Score-based Generative Models and Diffusion Models

Open DMQA Seminar
2022.02.11

조한샘

고려대학교
KOREA UNIVERSITY

CONTENTs

- Deep Generative Models
- Score-based Generative Models (NCSN)
- Diffusion Models (DDPM)
- Score-based Generative Modeling Through SDEs

발표자 소개

- 조한샘
  - ✓ Data Mining & Quality Analytics Lab
  - ✓ 석사과정 (2020.09~)
- 관심 연구 분야
  - ✓ Deep Generative Models
- Contact
  - ✓ chosam95@korea.ac.kr

Introduction

Image Generation on CIFAR-10

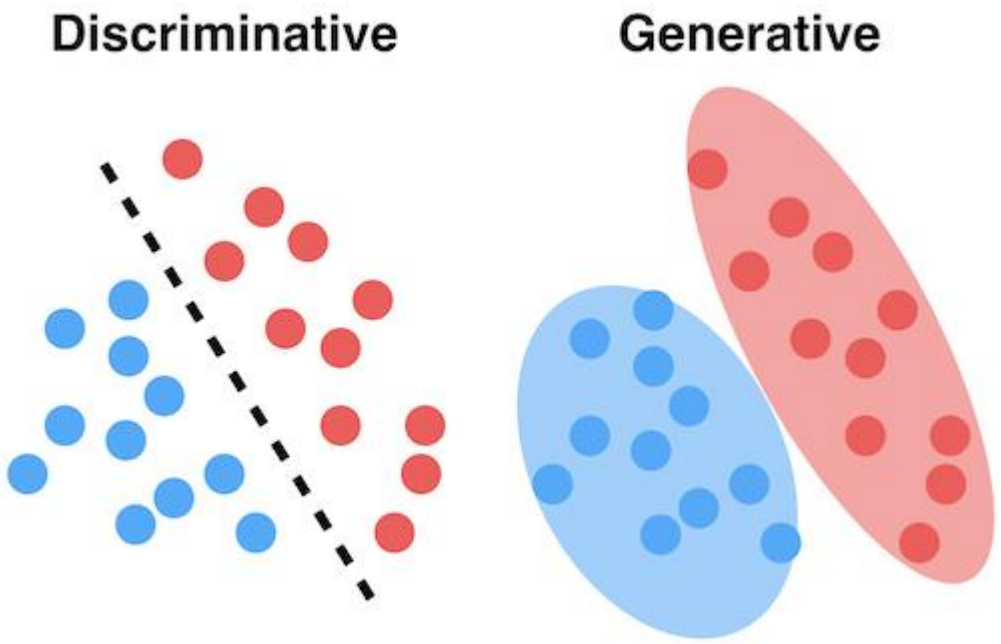
Rank	Model	FID	Inception score	bits/dimension	FID-10k-test	Paper	Code	Result	Year	Tags
1	**LSGM** (FID)	2.10		3.43		Score-based Generative Modeling in Latent Space			2021	VAE, Score-based
2	**LSGM** (balanced)	2.17		2.95		Score-based Generative Modeling in Latent Space			2021	VAE, Score-based
3	**NCSN++**	2.20	9.73			Score-Based Generative Modeling through Stochastic Differential Equations			2020	Score-based
4	**CLD-SGM** (EM-QS)	2.23				Score-Based Generative Modeling with Critically-Damped Langevin Diffusion			2021	
5	**CLD-SGM** (Prob. Flow)	2.25		3.31		Score-Based Generative Modeling with Critically-Damped Langevin Diffusion			2021	

https://paperswithcode.com/sota/image-generation-on-cifar-10

Deep Generative Models

Discriminative models vs Generative models

- Discriminative models: $P(Y|X)$를 모델링, decision boundary를 찾는 것이 목적 / classification
- Generative models: $P(X)$ 를 모델링, 데이터의 분포를 찾는 것이 목적 / data generation

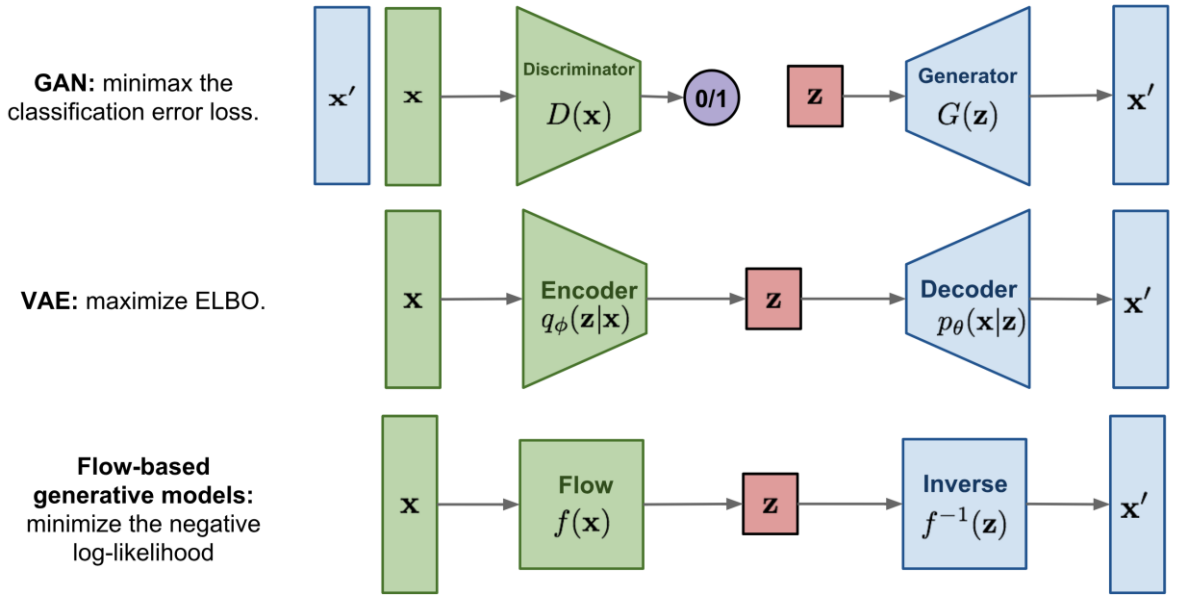


https://betterprogramming.pub/generative-vs-discriminative-models-d26def8fd64a

Deep Generative Models

Deep Generative Models

- GAN: 학습의 불안전성
- VAE: 직접 Likelihood를 최대화 하지 못함
- Flow-based: 역함수가 존재하는 구조에서만 사용가능



https://lilianweng.github.io/lil-log/2018/10/13/flow-based-deep-generative-models.html

Score-based Generative Models

Generative Modeling by Estimating Gradients of the Data Distribution (Song et al., 2019)

- NeurIPS 2019
- 2022년 02월 10일 기준: 235회 인용

Generative Modeling by Estimating Gradients of the Data Distribution

Yang Song
Stanford University
yangsong@cs.stanford.edu

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

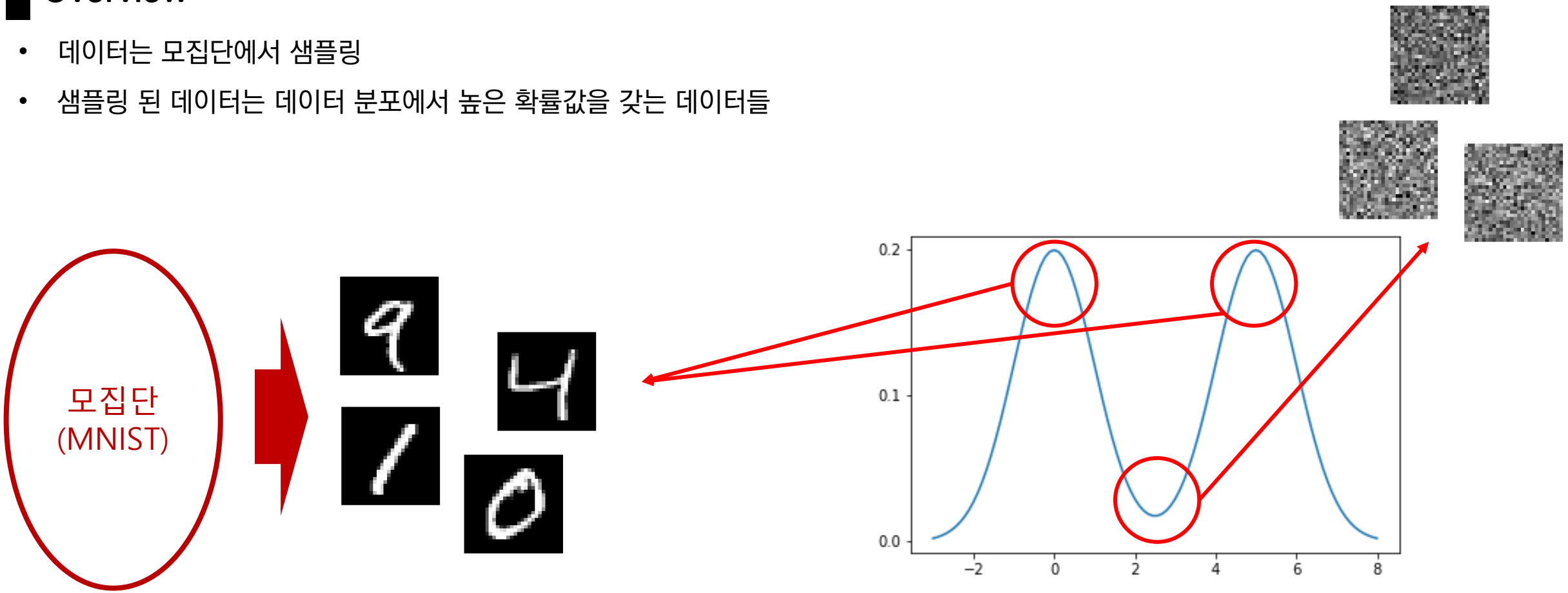
Abstract

We introduce a new generative model where samples are produced via Langevin dynamics using gradients of the data distribution estimated with score matching. Because gradients can be ill-defined and hard to estimate when the data resides on low-dimensional manifolds, we perturb the data with different levels of Gaussian noise, and jointly estimate the corresponding scores, *i.e.*, the vector fields of gradients of the perturbed data distribution for all noise levels. For sampling, we propose an annealed Langevin dynamics where we use gradients corresponding to gradually decreasing noise levels as the sampling process gets closer to the data manifold. Our framework allows flexible model architectures, requires no sampling during training or the use of adversarial methods, and provides a learning objective that can be used for principled model comparisons. Our models produce samples comparable to GANs on MNIST, CelebA and CIFAR-10 datasets, achieving a new state-of-the-art inception score of 8.87 on CIFAR-10. Additionally, we demonstrate that our models learn effective representations via image inpainting experiments.

Score-based Generative Models

Overview

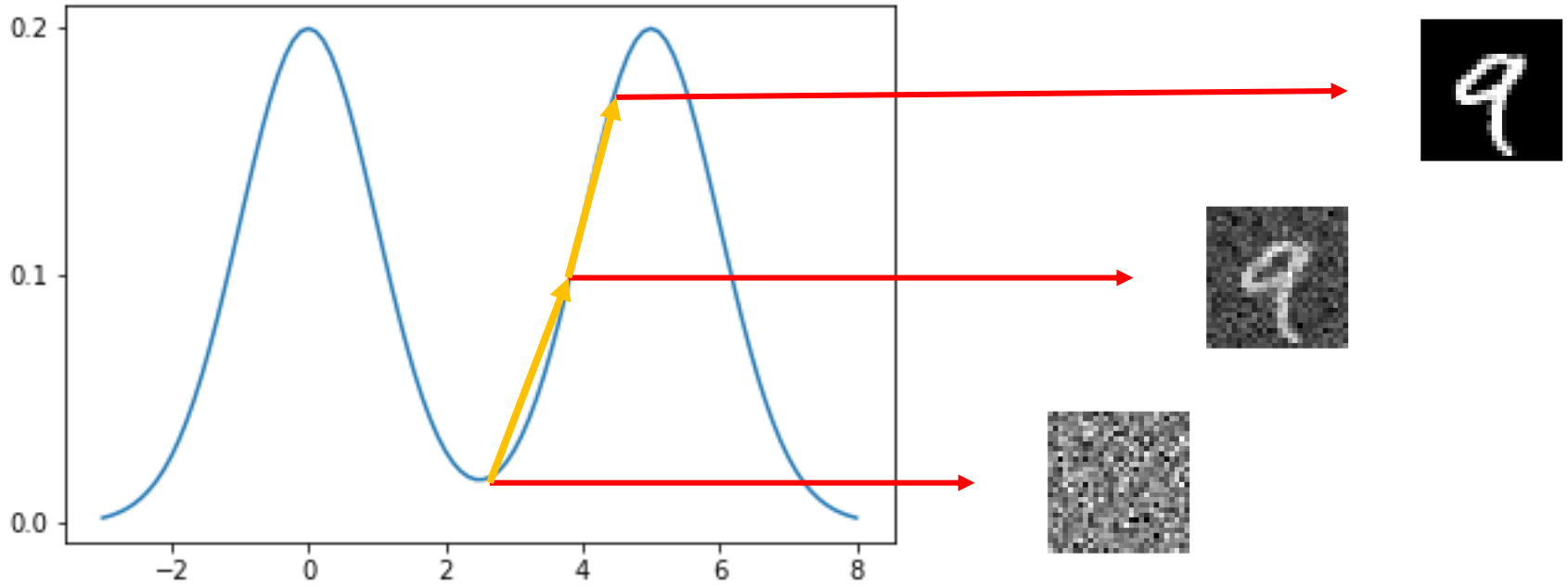
- 데이터는 모집단에서 샘플링
- 샘플링 된 데이터는 데이터 분포에서 높은 확률값을 갖는 데이터들



Score-based Generative Models

Overview

- 데이터 공간상에서 임의의 데이터 생성 (랜덤 노이즈)
- 확률밀도함수의 기울기를 계산 후 확률 값이 높아지는 방향으로 데이터 업데이트
- 확률값이 높은 곳에 도달하면 샘플링된 데이터와 유사한 데이터 생성 가능



Score-based Generative Models

Score-based Generative Models

- Score: 확률밀도함수의 미분
- 입력 데이터 x에 대한 미분이다!

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

Multivariate Normal

$$p(x) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$

$$\nabla_x log p(x) = -\Sigma^{-1}(x-\mu)$$

Example

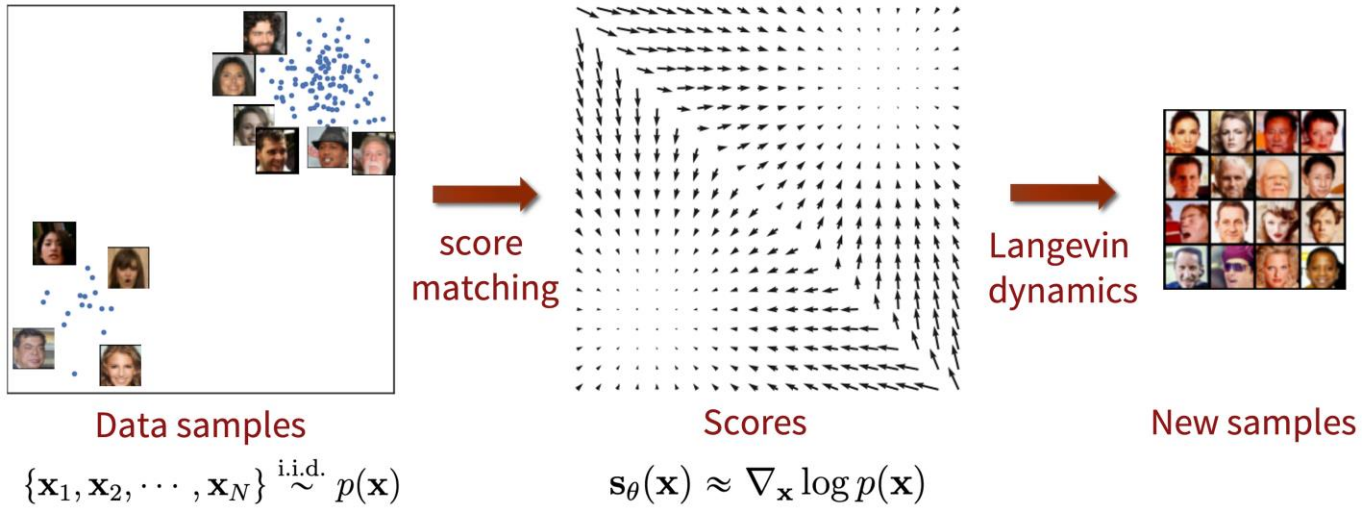
$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, x = \begin{bmatrix} 0 \\ 0.5 \end{bmatrix}$$

$$\nabla_x log p(x) = \begin{bmatrix} 0 \\ -0.5 \end{bmatrix}$$

Score-based Generative Models

Score-based Generative Models

- 데이터의 분포를 모르지만 **Score**만 알면 데이터 생성 가능
- Score를 데이터로부터 추정 (Score matching) / **Training**
- 추정된 score를 바탕으로 새로운 데이터 sampling (Langevin dynamics) / **Testing**



https://yang-song.github.io/blog/2021/score/

Score-based Generative Models

Score matching

- 데이터 x 에 대해 score값을 계산해주는 모델(Score Network) 만들자!

True score　　Score Network

$$Loss = \frac{1}{2} E_{p_{data}(x)} [\| \nabla_x log p(x) - s_\theta(x) \|_2^2]$$

Score Matching (2005)

Jacobian Matrix (d*d)

$$E_{p_{data}(x)} \left[tr\left(\nabla_x s_\theta(x)\right) + \frac{1}{2} \| s_\theta(x) \|_2^2 \right]$$

Denoising Score Matching (2011)

$$\frac{1}{2} E_{q_\sigma(\tilde{x}|x) p_{data}(x)} [\| s_\theta(\tilde{x}) - \nabla_x log q_\sigma(\tilde{x}|x) \|_2^2]$$

Hyvärinen, A., & Dayan, P. (2005). Estimation of non-normalized statistical models by score matching. Journal of Machine Learning Research, 6(4).
Vincent, P. (2011).A connection between score matching and denoising autoencoders. Neural computation, 23(7), 1661-1674.

Score-based Generative Models

Denoising score matching

- $q_\sigma(\tilde{x}|x)$: 가우시안 노이즈
- $\tilde{x}$: 입력 데이터 x에 노이즈 추가
- 노이즈가 추가된 데이터의 score 예측
- 노이즈가 충분히 작으면 원래 데이터의 score 예측 가능

$$\frac{1}{2}E_{q_\sigma(\tilde{x}|x)p_{data}(x)}[\|s_\theta(\tilde{x}) - \nabla_x log q_\sigma(\tilde{x}|x)\|_2^2]$$

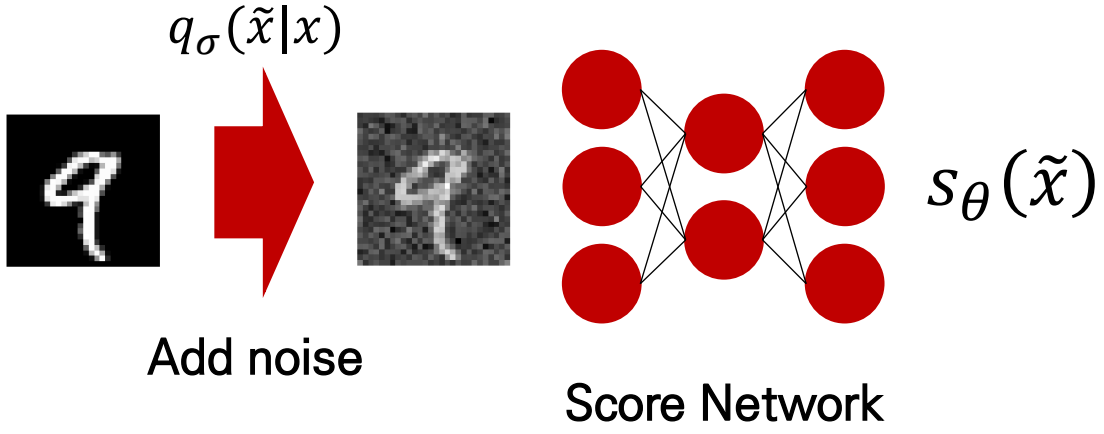
Small noise

$$s_{\theta^*}(x) = \nabla_x log q_\sigma(x) \approx \nabla_x log p_{data}(x)$$

Score-based Generative Models

Score network

- Denoising score matching을 통해 score network 학습
- Input: 데이터($\tilde{x}$) / Output: 데이터의 score
- Score network: $R^d \rightarrow R^d$
- 논문에서는 U-net 구조 사용



Network Output

Score

$$\frac{1}{2}E_{q_\sigma(\tilde{x}|x)p_{data}(x)}[\|s_\theta(\tilde{x}) - \nabla_x log q_\sigma(\tilde{x}|x)\|_2^2]$$

$$-\Sigma^{-1}(\tilde{x}-\mu)$$

Score-based Generative Models

Langevin dynamics

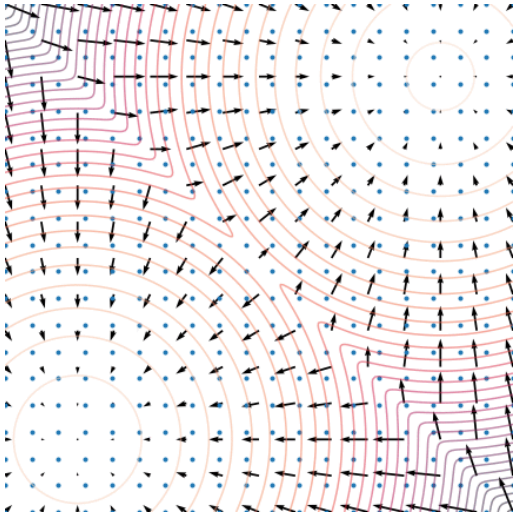
- Score network 잘 학습되었다면 모든 데이터 공간상에서 score 계산할 수 있다
- 임의의 데이터(랜덤 노이즈)에서 시작
- 현재 시점에서 추정된 score를 기반으로 데이터 업데이트
- Score 타고 올라가다 보면 높은 확률값(샘플링 데이터와 유사한)의 데이터 생성 가능

$$\tilde{x}_t = \tilde{x}_{t-1} + \frac{\epsilon}{2}\nabla_x log p(\tilde{x}_{t-1}) + \sqrt{\epsilon} z_t$$

이전시점 data

Score

Random Noise

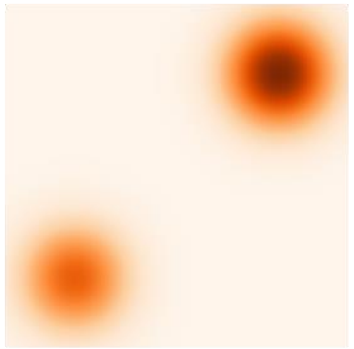


https://yang-song.github.io/blog/2021/score/

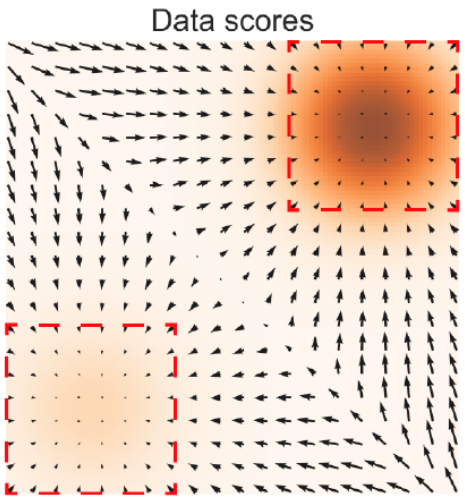
Score-based Generative Models

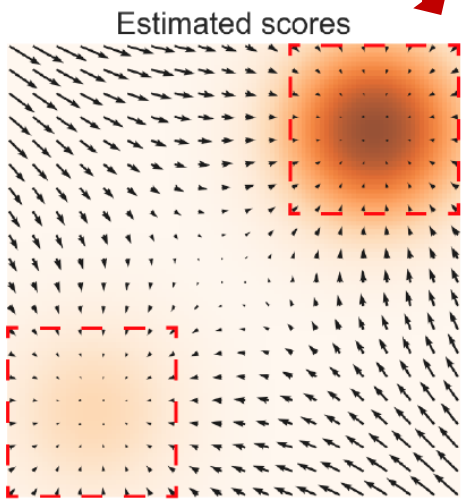
Problem in Low Density Regions (Inaccurate score estimation)

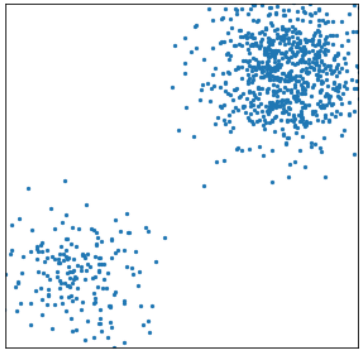
- 데이터는 확률값이 높은 부분에서 샘플링
- 확률값이 낮은 공간에서 score 정보가 부정확



Data distribution





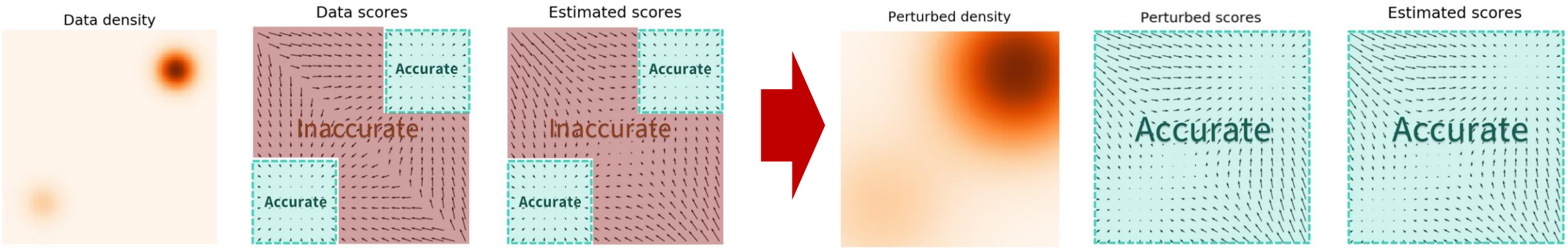


Sampling

Score-based Generative Models

Noise Conditional Score Networks

- **Problem:** 확률값이 낮은 공간에서 score 추정이 부정확
- **Solution:** 데이터에 노이즈를 추가한 후 score를 추정하자
- 노이즈가 추가된 데이터가 확률값이 낮은 공간을 채워서 score 추정 가능

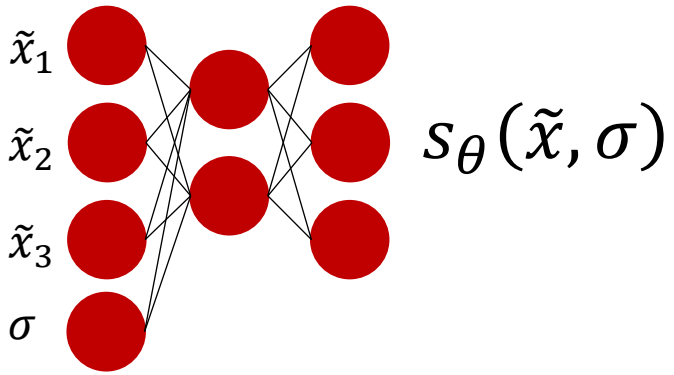


https://yang-song.github.io/blog/2021/score/

Score-based Generative Models

Noise Conditional Score Networks

- Input: 데이터($\tilde{x}$) + 노이즈(σ) / Output: score
- $q_\sigma(\tilde{x}|x) = N(\tilde{x}|x, \sigma^2 I)$
- 사전에 정의된 다양한 크기의 σ^2사용



Noise Conditional Score Network (NCSN)

$$\frac{1}{2} E_{q_\sigma(\tilde{x}|x)p_{data}(x)}\left[\left\| s_\theta(\tilde{x}) - \nabla_x log q_\sigma(\tilde{x}|x) \right\|_2^2\right]$$

Add noise condition

Gaussian distribution

$$\frac{1}{2} E_{q_\sigma(\tilde{x}|x)p_{data}(x)}\left[\left\| s_\theta(\tilde{x}, \sigma) - \frac{\tilde{x} - x}{\sigma^2} \right\|_2^2\right]$$

Score-based Generative Models

Annealed Langevin dynamics

- Noise schedule: 노이즈 크기를 감소시키며 샘플링 진행
- Gradient ascent: T step만큼 데이터 업데이트

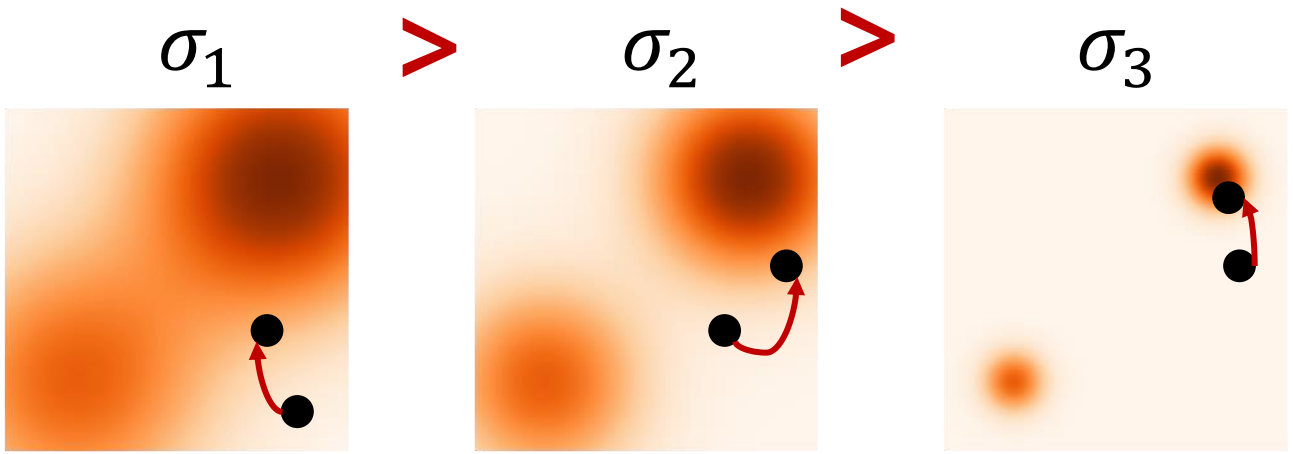
Noise schedule(1 → L)

Gradient ascent step(1 → T)

Algorithm 1 Annealed Langevin dynamics.

Require: $\{\sigma_i\}_{i=1}^{L}, \epsilon, T$.

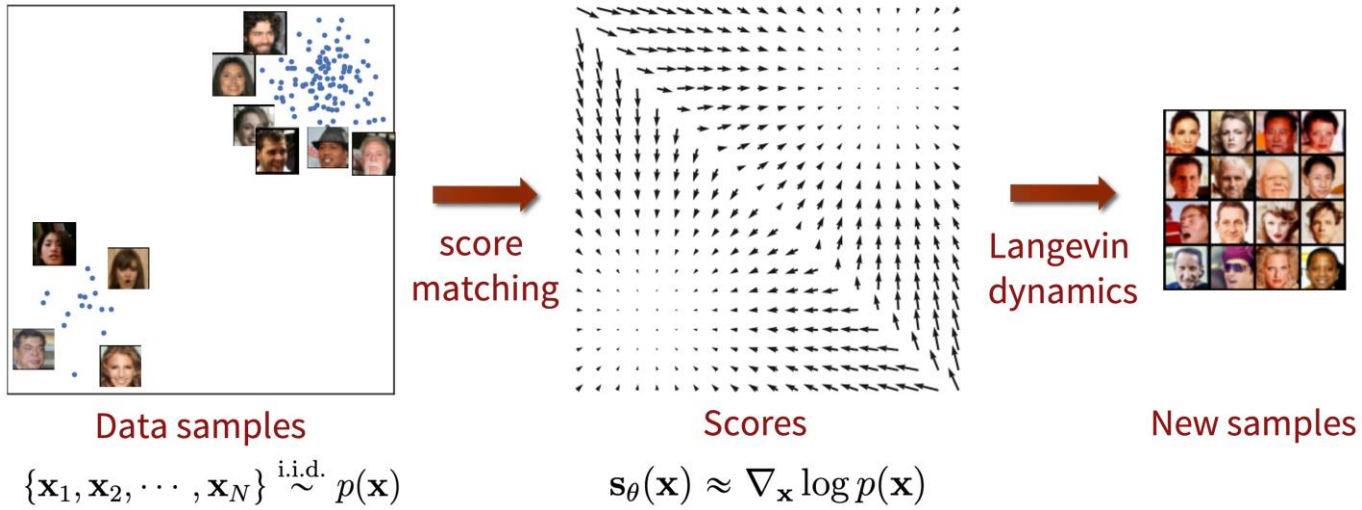
1: Initialize $\tilde{\mathbf{x}}_0$
2: **for** $i \leftarrow 1$ to L **do**
3: $\quad \alpha_i \leftarrow \epsilon \cdot \sigma_i^2 / \sigma_L^2$ ▷ α_i is the step size.
4: $\quad$ **for** $t \leftarrow 1$ to T **do**
5: $\quad\quad$ Draw $\mathbf{z}_t \sim \mathcal{N}(0, I)$
6: $\quad\quad \tilde{\mathbf{x}}_t \leftarrow \tilde{\mathbf{x}}_{t-1} + \frac{\alpha_i}{2}\mathbf{s}_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_{t-1}, \sigma_i) + \sqrt{\alpha_i}\,\mathbf{z}_t$
7: $\quad$ **end for**
8: $\quad \tilde{\mathbf{x}}_0 \leftarrow \tilde{\mathbf{x}}_T$
9: **end for**
return $\tilde{\mathbf{x}}_T$



Score-based Generative Models

Score-based Generative Models

- Score matching: Noise Conditional Score Networks(NCSN)
- Generation: Annealed Langevin dynamics



https://yang-song.github.io/blog/2021/score/

Diffusion Models

Denoising Diffusion Probabilistic Models (Ho et al., 2020)

- NeurIPS 2020
- 2022년 02월 10일 기준: 180회 인용

Denoising Diffusion Probabilistic Models

Jonathan Ho
UC Berkeley
jonathanho@berkeley.edu

Ajay Jain
UC Berkeley
ajayj@berkeley.edu

Pieter Abbeel
UC Berkeley
pabbeel@cs.berkeley.edu

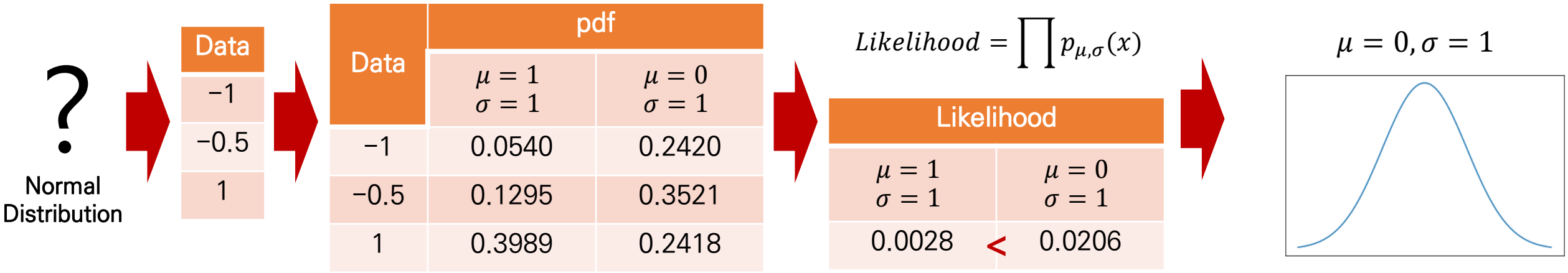
Abstract

We present high quality image synthesis results using diffusion probabilistic models, a class of latent variable models inspired by considerations from nonequilibrium thermodynamics. Our best results are obtained by training on a weighted variational bound designed according to a novel connection between diffusion probabilistic models and denoising score matching with Langevin dynamics, and our models naturally admit a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding. On the unconditional CIFAR10 dataset, we obtain an Inception score of 9.46 and a state-of-the-art FID score of 3.17. On 256x256 LSUN, we obtain sample quality similar to ProgressiveGAN. Our implementation is available at https://github.com/hojonathanho/diffusion.

Diffusion Models

Maximum Likelihood Estimation (MLE)

- PDF: 데이터가 해당 분포로 부터 샘플링 될 확률
- Likelihood: 주어진 파라미터를 이용한 분포가 모집단의 분포일 확률
- 주어진 데이터를 바탕으로 모집단의 분포와 유사할 확률이 가장 높은 파라미터 찾기



Diffusion Models

Maximum Likelihood Estimation (MLE)

- Likelihood를 파라미터로 미분 후 0값이 되는 지점

$$\frac{\partial Likelihood}{\partial \mu} = 0$$

$$\frac{\partial Likelihood}{\partial \sigma} = 0$$

$$\mu_{MLE} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\sigma_{MLE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$$

Diffusion Models

Variational AutoEncoder (VAE)

- Latent variable(z): 데이터를 설명해주는 특징값
- 직접 $p_\theta(x)$를 계산하기 어려움 → latent variable을 데이터 x에서부터 생성
- $p_\theta(z|x)$: True distribution
- $q_\phi(z|x)$: Model (Encoder)

$$Likelihood = \prod p_\theta(x)$$

$$p_\theta(x) = \int p_\theta(x|z)p_\theta(z)dz$$

Appendix 1

$$D_{KL}\left(q_\phi(z|x) \parallel p_\theta(z|x)\right)$$

$$= logp_\theta(x) + D_{KL}\left(q_\phi(z|x) \parallel p_\theta(z)\right) - E_{z\sim q_\phi(z|x)} logp_\theta(z|x)$$

Diffusion Models

Variational AutoEncoder (VAE)

- ①: KL Divergence 최소화
- ②: Likelihood 최대화

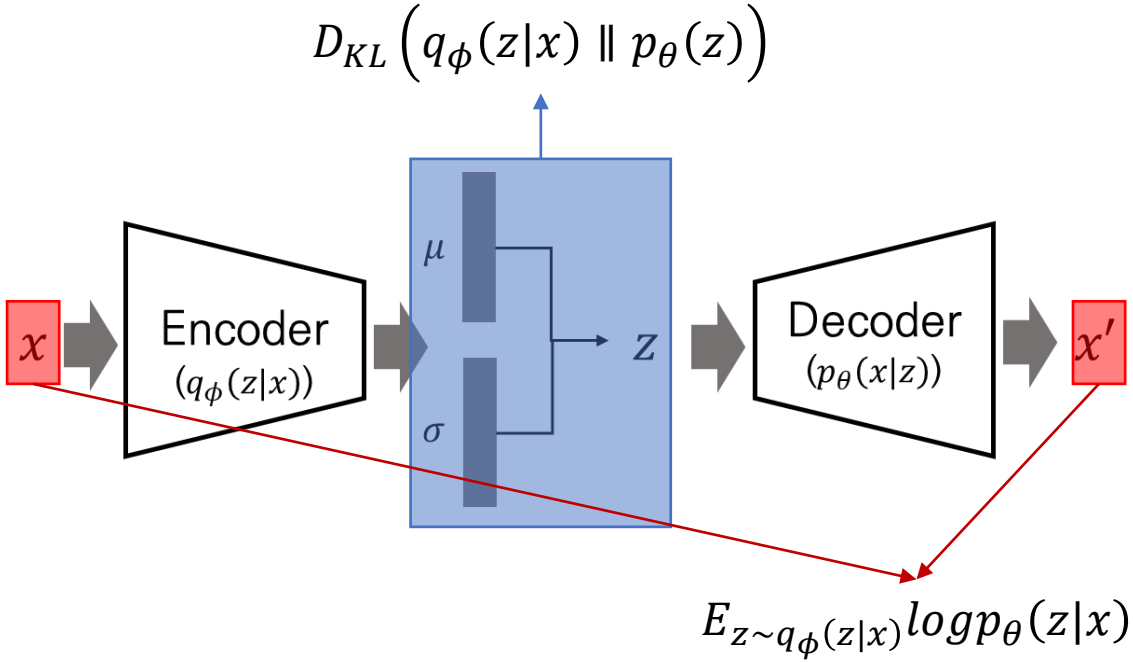
$$Loss_{VAE}$$

$$= \overset{①}{D_{KL}}\left(q_\phi(z|x) \parallel p_\theta(z|x)\right) \overset{②}{-} logp_\theta(x)$$

$$= D_{KL}\left(q_\phi(z|x) \parallel p_\theta(z)\right) - E_{z\sim q_\phi(z|x)} logp_\theta(z|x)$$

z가 사전에 정의된 분포와 거리 ↓

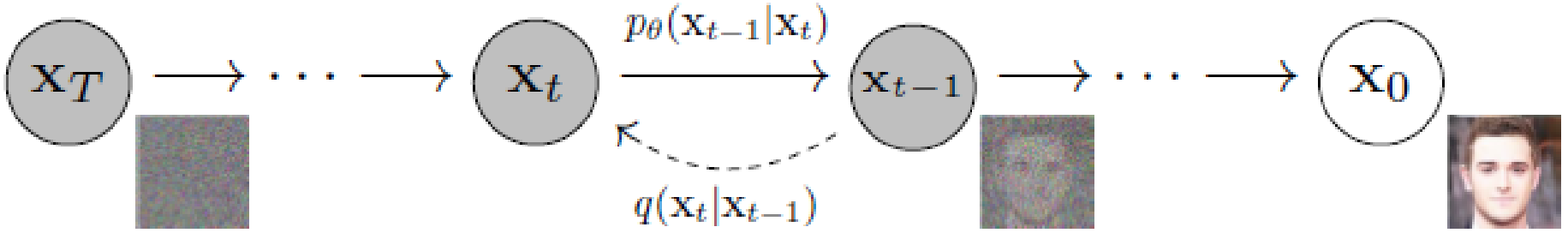
Reconstruction Loss



Diffusion Models

Denoising Diffusion Probabilistic Models (DDPM)

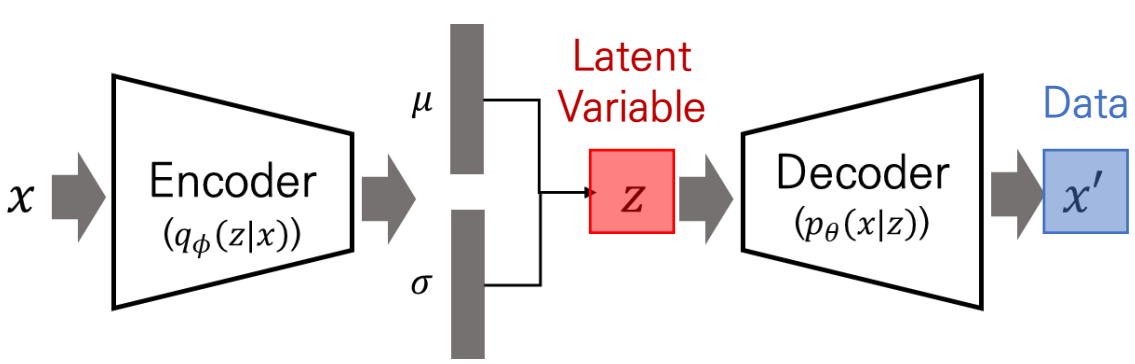
- Forward process: 데이터(x_0) + 노이즈 → 랜덤 노이즈(x_T)
- Reverse process: 랜덤 노이즈 (x_T) + 노이즈 제거 → 데이터(x_0)
- 노이즈를 제거하는 reverse process를 학습할 수 있다면 랜덤 노이즈로부터 데이터 생성 가능



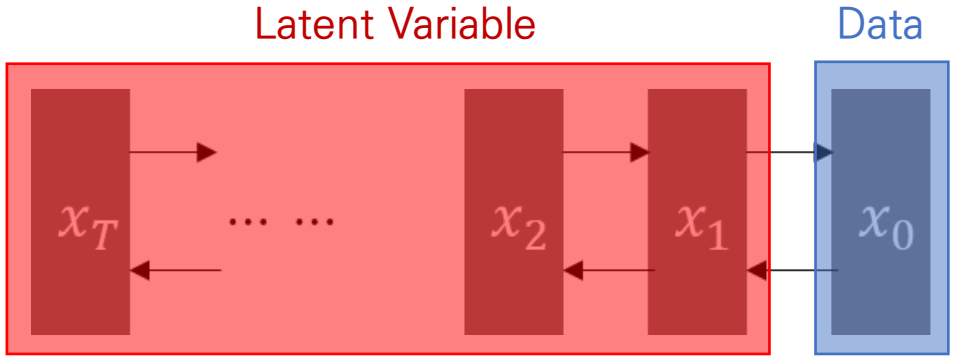
Diffusion Models

Denoising Diffusion Probabilistic Models (DDPM)

- VAE: Latent variable(z)
- Diffusion Models: Latent variable이 Markov chain ($x_1 \sim x_T$)
- Random process: 확률 변수들의 나열
- Markov chain: 이전 시점의 변수에만 영향을 받은 random process ($\mathrm{P}(X_{n+1}|X_n) = \mathrm{P}(X_{n+1}|X_n, X_{n-1}, \dots, X_0)$)



VAE



DDPM

Diffusion Models

Denoising Diffusion Probabilistic Models (DDPM)

- Data: $x \rightarrow x_0$
- Latent variable: $\mathrm{z} \rightarrow x_1 \sim x_T(x_{1:T})$

$$Loss_{VAE}$$

$$= D_{KL}\left(q_\phi(z|x) \parallel p_\theta(z|x)\right) - log p_\theta(x)$$

$$Loss_{DDPM}$$

$$= D_{KL}\left(q_\phi(x_{1:T}|x_0) \parallel p_\theta(x_{1:T}|x_0)\right) - log p_\theta(x_0)$$

Diffusion Models

Denoising Diffusion Probabilistic Models (DDPM)

- DDPM Loss 유도

$$Loss_{DDPM}$$

$$= D_{KL}\left(q_\phi(x_{1:T}|x_0) \parallel p_\theta(x_{1:T}|x_0)\right) - log p_\theta(x_0)$$

$$= E_{q_\phi(x_{0:T})}\left[log \frac{q_\phi(x_{1:T}|x_0)}{p_\theta(x_{0:T})}\right]$$

$$= E_{q_\phi(x_{0:T})}\left[D_{KL}\left(q(x_T|x_0) \parallel p_\theta(x_T)\right) + \sum_{t=2}^{T} D_{KL}\left(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t)\right) - log p_\theta(x_0|x_1)\right]$$

Appendix 2

Diffusion Models

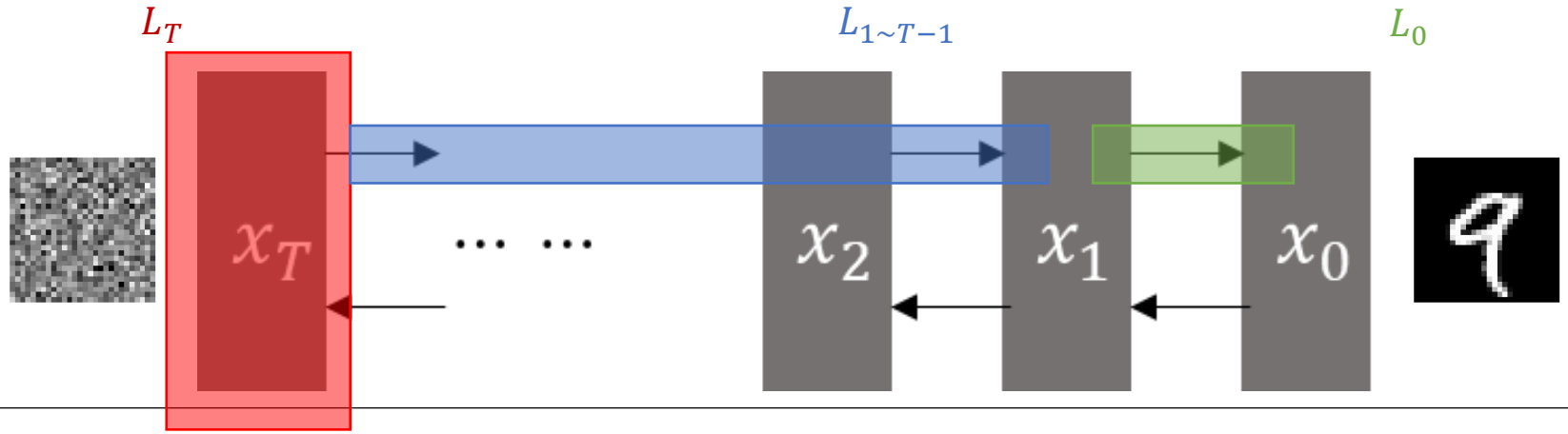
Denoising Diffusion Probabilistic Models (DDPM)

- Forward process: $q(x_t|x_{t-1}) \rightarrow Gaussian$
- Reverse process: $q(x_{t-1}|x_t) \rightarrow Gaussian$

Small noise

- $q(x_{t-1}|x_t, x_0) = \mathrm{N}\left(x_{t-1};\ \tilde{\mu}(x_t, x_0),\ \tilde{\beta}_t I\right) \rightarrow True\ Reverse\ Process$
- $p_\theta(x_{t-1}|x_t) \rightarrow Model$

$$Loss_{DDPM} = E_{q_\phi(x_{0:T})}\left[D_{KL}\left(q(x_T|x_0) \parallel p_\theta(x_T)\right) + \sum_{t=2}^{T} D_{KL}\left(q(x_{t-1}|x_t, x_0) \parallel p_\theta(x_{t-1}|x_t)\right) - log p_\theta(x_0|x_1)\right]$$



Diffusion Models

Denoising Diffusion Probabilistic Models (DDPM)

$$\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

Forward process noise

- $q(x_{t-1}|x_t, x_0) = \mathrm{N}(x_{t-1};\ \tilde{\mu}(x_t, x_0),\ \tilde{\beta}_t I) \rightarrow True\ Reverse\ Process$
- Model: $\mu_\theta(x_{t-1}|x_t)$
- 논문에서는 U-net 기반의 Pixel CNN++ 사용

$$Loss_{DDPM}$$

$$= E_{x_0,z}[\|\tilde{\mu}(x_t, x_0) - \mu_\theta(x_t, t)\|^2]$$

$$= E_{x_0,z}[\|z_t - z_\theta(x_t, t)\|^2]$$

Appendix 3

$$z_t \sim N(0,1)$$
$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}z_t$$

Diffusion Models

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

Denoising Diffusion Probabilistic Models (DDPM)

- Training

t 시점 데이터 생성

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

x_0

x_t

x_t

t

Model

$z_\theta(x_t, t)$

랜덤 노이즈 생성

$z_t \sim N(0,1)$

$E_{x_0,z}[\|z_t - z_\theta(x_t, t)\|^2]$

Model output

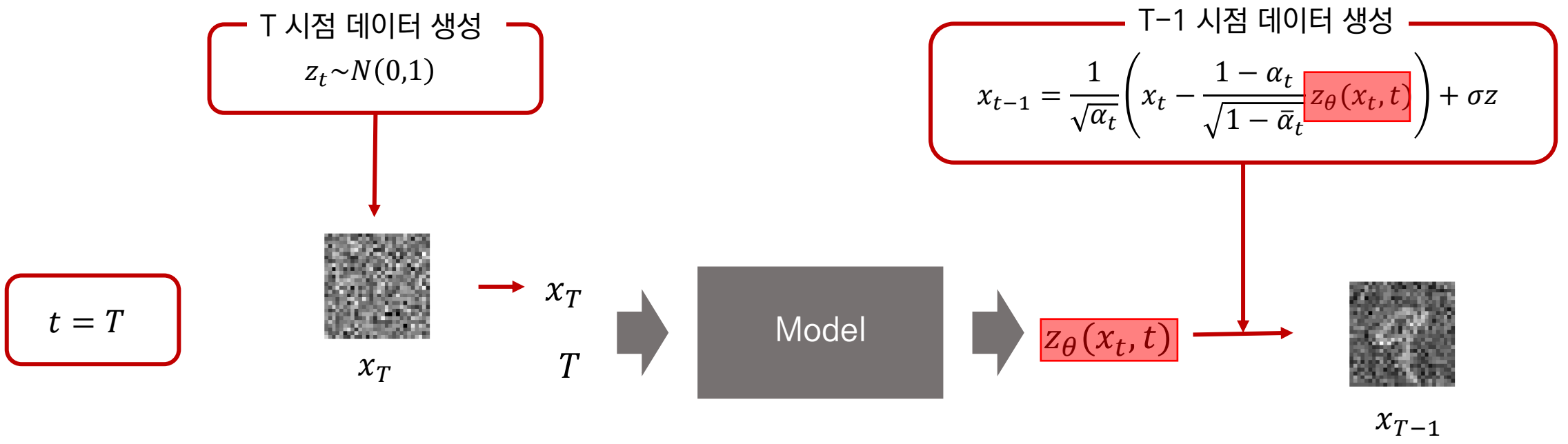
Diffusion Models

Denoising Diffusion Probabilistic Models (DDPM)

- Testing: x_T에서 x_{T-1} 생성

Algorithm 2 Sampling

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



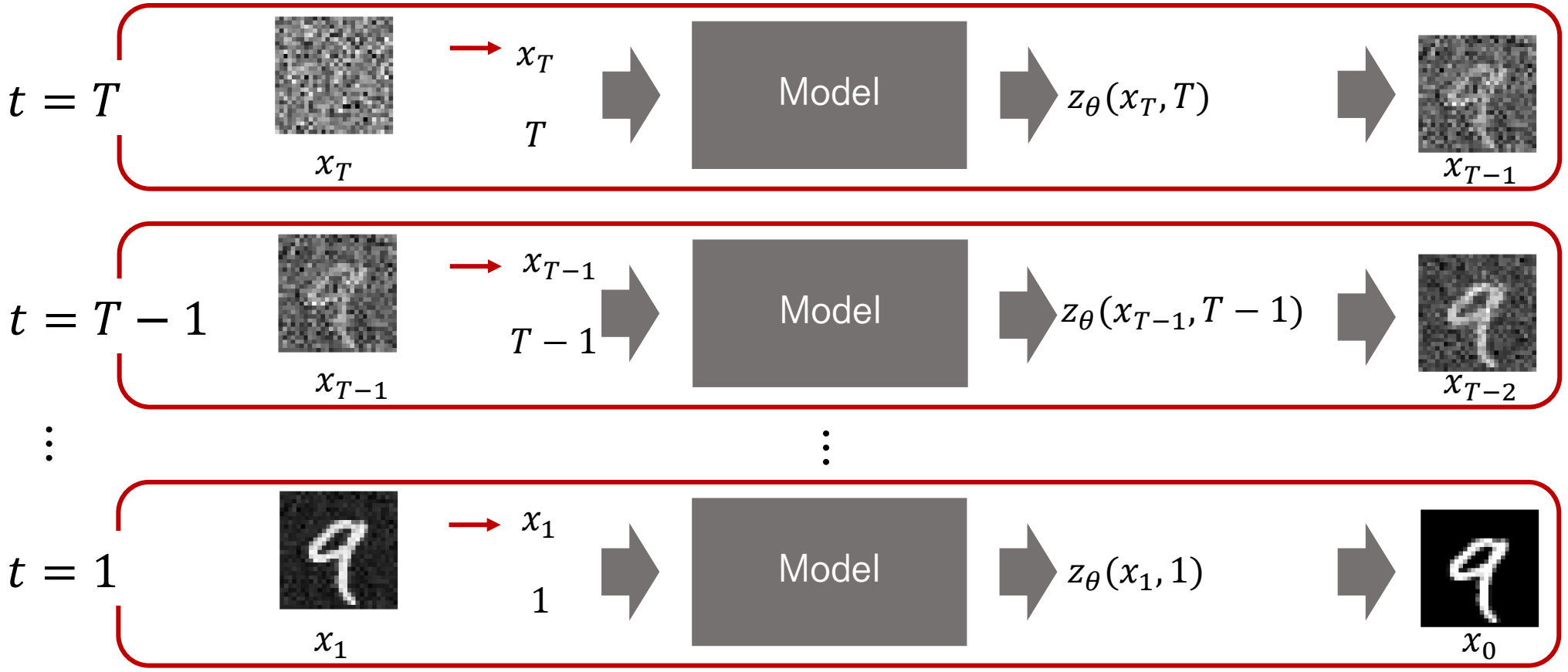
Diffusion Models

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** $\mathbf{x}_0$

Denoising Diffusion Probabilistic Models (DDPM)

- Testing: x_T에서 x_0 생성

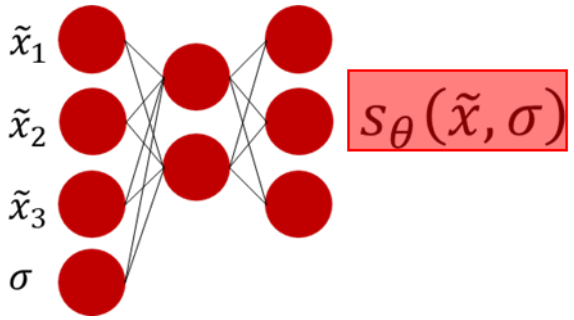


Diffusion Models

NCSN vs DDPM

- Training: 목적식이 유사

NCSN



$s_\theta(\tilde{x}, \sigma)$

$$\frac{1}{2} E_{q_\sigma(\tilde{x}|x) p_{data}(x)} \left[\left\| s_\theta(\tilde{x}, \sigma) - \frac{\tilde{x} - x}{\sigma^2} \right\|_2^2 \right]$$

DDPM

x_0

x_t

x_t

t

Model

$z_\theta(x_t, t)$

$$E_{x_0, z}[\| z_t - z_\theta(x_t, t) \|^2]$$

Diffusion Models

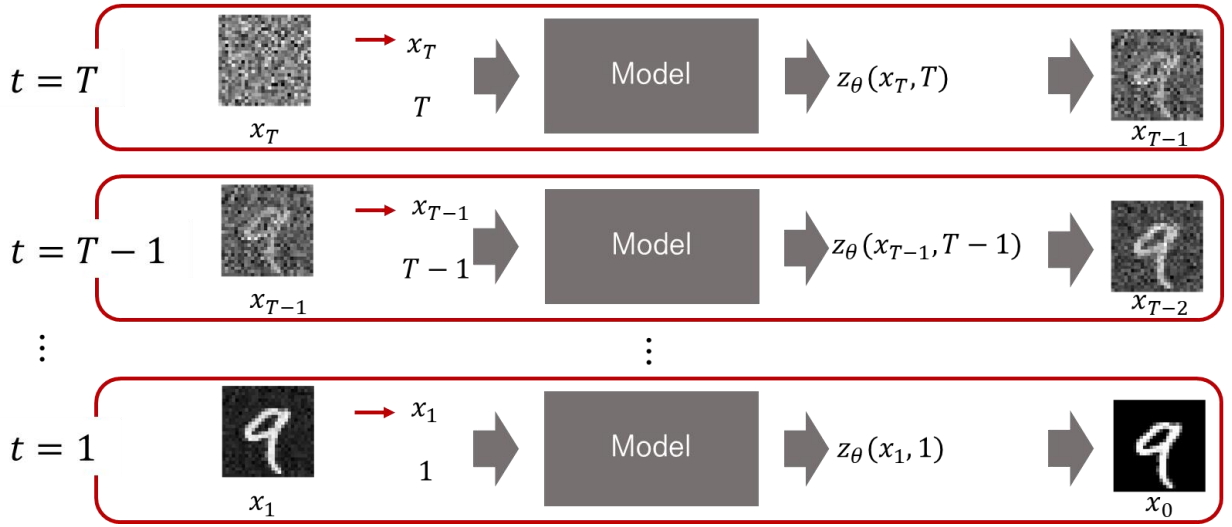
NCSN vs DDPM

- Testing: 데이터를 생성하는 식이 유사

$$\tilde{x}_t = \tilde{x}_{t-1} + \frac{\epsilon}{2}\nabla_x log p(\tilde{x}_{t-1}) + \sqrt{\epsilon} z_t$$

이전시점 data　　Score　　Random Noise

NCSN



DDPM

Score-based Generative Models Through SDEs

Score-based Generative Models Through Stochastic Differential Equations

- ICLR 2021 (Outstanding Paper Award)
- 2022년 02월 10일 기준: 121회 인용

SCORE-BASED GENERATIVE MODELING THROUGH STOCHASTIC DIFFERENTIAL EQUATIONS

Yang Song*
Stanford University
yangsong@cs.stanford.edu

Jascha Sohl-Dickstein
Google Brain
jaschasd@google.com

Diederik P. Kingma
Google Brain
durk@google.com

Abhishek Kumar
Google Brain
abhishk@google.com

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Ben Poole
Google Brain
pooleb@google.com

ABSTRACT

Creating noise from data is easy; creating data from noise is generative modeling. We present a stochastic differential equation (SDE) that smoothly transforms a complex data distribution to a known prior distribution by slowly injecting noise, and a corresponding reverse-time SDE that transforms the prior distribution back into the data distribution by slowly removing the noise. Crucially, the reverse-time SDE depends only on the time-dependent gradient field (a.k.a., score) of the perturbed data distribution. By leveraging advances in score-based generative modeling, we can accurately estimate these scores with neural networks, and use numerical SDE solvers to generate samples. We show that this framework encapsulates previous approaches in score-based generative modeling and diffusion probabilistic modeling, allowing for new sampling procedures and new modeling capabilities. In particular, we introduce a predictor-corrector framework to correct errors in the evolution of the discretized reverse-time SDE. We also derive an equivalent neural ODE that samples from the same distribution as the SDE, but additionally enables exact likelihood computation, and improved sampling efficiency. In addition, we provide a new way to solve inverse problems with score-based models, as demonstrated with experiments on class-conditional generation, image inpainting, and colorization. Combined with multiple architectural improvements, we achieve record-breaking performance for unconditional image generation on CIFAR-10 with an Inception score of 9.89 and FID of 2.20, a competitive likelihood of 2.99 bits/dim, and demonstrate high fidelity generation of 1024×1024 images for the first time from a score-based generative model.

Score-based Generative Modeling with SDEs

ODE and SDE

- 방정식 → **Solve** → 해
- Ordinary Differential Equation(ODE) → **Solve** → x(t)

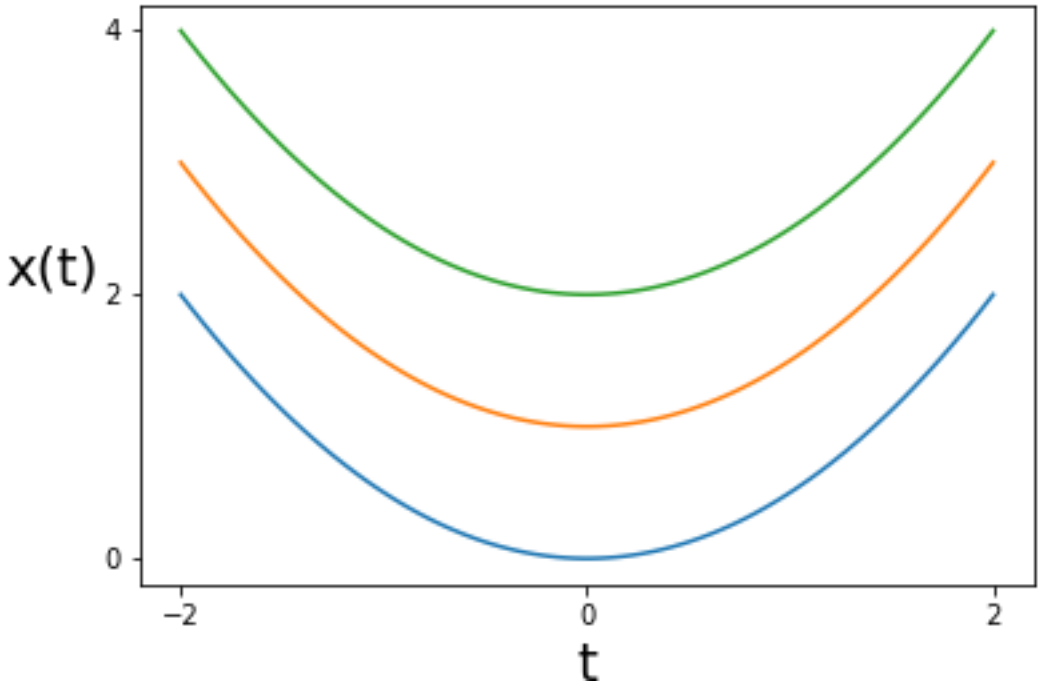
$$dx = f(x,t)dt$$

Example

① $\frac{dx}{dt} = t$

② $\int \frac{dx}{dt} dt = \int t\, dt$

③ $x(t) = \frac{1}{2}t^2 + C$



Score-based Generative Modeling with SDEs

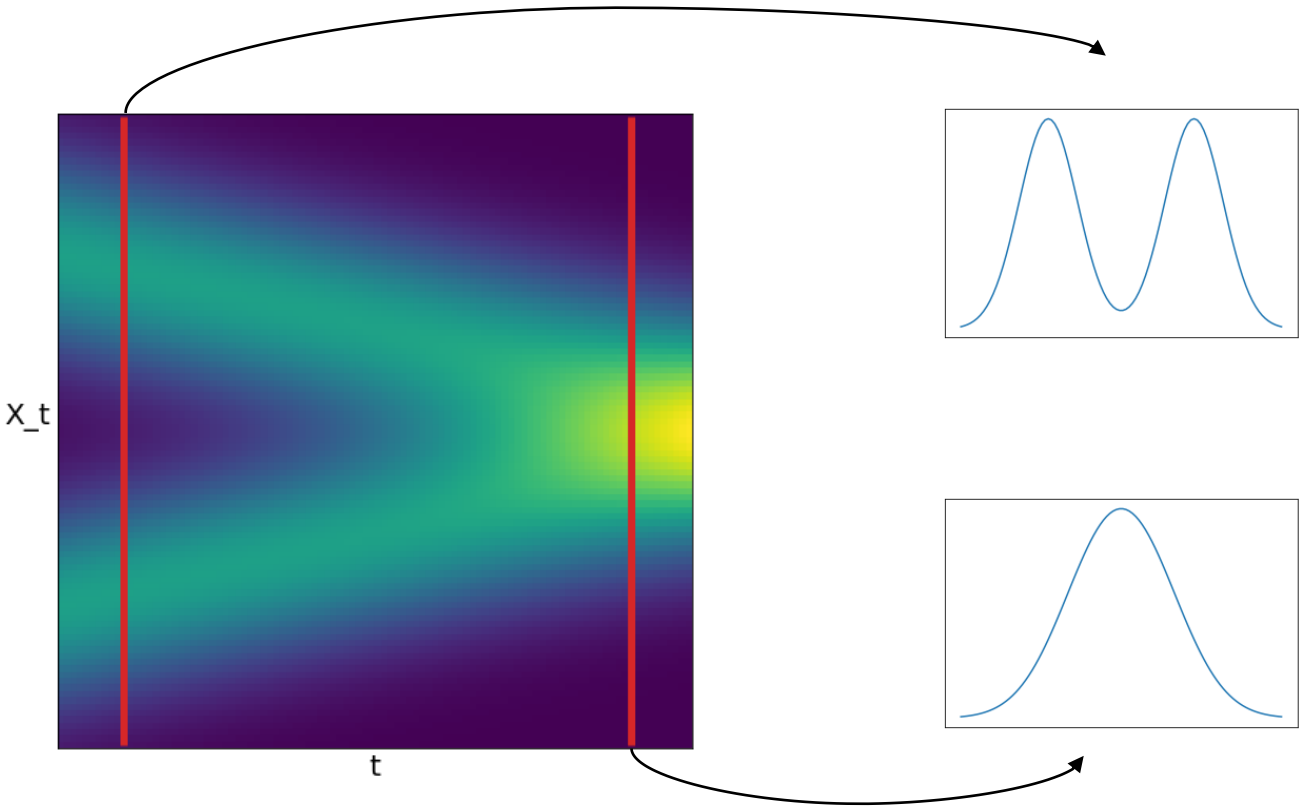
ODE and SDE

- Stochastic Differential Equation(SDE) → **Solve** → $\{X_t\}$ (Random process)

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

ODE Randomness

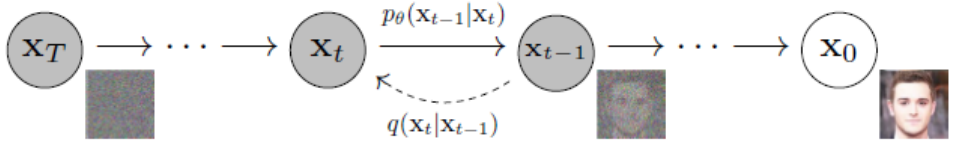
Solve



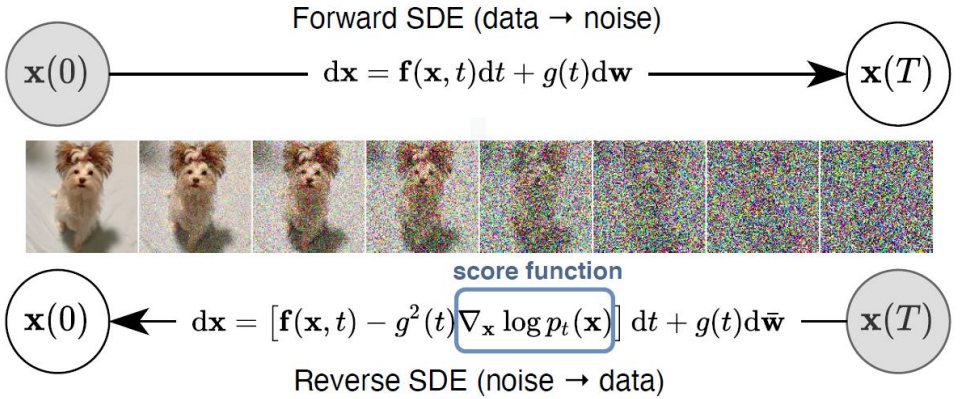
Score-based Generative Modeling with SDEs

Score-based Generative Modeling with SDEs

- NCSN/DDPM의 continuous 버전
- Forward SDE: 노이즈를 추가하는 과정
- Reverse SDE: 노이즈를 제거하는 과정
- Reverse SDE에 score function이 존재



DDPM



DDPM cont

Score-based Generative Modeling with SDEs

Score-based Generative Modeling with SDEs

- Forward SDE의 정의에 따라서 NCSN과 DDPM
- NCSN → Variance Exploding SDE (VE-SDE)
- DDPM → Variance Preserving SDE (VP-SDE)

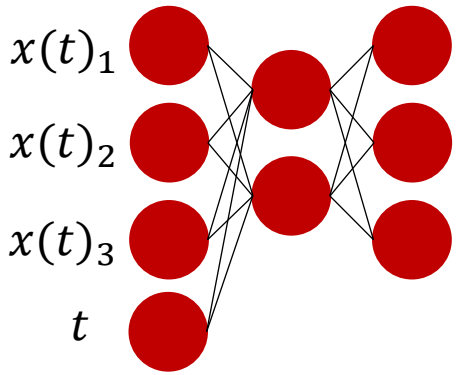
NCSN $x_i = x_{i-1} + \sqrt{\sigma_i^2 - \sigma_{i-1}^2} z_{i-1}$ → $dx = \sqrt{\frac{d[\sigma^2(t)]}{dt}} dw$ VE-SDE

DDPM $x_i = \sqrt{1-\beta_i} x_{i-1} + \sqrt{\beta_i} z_{i-1}$ → $dx = -\frac{1}{2}\beta(t)x\,dt + \sqrt{\beta(t)}dw$ VP-SDE

Score-based Generative Modeling with SDEs

Score-based Generative Modeling with SDEs

- **Training**: Score network 학습
- 네트워크 깊이 ↑ (NCSN++ cont, DDPM++ cont)



$s_\theta(x(t), t)$

NCSN++

Network Output Score

$$\frac{1}{2} E_{p(x(t)|x(0))p(x(0)} [\| s_\theta(x(t), t) - \nabla_x log p(x(t)|x(0)) \|_2^2]$$

Score-based Generative Modeling with SDEs

Score-based Generative Modeling with SDEs

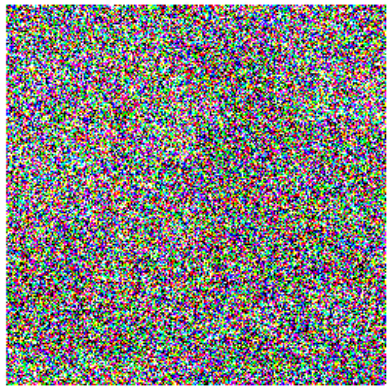
- **Testing**: Solve Reverse SDE
- SDE를 풀 수 있는 다양한 solver들 적용 가능 (Euler-Maruyama, Runge-Kutta …)
- Predictor: Reverse SDE를 풀어서 데이터 생성
- Corrector: Predictor에서 생성된 데이터에 Annealed Langevin dynamics 적용

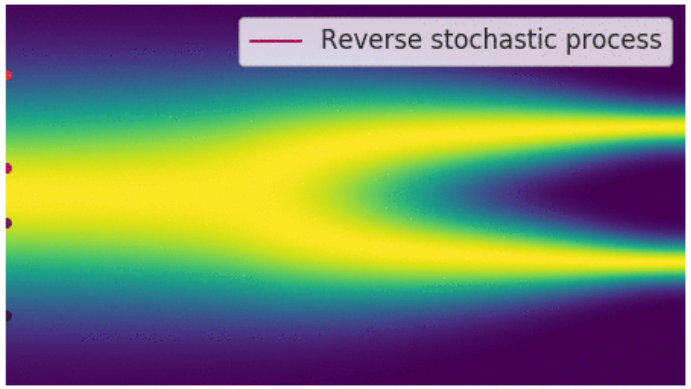
Algorithm 2 PC sampling (VE SDE)

$\mathbf{x}_N \sim \mathcal{N}(0, \sigma_{\max}^2 \mathbf{I})$
for $i = N-1$ **to** 0 **do**
  $\mathbf{x}'_i \leftarrow \mathbf{x}_{i+1} + (\sigma_{i+1}^2 - \sigma_i^2)\mathbf{s}_{\theta*}(\mathbf{x}_{i+1}, \sigma_{i+1})$
  $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$
  $\mathbf{x}_i \leftarrow \mathbf{x}'_i + \sqrt{\sigma_{i+1}^2 - \sigma_i^2}\mathbf{z}$
  for $j = 1$ **to** M **do**
    $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$
    $\mathbf{x}_i \leftarrow \mathbf{x}_i + \epsilon_i \mathbf{s}_{\theta*}(\mathbf{x}_i, \sigma_i) + \sqrt{2\epsilon_i}\mathbf{z}$
return $\mathbf{x}_0$

Algorithm 3 PC sampling (VP SDE)

$\mathbf{x}_N \sim \mathcal{N}(0, \mathbf{I})$
for $i = N-1$ **to** 0 **do**
  $\mathbf{x}'_i \leftarrow (2 - \sqrt{1-\beta_{i+1}})\mathbf{x}_{i+1} + \beta_{i+1}\mathbf{s}_{\theta*}(\mathbf{x}_{i+1}, i+1)$
  $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$
  $\mathbf{x}_i \leftarrow \mathbf{x}'_i + \sqrt{\beta_{i+1}}\mathbf{z}$ — Predictor
  for $j = 1$ **to** M **do** — Corrector
    $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$
    $\mathbf{x}_i \leftarrow \mathbf{x}_i + \epsilon_i \mathbf{s}_{\theta*}(\mathbf{x}_i, i) + \sqrt{2\epsilon_i}\mathbf{z}$
return $\mathbf{x}_0$





https://yang-song.github.io/blog/2021/score/

Conclusion

Score-based Generative Models and Diffusion Models

- Score-based Generative Models (NCSN)

  랜덤 노이즈에서 시작해 score값을 따라 높은 확률값이 있는 공간에서 데이터 생성

- Diffusion Models (DDPM)

  노이즈를 제거하는 과정을 학습해 랜덤 노이즈로부터 데이터 생성

- Score-based Generative Modeling with SDEs

  SDE라는 구조 내에서 NCSN과 DDPM을 통합

Appendix 1

VAE Loss 유도

$$
\begin{aligned}
& D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \\
&= \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})} d\mathbf{z} \\
&= \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x}) p_\theta(\mathbf{x})}{p_\theta(\mathbf{z}, \mathbf{x})} d\mathbf{z} && ; \text{Because } p(z|x)=p(z,x)/p(x) \\
&= \int q_\phi(\mathbf{z}|\mathbf{x}) \big(\log p_\theta(\mathbf{x}) + \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}, \mathbf{x})} \big) d\mathbf{z} \\
&= \log p_\theta(\mathbf{x}) + \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}, \mathbf{x})} d\mathbf{z} && ; \text{Because } \int q(z|x)dz=1 \\
&= \log p_\theta(\mathbf{x}) + \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{x}|\mathbf{z}) p_\theta(\mathbf{z})} d\mathbf{z} && ; \text{Because } p(z,x)=p(x|z)p(z) \\
&= \log p_\theta(\mathbf{x}) + \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} [\log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z})} - \log p_\theta(\mathbf{x}|\mathbf{z})] \\
&= \log p_\theta(\mathbf{x}) + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})) - \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})
\end{aligned}
$$

https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html#vae-variational-autoencoder

Appendix 2

DDPM Loss 유도 (1)

$$
\begin{aligned}
-\log p_\theta(\mathbf{x}_0) &\leq -\log p_\theta(\mathbf{x}_0) + D_{\mathrm{KL}}(q(\mathbf{x}_{1:T}|\mathbf{x}_0) \| p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0)) \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_{\mathbf{x}_{1:T}\sim q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})/p_\theta(\mathbf{x}_0)}\right] \\
&= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_q\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} + \log p_\theta(\mathbf{x}_0)\right] \\
&= \mathbb{E}_q\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\right] \\
\text{Let } L_{\mathrm{VLB}} &= \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\right] \geq -\mathbb{E}_{q(\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0)
\end{aligned}
$$

$$
\begin{aligned}
L_{\mathrm{VLB}} &= \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\right] \\
&= \mathbb{E}_q\left[\log \frac{\prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_T)\prod_{t=1}^T p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\right] \\
&= \mathbb{E}_q\left[-\log p_\theta(\mathbf{x}_T) + \sum_{t=1}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}\right] \\
&= \mathbb{E}_q\left[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1})}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\right] \\
&= \mathbb{E}_q\left[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \left(\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} \cdot \frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}\right) + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\right] \\
&= \mathbb{E}_q\left[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \sum_{t=2}^T \log \frac{q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\right] \\
&= \mathbb{E}_q\left[-\log p_\theta(\mathbf{x}_T) + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} + \log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{q(\mathbf{x}_1|\mathbf{x}_0)} + \log \frac{q(\mathbf{x}_1|\mathbf{x}_0)}{p_\theta(\mathbf{x}_0|\mathbf{x}_1)}\right] \\
&= \mathbb{E}_q\left[\log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{p_\theta(\mathbf{x}_T)} + \sum_{t=2}^T \log \frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0)}{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)} - \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)\right] \\
&= \mathbb{E}_q[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_T))}_{L_T} + \sum_{t=2}^T \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{-\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}]
\end{aligned}
$$

https://lilianweng.github.io/lil-log/2021/07/11/diffusion-models.html#what-are-diffusion-models

Appendix 3

DDPM Loss 유도 (2)

It is noteworthy that the reverse conditional probability is tractable when conditioned on $\mathbf{x}_0$:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I})$$

Using Bayes' rule, we have:

$$\begin{aligned} q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) &= q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)} \\ &\propto \exp\Big(-\frac{1}{2}\big(\frac{(\mathbf{x}_t - \sqrt{\alpha_t}\mathbf{x}_{t-1})^2}{\beta_t} + \frac{(\mathbf{x}_{t-1} - \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0)^2}{1-\bar{\alpha}_{t-1}} - \frac{(\mathbf{x}_t - \sqrt{\bar{\alpha}_t}\mathbf{x}_0)^2}{1-\bar{\alpha}_t}\big)\Big) \\ &= \exp\Big(-\frac{1}{2}\big((\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}})\mathbf{x}_{t-1}^2 - (\frac{2\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{2\sqrt{\bar{\alpha}_t}}{1-\bar{\alpha}_t}\mathbf{x}_0)\mathbf{x}_{t-1} + C(\mathbf{x}_t, \mathbf{x}_0)\big)\Big) \end{aligned}$$

where $C(\mathbf{x}_t, \mathbf{x}_0)$ is some function not involving $\mathbf{x}_{t-1}$ and details are omitted. Following the standard Gaussian density function, the mean and variance can be parameterized as follows:

$$\begin{aligned} \tilde{\beta}_t &= 1/(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}) = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t \\ \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) &= (\frac{\sqrt{\alpha_t}}{\beta_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_t}}{1-\bar{\alpha}_t}\mathbf{x}_0)/(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}}) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 \end{aligned}$$

Thanks to the nice property, we can represent $\mathbf{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\mathbf{z}_t)$ and plug it into the above equation and obtain:

$$\begin{aligned} \tilde{\boldsymbol{\mu}}_t &= \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\mathbf{z}_t) \\ &= \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\mathbf{z}_t\Big) \end{aligned}$$

Parameterization of L_t for Training Loss

Recall that we need to learn a neural network to approximate the conditioned probability distributions in the reverse diffusion process, $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$. We would like to train $\boldsymbol{\mu}_\theta$ to predict $\tilde{\boldsymbol{\mu}}_t = \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\mathbf{z}_t\Big)$. Because $\mathbf{x}_t$ is available as input at training time, we can reparameterize the Gaussian noise term instead to make it predict $\mathbf{z}_t$ from the input $\mathbf{x}_t$ at time step t:

$$\begin{aligned} \boldsymbol{\mu}_\theta(\mathbf{x}_t, t) &= \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\mathbf{z}_\theta(\mathbf{x}_t, t)\Big) \\ \text{Thus } \mathbf{x}_{t-1} &= \mathcal{N}(\mathbf{x}_{t-1}; \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\mathbf{z}_\theta(\mathbf{x}_t, t)\Big), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)) \end{aligned}$$

The loss term L_t is parameterized to minimize the difference from $\tilde{\boldsymbol{\mu}}$:

$$\begin{aligned} L_t &= \mathbb{E}_{\mathbf{x}_0, \mathbf{z}}\Big[\frac{1}{2\|\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)\|_2^2}\|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_\theta(\mathbf{x}_t, t)\|^2\Big] \\ &= \mathbb{E}_{\mathbf{x}_0, \mathbf{z}}\Big[\frac{1}{2\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\mathbf{z}_t\Big) - \frac{1}{\sqrt{\alpha_t}}\Big(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\mathbf{z}_\theta(\mathbf{x}_t, t)\Big)\|^2\Big] \\ &= \mathbb{E}_{\mathbf{x}_0, \mathbf{z}}\Big[\frac{\beta_t^2}{2\alpha_t(1-\bar{\alpha}_t)\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\mathbf{z}_t - \mathbf{z}_\theta(\mathbf{x}_t, t)\|^2\Big] \\ &= \mathbb{E}_{\mathbf{x}_0, \mathbf{z}}\Big[\frac{\beta_t^2}{2\alpha_t(1-\bar{\alpha}_t)\|\boldsymbol{\Sigma}_\theta\|_2^2}\|\mathbf{z}_t - \mathbf{z}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\mathbf{z}_t, t)\|^2\Big] \end{aligned}$$

Simplification

Empirically, Ho et al. (2020) found that training the diffusion model works better with a simplified objective that ignores the weighting term:

$$L_t^{\text{simple}} = \mathbb{E}_{\mathbf{x}_0, \mathbf{z}_t}\Big[\|\mathbf{z}_t - \mathbf{z}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\mathbf{z}_t, t)\|^2\Big]$$

The final simple objective is:

$$L_{\text{simple}} = L_t^{\text{simple}} + C$$

where C is a constant not depending on θ.

https://lilianweng.github.io/lil-log/2021/07/11/diffusion-models.html#what-are-diffusion-models