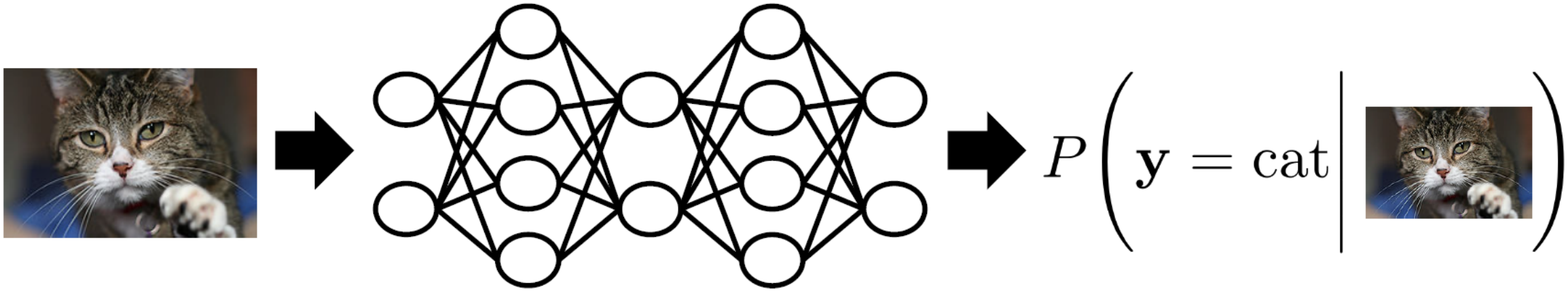
Deep Generative Modeling

Discriminative vs. Generative Models

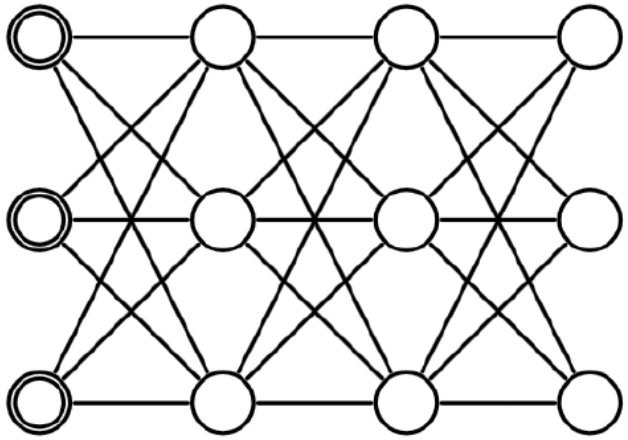
- Given an observed variable $\boldsymbol{x}$ and a target variable $\boldsymbol{y}$,
- Discriminative modeling estimates the conditional prob $p(\boldsymbol{y}|\boldsymbol{x})$
  - E.g., logistic regression, ImageNet classifiers, etc.



Discriminative vs. Generative Models

- Given an observed variable $\boldsymbol{x}$ and a target variable $\boldsymbol{y}$,
- Generative modeling estimates the joint distribution $p(\boldsymbol{x}, \boldsymbol{y})$
  - E.g., Boltzmann machines, Gaussian mixture models, etc.

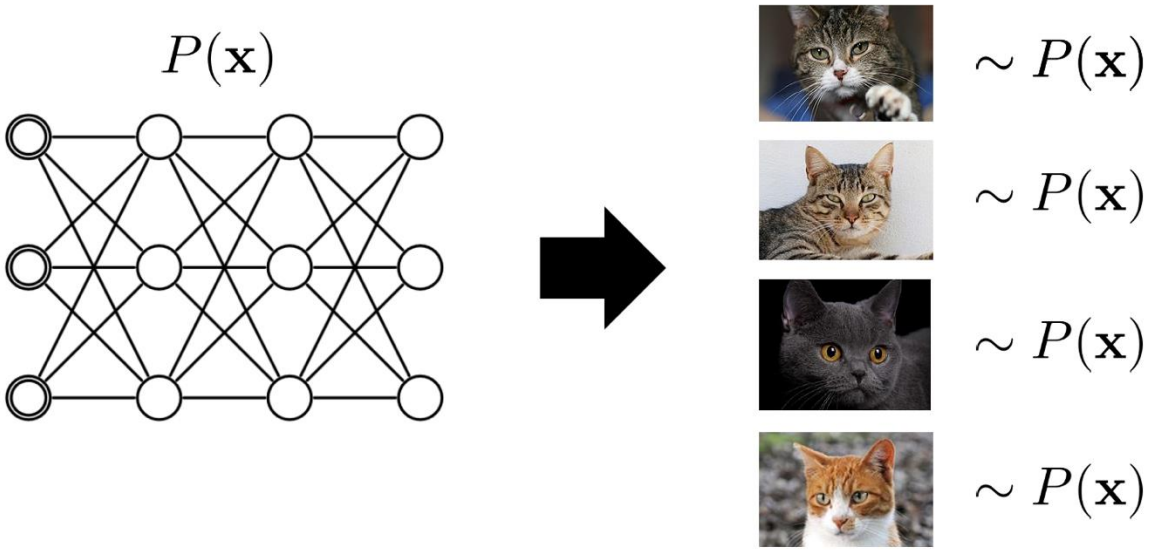
$\mathbf{y} = \mathrm{cat}$



$$P\left(\mathbf{y} = \mathrm{cat},\ \right)$$

Why Generative Models?

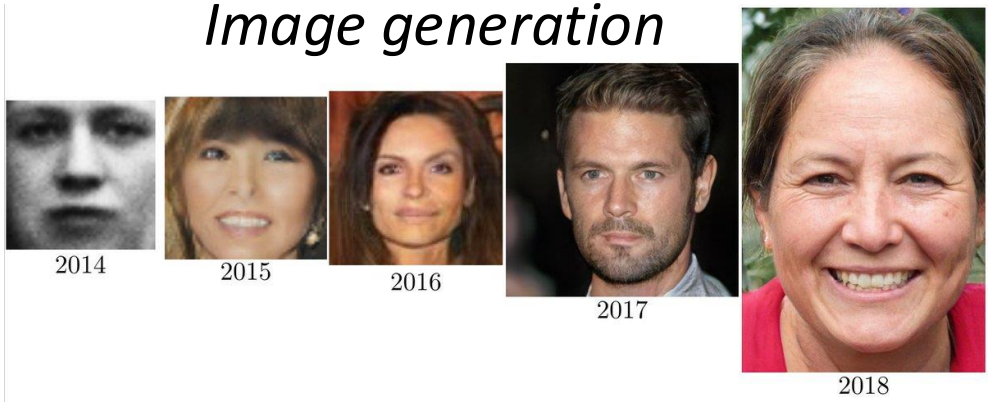
- Without assuming $\boldsymbol{y}$, generative models learn $p(\boldsymbol{x})$ from given data.
- $p(\boldsymbol{x})$ enables use to generate new data similar to the training dataset.
- We can use various sampling methods for generation based on $p(\boldsymbol{x})$.



Why Generative Models?

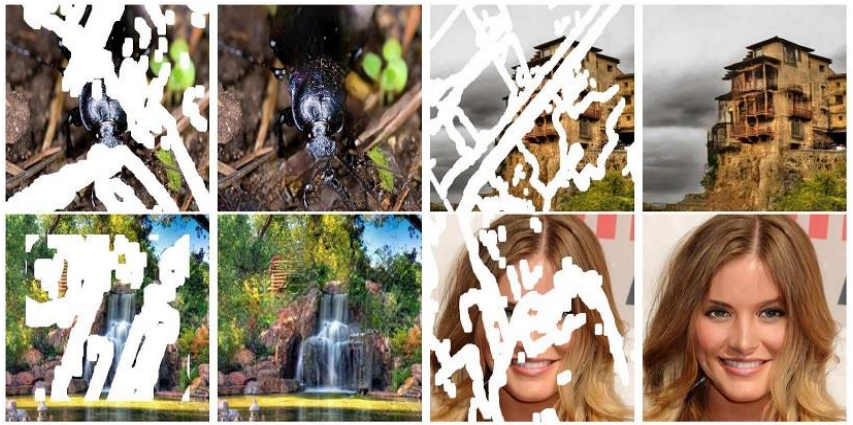
- Many real-world problems can be formulated as generation tasks.

Image generation



p(high res | low res)

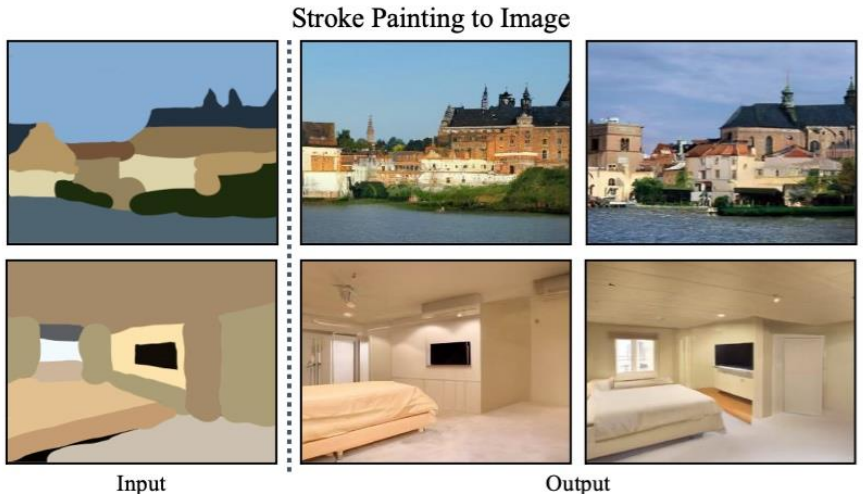
p(full image | masked image)



p(color image | grayscale)

Why Generative Models?

- Many real-world problems can be formulated as generation tasks.



Why Generative Models?

- Many real-world problems can be formulated as generation tasks.

Custom prompt

To get an A+ in deep generative models, students have to

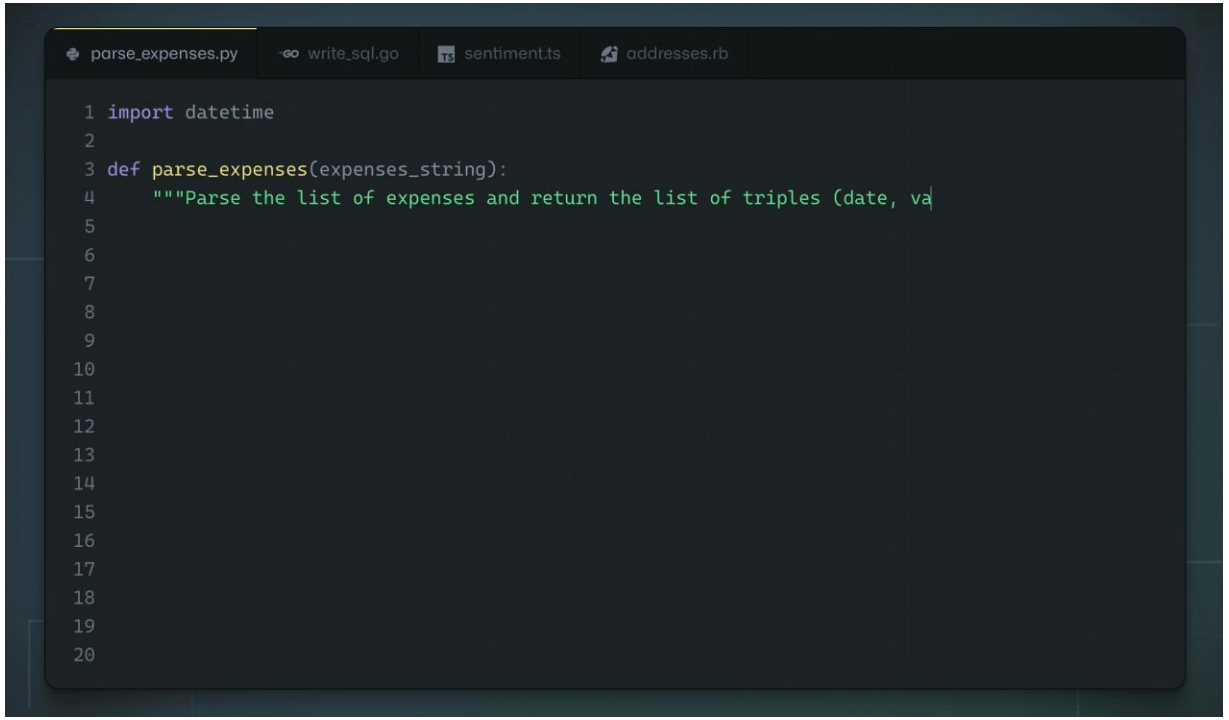
Completion

To get an A+ in deep generative models, students have to be willing to work with problems that are a whole lot more interesting than, say, the ones that most students work on in class. If you're a great student, the question above can be avoided and you'll be able to do great work, but if you're not, you will need to go beyond the basics before getting good.

Now to be clear, this advice is not just for the deep-learning crowd; it is good advice for any student who is taking his or her first course in machine learning.

The key point is that if you have a deep, deep brain of a computer scientist, that's just as important to you.

p(next word | prev words)



Why Generative Models?

- Many real-world problems can be formulated as generation tasks.

TEXT PROMPT an armchair in the shape of an avocado. . . .

AI-GENERATED IMAGES

Edit prompt or view more images↓

p(image | caption)

TEXT PROMPT a store front that has the word ‘openai’ written on it. . . .

AI-GENERATED IMAGES

Why Generative Models?

- Many real-world problems can be formulated as generation tasks.

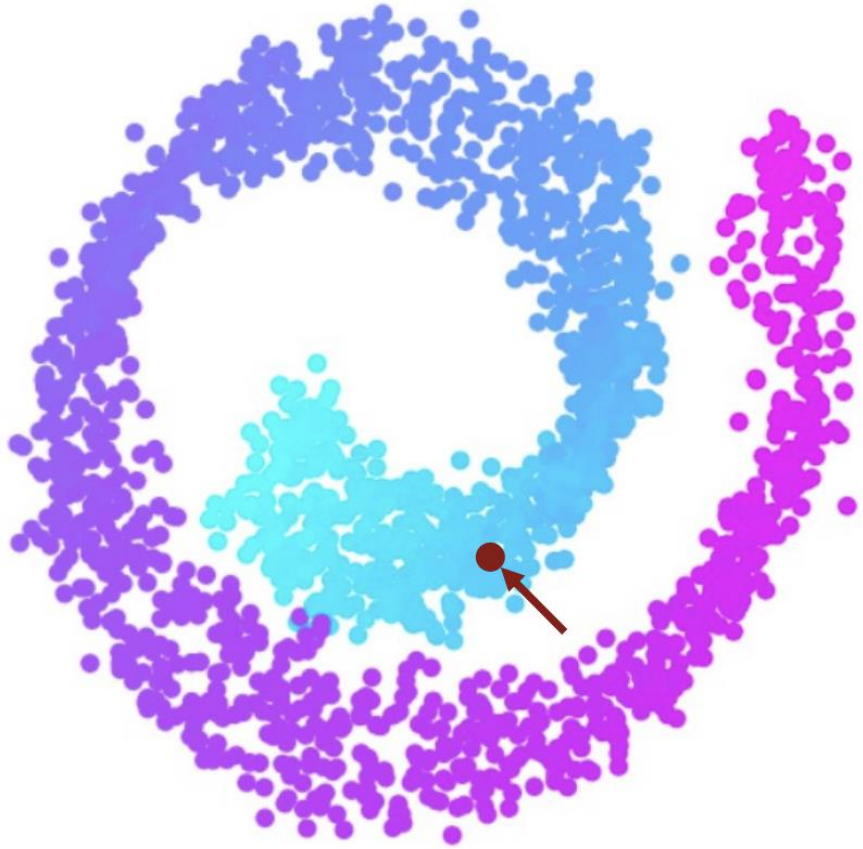
Midjourney: Prompt: "a hyper-realistic image of Harry Potter"

Generation and Sampling

- How can we generate a new realistic photo?

Generation and Sampling

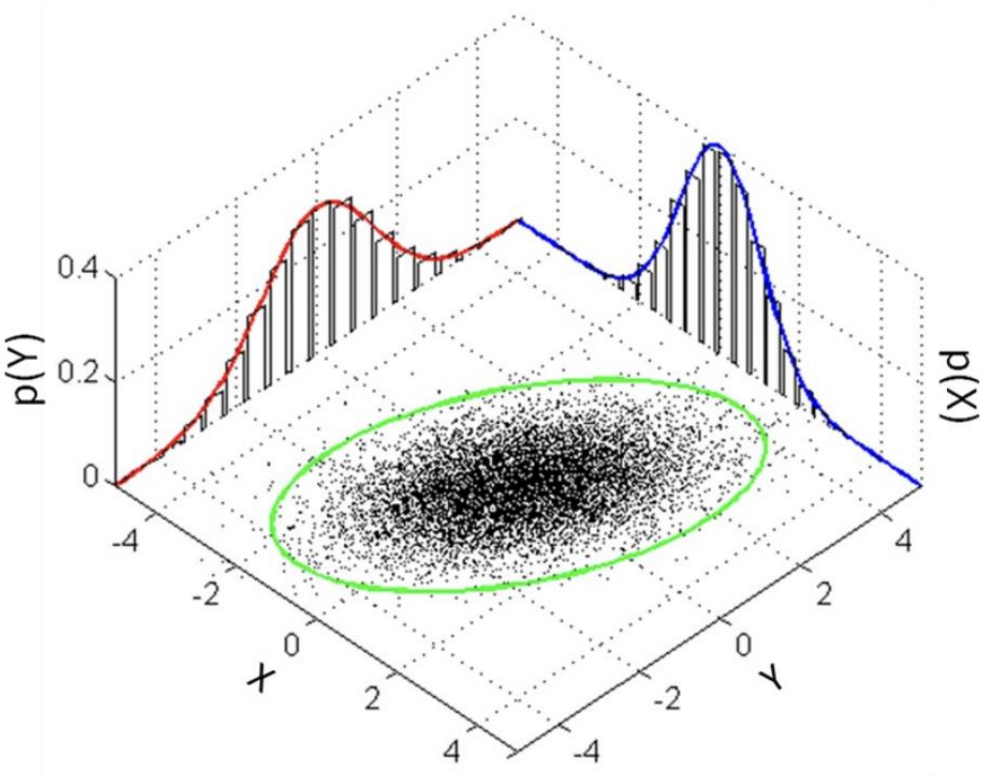
- How can we sample a new plausible 2D point?



Statistical Perspective

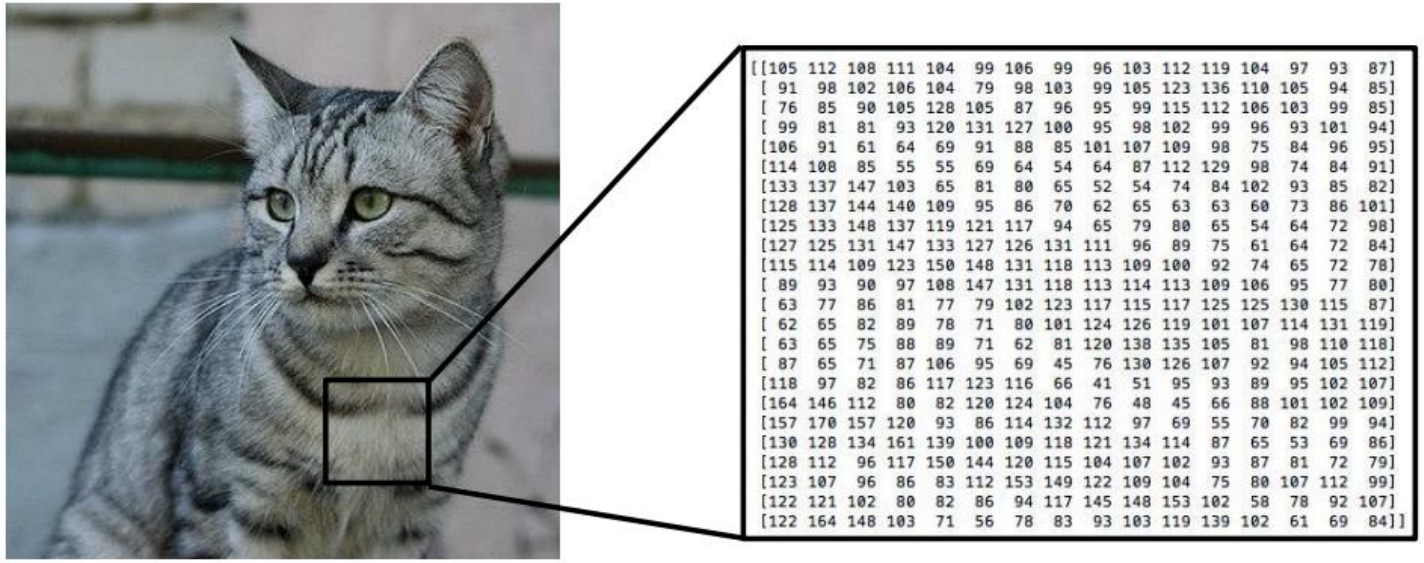
- We will view generation/sampling as
  - there being a probability distribution of the data, and
  - the given points are samples from the probability distribution.

- If the probability distribution is known, then for some specific distributions, we can sample from them directly.



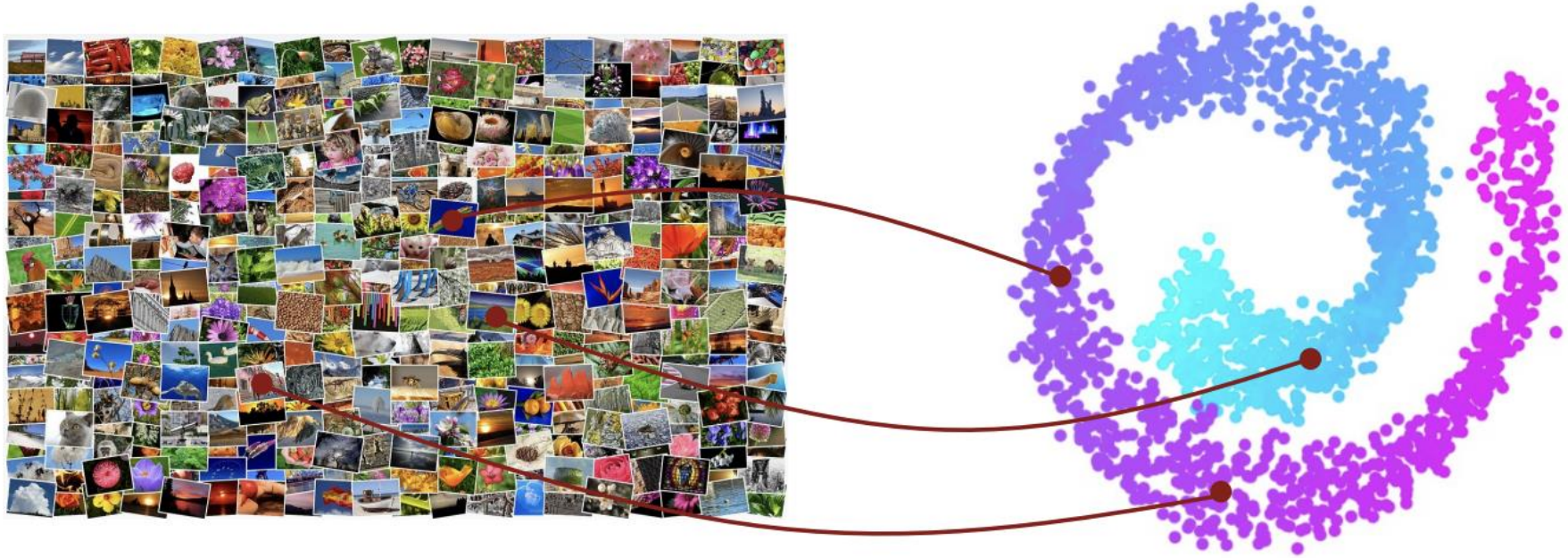
Statistical Perspective for Real Images

- Let’s consider RGB images with a resolution of 256x256.
- An image can be represented by a 256x256x3 vector.
- This means that an image is a point in a 256x256x3-dimensional space.



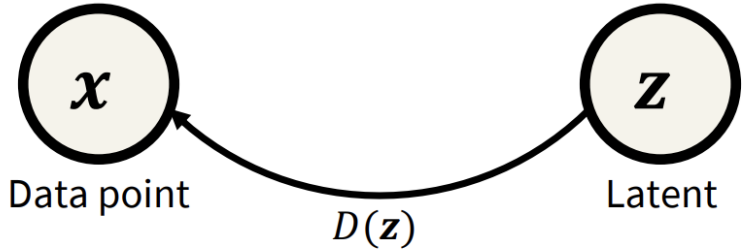
Statistical Perspective for Real Images

- Let's consider real images $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ as samples from a data distribution $p(\boldsymbol{x})$ 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 $p(\boldsymbol{z})$ (e.g., a standard normal distribution) to the data distribution $p(\boldsymbol{x})$
  - $\boldsymbol{z}$: Latent variable, $p(\boldsymbol{z})$: Latent distribution
- Sample from $p(\boldsymbol{z})$ and map it to a data point.



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

Let's use a neural network for $D(\boldsymbol{z})$!

Variational Autoencoder

Autoencoder (AE)

- A starting point is the autoencoder (AE).
  - AE consists of two networks: a deterministic encoder and a deterministic decoder.

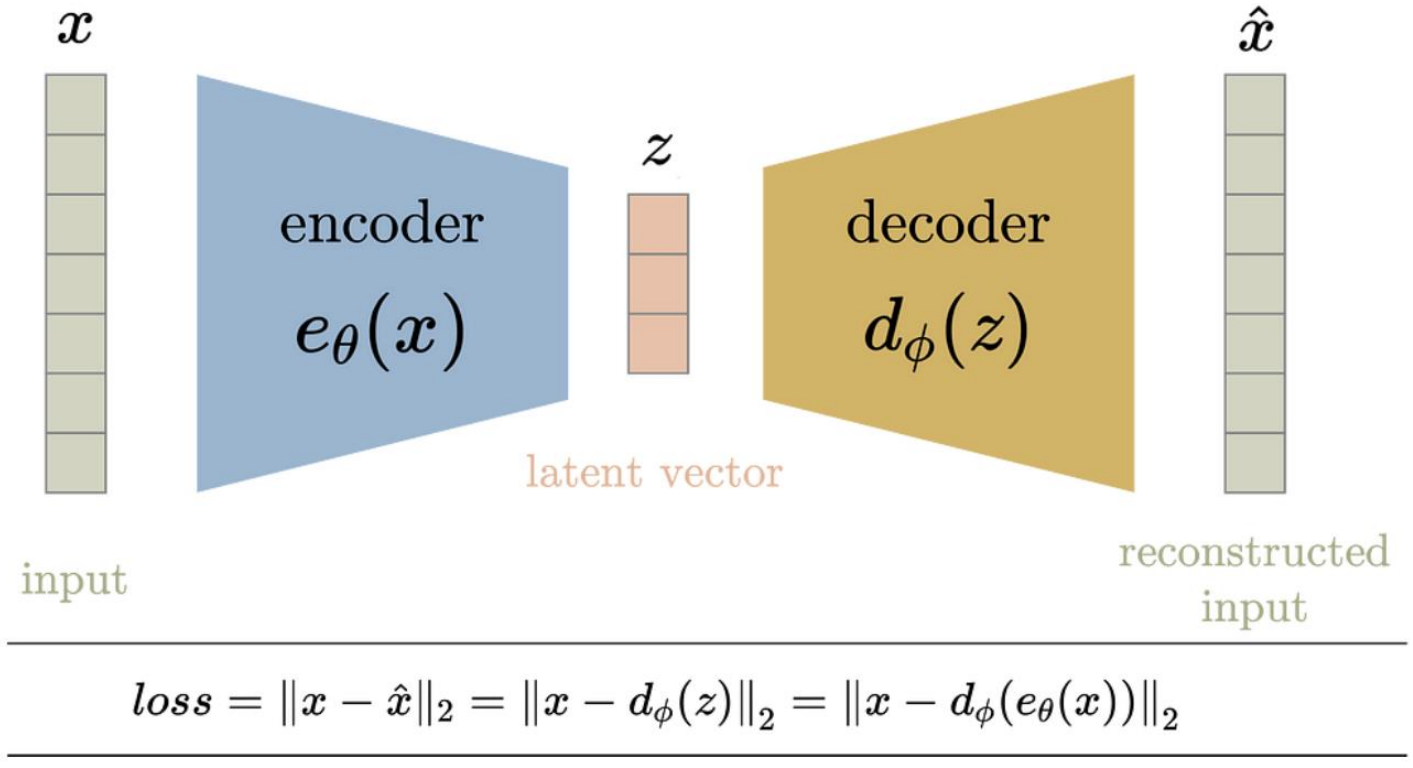


Figure: https://towardsdatascience.com/difference-between-autoencoder-ae-and-variational-autoencoder-vae-ed7be1c038f2/

Autoencoder (AE)

- What we need is the decoder (latent → input data).
  - AE enables accurate reconstruction, but the latent space is unstructured: randomly sampling latent codes usually produces meaningless outputs.
  - How do we guarantee that a latent is mapped to a data point?

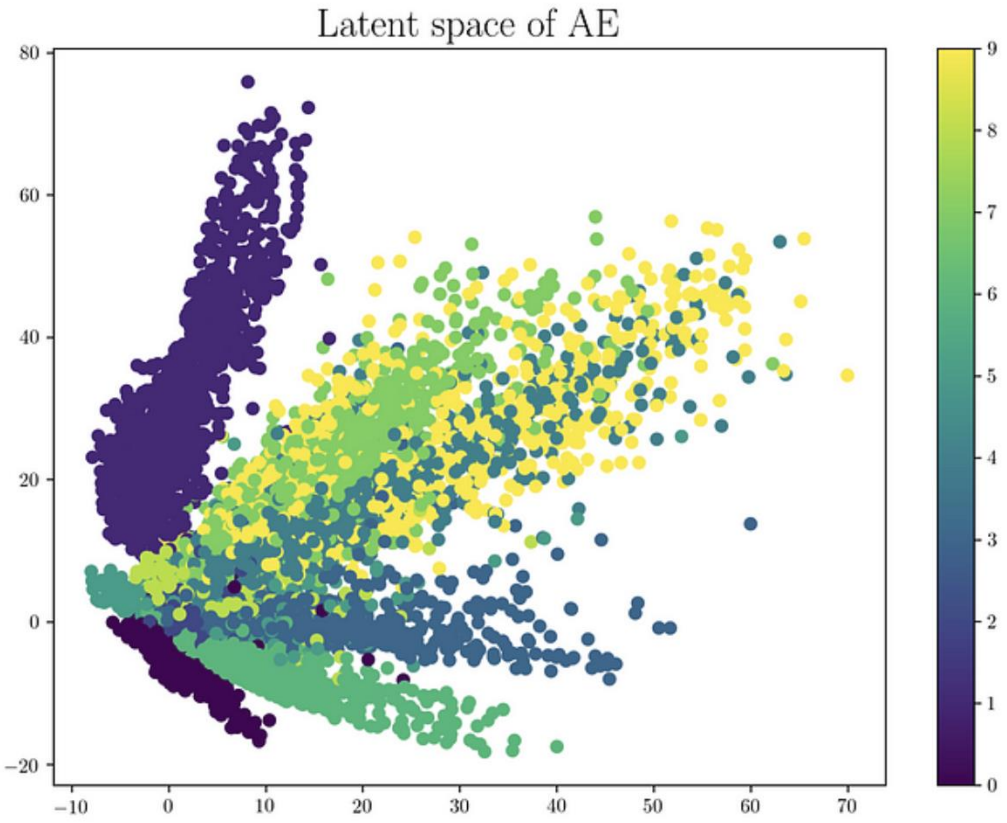
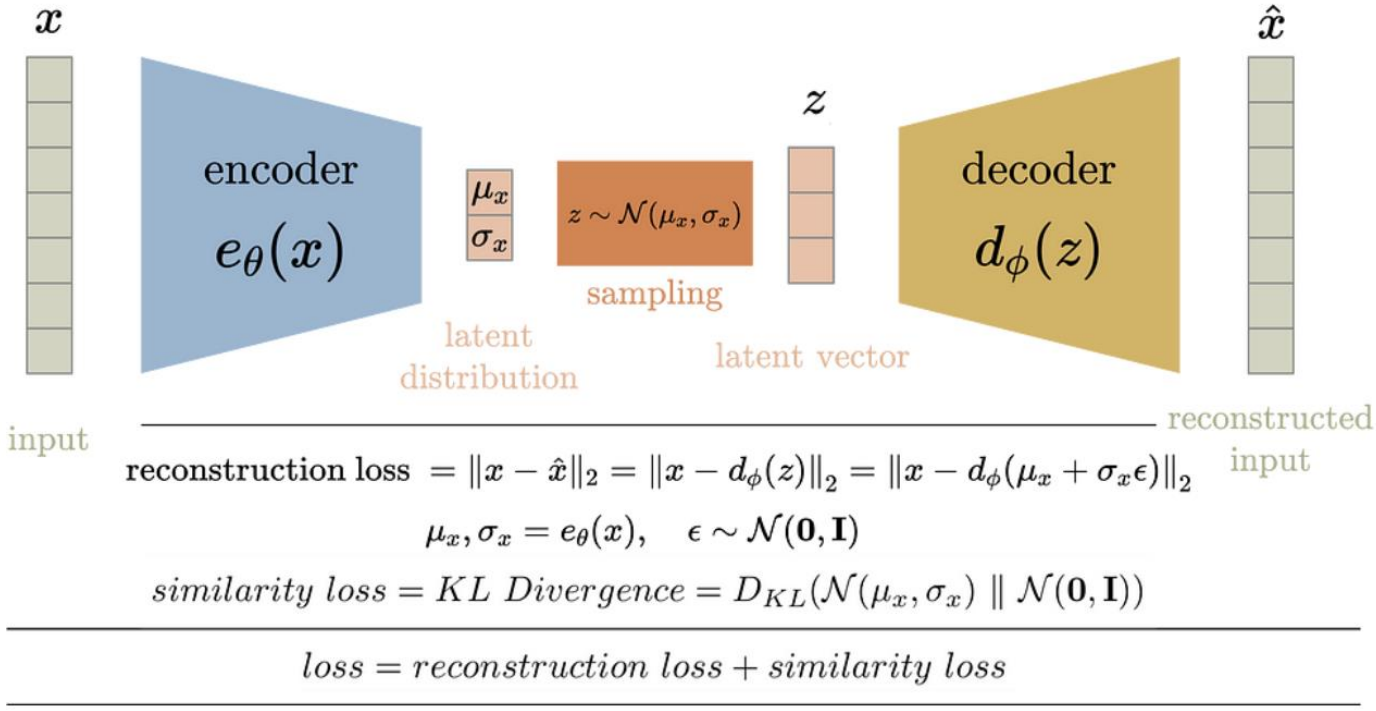


Figure: https://towardsdatascience.com/difference-between-autoencoder-ae-and-variational-autoencoder-vae-ed7be1c038f2/

Variational Autoencoder (VAE)

- VAE imposes a probabilistic structure on the latent space.
  - AE (simple reconstruction) → a generative model



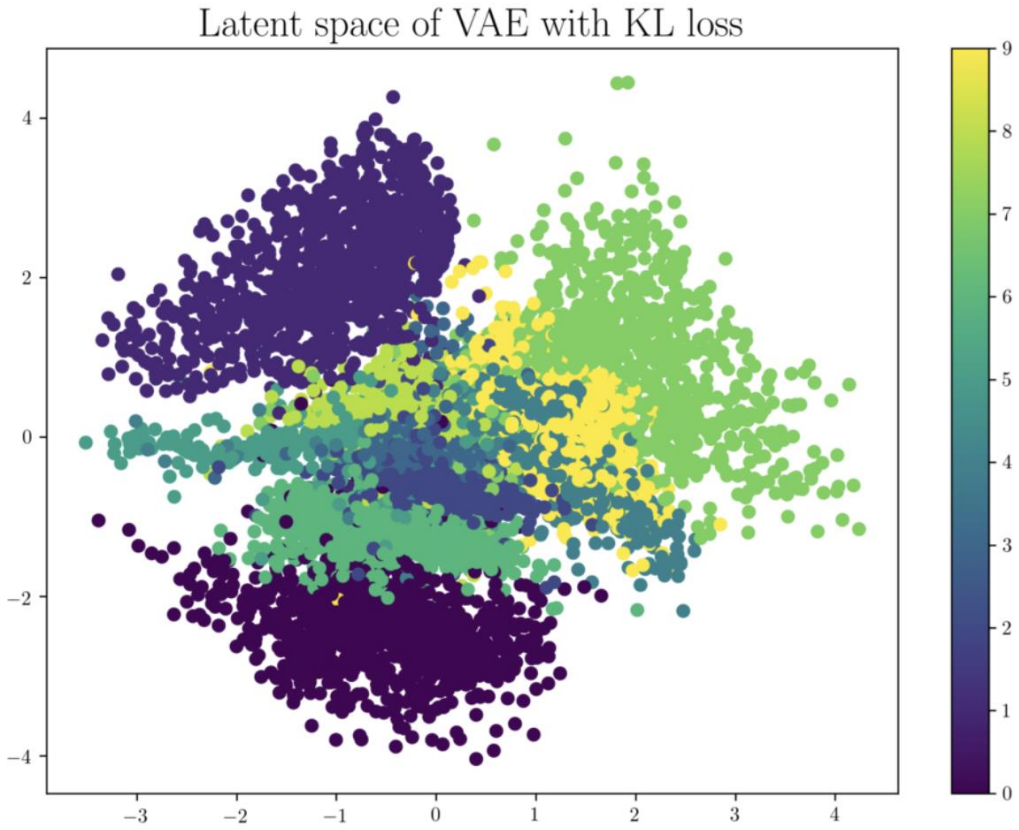
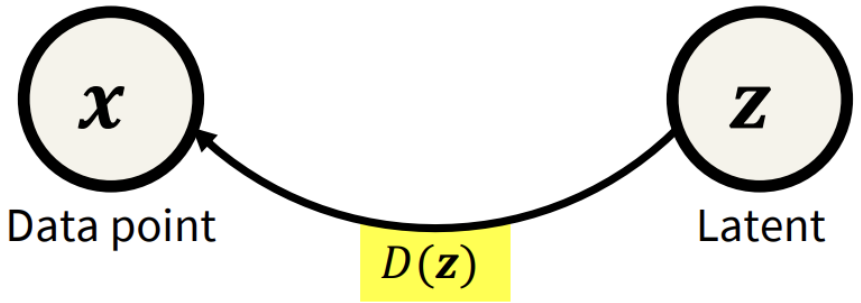


Figure: https://towardsdatascience.com/difference-between-autoencoder-ae-and-variational-autoencoder-vae-ed7be1c038f2/

Variational Autoencoder (VAE)

- Let's represent the mapping from the latent distribution $p(\boldsymbol{z})$ to the data distribution $p(\boldsymbol{x})$ as a conditional distribution $p(\boldsymbol{x}|\boldsymbol{z})$.
- Consider the decoder (generator) as predicting the mean of the conditional distribution $p(\boldsymbol{x}|\boldsymbol{z})$:

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



(Appendix) Basics

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

(Appendix) 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 $\boldsymbol{x}$.

(Appendix) Marginal Distribution

- Q) Given the joint probability table below, what is $p(\boldsymbol{x} = 1)$?

		$\boldsymbol{y}$		
		1	2	3
$\boldsymbol{x}$	1	0.32	0.03	0.01
	2	0.06	0.24	0.02
	3	0.02	0.03	0.27

(Appendix) 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.

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

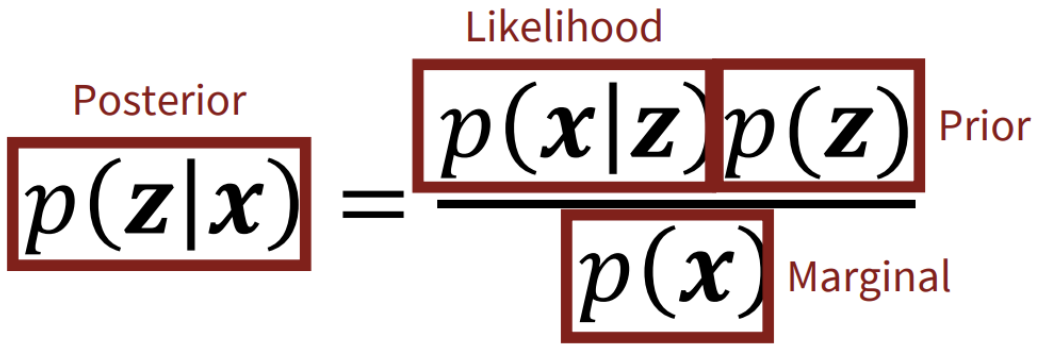
(Appendix) Expected Value

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

		$\boldsymbol{y}$		
		1	2	3
$\boldsymbol{x}$	1	0.32	0.03	0.01
	2	0.06	0.24	0.02
	3	0.02	0.03	0.27

(Appendix) Bayes' Rule

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



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

(Appendix) Bayes' Rule

- Q) What is $p(\boldsymbol{y} = 2|\boldsymbol{x} = 1)$?

		$\boldsymbol{y}$		
		1	2	3
$\boldsymbol{x}$	1	0.32	0.03	0.01
	2	0.06	0.24	0.02
	3	0.02	0.03	0.27

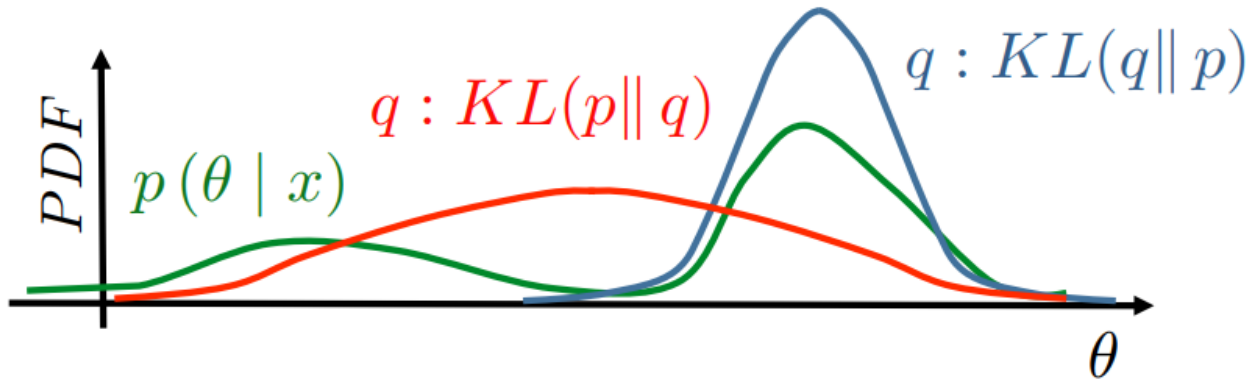
(Appendix) Kullback-Leibler Divergence

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

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

(Appendix) Kullback-Leibler Divergence

- Properties
  - $D_{\mathrm{KL}}(p||q) \geq 0$
  - $D_{\mathrm{KL}}(p||q) = 0 \Leftrightarrow p(\boldsymbol{x}) = q(\boldsymbol{x})$
  - $D_{\mathrm{KL}}(p||q) \neq D_{\mathrm{KL}}(q||p)$



- In general, $D_{\mathrm{KL}}(p||q)$ is called forward KL, whereas $D_{\mathrm{KL}}(q||p)$ is called reverse KL.
  - Forward KL is known as zero avoiding while reverse KL is known as zero forcing.

(Appendix) Kullback-Leibler Divergence

- Q)
  When $p(\boldsymbol{x}) = N(\boldsymbol{x}; \boldsymbol{\mu}, \sigma^2 \mathbf{I})$ and $q(\boldsymbol{x}) = N(\boldsymbol{x}; \mathbf{0}, \mathbf{I})$, what is $D_{\mathrm{KL}}\,(p||q)$?

$$\text{If } p(\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{x};\ \boldsymbol{\mu}_p, \sigma_p^2 \mathbf{I}\right),\ q(\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{\mu}_q, \sigma_q^2 \mathbf{I}\right),\ \boldsymbol{x} \in \mathbb{R}^{\boldsymbol{d}},$$

$$D_{\mathrm{KL}}(p \parallel q) = \frac{1}{2}\left[d\left(\frac{\sigma_p^2}{\sigma_q^2} - 1 - \ln\frac{\sigma_p^2}{\sigma_q^2}\right) + \frac{\left\|\boldsymbol{\mu}_p - \boldsymbol{\mu}_q\right\|^2}{\sigma_q^2}\right].$$

When $\boldsymbol{\mu}_q = 0, \sigma_q = 1,$

$$D_{\mathrm{KL}}(p \parallel q) = \frac{1}{2}\left[d\left(\sigma_p^2 - 1 - \ln\sigma_p^2\right) + \left\|\boldsymbol{\mu}_p\right\|^2\right]$$

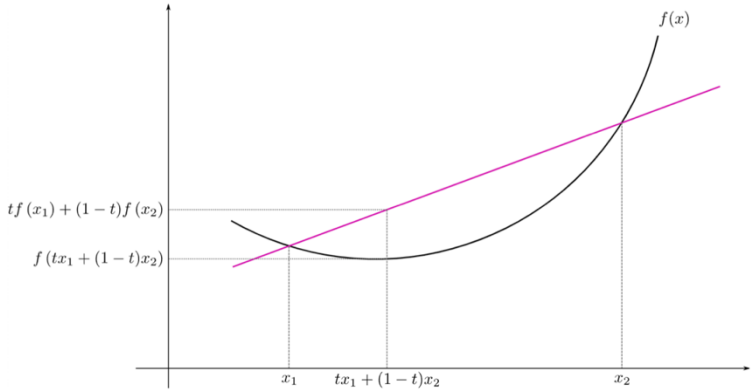
(Appendix) Jensen's Inequality

- f is a convex function if

$$f(t\boldsymbol{x}_1 + (1-t)\boldsymbol{x}_2) \leq tf(\boldsymbol{x}_1) + (1-t)f(\boldsymbol{x}_2)$$

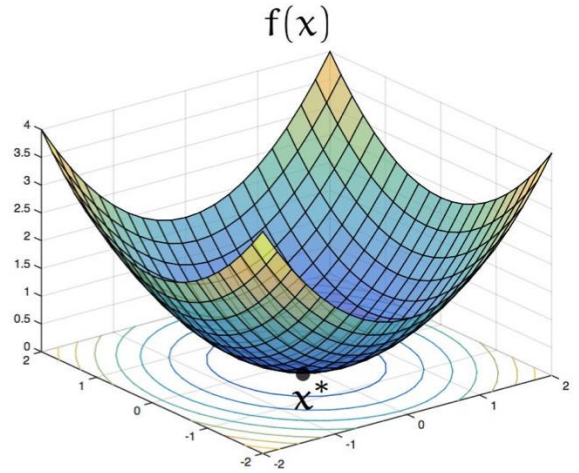
convex combination

for all $\boldsymbol{x}_1, \boldsymbol{x}_2$ and $t \in [0,1]$



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

for all $\boldsymbol{x}_i$ such that $t_i \geq 0$ and $\sum_i t_i = 1$



(Appendix) Jensen's Inequality

- If $\boldsymbol{x}$ is a random variable and f is a convex function, then

$$f\left(\mathbb{E}_{p(\boldsymbol{x})}[\boldsymbol{x}]\right) \leq \mathbb{E}_{p(\boldsymbol{x})}[f(\boldsymbol{x})]$$

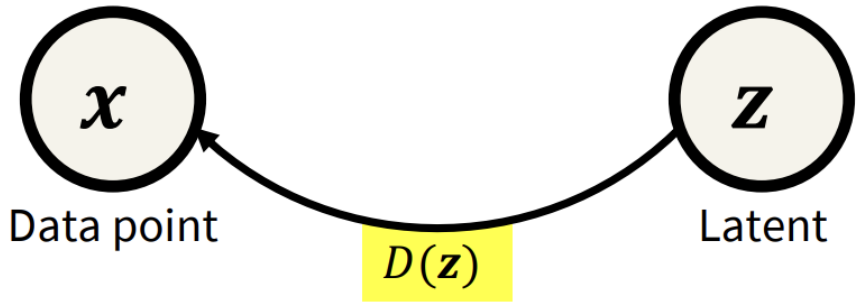
  since the expected value is a convex combination.

- Q) What if f is a concave function, such as $\log$?

Back to Variational Autoencoder (VAE)

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

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



Variational Autoencoder (VAE)

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

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

- How to compute the integral?
  - Monte-Carlo method for $\boldsymbol{x}$ and $\boldsymbol{z}$ takes too much time. → Intractable

Variational Autoencoder (VAE)

- Or, we can compute

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

But this conditional distribution is unknown.

- We cannot directly maximize $p(\boldsymbol{x}) = \frac{p(\boldsymbol{x},\boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x})}$ since $p(\boldsymbol{z}|\boldsymbol{x})$ is unknown.

Variational Autoencoder (VAE)

- Let's think about the lower bound of $\log p(\boldsymbol{x})$:

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

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

Variational Autoencoder (VAE)

- $\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right]$ is the expected value over $\boldsymbol{z}$ sampled from the variational distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$.

- Why $\log p(\boldsymbol{x}) \geq \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x},\boldsymbol{z})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\right]$?
  - Because of the Jensen's inequality.

Evidence Lower Bound (ELBO)

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

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

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

Concave function

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

Evidence Lower Bound (ELBO)

- Let's decompose ELBO:

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

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

$$= \boxed{\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z})]} - \boxed{D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z})\right)}$$

Reconstruction term to be *maximized*.

Prior matching term to be *minimized*.

Evidence Lower Bound (ELBO)

$$\mathcal{L}_{\mathrm{ELBO}} = \mathbb{E}_{q_{\phi}(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z})] - D_{\mathrm{KL}}\big(q_{\phi}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\big)$$

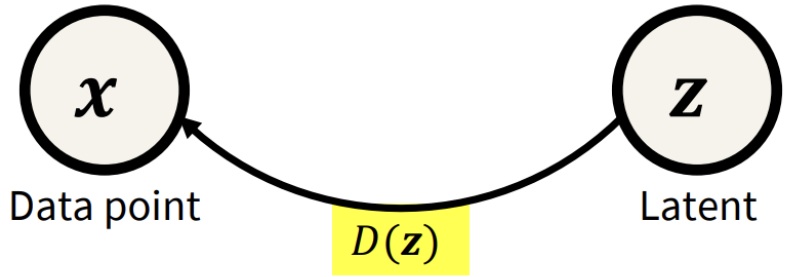
- Reconstruction
  - Accurate recovery of $\boldsymbol{x}$ from its latent code $\boldsymbol{z}$
  - With Gaussian assumptions, it reduces the familiar reconstruction loss of AE.
- Latent KL
  - Encourages $q_{\phi}(\boldsymbol{z}|\boldsymbol{x})$ to stay close to (a simple Gaussian) prior $p(\boldsymbol{z})$.
  - Shapes the latent space into a smooth and continuous structure.

Variational Autoencoder (VAE)

- In VAE, we want model $p(\boldsymbol{x})$ as

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

where $p(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}; D(\boldsymbol{z}), \sigma^2 I)$ and $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}; \mathbf{0}, \mathbf{I})$.

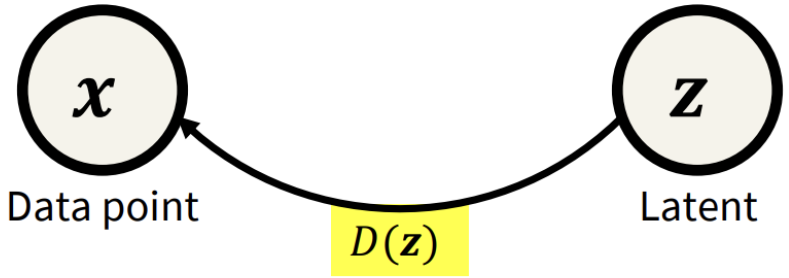


Variational Autoencoder (VAE)

- In VAE, we want model $p(\boldsymbol{x})$ as

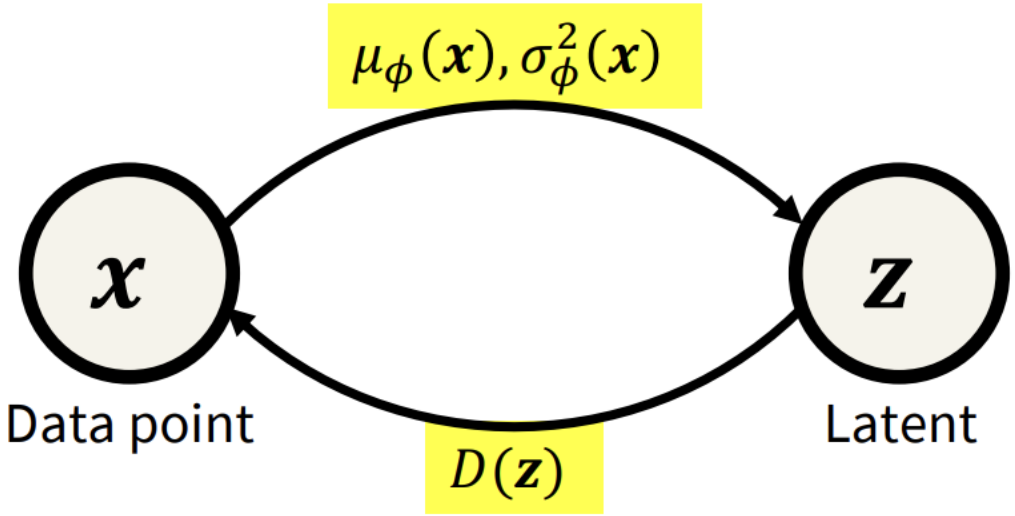
$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z} = \int p_\theta(\boldsymbol{x}|\boldsymbol{z}) p(\boldsymbol{z}) d\boldsymbol{z}$$

where $p_\theta(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x};\ D_\theta(\boldsymbol{z}), \sigma^2 I)$ and $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}; \mathbf{0}, \mathbf{I})$.



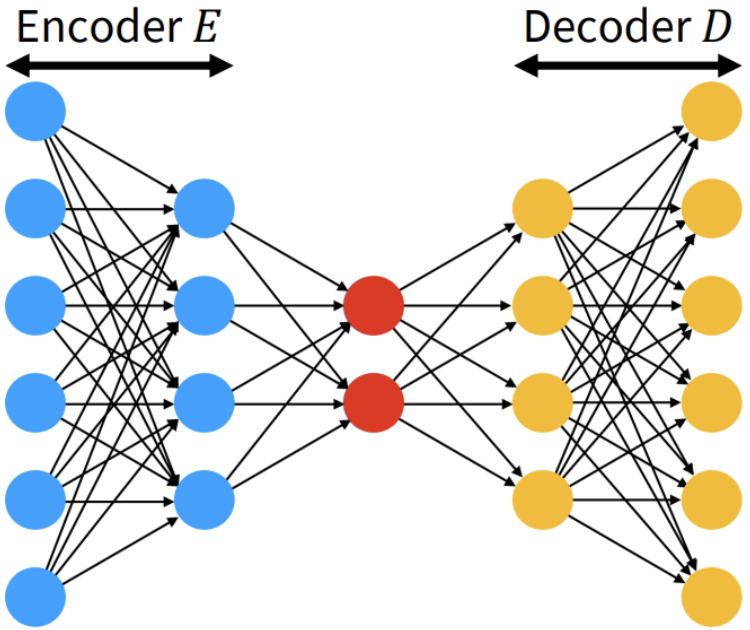
Evidence Lower Bound (ELBO)

- How to maximize $\log p(\boldsymbol{x})$? Maximize ELBO!
- We need the proxy distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ → Encoder
- Let $q_\phi(\boldsymbol{z}|\boldsymbol{x}) = N\left(\boldsymbol{z}; \boldsymbol{\mu}_\phi(\boldsymbol{x}), \sigma_\phi^2(\boldsymbol{x})\mathbf{I}\right)$



Variational Autoencoder (VAE)

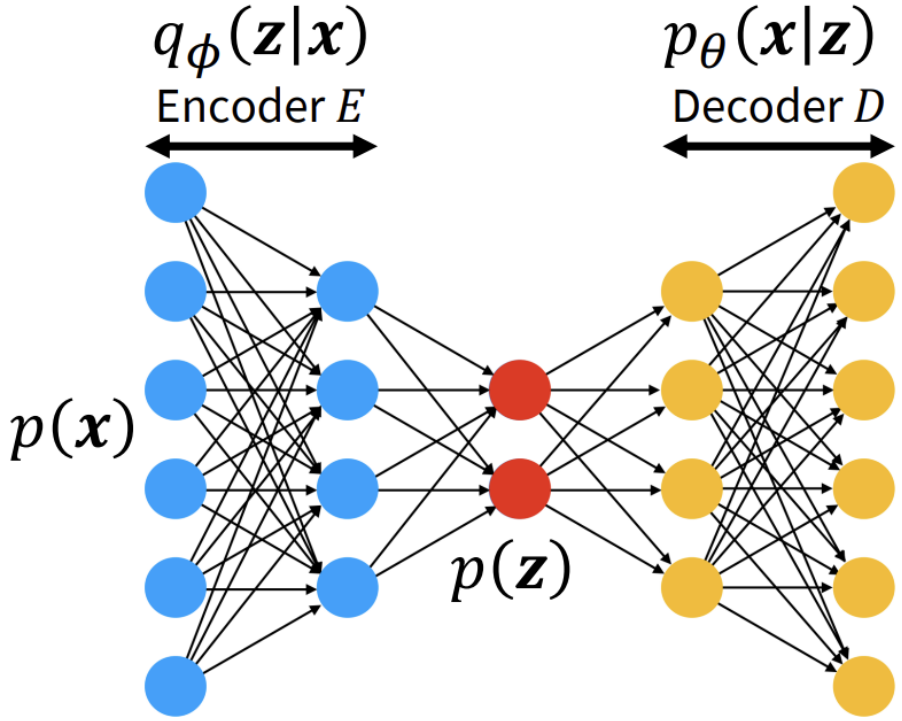
- Same with autoencoder but,
- From input $\boldsymbol{x}$, the encoder predicts $\boldsymbol{\mu}_{\phi}(\boldsymbol{x}), \sigma_{\phi}^{2}(\boldsymbol{x})\mathbf{I}$.
- The decoder takes a sample $\boldsymbol{z}\sim N\left(\boldsymbol{z}; \boldsymbol{\mu}_{\phi}(\boldsymbol{x}), \sigma_{\phi}^{2}(\boldsymbol{x})\mathbf{I}\right)$ as input.



Variational Autoencoder (VAE)

- Summary

Data distribution	$p(\boldsymbol{x})$
Encoder	$q_\phi(\boldsymbol{z}\|\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{z}; \boldsymbol{\mu}_\phi(\boldsymbol{x}), \sigma_\phi^2(\boldsymbol{x})\mathbf{I}\right)$
Latent distribution	$p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}; \mathbf{0}, \mathbf{I})$
Decoder	$p_\theta(\boldsymbol{x}\|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{x}; D_\theta(\boldsymbol{z}), \sigma^2 I)$



VAE Training

- How to maximize ELBO?

$$\underset{\theta,\phi}{\operatorname{argmax}}\, \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - D_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\big)$$

- Approximates using a Monte Carlo estimate:

$$\underset{\theta,\phi}{\operatorname{argmax}} \frac{1}{N}\sum_{i=1}^{N} \log p_\theta(\boldsymbol{x}|\boldsymbol{z}^{(i)}) - D_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\big)$$

where $\boldsymbol{z}^{(i)} \sim q_\phi(\boldsymbol{z}|\boldsymbol{x})$ for the given $\boldsymbol{x}$

VAE Training

- How to maximize ELBO?

$$\underset{\theta,\phi}{\operatorname{argmax}}\, \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - D_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\big)$$

- Approximates using a Monte Carlo estimate:

$$\underset{\theta,\phi}{\operatorname{argmax}} \frac{1}{N}\sum_{i=1}^{N} \log p_\theta(\boldsymbol{x}|\boldsymbol{z}^{(i)}) - D_{\mathrm{KL}}\big(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})\big)$$

Reparameterization trick

where $\boldsymbol{z}^{(i)} = \boldsymbol{\mu}_\phi(\boldsymbol{x}) + \sigma_\phi(\boldsymbol{x})\boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \mathbf{I})$

VAE Training

- Recall $p_\theta(\boldsymbol{x}|\boldsymbol{z}) = N(\boldsymbol{x}; D_\theta(\boldsymbol{z}), \sigma^2 \mathbf{I})$

$$\log p_\theta\left(\boldsymbol{x}|\boldsymbol{z}^{(i)}\right) = \log\left(\frac{1}{\sqrt{(2\pi\sigma^2)^d}} \exp\left(-\frac{\|\boldsymbol{x} - D_\theta(\boldsymbol{z})\|^2}{2\sigma^2}\right)\right)$$

$$= -\frac{1}{2\sigma^2}\|\boldsymbol{x} - D_\theta(\boldsymbol{z})\|^2 - \log\sqrt{(2\pi\sigma^2)^d}$$

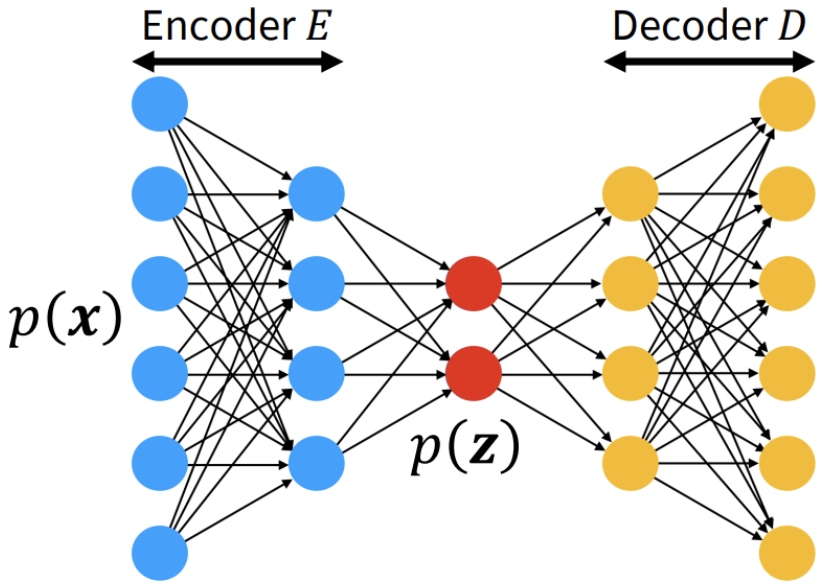
This is why it is called the reconstruction term.

Constant

VAE Training

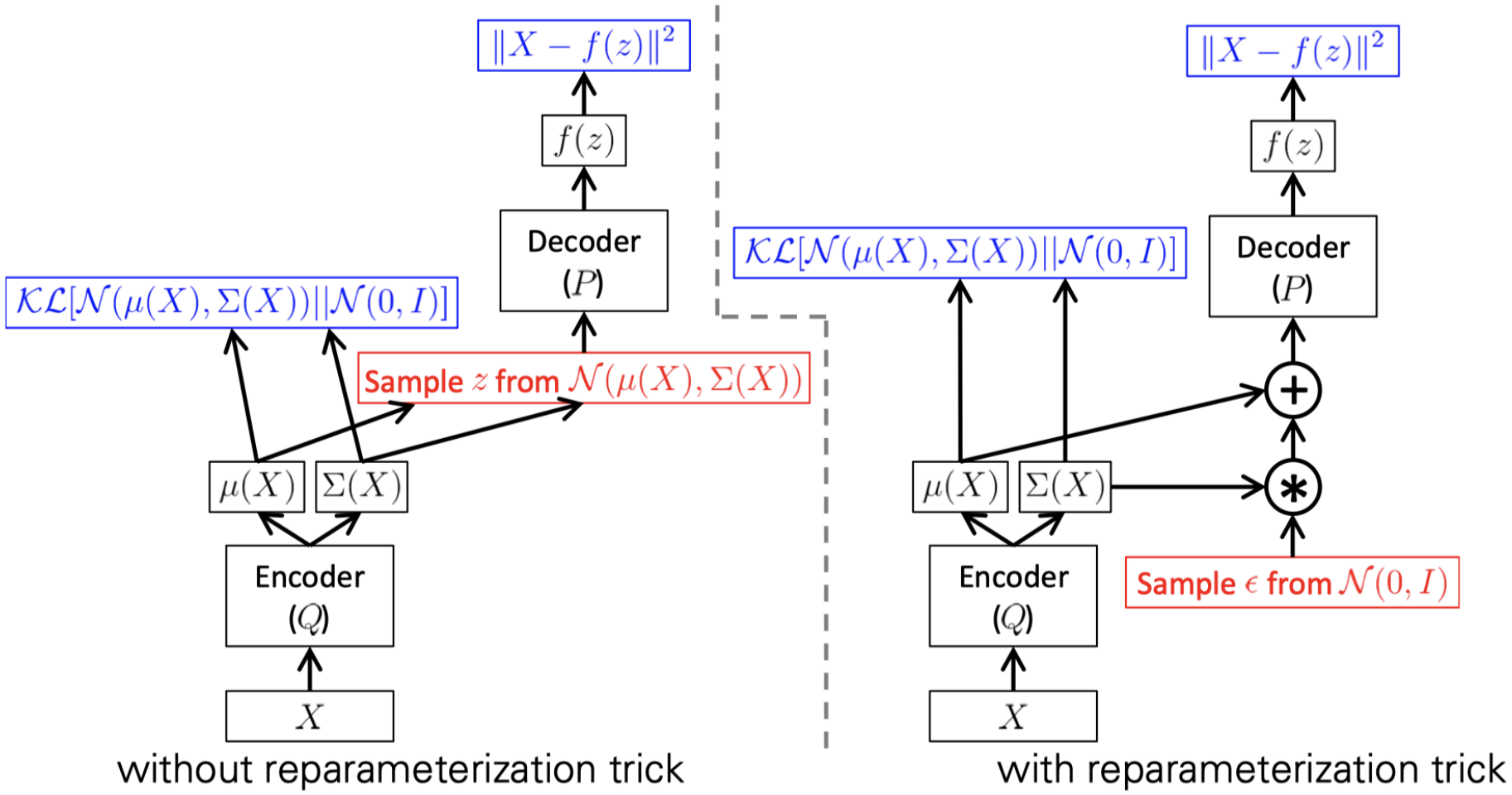
1. Feed a data point $\boldsymbol{x}$ to the encoder to predict and $\boldsymbol{\mu}_{\phi}(\boldsymbol{x})$ and $\sigma_{\phi}^{2}(\boldsymbol{x})$.
2. Sample a latent variable $\boldsymbol{z}$ from $q_{\phi}(\boldsymbol{z}|\boldsymbol{x}) = N\left(\boldsymbol{z}; \boldsymbol{\mu}_{\phi}(\boldsymbol{x}), \sigma_{\phi}^{2}(\boldsymbol{x})\mathbf{I}\right)$.
3. Feed $\boldsymbol{z}$ to the decoder to predict $\hat{\boldsymbol{x}} = D_{\theta}(\boldsymbol{z})$.
4. Perform the gradient descent through the negative ELBO.

Q) Why is the sampling differentiable?



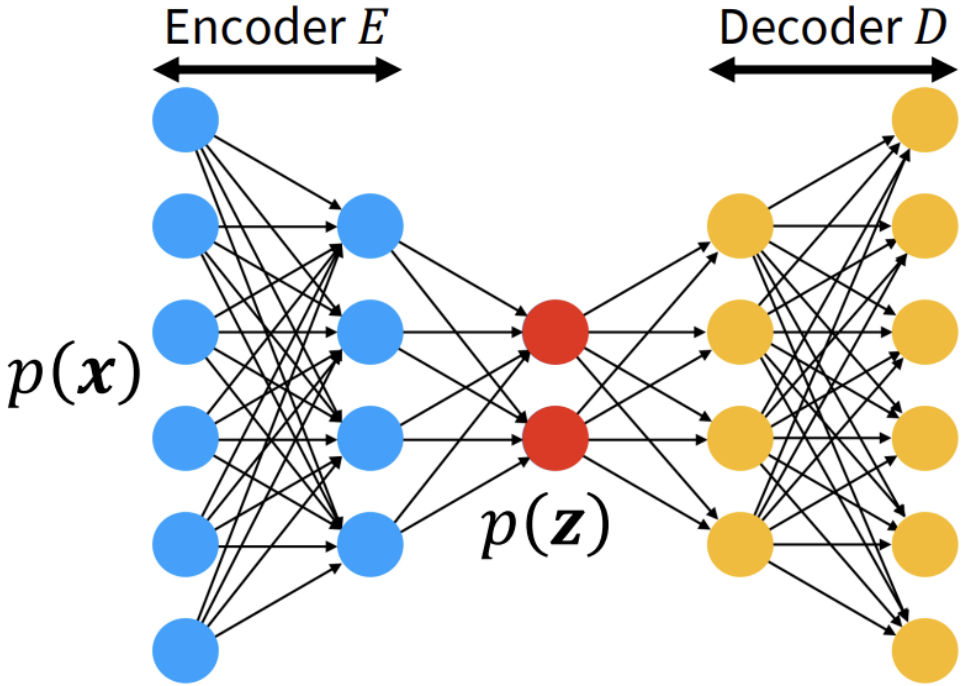
VAE Training

Q) Why is the sampling differentiable?



VAE Generation

1. Sample a latent variable $\boldsymbol{z}$ from $p(\boldsymbol{z}) = N(\boldsymbol{z}; \mathbf{0}, \mathbf{I})$.
2. Feed $\boldsymbol{z}$ to the decoder to predict $\hat{\boldsymbol{x}} = D_\theta(\boldsymbol{z})$.

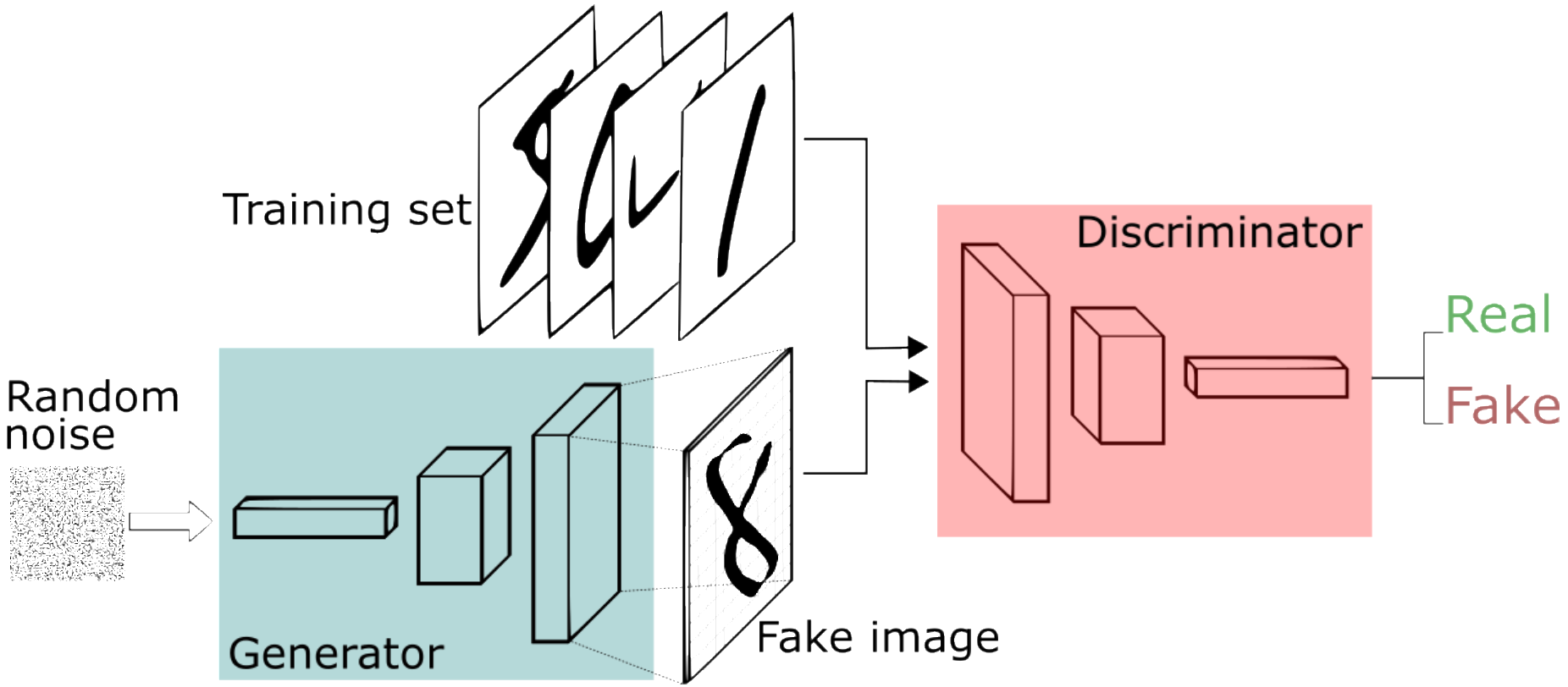


Samples from VAEs

Generative Adversarial Networks

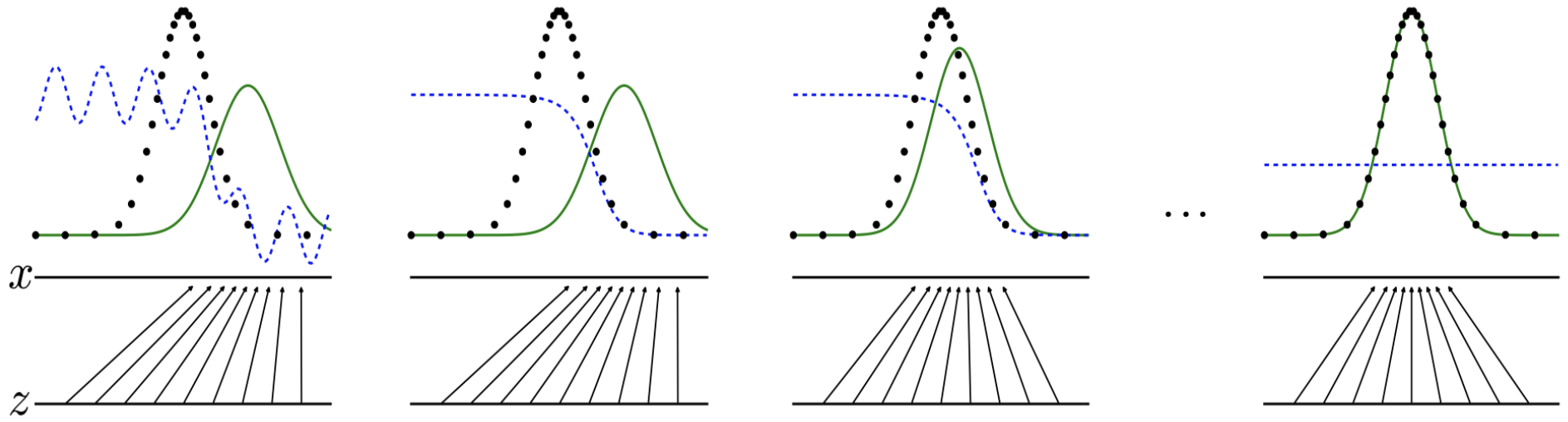
Generative Adversarial Networks

- Two player minimax game



Training Objective of GAN

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



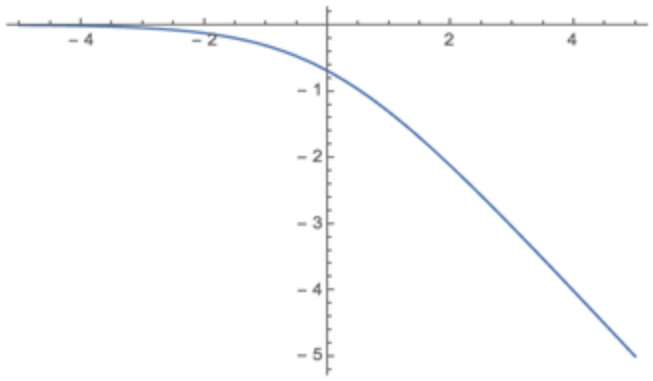
Training GAN

- Heuristics for training

In practice, equation 1 may not provide sufficient gradient for G to learn well. Early in learning, when G is poor, D can reject samples with high confidence because they are clearly different from the training data. In this case, $\log(1 - D(G(\boldsymbol{z})))$ saturates. Rather than training G to minimize $\log(1 - D(G(\boldsymbol{z})))$ we can train G to maximize $\log D(G(\boldsymbol{z}))$. This objective function results in the same fixed point of the dynamics of G and D but provides much stronger gradients early in learning.

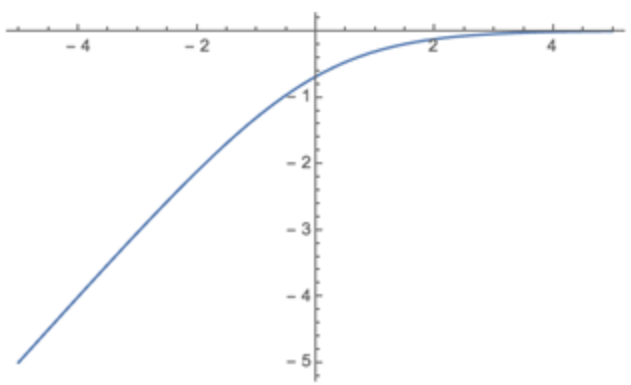
$\log(1 - x)$

```
Plot[Log[1-(1+E^(-x))^(-1)], {x, -5.0, 5.0}]
```



$\log(x)$

```
Plot[Log[(1+E^(-x))^(-1)], {x, -5.0, 5.0}]
```



- Instead of flipping the sign on V(D,G), we flip the target!
- Not minimax game
- Theoretical analysis does not hold

Training GAN

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

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:

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

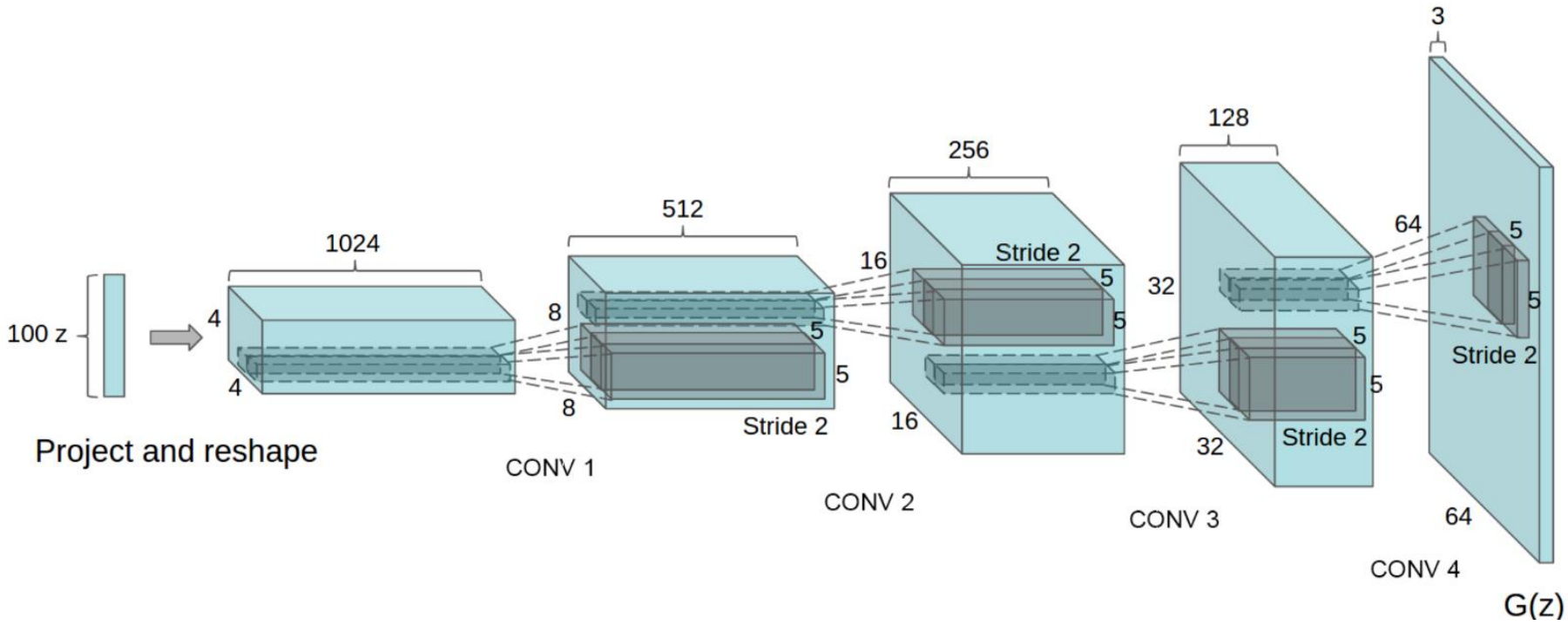
Note that this algorithm does not work in practice, but for the theoretical analysis.

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

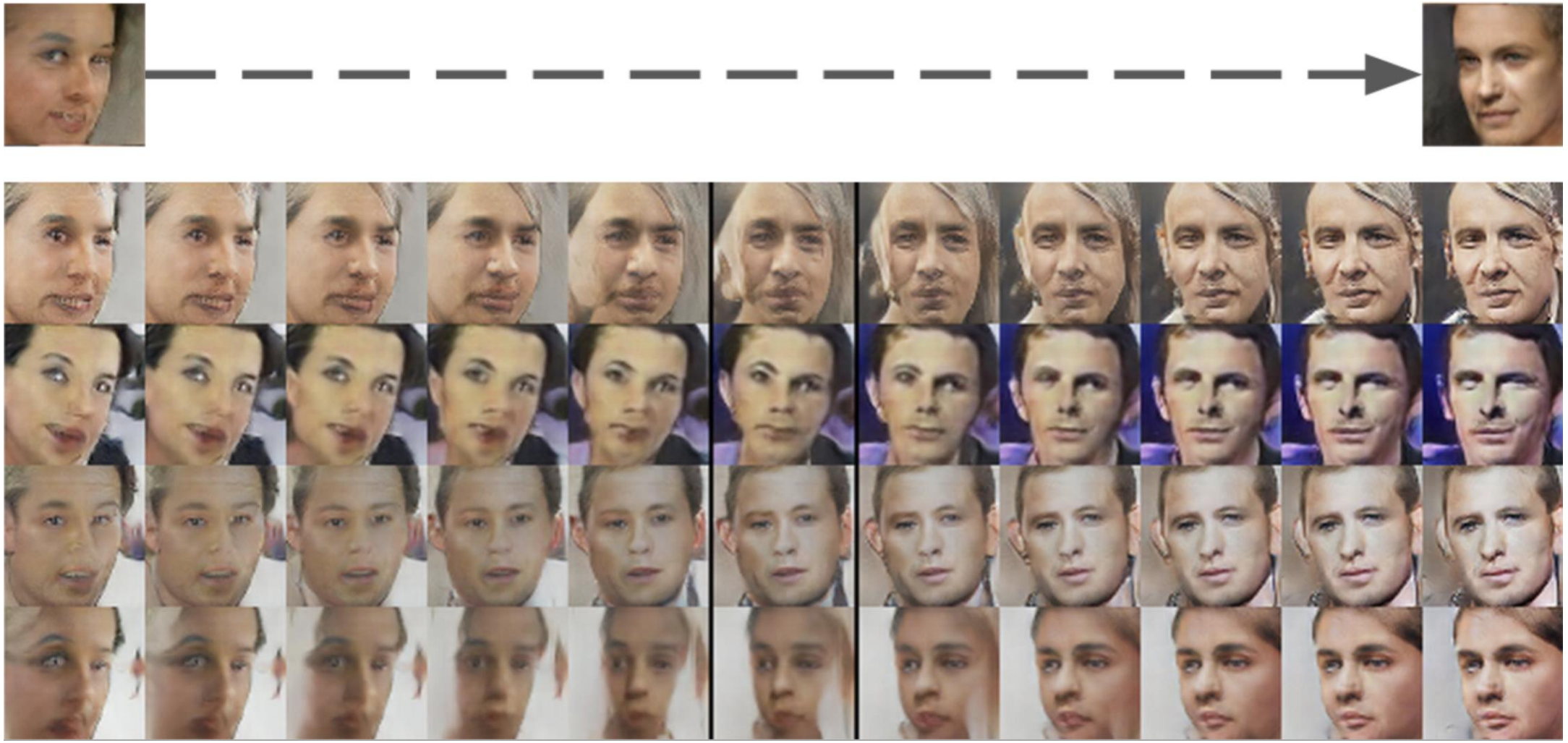
Deep Convolutional GAN

- Deep Convolutional GAN (Radford et al., 2015)
  - introduces some tips to make GAN training more stable
  - fully convolutional layers, leaky ReLU in D, Adam optimizer, batch norm, etc.



Deep Convolutional GAN (Radford et al., 2015)

Deep Convolutional GAN (Radford et al., 2015)



Deep Convolutional GAN (Radford et al., 2015)