



Denoising Diffusion Probabilistic Model

Suk-Hwan Lee

Artificial Intelligence

Creating the Future

Dong-A University

**Division of Computer Engineering &
Artificial Intelligence**

References

Jonathan Ho, Ajay Jain, Pieter Abbeel, "Denoising Diffusion Probabilistic Model," NeurIPS 2020
UC Berkeley

<https://github.com/hojonathanho/diffusion>
<https://hojonathanho.github.io/diffusion/>

[DPM]

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

[DDPM]

PR12-409, https://www.youtube.com/watch?v=1j0W_lu55nc
<http://dsba.korea.ac.kr/seminar/?mod=document&uid=2352>

[Blog]

<https://happy-jihye.github.io/diffusion/diffusion-1/>

Denoising Diffusion Probabilistic Models (DDPM)

Jonathan Ho, Ajay Jain, Pieter Abbeel
NeurIPS 2020
UC Berkeley
<https://github.com/hojonathanho/diffusion>
<https://hojonathanho.github.io/diffusion/>

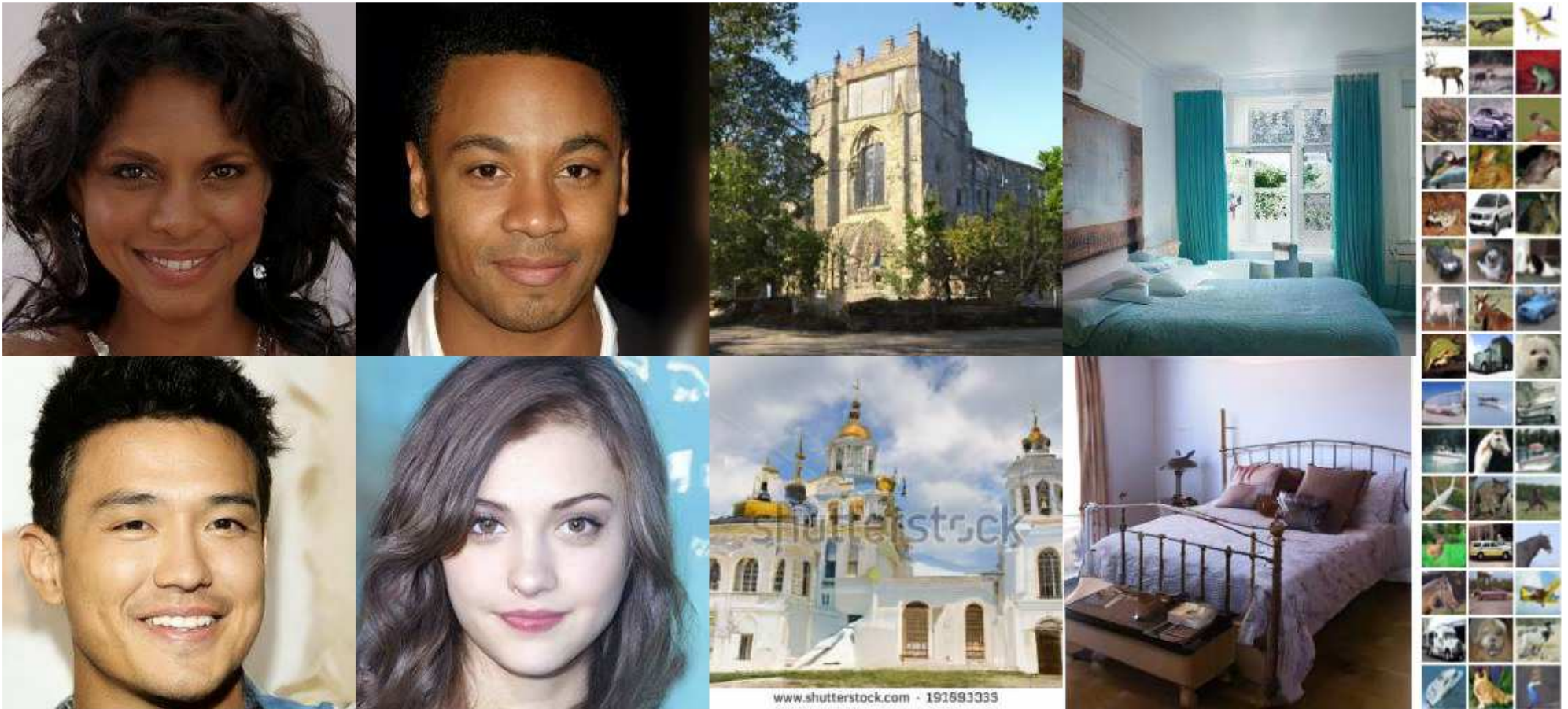
PR12-409, https://www.youtube.com/watch?v=1j0W_lu55nc

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**.
- 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.

Denoising Diffusion Probabilistic Models (DDPM)

PR12-409, https://www.youtube.com/watch?v=1j0W_lu55nc



Denoising Diffusion Probabilistic Models (DDPM)

Generative Models

PR12-409, https://www.youtube.com/watch?v=1j0W_lu55nc

“To know the distribution(manifold) of data”

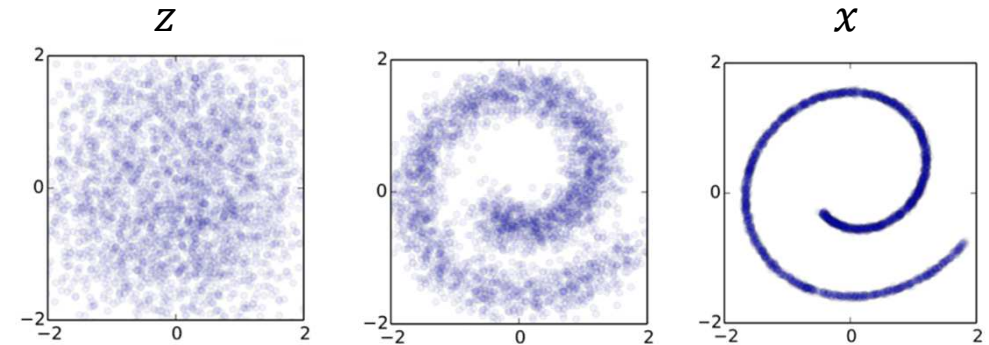
Task the sampling from Distribution

Variational Autoencoders

GANs

Normalizing Flow

<https://www.youtube.com/watch?v=TywKpiZJ7P4>



High Entropy
Low Information

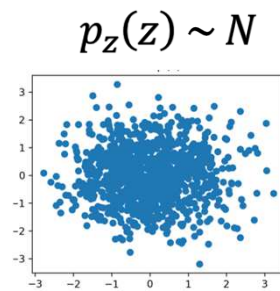
Low Entropy
High Information

Denoising Diffusion Probabilistic Models (DDPM)

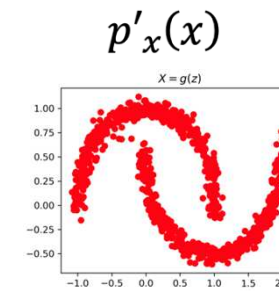
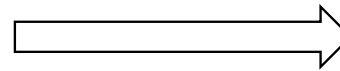
Generative Models

<http://dsba.korea.ac.kr/seminar/?mod=document&uid=2352>

Generative Model : Latent variable model



Trained model



- ✓ *Simple distribution*
- ✓ *Tractable(Gaussian)*

Input

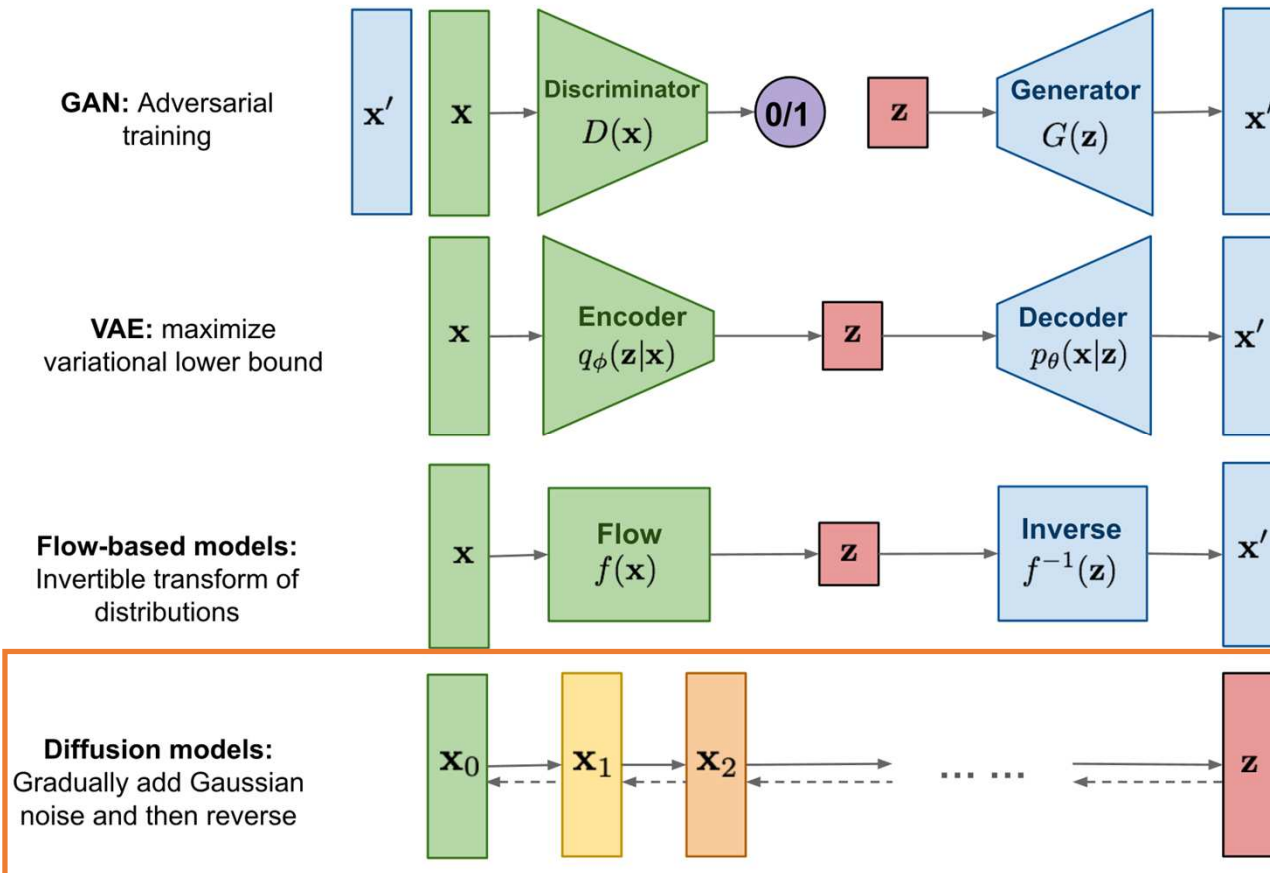
- ✓ *Complex distribution*
- ✓ *Visual, Audio patterns*

Output

- 결국 생성 모델로부터 원하는 것은 매우 간단한 분포(z)를 특정한 패턴을 갖는 분포로 변환(mapping, transformation, sampling)하는 것
- 그렇기에 대부분의 생성모델이 주어진 입력 데이터로부터 latent variable(z)을 얻어내고, 이를 변환하는 역량을 학습하고자 함

Denoising Diffusion Probabilistic Models (DDPM)

Generative Models



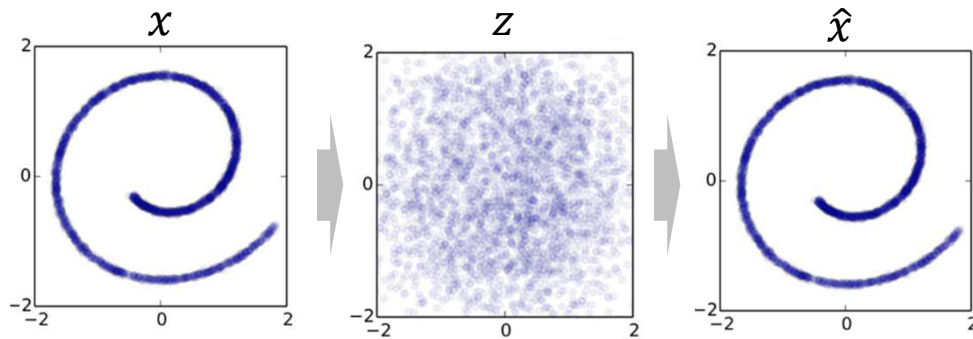
- 반복적인 변화(**iterative transformation**)를 활용한다는 점에서 **Flow-based models** 와 유사
- 분포에 대한 **Variational Inference**을 통한 학습을 진행한다는 점은 **VAE**와 유사
- 최근에는 Diffusion 모델의 학습에 Adversarial training을 활용하기도 함 ([Diffusion-GAN, 2022](#))

Denoising Diffusion Probabilistic Models (DDPM)

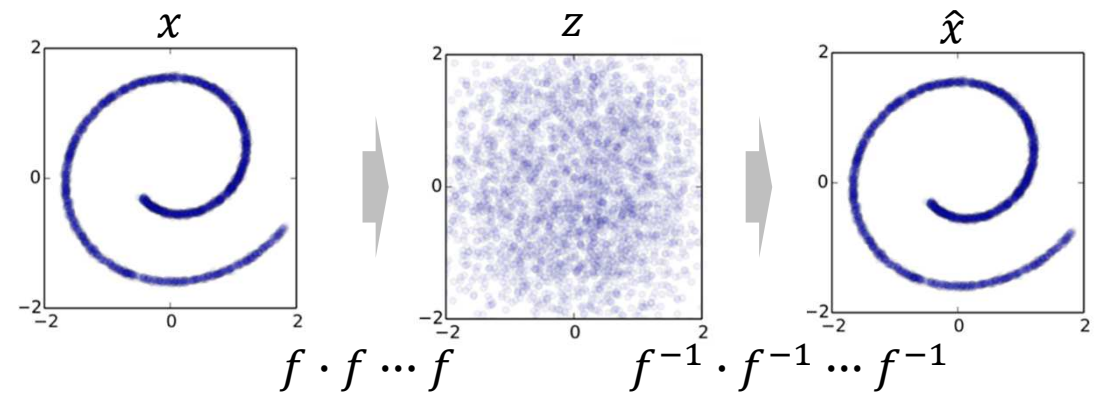
Generative Models

PR12-409, https://www.youtube.com/watch?v=1j0W_lu55nc

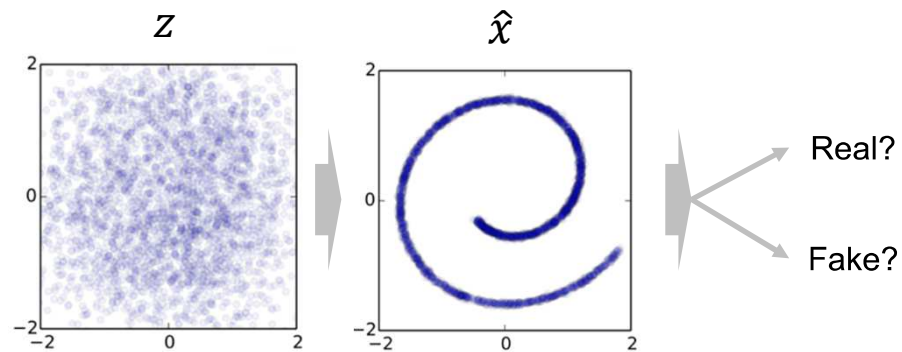
Variational Autoencoders



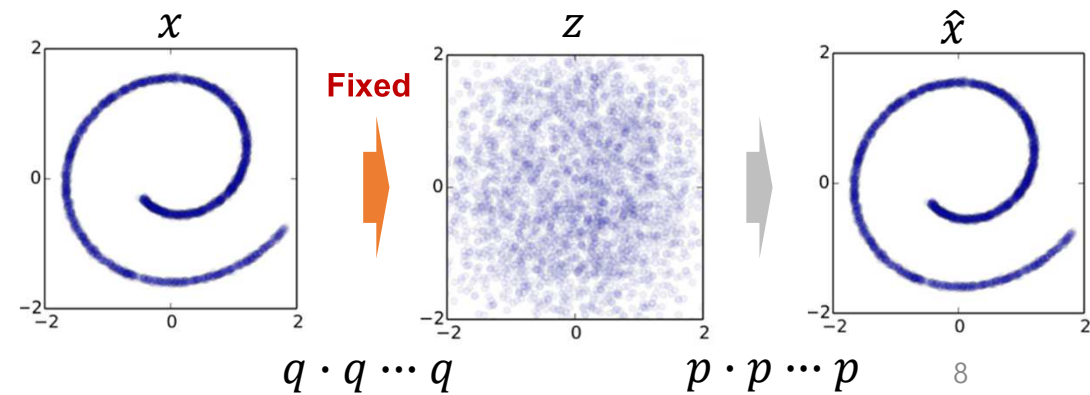
Normalizing Flow



GANs

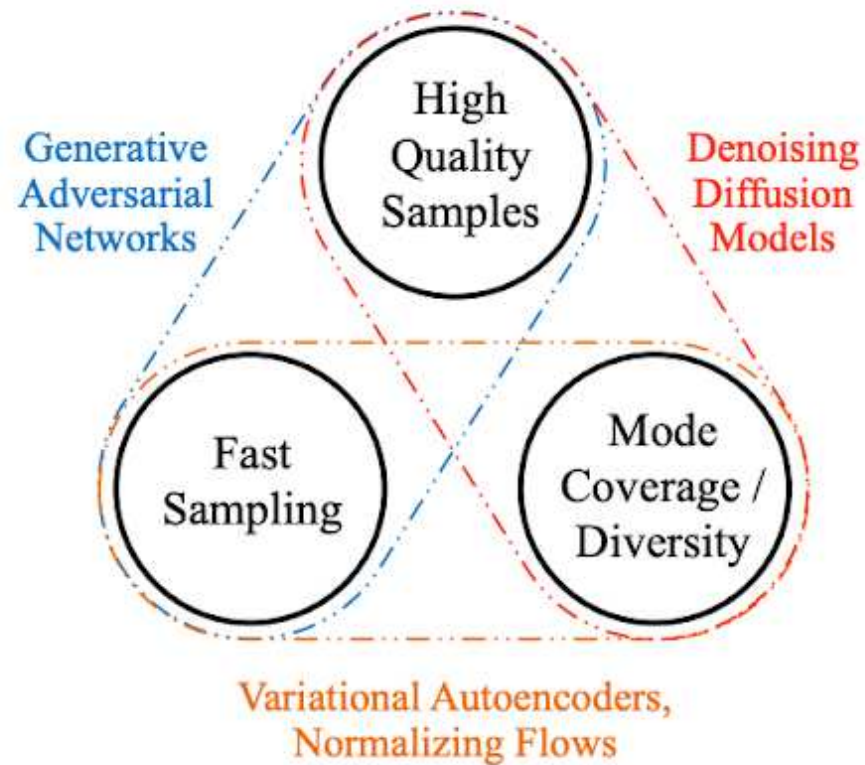


Diffusion Model



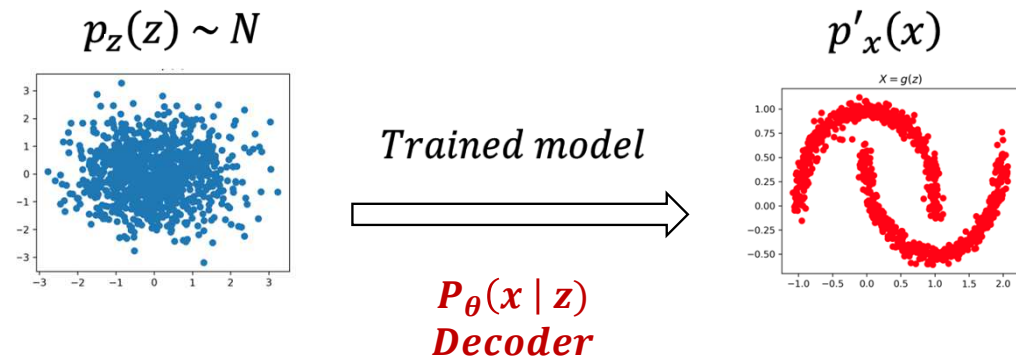
Denoising Diffusion Probabilistic Models (DDPM)

Generative Models



Generative Models

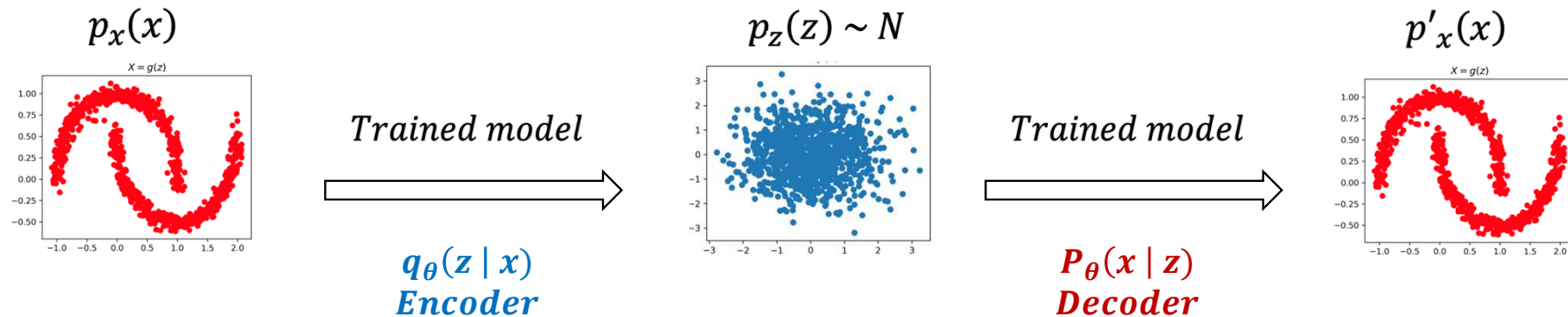
Variational Autoencoders



- 학습된 **Decoder network**를 통해 latent variable을 특정한 패턴의 분포로 mapping

Generative Models

Variational Autoencoders

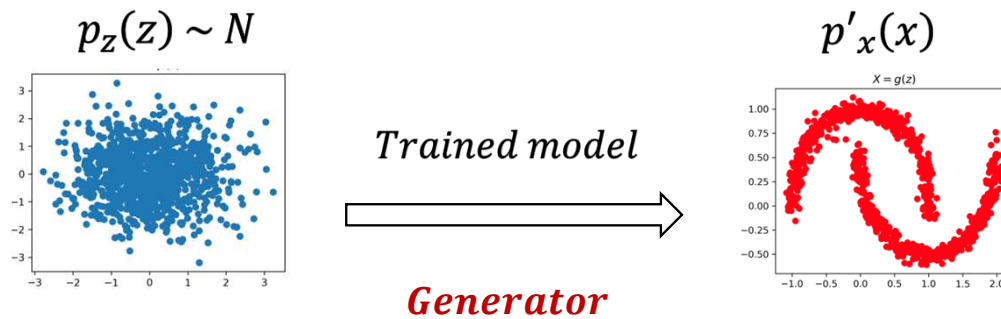


- **Encoder**를 모델 구조에 추가해, **Latent variable** / **Encoder** / **Decoder**를 모두 학습

$$P_\theta(x) = \frac{P_\theta(x, z)}{p_\theta(z|x)} = \frac{P_\theta(x|z)P_\theta(z)}{p_\theta(z|x)} = \text{likelihood}$$

Generative Models

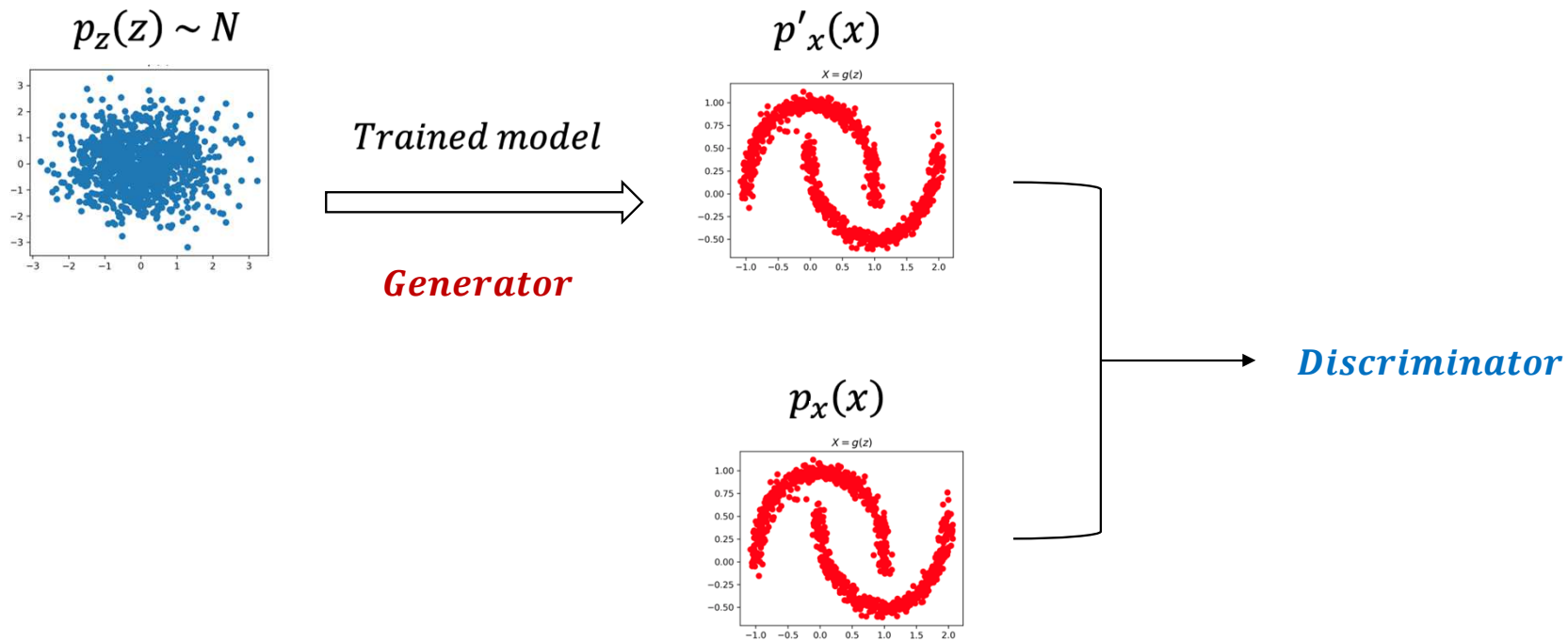
GAN



- 학습된 **Generator**를 통해 latent variable을 특정한 패턴의 분포로 mapping

Generative Models

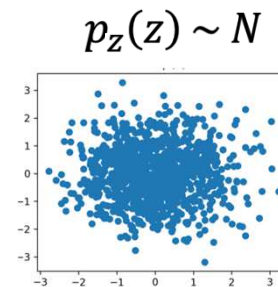
GAN



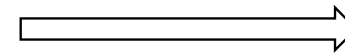
- **Discriminator**를 모델 구조에 추가해, **Generator**를 학습

Generative Models

Flow-based Model



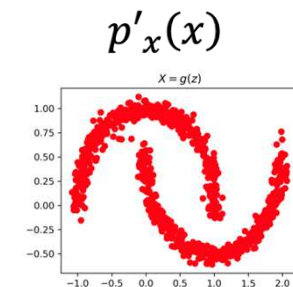
Trained model



$$f^{-1} \circ f^{-1} \circ \dots \circ f^{-1}$$

Flow (inverse)

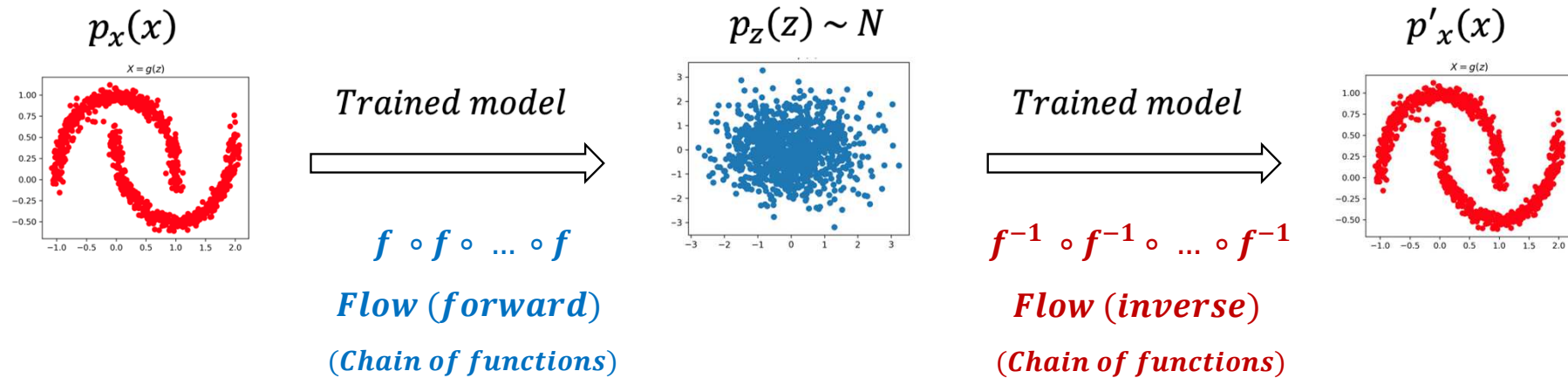
(Chain of functions)



- 학습된 **Flow model**의 **Inverse mapping**을 통해 latent variable을 특정한 패턴의 분포로 mapping

Generative Models

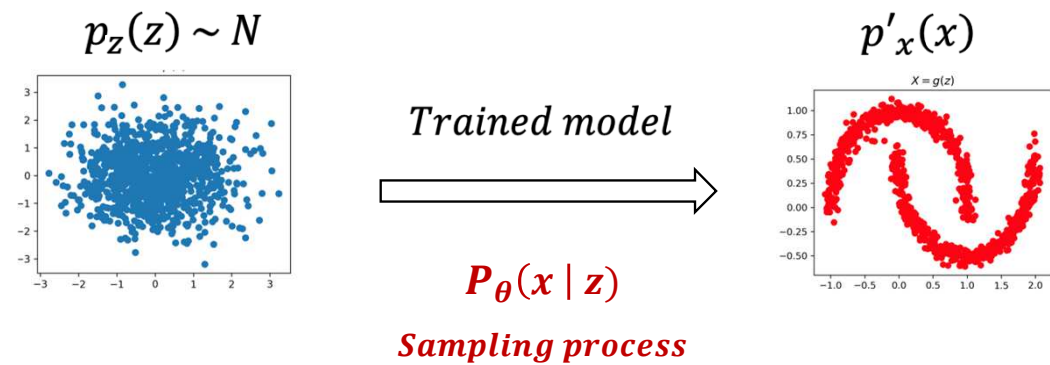
Flow-based Model



- 생성에 활용되는 **Inverse mapping**을 학습하기 위해 **Invertible function**을 학습

Generative Models

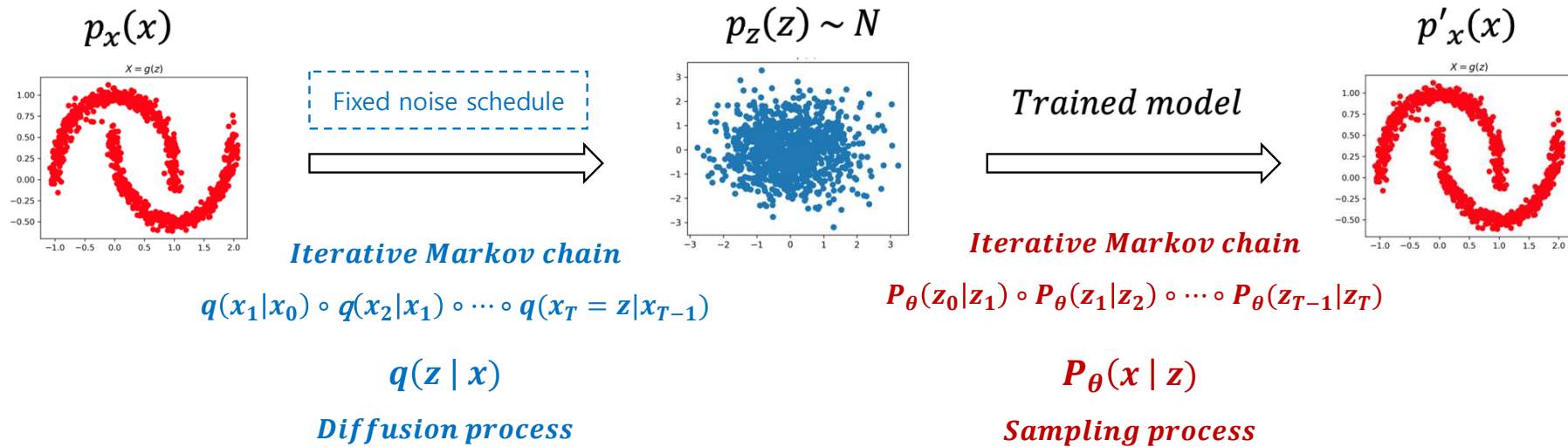
Diffusion based generative model



- 학습된 **Diffusion model**의 조건부 확률 분포 $P_\theta(x | z)$ 를 통해 특정한 패턴의 분포 획득

Generative Models

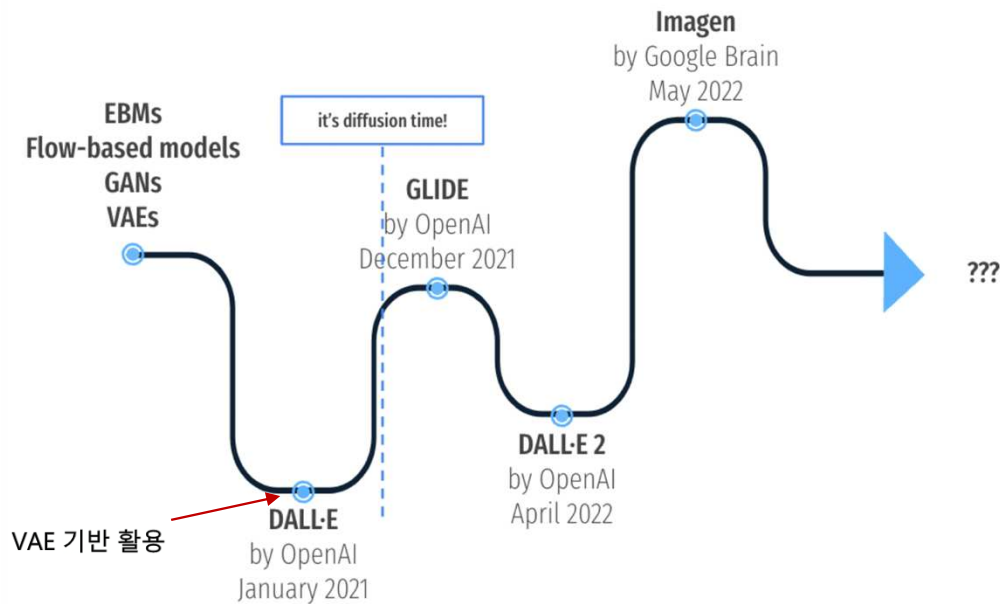
Diffusion based generative model



- 생성에 활용되는 조건부 확률 분포 $P_\theta(x | z)$ 을 학습하기 위해 *Diffusion process* $q(z | x)$ 를 활용

Diffusion Model Overview

➤ Diffusion Model 활용 : Text-to-Image generation



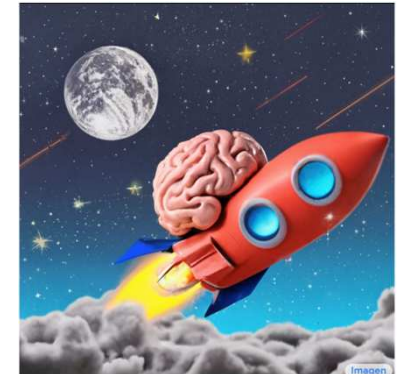
*"An astronaut riding a horse
in a photorealistic style"*

DALL-E 2



*"A brain riding a rocketship
heading towards the moon"*

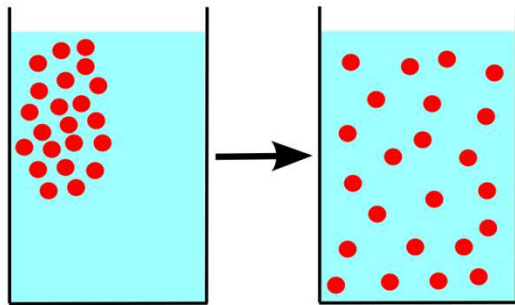
Imagen



- 2021년부터 Diffusion based generative model을 활용한 다양한 Text-to-Image generation 방법론이 제안되며 눈길을 끄
- 이미지, 오디오 등 다양한 양식의 벤치마크 데이터에서 리더보드 상위권에 오르며 generation 성능을 확인함

Diffusion Model Overview

➤ Diffusion = '확산'



$$q(\mathbf{x}^{(0:T)})$$

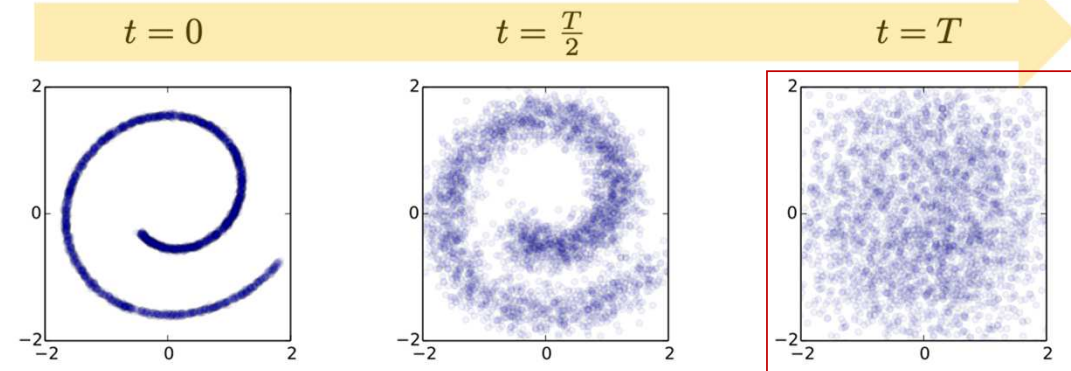
- 물리 통계 동역학("Thermodynamics")에서 특정한 물질 (액, 기체)이 조금씩 번지며 같은 농도로 바뀌는 현상
- 물질들의 특정한 분포가 서서히 와해되는 과정
- *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*(2015, ICML)에서 비지도학습을 위한 방법론으로 첫 활용

From *Deep Unsupervised Learning using Nonequilibrium Thermodynamics*(2015, ICML)

Diffusion process

Explicit Pattern

Gaussian Noise



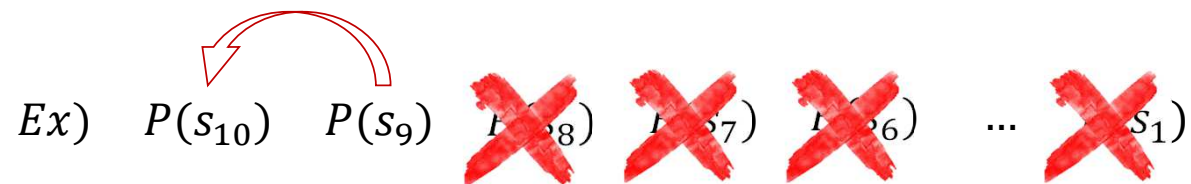
- 특정한 데이터의 패턴이 서서히 반복적인 과정을 거쳐 와해되는 과정("농도가 균일해지는")을 'Diffusion process'라 명명

Diffusion Model Overview

➤ Markov Chain

- **Markov 성질**을 갖는 **이산 확률과정**
 - ✓ **Markov 성질** : “특정 상태의 확률($t+1$)은 오직 현재(t)의 상태에 의존한다”
 - ✓ **이산 확률과정** : 이산적인 시간(0초, 1초, 2초, ..) 속에서의 확률적 현상

$$P[s_{t+1}|s_t] = P[s_{t+1}|s_1, \dots, s_t]$$



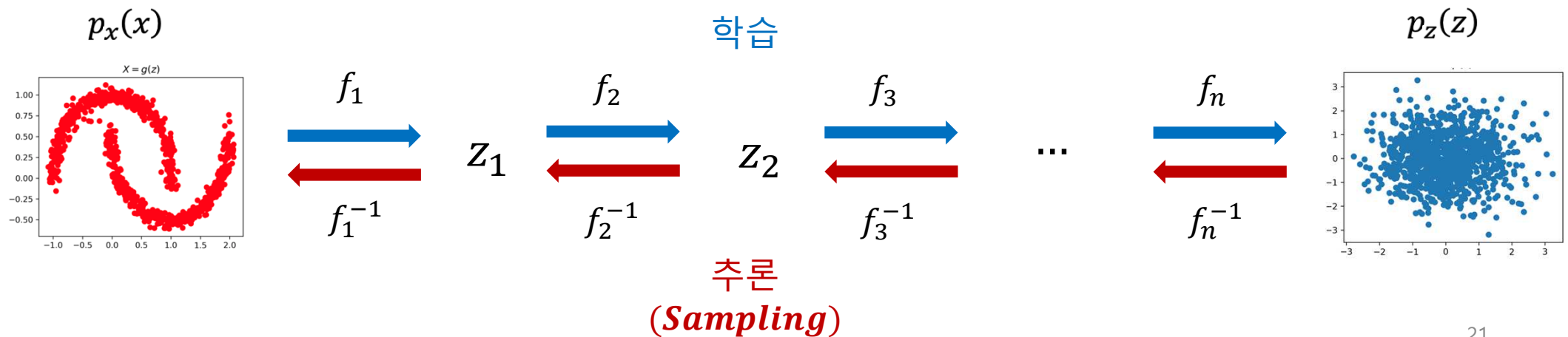
- ✓ 예 : “내일의 날씨는 오늘의 날씨만 보고 알 수 있다.” (내일의 날씨는 오로지 오늘의 날씨 만을 조건부로 하는 확률적 과정)

Diffusion Model Overview

➤ Normalizing Flow

- 심층 신경망 기반 확률적 생성 모형 중 하나
- 잠재 변수(z) 기반 확률적 생성모형으로서, 잠재 변수(z) 획득에 '변수 변환' 공식을 활용

✓ 변수변환 공식
$$p_x(x) = p_z(z) \left| \frac{dz}{dx} \right|$$



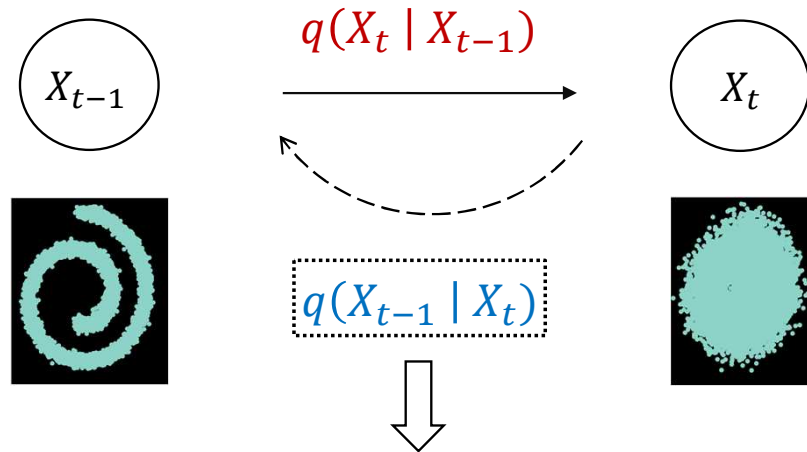
Diffusion Model Overview

➤ Diffusion Model

- **Diffusion model**은 Generative model로서 학습된 데이터의 패턴을 생성해내는 역할을 함
- 패턴 생성 과정을 학습하기 위해 고의적으로 패턴을 무너트리고(Noising), 이를 다시 복원하는 조건부 PDF를 학습(Denoising)

Diffusion process

Reverse process



$q(X_t | X_{t-1})$ 로 부터는 조건부가 바뀐 $q(X_{t-1} | X_t)$ 는 Inference과정에서 활용할 수 없음
다만, $q(X_t | X_{t-1})$ 가 gaussian이면, $q(X_{t-1} | X_t)$ 도 gaussian이라는 점은 이미 증명됨 (β_t 가 매우 작을 때)

(Feller, W., 1949)

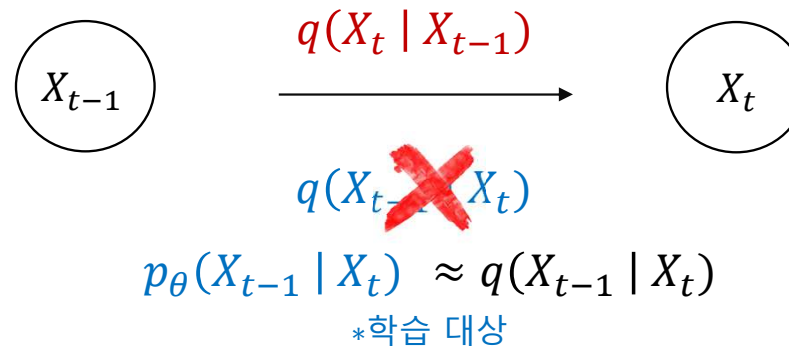
Diffusion Model Overview

➤ Diffusion Model

- **Diffusion model**은 Generative model로서 학습된 데이터의 패턴을 생성해내는 역할을 함
- 패턴 생성 과정을 학습하기 위해 고의적으로 패턴을 무너트리고(Noising), 이를 다시 복원하는 조건부 PDF를 학습(Denoising)

Diffusion process

Reverse process



- ✓ $p_\theta(X_{t-1} | X_t)$ 를 상정해 $q(X_{t-1} | X_t)$ 를 approximation
- ✓ 따라서 Diffusion model은 $p_\theta(X_{t-1} | X_t) \approx q(X_{t-1} | X_t)$ 되도록 학습

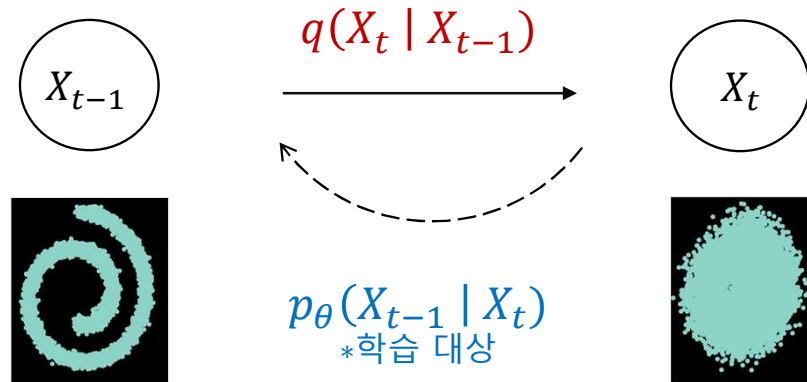
Diffusion Model Overview

➤ Diffusion Model

- **Diffusion model**은 Generative model로서 학습된 데이터의 패턴을 생성해내는 역할을 함
- 패턴 생성 과정을 학습하기 위해 고의적으로 패턴을 무너트리고(Noising), 이를 다시 복원하는 조건부 PDF를 학습(Denoising)

Diffusion process

Reverse process



✓ 하지만 이 변화(Noising, Denoising)를 하나의 단일 step transformation으로 학습하는 것은 매우 어려운 과제

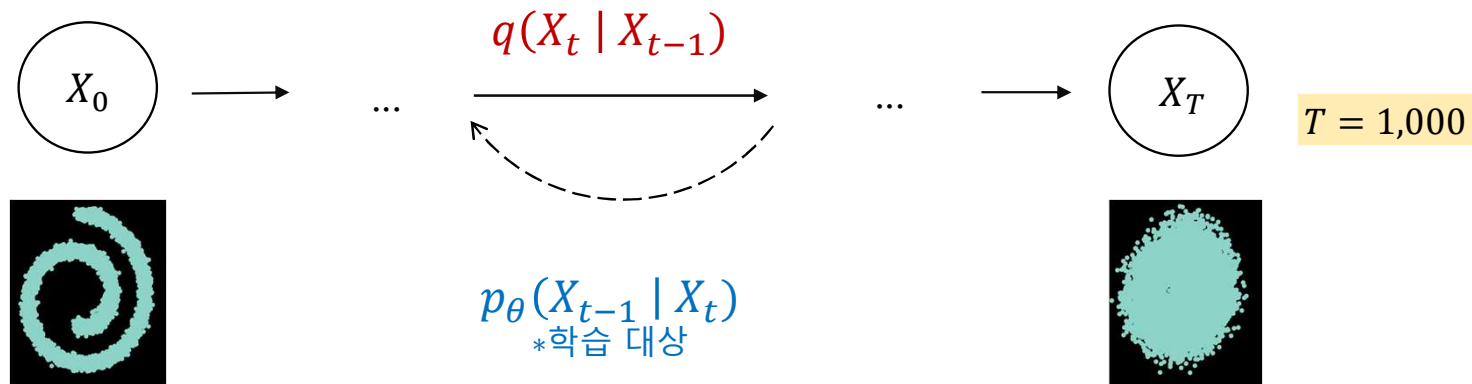
Diffusion Model Overview

➤ Diffusion Model

- **Diffusion model**은 Generative model로서 학습된 데이터의 패턴을 생성해내는 역할을 함
- 패턴 생성 과정을 학습하기 위해 고의적으로 패턴을 무너트리고(Noising), 이를 다시 복원하는 조건부 PDF를 학습(Denoising)

Diffusion process

Reverse process



- 2개의 각 Process(Diffusion, Reverse) 내 변화 과정은 **Markov Chain**으로 매우 많은 단계로 쪼개져 구성됨
 - ✓ $p_\theta(X_{t-1} | X_t)$ 의 학습은 결국 **"large number of small perturbations"**를 추정(estimate)하는 것

Denoising Diffusion Probabilistic Models (DDPM)

Diffusion Model : Diffusion Process

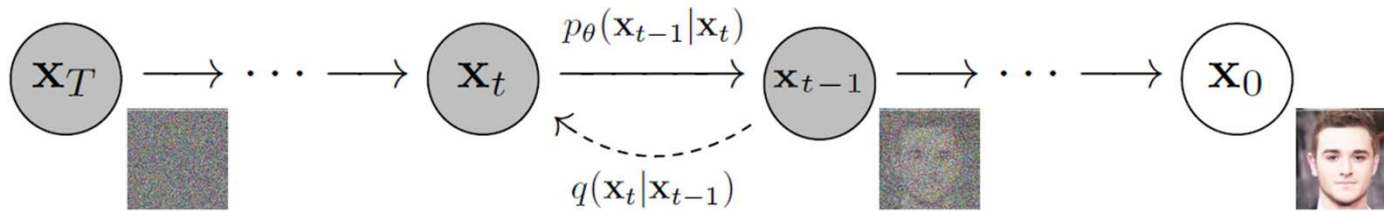


Figure 2: The directed graphical model considered in this work

Diffusion Process

• Reverse Process

학습

Diffusion models [53] are latent variable models of the form $p_\theta(\mathbf{x}_0) := \int p_\theta(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T}$, where $\mathbf{x}_1, \dots, \mathbf{x}_T$ are latents of the same dimensionality as the data $\mathbf{x}_0 \sim q(\mathbf{x}_0)$. The joint distribution $p_\theta(\mathbf{x}_{0:T})$ is called the *reverse process*, and it is defined as a Markov chain with learned Gaussian transitions starting at $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$:

$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)) \quad (1)$$

• Forward Process

설정

What distinguishes diffusion models from other types of latent variable models is that the approximate posterior $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$, called the *forward process* or *diffusion process*, is fixed to a Markov chain that gradually adds Gaussian noise to the data according to a variance schedule $\beta_1, \dots, \beta_T$:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1}), \quad q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad (2)$$

↑
↑

Denoising Diffusion Probabilistic Models (DDPM)

Diffusion Model : Diffusion Process

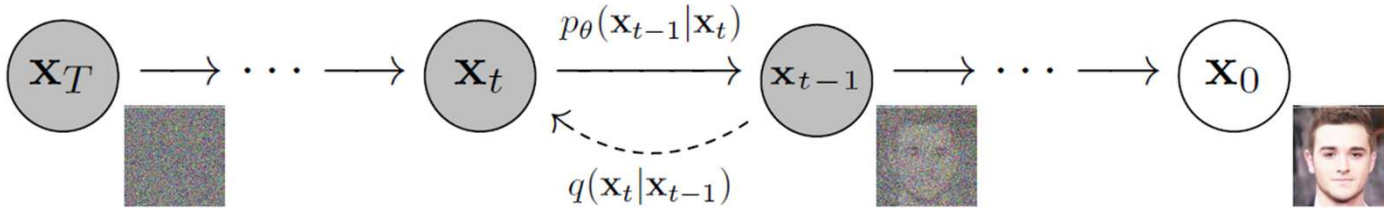


Figure 2: The directed graphical model considered in this work

Diffusion Loss

Training is performed by optimizing the usual variational bound on negative log likelihood:

$$\mathbb{E} [-\log p_\theta(\mathbf{x}_0)] \leq \mathbb{E}_q \left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right] = \mathbb{E}_q \left[-\log p(\mathbf{x}_T) - \sum_{t \geq 1} \log \frac{p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1})} \right] =: L \quad (3)$$

The forward process variances β_t can be learned by reparameterization [33] or held constant as hyperparameters, and expressiveness of the reverse process is ensured in part by the choice of Gaussian conditionals in $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, because both processes have the same functional form when β_t are small [53]. A notable property of the forward process is that it admits sampling $\mathbf{x}_t$ at an arbitrary timestep t in closed form: using the notation $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^t \alpha_s$, we have

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I}) \quad (4)$$

Denoising Diffusion Probabilistic Models (DDPM)

Diffusion Model : Diffusion Process

Diffusion Loss

Efficient training is therefore possible by optimizing random terms of L with stochastic gradient descent. Further improvements come from variance reduction by rewriting L (3) as:

$$\mathbb{E}_q \left[\underbrace{D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1} \underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} - \underbrace{\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \right] \quad (5)$$

(See Appendix A for details. The labels on the terms are used in Section 3.) Equation (5) uses KL divergence to directly compare $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ against forward process posteriors, which are tractable when conditioned on $\mathbf{x}_0$:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\boldsymbol{\beta}}_t \mathbf{I}), \quad (6)$$

$$\text{where } \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t} \mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \mathbf{x}_t \quad \text{and} \quad \tilde{\boldsymbol{\beta}}_t := \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \beta_t \quad (7)$$

Consequently, all KL divergences in Eq. (5) are comparisons between Gaussians, so they can be calculated in a Rao-Blackwellized fashion with closed form expressions instead of high variance Monte Carlo estimates.

$$\mathbb{E}_q \left[\underbrace{D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1} \underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} - \underbrace{\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \right]$$

Learning β_t

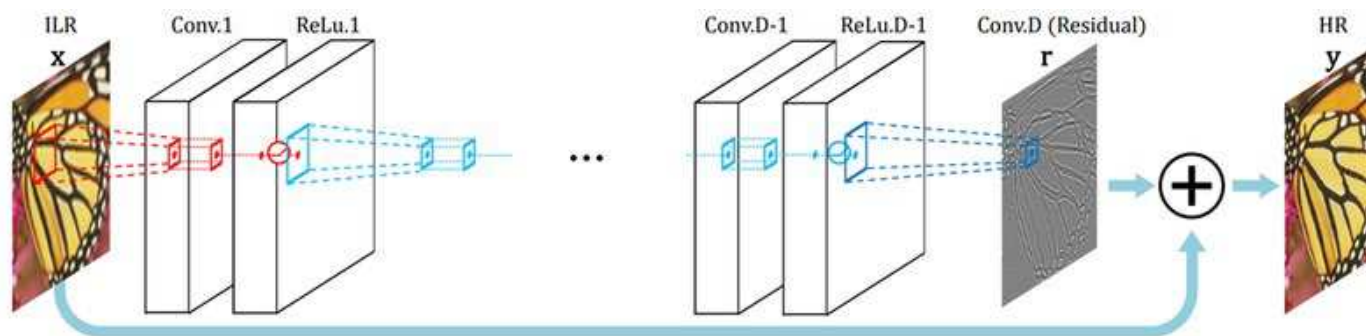
Regualization
Reconstruction

Denoising Diffusion Probabilistic Models (DDPM)

Diffusion Model : Diffusion Process

PR12-409, https://www.youtube.com/watch?v=1j0W_lu55nc

Residual Estimation



Accurate Image Super-Resolution Using Very Deep Convolutional Networks [CVPR 2016]

$$p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \mu_{\theta}(\mathbf{x}_t, t), \Sigma_{\theta}(\mathbf{x}_t, t))$$

$$\mu_{\theta}(\mathbf{x}_t, t) = \tilde{\mu}_t\left(\mathbf{x}_t, \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t}\epsilon_{\theta}(\mathbf{x}_t, t))\right) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}}\epsilon_{\theta}(\mathbf{x}_t, t)\right)$$

예측

Residual ϵ_{θ} 먼저 예측

Denoising Diffusion Probabilistic Models (DDPM)

Diffusion Model : Diffusion Process

PR12-409, https://www.youtube.com/watch?v=1j0W_lu55nc

Loss Simplification

$$\mathbb{E}_q \left[\underbrace{D_{\text{KL}}(q(\mathbf{x}_T | \mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1} \underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t))}_{L_{t-1}} - \underbrace{\log p_\theta(\mathbf{x}_0 | \mathbf{x}_1)}_{L_0} \right]$$

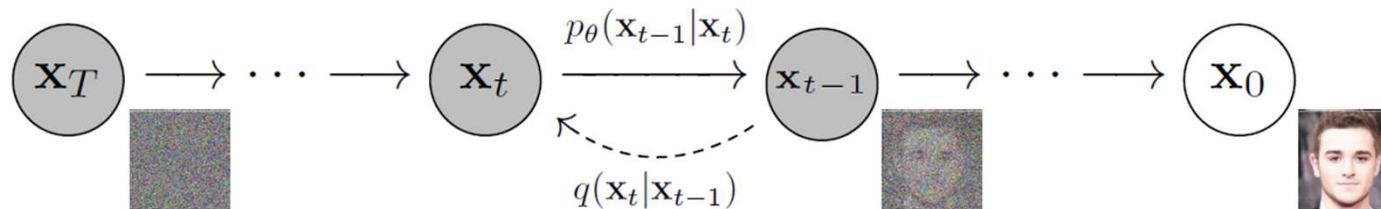
↓

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

Loss Simplification #1 –Deleting Regularization Term

“Not to learn, just to fix β_t ”

Inductive bias를 늘리는 방향



- We chose the β_t schedule from a set of constant, linear, and quadratic schedules, all constrained so that $L_T \approx 0$. We set $T = 1000$ without a sweep, and we chose a linear schedule from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$.

Denoising Diffusion Probabilistic Models (DDPM)

Diffusion Model : Diffusion Process

PR12-409, https://www.youtube.com/watch?v=1j0W_lu55nc

Loss Simplification #2 – Not to learn variance

$$p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_{\theta}(\mathbf{x}_t, t), \boldsymbol{\Sigma}_{\theta}(\mathbf{x}_t, t))$$

$$\boldsymbol{\Sigma}_{\theta}(\mathbf{x}_t, t) = \sigma_t^2 \mathbf{I} \quad \text{Set the variance to untrained time dependent constants}$$

$$\sigma_t^2 = \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \beta_t \quad \text{Or} \quad \sigma_t^2 = \beta_t \quad \text{Has similar results}$$

Loss Simplification

$$\mathbb{E}_q \left[\underbrace{D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1} \underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} - \underbrace{\log p_{\theta}(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \right]$$

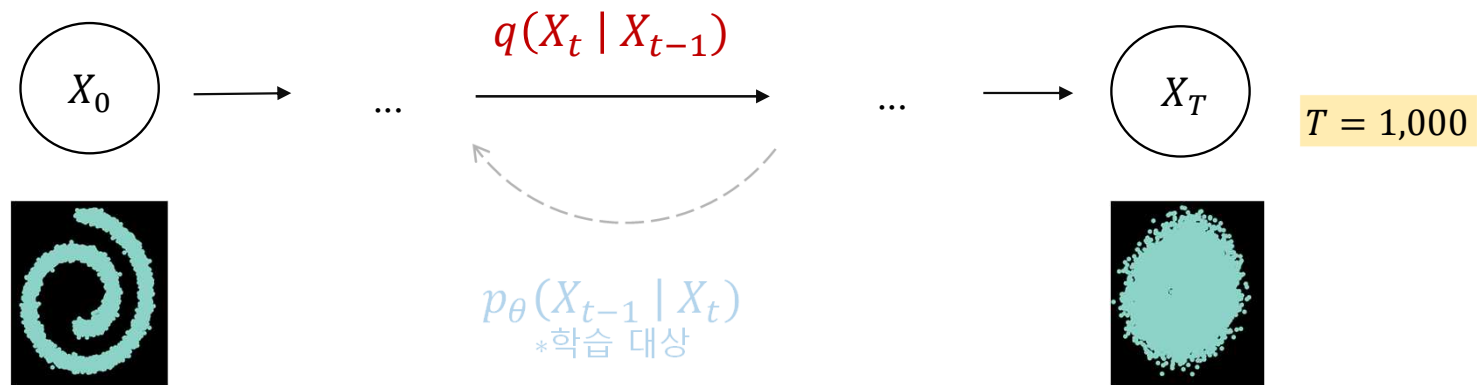
$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \|\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}_{\theta}(\mathbf{x}_t, t)\|^2 \right] + C \quad \leftarrow \quad \boldsymbol{\mu}_{\theta}(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_{\theta}(\mathbf{x}_t, t) \right)$$

$$\mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}} \left[\frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1-\bar{\alpha}_t)} \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\theta}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t} \boldsymbol{\epsilon}, t) \right\|^2 \right]$$

Diffusion Model : Diffusion Process

□ Phase 1 : Diffusion Forward Process

- Diffusion process는 gaussian noise를 점진적으로 주입하는 과정
 - ✓ 조건부 Gaussian $q(X_t | X_{t-1})$ 의 Markov chain으로 구성됨

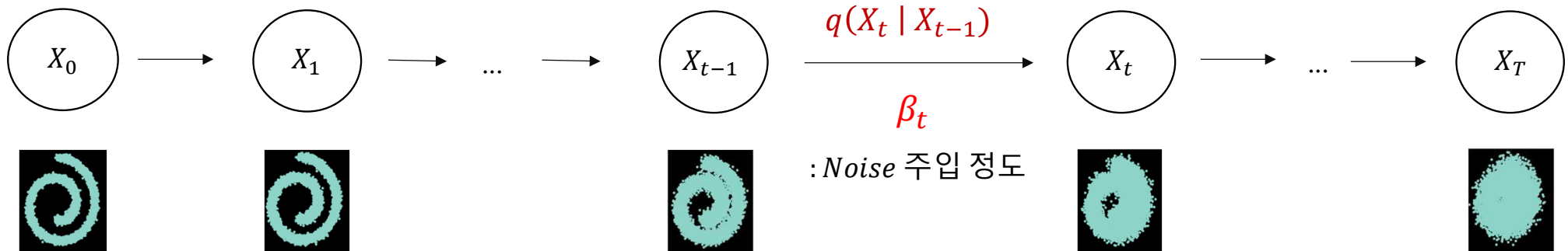


Diffusion Model : Diffusion Process

□ Phase 1 : Diffusion Forward Process

- Diffusion process는 gaussian noise를 점진적으로 주입하는 과정
 - ✓ 주입되는 gaussian noise 크기는 사전적으로 정의되고, 이를 β_t 로 표기

$$q(X_t | X_{t-1}) := N(X_t; \mu_{X_{t-1}}, \Sigma_{X_{t-1}}) := N(X_t; \sqrt{1 - \beta_t} X_{t-1}, \beta_t \cdot I)$$



Diffusion Model : Diffusion Process

□ Phase 1 : Diffusion Forward Process

- 주입되는 gaussian noise 크기는 사전적으로 정의되고, 이를 β_t 로 표기
- β_t 의 사전적 정의(scheduling)는 크게 3가지를 고려함
 - ✓ Linear schedule
 - ✓ Quad schedule
 - ✓ Sigmoid schedule

o Code Example

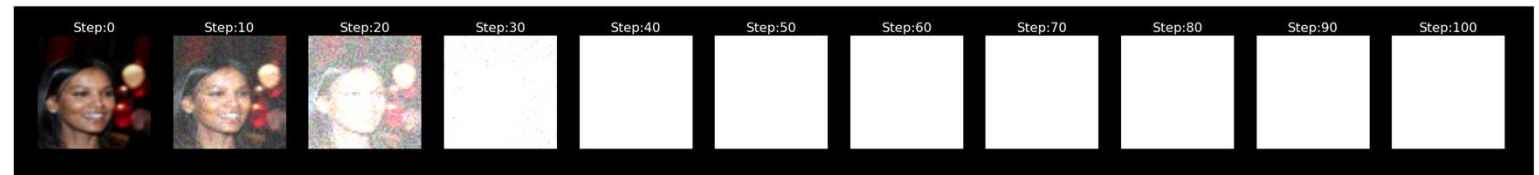
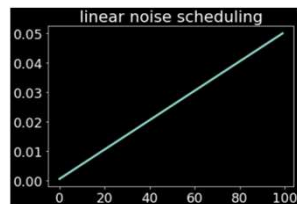
```
def make_beta_schedule(schedule='linear', n_timesteps=1000, start=1e-4, end=0.02):
    if schedule == 'linear':
        betas = torch.linspace(start, end, n_timesteps)
    elif schedule == "quad":
        betas = torch.linspace(start ** 0.5, end ** 0.5, n_timesteps) ** 2
    elif schedule == "sigmoid":
        betas = torch.linspace(-6, 6, n_timesteps)
        betas = torch.sigmoid(betas) * (end - start) + start
    return betas
```

Diffusion Model : Diffusion Process

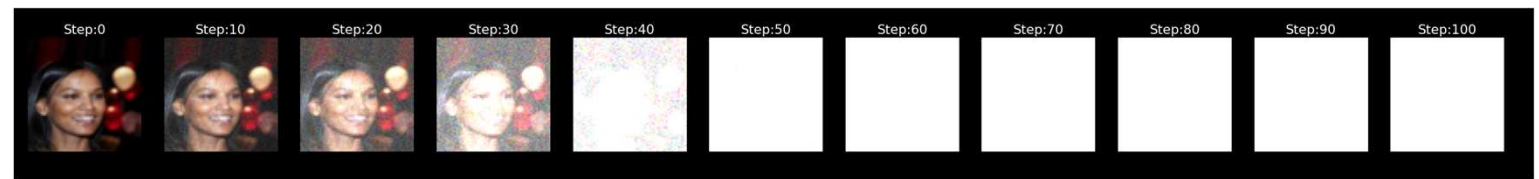
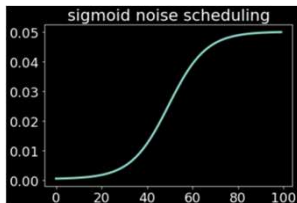
□ Phase 1 : Diffusion Forward Process

- β_t 의 사전적 정의(scheduling)는 크게 3가지를 고려함

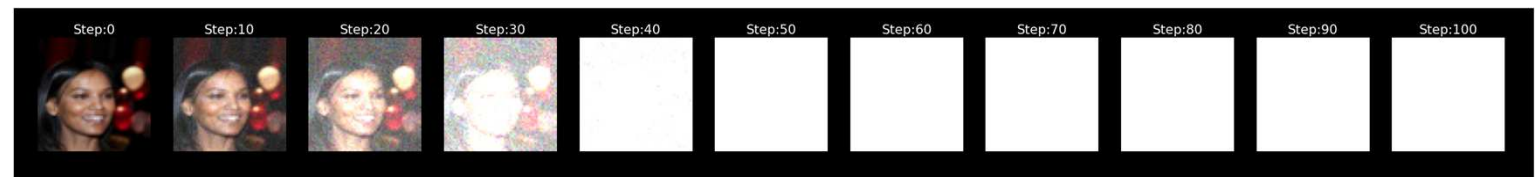
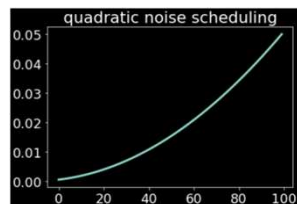
Linear scheduling



Sigmoid scheduling



Quadratic scheduling



Denoising Diffusion Probabilistic Models (DDPM)

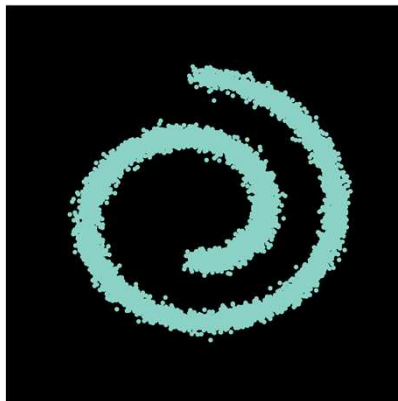
<http://dsba.korea.ac.kr/seminar/?mod=document&uid=2352>

Diffusion Model : Diffusion Process

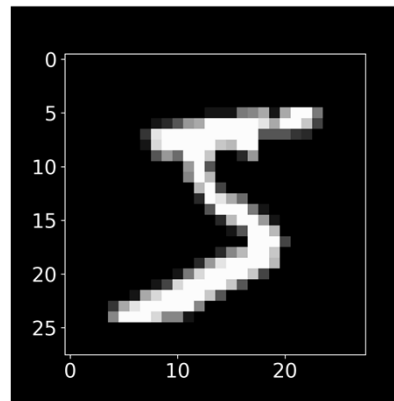
□ Phase 1 : Diffusion Forward Process

- Diffusion Process GIF

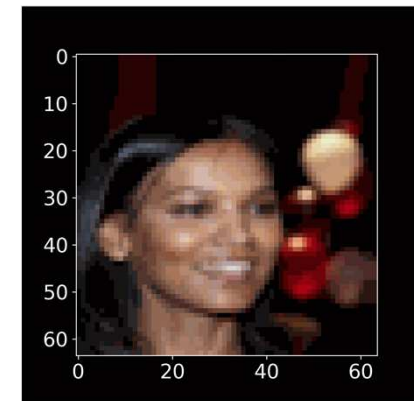
Swiss-roll



MNIST



CELEB A



Diffusion Model : Diffusion Process

□ Phase 1 : Diffusion Forward Process

- Diffusion process는 gaussian noise를 점진적으로 주입하는 과정 (**= Conditional Gaussian distribution**)

$$q(X_t | X_{t-1}) := N(X_t; \mu_{X_{t-1}}, \Sigma_{X_{t-1}}) := N(X_t; \sqrt{1 - \beta_t} X_{t-1}, \beta_t \cdot I)$$

$$\checkmark = \sqrt{1 - \beta_t} X_{t-1} + \sqrt{\beta_t} \epsilon_{t-1} \quad * \epsilon \sim N(0, I) \quad \text{(Reparameterization trick)}$$

o Code Example

```
def forward_process(x_start, n_steps, noise=None):
    """
    Diffuse the data (t == 0 means diffused for 1 step)

    x_start:tensor = dataset at start step(X_0)
    n_steps:int = the number of convert iterations in diffusion process
    noise:scheduled noise set
    """
    x_sequence = [x_start] # initial 'x_seq' which is filled with originl data at first.
    for n in range(n_steps):

        beta_t = noise[n]
        x_t_1 = x_sequence[-1]
        epsilon_t_1 = torch.rand_like(x_t_1)

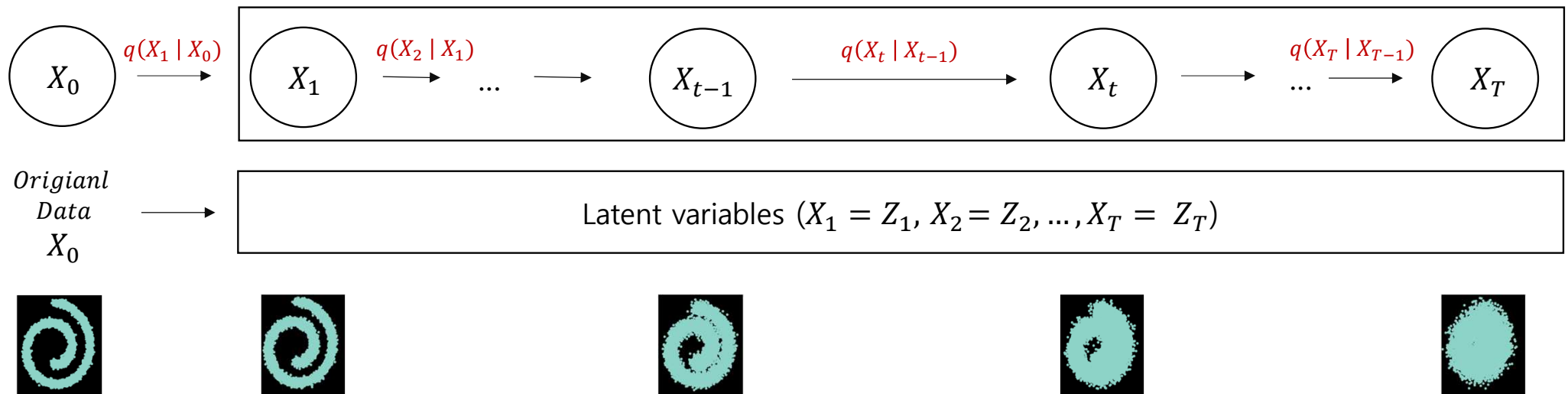
        x_t = (torch.sqrt(1-beta_t) * x_t_1) + (torch.sqrt(beta_t) * epsilon_t_1)
        x_sequence.append(x_t)

    return x_seq
```

Diffusion Model : Diffusion Process

Phase 1 : Diffusion Forward Process

- Diffusion process는 점진적으로 단계적 gaussian noise를 갖는 다수의 latent variable($X_1, X_2, \dots, X_T$)를 획득
- 다수의 latent variable($X_1, X_2, \dots, X_T$)를 상정한다는 점에서 Hierarchical VAE와 유사한 접근

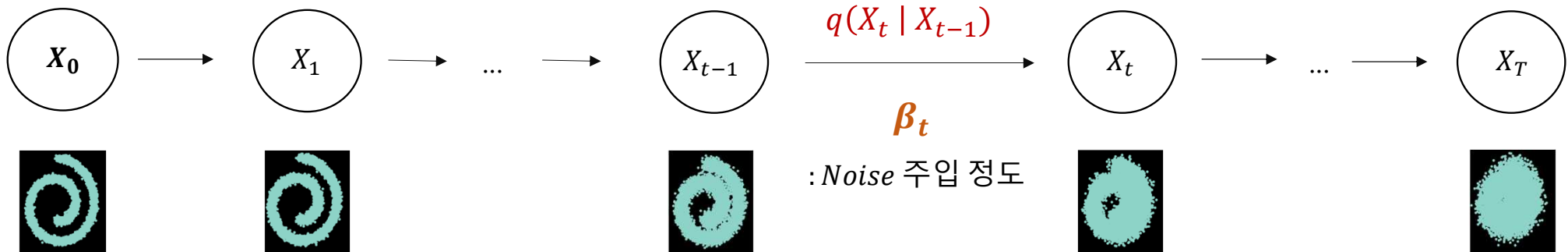


Diffusion Model : Diffusion Process

Phase 1 : Diffusion Forward Process

- Diffusion process는 Conditional Gaussian의 joint-distribution으로서, X_0 를 조건부로 latent variables($X_{1:T}$)를 생성해내는 과정
- ✓ 가장 마지막 latent variable($X_T = Z_T$)로 pure isotropic gaussian을 획득

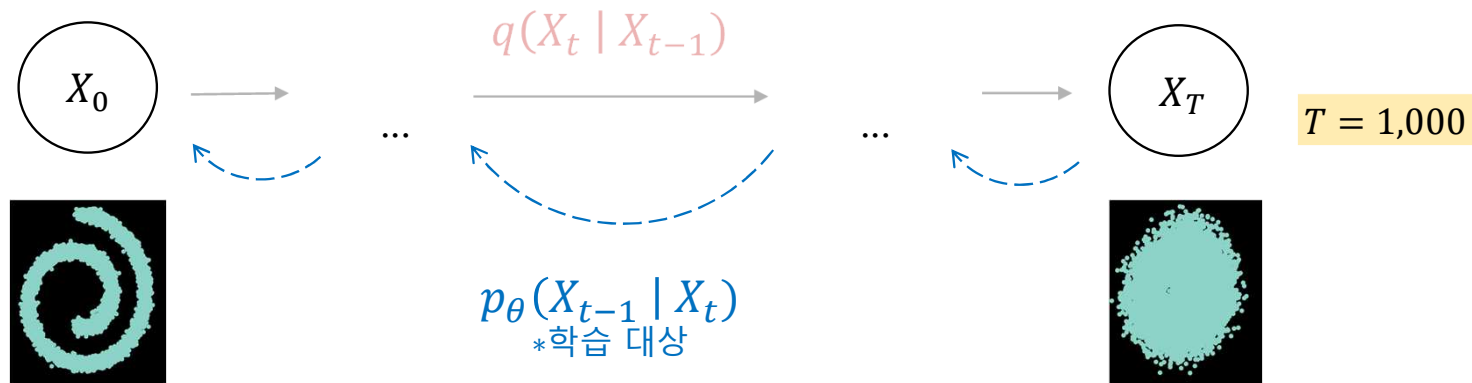
$$q(X_{1:T} | X_0) := \prod_{t=1}^T q(X_t | X_{t-1}), \quad q(X_t | X_{t-1}) := N(X_t; \mu_{X_{t-1}}, \Sigma_{X_{t-1}}) := N(X_t; \sqrt{1 - \beta_t} X_{t-1}, \beta_t \cdot I)$$



Diffusion Model : Diffusion Process

□ Phase 2 : Diffusion Reverse Process

- **Diffusion model**은 Generative model로서 **학습된 데이터의 패턴을 생성해내는 역할**을 함
- 패턴 생성 과정을 학습하기 위해 고의적으로 패턴을 무너트리고(Noising), 이를 다시 복원하는 조건부 pdf를 학습함(Denoising)
Diffusion process *Reverse process*
- Reverse process는 Diffusion process의 역 과정(Denoising)을 학습



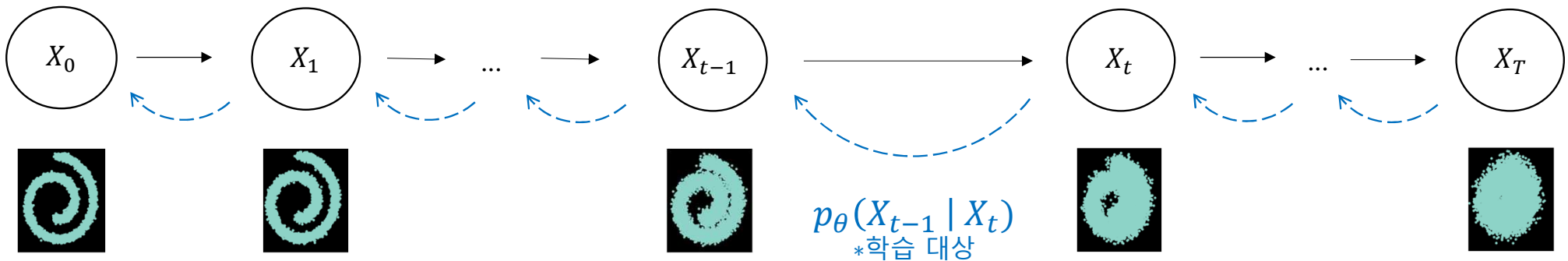
Diffusion Model : Diffusion Process

Phase 2 : Diffusion Reverse Process

- Reverse process는 Diffusion process의 역(reverse) 과정으로, **gaussian noise**를 제거해가며 특정한 패턴을 만들어가는 과정

$$p_{\theta}(X_{0:T}) := p(X_T) \prod_{t=1}^T p_{\theta}(X_{t-1} | X_t), \quad p_{\theta}(X_{t-1} | X_t) := N(X_{t-1}; \underbrace{\mu_{\theta}(X_t, t)}_{\text{학습 대상}}, \underbrace{\Sigma_{\theta}(X_t, t)}_{\text{학습 대상}})$$

학습 대상
(mean & variance function)



Diffusion Model : Diffusion Process

□ Phase 2 : Diffusion Reverse Process

- Reverse process는 Diffusion process의 역(reverse) 과정으로, **gaussian noise**를 제거해가며 특정한 패턴을 만들어가는 과정

$$p_{\theta}(X_{0:T}) := p(X_T) \prod_{t=1}^T p_{\theta}(X_{t-1} | X_t), \quad p_{\theta}(X_{t-1} | X_t) := N(X_{t-1}; \underbrace{\mu_{\theta}(X_t, t)}_{\text{학습 대상}}, \underbrace{\Sigma_{\theta}(X_t, t)}_{\text{학습 대상}})$$

학습 대상
(mean & variance function)

o Code example

```
def p_mean_variance(model, x, t):  
    # Make model prediction  
    ✓ out = model(x, t.to(device))  
  
    # Extract the mean and variance  
    mean, log_var = torch.split(out, 2, dim=-1)  
    var = torch.exp(log_var)  
    return mean, log_var
```

Denoising Diffusion Probabilistic Models (DDPM)

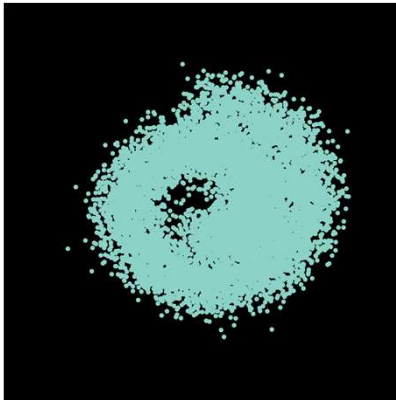
<http://dsba.korea.ac.kr/seminar/?mod=document&uid=2352>

Diffusion Model : Diffusion Process

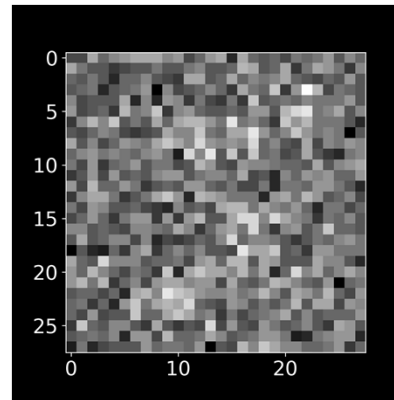
□ Phase 2 : Diffusion Reverse Process

- Reverse process gif

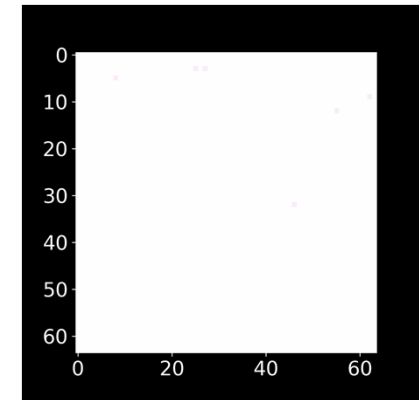
Swiss-roll



MNIST



CELEB A



Diffusion Model : Diffusion Process

□ Phase 2 : Diffusion Reverse Process Loss

$$\begin{aligned} Loss_{diffusion} &:= D_{KL}(q(z | x_0) || P_{\theta}(z)) + \sum_{t=2} D_{KL}(q(x_{t-1} | x_t, x_0) || P_{\theta}(x_{t-1} | x_t)) - E_q[\log P_{\theta}(x_0 | x_1)] \\ &:= \text{Negative log likelihood} (= \mathbb{E}_{x_T \sim q(x_T | x_0)} [-\log p_{\theta}(x_0)]) \end{aligned}$$

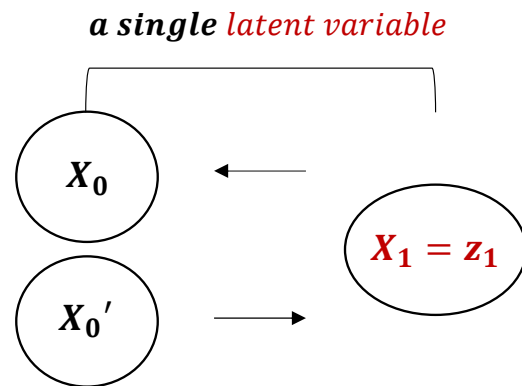
- Diffusion의 Loss는 VAE와 비교 이해 가능
- 자세한 도출 과정 참고 Blog ["\[논문공부\] Denoising Diffusion Probabilistic Models \(DDPM\) 설명 \(Blue collar Developer\)"](#)

Diffusion Model : Diffusion Process

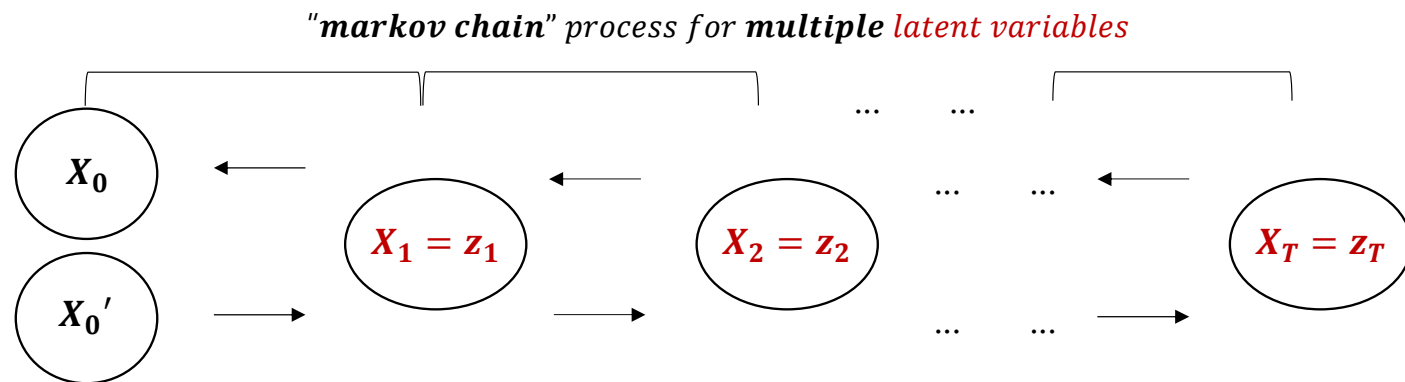
□ Phase 2 : Diffusion Reverse Process Loss

- VAE와 Diffusion의 구조 비교

✓ VAE



✓ Diffusion

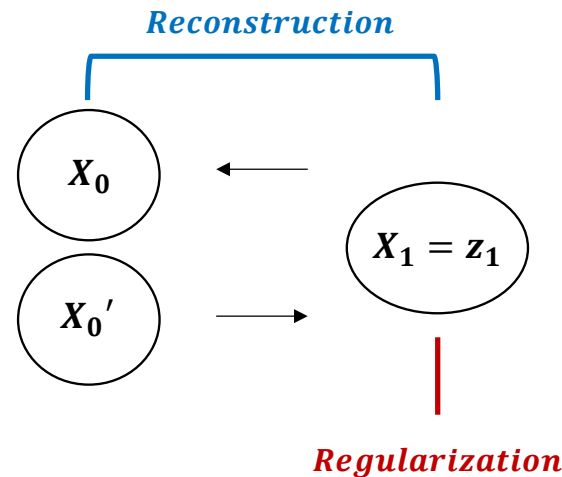


Diffusion Model : Diffusion Process

Phase 2 : Diffusion Reverse Process Loss

- VAE와 Diffusion의 구조 비교

✓ VAE



Regularizer on Encoder

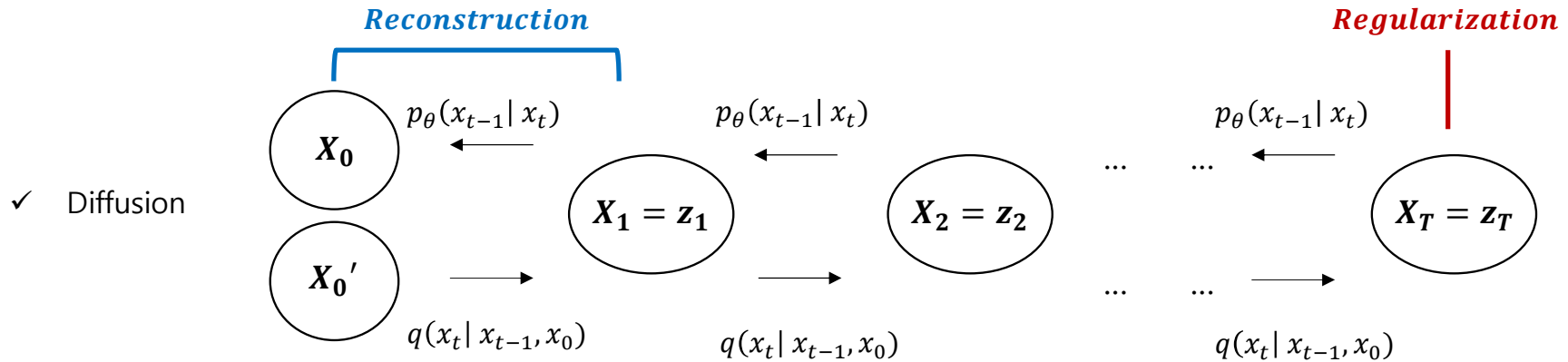
Reconstruction on Decoder

$$Loss_{VAE} = D_{KL}(q(z | x) || p_{\theta}(z)) - E_{z \sim q(z|x)} [\log P_{\theta}(x | z)]$$

Diffusion Model : Diffusion Process

Phase 2 : Diffusion Reverse Process Loss

- VAE와 Diffusion의 구조 비교

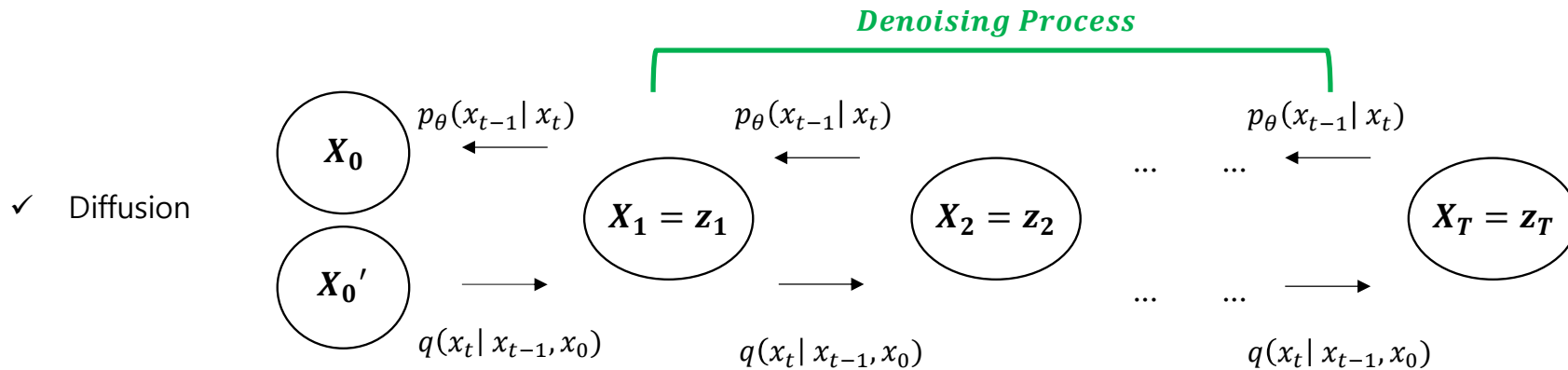


$$\begin{aligned}
 LOSS_{Diffusion} &= D_{KL}(q(z | x_0) \| P_\theta(x_0 | z)) - E_{z \sim q(z|x)}[\log P_\theta(z)] \\
 &= \underbrace{D_{KL}(q(z | x_0) \| P_\theta(z))}_{\text{Regularization}} + \sum_{t=2} D_{KL}(q(x_{t-1} | x_t, x_0) \| P_\theta(x_{t-1} | x_t)) - \underbrace{E_q[\log P_\theta(x_0 | x_1)]}_{\text{Reconstruction}}
 \end{aligned}$$

Diffusion Model : Diffusion Process

Phase 2 : Diffusion Reverse Process Loss

- VAE와 Diffusion의 구조 비교



$$\begin{aligned}
 \text{Loss}_{\text{Diffusion}} &= D_{KL}(q(z | x_0) \| P_\theta(x_0 | z)) - E_{z \sim q(z|x)} [\log P_\theta(z)] \\
 &= D_{KL}(q(z | x_0) \| P_\theta(z)) + \underbrace{\sum_{t=2} D_{KL}(q(x_{t-1} | x_t, x_0) \| P_\theta(x_{t-1} | x_t))}_{\text{Denoising Process}} - E_q[\log P_\theta(x_0 | x_1)]
 \end{aligned}$$

Diffusion Model : Diffusion Process

□ Phase 2 : Diffusion Reverse Process Loss

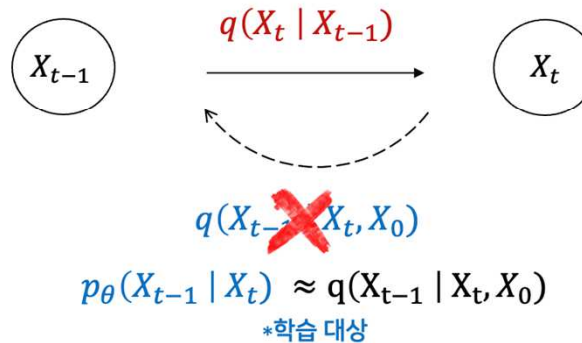
- Diffusion model의 denoising process 학습 : $p_{\theta}(X_{t-1} | X_t) \approx q(X_{t-1} | X_t, X_0)$

□ Diffusion model

- Diffusion model은 Generative model로서 학습된 데이터의 패턴을 생성해내는 역할을 함
- 패턴 생성 과정을 학습하기 위해 고의적으로 패턴을 무너트리고(Noising), 이를 다시 복원하는 조건부 PDF를 학습(Denoising)

Diffusion process

Reverse process

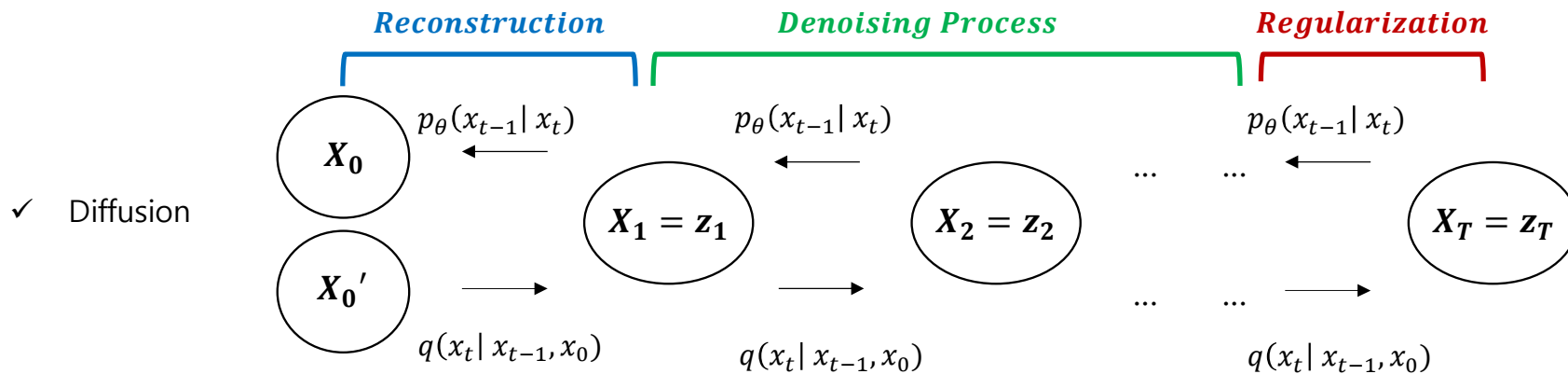


- ✓ $p_{\theta}(X_{t-1} | X_t)$ 를 상정해 $q(X_{t-1} | X_t, X_0)$ 를 approximation
- ✓ 따라서 Diffusion model은 $p_{\theta}(X_{t-1} | X_t) \approx q(X_{t-1} | X_t, X_0)$ 되도록 학습

Diffusion Model : Diffusion Process

Phase 2 : Diffusion Reverse Process Loss

- VAE와 Diffusion의 구조 비교



$$\begin{aligned}
 LOSS_{Diffusion} &= D_{KL}(q(z | x_0) || P_\theta(x_0 | z)) - E_{z \sim q(z|x)}[\log P_\theta(z)] \\
 &= \underbrace{D_{KL}(q(z | x_0) || P_\theta(z))}_{\text{Regularization}} + \sum_{t=2} \underbrace{D_{KL}(q(x_{t-1} | x_t, x_0) || P_\theta(x_{t-1} | x_t))}_{\text{Denoising Process}} - \underbrace{E_q[\log P_\theta(x_0 | x_1)]}_{\text{Reconstruction}}
 \end{aligned}$$

Diffusion Model : Diffusion Process

- Phase 2 : Diffusion Reverse Process Loss

Regularizer on Encoder

Reconstruction on Decoder

$$LOSS_{VAE} = D_{KL}(q(z | x) || p_{\theta}(z)) - E_{z \sim q(z|x)} [\log P_{\theta}(x | z)]$$

$$LOSS_{Diffusion} = D_{KL}(q(z | x_0) || P_{\theta}(x_0 | z)) - E_{z \sim q(z|x)} [\log P_{\theta}(z)]$$

$$= D_{KL}(q(z | x_0) || P_{\theta}(z)) + \sum_{t=2} D_{KL}(q(x_{t-1} | x_t, x_0) || P_{\theta}(x_{t-1} | x_t)) - E_q [\log P_{\theta}(x_0 | x_1)]$$

Regularization

Denoising Process

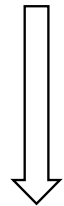
Reconstruction

DDPM

- Choose the **variances β_t of the forward process** and the **model architecture** and **Gaussian distribution parameterization of the reverse process**.
- Establish a **new explicit connection between diffusion models and denoising score matching** (Section 3.2) that leads to a simplified, **weighted variational bound objective** for diffusion models (Section 3.4).
- Ultimately, our model design is justified by **simplicity** and **empirical** results (Section 4).

$$\begin{aligned}
 LOSS_{Diffusion} &= D_{KL}(q(z | x_0) || P_{\theta}(x_0 | z)) - E_{z \sim q(z|x)} [\log P_{\theta}(z)] \\
 &= D_{KL}(q(z | x_0) || P_{\theta}(z)) + \sum_{t=2} D_{KL}(q(x_{t-1} | x_t, x_0) || P_{\theta}(x_{t-1} | x_t)) - E_q[\log P_{\theta}(x_0 | x_1)]
 \end{aligned}$$

Regularizer on Encoder
Denoising Process
Reconstruction on Decoder



DDPM에서는 Loss가 굉장히 간단한 식으로 정의됨

$$LOSS_{DDPM} = \mathbb{E}_{x_0, \epsilon} \left[\left| \epsilon - \epsilon_{\theta} \left(\sqrt{\tilde{\alpha}_t} + \sqrt{1 - \tilde{\alpha}_t} \epsilon, t \right) \right|^2 \right]$$

DDPM

□ Forward Process and L_T

We ignore the fact that the forward process variances β_t are learnable by reparameterization and instead fix them to constants (see Section 4 for details). Thus, in our implementation, the approximate posterior q has no learnable parameters, so L_T is a constant during training and can be ignored.

1. 학습 목적식에서 **Regularization term** 제외

- ✓ 굳이 학습시키지 않아도 fixed noise scheduling으로 필요한 'isotropic gaussian' 획득 가능하기 때문

$$Loss_{Diffusion} = \underbrace{D_{KL}(q(\mathbf{x}_T | \mathbf{x}_0) \| P_\theta(z))}_{\text{Regularization}} + \sum_{t=2} \underbrace{D_{KL}(q(x_{t-1} | x_t, x_0) \| P_\theta(x_{t-1} | x_t))}_{\text{Denoising Process}} - \underbrace{E_q[\log P_\theta(x_0 | x_1)]}_{\text{Reconstruction}}$$

$$\mathbb{E}_q \left[\underbrace{D_{KL}(q(\mathbf{x}_T | \mathbf{x}_0) \| p(\mathbf{x}_T))}_{\text{Regularization}} + \sum_{t>1} \underbrace{D_{KL}(q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) \| p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t))}_{L_{t-1}} - \underbrace{\log p_\theta(\mathbf{x}_0 | \mathbf{x}_1)}_{L_0} \right]$$

Learning β_t

DDPM

□ Forward Process and L_T

We ignore the fact that the forward process variances β_t are learnable by reparameterization and instead fix them to constants (see Section 4 for details). Thus, in our implementation, the approximate posterior q has no learnable parameters, so L_T is a constant during training and can be ignored.

❖ $\{\beta_1 = 10^{-4}, \dots, \beta_T = 0.02\}$ 로 Noise를 Linear하게 증가시키는 "Linear noise scheduling"을 설정할 경우

최종적인 latent variable ($Z_T = X_T$) 은 아래와 같은 분포를 갖게 됨

$$q(X_T | X_0) = N(X_T ; 0.00635 \cdot x_0, 0.99995 \cdot I)$$

$$q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I})$$

$$\alpha_t := 1 - \beta_t \quad \bar{\alpha}_t := \prod_{s=1}^t \bar{\alpha}_s$$

✓ 충분한 isotropic gaussian을 확보할 수 있다고 주장함

$$\mathbb{E}_q \left[\underbrace{D_{\text{KL}}(q(\mathbf{x}_T | \mathbf{x}_0) \| p(\mathbf{x}_T))}_{\text{Regualization}} + \sum_{t>1} \underbrace{D_{\text{KL}}(q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) \| p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t))}_{L_{t-1}} - \underbrace{\log p_{\theta}(\mathbf{x}_0 | \mathbf{x}_1)}_{L_0} \right]$$

✗
Learning β_t

Regualization
Reconstruction

DDPM

Reverse Process and $L_{1:T-1}$

Now we discuss our choices in $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$ for $1 < t \leq T$. First, we set $\Sigma_\theta(\mathbf{x}_t, t) = \sigma_t^2 \mathbf{I}$ to untrained time dependent constants. Experimentally, both $\sigma_t^2 = \beta_t$ and $\sigma_t^2 = \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \beta_t$ had similar results. The first choice is optimal for $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and the second is optimal for $\mathbf{x}_0$ deterministically set to one point. These are the two extreme choices corresponding to upper and lower bounds on reverse process entropy for data with coordinatewise unit variance [53].

- ② Second, to represent the mean $\mu_\theta(\mathbf{x}_t, t)$, we propose a specific parameterization motivated by the following analysis of L_t . With $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2 \mathbf{I})$, we can write:

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)\|^2 \right] + C \quad (8)$$

where C is a constant that does not depend on θ . So, we see that the most straightforward parameterization of μ_θ is a model that predicts $\tilde{\mu}_t$, the forward process posterior mean. However, we can expand Eq. (8) further by reparameterizing Eq. (4) as $\mathbf{x}_t(\mathbf{x}_0, \epsilon) = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$ for $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and applying the forward process posterior formula (7):

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

where $\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}} \beta_t}{1 - \bar{\alpha}_t} \mathbf{x}_0 + \frac{\sqrt{\bar{\alpha}_t} (1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} \mathbf{x}_t$ and $\tilde{\beta}_t := \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \beta_t$

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I})$$

DDPM

Reverse Process

$$L_{t-1} - C = \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{1}{2\sigma_t^2} \left\| \tilde{\mu}_t \left(\mathbf{x}_t(\mathbf{x}_0, \epsilon), \frac{1}{\sqrt{\bar{\alpha}_t}} (\mathbf{x}_t(\mathbf{x}_0, \epsilon) - \sqrt{1 - \bar{\alpha}_t} \epsilon) \right) - \mu_\theta(\mathbf{x}_t(\mathbf{x}_0, \epsilon), t) \right\|^2 \right] \quad (9)$$

$$= \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{1}{2\sigma_t^2} \left\| \frac{1}{\sqrt{\bar{\alpha}_t}} \left(\mathbf{x}_t(\mathbf{x}_0, \epsilon) - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon \right) - \mu_\theta(\mathbf{x}_t(\mathbf{x}_0, \epsilon), t) \right\|^2 \right] \quad (10)$$

↓ Eq. (9)

Equation (10) reveals that μ_θ must predict $\frac{1}{\sqrt{\bar{\alpha}_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon \right)$ given $\mathbf{x}_t$. Since $\mathbf{x}_t$ is available as input to the model, we may choose the parameterization

$$\mu_\theta(\mathbf{x}_t, t) = \tilde{\mu}_t \left(\mathbf{x}_t, \frac{1}{\sqrt{\bar{\alpha}_t}} (\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \epsilon_\theta(\mathbf{x}_t, t)) \right) = \frac{1}{\sqrt{\bar{\alpha}_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) \quad (11)$$

↓

where ϵ_θ is a function approximator intended to predict ϵ from $\mathbf{x}_t$. To sample $\mathbf{x}_{t-1} \sim p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)$ is to compute $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\bar{\alpha}_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$, where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The complete sampling procedure, Algorithm 2, resembles Langevin dynamics with ϵ_θ as a learned gradient of the data density. Furthermore, with the parameterization (11), Eq. (10) simplifies to:

$$\mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1 - \bar{\alpha}_t)} \left\| \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t) \right\|^2 \right] \quad (12)$$

which resembles denoising score matching over multiple noise scales indexed by t [55]. As Eq. (12) is equal to (one term of) the variational bound for the Langevin-like reverse process (11), we see that optimizing an objective resembling denoising score matching is equivalent to using variational inference to fit the finite-time marginal of a sampling chain resembling Langevin dynamics.

DDPM

Reverse Process and $L_{1:T-1}$

To summarize, we can train the reverse process mean function approximator μ_θ to predict $\tilde{\mu}_t$, or by modifying its parameterization, we can train it to predict ϵ . (There is also the possibility of predicting $\mathbf{x}_0$, but we found this to lead to worse sample quality early in our experiments.) We have shown that the ϵ -prediction parameterization both resembles Langevin dynamics and simplifies the diffusion model's variational bound to an objective that resembles denoising score matching. Nonetheless, it is just another parameterization of $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, so we verify its effectiveness in Section 4 in an ablation where we compare predicting ϵ against predicting $\tilde{\mu}_t$.

Algorithm 1 Training

```

1: repeat
2:    $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ 
3:    $t \sim \text{Uniform}(\{1, \dots, T\})$ 
4:    $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
5:   Take gradient descent step on
        $\nabla_\theta \|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon, t)\|^2$ 
6: until converged
    
```

Algorithm 2 Sampling

```

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

DDPM

Reverse Process and $L_{1:T-1}$

2. *Denoising Process* 의 목적식 재구성

$$LOSS_{Diffusion} = \underbrace{D_{KL}(q(z|x_0) || P_\theta(z))}_{\text{Regularization}} + \underbrace{\sum_{t=2}^T D_{KL}(q(x_{t-1} | x_t, x_0) || P_\theta(x_{t-1} | x_t))}_{\text{Denoising Process}} - E_q[\log P_\theta(x_0 | x_1)]$$

$$\mathbb{E}_q \left[\underbrace{D_{KL}(q(\mathbf{x}_T | \mathbf{x}_0) || p(\mathbf{x}_T))}_{\text{Regularization}} + \sum_{t>1} \underbrace{D_{KL}(q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) || p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t))}_{L_{t-1}} - \underbrace{\log p_\theta(\mathbf{x}_0 | \mathbf{x}_1)}_{L_0} \right]$$

DDPM

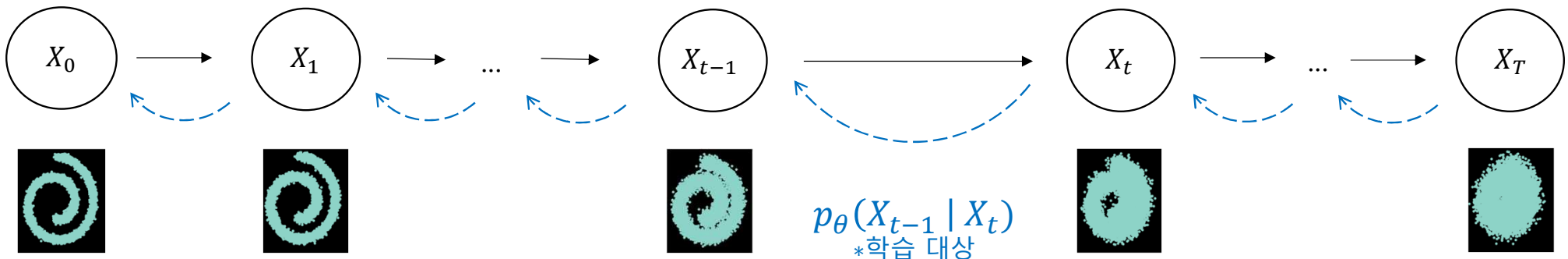
Reverse Process and $L_{1:T-1}$

2. *Denoising Process* 의 목적식 재구성

- Reverse process는 Diffusion process의 역(reverse) 과정으로, **gaussian noise**를 제거(**Denoising**)해가며 특정한 패턴을 만들어가는 과정

$$p_{\theta}(X_{0:T}) := p(X_T) \prod_{t=1}^T p_{\theta}(X_{t-1} | X_t), \quad p_{\theta}(X_{t-1} | X_t) := N(X_{t-1}; \underbrace{\mu_{\theta}(X_t, t)}_{\text{학습 대상}}, \underbrace{\Sigma_{\theta}(X_t, t)}_{\text{학습 대상}})$$

학습 대상
(mean & variance function)



DDPM

□ Reverse Process and $L_{1:T-1}$

2. *Denoising Process* 의 목적식 재구성

1) $\Sigma_{\theta}(X_t, t)$ 의 상수화 :

✓ 학습 대상(*mean & variance function*)

$$(1) \mu_{\theta}(X_t, t) \rightarrow \mu_{\theta}(X_t, t)$$

$$(2) \Sigma_{\theta}(X_t, t) \rightarrow \text{Constant (time dependent)} = \sigma_t^2 \cdot I = \tilde{\beta}_t \cdot I = t\text{시점까지의 누적된 noise}$$

$$\bullet \sigma_t^2 = \tilde{\beta}_t = \frac{1 - \overline{\alpha}_{t-1}}{1 - \overline{\alpha}_t} \beta_t \quad \left(\overline{\alpha}_t := \prod_{s=1}^t \alpha_s, \alpha_t = 1 - \beta_t \right)$$

DDPM

□ Reverse Process and $L_{1:T-1}$

2. *Denoising Process* 의 목적식 재구성

1) $\Sigma_\theta(X_t, t)$ 의 상수화 :

- DDPM 이전

$$p_\theta(X_{0:T}) := p(X_T) \prod_{t=1}^T p_\theta(X_{t-1} | X_t), \quad p_\theta(X_{t-1} | X_t) := N(X_{t-1}; \underbrace{\mu_\theta(X_t, t)}_{\text{학습 대상}}, \underbrace{\Sigma_\theta(X_t, t)}_{\text{학습 대상}})$$

학습 대상 (*mean & variance function*)

- DDPM 이후

$$p_\theta(X_{0:T}) := p(X_T) \prod_{t=1}^T p_\theta(X_{t-1} | X_t), \quad p_\theta(X_{t-1} | X_t) := N(X_{t-1}; \underbrace{\mu_\theta(X_t, t)}_{\text{학습 대상}}, \sigma_t^2 \cdot I)$$

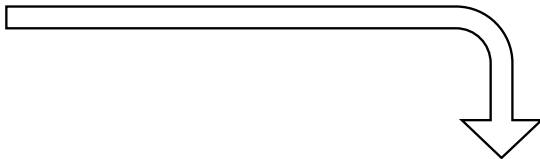
학습 대상 (*mean function*)

DDPM

Reverse Process and $L_{1:T-1}$

2. *Denoising Process* 의 목적식 재구성

2) $\mu_\theta(x_t, t)$ 을 새롭게 정의 : Denoising matching

$$\sum_{t=2} D_{KL}(q(x_{t-1} | x_t, x_0) \| P_\theta(x_{t-1} | x_t))$$


$$\begin{aligned} q(x_{t-1} | x_t, x_0) &= N(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t \cdot I) \\ p_\theta(x_{t-1} | x_t) &= N(x_{t-1}; \mu_\theta(x_t, t), \tilde{\beta}_t \cdot I) \end{aligned}$$

$$\rightarrow \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)\|^2 \right]$$

Denoising Process

DDPM

□ Reverse Process and $L_{1:T-1}$

2. *Denoising Process* 의 목적식 재구성

2) $\mu_\theta(x_t, t)$ 을 새롭게 정의 : *Denoising matching*

$$\mathbf{x}_t(\mathbf{x}_0, \epsilon) = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon \text{ for } \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad (\bar{\alpha}_t := \prod_{s=1}^t \alpha_s, \alpha_t = 1 - \beta_t)$$

$$\begin{aligned} \because q(x_t | x_{t-1}) &:= N(x_{t-1}; \sqrt{1 - \beta_t} x_{t-1}, \beta_t \cdot I), \\ x_t &= \sqrt{1 - \beta_t} x_{t-1} + \sqrt{\beta_t} \epsilon_{t-1} \end{aligned} \quad * \epsilon \sim N(0, I)$$

위에서 정의한 x_t 를 아래의 식 (1)에 대입하면, 식 (2)와 같이 정리된다.

$$(1) \quad \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)\|^2 \right]$$

$$(2) \quad \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{1}{2\sigma_t^2} \left\| \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t(\mathbf{x}_0, \epsilon) - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon \right) - \mu_\theta(\mathbf{x}_t(\mathbf{x}_0, \epsilon), t) \right\|^2 \right]$$

DDPM

□ Reverse Process and $L_{1:T-1}$

2. *Denoising Process* 의 목적식 재구성

2) $\mu_\theta(x_t, t)$ 을 새롭게 정의 : *Denoising matching*

$$(2) \quad \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{1}{2\sigma_t^2} \left\| \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t(\mathbf{x}_0, \epsilon) - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon \right) - \mu_\theta(\mathbf{x}_t(\mathbf{x}_0, \epsilon), t) \right\|^2 \right]$$

식 (2)에 따르면, 학습할 **mean function** $\mu_\theta(x_t, t)$ 는 주어진 t 시점에 $\frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon \right)$ 를 예측해내야 한다.

x_t (input)와 t 는 주어진므로, $\mu_\theta(x_t, t)$ 에게 남은 예측 대상은 t 시점의 **noise(ϵ)** 뿐이다.

따라서 아래 식 (3)과 같이 학습 모델 $\mu_\theta(x_t, t)$ 을 새롭게 정의한다.

$$(3) \quad \mu_\theta(\mathbf{x}_t, t) = \tilde{\mu}_t \left(\mathbf{x}_t, \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \epsilon_\theta(\mathbf{x}_t) \right) \right) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right)$$

DDPM

□ Reverse Process and $L_{1:T-1}$

2. *Denoising Process* 의 목적식 재구성

2) $\mu_\theta(x_t, t)$ 을 새롭게 정의 : *Denoising matching*

$$(2) \quad \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{1}{2\sigma_t^2} \left\| \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t(\mathbf{x}_0, \epsilon) - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon \right) - \boldsymbol{\mu}_\theta(\mathbf{x}_t(\mathbf{x}_0, \epsilon), t) \right\|^2 \right]$$

$$(3) \quad \boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \tilde{\boldsymbol{\mu}}_t \left(\mathbf{x}_t, \frac{1}{\sqrt{\bar{\alpha}_t}} \left(\mathbf{x}_t - \sqrt{1 - \bar{\alpha}_t} \epsilon_\theta(\mathbf{x}_t) \right) \right) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right)$$

식 (2)와 식 (3)을 조합하면 아래 식 (4)와 같은 새로운 목적식이 정의된다.

$$(4) \quad \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1 - \bar{\alpha}_t)} \left\| \epsilon - \epsilon_\theta \left(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t \right) \right\|^2 \right]$$

DDPM

□ Reverse Process and $L_{1:T-1}$

2. *Denoising Process* 의 목적식 재구성

2) $\mu_\theta(x_t, t)$ 을 새롭게 정의 : *Denoising matching*

$$(4) \quad \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1 - \bar{\alpha}_t)} \left\| \epsilon - \epsilon_\theta \left(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t \right) \right\|^2 \right]$$

결국 DDPM model(ϵ_θ)이 학습해야 하는 것은 주어진 t 시점의 gaussian noise(ϵ) 가 된다.

이처럼 각 시점의 다양한 scale의 gaussian noise를 예측해, denoising에 활용하고자 하는 것이 DDPM의 지향점이다.

최종적으로는 계수 term을 제외한 아래 식 (5)의 Loss를 사용

$$(5) \quad L_{\text{simple}}(\theta) := \mathbb{E}_{t, \mathbf{x}_0, \epsilon} \left[\left\| \epsilon - \epsilon_\theta \left(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t \right) \right\|^2 \right]$$

DDPM

□ Data Scaling, Reverse Process Decoder, and L_0

We assume that image data consists of integers in $\{0, 1, \dots, 255\}$ scaled linearly to $[-1, 1]$. This ensures that the neural network reverse process operates on consistently scaled inputs starting from the standard normal prior $p(\mathbf{x}_T)$. To obtain discrete log likelihoods, we set the last term of the reverse process to an independent discrete decoder derived from the Gaussian $\mathcal{N}(\mathbf{x}_0; \boldsymbol{\mu}_\theta(\mathbf{x}_1, 1), \sigma_1^2 \mathbf{I})$:

$$p_\theta(\mathbf{x}_0|\mathbf{x}_1) = \prod_{i=1}^D \int_{\delta_-(x_0^i)}^{\delta_+(x_0^i)} \mathcal{N}(x; \mu_\theta^i(\mathbf{x}_1, 1), \sigma_1^2) dx \quad (13)$$

$$\delta_+(x) = \begin{cases} \infty & \text{if } x = 1 \\ x + \frac{1}{255} & \text{if } x < 1 \end{cases} \quad \delta_-(x) = \begin{cases} -\infty & \text{if } x = -1 \\ x - \frac{1}{255} & \text{if } x > -1 \end{cases}$$

where D is the data dimensionality and the i superscript indicates extraction of one coordinate. (It would be straightforward to instead incorporate a more powerful decoder like a conditional autoregressive model, but we leave that to future work.) Similar to the discretized continuous distributions used in VAE decoders and autoregressive models [34, 52], our choice here ensures that the variational bound is a lossless codelength of discrete data, without need of adding noise to the data or incorporating the Jacobian of the scaling operation into the log likelihood. At the end of sampling, we display $\boldsymbol{\mu}_\theta(\mathbf{x}_1, 1)$ noiselessly.

DDPM

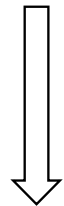
□ DDPM Loss

$$\begin{aligned} LOSS_{Diffusion} &= D_{KL}(q(z | x_0) || P_{\theta}(x_0 | z)) - E_{z \sim q(z|x)}[\log P_{\theta}(z)] \\ &= D_{KL}(q(z | x_0) || P_{\theta}(z)) + \sum_{t=2} D_{KL}(q(x_{t-1} | x_t, x_0) || P_{\theta}(x_{t-1} | x_t)) - E_q[\log P_{\theta}(x_0 | x_1)] \end{aligned}$$

Regularizer on Encoder

Denoising Process

Reconstruction on Decoder



DDPM에서는 Loss가 굉장히 간단한 식으로 정의됨

$$(5) \quad LOSS_{DDPM} = \mathbb{E}_{x_0, \epsilon} \left[\left| \epsilon - \epsilon_{\theta} \left(\sqrt{\tilde{\alpha}_t} + \sqrt{1 - \tilde{\alpha}_t} \epsilon, t \right) \right|^2 \right]$$

Experimental Results

□ Sample quality

Table 1: CIFAR10 results. NLL measured in bits/dim.

Model	IS	FID	NLL Test (Train)
Conditional			
EBM [11]	8.30	37.9	
JEM [17]	8.76	38.4	
BigGAN [3]	9.22	14.73	
StyleGAN2 + ADA (v1) [29]	10.06	2.67	
Unconditional			
Diffusion (original) [53]			≤ 5.40
Gated PixelCNN [59]	4.60	65.93	3.03 (2.90)
Sparse Transformer [7]			2.80
PixelIQN [43]	5.29	49.46	
EBM [11]	6.78	38.2	
NCSNv2 [56]		31.75	
NCSN [55]	8.87 ± 0.12	25.32	
SNGAN [39]	8.22 ± 0.05	21.7	
SNGAN-DDLS [4]	9.09 ± 0.10	15.42	
StyleGAN2 + ADA (v1) [29]	9.74 ± 0.05	3.26	
Ours (L , fixed isotropic Σ)	7.67 ± 0.13	13.51	≤ 3.70 (3.69)
Ours (L_{simple})	9.46 ± 0.11	3.17	≤ 3.75 (3.72)

- DDPM의 FID score는 3.17로서 Unconditional 생성모형 중 가장 높은 sample quality를 보임
- 계수 Term을 제외한 경우($Ours : L_{\text{simple}}$)는 제외하지 않은 경우 ($Ours : L$)에 비해 NLL이 높지만, Sample quality(FID score)는 월등히 높음을 확인할 수 있음

Experimental Results

□ Sample quality

Table 2: Unconditional CIFAR10 reverse process parameterization and training objective ablation. Blank entries were unstable to train and generated poor samples with out-of-range scores.

	Objective	IS	FID
$\tilde{\mu}$ prediction (baseline)			
(1)	L , learned diagonal Σ	7.28 ± 0.10	23.69
	L , fixed isotropic Σ	8.06 ± 0.09	13.22
(2)	$\ \tilde{\mu} - \tilde{\mu}_\theta\ ^2$	—	—
ϵ prediction (ours)			
(3)	L , learned diagonal Σ	—	—
	L , fixed isotropic Σ	7.67 ± 0.13	13.51
(4)	$\ \tilde{\epsilon} - \epsilon_\theta\ ^2$ (L_{simple})	9.46 ± 0.11	3.17

$$(1) \quad \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)\|^2 \right]$$

$$(2) \quad \mathbb{E}_q \left[\|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)\|^2 \right]$$

$$(3) \quad \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1 - \bar{\alpha}_t)} \left\| \epsilon - \epsilon_\theta \left(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t \right) \right\|^2 \right]$$

$$(4) \quad \mathbb{E}_{t, \mathbf{x}_0, \epsilon} \left[\left\| \epsilon - \epsilon_\theta \left(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t \right) \right\|^2 \right]$$

매우 불안정한 학습 및 성능이 매우 악화된 경우, "—" 로 표기

- $\tilde{\mu}$ 예측 목적식의 경우, Fixed variance를 사용했을 때, 성능이 향상됨. 다만, 목적식의 간단화(simplification)에서는 성능이 매우 악화됨
- ϵ 예측 목적식의 경우, Fixed variance를 사용할 경우 & 목적식의 간단화를 적용할 경우 성능이 크게 향상됨

Experimental Results

□ Sample quality

	ϵ prediction (ours)	
(3) L , learned diagonal Σ	–	–
(3) L , fixed isotropic Σ	7.67 ± 0.13	13.51
(4) $\ \tilde{\epsilon} - \epsilon_\theta\ ^2 (L_{\text{simple}})$	9.46 ± 0.11	3.17

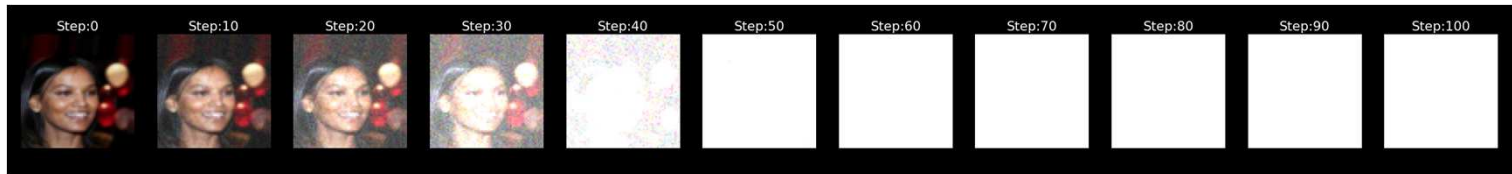
Coefficient term

$$(3) \quad \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1 - \bar{\alpha}_t)} \left\| \epsilon - \epsilon_\theta \left(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t \right) \right\|^2 \right]$$

$$(4) \quad \mathbb{E}_{t, \mathbf{x}_0, \epsilon} \left[\left\| \epsilon - \epsilon_\theta \left(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t \right) \right\|^2 \right]$$

Small “t”

Large “t”



- 목적식 (3) : *Coefficient term* 은 t 가 증가할 수록 그 값이 작아지는 경향을 갖게 됨
 - 따라서 *Coefficient term* 이 존재할 경우, large t 시점(더 noisy)의 Loss가 상대적으로 down-weight 되는 효과가 발생
- 목적식 (4) : *Coefficient term* 을 제거함으로써, large t 시점(더 noisy)의 Loss 비중을 더 높일 수 있음
 - 이를 통해 모델이 noise가 심한 이미지(large t 시점의 상태)의 denoising에 집중하도록 유도

Experimental Results

□ Progressive generation

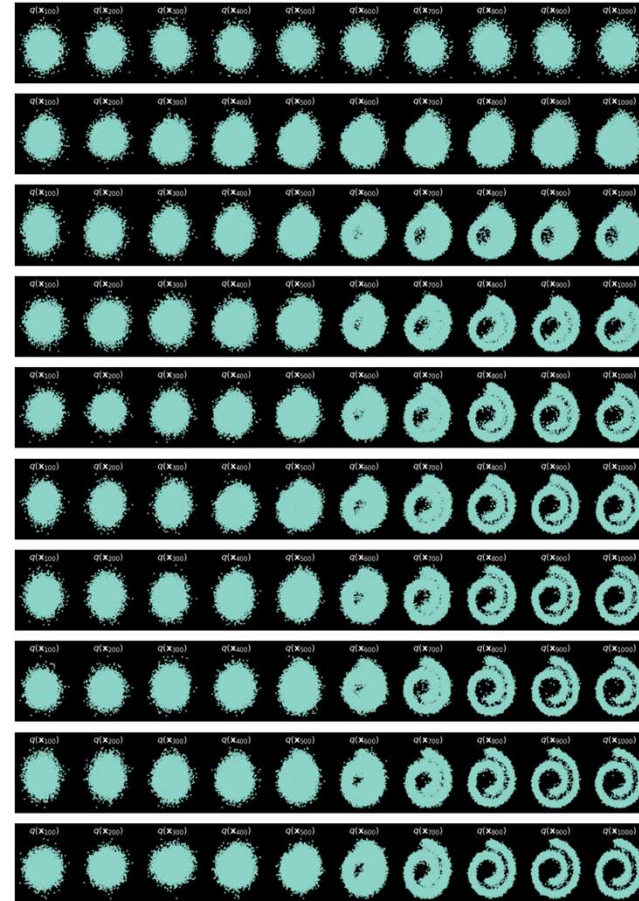


Diffusion "tutorial code"

□ Tutorial code

- *tutorial code* [[Link](https://github.com/acids-ircam/diffusion_models)]
 - ✓ Reference : https://github.com/acids-ircam/diffusion_models

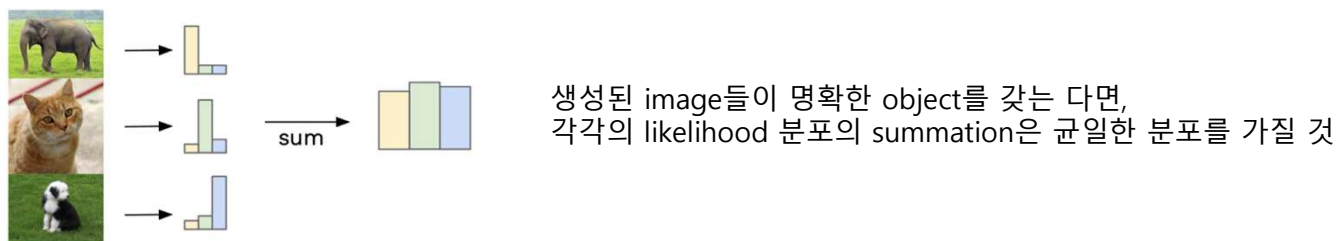
[From blog post from 'A simple explanation of the Inception Score\(David Mack\)'](#)



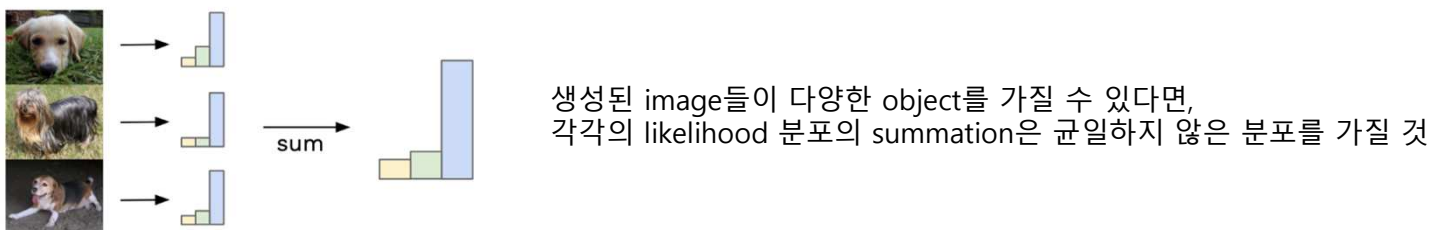
IS score

□ Inception Score : the higher, the better

- *Improved Techniques for Training GANs*(Tim Salimans, 2016)에서 제안된 generated image sample quality 평가 지표
- Inception image classifier를 활용해 'inception'이라는 명칭을 사용함
- 2가지 특성을 통해 생성된 이미지의 품질을 평가. 아래 2가지를 만족할 수록 높은 score 획득
 - ✓ Image quality : 생성된 이미지가 명확히 object를 표현하고 있는 지(realistic?)



- ✓ Image diversity : 다양한 object가 생성되는 지('dog' class : {Bulldog, Poodle, ...})



둘 간의 KL divergence로
Inception Score 산출

FID score

□ FID(Frechet Inception Distance) score : the lower, the better

- Pre-trained Image classification model을 활용해 추출한 feature representation 간의 거리(distance)를 score로 활용
 - 거리(distance)는 Frechet distance로 측정
 - o Frechet distance는 곡선을 이루는 points의 위치와 순서를 고려해 두 곡선 간 유사도를 측정하는 지표

예를 들어, 일변량 정규분포 간 Frechet distance는 아래와 같이 측정됨

$$d(X, Y) = (\mu_X - \mu_Y)^2 + (\sigma_X - \sigma_Y)^2$$

- Feature 정보는 후반부 layer에서 추출한 high level representation을 활용
- FID score의 계산은 아래와 같은 순서로 진행

1. *Pretrained image classification model define(ex: inception model)*

2. *Compute embedding(feature representation)*

2 – 1) *Embedding of real image*

2 – 2) *Embedding of generated image*

3. *Calculate "Frechet distance" between 2 – 1) and 2 – 2)*