Deep Learning for Computer Vision

Deep Generative Models: Introduction to Diffusion Models

Vineeth N Balasubramanian

Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

Homework Review

Question: Why is MI Gap and not MI used as a metric for disentanglement?

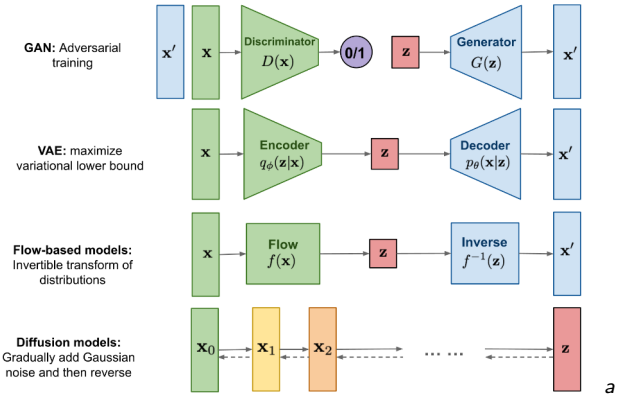
Homework Review

Question: Why is MI Gap and not MI used as a metric for disentanglement?

- MI Gap penalizes unaligned latent variables (contains information about more than one generative factor) $\rightarrow$ undesirable for disentanglement
- If one latent variable reliably models a generative factor, it is not required for other latent variables to be informative about this factor
- Read Chen et al, Isolating Sources of Disentanglement in Variational Autoencoders, NeurIPS 2018 for more information!

Recall: Deep Generative Models

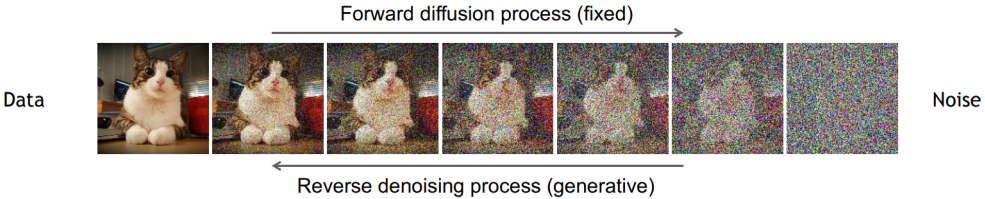
- Generative models aim to learn the underlying distribution of a dataset to generate new samples that are similar to the original data.
- Different approaches:
  - **Generative Adversarial Networks (GANs):** Competitive process between a generator and a discriminator
  - **Variational Autoencoders (VAEs):** Probabilistic approach
  - **Normalizing Flows:** Invertible transformations to model data distributions
  - **Diffusion models:** Generate data by reversing a gradual noising process



[a] *Image Credit: Lilian Weng: What are Diffusion Models? (blog)*

Diffusion Models: An Introduction

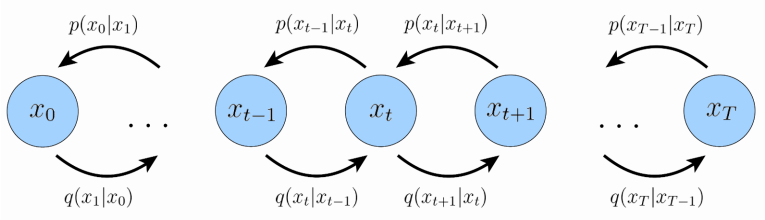
- Class of generative models inspired by physical process of diffusion, where particles spread out over time due to random motion
- Consist of:
  - a **forward process** that gradually adds noise to data, resulting in a distribution close to Gaussian
  - a **reverse process** that denoises to generate samples, starting from pure noise
- Models trained to accurately reverse the diffusion process, learning to recover the original data distribution from the noisy distribution



[1] *Image Credit: CVPR 2023 Tutorial on Diffusion Models*

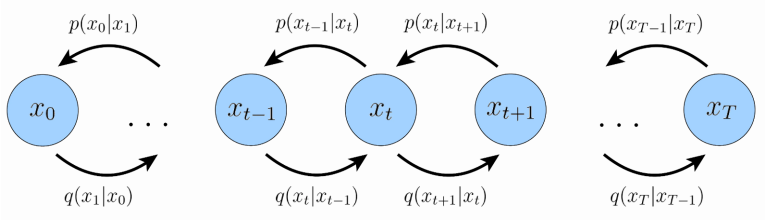
Denoising Diffusion Probabilistic Models (DDPMs)

- DDPMs: Specific type of diffusion model that uses a probabilistic approach for the reverse diffusion process, modeling it as a sequence of learned conditional distributions



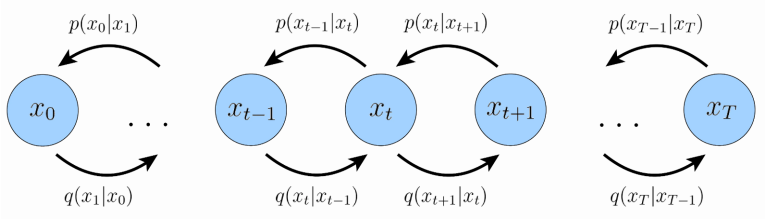
Denoising Diffusion Probabilistic Models (DDPMs)

- DDPMs: Specific type of diffusion model that uses a probabilistic approach for the reverse diffusion process, modeling it as a sequence of learned conditional distributions
- Characterized by a Markovian forward process, where each step adds a small amount of noise, and a learned reverse process parameterized by a neural network, which aims to denoise the data at each step



Denoising Diffusion Probabilistic Models (DDPMs)

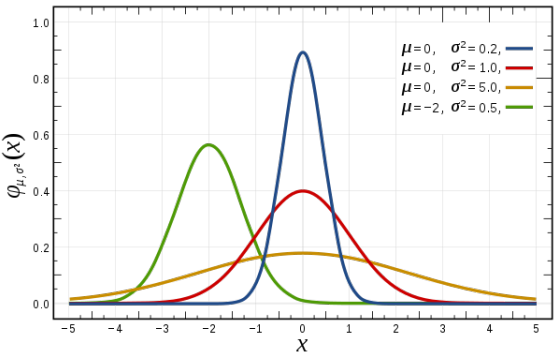
- DDPMs: Specific type of diffusion model that uses a probabilistic approach for the reverse diffusion process, modeling it as a sequence of learned conditional distributions
- Characterized by a Markovian forward process, where each step adds a small amount of noise, and a learned reverse process parameterized by a neural network, which aims to denoise the data at each step
- Shown impressive results in generating high-quality samples for various types of data, including images, audio, and molecular structures, demonstrating versatility and effectiveness in generative modeling



Preliminaries: Gaussian Distribution

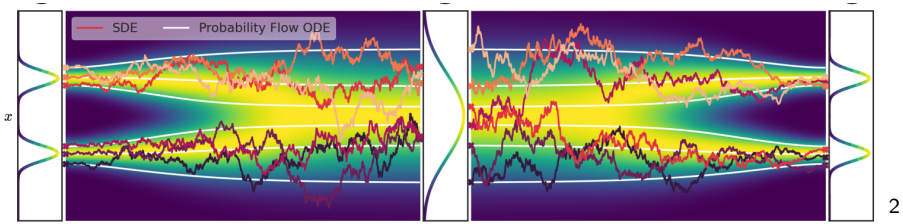
- Gaussian distribution, also known as normal distribution, is a probability distribution characterized by its bell-shaped curve
- Defined by two parameters: *mean* (μ) and *variance* (σ^2)
- Probability density function (PDF) of a Gaussian distribution:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$



Preliminaries: Stochastic Processes

- Collection of random variables $\{X_t\}$ indexed by time t, representing evolution of a system over time
- Each realization of a stochastic process is a potential path or trajectory that the system can follow
- Used to model phenomena with inherent randomness; e.g. stock prices, temperature fluctuations, population dynamics
- Key properties: Stationarity, Independence, Markov property



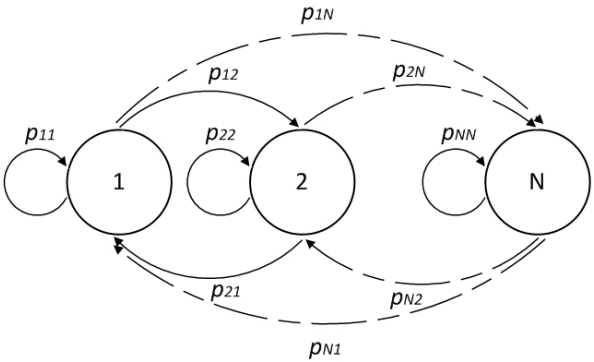
[2] *Image Credit: Song et al, Score-based Generative Modeling through Stochastic Differential Equations, ICLR 2021*

Preliminaries: Markov Chains

- **Markov chain:** A stochastic process with Markov property: future state depends only on current state
- Transition matrix P defines the probabilities of moving from one state to another:

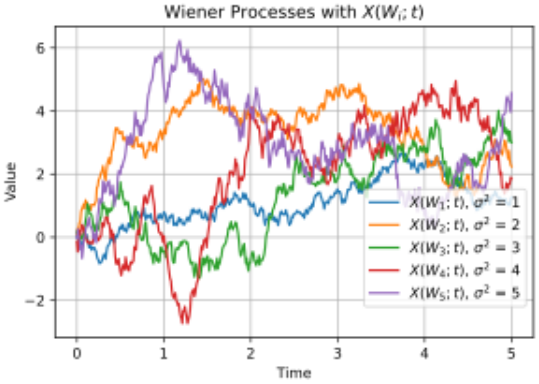
$$P = \begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1n} \\ P_{21} & P_{22} & \cdots & P_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ P_{n1} & P_{n2} & \cdots & P_{nn} \end{bmatrix},$$

where P_{ij} is probability of transitioning from state i to state j



Preliminaries: Wiener Process (Brownian Motion)

- Wiener process, also known as Brownian motion, is a continuous-time stochastic process with the following properties:
  - $W_0 = 0$ (process starts at zero)
  - $W_t - W_s \sim \mathcal{N}(0, t-s)$ for $0 \leq s < t$ (increments are normally distributed).
  - Increments independent for non-overlapping time intervals
  - Paths of process are continuous with probability 1
- Fundamental building block of many models in finance, physics, and engineering; e.g. Black-Scholes model for option pricing



Preliminaries: Stochastic Differential Equations (SDEs)

- SDEs used to model systems with random behavior, influenced by both deterministic and stochastic components
- General form of an SDE:

$$dX_t = \mu(X_t, t)dt + \sigma(X_t, t)dW_t$$

where:

  - X_t is the state variable at time t
  - $\mu(X_t, t)$ is the drift coefficient, representing deterministic trends
  - $\sigma(X_t, t)$ is the diffusion coefficient, representing random fluctuations
  - dW_t is an increment of a Wiener process (Brownian motion), modeling the random noise

- Used to describe the dynamics of systems where noise plays a significant role, such as in financial markets, physical processes, biological systems[3]

[3] *Song et al, Score-based Generative Modeling through Stochastic Differential Equations, ICLR 2021*

DDPMs: Theoretical Foundations

- DDPMs based on stochastic differential equations, which model systems with random perturbations

DDPMs: Theoretical Foundations

- DDPMs based on stochastic differential equations, which model systems with random perturbations
- **Forward diffusion process** adds noise to data in a Markov chain, gradually degrading the signal

DDPMs: Theoretical Foundations

- DDPMs based on stochastic differential equations, which model systems with random perturbations
- **Forward diffusion process** adds noise to data in a Markov chain, gradually degrading the signal
- **Reverse diffusion process** is a learned Markov chain that denoises the data, reconstructing the original signal from noise

DDPMs: Theoretical Foundations

- DDPMs based on stochastic differential equations, which model systems with random perturbations
- **Forward diffusion process** adds noise to data in a Markov chain, gradually degrading the signal
- **Reverse diffusion process** is a learned Markov chain that denoises the data, reconstructing the original signal from noise
- Training objective maximizes likelihood of reverse process generating the original data from noise, by minimizing discrepancy between original data and denoised samples

DDPMs: Theoretical Foundations

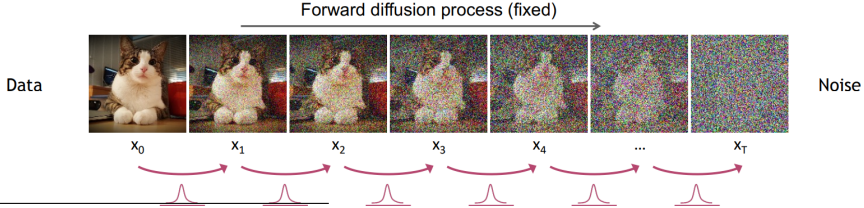
- DDPMs based on stochastic differential equations, which model systems with random perturbations
- **Forward diffusion process** adds noise to data in a Markov chain, gradually degrading the signal
- **Reverse diffusion process** is a learned Markov chain that denoises the data, reconstructing the original signal from noise
- Training objective maximizes likelihood of reverse process generating the original data from noise, by minimizing discrepancy between original data and denoised samples
- Success depends on design of forward and reverse processes, ensuring accurate capture and reconstruction of data distribution

Forward Diffusion Process

- Defined by a sequence of Gaussian transitions:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I}),$$

where β_t is the variance of the noise added at step t



[4] *Image Credit: CVPR 2023 Tutorial on Diffusion Models*

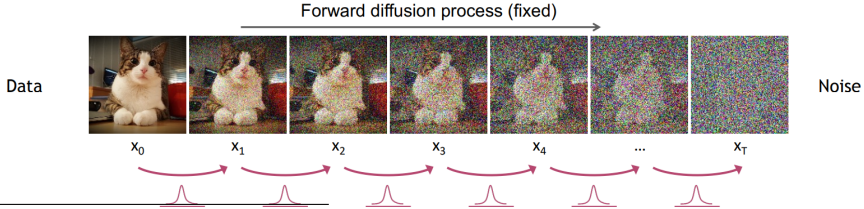
Forward Diffusion Process

- Defined by a sequence of Gaussian transitions:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I}),$$

  where β_t is the variance of the noise added at step t
- Joint distribution of forward process given by:

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}).$$



[4] *Image Credit: CVPR 2023 Tutorial on Diffusion Models*

Diffusion Kernel

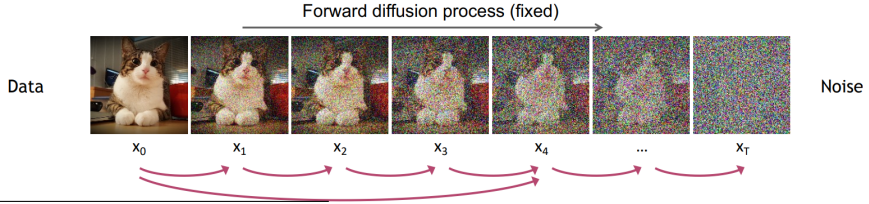
- Diffusion kernel is defined as:

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I})$$

  where $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$
- Sampling from the diffusion kernel given by:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I})$$

- As $t \to T$, $\bar{\alpha}_T \to 0$, leading to $x_T \sim \mathcal{N}(0, \mathbf{I})$

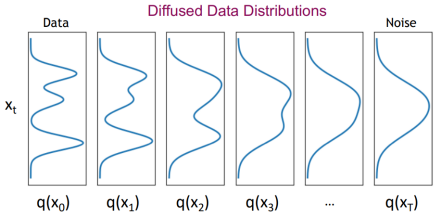


5

[5] *Image Credit: CVPR 2023 Tutorial on Diffusion Models*

Distribution Evolution in Forward Diffusion

- In forward diffusion, distribution of data evolves as noise is gradually added



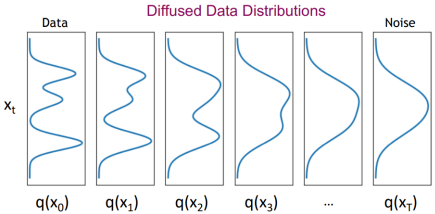
Distribution Evolution in Forward Diffusion

- In forward diffusion, distribution of data evolves as noise is gradually added
- Distribution at time t, $q(x_t)$, can be expressed as integral over the joint distribution $q(x_0, x_t)$:

$$q(x_t) = \int q(x_0, x_t) dx_0,$$

where joint distribution can be further decomposed as:

$$q(x_0, x_t) = q(x_t|x_0)q(x_0)$$



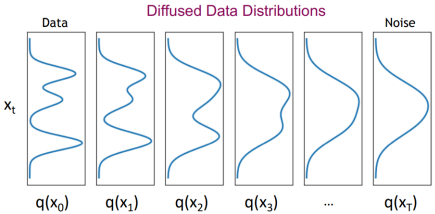
Distribution Evolution in Forward Diffusion

- In forward diffusion, distribution of data evolves as noise is gradually added
- Distribution at time t, $q(x_t)$, can be expressed as integral over the joint distribution $q(x_0, x_t)$:

$$q(x_t) = \int q(x_0, x_t) dx_0,$$

where joint distribution can be further decomposed as:

$$q(x_0, x_t) = q(x_t|x_0)q(x_0)$$



- This decomposition reflects the combination of input data distribution $q(x_0)$ and diffusion kernel $q(x_t|x_0)$

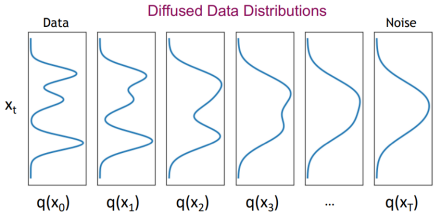
Distribution Evolution in Forward Diffusion

- In forward diffusion, distribution of data evolves as noise is gradually added
- Distribution at time t, $q(x_t)$, can be expressed as integral over the joint distribution $q(x_0, x_t)$:

$$q(x_t) = \int q(x_0, x_t) dx_0,$$

where joint distribution can be further decomposed as:

$$q(x_0, x_t) = q(x_t|x_0)q(x_0)$$



- This decomposition reflects the combination of input data distribution $q(x_0)$ and diffusion kernel $q(x_t|x_0)$
- **Ancestral sampling:** To sample from $q(x_t)$, we first sample $x_0 \sim p(x_0)$ and then apply the diffusion kernel to obtain $x_t \sim q(x_t|x_0)$

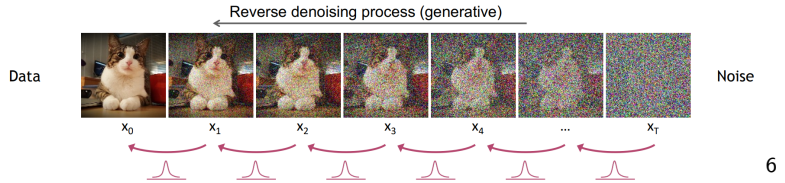
Reverse Diffusion Process

- Reverse diffusion process is defined as:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

  where μ_θ and Σ_θ are learned functions parameterized by θ
- The reverse process reconstructs x_{t-1} from x_t by denoising, guided by the learned parameters
- Joint distribution of the reverse process is given by:

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$



6

[6] *Image Credit: CVPR 2023 Tutorial on Diffusion Models*

Denoising Function in Reverse Diffusion

- Denoising function $\mu_\theta(x_t, t)$ is key in DDPMs. It guides the reverse diffusion process
- It is defined as:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{1-\beta_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right)$$

where $\epsilon_\theta(x_t, t)$ is the neural network's output. This neural network predicts the noise ϵ added during the forward process.

Denoising Function in Reverse Diffusion

- Denoising function $\mu_\theta(x_t, t)$ is key in DDPMs. It guides the reverse diffusion process
- It is defined as:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{1-\beta_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right)$$

  where $\epsilon_\theta(x_t, t)$ is the neural network's output. This neural network predicts the noise ϵ added during the forward process.
- Denoising function aims to reconstruct the original data from noisy observations
- At each step, it estimates the added noise and subtracts it
- This process moves from a noisy observation x_t towards the clean data x_0
- Accurate noise prediction is crucial for high-quality data reconstruction

Derivation of Denoising Function μ_θ (1/2)

- Start with the forward diffusion process:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$$

where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$

Derivation of Denoising Function μ_θ (1/2)

- Start with the forward diffusion process:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$$

  where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$

- Rearrange to express x_0 in terms of x_t and ϵ:

$$x_0 = \frac{1}{\sqrt{\bar{\alpha}_t}} x_t - \frac{\sqrt{1 - \bar{\alpha}_t}}{\sqrt{\bar{\alpha}_t}} \epsilon$$

Derivation of Denoising Function μ_θ (1/2)

- Start with the forward diffusion process:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$$

where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$

- Rearrange to express x_0 in terms of x_t and ϵ:

$$x_0 = \frac{1}{\sqrt{\bar{\alpha}_t}} x_t - \frac{\sqrt{1 - \bar{\alpha}_t}}{\sqrt{\bar{\alpha}_t}} \epsilon$$

- For the reverse process, we aim to predict x_{t-1} from x_t. The forward process for x_{t-1} is given by:

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}} x_0 + \sqrt{1 - \bar{\alpha}_{t-1}} \epsilon'$$

where $\epsilon' \sim \mathcal{N}(0, \mathbf{I})$

Derivation of Denoising Function μ_θ (2/2)

- Substitute the expression for x_0 into the equation for x_{t-1}:

$$\begin{aligned} x_{t-1} &= \sqrt{\bar{\alpha}_{t-1}}\left(\frac{1}{\sqrt{\bar{\alpha}_t}}x_t - \frac{\sqrt{1-\bar{\alpha}_t}}{\sqrt{\bar{\alpha}_t}}\epsilon\right) + \sqrt{1-\bar{\alpha}_{t-1}}\epsilon' \\ &= \frac{\sqrt{\bar{\alpha}_{t-1}}}{\sqrt{\bar{\alpha}_t}}x_t + \left(\sqrt{1-\bar{\alpha}_{t-1}} - \frac{\sqrt{\bar{\alpha}_{t-1}}\sqrt{1-\bar{\alpha}_t}}{\sqrt{\bar{\alpha}_t}}\right)\epsilon' \end{aligned}$$

Derivation of Denoising Function μ_θ (2/2)

- Substitute the expression for x_0 into the equation for x_{t-1}:

$$\begin{aligned} x_{t-1} &= \sqrt{\bar{\alpha}_{t-1}}\left(\frac{1}{\sqrt{\bar{\alpha}_t}}x_t - \frac{\sqrt{1-\bar{\alpha}_t}}{\sqrt{\bar{\alpha}_t}}\epsilon\right) + \sqrt{1-\bar{\alpha}_{t-1}}\epsilon' \\ &= \frac{\sqrt{\bar{\alpha}_{t-1}}}{\sqrt{\bar{\alpha}_t}}x_t + \left(\sqrt{1-\bar{\alpha}_{t-1}} - \frac{\sqrt{\bar{\alpha}_{t-1}}\sqrt{1-\bar{\alpha}_t}}{\sqrt{\bar{\alpha}_t}}\right)\epsilon' \end{aligned}$$

- The denoising function μ_θ aims to estimate x_{t-1} from x_t, leading to the typical form:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{1-\beta_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right)$$

where $\epsilon_\theta(x_t, t)$ is the neural network's estimate of the noise ϵ

Toy Example: Diffusion Process

- Let's consider a simple one-dimensional example with a linear noise schedule
- Initial data point $x_0 = 5$, noise schedule $\beta_t = 0.2$ for all timesteps.

Timestep	Scale Factor	Noise	Noisy Data
0	-	-	5
1	$\sqrt{1-0.2}$	ϵ_1	$5\sqrt{1-0.2} + \sqrt{0.2}\epsilon_1$
2	$\sqrt{1-0.2}$	ϵ_2	$5(\sqrt{1-0.2})^2 + \sqrt{0.2}(\epsilon_1 + \epsilon_2)$
...	...	...	...
T	$\sqrt{1-0.2}$	ϵ_T	$5(\sqrt{1-0.2})^T + \sqrt{0.2}\sum_{t=1}^{T}\epsilon_t$

- As the number of timesteps increases, the data point becomes increasingly noisy

Toy Example: Reverse Process

- Continuing from the previous toy example, let's consider the reverse process to denoise the data
- Assume the neural network predicts the noise $\epsilon_\theta(x_t, t)$ accurately

Timestep	Predicted Noise	Denoised Data
T	ϵ_T	$5(\sqrt{1-0.2})^T + \sqrt{0.2}\sum_{t=1}^{T}\epsilon_t$
T-1	ϵ_{T-1}	$5(\sqrt{1-0.2})^{T-1} + \sqrt{0.2}\sum_{t=1}^{T-1}\epsilon_t$
...	...	...
1	ϵ_1	$5\sqrt{1-0.2} + \sqrt{0.2}\epsilon_1$
0	-	5

- The reverse process iteratively removes the predicted noise, gradually recovering the original data point

Denoising Objective Function

- Objective of DDPMs is to maximize the log-likelihood of the data under the reverse diffusion process:

$$\max_{\theta} \mathbb{E}_{q(x_{0:T})} \left[\log p_\theta(x_{0:T}) \right]$$

where $q(x_{0:T})$ represents the forward diffusion process. What should be the objective function to train?

Denoising Objective Function

- Objective of DDPMs is to maximize the log-likelihood of the data under the reverse diffusion process:

$$\max_{\theta} \mathbb{E}_{q(x_{0:T})} \left[\log p_\theta(x_{0:T}) \right]$$

  where $q(x_{0:T})$ represents the forward diffusion process. What should be the objective function to train?

- Objective should encourage the model to accurately reconstruct the original data x_0 from the noise x_T through the reverse process

Denoising Objective Function

- Objective of DDPMs is to maximize the log-likelihood of the data under the reverse diffusion process:

$$\max_{\theta} \mathbb{E}_{q(x_{0:T})} \left[\log p_\theta(x_{0:T}) \right]$$

  where $q(x_{0:T})$ represents the forward diffusion process. What should be the objective function to train?
- Objective should encourage the model to accurately reconstruct the original data x_0 from the noise x_T through the reverse process
- Log likelihood can be decomposed into a sum of conditional log likelihoods:

$$\log p_\theta(x_{0:T}) = \log p(x_T) + \sum_{t=1}^{T} \log p_\theta(x_{t-1}|x_t)$$

- Training objective (negative log likelihood) is minimized using Stochastic Gradient Descent (SGD)

Training DDPMs

- Reverse process is parameterized by a neural network with parameters θ, optimized via SGD
- Training loss builds on the denoising objective, formulated as a weighted sum of squared errors:

$$\mathcal{L}(\theta) = \sum_{t=1}^{T} w_t \mathbb{E}_{q(x_t|x_0)} \left[\|x_0 - \mu_\theta(x_t, t)\|^2 \right]$$

where w_t are weights emphasizing different stages of the reverse process, $\mu_\theta(x_t, t)$ is predicted mean of x_0 given x_t

Training DDPMs

- Reverse process is parameterized by a neural network with parameters θ, optimized via SGD
- Training loss builds on the denoising objective, formulated as a weighted sum of squared errors:

$$\mathcal{L}(\theta) = \sum_{t=1}^{T} w_t \mathbb{E}_{q(x_t|x_0)} \left[\|x_0 - \mu_\theta(x_t, t)\|^2 \right]$$

  where w_t are weights emphasizing different stages of the reverse process, $\mu_\theta(x_t, t)$ is predicted mean of x_0 given x_t
- Weights w_t can be adjusted based on signal-to-noise ratio at each timestep, balancing importance of early and late stages of denoising. This ensures that model focuses on accurate reconstruction at each stage of reverse process

Training Algorithm

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

Sampling from DDPMs (1/2)

- To generate new samples, the reverse diffusion process is initiated with a sample from the standard Gaussian distribution:

$$x_T \sim \mathcal{N}(0, \mathbf{I})$$

Sampling from DDPMs (1/2)

- To generate new samples, the reverse diffusion process is initiated with a sample from the standard Gaussian distribution:

$$x_T \sim \mathcal{N}(0, \mathbf{I})$$

- The reverse process then iteratively denoises x_T to obtain the sample x_0. At each step t, the distribution of x_{t-1} given x_t is modeled as:

$$x_{t-1} \sim \mathcal{N}(\mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

where $\mu_\theta(x_t, t)$ is predicted mean for the denoised data and $\Sigma_\theta(x_t, t)$ is predicted covariance

Sampling from DDPMs (1/2)

- To generate new samples, the reverse diffusion process is initiated with a sample from the standard Gaussian distribution:

$$x_T \sim \mathcal{N}(0, \mathbf{I})$$

- The reverse process then iteratively denoises x_T to obtain the sample x_0. At each step t, the distribution of x_{t-1} given x_t is modeled as:

$$x_{t-1} \sim \mathcal{N}(\mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

  where $\mu_\theta(x_t, t)$ is predicted mean for the denoised data and $\Sigma_\theta(x_t, t)$ is predicted covariance

- Parameters μ_θ and Σ_θ are learned during the training phase; used to guide the denoising process in reverse diffusion

Sampling from DDPMs (2/2)

- Iterative denoising process continues until x_0 is reached, which is the generated sample representing the model's approximation of the data distribution

Sampling from DDPMs (2/2)

- Iterative denoising process continues until x_0 is reached, which is the generated sample representing the model's approximation of the data distribution
- Quality of generated samples depends on accuracy of learned reverse process in modeling the true data distribution

Sampling from DDPMs (2/2)

- Iterative denoising process continues until x_0 is reached, which is the generated sample representing the model's approximation of the data distribution
- Quality of generated samples depends on accuracy of learned reverse process in modeling the true data distribution
- The variance schedule $\Sigma_\theta(x_t, t)$ can be adapted during sampling to control the trade-off between sample diversity and fidelity. A higher variance may increase diversity but reduce fidelity, and vice versa

Sampling from DDPMs (2/2)

- Iterative denoising process continues until x_0 is reached, which is the generated sample representing the model's approximation of the data distribution
- Quality of generated samples depends on accuracy of learned reverse process in modeling the true data distribution
- The variance schedule $\Sigma_\theta(x_t, t)$ can be adapted during sampling to control the trade-off between sample diversity and fidelity. A higher variance may increase diversity but reduce fidelity, and vice versa
- Sampling process is inherently stochastic, with randomness introduced at each step through the sampling from the Gaussian distribution. This leads to a variety of generated samples, capturing the diversity of the data distribution

Sampling Algorithm

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 $\mathbf{x}_0$

Schedules for β and α Values

- β_t controls noise level at each step of forward diffusion process. It is typically chosen to increase gradually, ensuring a smooth transition from data to noise
- $\alpha_t = 1 - \beta_t$ represents retention rate of the data's original signal at each step. It decreases gradually, reflecting increasing noise level

Schedules for β and α Values

- β_t controls noise level at each step of forward diffusion process. It is typically chosen to increase gradually, ensuring a smooth transition from data to noise
- $\alpha_t = 1 - \beta_t$ represents retention rate of the data's original signal at each step. It decreases gradually, reflecting increasing noise level
- Cumulative product $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$ indicates overall retention rate up to step $t \rightarrow$ crucial for computing mean of reverse process distribution
- Schedules are designed to balance trade-off between model's ability to capture complex data distributions and training stability – a well-chosen schedule can significantly impact model's performance

Variational Inference in DDPMs: Formulation

- **Goal:** Approximate the intractable posterior $p(x_{0:T}|x_0)$ with a tractable distribution $q(x_{0:T})$

Variational Inference in DDPMs: Formulation

- **Goal:** Approximate the intractable posterior $p(x_{0:T}|x_0)$ with a tractable distribution $q(x_{0:T})$
- Variational lower bound (ELBO):

$$\mathcal{L}(\theta) = \mathbb{E}_{q(x_{0:T})}\left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right]$$

Variational Inference in DDPMs: Formulation

- **Goal:** Approximate the intractable posterior $p(x_{0:T}|x_0)$ with a tractable distribution $q(x_{0:T})$
- Variational lower bound (ELBO):

$$\mathcal{L}(\theta) = \mathbb{E}_{q(x_{0:T})}\left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right]$$

- Decompose the lower bound into data likelihood, prior, and KL divergence terms:

$$\begin{aligned}\mathcal{L}(\theta) &= \mathbb{E}_{q(x_{0:T})}\left[\log p(x_0) + \sum_{t=1}^{T} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right] \\ &= \mathbb{E}_{q(x_{0:T})}\left[\log p(x_0)\right] - \sum_{t=1}^{T} \mathsf{KL}\left(q(x_t|x_{t-1}) \| p_\theta(x_{t-1}|x_t)\right)\end{aligned}$$

Variational Inference in DDPMs: Mathematical Derivation (1/2)

- Start with ELBO:

$$\mathcal{L}(\theta) = \mathbb{E}_{q(x_{0:T})}\left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right]$$

Variational Inference in DDPMs: Mathematical Derivation (1/2)

- Start with ELBO:

$$\mathcal{L}(\theta) = \mathbb{E}_{q(x_{0:T})}\left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right]$$

- Expand $p_\theta(x_{0:T})$ using the chain rule of probability:

$$p_\theta(x_{0:T}) = p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$

Variational Inference in DDPMs: Mathematical Derivation (1/2)

- Start with ELBO:

$$\mathcal{L}(\theta) = \mathbb{E}_{q(x_{0:T})}\left[\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right]$$

- Expand $p_\theta(x_{0:T})$ using the chain rule of probability:

$$p_\theta(x_{0:T}) = p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$

- Substitute this into ELBO and separate terms:

$$\begin{aligned}\mathcal{L}(\theta) &= \mathbb{E}_{q(x_{0:T})}\left[\log p(x_T) + \sum_{t=1}^{T} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right] \\ &= \mathbb{E}_{q(x_{0:T})}\left[\log p(x_T)\right] + \sum_{t=1}^{T} \mathbb{E}_{q(x_{t-1},x_t)}\left[\log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right]\end{aligned}$$

Variational Inference in DDPMs: Mathematical Derivation (2/2)

- Continue with the decomposition:

$$\mathcal{L}(\theta) = \mathbb{E}_{q(x_{0:T})}\left[\log p(x_T)\right] - \sum_{t=1}^{T} \mathsf{KL}\left(q(x_t|x_{t-1}) \| p_\theta(x_{t-1}|x_t)\right)$$

$$= \mathbb{E}_{q(x_T)}\left[\log p(x_T)\right] - \sum_{t=1}^{T} \mathsf{KL}\left(q(x_t|x_{t-1}) \| p_\theta(x_{t-1}|x_t)\right)$$

- KL divergence terms measure discrepancy between forward and reverse processes

Variational Inference in DDPMs: Mathematical Derivation (2/2)

- Continue with the decomposition:

$$\mathcal{L}(\theta) = \mathbb{E}_{q(x_{0:T})}\left[\log p(x_T)\right] - \sum_{t=1}^{T} \mathsf{KL}\left(q(x_t|x_{t-1}) \| p_\theta(x_{t-1}|x_t)\right)$$

$$= \mathbb{E}_{q(x_T)}\left[\log p(x_T)\right] - \sum_{t=1}^{T} \mathsf{KL}\left(q(x_t|x_{t-1}) \| p_\theta(x_{t-1}|x_t)\right)$$

- KL divergence terms measure discrepancy between forward and reverse processes
- To maximize ELBO, we minimize KL divergence terms, aligning the reverse process with the forward process

Variational Inference in DDPMs: Mathematical Derivation (2/2)

- Continue with the decomposition:

$$\mathcal{L}(\theta) = \mathbb{E}_{q(x_{0:T})}\left[\log p(x_T)\right] - \sum_{t=1}^{T} \mathsf{KL}\left(q(x_t|x_{t-1}) \| p_\theta(x_{t-1}|x_t)\right)$$

$$= \mathbb{E}_{q(x_T)}\left[\log p(x_T)\right] - \sum_{t=1}^{T} \mathsf{KL}\left(q(x_t|x_{t-1}) \| p_\theta(x_{t-1}|x_t)\right)$$

- KL divergence terms measure discrepancy between forward and reverse processes
- To maximize ELBO, we minimize KL divergence terms, aligning the reverse process with the forward process
- This optimization is typically performed using SGD, updating the parameters θ to improve the reverse process

Score-Based Models: Overview

- **Score-based models**, also known as *energy-based models*, focus on learning the score function, which is the gradient of the log-density of the data distribution:

$$s(x) = \nabla_x \log p(x)$$

Score-Based Models: Overview

- **Score-based models**, also known as *energy-based models*, focus on learning the score function, which is the gradient of the log-density of the data distribution:

$$s(x) = \nabla_x \log p(x)$$

- Score function captures the direction and magnitude of the steepest ascent in the data density, providing information about the structure of the data distribution

Score-Based Models: Overview

- **Score-based models**, also known as *energy-based models*, focus on learning the score function, which is the gradient of the log-density of the data distribution:

$$s(x) = \nabla_x \log p(x)$$

- Score function captures the direction and magnitude of the steepest ascent in the data density, providing information about the structure of the data distribution
- Score-based models trained to approximate true score function using a parameterized model $s_\theta(x)$
- This formulation is useful as it allows model to capture complex data distributions without explicitly modeling the density $p(x)$

Training Score-Based Models

- Score-based models trained by minimizing the discrepancy between estimated score $s_\theta(x)$ and true score $s(x)$, typically using a loss function based on Fisher divergence:

$$\mathcal{L}(\theta) = \mathbb{E}_{p(x)}\left[\frac{1}{2}\|s_\theta(x) - s(x)\|^2\right]$$

Training Score-Based Models

- Score-based models trained by minimizing the discrepancy between estimated score $s_\theta(x)$ and true score $s(x)$, typically using a loss function based on Fisher divergence:

$$\mathcal{L}(\theta) = \mathbb{E}_{p(x)}\left[\frac{1}{2}\|s_\theta(x) - s(x)\|^2\right]$$

- In practice, true score $s(x)$ is unknown and is replaced by perturbed samples $x' = x + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$; the model is trained to denoise these samples

Training Score-Based Models

- Score-based models trained by minimizing the discrepancy between estimated score $s_\theta(x)$ and true score $s(x)$, typically using a loss function based on Fisher divergence:

$$\mathcal{L}(\theta) = \mathbb{E}_{p(x)}\left[\frac{1}{2}\|s_\theta(x) - s(x)\|^2\right]$$

- In practice, true score $s(x)$ is unknown and is replaced by perturbed samples $x' = x + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$; the model is trained to denoise these samples
- The training objective becomes:

$$\mathcal{L}(\theta) = \mathbb{E}_{p(x),\epsilon}\left[\frac{1}{2}\|s_\theta(x') - \frac{\epsilon}{\sigma^2}\|^2\right]$$

Training Score-Based Models

- Score-based models trained by minimizing the discrepancy between estimated score $s_\theta(x)$ and true score $s(x)$, typically using a loss function based on Fisher divergence:

$$\mathcal{L}(\theta) = \mathbb{E}_{p(x)}\left[\frac{1}{2}\|s_\theta(x) - s(x)\|^2\right]$$

- In practice, true score $s(x)$ is unknown and is replaced by perturbed samples $x' = x + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$; the model is trained to denoise these samples
- The training objective becomes:

$$\mathcal{L}(\theta) = \mathbb{E}_{p(x),\epsilon}\left[\frac{1}{2}\|s_\theta(x') - \frac{\epsilon}{\sigma^2}\|^2\right]$$

- This training strategy effectively leverages the score function's ability to characterize the data distribution through the gradients of its log density

Connection to DDPMs

- Both DDPMs and score-based models involve learning to denoise data through a diffusion process!

Connection to DDPMs

- Both DDPMs and score-based models involve learning to denoise data through a diffusion process!
- In DDPMs, the denoising function $\mu_\theta(x_t, t)$ at each diffusion step can be interpreted as an estimate of the score of the perturbed data distribution:

$$\mu_\theta(x_t, t) \approx \frac{1}{\sqrt{1-\bar{\alpha}_t}} s_\theta(x_t)$$

Connection to DDPMs

- Both DDPMs and score-based models involve learning to denoise data through a diffusion process!
- In DDPMs, the denoising function $\mu_\theta(x_t, t)$ at each diffusion step can be interpreted as an estimate of the score of the perturbed data distribution:

$$\mu_\theta(x_t, t) \approx \frac{1}{\sqrt{1-\bar{\alpha}_t}} s_\theta(x_t)$$

- This connection implies that training a DDPM can be viewed as learning an implicit score function, which is used to guide the reverse diffusion process
- The interplay between DDPMs and score-based models highlights the importance of the score function in capturing the underlying structure of the data distribution and guiding the generation of new samples

Score-Based Models and Sampling

- Sampling from score-based models involves integrating a stochastic differential equation (SDE) driven by the learned score function:

$$dx = s_\theta(x)dt + \sqrt{2}dW$$

  where W is a Wiener process
- This SDE describes a continuous-time diffusion process that gradually transforms noise into samples from the data distribution, guided by the learned score function

Score-Based Models and Sampling

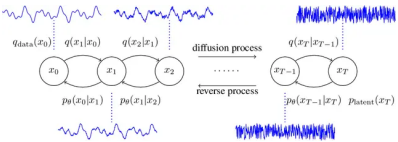
- Sampling from score-based models involves integrating a stochastic differential equation (SDE) driven by the learned score function:

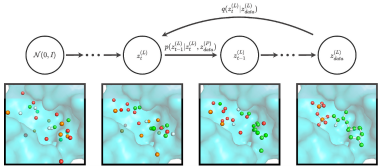
$$dx = s_\theta(x)dt + \sqrt{2}dW$$

  where W is a Wiener process
- This SDE describes a continuous-time diffusion process that gradually transforms noise into samples from the data distribution, guided by the learned score function
- The connection with DDPMs provides a discrete-time counterpart to this continuous-time sampling process, with the reverse diffusion steps corresponding to discrete updates driven by the implicit score function
- The effectiveness of score-based models in sampling hinges on the accuracy of the learned score function in capturing the intricate details of the data distribution

Applications of DDPMs

- Successfully applied in various domains, including image generation, audio synthesis, and molecular design
- **Image generation:** DDPMs can produce high-quality, diverse samples competitive with those generated by GANs
- **Audio synthesis:** DDPMs used to generate realistic and coherent audio waveforms, such as speech and music
- **Molecular design:** DDPMs used to generate novel molecular structures with desired properties, aiding in drug discovery and materials science





Limitations and Challenges

- While DDPMs have shown impressive results, they are not without limitations and challenges. What could they be?

Limitations and Challenges

- While DDPMs have shown impressive results, they are not without limitations and challenges. What could they be?
- Challenges:
  1. Computational cost of training and sampling, as diffusion process involves multiple steps and requires evaluating a neural network at each step

Limitations and Challenges

- While DDPMs have shown impressive results, they are not without limitations and challenges. What could they be?
- Challenges:
  1. Computational cost of training and sampling, as diffusion process involves multiple steps and requires evaluating a neural network at each step
  2. Choice of noise schedules (β and α values), which can significantly impact model's performance and stability

Limitations and Challenges

- While DDPMs have shown impressive results, they are not without limitations and challenges. What could they be?
- Challenges:
  1. Computational cost of training and sampling, as diffusion process involves multiple steps and requires evaluating a neural network at each step
  2. Choice of noise schedules (β and α values), which can significantly impact model's performance and stability
  3. Ensuring diversity in generated samples and avoiding mode collapse can be difficult, especially in high-dimensional spaces

Comparison with Other Generative Models

- Generative models often face a trilemma between *fast sampling*, *mode convergence*, and *high-quality samples*. Can you think of how GANs, VAEs and DDPMs may handle these?

Comparison with Other Generative Models

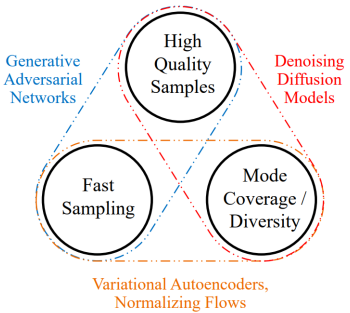
- Generative models often face a trilemma between *fast sampling*, *mode convergence*, and *high-quality samples*. Can you think of how GANs, VAEs and DDPMs may handle these?
- **GANs** generate high-quality samples quickly but may suffer from mode collapse, where they fail to capture the diversity of the data distribution

Comparison with Other Generative Models

- Generative models often face a trilemma between *fast sampling*, *mode convergence*, and *high-quality samples*. Can you think of how GANs, VAEs and DDPMs may handle these?
- **GANs** generate high-quality samples quickly but may suffer from mode collapse, where they fail to capture the diversity of the data distribution
- **VAEs** provide better mode coverage due to their probabilistic nature but often produce lower-quality samples compared to GANs

Comparison with Other Generative Models

- Generative models often face a trilemma between *fast sampling*, *mode convergence*, and *high-quality samples*. Can you think of how GANs, VAEs and DDPMs may handle these?
- **GANs** generate high-quality samples quickly but may suffer from mode collapse, where they fail to capture the diversity of the data distribution
- **VAEs** provide better mode coverage due to their probabilistic nature but often produce lower-quality samples compared to GANs
- **DDPMs** excel in generating high-quality samples with good mode coverage, but the iterative denoising process leads to slower sampling times



Practical Considerations

- Training DDPMs requires careful tuning of hyperparameters, including the noise schedule and learning rate
- Choice of architecture for the neural network is also crucial, with larger models typically providing better performance
- Sampling from DDPMs can be computationally intensive, as it involves multiple steps of denoising; techniques like sub-sampling can be used to speed up the process.
- Implementations of DDPMs should leverage parallel computing and hardware acceleration to manage computational demands

Homework

Readings

- Lilian Weng, What are Diffusion Models?
- (YouTube video) Tutorial on Denoising Diffusion-based Generative Modeling: Foundations and Applications