Diffusion

Presented By: Massa Baali

Discriminative vs. Generative Models

- **Discriminative models learn to discriminate**
  - Determine the class given the input
    - Compute P(y|x)
- **Generative models can generate**
  - Produce more instances like the training data
    - Compute and/or draw from P(x,y)

Discriminative vs. Generative Models

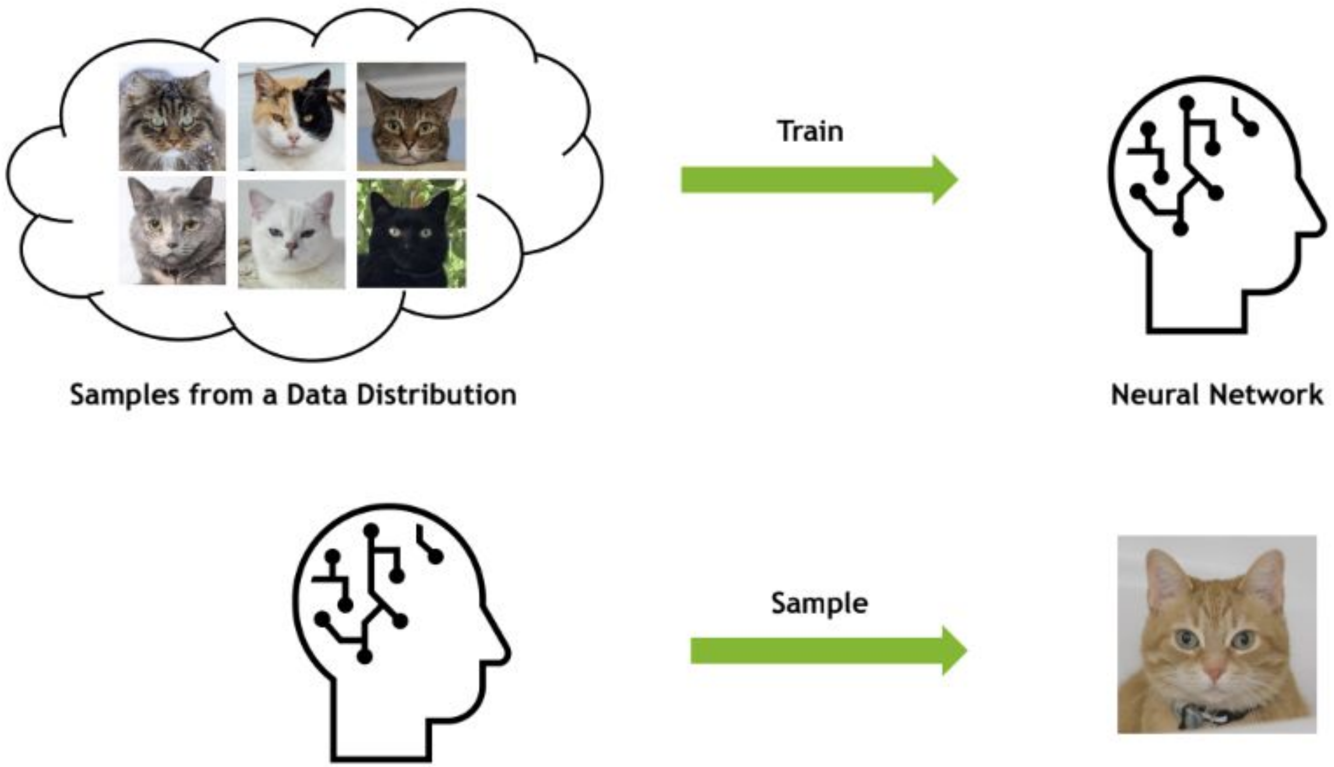
Given a distribution of inputs X and labels Y

Discriminative models	Generative models
Discriminative models learn conditional distribution P(Y | X)	Generative models learn the joint distribution P(Y, X)
Learns decision boundary between classes.	Learns actual probability distribution of data.
Limited scope. Can only be used for classification tasks.	Can do both generative and discriminative tasks.

Harder problem, requires a deeper understanding of the distribution than discriminative models.

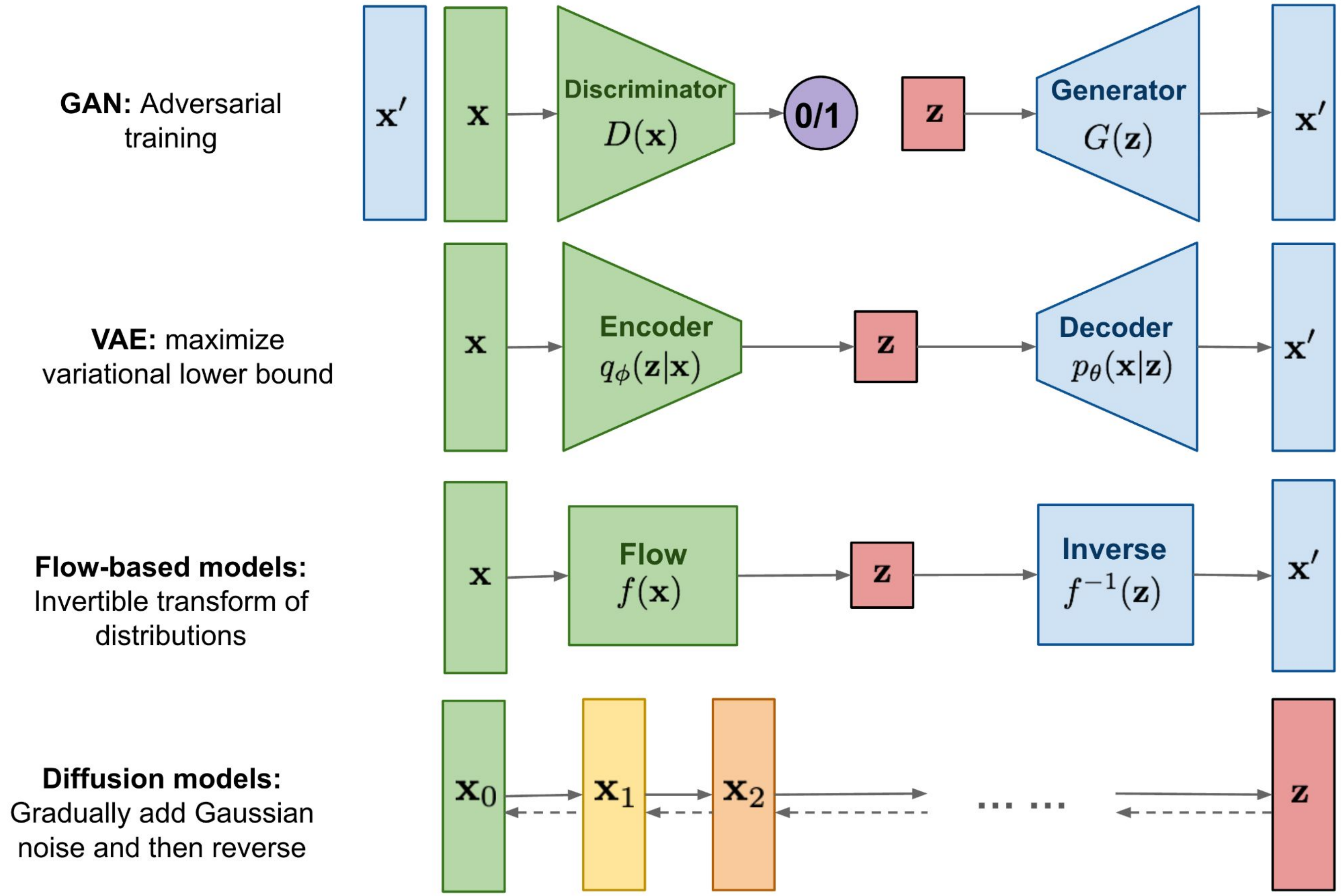
Deep Generative learning

Learning to generate data



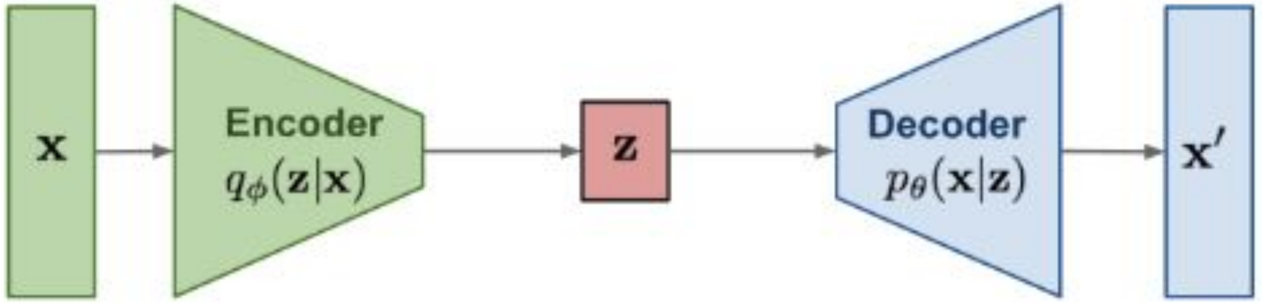
Slide credit to: https://cvpr2022-tutorial-diffusion-models.github.io/

Generative Models



https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

VAE (Recap)

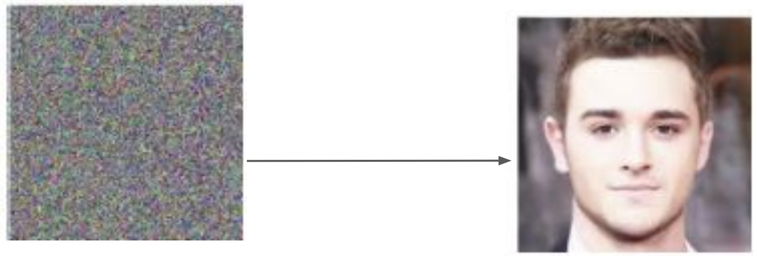


A generative model that learns to create new samples by:

- Compressing samples into a small code (*latent space*)
- Reconstructing samples from these codes

https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

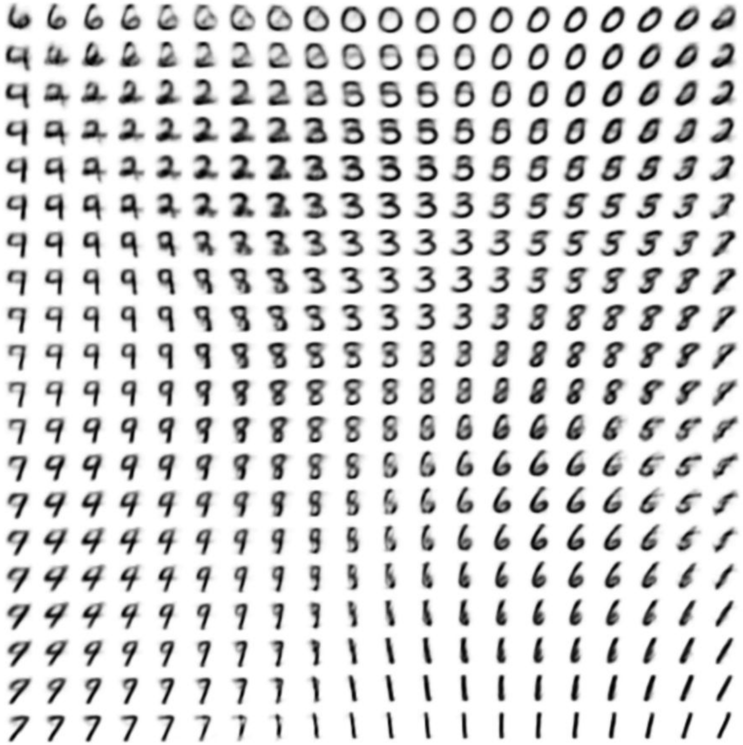
Limitations of VAEs



The decoder must transform a standard Gaussian *all the way* to the target distribution in **one step**.

- Too large a gap to bridge in one step
- The decoder has to do ALL the work at once

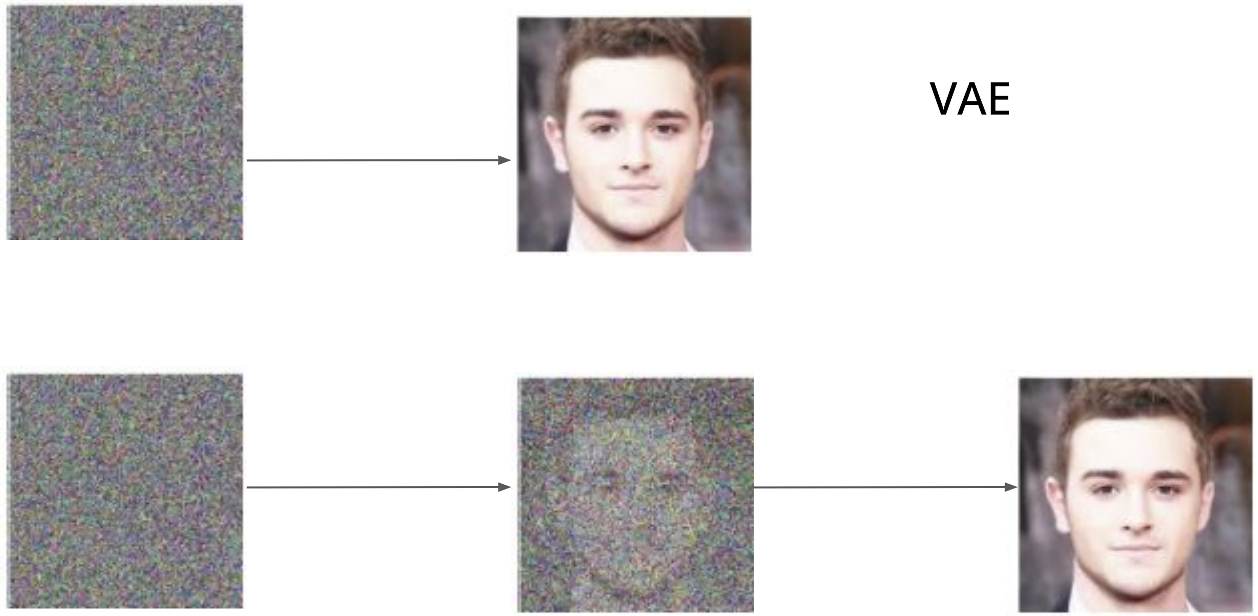
Limitations of VAEs



Result: Blurry, Low-Quality Images

Kingma et al. Auto-Encoding Variational Bayes

From Single-Step to Multi-Step Generation



VAE

Diffusion

Hierarchicals VAEs

- Stack multiple VAEs with intermediate latent variables
- Each level: $x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_t$
- Each step is a small, learnable transformation

Diffusion Models: A Special Case

- Can be viewed as a hierarchical VAE with specific choices:
  - All latent variables have the same dimension as the original image
  - Forward process (encoder) is fixed (just adds Gaussian noise)
  - Only learn the reverse process (decoder/denoising)
  - Many steps (e.g., T = 1000)

Poll 1

Why do VAEs produce blurry images?

- A) Not enough training data
- B) Bad optimizer choice
- C) One-step jump from noise to image is too hard
- D) Network architecture is too small

Poll 1

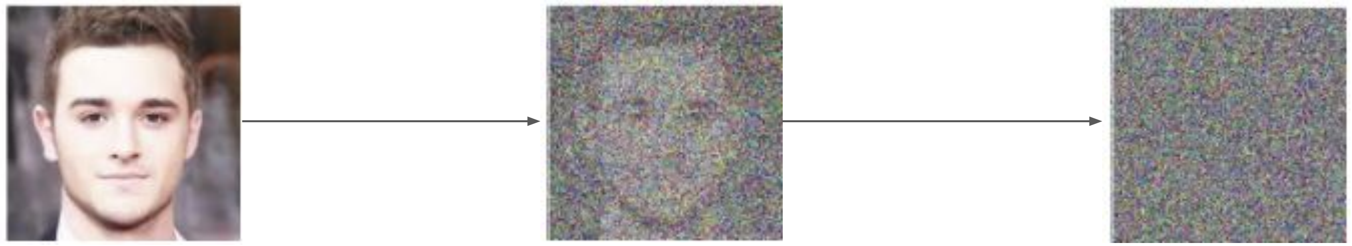
Why do VAEs produce blurry images?

- A) Not enough training data
- B) Bad optimizer choice
- C) One-step jump from noise to image is too hard
- D) Network architecture is too small

Outline

- Definition of Diffusion
- Importance of Diffusion
- Explaining the process of diffusion
- Denoising Diffusion Implicit Models (DDIM)
- Diffusion Models from Stochastic Differential Equations and Score Matching Perspective
- Classifier-free Guidance (CFG)
- Performance Metrics: FID, IS, Precision and Recall
- Applications of Diffusion Models

Definition

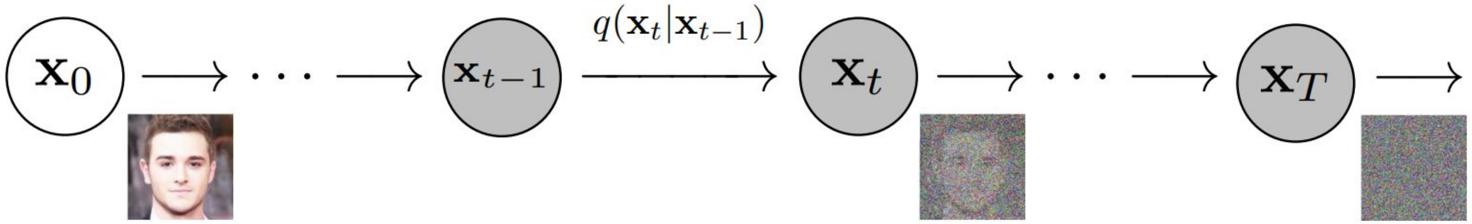


Adapted from: Ho, J., Jain, A. and Abbeel, P., 2020. Denoising diffusion probabilistic models. Advances in neural information processing systems, 33, pp.6840-6851.

Importance of Diffusion Models

- **High-Quality Outputs:** Exceptional generation of images, audio, and more with high fidelity.
- **Stable Training:** More consistent training outcomes compared to other models like GANs.

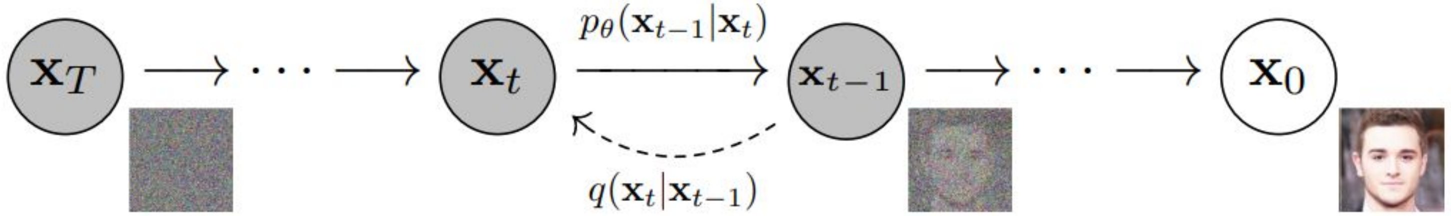
Forward Process



- **Gradual Noise Addition:** Over t time steps, noise is progressively added to an image.
- **Markov Chain Model:** The process, denoted by q takes the form of Markov Chain where the distribution at a particular time steps only depends on the sample from the immediate previous step.

Adapted from:
https://www.assemblyai.com/blog/diffusion-models-for-machine-learning-introduction/

Reverse Process



- **Denoise:** The model learns to gradually remove noise from the data at each step.
- **Reconstruction:** Aims to reconstruct the original image or data from its noisy version.
- The main goal of training a diffusion model is learning the reverse process specifically training pθ(xt−1 | xt).

Adapted from:
https://www.assemblyai.com/blog/diffusion-models-for-machine-learning-introduction/

How do we add noise?

- **Gaussian Noise:** The noise added in diffusion models follows a Gaussian (normal) distribution. This means at each diffusion step, we inject some random noise that has the familiar "bell curve" distribution.
- **Standard Normal Sampling:** We sample the noise from a standard normal distribution $\mathcal{N}(0, \boldsymbol{I})$, which has a mean of 0 and a variance of 1.

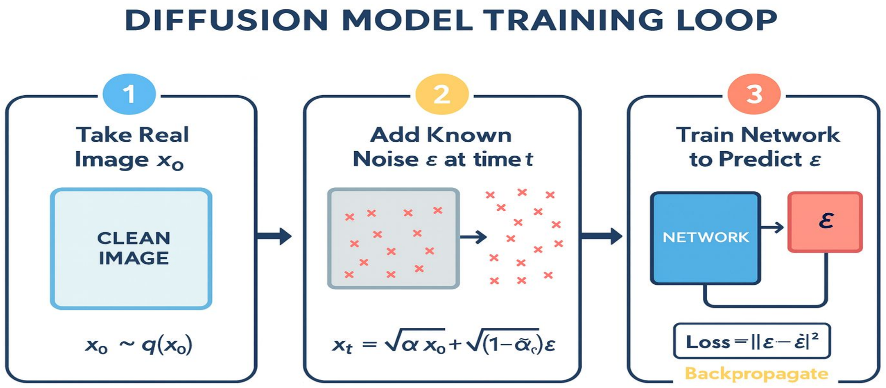
What Does the Model Actually Learn?

The Task:

- Input: Noisy image x_t, timestep t
- Output: Predict the noise ε that was added

Why This Works:

- If we know the noise, we can subtract it
- Subtracting noise = denoising = moving toward clean data



Training Objective

- **Minimize Error:** The goal is to minimize the Mean Squared Error (MSE) between the model's predicted noise and the actual noise added during the forward process.
- **High-Quality Reconstruction:** Effective training enables the model to accurately reconstruct original data from noise, enhancing its capability to generate high-quality samples.

$$L_{\text{simple}}(\theta) := \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t\right)\right\|^2\right]$$

Training

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
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

Algorithm 2 Sampling

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

Poll 2

What does the neural network actually predict during training?

- A) The clean image x_0
- B) The noisy image x_t
- C) The noise ε that was added
- D) The next timestep t+1

Poll 2

What does the neural network actually predict during training?

- A) The clean image x_0
- B) The noisy image x_t
- C) The noise ε that was added
- D) The next timestep t+1

Outline

- Definition of Diffusion
- Importance of Diffusion
- Explaining the process of diffusion
- Denoising Diffusion Implicit Models (DDIM)
- Diffusion Models from Stochastic Differential Equations and Score Matching Perspective
- Classifier-free Guidance (CFG)
- Performance Metrics: FID, IS, Precision and Recall
- Applications of Diffusion Models

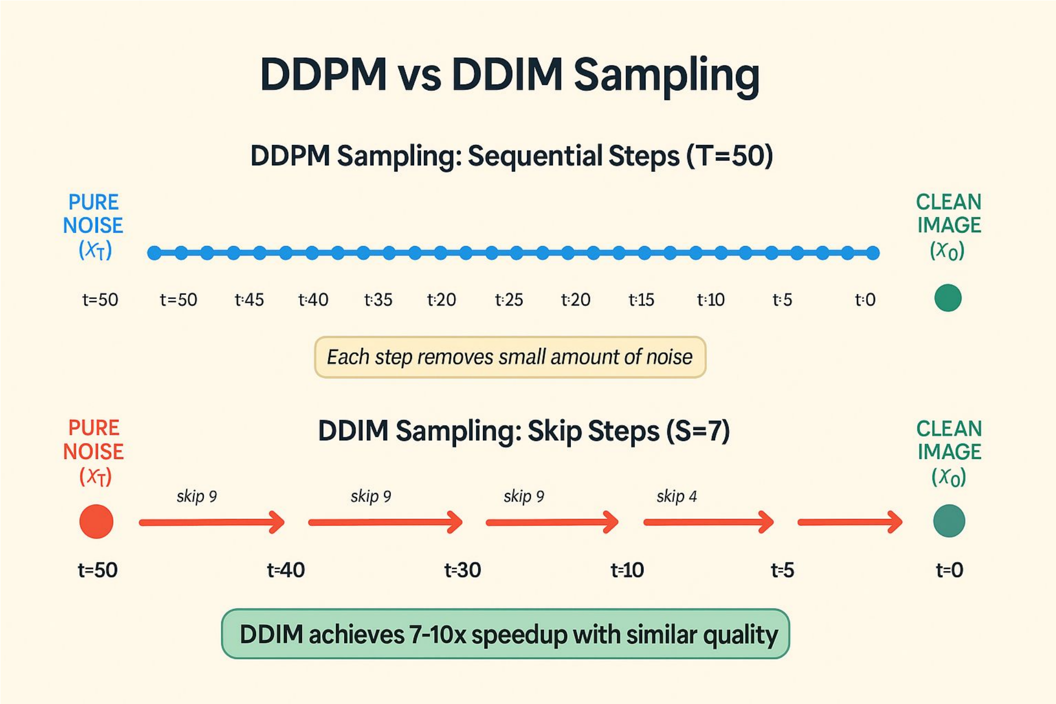
The Speed Problem

DDPM Challenge:

- Must run model T=1000 times sequentially
- Each step removes tiny amount of noise
- Generation takes minutes per image

DDIM Solution:

- Skip timesteps intelligently
- Use 10-50 steps instead of 1000
- 10-100× faster with similar quality



Denoising Diffusion Implicit Models (DDIM)

Limitations of DDPM Inference

Sequential Denoising: Must process each timestep in reverse to remove Gaussian noise. This happens due to the markov chain structure of the reverse process.

Algorithm 2 Sampling

$x_T \sim \mathcal{N}(0, I)$ ▷*Initial isotropic gaussian noise sampling*
for t = T, ..., 1 **do**
  $z \sim \mathcal{N}(0, I)$ if $t > 1$ else $z = 0$ ▷*Sample random noise (if not last step)*
  $\tilde{\epsilon} = \epsilon_\theta(x_t, t)$ ▷*Estimated noise in current noisy data*
  $\tilde{x_0} = \frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \sqrt{1 - \bar{\alpha}_t}\tilde{\epsilon})$ ▷*Estimated x_0 from estimated noise*
  $\tilde{\mu} = \mu_t(x_t, \tilde{x_0}) \left(= \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, t)\right)\right)$ ▷*Mean for previous step sampling*
  $x_{t-1} = \tilde{\mu} + \sigma_t z$ ▷*Previous step sampling*
end for
return x_0

DDIM Overview

Random Sampling: In DDPM the inference step involves stochastic (random) sampling to reverse the diffusion process.

$$\underbrace{\tilde{x}_{t-1} = \mu_\theta(\tilde{x}_t, t)}_{\text{denoising function}} + \underbrace{\sigma_t}_{\text{noise schedule}}(z_t - \underbrace{\epsilon_\theta(\tilde{x}_t, t)}_{\text{estimated noise}})$$

DDIM Overview

Deterministic Sampling: DDIM uses a non-Markovian process allowing for fewer timesteps.It modifies the inference step by modifying the reverse process, making the process deterministic.

$$\boldsymbol{x}_{t-1} = \sqrt{\alpha_{t-1}} \underbrace{\left(\frac{\boldsymbol{x}_t - \sqrt{1-\alpha_t}\epsilon_\theta^{(t)}(\boldsymbol{x}_t)}{\sqrt{\alpha_t}} \right)}_{\text{" predicted } \boldsymbol{x}_0 \text{"}} + \underbrace{\sqrt{1-\alpha_{t-1}-\sigma_t^2} \cdot \epsilon_\theta^{(t)}(\boldsymbol{x}_t)}_{\text{"direction pointing to } \boldsymbol{x}_t \text{"}} + \underbrace{\sigma_t \epsilon_t}_{\text{random noise}}$$

Poll 3

DDIM is 10-50× faster than DDPM because:

- A) It uses a smaller neural network
- B) It skips timesteps intelligently
- C) It trains faster
- D) It uses a better GPU

Poll 3

DDIM is 10-50× faster than DDPM because:

- A) It uses a smaller neural network
- B) It skips timesteps intelligently
- C) It trains faster
- D) It uses a better GPU

Outline

- Definition of Diffusion
- Importance of Diffusion
- Explaining the process of diffusion
- Denoising Diffusion Implicit Models (DDIM)
- Diffusion Models from Stochastic Differential Equations and Score Matching Perspective
- Classifier-free Guidance (CFG)
- Performance Metrics: FID, IS, Precision and Recall
- Applications of Diffusion Models

Why Move to Continuous Time?

Discrete View (DDPM):

- Fixed timesteps: $t = 1, 2, 3, \ldots, 1000$
- Fixed noise schedule: $\beta_1, \beta_2, \ldots, \beta_{1000}$
- Limited flexibility

Continuous View (SDE):

- Continuous time: $t \in [0, T]$
- Smooth noise function: $\beta(t)$
- Can discretize flexibly at sampling time

From ODEs to SDEs - Modeling Diffusion

- Diffusion Models (Discrete View)
  - Forward: Add noise in T discrete steps (e.g., T=1000)
  - Backward: Remove noise in T discrete steps

Problem:

- Can't flexibly adjust ***steps*** during generation
- Limited flexibility in how we add/remove noise
- Can't use powerful continuous-time solvers

From ODEs to SDEs - Modeling Diffusion

Why SDEs for Diffusion Models?

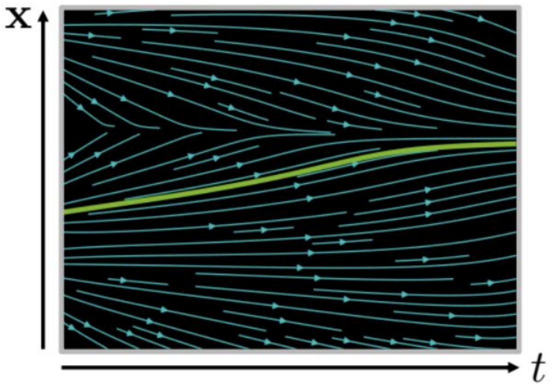
- Deterministic Sampling via Probability Flow ODEs
  - Derive equivalent ODE from any diffusion SDE (no randomness in generation)
  - Enable exact reconstruction, smooth interpolation, and image editing

- Unified Framework
  - DDPM, score-based, and flow models as special cases of one SDE
  - Transfer insights: train with one method, sample with another

Ordinary Differential Equations

Ordinary Differential Equation (ODE):

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}, t) \quad \text{or} \quad \mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t$$



Analytical Solution:

$$\mathbf{x}(t) = \mathbf{x}(0) + \int_0^t \mathbf{f}(\mathbf{x}, \tau)\mathrm{d}\tau$$

Iterative Numerical Solution:

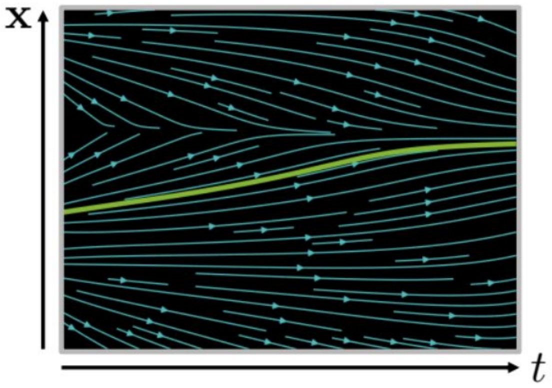
$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t$$

Slide credit to: https://cvpr2022-tutorial-diffusion-models.github.io/

Stochastic Differential Equations

Ordinary Differential Equation (ODE):

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \mathbf{f}(\mathbf{x}, t) \quad \text{or} \quad \mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t$$



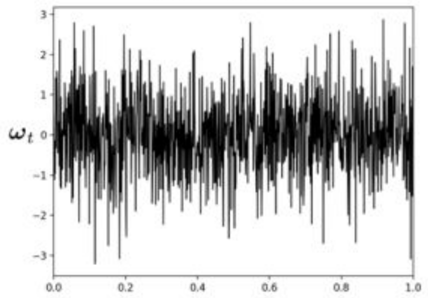
Analytical Solution: $\mathbf{x}(t) = \mathbf{x}(0) + \int_0^t \mathbf{f}(\mathbf{x}, \tau)\mathrm{d}\tau$

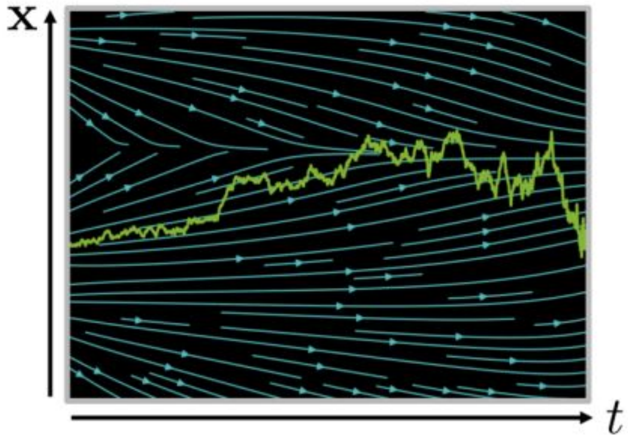
Iterative Numerical Solution: $\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t$

Stochastic Differential Equation (SDE):

$$\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} = \underbrace{\mathbf{f}(\mathbf{x}, t)}_{\text{drift coefficient}} + \underbrace{\sigma(\mathbf{x}, t)}_{\text{diffusion coefficient}}\boldsymbol{\omega}_t \longleftarrow \text{Wiener Process (Gaussian White Noise)}$$

$$\left(\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + \sigma(\mathbf{x}, t)\mathrm{d}\boldsymbol{\omega}_t \right)$$

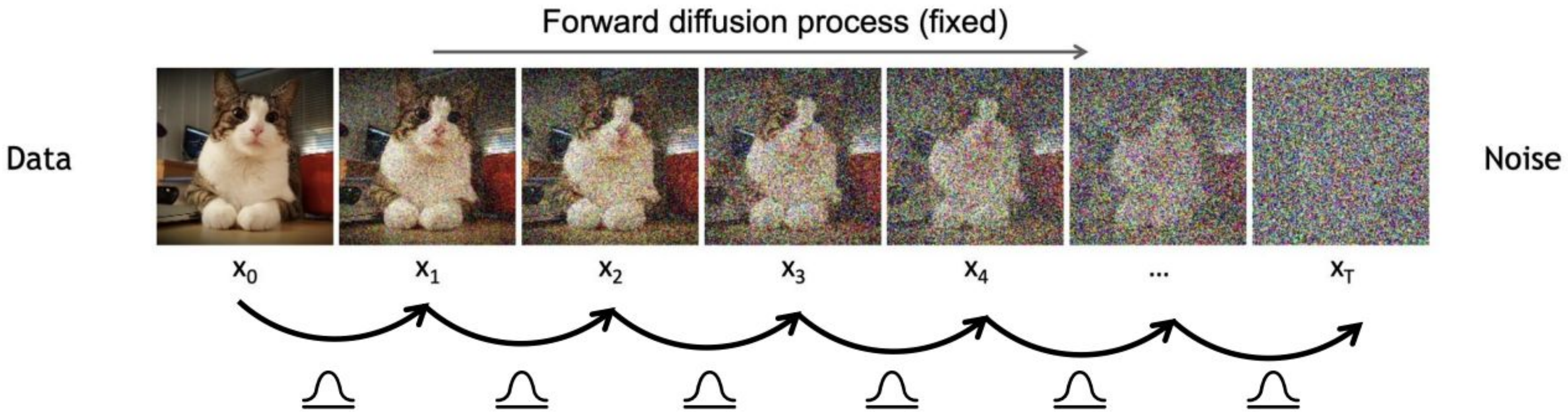




$$\mathbf{x}(t + \Delta t) \approx \mathbf{x}(t) + \mathbf{f}(\mathbf{x}(t), t)\Delta t + \sigma(\mathbf{x}(t), t)\sqrt{\Delta t}\,\mathcal{N}(\mathbf{0}, \mathbf{I})$$

Slide credit to: https://cvpr2022-tutorial-diffusion-models.github.io/

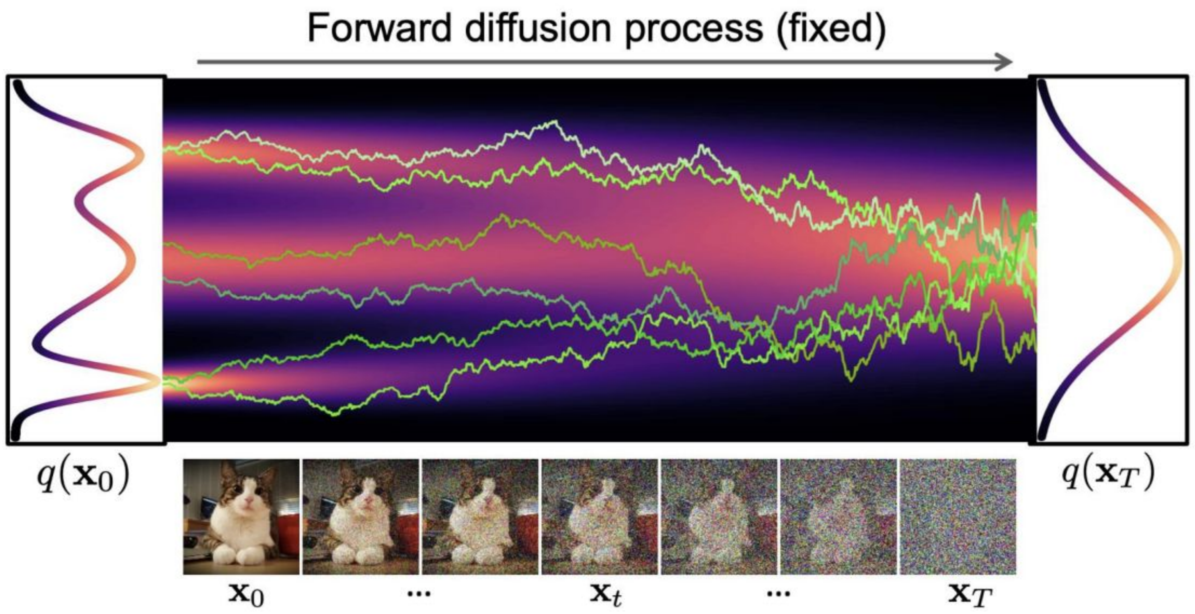
Forward Diffusion Process as SDEs



$$\mathrm{d}\mathbf{x}_t = \underbrace{-\frac{1}{2}\beta(t)\mathbf{x}_t \,\mathrm{d}t}_{\substack{\text{drift term} \\ \text{(pulls towards mode)}}} + \underbrace{\sqrt{\beta(t)}\,\mathrm{d}\boldsymbol{\omega}_t}_{\substack{\text{diffusion term} \\ \text{(injects noise)}}}$$

Slide credit to: https://cvpr2022-tutorial-diffusion-models.github.io/

Forward Diffusion Process as SDEs



$$\mathrm{d}\mathbf{x}_t = \underbrace{-\frac{1}{2}\beta(t)\mathbf{x}_t\,\mathrm{d}t}_{\substack{\text{drift term} \\ \text{(pulls towards mode)}}} + \underbrace{\sqrt{\beta(t)}\,\mathrm{d}\boldsymbol{\omega}_t}_{\substack{\text{diffusion term} \\ \text{(injects noise)}}}$$

Slide credit to: https://cvpr2022-tutorial-diffusion-models.github.io/

Forward Diffusion Process as SDEs

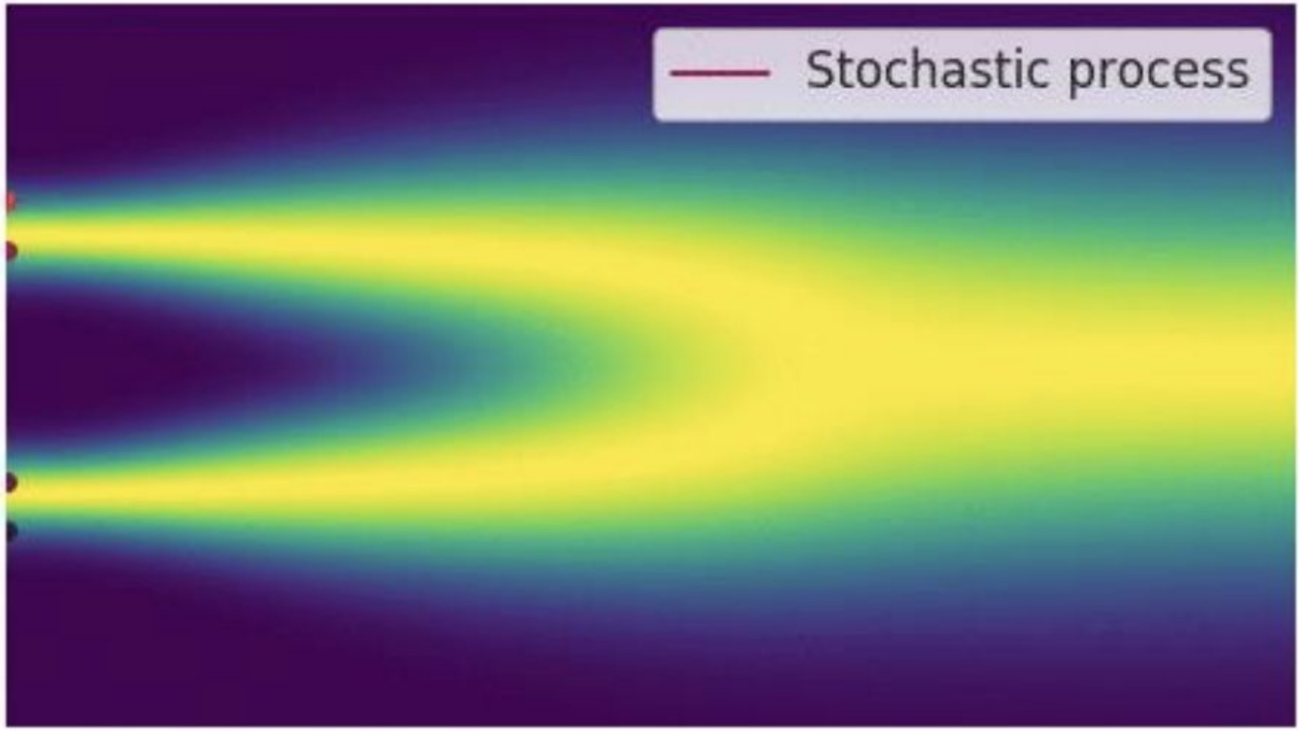
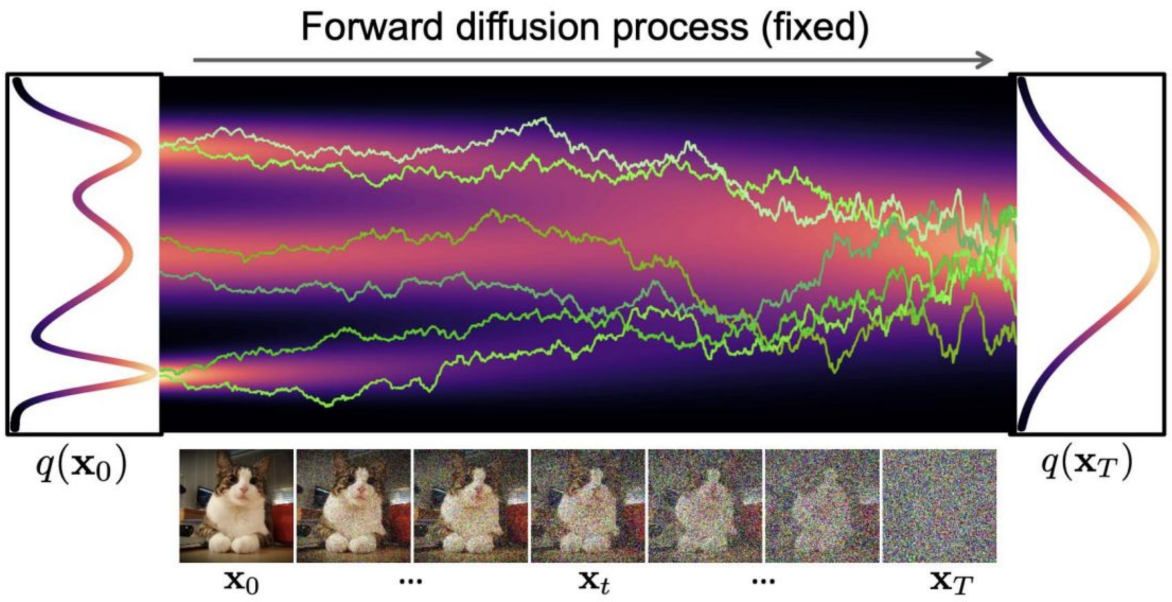


Figure credit to: https://yang -song.net/blog/2021/score/

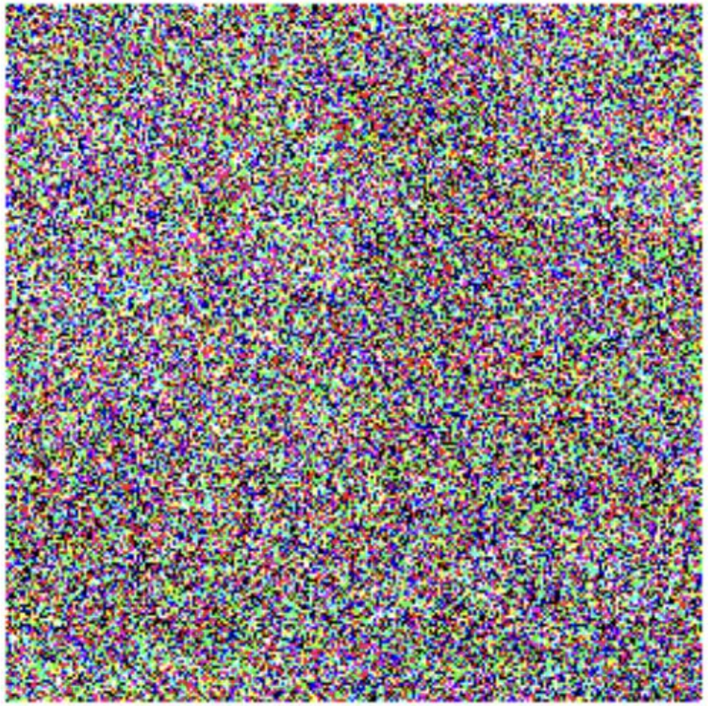
Generative Reverse SDEs



$$\mathrm{d}\mathbf{x}_t = \overbrace{\left[-\frac{1}{2}\beta(t)\mathbf{x}_t - \beta(t)\underbrace{\nabla_{\mathbf{x}_t}\log q_t(\mathbf{x}_t)}_{\text{“Score Function”}}\right]}^{\text{drift term}}\mathrm{d}t + \overbrace{\sqrt{\beta(t)}\,\mathrm{d}\bar{\boldsymbol{\omega}}_t}^{\text{diffusion term}}$$

Slide credit to: https://cvpr2022-tutorial-diffusion-models.github.io/

Generative Reverse SDEs



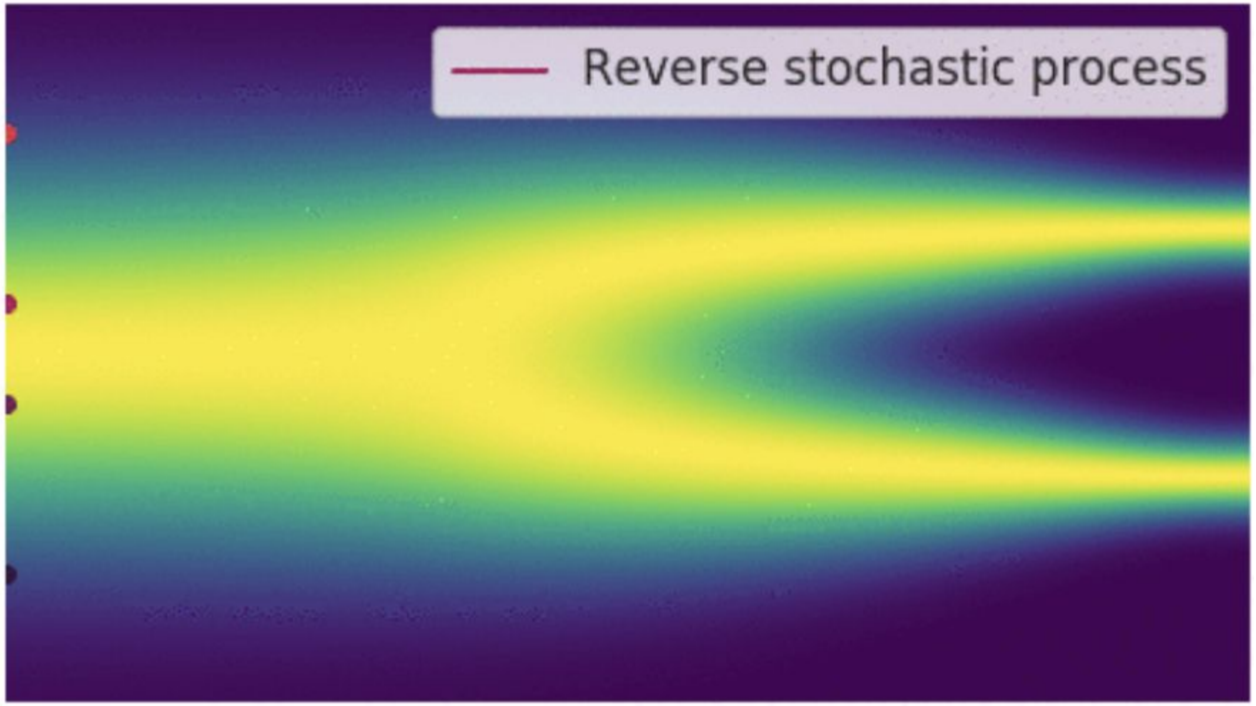


Figure credit to: https://yang -song.net/blog/2021/score/

The Score Function: Why It Matters

Traditional Approach:

Learn $p(x)$ directly → requires normalizing constant Z → Z is intractable for complex data

Score-Based Approach:

Learn $\nabla_x \log p(x)$ instead → no Z needed! → Z cancels in gradient

The Score Function: Why It Matters

Connection to Diffusion:

Predicting noise ε ⟺ Estimating score ∇_x log p(x)

Same objective, different interpretation

Score tells us direction toward high probability

Score Matching

- General form of probability density function

$$p_\theta(\mathbf{x}) = \frac{e^{-f_\theta(\mathbf{x})}}{Z_\theta}$$

- Maximizing the log-likelihood requires us to know Z_θ
  - – Often intractable

- Instead, we can model the score function

$$\nabla_{\mathbf{x}} \log p(\mathbf{x})$$

Slide credit to: Hao Chen https://deeplearning.cs.cmu.edu/S25/document/slides/lec23.diffusion_s25.pdf

Denoising Score Matching

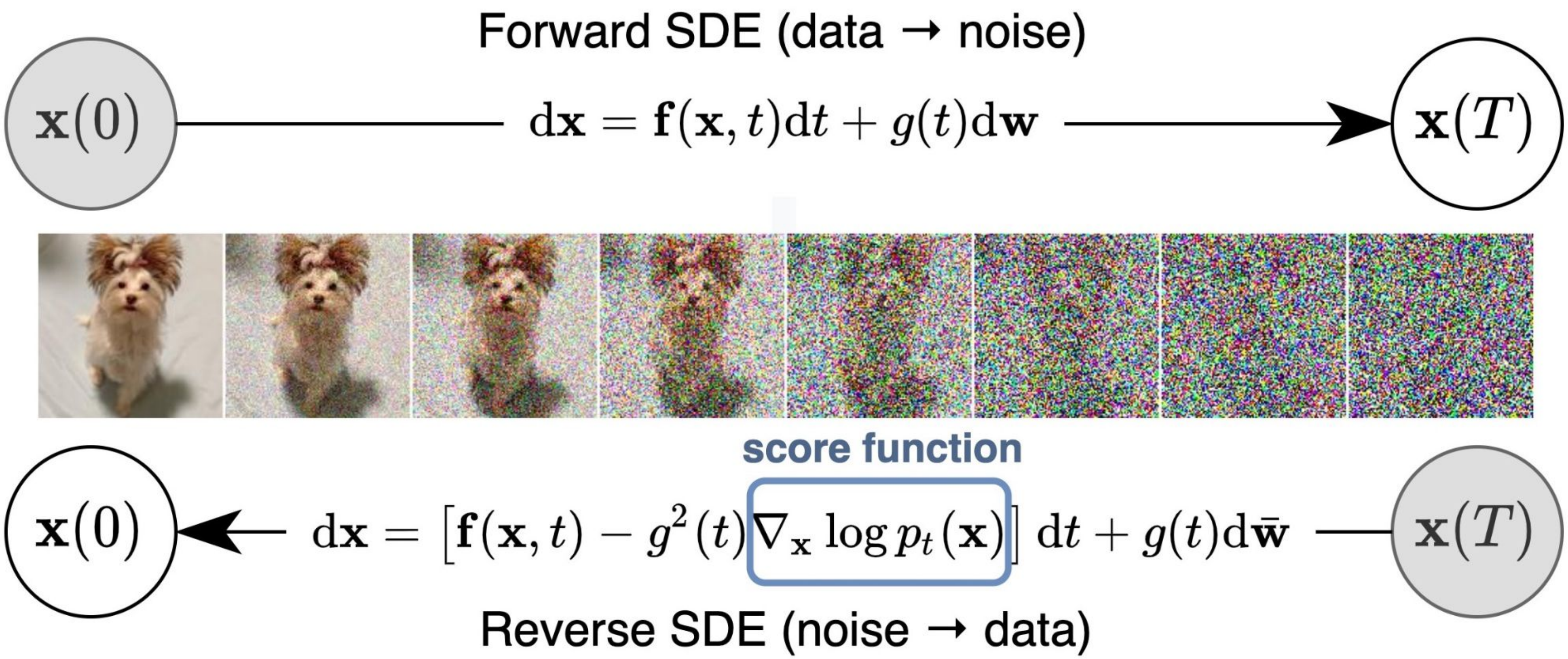


Figure credit to: https://yang-song.net/blog/2021/score/

Weighted Diffusion Objective

Denoising score matching objective with loss weighting

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{t\sim\mathcal{U}(0,T)}\mathbb{E}_{\mathbf{x}_0\sim q_0(\mathbf{x}_0)}\mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})} \frac{\lambda(t)}{\sigma_t^2}\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\mathbf{x}_t, t\right)\right\|_2^2$$

Problem: AND Why weighting λ(t)?

- Different timesteps contribute differently
- Early timesteps (high noise): easy to predict, less important
- Late timesteps (low noise): hard to predict, very important

Common choices:

- λ(t) = 1: Simple, uniform weighting (DDPM)
- λ(t) = σ_t²: Better perceptual quality
- λ(t) = β(t): Maximum likelihood

Poll 4

The score function $\nabla \log p(x)$ points toward:

- A) Random directions
- B) Low probability regions
- C) High probability regions (data)
- D) The origin

Poll 4

The score function $\nabla \log p(x)$ points toward:

- A) Random directions
- B) Low probability regions
- C) High probability regions (data)
- D) The origin

Outline

- Definition of Diffusion
- Importance of Diffusion
- Explaining the process of diffusion
- Denoising Diffusion Implicit Models (DDIM)
- Diffusion Models from Stochastic Differential Equations and Score Matching Perspective
- Classifier-free Guidance (CFG)
- Performance Metrics: FID, IS, Precision and Recall
- Applications of Diffusion Models

Conditional Generation: The Control Problem

Unconditional Diffusion:

- Sample: Random noise → Random image from training distribution
- No control over output class/content

The Need for Control:

- "Draw a sunset over mountains"
- "Create a portrait in Van Gogh style"

Conditional Generation: The Control Problem

1. **Classifier Guidance:** Train separate classifier, use gradients
   - ✗ Requires training extra model
   - ✗ Classifier can be noisy at high *t*
2. **Classifier-Free Guidance (CFG):** Train one model for both
   - ✓ Single model
   - ✓ Better quality

Classifier-free Guidance (CFG)

Limited Class Control: Previous method is incapable of generating an image for a given class

Purpose of CFG: it allows for targeted generation of images by conditioning the model on a specific class label.

How Classifier-Free Guidance Works

Training:

- Randomly drop condition 10-20% of time
- Model learns both:
  - $\epsilon\_\theta(x\_t, \text{class}) \leftarrow$ conditional
  - $\epsilon\_\theta(x\_t, \varnothing) \leftarrow$ unconditional

Sampling:

$$\tilde{\varepsilon} = \varepsilon\_\text{uncond} + \gamma(\varepsilon\_\text{cond} - \varepsilon\_\text{uncond})$$

$$\underset{\text{baseline}}{\uparrow} \qquad \underset{\text{scale}}{\uparrow} \quad \underset{\text{direction toward class}}{\uparrow}$$

Guidance scale γ:

- $\gamma = 0$: Ignore condition (random)
- $\gamma = 1$: Standard conditioning
- $\gamma > 1$: Stronger conditioning (more typical of class)

Classifier-free Guidance (CFG)

Algorithm 1 Classifier guided diffusion sampling, given a diffusion model $(\mu_\theta(x_t), \Sigma_\theta(x_t))$, classifier $p_\phi(y|x_t)$, and gradient scale s.

Input: class label y, gradient scale s
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for all t from T to 1 **do**
  $\mu, \Sigma \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$
  $x_{t-1} \leftarrow$ sample from $\mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log p_\phi(y|x_t), \Sigma)$
end for
return x_0

Outline

- Definition of Diffusion
- Importance of Diffusion
- Explaining the process of diffusion
- Denoising Diffusion Implicit Models (DDIM)
- Diffusion Models from Stochastic Differential Equations and Score Matching Perspective
- Classifier-free Guidance (CFG)
- Performance Metrics: FID, IS, Precision and Recall
- Applications of Diffusion Models

Performance Metrics

Fréchet Inception Distance (FID)

Measures the distance between feature vectors of real and generated images; lower scores indicate better image quality and similarity to real images.

Inception Score (IS)

Assesses the diversity and clarity of generated images using a pre-trained model; higher scores denote better image clarity and variety.

Precision and Recall

Evaluates the quality and diversity of generated images; high precision indicates realistic images, while high recall shows variety close to the actual dataset.

Outline

- Definition of Diffusion
- Importance of Diffusion
- Explaining the process of diffusion
- Denoising Diffusion Implicit Models (DDIM)
- Diffusion Models from Stochastic Differential Equations and Score Matching Perspective
- Classifier-free Guidance (CFG)
- Performance Metrics: FID, IS, Precision and Recall
- Applications of Diffusion Models

Latent DDPM

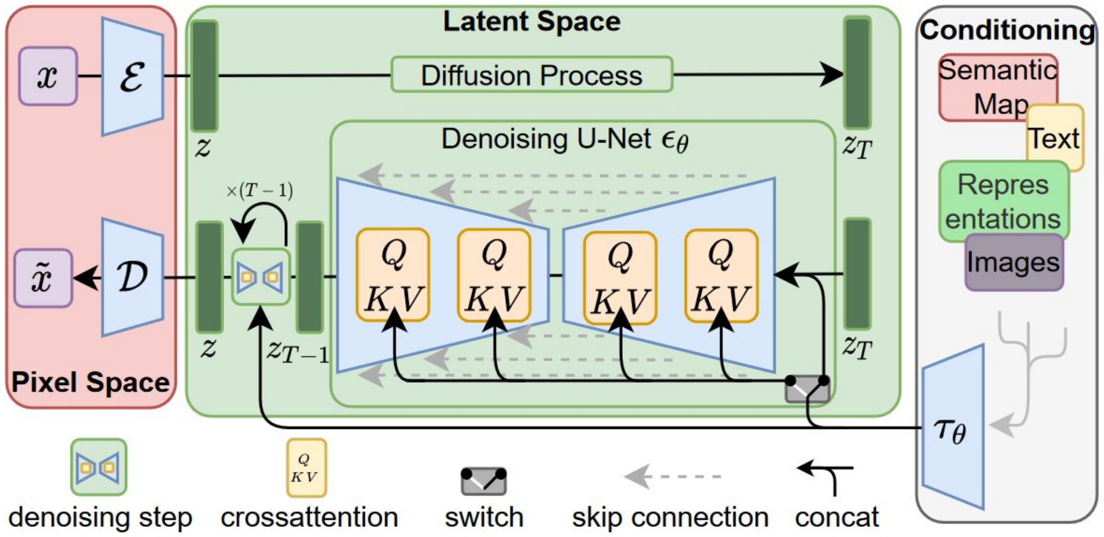
Why Latent Space?

- **Efficiency:** Running diffusion models directly in pixel space is computationally expensive. Latent Diffusion Models (LDMs) operate in a compressed latent space, drastically reducing computational cost and complexity.
- **Data Compression:** LDMs capture the essential ***features*** of high-dimensional data like images in a more compact and structured form, enhancing processing efficiency.
- **Improved Performance:** Working in latent space focuses the model on **relevant aspects** of the *data*, improving both the efficiency and effectiveness of the generation process.

Latent DDPM

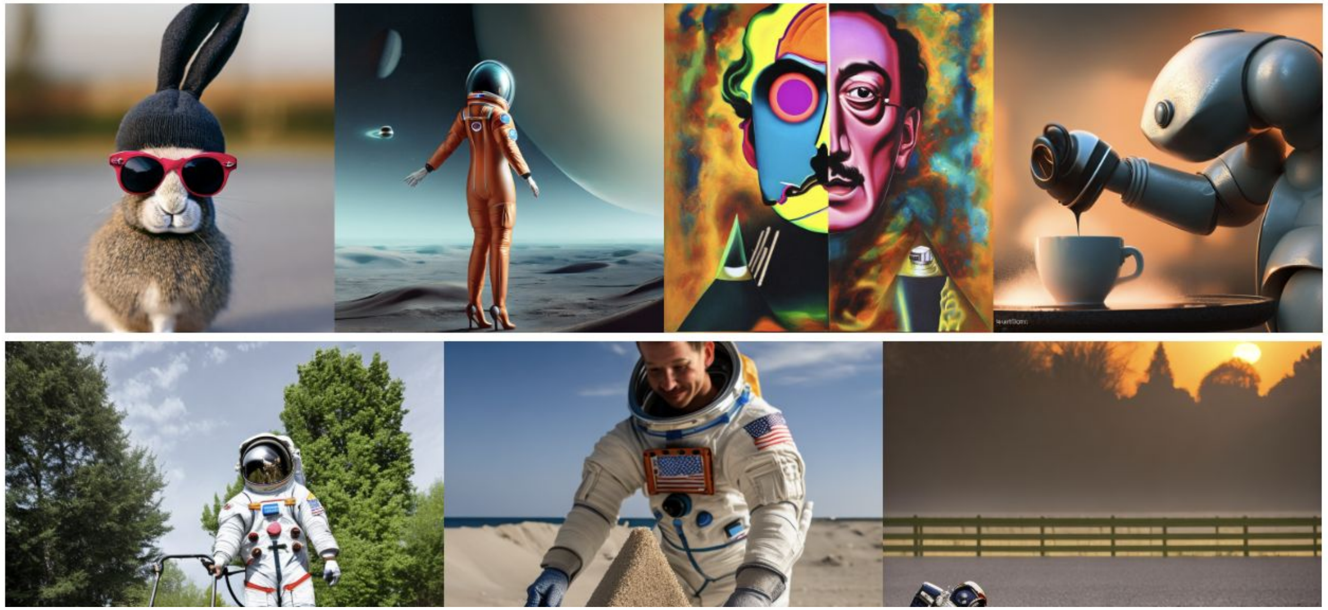
- Map data into latent space using VAE or any other similar approach

1. **Encoder:** Compresses input data into a latent representation.
2. **Decoder:** Reconstructs the original data from the latent representation.



Stable Diffusion

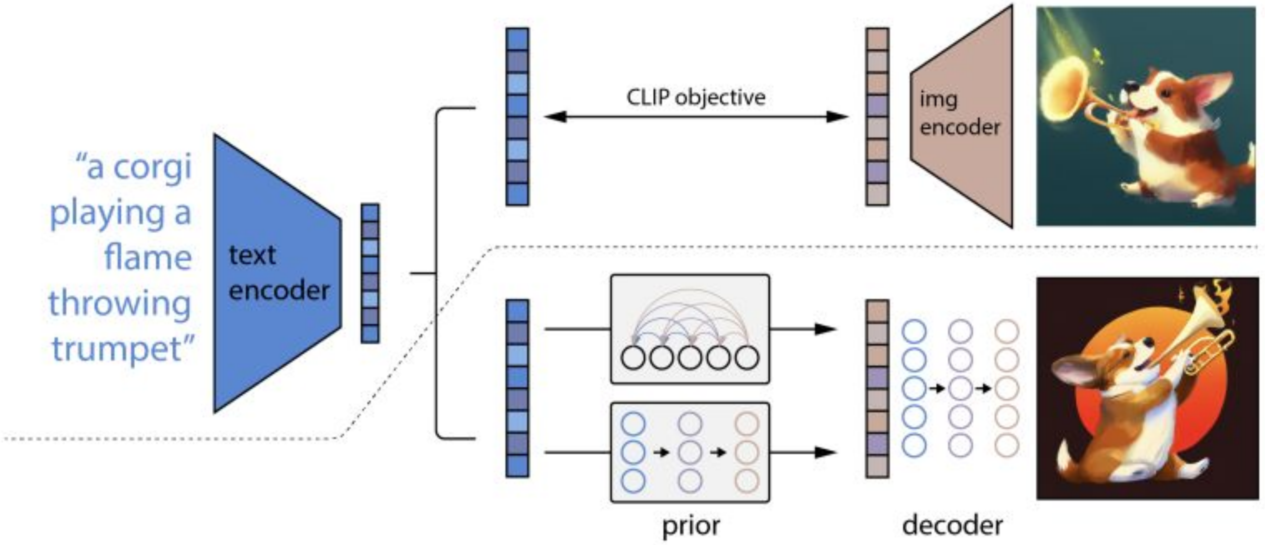
- Stable Diffusion is a latent text-to-image diffusion model.



Stability AI. https://github.com/Stability-AI/stablediffusion

DALLE

- DALLE is a text-to-image generative model that creates coherent, high-resolution images from natural-language prompts.



Ramesh et al. Hierarchical Text-Conditional Image Generation with CLIP Latents

DiT

- DiT (Diffusion Transformer) is a diffusion model architecture that replaces U-Nets with Transformers to more effectively denoise and generate high-quality images.

Peebles et al. Scalable Diffusion Models with Transformers. 2020.

MAR

- An autoregressive model with diffusion loss

Li et al. Autoregressive Image Generation without Vector Quantization. 2024.

Key Takeaways

1. Many Small Steps Beat One Big Jump

VAEs try to generate in one step and fail (blurry images). Diffusion uses 1000 tiny steps instead. Each small change is easy to learn. Result: sharp, high-quality outputs.

2. Forward is Fixed, Reverse is Learning

Forward process: add noise using a fixed equation (no training needed). Reverse process: train neural network to predict the noise. Training objective: MSE loss.

3. Three Views, Same Model

Denoising, score-based, and SDEs are equivalent perspectives. Understanding one helps understand all.

Thank you :-)