

# Introduction to Diffusion Models (October 11, 2023)

Santiago VELASCO-FORERO  
<http://cmm.ensmp.fr/~velasco/>

MINES Paris  
PSL Research University  
Center for Mathematical Morphology



# Contents

## 1 Introduction

## 2 Generative models

- Classical Generative Modeling
- Autoencoders
- Variational Autoencoder

## 3 Generative Adversarial Networks (GANs)

## 4 Variational Diffusion Models

## 5 References

# Contents

1 Introduction

2 Generative models

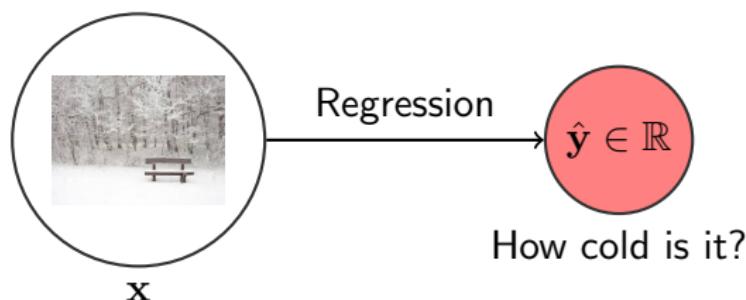
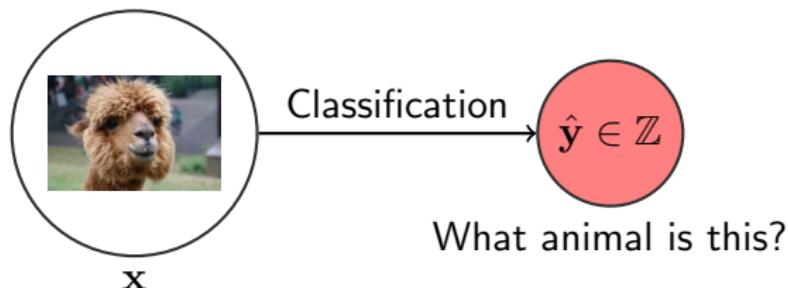
3 Generative Adversarial Networks (GANs)

4 Variational Diffusion Models

5 References

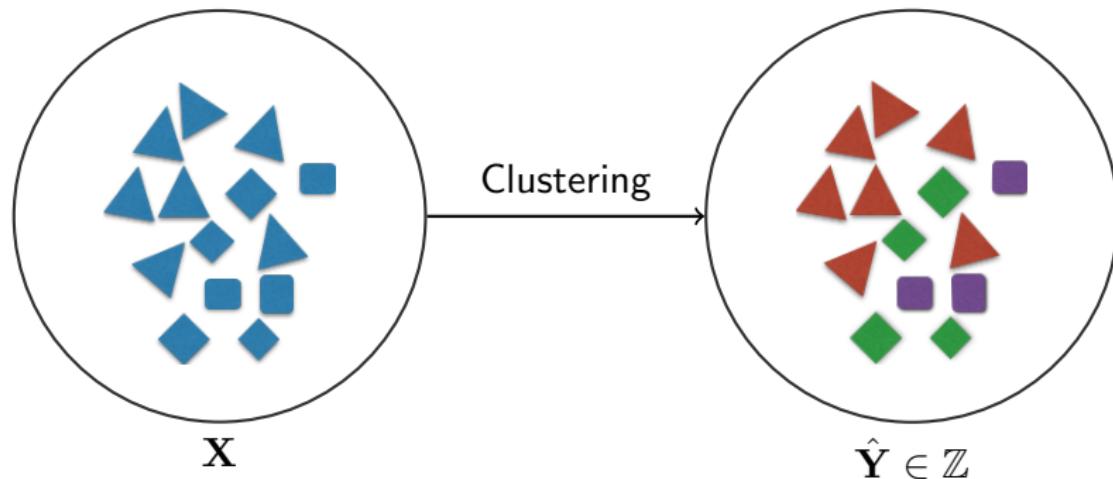
## Supervised Learning

Given a labeled dataset  $(\mathbf{X}, \mathbf{Y})$ , we would like to learn a mapping from data space to label space.



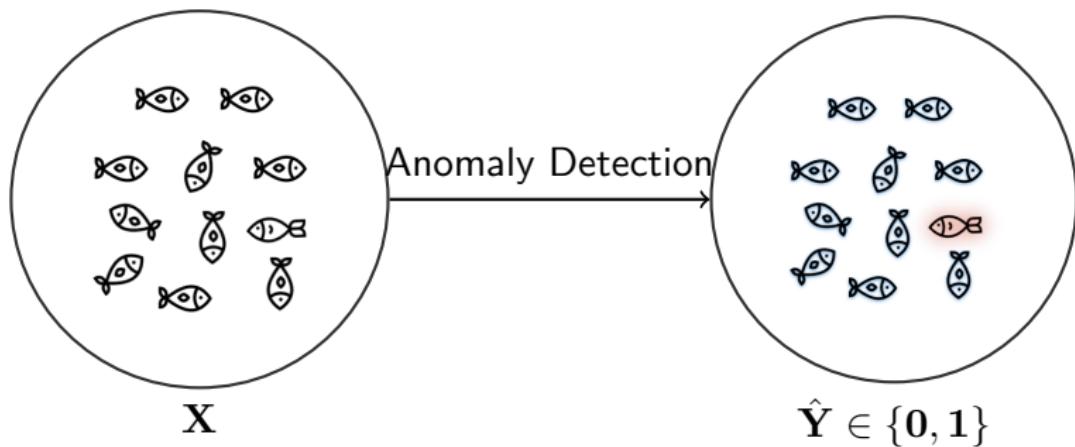
# Unsupervised Learning: Clustering

Given an unlabeled dataset ( $\mathbf{X}$ ), we would like to learn: **How to group objects into categories?**



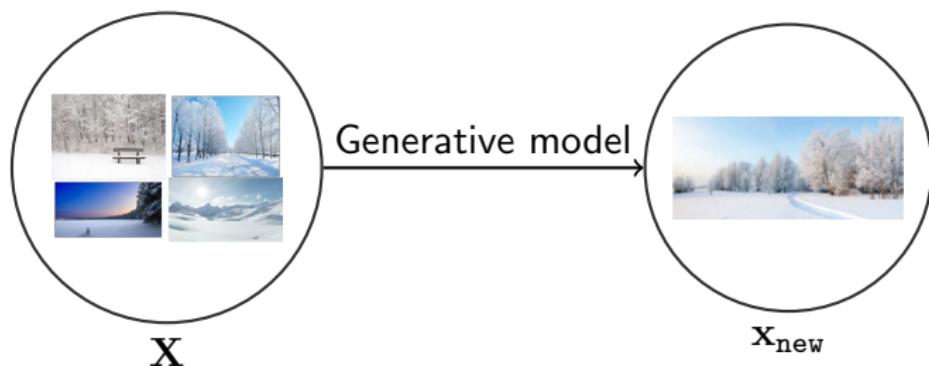
## Anomaly detection

Given an unlabeled dataset ( $\mathbf{X}$ ), we would like to learn: How to identify observations differing significantly from the majority of data?



## Unsupervised learning: Generative Models

Given an unlabeled dataset ( $\mathbf{X}$ ), we would like to learn: How to generate a new observation from the same distribution (unknown) of dataset?



# Contents

## 1 Introduction

## 2 Generative models

- Classical Generative Modeling
- Autoencoders
- Variational Autoencoder

## 3 Generative Adversarial Networks (GANs)

## 4 Variational Diffusion Models

## 5 References

## Goal

Generative models "learn" a joint distribution over a dataset. They are mostly used for sampling applications or density estimation.

- ① (Sampling) A generative model learns to fit a model over observations so one can sample novel data from the model,  
 $\mathbf{x}_{new} \sim p_{\theta}(\mathbf{x})$
- ② (Density estimation) A model learns to estimate the probability of observations. Given a datapoint  $\mathbf{x}$  the model, what is the probability of  $\mathbf{x}$  belong to the dataset?

## Generative models

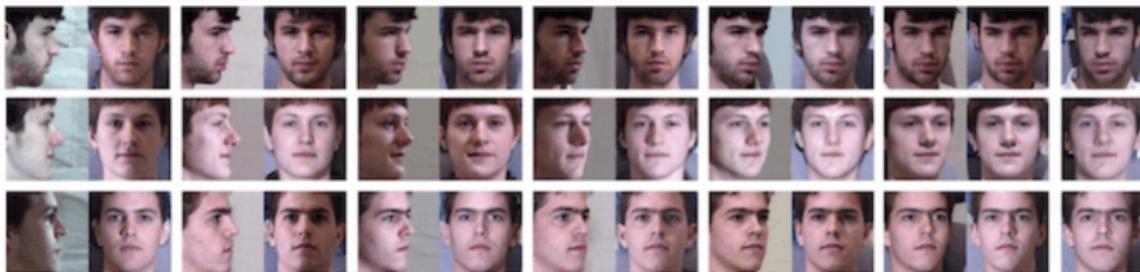
- Given observed samples i.i.d.  $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$  from a distribution of interest (unknown) in  $\mathbb{R}^p$ , the goal of a *generative model* is to learn to model its true data distribution  $p(\mathbf{x})$ .

## Generative models

- Given observed samples i.i.d.  $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$  from a distribution of interest (unknown) in  $\mathbb{R}^p$ , the goal of a *generative model* is to learn to model its true data distribution  $p(\mathbf{x})$ .
- After learning, we can *generate* new samples from our approximate model.
- In most of the applications, one is interested in conditional generative models, for instance, generate an image **from** a text, or generate an image **from** another image, so on.

# Examples of Conditional generation.

## ① Face Frontal View Generation



- ② Photos to Emojis
- ③ Image Editing
- ④ Image De-raining
- ⑤ Text-to-Image
- ⑥ Many others...

# Examples of Conditional generation.

- ① Face Frontal View Generation
- ② Photos to Emojis
- ③ Image Editing
- ④ Image De-raining



- ⑤ Text-to-Image
- ⑥ Many others...

# Contents

## 1 Introduction

## 2 Generative models

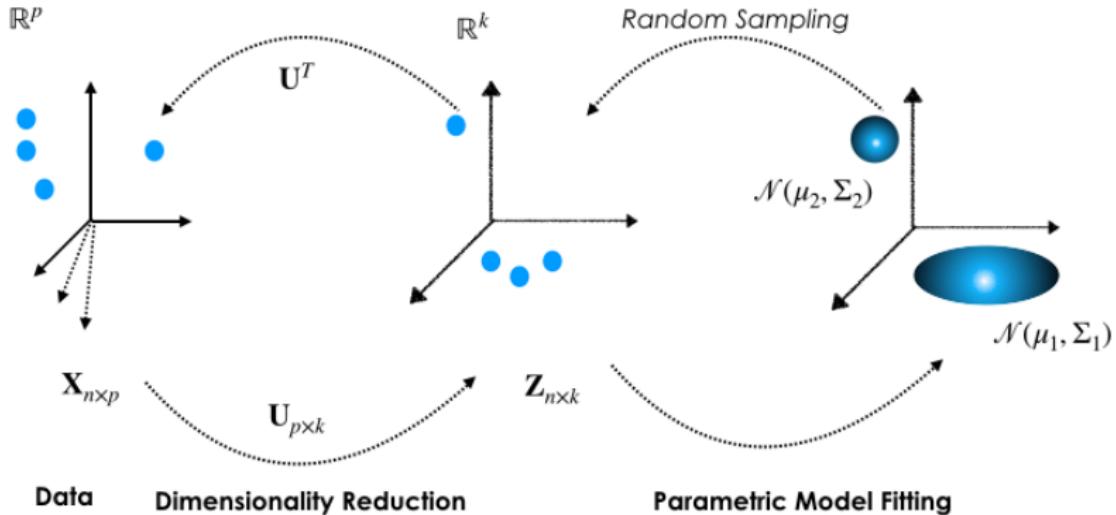
- Classical Generative Modeling
- Autoencoders
- Variational Autoencoder

## 3 Generative Adversarial Networks (GANs)

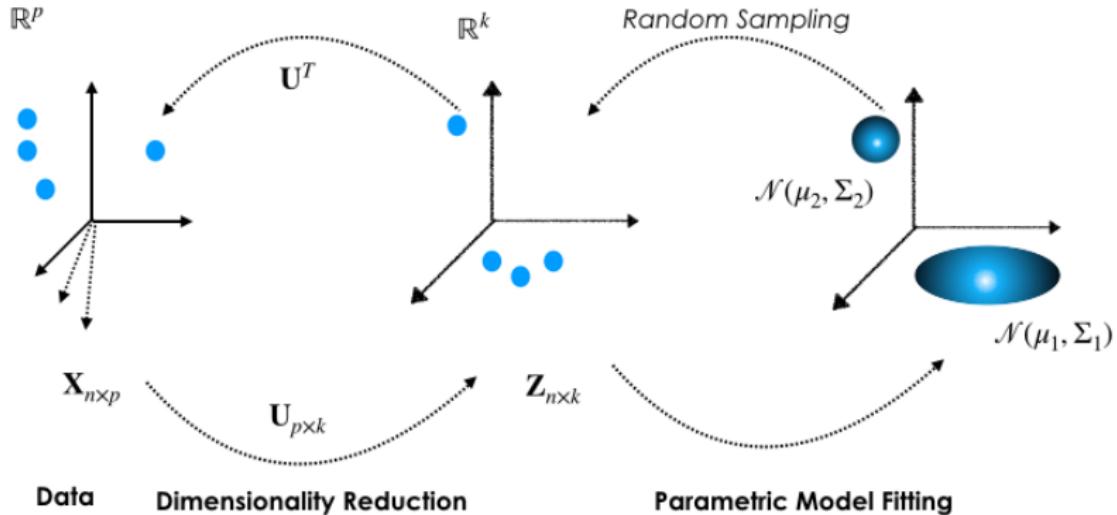
## 4 Variational Diffusion Models

## 5 References

# Classical Generative Modeling

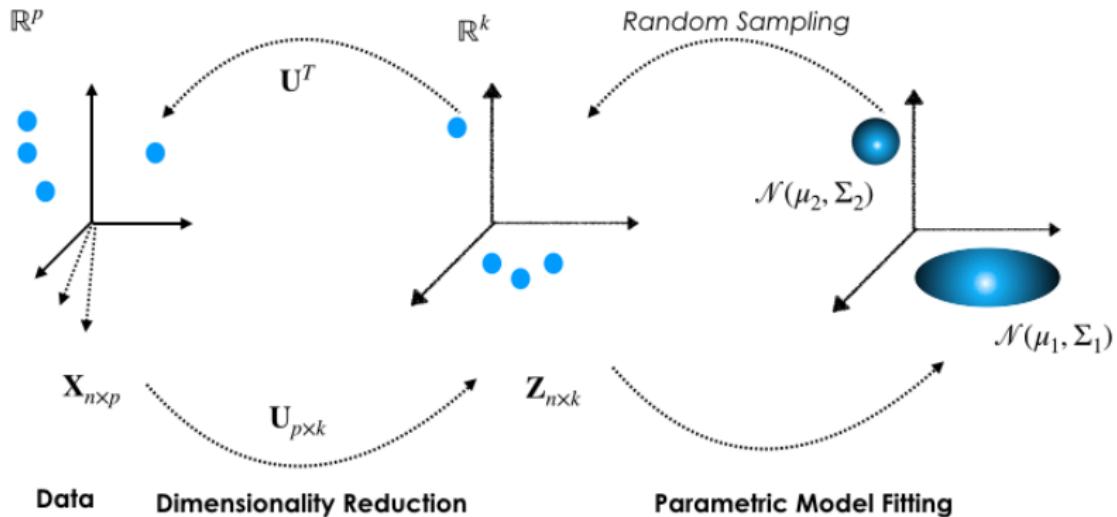


# Classical Generative Modeling



- Dimensionality reduction: Usually linear by maximal variance subspace (Principal Component Analysis)
- Distribution Modeling: Usually Mixed Multivariate Gaussian Distribution

# Classical Generative Modeling



## Limitations:

- Nonlinear relations are difficult to be considered. For instance: Spatial Correlation between Pixels.
- Distribution Modeling: Fitting parametric models in high-dimensions is hard. *Curse of Dimensionality*

## Generative models

- Given observed samples i.i.d.  $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$  from a distribution of interest (unknow) in  $\mathbb{R}^p$ , the goal of a *generative model* is to learn to model its true data distribution  $p(\mathbf{x})$ .
- After learning, we can *generate* new samples from our approximate model.
- Closed form for the model is not required

# Contents

## 1 Introduction

## 2 Generative models

- Classical Generative Modeling
- **Autoencoders**
- Variational Autoencoder

## 3 Generative Adversarial Networks (GANs)

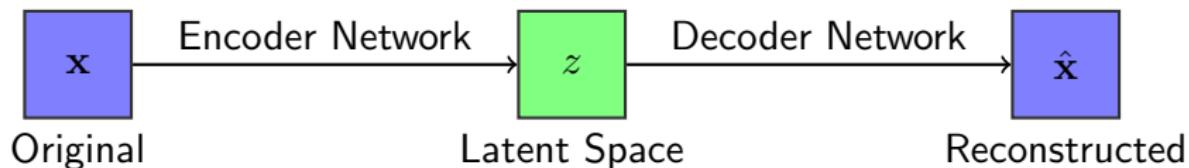
## 4 Variational Diffusion Models

## 5 References

# Autoencoders

Autoencoders are neural networks whose purpose is twofold:

- ① To compress some input data by transforming it from the input domain to another space, known as the *latent space* (code).
- ② To take this latent representation and transform it back to the original space, such that the output is *similar* to the input.



The loss function for a given input vector is usually the reconstruction error:

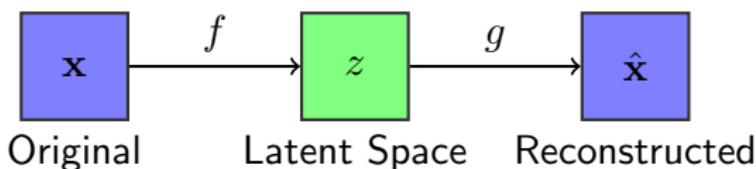
$$L(\mathbf{x}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2$$

# Autoencoder

- An *autoencoder* is a neural network that is trained to attempt to copy its input to its output.
- The network may be viewed as consisting of two parts: an encoder function  $z = f(\mathbf{x})$  and a decoder that produces a reconstruction  $\hat{\mathbf{x}} = g(f(\mathbf{x}))$ .
- The composition of  $f$  and  $g$  is called the *reconstruction function*
- If an autoencoder succeeds to learn  $g(f(\mathbf{x})) = \mathbf{x}$  everywhere, then it is not especially useful (overfitting).
- The learning process consists in minimizing the loss function:

$$L(\mathbf{x}, g(f(\mathbf{x}))), \quad (1)$$

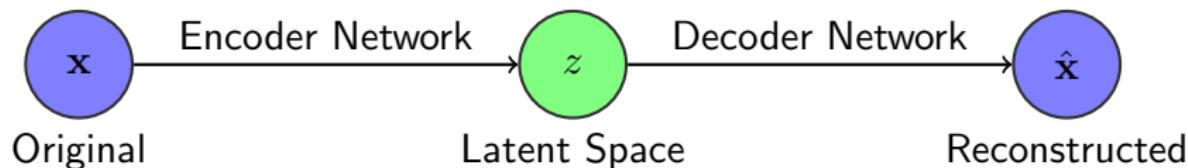
where  $L$  is a loss function, such as mean squared error.



# Autoencoders

Autoencoders are neural networks whose purpose is twofold:

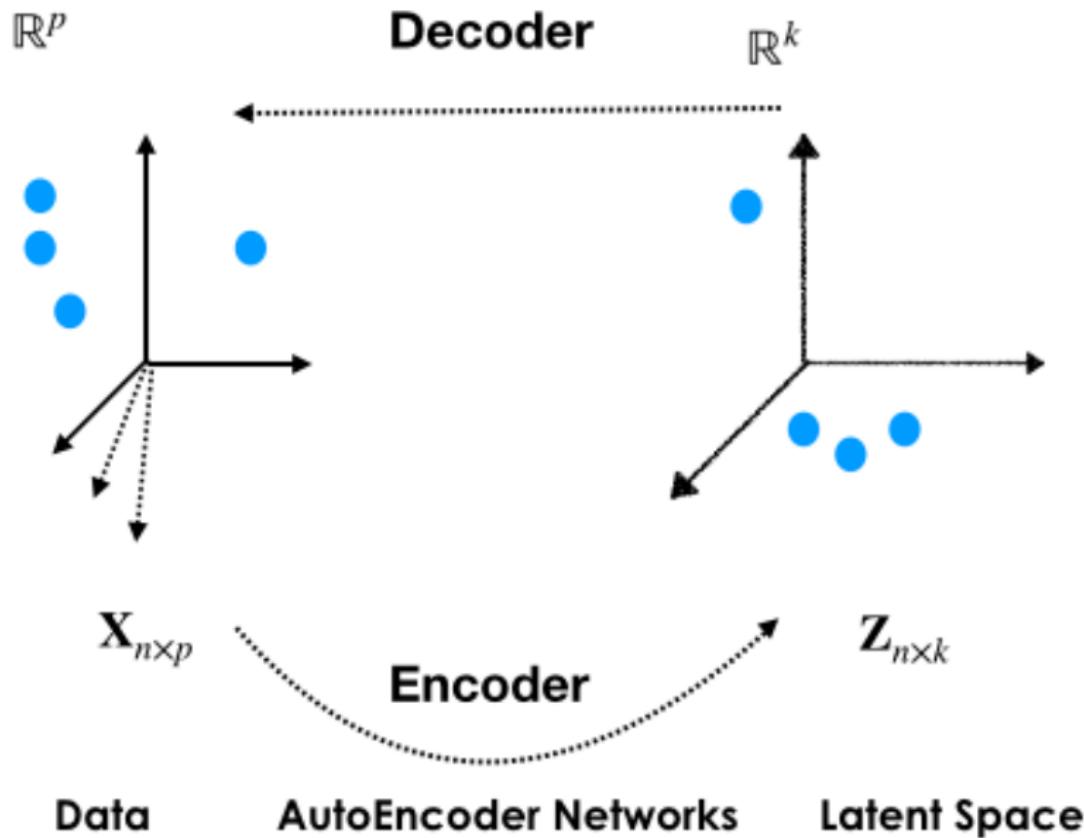
- ① To compress some input data by transforming it from the input domain to another space, known as the *latent space* (code).
- ② To take this latent representation and transform it back to the original space, such that the output is *similar* to the input.



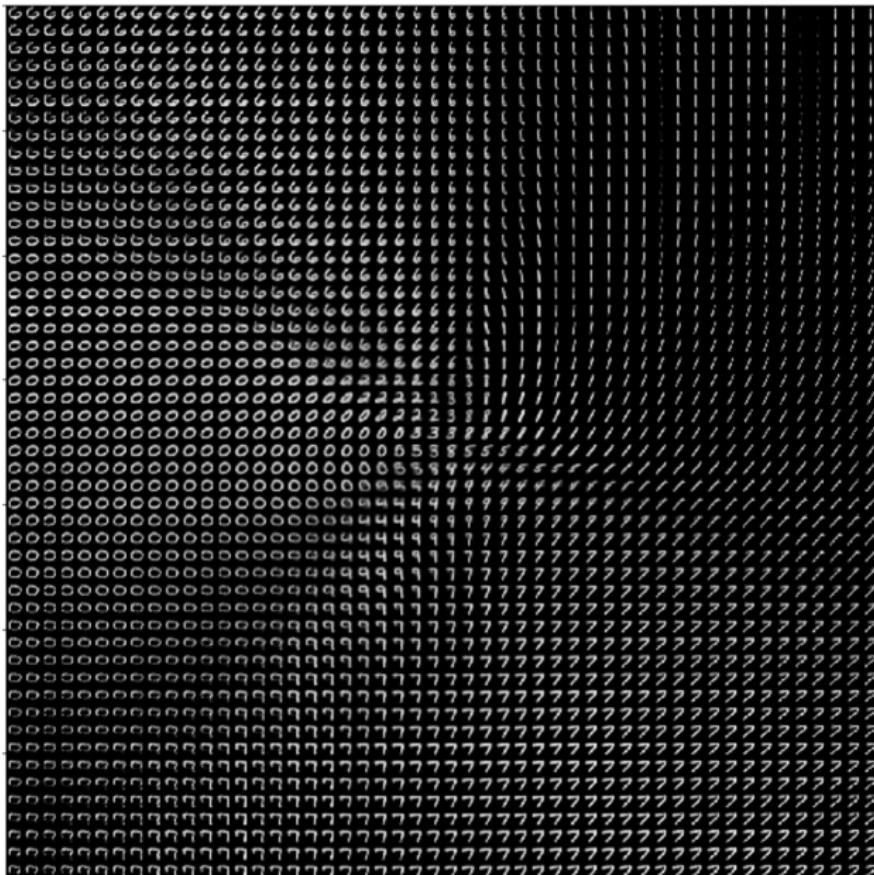
The loss function for a given input vector is usually the reconstruction error:

$$L(\mathbf{x}) = \|\mathbf{x} - \hat{\mathbf{x}}\|^2$$

## AutoEncoder Scheme



## Example: $k = 2$ , latent space for MNIST dataset



# Contents

## 1 Introduction

## 2 Generative models

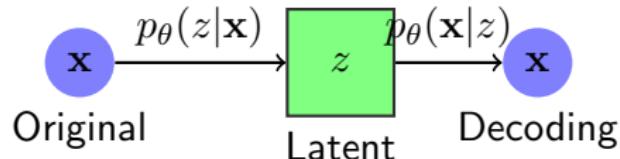
- Classical Generative Modeling
- Autoencoders
- Variational Autoencoder

## 3 Generative Adversarial Networks (GANs)

## 4 Variational Diffusion Models

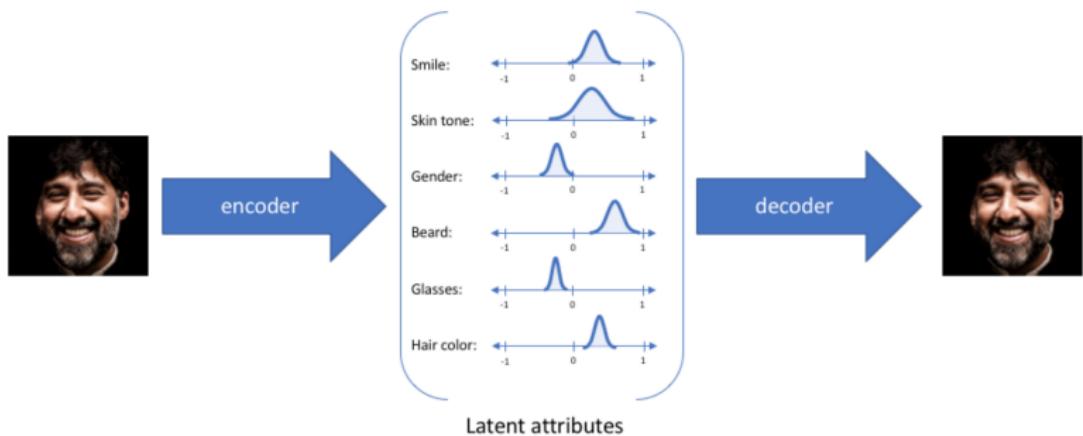
## 5 References

# Variational Autoencoder



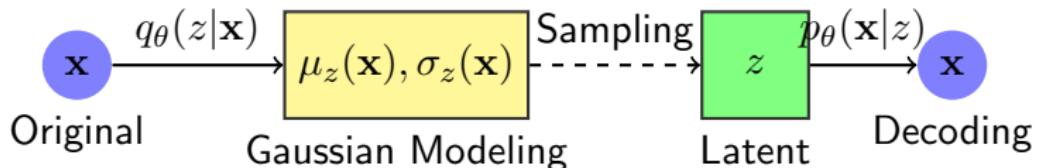
- Training via maximum likelihood of  $p(\mathbf{x})$
- Intractability: the true posterior density  $p_\theta(z|\mathbf{x})$  cannot be calculated in polynomial time
- Solution: Approximate  $p_\theta(z|\mathbf{x})$  by means of  $q_\theta(z|\mathbf{x}) = \mathcal{N}(z; \mu_z(\mathbf{x}), \sigma_z(\mathbf{x}))$

# Latent space intuition (variational case)



Credits:  
<https://www.jeremyjordan.me/variational-autoencoders/>

# Variational Autoencoder



- Gaussian modeling
- Training via maximum likelihood of  $p(\mathbf{x})$
- Learning the parameters  $\theta$  via backpropagation?

## Training via maximum likelihood

Assume we would like to compute the likelihood of an image  $\mathbf{x}$  from the training set:

$$\begin{aligned}\mathcal{L}(\mathbf{x}) &= \log(p(\mathbf{x})) \\ &= \sum_z q(z|\mathbf{x}) \log(p(\mathbf{x})) \\ &= \sum_z q(z|\mathbf{x}) \log\left(\frac{p(z, \mathbf{x})}{p(z|\mathbf{x})}\right) \\ &= \sum_z q(z|\mathbf{x}) \log\left(\frac{p(z, \mathbf{x})q(z|\mathbf{x})}{q(z|\mathbf{x})p(z|\mathbf{x})}\right) \\ &= \sum_z q(z|\mathbf{x}) \log\left(\frac{p(z, \mathbf{x})}{q(z|\mathbf{x})}\right) + \sum_z q(z|\mathbf{x}) \log\left(\frac{q(z|\mathbf{x})}{p(z|\mathbf{x})}\right) \\ &= \sum_z q(z|\mathbf{x}) \log\left(\frac{p(z, \mathbf{x})}{q(z|\mathbf{x})}\right) + \underbrace{D_{KL}(q(z|\mathbf{x}), p(z|\mathbf{x}))}_{\geq 0}\end{aligned}$$

## Kullback-Leibler Divergence

They introduced the concept of the KL divergence in 1951.

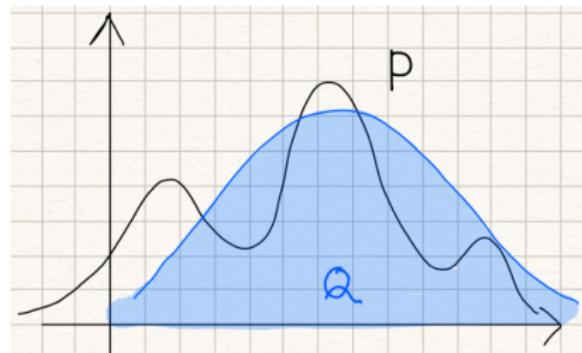
$$D_{KL}(p||q) = \int p(x) \log \frac{p(x)}{q(x)} dx \quad (2)$$

# Kullback-Leibler Divergence

They introduced the concept of the KL divergence in 1951.

$$D_{KL}(p||q) = \int p(x) \log \frac{p(x)}{q(x)} dx \quad (2)$$

- KL divergence is non-negative
- KL divergence is asymmetric
- $D_{KL}(p||q) = \mathbb{E}_{x \sim p(x)} \log \frac{p(x)}{q(x)}$



## Training via maximum likelihood

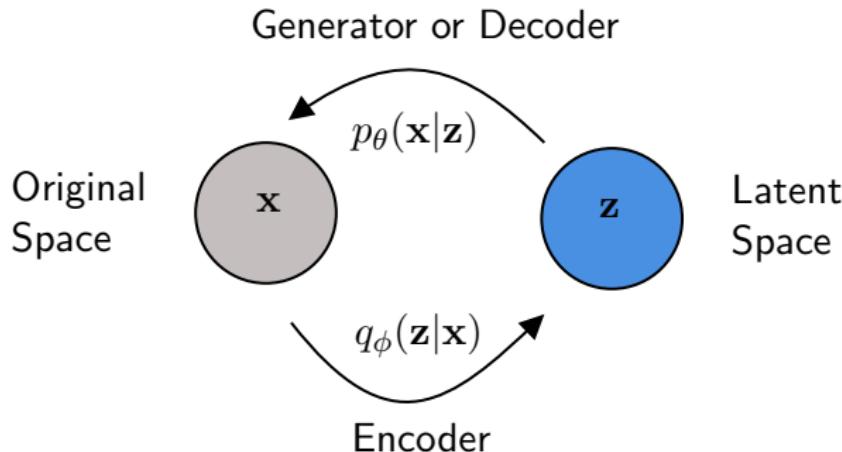
Assume we would like to compute the likelihood of an image  $\mathbf{x}$  from the training set:

$$\begin{aligned}\mathcal{L}(\mathbf{x}) &= \log(p(\mathbf{x})) \\ &= \sum_z q(z|\mathbf{x}) \log(p(\mathbf{x})) \\ &= \sum_z q(z|\mathbf{x}) \log\left(\frac{p(z, \mathbf{x})}{p(z|\mathbf{x})}\right) \\ &= \sum_z q(z|\mathbf{x}) \log\left(\frac{p(z, \mathbf{x})q(z|\mathbf{x})}{q(z|\mathbf{x})p(z|\mathbf{x})}\right) \\ &= \sum_z q(z|\mathbf{x}) \log\left(\frac{p(z, \mathbf{x})}{q(z|\mathbf{x})}\right) + \sum_z q(z|\mathbf{x}) \log\left(\frac{q(z|\mathbf{x})}{p(z|\mathbf{x})}\right) \\ &= \sum_z q(z|\mathbf{x}) \log\left(\frac{p(z, \mathbf{x})}{q(z|\mathbf{x})}\right) + \underbrace{D_{KL}(q(z|\mathbf{x}), p(z|\mathbf{x}))}_{\geq 0} \\ &= \mathcal{L}^{lvb}(\mathbf{x}) + D_{KL}(q(z|\mathbf{x}), p(z|\mathbf{x})) \\ &\geq \mathcal{L}^{lvb}(\mathbf{x})\end{aligned}$$

$$\begin{aligned}
\mathcal{L}(\mathbf{x}) \geq \mathcal{L}^{lvb}(\mathbf{x}) &= \sum_z q(z|\mathbf{x}) \log \left( \frac{p(z, \mathbf{x})}{q(z|\mathbf{x})} \right) \\
&= \sum_z q(z|\mathbf{x}) \log \left( \frac{p(\mathbf{x}|z)p(z)}{q(z|\mathbf{x})} \right) \\
&= \sum_z q(z|\mathbf{x}) \log(p(\mathbf{x}|z)) + \sum_z q(z|\mathbf{x}) \log \left( \frac{p(z)}{q(z|\mathbf{x})} \right) \\
&= \mathbb{E}_{q(z|\mathbf{x})} \log(p(\mathbf{x}|z)) - D_{KL}(q(z|\mathbf{x}), p(z)) \\
&= \underbrace{\mathbb{E}_{q(z|\mathbf{x})} \log(p(\mathbf{x}|z))}_{\text{Expected Reconstruction}} - \underbrace{D_{KL}(q(z|\mathbf{x}), p(z))}_{\text{Regularization } \mathcal{N}(0, 1)}
\end{aligned}$$

- First term implies the use of many realization of sampling process (in practice we only have a few of samples per training example!)
- Second term is simply a formula for diagonal multivariate Gaussian distribution.

## Variational Framework



- The  $q(z|x)$  defines a distribution over the latent space  $z$  for the observations  $x$
- The  $p(x|z)$  defines a distribution over the original by decoding latent variable into the observations space.
- $\phi, \theta$  denote parameters to be learned.
- The generation is performed by taking a point in the latent space and applying the decoder.

## Latent variable intuition

We can imagine the latent variables and data we observed as modelled by joint distribution  $p(\mathbf{x}, \mathbf{z})$

One approach of generative modelling, called *likelihood-based*, is to learn a model to maximise the likelihood  $p(\mathbf{x})$  of all observed  $\mathbf{x}$

- ① Marginalize:  $p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z}$
- ② Chain rule of probability:  $p(\mathbf{x}) = \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{z}|\mathbf{x})}$

## Latent variable intuition

We can imagine the latent variables and data we observed as modelled by joint distribution  $p(\mathbf{x}, \mathbf{z})$

One approach of generative modelling, called *likelihood-based*, is to learn a model to maximise the likelihood  $p(\mathbf{x})$  of all observed  $\mathbf{x}$

① Marginalize:  $p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z}$

② Chain rule of probability:  $p(\mathbf{x}) = \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{z}|\mathbf{x})}$

The main difficulties of:

- (1) is that it involves integrating out all latent variables  $\mathbf{z}$ , which is intractable for complex models.
- (2) is that it involves having access to  $p(\mathbf{z}|\mathbf{x})$ , the truth latent encoder.

## Latent variable intuition

We can imagine the latent variables and data we observed as modelled by joint distribution  $p(\mathbf{x}, \mathbf{z})$

One approach of generative modelling, called *likelihood-based*, is to learn a model to maximise the likelihood  $p(\mathbf{x})$  of all observed  $\mathbf{x}$

① Marginalize:  $p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z}$

② Chain rule of probability:  $p(\mathbf{x}) = \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{z}|\mathbf{x})}$

The main difficulties of:

(1) is that it involves integrating out all latent variables  $\mathbf{z}$ , which is intractable for complex models.

(2) is that it involves having access to  $p(\mathbf{z}|\mathbf{x})$ , the truth latent encoder. **Solution: Find a lower bound of the log likelihood of the observed data (called the *Evidence Lower Bound*, ELBO)**

## ELBO derivation

$$\log p(\mathbf{x}) = \log p(\mathbf{x})$$

## ELBO derivation

$$\begin{aligned}\log p(\mathbf{x}) &= \log p(\mathbf{x}) \\ &= \log p(\mathbf{x}) \int q_{\phi}(\mathbf{z}|\mathbf{x}) d\mathbf{z} \quad (\text{Multiplying by 1})\end{aligned}$$

## ELBO derivation

$$\begin{aligned}\log p(\mathbf{x}) &= \log p(\mathbf{x}) \\ &= \log p(\mathbf{x}) \int q_{\phi}(\mathbf{z}|\mathbf{x}) d\mathbf{z} \quad (\text{Multiplying by 1}) \\ &= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})}(\log p(\mathbf{x})) \quad (\text{Def. Expected value})\end{aligned}$$

## ELBO derivation

$$\begin{aligned}\log p(\mathbf{x}) &= \log p(\mathbf{x}) \\&= \log p(\mathbf{x}) \int q_{\phi}(\mathbf{z}|\mathbf{x}) d\mathbf{z} \quad (\text{Multiplying by 1}) \\&= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})}(\log p(\mathbf{x})) \quad (\text{Def. Expected value}) \\&= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left( \log \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{z}|\mathbf{x})} \right) \quad (\text{Chain Rule of probability})\end{aligned}$$

## ELBO derivation

$$\begin{aligned}\log p(\mathbf{x}) &= \log p(\mathbf{x}) \\&= \log p(\mathbf{x}) \int q_{\phi}(\mathbf{z}|\mathbf{x}) d\mathbf{z} \quad (\text{Multiplying by 1}) \\&= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})}(\log p(\mathbf{x})) \quad (\text{Def. Expected value}) \\&= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left( \log \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{z}|\mathbf{x})} \right) \quad (\text{Chain Rule of probability}) \\&= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left( \log \frac{p(\mathbf{x}, \mathbf{z})q_{\phi}(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})q_{\phi}(\mathbf{z}|\mathbf{x})} \right)\end{aligned}$$

## ELBO derivation

$$\begin{aligned}\log p(\mathbf{x}) &= \log p(\mathbf{x}) \\&= \log p(\mathbf{x}) \int q_{\phi}(\mathbf{z}|\mathbf{x}) d\mathbf{z} \quad (\text{Multiplying by 1}) \\&= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})}(\log p(\mathbf{x})) \quad (\text{Def. Expected value}) \\&= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left( \log \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{z}|\mathbf{x})} \right) \quad (\text{Chain Rule of probability}) \\&= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left( \log \frac{p(\mathbf{x}, \mathbf{z})q_{\phi}(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})q_{\phi}(\mathbf{z}|\mathbf{x})} \right) \\&= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left( \log \frac{p(\mathbf{x}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \right) + \underbrace{\mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left( \log \frac{q_{\phi}(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})} \right)}_{\text{KL Divergence} \geq 0}\end{aligned}$$

## ELBO derivation

$$\begin{aligned}\log p(\mathbf{x}) &= \log p(\mathbf{x}) \\&= \log p(\mathbf{x}) \int q_{\phi}(\mathbf{z}|\mathbf{x}) d\mathbf{z} \quad (\text{Multiplying by 1}) \\&= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})}(\log p(\mathbf{x})) \quad (\text{Def. Expected value}) \\&= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left( \log \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{z}|\mathbf{x})} \right) \quad (\text{Chain Rule of probability}) \\&= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left( \log \frac{p(\mathbf{x}, \mathbf{z})q_{\phi}(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})q_{\phi}(\mathbf{z}|\mathbf{x})} \right) \\&= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left( \log \frac{p(\mathbf{x}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \right) + \underbrace{\mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left( \log \frac{q_{\phi}(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})} \right)}_{\text{KL Divergence} \geq 0}\end{aligned}$$

The KL divergence between the approximate posterior  $q_{\phi}(\mathbf{z}|\mathbf{x})$  and the true posterior  $p(\mathbf{z}|\mathbf{x})$ .

## ELBO derivation

$$\begin{aligned}\log p(\mathbf{x}) &= \log p(\mathbf{x}) \\&= \log p(\mathbf{x}) \int q_{\phi(\mathbf{z}|\mathbf{x})} d\mathbf{z} \quad (\text{Multiplying by 1}) \\&= \mathbb{E}_{q_{\phi(\mathbf{z}|\mathbf{x})}}(\log p(\mathbf{x})) \quad (\text{Def. Expected value}) \\&= \mathbb{E}_{q_{\phi(\mathbf{z}|\mathbf{x})}} \left( \log \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{z}|\mathbf{x})} \right) \quad (\text{Chain Rule of probability}) \\&= \mathbb{E}_{q_{\phi(\mathbf{z}|\mathbf{x})}} \left( \log \frac{p(\mathbf{x}, \mathbf{z})q_{\phi}(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})q_{\phi}(\mathbf{z}|\mathbf{x})} \right) \\&= \mathbb{E}_{q_{\phi(\mathbf{z}|\mathbf{x})}} \left( \log \frac{p(\mathbf{x}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \right) + \underbrace{\mathbb{E}_{q_{\phi(\mathbf{z}|\mathbf{x})}} \left( \log \frac{q_{\phi}(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})} \right)}_{\text{KL Divergence} \geq 0}\end{aligned}$$

## ELBO derivation

$$\begin{aligned}\log p(\mathbf{x}) &= \log p(\mathbf{x}) \\&= \log p(\mathbf{x}) \int q_{\phi}(\mathbf{z}|\mathbf{x}) d\mathbf{z} \quad (\text{Multiplying by 1}) \\&= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})}(\log p(\mathbf{x})) \quad (\text{Def. Expected value}) \\&= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left( \log \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{z}|\mathbf{x})} \right) \quad (\text{Chain Rule of probability}) \\&= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left( \log \frac{p(\mathbf{x}, \mathbf{z})q_{\phi}(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})q_{\phi}(\mathbf{z}|\mathbf{x})} \right) \\&= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left( \log \frac{p(\mathbf{x}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \right) + \underbrace{\mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left( \log \frac{q_{\phi}(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})} \right)}_{\text{KL Divergence} \geq 0} \\&\geq \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} \left( \log \frac{p(\mathbf{x}, \mathbf{z})}{q_{\phi}(\mathbf{z}|\mathbf{x})} \right)\end{aligned}$$

The KL divergence between the approximate posterior  $q_{\phi}(\mathbf{z}|\mathbf{x})$  and the true posterior  $p(\mathbf{z}|\mathbf{x})$ .

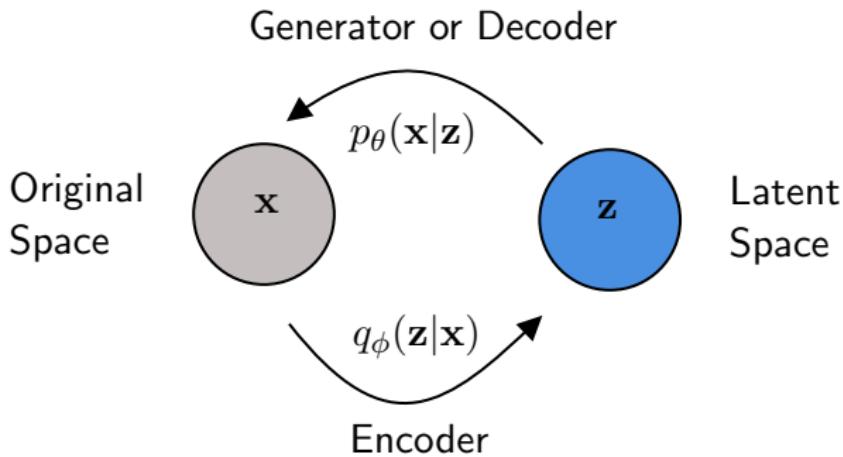
## Evidence Lower Bound (ELBO)

The goal is maximise the

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

$$\begin{aligned}\log p(\mathbf{x}) &\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left( \log \frac{p(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right) \\ &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left( \log \frac{p_{\theta}(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right) \\ &= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_{\theta}(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction term}} - \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left( \log \frac{p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right)}_{\text{KL Divergence} \geq 0}\end{aligned}$$

The KL divergence between the approximate posterior  $q_\phi(\mathbf{z}|\mathbf{x})$  and a prior term  $p(\mathbf{z})$ , usually considered as Gaussian Distribution.



## Interpretation

- ① The first term measures the reconstruction likelihood of the decoder. Ensure that the learned distribution models effectively the problem.
- ② The second term gives a prior belief of variables on latent space. Minimising this term encourages the encoder to learn the prior distribution instead of collapsing into a point (Dirac delta function).

# Contents

## 1 Introduction

## 2 Generative models

- Classical Generative Modeling
- Autoencoders
- Variational Autoencoder

## 3 Generative Adversarial Networks (GANs)

## 4 Variational Diffusion Models

## 5 References

# Contents

- 1 Introduction
- 2 Generative models
- 3 Generative Adversarial Networks (GANs)
- 4 Variational Diffusion Models
- 5 References

# Generative Adversarial Networks (GANs)

We require that the discriminator  $D$  recognises examples from the  $P_{\text{real}}(\mathbf{x})$  distribution,

$$\mathbb{E}_{\mathbf{x} \sim P_{\text{real}}(\mathbf{x})} [\log D(\mathbf{x})] \quad \text{Decision over Real Data}$$

where  $\mathbb{E}$  denotes the expectation. This term comes from the “1” class of the log-loss function.

Additionally, we would like to trick the discriminator via a good generator  $G$ . Thus, the term comes from “0” class of the log-loss function:

$$\mathbb{E}_{z \sim P_z(z)} [\log(1 - D(G(z)))] \quad \text{Decision over Fake Data}$$

## Generative Adversarial Networks (GANs)

Questions: Optimal Discriminator? Optimal Generator? We define:

$$V(G, D) := \mathbb{E}_{\mathbf{x} \sim P_{\text{real}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{z \sim P_z(z)} [\log(1 - D(G(z)))]$$

What should be an optimal discriminator?

## Generative Adversarial Networks (GANs)

Questions: Optimal Discriminator? Optimal Generator? We define:

$$V(G, D) := \mathbb{E}_{\mathbf{x} \sim P_{\text{real}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{z \sim P_z(z)} [\log(1 - D(G(z)))]$$

What should be an optimal discriminator?

$$D^* = \arg \max_D V(G, D)$$

Now, given this optimal discriminator  $D^*$ , what should be an optimal generator?

## Generative Adversarial Networks (GANs)

Questions: Optimal Discriminator? Optimal Generator? We define:

$$V(G, D) := \mathbb{E}_{\mathbf{x} \sim P_{\text{real}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{z \sim P_z(z)} [\log(1 - D(G(z)))]$$

What should be an optimal discriminator?

$$D^* = \arg \max_D V(G, D)$$

Now, given this optimal discriminator  $D^*$ , what should be an optimal generator?

$$G^* = \arg \min_G V(G, D^*)$$

This is called the *minimax formulation*, since the generator and discriminator are playing a *zero-sum game* against each other:

$$\min_G \max_D V(G, D) \tag{3}$$

## Generative Adversarial Networks (GANs)

$$\begin{aligned} & \min_G \max_D V(G, D) := \\ = & \mathbb{E}_{\mathbf{x} \sim P_{\text{real}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{z \sim P_z(z)} [\log(1 - D(G(z)))] \\ & \text{by Radon-Nikodym Theorem} \\ = & \mathbb{E}_{\mathbf{x} \sim P_{\text{real}}(\mathbf{x})} [\log D(\mathbf{x})] + \mathbb{E}_{x \sim P_{g(\mathbf{x})}} [\log(1 - D(x))] \\ = & \int_x (P_{\text{real}}(\mathbf{x})[\log D(\mathbf{x})] + P_{g(\mathbf{x})}[\log(1 - D(x))]) dx \end{aligned}$$

$$V(G, D) = \int_x \left( P_{\text{real}}(\mathbf{x})[\log D(\mathbf{x})] + P_{g(\mathbf{x})}[\log(1 - D(x))] \right) dx \quad (4)$$

- Note that  $f(x) = a \log(x) + b \log(1 - x)$ , has minimum in

$$x^* = \frac{a}{a+b}$$

$$V(G, D) = \int_x \left( P_{\text{real}}(\mathbf{x})[\log D(\mathbf{x})] + P_{g(\mathbf{x})}[\log(1 - D(x))] \right) dx \quad (4)$$

- Note that  $f(x) = a \log(x) + b \log(1 - x)$ , has minimum in  $x^* = \frac{a}{a+b}$
- Then If  $G$  is fixed,
- $P_{\text{real}}(\mathbf{x}) \log D(\mathbf{x}) + P_{g(\mathbf{x})} \log(1 - D(x))$  the optimal discriminator  $D^*(\mathbf{x}) = \frac{P_{\text{real}}(\mathbf{x})}{P_{\text{real}}(\mathbf{x}) + P_g(\mathbf{x})}$

Given an optimal discriminator  $D^*(\mathbf{x})$ , find  $\min_G V(G, D^*)$ .

$$\int_x (P_{\text{real}}(\mathbf{x}) \log \left( \frac{P_{\text{real}}(\mathbf{x})}{P_{\text{real}}(\mathbf{x}) + P_g(\mathbf{x})} \right) + P_{g(\mathbf{x})} \log \left( 1 - \frac{P_{\text{real}}(\mathbf{x})}{P_{\text{real}}(\mathbf{x}) + P_g(\mathbf{x})} \right))$$

---

<sup>1</sup>Divergence Jensen–Shannon is a symmetrized and smoothed version of the Kullback–Leibler divergence,  $\text{Div}_{\text{JS}}(p||q) = \frac{1}{2} \text{Div}_{\text{KL}}(p||m) + \frac{1}{2} \text{Div}_{\text{KL}}(q||m)$  where  $m = p + q$

Given an optimal discriminator  $D^*(\mathbf{x})$ , find  $\min_G V(G, D^*)$ .

$$\int_x (P_{\text{real}}(\mathbf{x}) \log \left( \frac{P_{\text{real}}(\mathbf{x})}{P_{\text{real}}(\mathbf{x}) + P_g(\mathbf{x})} \right) + P_{g(\mathbf{x})} \log \left( 1 - \frac{P_{\text{real}}(\mathbf{x})}{P_{\text{real}}(\mathbf{x}) + P_g(\mathbf{x})} \right))$$

by means of Divergence Jensen–Shannon<sup>1</sup> we can obtain:

$$Div_{\text{JS}}(P_{\text{real}} || P_g) = \frac{1}{2} (\log 4 + \min_G V(G, D^*)) \quad (5)$$

---

<sup>1</sup>Divergence Jensen–Shannon is a symmetrized and smoothed version of the Kullback–Leibler divergence,  $Div_{\text{JS}}(p || q) = \frac{1}{2} Div_{\text{KL}}(p || m) + \frac{1}{2} Div_{\text{KL}}(q || m)$  where  $m = p + q$

Summarising, we have:

- Optimal Discriminator:  $D^*(\mathbf{x}) = \frac{P_{\text{real}}(\mathbf{x})}{P_{\text{real}}(\mathbf{x}) + P_g(\mathbf{x})}$
- $\min_G V(G, D^*) = 2\text{Div}_{\text{JS}}(P_{\text{real}} || P_g) - 2 \log 2$

Summarising, we have:

- Optimal Discriminator:  $D^*(\mathbf{x}) = \frac{P_{\text{real}}(\mathbf{x})}{P_{\text{real}}(\mathbf{x}) + P_g(\mathbf{x})}$
- $\min_G V(G, D^*) = 2\text{Div}_{\text{JS}}(P_{\text{real}} || P_g) - 2 \log 2$

Thus,

- $\min_G V(G, D^*)$  is obtained when  $P_{\text{real}} = P_g$ ,

Summarising, we have:

- Optimal Discriminator:  $D^*(\mathbf{x}) = \frac{P_{\text{real}}(\mathbf{x})}{P_{\text{real}}(\mathbf{x}) + P_g(\mathbf{x})}$
- $\min_G V(G, D^*) = 2\text{Div}_{\text{JS}}(P_{\text{real}} || P_g) - 2 \log 2$

Thus,

- $\min_G V(G, D^*)$  is obtained when  $P_{\text{real}} = P_g$ ,
- $D^*(\mathbf{x}) = \frac{1}{2}$

Summarising, we have:

- Optimal Discriminator:  $D^*(\mathbf{x}) = \frac{P_{\text{real}}(\mathbf{x})}{P_{\text{real}}(\mathbf{x}) + P_g(\mathbf{x})}$
- $\min_G V(G, D^*) = 2\text{Div}_{\text{JS}}(P_{\text{real}} || P_g) - 2 \log 2$

Thus,

- $\min_G V(G, D^*)$  is obtained when  $P_{\text{real}} = P_g$ ,
- $D^*(\mathbf{x}) = \frac{1}{2}$
- and  $\min_G \max_D V(G, D) = -2 \log 2$

## Proof with more details

- ① Annotated Proof of GANs [https://srome.github.io/  
An-Annotated-Proof-of-Generative-Adversarial-Networks-](https://srome.github.io/An-Annotated-Proof-of-Generative-Adversarial-Networks-)
- ② Generative Adversarial Nets:  
[\(https://arxiv.org/pdf/1406.2661.pdf\)](https://arxiv.org/pdf/1406.2661.pdf)

## Training GANs: Training the Discriminator:

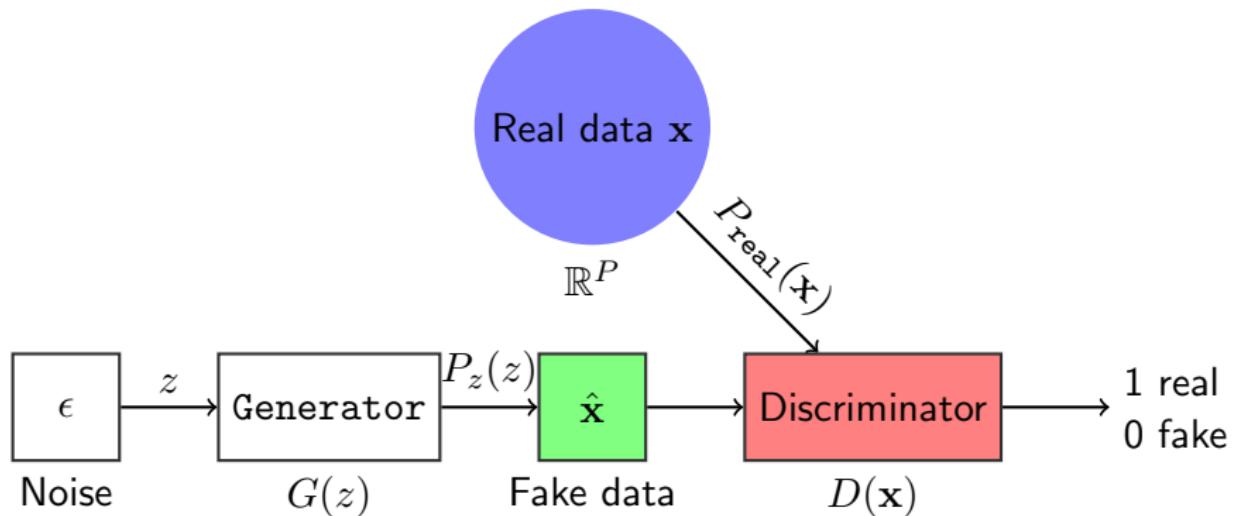


Figure: Only Discriminator should be updated

## Training GANs: Training the Generator

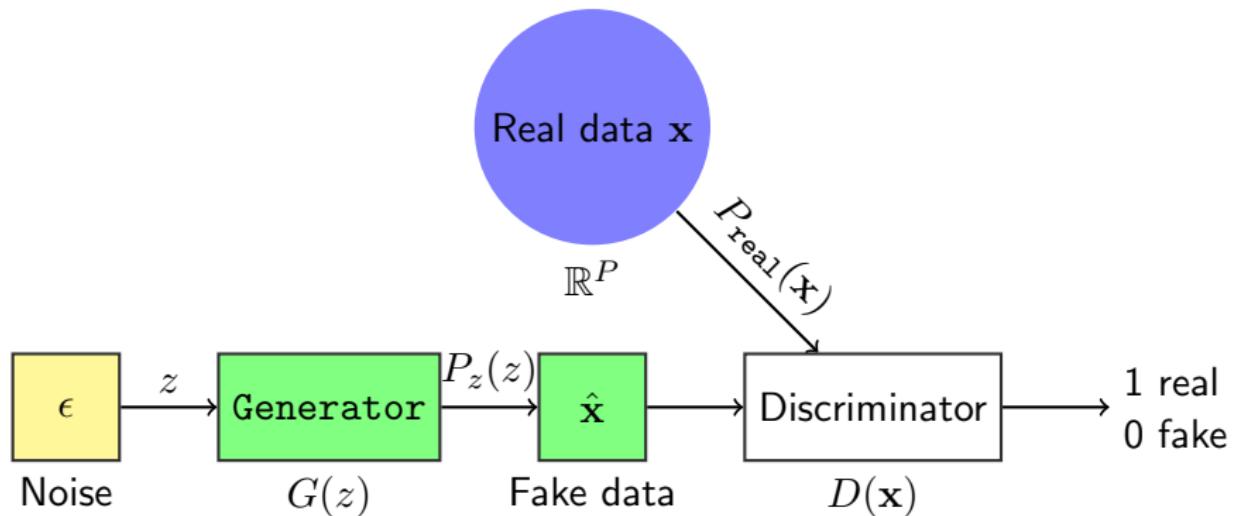


Figure: Discriminator should be set to "non-trainable" weights

# How to Train a GAN?

Bad news: GANs are hard to train. **Tips and tricks to make GANs work** <https://github.com/soumith/ganhacks>

- Use others loss functions.
- Avoid Sparse Gradients: ReLU, MaxPool
- Use Deep Convolutional Generative Adversarial Networks when you can. It works!
- Use SGD for discriminator and ADAM for generator
- Use Dropouts in G in both train and test phase
- ...

“In theory, there is no difference between theory and practice. In practice, there is.”

## High-resolution GANs

Progressive Growing of GANs for Improved Quality, Stability, and Variation [Karras et al., 2017]

<https://www.youtube.com/watch?v=G06dEcZ-QTg>

## Applications of GANs: Earn money!

# Applications of GANs: Earn money!

23-25 Oct, 2018: Auction house "Christies" sold a portrait for 432,000 dollars that had been generated by a GANs



LOT 363

*Edmond de Belamy, from La Famille de Belamy*

Price realised ⓘ

USD 432,500

Estimate ⓘ

USD 7,000 - USD 10,000

Follow lot



+ Add to Interests

*Edmond de Belamy, from La Famille de Belamy*

generative Adversarial Network print, on canvas, 2018, signed with GAN model loss function in ink by the publisher, from a series of eleven unique images, published by Obvious Art, Paris, with original gilded wood frame  
S. 27 ½ x 27 ½ in (700 x 700 mm.)

## GANs to improve simulations

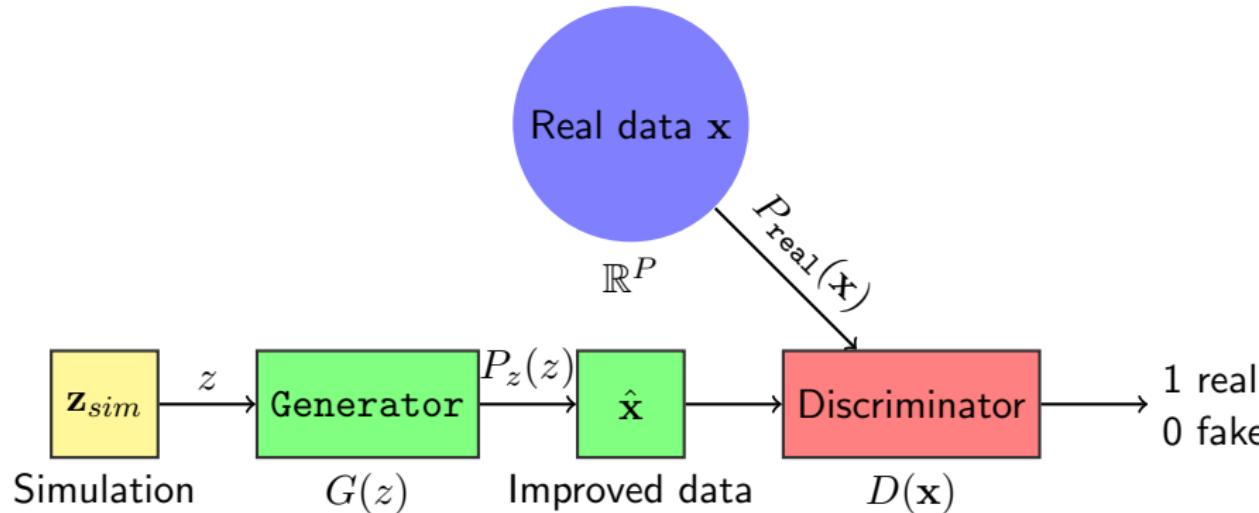


Figure: In this case, Generator is called Refiner

## Application: Refine simulations

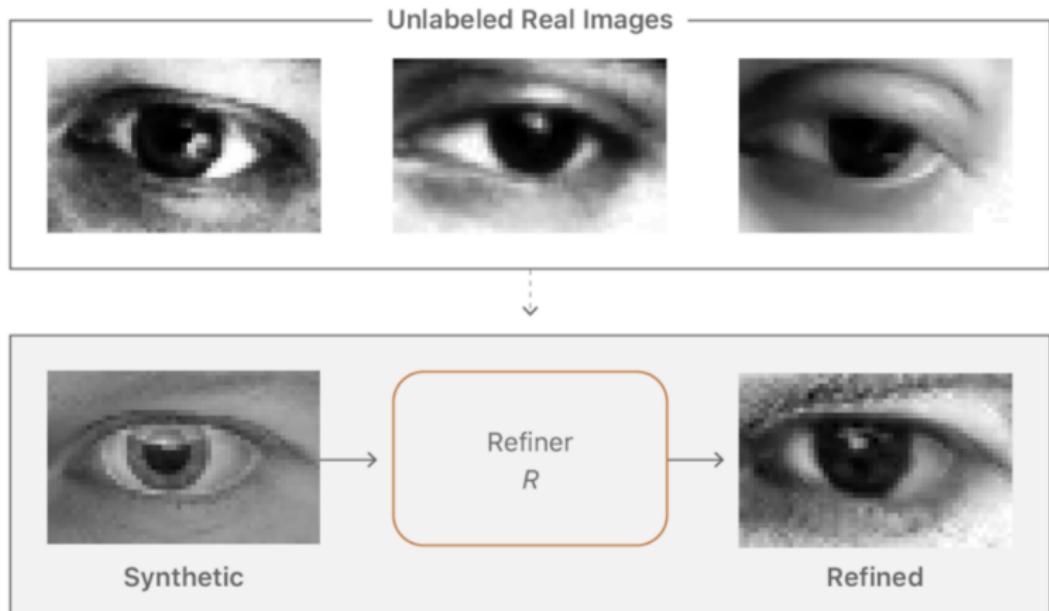
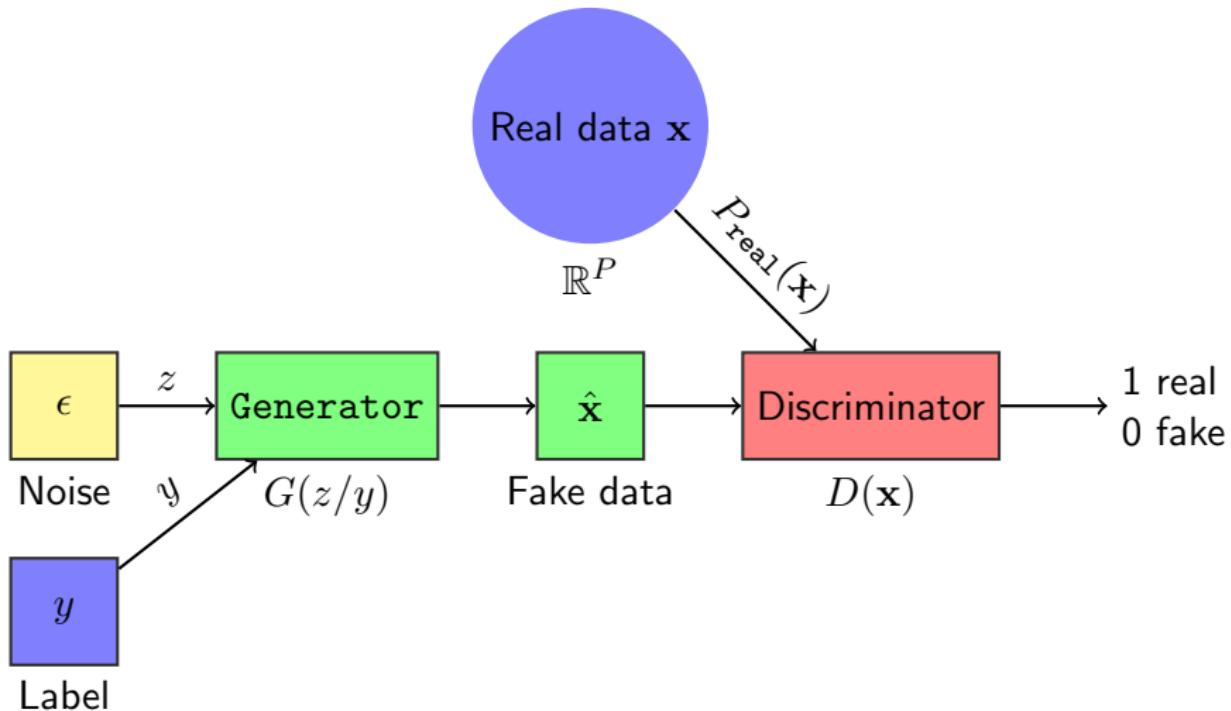


Figure: Learning from Simulated and Unsupervised Images through Adversarial Training [Shrivastava et al., 2017]

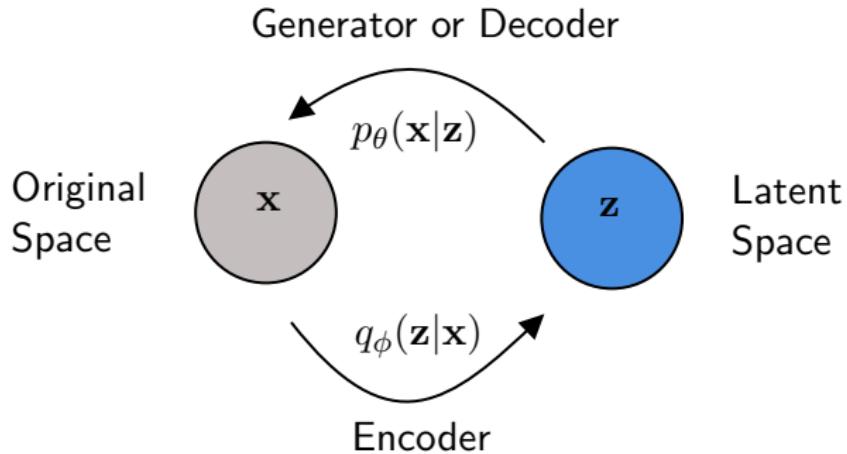
# Conditional Generative Adversarial Networks (cGANs)



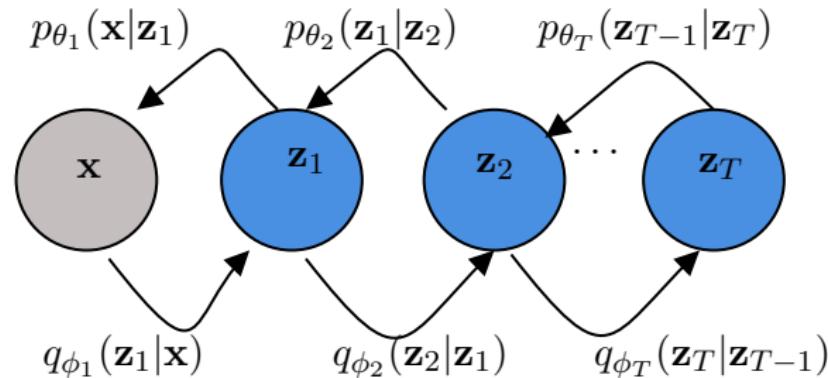
# Contents

- 1 Introduction
- 2 Generative models
- 3 Generative Adversarial Networks (GANs)
- 4 Variational Diffusion Models
- 5 References

# Variational Autoencoder



## Markovian variational autoencoder with $T$ hierarchical latent spaces



The generative process is modeled as a Markov Chain, where each latent value  $\mathbf{z}_t$  is generated only from the previous latent  $\mathbf{z}_{t+1}$ .

## Markovian variational autoencoder with $T$ hierarchical latent spaces

The joint distribution and the posterior of a Markovian Hierarchical VAE as:

$$p(\mathbf{x}, \mathbf{z}_{1:T}) = p(\mathbf{z}_T) p_{\theta_1}(\mathbf{x}|\mathbf{z}_1) \prod_{t=2}^T p_{\theta_t}(\mathbf{z}_{t-1}|\mathbf{z}_t), \quad (6)$$

$$q_{\phi}(\mathbf{z}_{1:T}|\mathbf{x}) = q_{\phi_1}(\mathbf{z}_1|\mathbf{x}) \prod_{t=2}^T q_{\phi_t}(\mathbf{z}_t|\mathbf{z}_{t-1}) \quad (7)$$

## ELBO derivation

$$\log p(\mathbf{x}) = \log \int p(\mathbf{x}, \mathbf{z}_{1:T}) d\mathbf{z}_{1:T} \quad (8)$$

$$= \log \int \frac{p(\mathbf{x}, \mathbf{z}_{1:T}) q_\phi(\mathbf{z}_{1:T} | \mathbf{x})}{q_\phi(\mathbf{z}_{1:T} | \mathbf{x})} d\mathbf{z}_{1:T} \quad (9)$$

$$= \log \mathbb{E}_{q_\phi(\mathbf{z}_{1:T} | \mathbf{x})} \left( \frac{p(\mathbf{x}, \mathbf{z}_{1:T})}{q_\phi(\mathbf{z}_{1:T} | \mathbf{x})} \right) \quad (10)$$

$$\geq \mathbb{E}_{q_\phi(\mathbf{z}_{1:T} | \mathbf{x})} \log \left( \frac{p(\mathbf{x}, \mathbf{z}_{1:T})}{q_\phi(\mathbf{z}_{1:T} | \mathbf{x})} \right) \quad (11)$$

## Variational Diffusion Models

That is a Markovian variational autoencoder with the following restrictions:

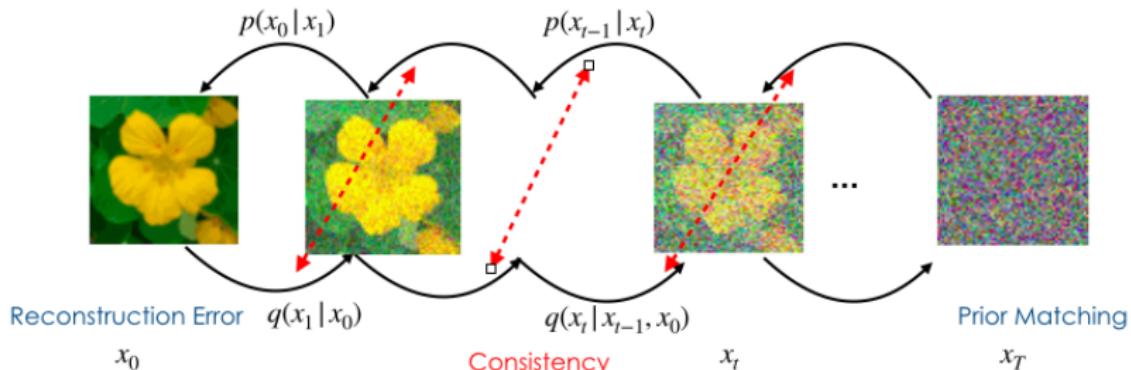
- ① The latent dimension is exactly equal to the data dimension
- ② The latent encoder at each timestep is predefined as a linear Gaussian model. A Gaussian distribution centred around the output of the previous timestep.
- ③ After the final timestep  $T$  the distribution in the latent space is a standard Gaussian distribution.

Use double conditioning.

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_0) \quad (12)$$

$$\begin{aligned} \log p(\mathbf{x}) &\geq \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x})} \log \left( \frac{p(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right) \\ &= \dots \\ &= \underbrace{\mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)} [\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)]}_{\text{Reconstruction term}} - \underbrace{\mathbb{E}_{q(\mathbf{x}_T|\mathbf{x}_0)} \left( \log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_{T-1})} \right)}_{\text{KL} \geq 0, \text{ Prior Matching Term}} \\ &+ \sum_{t=1}^T \underbrace{\mathbb{E}_{q(\mathbf{x}_{t-1}, \mathbf{x}_{t+1}|\mathbf{x}_0)} \left( \log \frac{p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \right)}_{\text{KL} \geq 0, \text{ Consistency term}} \end{aligned}$$

# Variational Diffusion Models



We must be able to “**undo**” the forward noising process:

- Given a target  $x_0$ , you must be able to *reverse* the process of noising to reconstruct  $x_0$ .
- This noising process happens in small increments.

At the time-step  $T$ , the process should be a *random noise*:

- After iteratively adding noise  $T$  times, the result must be independent of the starting point  $x_0$

## Forward Diffusion

Let  $\alpha_t$  be the noise schedule for  $t \in \{0, \dots, T - 1\}$ .  $\alpha_t$  decreases from 1 to 0 as  $t$  increases. We can define a *forward* noising function as the distribution:

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha} \mathbf{x}_{t-1}, (1 - \alpha_t) \mathbf{I}) \quad (13)$$

## Forward Diffusion

Let  $\alpha_t$  be the noise schedule for  $t \in \{0, \dots, T - 1\}$ .  $\alpha_t$  decreases from 1 to 0 as  $t$  increases. We can define a *forward* noising function as the distribution:

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha} \mathbf{x}_{t-1}, (1 - \alpha_t) \mathbf{I}) \quad (13)$$

We can also arbitrary sample

$$\mathbf{x}_t \sim q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I}) \text{ where } \bar{\alpha}_t = \prod_{i=1}^t \alpha_i$$

## Reverse Diffusion

The “reverse” denoising function as the posterior distribution:

$$\begin{aligned} q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) &= \mathcal{N}(\mathbf{x}_{t-1}; \frac{1}{1 - \bar{\alpha}_t} (\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})\mathbf{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1 - \alpha_t)\mathbf{x}_0) \\ &\quad, \frac{(1 - \alpha_t)(1 - \bar{\alpha}_t)}{1 - \bar{\alpha}_{t-1}} \mathbf{I}) \\ &= \mathcal{N}(\mathbf{x}_{t-1}; \mu_q(\mathbf{x}_t, \mathbf{x}_0), \Sigma_q(t)) \end{aligned}$$

We call the reverse diffusion process parameterised by a model

$$\begin{aligned} p(\mathbf{x}_{t-1} | \mathbf{x}_t) &= \mathcal{N}(\mathbf{x}_{t-1}; \frac{1}{1 - \bar{\alpha}_t} (\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})\mathbf{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1 - \alpha_t)\hat{\mathbf{x}}_0), \\ &\quad \frac{(1 - \alpha_t)(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \mathbf{I}) \end{aligned}$$

Where the model defines the distribution over  $\hat{\mathbf{x}}_0$ :

$$p(\hat{\mathbf{x}}_0 | \mathbf{x}_t, \mathbf{c}) = \text{model}_\theta(\mathbf{x}_t, \mathbf{c})$$

## Reverse Diffusion Example

Consider a single pixel of a greyscale image, where the value ranges from 0 to 1

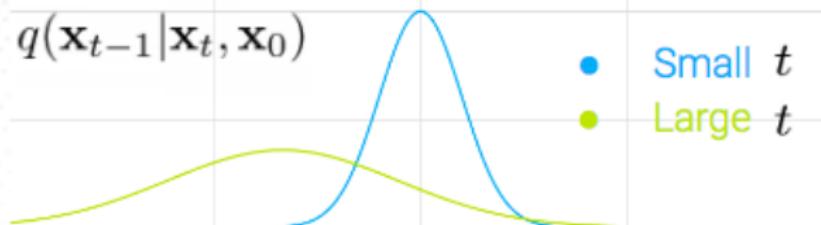
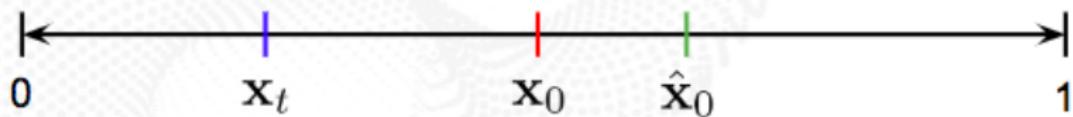
- Current value at  $\mathbf{x}_t$
- Target value at  $\mathbf{x}_0$
- Model prediction at  $\hat{\mathbf{x}}_0$

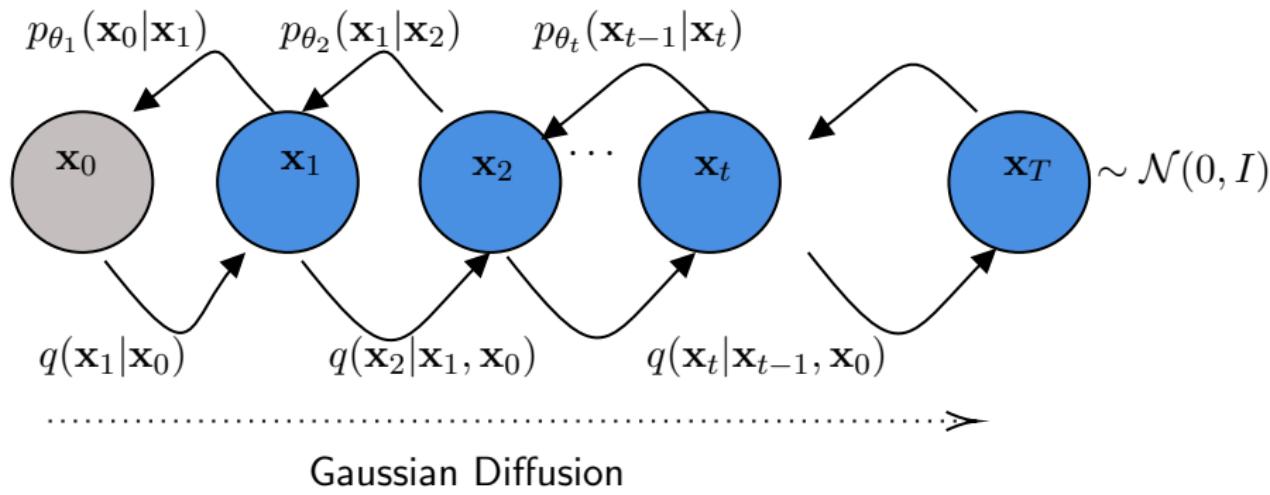
## Reverse Diffusion Example

Consider a single pixel of a greyscale image, where the value ranges from 0 to 1

- Current value at  $\mathbf{x}_t$
- Target value at  $\mathbf{x}_0$
- Model prediction at  $\hat{\mathbf{x}}_0$

$$\frac{\mu_p(\mathbf{x}_t, \hat{\mathbf{x}}_0)}{\mu_q(\mathbf{x}_t, \mathbf{x}_0)}$$





# Diffusion models

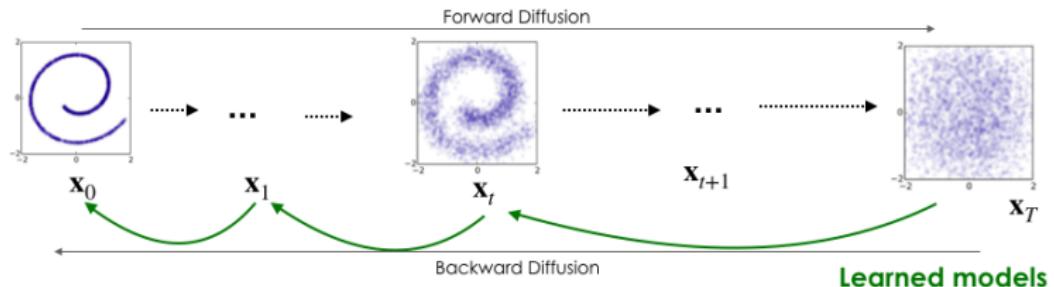


Figure: Diffusion models

## Denoising Diffusion models

Over the last two years massive advances in the field of image synthesis with generative models have been made.

- ① DALL-E 2
- ② Midjourney
- ③ Stable Diffusion

# Stable Diffusion models

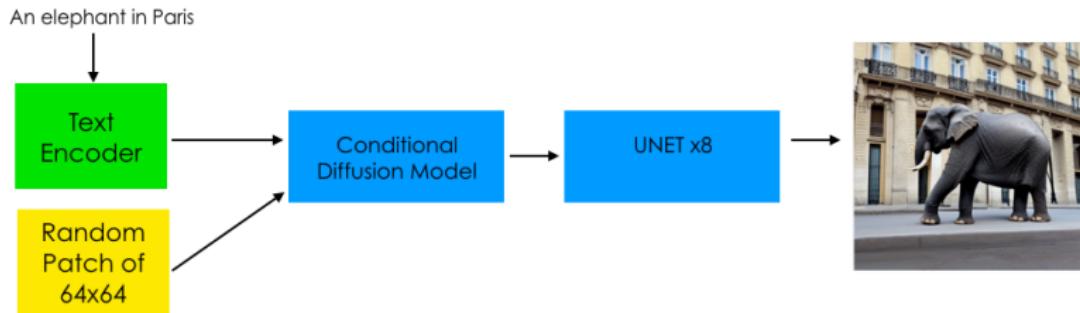


Figure: StableDiffusion

# Stable Diffusion models

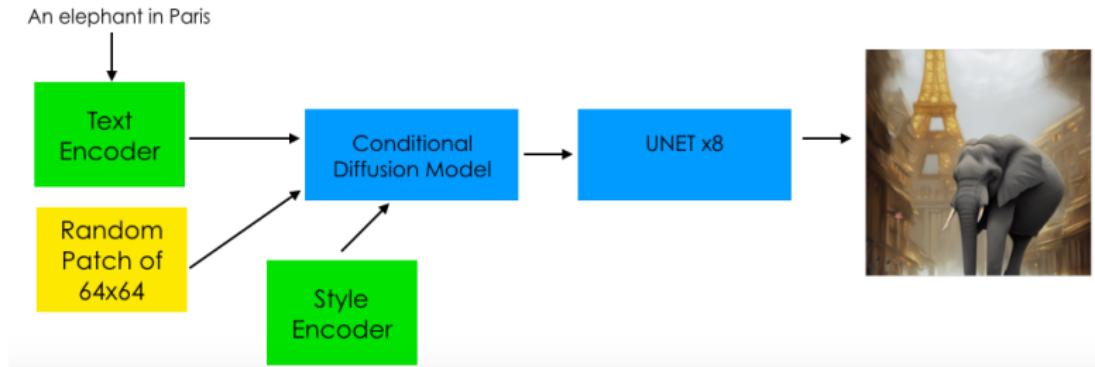


Figure: StableDiffusion

## Example: Colab

- ① Conditional GANs

<https://colab.research.google.com/drive/12C-ngdYQrEB8scko6uh9ZQEoQAcpa6uC?usp=sharing>

- ② Understanding Diffusion

<https://colab.research.google.com/drive/1dgml5rx6QKq7LSzwho6oWm7TvEdxgICf?usp=sharing>

- ③ Stable Diffusion Model

[https://colab.research.google.com/drive/1xmskXN15B3vK\\_HtwKy\\_HNF1KgCPgemiT?usp=sharing](https://colab.research.google.com/drive/1xmskXN15B3vK_HtwKy_HNF1KgCPgemiT?usp=sharing)

# Contents

- 1 Introduction
- 2 Generative models
- 3 Generative Adversarial Networks (GANs)
- 4 Variational Diffusion Models
- 5 References

## References I

- [Antipov et al., 2017] Antipov, G., Baccouche, M., and Dugelay, J.-L. (2017). Face aging with conditional generative adversarial networks. In *IEEE International Conference on Image Processing (ICIP)*, pages 2089–2093. IEEE.
- [Karras et al., 2017] Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*.
- [Shrivastava et al., 2017] Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., and Webb, R. (2017). Learning from simulated and unsupervised images through adversarial training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.