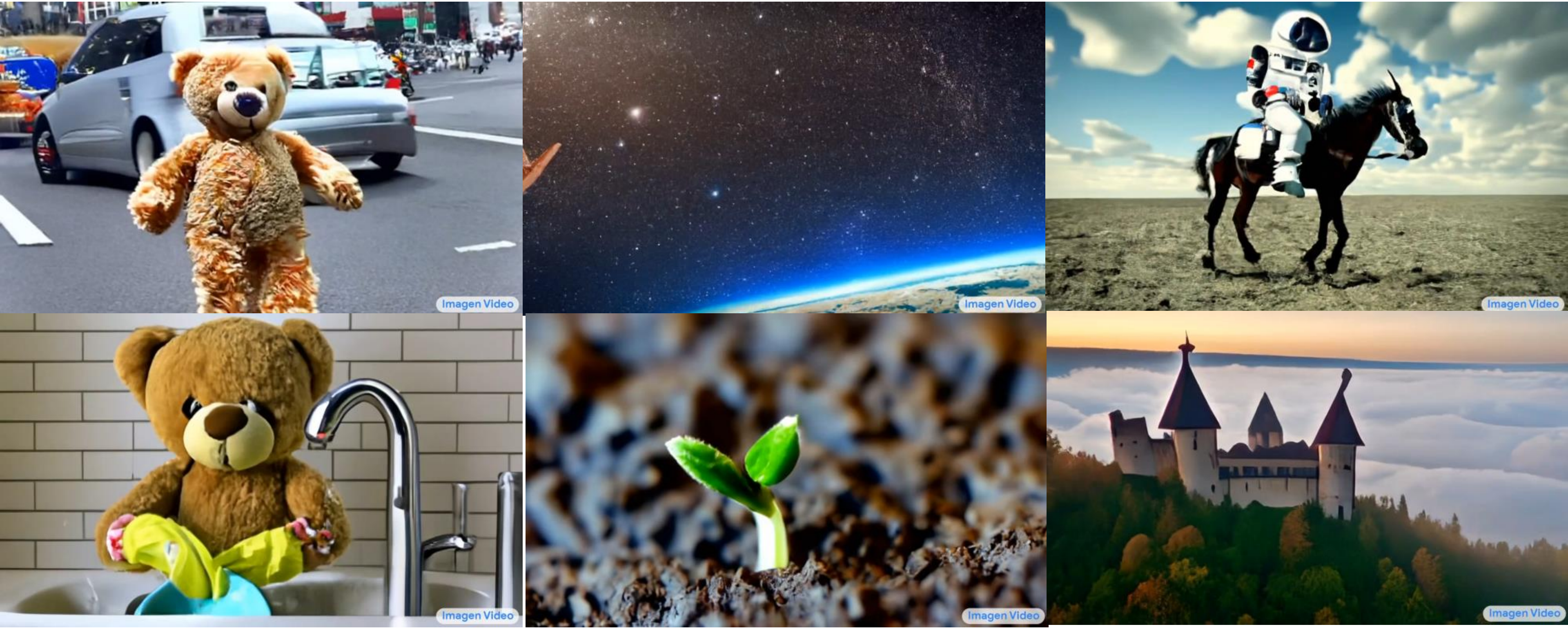


Videos generated by diffusion models

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
  - 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
  - Applications: Text-to-Image, Counterfactual, Inpainting
  - Formulation
  - Network
  - Latent Diffusion Model (*Stable Diffusion*)

Intro Diffusion Models

Reminder: Generative Models

Discriminative Models

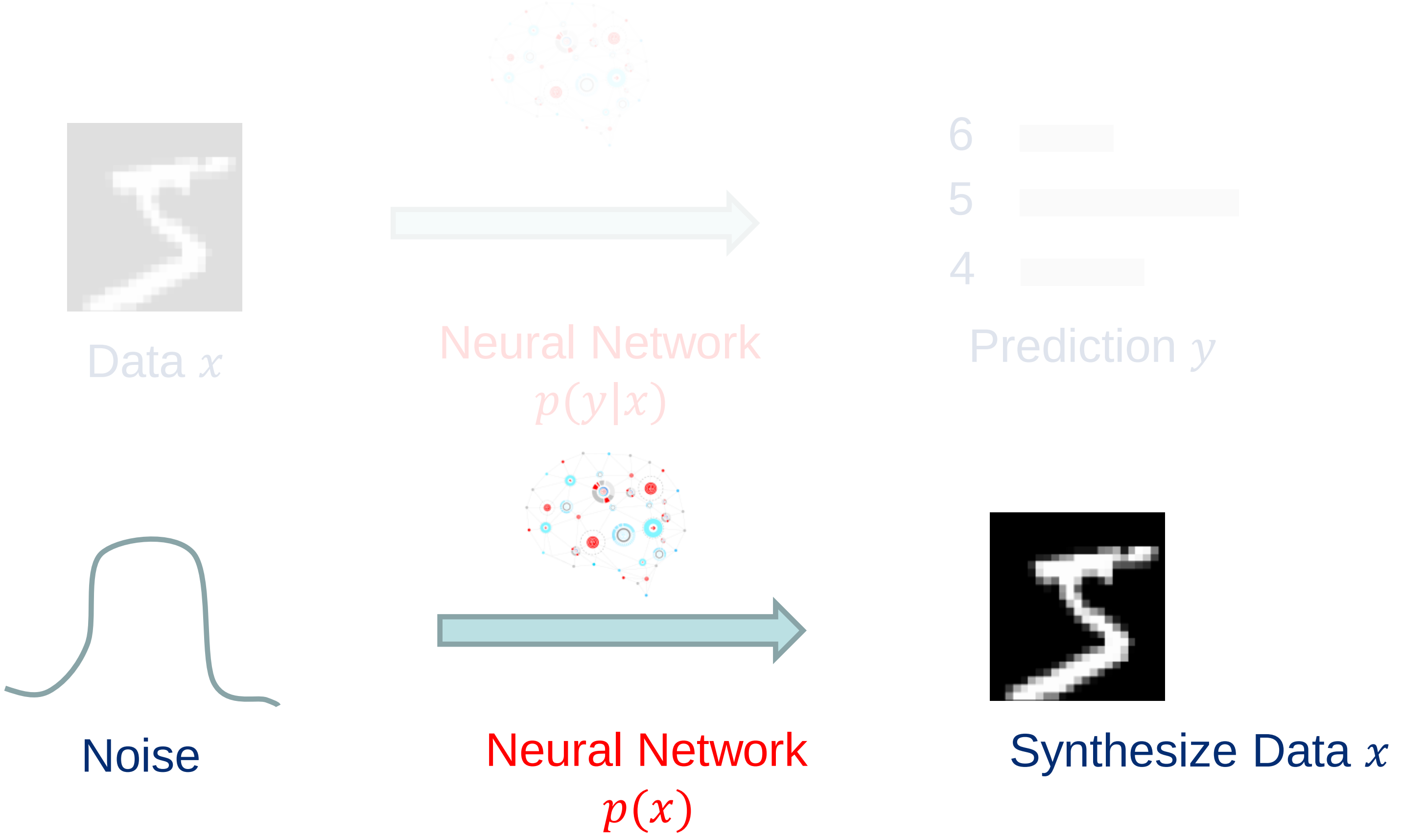
Data x

Neural Network

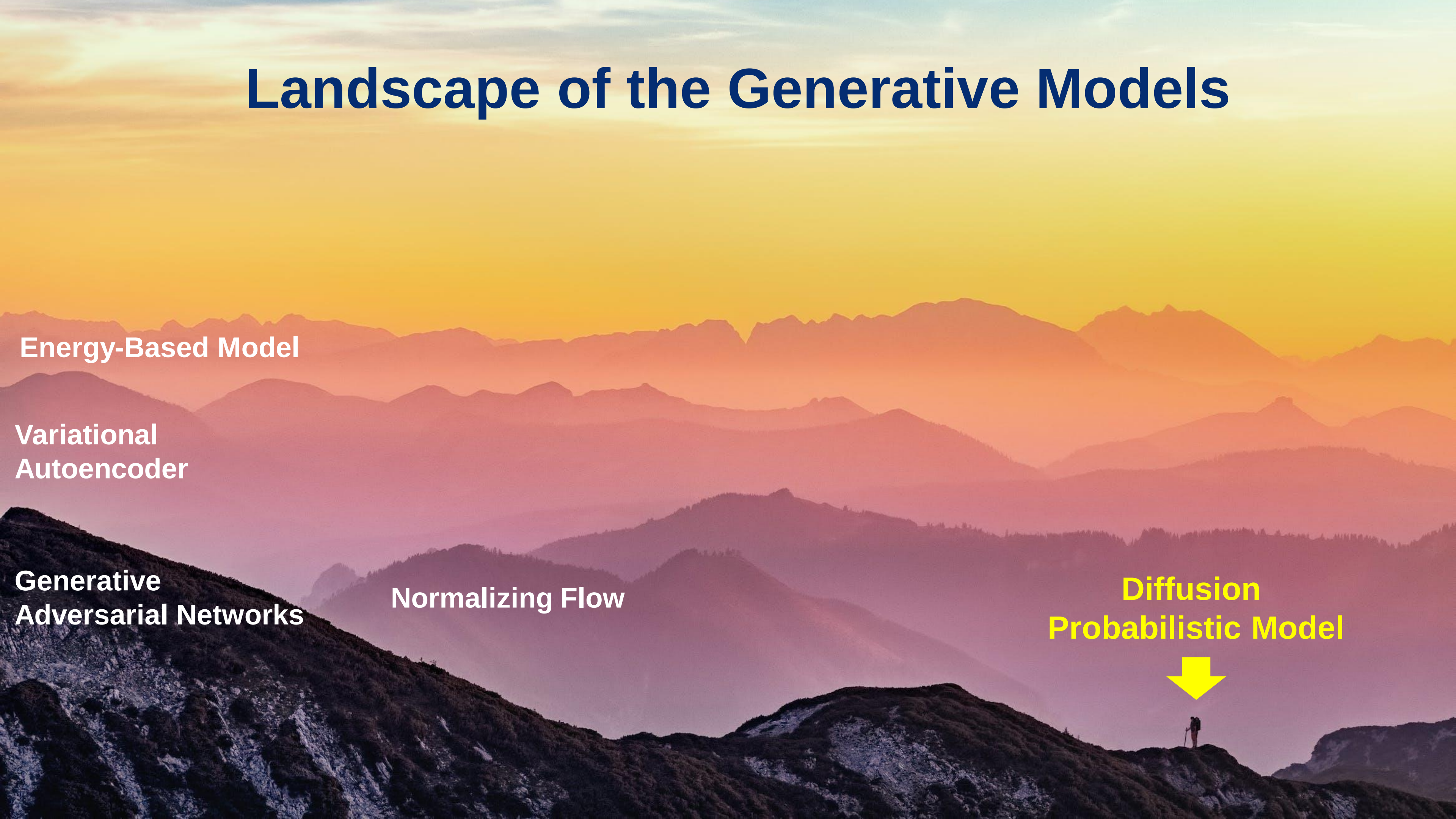
$p(y|x)$

Prediction y

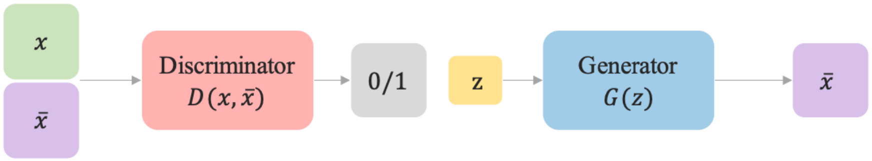
Generative Models



Landscape of the Generative Models

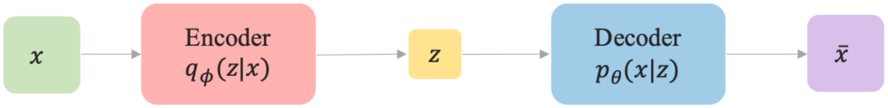

Landscape of the Generative Models
Energy-Based Model
Variational Autoencoder
Generative Adversarial Networks
Normalizing Flow
Diffusion Probabilistic Model

Why Diffusion Models?



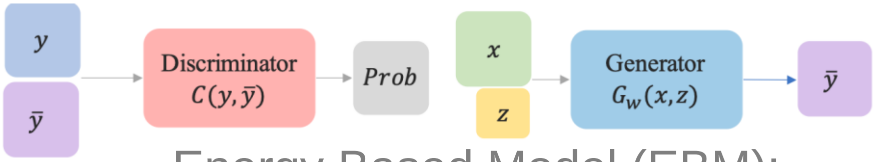
Generative Adversarial Network (GAN):

training additional discriminators



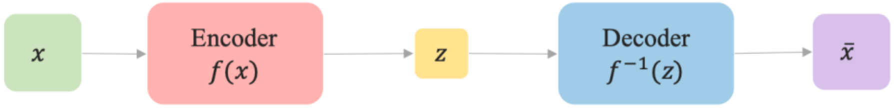
Variational Auto Encoder (VAE):

require aligning posterior distributions



Energy Based Model (EBM):

intractable partition functions



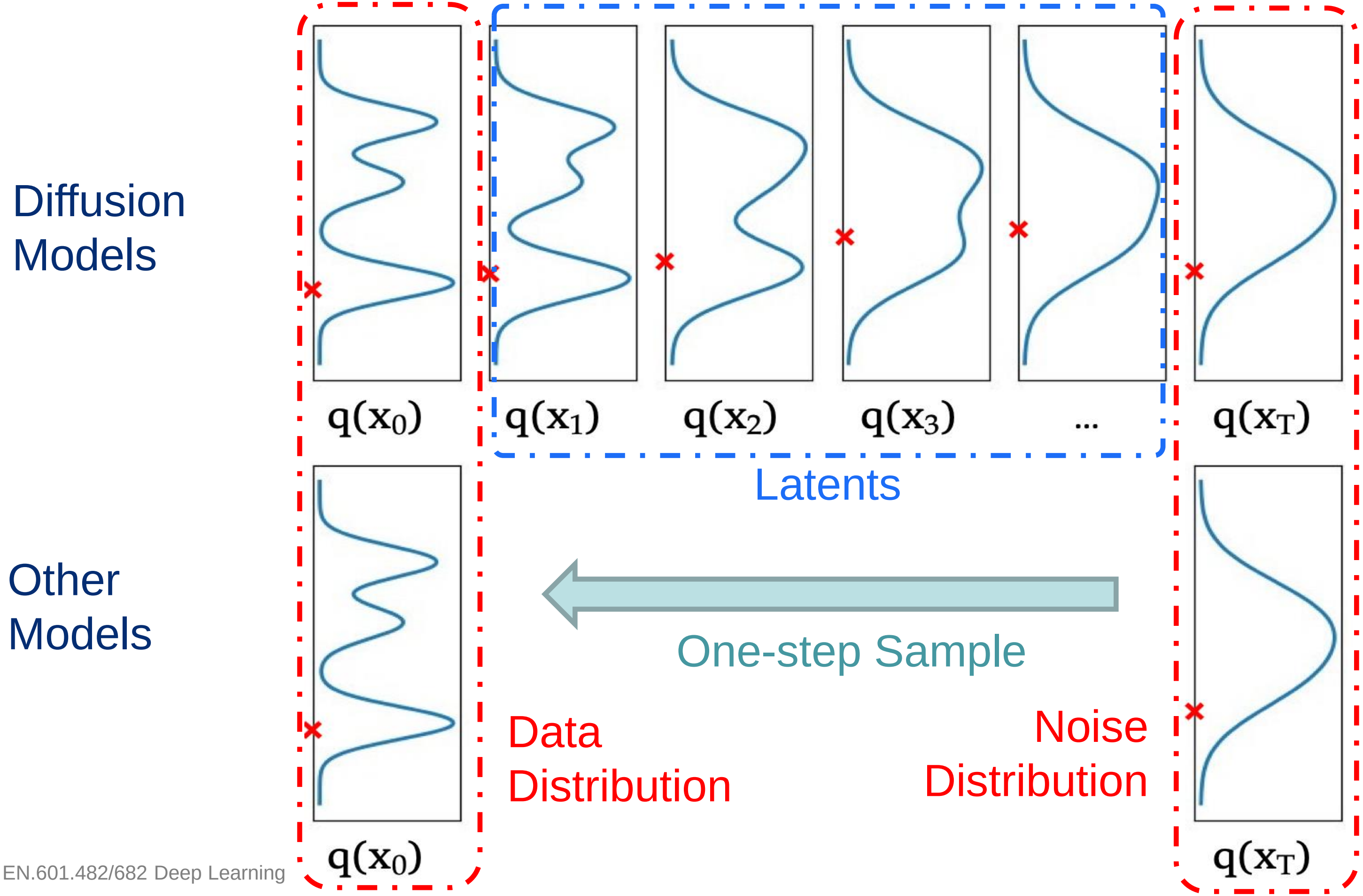
Normalizing Flow (NF):

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

Why Diffusion Models?



The Power of Diffusion Models: Text-to-Image Generation

DALL·E 2 (OpenAI)

"A teddy bear on a skateboard in times square"

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

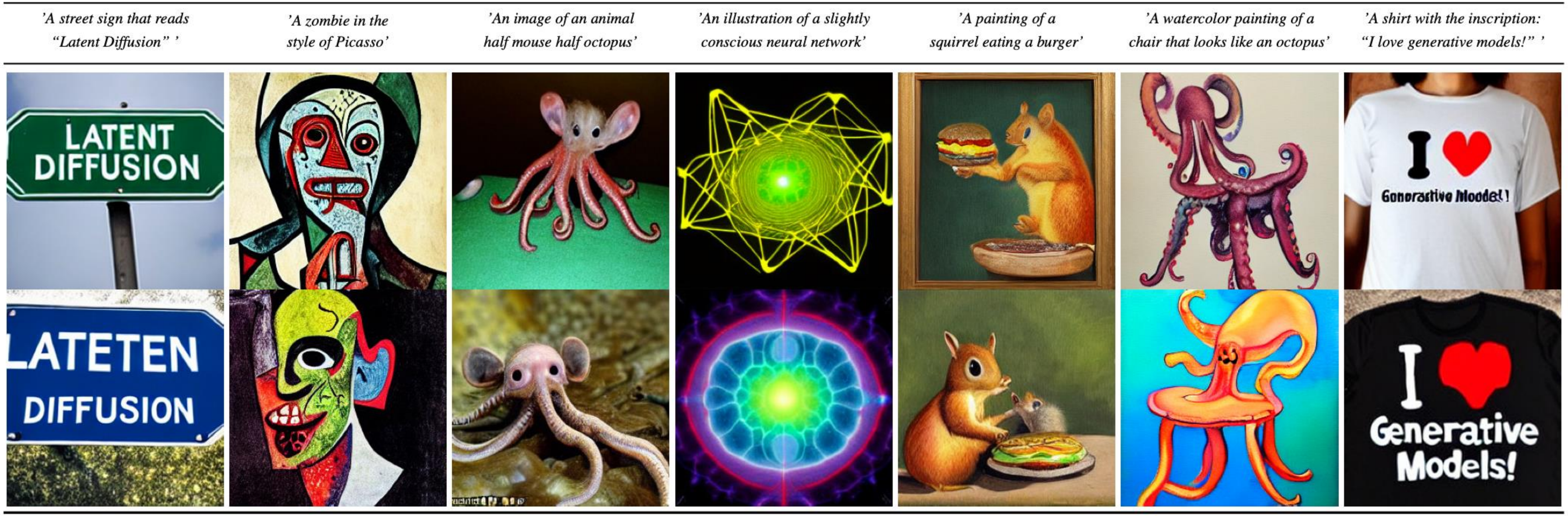
Imagen (Google)

"A group of teddy bears in suit in a corporate office celebrating the birthday of their friend. There is a pizza cake on the desk."

https://imagen.research.google

The Power of Diffusion Models: Text-to-Image Generation

Stable Diffusion (Stability AI)



https://stability.ai/blog/stable-diffusion-public-release

High-Resolution Image Synthesis with Latent Diffusion Models CVPR 2022

The Power of Diffusion Models: Text-to-Image Generation

Midjourney v5



The Power of Diffusion Models: Text-to-Image Generation

Midjourney v5

- **Wider Stylistic Range**
- **Higher Resolution**
- **Greater Clarity and Precision**
- **Broader Aspect Ratio Options**

Intro Diffusion Models

Denoising Diffusion Probabilistic Model

(DDPM)

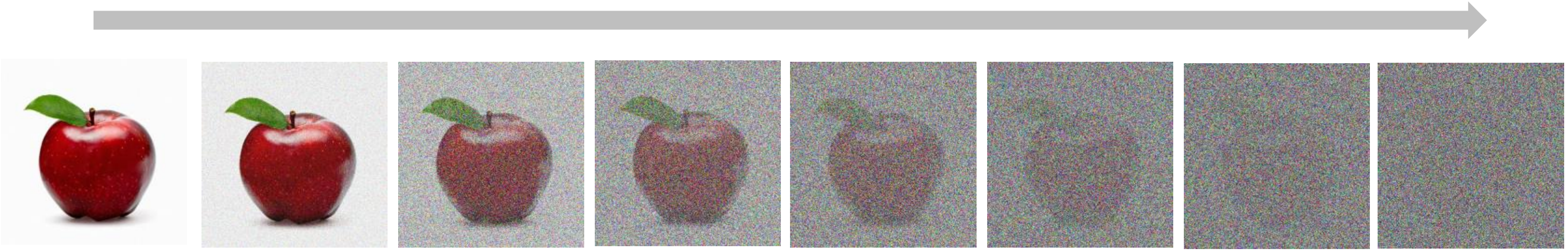
Basic Concept of Diffusion

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

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.

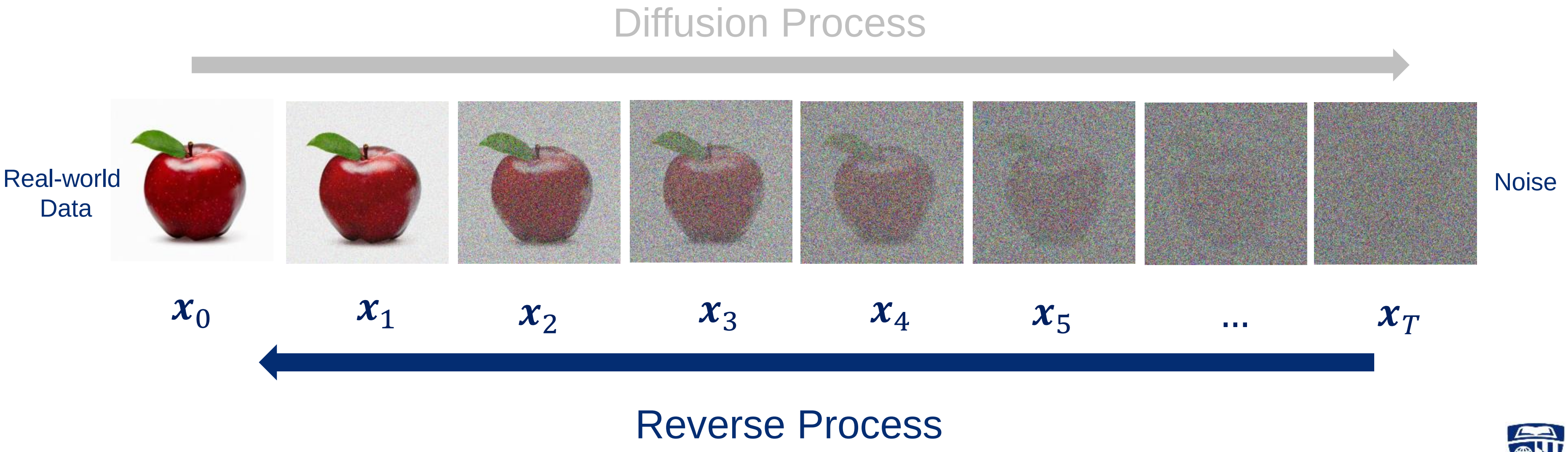
Diffusion Process



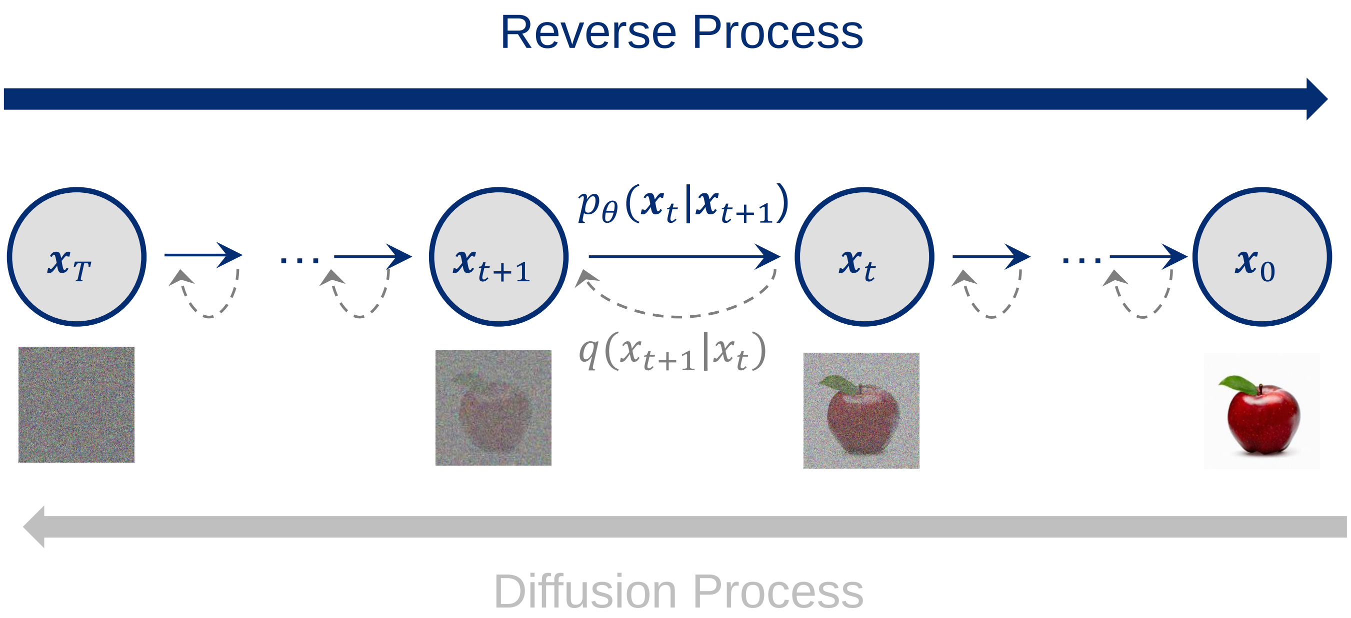
$\boldsymbol{x}_0$	$\boldsymbol{x}_1$	$\boldsymbol{x}_2$	$\boldsymbol{x}_3$	$\boldsymbol{x}_4$	$\boldsymbol{x}_5$	…	$\boldsymbol{x}_T$
Real-world Data							Noise

What is Diffusion Probabilistic Model?

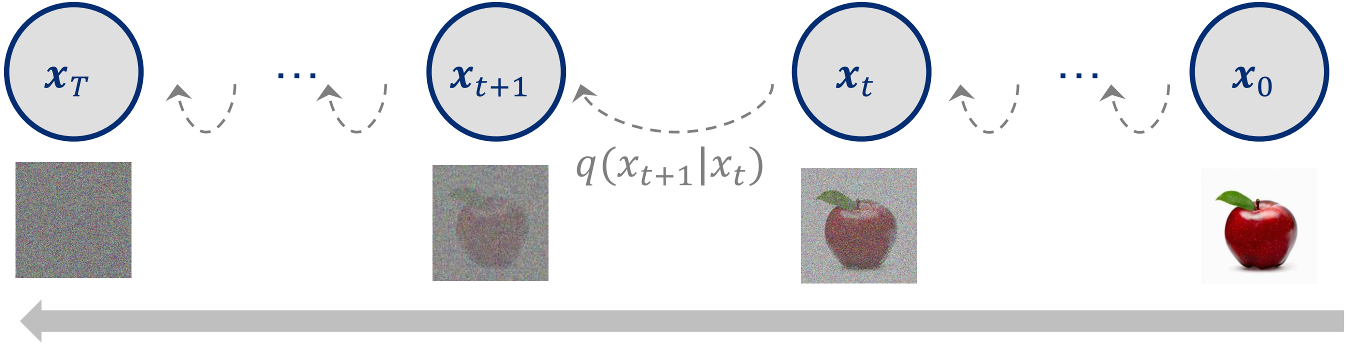
- Consists of two processes.
  - Diffusion/Forward process: gradually add noise to the input
  - Reverse process: learns to denoise -> generate new data



DDPM In the View of a Directed Graph

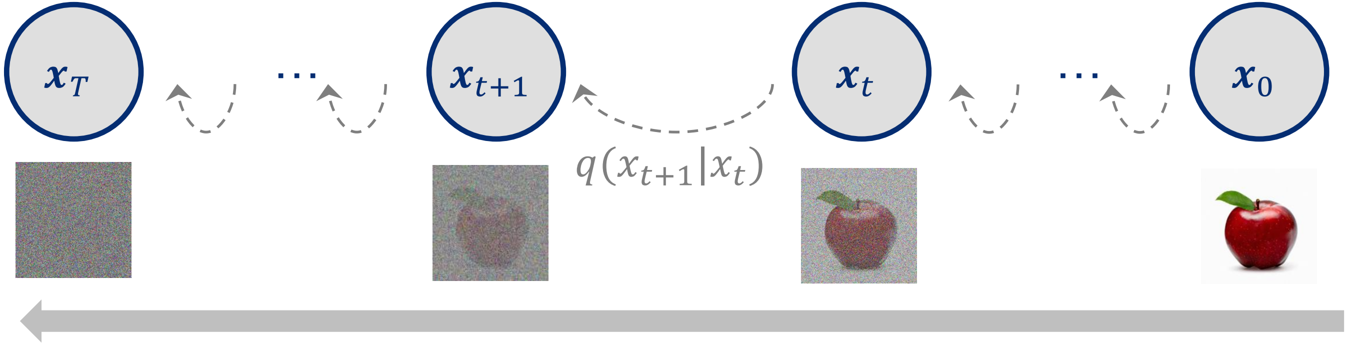


Diffusion/Forward Process (1/3)



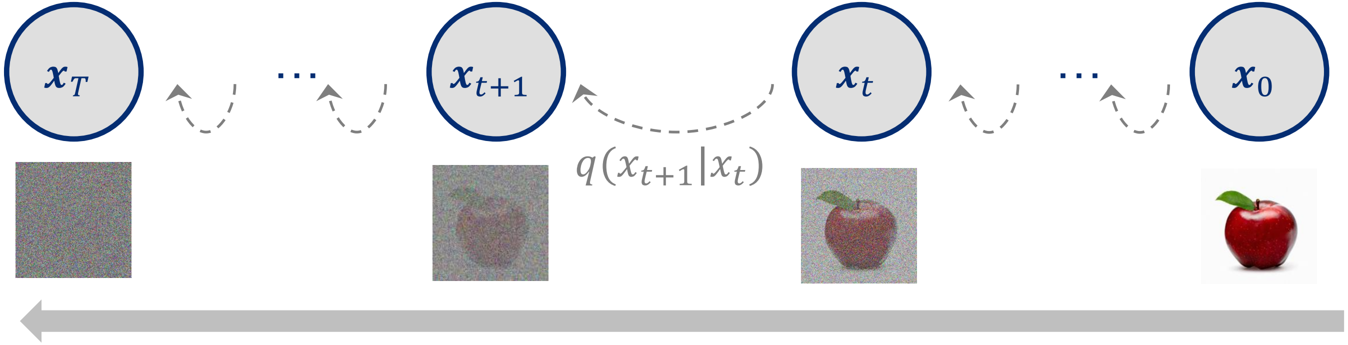
- Motivation: transforms the starting state ($\boldsymbol{x}_0$) into the tractable noise ($\boldsymbol{x}_i$)

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 $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$ as the **diffusion process**.

Diffusion/Forward Process (1/3)

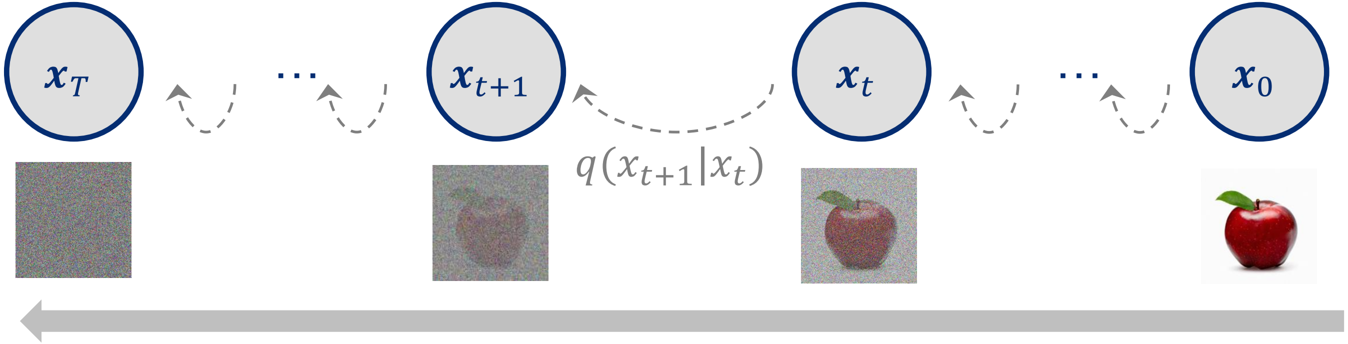


- Motivation: transforms the starting state ($\boldsymbol{x}_0$) into the tractable noise ($\boldsymbol{x}_i$)
- Formally, we call the joint distribution $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$ as the **diffusion process**.
- In DDPM, $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$ is defined as a Markov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

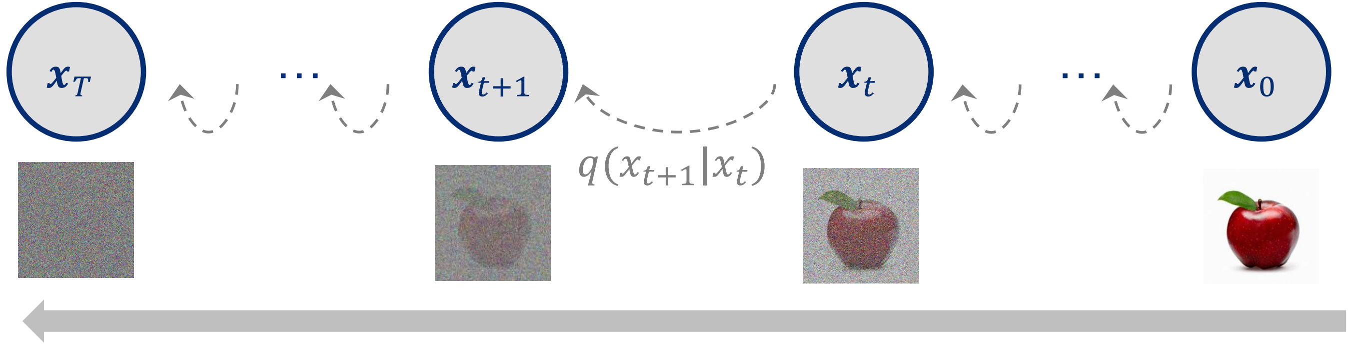


- Motivation: transforms the starting state ($\boldsymbol{x}_0$) into the tractable noise ($\boldsymbol{x}_i$)
- Formally, we call the joint distribution $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$ as the **diffusion process**.
- In DDPM, $q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$ is defined as a Markov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

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

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:

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

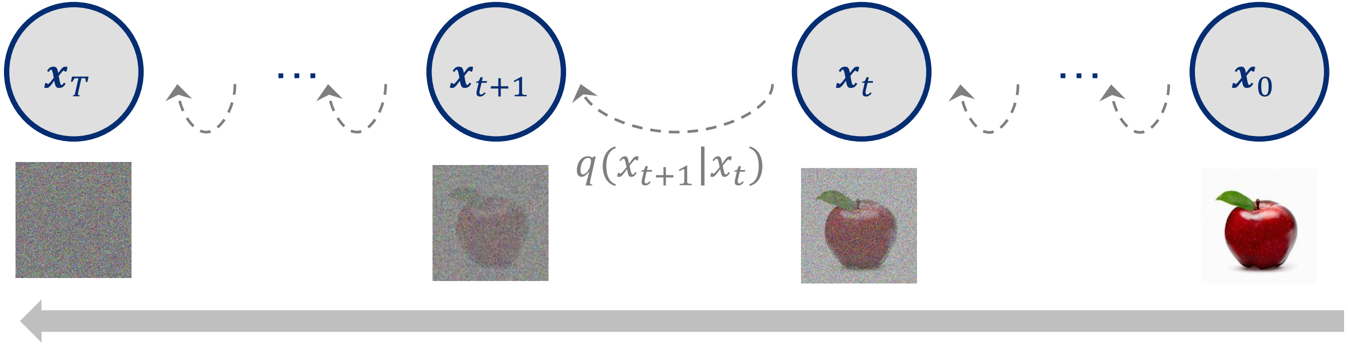
Transaction kernel

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

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

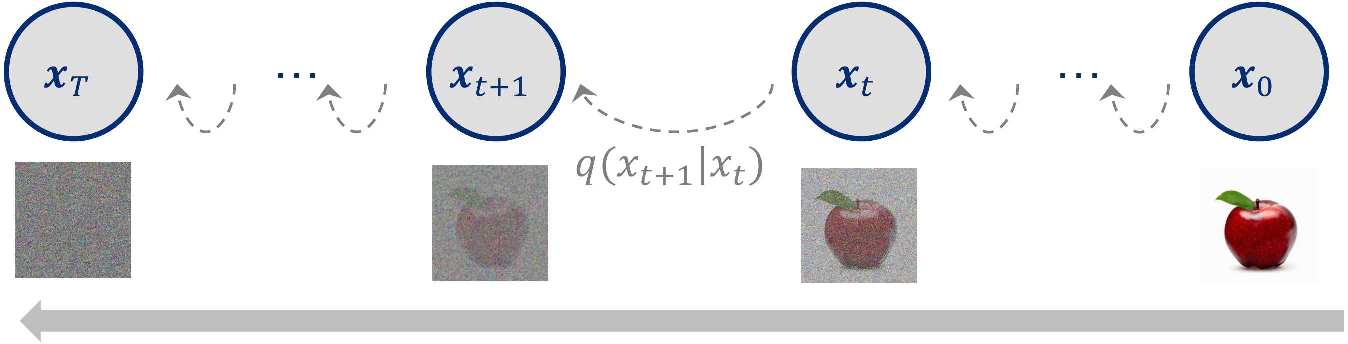
Parameters in the range of $(0,1)$.

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)

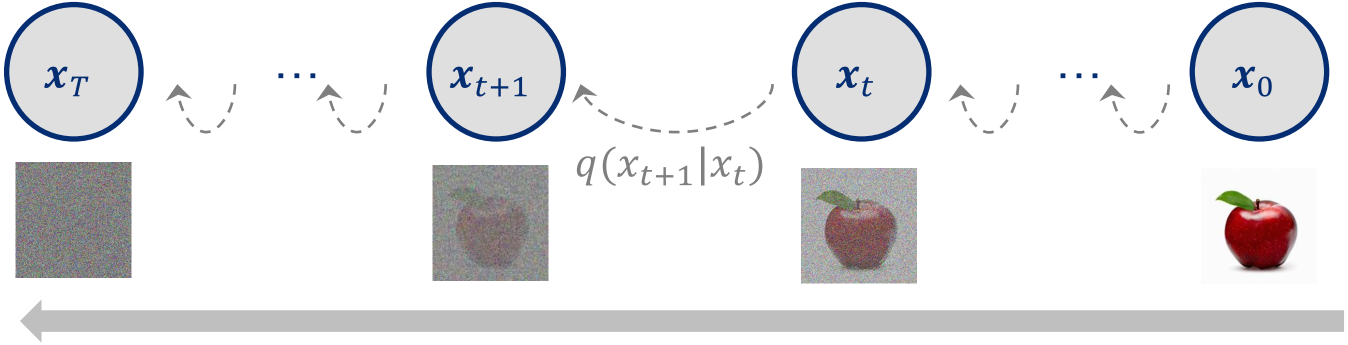


Q: Why Gaussian perturbation i.e., $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}\left(\boldsymbol{x}_t \middle| \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t \boldsymbol{I}\right)$?

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

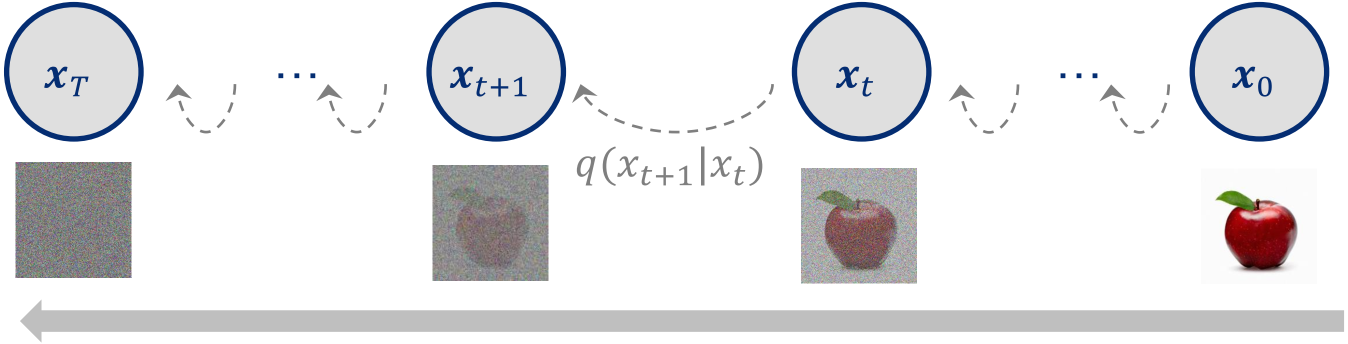
A: Composition of Gaussians is still Gaussian

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

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

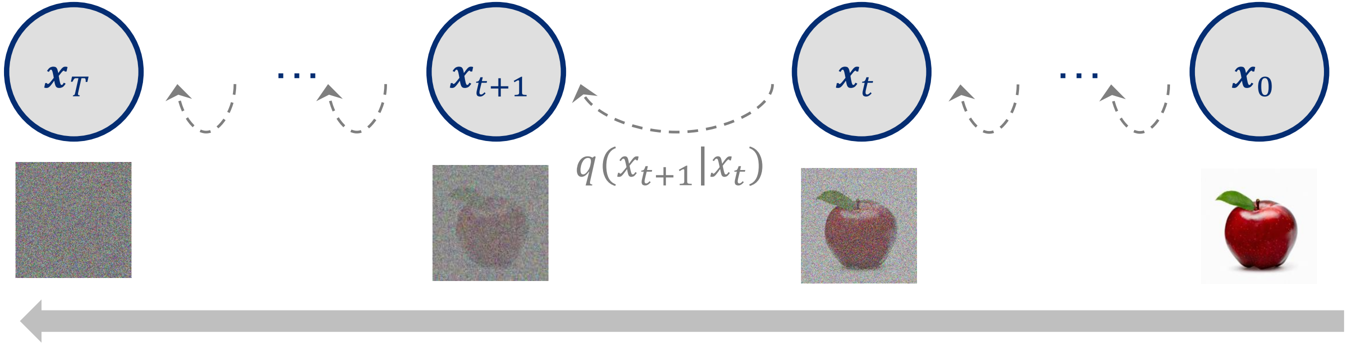
$$\lim_{n\to\infty} \overline{\alpha_t} = 0 \quad \text{and} \quad \lim_{t\to\infty} q(\boldsymbol{x}_t) = \lim_{t\to\infty} q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(0, \boldsymbol{I}).$$

Diffusion/Forward Process (3/3)



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

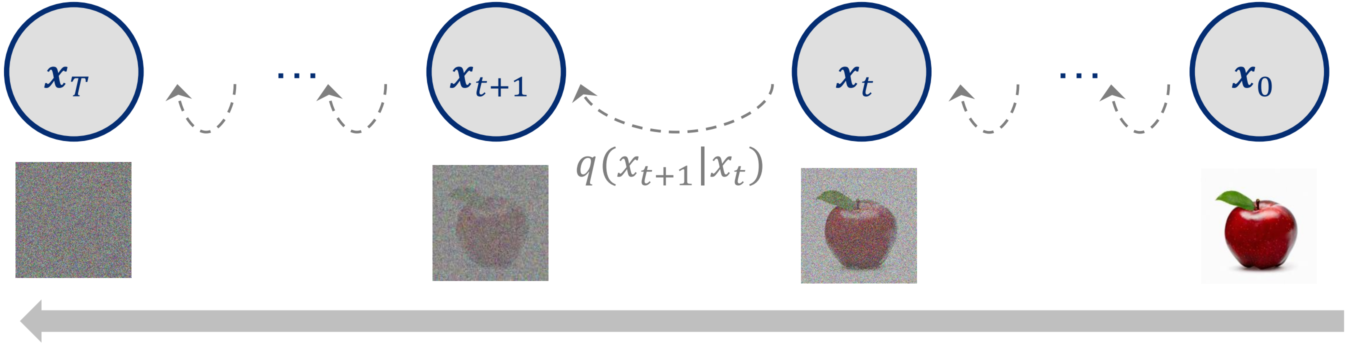
Diffusion/Forward Process (3/3)



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)

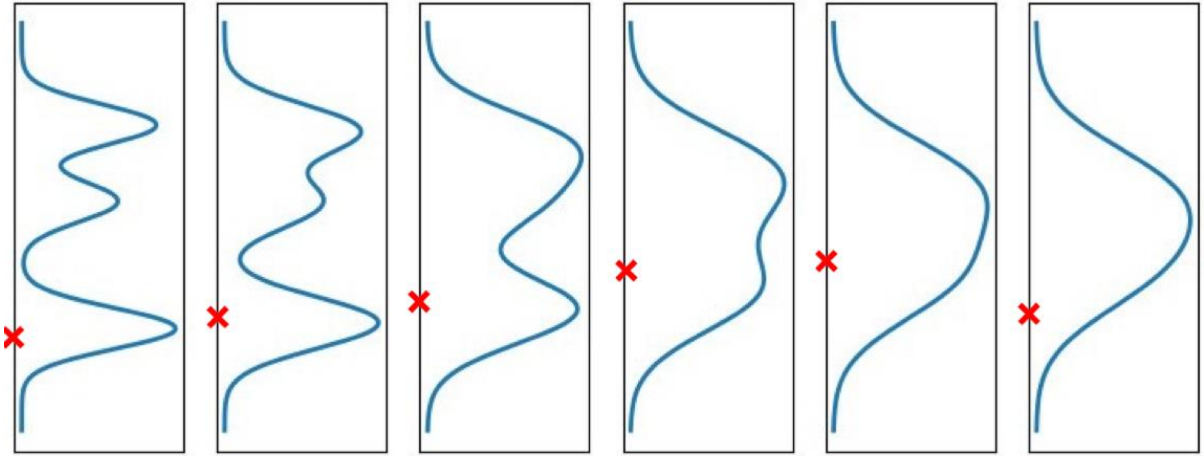


Q: How to sample from $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}\left(\boldsymbol{x}_t \middle| \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t \boldsymbol{I}\right)$?

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

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 $\epsilon \sim \mathcal{N}(0, \boldsymbol{I})$.

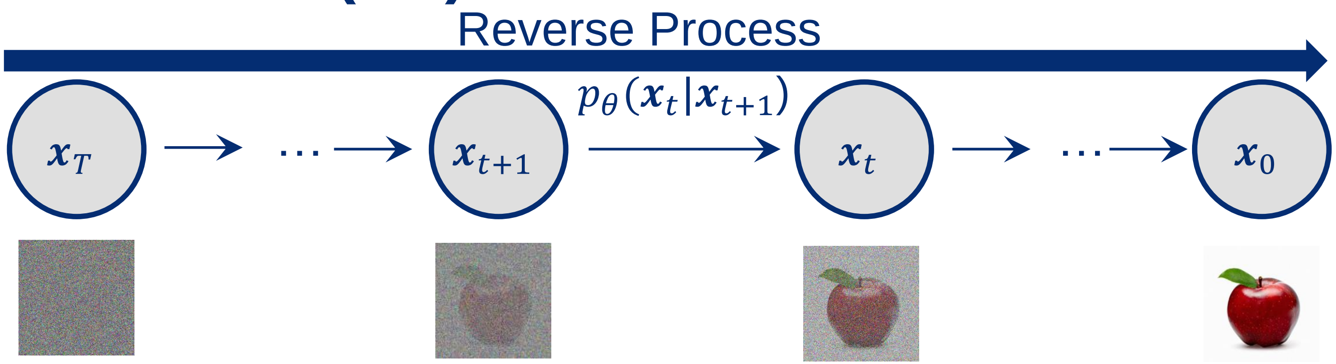
What Happen in the Diffusion/Forward Process?



$q(x_0)$ $\quad$ $q(x_1)$ $\quad$ $q(x_2)$ $\quad$ $q(x_3)$ $\quad$ ... $\quad$ $q(x_T) = \mathcal{N}(0, \boldsymbol{I})$

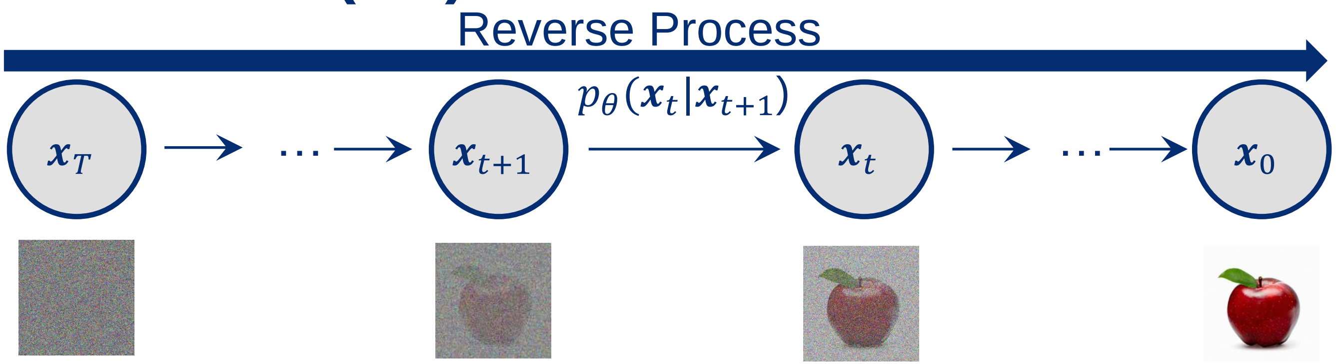
$$\underbrace{q(\boldsymbol{x}_t)}_{\text{Diffused Data Distribution}} = \int \underbrace{q(\boldsymbol{x}_0, \boldsymbol{x}_t)}_{\text{Joint Data Distribution}} d\boldsymbol{x}_0 = \int \underbrace{q(\boldsymbol{x}_0)}_{\text{Input Data Distribution}} \underbrace{q(\boldsymbol{x}_t | \boldsymbol{x}_0)}_{\text{Diffusion Kernel}} d\boldsymbol{x}_0$$

Reverse Process (1/2)



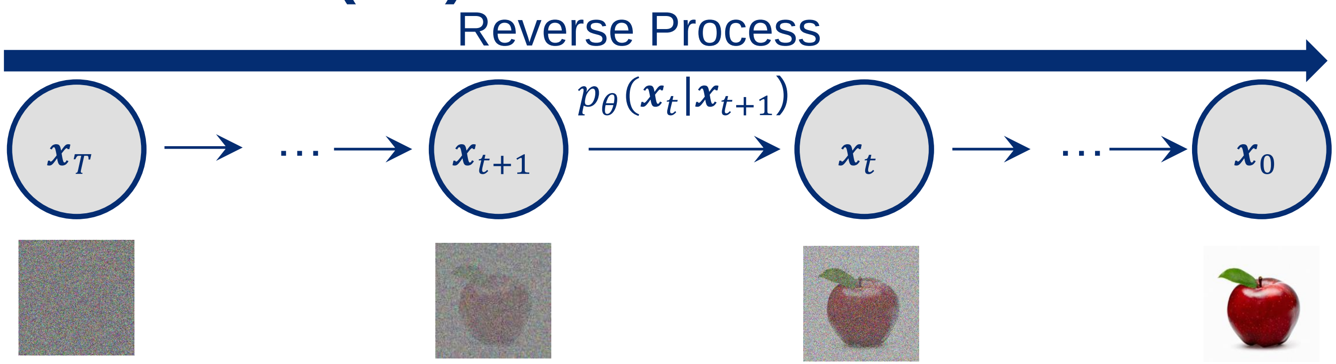
- Motivation: Reverse $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ to reconstruct image ($\boldsymbol{x}_0$) from noise ($\boldsymbol{x}_T$).

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**.

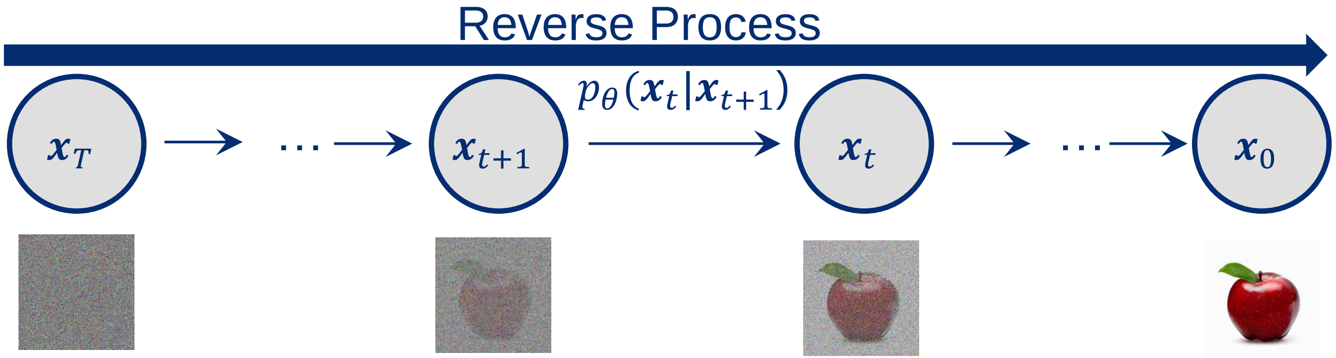
Reverse Process (1/2)



- 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.

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

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

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

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

Reverse Process (2/2)

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

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

Reverse Process (2/2)

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

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

- 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

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

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.*, $D_{KL}\left(\mathrm{q}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) \middle|\middle| p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)\right)$, which can be simplified to:

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

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.*, $D_{KL}(\mathrm{q}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)||p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t))$, 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 $\widetilde{\boldsymbol{\mu}_t}(\boldsymbol{x}_t, \boldsymbol{\varepsilon}) = \frac{1}{\sqrt{\alpha_t}}(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\boldsymbol{\varepsilon})$.

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

- It leads the loss function to be

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

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

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

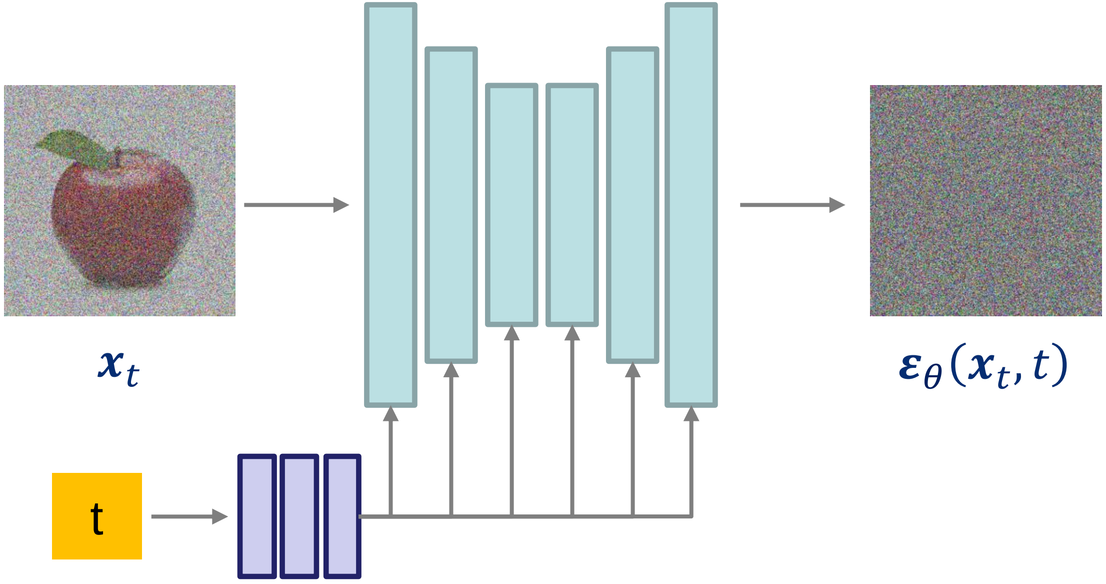
- where t is uniform between 1 and T.

Algorithm 1 Training

1: **repeat**
2: $\quad \mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $\quad t \sim \mathrm{Uniform}(\{1, \dots, T\})$
4: $\quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: $\quad$ 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 $\varepsilon_\theta(x_t, t)$

U-Net 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

Sampling Process

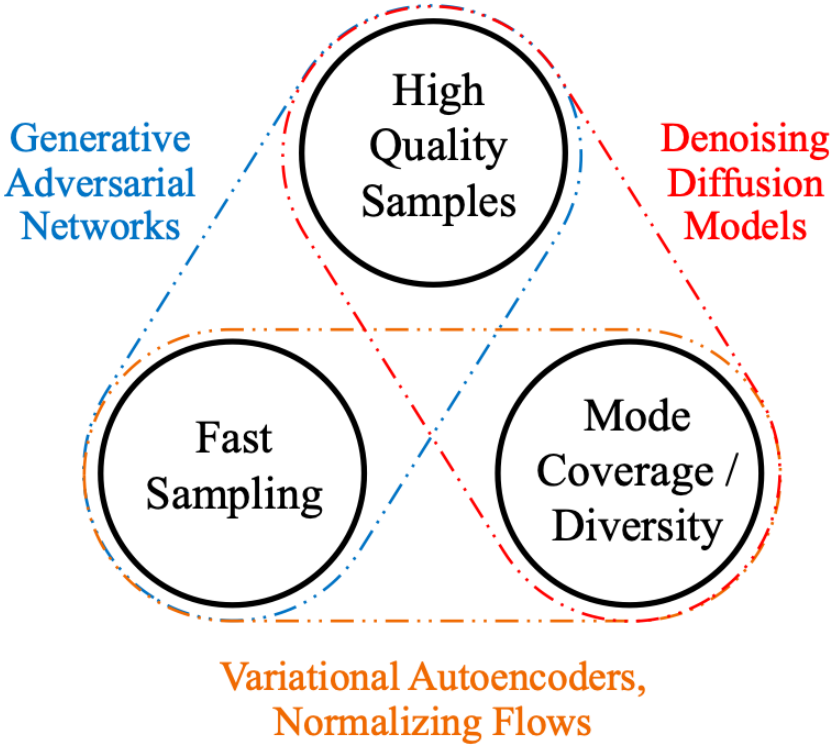
- **Goal** Generate a sample $\widehat{\boldsymbol{x}_0}$ from the Gaussian $\boldsymbol{x}_T$.
- **Limitation** Slow. Take 20 hours to sample 50k images of size 32 × 32 on a NVIDIA 2080Ti (*vs.* a GAN takes less than 1 min)

Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ **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 Model



Tackling the Generative Learning Trilemma with Denoising Diffusion GANs ICLR 2022

Intro Diffusion Models

Conditional Diffusion Model

Applications of Conditional Diffusion Models

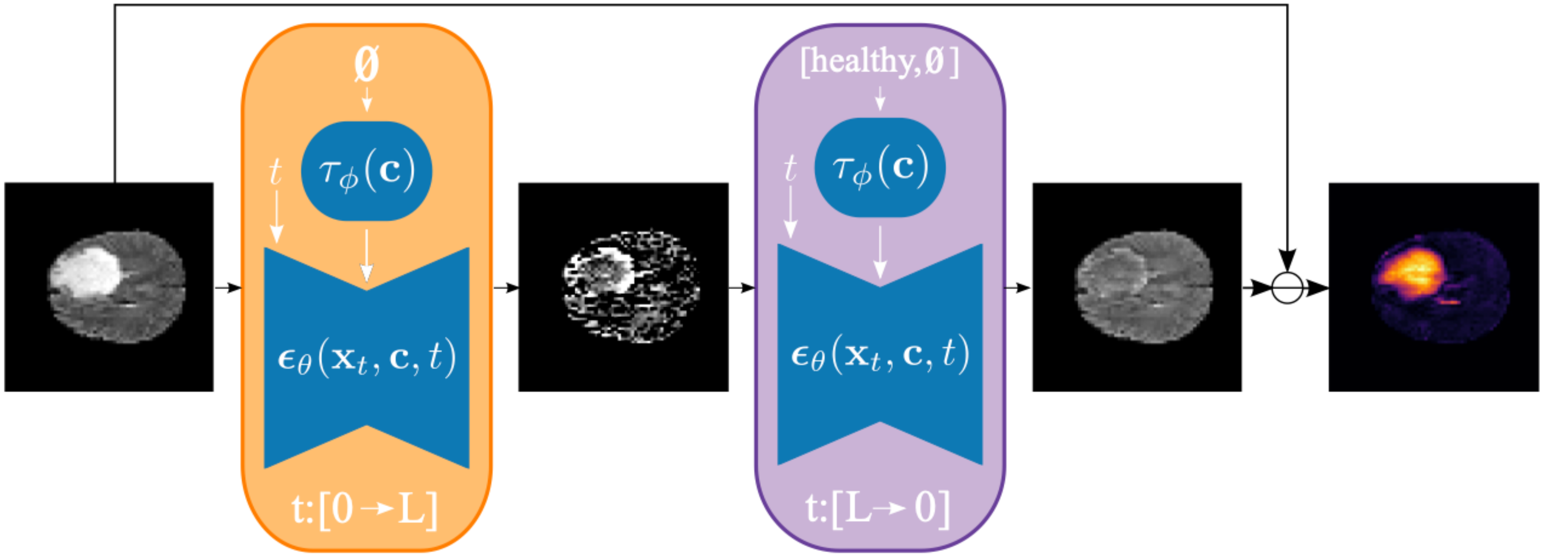
Text-to-Image Generation

"A teddy bear on a skateboard in times square"

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

Applications of Conditional Diffusion Models

Counterfactual Generation



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

Applications of Conditional Diffusion Models

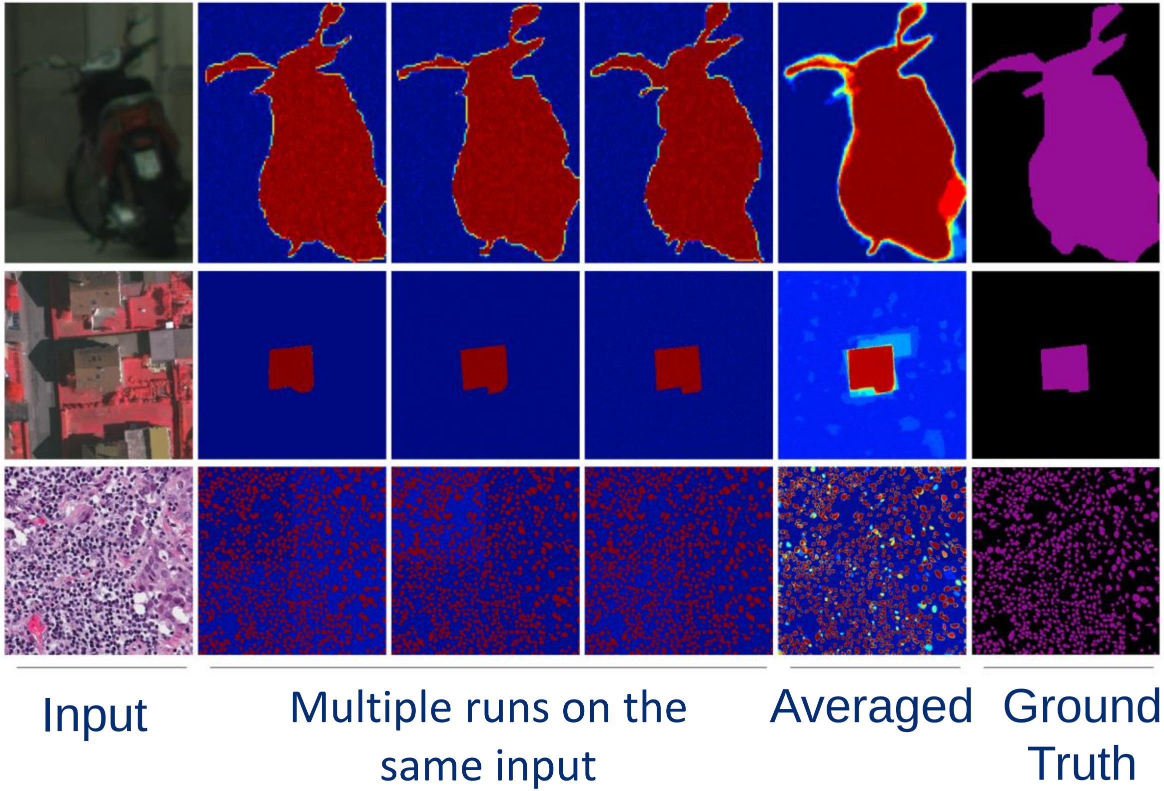
Image Inpainting

Randomness

RePaint: Inpainting using Denoising Diffusion Probabilistic Models , CVPR 2022

Applications of Conditional Diffusion Models

Image Segmentation



Include Condition to Reverse Process

- Conditional Reverse Process:

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

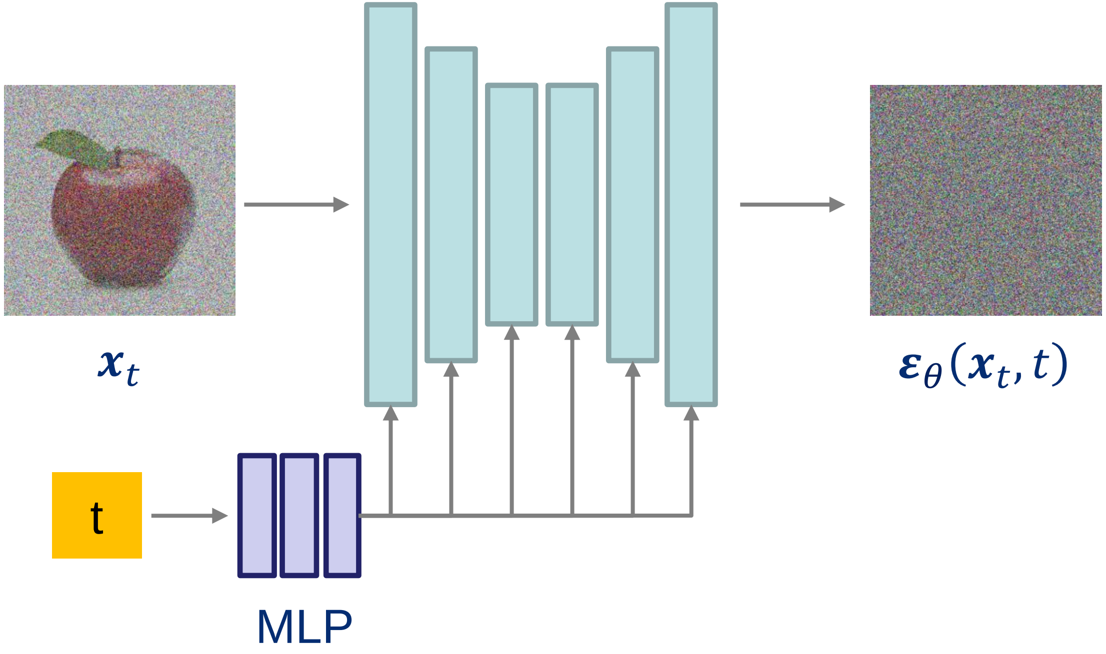
$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{c}) = \mathcal{N}(\boldsymbol{x}_{t-1}|\boxed{\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t, \boldsymbol{c})}, \boldsymbol{\Sigma}_\theta(\boldsymbol{x}_t, t, \boldsymbol{c}))$$

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.

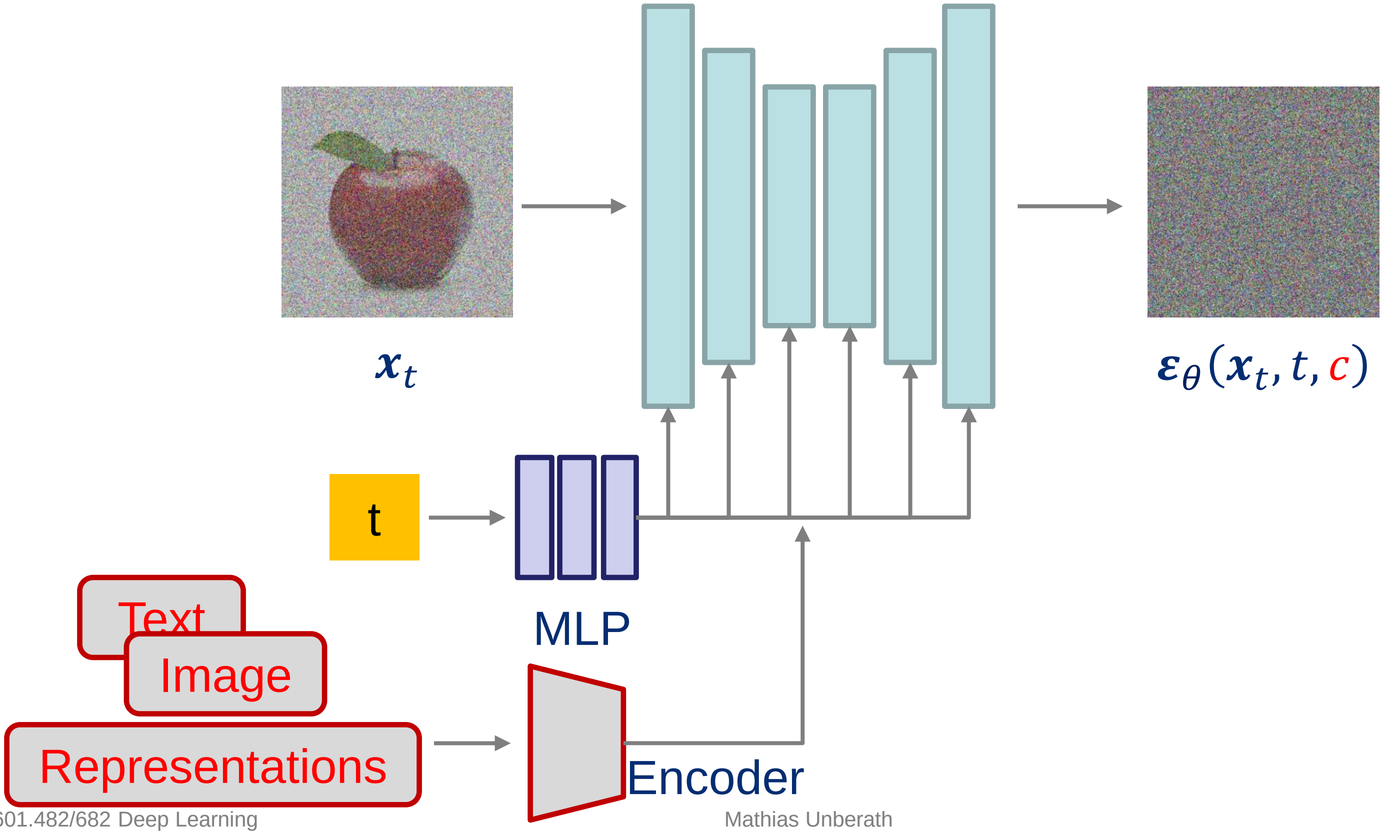
Conditional Denoising Network $\varepsilon_\theta(\boldsymbol{x}_t, t, \boldsymbol{c})$

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



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

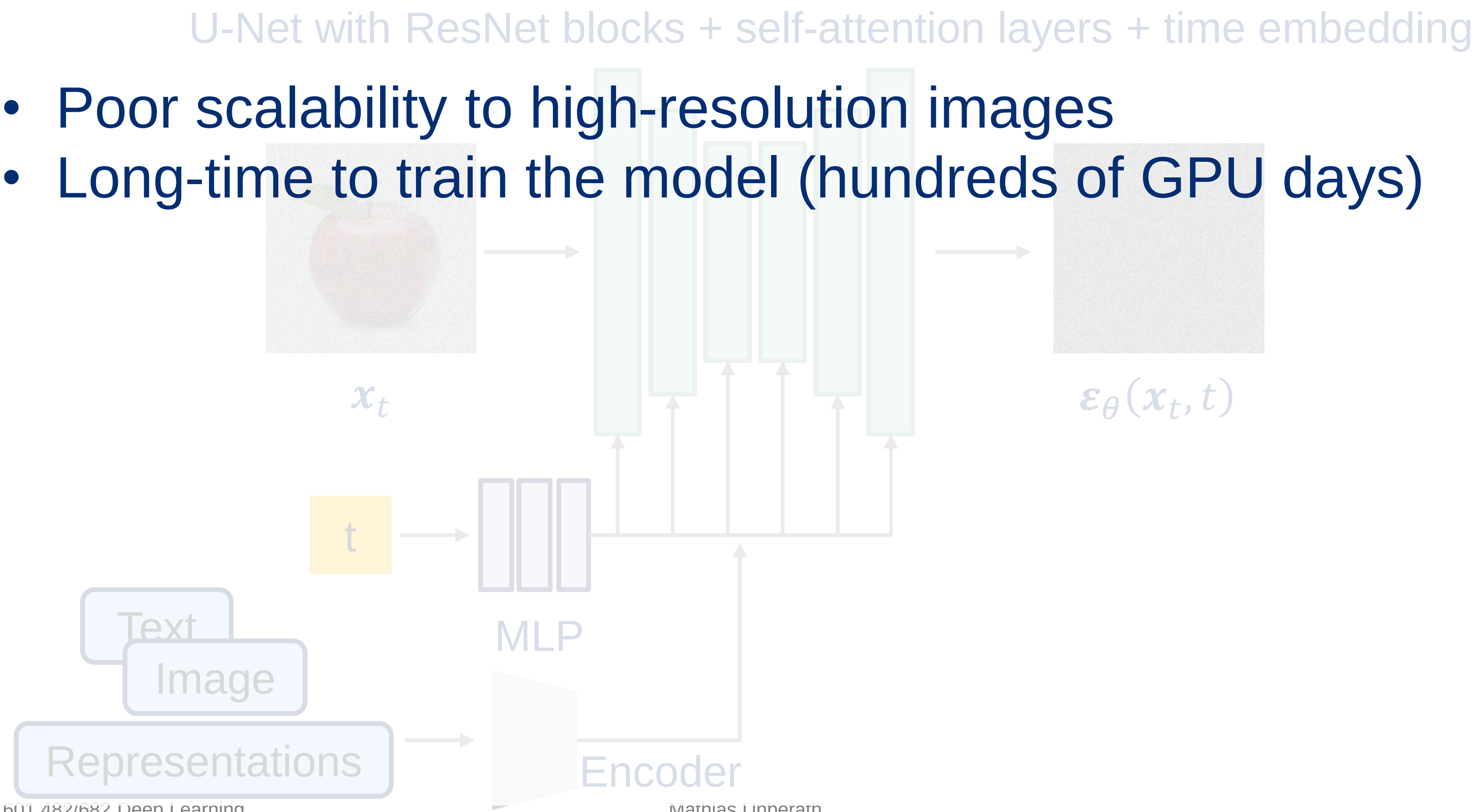
U-Net with ResNet blocks + self-attention layers + time embedding



Limitations of the Pixel-wise Denoising

Limitations of the Pixel-wise Denoising

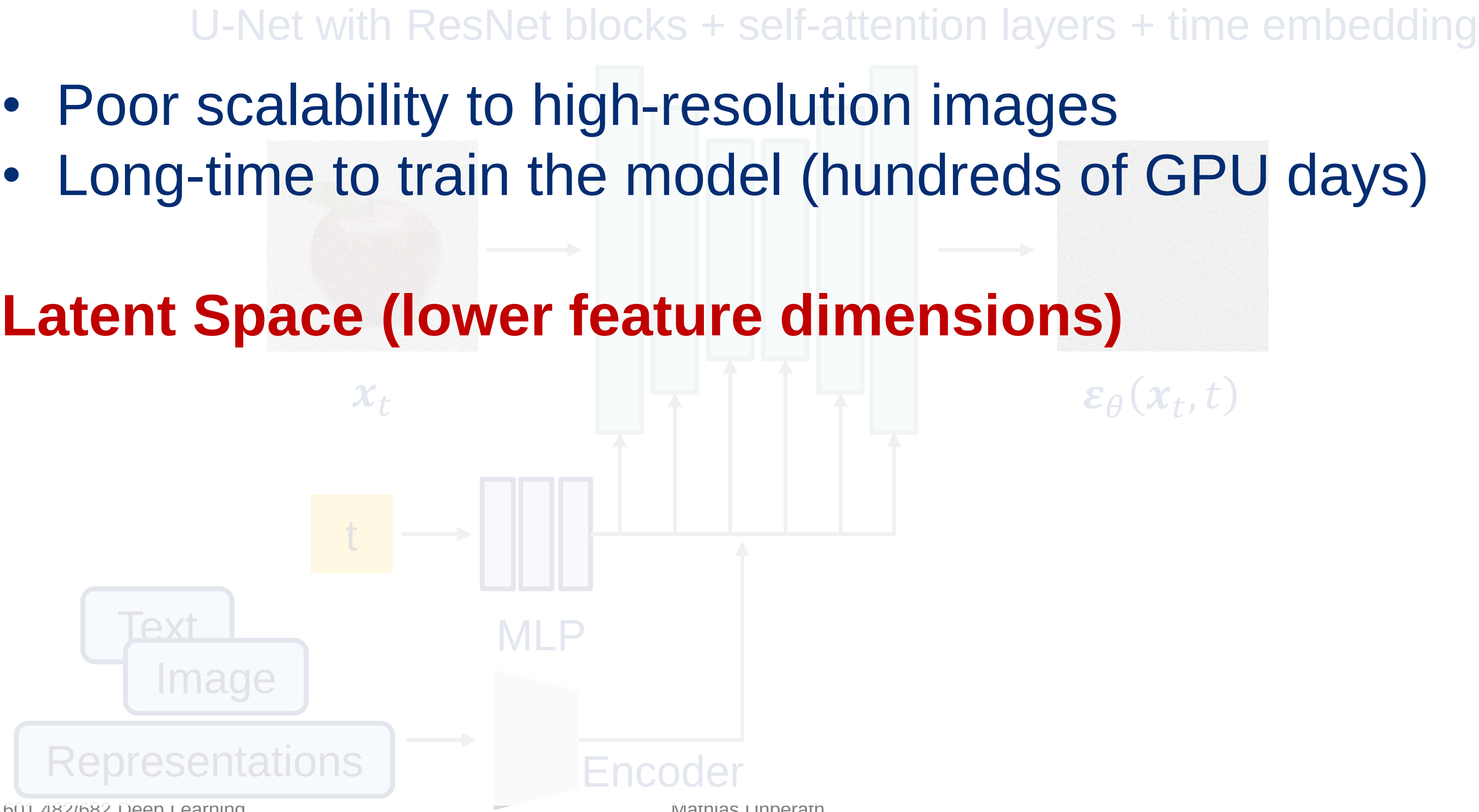
- Poor scalability to high-resolution images
- Long-time to train the model (hundreds of GPU days)



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)



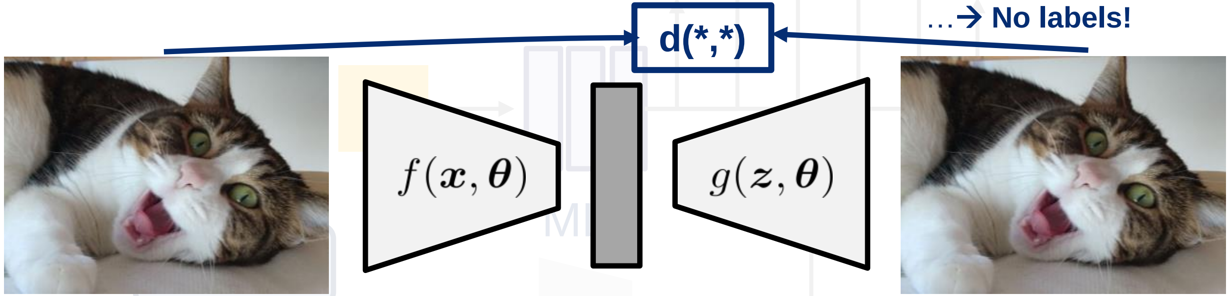
Limitations of the Pixel-wise Denoising

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

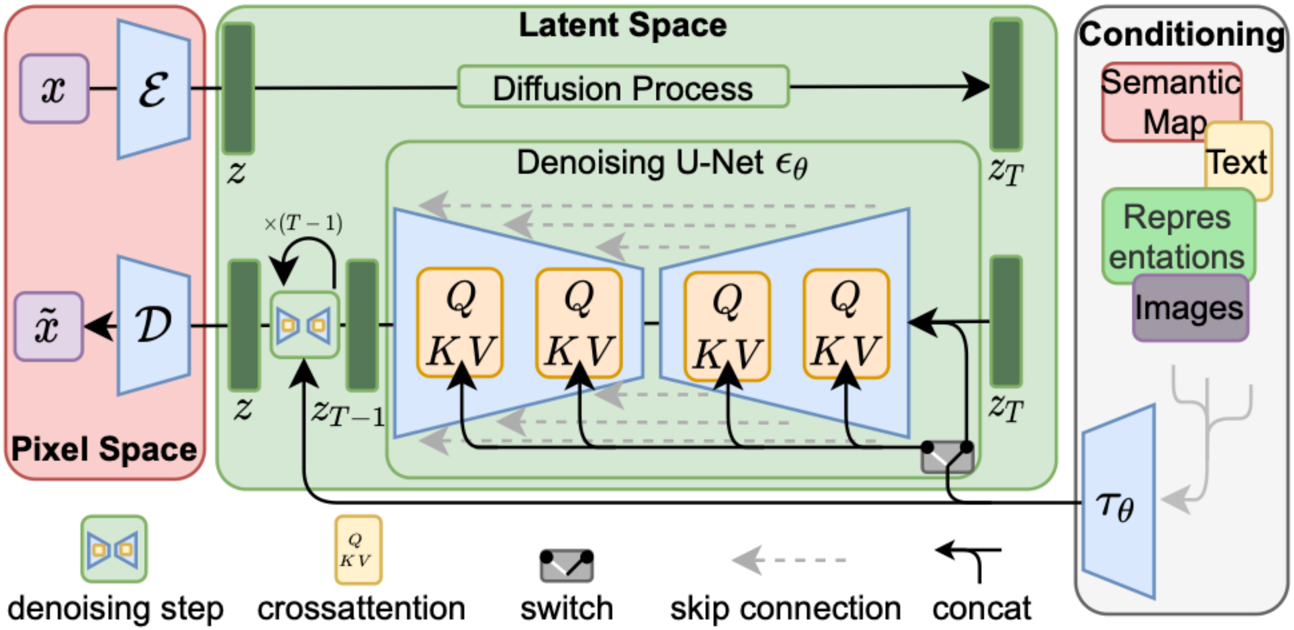
Latent Space (lower feature dimensions)

- AutoEncoder

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

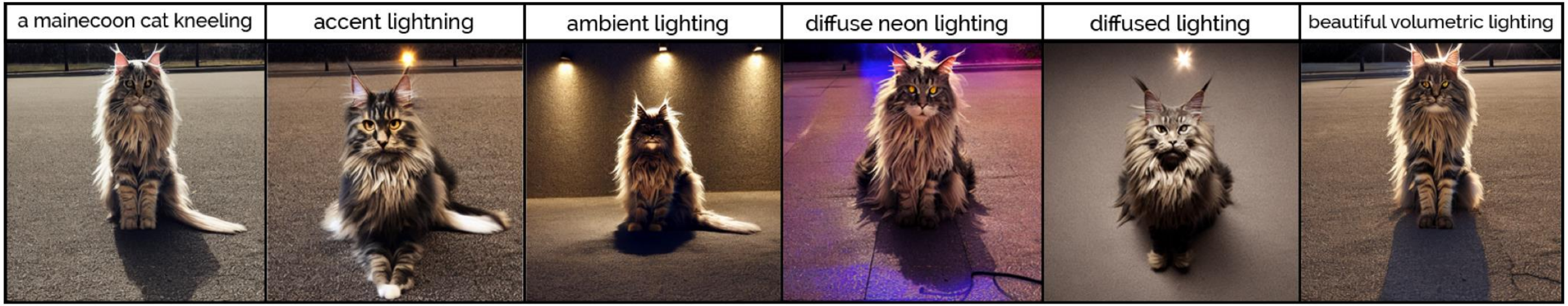


Laten Diffusion Model



High-Resolution Image Synthesis with Latent Diffusion Models CVPR 2022

The Power of Prompt Engineering in Diffusion Model



Adding ‘Lighting’ Words

https://arxiv.org/pdf/2211.15462.pdf


atmospheric lighting
back light
beautiful lightning
volumetric lighting
casting lighting
cinema lighting
dark shadows
diffuse lighting
diffuse neon lighting
dramatic cinematic lighting
dynamic volumetric lighting
eerie lighting

Stable Diffusion Prompts

The Stable Diffusion prompts search engine.

Explore millions of AI generated images and create collections of prompts. Search generative visuals for everyone by AI artists everywhere in our 12 million prompts database.

Create better prompts. Generative visuals for everyone. By AI artists everywhere.

Search prompts... Search

https://stablediffusionweb.com/prompts

Intro Diffusion Models

Questions?