Introduction to Diffusion Models

Part 1: Introduction, Formulation and Theory

Leonid Sigal and Siddhesh Khandelwal

Introduction : Generative Models

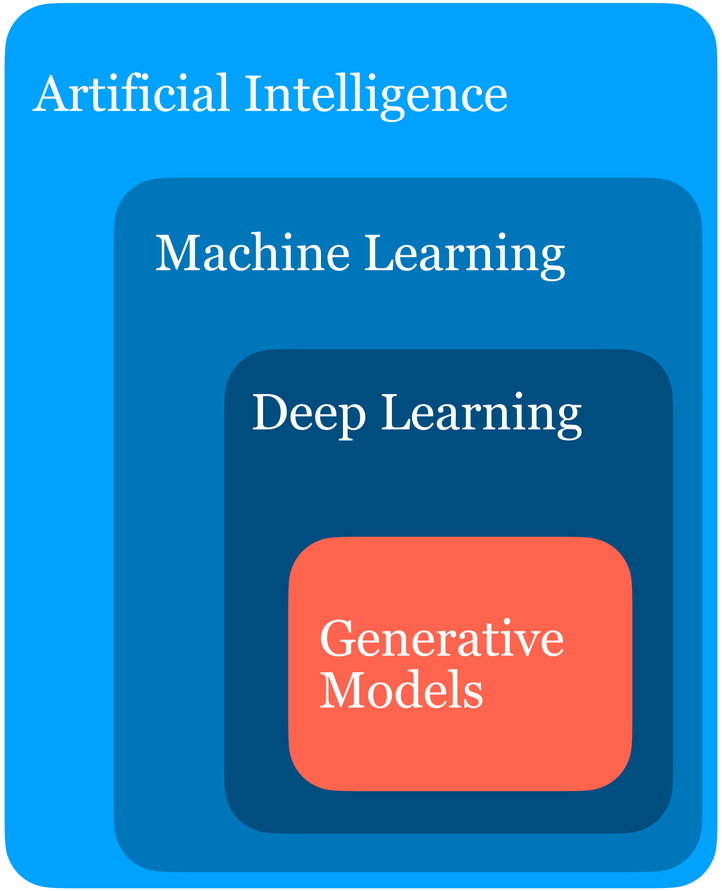
- Most AI methods are **predictive** in nature.
  - Observe and classify patterns in content or predict a new pattern
  - Ex: Label objects in an image.

- **Generative Models,** on the other hand, are capable of **creating** text, images, videos, and other types of content.
  - These models can be **conditioned** on a text prompt to get the desired output.

Introduction : Generative Models

- Most AI methods are **predictive** in nature.
  - Observe and classify patterns in content or predict a new pattern
  - Ex: Label objects in an image.

- **Generative Models,** on the other hand, are capable of **creating** text, images, videos, and other types of content.
  - These models can be **conditioned** on a text prompt to get the desired output.



Introduction : Generative Models

Introduction : Generative Models

Samples from a Data Distribution

Train

Neural Network

Introduction : Generative Models

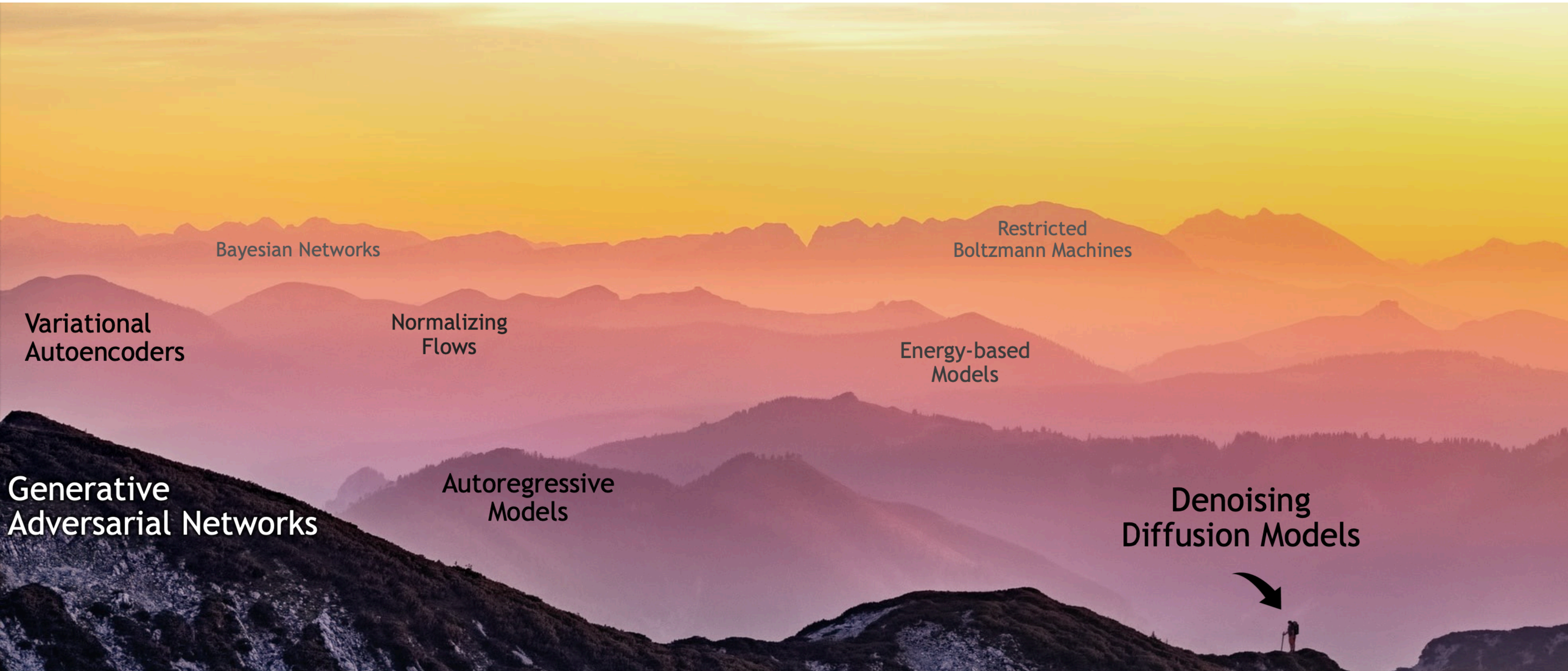
Samples from a Data Distribution

Train

Neural Network

Sample

Introduction : Generative Models



Properties of Good Generative Model

Properties of Good Generative Model

High Quality Samples

Mode Coverage / Diversity

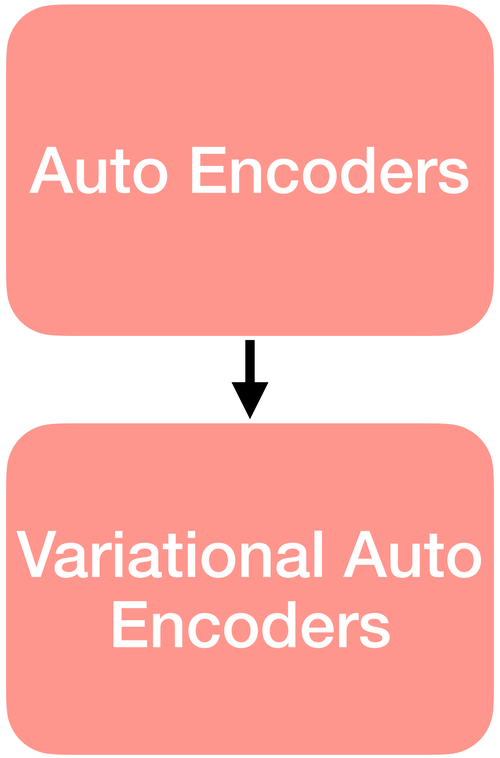
Fast Sampling

Overview

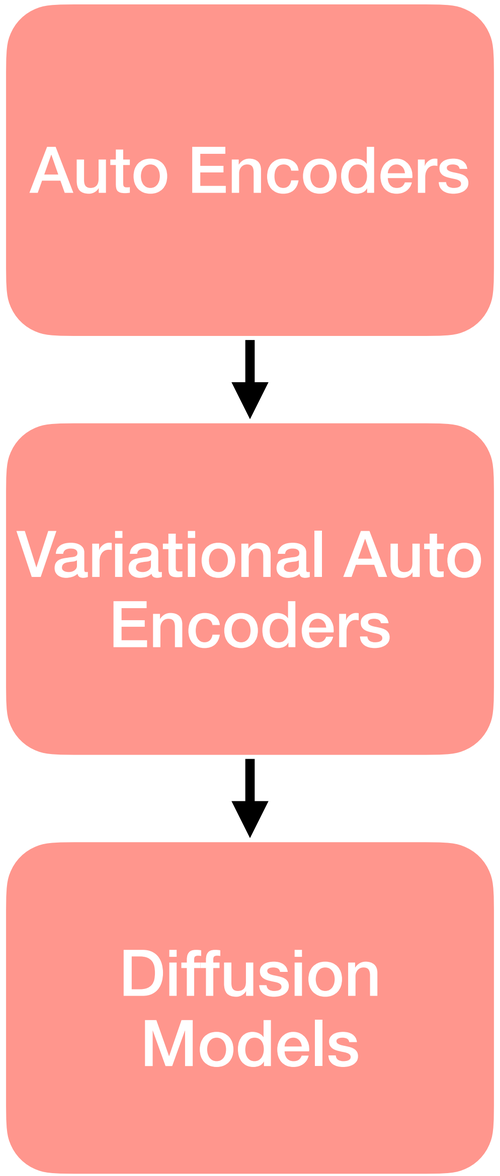
Overview

Auto Encoders

Overview



Overview


Auto Encoders
Variational Auto Encoders
Diffusion Models

Introduction : Auto Encoders

- An autoencoder is a neural network whose job is to take an input **x** and predict the same output **x**

Introduction : Auto Encoders

- An autoencoder is a neural network whose job is to take an input **x** and predict the same output **x**
  - Consists of two components
    Encoder: transforms the input into a latent representation

Image taken from https://en.wikipedia.org/wiki/Variational_autoencoder

Introduction : Auto Encoders

- An autoencoder is a neural network whose job is to take an input **x** and predict the same output **x**
  - Consists of two components
    Encoder: transforms the input into a latent representation

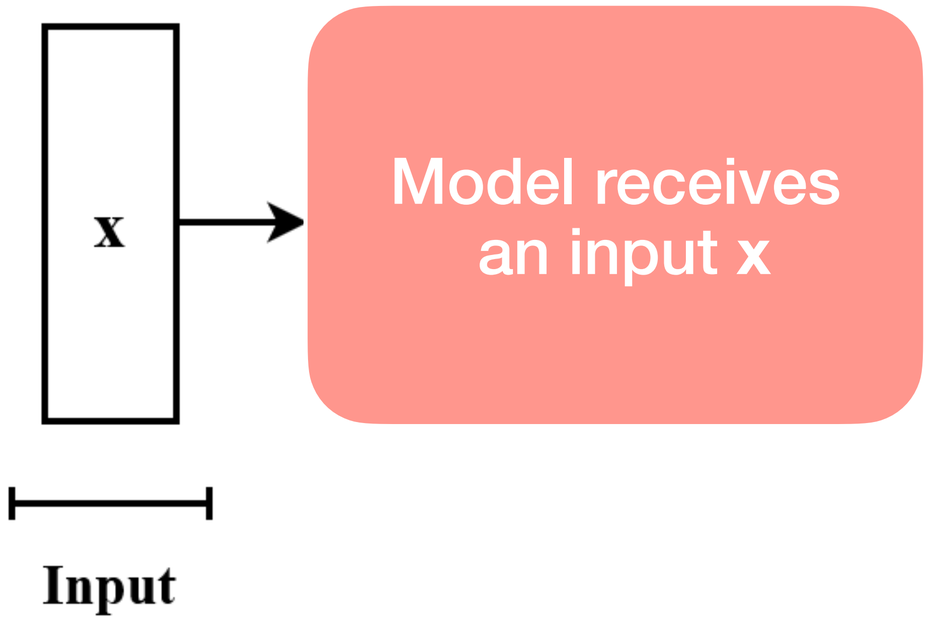


Image taken from https://en.wikipedia.org/wiki/Variational_autoencoder

Introduction : Auto Encoders

- An autoencoder is a neural network whose job is to take an input **x** and predict the same output **x**
  - Consists of two components
    Encoder: transforms the input into a latent representation

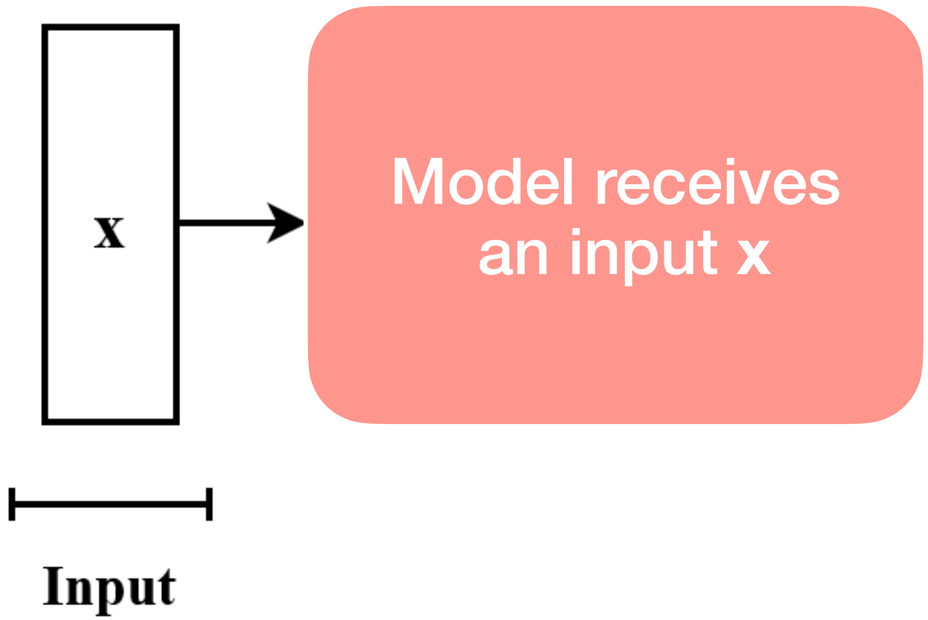


Image taken from https://en.wikipedia.org/wiki/Variational_autoencoder

Introduction : Auto Encoders

- An autoencoder is a neural network whose job is to take an input **x** and predict the same output **x**
  - Consists of two components
    Encoder: transforms the input into a latent representation

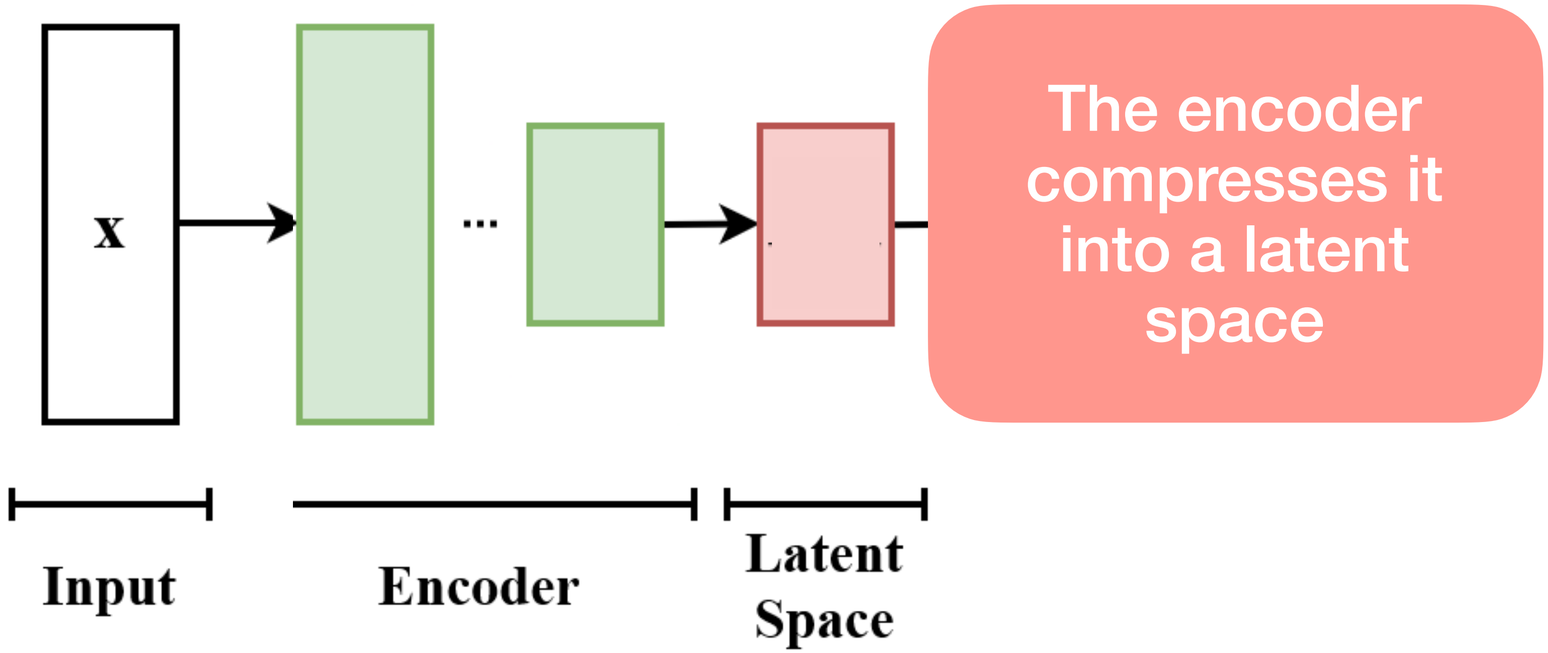


Image taken from https://en.wikipedia.org/wiki/Variational_autoencoder

Introduction : Auto Encoders

- An autoencoder is a neural network whose job is to take an input **x** and predict the same output **x**
  - Consists of two components
    Encoder: transforms the input into a latent representation
    Decoder: recreates the input from the latent representation

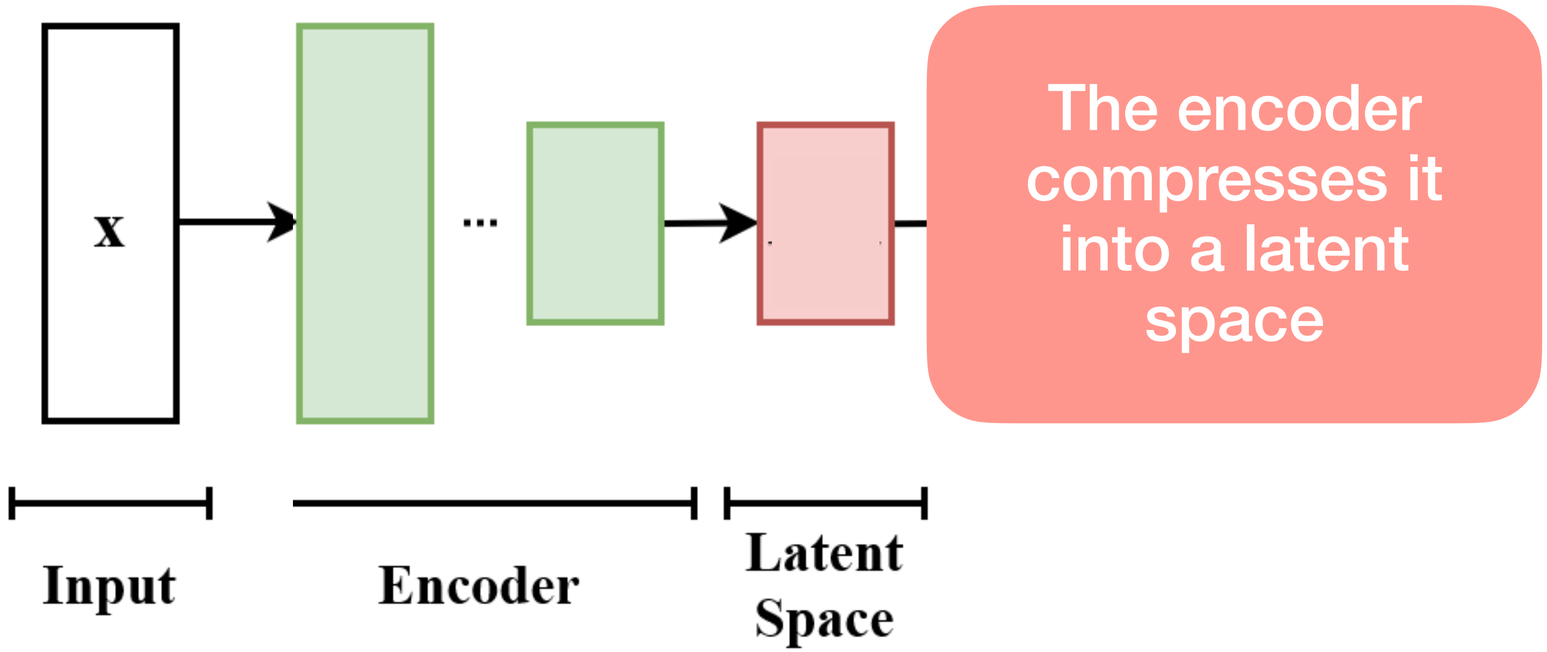
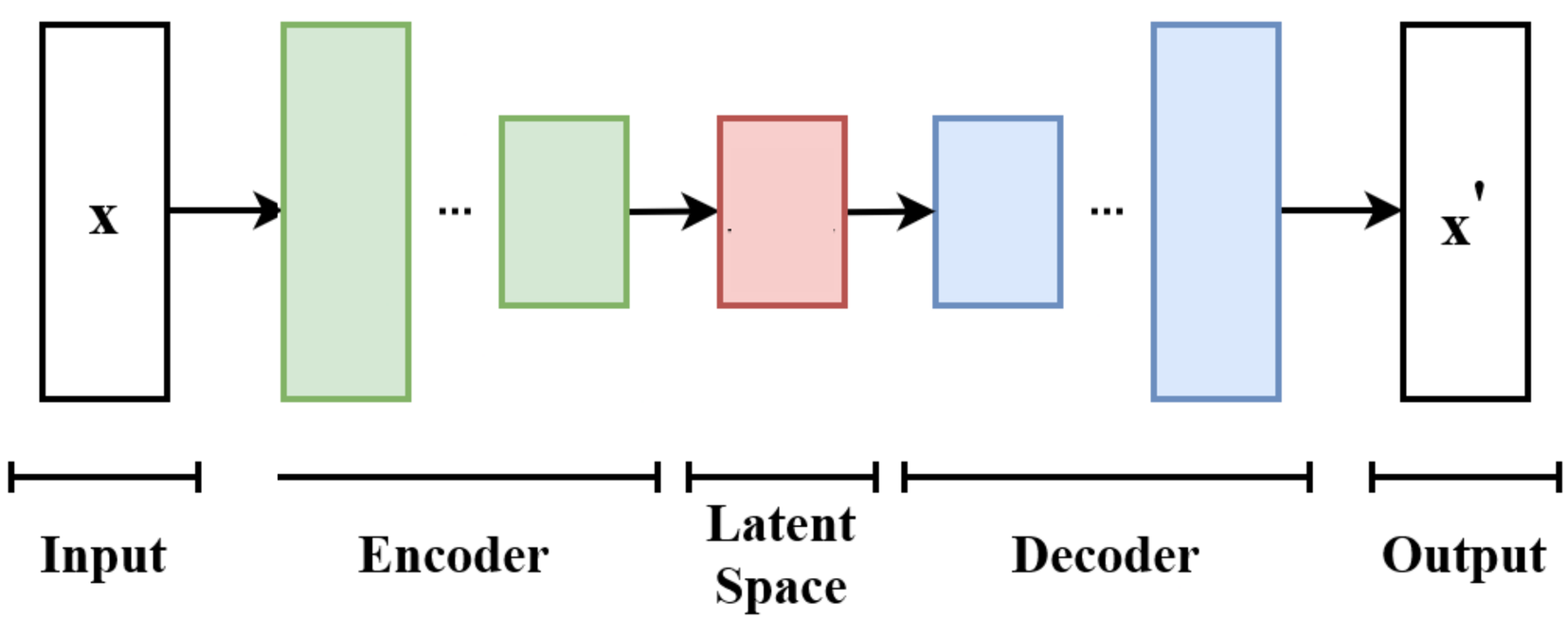


Image taken from https://en.wikipedia.org/wiki/Variational_autoencoder

Introduction : Auto Encoders

- An autoencoder is a neural network whose job is to take an input **x** and predict the same output **x**
  - Consists of two components
    Encoder: transforms the input into a latent representation
    Decoder: recreates the input from the latent representation

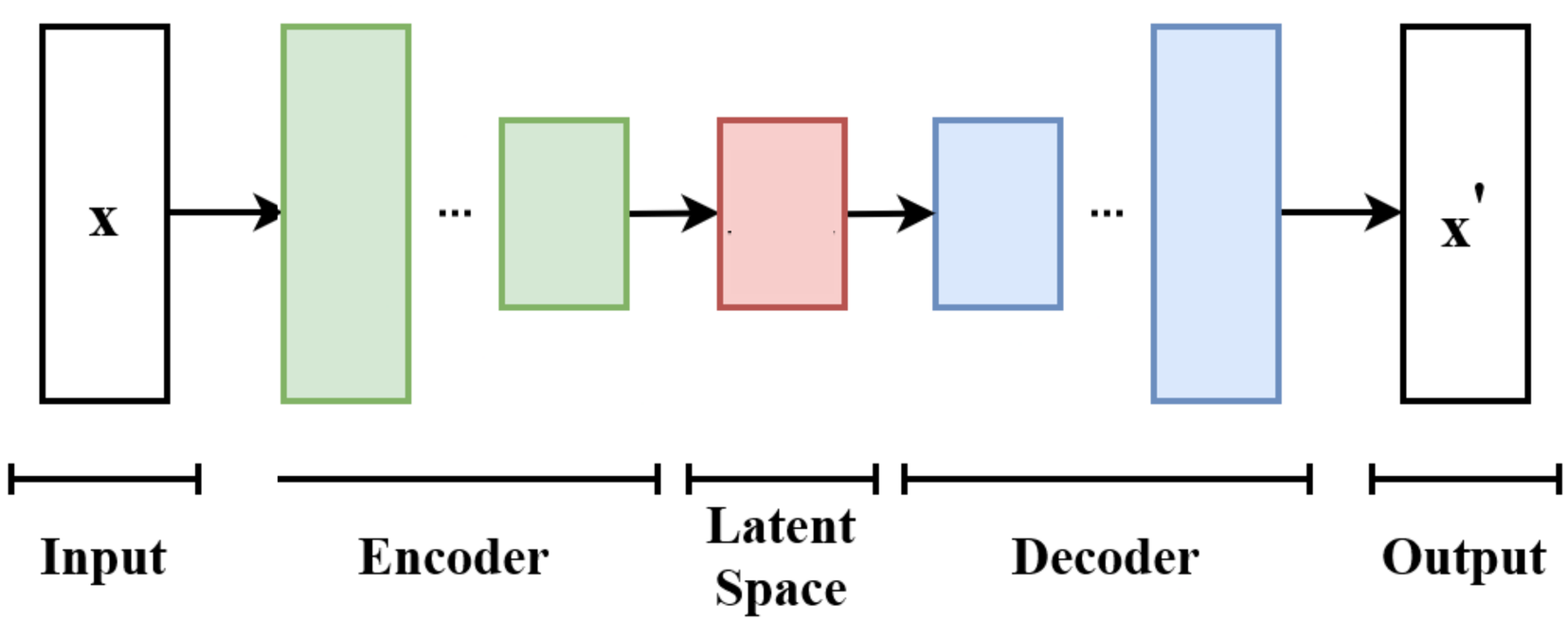


The decoder produces x' as similar as possible to x

Image taken from https://en.wikipedia.org/wiki/Variational_autoencoder

Introduction : Auto Encoders

- An autoencoder is a neural network whose job is to take an input **x** and predict the same output **x**
  - Consists of two components
    Encoder: transforms the input into a latent representation
    Decoder: recreates the input from the latent representation

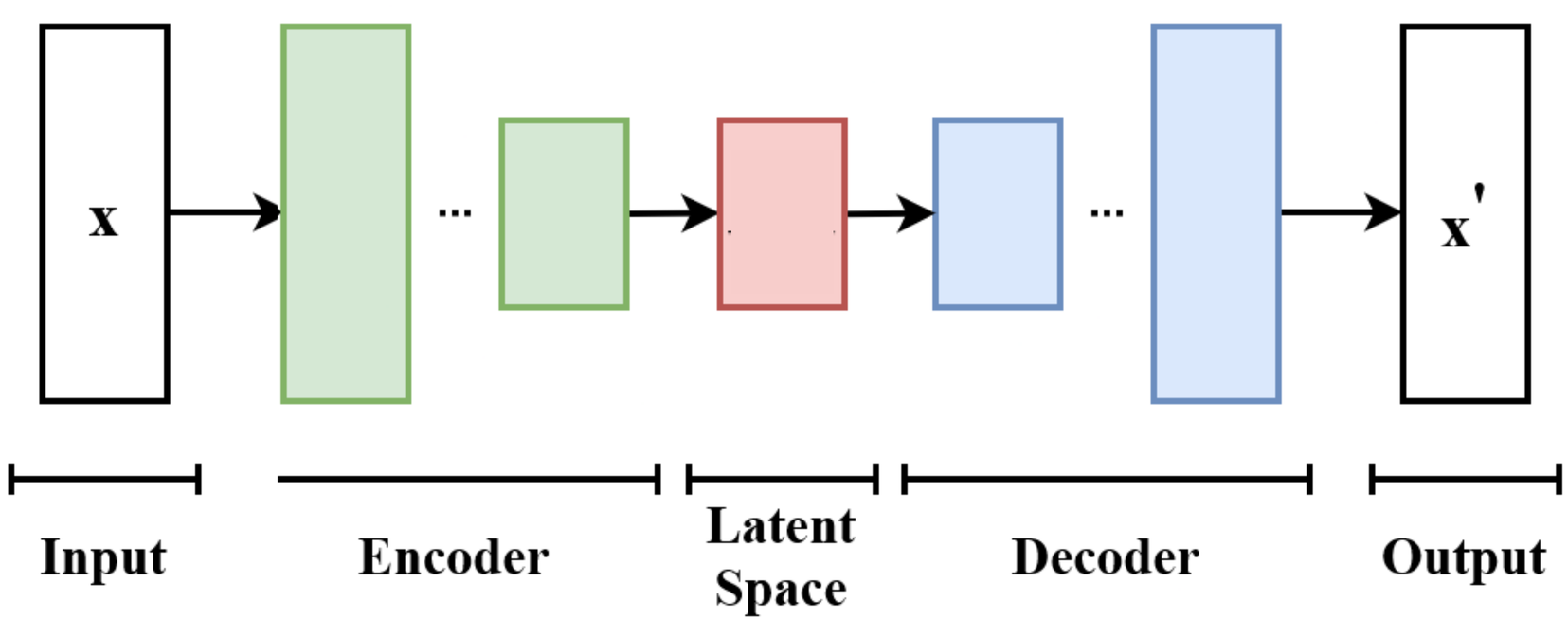


The decoder produces x' as similar as possible to x

Image taken from https://en.wikipedia.org/wiki/Variational_autoencoder

Introduction : Auto Encoders

- An autoencoder is a neural network whose job is to take an input **x** and predict the same output **x**
  - Consists of two components
    Encoder: transforms the input into a latent representation
    Decoder: recreates the input from the latent representation



The latent space size is much smaller than the input to make this non-trivial

Image taken from https://en.wikipedia.org/wiki/Variational_autoencoder

Introduction : Auto Encoders

- An autoencoder is a neural network whose job is to take an input **x** and predict the same output **x**
  - Consists of two components
    Encoder: transforms the input into a latent representation
    Decoder: recreates the input from the latent representation

- Autoencoders essentially perform **compression**
  - Map high-dimensional data to a lower-dimensional latent code.

Image taken from https://en.wikipedia.org/wiki/Variational_autoencoder

Introduction : Auto Encoders

- Autoencoders can technically be used for data generation.
  - Pass an arbitrary latent code into the decoder.

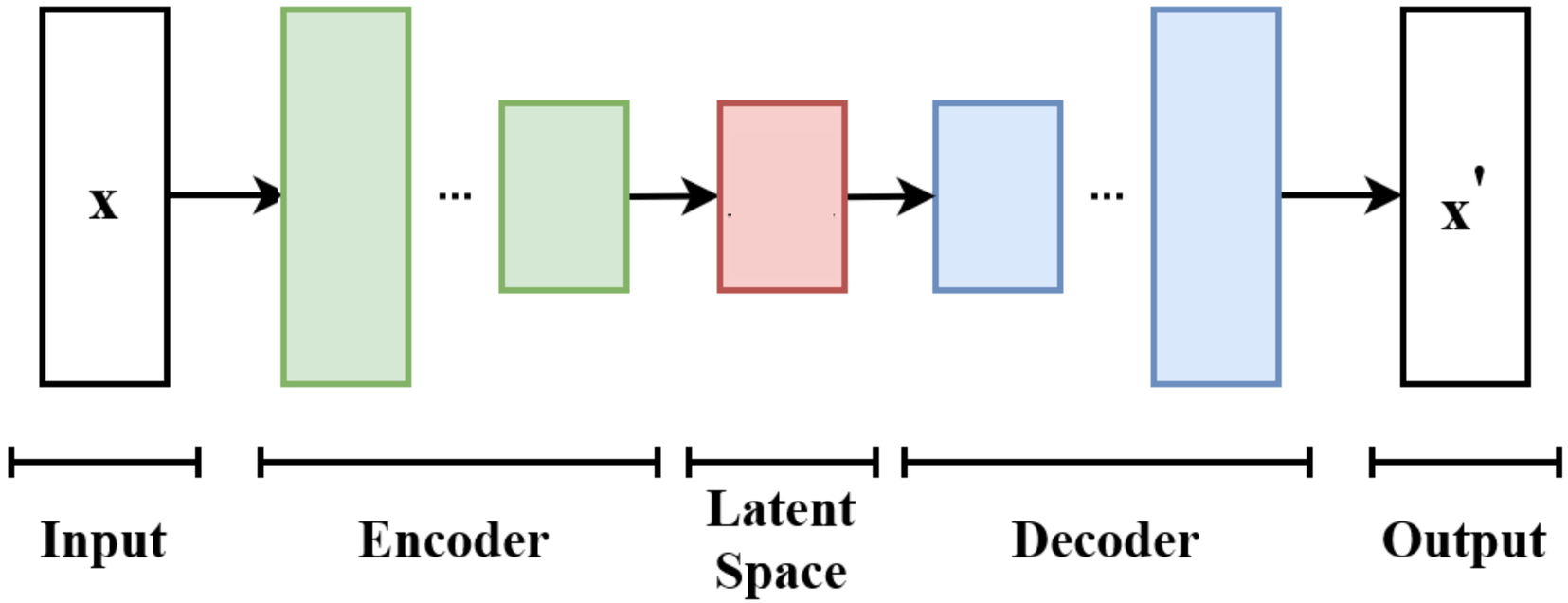


Image taken from https://en.wikipedia.org/wiki/Variational_autoencoder

Introduction : Auto Encoders

- Autoencoders can technically be used for data generation.
  - Pass an arbitrary latent code into the decoder.

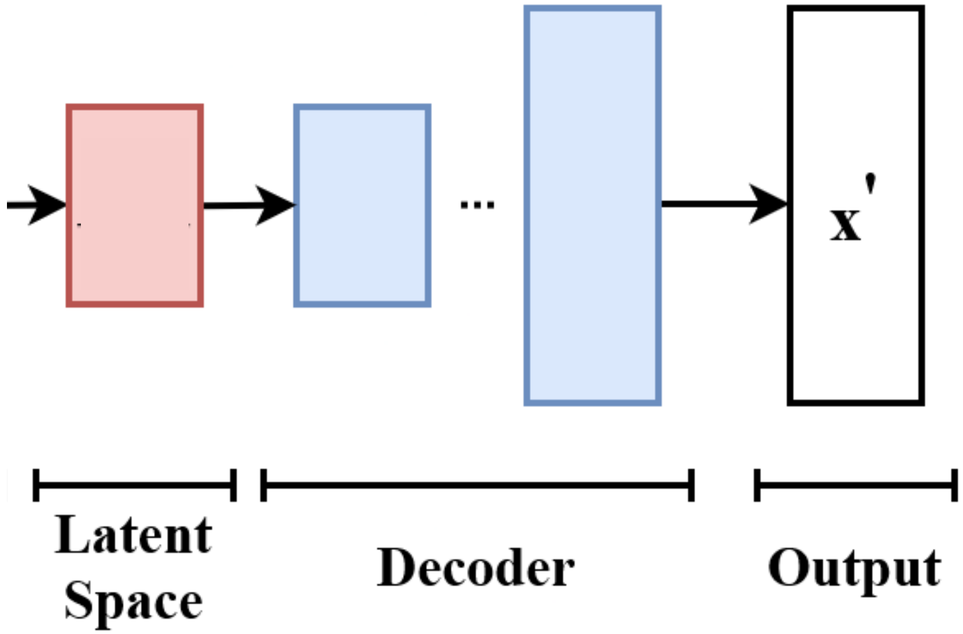


Image taken from https://en.wikipedia.org/wiki/Variational_autoencoder

Introduction : Auto Encoders

- Autoencoders can technically be used for data generation.
  - Pass an arbitrary latent code into the decoder.

- However, the fundamental problem with autoencoders is that the latent space they learn may **not be continuous**, or allow easy interpolation.

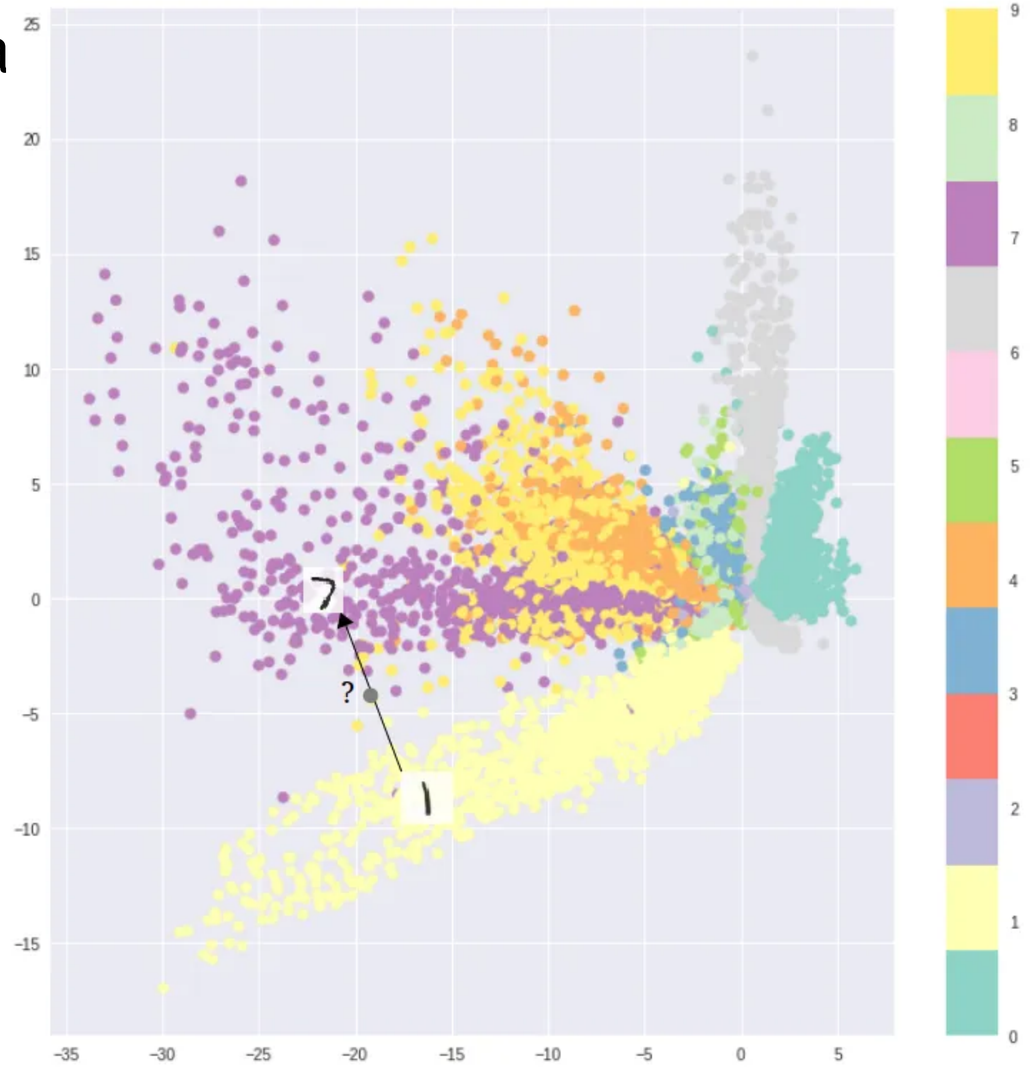


Image taken from https://towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf

Introduction : Variational Auto Encoders

- Variational Auto Encoders (**VAEs**) are a popular class of generative models.
  - Contrary to autoencoders, they learn to map the input to a **continuous latent representation**.

Image taken from https://en.wikipedia.org/wiki/Variational_autoencoder

Introduction : Variational Auto Encoders

- Variational Auto Encoders (**VAEs**) are a popular class of generative models.
  - Contrary to autoencoders, they learn to map the input to a **continuous latent representation**.

Image taken from https://en.wikipedia.org/wiki/Variational_autoencoder

Introduction : Variational Auto Encoders

- Variational Auto Encoders (**VAEs**) are a popular class of generative models.
  - Contrary to autoencoders, they learn to map the input to a **continuous latent representation**.

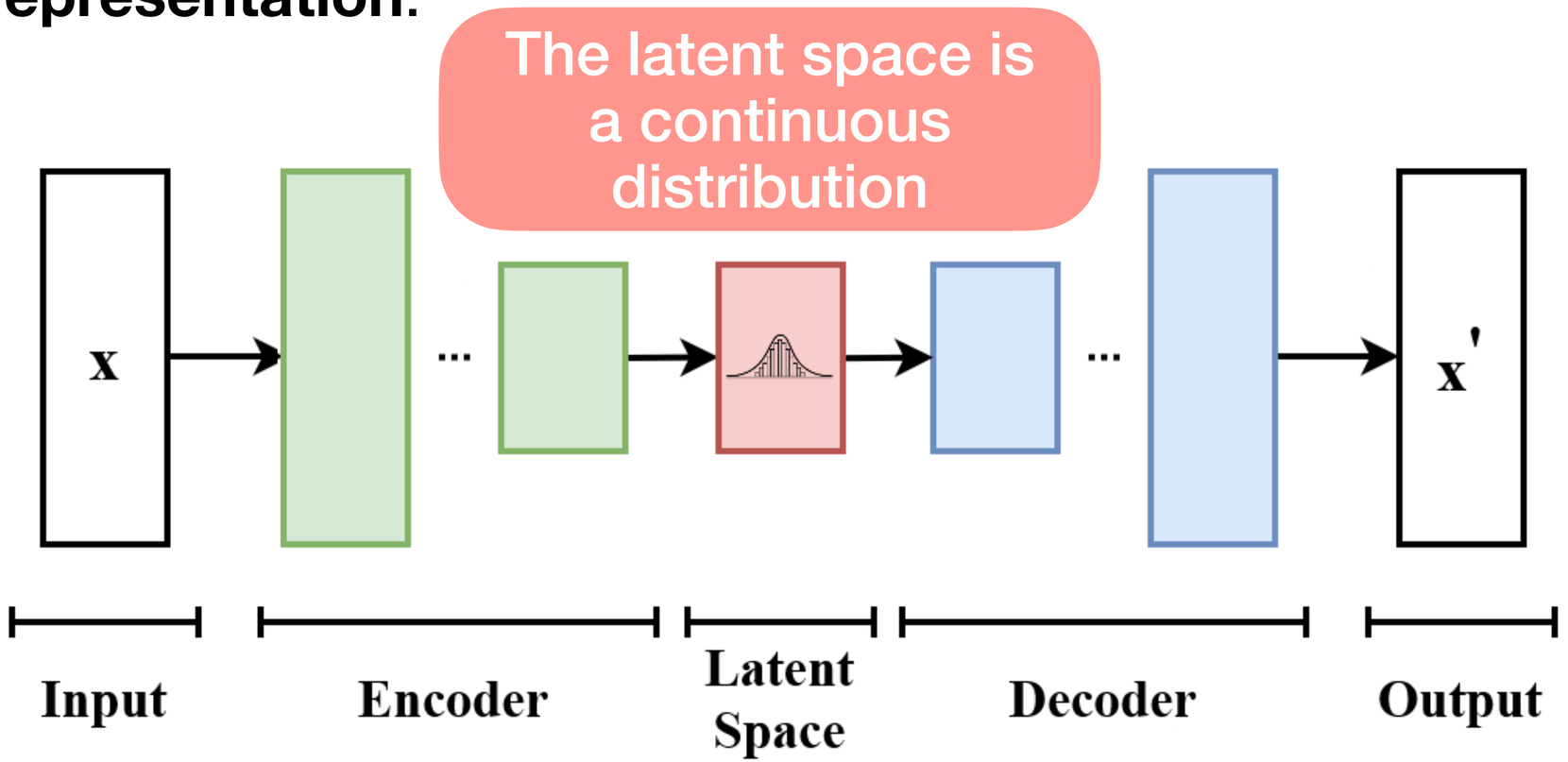


Image taken from https://en.wikipedia.org/wiki/Variational_autoencoder

Introduction : Variational Auto Encoders

- The encoder and decoder in a VAE are essentially modelling two different distributions.

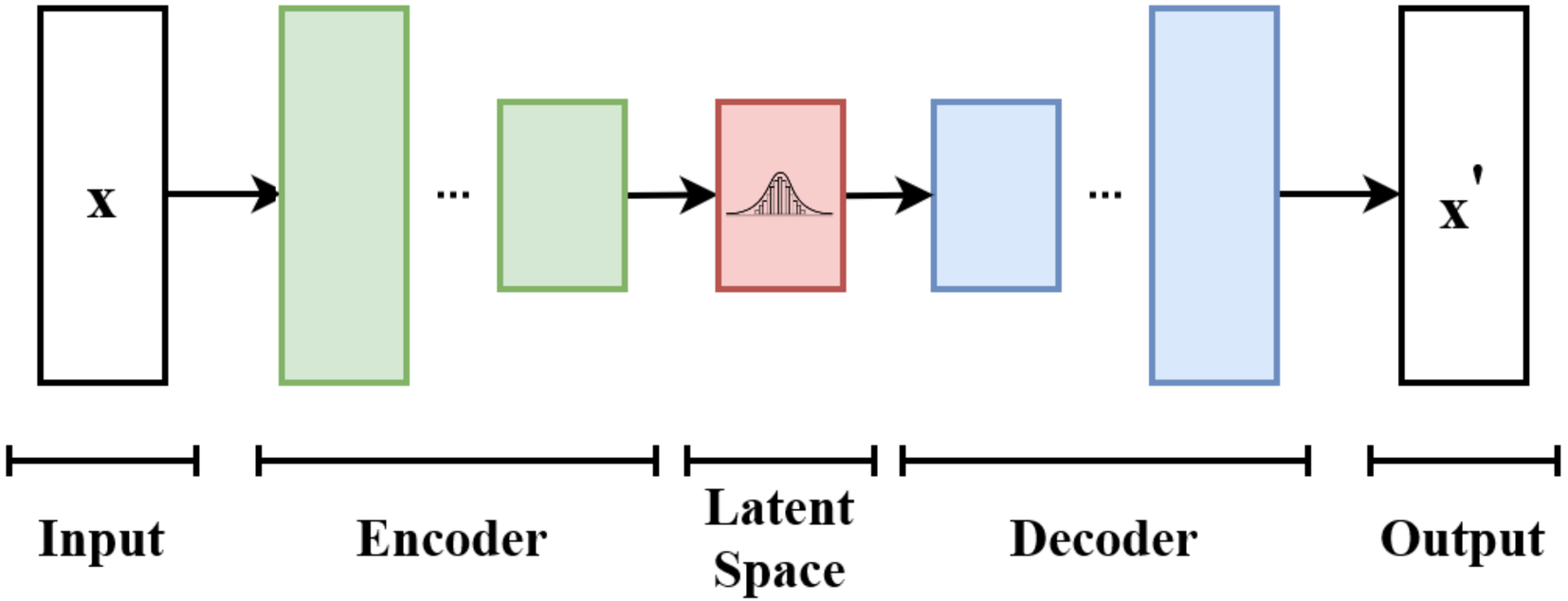


Image taken from https://en.wikipedia.org/wiki/Variational_autoencoder

Introduction : Variational Auto Encoders

- The encoder and decoder in a VAE are essentially modelling two different distributions.

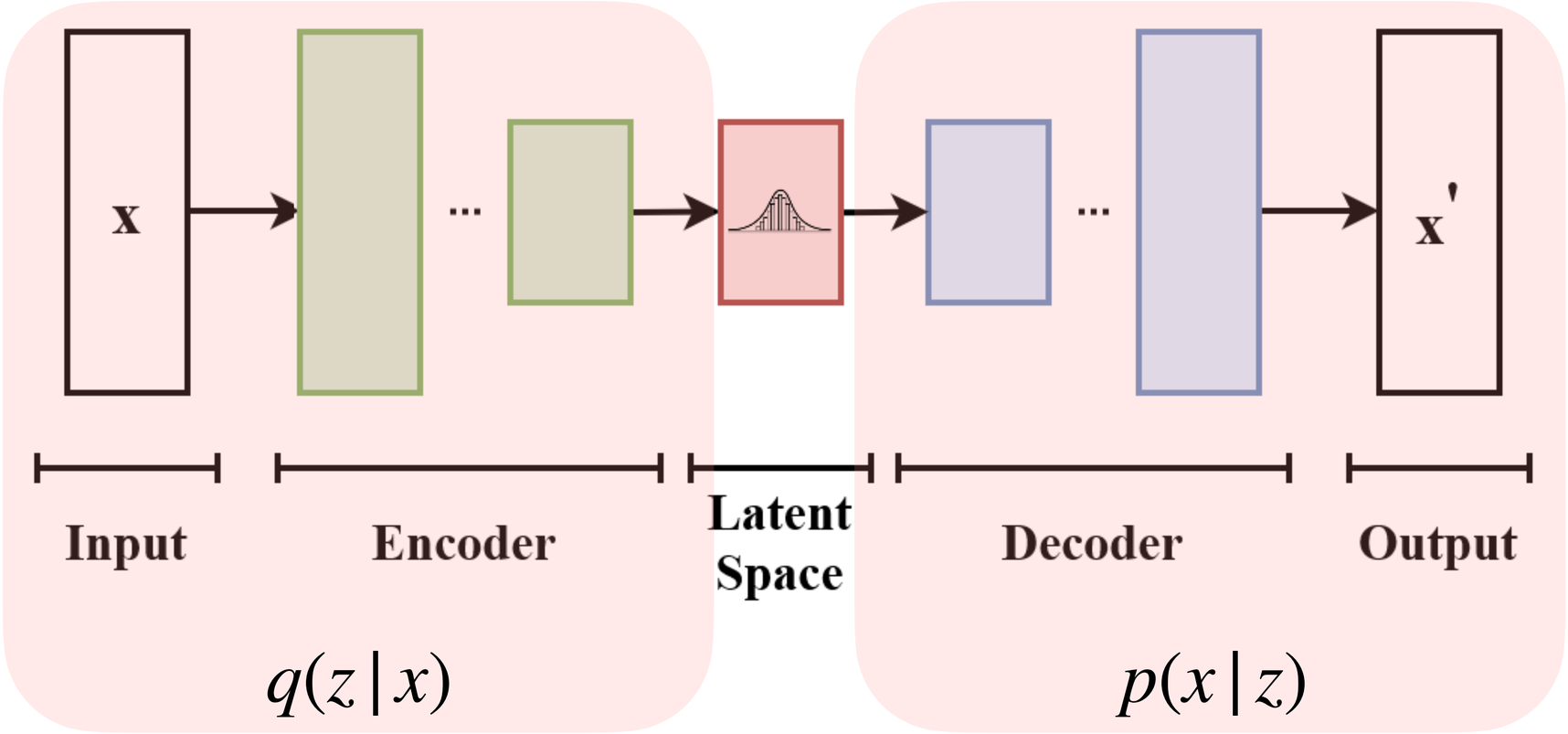


Image taken from https://en.wikipedia.org/wiki/Variational_autoencoder

Introduction : Variational Auto Encoders

- The encoder and decoder in a VAE are essentially modelling two different distributions.

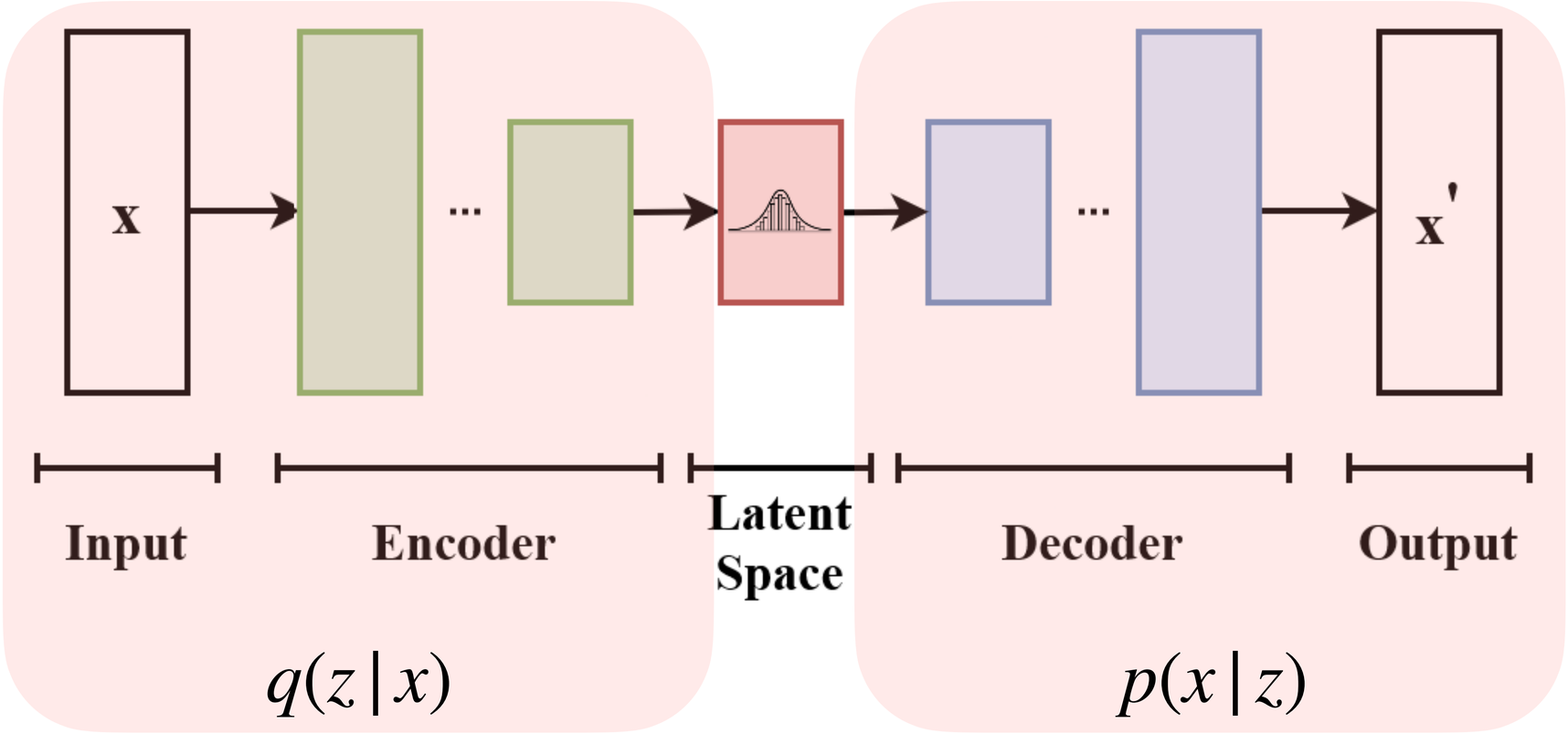


Image taken from https://en.wikipedia.org/wiki/Variational_autoencoder

Introduction : Variational Auto Encoders

- The encoder and decoder in a VAE are essentially modelling two different distributions.

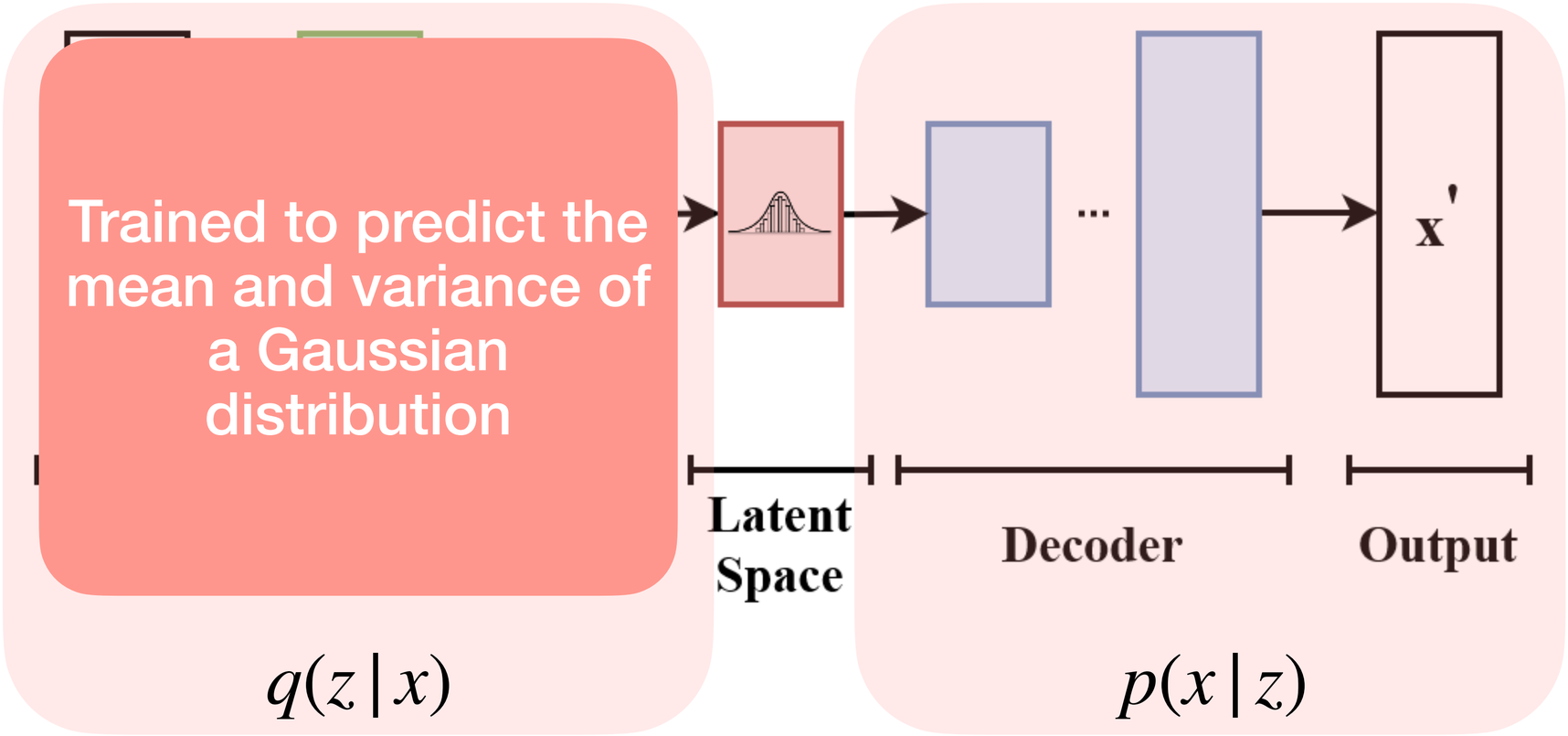


Image taken from https://en.wikipedia.org/wiki/Variational_autoencoder

Introduction : Variational Auto Encoders

- The encoder and decoder in a VAE are essentially modelling two different distributions.

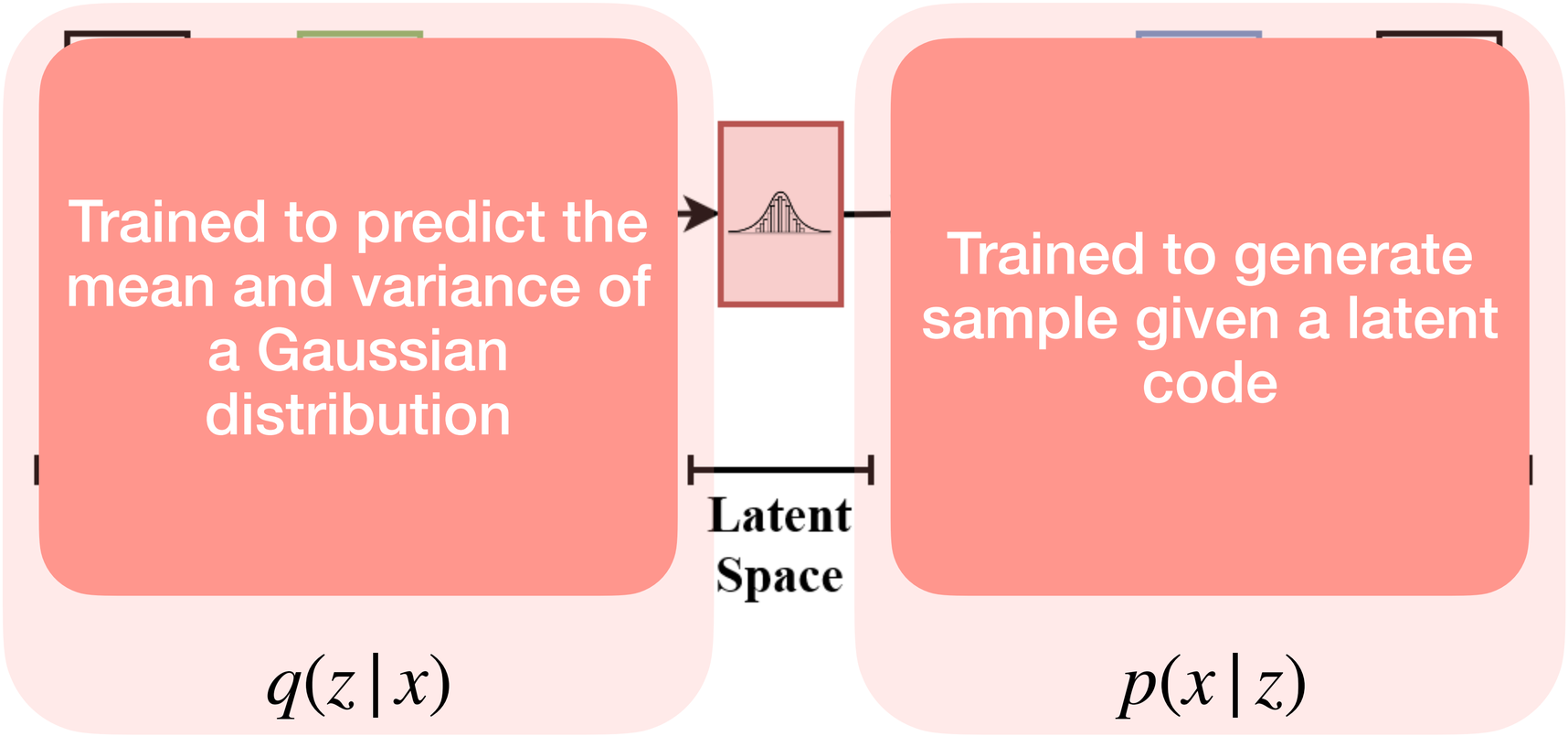


Image taken from https://en.wikipedia.org/wiki/Variational_autoencoder

Introduction : Variational Auto Encoders

- VAEs can also be conditioned on a label / class to generate specific output categories.

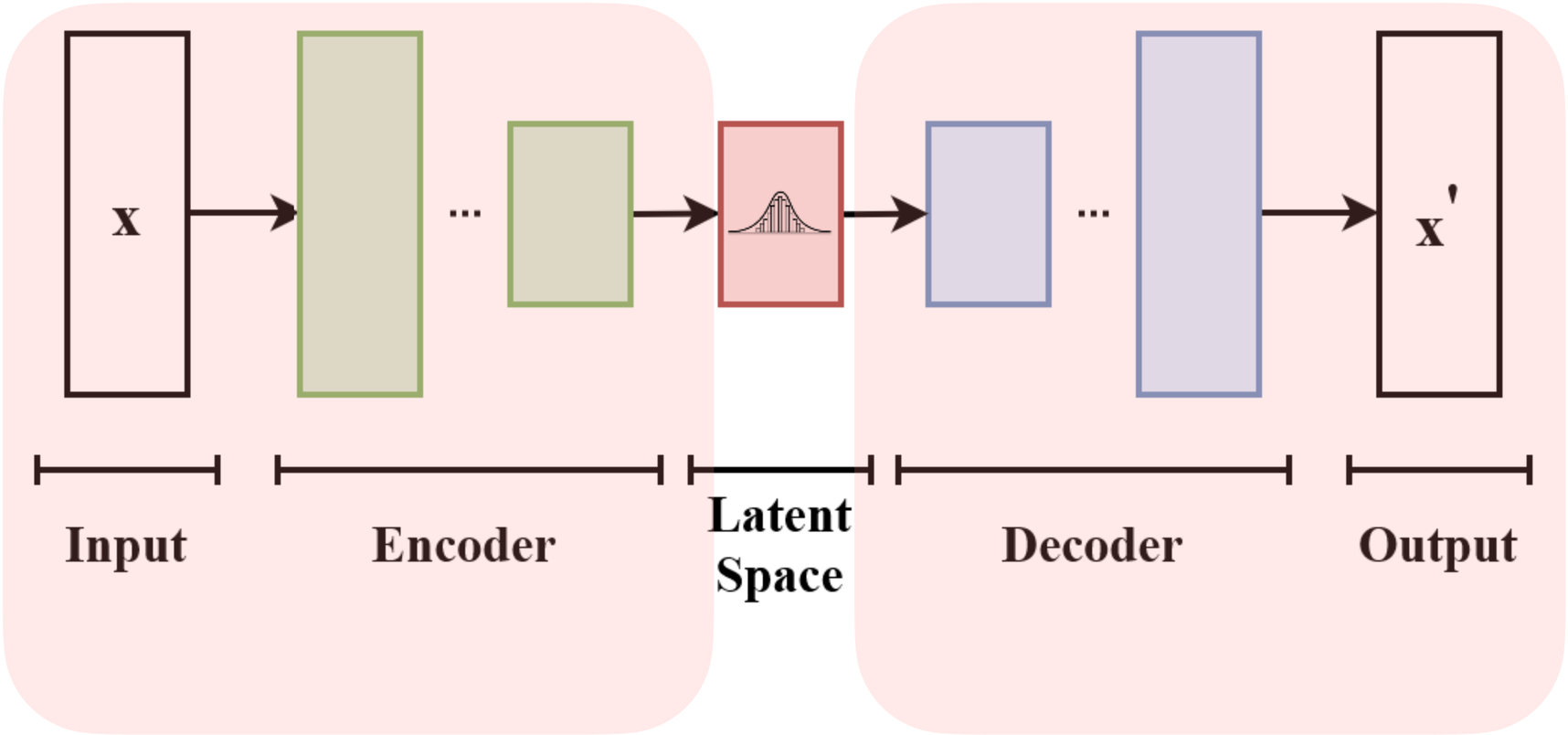


Image taken from https://en.wikipedia.org/wiki/Variational_autoencoder

Introduction : Variational Auto Encoders

- VAEs can also be conditioned on a label / class to generate specific output categories.

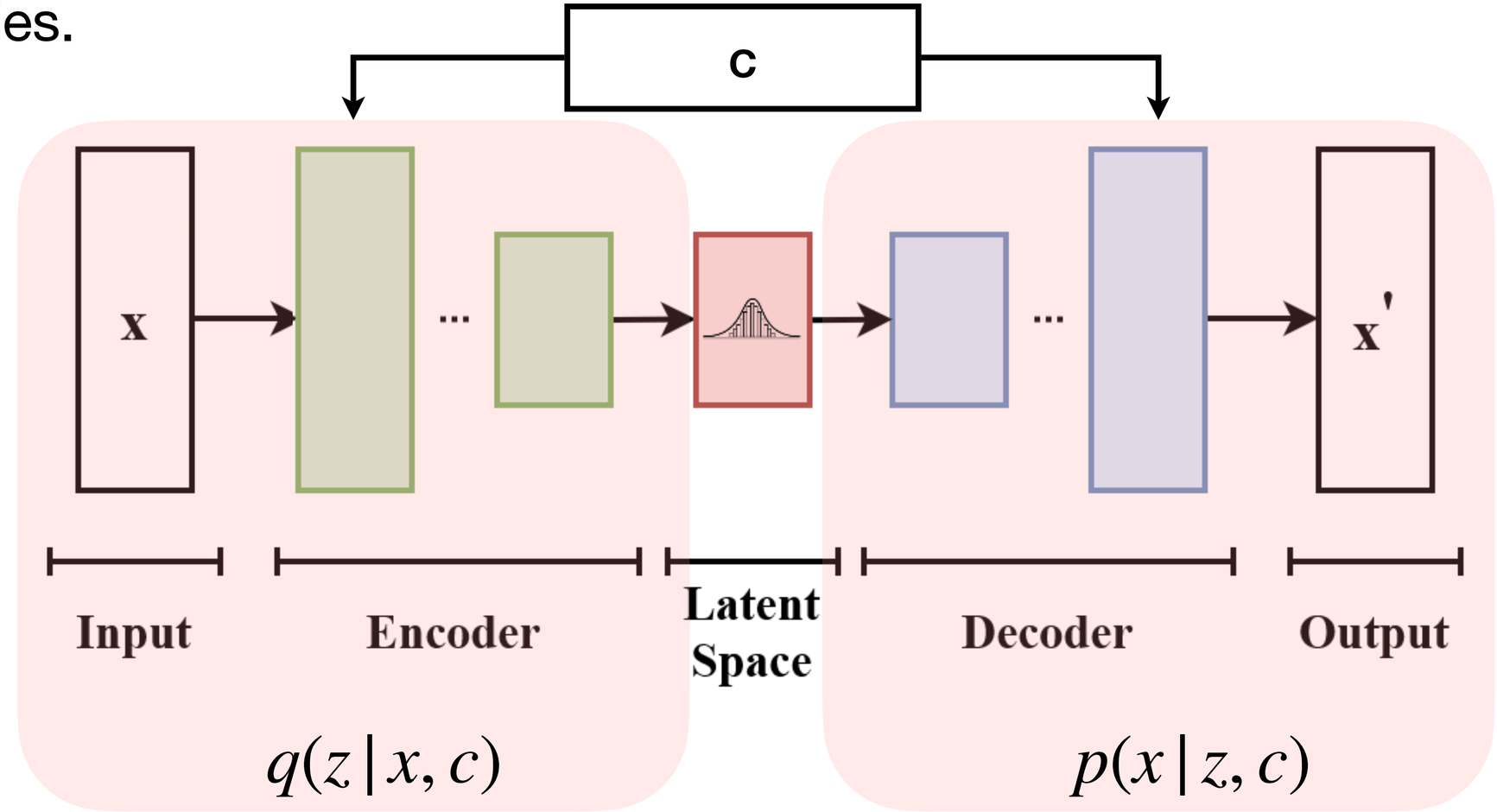
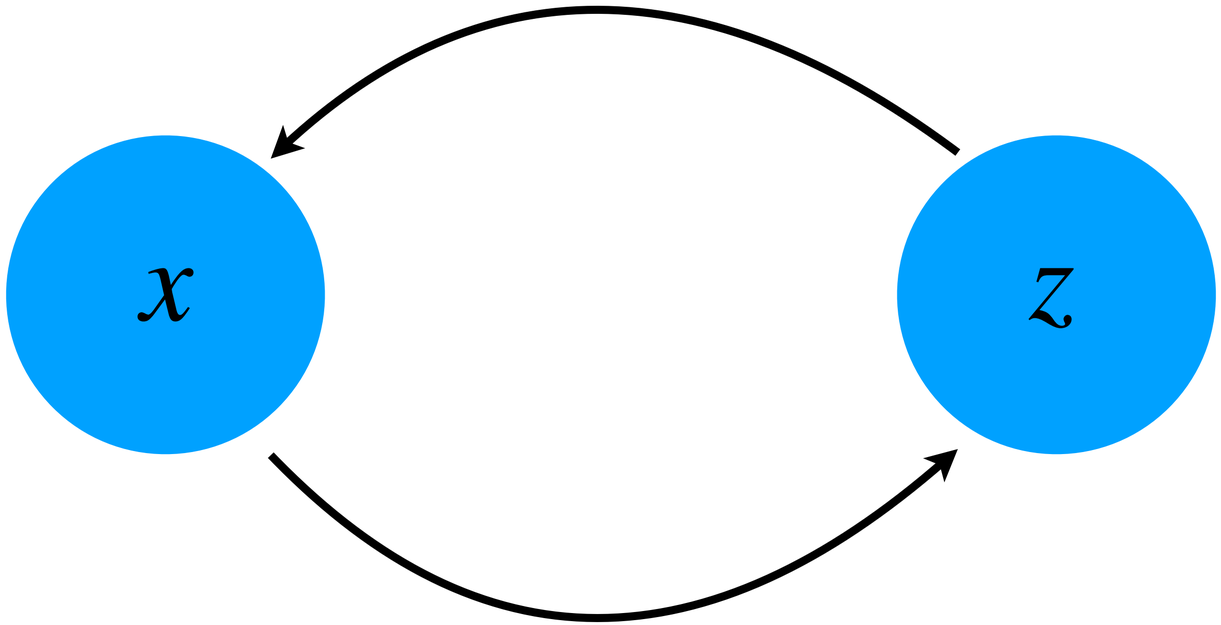


Image taken from https://en.wikipedia.org/wiki/Variational_autoencoder

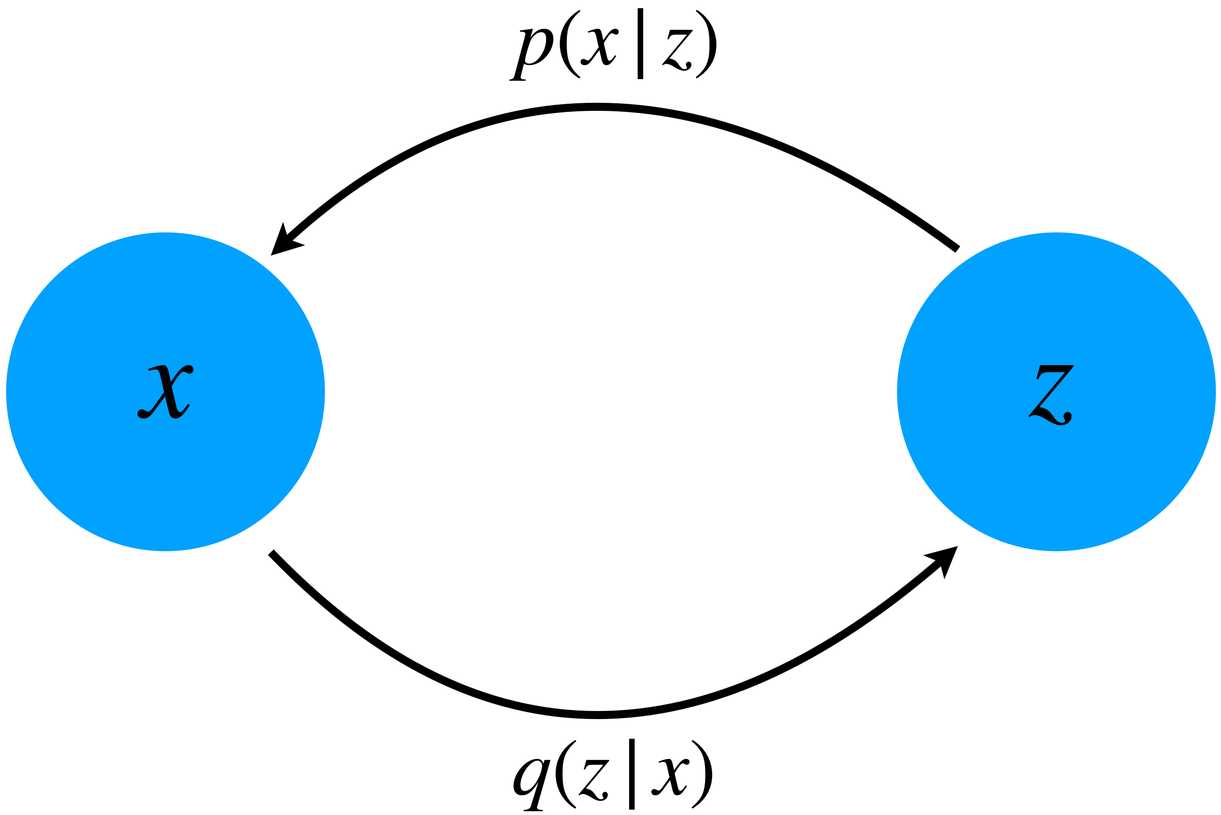
Introduction : Variational Auto Encoders

- VAEs can be represented as a cyclic graph.

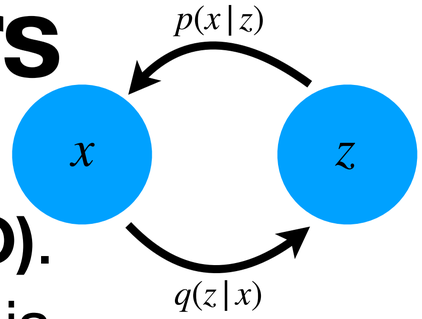


Introduction : Variational Auto Encoders

- VAEs can be represented as a cyclic graph.

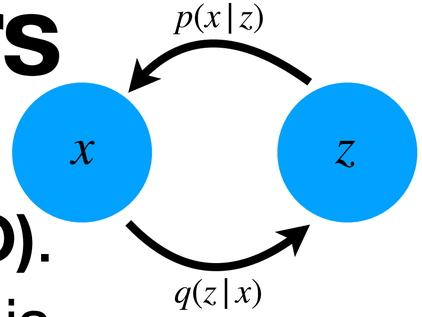


Introduction : Variational Auto Encoders



- Training VAEs involve maximizing the **E**vidence **L**ower **Bo**und **(ELBO)**.
  - This is because directly maximizing the likelihood of the data **p(x)** is difficult.

Introduction : Variational Auto Encoders

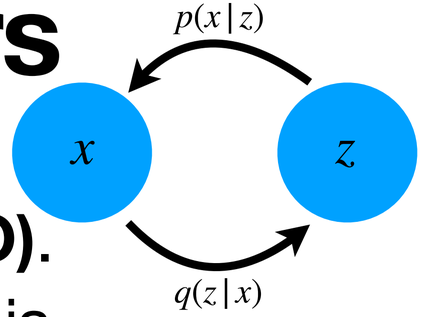


- Training VAEs involve maximizing the **E**vidence **L**ower **Bo**und **(ELBO)**.
  - This is because directly maximizing the likelihood of the data **p(x)** is difficult.

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

Introducing additional variable

Introduction : Variational Auto Encoders



- Training VAEs involve maximizing the **E**vidence **L**ower **Bo**und **(ELBO)**.
  - This is because directly maximizing the likelihood of the data **p(x)** is difficult.

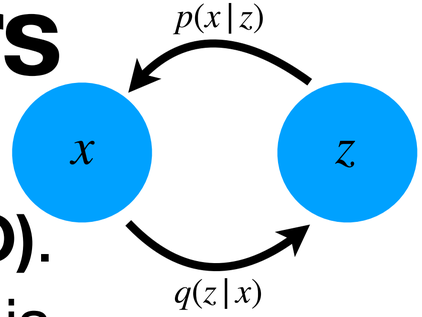
$$\log p(\boldsymbol{x}) = \log \int p(\boldsymbol{x}, \boldsymbol{z}) d\boldsymbol{z}$$

Introducing additional variable

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

Multiplying by 1

Introduction : Variational Auto Encoders



- Training VAEs involve maximizing the **E**vidence **L**ower **Bo**und **(ELBO)**.
  - This is because directly maximizing the likelihood of the data **p(x)** is difficult.

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

Introducing additional variable

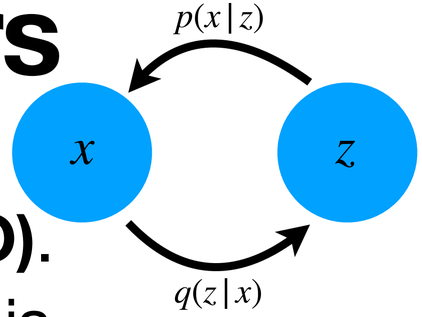
$$= \log \int \frac{p(\boldsymbol{x}, \boldsymbol{z}) q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} d\boldsymbol{z}$$

Multiplying by 1

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

Definition of Expectation

Introduction : Variational Auto Encoders



- Training VAEs involve maximizing the **E**vidence **L**ower **Bo**und **(ELBO)**.
  - This is because directly maximizing the likelihood of the data **p(x)** is difficult.

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

Introducing additional variable

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

Multiplying by 1

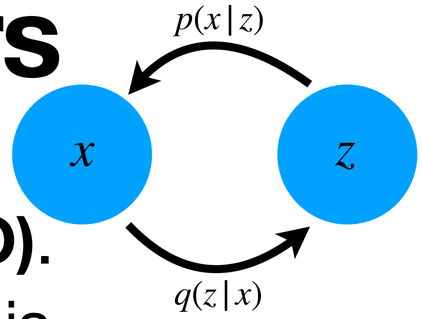
$$= \log \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \left[\frac{p(\boldsymbol{x}, \boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})} \right]$$

Definition of Expectation

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

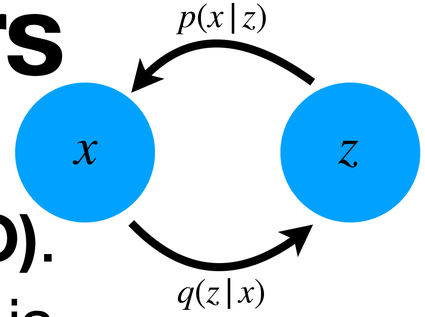
Jensen’s Inequality

Introduction : Variational Auto Encoders



- Training VAEs involve maximizing the **E**vidence **L**ower **Bo**und **(ELBO)**.
  - This is because directly maximizing the likelihood of the data **p(x)** is difficult.

Introduction : Variational Auto Encoders

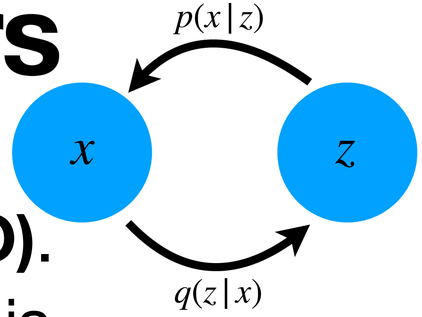


- Training VAEs involve maximizing the **E**vidence **L**ower **Bo**und **(ELBO)**.
  - This is because directly maximizing the likelihood of the data **p(x)** is difficult.

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

Chain Rule

Introduction : Variational Auto Encoders



- Training VAEs involve maximizing the **E**vidence **L**ower **Bo**und **(ELBO)**.
  - This is because directly maximizing the likelihood of the data **p(x)** is difficult.

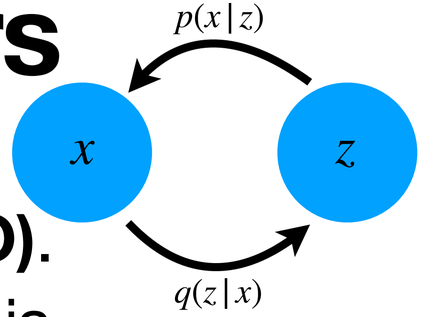
$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x},\boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\right] = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{q_{\boldsymbol{\phi}}(\bolds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

Chain Rule

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

Split Expectation

Introduction : Variational Auto Encoders



- Training VAEs involve maximizing the **E**vidence **L**ower **Bo**und **(ELBO)**.
  - This is because directly maximizing the likelihood of the data **p(x)** is difficult.

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

Chain Rule

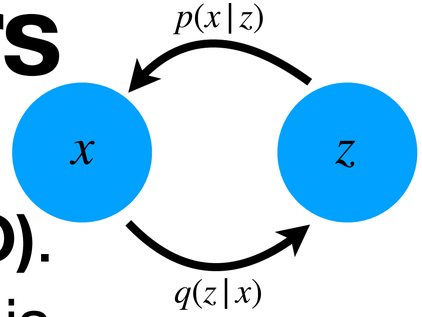
$$= \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right] + \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{z})}{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\right]$$

Split Expectation

$$= \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right]}_{\text{reconstruction term}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z}))}_{\text{prior matching term}}$$

Definition of KL Divergence

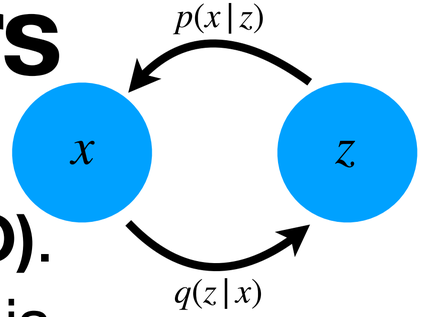
Introduction : Variational Auto Encoders



- Training VAEs involve maximizing the **E**vidence **L**ower **Bo**und **(ELBO)**.
  - This is because directly maximizing the likelihood of the data **p(x)** is difficult.

$$= \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right]}_{\text{reconstruction term}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z}))}_{\text{prior matching term}}$$

Introduction : Variational Auto Encoders

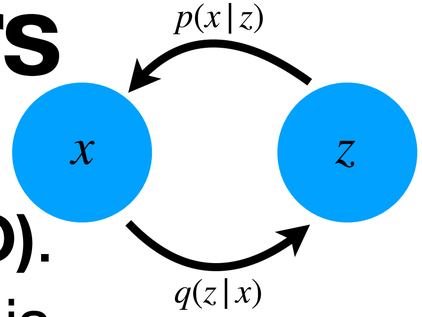


- Training VAEs involve maximizing the **E**vidence **L**ower **Bo**und **(ELBO)**.
  - This is because directly maximizing the likelihood of the data **p(x)** is difficult.

$$= \underbrace{\mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\right]}_{\text{reconstruction term}} - \underbrace{D_{\mathrm{KL}}(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z}))}_{\text{prior matching term}}$$

Reconstruction term that ensures the decoder regenerates the input.

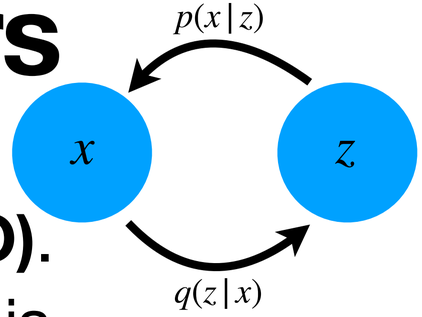
Introduction : Variational Auto Encoders



- Training VAEs involve maximizing the **E**vidence **L**ower **Bo**und **(ELBO)**.
  - This is because directly maximizing the likelihood of the data **p(x)** is difficult.

$$= \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right]}_{\text{reconstruction term}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z}))}_{\text{prior matching term}}$$

Introduction : Variational Auto Encoders



- Training VAEs involve maximizing the **E**vidence **L**ower **Bo**und **(ELBO)**.
  - This is because directly maximizing the likelihood of the data **p(x)** is difficult.

$$= \underbrace{\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right]}_{\text{reconstruction term}} - \underbrace{D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\boldsymbol{z}))}_{\text{prior matching term}}$$

Regularization to ensure that the Gaussian parameters output by the encoder are close to a standard Gaussian distribution

Introduction : Variational Auto Encoders

- As VAEs learn a **continuous latent representation**, it can be used for data generation.

Image taken from https://mbernste.github.io/posts/vae/

Introduction : Variational Auto Encoders

- As VAEs learn a **continuous latent representation**, it can be used for data generation.

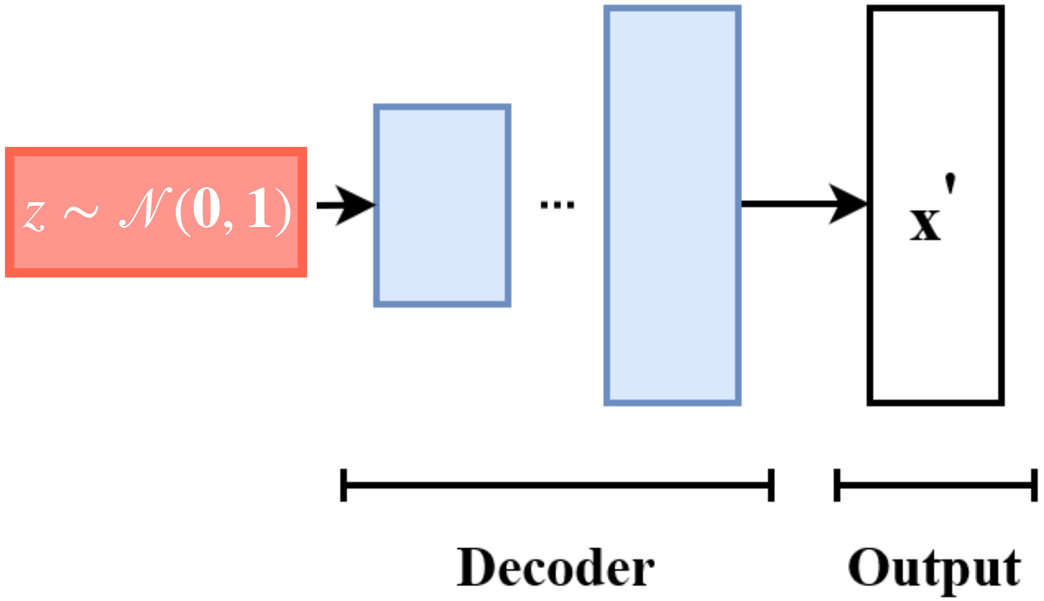
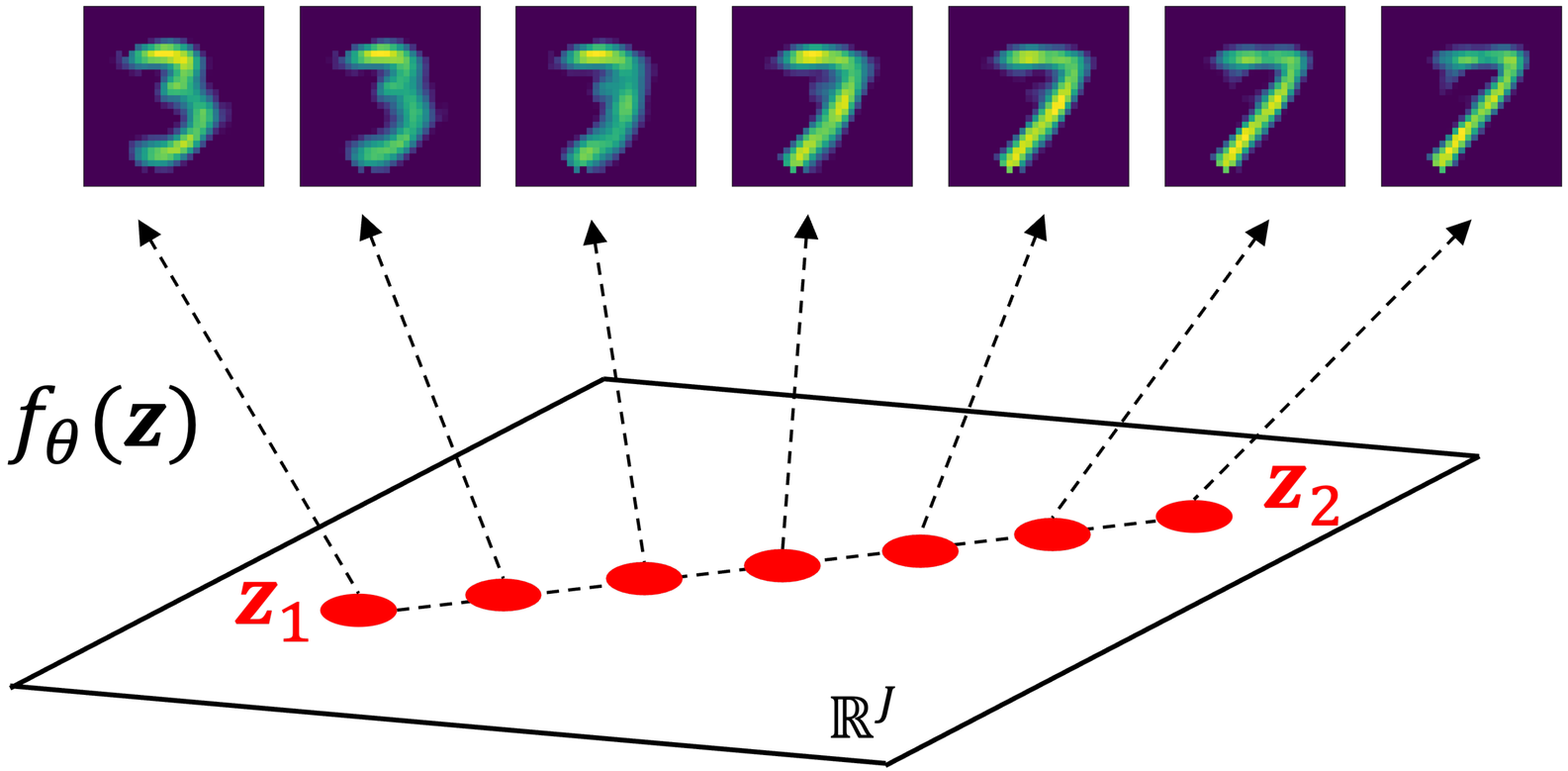


Image taken from https://mbernste.github.io/posts/vae/

Introduction : Variational Auto Encoders

- As VAEs learn a **continuous latent representation**, it can be used for data generation.



Continuous latent space enables data generation from any point

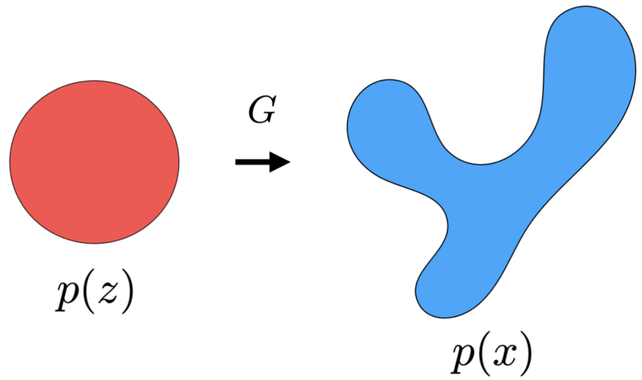
Image taken from https://mbernste.github.io/posts/vae/

Introduction : Variational Auto Encoders

- **VAE** effectively defines a mapping ***G*** from a simple (Gaussian) latent distributions ***p***(***z***) to an arbitrarily complex data distribution ***p***(***x***)

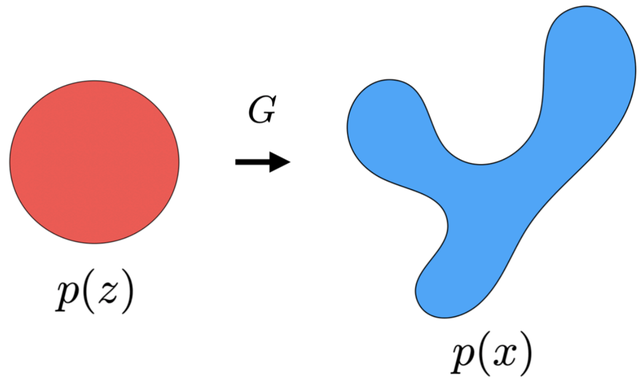
Introduction : Variational Auto Encoders

- **VAE** effectively defines a mapping ***G*** from a simple (Gaussian) latent distributions ***p***(***z***) to an arbitrarily complex data distribution ***p***(***x***)



Introduction : Variational Auto Encoders

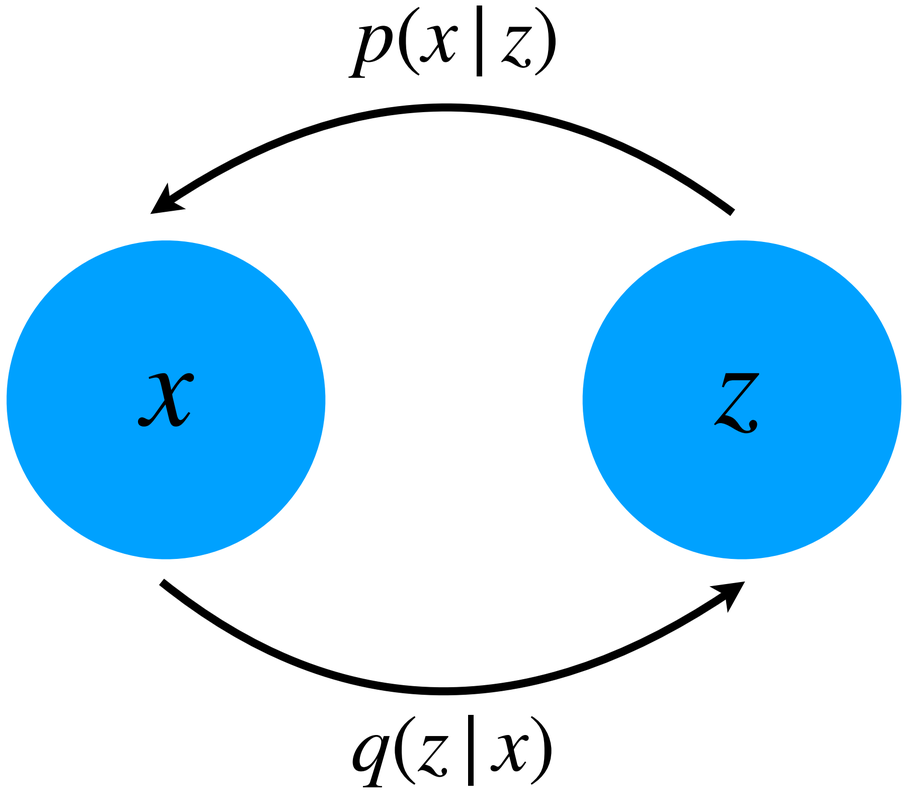
- **VAE** effectively defines a mapping ***G*** from a simple (Gaussian) latent distributions ***p***(***z***) to an arbitrarily complex data distribution ***p***(***x***)



- **Diffusion model** does something very similar, but through a series of parametrized transformations, ensuring that each transformation can be simpler and more complex data distributions can be handled more easily as a result

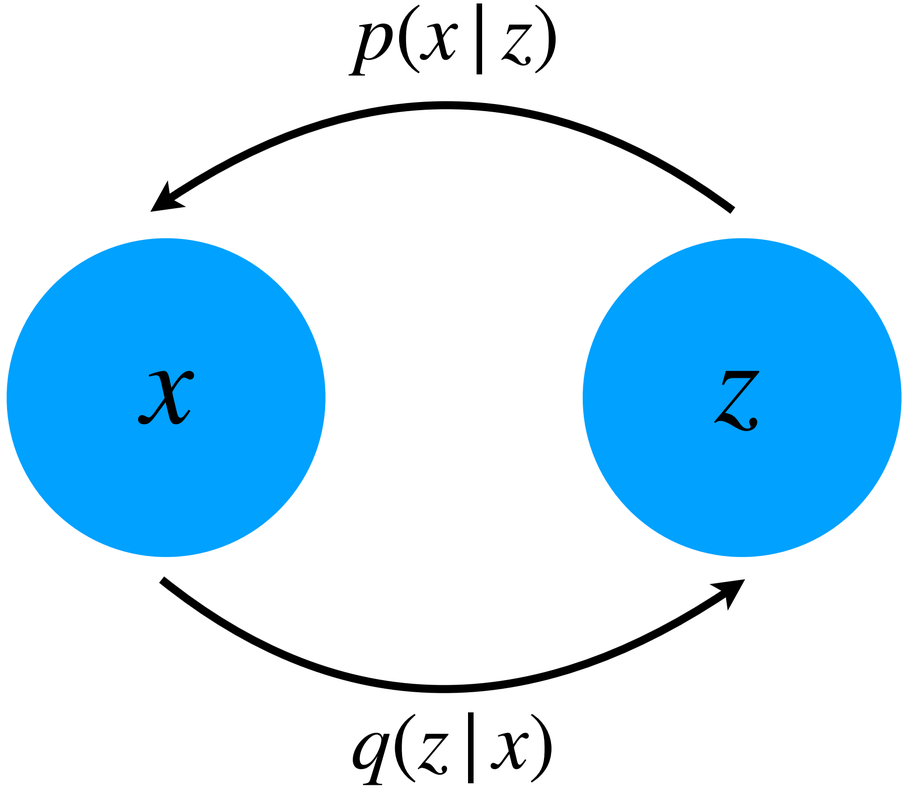
Variational Diffusion Models

- The easiest way to think about a diffusion model is as a Markovian hierarchical variational autoencoder with some restrictions.



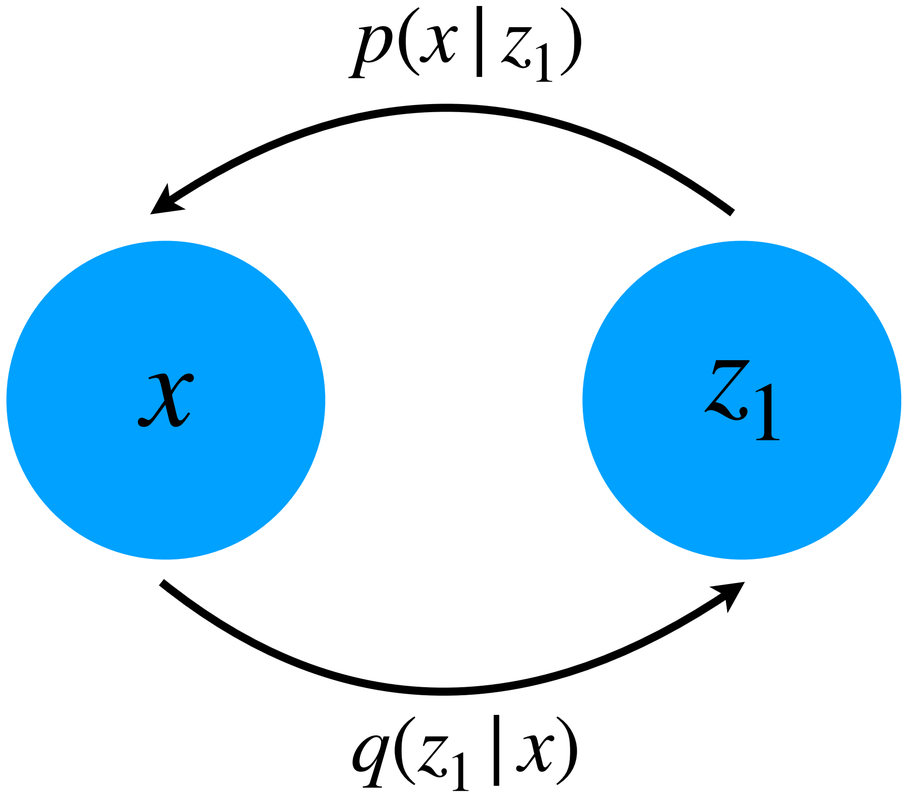
Variational Diffusion Models

- The easiest way to think about a diffusion model is as a Markovian hierarchical variational autoencoder with some restrictions.



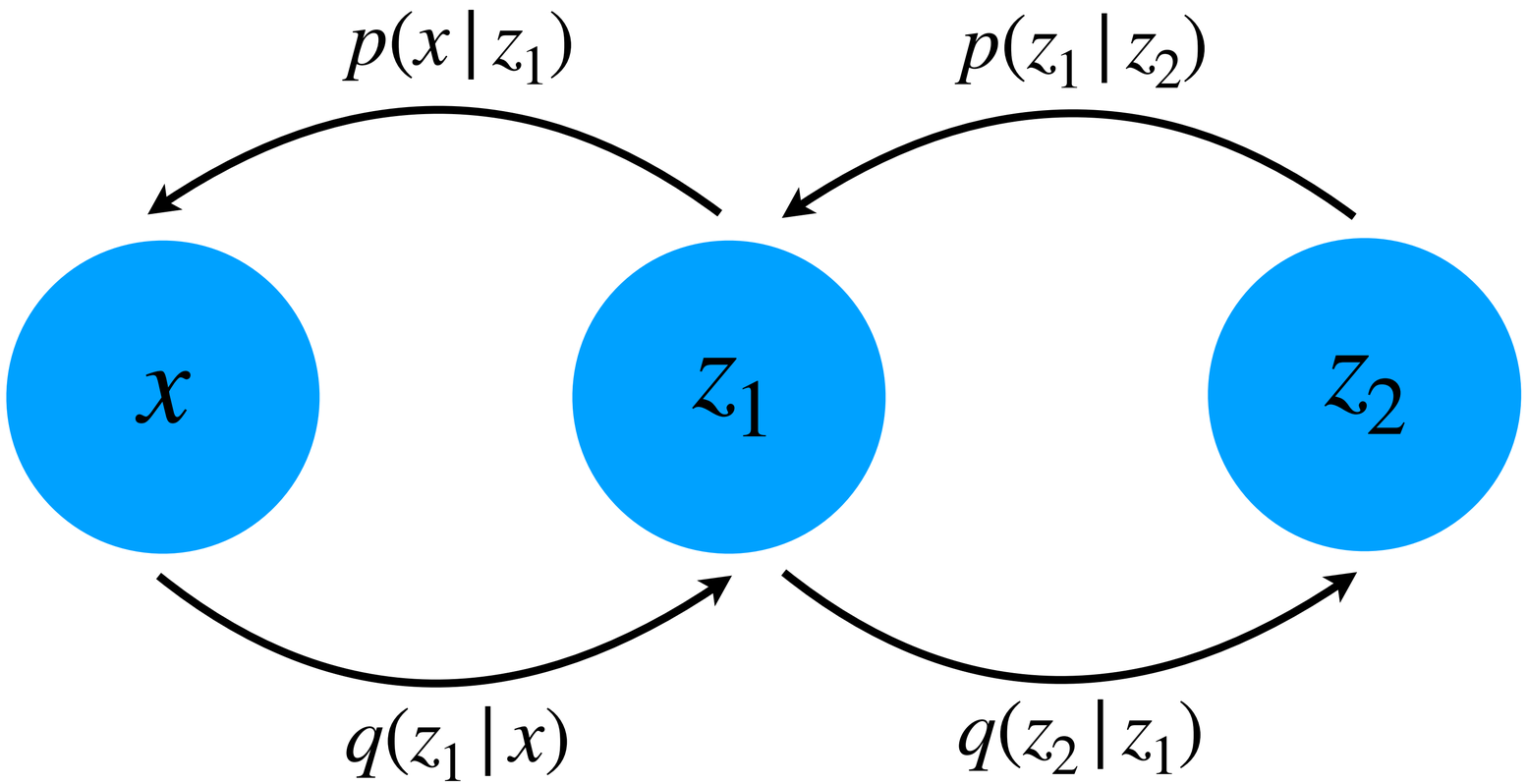
Variational Diffusion Models

- The easiest way to think about a diffusion model is as a Markovian hierarchical variational autoencoder with some restrictions.



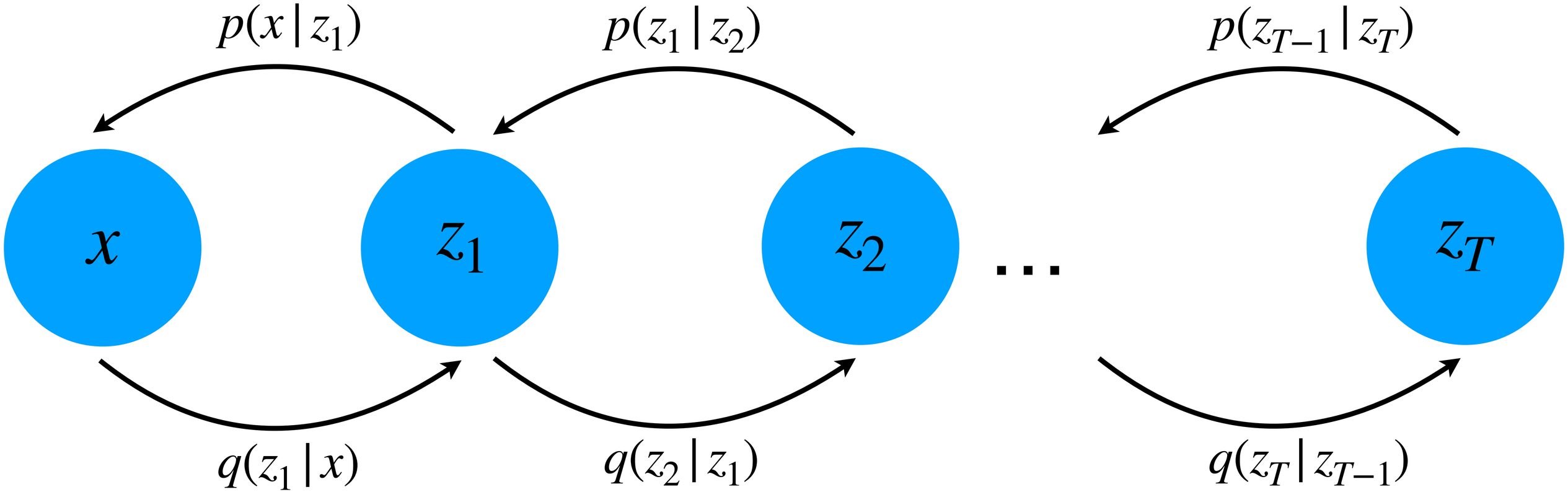
Variational Diffusion Models

- The easiest way to think about a diffusion model is as a Markovian hierarchical variational autoencoder with some restrictions.



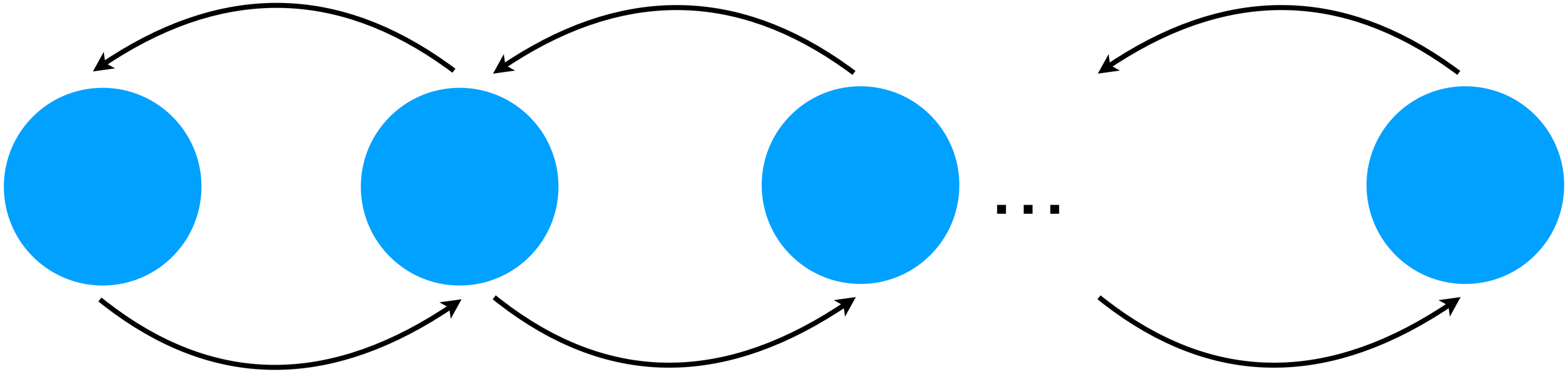
Variational Diffusion Models

- The easiest way to think about a diffusion model is as a Markovian hierarchical variational autoencoder with some restrictions.



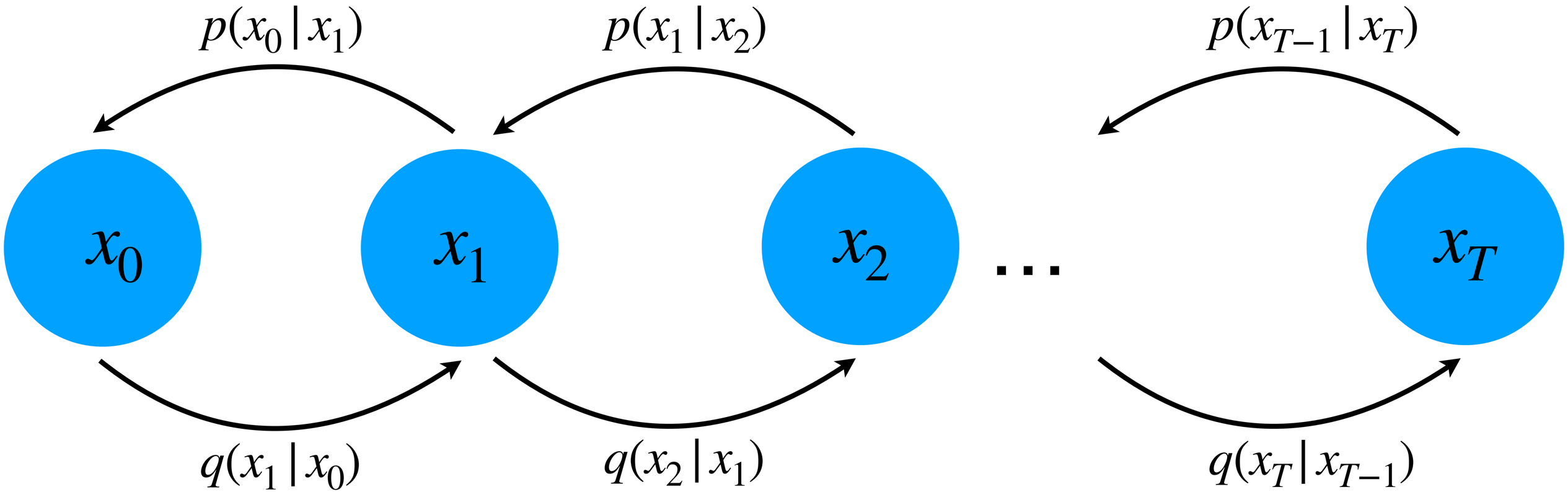
Variational Diffusion Models

- The easiest way to think about a diffusion model is as a Markovian hierarchical variational autoencoder with some restrictions.
  - All latent dimensions **z** are **the same** as the data dimension **x**



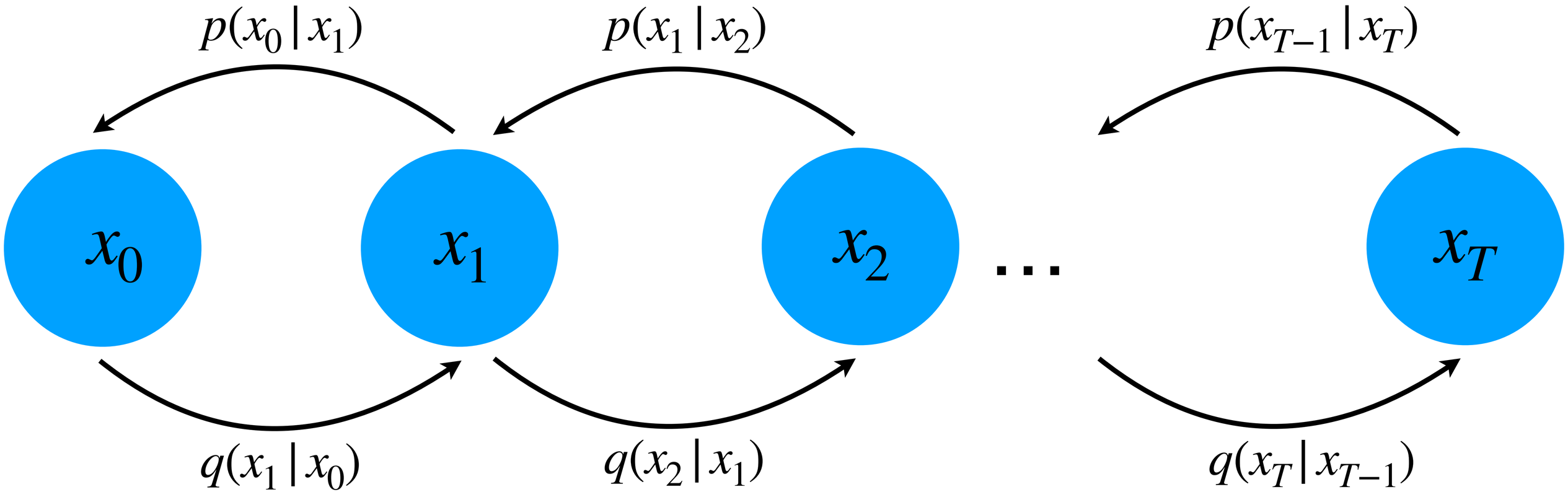
Variational Diffusion Models

- The easiest way to think about a diffusion model is as a Markovian hierarchical variational autoencoder with some restrictions.
  - All latent dimensions **z** are **the same** as the data dimension **x**



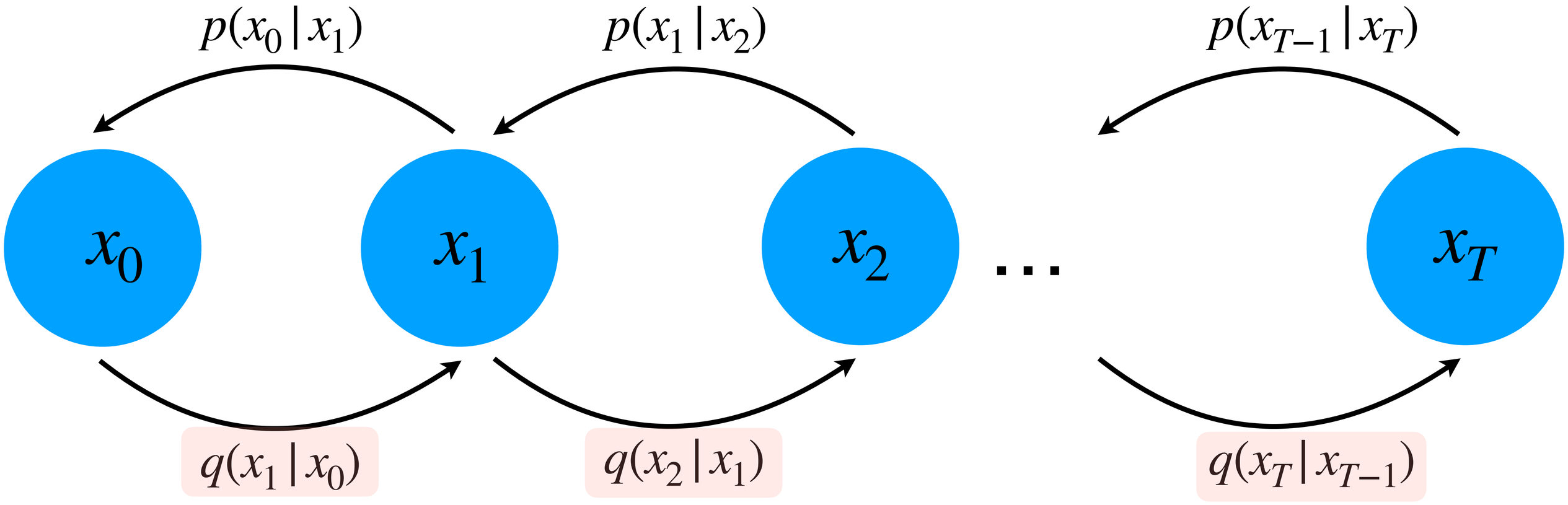
Variational Diffusion Models

- The easiest way to think about a diffusion model is as a Markovian hierarchical variational autoencoder with some restrictions.
  - Encoder at each step is **not learned;** predefined as a Gaussian distribution centered around the output of the previous step.



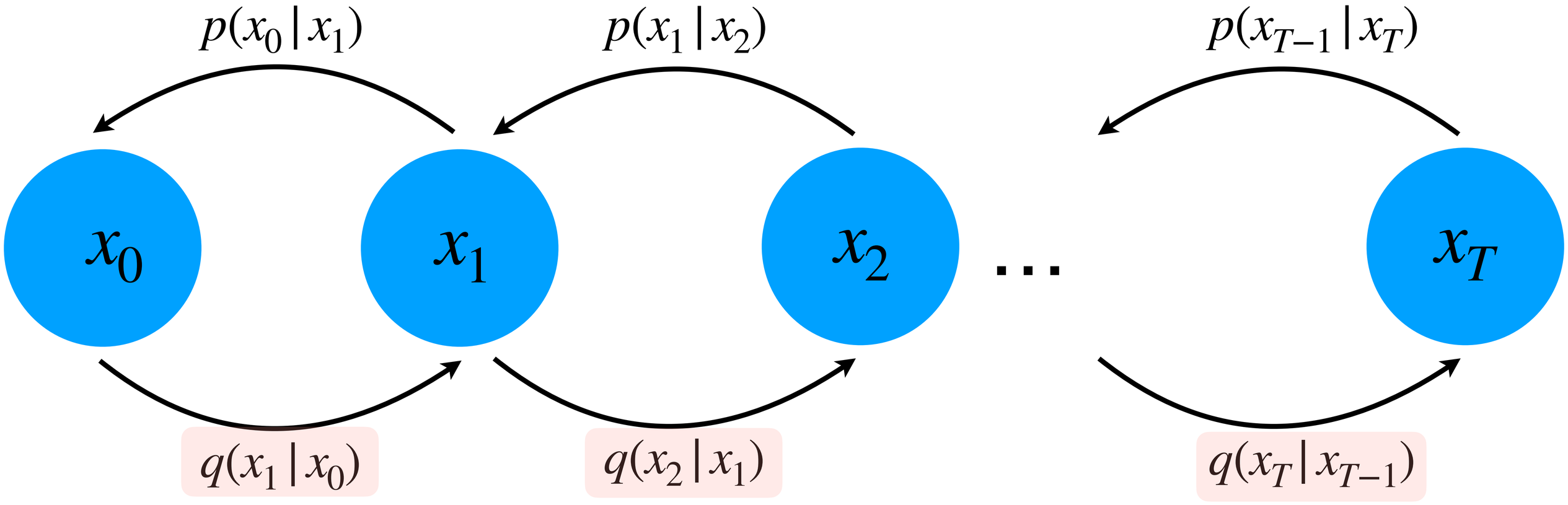
Variational Diffusion Models

- The easiest way to think about a diffusion model is as a Markovian hierarchical variational autoencoder with some restrictions.
  - Encoder at each step is **not learned;** predefined as a Gaussian distribution centered around the output of the previous step.



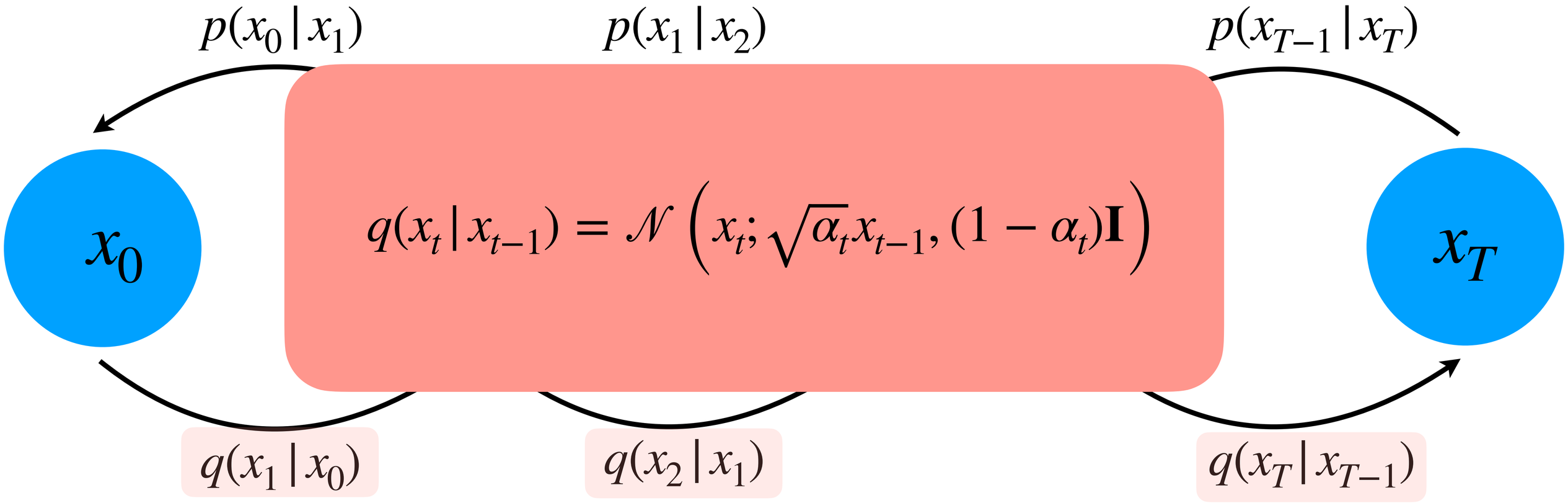
Variational Diffusion Models

- The easiest way to think about a diffusion model is as a Markovian hierarchical variational autoencoder with some restrictions.
  - Encoder at each step is **not learned;** predefined as a Gaussian distribution centered around the output of the previous step.



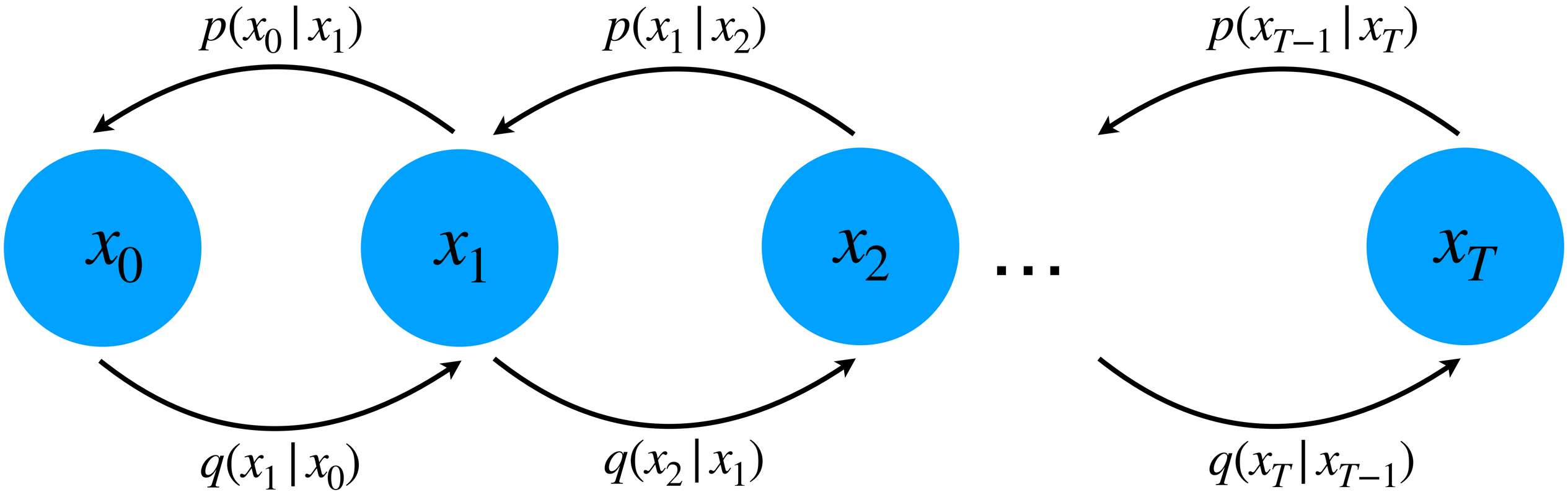
Variational Diffusion Models

- The easiest way to think about a diffusion model is as a Markovian hierarchical variational autoencoder with some restrictions.
  - Encoder at each step is **not learned;** predefined as a Gaussian distribution centered around the output of the previous step.



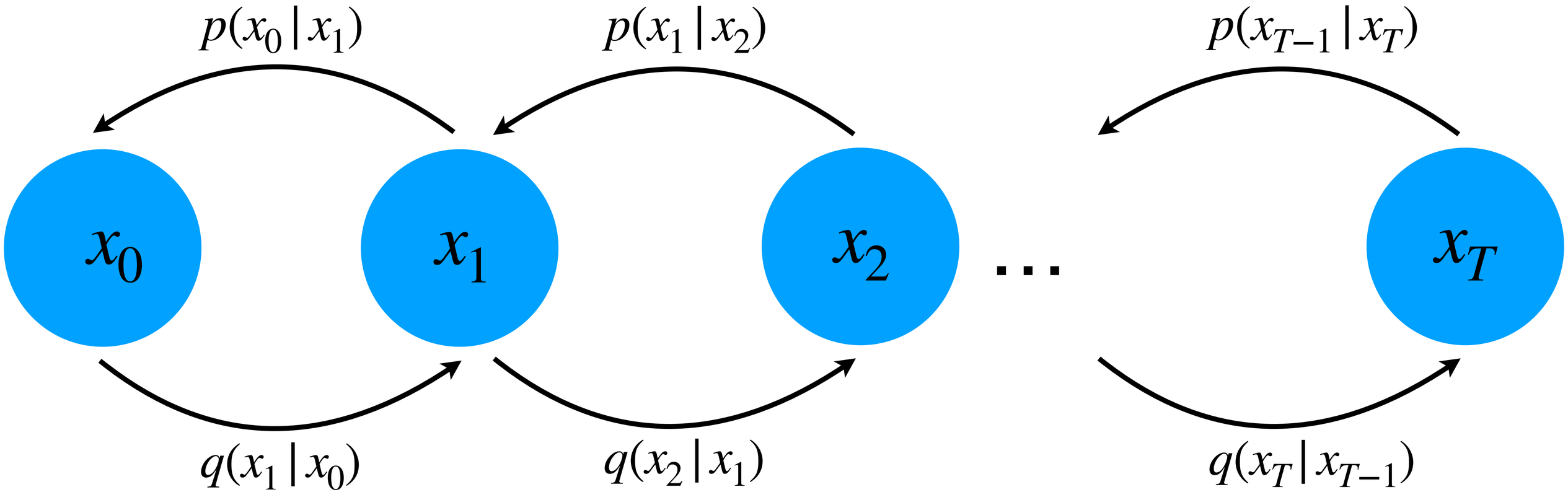
Variational Diffusion Models

- The easiest way to think about a diffusion model is as a Markovian hierarchical variational autoencoder with some restrictions.
  - The Gaussian parameters of the encoder are set in a way such that the distribution at the final timestep **T** is a standard Gaussian



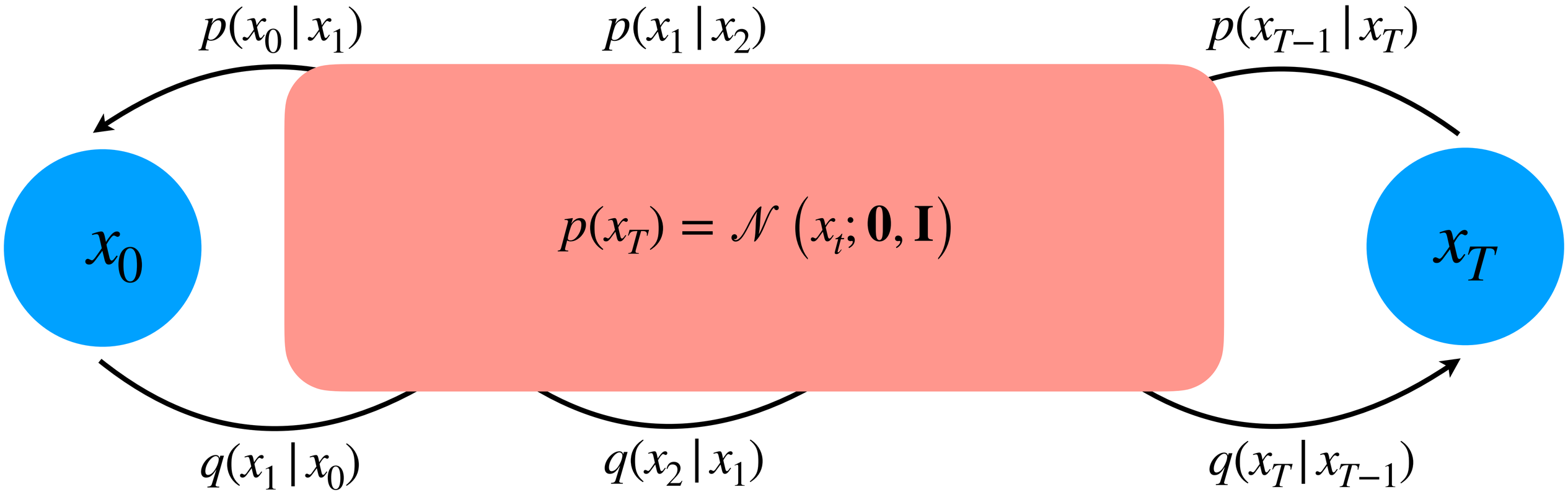
Variational Diffusion Models

- The easiest way to think about a diffusion model is as a Markovian hierarchical variational autoencoder with some restrictions.
  - The Gaussian parameters of the encoder are set in a way such that the distribution at the final timestep **T** is a standard Gaussian



Variational Diffusion Models

- The easiest way to think about a diffusion model is as a Markovian hierarchical variational autoencoder with some restrictions.
  - The Gaussian parameters of the encoder are set in a way such that the distribution at the final timestep **T** is a standard Gaussian



Variational Diffusion Models

- Diffusion models can be split into two processes.

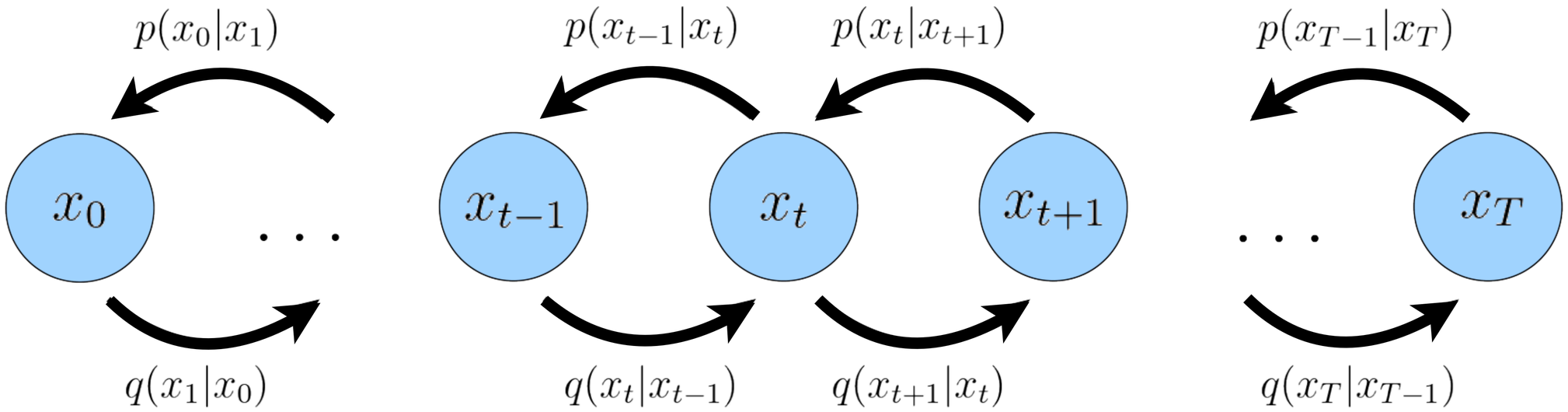


Image taken from https://arxiv.org/pdf/2208.11970

Variational Diffusion Models

- Diffusion models can be split into two processes.
  - **Forward Process:** Progressively corrupt an input by adding Gaussian noise until it becomes completely identical to pure Gaussian noise. **Not Learned.**

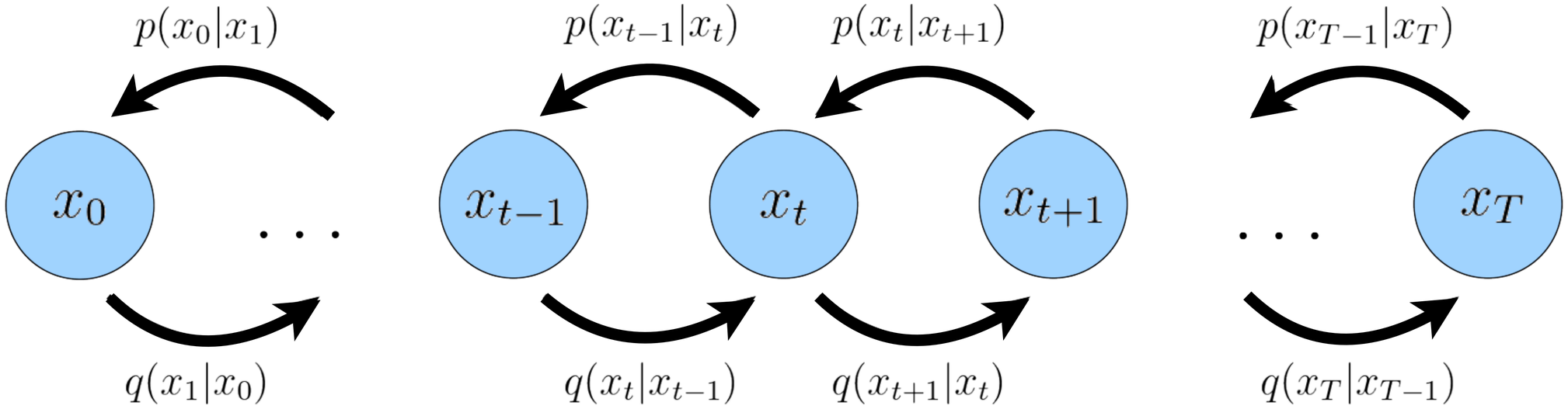


Image taken from https://arxiv.org/pdf/2208.11970

Variational Diffusion Models

- Diffusion models can be split into two processes.
  - **Forward Process:** Progressively corrupt an input by adding Gaussian noise until it becomes completely identical to pure Gaussian noise. **Not Learned.**

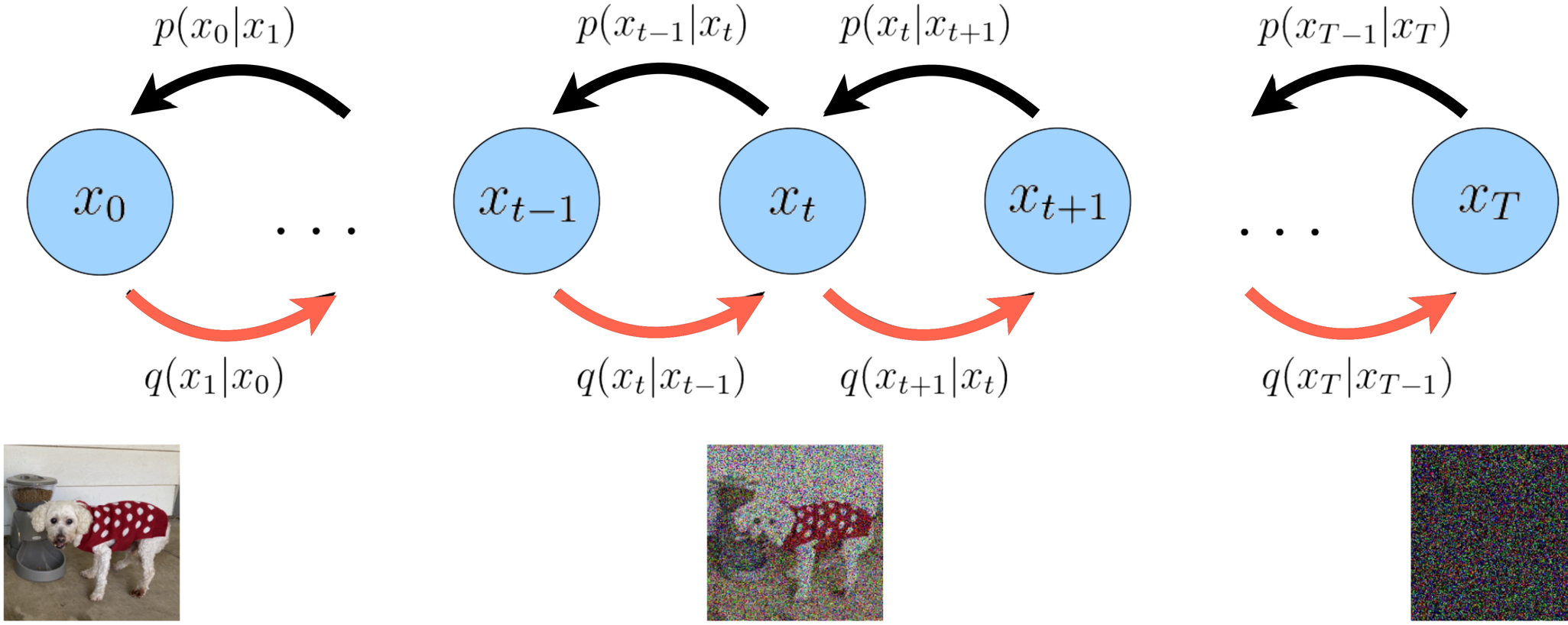


Image taken from https://arxiv.org/pdf/2208.11970

Variational Diffusion Models

- Diffusion models can be split into two processes.
  - **Reverse Process:** Noise is transformed back into a sample from the target distribution. **Learned**.

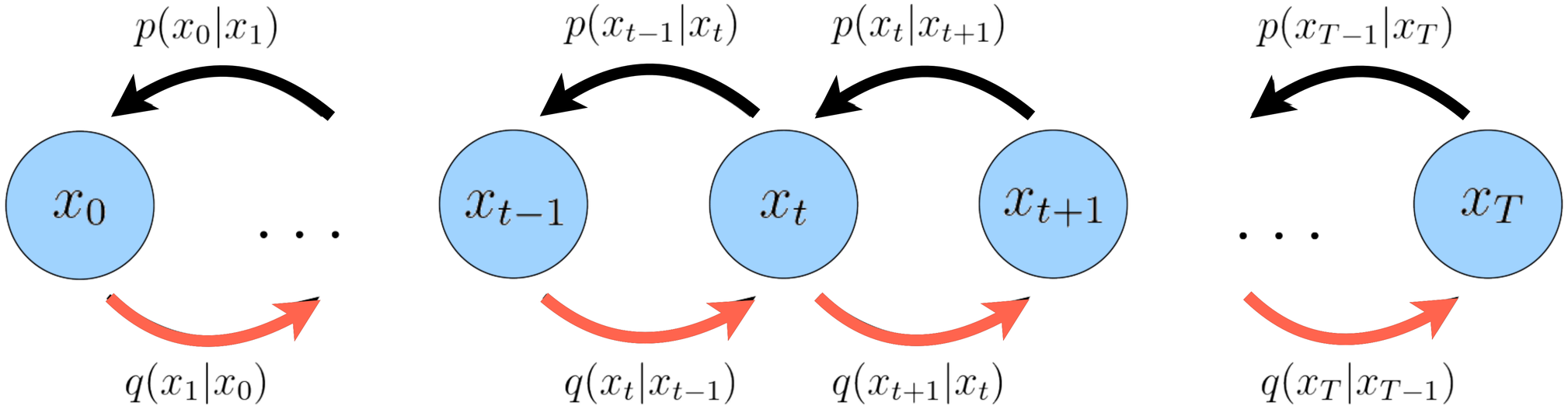


Image taken from https://arxiv.org/pdf/2208.11970

Variational Diffusion Models

- Diffusion models can be split into two processes.
  - **Reverse Process:** Noise is transformed back into a sample from the target distribution. **Learned**.

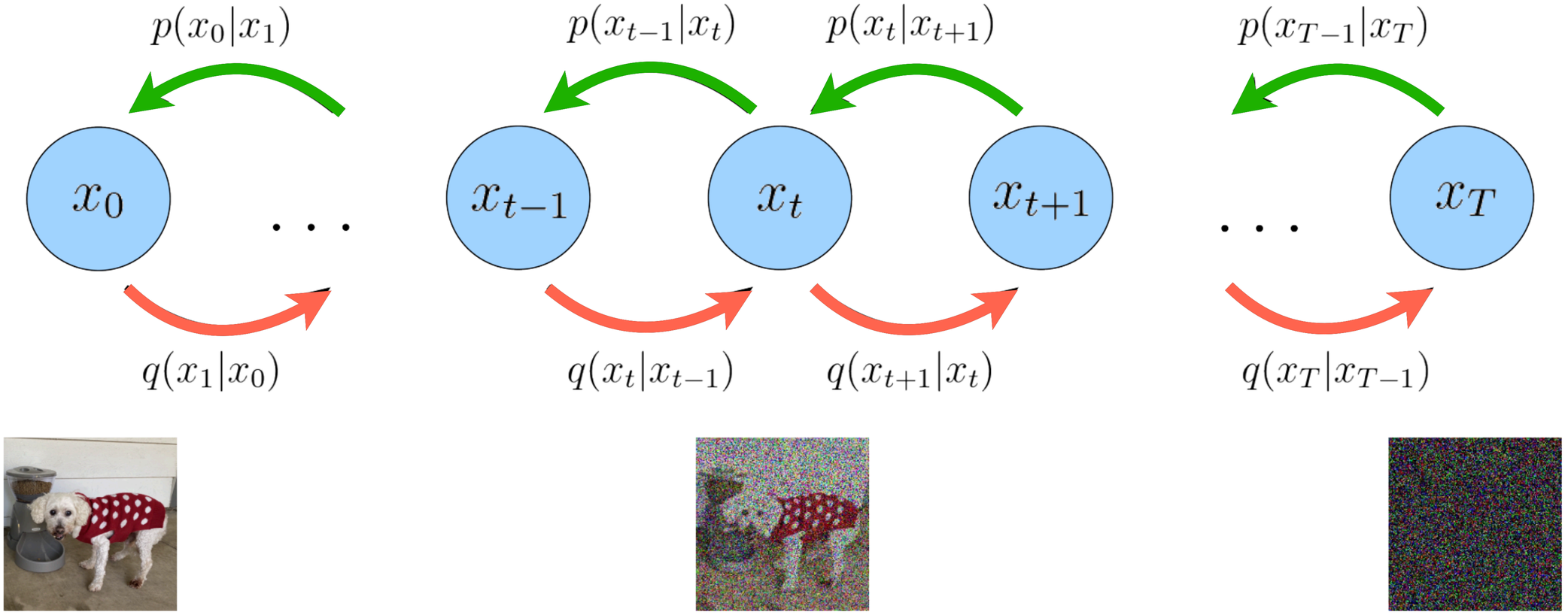
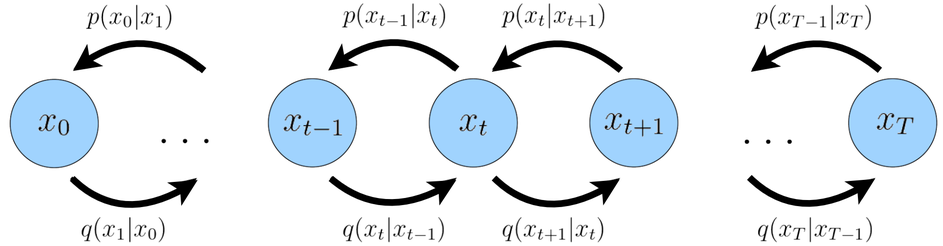


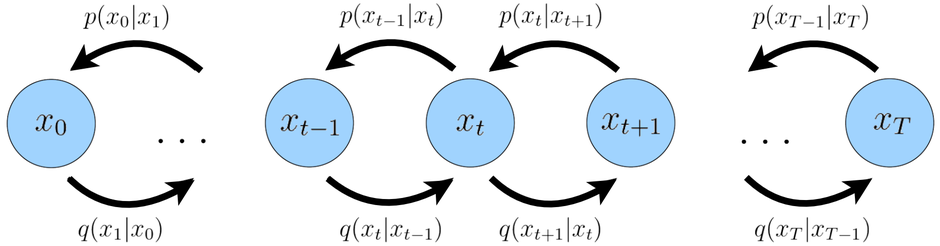
Image taken from https://arxiv.org/pdf/2208.11970

Variational Diffusion Models: Training



- Similar to VAEs, training diffusion models involves maximizing the **ELBO**

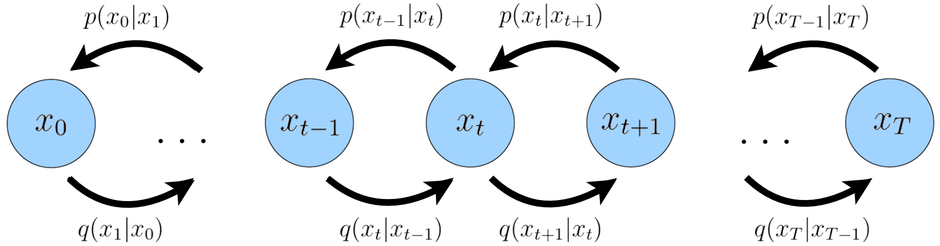
Variational Diffusion Models: Training



- Similar to VAEs, training diffusion models involves maximizing the **ELBO**

$$\underbrace{\mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)\right]}_{\text{reconstruction term}} - \underbrace{D_{\mathrm{KL}}(q(\boldsymbol{x}_T|\boldsymbol{x}_0) \parallel p(\boldsymbol{x}_T))}_{\text{prior matching term}} - \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

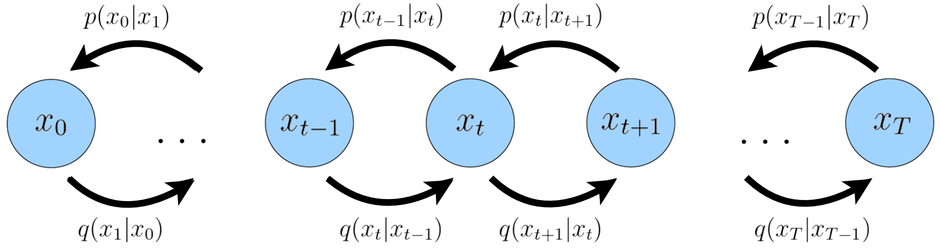
Variational Diffusion Models: Training



- Similar to VAEs, training diffusion models involves maximizing the **ELBO**

$$\underbrace{\mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)\right]}_{\text{reconstruction term}} - \underbrace{D_{\mathrm{KL}}(q(\boldsymbol{x}_T|\boldsymbol{x}_0) \parallel p(\boldsymbol{x}_T))}_{\text{prior matching term}} - \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

Variational Diffusion Models: Training

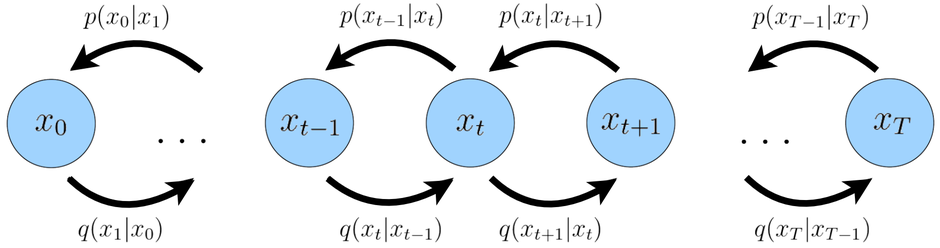


- Similar to VAEs, training diffusion models involves maximizing the **ELBO**

$$\underbrace{\mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)\right]}_{\text{reconstruction term}} - \underbrace{D_{\mathrm{KL}}(q(\boldsymbol{x}_T|\boldsymbol{x}_0) \parallel p(\boldsymbol{x}_T))}_{\text{prior matching term}} - \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

Reconstruction term that is similar to the ELBO in a VAE.

Variational Diffusion Models: Training

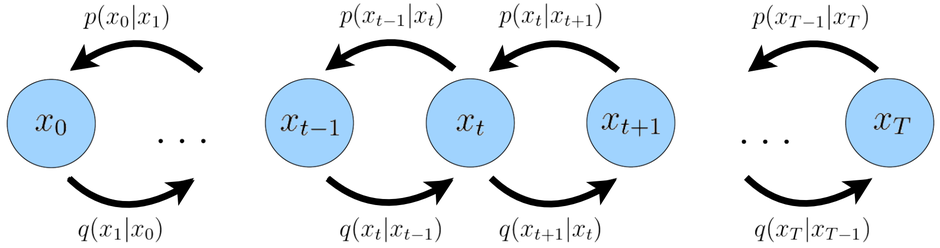


- Similar to VAEs, training diffusion models involves maximizing the **ELBO**

$$\underbrace{\mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)\right]}_{\text{reconstruction term}} - \underbrace{D_{\mathrm{KL}}(q(\boldsymbol{x}_T|\boldsymbol{x}_0) \parallel p(\boldsymbol{x}_T))}_{\text{prior matching term}} - \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

Represents how close the distribution of the final noised input x_T is to a standard Gaussian prior. This term has no trainable parameters and is ignored during training.

Variational Diffusion Models: Training

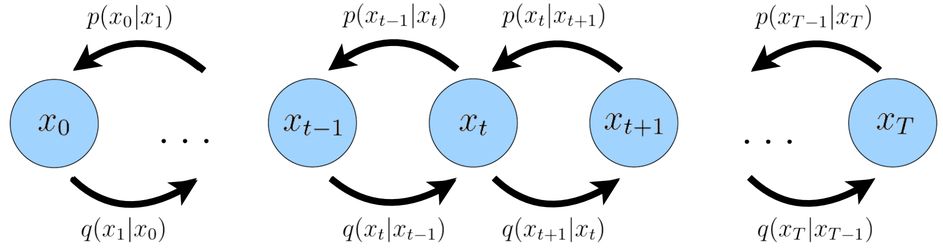


- Similar to VAEs, training diffusion models involves maximizing the **ELBO**

$$\underbrace{\mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1)\right]}_{\text{reconstruction term}} - \underbrace{D_{\mathrm{KL}}(q(\boldsymbol{x}_T|\boldsymbol{x}_0) \parallel p(\boldsymbol{x}_T))}_{\text{prior matching term}} - \sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

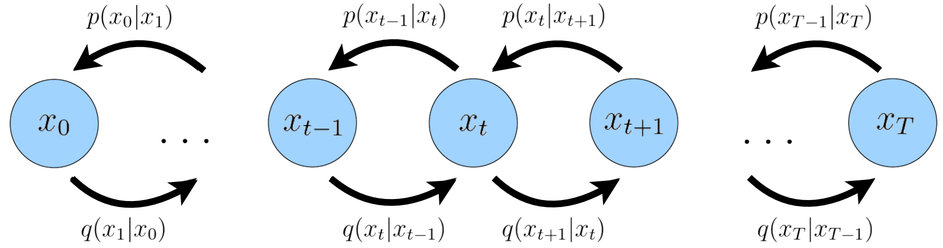
The decoder $p_\theta\left(x_{t-1} \mid x_t\right)$ is learned to be an approximation to the ground truth denoising transition $q\left(x_{t-1} \mid x_t, x_0\right)$

Variational Diffusion Models: Training



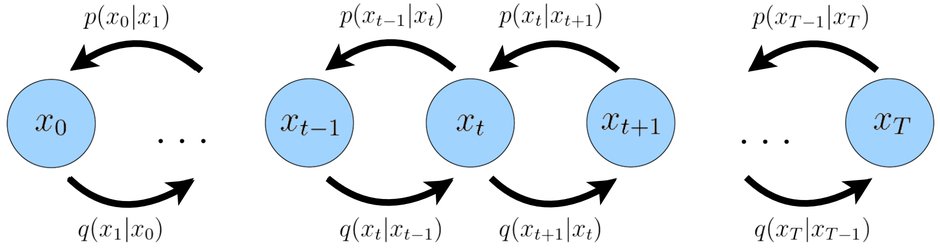
$$\sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

Variational Diffusion Models: Training



$$\sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

Variational Diffusion Models: Training

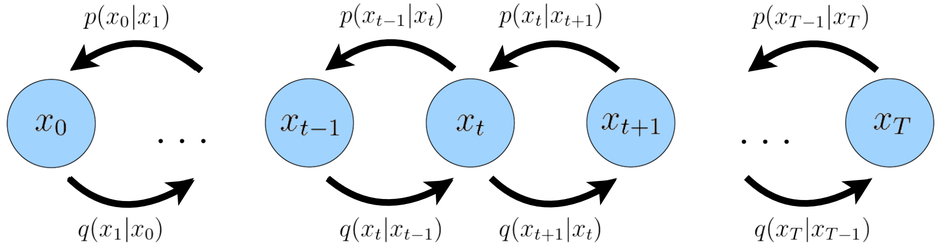


$$\sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

$$q\left(x_{t-1}\,|\,x_t, x_0\right) = \frac{q(x_t\,|\,x_{t-1}, x_0)\;q(x_{t-1}\,|\,x_0)}{q(x_t\,|\,x_0)}$$

Bayes Rule

Variational Diffusion Models: Training



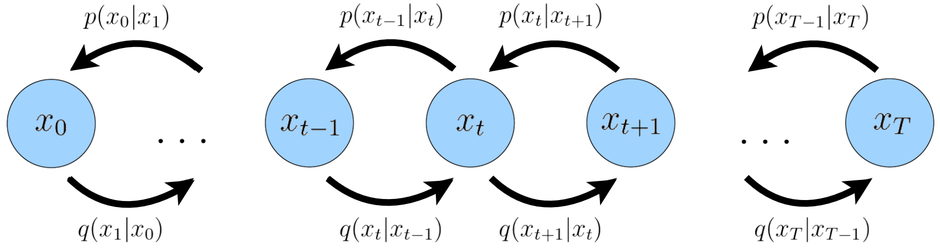
$$\sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

$$q\left(x_{t-1}\,|\,x_t, x_0\right) = \frac{\overbrace{q(x_t\,|\,x_{t-1})}^{=\,q(x_t\,|\,x_{t-1})}\; q(x_{t-1}\,|\,x_0)}{q(x_t\,|\,x_0)}$$

Bayes Rule

Markov Property

Variational Diffusion Models: Training



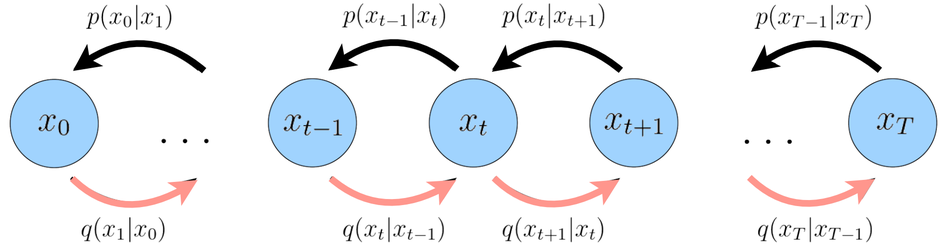
$$\sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

$$q\left(x_{t-1}\,|\,x_t, x_0\right) = \frac{\overbrace{q(x_t\,|\,x_{t-1}, x_0)}^{=\,q(x_t\,|\,x_{t-1})}\; q(x_{t-1}\,|\,x_0)}{q(x_t\,|\,x_0)}$$

Bayes Rule

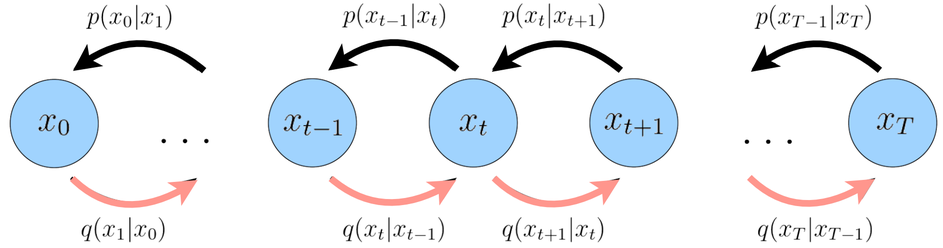
Markov Property

Variational Diffusion Models: Forward Process



- **Assumption:** Encoder at each step is a Gaussian distribution centered around the output of the previous step.

Variational Diffusion Models: Forward Process

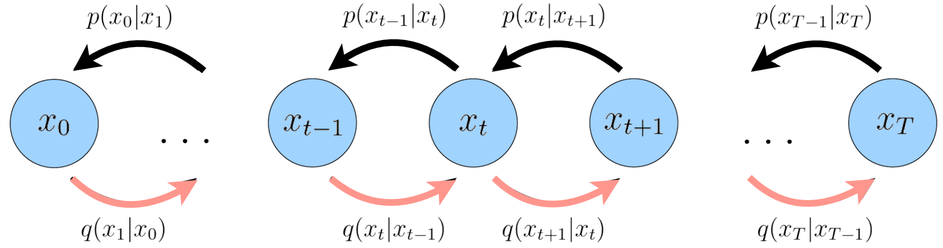


- **Assumption:** Encoder at each step is a Gaussian distribution centered around the output of the previous step.

$$q(x_t \mid x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{\alpha_t} x_{t-1}, (1-\alpha_t)\mathbf{I}\right)$$

$\{\alpha_t \in (0,1)\}_{t=1}^{T}$ is a predefined variance schedule

Variational Diffusion Models: Forward Process



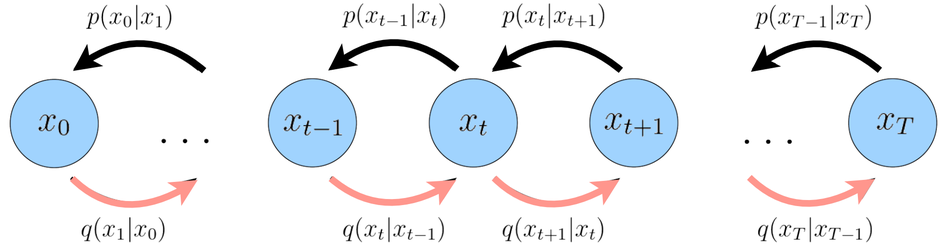
- **Assumption:** Encoder at each step is a Gaussian distribution centered around the output of the previous step.

$$q(x_t \mid x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{\alpha_t} x_{t-1}, (1-\alpha_t)\mathbf{I}\right)$$

$\{\alpha_t \in (0,1)\}_{t=1}^{T}$ is a predefined variance schedule

- As $\mathbf{x_0}$ is the initial input, obtaining the noisy input for any arbitrary timestep involves solving the following,

Variational Diffusion Models: Forward Process



- **Assumption:** Encoder at each step is a Gaussian distribution centered around the output of the previous step.

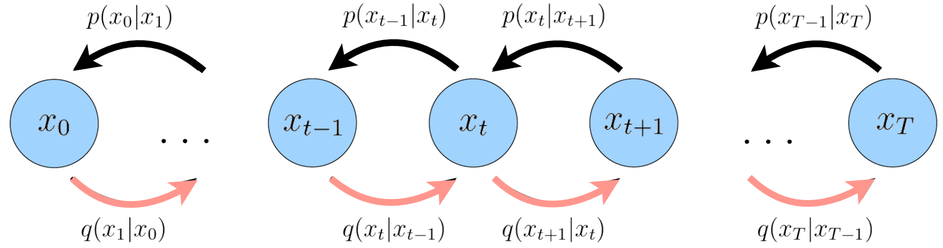
$$q(x_t \mid x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{\alpha_t} x_{t-1}, (1-\alpha_t)\mathbf{I}\right)$$

$\{\alpha_t \in (0,1)\}_{t=1}^{T}$ is a predefined variance schedule

- As $\mathbf{x_0}$ is the initial input, obtaining the noisy input for any arbitrary timestep involves solving the following,

$$q(x_t \mid x_0) = q(x_t \mid x_{t-1}) \cdot q(x_{t-1} \mid x_{t-2}) \ \ldots \ \ q(x_1 \mid x_0)$$

Variational Diffusion Models: Forward Process



- **Assumption:** Encoder at each step is a Gaussian distribution centered around the output of the previous step.

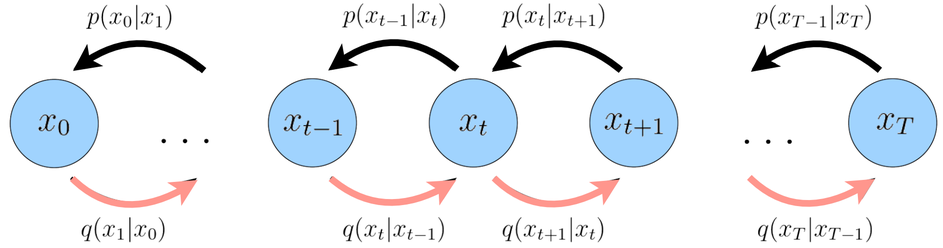
$$q(x_t \mid x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{\alpha_t} x_{t-1}, (1-\alpha_t)\mathbf{I}\right)$$

$\{\alpha_t \in (0,1)\}_{t=1}^{T}$ is a predefined variance schedule

- As $\mathbf{x_0}$ is the initial input, obtaining the noisy input for any arbitrary timestep involves solving the following,

$$q(x_t \mid x_0) = q(x_t \mid x_{t-1}) \cdot q(x_{t-1} \mid x_{t-2}) \ \ldots \ \ q(x_1 \mid x_0)$$

Variational Diffusion Models: Forward Process



- **Assumption:** Encoder at each step is a Gaussian distribution centered around the output of the previous step.

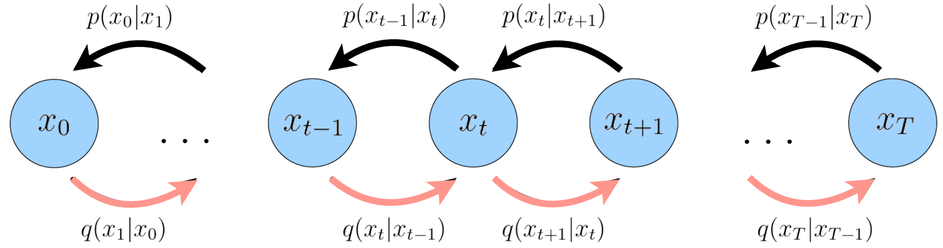
$$q(x_t \mid x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{\alpha_t} x_{t-1}, (1-\alpha_t)\mathbf{I}\right)$$

$\{\alpha_t \in (0,1)\}_{t=1}^{T}$ is a predefined variance schedule

- As $\mathbf{x_0}$ is the initial input, obtaining the noisy input for any arbitrary timestep involves solving the following,

$$q(x_t \mid x_0) = \prod_{i=1}^{t} q(x_i \mid x_{i-1})$$

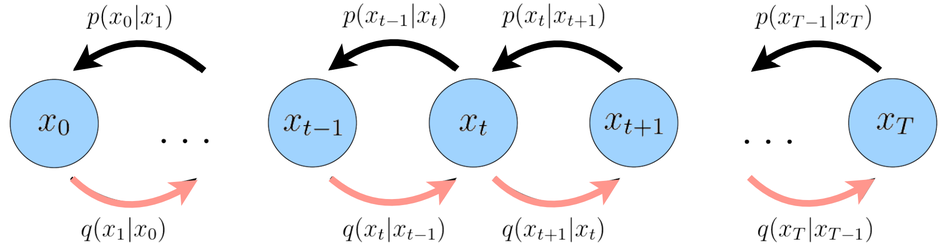
Variational Diffusion Models: Forward Process



$$q(x_t \mid x_0) = \prod_{i=1}^{t} q(x_i \mid x_{i-1})$$

- Naively computing this is **slow.**
  - Given a new input, one would need to go through this chain of length t, computing intermediate noisy inputs.

Variational Diffusion Models: Forward Process

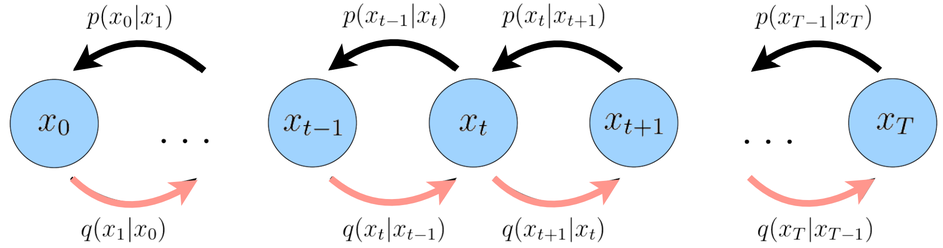


$$q(x_t \mid x_0) = \prod_{i=1}^{t} q(x_i \mid x_{i-1})$$

- Naively computing this is **slow.**
  - Given a new input, one would need to go through this chain of length t, computing intermediate noisy inputs.

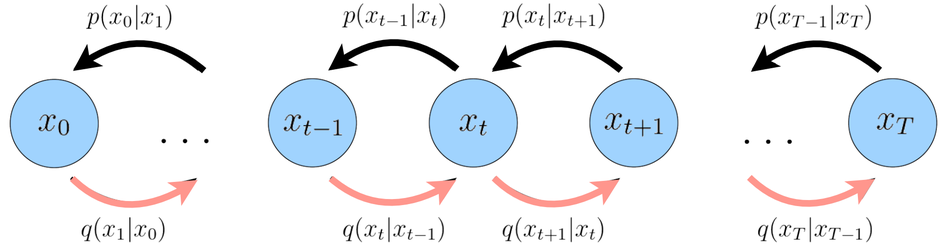
- However, as we can leverage properties of Gaussians to make this efficient.

Variational Diffusion Models: Forward Process



$$x_t \sim q(x_t \mid x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{\alpha_t} x_{t-1}, (1-\alpha_t)\mathbf{I}\right)$$

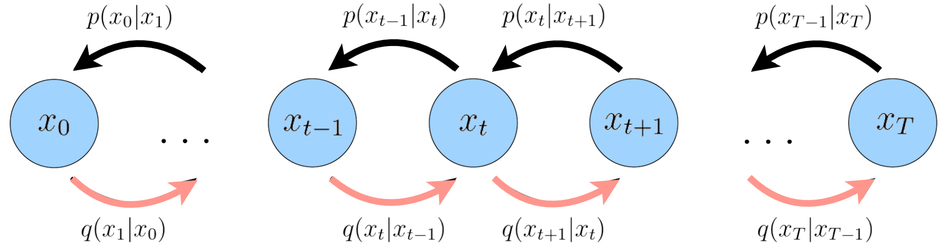
Variational Diffusion Models: Forward Process



$$x_t \sim q(x_t \mid x_{t-1}) = \sqrt{\alpha_t} x_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1} \quad \epsilon_{t-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

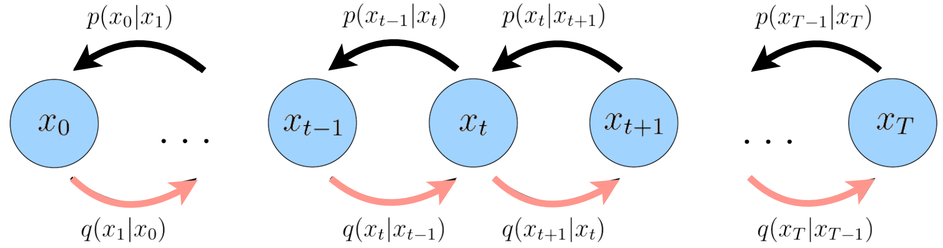
Reparametrization Trick

Variational Diffusion Models: Forward Process



$$x_t \sim q(x_t \mid x_{t-1}) = \sqrt{\alpha_t} x_{t-1} + \sqrt{1 - \alpha_t} \epsilon_{t-1}$$

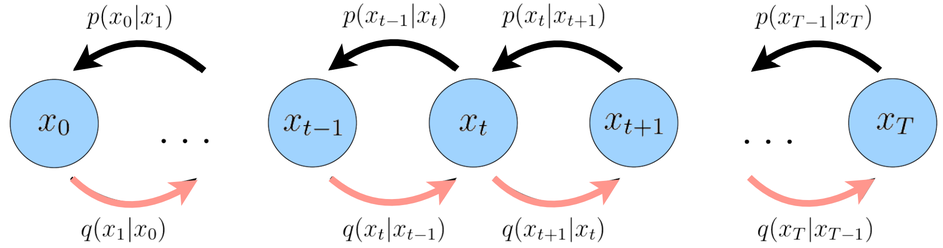
Variational Diffusion Models: Forward Process



$$x_t \sim q(x_t \mid x_{t-1}) = \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_{t-2}}\epsilon_{t-2}\right) + \sqrt{1-\alpha_t}\epsilon_{t-1}$$

Recursion

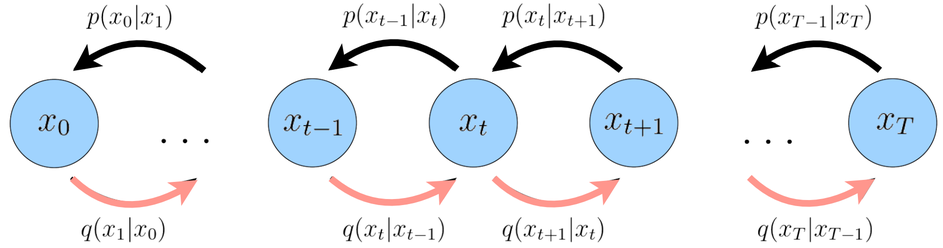
Variational Diffusion Models: Forward Process



$$x_t \sim q(x_t \mid x_{t-1}) = \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_{t-2}}\epsilon_{t-2}\right) + \sqrt{1-\alpha_t}\epsilon_{t-1}$$

Recursion

Variational Diffusion Models: Forward Process

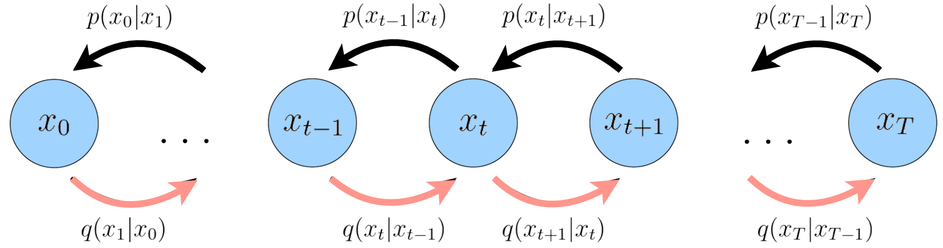


$$x_t \sim q(x_t \mid x_{t-1}) = \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_{t-2}}\epsilon_{t-2}\right) + \sqrt{1-\alpha_t}\epsilon_{t-1}$$

Recursion

$$= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} \quad + \sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\epsilon_{t-2} + \sqrt{1-\alpha_t}\epsilon_{t-1}$$

Variational Diffusion Models: Forward Process



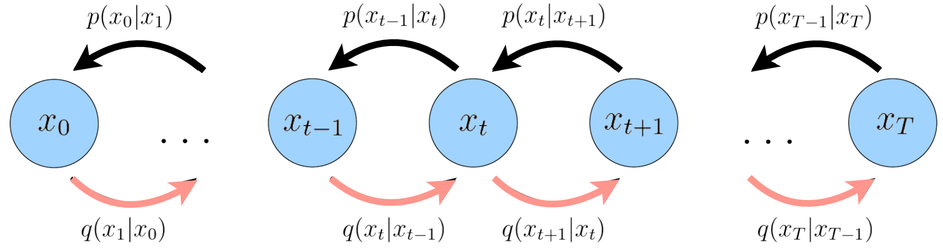
$$x_t \sim q(x_t \mid x_{t-1}) = \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_{t-2}}\epsilon_{t-2}\right) + \sqrt{1-\alpha_t}\epsilon_{t-1}$$

Recursion

$$= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\epsilon_{t-2} + \sqrt{1-\alpha_t}\epsilon_{t-1}$$

$$\sim \mathcal{N}\left(\mathbf{0}, (\alpha_t - \alpha_t\alpha_{t-1})\mathbf{I}\right)$$

Variational Diffusion Models: Forward Process



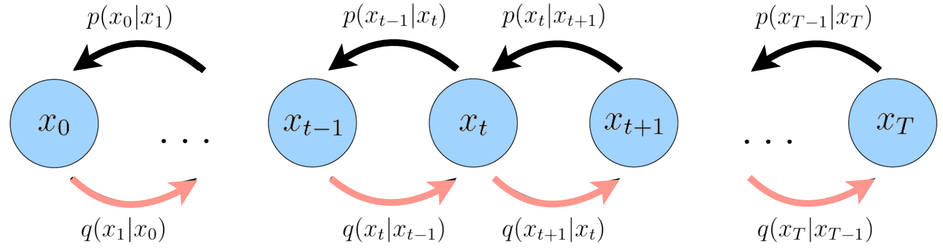
$$x_t \sim q(x_t \mid x_{t-1}) = \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_{t-2}}\epsilon_{t-2}\right) + \sqrt{1-\alpha_t}\epsilon_{t-1}$$

Recursion

$$= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\epsilon_{t-2} + \sqrt{1-\alpha_t}\epsilon_{t-1}$$

$$\sim \mathcal{N}\left(\mathbf{0}, (\alpha_t - \alpha_t\alpha_{t-1})\mathbf{I}\right) \qquad \sim \mathcal{N}\left(\mathbf{0}, (1-\alpha_t)\mathbf{I}\right)$$

Variational Diffusion Models: Forward Process



$$x_t \sim q(x_t \mid x_{t-1}) = \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_{t-2}}\epsilon_{t-2}\right) + \sqrt{1-\alpha_t}\epsilon_{t-1}$$

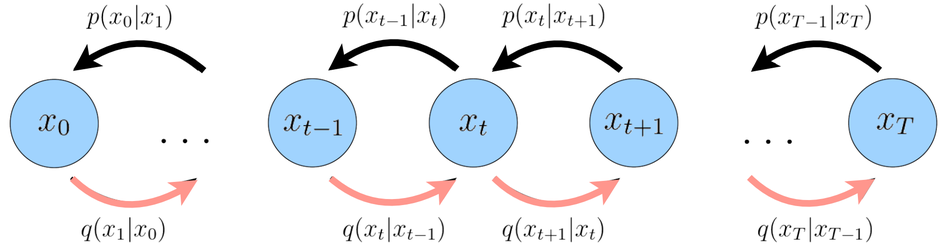
Recursion

$$= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\epsilon_{t-2} + \sqrt{1-\alpha_t}\epsilon_{t-1}$$

$$\sim \mathcal{N}\left(\mathbf{0}, (\alpha_t - \alpha_t\alpha_{t-1})\mathbf{I}\right) \qquad \sim \mathcal{N}\left(\mathbf{0}, (1-\alpha_t)\mathbf{I}\right)$$

Sum of two independent Gaussian random variables is a Gaussian.
Mean = sum of the two means
Variance = sum of the two variances

Variational Diffusion Models: Forward Process



$$x_t \sim q(x_t \mid x_{t-1}) = \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_{t-2}}\epsilon_{t-2}\right) + \sqrt{1-\alpha_t}\epsilon_{t-1}$$

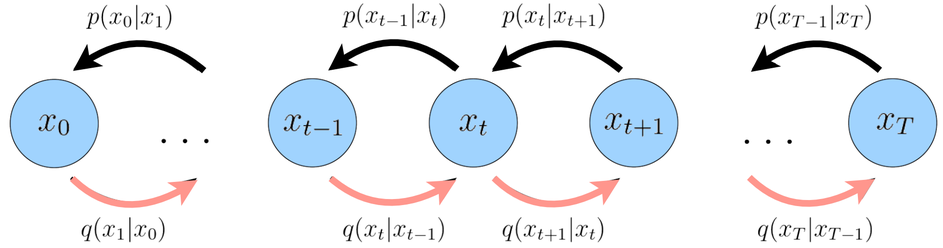
Recursion

$$= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{\alpha_t - \alpha_t\alpha_{t-1}}\epsilon_{t-2} + \sqrt{1-\alpha_t}\epsilon_{t-1}$$

$$\sim \mathcal{N}\left(\mathbf{0}, (\alpha_t - \alpha_t\alpha_{t-1})\mathbf{I}\right) \qquad \sim \mathcal{N}\left(\mathbf{0}, (1-\alpha_t)\mathbf{I}\right)$$

Sum of two independent Gaussian random variables is a Gaussian.
Mean = sum of the two means
Variance = sum of the two variances

Variational Diffusion Models: Forward Process



$$x_t \sim q(x_t \mid x_{t-1}) = \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_{t-2}}\epsilon_{t-2}\right) + \sqrt{1-\alpha_t}\epsilon_{t-1}$$

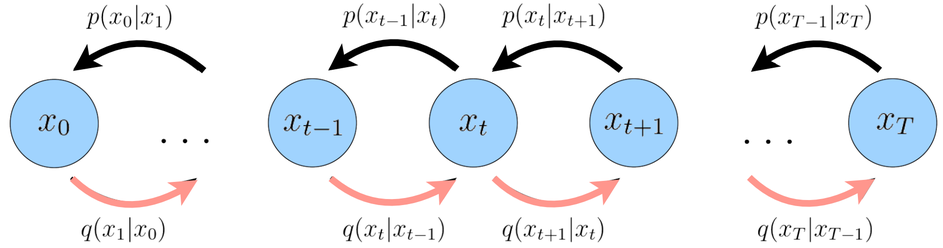
Recursion

$$= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} \quad + \sqrt{1-\alpha_t\alpha_{t-1}}\epsilon^*_{t-2}$$

$$\sim \mathcal{N}\left(\mathbf{0}, (1-\alpha_t\alpha_{t-1})\mathbf{I}\right)$$

Sum of two independent Gaussian random variables is a Gaussian.
Mean = sum of the two means
Variance = sum of the two variances

Variational Diffusion Models: Forward Process



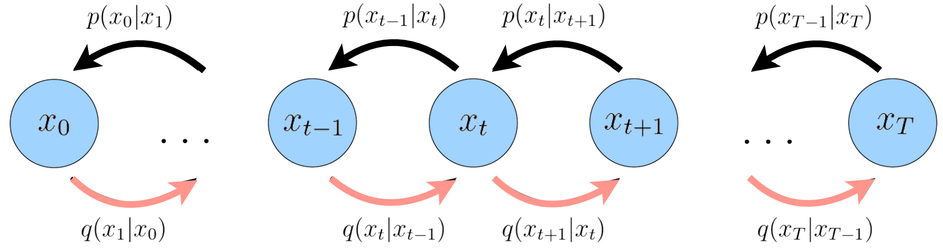
$$x_t \sim q(x_t \mid x_{t-1}) = \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_{t-2}}\epsilon_{t-2}\right) + \sqrt{1-\alpha_t}\epsilon_{t-1}$$

Recursion

$$= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} \quad + \sqrt{1-\alpha_t\alpha_{t-1}}\epsilon^*_{t-2}$$

$$\sim \mathcal{N}\left(\mathbf{0}, (1-\alpha_t\alpha_{t-1})\mathbf{I}\right)$$

Variational Diffusion Models: Forward Process



$$x_t \sim q(x_t \mid x_{t-1}) = \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_{t-2}}\epsilon_{t-2}\right) + \sqrt{1-\alpha_t}\epsilon_{t-1}$$

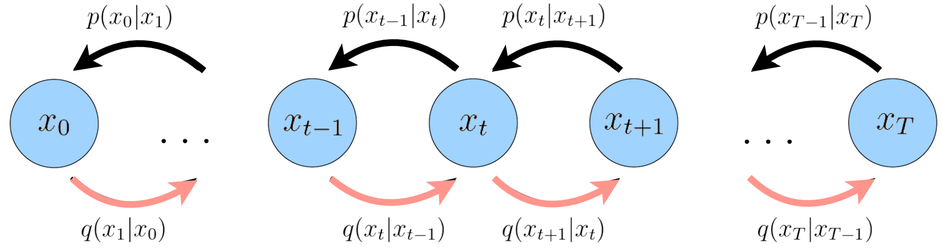
Recursion

$$= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} \quad + \sqrt{1-\alpha_t\alpha_{t-1}}\epsilon^*_{t-2}$$

$$\sim \mathcal{N}\left(\mathbf{0}, (1-\alpha_t\alpha_{t-1})\mathbf{I}\right)$$

$$= \ldots$$

Variational Diffusion Models: Forward Process



$$x_t \sim q(x_t \mid x_{t-1}) = \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_{t-2}}\epsilon_{t-2}\right) + \sqrt{1-\alpha_t}\epsilon_{t-1}$$

Recursion

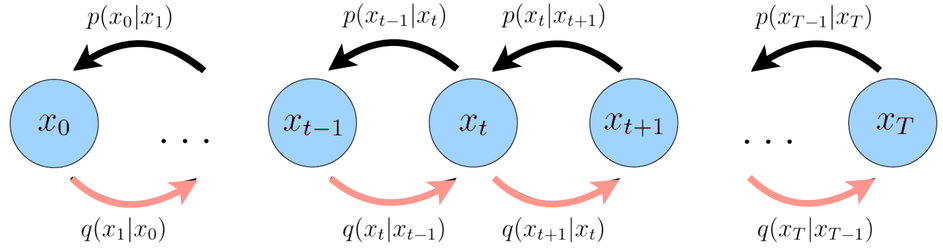
$$= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} \quad + \sqrt{1-\alpha_t\alpha_{t-1}}\epsilon^*_{t-2}$$

$$\sim \mathcal{N}\left(\mathbf{0}, (1-\alpha_t\alpha_{t-1})\mathbf{I}\right)$$

$$= \dots$$

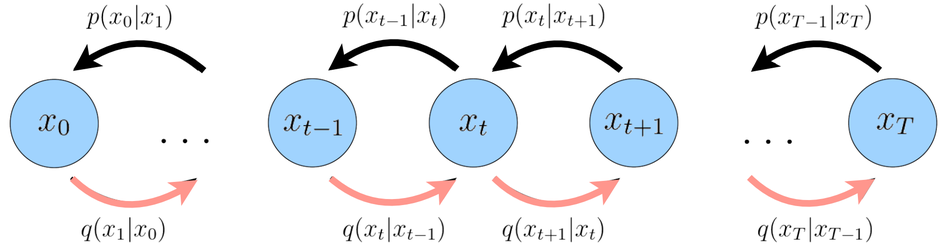
$$= \sqrt{\prod_{i=1}^{t}\alpha_i}x_0 + \sqrt{1-\prod_{i=1}^{t}\alpha_i}\epsilon^*_0$$

Variational Diffusion Models: Forward Process



$$x_t \sim q(x_t \mid x_{t-1}) = \sqrt{\prod_{i=1}^{t} \alpha_i} x_0 + \sqrt{1 - \prod_{i=1}^{t} \alpha_i} \epsilon_0^*$$

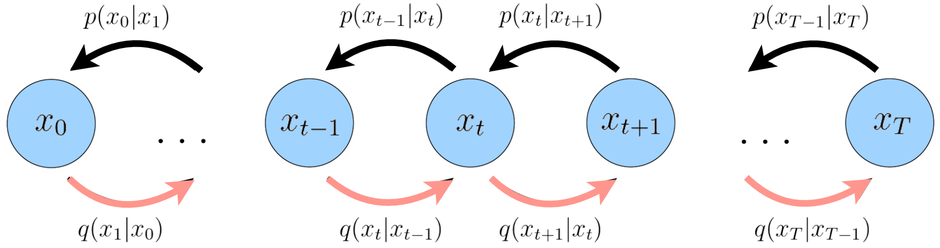
Variational Diffusion Models: Forward Process



$$x_t \sim q(x_t \mid x_{t-1}) = \sqrt{\prod_{i=1}^{t} \alpha_i} x_0 + \sqrt{1 - \prod_{i=1}^{t} \alpha_i} \epsilon_0^*$$

$$\sqrt{\bar{\alpha}_t}$$

Variational Diffusion Models: Forward Process

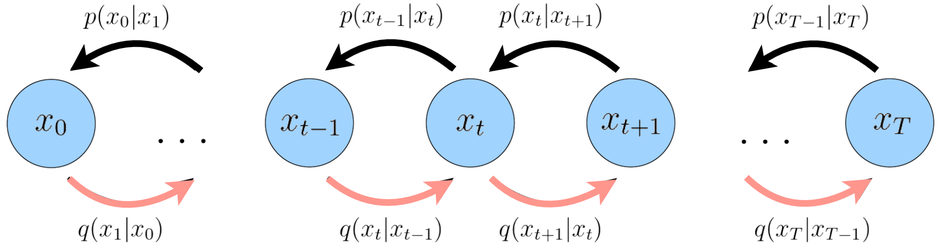


$$x_t \sim q(x_t \mid x_{t-1}) = \sqrt{\prod_{i=1}^{t} \alpha_i} x_0 + \sqrt{1 - \prod_{i=1}^{t} \alpha_i} \epsilon_0^*$$

$\sqrt{\bar{\alpha}_t}$

$\sqrt{1 - \bar{\alpha}_t}$

Variational Diffusion Models: Forward Process

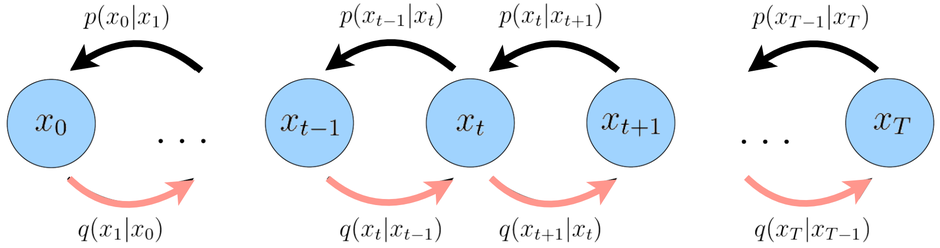


$$x_t \sim q(x_t \mid x_{t-1}) = \sqrt{\prod_{i=1}^{t} \alpha_i} x_0 + \sqrt{1 - \prod_{i=1}^{t} \alpha_i} \epsilon_0^*$$

$\sqrt{\bar{\alpha}_t}$ $\qquad$ $\sqrt{1 - \bar{\alpha}_t}$

$$= \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon_0^*$$

Variational Diffusion Models: Forward Process



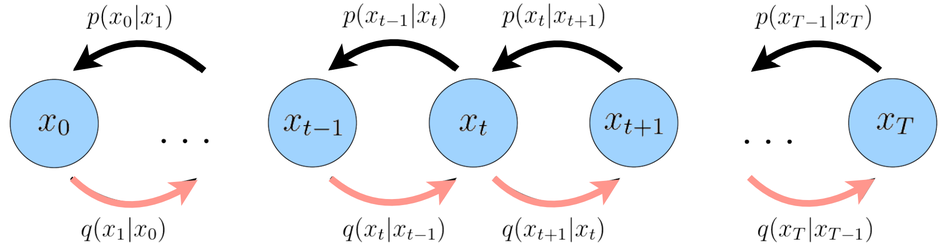
$$x_t \sim q(x_t \mid x_{t-1}) = \sqrt{\prod_{i=1}^{t} \alpha_i} x_0 + \sqrt{1 - \prod_{i=1}^{t} \alpha_i} \epsilon_0^*$$

$\sqrt{\bar{\alpha}_t}$ $\qquad$ $\sqrt{1 - \bar{\alpha}_t}$

$$x_t \sim q(x_t \mid x_0) = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon_0^*$$

$$\sim \mathcal{N}\left(\sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t)\mathbf{I}\right)$$

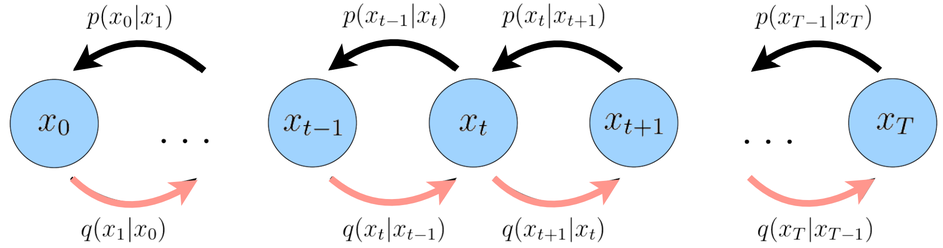
Variational Diffusion Models: Forward Process



$$x_t \sim q(x_t \mid x_0) \quad = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon_0^*$$

$$\sim \mathcal{N}\left(\sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t)\mathbf{I}\right)$$

Variational Diffusion Models: Forward Process

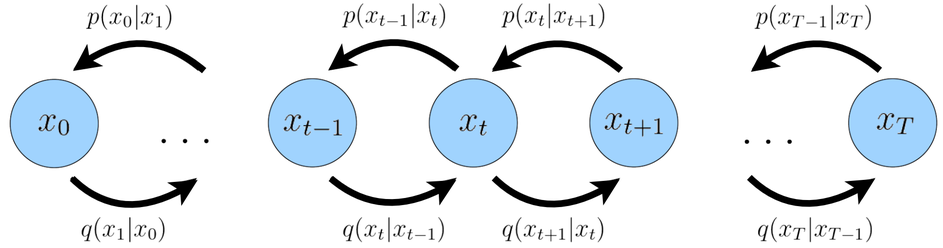


$$x_t \sim q(x_t \mid x_0) \quad = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon_0^*$$

$$\sim \mathcal{N}\left(\sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t)\mathbf{I}\right)$$

- This enables computing noised inputs at an arbitrary timestep directly from the denoised input.
  - Intuitively, setting $\bar{\alpha}_1 > \bar{\alpha}_2 > \ldots > \bar{\alpha}_T$ makes the sample noise with larger step size.

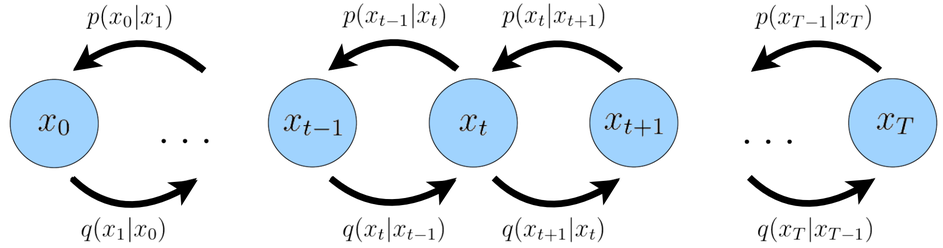
Variational Diffusion Models: Forward Process



$$\sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

- Recall that training diffusion models involves computing the denoising matching term.

Variational Diffusion Models: Forward Process



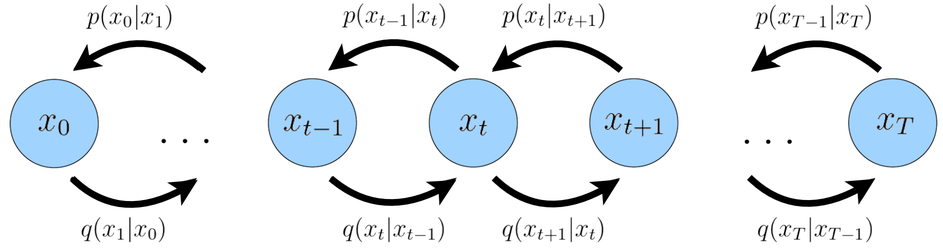
$$\sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

- Recall that training diffusion models involves computing the denoising matching term.

$$q\left(x_{t-1}\,|\,x_t, x_0\right) = \frac{q(x_t\,|\,x_{t-1}, x_0)\;\; q(x_{t-1}\,|\,x_0)}{q(x_t\,|\,x_0)}$$

Bayes Rule

Variational Diffusion Models: Forward Process



$$\sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

- Recall that training diffusion models involves computing the denoising matching term.

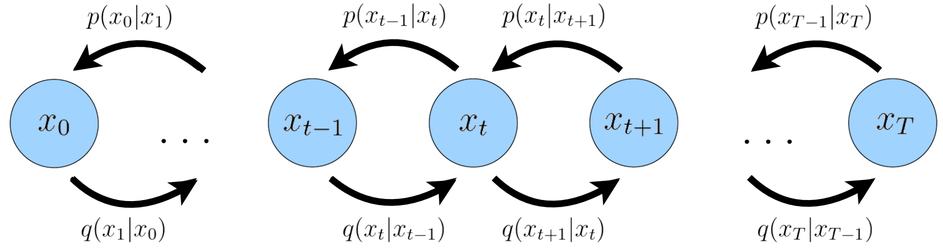
$$q\left(x_{t-1}|x_t,x_0\right) = \frac{q(x_t|x_{t-1},x_0)\; q(x_{t-1}|x_0)}{q(x_t|x_0)}$$

$= q(x_t|x_{t-1})$

Bayes Rule

Markov Property

Variational Diffusion Models: Forward Process



$$\sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

- Recall that training diffusion models involves computing the denoising matching term.

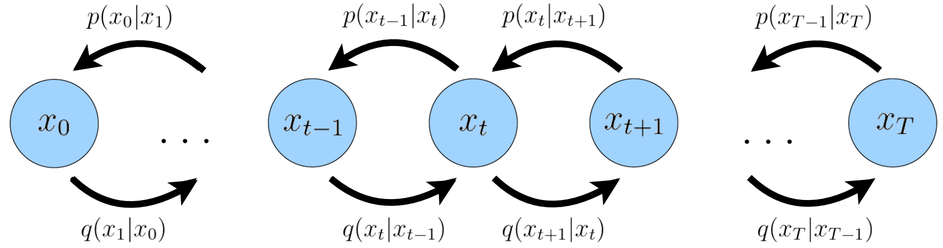
$$q\left(x_{t-1}\,|\,x_t, x_0\right) = \frac{\overbrace{q(x_t\,|\,x_{t-1}, x_0)}^{= q(x_t\,|\,x_{t-1})} \overbrace{q(x_{t-1}\,|\,x_0)}^{\mathcal{N}\left(\sqrt{\bar{\alpha}_{t-1}}x_0, (1-\bar{\alpha}_{t-1})\mathbf{I}\right)}}{\underbrace{q(x_t\,|\,x_0)}_{\mathcal{N}\left(\sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I}\right)}}$$

Bayes Rule

Markov Property

Derivation Shown Earlier

Variational Diffusion Models: Forward Process



$$\sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

- Recall that training diffusion models involves computing the denoising matching term.

$$q\left(x_{t-1}|x_t,x_0\right) = \frac{\overbrace{q(x_t|x_{t-1},x_0)}^{=\,q(x_t|x_{t-1})}\ \overbrace{q(x_{t-1}|x_0)}^{\mathcal{N}\left(\sqrt{\bar{\alpha}_{t-1}}x_0,(1-\bar{\alpha}_{t-1})\mathbf{I}\right)}}{\underbrace{q(x_t|x_0)}_{\mathcal{N}\left(\sqrt{\bar{\alpha}_t}x_0,(1-\bar{\alpha}_t)\mathbf{I}\right)}}$$

Bayes Rule

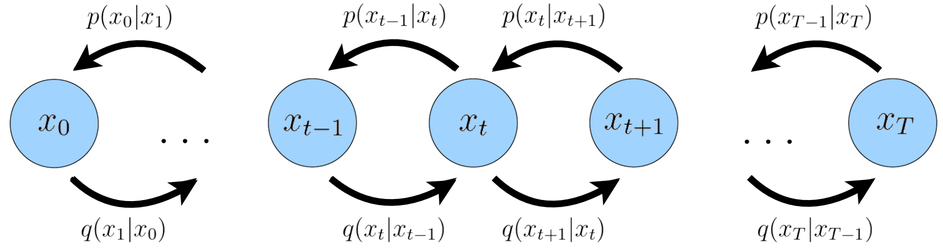
Markov Property

Derivation Shown Earlier

- Plugging in the distributions and simplifying,

$$q\left(x_{t-1}|x_t,x_0\right) \propto \mathcal{N}(\boldsymbol{x}_{t-1};\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t+\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t},\frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{I})$$

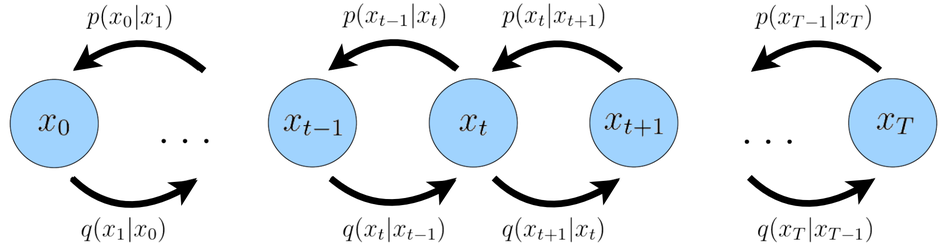
Variational Diffusion Models: Forward Process



$$\sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

$$q\left(x_{t-1}\,|\,x_t, x_0\right) \propto \mathcal{N}(\boldsymbol{x}_{t-1}; \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}, \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{I})$$

Variational Diffusion Models: Forward Process

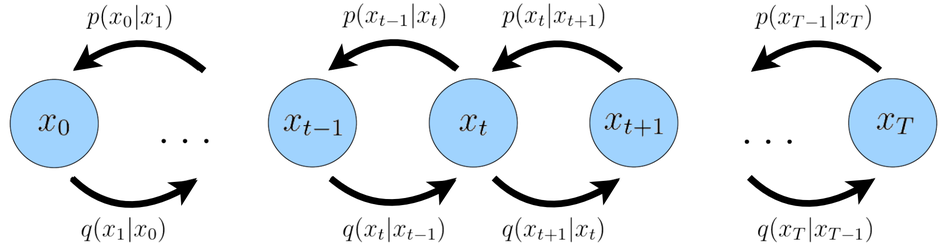


$$\sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

$$q\left(x_{t-1}\,|\,x_t, x_0\right) \propto \mathcal{N}(\boldsymbol{x}_{t-1}; \underbrace{\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}}_{\mu_q(\boldsymbol{x}_t,\boldsymbol{x}_0)}, \frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{I})$$

Dependent on noisy and ground truth input

Variational Diffusion Models: Forward Process



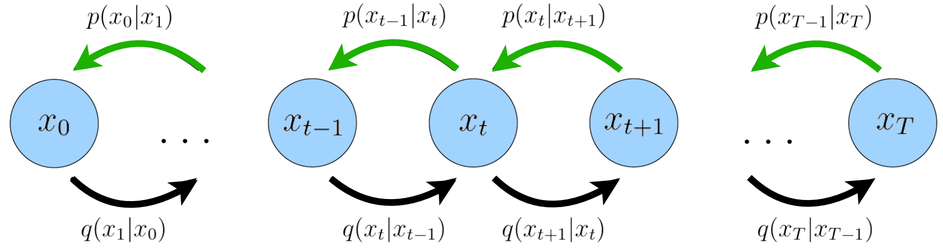
$$\sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

$$q\left(x_{t-1}\,|\,x_t, x_0\right) \propto \mathcal{N}(\boldsymbol{x}_{t-1}; \underbrace{\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}}_{\mu_q(\boldsymbol{x}_t,\boldsymbol{x}_0)}, \underbrace{\frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{I}}_{\boldsymbol{\Sigma}_q(t)})$$

Dependent on noisy and ground truth input

Only dependent on time t

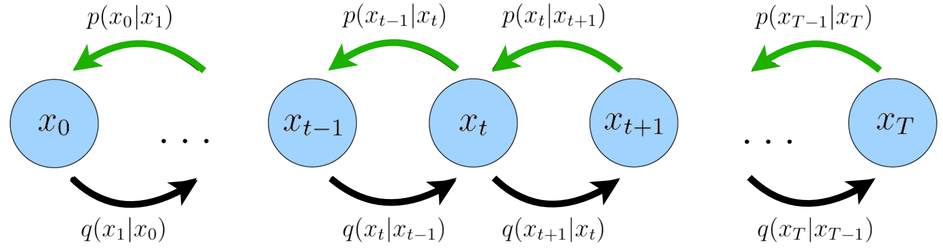
Variational Diffusion Models: Reverse Process



$$\sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

- The goal of the reverse process is to transform noise to a sample from the data distribution.

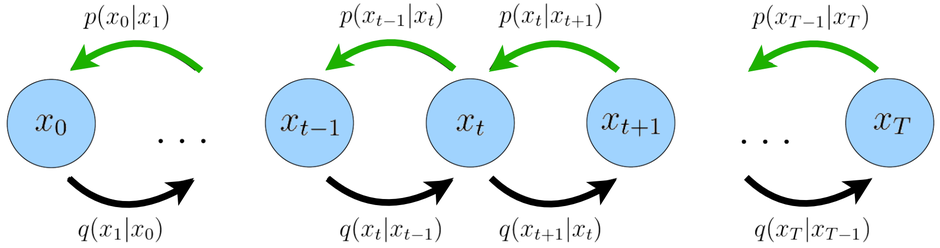
Variational Diffusion Models: Reverse Process



$$\sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

- The goal of the reverse process is to transform noise to a sample from the data distribution.

Variational Diffusion Models: Reverse Process

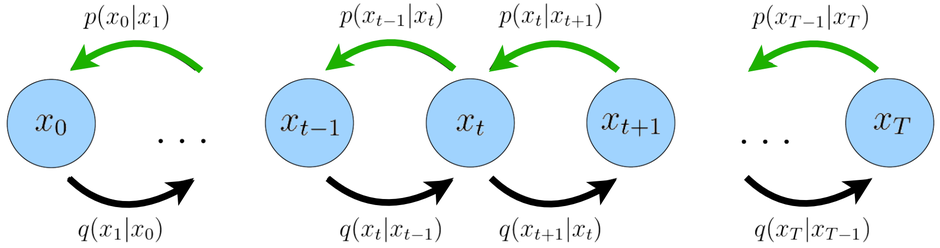


$$\sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

- The goal of the reverse process is to transform noise to a sample from the data distribution.

- Ideally if we can reverse the forward process and sample from $q(x_{t-1} \mid x_t)$, we can easily convert noise to a sample.

Variational Diffusion Models: Reverse Process

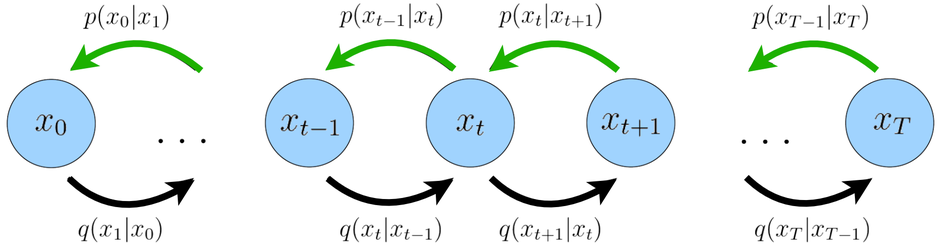


$$\sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

- The goal of the reverse process is to transform noise to a sample from the data distribution.

- Ideally if we can reverse the forward process and sample from $q(x_{t-1} \mid x_t)$, we can easily convert noise to a sample.
  - $q(x_{t-1} \mid x_t)$ is hard to estimate as it requires computing marginals.
  - Instead learn a model $p_\theta(x_{t-1} \mid x_t)$ to approximate the true reverse process.

Variational Diffusion Models: Reverse Process

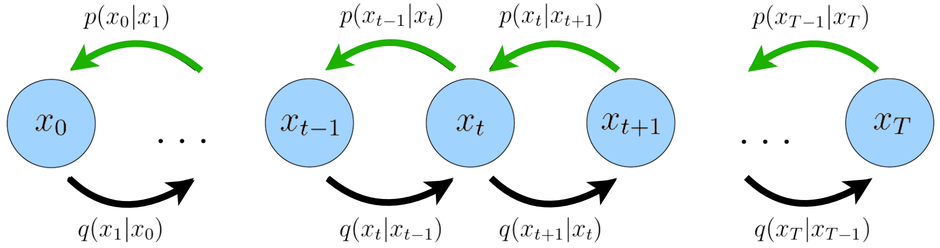


$$\sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

$$q\left(x_{t-1}\,|\,x_t,x_0\right) \propto \mathcal{N}(\boldsymbol{x}_{t-1}; \underbrace{\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}}_{\mu_q(\boldsymbol{x}_t,\boldsymbol{x}_0)}, \underbrace{\frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{I}}_{\boldsymbol{\Sigma}_q(t)})$$

- Following the KL optimization term, in order to ensure $p_\theta(x_{t-1}\,|\,x_t)$ approximates the ground-truth transition $q(x_{t-1}\,|\,x_t,x_0)$ as closely as possible, it is modelled as a Gaussian.

Variational Diffusion Models: Reverse Process

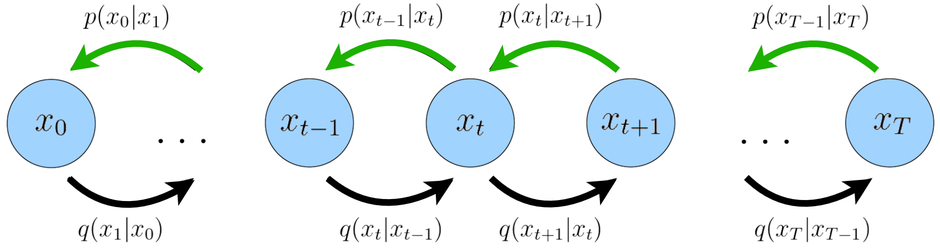


$$\sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

$$q\left(x_{t-1}\,|\,x_t, x_0\right) \propto \mathcal{N}(\boldsymbol{x}_{t-1}; \underbrace{\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}}_{\mu_q(\boldsymbol{x}_t,\boldsymbol{x}_0)}, \underbrace{\frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{I}}_{\boldsymbol{\Sigma}_q(t)})$$

- Following the KL optimization term, in order to ensure $p_\theta(x_{t-1}\,|\,x_t)$ approximates the ground-truth transition $q(x_{t-1}\,|\,x_t, x_0)$ as closely as possible, it is modelled as a Gaussian.
  - The mean is parameterized as the function $\mu_\theta(x_t, t)$

Variational Diffusion Models: Reverse Process

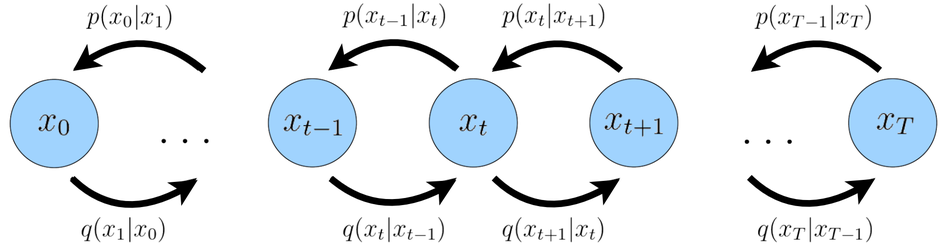


$$\sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

$$q\left(x_{t-1}\,|\,x_t, x_0\right) \propto \mathcal{N}(\boldsymbol{x}_{t-1}; \underbrace{\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}}_{\mu_q(\boldsymbol{x}_t,\boldsymbol{x}_0)}, \underbrace{\frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{I}}_{\boldsymbol{\Sigma}_q(t)})$$

- Following the KL optimization term, in order to ensure $p_\theta(x_{t-1}\,|\,x_t)$ approximates the ground-truth transition $q(x_{t-1}\,|\,x_t, x_0)$ as closely as possible, it is modelled as a Gaussian.
  - The mean is parameterized as the function $\mu_\theta(x_t, t)$
  - The variance is set to be the same as $\Sigma_q(t)$

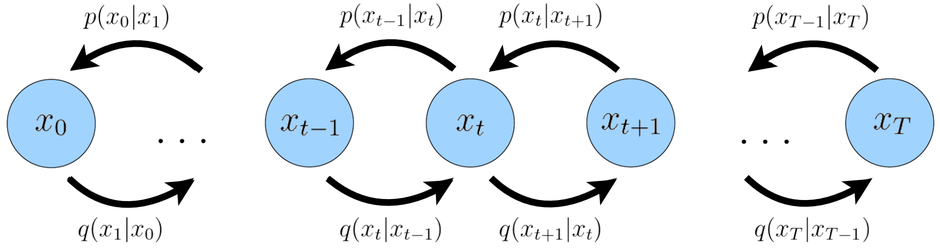
Variational Diffusion Models: Simplifying Training



$$\sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

- As it is ensured that the variances of $p_\theta(x_{t-1}\,|\,x_t)$ and $q\left(x_{t-1}\,|\,x_t, x_0\right)$ match exactly, optimizing the KL divergence can be simplified to,

Variational Diffusion Models: Simplifying Training



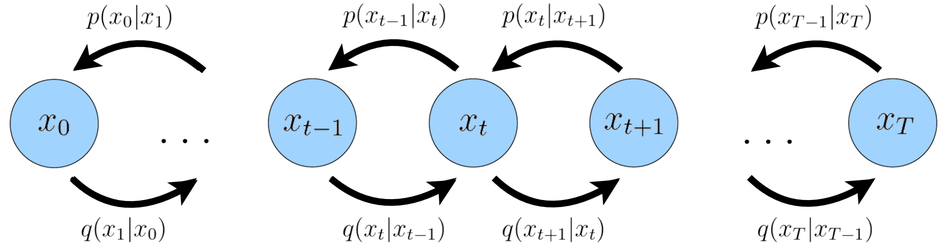
$$\sum_{t=2}^{T} \underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))\right]}_{\text{denoising matching term}}$$

- As it is ensured that the variances of $p_\theta(x_{t-1}\,|\,x_t)$ and $q\left(x_{t-1}\,|\,x_t, x_0\right)$ match exactly, optimizing the KL divergence can be simplified to,

$$\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)) = \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)}\left[\|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\|_2^2\right]$$

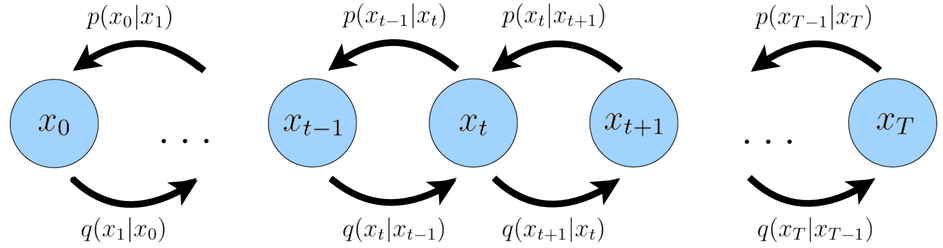
Reduces to minimizing the difference between the means of the two distributions

Variational Diffusion Models: Simplifying Training



$$\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)) = \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \left[\|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\|_2^2 \right]$$

Variational Diffusion Models: Simplifying Training

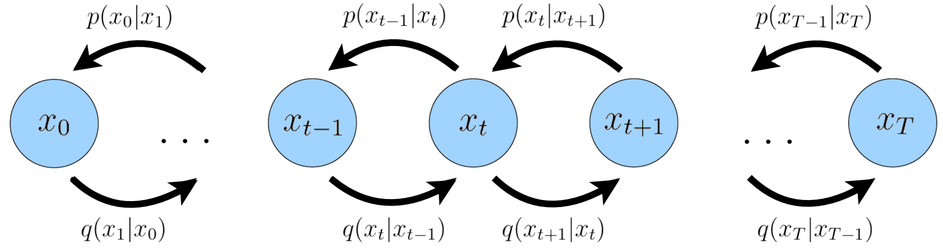


$$\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)) = \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \left[\|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\|_2^2 \right]$$

$$\boldsymbol{\mu}_q(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}$$

From derivation shown earlier

Variational Diffusion Models: Simplifying Training



$$\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)) = \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \left[\|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\|_2^2 \right]$$

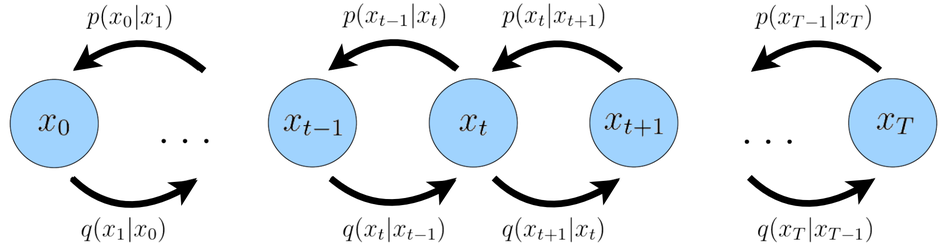
$$\boldsymbol{\mu}_q(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}$$

From derivation shown earlier

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)}{1-\bar{\alpha}_t}$$

Defined to match the above mean closely

Variational Diffusion Models: Simplifying Training



$$\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)) = \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \left[\|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\|_2^2 \right]$$

$$\boldsymbol{\mu}_q(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}$$

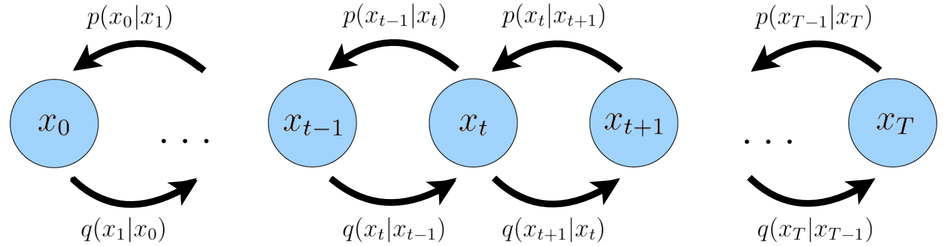
From derivation shown earlier

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t + \sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)}{1-\bar{\alpha}_t}$$

Defined to match the above mean closely

- $\hat{x}_\theta(x_t, t)$ is parametrized by a neural network.
  - Predicts the original input $\mathbf{x_0}$ from a noised input $\mathbf{x_t}$ at time t

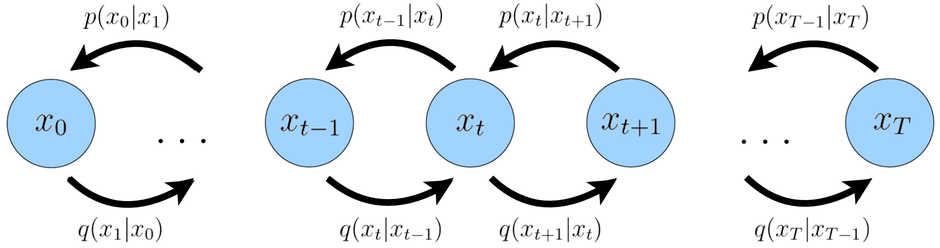
Variational Diffusion Models: Simplifying Training



$$\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))$$

$$= \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \left[\|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\|_2^2\right]$$

Variational Diffusion Models: Simplifying Training

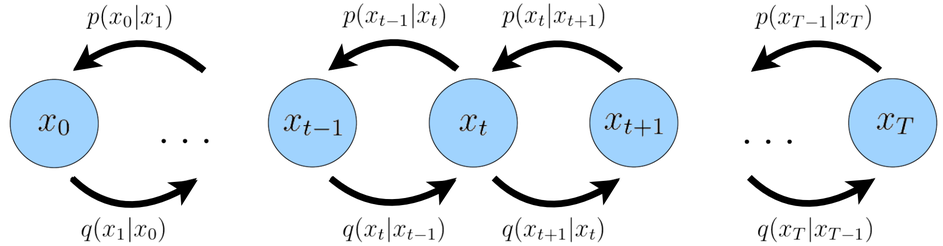


$$\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))$$

$$= \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \frac{\bar{\alpha}_{t-1}(1-\alpha_t)^2}{(1-\bar{\alpha}_t)^2} \left[\|\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \boldsymbol{x}_0\|_2^2\right]$$

Optimizing a diffusion model simplifies to learning a neural network to predict the original ground truth input from an arbitrary noised version of it

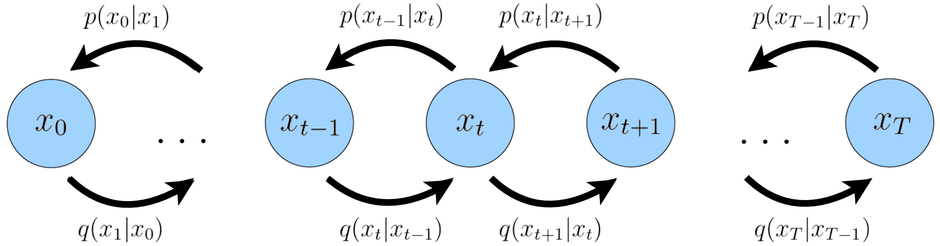
Variational Diffusion Models: Equivalent View



$$x_t \sim q(x_t \mid x_0) \quad = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon_0^*$$

$$\sim \mathcal{N}\left(\sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t)\mathbf{I}\right)$$

Variational Diffusion Models: Equivalent View



$$x_t \sim q(x_t \mid x_0) \quad = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon_0^*$$

$$\sim \mathcal{N}\left(\sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t)\mathbf{I}\right)$$

$$x_0 = \frac{x_t - \sqrt{1 - \bar{\alpha}_t} \epsilon_0^*}{\sqrt{\bar{\alpha}_t}}$$

Knowing the noise added and the noisy input is sufficient to give the denoised input

Variational Diffusion Models: Equivalent View

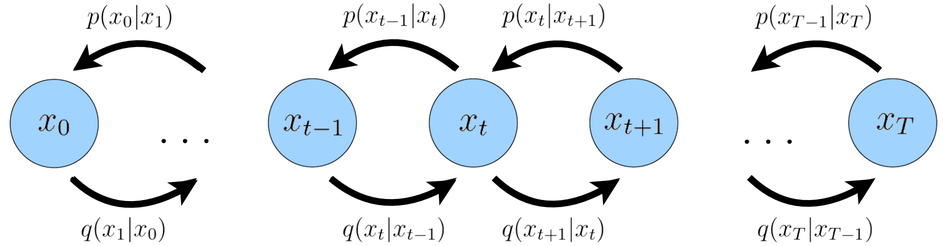
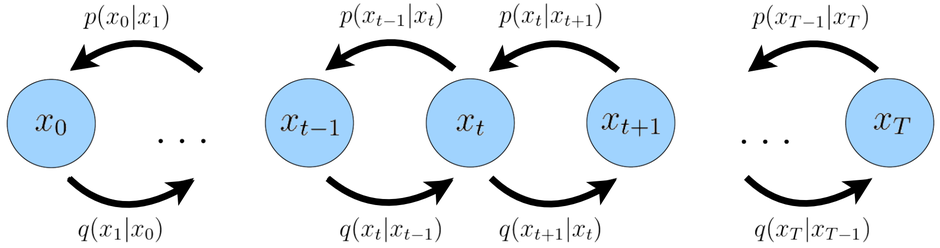


Image taken from https://arxiv.org/pdf/2208.11970

Variational Diffusion Models: Equivalent View



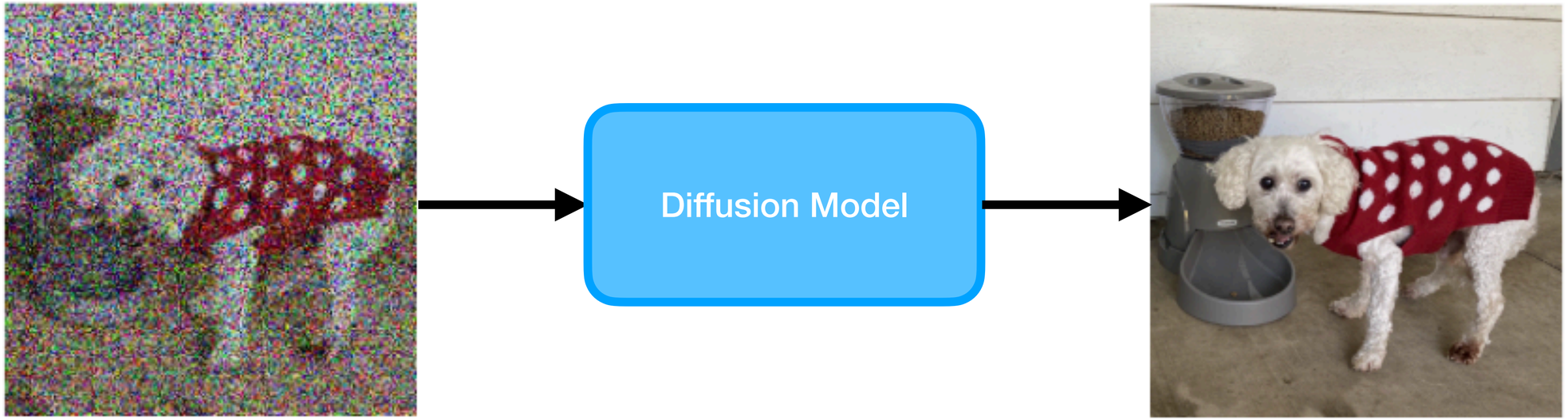
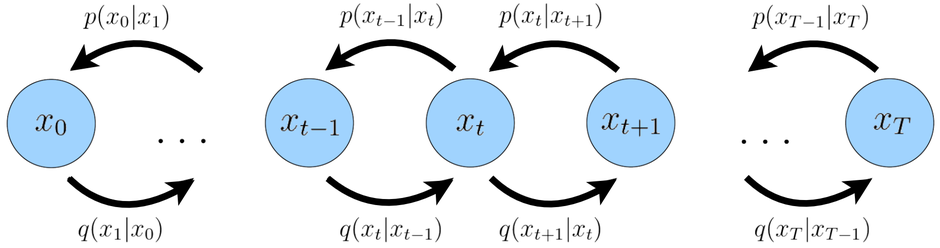


Image taken from https://arxiv.org/pdf/2208.11970

Variational Diffusion Models: Equivalent View



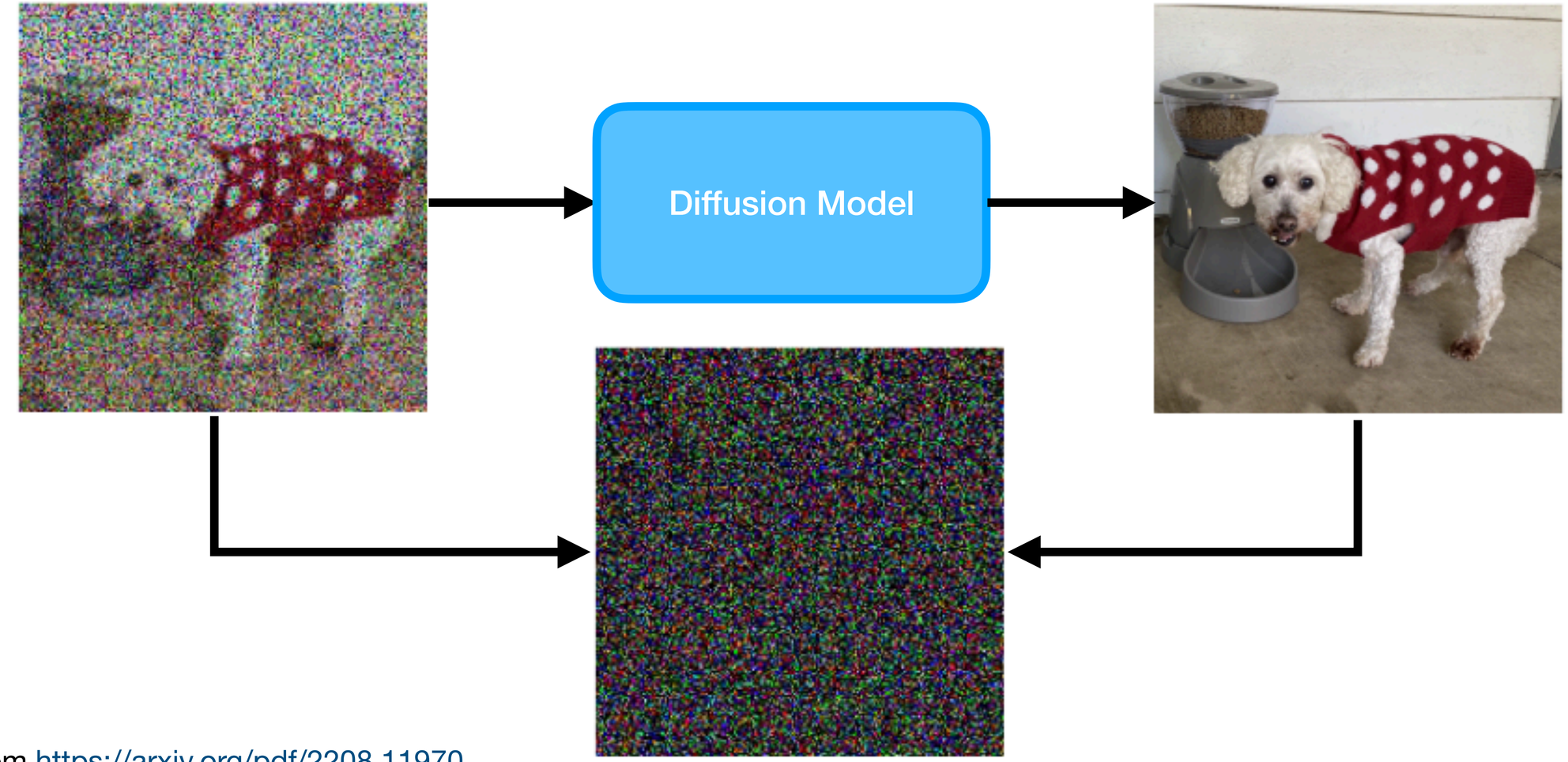
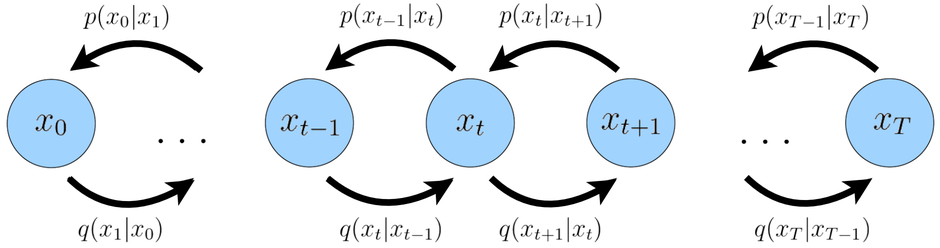


Image taken from https://arxiv.org/pdf/2208.11970

Variational Diffusion Models: Equivalent View



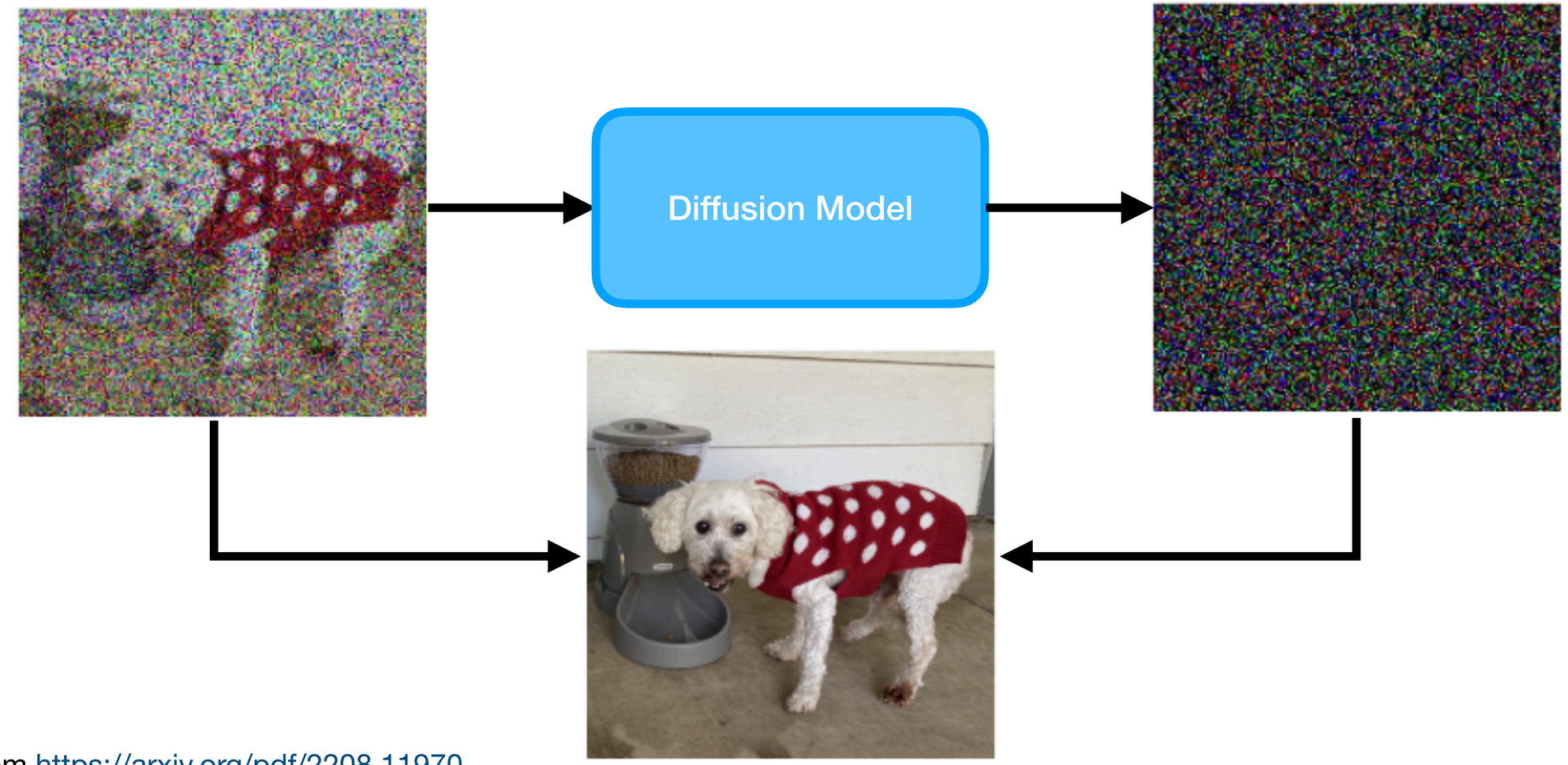
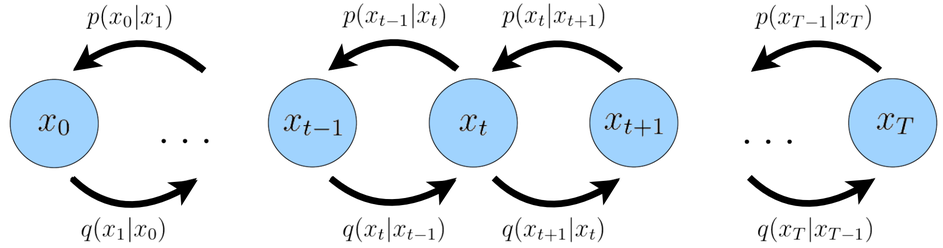


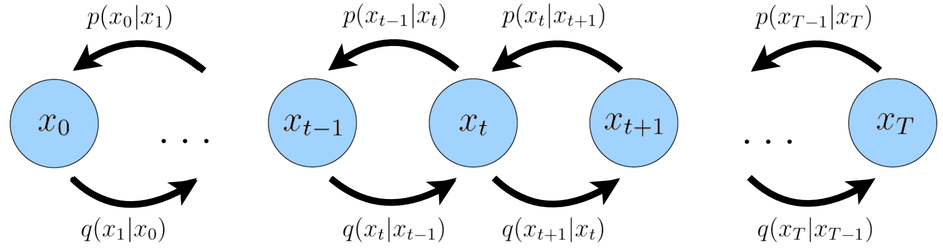
Image taken from https://arxiv.org/pdf/2208.11970

Variational Diffusion Models: Equivalent View



$$\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)) = \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \left[\|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\|_2^2 \right]$$

Variational Diffusion Models: Equivalent View

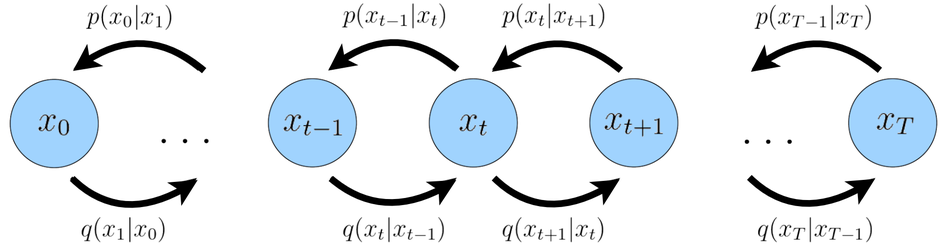


$$\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)) = \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \left[\|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\|_2^2 \right]$$

$$\boldsymbol{\mu}_q(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{1}{\sqrt{\alpha_t}} \boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}} \boldsymbol{\epsilon}_0$$

Alternate Formulation using equivalence

Variational Diffusion Models: Equivalent View



$$\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)) = \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \left[\|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\|_2^2 \right]$$

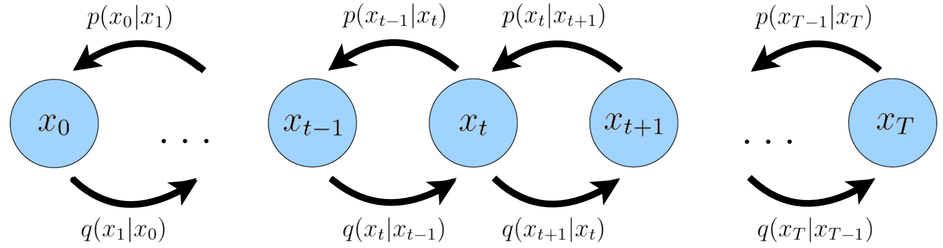
$$\boldsymbol{\mu}_q(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{1}{\sqrt{\alpha_t}} \boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}} \boldsymbol{\epsilon}_0$$

Alternate Formulation using equivalence

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}} \boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}} \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$$

Defined to match the above mean closely

Variational Diffusion Models: Equivalent View



$$\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)) = \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \left[\|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\|_2^2 \right]$$

$$\boldsymbol{\mu}_q(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{1}{\sqrt{\alpha_t}} \boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}} \boldsymbol{\epsilon}_0$$

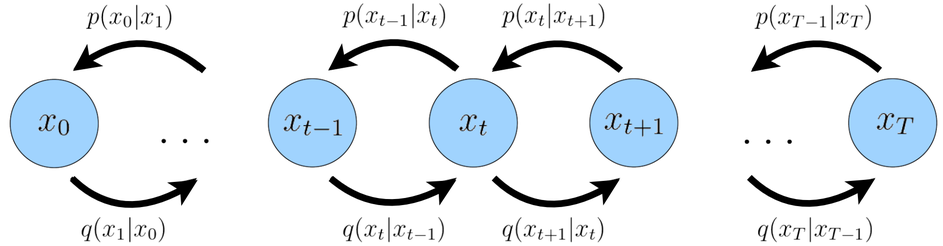
Alternate Formulation using equivalence

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}} \boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}} \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$$

Defined to match the above mean closely

- $\hat{\epsilon}_\theta(x_t, t)$ is parametrized by a neural network.
  - Predicts the added noise ϵ_0^* from a noised input $\mathbf{x_t}$ at time t

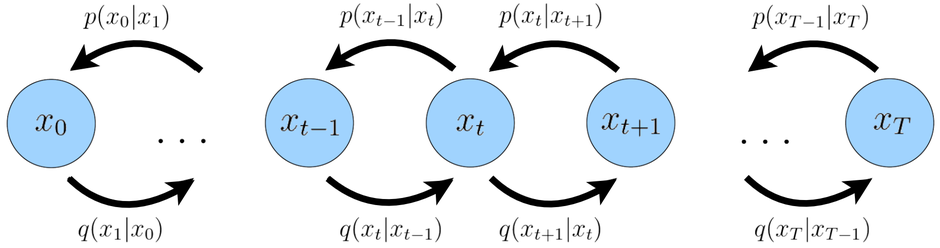
Variational Diffusion Models: Equivalent View



$$\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))$$

$$= \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \left[\|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\|_2^2\right]$$

Variational Diffusion Models: Equivalent View

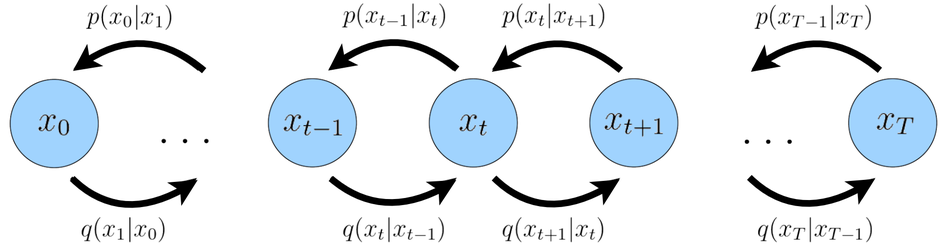


$$\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))$$

$$= \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2}{(1-\bar{\alpha}_t)\alpha_t} \left[\|\boldsymbol{\epsilon}_0 - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\|_2^2\right]$$

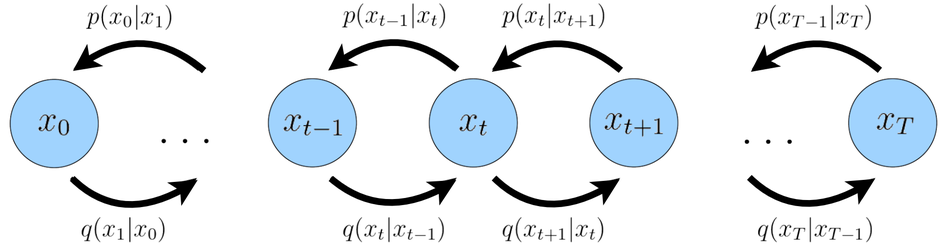
Equivalently, optimizing a diffusion model can also be simplified to learning a neural network to predict the source noise added to x_0 to get x_t

Variational Diffusion Models: Third Interpretation



$$\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)) = \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \left[\|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\|_2^2 \right]$$

Variational Diffusion Models: Third Interpretation

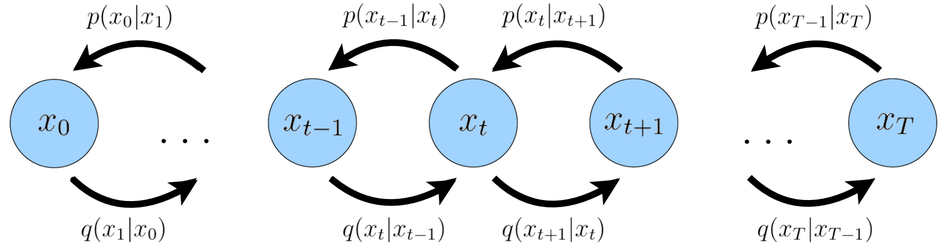


$$\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)) = \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \left[\|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\|_2^2 \right]$$

$$\boldsymbol{\mu}_q(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{1}{\sqrt{\alpha_t}} \boldsymbol{x}_t + \frac{1-\alpha_t}{\sqrt{\alpha_t}} \nabla \log p(\boldsymbol{x}_t)$$

Alternate Formulation using score function

Variational Diffusion Models: Third Interpretation



$$\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)) = \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \left[\|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\|_2^2 \right]$$

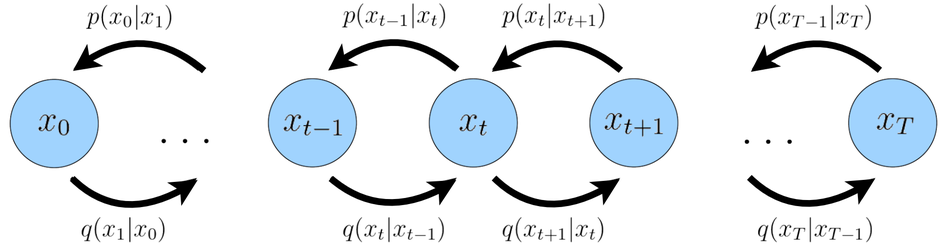
$$\boldsymbol{\mu}_q(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{1}{\sqrt{\alpha_t}} \boldsymbol{x}_t + \frac{1-\alpha_t}{\sqrt{\alpha_t}} \nabla \log p(\boldsymbol{x}_t)$$

Alternate Formulation using score function

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}} \boldsymbol{x}_t + \frac{1-\alpha_t}{\sqrt{\alpha_t}} \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$$

Defined to match the above mean closely

Variational Diffusion Models: Third Interpretation



$$\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)) = \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \left[\|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\|_2^2 \right]$$

$$\boldsymbol{\mu}_q(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{1}{\sqrt{\alpha_t}} \boldsymbol{x}_t + \frac{1-\alpha_t}{\sqrt{\alpha_t}} \nabla \log p(\boldsymbol{x}_t)$$

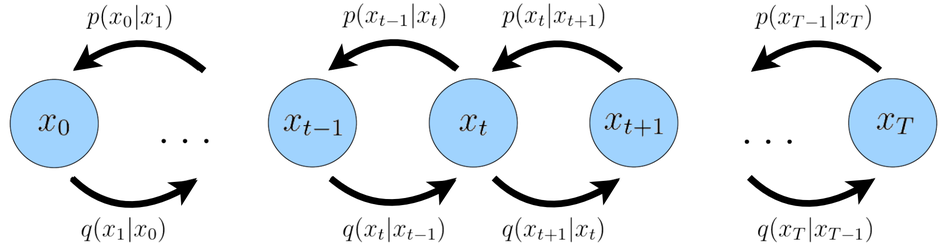
Alternate Formulation using score function

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}} \boldsymbol{x}_t + \frac{1-\alpha_t}{\sqrt{\alpha_t}} \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$$

Defined to match the above mean closely

- $s_\theta(x_t, t)$ is parametrized by a neural network.
  - Predicts the score function $\nabla \log p(x_t)$ from a noised input **x_t** at time t

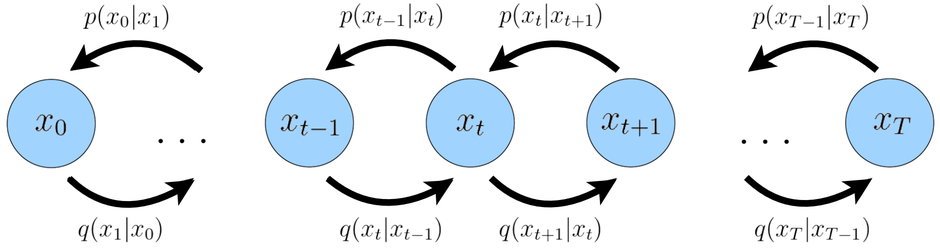
Variational Diffusion Models: Third Interpretation



$$\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))$$

$$= \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \left[\|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}_q\|_2^2 \right]$$

Variational Diffusion Models: Third Interpretation



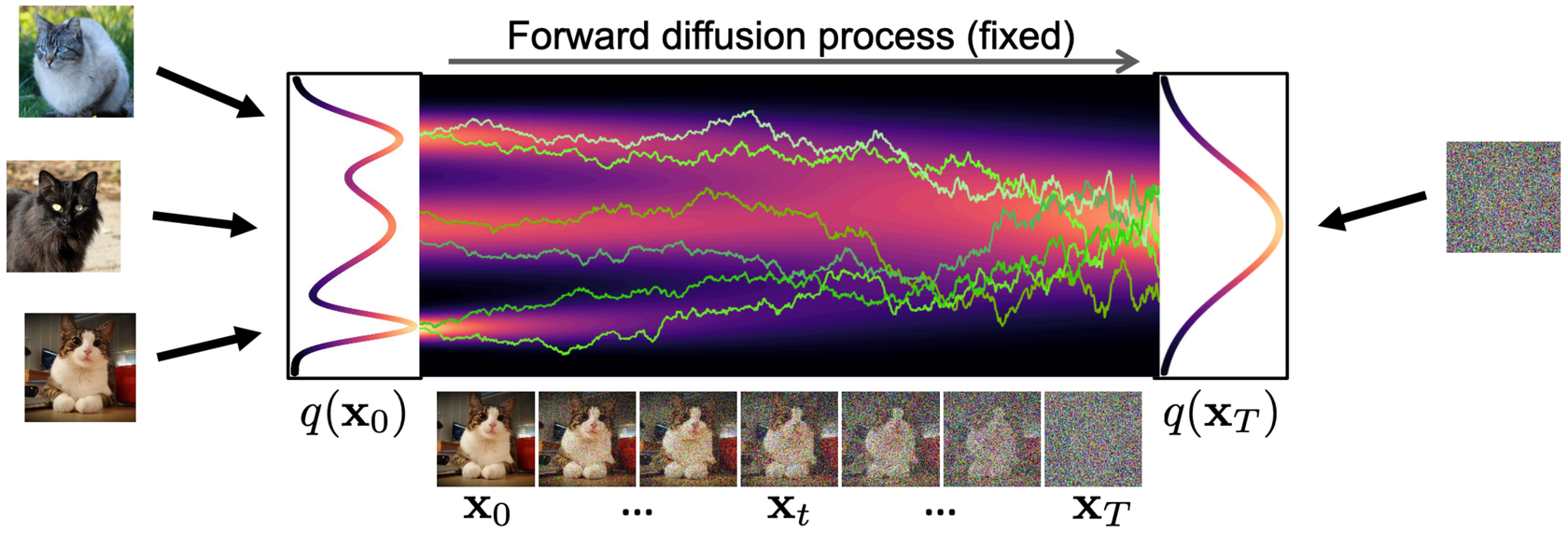
$$\arg\min_{\boldsymbol{\theta}} D_{\mathrm{KL}}(q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))$$

$$= \arg\min_{\boldsymbol{\theta}} \frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2}{\alpha_t} \left[\|\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) - \nabla \log p(\boldsymbol{x}_t)\|_2^2\right]$$

Equivalently, optimizing a diffusion model can also be simplified to learning a neural network to predict the score function, which is the gradient of x_t in data space, for a given t.

Alternative View of Diffusion Processes

- Both forward and reverse diffusion processes can be written as stochastic differential equations (SDEs)
- Reverse SDE governs warping from known simple noise distribution to the data distribution (can also be formulated using ODE)



Introduction to Diffusion Models

Part 2: Practice and Advance Topics

Leonid Sigal and Siddhesh Khandelwal

Practical Summary: Training

Algorithm 1 Training

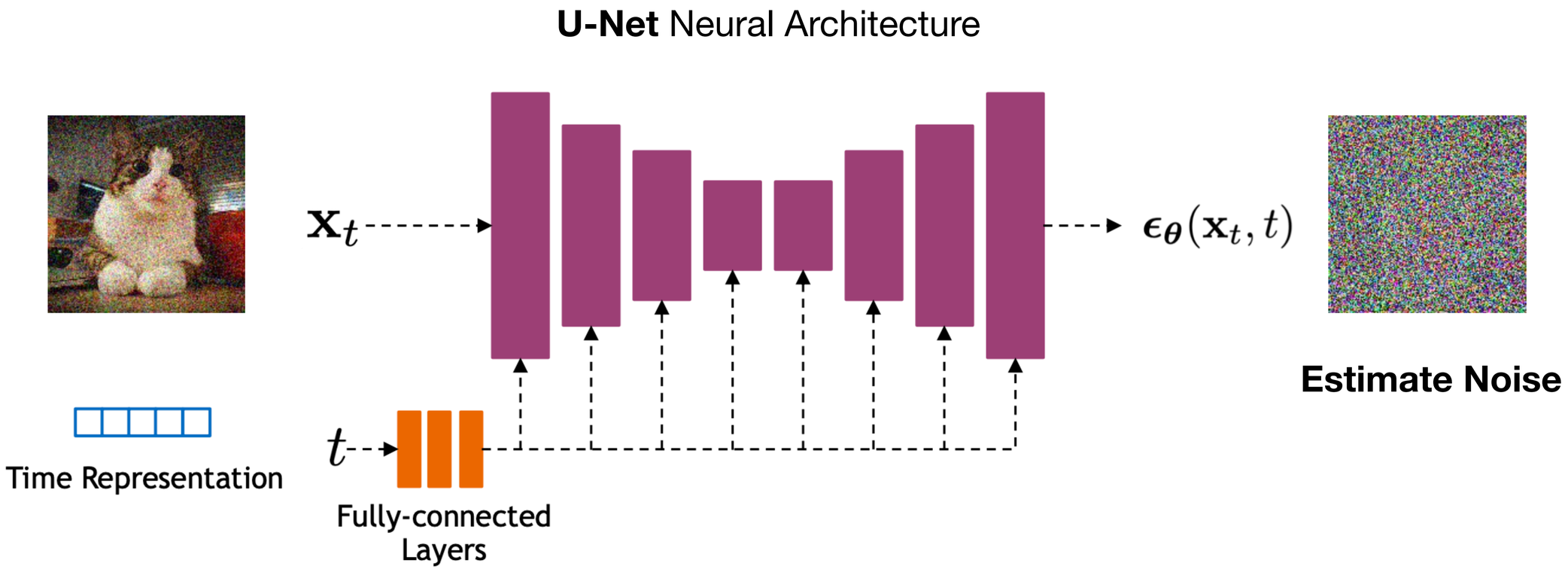
1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

Practical Summary: Sampling

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Real Example: Image Generation



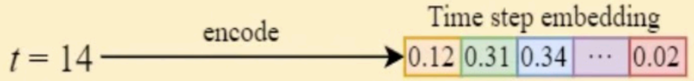
Real Example: Training for Image Generation

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

For each training step:

1. Randomly select a time step & encode it



From Steins (medium.com)

- **Step 1**: Choose and encode time step (usually uniformly from 1 to T=1000)

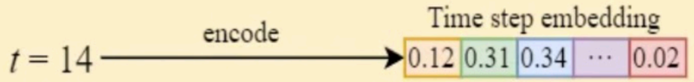
Real Example: Training for Image Generation

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

For each training step:

1. Randomly select a time step & encode it



	sin(t/1)	cos(t/1)	sin(t/10)	cos(t/10)	...
t=14	0.24	0.97	0.02	1.00	...

From Steins (medium.com)

- **Step 1**: Choose and encode time step (usually uniformly from 1 to T=1000)

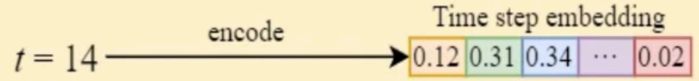
Real Example: Training for Image Generation

Algorithm 1 Training

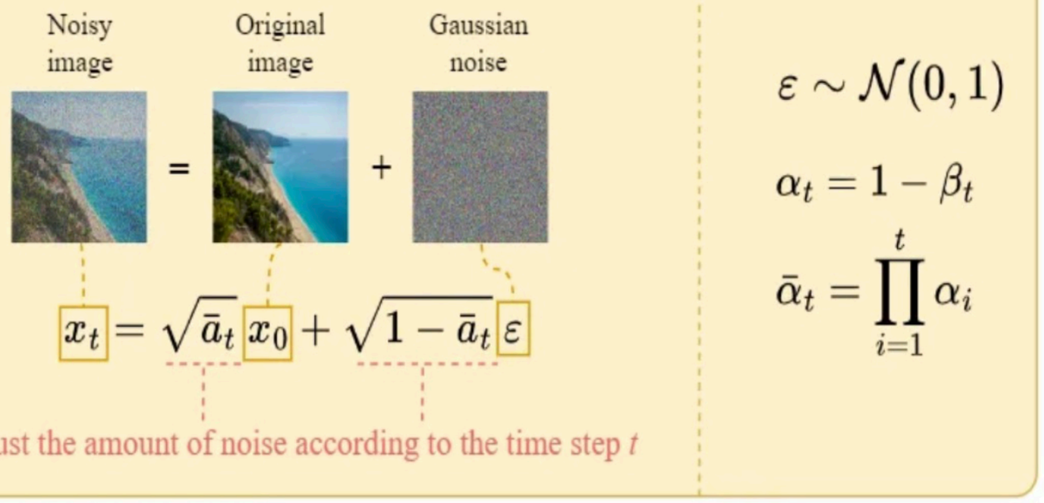
1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

For each training step:

1. Randomly select a time step & encode it



2. Add noise to image



From Steins (medium.com)

- **Step 2**: Add noise corresponding to sampled step t, this creates training data

Real Example: Training for Image Generation

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

For each training step:

1. Randomly select a time step & encode it

$t = 14$ → encode → Time step embedding: 0.12 | 0.31 | 0.34 | ⋯ | 0.02

2. Add noise to image

$$x_t = \sqrt{\bar{a}_t}\, x_0 + \sqrt{1-\bar{a}_t}\, \varepsilon$$

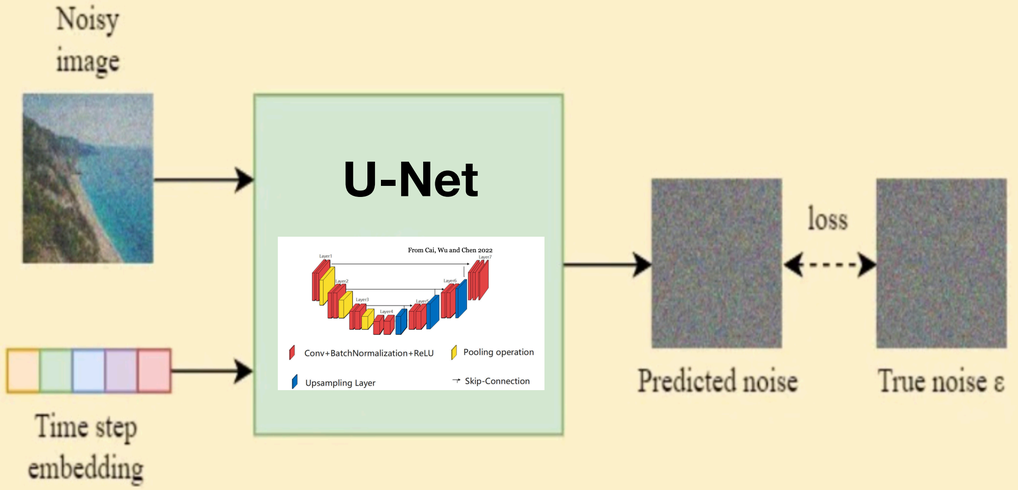
Adjust the amount of noise according to the time step t

$$\varepsilon \sim \mathcal{N}(0,1)$$

$$\alpha_t = 1 - \beta_t$$

$$\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$$

3. Train the UNet



From Steins (medium.com)

*step encoding simply added to the noisy sample representation

- **Step 3**: Train network to recover noise from corrupted sample

Real Example: Training for Image Generation

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

For each training step:

1. Randomly select a time s

$t = 14$

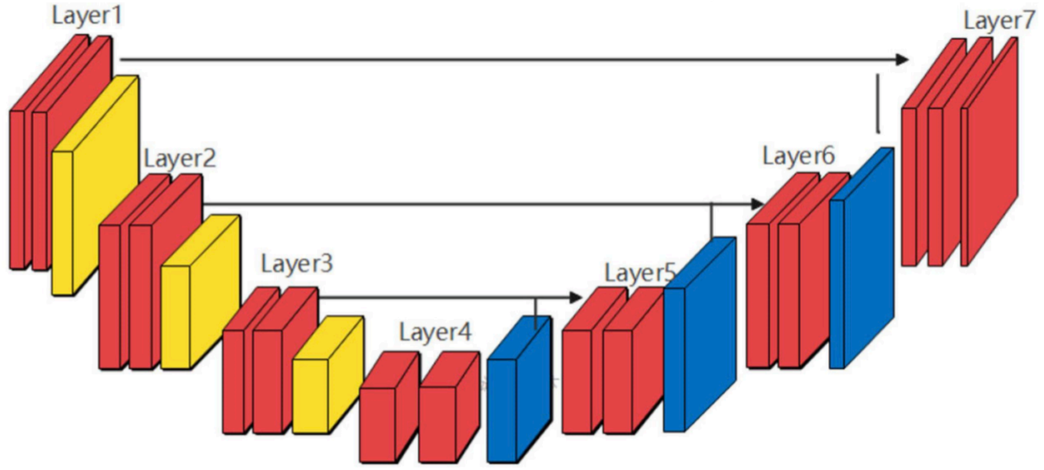
2. Add noise to image

Noisy image | Origi imag

$$x_t = \sqrt{\bar{a}_t}\, x_0$$

Adjust the amount of noise

From Cai, Wu and Chen 2022

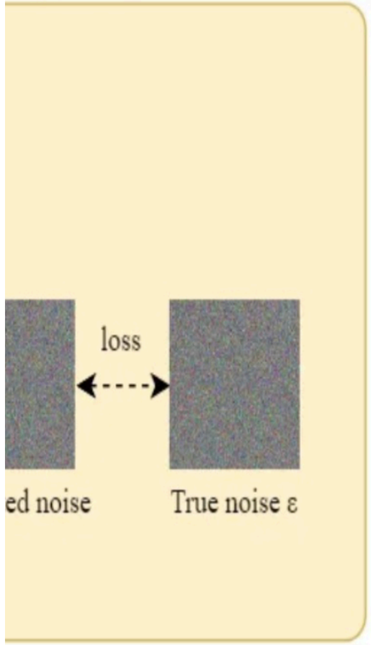


Conv+BatchNormalization+ReLU

Pooling operation

Upsampling Layer

Skip-Connection



eins (medium.com)

Real Example: Training for Image Generation

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

For each training step:

1. Randomly select a time s

$t = 14$

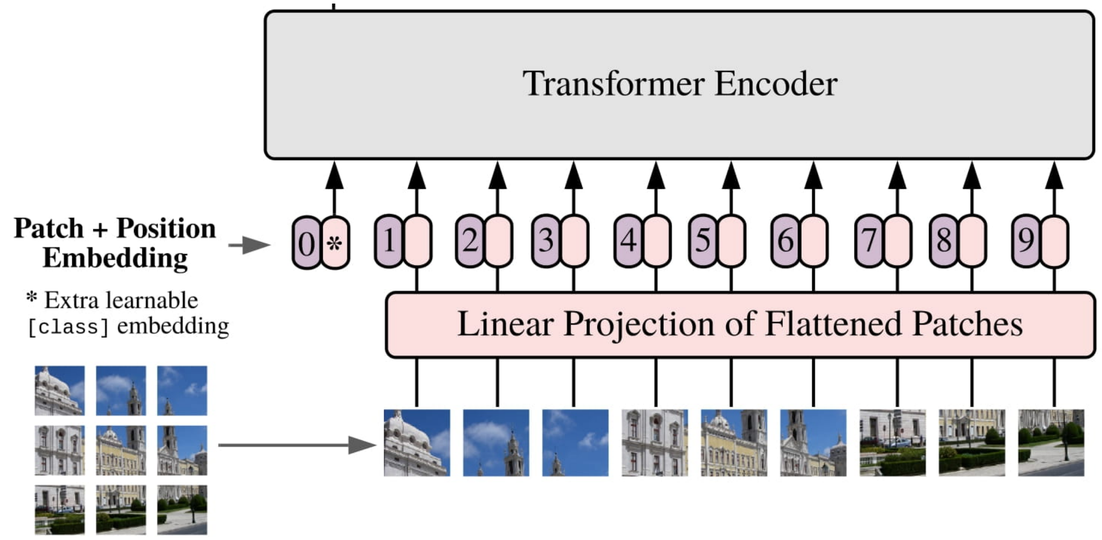
2. Add noise to image

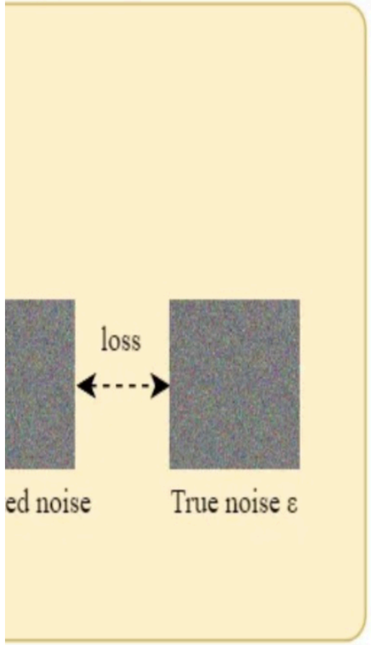
Noisy image

Origin image

$$x_t = \sqrt{\bar{a}_t}\, x_0$$

Adjust the amount of noise





eins (medium.com)

Real Example: Training for Image Generation

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

For each training step:

1. Randomly select a time s

$t = 14$

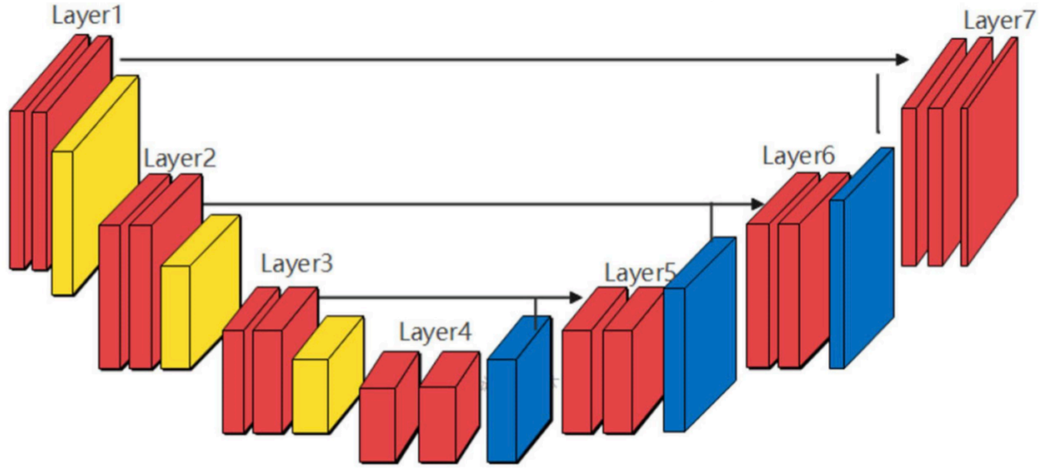
2. Add noise to image

Noisy image | Origi imag

$$x_t = \sqrt{\bar{a}_t}\, x_0$$

Adjust the amount of noise

From Cai, Wu and Chen 2022

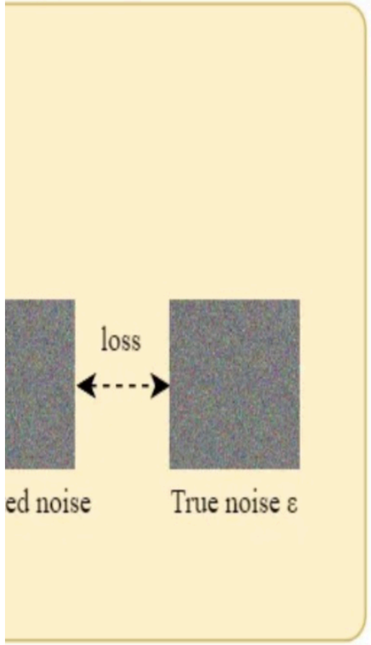


Conv+BatchNormalization+ReLU

Pooling operation

Upsampling Layer

Skip-Connection



eins (medium.com)

Real Example: Training for Image Generation

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

For each training step:

1. Randomly select a time step & encode it

$t = 14$ → encode → Time step embedding: 0.12 | 0.31 | 0.34 | ⋯ | 0.02

2. Add noise to image

Noisy image = Original image + Gaussian noise

$$x_t = \sqrt{\bar{a}_t}\, x_0 + \sqrt{1-\bar{a}_t}\, \varepsilon$$

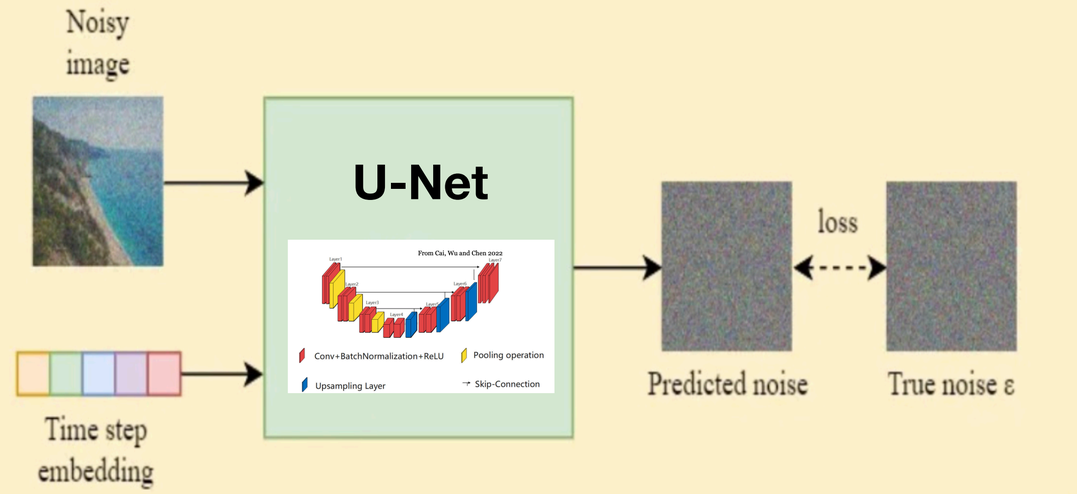
Adjust the amount of noise according to the time step t

$$\varepsilon \sim \mathcal{N}(0,1)$$

$$\alpha_t = 1 - \beta_t$$

$$\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$$

3. Train the UNet



From Steins (medium.com)

- **Important**: We are training one network (shared params) across all steps

Real Example: Inference for Image Generation

Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \dots, 1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
end for
return $\mathbf{x}_0$

1. Sample a Gaussian noise

$x_T \sim N(0, \mathrm{I})$ E.g. $T = 1000$

$x_{1000} \sim N(0, \mathrm{I})$

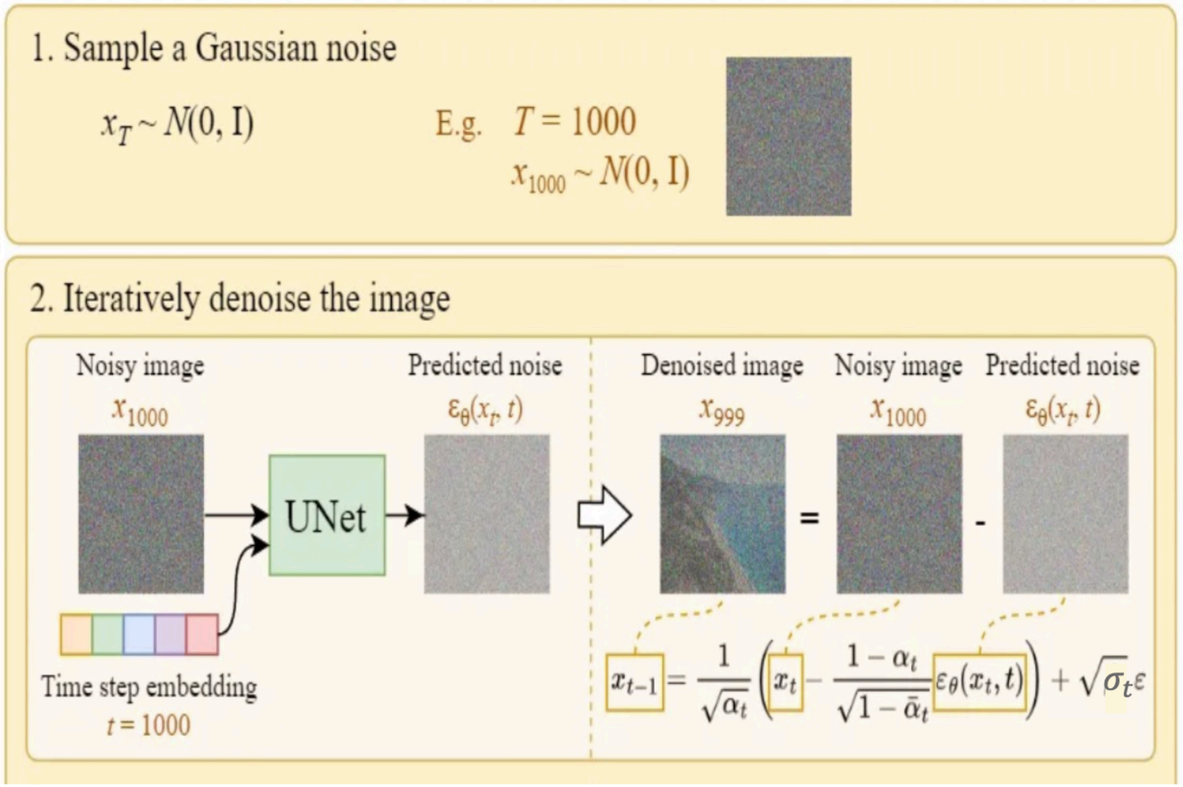
From Steins (medium.com)

- **Step 1**: Start from sampling pure noise

Real Example: Inference for Image Generation

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$



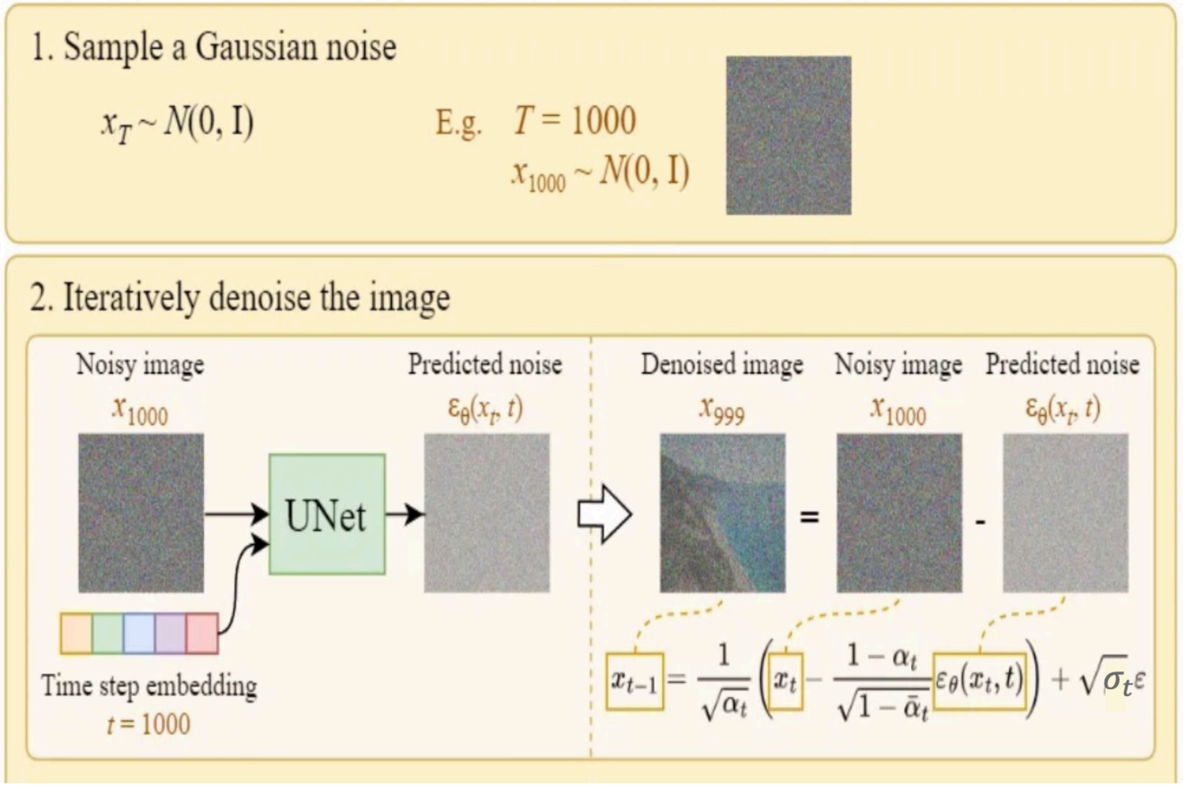
From Steins (medium.com)

- **Step 2**: Predict the noise using a trained (U-Net) model for t-1

Real Example: Inference for Image Generation

Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \dots, 1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
end for
return $\mathbf{x}_0$



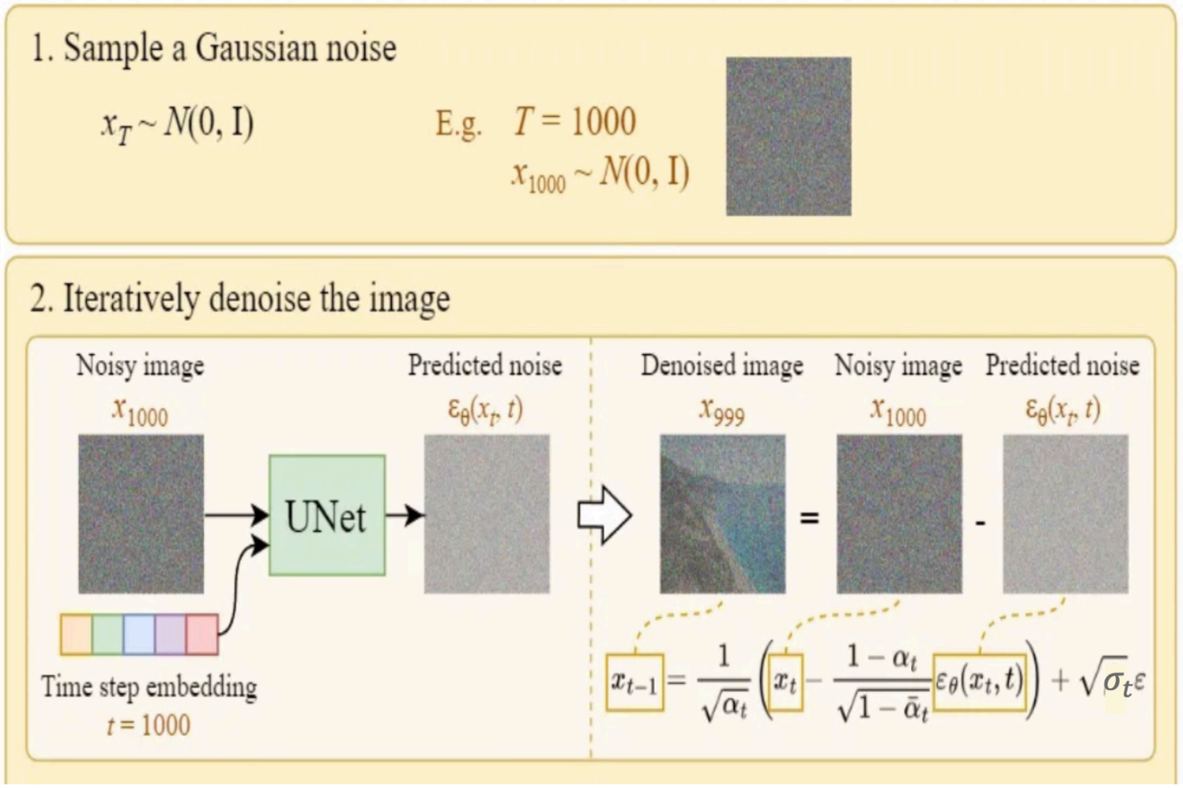
From Steins (medium.com)

- **Step 2**: Subtract predicted noise to obtain cleaner version of the sample

Real Example: Inference for Image Generation

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$



From Steins (medium.com)

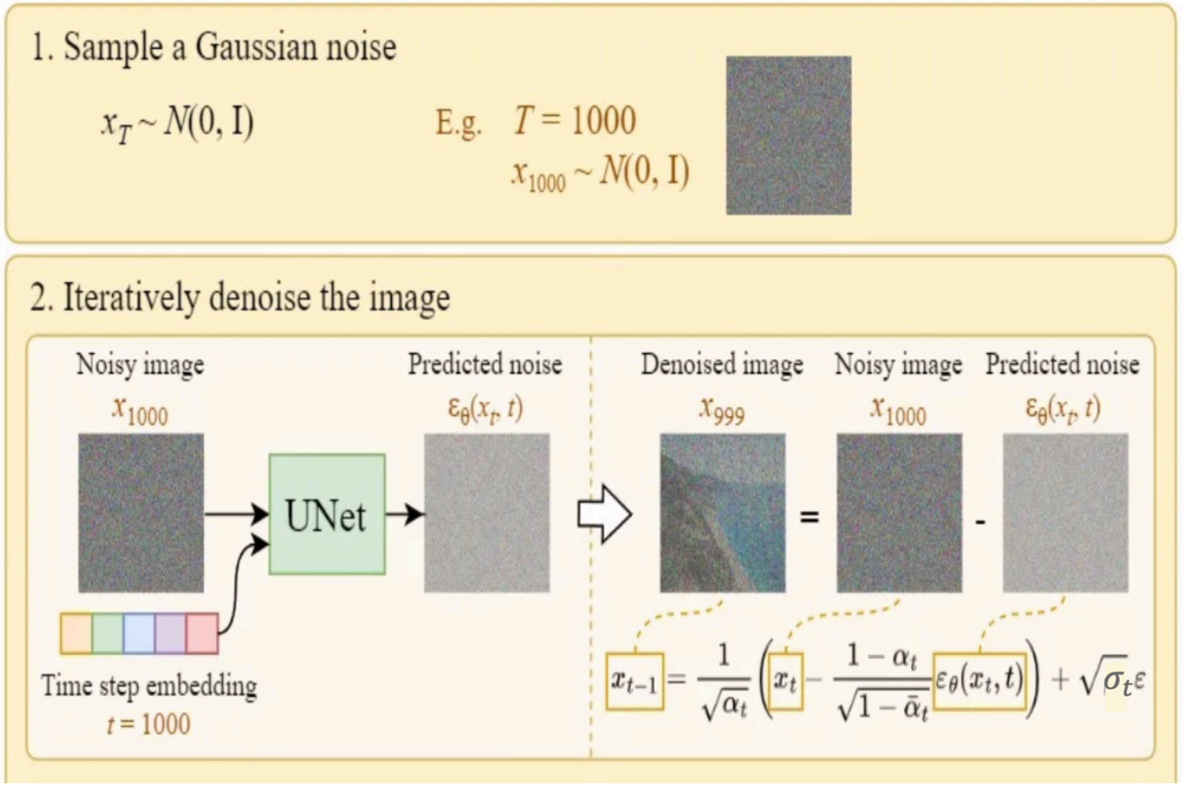
. . .

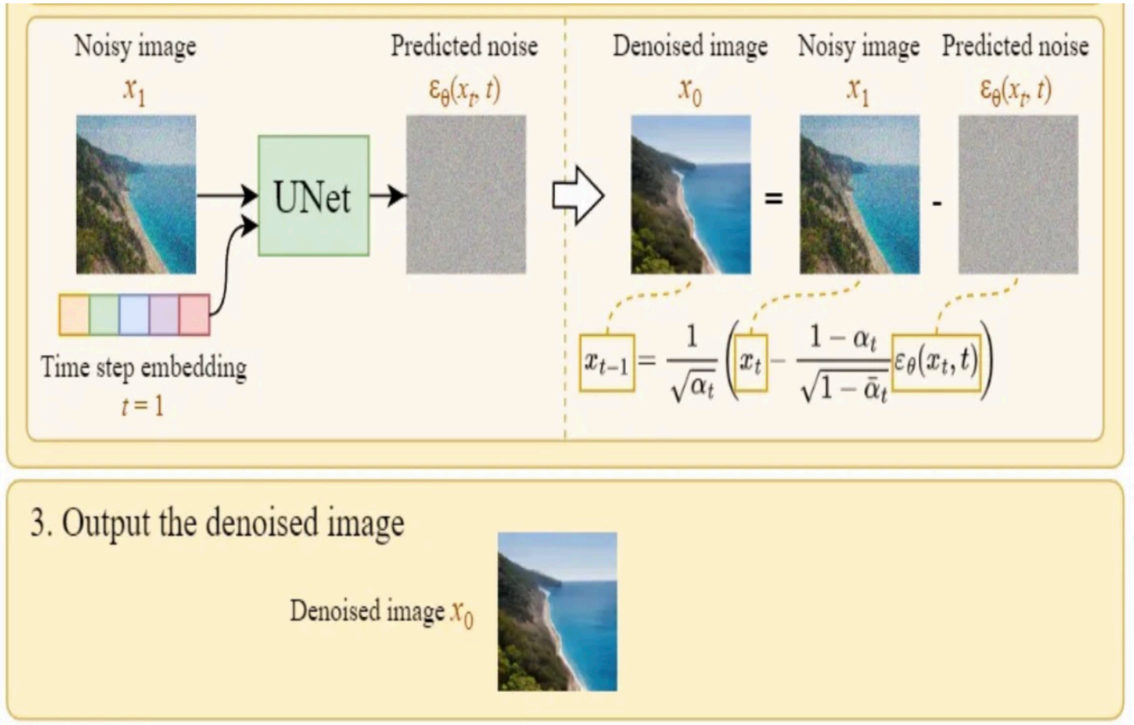
- Repeat many (T) times

Real Example: Inference for Image Generation

Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \dots, 1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
end for
return $\mathbf{x}_0$

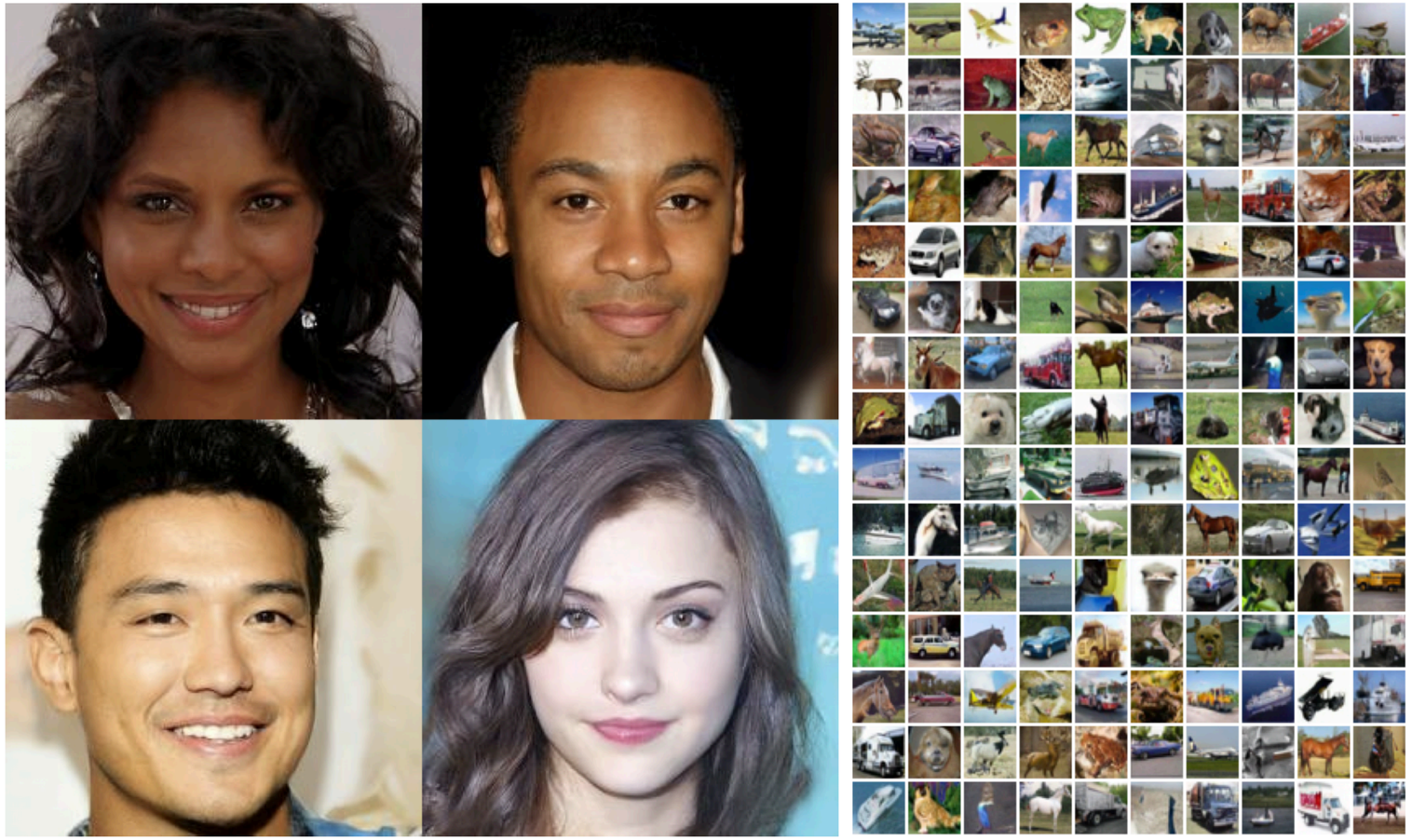




From Steins (medium.com)

- **Step 3**: The final output is a clean image corresponding to t=1

Real Example: Image Generation



[Ho, Jain and Abbeel (2020)]

Properties of Good Generative Model

High Quality Samples

Mode Coverage / Diversity

Fast Sampling

Accelerating Sampling

Advanced ODE / SDE Solvers

Distillation Techniques

Low-dim. / Latent Diffusion

Advanced Diffusion Processes

Accelerated Sampling: Denoising Diffusion Implicit Models (DDIM)

- **Theory**: Assume non-Markovian process (conditioning on original sample) and re-derive forward and backwards equations
- **Practice**:
  - Init: Let t = T = 1000, sample noise
    - **Step 1** : always predict t=1
    - **Step 2** : add noise to get to an intermediate step (like in training)
    - Continue this process
- **Nice property**: does not require any changes to training

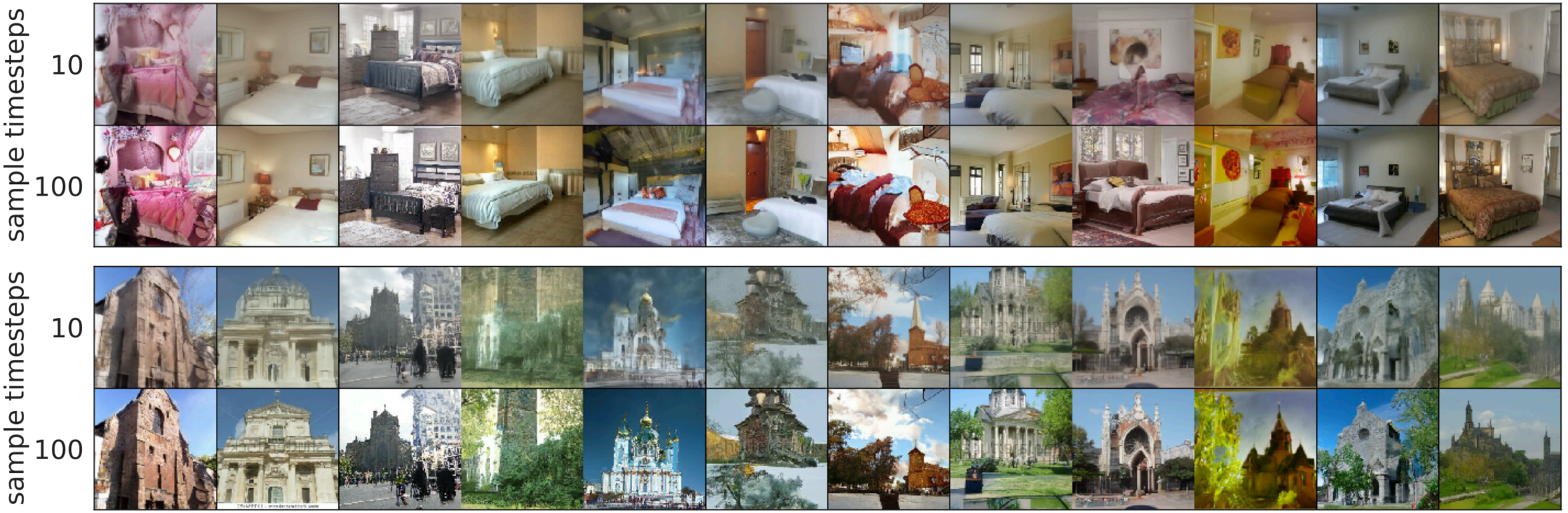
Advanced ODE / SDE Solvers

Distillation Techniques

Low-dim. / Latent Diffusion

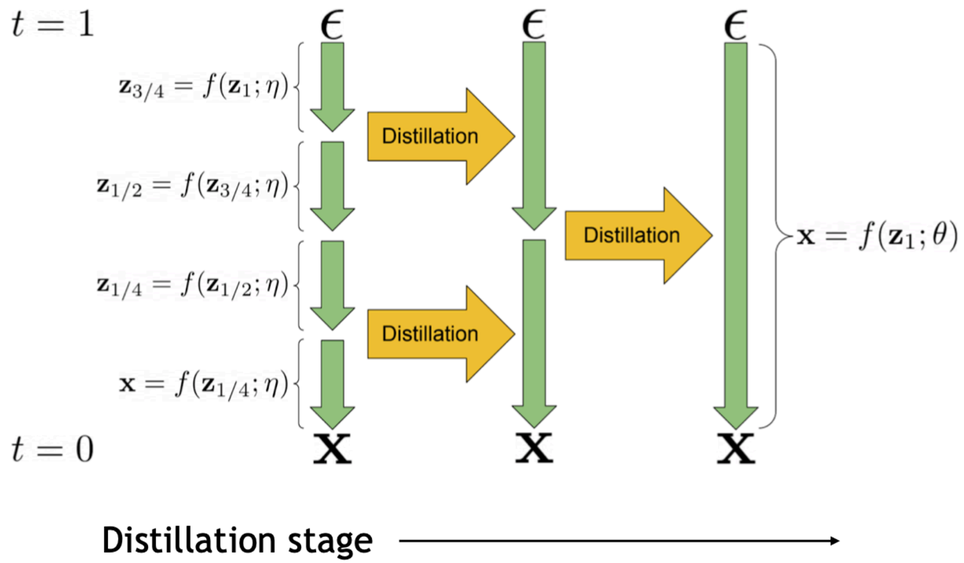
Advanced Diffusion Processes

Accelerated Sampling: Denoising Diffusion Implicit Models (DDIM)



Advanced ODE / SDE Solvers | Distillation Techniques | Low-dim. / Latent Diffusion | Advanced Diffusion Processes

Accelerated Sampling: Progressive Distillation



- Distill a sampler to the same model architecture but with fewer steps
  - e.g., a “student” model is learned to distill two adjacent sampling steps of the “teacher” model

Slide adopted from Denoising Diffusion Models Tutorial at CVPR 2023

[Salimans & Ho, “Progressive distillation for fast sampling of diffusion models”, ICLR 2022]

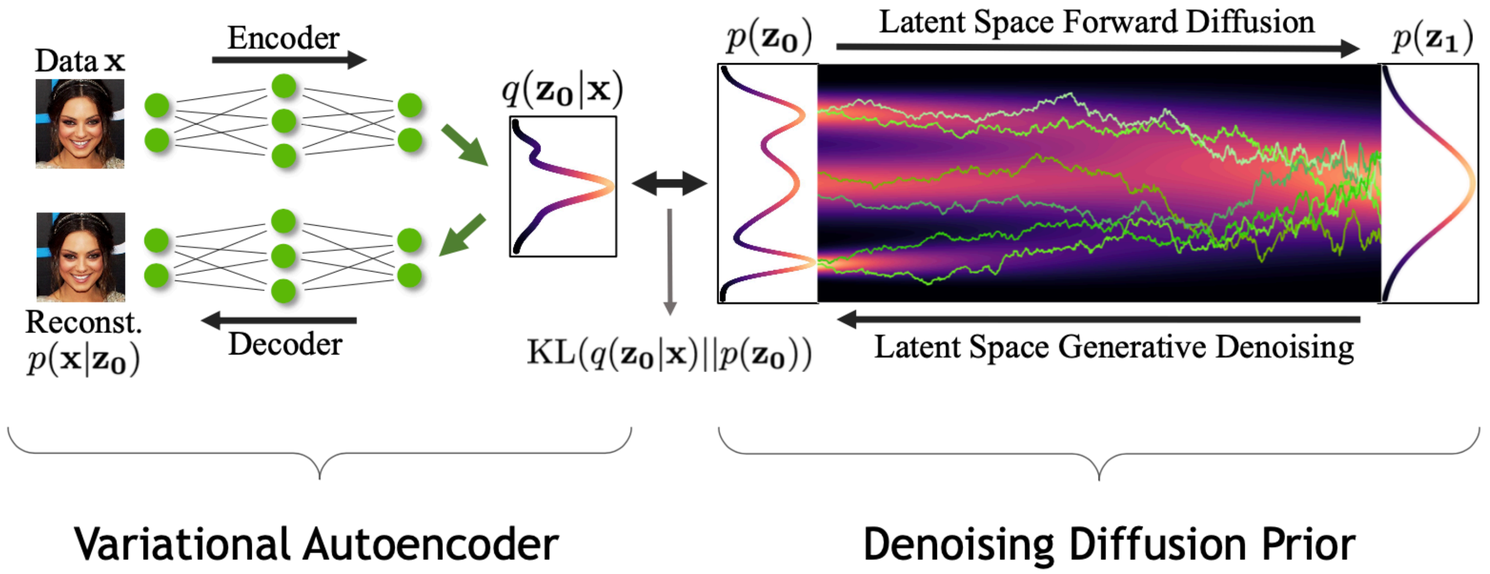
Advanced ODE / SDE Solvers

Distillation Techniques

Low-dim. / Latent Diffusion

Advanced Diffusion Processes

Accelerated Sampling: Latent Diffusion Model



- **Main idea**: Encode data into an embedding space and apply latent diffusion in this space

Slide adopted from Denoising Diffusion Models Tutorial at CVPR 2023

[Vahdat et al., “Score-based generative modeling in latent space”, NeurIPS 2021]

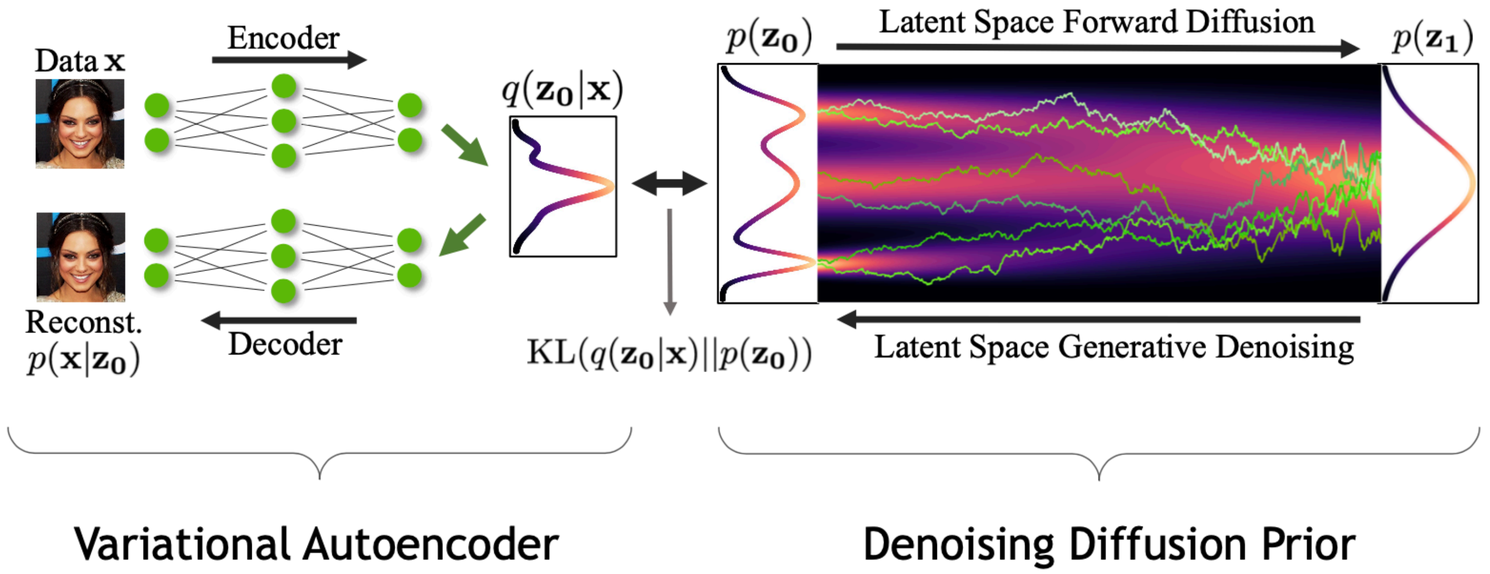
Advanced ODE / SDE Solvers

Distillation Techniques

Low-dim. / Latent Diffusion

Advanced Diffusion Processes

Accelerated Sampling: Latent Diffusion Model



- **Benefits**:
  - Distribution in latent space normal -> simpler denoising
  - Distribution in latent space low-dimensional -> faster sampling
  - Easier to apply to different data types (graphs, tables, text, 3d)

Slide adopted from Denoising Diffusion Models Tutorial at CVPR 2023

[Vahdat et al., “Score-based generative modeling in latent space”, NeurIPS 2021]

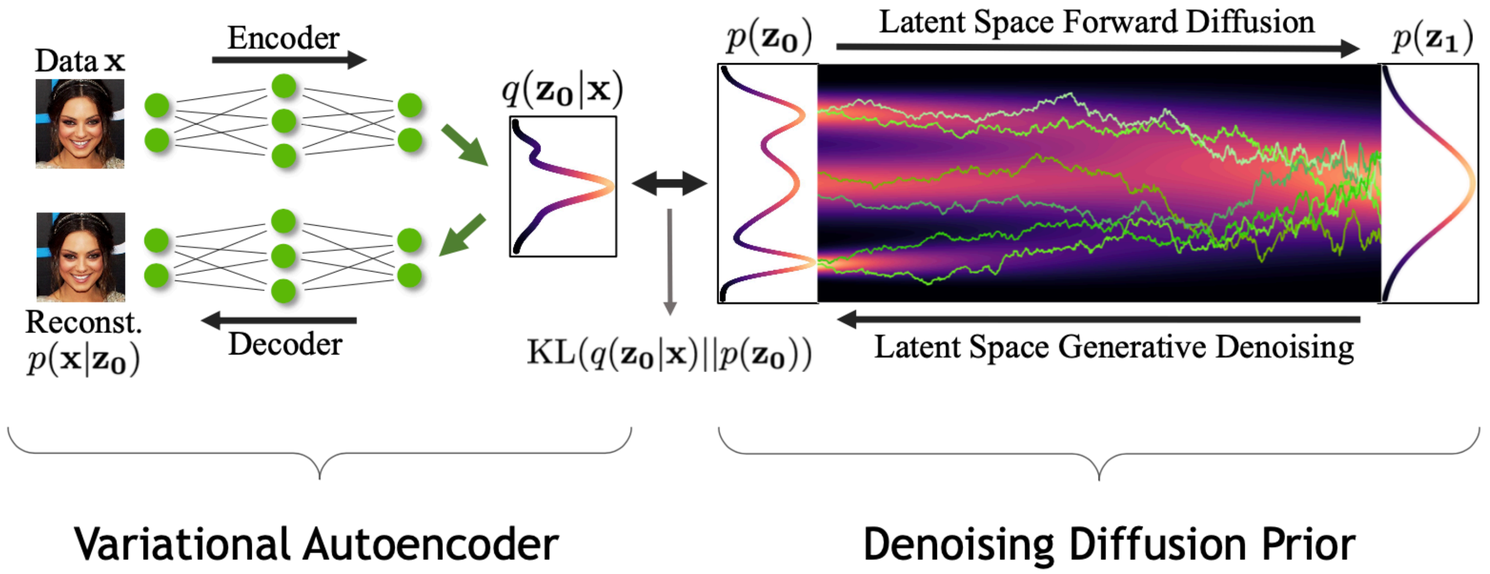
Advanced ODE / SDE Solvers

Distillation Techniques

Low-dim. / Latent Diffusion

Advanced Diffusion Processes

Accelerated Sampling: Latent Diffusion Model

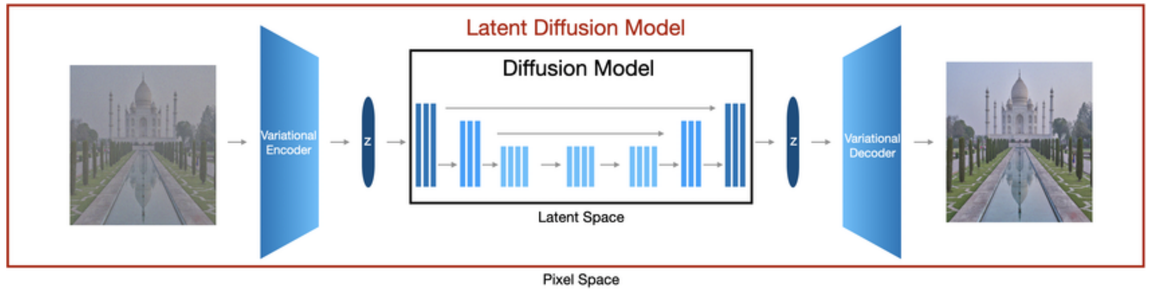


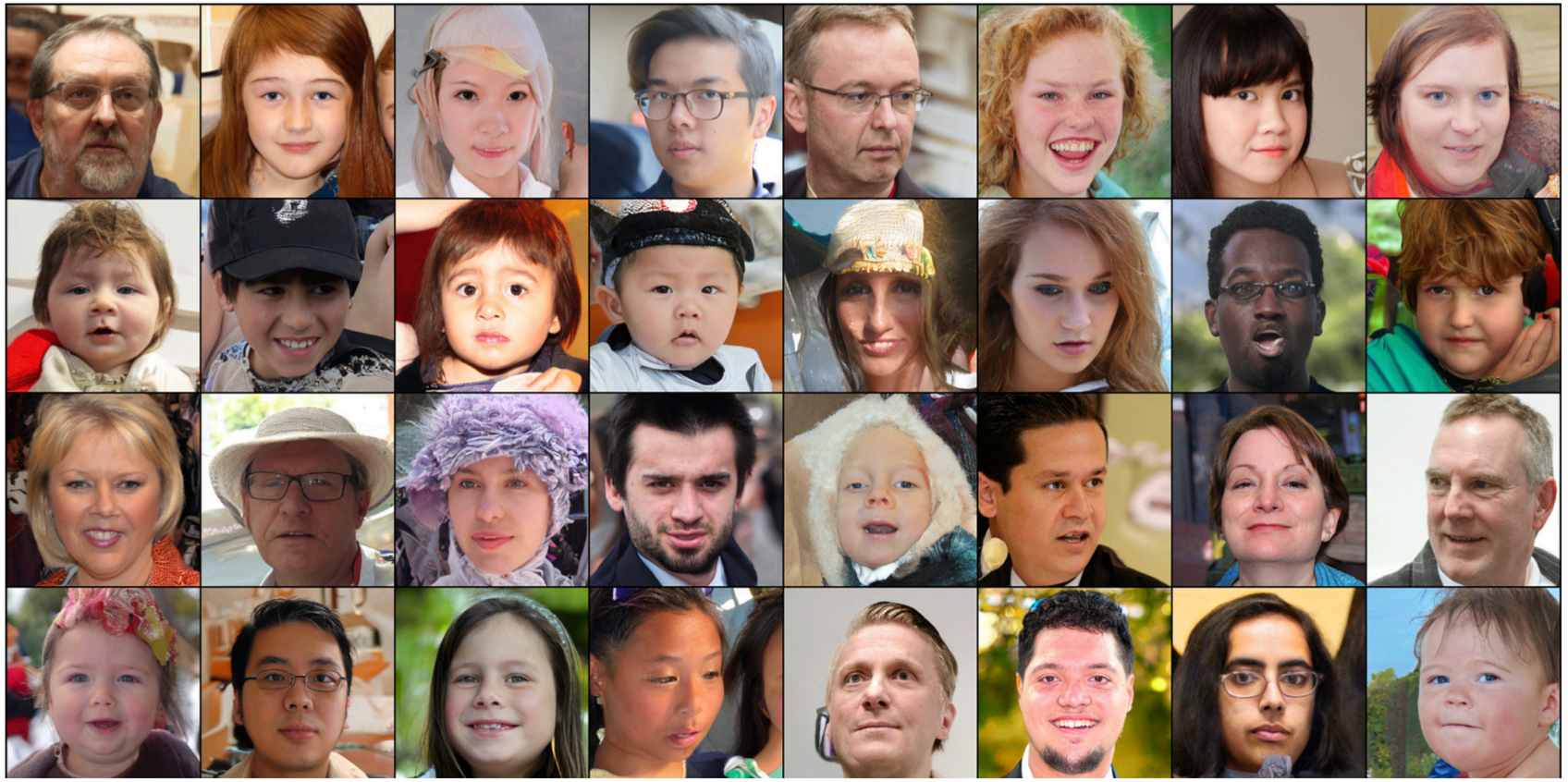
- **Two stage training**: train autoencoder first, then train the diffusion prior

Slide adopted from Denoising Diffusion Models Tutorial at CVPR 2023

[Rombach et al., “High-Resolution Image Synthesis with Latent Diffusion Models”, CVPR 2022]

Accelerated Sampling: Latent Diffusion Model





[Rombach et al., “High-Resolution Image Synthesis with Latent Diffusion Models”, CVPR 2022]

Accelerated Sampling: Latent Diffusion Model

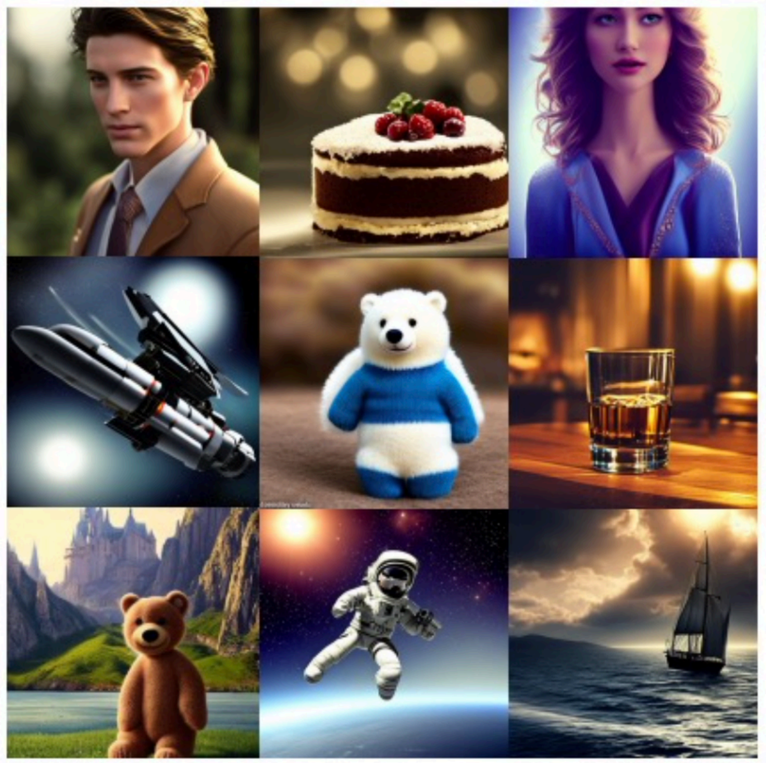
Advanced ODE / SDE Solvers

Distillation Techniques

Low-dim. / Latent Diffusion

Advanced Diffusion Processes

(a) 2 denoising steps



(b) 4 denoising steps

(c) 8 denoising steps

[Meng et al., "On Distillation of Guided Diffusion Models", CVPR 2023 (Award Candidate)]

Accelerating Sampling : Summary

Advanced ODE / SDE Solvers

e.g., **DDIM**

Simple
Requires no training
Could be inaccurate

Distillation Techniques

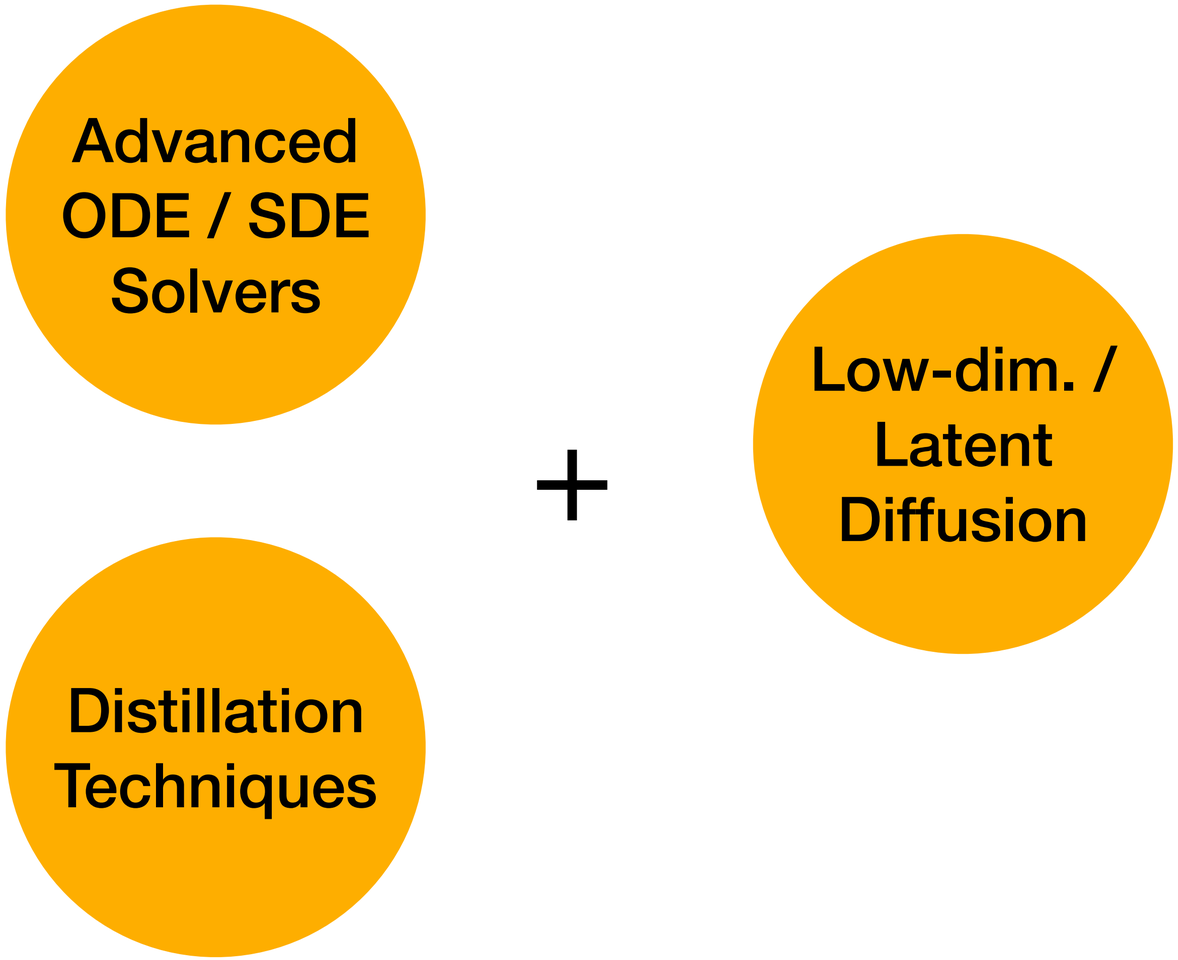
More complex to setup
Requires training
Usually more accurate

Low-dim. / Latent Diffusion

Moderate
Requires training of encoder
Usually works well

Advanced Diffusion Processes

Accelerating Sampling : Summary



Conditional Diffusion Models

- Generating raw samples of data is often not very useful in practice, much more practically useful is to generate data conditioned on some input (e.g., text to image generation)

DALL·E 2

“a propaganda poster depicting a cat dressed as french emperor napoleon holding a piece of cheese”

IMAGEN

“A photo of a raccoon wearing an astronaut helmet, looking out of the window at night.”

Slide adopted from Denoising Diffusion Models Tutorial at CVPR 2023

Conditional Diffusion Models:

Explicit Conditioning

Classifier Guidance

Classifier-free Guidance

Slide adopted from Denoising Diffusion Models Tutorial at CVPR 2023

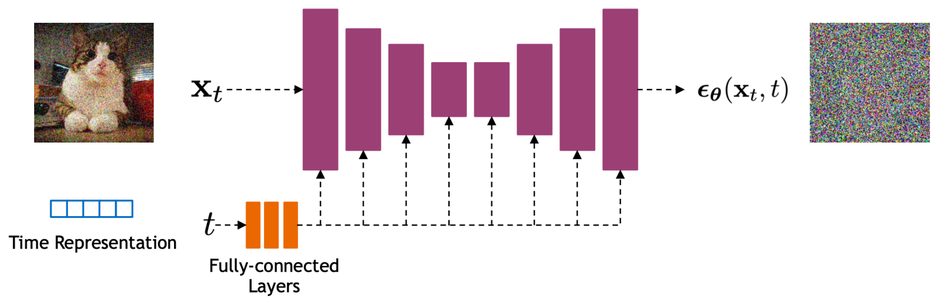
Conditional Diffusion Models: Explicit Conditioning

Explicit Conditioning

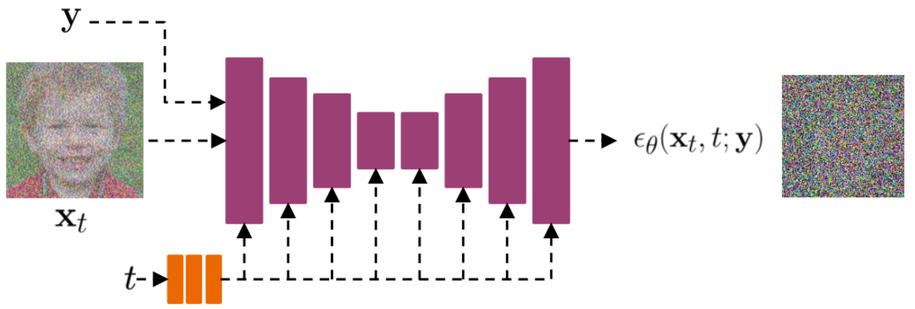
Classifier Guidance

Classifier-free Guidance

- Instead of training diffusion model to model ***p*(*x*)**, train it to model conditional distribution ***p*(*x*|*y*)**, where ***y*** is the conditioning variable (e.g., text) and ***x*** is the output (e.g., image) corresponding to the condition.
- Effectively requires no changes to the diffusion process (just data and fusion)



Unconditional Diffusion Model



Conditional Diffusion Model

Slide adopted from Denoising Diffusion Models Tutorial at CVPR 2023

Conditional Diffusion Models: Classifier Guidance

Explicit Conditioning

Classifier Guidance

Classifier-free Guidance

- When sampling from a diffusion model, we need an estimate of gradient p(x) with respect to the sample
- Using Bayes' rule we can express the gradient for conditional diffusion as

$$p(x_t|y) = \frac{p(y|x_t)p(x_t)}{p(y)}$$

$$\nabla_{x_t} \log p(x_t|y) = \nabla_{x_t} \log p(y|x_t) + \nabla_{x_t} \log p(x_t)$$

Slide adopted from Denoising Diffusion Models Tutorial at CVPR 2023

Conditional Diffusion Models: Classifier Guidance

Explicit Conditioning

Classifier Guidance

Classifier-free Guidance

- When sampling from a diffusion model, we need an estimate of gradient p(x) with respect to the sample
- Using Bayes' rule we can express the gradient for conditional diffusion as

$$p(x_t|y) = \frac{p(y|x_t)p(x_t)}{p(y)}$$

$$\nabla_{x_t} \log p(x_t|y) = \nabla_{x_t} \log p(y|x_t) + \nabla_{x_t} \log p(x_t)$$

Gradient of pre-trained classifier

Conditional Diffusion Models: Classifier Guidance

Explicit Conditioning

Classifier Guidance

Classifier-free Guidance

- When sampling from a diffusion model, we need an estimate of gradient p(x) with respect to the sample
- Using Bayes' rule we can express the gradient for conditional diffusion as

$$p(x_t|y) = \frac{p(y|x_t)p(x_t)}{p(y)}$$

$$\nabla_{x_t} \log p(x_t|y) = \nabla_{x_t} \log p(y|x_t) + \nabla_{x_t} \log p(x_t)$$

Standard diffusion gradient

Slide adopted from Denoising Diffusion Models Tutorial at CVPR 2023

Conditional Diffusion Models: Classifier Guidance

Explicit Conditioning

Classifier Guidance

Classifier-free Guidance

- When sampling from a diffusion model, we need an estimate of gradient p(x) with respect to the sample
- Using Bayes' rule we can express the gradient for conditional diffusion as

$$p(x_t|y) = \frac{p(y|x_t)p(x_t)}{p(y)}$$

$$\nabla_{x_t} \log p(x_t|y) = w \nabla_{x_t} \log p(y|x_t) + \nabla_{x_t} \log p(x_t)$$

Strength of guidance

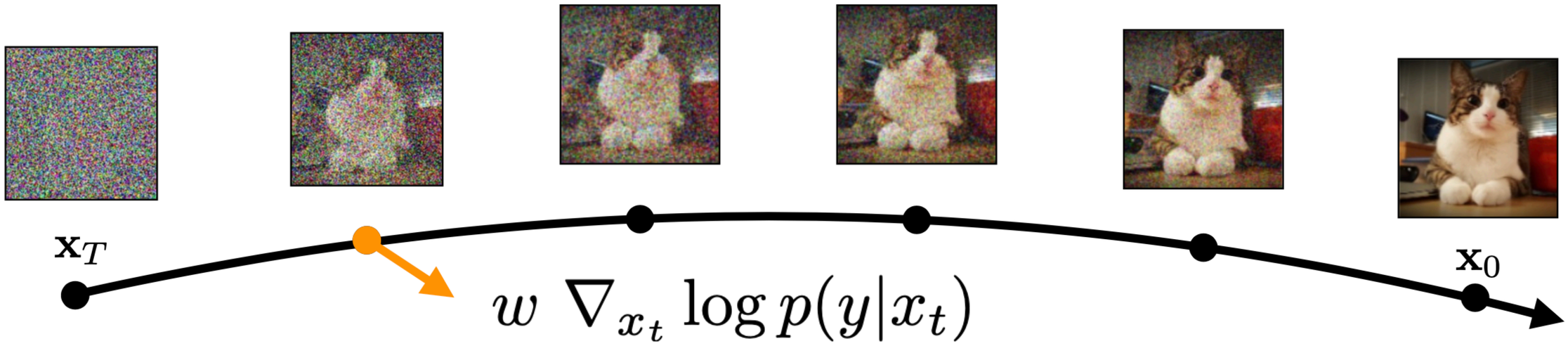
Slide adopted from Denoising Diffusion Models Tutorial at CVPR 2023

Conditional Diffusion Models: Classifier Guidance

Explicit Conditioning

Classifier Guidance

Classifier-free Guidance



- **Intuition**: Perturb the inference trajectory (series of de-noising steps) at each step using feedback from a pre-trained classifier.

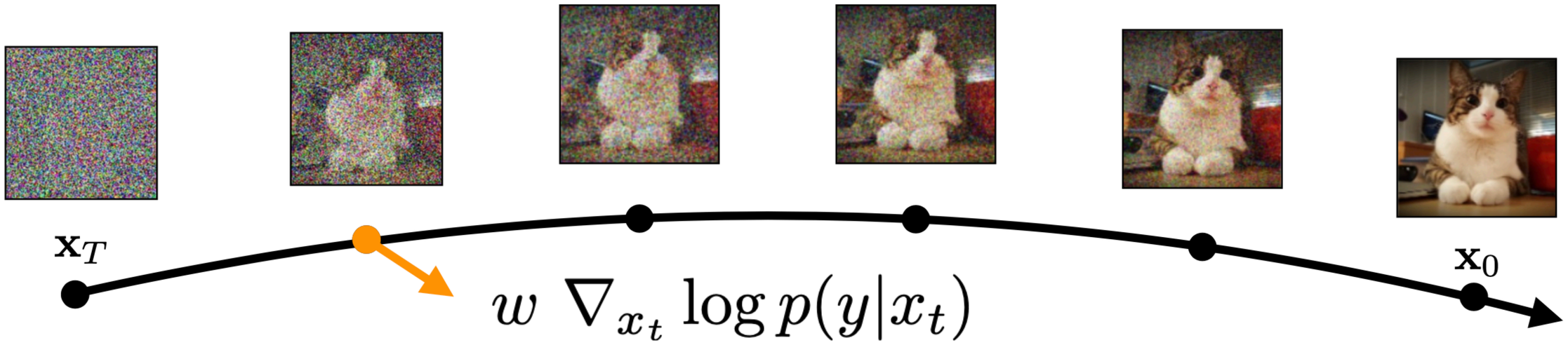
Slide adopted from EECS 442 Fall 2023

Conditional Diffusion Models: Classifier Guidance

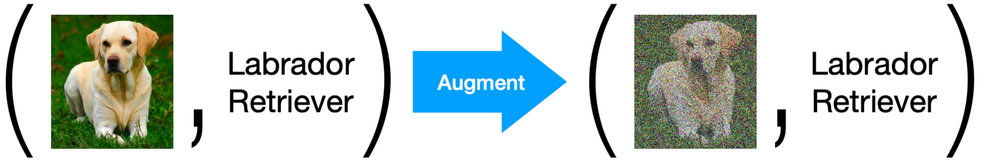
Explicit Conditioning

Classifier Guidance

Classifier-free Guidance



- **Issue**: Need to fine-tune classifier with noisy samples



Slide adopted from EECS 442 Fall 2023

Conditional Diffusion Models: Classifier Guidance

Explicit Conditioning

Classifier Guidance

Classifier-free Guidance

Guidance Weight 1.0

Guidance Weight 10.0

Slide adopted from EECS 442 Fall 2023

[Dhariwal and Nichol, “Diffusion Models Beat GANs on Image Synthesis”]

Conditional Diffusion Models: Classifier-free Guidance

Explicit Conditioning

Classifier Guidance

Classifier-free Guidance

$$\nabla_{x_t} \log p(x_t|y) = \nabla_{x_t} \log p(y|x_t) + \nabla_{x_t} \log p(x_t)$$

Bayes' Rule

Slide adopted from Denoising Diffusion Models Tutorial at CVPR 2023

Conditional Diffusion Models: Classifier-free Guidance

Explicit Conditioning

Classifier Guidance

Classifier-free Guidance

$$\nabla_{x_t} \log p(x_t|y) = \nabla_{x_t} \log p(y|x_t) + \nabla_{x_t} \log p(x_t)$$

Bayes' Rule

Bayes' Rule (again)

$$\nabla_{x_t} \log p(y|x_t) = \nabla_{x_t} \log p(x_t|y) - \nabla_{x_t} \log p(x_t)$$

Slide adopted from Denoising Diffusion Models Tutorial at CVPR 2023

Conditional Diffusion Models: Classifier-free Guidance

Explicit Conditioning

Classifier Guidance

Classifier-free Guidance

$$\nabla_{x_t} \log p(x_t|y) = \nabla_{x_t} \log p(y|x_t) + \nabla_{x_t} \log p(x_t)$$

Bayes' Rule

Bayes' Rule (again)

$$\nabla_{x_t} \log p(y|x_t) = \nabla_{x_t} \log p(x_t|y) - \nabla_{x_t} \log p(x_t)$$

Standard diffusion gradient

Slide adopted from Denoising Diffusion Models Tutorial at CVPR 2023

Conditional Diffusion Models: Classifier-free Guidance

Explicit Conditioning

Classifier Guidance

Classifier-free Guidance

$$\nabla_{x_t} \log p(x_t|y) = \nabla_{x_t} \log p(y|x_t) + \nabla_{x_t} \log p(x_t)$$

Bayes' Rule

Bayes' Rule (again)

$$\nabla_{x_t} \log p(y|x_t) = \nabla_{x_t} \log p(x_t|y) - \nabla_{x_t} \log p(x_t)$$

Conditional diffusion gradient

Slide adopted from Denoising Diffusion Models Tutorial at CVPR 2023

Conditional Diffusion Models: Classifier-free Guidance

Explicit Conditioning

Classifier Guidance

Classifier-free Guidance

$$\nabla_{x_t} \log p(x_t|y) = \nabla_{x_t} \log p(y|x_t) + \nabla_{x_t} \log p(x_t)$$

Bayes' Rule

Bayes' Rule (again)

$$\nabla_{x_t} \log p(y|x_t) = \nabla_{x_t} \log p(x_t|y) - \nabla_{x_t} \log p(x_t)$$

$$= (1+w)\ \nabla_{x_t} \log p(x_t|y) + w\ \nabla_{x_t} \log p(x_t)$$

- **Intuition**: instead of training a separate classifier, obtain an implicit classifier by jointly training conditional and unconditional diffusion at the same time
- Amounts to dropping the condition (e.g., zeroing it out) some fraction of the time

Slide adopted from Denoising Diffusion Models Tutorial at CVPR 2023

Conditional Diffusion Models: Classifier-free Guidance

Explicit Conditioning

Classifier Guidance

Classifier-free Guidance

Non-guidance

$\omega = 1$

$\omega = 3$

Slide adopted from Denoising Diffusion Models Tutorial at CVPR 2023

[Ho & Salimans, “Classifier-Free Diffusion Guidance”, 2021]