# Denoising Diffusion Models: Fundamentals, Implementations and Applications

Mingfei Sun

Department of Computer Science

University of Manchester

4 June, 2025

# Disclaimer

- Slides adapted from:
  Denoising Diffusion Models: A Generative Learning Big Bang

- Some images are taken from the blog:
  What are Diffusion Models

# Heads-up: $n$ warning(s), 0 error(s)

# Heads-up: $n$ warning(s), 0 error(s)

- This tutorial is math-intensive, containing many equations and derivations.
- This tutorial focuses specifically on de-noising diffusion models (DDPM), with alterations in conditional generation.
- My knowledge on generative models is rather limited: may not answer all you questions, sorry!
- I tend to speak fast and skip many details: try to avoid it this time.

## Heads-up: $n$ warning(s), 0 error(s)

- This tutorial is math-intensive, containing many equations and derivations.
- This tutorial focuses specifically on de-noising diffusion models (DDPM), with alterations in conditional generation.
- My knowledge on generative models is rather limited: may not answer all you questions, sorry!
- I tend to speak fast and skip many details: try to avoid it this time.

# Heads-up: $n$ warning(s), 0 error(s)

- This tutorial is math-intensive, containing many equations and derivations.
- This tutorial focuses specifically on de-noising diffusion models (DDPM), with alterations in conditional generation.
- My knowledge on generative models is rather limited: may not answer all you questions, sorry!
- I tend to speak fast and skip many details: try to avoid it this time.

# Heads-up: $n$ warning(s), 0 error(s)

- This tutorial is math-intensive, containing many equations and derivations.
- This tutorial focuses specifically on de-noising diffusion models (DDPM), with alterations in conditional generation.
- My knowledge on generative models is rather limited: may not answer all you questions, sorry!
- I tend to speak fast and skip many details: try to avoid it this time.

# Outline

# Outline

# Random variables and notations

- A *random variable* (RV) is a *function* that assigns a number to the outcome of a random experiment.
- When referring to the probability $P(X = x)$, we usually simply write $P(x)$.
- Likewise, instead of writing $P(X = x, Y = y)$, we simply write $P(x, y)$.
- A continuous RV $X$ takes values within one or more intervals of the real line.

# Probability density function

- We use **probability density functions** (pdf), $p(x)$, to describe a continuous RV $X$.
- We can use a **joint probability density function**, $p(x, y)$ to fully characterise two continuous random variables $X$ and $Y$.
- Typical pdf in diffusion models

$$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2 \mathbf{I}) \text{ with } \boldsymbol{\mu} \text{ as the mean and } \sigma^2 \mathbf{I} \text{ as the covariance matrix}$$

  - Normal distribution (aka Gaussian distribution) for scalar RV $x$:

$$\mathcal{N}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

  - Multivariate normal distribution for vector RV $\boldsymbol{x}$:

$$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \Sigma^{-1} (\boldsymbol{x}-\boldsymbol{\mu})\right)$$

- For continuous RVs, expectation and variance are defined as

$$\mu = \mathbb{E}[X] = \int_{-\infty}^{\infty} x p(x) \mathrm{d}x,$$

$$\sigma^2 = \mathrm{var}[X] = \mathbb{E}[(X-\mu)^2] = \int_{-\infty}^{\infty} (x-\mu)^2 p(x) \mathrm{d}x$$

# Rules of probability (continuous RVs) and Markov property

- **Sum rule of probability**. In the case of continuous RVs, we replace the sums we had before with an integral

$$p(x) = \int_{-\infty}^{\infty} p(x, y) \mathrm{d}y,$$

where $p(x)$ is known as the **marginal pdf**.

- **Product rule of probability**. The conditional pdf can be obtained as

$$p(x|y) = \frac{p(x, y)}{p(y)},$$

which can also be written as $p(x, y) = p(x|y)p(y)$.

- **Bayes' theorem** follows as

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)},$$

which can be verified through $p(x, y) = p(y|x)p(x)$ and $p(x, y) = p(x|y)p(y)$.

- **Markov property**. For all $x_{t-1}, x_t, x_{t+1}$:

$$p(x_{t+1}|x_t) = p(x_{t+1}|x_t, x_{t-1}, \ldots, x_0),$$

i.e., $x_t$ is a sufficient statistic for the history $(x_0, x_1, \ldots, x_t)$.

# Rules of probability (continuous RVs) and Markov property

- **Sum rule of probability**. In the case of continuous RVs, we replace the sums we had before with an integral

$$p(x) = \int_{-\infty}^{\infty} p(x, y) \mathrm{d}y,$$

where $p(x)$ is known as the **marginal pdf**.

- **Product rule of probability**. The conditional pdf can be obtained as

$$p(x|y) = \frac{p(x, y)}{p(y)},$$

which can also be written as $p(x, y) = p(x|y)p(y)$.

- **Bayes' theorem** follows as

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)},$$

which can be verified through $p(x, y) = p(y|x)p(x)$ and $p(x, y) = p(x|y)p(y)$.

- **Markov property**. For all $x_{t-1}, x_t, x_{t+1}$:

$$p(x_{t+1}|x_t) = p(x_{t+1}|x_t, x_{t-1}, \ldots, x_0),$$

i.e., $x_t$ is a sufficient statistic for the history $(x_0, x_1, \ldots, x_t)$.

# Rules of probability (continuous RVs) and Markov property

- **Sum rule of probability**. In the case of continuous RVs, we replace the sums we had before with an integral

$$p(x) = \int_{-\infty}^{\infty} p(x, y) \mathrm{d}y,$$

where $p(x)$ is known as the **marginal pdf**.

- **Product rule of probability**. The conditional pdf can be obtained as

$$p(x|y) = \frac{p(x, y)}{p(y)},$$

which can also be written as $p(x, y) = p(x|y)p(y)$.

- **Bayes' theorem** follows as

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)},$$

which can be verified through $p(x, y) = p(y|x)p(x)$ and $p(x, y) = p(x|y)p(y)$.

- **Markov property**. For all $x_{t-1}, x_t, x_{t+1}$:

$$p(x_{t+1}|x_t) = p(x_{t+1}|x_t, x_{t-1}, \ldots, x_0),$$

i.e., $x_t$ is a sufficient statistic for the history $(x_0, x_1, \ldots, x_t)$.

# Rules of probability (continuous RVs) and Markov property

- **Sum rule of probability**. In the case of continuous RVs, we replace the sums we had before with an integral

$$p(x) = \int_{-\infty}^{\infty} p(x, y) \mathrm{d}y,$$

where $p(x)$ is known as the **marginal pdf**.

- **Product rule of probability**. The conditional pdf can be obtained as

$$p(x|y) = \frac{p(x, y)}{p(y)},$$

which can also be written as $p(x, y) = p(x|y)p(y)$.

- **Bayes' theorem** follows as

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)},$$

which can be verified through $p(x, y) = p(y|x)p(x)$ and $p(x, y) = p(x|y)p(y)$.

- **Markov property**. For all $x_{t-1}, x_t, x_{t+1}$:

$$p(x_{t+1}|x_t) = p(x_{t+1}|x_t, x_{t-1}, \ldots, x_0),$$

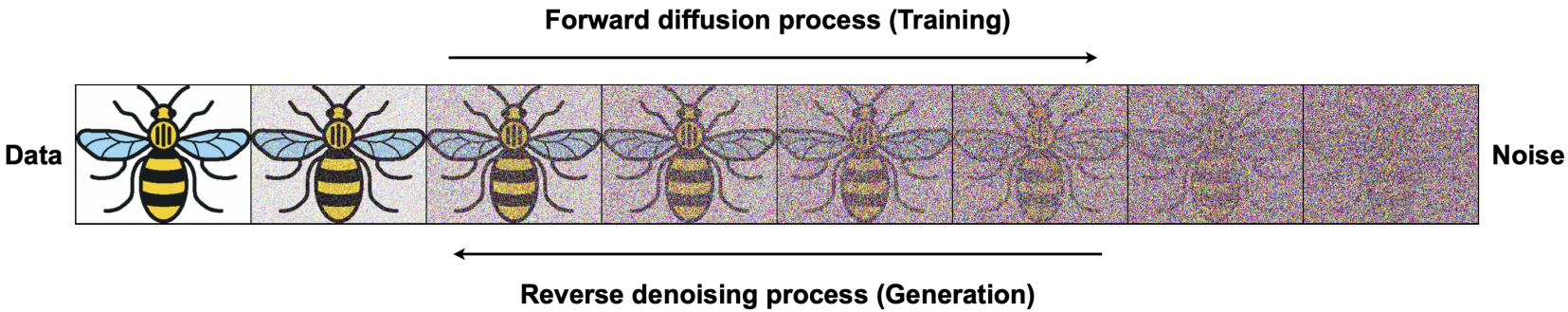
i.e., $x_t$ is a sufficient statistic for the history $(x_0, x_1, \ldots, x_t)$.

# Outline

# Denoising Diffusion Models

Denoising Diffusion models consist of two processes:

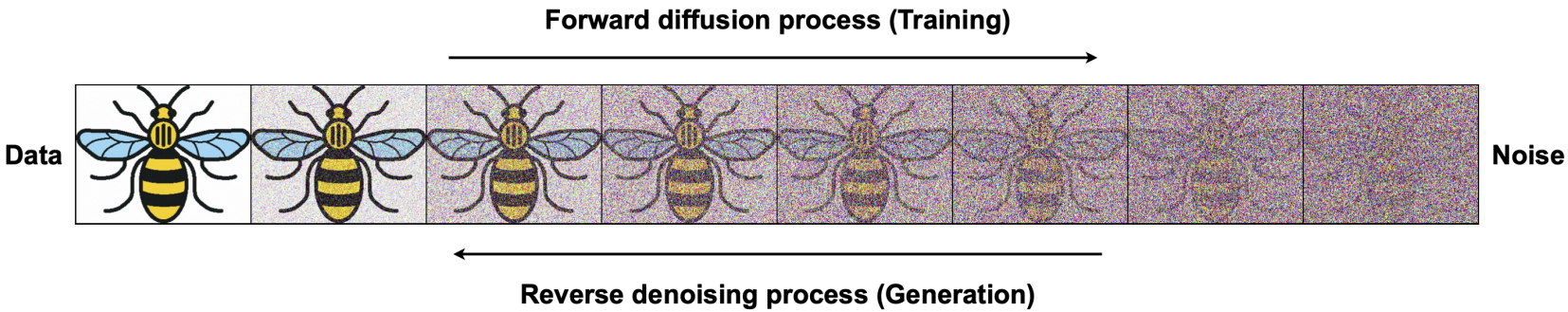
- Forward diffusion process that gradually adds noise to input
- Reverse denoising process that learns to generate data by denoising



# Denoising Diffusion Models

Denoising Diffusion models consist of two processes:

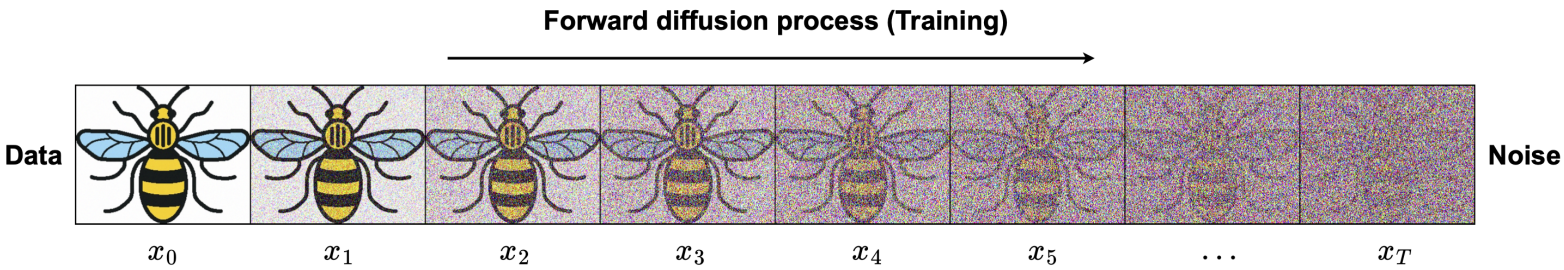
- Forward diffusion process that gradually adds noise to input
- Reverse denoising process that learns to generate data by denoising



# Forward Diffusion Models

The formal definition of the forward process in $T$ steps:

- Add small amount of Gaussian noise to the sample in $T$ steps, producing a sequence of noisy samples $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T$.




- The step sizes are controlled by a variance schedule $\{\beta_t \in (0,1)\}_{t=1}^{T}$.

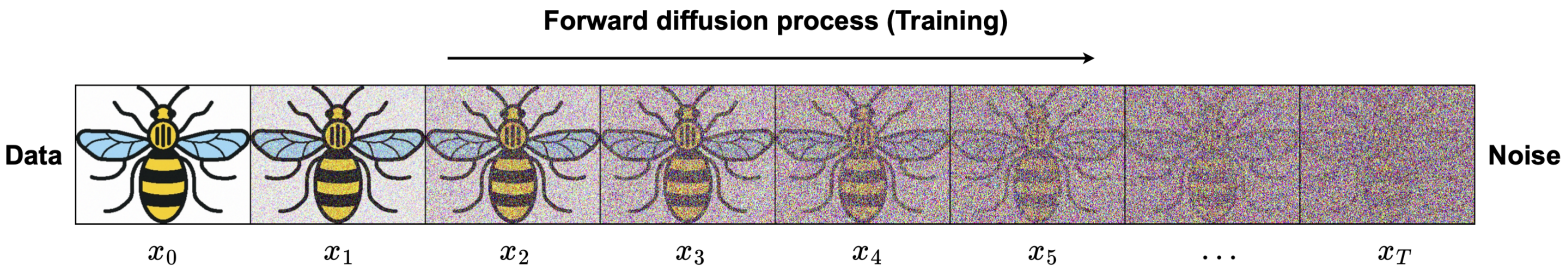
$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \underbrace{\mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t\mathbf{I})}_{\boldsymbol{x}_t \sim \mathcal{N}(\mu, \sigma^2) \text{ with } \mu = \sqrt{1-\beta_t}\boldsymbol{x}_{t-1} \text{ and } \sigma^2\mathbf{I} = \beta_t\mathbf{I}}$$

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

# Forward Diffusion Models

The formal definition of the forward process in $T$ steps:

- Add small amount of Gaussian noise to the sample in $T$ steps, producing a sequence of noisy samples $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T$.




- The step sizes are controlled by a variance schedule $\{\beta_t \in (0,1)\}_{t=1}^{T}$.

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \underbrace{\mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t\mathbf{I})}_{\boldsymbol{x}_t \sim \mathcal{N}(\mu, \sigma^2) \text{ with } \mu = \sqrt{1-\beta_t}\boldsymbol{x}_{t-1} \text{ and } \sigma^2\mathbf{I} = \beta_t\mathbf{I}}$$

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

# Diffusion Kernel

A nice property of the forward process is that we can sample $\boldsymbol{x}_t$ at any arbitrary step $t$ in a closed form. Define $\alpha_t \triangleq 1 - \beta_t$ and $\bar{\alpha}_t \triangleq \prod_{i=1}^{t} \alpha_i$.

According to $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t\mathbf{I})$:

$$
\begin{aligned}
\boldsymbol{x}_t &= \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon} && ;\ \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \\
&= \sqrt{\alpha_t}\underbrace{\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}\right)}_{\boldsymbol{x}_{t-1}} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon} && ;\ \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\boldsymbol{\epsilon} && ;\ \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \\
&= \dots \\
&= \sqrt{\alpha_t \dots \alpha_1}\boldsymbol{x}_0 + \sqrt{1-\alpha_t \dots \alpha_1}\boldsymbol{\epsilon} \\
&= \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} && ;\ \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})
\end{aligned}
$$

**Diffusion Kernel:** $\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$

Or equivalently,

**Diffusion Kernel:** $q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$

# Diffusion Kernel

A nice property of the forward process is that we can sample $\boldsymbol{x}_t$ at any arbitrary step $t$ in a closed form. Define $\alpha_t \triangleq 1 - \beta_t$ and $\bar{\alpha}_t \triangleq \prod_{i=1}^{t} \alpha_i$.

According to $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t\mathbf{I})$:

$$
\begin{aligned}
\boldsymbol{x}_t &= \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon} && ; \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \\
&= \sqrt{\alpha_t}\underbrace{\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}\right)}_{\boldsymbol{x}_{t-1}} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon} && ; \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\boldsymbol{\epsilon} && ; \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \\
&= \dots \\
&= \sqrt{\alpha_t \dots \alpha_1}\boldsymbol{x}_0 + \sqrt{1-\alpha_t \dots \alpha_1}\boldsymbol{\epsilon} \\
&= \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} && ; \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})
\end{aligned}
$$

**Diffusion Kernel:** $\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$

Or equivalently,

**Diffusion Kernel:** $q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$

# Diffusion Kernel

A nice property of the forward process is that we can sample $\boldsymbol{x}_t$ at any arbitrary step $t$ in a closed form. Define $\alpha_t \triangleq 1 - \beta_t$ and $\bar{\alpha}_t \triangleq \prod_{i=1}^{t} \alpha_i$.

According to $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t\mathbf{I})$:

$$\begin{aligned}
\boldsymbol{x}_t &= \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon} && ; \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \\
&= \sqrt{\alpha_t}\underbrace{\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}\right)}_{\boldsymbol{x}_{t-1}} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon} && ; \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\boldsymbol{\epsilon} && ; \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \\
&= \dots \\
&= \sqrt{\alpha_t \dots \alpha_1}\boldsymbol{x}_0 + \sqrt{1-\alpha_t \dots \alpha_1}\boldsymbol{\epsilon} \\
&= \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} && ; \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})
\end{aligned}$$

**Diffusion Kernel:** $\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$

Or equivalently,

**Diffusion Kernel:** $q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$

# Diffusion Kernel

A nice property of the forward process is that we can sample $\boldsymbol{x}_t$ at any arbitrary step $t$ in a closed form. Define $\alpha_t \triangleq 1 - \beta_t$ and $\bar{\alpha}_t \triangleq \prod_{i=1}^{t} \alpha_i$.

According to $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t\mathbf{I})$:

$$
\begin{aligned}
\boldsymbol{x}_t &= \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon} && ; \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \\
&= \sqrt{\alpha_t}\underbrace{\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}\right)}_{\boldsymbol{x}_{t-1}} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon} && ; \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\boldsymbol{\epsilon} && ; \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \\
&= \dots \\
&= \sqrt{\alpha_t \dots \alpha_1}\boldsymbol{x}_0 + \sqrt{1-\alpha_t \dots \alpha_1}\boldsymbol{\epsilon} \\
&= \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} && ; \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})
\end{aligned}
$$

**Diffusion Kernel:** $\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$

Or equivalently,

**Diffusion Kernel:** $q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$

# Diffusion Kernel

A nice property of the forward process is that we can sample $\boldsymbol{x}_t$ at any arbitrary step $t$ in a closed form. Define $\alpha_t \triangleq 1 - \beta_t$ and $\bar{\alpha}_t \triangleq \prod_{i=1}^{t} \alpha_i$.

According to $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t\mathbf{I})$:

$$
\begin{aligned}
\boldsymbol{x}_t &= \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon} && ; \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \\
&= \sqrt{\alpha_t}\underbrace{\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}\right)}_{\boldsymbol{x}_{t-1}} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon} && ; \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\boldsymbol{\epsilon} && ; \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \\
&= \ldots \\
&= \sqrt{\alpha_t \ldots \alpha_1}\boldsymbol{x}_0 + \sqrt{1-\alpha_t \ldots \alpha_1}\boldsymbol{\epsilon} \\
&= \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} && ; \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})
\end{aligned}
$$

**Diffusion Kernel:** $\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$

Or equivalently,

**Diffusion Kernel:** $q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$

# Diffusion Kernel

A nice property of the forward process is that we can sample $\boldsymbol{x}_t$ at any arbitrary step $t$ in a closed form. Define $\alpha_t \triangleq 1 - \beta_t$ and $\bar{\alpha}_t \triangleq \prod_{i=1}^{t} \alpha_i$.

According to $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t\mathbf{I})$:

$$
\begin{aligned}
\boldsymbol{x}_t &= \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon} && ; \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \\
&= \sqrt{\alpha_t}\underbrace{\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}\right)}_{\boldsymbol{x}_{t-1}} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon} && ; \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\boldsymbol{\epsilon} && ; \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \\
&= \dots \\
&= \sqrt{\alpha_t \dots \alpha_1}\boldsymbol{x}_0 + \sqrt{1-\alpha_t \dots \alpha_1}\boldsymbol{\epsilon} \\
&= \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} && ; \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})
\end{aligned}
$$

**Diffusion Kernel:** $x_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$

Or equivalently,

**Diffusion Kernel:** $q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$

# Diffusion Kernel

A nice property of the forward process is that we can sample $\boldsymbol{x}_t$ at any arbitrary step $t$ in a closed form. Define $\alpha_t \triangleq 1 - \beta_t$ and $\bar{\alpha}_t \triangleq \prod_{i=1}^{t} \alpha_i$.

According to $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t\mathbf{I})$:

$$
\begin{aligned}
\boldsymbol{x}_t &= \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon} && ;\ \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \\
&= \sqrt{\alpha_t}\underbrace{\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}\right)}_{\boldsymbol{x}_{t-1}} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon} && ;\ \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\boldsymbol{\epsilon} && ;\ \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \\
&= \dots \\
&= \sqrt{\alpha_t \dots \alpha_1}\boldsymbol{x}_0 + \sqrt{1-\alpha_t \dots \alpha_1}\boldsymbol{\epsilon} \\
&= \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} && ;\ \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})
\end{aligned}
$$

**Diffusion Kernel:** $x_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$

Or equivalently,

**Diffusion Kernel:** $q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$

# Diffusion Kernel

A nice property of the forward process is that we can sample $\boldsymbol{x}_t$ at any arbitrary step $t$ in a closed form. Define $\alpha_t \triangleq 1 - \beta_t$ and $\bar{\alpha}_t \triangleq \prod_{i=1}^{t} \alpha_i$.

According to $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t\mathbf{I})$:

$$
\begin{aligned}
\boldsymbol{x}_t &= \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon} && ;\ \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \\
&= \sqrt{\alpha_t}\underbrace{\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}\right)}_{\boldsymbol{x}_{t-1}} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon} && ;\ \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\boldsymbol{\epsilon} && ;\ \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1}) \\
&= \dots \\
&= \sqrt{\alpha_t \dots \alpha_1}\boldsymbol{x}_0 + \sqrt{1-\alpha_t \dots \alpha_1}\boldsymbol{\epsilon} \\
&= \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} && ;\ \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})
\end{aligned}
$$

**Diffusion Kernel:** $\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$

Or equivalently,

**Diffusion Kernel:** $q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$

# What happens to a distribution in the forward process

For forward process:

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon},$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\beta_t$ (a.k.a the variance scheduler) is designed such that $\bar{\alpha}_T \to 0$ and $q(\boldsymbol{x}_T|\boldsymbol{x}_0) \approx \mathcal{N}(\boldsymbol{x}_T; \mathbf{0}, \mathbf{I})$.

So far, we discussed the diffusion kernel $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$ but what about $q(\boldsymbol{x}_t)$?

$$\underbrace{q(\boldsymbol{x}_t)}_{\substack{\text{Diffused} \\ \text{data distribution}}} = \int \underbrace{q(\boldsymbol{x}_0, \boldsymbol{x}_t)}_{\substack{\text{Joint} \\ \text{distribution}}} d\boldsymbol{x}_0 = \int \underbrace{q(\boldsymbol{x}_0)}_{\substack{\text{Input data} \\ \text{distribution}}} \underbrace{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}_{\substack{\text{Diffusion} \\ \text{kernel}}} d\boldsymbol{x}_0$$

We can sample $\boldsymbol{x}_t \sim q(\boldsymbol{x}_t)$ by first sampling $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$ and then sampling $\boldsymbol{x}_t \sim q(\boldsymbol{x}_t|\boldsymbol{x}_0)$ (i.e., ancestral sampling).

# What happens to a distribution in the forward process

For forward process:

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon},$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\beta_t$ (a.k.a the variance scheduler) is designed such that $\bar{\alpha}_T \to 0$ and $q(\boldsymbol{x}_T|\boldsymbol{x}_0) \approx \mathcal{N}(\boldsymbol{x}_T; \mathbf{0}, \mathbf{I})$.

So far, we discussed the diffusion kernel $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$ but what about $q(\boldsymbol{x}_t)$?

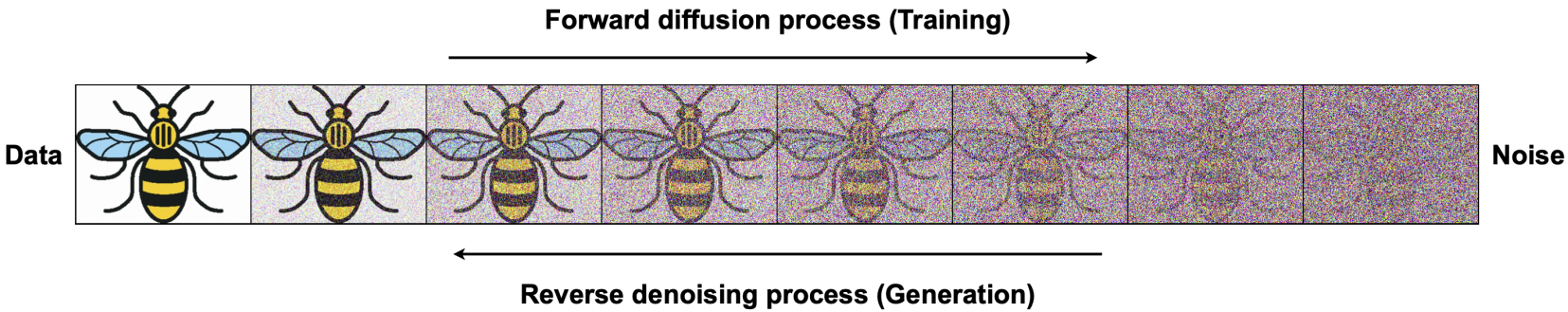
$$\underbrace{q(\boldsymbol{x}_t)}_{\substack{\text{Diffused} \\ \text{data distribution}}} = \int \underbrace{q(\boldsymbol{x}_0, \boldsymbol{x}_t)}_{\substack{\text{Joint} \\ \text{distribution}}} d\boldsymbol{x}_0 = \int \underbrace{q(\boldsymbol{x}_0)}_{\substack{\text{Input data} \\ \text{distribution}}} \underbrace{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}_{\substack{\text{Diffusion} \\ \text{kernel}}} d\boldsymbol{x}_0$$

We can sample $\boldsymbol{x}_t \sim q(\boldsymbol{x}_t)$ by first sampling $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$ and then sampling $\boldsymbol{x}_t \sim q(\boldsymbol{x}_t|\boldsymbol{x}_0)$ (i.e., ancestral sampling).

# Denoising Diffusion Models

Denoising Diffusion models consist of two processes:

- Forward diffusion process that gradually adds noise to input
- Reverse denoising process that learns to generate data by denoising



# Generative Learning by Denoising

Heads-up: we use $q$ for the forward process and $p$ for the reverse process.

Recall, that the denoising process is designed such that $p(\boldsymbol{x}_T) \approx \mathcal{N}(\boldsymbol{x}_T; \mathbf{0}, \mathbf{I})$

**Generation**:

- Sample $\boldsymbol{x}_T \sim \mathcal{N}(\boldsymbol{x}_T; \mathbf{0}, \mathbf{I})$
- Iteratively sample $\boldsymbol{x}_{t-1} \sim \underbrace{p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)}_{\text{True denoising distribution}}$

Can we parameterize $p(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$? Yes, we can use a Normal distribution if $\beta_t$ is small in each forward diffusion step. [1]

[1] For Gaussian diffusion, for continuous diffusion (limit of small step size $\beta$), the reversal of the diffusion process has the identical functional form as the forward process. Since $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ is a Gaussian distribution, and if $\beta_t$ is small, then $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ will also be a Gaussian distribution.

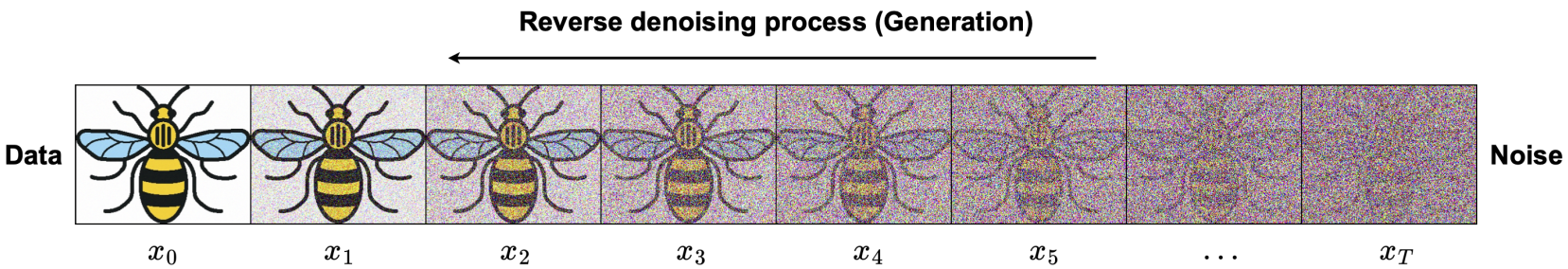
# Reverse Denoising Process

Formal definition of reverse processes in $T$ steps:

$$p(\boldsymbol{x}_T) = \mathcal{N}(\boldsymbol{x}_T; \mathbf{0}, \mathbf{I})$$

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}; \underbrace{\mu_\theta(\boldsymbol{x}_t, t)}_{\text{Trainable network (U-net, Denoising autoencoder)}}, \sigma_t^2 \mathbf{I})$$

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



# Learning Denoising Model: Variational upper bound

For training, we can form variational upper bound

$$\nabla_\theta \mathcal{L} = \nabla_\theta \mathbb{E}_{q(\boldsymbol{x}_0) q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \left[ \sum_{t=1}^{T} \log \frac{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)}{p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)} \right]$$

$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ follows Gaussian distribution $\mathcal{N}\left(\tilde{\mu}(\boldsymbol{x}_t, \boldsymbol{x}_0), \tilde{\beta}_t \mathbf{I}\right)$, where $\tilde{\mu}(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{1}{\sqrt{\alpha_t}} \left( \boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon} \right)$.
Reverse transition $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}; \mu_\theta(\boldsymbol{x}_t, t), \sigma_t^2 \mathbf{I})$.

Since $\boldsymbol{x}_t$ is available as input to the model, we may choose the parameterization

$$\boxed{\mu_\theta(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}} \left( \boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t) \right)}$$

Using a few simple arithmetic operations and $\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$, we can write down the variational objective for two Gaussian distributions as:

$$\min_\theta \mathbb{E}_{\underbrace{\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)}_{\text{data distribution}}, \underbrace{t \sim \mathcal{U}\{1, T\}}_{\text{Diffusion steps}}, \underbrace{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}_{\text{Gaussian noises}}} \left[ \lambda_t \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta \left( \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t \right) \right\|^2 \right]$$

Ho et al. [2] observe that simply setting $\lambda_t$ to 1 for all $t$ works best in practice.

[2] Denoising diffusion probabilistic models, Ho et al, 2020

# Learning Denoising Model: Variational upper bound

For training, we can form variational upper bound

$$\nabla_\theta \mathcal{L} = \nabla_\theta \mathbb{E}_{q(\boldsymbol{x}_0) q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)} \left[ \sum_{t=1}^{T} \log \frac{q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)}{p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)} \right]$$

$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0)$ follows Gaussian distribution $\mathcal{N}\left(\tilde{\mu}(\boldsymbol{x}_t, \boldsymbol{x}_0), \tilde{\beta}_t \mathbf{I}\right)$, where
$\tilde{\mu}(\boldsymbol{x}_t, \boldsymbol{x}_0) = \frac{1}{\sqrt{\alpha_t}} \left( \boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon} \right)$.
Reverse transition $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}; \mu_\theta(\boldsymbol{x}_t, t), \sigma_t^2 \mathbf{I})$.

Since $\boldsymbol{x}_t$ is available as input to the model, we may choose the parameterization

$$\boxed{\mu_\theta(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}} \left( \boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t) \right)}$$

Using a few simple arithmetic operations and $\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$, we can write down the variational objective for two Gaussian distributions as:

$$\min_\theta \mathbb{E}_{\underbrace{\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)}_{\text{data distribution}}, \underbrace{t \sim \mathcal{U}\{1, T\}}_{\text{Diffusion steps}}, \underbrace{\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}_{\text{Gaussian noises}}} \left[ \lambda_t \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta \left( \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t \right) \right\|^2 \right]$$

Ho et al. [2] observe that simply setting $\lambda_t$ to 1 for all $t$ works best in practice.

[2] Denoising diffusion probabilistic models, Ho et al, 2020

# Generating New Samples

Denoising steps:

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}; \mu_\theta(\boldsymbol{x}_t, t), \sigma_t^2 \mathbf{I})$$

and

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

Note that $\boldsymbol{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, a new sample $\boldsymbol{x}_0$ is generated by recursively applying

$$\begin{aligned} \boldsymbol{x}_{t-1} &= \mu_\theta(\boldsymbol{x}_t, t) + \sigma_t \boldsymbol{z} && ; \boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \forall t = T, ..., 1 \\ &= \frac{1}{\sqrt{\alpha_t}} \left( \boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t) \right) + \sigma_t \boldsymbol{z} && ; \boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \forall t = T, ..., 1 \end{aligned}$$

- Ho et al. [3] suggest to set $\sigma_t$ to $\sigma_t^2 = \beta_t$ or $\sigma_t^2 = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$, yielding similar experimental results.
- Nichol and Dhariwal [4] propose to learn an interpolation between between $\beta_t$ and $\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$

[3] Denoising diffusion probabilistic models, Ho et al, 2020

[4] Improved denoising diffusion probabilistic models, Alexander Quinn Nichol and Prafulla Dhariwal, 2021

# Combined: Training and Sampling

Practical algorithms for Denoising Diffuion Probabilistic Models (DDPM):

**Algorithm** Training

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

**Algorithm** Sampling

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

# Combined: Training and Sampling

Practical algorithms for Denoising Diffuion Probabilistic Models (DDPM):

**Algorithm** Training

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

**Algorithm** Sampling

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

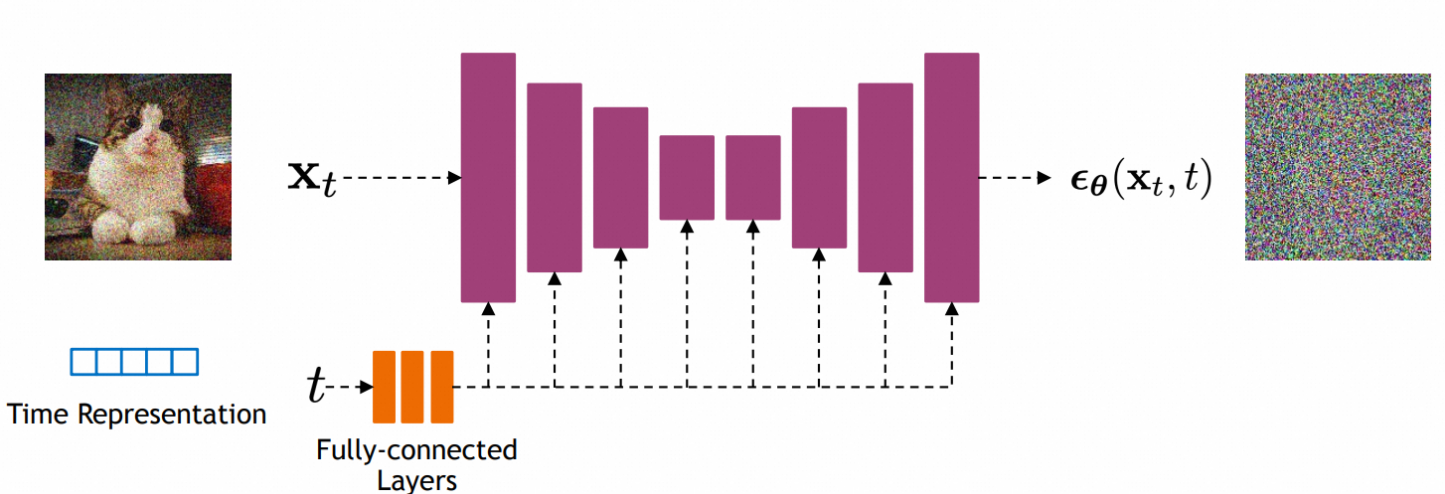
# Implementation considerations

**Algorithm** Training

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

**Network architectures for $\epsilon_\theta$:**

- Diffusion models often use U-Net architectures with ResNet blocks and self-attention layers to represent $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$
- Time representation: sinusoidal positional embeddings or random Fourier features



# Implementation considerations

**Algorithm** Training

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

**Algorithm** Sampling

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

**Noise and variance scheduler**:

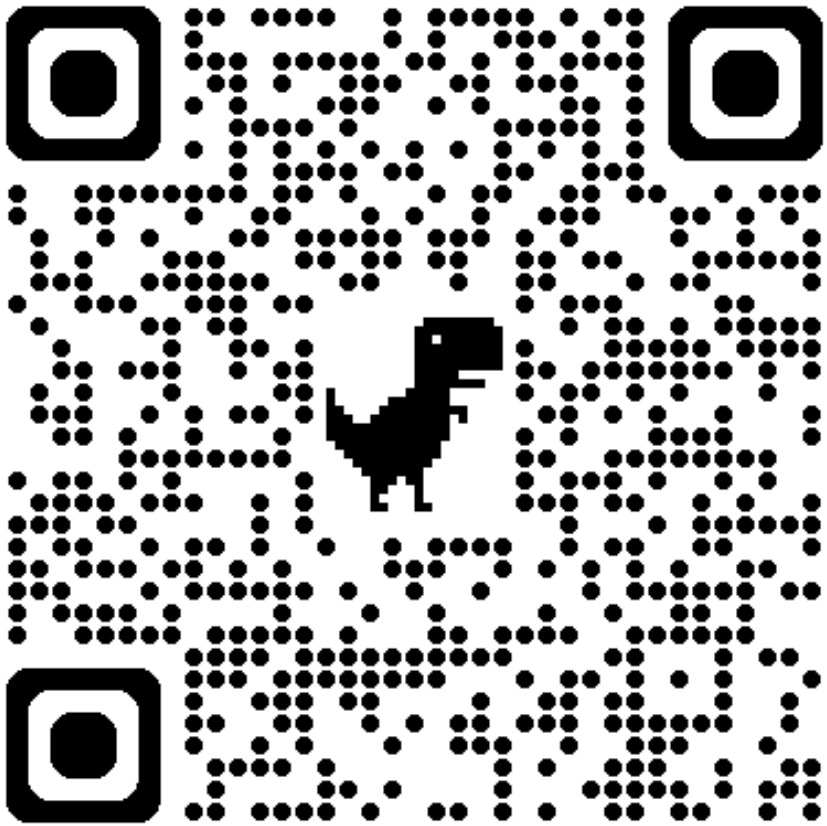
- Set $T = 1000$ and the forward process variances to constants increasing linearly from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$.
- $\alpha_t \triangleq 1 - \beta_t$ and $\bar{\alpha}_t \triangleq \prod_{i=1}^{t} \alpha_i$.
- Diagonal covariance matrix and $\sigma_t^2 = \beta_t$.

# Give me the code, please!

Talk is cheap, show me the code!!!

# Give me the code, please!

Educational implementations of DDPM, Guided DDPM, Conditional DDPM, and Classifier Free Diffusion, on MNIST dataset, i.e., generating digits



`https://github.com/mingfeisun/diffusion-models`

# Outline

# Applications beyond Image Generation

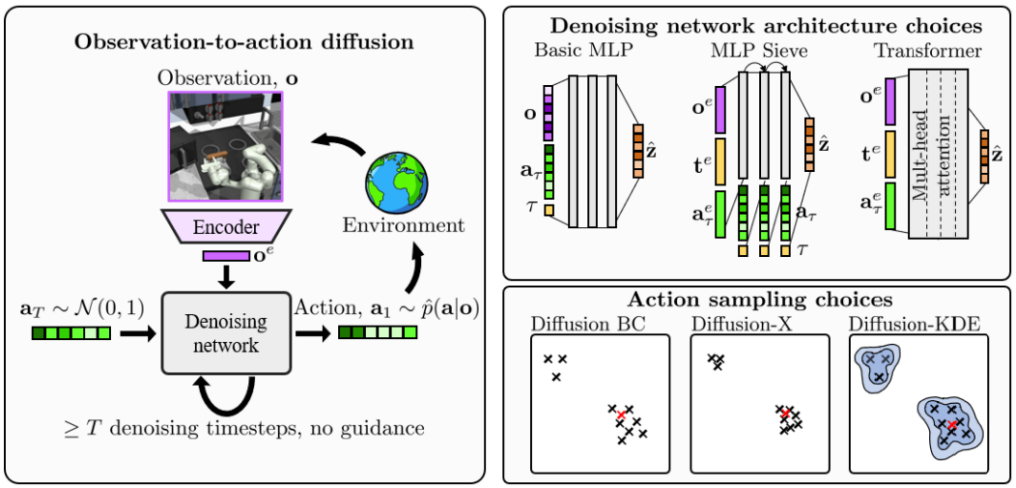
Diffusion models are good at modeling complex distributions

- Policies in reinforcement learning and imitation learning are conditional distributions:
  - Diffusion models as implicit parameterization of policies
  - Diffusion models to generate decision-making rules
  - Example: Imitating Human Behaviour with Diffusion Models, ICLR 2023
- Inference under constraints can be interpreted as constraints-guided sampling
  - Inference without constraints as generative modeling
  - Guide the sampling process with constraints-induced classifiers
  - Example: Effective Generation of Feasible Solutions for Integer Programming via Guided Diffusion, SIGKDD 2024

# Diffusion Models for Imitation Learning

## Explicit conditional diffusion models as policy representations[5]

- Policies as observation-to-action generative models




- Denoising Diffusion Probabilistic Models

$$\mathcal{L}_{\text{DDPM}} \triangleq \mathbb{E}_{o,a,t,z}\left[||\epsilon_\theta(o, a_t, t) - z||\right]$$

with $a_\tau = \sqrt{\bar{\alpha}_t}a + \sqrt{1-\bar{\alpha}_t}z$ for variance scheduler $\bar{\alpha}_t$, and denosing

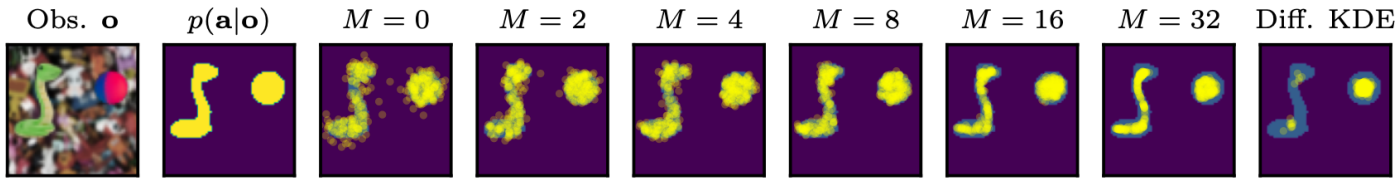
$$a_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(a_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\underbrace{\epsilon(o, a_t, t)}_{\text{Explicit conditioning}}\right) + \sigma_t z$$

[5] Imitating Human Behaviour with Diffusion Models, ICLR 2023

# Diffusion Models for Imitation Learning

**Explicit conditional diffusion models as policy representations**[6]

- "A bad action could be selected during a roll-out"
- Reliable sampling schemes:
  - **Diffusion-X**: extra denoising iterations for a sample
  - **Diffusion-KDE**: kernel-density estimator (KDE) fitting




- Imitating human behavior [Counter-Strike: Global Offensive (CSGO)]


[6] Imitating Human Behaviour with Diffusion Models, ICLR 2023

# Diffusion Models for Inference under Constraints

## Guided sampling as inference under constraints[7]

Integer Programming

$$\min_{\mathbf{x}} \mathbf{c}^\top \mathbf{x} \qquad \text{subject to } \mathbf{A}\mathbf{x} \leq \mathbf{b}, \qquad \mathbf{x} \in \mathbb{Z}^n$$

where $\mathbf{c} \in \mathbb{R}^n$ denotes the objective coefficient, $\mathbf{A} \in \mathbb{R}^{m \times n}$ is the coefficient matrix of constraints and $\mathbf{b} \in \mathbb{R}^m$ represents the right-hand-side vector.

- Solving large-scale Integer Programming needs an initial good guess of feasible solutions
- Introduce constraint guidance by designing each transition probability as

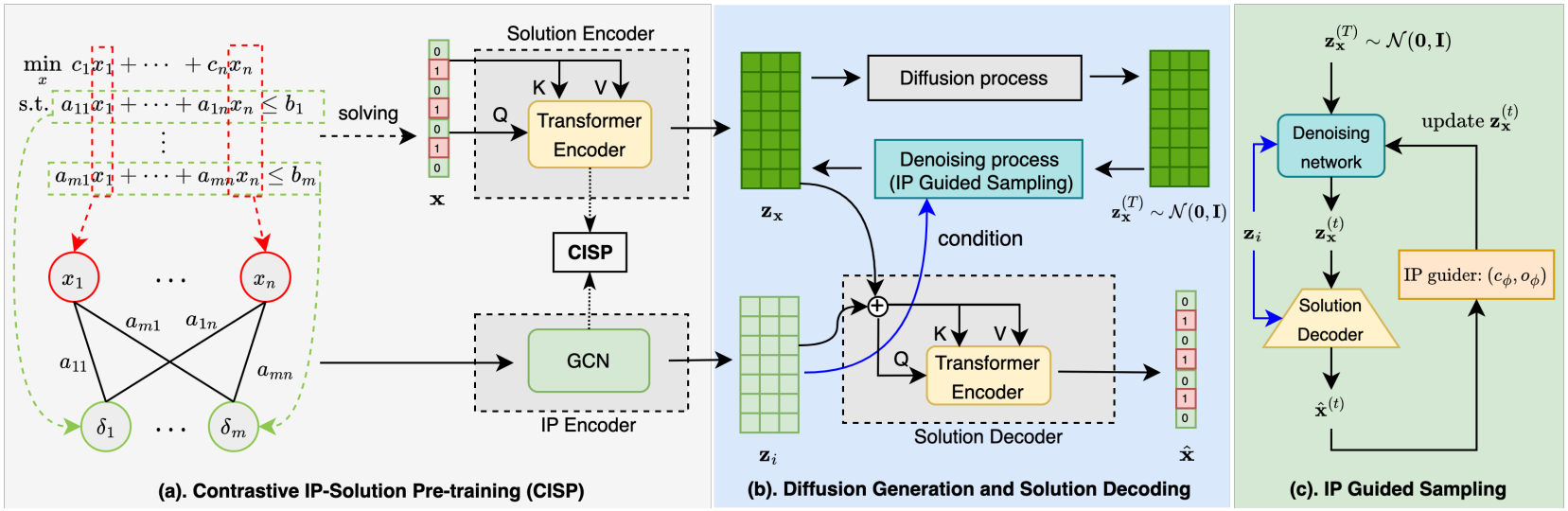
$$p_{\theta,\phi}(\mathbf{z}_{\mathbf{x}}^{(t)} | \mathbf{z}_{\mathbf{x}}^{(t+1)}, \mathbf{z}_i, \mathbf{A}, \mathbf{b}) = Z p_\theta(\mathbf{z}_{\mathbf{x}}^{(t)} | \mathbf{z}_{\mathbf{x}}^{(t+1)}, \mathbf{z}_i) e^{-s c_\phi(\mathbf{z}_{\mathbf{x}}^{(t)}, \mathbf{z}_i, \mathbf{A}, \mathbf{b})}, \qquad (1)$$

  where $c_\phi(\mathbf{z}_{\mathbf{x}}^{(t)}, \mathbf{z}_i, \mathbf{A}, \mathbf{b}) = \sum_{k=1}^m \max(\mathbf{a}_k^T \mathbf{d}_\phi(\mathbf{z}_{\mathbf{x}}^{(t)}, \mathbf{z}_i) - b_k, 0)$ measures the violation of constraints.
- Consider the Taylor expansion for $c_\phi$ at $\mathbf{z}_{\mathbf{x}}^{(t)} = \boldsymbol{\mu}$: $c_\phi(\mathbf{z}_{\mathbf{x}}^{(t)}, \mathbf{z}_i, \mathbf{A}, \mathbf{b}) = (\mathbf{z}_{\mathbf{x}}^{(t)} - \boldsymbol{\mu}) \nabla_{\mathbf{z}_{\mathbf{x}}^{(t)}} c_\phi(\mathbf{z}_{\mathbf{x}}^{(t)}, \mathbf{z}_i, \mathbf{A}, \mathbf{b})|_{\mathbf{z}_{\mathbf{x}}^{(t)} = \boldsymbol{\mu}} + C_1$. Hence, train a "classifer" to predict the constraint violation.

[7] Effective Generation of Feasible Solutions for Integer Programming via Guided Diffusion, SIGKDD 2024

# Diffusion Models for Integer Programming

## Guided sampling as inference under constraints[8]



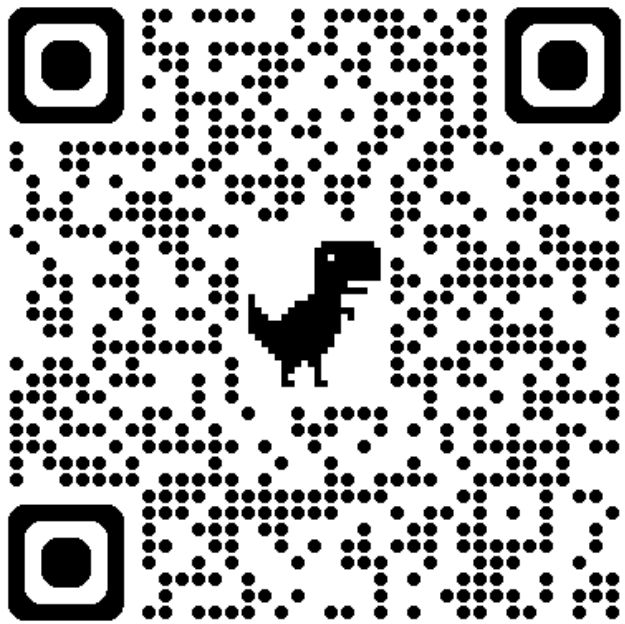

1. Trains IP Encoder and Solution Encoder to obtain embeddings.

2. Jointly train diffusion models and the solution decoder to capture solution distributions.

3. Guide diffusion sampling with objectives and constraints.

[8] Effective Generation of Feasible Solutions for Integer Programming via Guided Diffusion, SIGKDD 2024

# *Thank You*



https://github.com/mingfeisun/diffusion-models

mingfei.sun@manchester.ac.uk
https://mingfeisun.github.io/