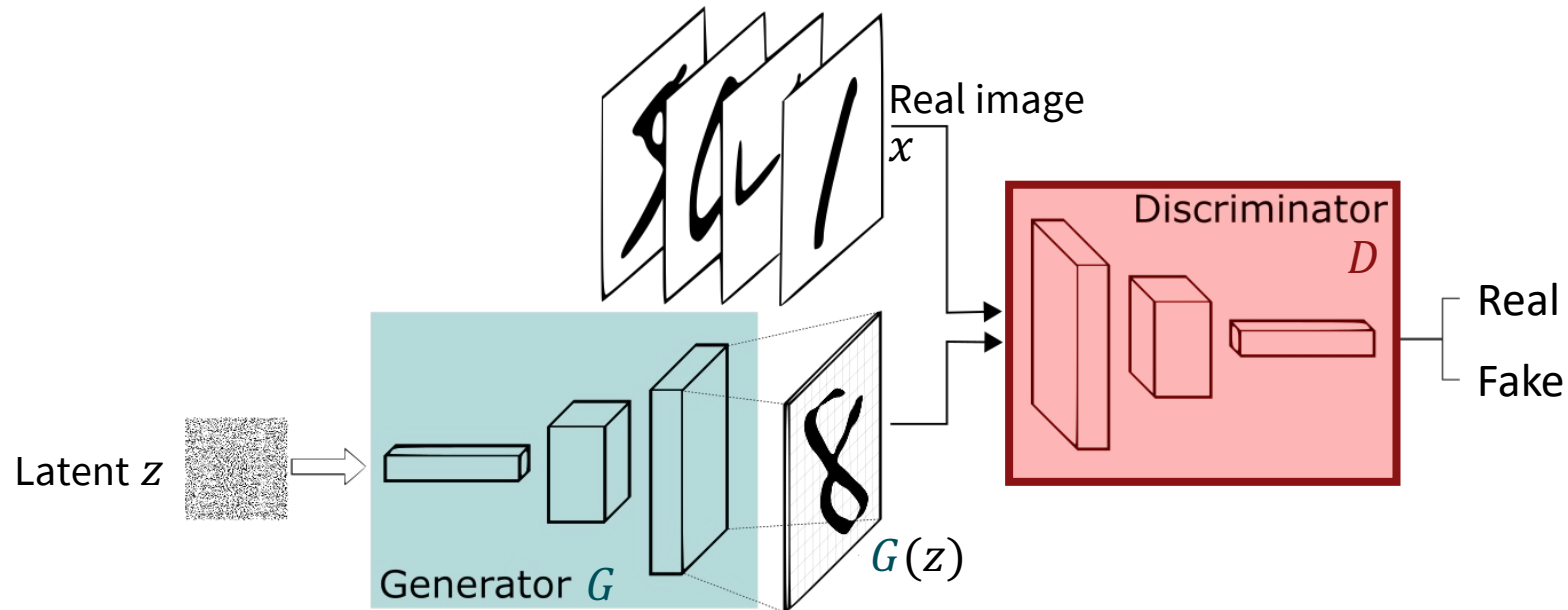
The probability of the *fake* image $G(z)$ being a real image.



Generative Adversarial Network (GAN)

Loss function:

$$\min_G \max_D V(D, G) = \underbrace{\mathbb{E}_{x \sim p(x)} [\log D(x)]}_{\text{Discriminator \textit{increases} the probability for the real image } x \dots} + \underbrace{\mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))]}_{\text{and \textit{decreases} the probability for the synthetic image } G(z).}$$

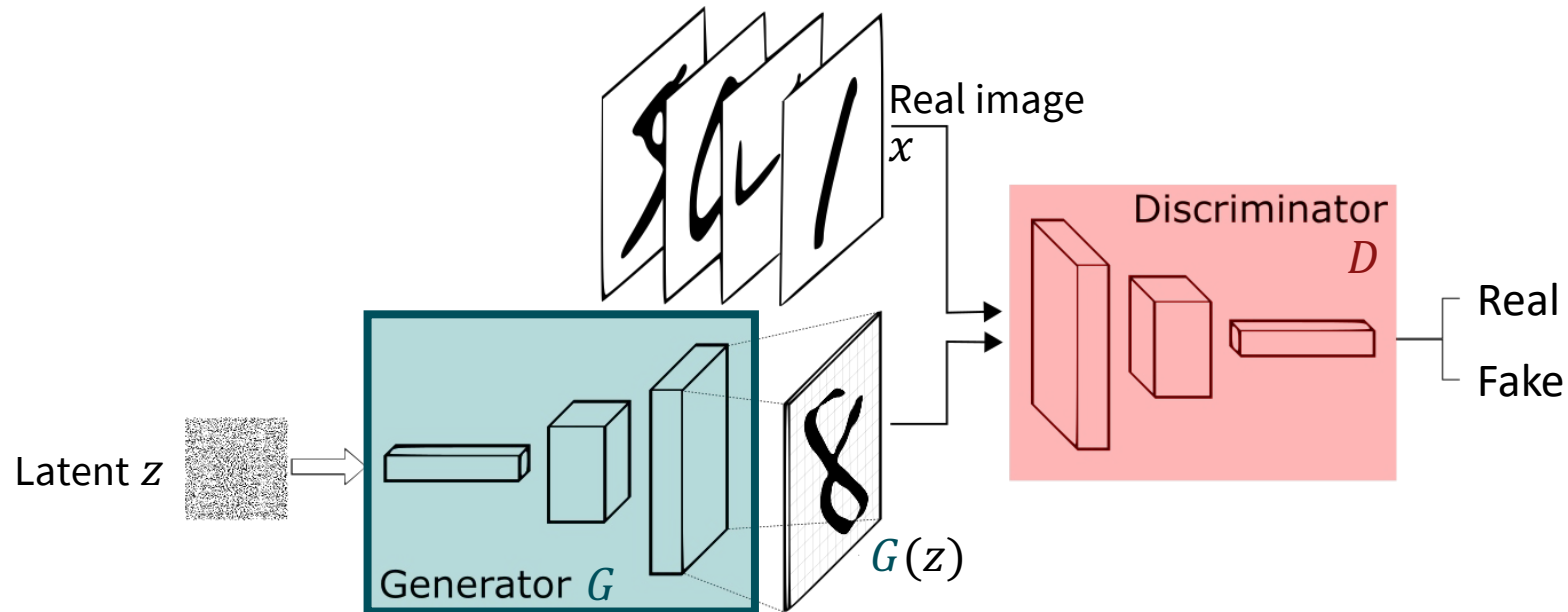


Generative Adversarial Network (GAN)

Loss function:

$$\min_{\underline{G}} \max_D V(D, G) = \mathbb{E}_{x \sim p(x)} [\log D(x)] + \mathbb{E}_{z \sim p(z)} [\log(1 - D(G(z)))]$$

Generator *increases*
the probability for the
synthetic image $G(z)$.



Generative Adversarial Network (GAN)

Loss function:

$$\min_{\underline{G}} \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))]$$

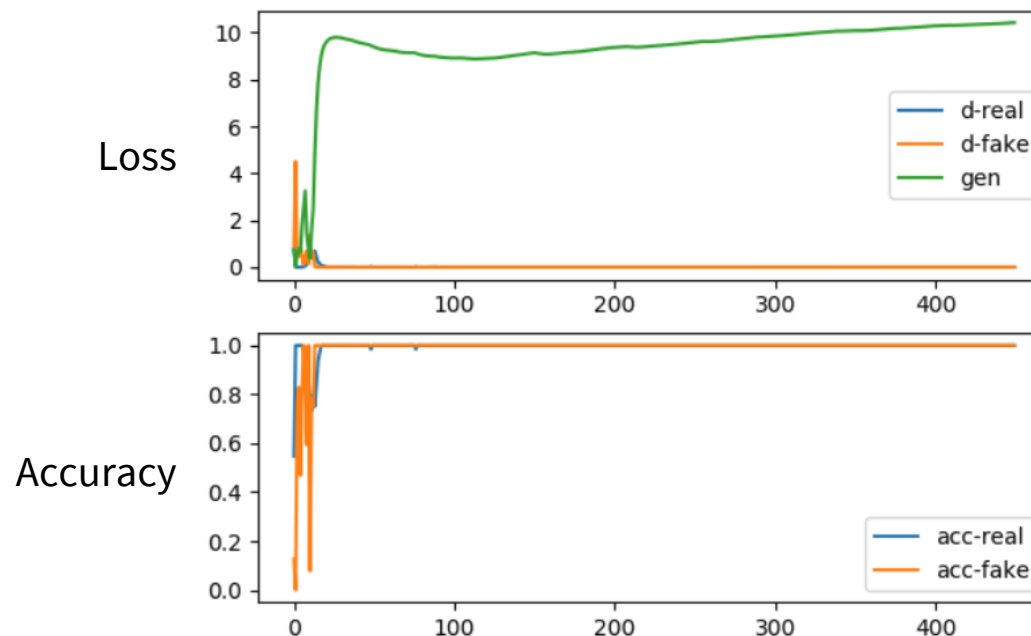
A **min-max (minimax)** optimization problem, which is known as **very difficult to solve!**

Challenges in GAN Training

1. Non-convergence & Instability
2. Mode collapse

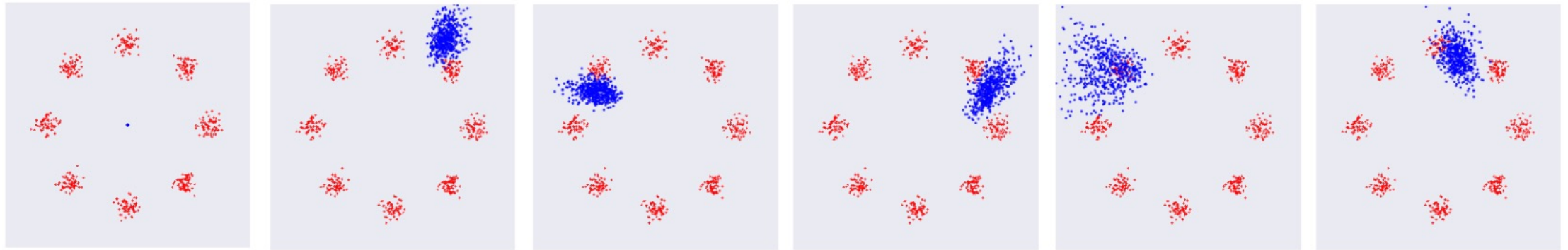
Non-Convergence & Instability

The **discriminator becomes too strong** and can easily distinguish between real and fake samples, leading to near-zero gradients for the generator and halting its learning.



Mode Collapse

The generator tends to **collapse** to a few modes of the data distribution instead of capturing the full diversity of the true data distribution.



Advances in GAN

Index	Software name	Language	Backend	Link
1	CGAN	Python	PyTorch	https://github.com/Lornatang/CGAN-PyTorch
2	DCGAN	Python	PyTorch	https://github.com/Natsu6767/DCGAN-PyTorch
3	AAEs	Python	TensorFlow	https://github.com/conan7882/adversarial-autoencoders
4	InfoGAN	Python	TensorFlow	https://github.com/openai/InfoGAN
5	SAD-GAN	—	—	—
6	LSGAN	Python	PyTorch	https://github.com/xudonmao/LSGAN
7	SRGAN	Python	TensorFlow	https://github.com/tensorlayer/SRGAN
8	WGAN	Python	PyTorch	https://github.com/Zeleni9/pytorch-wgan
9	CycleGAN	Python	TensorFlow	https://github.com/junyanz/CycleGAN
10	ProGAN	Python	PyTorch	https://github.com/tkarras/progressive_growing_of_gans
11	MidiNet	Python	TensorFlow	https://github.com/RichardYang40148/MidiNet
12	SN-GAN	Python	PyTorch	https://github.com/hanyoseob/pytorch-SNGAN
13	RGAN	Python	TensorFlow	https://github.com/ratschlab/RGAN
14	StarGAN	Python	PyTorch	https://github.com/yunjey/stargan
15	BigGAN	Python	PyTorch	https://github.com/ajbrock/BigGAN-PyTorch
16	MI-GAN	Python	TensorFlow	https://github.com/hazratalli/MI-GAN
17	AttGAN	Python	TensorFlow	https://github.com/LynnHo/AttGAN-Tensorflow
18	PATE-GAN	Python	TensorFlow	https://github.com/vanderschaarlab/mlforhealthlabpub/tree/main/alg/pategan
19	DM-GAN	Python	PyTorch	https://github.com/MinfengZhu/DM-GAN
20	SinGAN	Python	PyTorch	https://github.com/tamarott/SinGAN
21	POLY-GAN	Python	PyTorch	https://github.com/nile649/POLY-GAN
22	MIEGAN	—	—	—
23	VQGAN	Python	PyTorch	https://github.com/dome272/VQGAN-pytorch
24	DALL-E	Python	PyTorch	https://github.com/lucidrains/DALLE-pytorch
25	CEGAN	—	—	—
26	Seismogen	Python	PyTorch	https://github.com/Miffka/seismogen
27	MetroGAN	Python	PyTorch	https://github.com/zwj-Giser/MetroGAN
28	M3GAN	Python	PyTorch	https://github.com/SLZWVICTOR/M3GAN
29	CNTS	Python	PyTorch	https://github.com/BomBooooo/CNTS/tree/main
30	RidgeGAN	Python	PyTorch	https://github.com/rahisha-thottolil/ridgegan

StyleGAN2



Variational Autoencoder (VAE)

Kingma and Welling , Auto-Encoding Variational Bayes, ICLR 2014.

Variational Autoencoder (VAE)

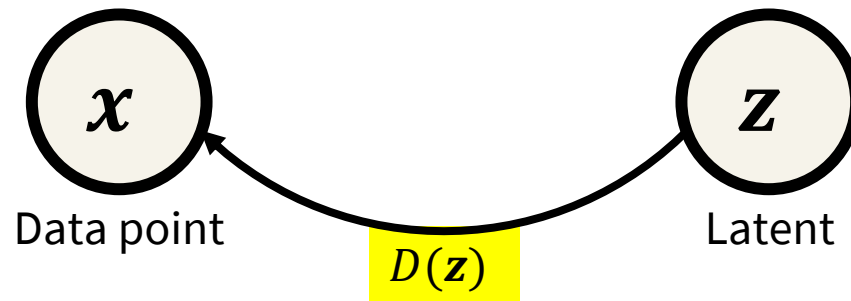
- How to make a generative model *without* solving a **minimax** problem?
- Let's represent the mapping from the latent distribution $p(\mathbf{z})$ to the data distribution $p(\mathbf{x})$ as a **conditional distribution** $p(\mathbf{x}|\mathbf{z})$.

Variational Autoencoder (VAE)

Let's consider the **decoder (generator)** as predicting the **mean** of the **conditional distribution** $p(\mathbf{x}|\mathbf{z})$:

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; D(\mathbf{z}), \underline{\sigma^2 \mathbf{I}})$$

Fixed variance




Basics

- Marginal distribution
- Expected value
- Bayes' rule
- Kullback–Leibler (KL) Divergence
- Jensen's inequality

Marginal Distribution

The **marginal** distribution of a **subset** of a set of random variables is the probability distribution of the variables contained in the subset.

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z}$$


Take the integral over \boldsymbol{z} to marginalize \boldsymbol{x} .

Marginal Distribution

Q. Given the joint probability table below, what is $p(x = 1)$?

		y		
		1	2	3
x	1	0.32	0.03	0.01
	2	0.06	0.24	0.02
	3	0.02	0.03	0.27

Expected Value

The **expected value** is the arithmetic mean of the possible values a random variable can take, **weighted** by the probability of those outcomes.

$$\mathbb{E}_{p(\boldsymbol{x})}[\boldsymbol{x}] = \int \boldsymbol{x} \cdot p(\boldsymbol{x}) d\boldsymbol{x}$$

Marginal Distribution

Q. What is $\mathbb{E}_{p(x)}[x]$?

		y		
		1	2	3
x	1	0.32	0.03	0.01
	2	0.06	0.24	0.02
	3	0.02	0.03	0.27

Bayes' Rule

Bayes' rule is a mathematical formula used to determine the **conditional probability** of events.

$$\overset{\text{Posterior}}{\boxed{p(\mathbf{z}|\mathbf{x})}} = \frac{\overset{\text{Likelihood}}{\boxed{p(\mathbf{x}|\mathbf{z})}} \overset{\text{Prior}}{\boxed{p(\mathbf{z})}}}{\underset{\text{Marginal}}{\boxed{p(\mathbf{x})}}}$$

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

Marginal Distribution

Q. What is $p(y = 2|x = 1)$?

		y		
		1	2	3
x	1	0.32	0.03	0.01
	2	0.06	0.24	0.02
	3	0.02	0.03	0.27

Kullback–Leibler (KL) Divergence

Kullback–Leibler (KL) divergence is a measure of how one probability distribution p is different from a reference probability distribution q :

$$D_{\text{KL}}(p \parallel q) = \int p(\mathbf{x}) \log \left(\frac{p(\mathbf{x})}{q(\mathbf{x})} \right) d\mathbf{x} = \mathbb{E}_{p(\mathbf{x})} \left[\log \left(\frac{p(\mathbf{x})}{q(\mathbf{x})} \right) \right]$$

Q. What is $D_{\text{KL}}(p \parallel p)$?

Kullback–Leibler (KL) Divergence

Q. Homework

When $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \sigma^2 \mathbf{I})$ and $q(\mathbf{x}) = \mathcal{N}(\mathbf{x}; \mathbf{0}, \mathbf{I})$,

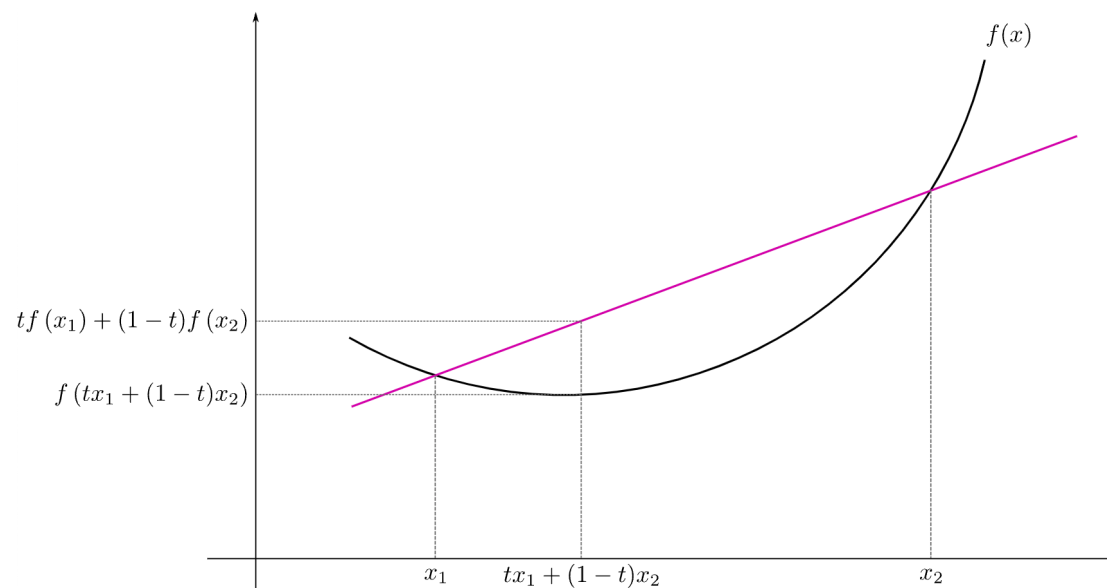
What is $D_{\text{KL}}(p \parallel q)$?

Jensen's Inequality

f is a **convex** function if

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2)$$

for all x_1, x_2 , and $t \in [0,1]$.



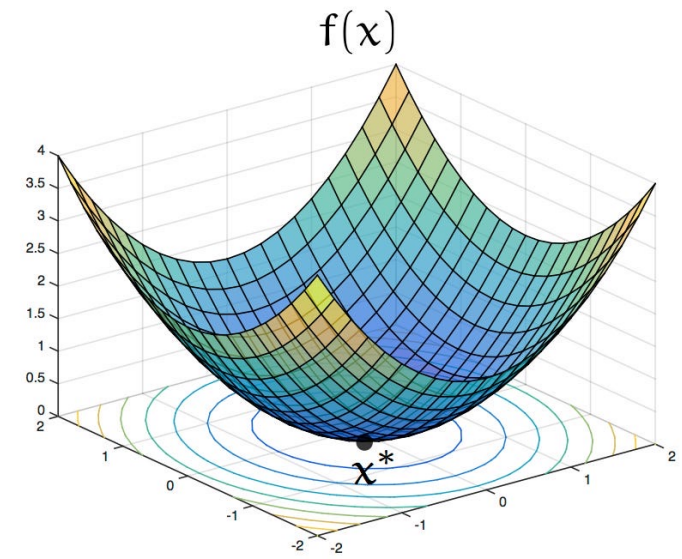
Jensen's Inequality

f is a **convex** function if

$$f\left(\sum_i t_i \mathbf{x}_i\right) \leq \sum_i t_i f(\mathbf{x}_i)$$

Affine combination \swarrow

for all \mathbf{x}_i and $t \in [0,1]$.



Jensen's Inequality

if \mathbf{x} is a random variable and f is a **convex** function, then

$$f(\mathbb{E}_{p(\mathbf{x})}[\mathbf{x}]) \leq \mathbb{E}_{p(\mathbf{x})}[f(\mathbf{x})]$$

Since the expected value is an affine combination.

Q. What if f is a **concave** function?

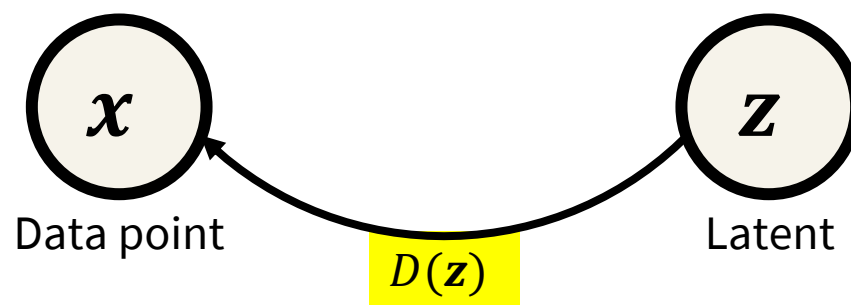
Back to VAE...

- Marginal distribution
- Expected value
- Bayes' rule
- Kullback–Leibler (KL) Divergence
- Jensen's inequality

Variational Autoencoder (VAE)

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$$

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}; D(\mathbf{z}), \sigma^2 \mathbf{I})$$



Variational Autoencoder (VAE)

For all given real images \mathbf{x} , we want to maximize the marginal probability:

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$$

How to compute the **integral**?

Monte-Carlo method for \mathbf{x} and \mathbf{z} takes too much time...
→ Intractable.

Variational Autoencoder (VAE)

Or, we can compute

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

But this conditional distribution
is unknown.

Evidence Lower Bound (ELBO)

- We cannot directly maximize $p(\mathbf{x}) = \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{z}|\mathbf{x})}$ since $p(\mathbf{z}|\mathbf{x})$ is unknown.
- Let's think about the **lower bound** of $\log p(\mathbf{x})$:

$$\log p(\mathbf{x}) \geq \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \right]$$

Evidence Lower Bound (ELBO)

- $q_{\phi}(\mathbf{z}|\mathbf{x})$ is a **variational distribution** with parameters ϕ .
- E.g., a Gaussian distribution with mean and variance as parameters.
- Consider $q_{\phi}(\mathbf{z}|\mathbf{x})$ as an **arbitrary conditional distribution** that may not be the same with $p(\mathbf{z}|\mathbf{x})$.
- We use the **proxy** distribution $q_{\phi}(\mathbf{z}|\mathbf{x})$ since we don't know $p(\mathbf{z}|\mathbf{x})$.

Evidence Lower Bound (ELBO)

- $\mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \right]$ is the expected value over \mathbf{z} sampled from the variational distribution $q_{\phi}(\mathbf{z}|\mathbf{x})$.
- Why $\log p(\mathbf{x}) \geq \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \right]$?
Because of the Jensen's inequality.

Evidence Lower Bound (ELBO)

$$\log p(\mathbf{x}) = \log \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z}$$

$$= \log \int p(\mathbf{x}, \mathbf{z}) \frac{q_\phi(\mathbf{z}|\mathbf{x})}{q_\phi(\mathbf{z}|\mathbf{x})} d\mathbf{z}$$

$$= \boxed{\log} \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\frac{p(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right]$$

Concave
function

$$\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right]$$

Evidence Lower Bound (ELBO)

Q. Homework

What is the **Jensen gap** $\left(\log p(\mathbf{x}) - \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \right] \right)$?

Evidence Lower Bound (ELBO)

Let's decompose ELBO:

$$\begin{aligned}\mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \right] &= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left[\log \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \right] \\&= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x}|\mathbf{z})] - \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left[\log \frac{q_{\phi}(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})} \right] \\&= \boxed{\mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x}|\mathbf{z})]} - \boxed{D_{KL} \left(q_{\phi}(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z}) \right)}\end{aligned}$$

Reconstruction term
to be *maximized*.

Prior matching term
to be *minimized*.