Diffusion Models

Diffusion Models

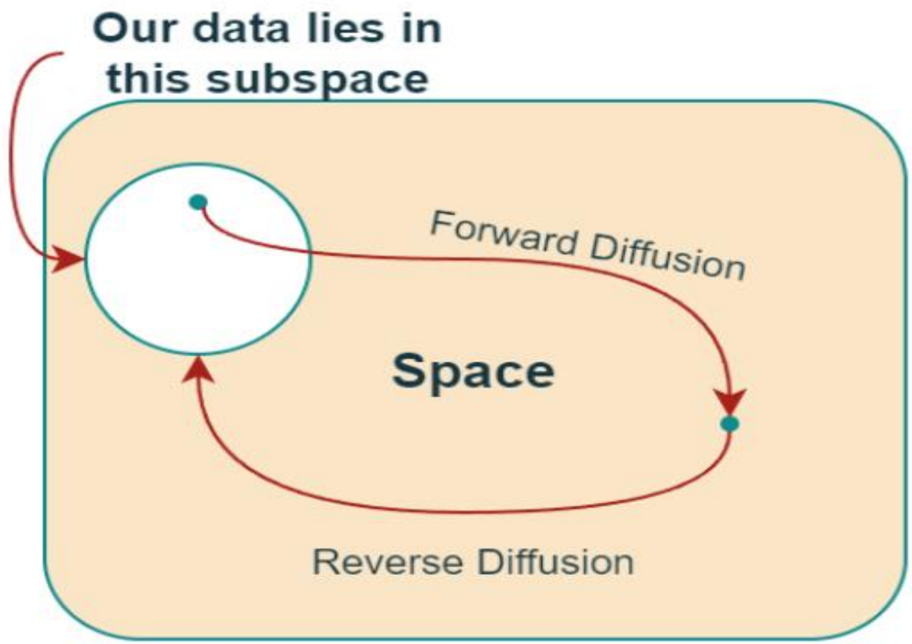
A hedgehog using a calculator.

A corgi wearing a red bowtie and a purple

A transparent sculpture- a duck made out of glas

High-level overview

- Diffusion models are probabilistic models used for image generation
- They involve reversing the process of gradually degrading the data
- Consist of two processes:
  - **The forward process**: data is progressively destroyed by adding noise across multiple time steps
  - **The reverse process**: using a neural network, noise is sequentially removed to obtain the original data



A high-level conceptual overview of the entire image space.

Three Categories

- Denoising Diffusion Probabilistic Models (DDPM)
- Noise Conditioned Score Networks (NCSN)
- Stochastic Differential Equations (SDE)

Notations

$p(x_0)$ - data distribution

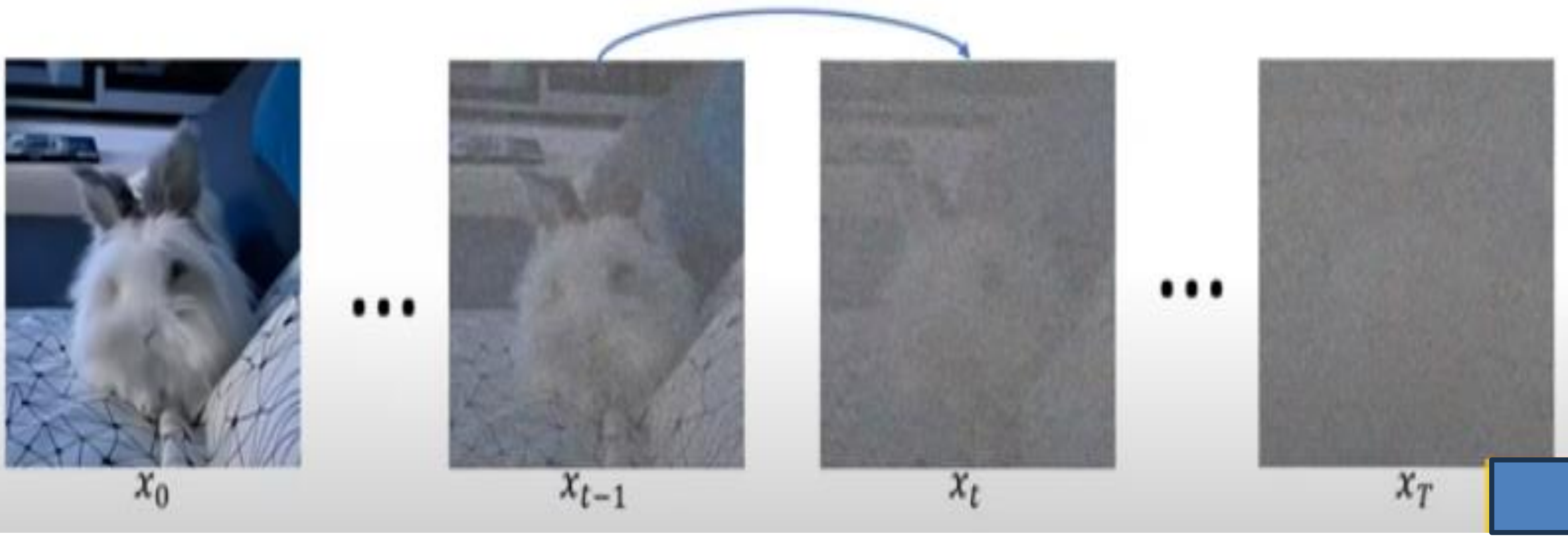
$\mathcal{N}(x; \mu, \sigma \cdot I)$ - Gaussian distribution

Random Variable (image) Mean Vector Covariance matrix. I is the identity matrix

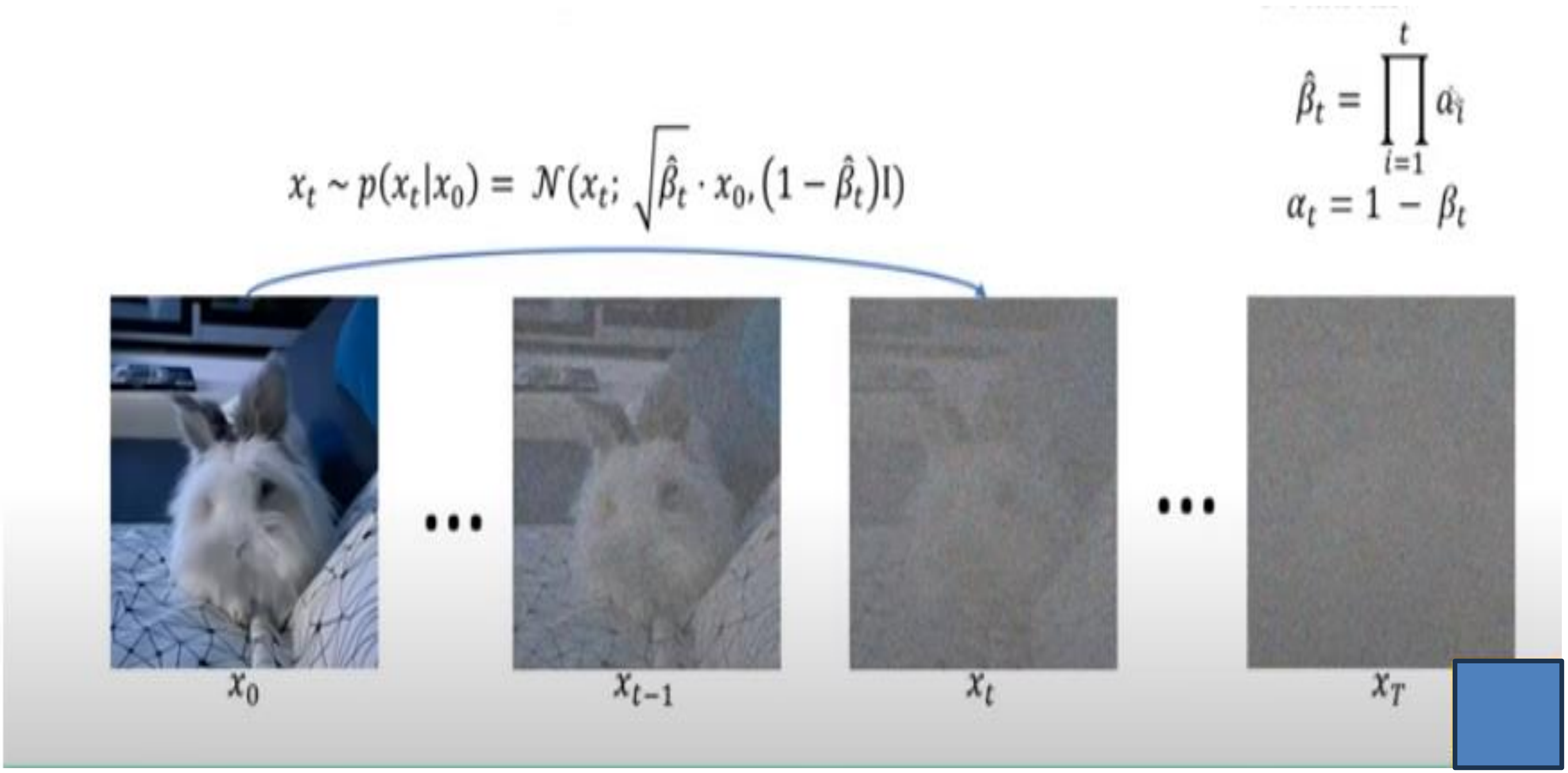
Forward Process (Iterative)

$$\beta_t \ll 1\,,\, t = \overline{1,T}$$

$$x_t \sim p(x_t | x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t} \cdot x_{t-1}, \beta_t \mathrm{I})$$



Forward Process (One Shot)

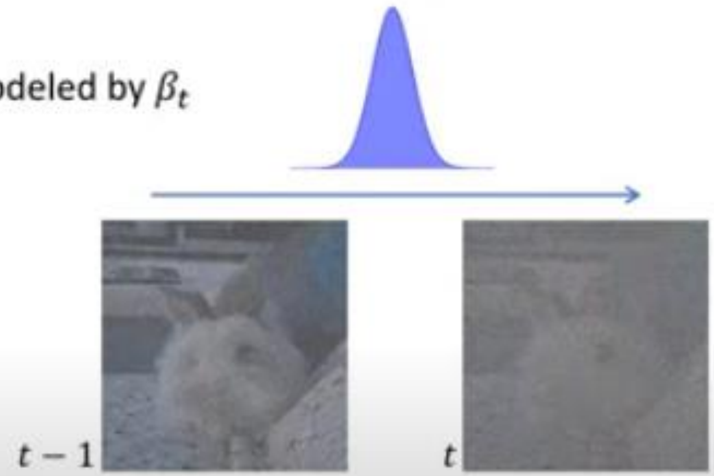


DDPMs. Properties of β_t

✓ 1. $\beta_t \ll 1\ ,t = \overline{1,T}$

$$x_t \sim p(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t} \cdot x_{t-1}, \beta_t \mathrm{I})$$

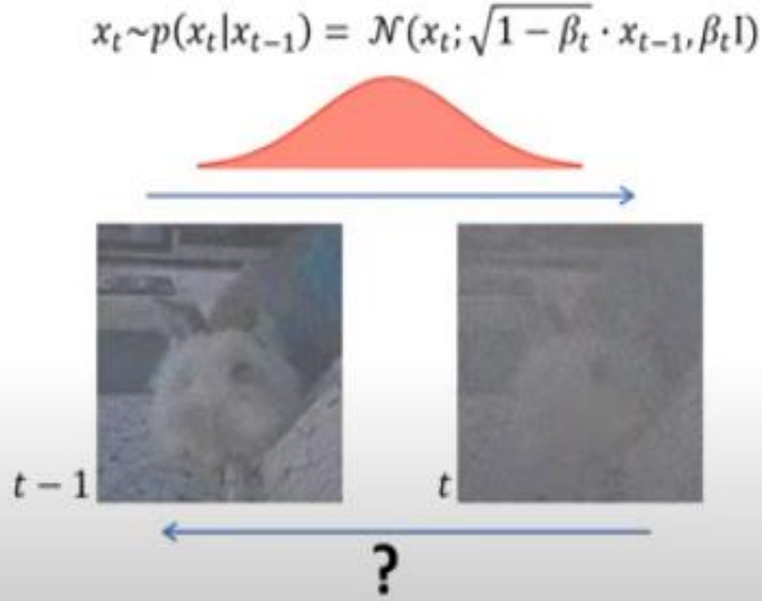
x_t is created with a small step modeled by β_t



x_{t-1} comes from region close to x_t,
therefore we can model with Gaussian

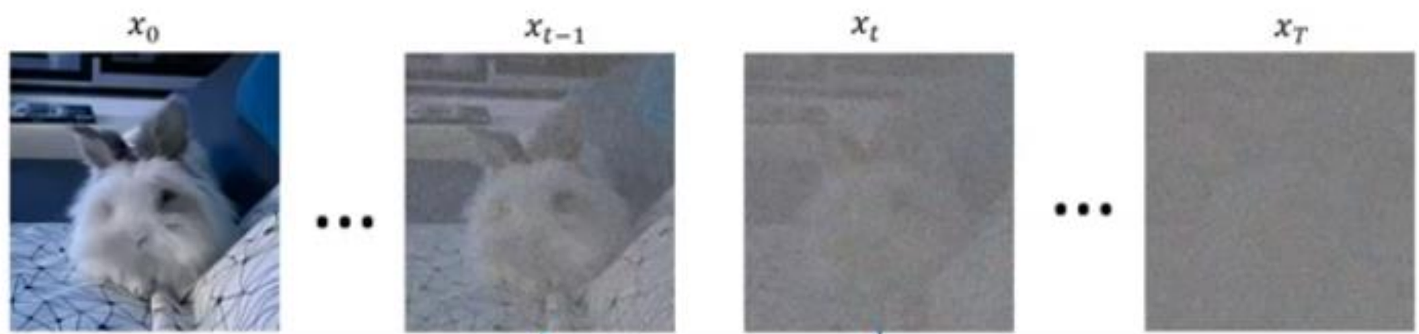
DDPMs. Properties of β_t

❌ 1. $\beta_t \ll 1$, $t = \overline{1,T}$



DDPMs. Training objective

Remember that:



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

Reverse process

Neural Network

Cross Entropy and KL (Kullback-Leibler) divergence

- Entropy: $E(P) = - \Sigma_i P(i) \log P(i)$
- Cross Entropy: $C(P) = - \Sigma_i P(i) \log Q(i)$
- KL divergence: $D_{KL}(P \,||\, Q) = \Sigma_i P(i) \log[P(i)/Q(i)] = \Sigma_i P(i)[\log P(i) - \log Q(i)]$

DDPMs. Training Objective

$$\min_{\theta} \mathbb{E}_{x_0 \sim p(x_0)} \left[-\log p_\theta(x_0|x_1) + KL(p(x_T|x_0)||p_\theta(x_T)) + \sum_{t=2}^{T} KL(p(x_{t-1}|x_t,\ x_0)||p_\theta(x_{t-1}|x_t)) \right]$$

$$\min_{\theta} \mathbb{E}_{x_0 \sim p(x_0)} \left[-\log p_\theta(x_0|x_1) + \sum_{t=2}^{T} KL(p(x_{t-1}|x_t,\ x_0)||p_\theta(x_{t-1}|x_t)) \right]$$

Notations:

$$\hat{\beta}_t = \prod_{i=1}^{t} \alpha_i$$

$$\alpha_t = 1 - \beta_t$$

$$p(x_{t-1}|x_t, x_0) = \mathcal{N}\left(x_{t-1};\ \tilde{\mu}(x_t, x_0), \tilde{\beta}_t I\right)$$

$$\tilde{\mu}(x_t, x_0) = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \hat{\beta}_t}} z_t \right), z_t \sim \mathcal{N}(0, \mathrm{I})$$

$$\tilde{\beta}_t = \frac{1 - \hat{\beta}_{t-1}}{1 - \hat{\beta}_t} \cdot \beta_t$$

$$\min_{\theta} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_{x_0 \sim p(x_0), z_t \sim \mathcal{N}(0,\mathrm{I})} \left\| z_t - z_\theta(x_t, t) \right\|_2^2$$

DDPMs. Training Algorithm

$$\min_{\theta} \underbrace{\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{x_0 \sim p(x_0), z_t \sim \mathcal{N}(0,\mathrm{I})} \|z_t - z_\theta(x_t, t)\|_2^2}_{\mathcal{L}_{simple}}$$

Training algorithm:

$$\hat{\beta}_t = \prod_{i=1}^{t} \alpha_i$$

Repeat

$x_0 \sim p(x_0)$

$t \sim \mathcal{U}(\{1, \dots, T\})$

$z_t \sim \mathcal{N}(0, \mathrm{I})$

$x_t = \sqrt{\hat{\beta}_t} \cdot x_0 + \sqrt{(1 - \hat{\beta}_t)} z_t$

$\theta = \theta - lr \cdot \nabla_\theta \mathcal{L}_{simple}$

Until convergence

DDPMs. Training Algorithm

$$\min_\theta \underbrace{\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{x_0\sim p(x_0), z_t\sim\mathcal{N}(0,\mathrm{I})} \|z_t - z_\theta(x_t, t)\|_2^2}_{\mathcal{L}_{simple}}$$

$$\hat{\beta}_t = \prod_{i=1}^{t} \alpha_i$$

Training algorithm:

Repeat

$x_0 \sim p(x_0)$ %We sample an image from our data set

$t \sim \mathcal{U}(\{1, \ldots, T\})$ %choose randomly a time step t of the forward process

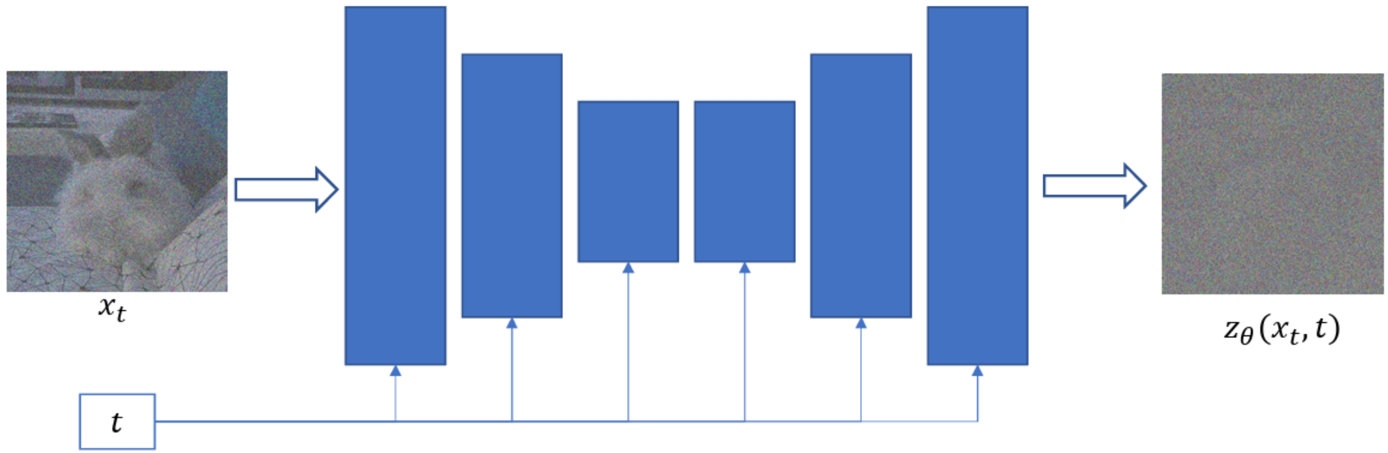
$z_t \sim \mathcal{N}(0, \mathrm{I})$ %sample the noise z_t

$x_t = \sqrt{\hat{\beta}_t} \cdot x_0 + \sqrt{(1-\hat{\beta}_t)} z_t$ % Get noisy image

$\theta = \theta - lr \cdot \nabla_\theta \mathcal{L}_{simple}$ %Update neural network weights

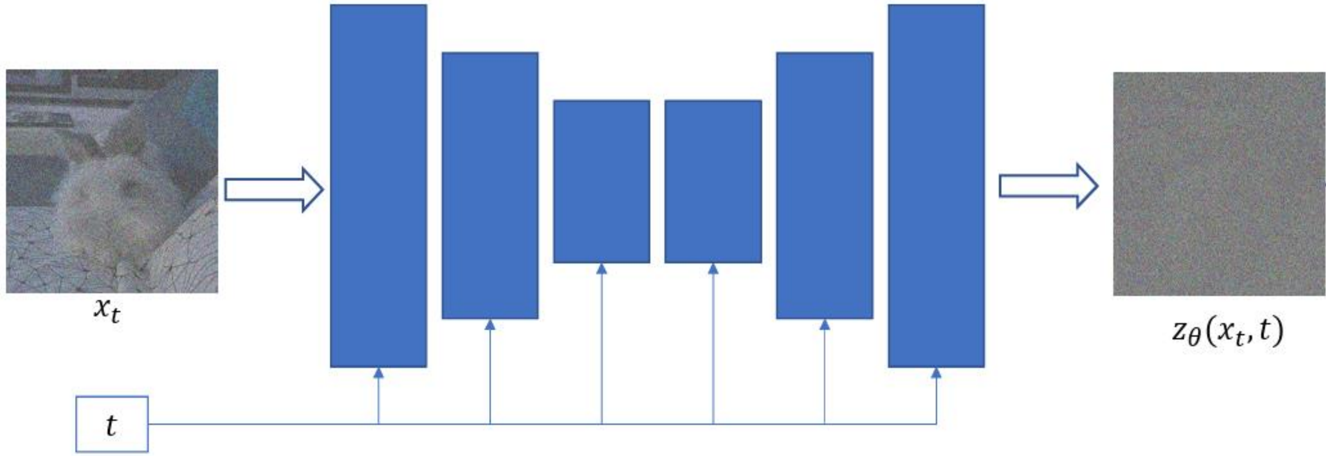
Until convergence

DDPMs. Sampling



- Pass the current noisy image along with t to the neural network
- With the resultant z_θ compute the mean of the gaussian distribution

DDPMs. Sampling



Sample the image x_{t-1} for the next iteration

x_{t-1}

$$\sim \mathcal{N}\left(x_{t-1}, \overbrace{\frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \hat{\beta}_t}} z_\theta(x_t, t)\right)}^{\mu_\theta(x_t, t)}, \sigma_t^2 \mathrm{I}\right)$$

Three Categories

- Denoising Diffusion Probabilistic Models (DDPM)
- Noise Conditioned Score Networks (NCSN)
- Stochastic Differential Equations (SDE)

Score Function

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

- Direction we need to change the input x such that the density becomes greater

Score Function

- Second formulation of Diffusion Model

- Langevin dynamics method
  - Starts from a random sample
  - Apply iterative updates with the score function to modify the sample
  - Result will have a higher chance of being a sample of the true distribution p(x)

Naïve score-based model

- **Score**: gradient of the logarithm of the probability density with respect to the input

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

- Langevin dynamics

$$x_i = x_{i-1} + \frac{\gamma}{2}\nabla_x \log p(x) + \sqrt{\gamma} \cdot \omega_i$$

Step size – controls the magnitude of the update in the direction of the score

Score – estimated by the score network

Noise – random gaussian noise N(0, I)

Naïve score-based model

- The score is approximated with a neural network
- Score network is trained using score matching

$$\mathbb{E}_{x \sim p(x)} \| s_\theta(x) - \nabla_x \log p(x) \|_2^2$$

- Denoising score matching:
  - ➢ Add small noise to each sample of the data:

$$x' \sim \mathcal{N}(x', x, \sigma \cdot I) = p_\sigma(x')$$

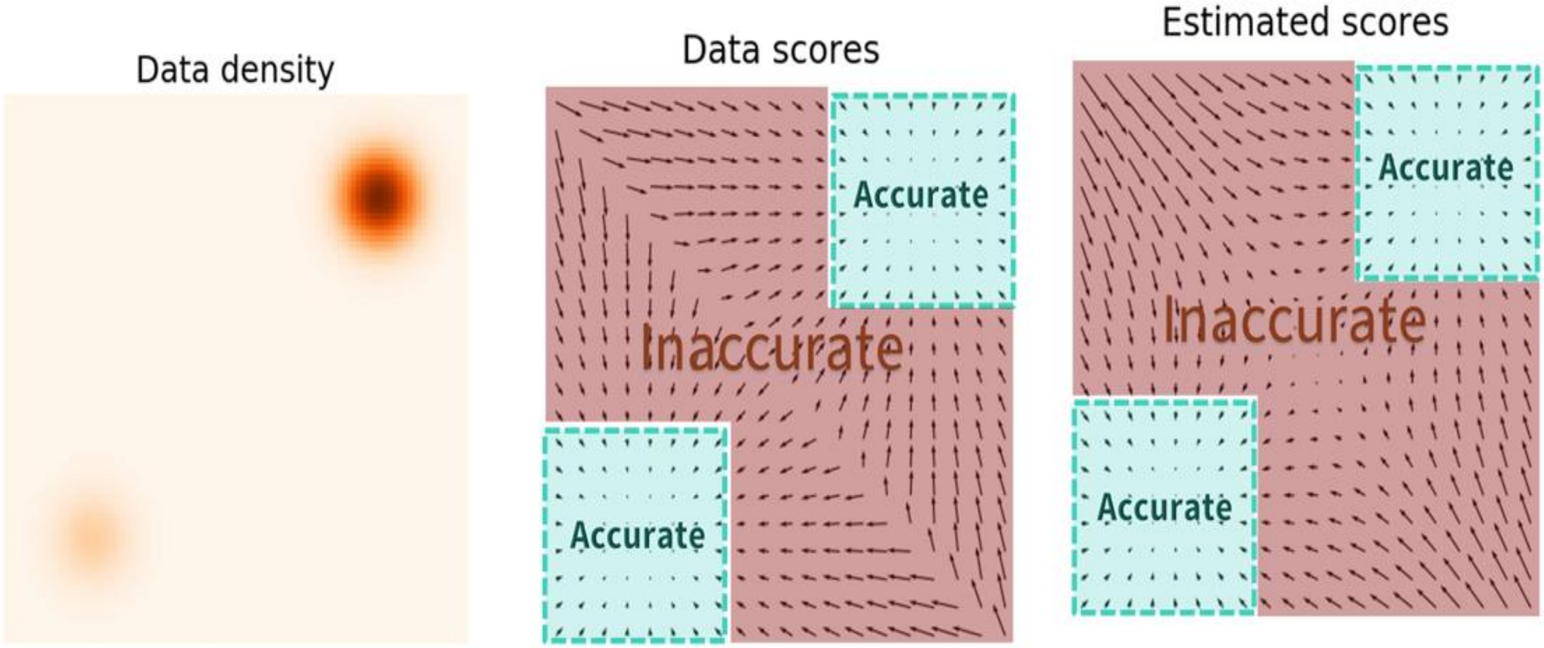
  - ➢ Objective

$$\mathbb{E}_{x' \sim p_\sigma(x')} \| s_\theta(x') - \nabla_{x'} \log p_\sigma(x') \|_2^2$$

  - ➢ After training:

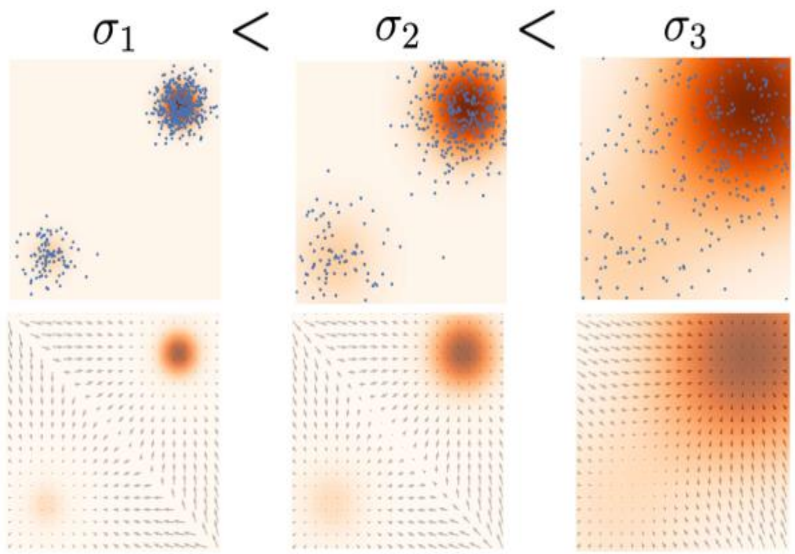
$$s_\theta(x) \approx \nabla_x \log p\ (x)$$

Naïve score-based model. Problems



Noise Conditioned Score Network (NCSNs)

- Solution:
  - ➢Perturb the data with random Gaussian noise at different scales
  - ➢Learn score estimations noisy distributions via a single score network



Credit images Yang Song: https://yang-song.net/blog/2021/score/

Noise Conditioned Score Network (NCSNs)

- Given a sequence of Gaussian noise scales $\sigma_1 < \sigma_2 < \cdots < \sigma_T$ such that:
  - $p_{\sigma_1}(x) \approx p(x_0)$
  - Approximating the true data distribution
  - $p_{\sigma_T}(x) \approx \mathcal{N}(0, I)$
  - Almost equally with the standard gaussian distribution.
- And the forward process i.e. noise perturbation given by:

$$p_{\sigma_t}(x_t | x) = \mathcal{N}(x_t; x, \sigma_t^2 \cdot I) = \frac{1}{\sigma_t \cdot \sqrt{2\pi}} \cdot \exp\left(\frac{-1}{2} \cdot \left(\frac{x_t - x}{\sigma_t}\right)^2\right)$$

- The gradient can be written as:

$$\nabla_{x_t} \log p_{\sigma_t}(x_t | x) = -\frac{x_t - x}{\sigma_t^2}$$

Noise Conditioned Score Network (NCSNs)

- Training the NCSN with denoising score matching, the following objective is minimized:

$$\mathcal{L}_{dsm} = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{p(x)}\mathbb{E}_{p_{\sigma_t}(x_t|x)} \left\| s_\theta(x_t, \sigma_t) + \frac{x_t - x}{\sigma_t^2} \right\|_2^2$$

Noise Conditioned Score Network (NCSNs)

- Training the NCSN with denoising score matching, the following objective is minimized:

$$\mathcal{L}_{dsm} = \frac{1}{T}\sum_{t=1}^{T} \lambda(\sigma_t)\mathbb{E}_{p(x)}\mathbb{E}_{p_{\sigma_t}(x_t|x)} \left\| s_\theta(x_t, \sigma_t) + \frac{x_t - x}{\sigma_t^2} \right\|_2^2$$

Weighting function

Noise Conditioned Score Network (NCSNs). Sampling

Langevin dynamics

Parameters:

N – number of iterations for Langevin dynamics

$\sigma_0 < \sigma_1 < \ldots < \sigma_T$ - noise scales

$\{\gamma_t \mid t = 1, T\}$ - update magnitude

Algorithm:

$x_T^0 \sim \mathcal{N}(0, I)$; %sample some standard gaussian noise

for t $= T, 1$ do: %start from the largest noise scale, which is denoted by the time step

  for i $= 1, N$ do: %for N iterations execute the Langevin dynamics updates

    $z_t \sim \mathcal{N}(0, I)$ % get noise

    $x_t^i = x_t^{i-1} + \frac{\gamma_t}{2} \cdot s_\theta\left(x_t^{i-1}, \sigma_t\right) + \sqrt{\gamma_t} z_t$ % update

  $x_{t-1}^0 = x_t^N$ % next iteration

return x_0

DDPM vs NCSN. Losses

DDPM: $\mathcal{L}_{simple} = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{x_0 \sim p(x_0), z_t \sim \mathcal{N}(0,\mathrm{I})} \left\| z_\theta(x_t, t) - z_t \right\|_2^2$

NCSN: $\mathcal{L}_{dsm} = \frac{1}{T}\sum_{t=1}^{T} \lambda(\sigma_t) \mathbb{E}_{x_0 \sim p(x_0),\, x_t \sim p_{\sigma_t}(x_t | x_0)} \left\| s_\theta(x_t, \sigma_t) + \frac{x_t - x_0}{\sigma_t^2} \right\|$

We can rewrite the noise z_t, as follows:

$$x_t \sim \mathcal{N}(x_t;\ x_0, \sigma_t^2 \mathrm{I}) \Rightarrow x_t = x_0 + \sigma_t \cdot z_t,\ z_t \sim \mathcal{N}(0, I) \Rightarrow z_t = \frac{x_t - x_0}{\sigma_t}$$

- So, $s_\theta(x_t, \sigma_t)$ learns to approximate a scaled negative noise $\frac{-z_t}{\sigma_t}$.

DDPM vs NCSN. Sampling

DDPM: $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\hat{\beta}_t}} z_\theta(x_t, t)\right) + \sqrt{\beta_t} \cdot z_t$

- Iterative updates are based on subtracting some form of noise from the noisy image.

NCSN: $x_t^i = x_t^{i-1} + \frac{\gamma_t}{2} \cdot s_\theta\left(x_t^{i-1}, \sigma_t\right) + \sqrt{\gamma_t} \cdot z_t$

- This is true also for NCSN because $s_\theta\left(x_t^{i-1}, \sigma_t\right)$ approximates the negative of the noise.

$$\mathcal{L}_{dsm} = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{p(x)}\mathbb{E}_{p_{\sigma_t}(x_t|x)} \left\| s_\theta(x_t, \sigma_t) + \frac{x_t - x}{\sigma_t^2} \right\|_2^2$$

$$\mathcal{L}_{dsm} = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{p(x)}\mathbb{E}_{p_{\sigma_t}(x_t|x)} \left\| s_\theta(x_t, \sigma_t) - (-\frac{x_t - x}{\sigma_t^2}) \right\|_2^2$$

$$\underbrace{\min_\theta \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{x_0 \sim p(x_0), z_t \sim \mathcal{N}(0, \mathrm{I})} \left\| z_t - z_\theta(x_t, t) \right\|_2^2}_{\mathcal{L}_{simple}}$$

$$z_t = \frac{x_t - x_0}{\sigma_t}$$

- Therefore, the generative processes defined by NCSN and DDPM are very similar.

Stochastic Differential Equations (SDEs)

- A generalized framework that can be applied over the previous two methods
- However, the diffusion process is continuous, given by an SDE
- Works by the same principle:
  - Gradually transforms the data distribution $p(x_0)$ into noise
  - Reverse the process to obtain the original data distribution

Stochastic Differential Equations (SDEs)

- The forward diffusion process is represented by the following SDE:

$$\frac{\partial x}{\partial t} = f(x,t) + \sigma(t)\omega_t \Leftrightarrow \partial x = f(x,t)\partial t + \sigma(t) \cdot \partial \omega_t$$

Notation for: $\mathcal{N}(0, \partial t)$

Function for drift coefficient: gradually nullifies the data x0

Function for diffusion coefficient: controls how much Gaussian noise is added

White Gaussian noise

Stochastic Differential Equations (SDEs)

- The reverse-time SDE is defined as:

$$\partial x = [f(x,t) - \sigma(t)^2 \cdot \nabla_x log\, p_t(x)]\partial t + \sigma(t) \cdot \partial\hat{\omega}$$

- The training objective is similar to NCSN, but adapted for continuous time:

$$\mathcal{L}_{dsm}^{*} = \mathbb{E}_t \left[\lambda(t)\mathbb{E}_{p(x_0)}\mathbb{E}_{p_t(x_t|x_0)}\left\| s_\theta(x_t,t) + \nabla_{x_t} log\, p_t(x_t|x_0)\right\|_2^2\right]$$

- The score function is used in the reverse-time SDE:
  - ➢ It employs a neural network to estimate the score function.
  - ➢ Then uses a numerical SDE solver to generate samples.

Stochastic Differential Equations (SDEs). NCSN

- The process of NCSN:

$$x_t \sim \mathcal{N}(x_t; x_{t-1}, (\sigma_t^2 - \sigma_{t-1}^2) \cdot I) \Rightarrow x_t = x_{t-1} + \sqrt{(\sigma_t^2 - \sigma_{t-1}^2)} \cdot z_t$$

- We can reformulate the above expression to look like a discretization of an SDE:

$$x_t - x_{t-1} = \sqrt{\frac{(\sigma_t^2 - \sigma_{t-1}^2)}{t - (t-1)}} \cdot z_t$$

- Translating the above discretization in the continuous case:

$$\partial x = \sqrt{\frac{\partial \sigma^2(t)}{\partial t}} \partial \omega(t)$$

$$\partial x = f(x,t)\partial t + \sigma(t) \cdot \partial \omega_t$$

Stochastic Differential Equations (SDEs). DDPM

- The process of DDPM:

$$x_i \sim \mathcal{N}\left(x_i; \sqrt{1-\beta_i} \cdot x_{i-1}, \beta_i \mathrm{I}\right) \Rightarrow x_i = \sqrt{1-\beta_i} \cdot x_{i-1} + \sqrt{\beta_i} \cdot z_i$$

- If we consider time step size $\Delta t = \frac{1}{T}$, instead of 1, and $\beta(t)\Delta t = \beta_i$:

$$x_t = \sqrt{1-\beta(t)\Delta t} \cdot x_{t-\Delta t} + \sqrt{\beta(t)\Delta t} \cdot z_t$$

- Using Taylor expansion of $\sqrt{1-\beta(t)\Delta t}$:

$$x_t \approx \left(1 - \frac{\beta(t)\Delta t}{2}\right) \cdot x_{t-\Delta t} + \sqrt{\beta(t)\Delta t} \cdot z_t$$

$$x_t \approx x_{t-\Delta t} - \frac{\beta(t)\Delta t}{2} \cdot x_{t-\Delta t} + \sqrt{\beta(t)\Delta t} \cdot z_t \Leftrightarrow x_t - x_{t-\Delta t} = -\frac{\beta(t)\Delta t}{2} \cdot x_t + \sqrt{\beta(t)\Delta t} \cdot z_t$$

- For the continuous case, the above becomes:

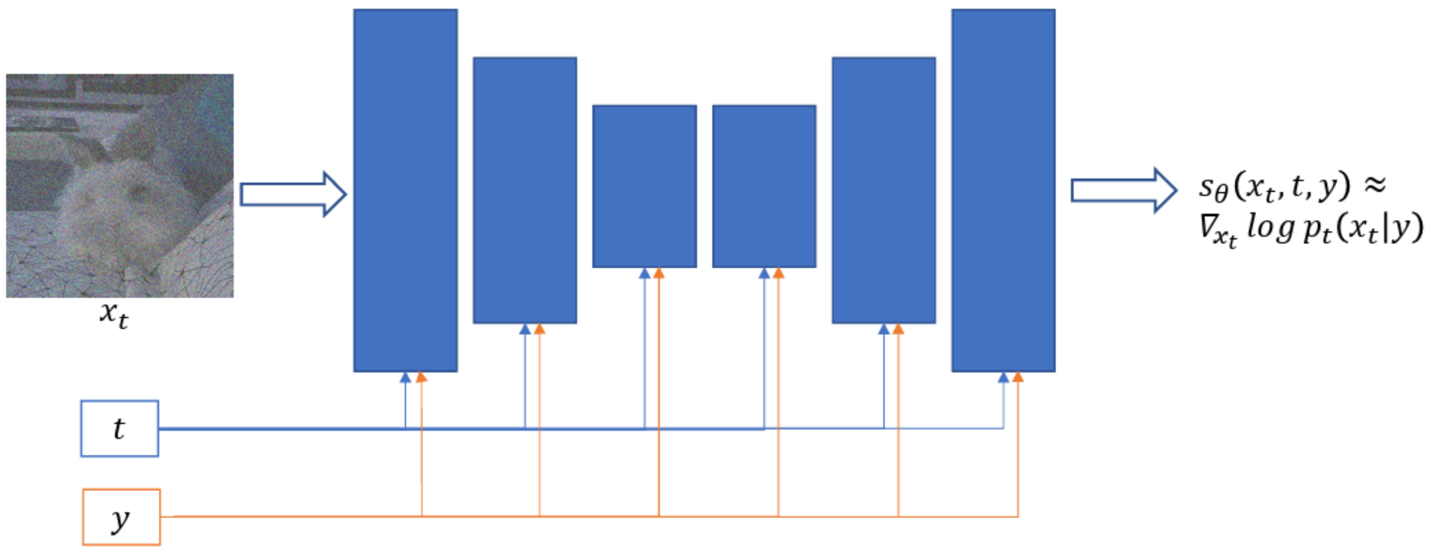
$$\partial x = -\frac{1}{2}\, \beta(t) x_t \partial t + \sqrt{\beta(t)}\partial \omega(t)$$

Conditional generation.

Diffusion models estimate the score function, $\boldsymbol{\nabla_{x_t} log\, p_t(x_t)}$ to sample from a distribution $\boldsymbol{p(x)}$.

Sampling from $\mathbf{p}(\boldsymbol{x}|\boldsymbol{y})$ requires the score function of this probability density, $\boldsymbol{\nabla_{x_t}\, log\, p_t(x_t|y)}$; $\boldsymbol{y}$ is condition.

Solution 1. **Conditional training:** train the model with an additional input y to estimate $\boldsymbol{\nabla_{x_t}\, log\, p_t(x_t|y)}$.



Conditional generation. Classifier Guidance

Diffusion models estimate the score function, $\boldsymbol{\nabla_{x_t} log\, p_t(x_t)}$ to sample from a distribution $\boldsymbol{p(x)}$.

Sampling from $\mathbf{p}(\boldsymbol{x}|\boldsymbol{y})$ requires the score function of this probability density, $\boldsymbol{\nabla_{x_t} log\, p_t(x_t|y)}$.

Solution 2. **Classifier guidance:**

Bayes rule:

$$p_t(x_t|y) = \frac{p_t(y|x_t) \cdot p_t(x_t)}{p_t(y)} \Leftrightarrow$$

Logarithm:

$$\log p_t(x_t|y) = \log p_t(y|x_t) + \log p_t(x_t) - \log p_t(y) \;\; \Leftrightarrow$$

Gradient:

$$\nabla_{x_t} \log p_t(x_t|y) = \nabla_{x_t} \log p_t(y|x_t) + \nabla_{x_t} \log p_t(x_t) \;\; - \nabla_{x_t} \log p_t(y) \Leftrightarrow$$

$$\boldsymbol{\nabla_{x_t} log\, p_t(x_t|y) = \nabla_{x_t} log\, p_t(y|x_t) + \nabla_{x_t} log\, p_t(x_t)}$$

Classifier

Unconditional diffusion model

Conditional generation. Classifier Guidance

Solution 2. **Classifier guidance:**

Guidance weight

$$\nabla_{x_t} \log p_t(x_t|y) = s \cdot \nabla_{x_t} \log p_t(y|x_t) + \nabla_{x_t} \log p_t(x_t)$$

$s = 1$

$s = 10$

Problem

- Need to have good gradients estimates at each step of denoising process
  - Need a classifier that is robust to noise added in the image.
  - Training of the classifier on noisy data, which can be problematic..

Conditional generation. Classifier-free Guidance

Solution 3. **Classifier-free guidance**

$$\nabla_{x_t} log\, p_t(x_t|y) = s \cdot \nabla_{x_t} log\, p_t(y|x_t) + \nabla_{x_t} log\, p_t(x_t)$$

Bayes rule:

$$p_t(y|x_t) = \frac{p_t(x_t|y) \cdot p_t(y)}{p_t(x_t)}$$

Logarithm:

$$\log p_t(y|x_t) = \log p_t(x_t|y) - \log p_t(x_t) + \log p_t(y)$$

Gradient

$$\nabla_{x_t} log\, p_t(y|x_t) = \nabla_{x_t} log\, p_t(x_t|y) - \nabla_{x_t} \log p_t(x_t)$$

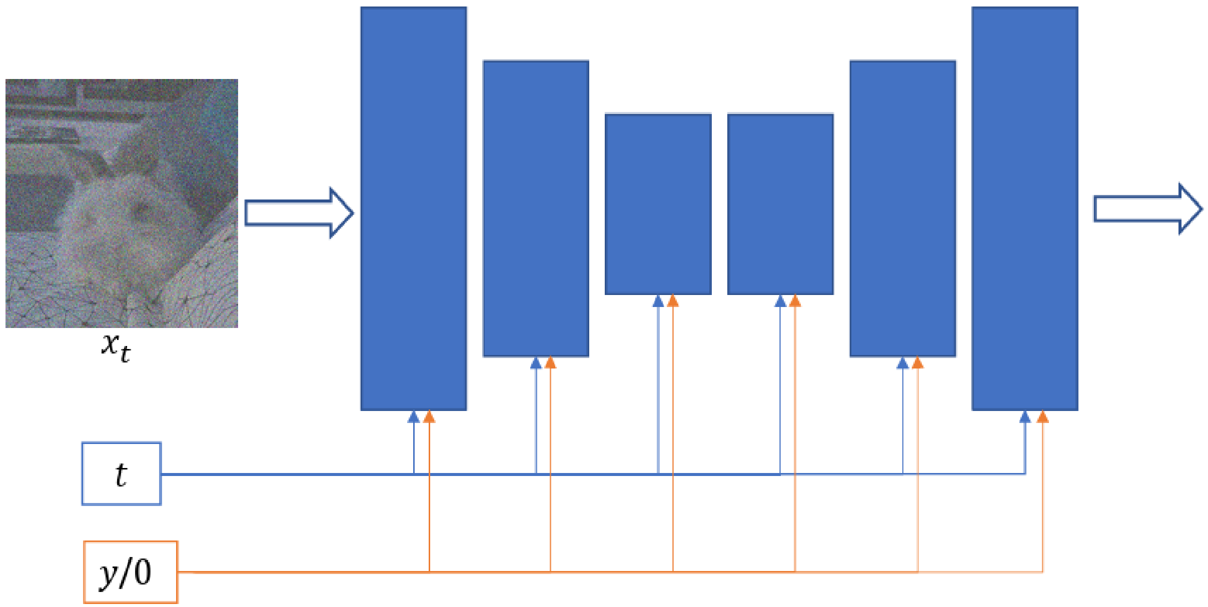
from above $\nabla_{x_t} log\, p_t(x_t|y) = s \cdot \nabla_{x_t} log\, p_t(y|x_t) + \nabla_{x_t} log\, p_t(x_t)$

$$\nabla_{x_t} log\, p_t(x_t|y) = s \cdot (\nabla_{x_t} log\, p_t(x_t|y) - \nabla_{x_t} \log p_t(x_t)) + \nabla_{x_t} log\, p_t(x_t)$$

$$\boldsymbol{\nabla_{x_t} log\, p_t(x_t|y) = s \cdot \nabla_{x_t} log\, p_t(x_t|y) + (1 - s) \cdot \nabla_{x_t} log\, p_t(x_t)}$$

Learned by a single model

Conditional generation. Classifier-free Guidance



$$s_\theta(x_t, t, y/0\,) \approx p_t(x_t|y)/p_t(x_t|0)$$

Thank You