Diffusion Models

10th Mar, 2025

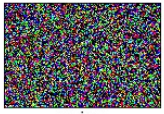
Speaker: Li Longlong

Why Generative Modeling [1]

A well-trained deep neural network can classify animal images $\mathbf{x} \in \mathbb{Z}^D$, $y \in \mathcal{Y} = \{\text{cat, dog, horse}\}$ with high confidence $p(y|\mathbf{x})$. However, as shown by Szegedy et al [1], adding small perturbations to an image can drastically alter the predicted label while keeping the image visually unchanged.

$p(y = cat|\mathbf{x}) = 0.90$
$p(y = dog|\mathbf{x}) = 0.05$
$p(y = horse|\mathbf{x}) = 0.05$

+



noise

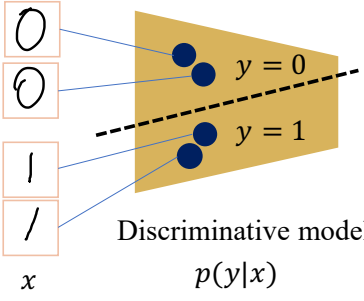
=

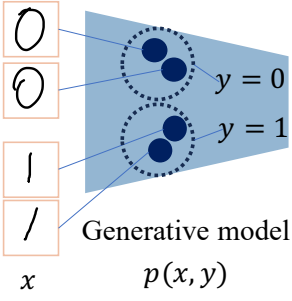
$p(y = cat|\mathbf{x}) = 0.05$
$p(y = dog|\mathbf{x}) = 0.05$
$p(y = horse|\mathbf{x}) = 0.90$

Therefore, beyond recognizing decision boundaries, the model should also capture the underlying data distributions.

Discriminative vs. Generative Models

	Discriminative Models	**Generative Models**
Goal	Learn decision boundary	Model data distribution
Probability	$p(Y\|X)$	$p(X, Y)$ or $p(X)$
Usage	Classification	Generation, Denoising
Examples	SVM, MLP, k-NN	GAN, VAE, Diffusion





Sampling from Bernoulli Space

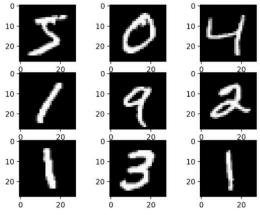
Bernoulli Distribution in MNIST

The MNIST dataset is often represented as binary (black/white) and can be modeled using a Bernoulli Distribution:

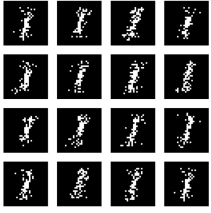
$$p(X = x) = p^x(1-p)^{1-x}, \quad x \in \{0, 1\}$$

where X is a random variable representing the pixel state, x is its specific value (0 for white, 1 for black), and p is the probability of a pixel being 1 (obtained from data statistics).

A. MNIST image

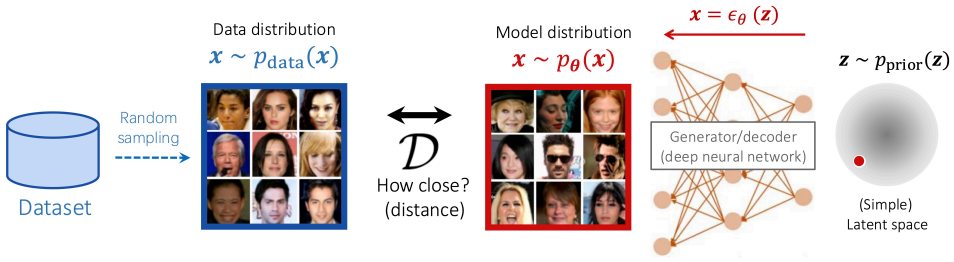


B. Generated from Bernoulli distribution



Deep Learning based Generative Models [2]

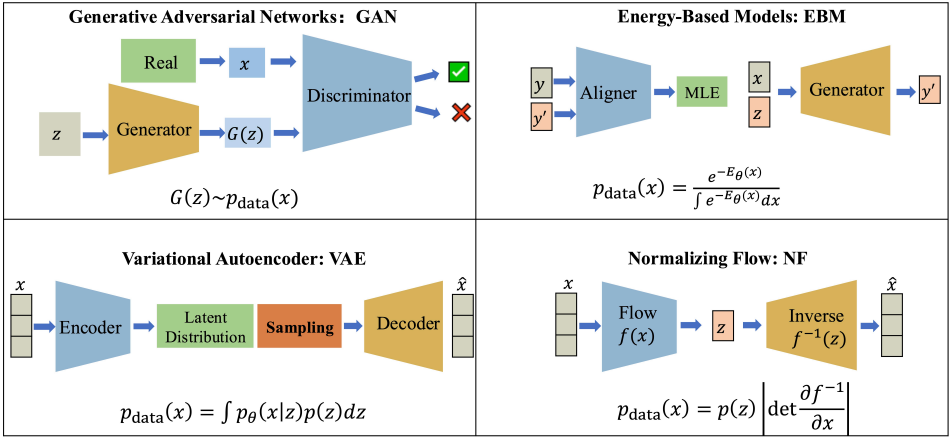
Generative models use neural networks to learn the target distribution.



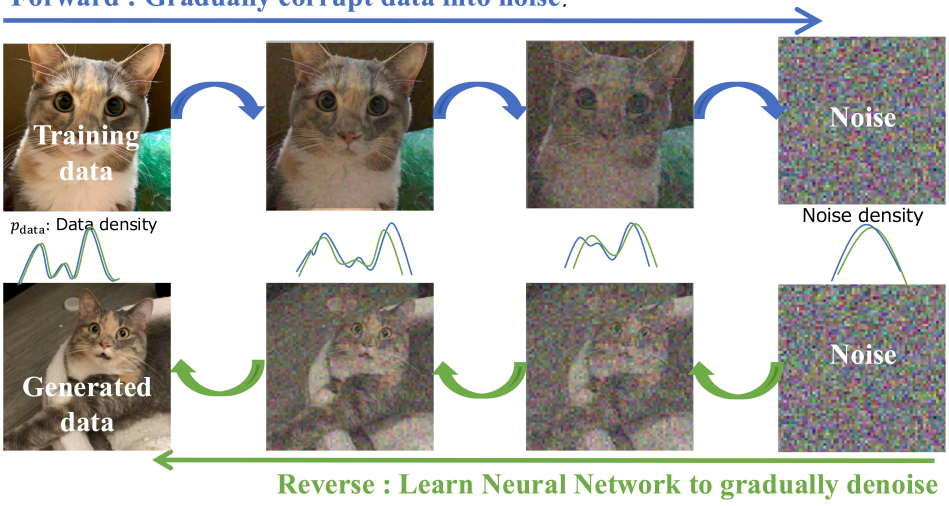
$$\min_\theta \mathcal{D}\left(p_\theta(x), p_{\text{data}}(x)\right)$$

where $\mathcal{D}$ is a divergence metric (e.g., KL divergence, Wasserstein distance).

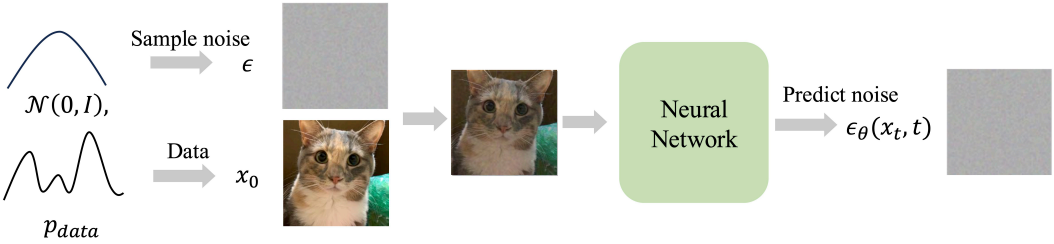
Comparison of Deep Generative Models [3]



Diffusion Model Architecture [4]



DDPM Forward Process [5]



Markov Chain-based:

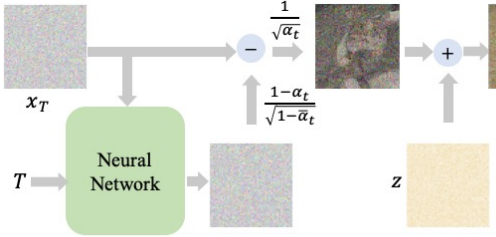
$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \Rightarrow x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I),$$

$$q(x_{1:T}) = q(x_0)\prod_{t=1}^{T} q(x_t|x_{t-1}) \Rightarrow x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \quad \alpha_t = 1-\beta_t, \quad \bar{\alpha}_t = \prod_{i=1}^{t}\alpha_i$$

Loss function:

$$\mathcal{L}(\theta) = \mathbb{E}_{x_0,\epsilon,t}\left[\|\epsilon_\theta(x_t, t) - \epsilon\|^2\right]$$

DDPM Reverse Process [5]



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

- $\mu_\theta(x_t, t)$ is the **mean** of the predicted distribution
- $\sum_\theta(x_t, t) = \beta_t I$ is the **variance**, controlling the noise level

Reverse Diffusion Mean Formula: $\mu_\theta(x_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right) + \sigma_t z$

where:

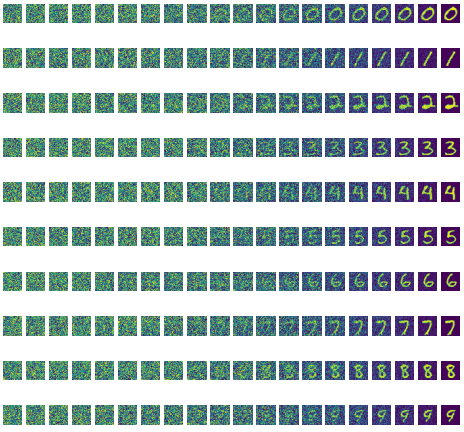
- $\epsilon_\theta(x_t, t)$ is the neural network's prediction of the added noise.
- $\sigma_t z$ represents random noise, ensuring that the generated data is not solely determined by the neural network but retains stochasticity.

DDPM Sampling Process on MNIST [6]

- **Step 1:** Start from Gaussian noise $x_T \sim \mathcal{N}(0, I)$.
- **Step 2:** Iteratively refine the sample by denoising:

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

- **Step 3:** Repeat for $t = T, T-1, \ldots, 1$ to reconstruct a clean sample.



SGM: Score-Based Generative Model [7, 8, 9]

Fokker-Planck Equation:

$$\frac{\partial p(x,t)}{\partial t} = -\nabla_x \cdot (p(x,t)v(x,t)) + D\nabla_x^2 p(x,t)$$

where:

- The first term $-\nabla_x \cdot (p(x,t)v(x,t))$ is the **drift term**, describing how the data moves along a certain direction.
- The second term $D\nabla_x^2 p(x,t)$ is the **diffusion term**, representing how noise propagates.

Forward Process (Stochastic Differential Equation):

$$dx = f(x,t)dt + g(t)dW_t$$

where:

- $f(x,t)$ is the **drift coefficient**, controls **mean variation**.
- $g(t)$ controls the **noise intensity**.
- dW_t represents a **standard Wiener process**.

SGM: Score-Based Generative Model [7, 8, 9]

Reverse Process (Inverse Stochastic Differential Equation):

$$dx = [f(x, t) - g(t)^2 \nabla_x \log p_t(x)]\, dt + g(t)\, d\bar{W}_t$$

where:

- $\nabla_x \log p_t(x)$ is the **score function**, which we need to estimate. Through the score function, the **Inverse SDE guides sampling toward high-density regions** of the data distribution.
- $g(t)\, d\bar{W}_t$ represents **random noise**, which introduces stochasticity in the reverse process.

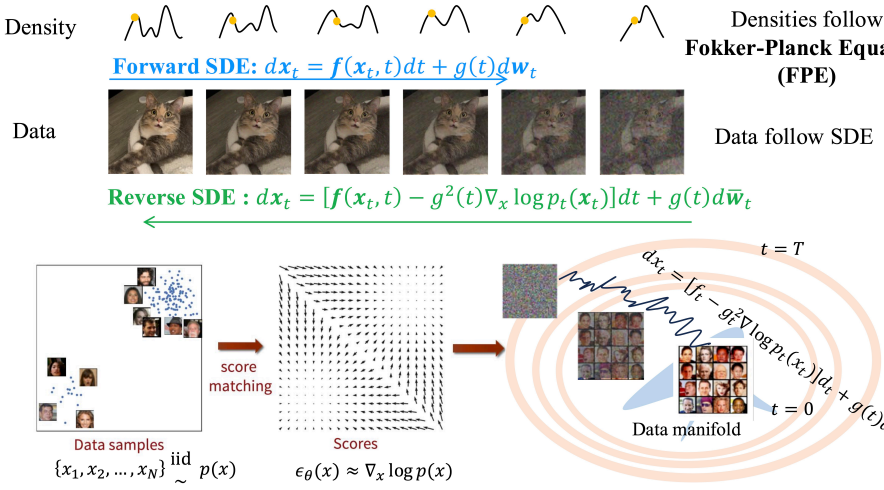
Score Matching Loss Function:

$$\mathcal{L}(\theta) = \mathbb{E}_{p_t(x)} \left[\lambda(t) \| \nabla_x \log p_t(x) + \epsilon_\theta(x, t) \|^2 \right] \Rightarrow \mathcal{L} = \mathbb{E}_{q(x_0, x_t)} \left[\| s_\theta(x_t, t) - \nabla_x \log p_t(x_t|x_0) \|^2 \right]$$

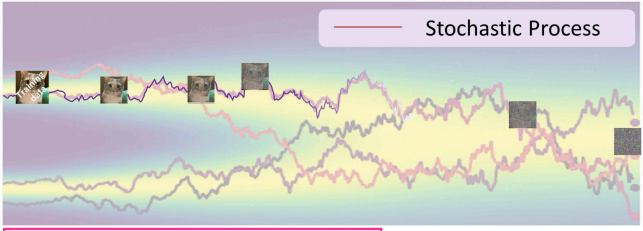
where:

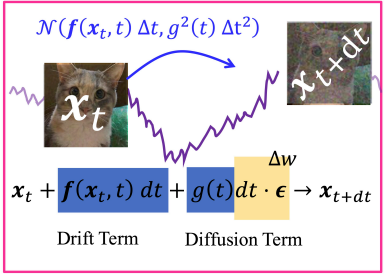
- $s_\theta(x, t)$ is the **neural network estimating the score function**.
- $p_t(x_t|x_0) = \mathcal{N}(x_t, \alpha_t x_0, \sigma_t^2 I) \Rightarrow \nabla_x \log p_t(x_t|x_0) = -\frac{x_t - \alpha_t x_0}{\sigma_t^2}$, derived from the Gaussian perturbation model
- $\lambda(t)$ is a weighting function controlling the loss scale at different timesteps.

Process of SGM



Special case of SGM (DDPM): Forward Process





The Markov Chain-based of DDPM

$$x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\,\epsilon$$

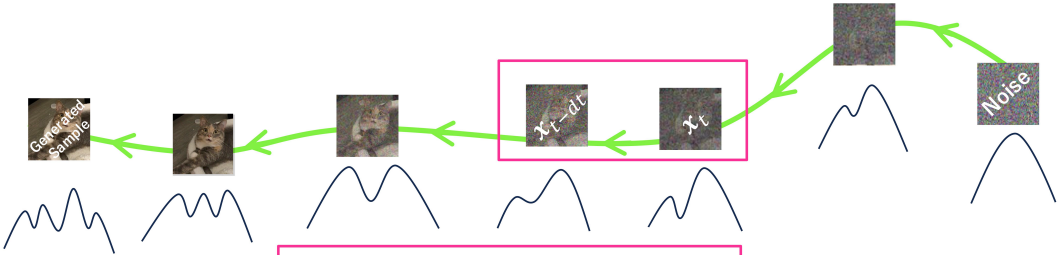
$$\Downarrow$$

$$dx = (\sqrt{1-\beta_t} - 1)x_{t-1} + \sqrt{\beta_t}\,\epsilon$$

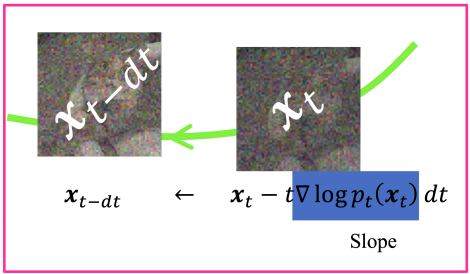
$$\Downarrow$$

$$dx = \underbrace{-\frac{1}{2}\beta_t x dt}_{f(x,t)} + \underbrace{\sqrt{\beta_t}dw}_{g(t)}$$

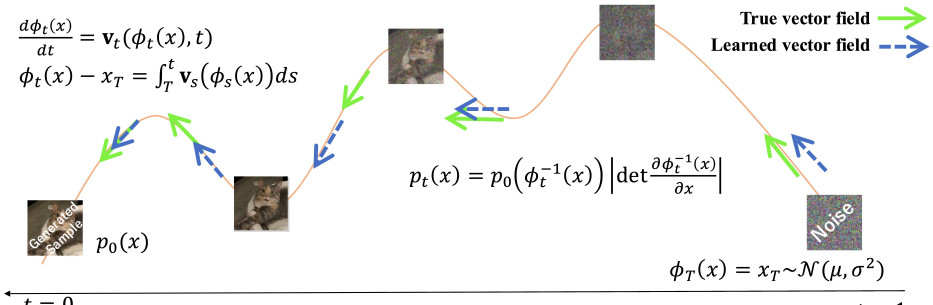
Special case of SGM (DDPM): Reverse Process



$$\frac{dx_t}{dt} = -t\nabla \log p_t(x_t)$$



Flow Matching (FM) [10]



where,
$\phi_t(x)$ is the flow path of the data distribution $p_t(x)$,
and $\mathbf{v}_t(x)$ is the flow direction of the data distribution $p_t(x)$.

$$\mathcal{L} = \mathbb{E}_{p_t(x)}\left[\left\|\mathbf{v}_{t,\theta}(x) - \mathbf{v}_t(x)\right\|^2\right] \Rightarrow \mathbb{E}_{t,q(z),p_t(x_t|z)}\left[\left\|\mathbf{v}_{t,\theta}(x) - \mathbf{v}_t(x_t|x_T)\right\|^2\right]$$

$$\phi_t(x) = tx_T + (1-t)x_0 \Rightarrow p_t(x|x_T) = \mathcal{N}(x|tx_T + (1-t)x_0)$$

where x_0 is the true data, and $p_0(x_0)$ is its distribution

$$\mathbf{v}_t(x|x_T) = \frac{x - x_T}{t},$$

From SGM (Score-Based Generative Models) to Flow Matching

SGM (Score-Based Generative Models) relies on stochastic differential equations (SDEs) for diffusion:

$$dx = f(x, t)dt + g(t)dW_t$$

Flow Matching, however, adopts a deterministic ODE:

$$\frac{d\phi_t(x)}{dt} = \mathbf{v}_t(\phi_t(x), t)$$

How does Flow Matching transition from SGM?

- In SGM, we learn the score function $\epsilon_\theta(x, t) \approx \nabla_x \log p_t(x)$ to estimate the direction of data movement.
- In Flow Matching, we directly learn the velocity field $\mathbf{v}_\theta(x, t)$, which determines the data transport trajectory without requiring the stochastic term dW_t.

This means:

- Flow Matching can be seen as a deterministic ODE formulation of SGM, eliminating stochasticity and leading to more stable training and sampling.
- Sampling only requires solving an ODE, bypassing the need for stochastic SDE inversion, thus improving efficiency and computational speed.

From SGM to Flow Matching

SGM (SDE):

- **Forward SDE:** $d\boldsymbol{x}_t = \boldsymbol{f}(\boldsymbol{x}_t, t)dt + g(t)d\boldsymbol{w}_t$
- **Reverse SDE :** $d\boldsymbol{x}_t = [\boldsymbol{f}(\boldsymbol{x}_t, t) - g^2(t)\nabla_x \log p_t(\boldsymbol{x}_t)]dt + g(t)d\bar{\boldsymbol{w}}_t$

Fokker-Planck Equation (FPE): $\frac{\partial p_t(x)}{\partial t} + \nabla_{\boldsymbol{x}} \cdot \left(p_t(x)\mathbf{v_t}(x)\right) + D\nabla_x^2 p_t(x) = 0$

Remove Diffusion term:

Probability flow ODE: $d\boldsymbol{x}_t = \left[\boldsymbol{f}(\boldsymbol{x}_t, t) - \frac{1}{2}g^2(t)\nabla_x \log p_t(\boldsymbol{x}_t)\right] dt$

Density

Remove Diffusion term : $d\boldsymbol{x}_t = \boldsymbol{f}(\boldsymbol{x}_t, t)dt + g(t)d\boldsymbol{w}_t$, $= 0$

Data

Probability flow ODE : $d\boldsymbol{x}_t = [\boldsymbol{f}(\boldsymbol{x}_t, t) - g^2(t)\nabla_x \log p_t(\boldsymbol{x}_t)]dt + g(t)d\bar{\boldsymbol{w}}_t$, $= 0$

Summary

	DDPM	SGM	FM
Core Concept	Markov chain noise addition & denoising	Score function of data distribution $\nabla_x \log p(x)$	Deterministic vector field $v_t(x)$
Mathematical Formulation	Discrete Markov chain: $q(x_t, x_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}\, x_{t-1}, \beta_t\, I)$	Stochastic Differential Equation (SDE): $dx = f(x,t)dt + g(t)dW_t$	Ordinary Differential Equation (ODE): $\frac{d\phi_t(x)}{dt} = v_\theta(x,t)$
Forward Process (Noise Addition)	Discrete noise addition	Continuous diffusion (SDE)	Deterministic flow transformation (ODE)
Stochasticity	Yes, Gaussian noise $\sigma_t z$	Yes, $g(t)d\overline{W}_t$	No, Fully deterministic
Reverse Process (Sampling)	Discrete denoising	Inverse SDE sampling	Solving Probability Flow ODE
Training Objective	Predict added noise: $\mathcal{L}(\theta) = \mathbb{E}\|\epsilon_\theta(x_t, t) - \epsilon\|^2$	Predict score function of data: $\mathcal{L}(\theta) = \mathbb{E}\|s_\theta(x_t, t) - \nabla_x \log p_t(x_t\|x_0)\|^2$	Train vector field $v_\theta(x,t)$ to match true velocity field $v_t(x)$: $\mathcal{L}(\theta) = \mathbb{E}\|v_\theta(x,t) - v_t(x_t\|x_T)\|^2$