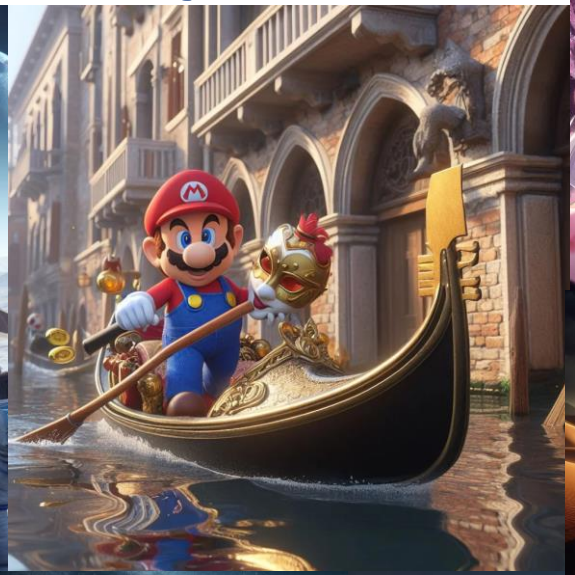
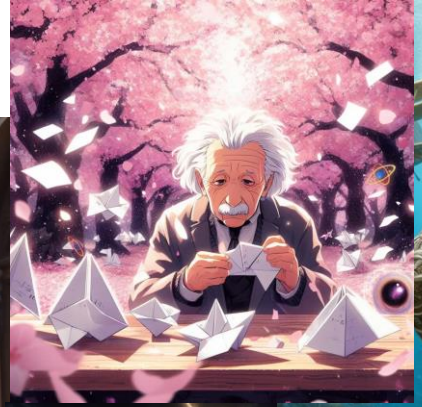
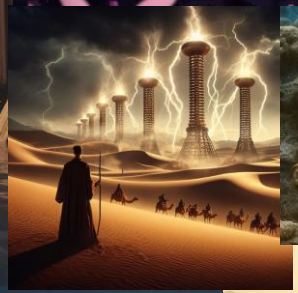
Images Generated by DALLE-3







JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

EN.601.482/682 Deep Learning

An Introduction to Diffusion Models

Mathias Unberath, PhD
Assistant Professor
Dept of Computer Science
Johns Hopkins University

Yiqing Shen
PhD Student
Dept of Computer Science
Johns Hopkins University

Agendas

- Recap: Generative Models
  - Density Modeling for Data Generation
  - Generative vs. Discriminative Models
  - VAE, GAE, Normalizing Flow, Energy Based Model
- Denoising Diffusion Probabilistic Model
  - Basic Concept & Definitions
  - Method Overview
  - Forward Process
  - Reverse Process
  - Training Objective
  - Denoising Network Architecture
  - Sampling Process
  - Comparisons with other Generative Models
- Conditional Diffusion Model
  - Fancy Applications
  - Formulation
  - Network
  - Latent Diffusion Model (*Stable Diffusion*)

Intro Diffusion Models

Recap: Generative Models

Density Modeling for Data Generation

- Assume that all data comes from a distribution $p_{data}(x)$, the distribution approximated by the model is denoted as $p_{theta}(x)$
- The goal of generative models is to approximate $p_{data}(x)$ with $p_{theta}(x)$
- We generate new data by sampling from the learned distribution $x \sim p_{theta}(x)$
- In practice, train models to maximize the expected log likelihood of $p_{theta}(x)$ (or minimizing negative log likelihood)/minimize divergence between $p_{theta}(x)$ and $p_{data}(x)$

$p_{data}(x)$: Unknown
$p_{theta}(x)$: Learnable and parameterize by theta

Training Objective from the Perspective of Density Modeling

VAE

$$
\begin{aligned}
L_{\mathrm{VAE}}(\theta, \phi) &= -\log p_\theta(\mathbf{x}) + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \\
&= -\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})) \\
\theta^*, \phi^* &= \arg\min_{\theta, \phi} L_{\mathrm{VAE}}
\end{aligned}
$$

$$
\boxed{-L_{\mathrm{VAE}} = \log p_\theta(\mathbf{x}) - D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \leq \log p_\theta(\mathbf{x})}
$$

minimizing negative log likelihood

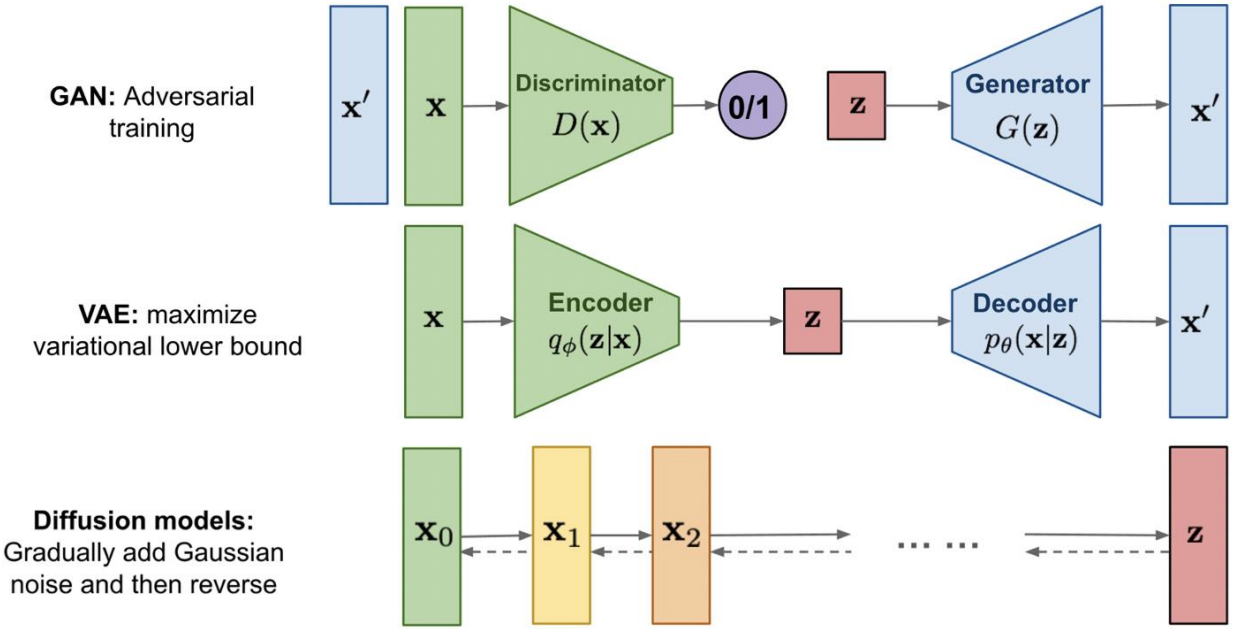
GAN

$$
\begin{aligned}
\min_G \max_D L(D, G) &= \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))] \\
&= \mathbb{E}_{x \sim p_r(x)}[\log D(x)] + \mathbb{E}_{x \sim p_g(x)}[\log(1 - D(x)]
\end{aligned}
$$

$$
\boxed{L(G, D^*) = 2D_{JS}(p_r \| p_g) - 2\log 2}
$$

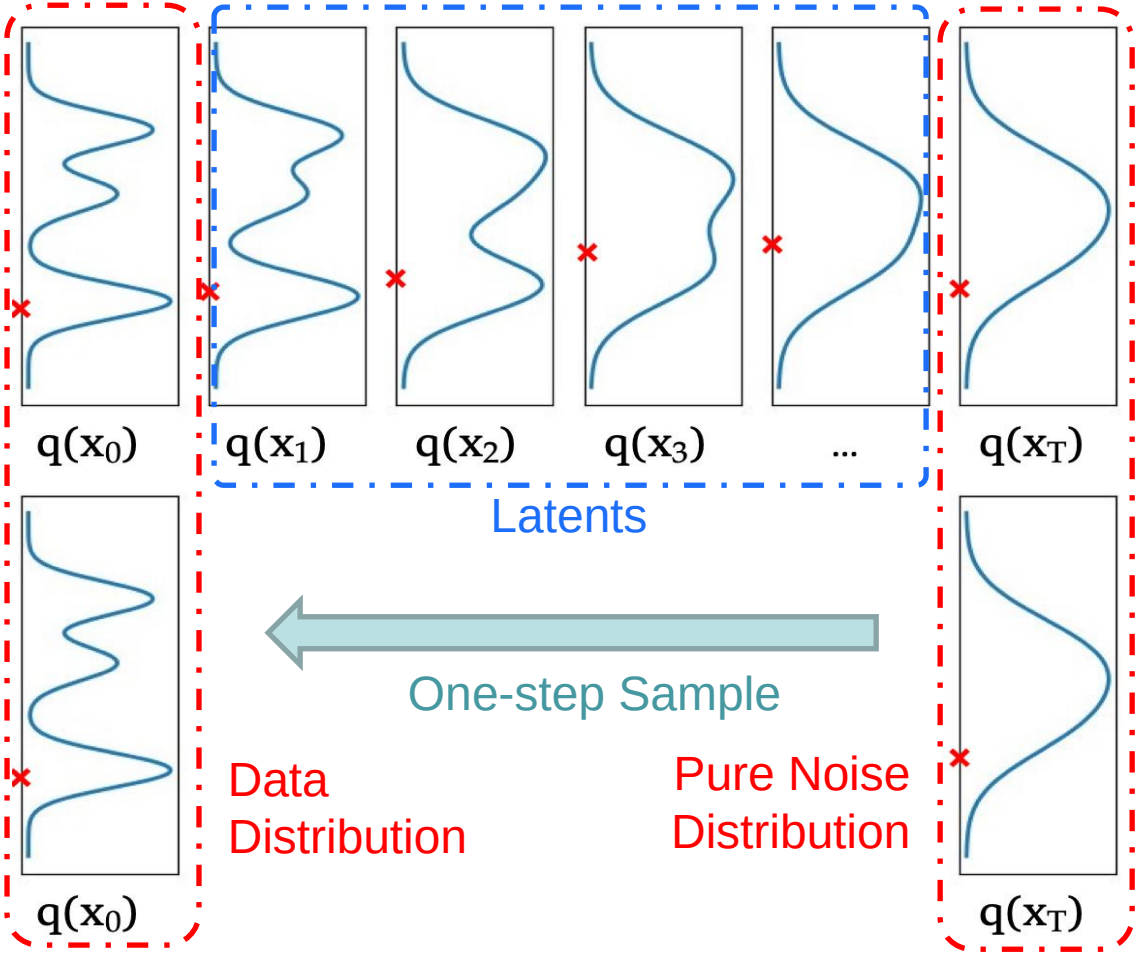
minimizing JS-divergence between the distribution of training set and the one model learned

Sampling Stage: Generate New Samples



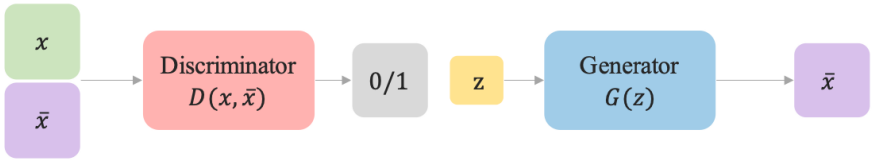
Why Diffusion Models?: Tractable probabilistic parameterization

Diffusion Models

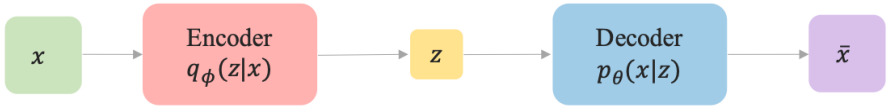


Other Models

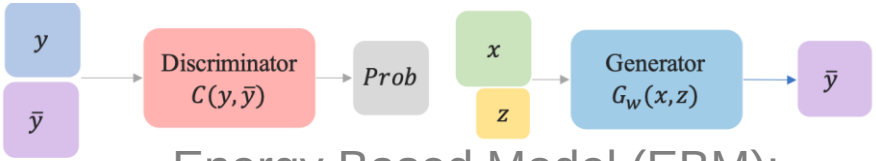
Why Diffusion Models?



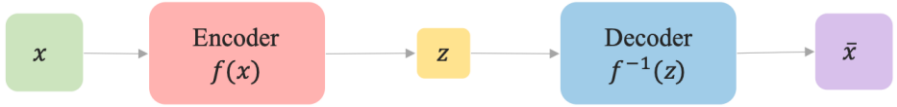
Generative Adversarial Network (GAN):
training additional discriminators



Variational Auto Encoder (VAE):
require aligning posterior distributions



Energy Based Model (EBM):
intractable partition functions



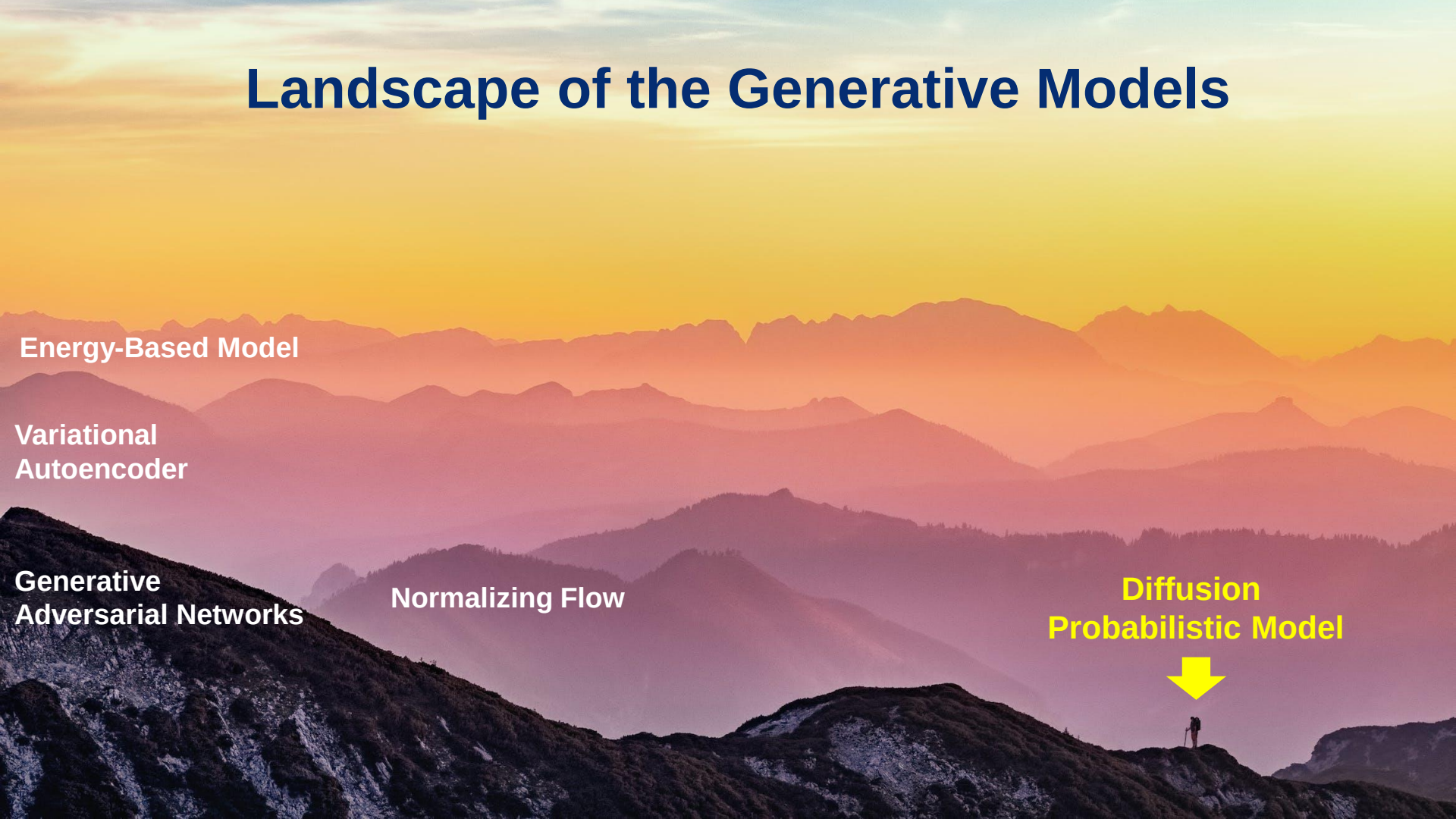
Normalizing Flow (NF):
imposing network constraints

Advantages of Diffusion Models:

- Tractable probabilistic parameterization for describing the generation process
- A stable training procedure with sufficient theoretical support
- A unified loss function design with high simplicity

Landscape of the Generative Models

Landscape of the Generative Models



Intro Diffusion Models

Denoising Diffusion Probabilistic Model (DDPM)

Denoising Diffusion Probabilistic Model Neurips 2020

Basic Concept of Diffusion

- **Diffusion** is the movement of anything (atoms, ions, molecules, energy) generally from a region of higher concentration to a region of lower concentration. (becomes more noisy)

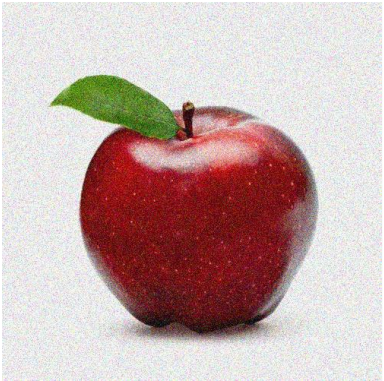
Basic Concept of Diffusion

- **Diffusion** is the movement of anything (atoms, ions, molecules, energy) generally from a region of higher concentration to a region of lower concentration.

What if we add a bunch of Gaussian noise to an image?

Basic Concept of Diffusion

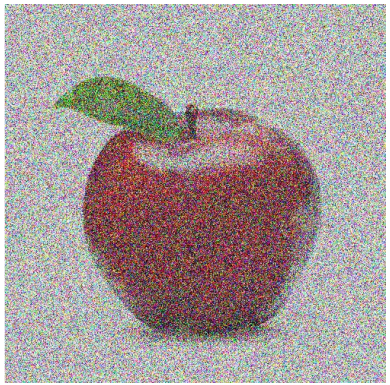
- **Diffusion** is the movement of anything (atoms, ions, molecules, energy) generally from a region of higher concentration to a region of lower concentration.



And again …

Basic Concept of Diffusion

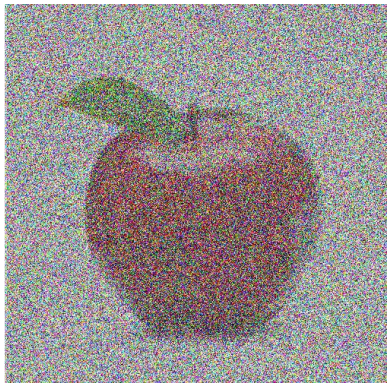
- **Diffusion** is the movement of anything (atoms, ions, molecules, energy) generally from a region of higher concentration to a region of lower concentration.



And again …

Basic Concept of Diffusion

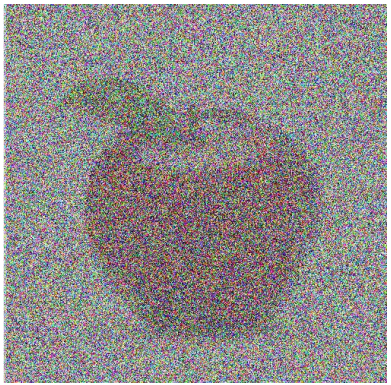
- **Diffusion** is the movement of anything (atoms, ions, molecules, energy) generally from a region of higher concentration to a region of lower concentration.



And again …

Basic Concept of Diffusion

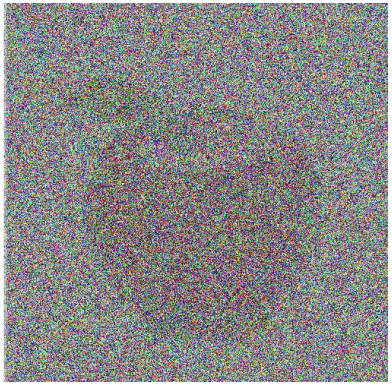
- **Diffusion** is the movement of anything (atoms, ions, molecules, energy) generally from a region of higher concentration to a region of lower concentration.



And again …

Basic Concept of Diffusion

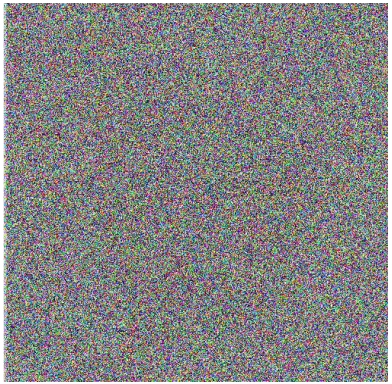
- **Diffusion** is the movement of anything (atoms, ions, molecules, energy) generally from a region of higher concentration to a region of lower concentration.



And again …

Basic Concept of Diffusion

- **Diffusion** is the movement of anything (atoms, ions, molecules, energy) generally from a region of higher concentration to a region of lower concentration.

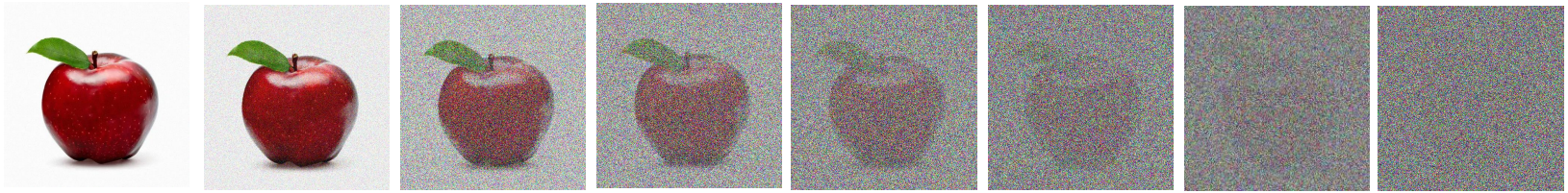


Until it resembles pure noise

Basic Concept of Diffusion

- **Diffusion** is the movement of anything (atoms, ions, molecules, energy) generally from a region of higher concentration to a region of lower concentration.

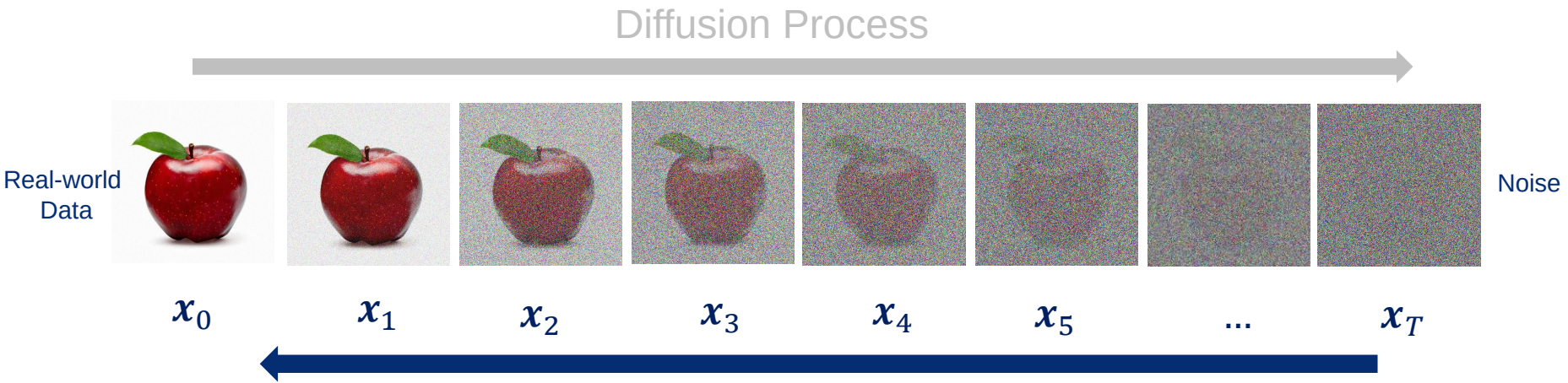
Diffusion Process



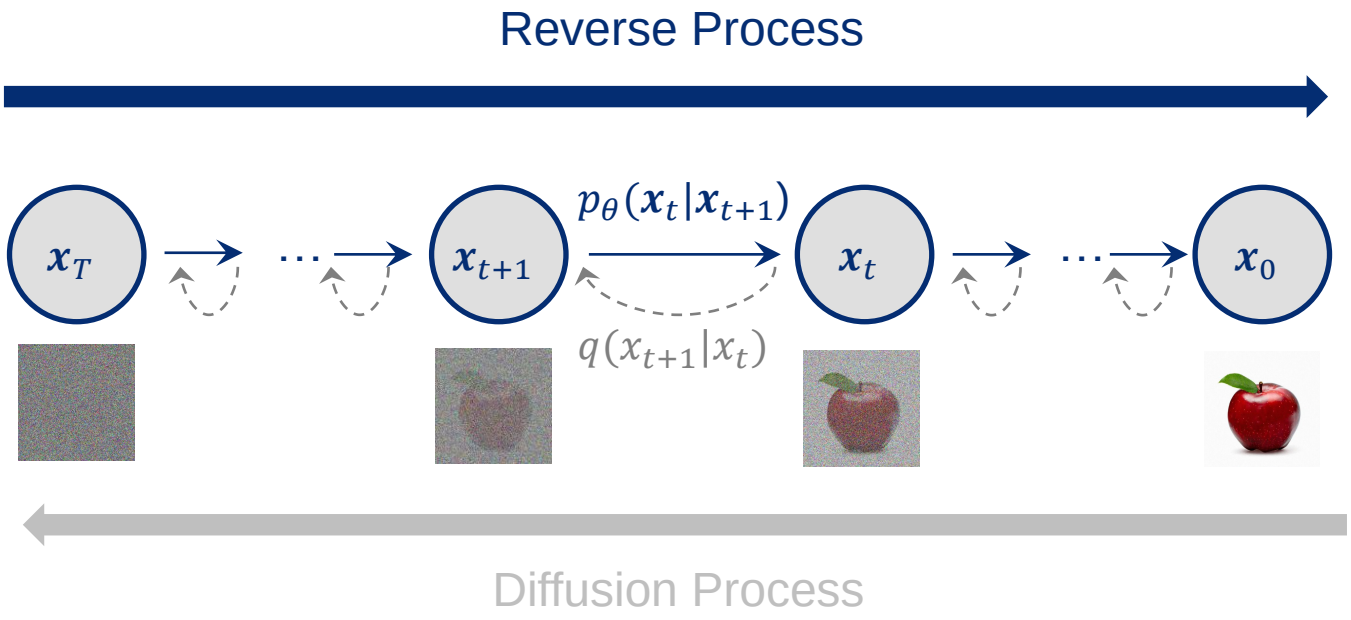
x_0 Real-world Data | x_1 | x_2 | x_3 | x_4 | x_5 | ... | x_T Noise

What is Diffusion Probabilistic Model?

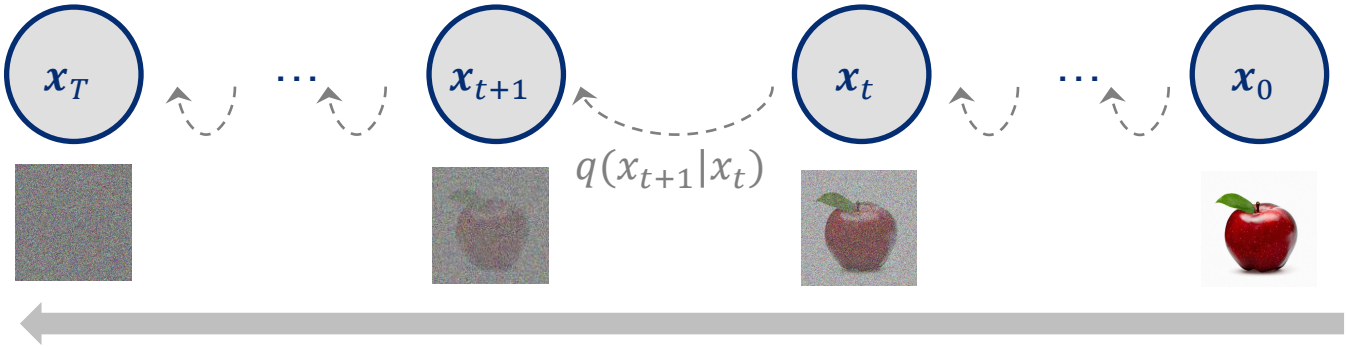
- Consists of two processes.
  - Diffusion/Forward process: gradually add noise to the input
  - Reverse process: learns to denoise -> generate new data



DDPM In the View of a Directed Graph

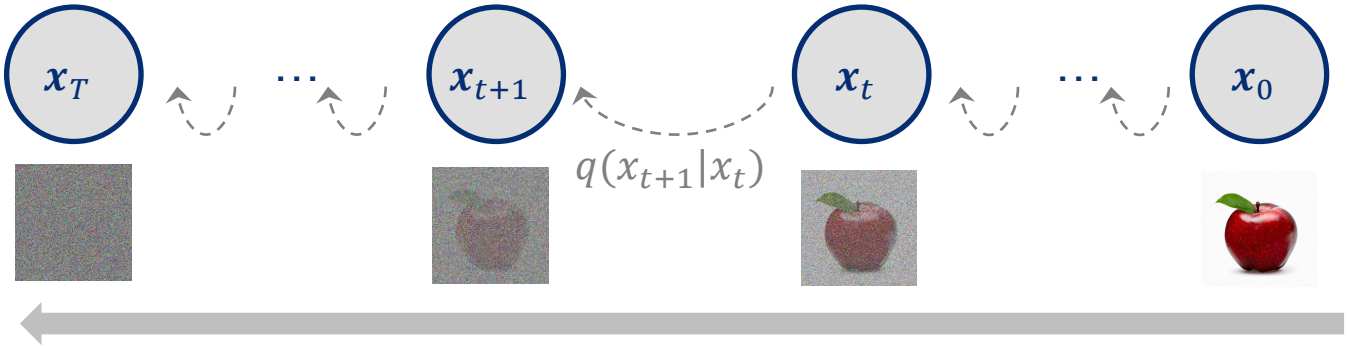


Diffusion/Forward Process (1/3)



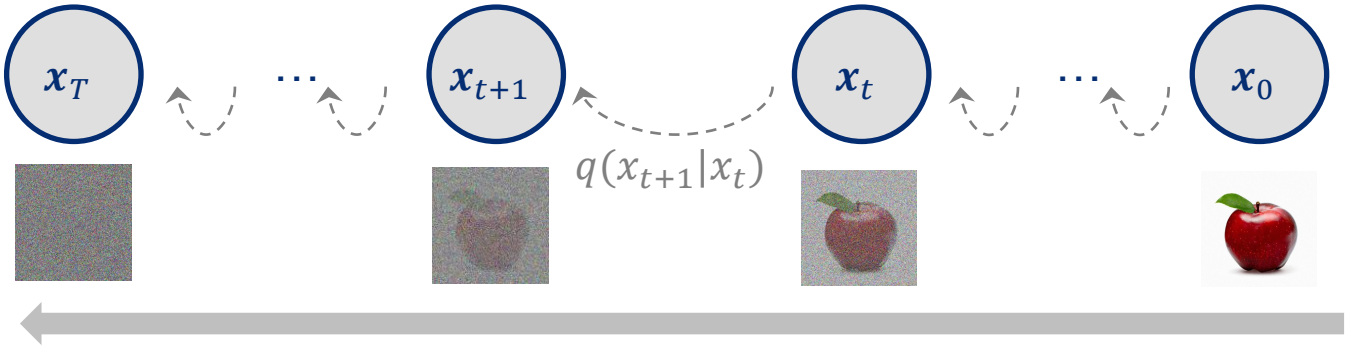
- Motivation: transforms the starting state ($\boldsymbol{x}_0$) into the tractable noise ($\boldsymbol{x}_i$)

Diffusion/Forward Process (1/3)



- Motivation: transforms the starting state ($\boldsymbol{x}_0$) into the tractable noise ($\boldsymbol{x}_i$)
- Formally, we call the joint distribution $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) = q(x_1, x_2, \ldots, xT|\boldsymbol{x}_0)$ as the **diffusion process**.

Diffusion/Forward Process (1/3)

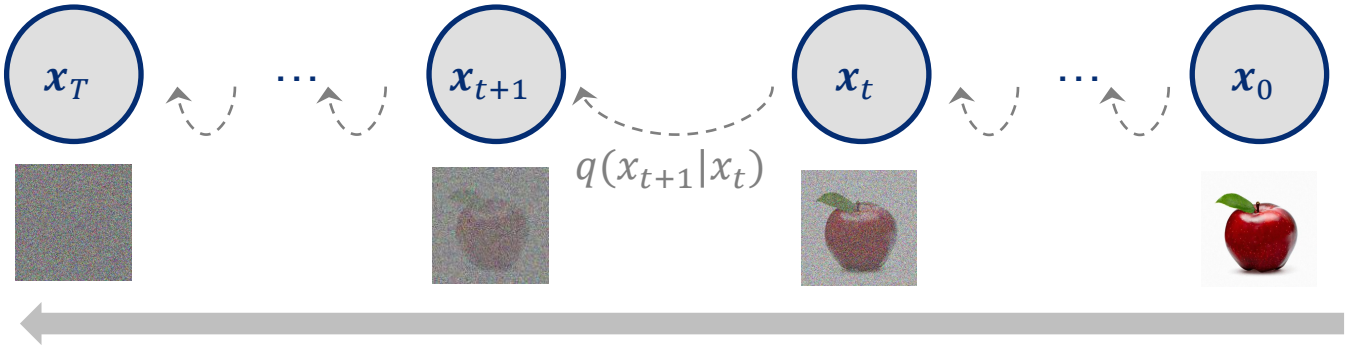


- Motivation: transforms the starting state ($\boldsymbol{x_0}$) into the tractable noise ($\boldsymbol{x_i}$)
- Formally, we call the joint distribution $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$ as the **diffusion process**.
- In DDPM, $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$ is defined as a Markov chain:

$$q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) = \prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{x}_{t-2}, \dots, \boldsymbol{x}_0)$$

Chain Rule (Probabilistic Properties)

Diffusion/Forward Process (1/3)

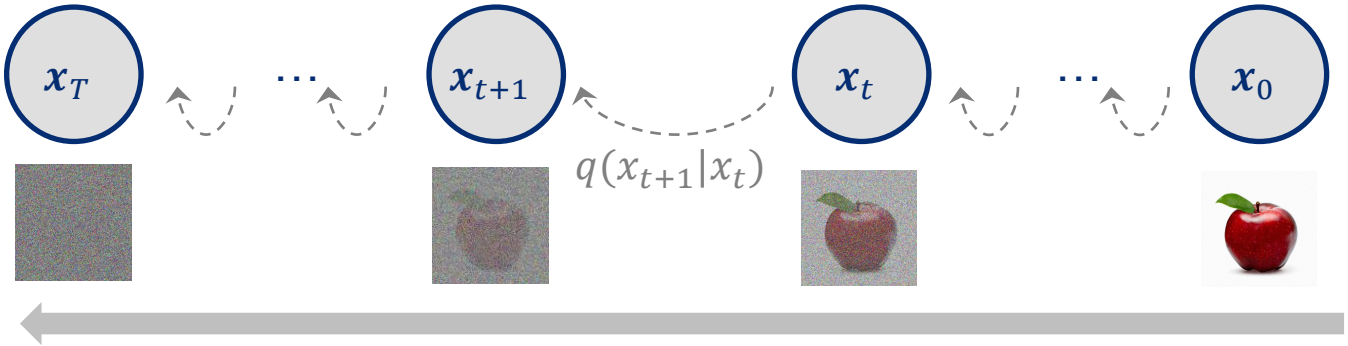


- Motivation: transforms the starting state ($\boldsymbol{x}_0$) into the tractable noise ($\boldsymbol{x}_i$)
- Formally, we call the joint distribution $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$ as the **diffusion process**.
- In DDPM, $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$ is defined as a Markov chain:

$$q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) = \prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$$

Markov Property -> Transaction kernel

Diffusion/Forward Process (1/3)



- Motivation: transforms the starting state ($\boldsymbol{x}_0$) into the tractable noise ($\boldsymbol{x}_i$)
- Formally, we call the joint distribution $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$ as the **diffusion process**.
- In DDPM, $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$ is defined as a Markov chain:

$$q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) = \prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$$

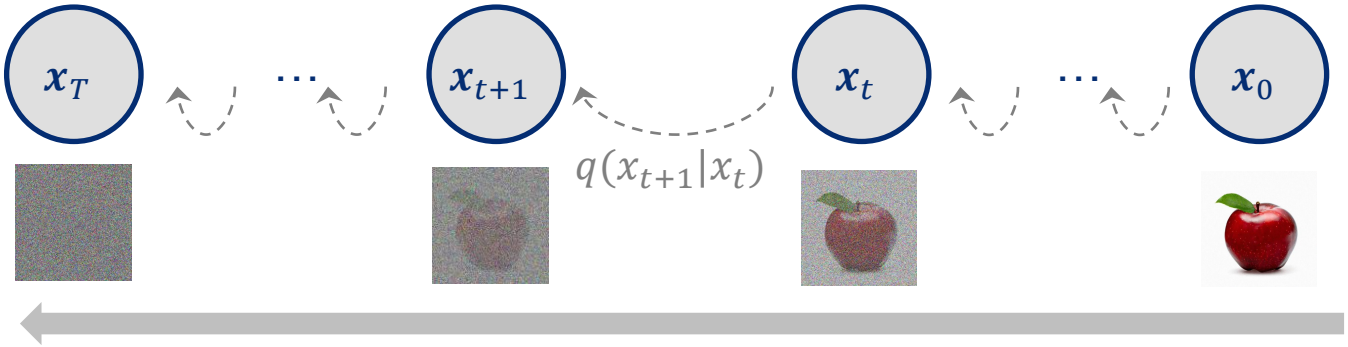
Transaction kernel

- The **transaction kernel** in DDPM is a Gaussian perturbation, i.e.

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}\left(\boldsymbol{x}_t \middle| \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t \boldsymbol{I}\right)$$

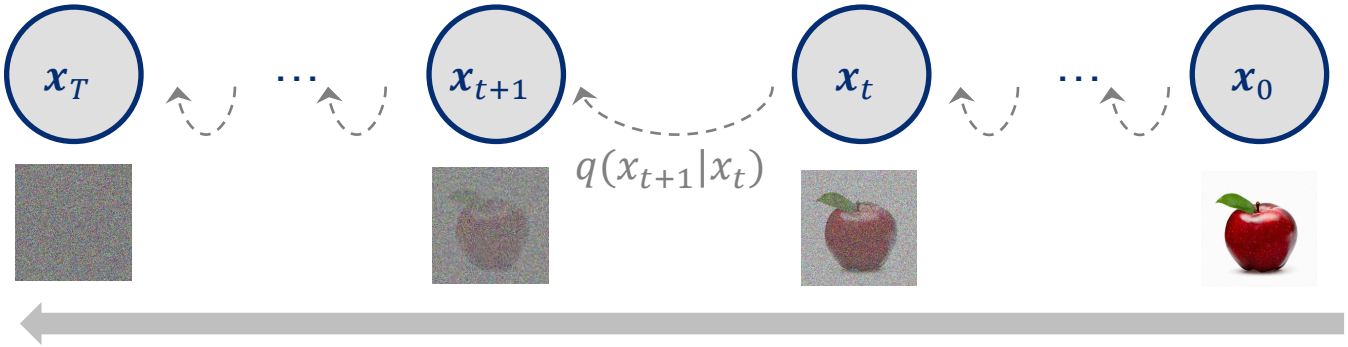
Parameters in the range of $(0,1)$.

Diffusion/Forward Process (2/3)



Q: Why Gaussian perturbation i.e., $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}\left(\boldsymbol{x}_t\middle|\sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t \boldsymbol{I}\right)$?

Diffusion/Forward Process (2/3)

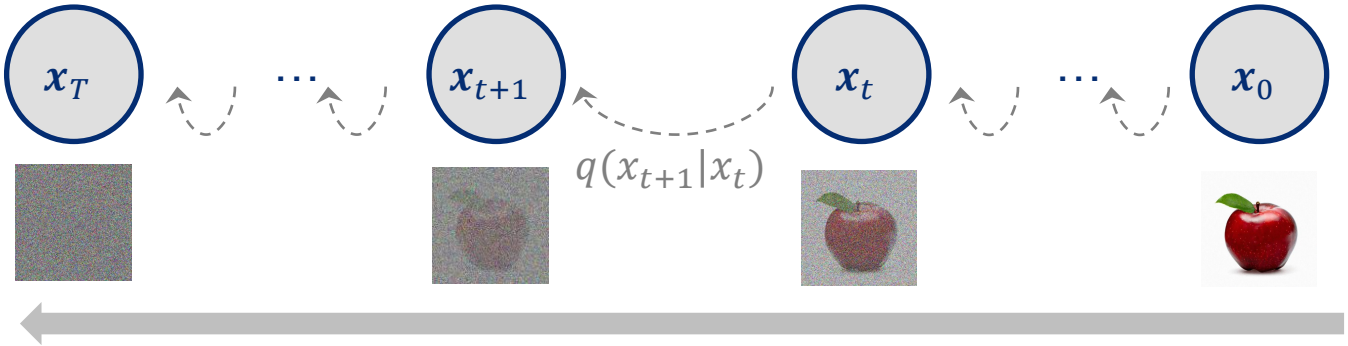


Q: Why Gaussian perturbation i.e., $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}\left(\boldsymbol{x}_t|\sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t \boldsymbol{I}\right)$?

A: Composition of Gaussians is still Gaussian

$$q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t|\sqrt{\overline{\alpha_t}}\boldsymbol{x}_{t-1}, (1-\overline{\alpha_t})I) \quad \text{where} \quad \overline{\alpha_t} = \prod_{s=1}^{t}(1-\beta_t)$$

Diffusion/Forward Process (2/3)



Q: Why Gaussian perturbation i.e., $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}\left(\boldsymbol{x}_t|\sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t \boldsymbol{I}\right)$?

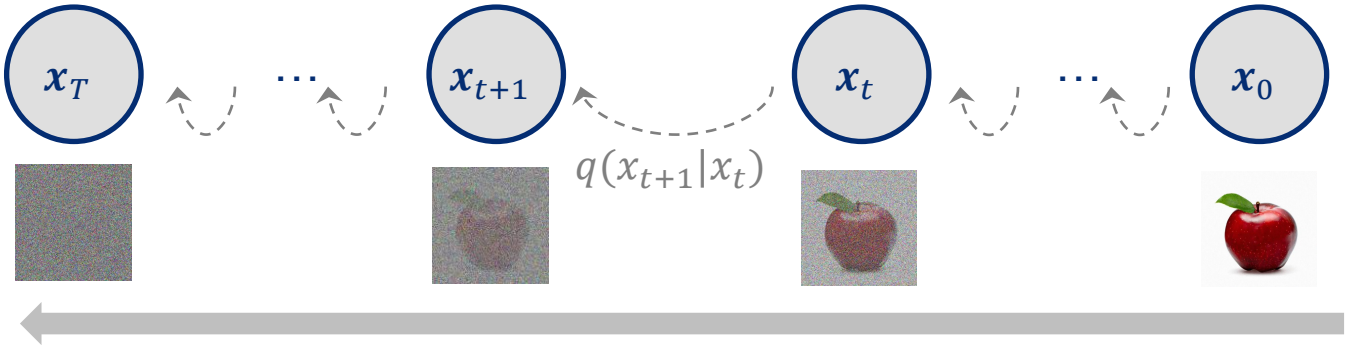
A: Composition of Gaussians is still Gaussian

$$q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t|\sqrt{\overline{\alpha_t}}\boldsymbol{x}_0, (1-\overline{\alpha_t})I) \quad \text{where} \quad \overline{\alpha_t} = \prod_{s=1}^{t}(1-\beta_t)$$

By choosing β_t properly (*e.g.,* all $\beta_t <$ Constant < 1), we have

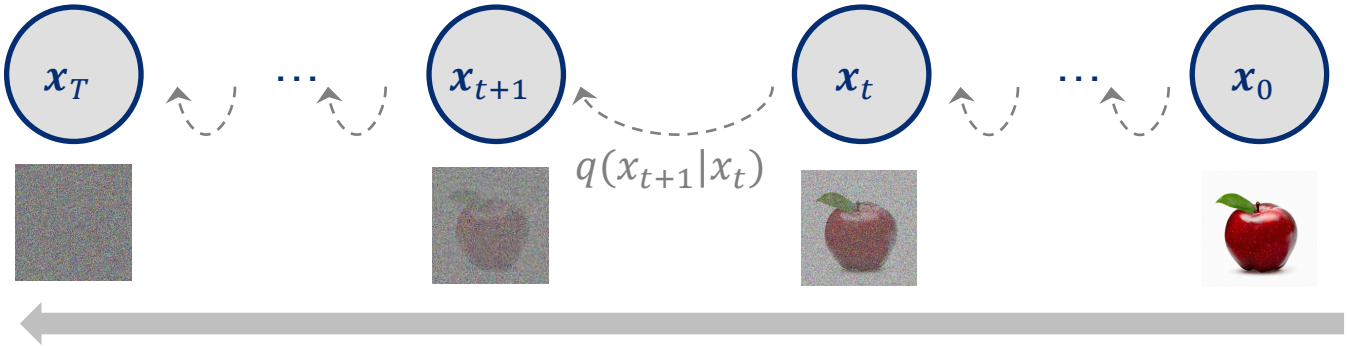
$$\lim_{n\to\infty} \overline{\alpha_t} = 0 \quad \text{and} \quad \lim_{t\to\infty} q(\boldsymbol{x}_t) = \lim_{t\to\infty} q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(0, \boldsymbol{I}).$$

Diffusion/Forward Process (3/3)



Q: How to sample from $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}\left(\boldsymbol{x}_t\middle|\sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t\boldsymbol{I}\right)$? (For the training purpose, we do computation with one data rather than on a distribution)

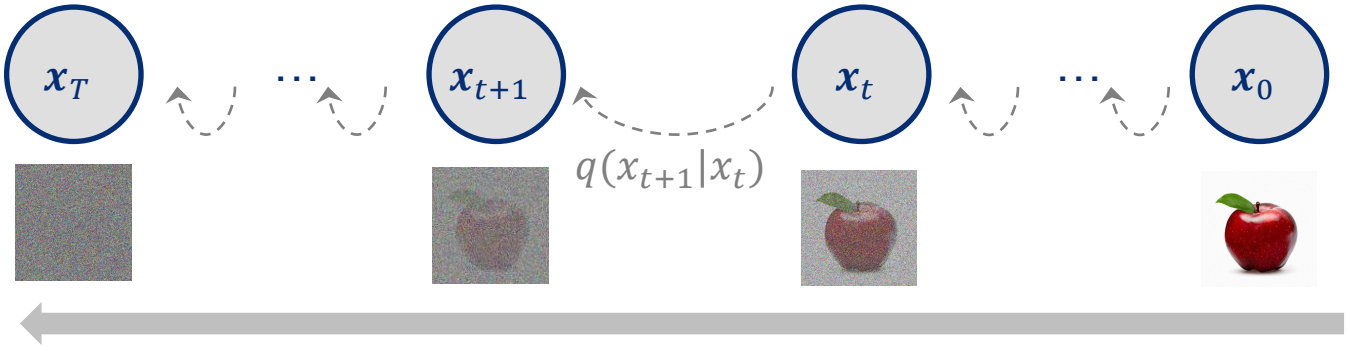
Diffusion/Forward Process (3/3)



Q: How to sample from $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}\left(\boldsymbol{x}_t\middle|\sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t\boldsymbol{I}\right)$?

A: $\boldsymbol{x}_t = \sqrt{1-\beta_t}\boldsymbol{x}_{t-1} + \beta_t \cdot \epsilon_{t-1}$ where $\epsilon_{t-1} \sim \mathcal{N}(0, \boldsymbol{I})$.

Diffusion/Forward Process (3/3)

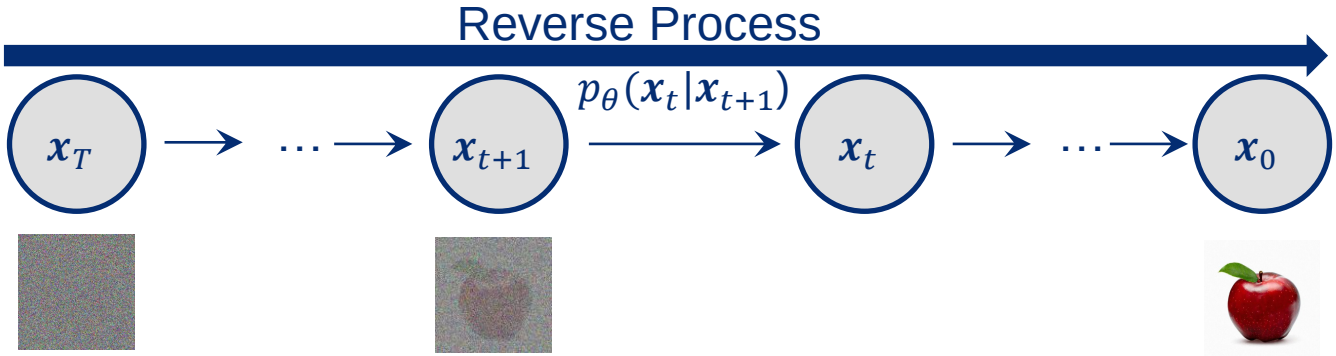


Q: How to sample from $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}\left(\boldsymbol{x}_t|\sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t \boldsymbol{I}\right)$?

A: $\boldsymbol{x}_t = \sqrt{1-\beta_t}\boldsymbol{x}_{t-1} + \beta_t \cdot \epsilon_{t-1}$ where $\epsilon_{t-1} \sim \mathcal{N}(0, \boldsymbol{I})$.

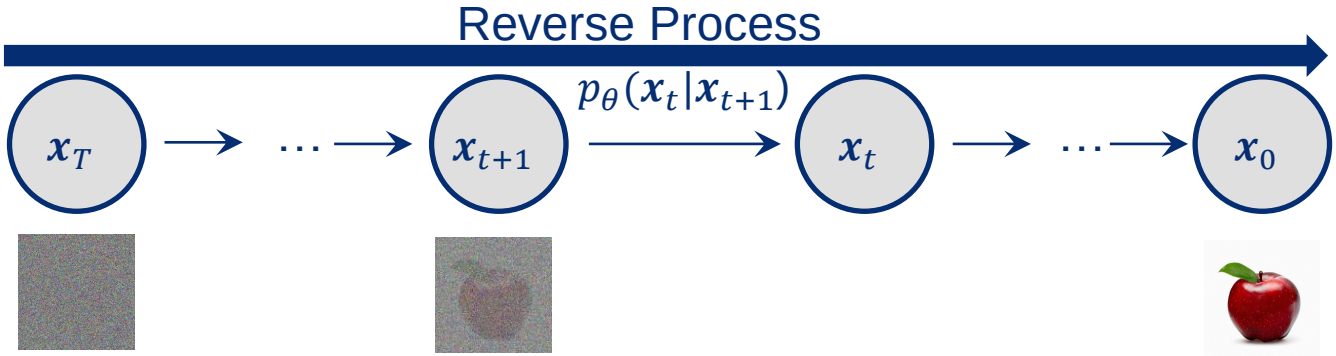
Similarly, $q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t|\sqrt{\overline{\alpha_t}}\boldsymbol{x}_0, (1-\overline{\alpha_t})I)$ yields $\boldsymbol{x}_t = \sqrt{\overline{\alpha_t}}\boldsymbol{x}_0 + (1-\overline{\alpha_t}) \cdot \epsilon$, where $\epsilon \sim \mathcal{N}(0, \boldsymbol{I})$.

Reverse Process (1/2)



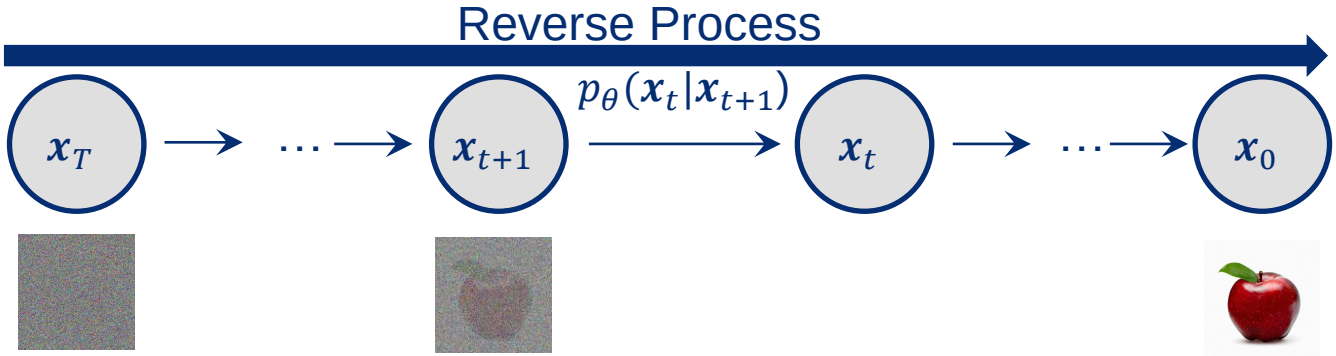
- Motivation: Reverse $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ to reconstruct image ($\boldsymbol{x}_0$) from noise ($\boldsymbol{x}_T$).

Reverse Process (1/2)



- Motivation: Reverse $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ to reconstruct image ($\boldsymbol{x}_0$) from noise ($\boldsymbol{x}_T$).
- Formally, we term the joint distribution $p_\theta(x_{0:T})$as the **reverse process**.

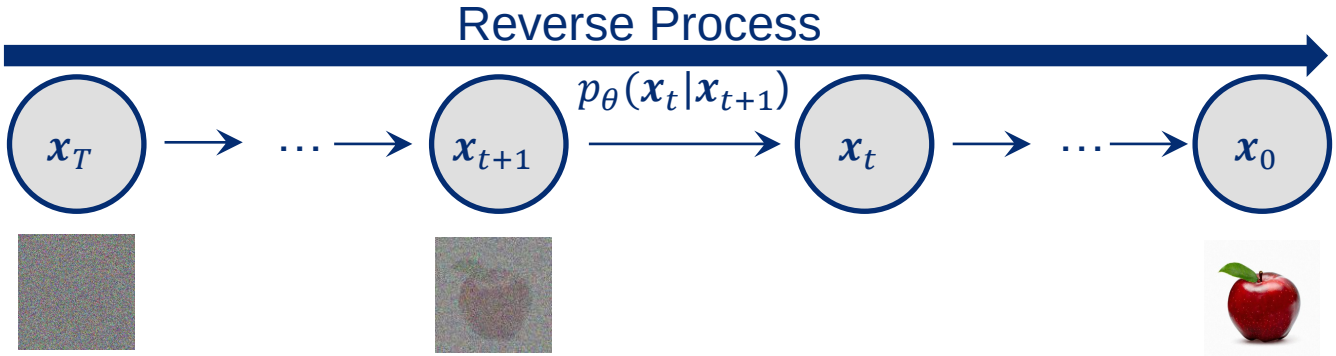
Reverse Process (1/2)



- Motivation: Reverse $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ to reconstruct image ($\boldsymbol{x}_0$) from noise ($\boldsymbol{x}_T$).
- Formally, we term the joint distribution $p_\theta(x_{0:T})$as the **reverse process**.
- In DDPM, $p_\theta(x_{0:T})$ is also a Markov chain, i.e.

$$p_\theta(x_{0:T}) = p_\theta(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)\,.$$

Reverse Process (1/2)



- Motivation: Reverse $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ to reconstruct image ($\boldsymbol{x}_0$) from noise ($\boldsymbol{x}_T$).
- Formally, we term the joint distribution $p_\theta(x_{0:T})$as the **reverse process**.
- In DDPM, $p_\theta(x_{0:T})$ is also a Markov chain, i.e.

$$p_\theta(x_{0:T}) = p_\theta(\boldsymbol{x}_T)\prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)\,.$$

$\boldsymbol{\mu}_\theta$ is learnable mapping (*e.g.,* U-Net).
$\boldsymbol{\Sigma}_\theta$ can be learnable, but simply set to $\sigma_t \boldsymbol{I}$

- Each factor $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ learns to approximate unknown q$(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ by:

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}|\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t, t))$$

Reverse Process (2/2)

- However, $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is not identifiable/intractable, using the Bayesian Rule:

$$f(\theta \mid x) = \frac{f(\theta, x)}{f(x)} = \frac{f(\theta)\, f(x \mid \theta)}{f(x)} \quad \Longrightarrow \quad q(x_{t-1} \mid x_t) = q(x_t \mid x_{t-1}) \frac{q(x_{t-1})}{q(x_t)}$$

Reverse Process (2/2)

- However, $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is not identifiable/intractable, using the Bayesian Rule:

• However, $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is not identifiable/intractable, using the Bayesian Rule:

$$q(x_{t-1} \mid x_t) = q(x_t \mid x_{t-1}) \frac{q(x_{t-1})}{q(x_t)}$$

$$q(x_t) = \int q(x_t \mid x_{t-1}) q(x_{t-1}) \mathrm{d}x$$

Need to integrate over the whole data distribution to derive

Q: Can we somehow find approximation that is tractable?

Reverse Process (2/2)

- However, $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is not tractable
- $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ is tractable, using the Bayesian Rule:

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

Reverse Process (2/2)

- However, $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is not tractable
- $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ is tractable, using the Bayesian Rule:

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

Markov assumption

$$q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t|\sqrt{\overline{\alpha_t}}\boldsymbol{x}_0, (1-\overline{\alpha_t})I)$$

Reverse Process (2/2)

- However, $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is not tractable
- $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ is tractable, using the Bayesian Rule:

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

Markov assumption

$$q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t|\sqrt{\overline{\alpha}_t}\boldsymbol{x}_0, (1-\overline{\alpha}_t)I)$$

$$\propto \exp\left(-\frac{1}{2}\left(\frac{(x_t - \sqrt{\alpha_t}x_{t-1})^2}{\beta_t} + \frac{(x_{t-1} - \sqrt{\overline{\alpha}_{t-1}}x_0)^2}{1-\overline{a}_{t-1}} - \frac{(x_t - \sqrt{\overline{\alpha}_t}x_0)^2}{1-\overline{a}_t}\right)\right)$$

$$= \exp\left(-\frac{1}{2}\left((\frac{\alpha_t}{\beta_t} + \frac{1}{1-\overline{\alpha}_{t-1}})x_{t-1}^2 - (\frac{2\sqrt{\alpha_t}}{\beta_t}x_t + \frac{2\sqrt{\overline{a}_{t-1}}}{1-\overline{\alpha}_{t-1}}x_0)x_{t-1} + C(x_t, x_0)\right)\right)$$

Reverse Process (2/2)

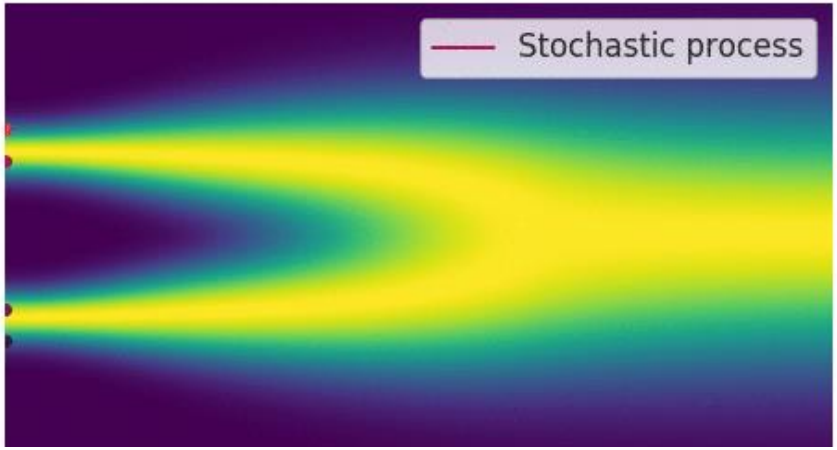
- However, $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is not tractable
- $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ is tractable, using the Bayesian Rule:

$$
\begin{aligned}
q(x_{t-1}|x_t, x_0) &= q(x_t|x_{t-1}, x_0)\frac{q(x_{t-1}|x_0)}{q(x_t|x_0)} \\
&\propto \exp\left(-\frac{1}{2}\left(\frac{(x_t - \sqrt{\alpha_t}x_{t-1})^2}{\beta_t} + \frac{(x_{t-1} - \sqrt{\bar{\alpha}_{t-1}}x_0)^2}{1-\bar{a}_{t-1}} - \frac{(x_t - \sqrt{\bar{\alpha}_t}x_0)^2}{1-\bar{a}_t}\right)\right) \\
&= \exp\left(-\frac{1}{2}\left(\left(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}\right)x_{t-1}^2 - \left(\frac{2\sqrt{\alpha_t}}{\beta_t}x_t + \frac{2\sqrt{\bar{a}_{t-1}}}{1-\bar{\alpha}_{t-1}}x_0\right)x_{t-1} + C(x_t, x_0)\right)\right)
\end{aligned}
$$

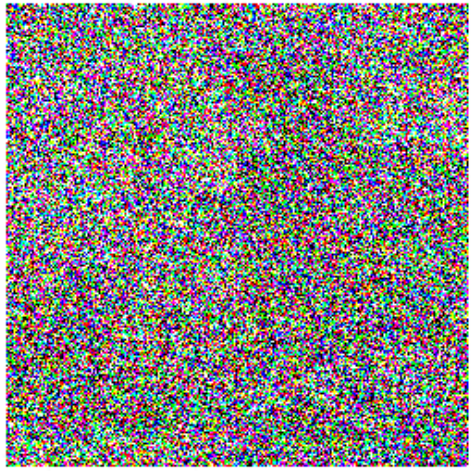
- In brief, we have $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \mathcal{N}\left(\boldsymbol{x}_{t-1}\middle|\widetilde{\boldsymbol{\mu}}_t(\boldsymbol{x}_t, \boldsymbol{x}_0), \widetilde{\beta}_t \boldsymbol{I}\right)$ with

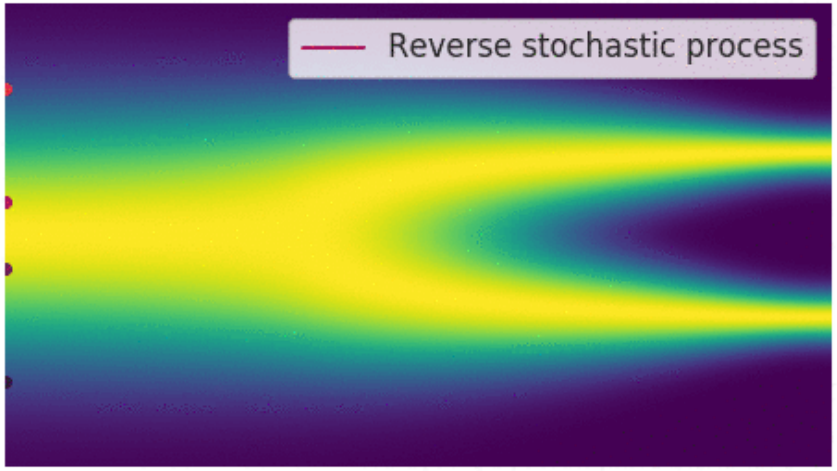
$$
\widetilde{\boldsymbol{\mu}}_t(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\boldsymbol{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{x}_t \text{ and } \widetilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t
$$

Visualization of the Forward and Reverse Processes



Forward process: converting the image distribution to pure noise





Reverse process: sampling from the image distribution, starting with pure noise

Training Objective (1/2)

- To approximate $\mathrm{q}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ with $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$, we define the loss to be the KL-divergence between them *i.e.*, $D_{KL}(\mathrm{q}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)\big|\big|p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))$, which can be simplified to:

$$\mathbb{E}_{\boldsymbol{x}_t \sim q}\left[\frac{1}{2\sigma_t^2}\|\widetilde{\boldsymbol{\mu}}_{\boldsymbol{t}}(\boldsymbol{x}_t, \boldsymbol{\varepsilon}) - \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)\|^2\right]$$

Training Objective (1/2)

- To approximate $\mathrm{q}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ with $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$, we define the loss to be the KL-divergence between them *i.e.*, $D_{KL}(\mathrm{q}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)\big|\big|p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))$, which can be simplified to:

$$\mathbb{E}_{\boldsymbol{x}_t \sim q}\left[\frac{1}{2\sigma_t^2}\|\widetilde{\boldsymbol{\mu}}_{\boldsymbol{t}}(\boldsymbol{x}_t, \boldsymbol{\varepsilon}) - \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)\|^2\right]$$

- It means that $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)$ tries to predict $\widetilde{\boldsymbol{\mu}}_{\boldsymbol{t}}(\boldsymbol{x}_t, \boldsymbol{\varepsilon}) = \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\boldsymbol{\varepsilon})$.

Training Objective (1/2)

- To approximate $\mathrm{q}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ with $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$, we define the loss to be the KL-divergence between them *i.e.*, $D_{KL}(\mathrm{q}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)\big|\big|p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))$, which can be simplified to:

$$\mathbb{E}_{\boldsymbol{x}_{t\sim q}}\left[\frac{1}{2\sigma_t^2}\|\widetilde{\boldsymbol{\mu}_t}(\boldsymbol{x}_t, \boldsymbol{\varepsilon}) - \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)\|^2\right]$$

- It means that $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)$ tries to predict $\widetilde{\boldsymbol{\mu}_t}(\boldsymbol{x}_t, \boldsymbol{\varepsilon}) = \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\boldsymbol{\varepsilon})$.
- We come to the the parametrization $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t))$ *where* $\boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t)$ intends to predict $\boldsymbol{\varepsilon}$ from $\boldsymbol{x}_t$.

Training Objective (1/2)

- To approximate $\mathrm{q}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ with $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$, we define the loss to be the KL-divergence between them *i.e.*, $D_{KL}(\mathrm{q}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)\big|\big|p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))$, which can be simplified to:

$$\mathbb{E}_{\boldsymbol{x}_t \sim q}\left[\frac{1}{2\sigma_t^2}\|\widetilde{\boldsymbol{\mu}}_{\boldsymbol{t}}(\boldsymbol{x}_t, \boldsymbol{\varepsilon}) - \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)\|^2\right]$$

- It means that $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)$ tries to predict $\widetilde{\boldsymbol{\mu}}_{\boldsymbol{t}}(\boldsymbol{x}_t, \boldsymbol{\varepsilon}) = \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\boldsymbol{\varepsilon})$.
- We come to the the parametrization $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t))$ *where* $\boldsymbol{\varepsilon}_\theta(\boldsymbol{x}_t, t)$ intends to predict $\boldsymbol{\varepsilon}$ from $\boldsymbol{x}_t$.
- It leads the loss function to be

$$L_t = \mathbb{E}_{\boldsymbol{x}_0, \boldsymbol{\varepsilon}}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\overline{\alpha_t})}\left\|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta\left(\sqrt{\overline{\alpha_t}}\boldsymbol{x}_0 + \sqrt{1-\overline{\alpha_t}}\boldsymbol{\varepsilon}, t\right)\right\|^2\right]$$

Known weights

$\boldsymbol{x}_t$

Training Objective (2/2)

- To simplify the formulation, we can re-weight $L_t = \mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{\varepsilon}}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\overline{\alpha_t})}\left\|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta\left(\sqrt{\overline{\alpha_t}}\boldsymbol{x}_0 + \sqrt{1-\overline{\alpha_t}}\boldsymbol{\varepsilon}, t\right)\right\|^2\right]$, which is empirically found beneficial to the sample quality

$$\mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{\varepsilon}}\left\|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta\left(\sqrt{\overline{\alpha_t}}\boldsymbol{x}_0 + \sqrt{1-\overline{\alpha_t}}\boldsymbol{\varepsilon}, t\right)\right\|^2$$

- where t is uniform between 1 and T.

Training Objective (2/2)

- To simplify the formulation, we can re-weight $L_t = \mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{\varepsilon}}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\overline{\alpha_t})}\left\|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta\left(\sqrt{\overline{\alpha_t}}\boldsymbol{x}_0 + \sqrt{1-\overline{\alpha_t}}\boldsymbol{\varepsilon}, t\right)\right\|^2\right]$, which is empirically found beneficial to the sample quality

$$\mathbb{E}_{\boldsymbol{x}_0,\boldsymbol{\varepsilon}}\left\|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_\theta\left(\sqrt{\overline{\alpha_t}}\boldsymbol{x}_0 + \sqrt{1-\overline{\alpha_t}}\boldsymbol{\varepsilon}, t\right)\right\|^2$$

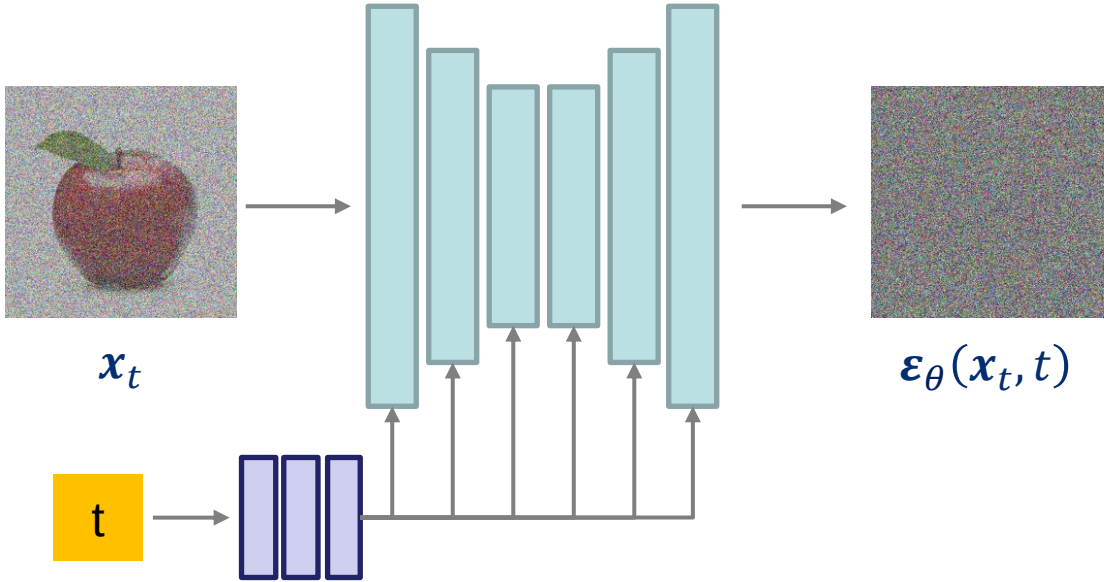
- where t is uniform between 1 and T.

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

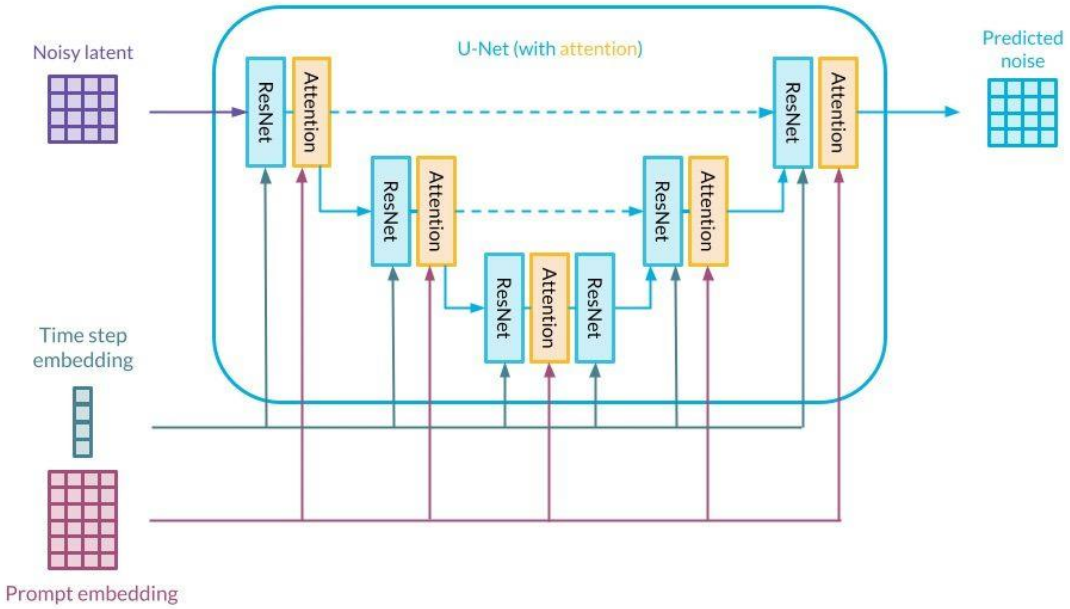
Denoising Network $\boldsymbol{\varepsilon_\theta(x_t, t)}$

U-Net (why?) with ResNet blocks + self-attention layers + time embedding



- Time Representation: Sinusoidal Positional Embeddings
- Time embeddings are fed to the residual blocks using either simple spatial addition or using adaptive group normalization layers

Take a Deeper Look into the Denosing Network



Take the U-Net model used for semantic segmentation and add:

- Positional embeddings
- Residual connections
- Attention modules
- Group normalization
- New activation functions like Swish, Mish, GeLU, etc

Sampling Process

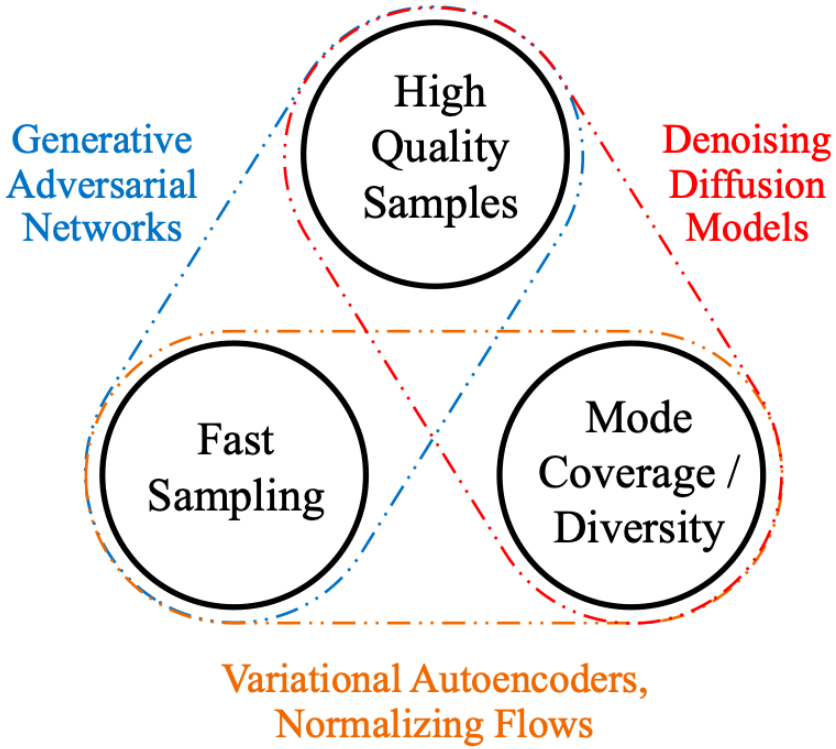
- **Goal** Generate a sample $\widehat{x_0}$ from the Gaussian x_T.
- **Limitation** Slow. Take 20 hours to sample 50k images of size 32 × 32 on a NVIDIA 2080Ti (*vs.* a GAN takes less than 1 min)

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 $\mathbf{x}_0$

Unconditional CIFAR10 progressive generation

Comparisons with Other Generative Models



Tackling the Generative Learning Trilemma with Denoising Diffusion GANs ICLR 2022

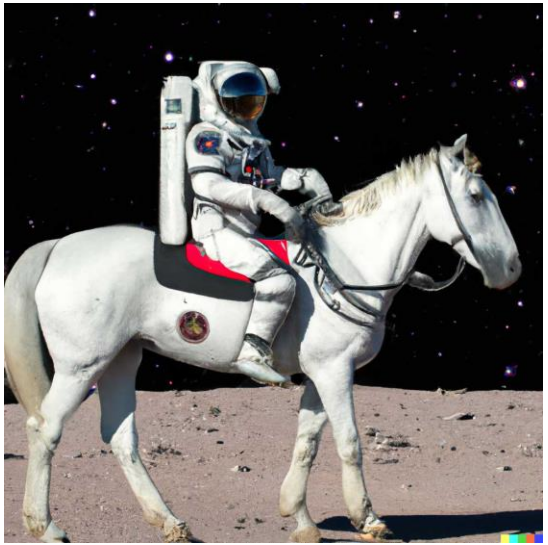
Intro Diffusion Models

Conditional Diffusion Model

Applications of Conditional Diffusion Models

Text-to-Image Generation (Dalle-2)

Teddy bears mixing sparkling chemicals as mad scientists



An astronaut riding a horse in a photorealistic style

A bowl of soup as a planet in the universe

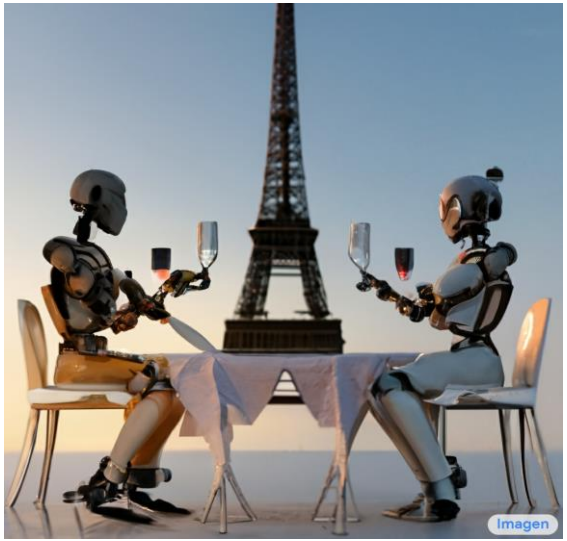
https://openai.com/product/dall-e-2

Applications of Conditional Diffusion Models

Text-to-Image Generation (Imagen)

A cute corgi lives in a house made of sushi

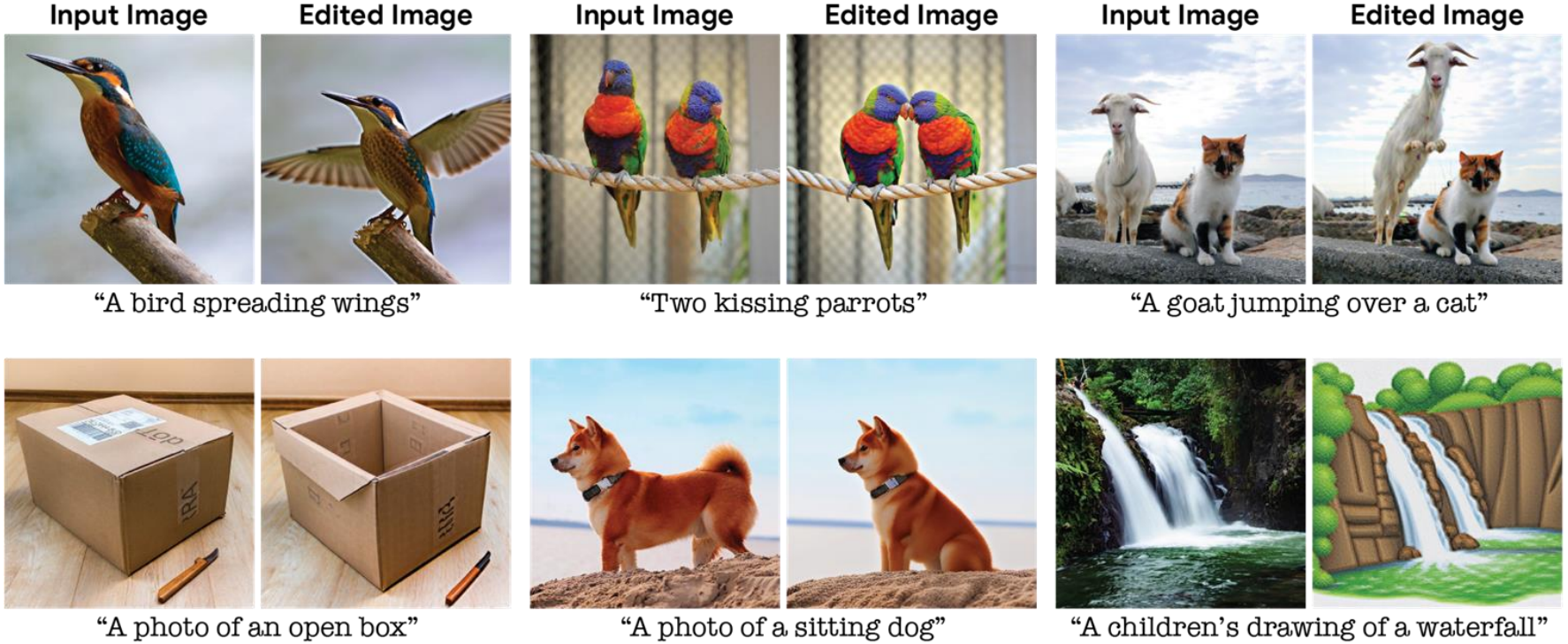
A majestic oil painting of a raccoon Queen wearing red French royal gown.



A robot couple fine-dining with the Eiffel Tower in the background

Applications of Conditional Diffusion Models

Image-to-Image Generation (Image Editing)



Applications of Conditional Diffusion Models

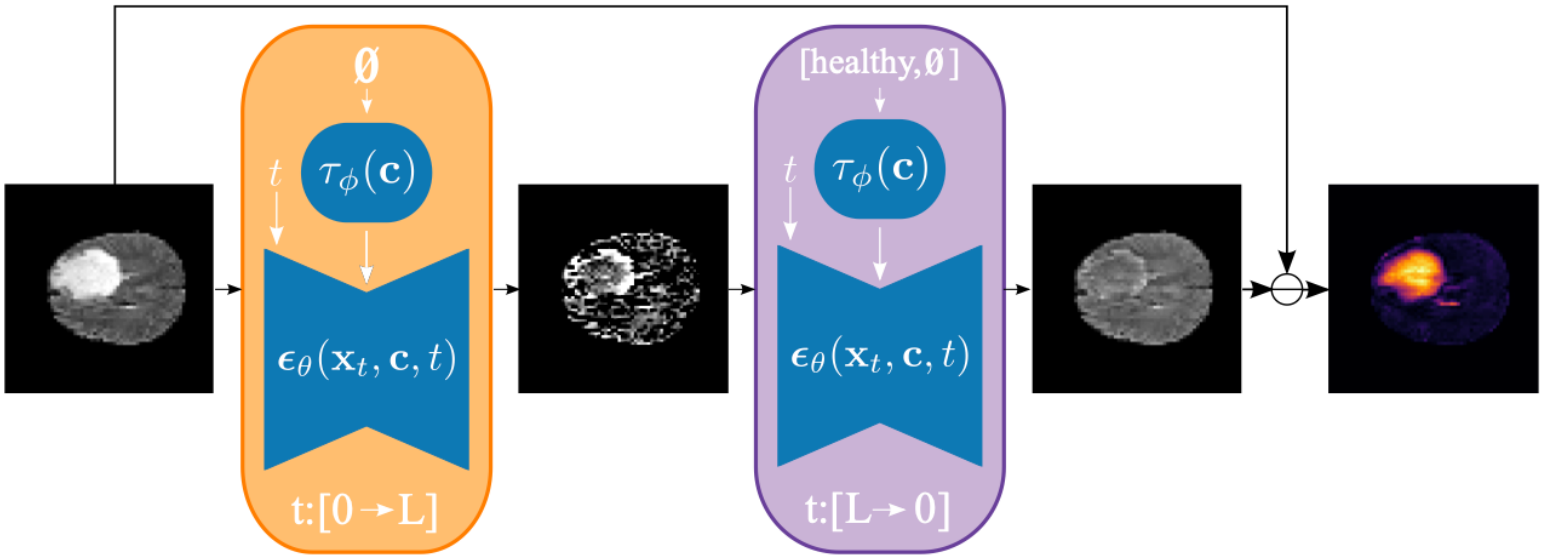
Image-to-Image Generation (Inpaiting)

Randomness

RePaint: Inpainting using Denoising Diffusion Probabilistic Models, CVPR 2022

Applications of Conditional Diffusion Models

Counterfactual Generation



What is Healthy? Generative Counterfactual Diffusion for Lesion Localization MICCAI/W 2022

Applications of Conditional Diffusion Models

Text to Video Generation

A confused grizzly bear in a calculus class

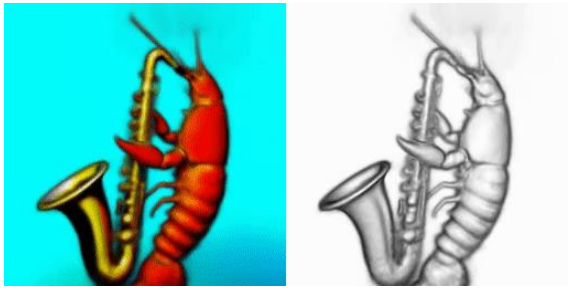
A golden retriever eating ice cream on a beautiful tropical beach at sunset, high resolution

A panda playing on a swing set

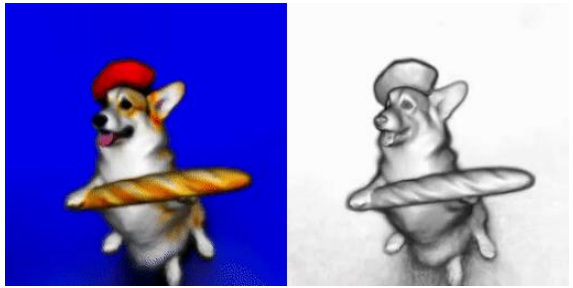
Applications of Conditional Diffusion Models

Text to 3D Generation (DreamFusion)

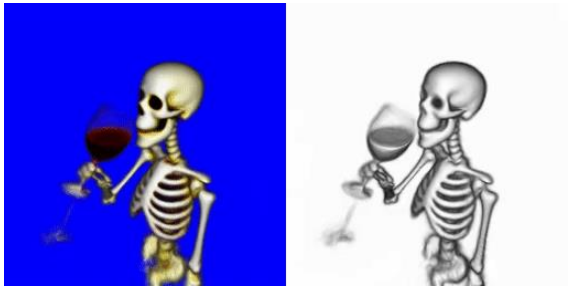
a fox holding a video game controller



a lobster playing the saxophone



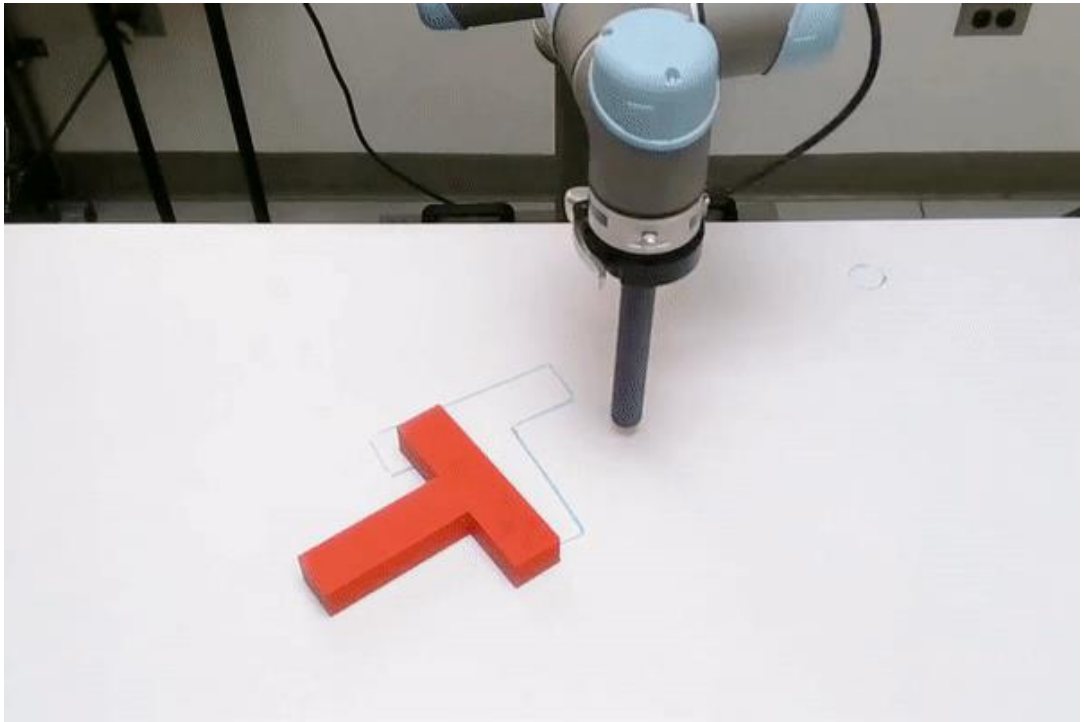
a corgi wearing a beret and holding a baguette, standing up on two hind legs



a human skeleton drinking a glass of red wine

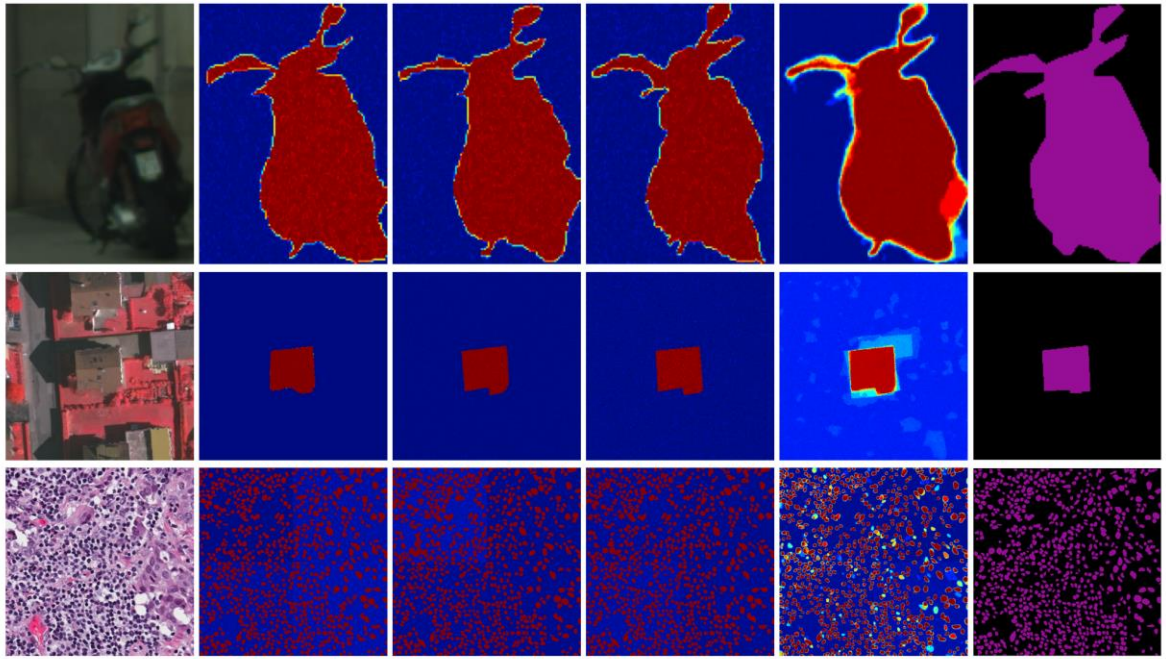
Applications of Conditional Diffusion Models

Diffusion Policy (Robotics)



Applications of Conditional Diffusion Models

Image Segmentation



Input | Multiple runs on the same input | Averaged | Ground Truth

Include Condition to Reverse Process

- Conditional Reverse Process:

$$p_\theta(x_{0:T}|\boldsymbol{c}) = p_\theta(\boldsymbol{x}_T)\prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{c})$$

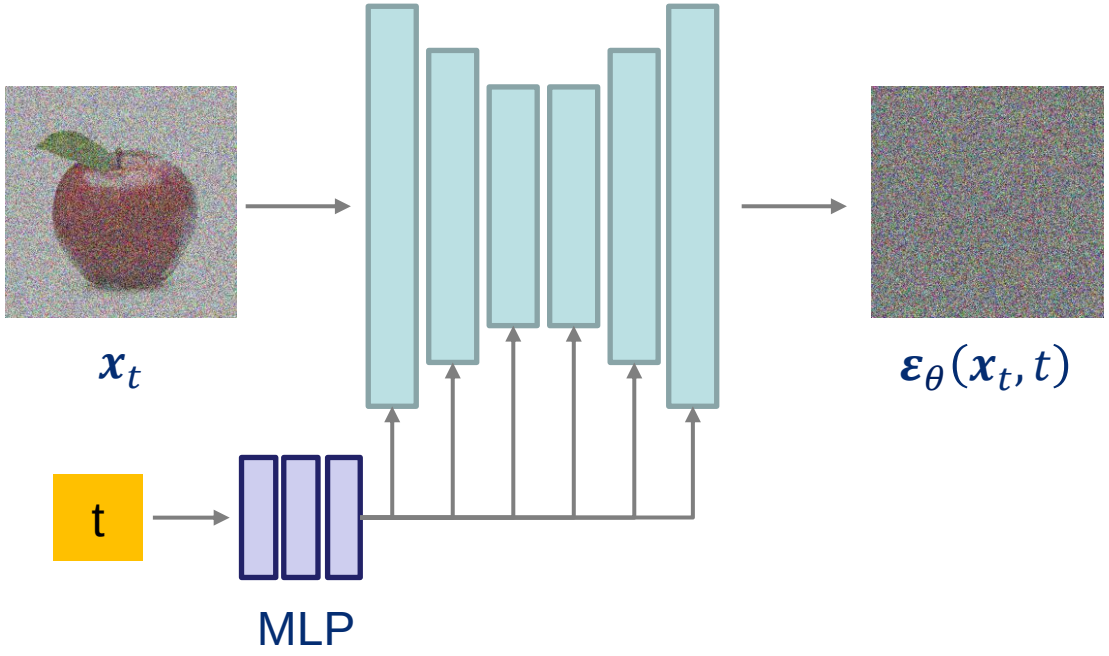
$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{c}) = \mathcal{N}(\boldsymbol{x}_{t-1}|\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t, \boldsymbol{c}), \boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t, t, \boldsymbol{c}))$$

Impose Conditions onto the Denoising UNet

- Scalar Conditioning (Representations): encode scalar as a vector embedding, simple spatial addition or adaptive group normalization layers.
- Image Conditioning: channel-wise concatenation of the conditional image.
- Text Conditioning: single vector embedding – spatial addition or adaptive group norm / a seq of vector embeddings - cross-attention.

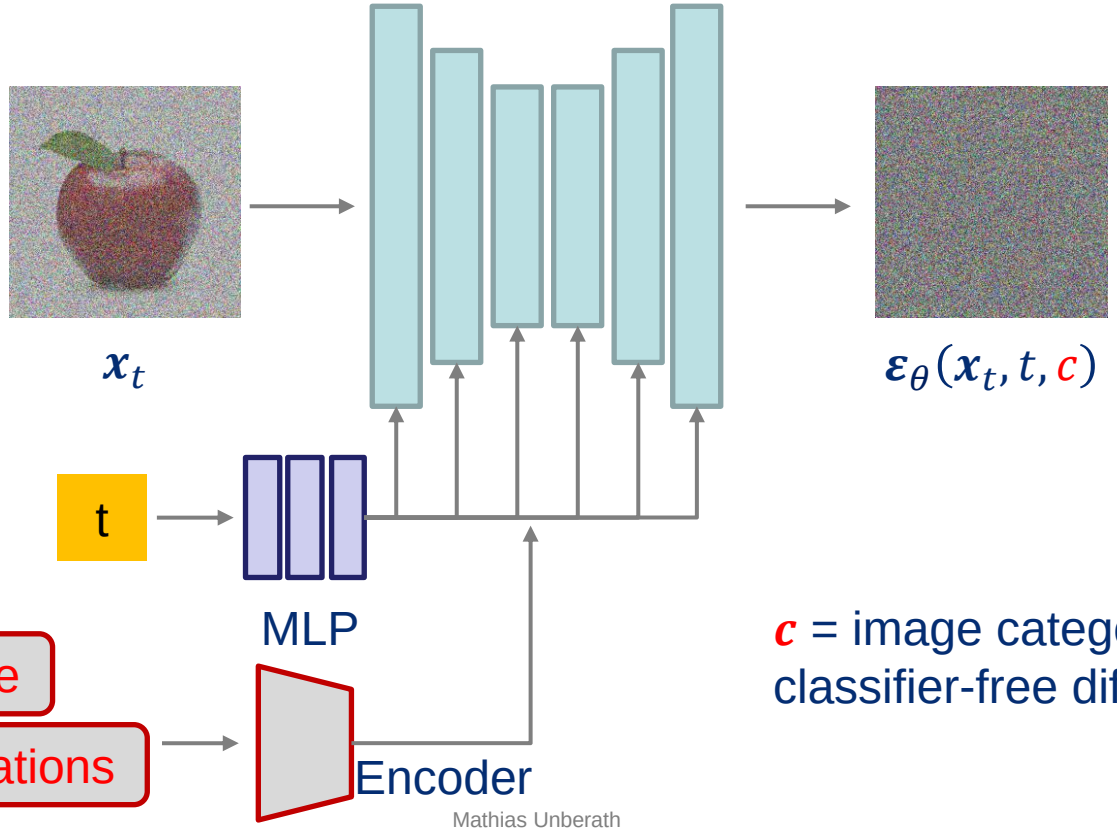
Conditional Denoising Network $\varepsilon_\theta(x_t, t, c)$

U-Net with ResNet blocks + self-attention layers + time embedding



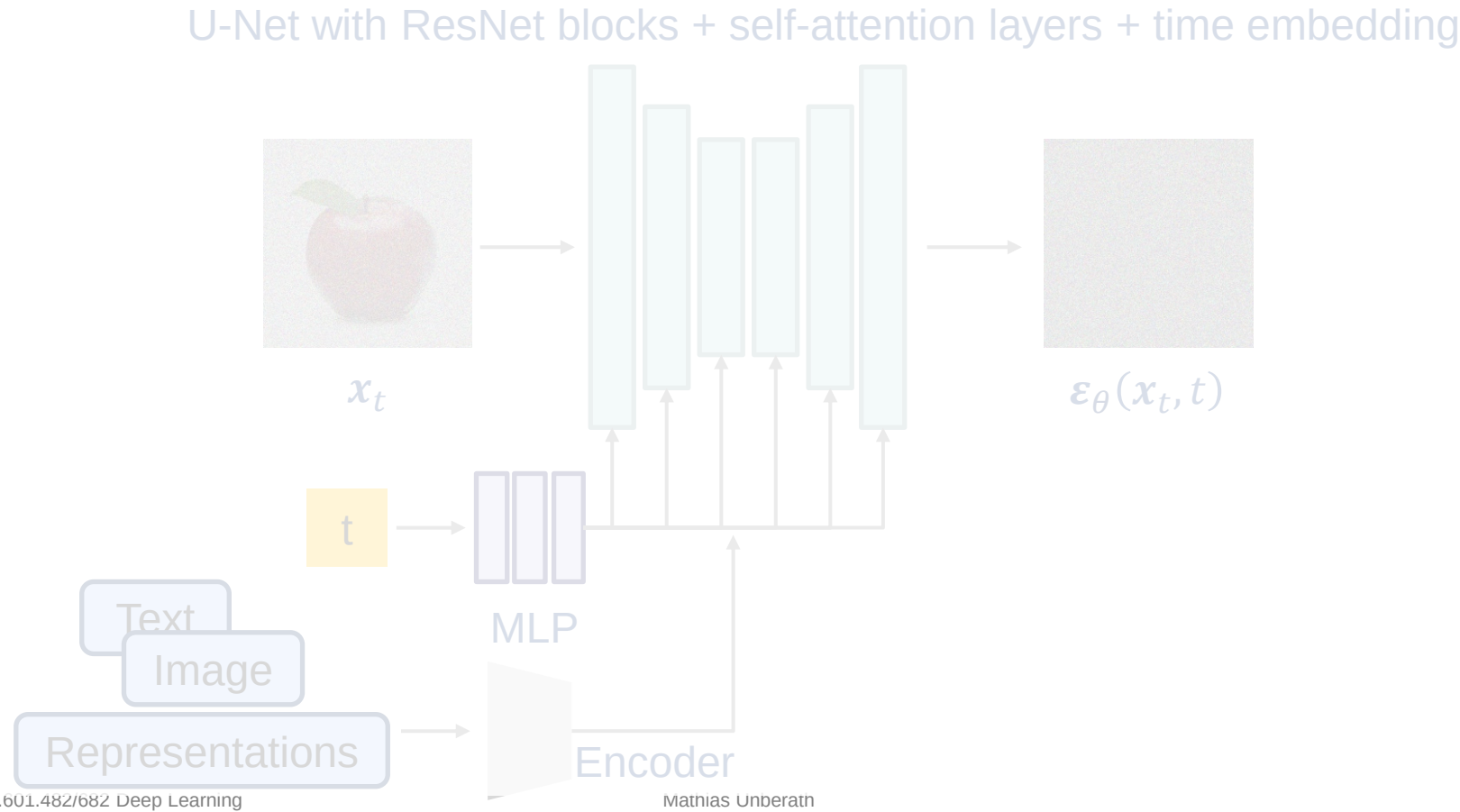
Conditional Denoising Network $\varepsilon_\theta(x_t, t, c)$

U-Net with ResNet blocks + self-attention layers + time embedding



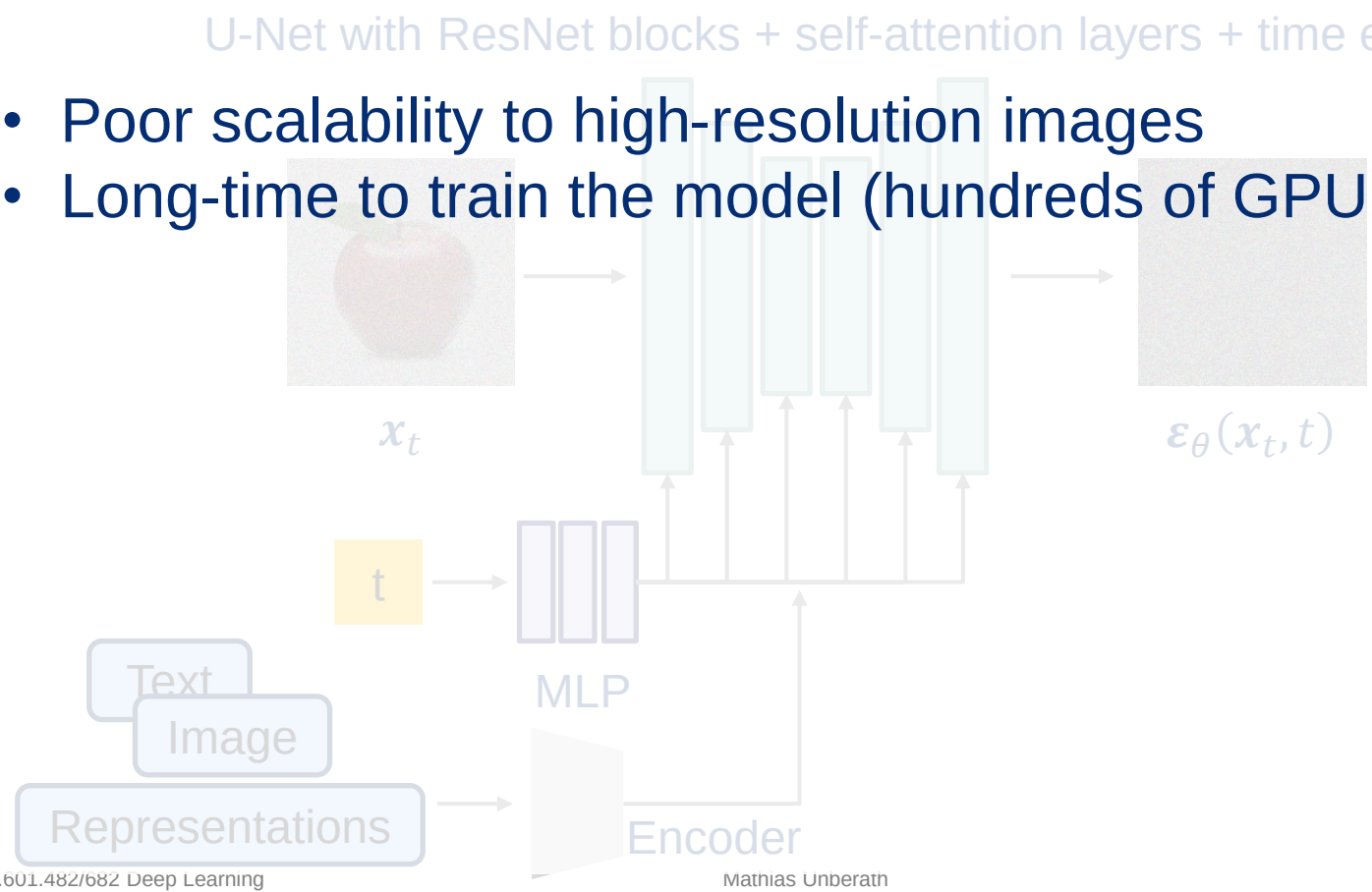
c = image category ->
classifier-free diffusion model

Limitations of the Pixel-wise Denoising



Limitations of the Pixel-wise Denoising

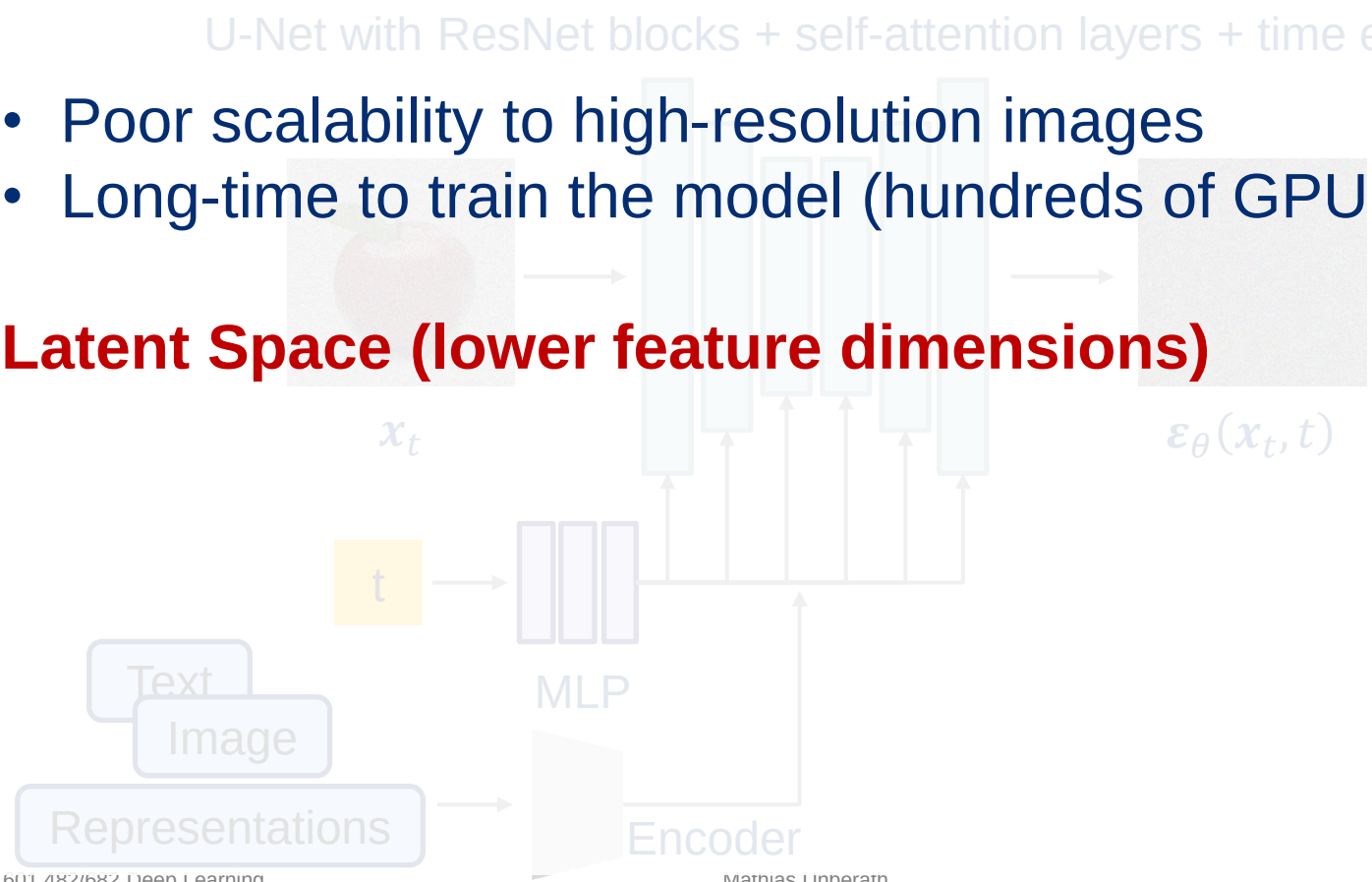
- Poor scalability to high-resolution images
- Long-time to train the model (hundreds of GPU days)



Limitations of the Pixel-wise Denoising

- Poor scalability to high-resolution images
- Long-time to train the model (hundreds of GPU days)

Latent Space (lower feature dimensions)



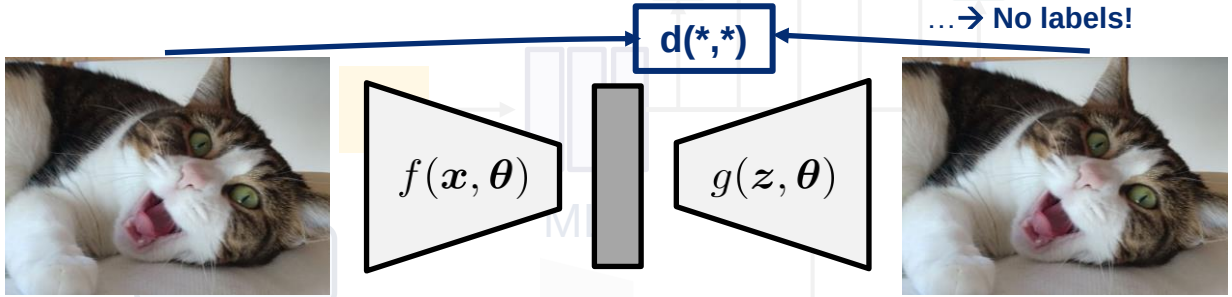
Limitations of the Pixel-wise Denoising

- Poor scalability to high-resolution images
- Long-time to train the model (hundreds of GPU days)

Latent Space (lower feature dimensions)

- AutoEncoder

Loss function: Can be arbitrary, e.g. L2, Mutual Information, …→ **No labels!**

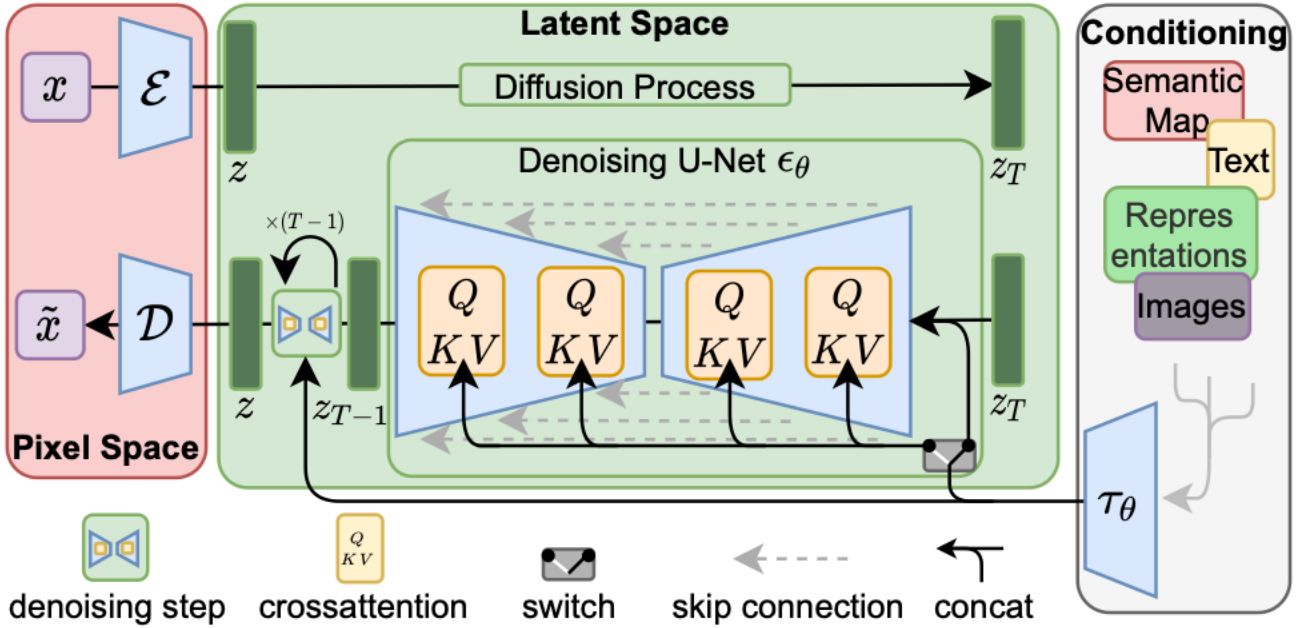


Latent Diffusion Model

- Train a compression model that strips away irrelevant high-level pixel-space details from an image and encodes it into a semantically equivalent low-dimensional latent
  - Also need a way to convert back from the latent space to the pixel space — naturally, we want an encoder-decoder architecture
  - Popular choices include VQ-VAE and VQ-GAN models

- Perform the diffusion process in this latent space. Benefits:
  - The diffusion process is only focusing on the relevant semantic bits of the data
  - Performing diffusion in a low-dimensional space is significantly faster

Laten Diffusion Model



- $\mathcal{E}$ — encoder to convert from pixels to latent
- $\mathcal{D}$ — decoder to convert from latents to pixels
- cross-attention for conditional generation
- U-Net style architecture as before

High-Resolution Image Synthesis with Latent Diffusion Models CVPR 2022

How much more efficient?

- Due to the fact that latent diffusion models (LDM) operate in a lower dimensional space, the required memory and compute is drastically reduced in comparison to pixel-space diffusion models.

- For instance, the current most popular LDM (stable diffusion) uses an autoencoder that has a reduction factor of 8. This essentially means that we perform diffusion on a 64x smaller latent and save 64x memory!

Inference Visualization

Intro Diffusion Models

Questions?