Grooming Diffusion

Basic diffusion

NAVER AI Lab

김준호

https://github.com/taki0112

NAVER AI LAB

Recall

NAVER AI LAB

Youtube

- The recipe of GANs
  - 2014 ~ 2020 GANs 모델 연구 요약 (기초 ~ 심화)
- The diffusion theory
  - diffusion을 이해하기 위한 이론 (기초)
- The applications of diffusion
  - Text-to-image 모델 소개 (심화)

완전히 똑같진 않으나, 복습하기에 좋을것입니다. :)

01
ML Research / Backbone Research

- Fundamental Machine Learning
- Machine Learning Optimization
- Visual Backbone
- Vision-and-Language
- Trustworthy AI

02
Generation Research

- Image-to-Image Translation
- Visual Generative Models
- Multi-modal Generation

03
Language Research

- Hyperscale AI
- Language Representations
- AI Ethics

04
HCI Research

- Large Language Models (LLM)-based HCI

05
Healthcare AI

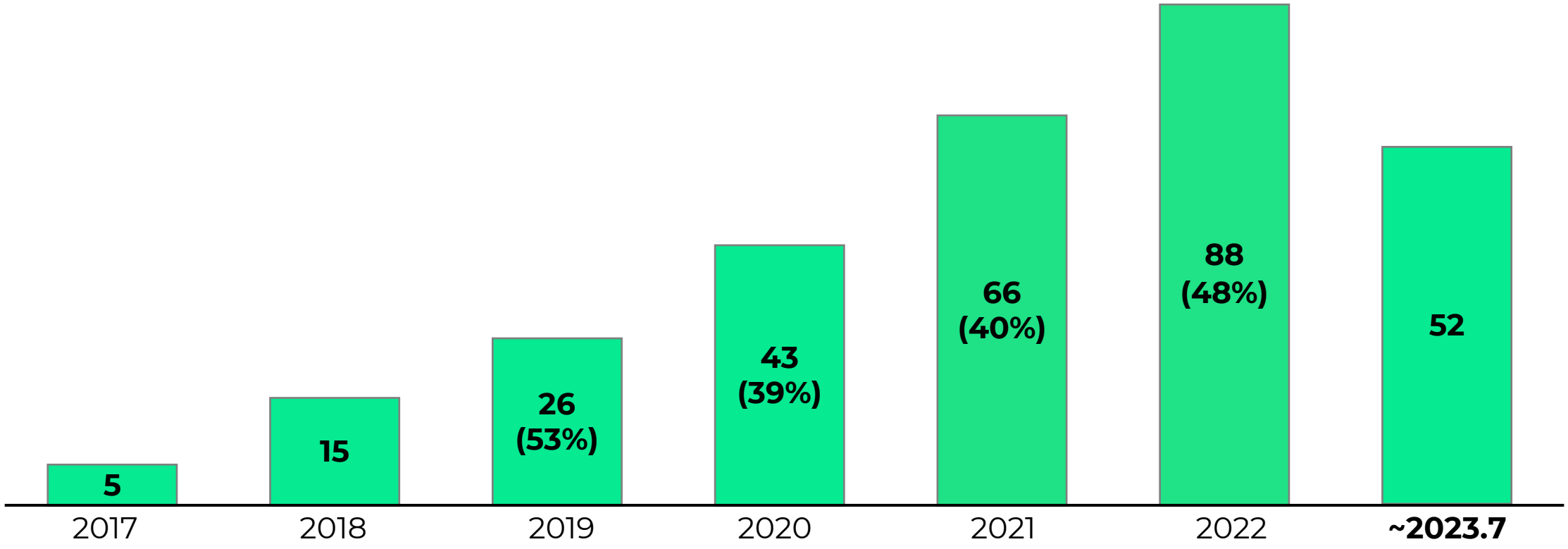
- Healthcare AI

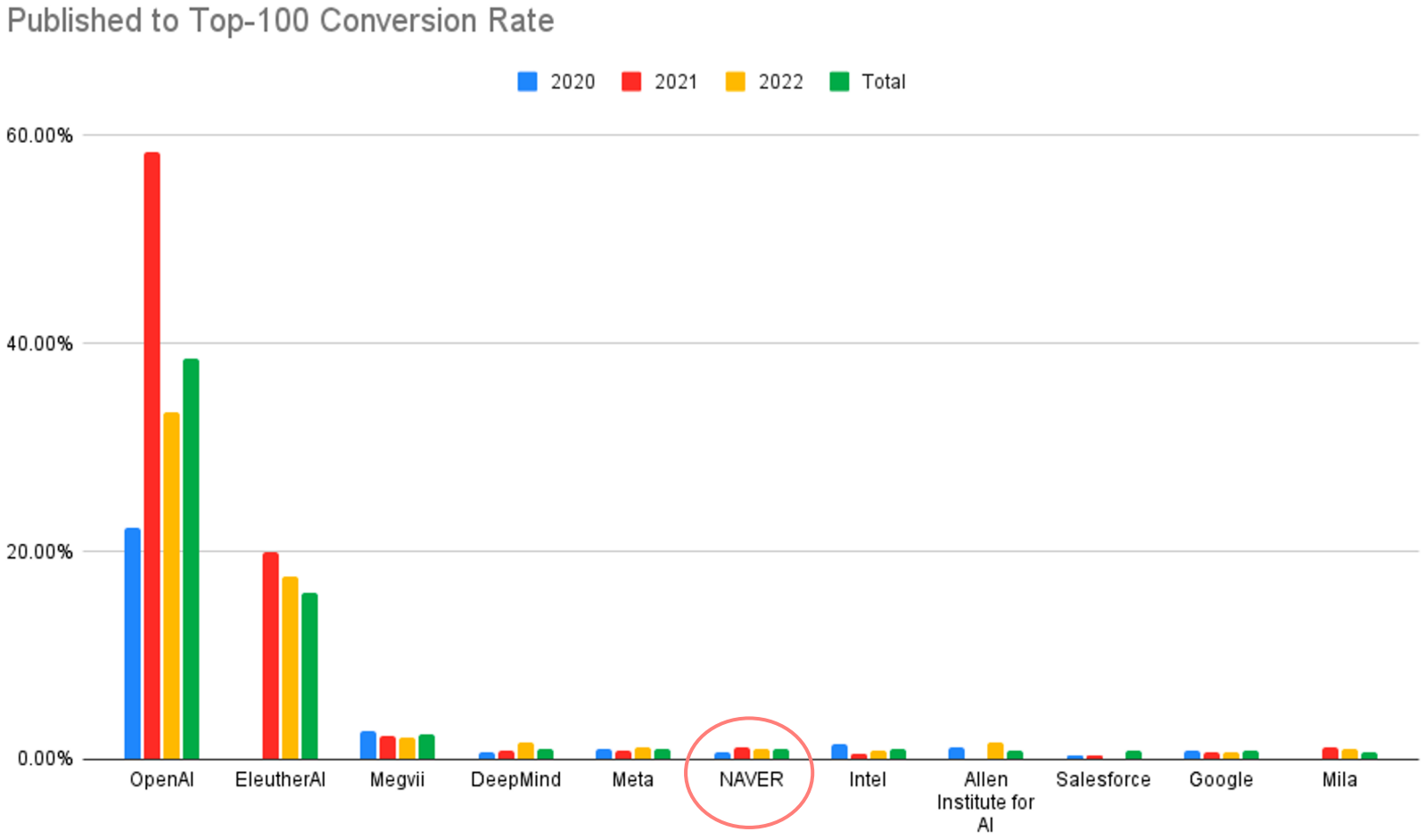
06
Neural 3D Research

- Neural Radiance Fields (NeRFs)

Publications @ Top AI Conferences of NAVER CLOVA

- NeurIPS, ICML, ICLR, CVPR, ICCV, ECCV, ACL, EMNLP, AAAI, ICASSP, Interspeech, …
- 40% of papers are applied to the services



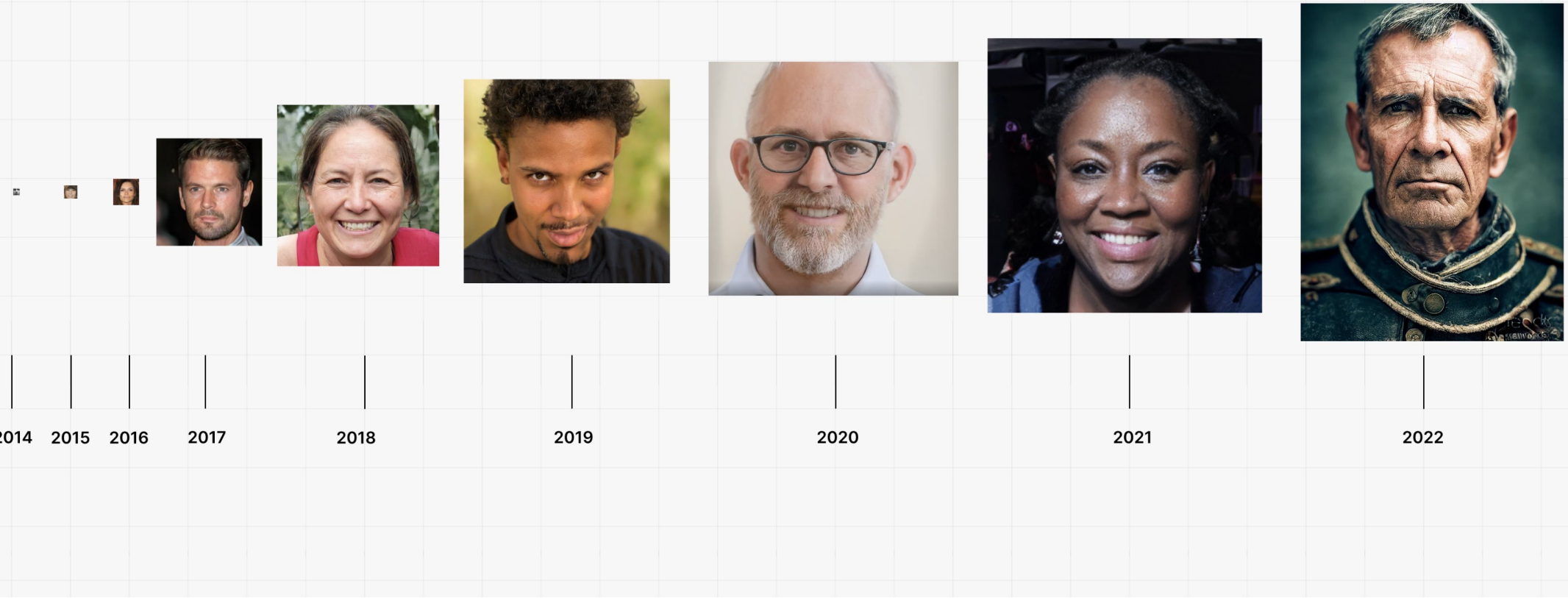


https://www.zeta-alpha.com/post/must-read-the-100-most-cited-ai-papers-in-2022

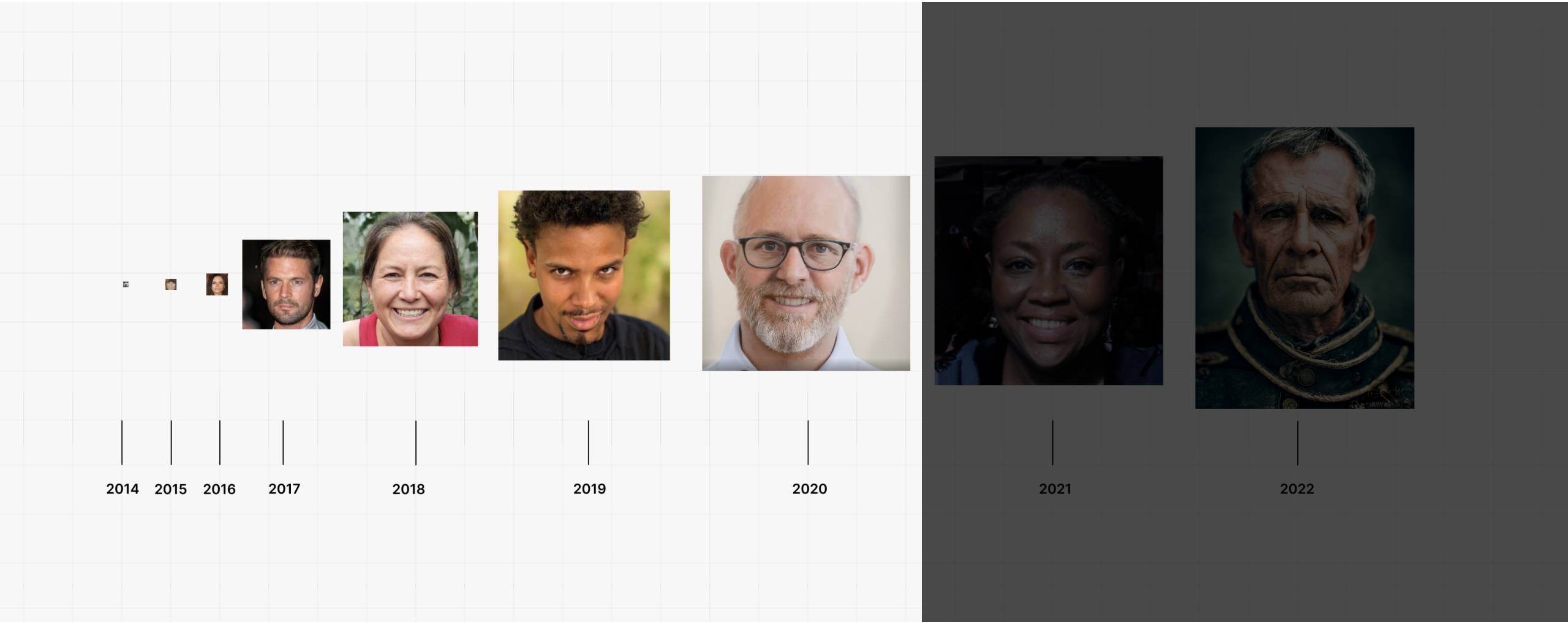
Intro

Overview

NAVER AI LAB



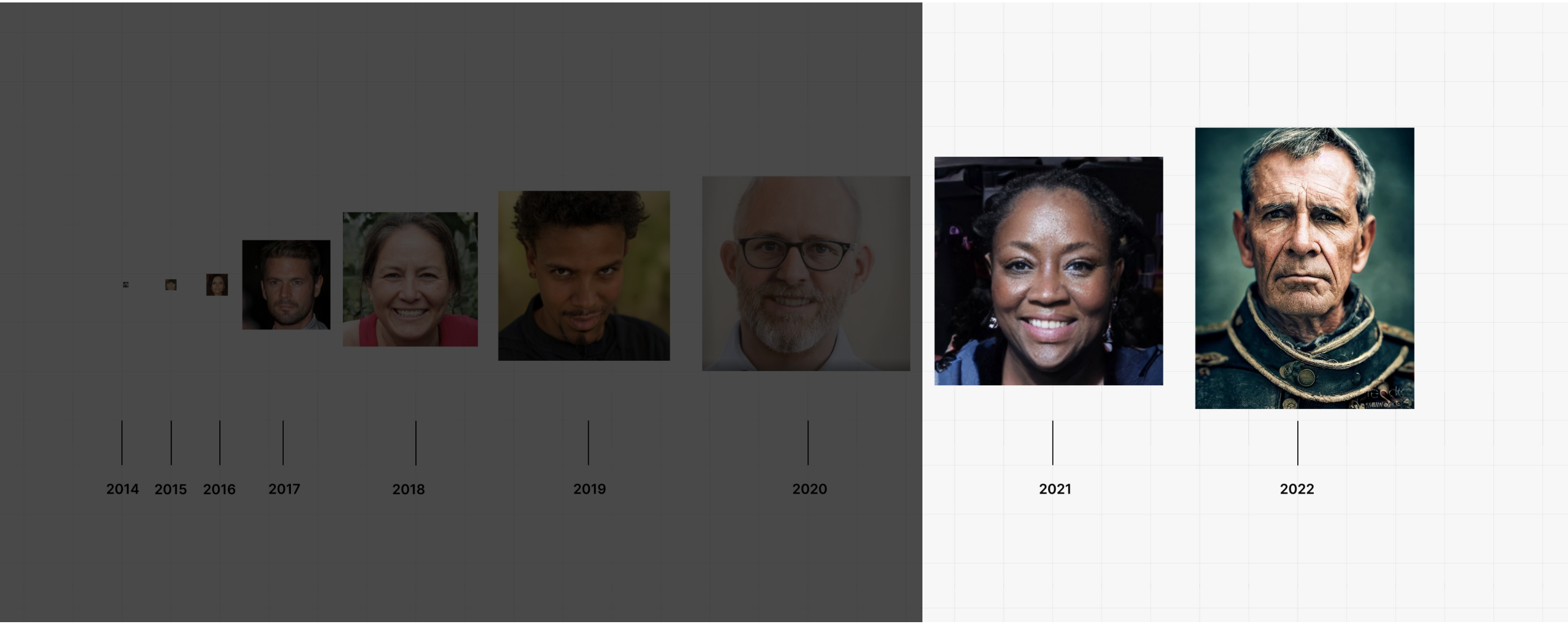
NAVER AI LAB



과거를 이해하는 시간

16개 논문

NAVER AI LAB



현재를 이해하고, 미래를 준비하는 시간

Grooming the diffusion

Intro | Summary of GANs

NAVER AI LAB

 # 2014 ~ 2016

- Unconditional generation
  - GANs
  - Diverse loss
- Conditional generation
  - ACGAN
  - Multi-task discriminator
  - Projection discriminator

Intro | Summary of GANs

NAVER AI LAB

2014 ~ 2016

- Unconditional generation
  - GANs
  - Diverse loss
- Conditional generation
  - ACGAN
  - Multi-task discriminator
  - Projection discriminator

2017 ~ 2018

- Progressive GAN
  - Progressive training
- BigGAN
  - Conditional batch normalization
  - Large scale
  - Truncation trick
- StyleGAN
  - Disentangle the latent space with mapping layer
  - Style Mixing (determine the coarse, middle, fine style)
    - A module = Global aspects
    - B module = Local aspects
  - Truncation trick

Intro | Summary of GANs

NAVER AI LAB

2014 ~ 2016

- Unconditional generation
  - GANs
  - Diverse loss
- Conditional generation
  - ACGAN
  - Multi-task discriminator
  - Projection discriminator

2017 ~ 2018

- Progressive GAN
  - Progressive training
- BigGAN
  - Conditional batch normalization
  - Large scale
  - Truncation trick
- StyleGAN
  - Disentangle the latent space with mapping layer
  - Style Mixing (determine the coarse, middle, fine style)
    - A module = Global aspects
    - B module = Local aspects
  - Truncation trick

2019 ~ 2020

- StyleGAN2
  - StyleGAN + Weight modulation + Lazy regularization.
- DiffAugment
  - Prevent the overfitting in a discriminator.
  - Apply the differentiable augmentation to generator & discriminator.
- ADA
  - Prevent the overfitting in a discriminator.
  - Apply the adaptively augmentation to generator & discriminator.

Intro | Summary of GANs

NAVER AI LAB

2014 ~ 2016

- Unconditional generation
  - GANs
  - Diverse loss
- Conditional generation
  - ACGAN
  - Multi-task discriminator
  - Projection discriminator

2017 ~ 2018

- Progressive GAN
  - Progressive training
- BigGAN
  - Conditional batch normalization
  - Large scale
  - Truncation trick
- StyleGAN
  - Disentangle the latent space with mapping layer
  - Style Mixing (determine the coarse, middle, fine style)
    - A module = Global aspects
    - B module = Local aspects
  - Truncation trick

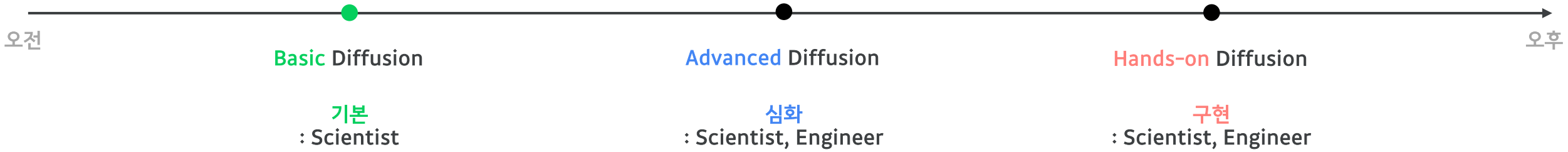
2019 ~ 2020

- StyleGAN2
  - StyleGAN + Weight modulation + Lazy regularization.
- DiffAugment
  - Prevent the overfitting in a discriminator.
  - Apply the differentiable augmentation to generator & discriminator.
- ADA
  - Prevent the overfitting in a discriminator.
  - Apply the adaptively augmentation to generator & discriminator.

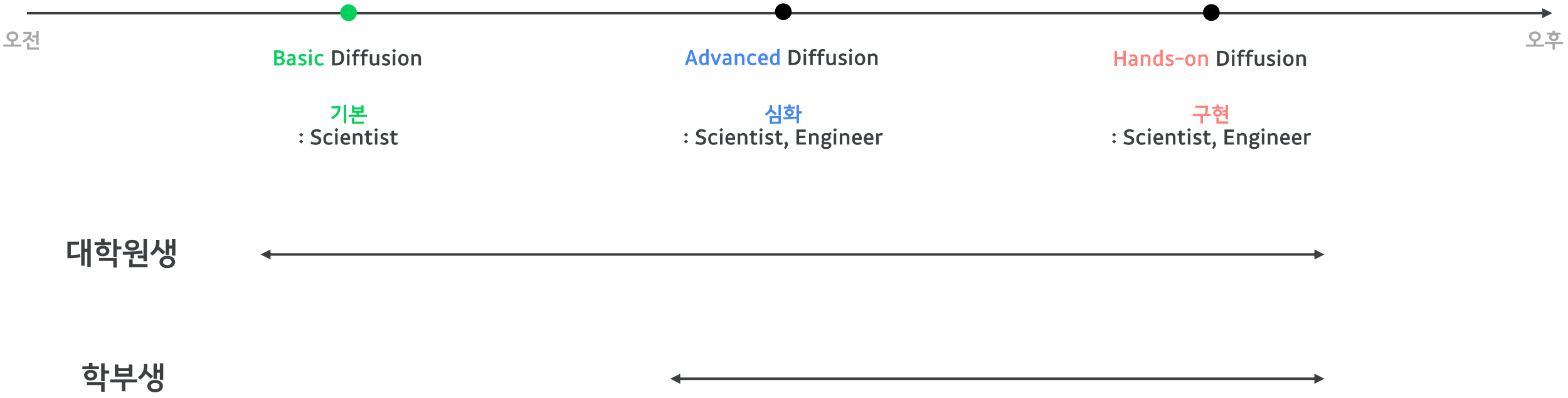
2014 ~ 2020: Techniques

- Consistency regularization
  - CR-GAN: augmented real images for discriminator.
  - bCR-GAN: augmented real & fake images for discriminator.
  - zCR-GAN: augmented latent codes for generator & discriminator.
  - ICR-GAN: bCR + zCR
- FSMR: Feature Statistics Mixing Regularization
  - Reduce style-bias in discriminator.
- GGDR: Generator Guided Discriminator Regularization
  - Dense supervision for discriminator.

NAVER AI LAB

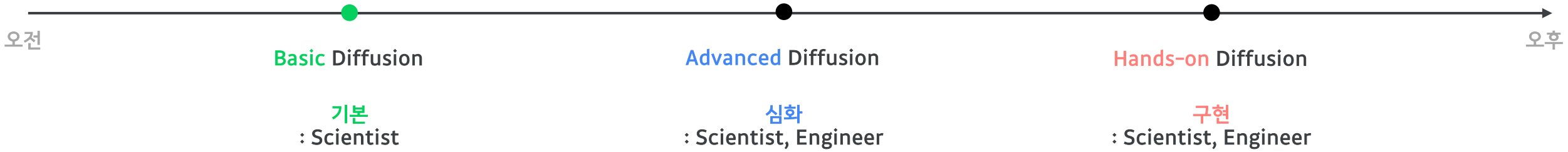

오전
Basic Diffusion
기본
: Scientist
Advanced Diffusion
심화
: Scientist, Engineer
Hands-on Diffusion
구현
: Scientist, Engineer
오후

NAVER AI LAB


오전
오후
Basic Diffusion
기본
: Scientist
Advanced Diffusion
심화
: Scientist, Engineer
Hands-on Diffusion
구현
: Scientist, Engineer
대학원생
학부생

NAVER AI LAB

github.com/taki0112/diffusion-pytorch



기본
: Scientist

심화
: Scientist, Engineer

구현
: Scientist, Engineer

대학원생

학부생

NAVER AI LAB

오전

Basic Diffusion

기본
: Scientist

Advanced Diffusion

심화
: Scientist, Engineer

Hands-on Diffusion

구현
: Scientist, Engineer

오후

2020

NeurIPS 2020

Denoising diffusion probabilistic models (DDPM)

Denoising diffusion implicit models (DDIM)

ICLR 2021

NeurIPS 2021

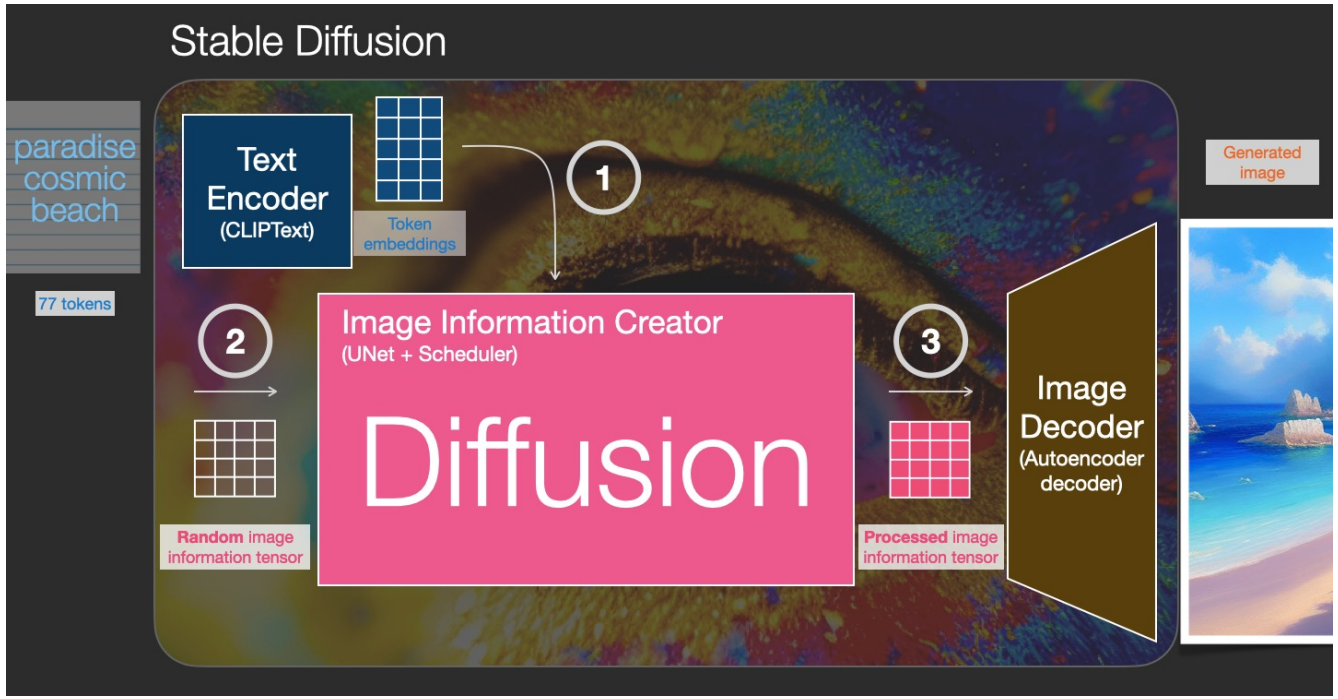
Diffusion Models Beat GANs on Image Synthesis

Tackling the Generative Learning Trilemma
with Denoising Diffusion GANs

ICLR 2022

2022

NAVER AI LAB



"An astronaut riding a horse in a photorealistic style"

↓

DALLE 2

↓

"A brain riding a rocketship heading towards the moon"

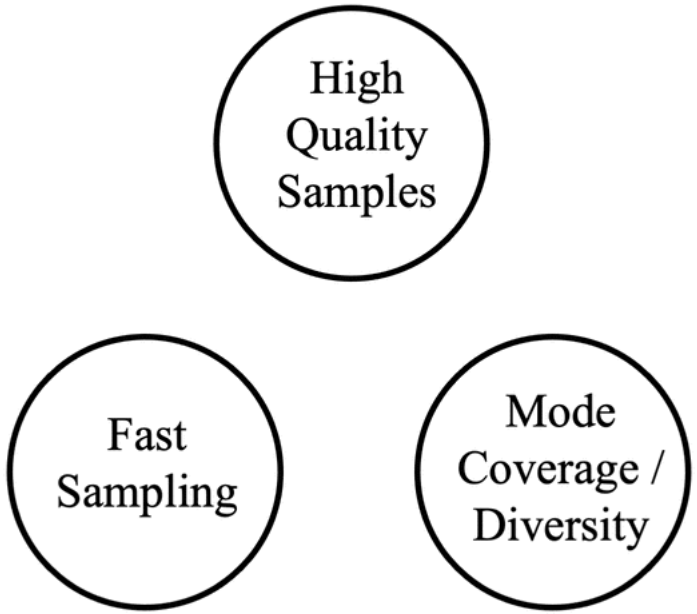
↓

Imagen

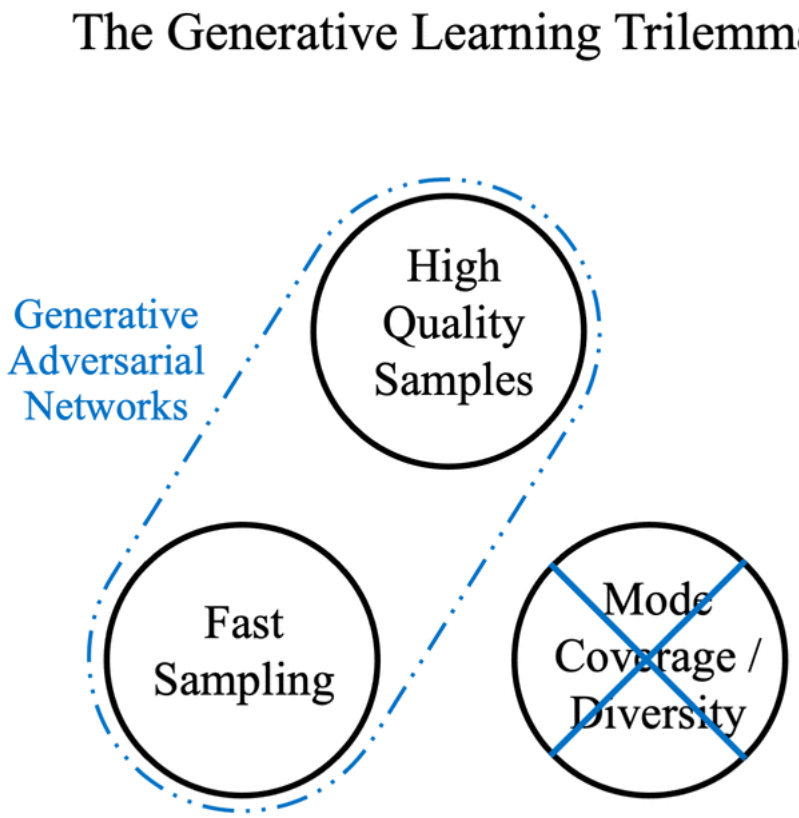
↓

NAVER AI LAB

The Generative Learning Trilemma



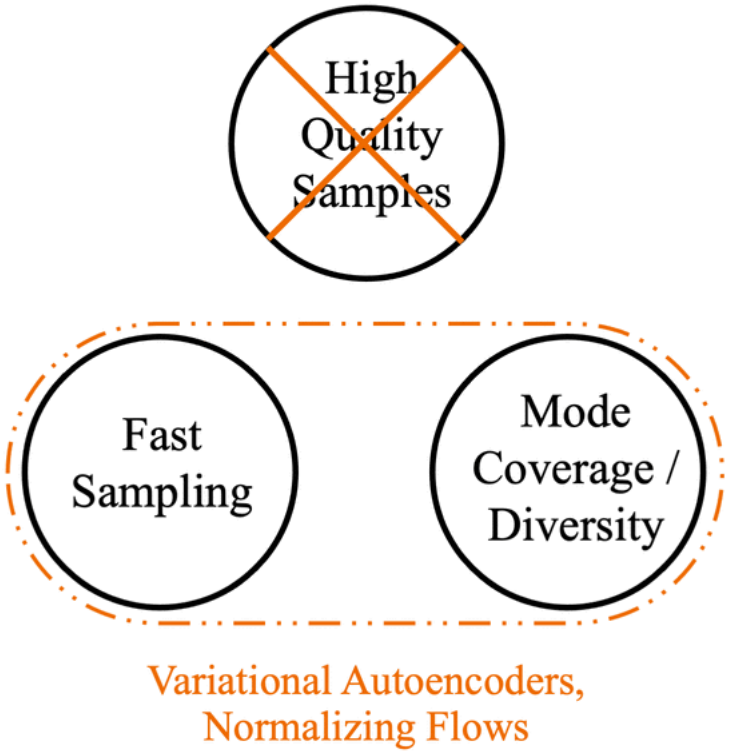
Tackling the Generative Learning Trilemma with Denoising Diffusion GANs. ICLR 2022



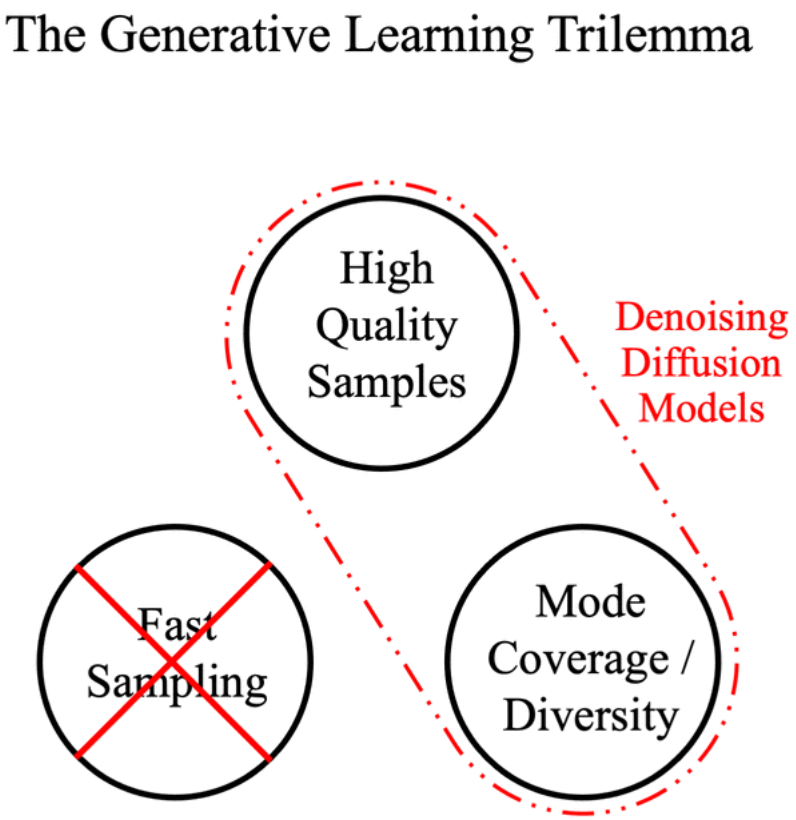
Tackling the Generative Learning Trilemma with Denoising Diffusion GANs. ICLR 2022

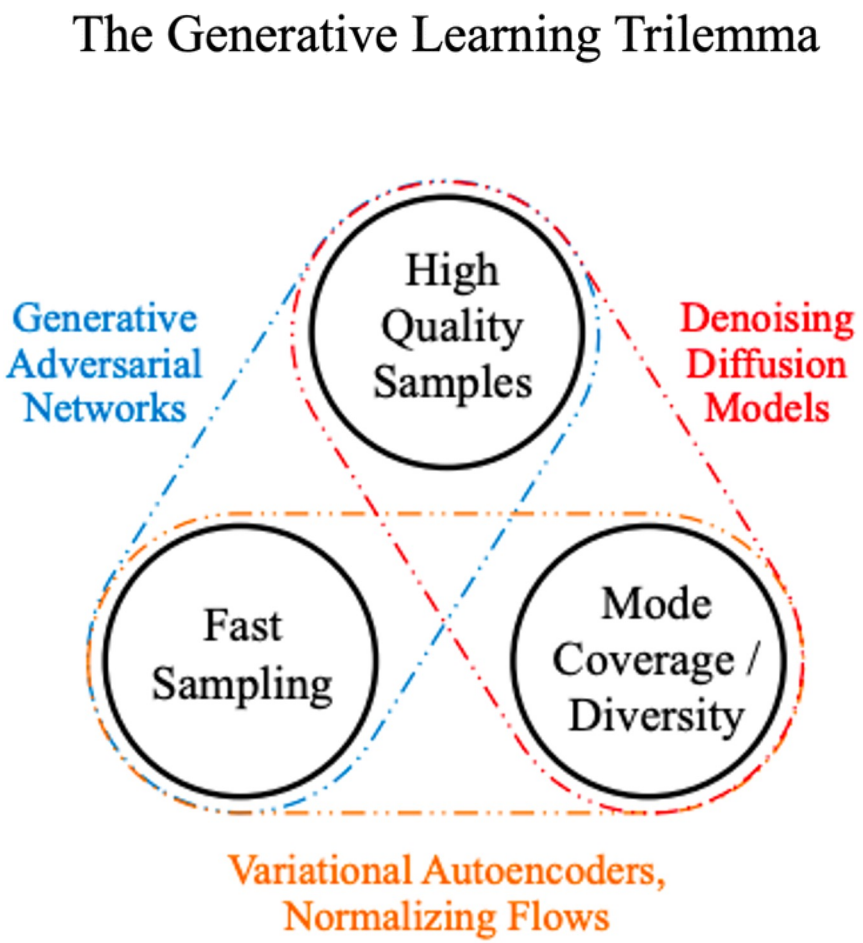
NAVER AI LAB

The Generative Learning Trilemma



Tackling the Generative Learning Trilemma with Denoising Diffusion GANs. ICLR 2022


The Generative Learning Trilemma
High Quality Samples
Denoising Diffusion Models
Fast Sampling
Mode Coverage / Diversity


The Generative Learning Trilemma
High Quality Samples
Generative Adversarial Networks
Denoising Diffusion Models
Fast Sampling
Mode Coverage / Diversity
Variational Autoencoders, Normalizing Flows

Diffusion process

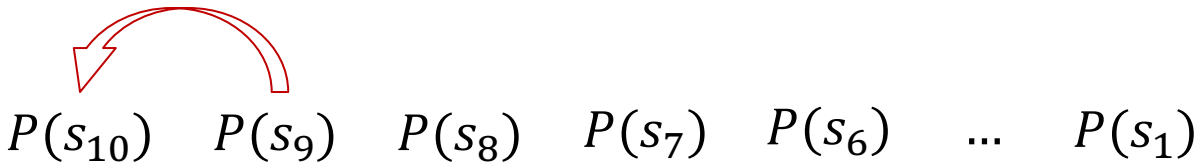
NAVER AI LAB

❑ Markov Chain

- Markov 성질을 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



$$P(s_{10}) \quad P(s_9) \quad P(s_8) \quad P(s_7) \quad P(s_6) \quad \dots \quad P(s_1)$$

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

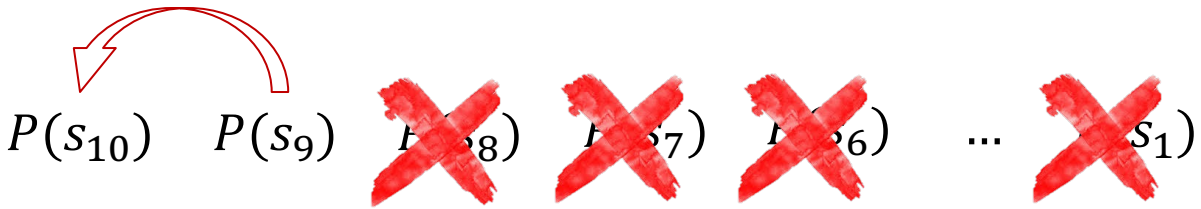
NAVER AI LAB

❑ Markov Chain

- Markov 성질을 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



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

NAVER AI LAB

+ noise

X_0

X_T

– noise

NAVER AI LAB

Diffusion process

+ noise

X_0

X_T

- noise

Denoising process

NAVER AI LAB

Forward process

Diffusion process

+ noise

X_0

X_T

– noise

Denoising process

Reverse process

NAVER AI LAB

Forward process

Diffusion process

$q(X_t \mid X_{t-1})$ + noise

X_0 X_T

– noise $q(X_{t-1} \mid X_t)$

Denoising process

Reverse process

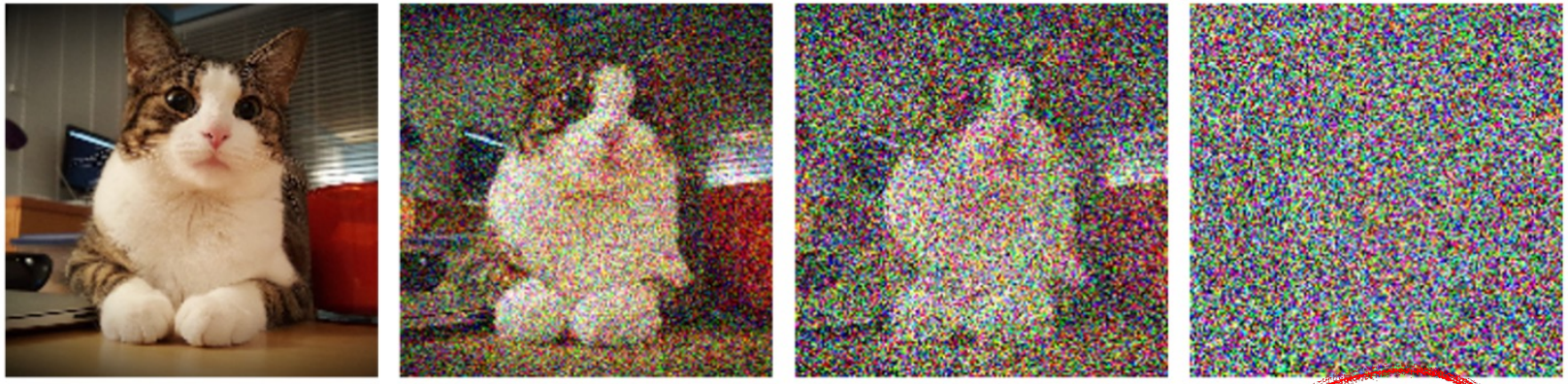
NAVER AI LAB

Forward process

Diffusion process

$q(X_t \mid X_{t-1})$

+ noise



X_0

X_T

- noise

$q(X_{t-1} \mid X_t)$

학습을 통해
가능하게 만들자

Denoising process

Reverse process

NAVER AI LAB

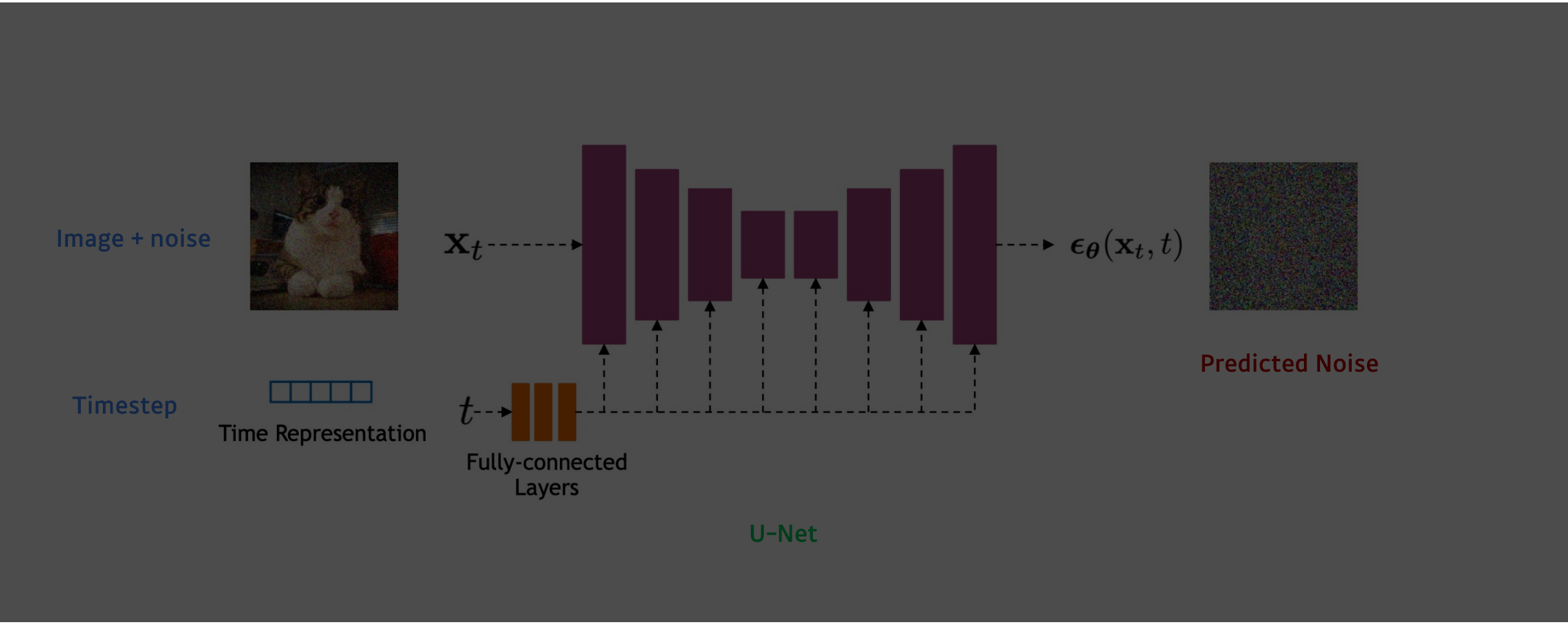


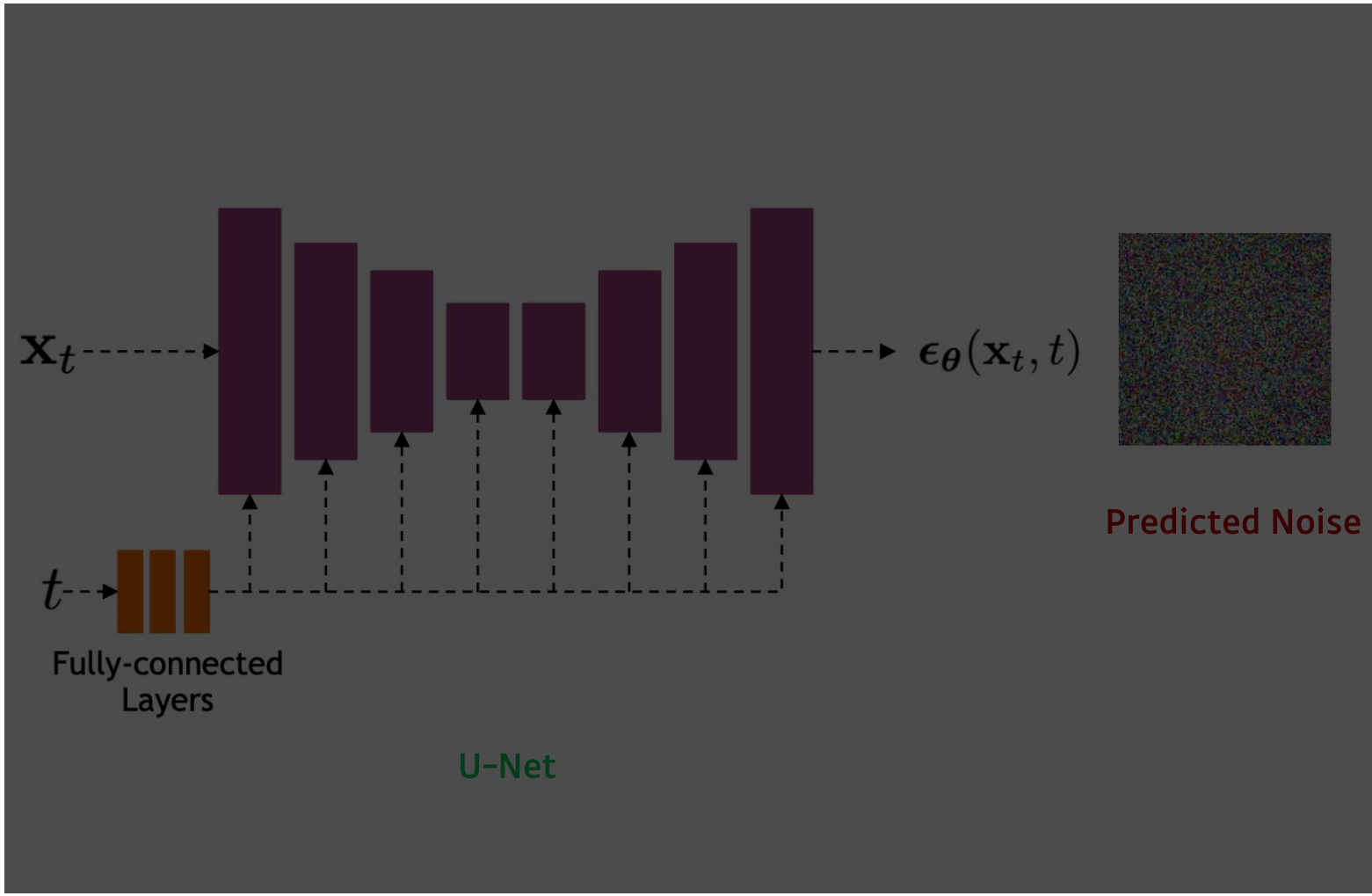
Image + noise
$\mathbf{x}_t$
$\epsilon_\theta(\mathbf{x}_t, t)$
Predicted Noise
Timestep
Time Representation
t
Fully-connected Layers
U-Net

NAVER AI LAB

Image + noise

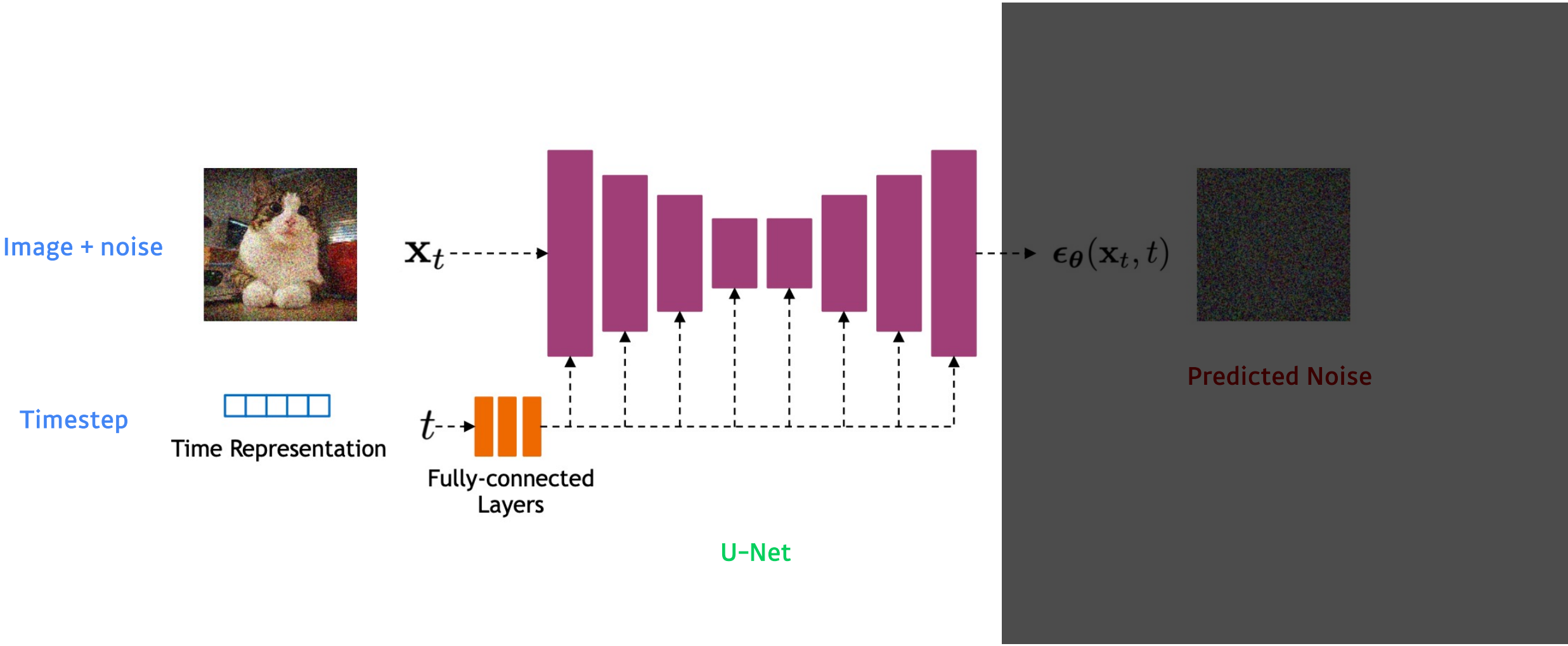
Timestep

Time Representation



Input

NAVER AI LAB



Input

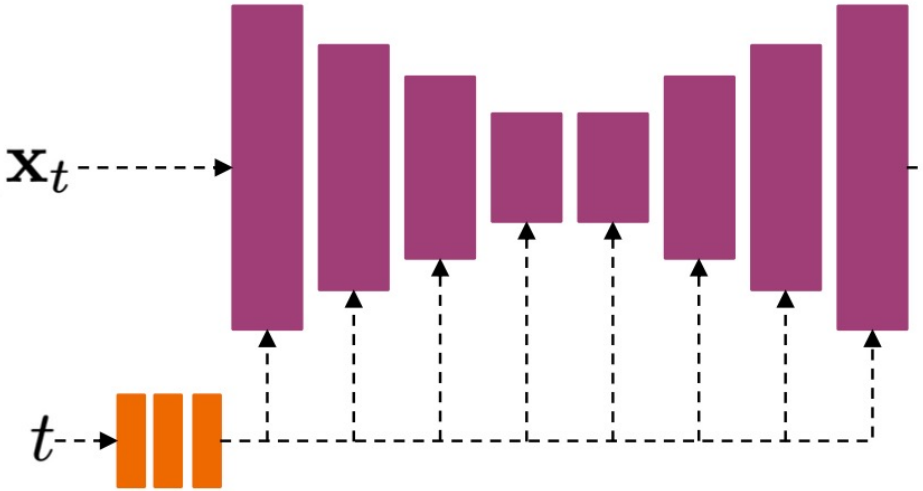
Network

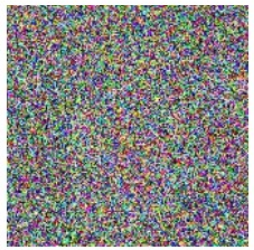
NAVER AI LAB

Image + noise

Timestep

Time Representation





Predicted Noise

U-Net

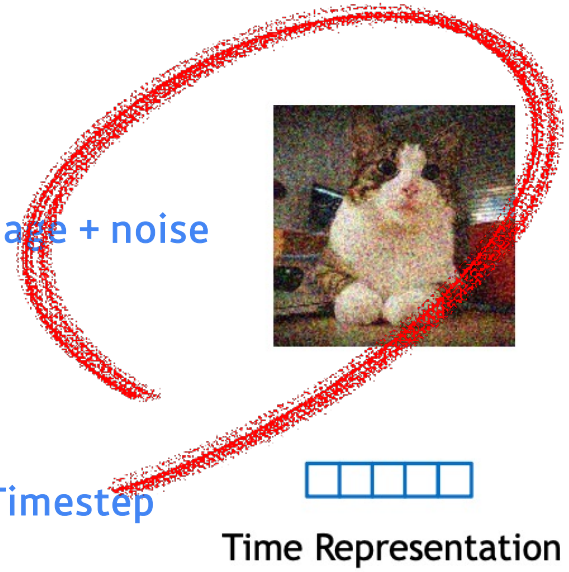
Input

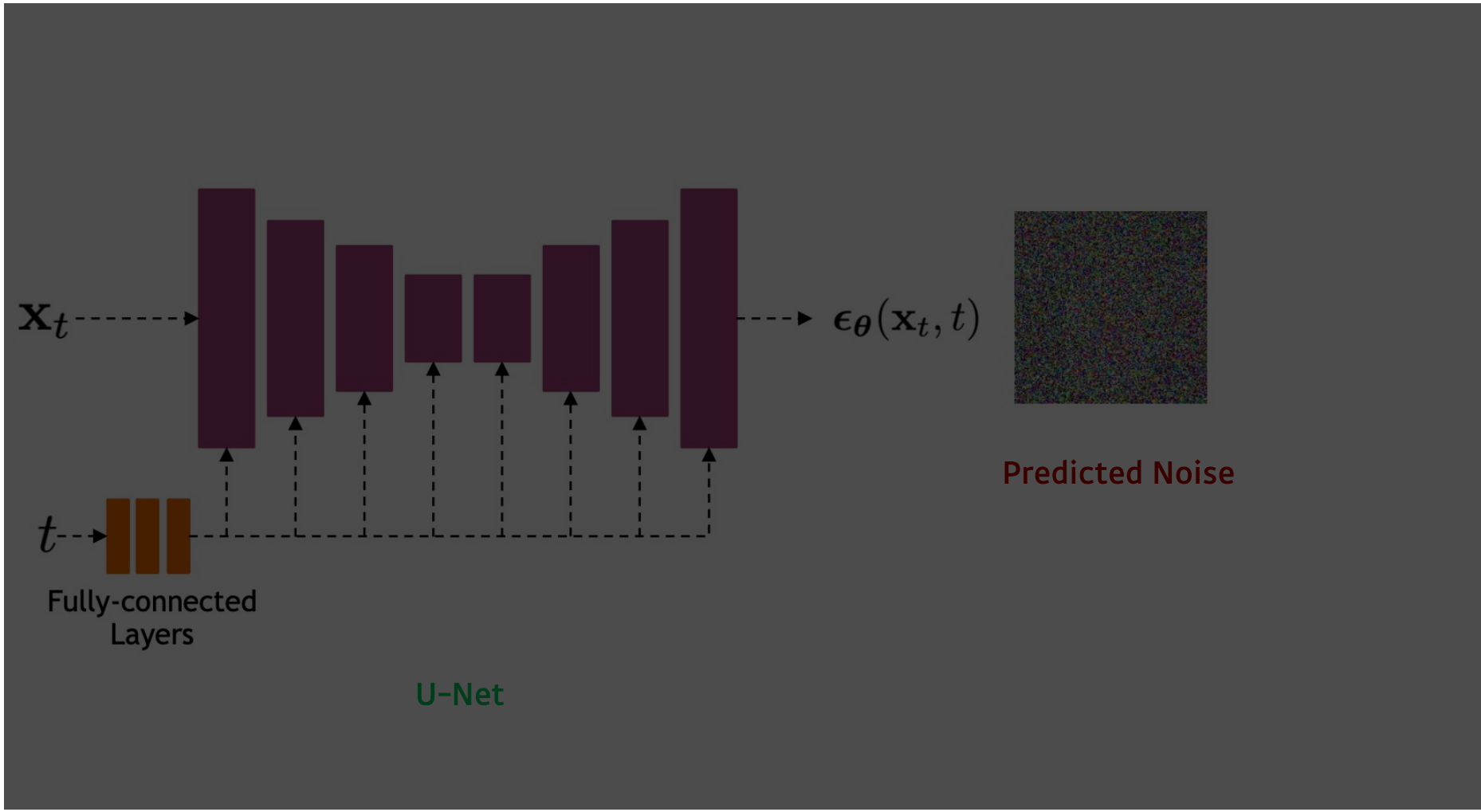
Network

Output

NAVER AI LAB

Forward

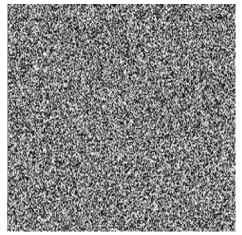




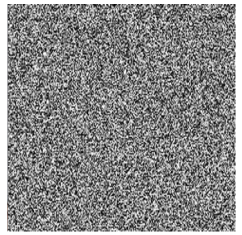
Input

Network

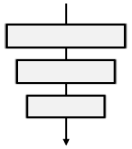
Output

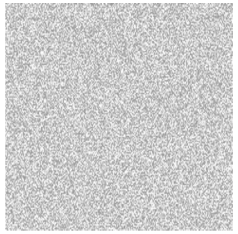


x_T

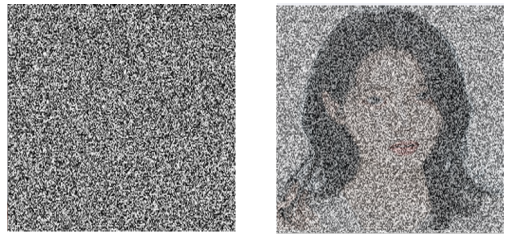


x_T

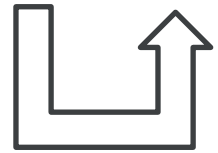


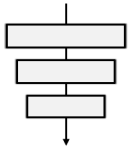


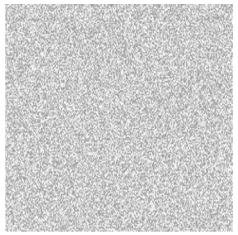
z_T



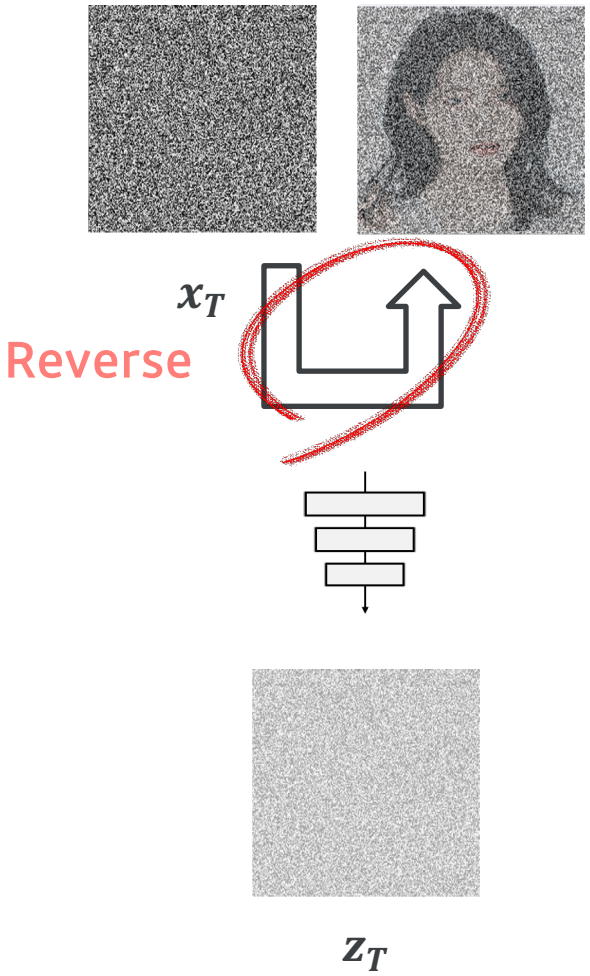
x_T

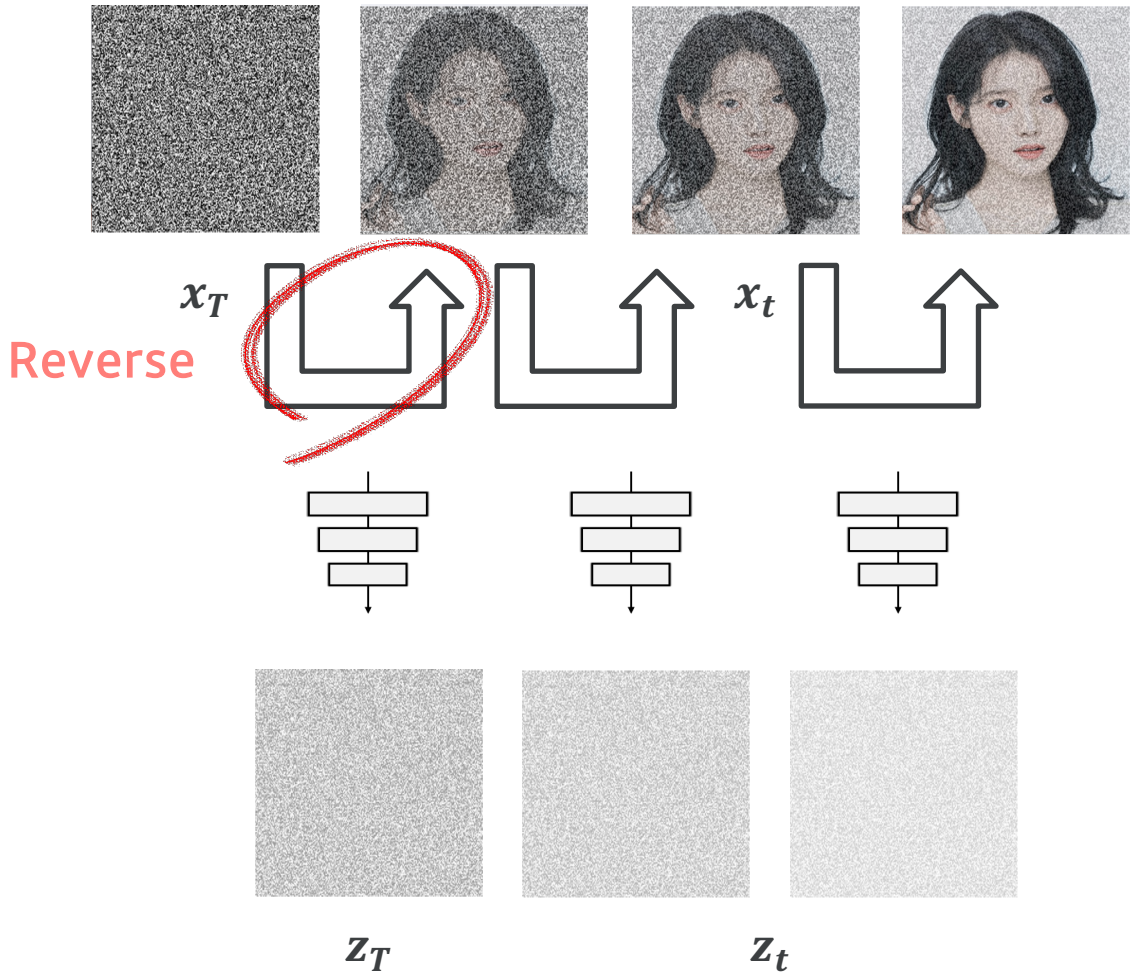






z_T


x_T
Reverse
z_T


x_T
x_t
Reverse
z_T
z_t


x_T
x_t
x_0
Reverse
z_T
z_t
z_1

NAVER AI LAB

DDPM

DDIM

Diffusion | Summary

NAVER AI LAB

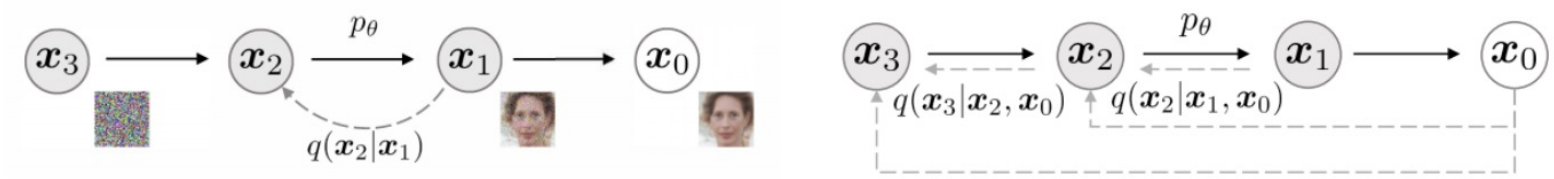


Figure 1: Graphical models for diffusion (left) and non-Markovian (right) inference models.

Denoising Diffusion Implicit Models

DDPM

- $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon,\ \epsilon \sim \mathcal{N}(0, \mathbf{I})$ `(Forward)`
  - $\beta_1 = 10^{-4}, \beta_T = 0.02$
  - $\alpha_t := 1 - \beta_t,\ \bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$
- $\epsilon - \epsilon_\theta(\mathbf{x}_t) = \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon)$ `(Loss)`
  - ϵ_θ = prediction network
- $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right) + \sqrt{\tilde{\beta}_t}\epsilon,\ \epsilon \sim \mathcal{N}(0, \mathbf{I})$ `(Reverse)`
  - $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$
    - $\tilde{\beta}_t = \beta_t$로 해도 성능차이 없음

DDIM

- `In paper,` DDIM α = DDPM $\bar{\alpha}$
- $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$ `(Forward)`
- $\epsilon - \epsilon_\theta(\mathbf{x}_t) = \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon)$ `(Loss)`
  - ϵ_θ = prediction network
- $\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\underbrace{\left(\frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(\mathbf{x}_t)}{\sqrt{\bar{\alpha}_t}}\right)}_{\text{predicted } \mathbf{x}_0 = f_\theta(\mathbf{x}_t)} + \underbrace{\sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\cdot\epsilon_\theta(\mathbf{x}_t)}_{\text{direction pointing to } \mathbf{x}_t} + \underbrace{\sigma_t\epsilon}_{\text{noise}}$

  `(Reverse)`
  - deterministic when $\sigma_t = 0$ → consistency (DDIM)
  - stochastic when $\sigma_t = 1$ → inconsistency (DDPM)

- Prob & Stats

- Distribution

- Advanced

- Prob & Stats
  - $X = N(\mu, \Sigma)$ mean, variance
    - $\varepsilon \sim N(0, I)$
  - $E(aX + b) = aE(x) + b$
  - $V(aX + b) = a^2V(X)$
  - $\sigma(aX + b) = |a|\sigma(X)$ standard deviation = var^2
  - $P(B \mid A) = P(A \mid B)\frac{P(B)}{P(A)}$
- Distribution
- Advanced

NAVER AI LAB

- Prob & Stats
  - $X = N(\mu, \Sigma)$ mean, variance
    - $\varepsilon \sim N(0, I)$
  - $E(aX + b) = aE(x) + b$
  - $V(aX + b) = a^2V(X)$
  - $\sigma(aX + b) = |a|\sigma(X)$ standard deviation = var^2
  - $P(B \mid A) = P(A \mid B)\frac{P(B)}{P(A)}$

- Distribution
  - $q(x)$: $real\ distribution$
  - $p_\theta(x)$: $network\ distribution$

- Advanced

- Prob & Stats
  - $X = N(\mu, \Sigma)$ mean, variance
    - $\varepsilon \sim N(0, I)$
  - $E(aX + b) = aE(x) + b$
  - $V(aX + b) = a^2V(X)$
  - $\sigma(aX + b) = |a|\sigma(X)$ standard deviation = var^2
  - $P(B \mid A) = P(A \mid B)\frac{P(B)}{P(A)}$

- Distribution
  - $q(x)$: $real\ distribution$
  - $p_\theta(x)$: $network\ distribution$

- Advanced
  - $N(x;\ \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} = f(x)$ probability density function (pdf)
  - $T_f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n = f(a) + f'(a)(x-a) + \frac{1}{2}f''(a)(x-a)^2 + \ \ldots$
    - $\approx f(a) + f'(a)(x-a)$

- Prob & Stats
  - $X = N(\mu, \Sigma)$ mean, variance
    - $\varepsilon \sim N(0, I)$
  - $E(aX + b) = aE(x) + b$
  - $V(aX + b) = a^2V(X)$
  - $\sigma(aX + b) = |a|\sigma(X)$ standard deviation = var^2
  - $P(B \mid A) = P(A \mid B)\frac{P(B)}{P(A)}$

- Distribution
  - $q(x)$: $real\ distribution$
  - $p_\theta(x)$: $network\ distribution$

- Advanced
  - $N(x;\ \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} = f(x)$ probability density function (pdf)
  - $T_f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n = f(a) + f'(a)(x-a) + \frac{1}{2}f''(a)(x-a)^2 + \ \ldots$
    - $\approx f(a) + f'(a)(x-a)$

Goal

Read a paper

NAVER AI LAB

3.1 DDPM Sampling with Manifold Constraint

In DDPMs (Ho et al., 2020), starting from a clean image $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$, a forward diffusion process $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ is described as a Markov chain that gradually adds Gaussian noise at every time steps t:

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

where $\{\beta\}_{t=0}^{T}$ is a variance schedule. By denoting $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$, the forward diffused sample at t, i.e. $\boldsymbol{x}_t$, can be sampled in one step as:

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \quad \text{where} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}). \tag{2}$$

As the reverse of the forward step $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is intractable, DDPM learns to maximize the variational lowerbound through a parameterized Gaussian transitions $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ with the parameter θ. Accordingly, the reverse process is approximated as Markov chain with learned mean and fixed variance, starting from $p(\boldsymbol{x}_T) = \mathcal{N}(\boldsymbol{x}_T; \boldsymbol{0}, \boldsymbol{I})$:

$$p_\theta(\boldsymbol{x}_{0:T}) := p_\theta(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t), \quad \text{where} \quad p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) := \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \sigma_t^2 \boldsymbol{I}). \tag{3}$$

where

$$\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) := \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right), \tag{4}$$

Here, $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$ is the diffusion model trained by optimizing the objective:

$$\min_\theta L(\theta), \quad \text{where} \quad L(\theta) := \mathbb{E}_{t, \boldsymbol{x}_0, \boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\|^2\right]. \tag{5}$$

After the optimization, by plugging learned score function into the generative (or reverse) diffusion process, one can simply sample from $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ by

$$\boldsymbol{x}_{t-1} = \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) + \sigma_t \boldsymbol{\epsilon} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right) + \sigma_t \boldsymbol{\epsilon} \tag{6}$$

In image translation using *conditional* diffusion models (Saharia et al., 2022a; Sasaki et al., 2021), the diffusion model $\boldsymbol{\epsilon}_\theta$ in (5) and (6) should be replaced with $\boldsymbol{\epsilon}_\theta(\boldsymbol{y}, \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)$ where $\boldsymbol{y}$ denotes the matched target image. Accordingly, the sample generation is tightly controlled by the matched target in a supervised manner, so that the image content change rarely happen. Unfortunately, the requirement of the *matched* targets for the training makes this approach impractical.

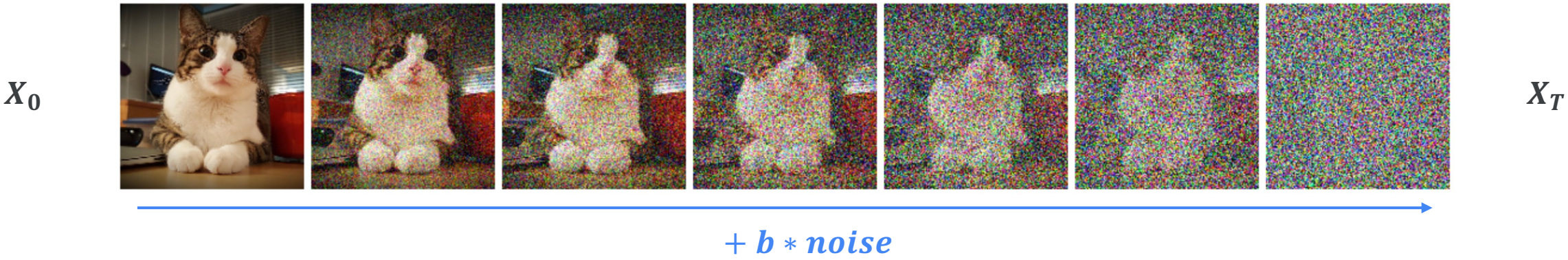
To address this, Dhariwal & Nichol (2021) proposed classifier-guided image translation using the unconditional diffusion model training as in (5) and a pre-trained classifier $p_\phi(\boldsymbol{y}|\boldsymbol{x}_t)$. Specifically, $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)$ in (4) and (6) are supplemented with the gradient of the classifier, i.e. $\hat{\mu}_\theta(\boldsymbol{x}_t, t) := \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) + \sigma_t \nabla_{\boldsymbol{x}_t} \log p_\phi(\boldsymbol{y}|\boldsymbol{x}_t)$. However, most of the classifiers, which should be separately trained, are not usually sufficient to control the content of the samples from the reverse diffusion process.

Forward process

Diffusion | Forward process

NAVER AI LAB

- Overview (mean, variance)
  - $q(X_t \mid X_{t-1}) = N(X_t;\ \mu_{X_{t-1}},\ \Sigma_{X_{t-1}})$
    - Conditional gaussian distribution



Diffusion | Forward process

NAVER AI LAB

- Overview
  - $q(X_t \mid X_{t-1}) = N(X_t;\ \mu_{X_{t-1}},\ \Sigma_{X_{t-1}})$ (mean, variance)
    - Conditional gaussian distribution

X_0

X_T

$+\ b * noise$

$$x_t = a * x_{t-1} + b * noise$$

Diffusion | Forward process

NAVER AI LAB

- Overview
  - $q(X_t \mid X_{t-1}) = N\left(X_t;\ \mu_{X_{t-1}},\ \Sigma_{X_{t-1}}\right) = N\left(X_t; \sqrt{1-\beta_t}X_{t-1}, \beta_t * I\right)$ (mean, variance)
    - Conditional gaussian distribution
    - $0 < \beta_1 < \beta_2 < \ ... < \beta_T < 1$
    - 0.0001 ~ 0.02

Note

$Var(aX) = a^2 Var(X)$

$$x_t = a * x_{t-1} + b * noise$$

$$\boldsymbol{x_t = \sqrt{1-\beta_t} * x_{t-1} + \sqrt{\beta_t} * noise}$$

Diffusion | Forward process

NAVER AI LAB

- Overview
  - $q(X_t \mid X_{t-1}) = N(X_t;\ \mu_{X_{t-1}},\ \Sigma_{X_{t-1}}) = N(X_t; \sqrt{1-\beta_t}X_{t-1}, \beta_t * I)$ (mean, variance)
    - Conditional gaussian distribution
    - $0 < \beta_1 < \beta_2 < \ ... < \beta_T < 1$
    - 0.0001 ~ 0.02
  - $x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\varepsilon$ (Reparameterization trick) x = mean + std * noise
    - $\varepsilon \sim N(0, I)$

Note

$Var(aX) = a^2 Var(X)$

X_0 X_T

$+ \beta * noise$

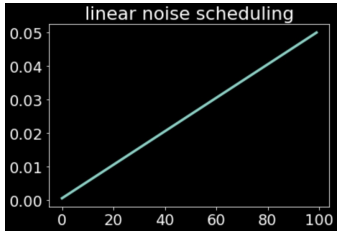
$x_t = a * x_{t-1} + b * noise$

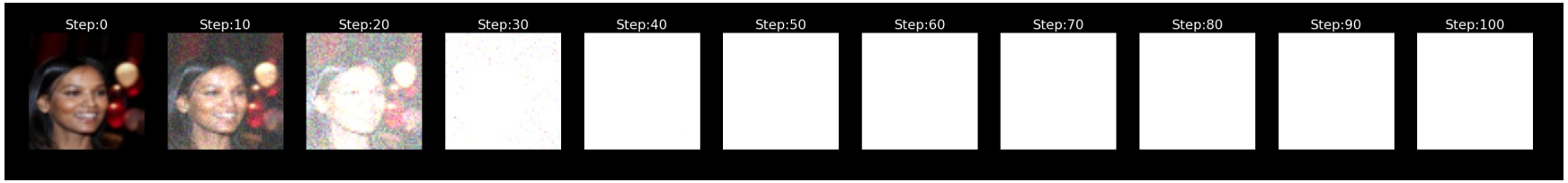
$$x_t = \sqrt{1-\beta_t} * x_{t-1} + \sqrt{\beta_t} * noise$$

NAVER AI LAB

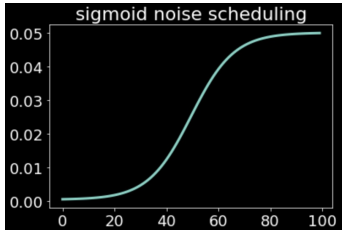
- Overview
  - Linear, Quad, Sigmoid, Cosine, ...

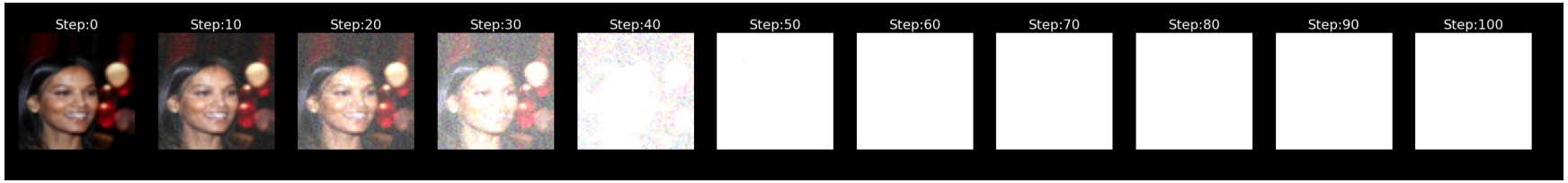
✓ Linear scheduling



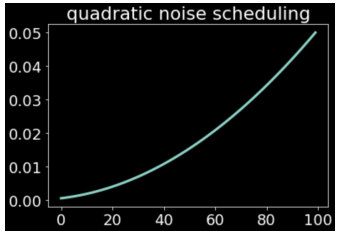


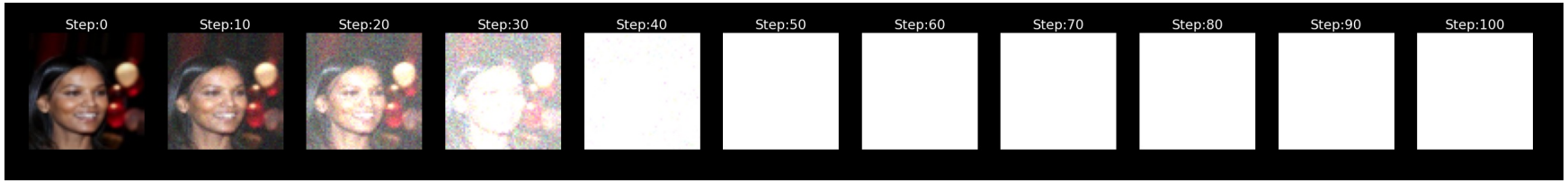
✓ Sigmoid scheduling





✓ Quadratic scheduling



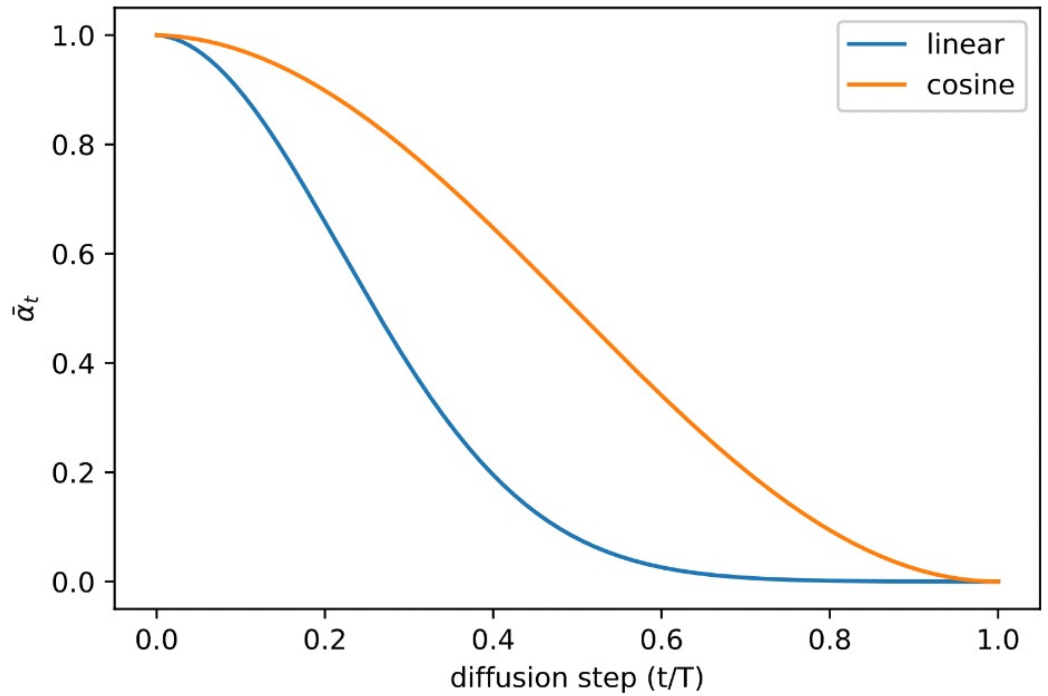


NAVER AI LAB

- Overview
  - Linear, Quad, Sigmoid, Cosine, ...

$$\beta_t = \text{clip}(1 - \frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}, 0.999) \quad \bar{\alpha}_t = \frac{f(t)}{f(0)} \quad \text{where } f(t) = \cos\left(\frac{t/T + s}{1+s} \cdot \frac{\pi}{2}\right)$$

where the small offset s is to prevent β_t from being too small when close to $t = 0$.



Diffusion | Forward process (x_t, x_0)

NAVER AI LAB

- Overview
  - $q(x_t \mid x_{t-1}) = N\left(x_t; \sqrt{1-\beta_t} x_{t-1}, \beta_t * I\right)$

Diffusion | Forward process (x_t, x_0)

NAVER AI LAB

- Overview
  - $q(x_t \mid x_{t-1}) = N\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t * I\right)$
  - $q(x_t \mid x_0) = N\left(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t) * I\right)$
    - $\alpha_t = 1 - \beta_t$
    - $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$

Diffusion | Forward process (x_t, x_0)

NAVER AI LAB

- Overview
  - $q(x_t \mid x_{t-1}) = N\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t * I\right)$
  - $q(x_t \mid x_0) = N\left(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t) * I\right)$
    - $\alpha_t = 1 - \beta_t$
    - $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$
  - $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$

Note

$0.0001 < \beta_t < 0.02$

- linear, cosine ...

Note

$$
\begin{aligned}
x_t &= \sqrt{\alpha_t}x_{t-1} + \sqrt{1-\alpha_t}z_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{\alpha_t}\sqrt{1-\alpha_{t-1}}z_{t-2} + \sqrt{1-\alpha_t}z_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}x_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{z}_{t-2} \ldots (*) \\
&= \cdots \\
&= \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}z
\end{aligned}
$$

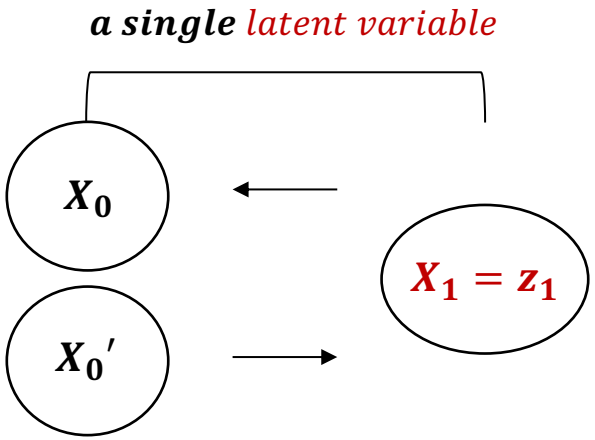
(*) $X \sim \mathcal{N}(\mu_X, \sigma_X^2)$와 $Y \sim \mathcal{N}(\mu_Y, \sigma_Y^2)$에서 $Z = X + Y$는 $Z \sim \mathcal{N}(\mu_X + \mu_Y, \sigma_X^2 + \sigma_Y^2)$입니다.

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

Objective for Reverse process

NAVER AI LAB

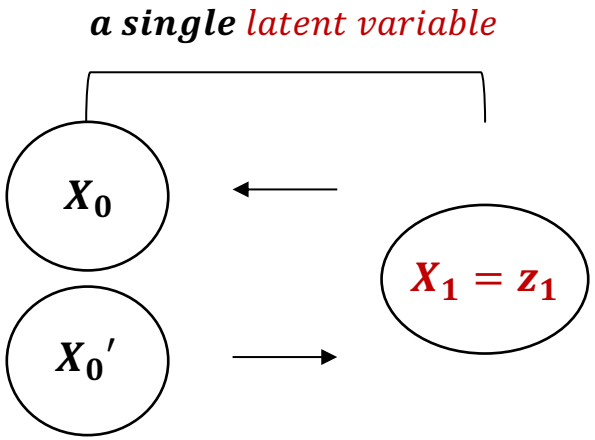
- ✓ VAE



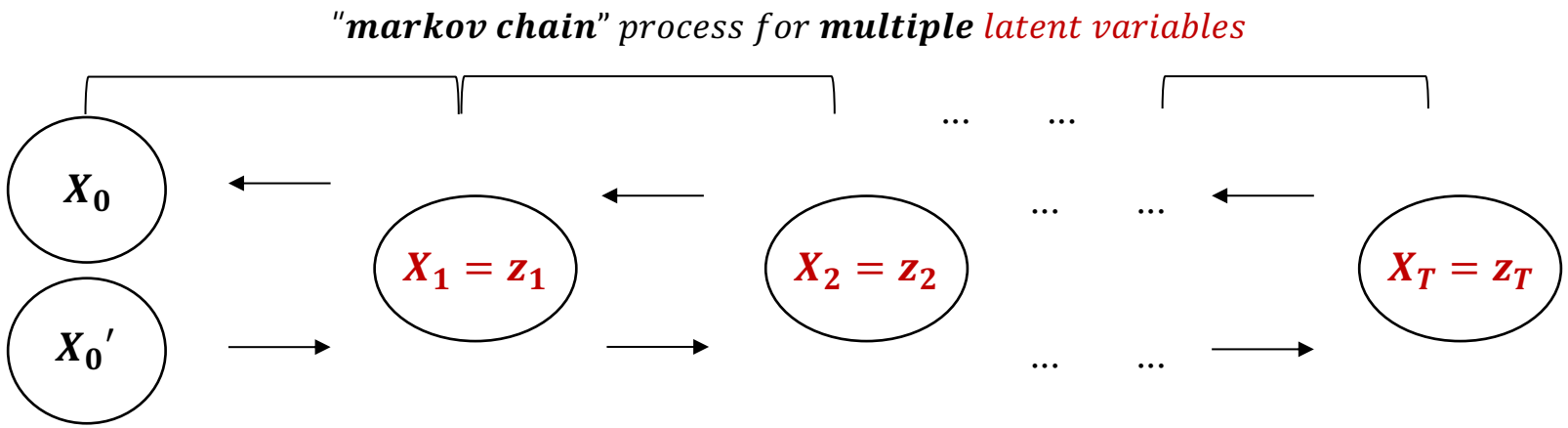
Diffusion | Objective for Reverse process

NAVER AI LAB

✓ VAE



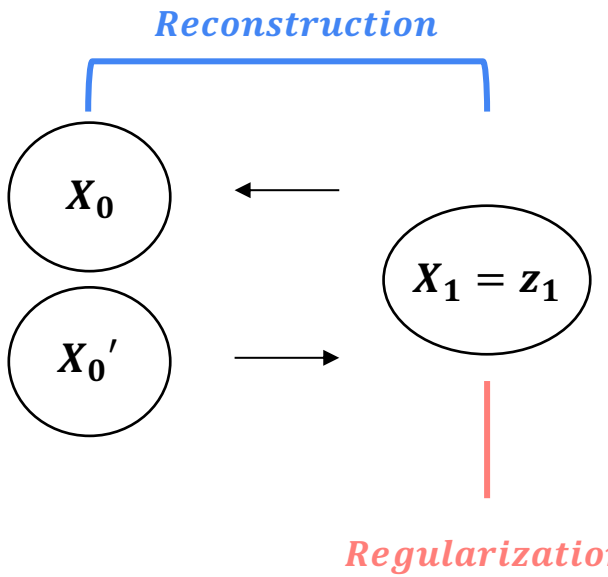
✓ Diffusion



www.youtube.com/watch?v=_JQSMhqXw-4

NAVER AI LAB

✓ VAE



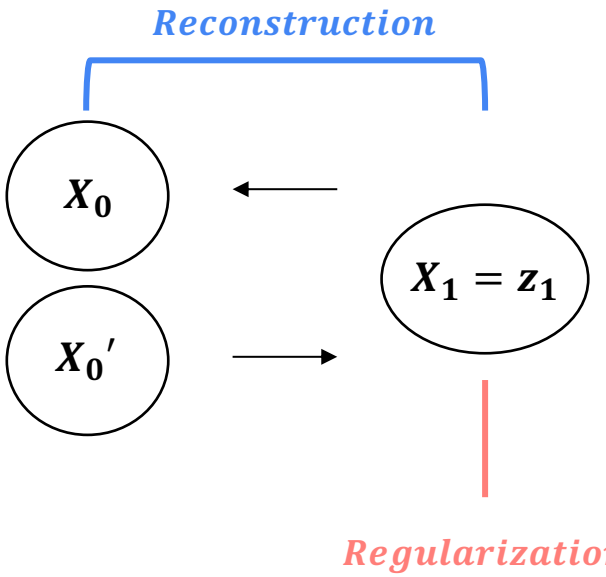
Regularizer on Encoder *Reconstruction on Decoder*

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

Diffusion | Objective for Reverse process

NAVER AI LAB

✓ VAE



Maximize $P_\theta(x)$

Variational autoencoder

$$\mathbb{E}_{x_T \sim q(x_T|x_0)}[-\log p_\theta(x_0)]$$

$$= \int_{-\infty}^{\infty} (-\log p_\theta(x_0)) \cdot q(x_T|x_0) dx_T \quad \because \text{definition of expectation}$$

$$= \int_{-\infty}^{\infty} \left(-\log \frac{p_\theta(x_0, x_T)}{p_\theta(x_T|x_0)}\right) \cdot q(x_T|x_0) dx_T \quad \because \text{bayes rule}, p_\theta(x_T|x_0) = \frac{p_\theta(x_T, x_0)}{p_\theta(x_0)}$$

$$= \int_{-\infty}^{\infty} (-\log \frac{p_\theta(x_0, x_T)}{p_\theta(x_T|x_0)} \cdot \frac{q(x_T|x_0)}{q(x_T|x_0)}) \cdot q(x_T|x_0) dx_T$$

$$\leq \int_{-\infty}^{\infty} \left(-\log \frac{p_\theta(x_0, x_T)}{q(x_T|x_0)}\right) \cdot q(x_T|x_0) dx_T \quad \because KL\ divergence > 0, \text{"ELBO"}$$

$$= \int_{-\infty}^{\infty} \left(-\log \frac{p_\theta(x_0|x_T) \cdot p_\theta(x_T)}{q(x_T|x_0)}\right) \cdot q(x_T|x_0) dx_T \quad \because bayes\ rule, p_\theta(x_0, x_T) = p_\theta(x_0|x_T) p_\theta(x_T)$$

$$= \int_{-\infty}^{\infty} (-\log p_\theta(x_0|x_T)) \cdot q(x_T|x_0) dx_T + \int_{-\infty}^{\infty} (-\log \frac{p_\theta(x_T)}{q(x_T|x_0)}) \cdot q(x_T|x_0) dx_T \quad \because separate\ \log$$

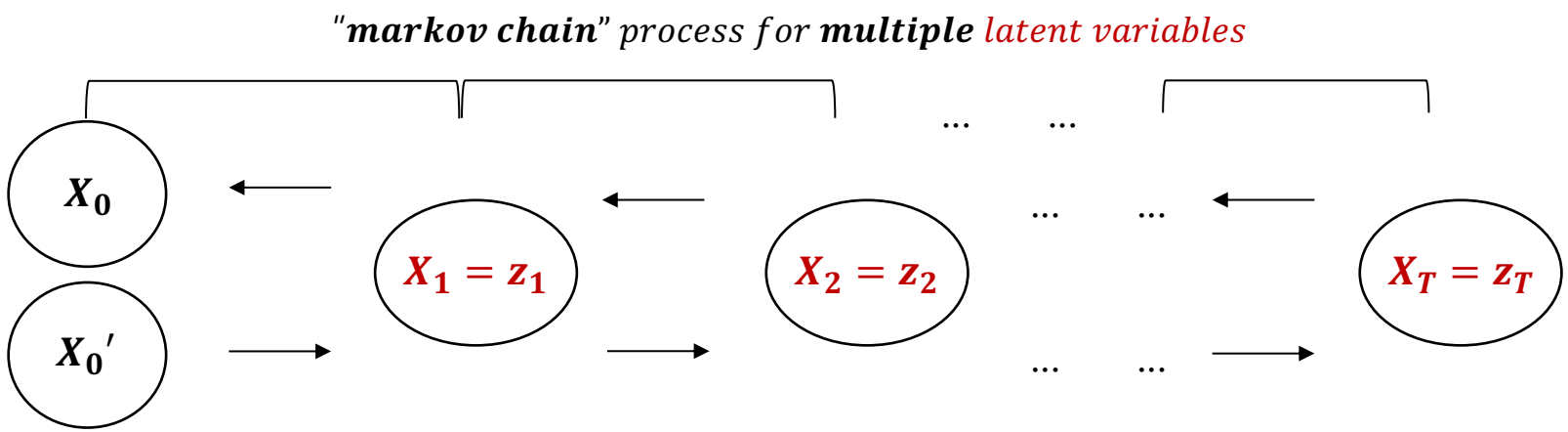
$$= \mathbb{E}_{x_T \sim q(x_T|x_0)}[-\log p_\theta(x_0|x_T)] + \mathbb{E}_{x_T \sim q(x_T|x_0)}[-\log \frac{p_\theta(x_T)}{q(x_T|x_0)}] \quad \because definition\ of\ expectation$$

Regularizer on Encoder *Reconstruction on Decoder*

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

NAVER AI LAB

- ✓ **Diffusion**



NAVER AI LAB

Maximize $P_\theta(x)$

✓ **Diffusion**

$$\mathbb{E}_{x_T\sim q(x_T|x_0)}[-\log p_\theta(x_0)]$$

① $= \mathbb{E}_{x_T\sim q(x_T|x_0)}\left[-\log \frac{p_\theta(x_0,x_1,x_2,\dots,x_T)}{p_\theta(x_1,x_2,x_3,\dots,x_T|x_0)}\right]$ ∵ bayes rule, $p_\theta(x_T|x_0)=\frac{p_\theta(x_T,x_0)}{p_\theta(x_0)}$

② $= \mathbb{E}_{x_T\sim q(x_T|x_0)}\left[-\log \frac{p_\theta(x_0,x_1,x_2,\dots,x_T)}{p_\theta(x_1,x_2,x_3,\dots,x_T|x_0)}\cdot\frac{q(x_{1:T}|x_0)}{q(x_{1:T}|x_0)}\right]$

③ $\leq \mathbb{E}_{x_T\sim q(x_T|x_0)}\left[-\log \frac{p_\theta(x_0,x_1,x_2,\dots,x_T)}{q(x_{1:T}|x_0)}\right]$ ∵ *KL divergence* > 0, "**ELBO**"

④ $= \mathbb{E}_{x_T\sim q(x_T|x_0)}\left[-\log \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right]$ ∵ *Notation*

⑤ $= \mathbb{E}_{x_T\sim q(x_T|x_0)}\left[-\log \frac{p_\theta(\boldsymbol{x_T})\prod_{t=1}^{T}p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t})}{\prod_{t=1}^{T}q(\boldsymbol{x_t}|\boldsymbol{x_{t-1}})}\right]$ ∵ *Below Markov chain property*

⑥ $= \mathbb{E}_{x_{1:T}\sim q(x_{1:T}|x_0)}\left[-\log p_\theta(\boldsymbol{x_T})-\sum_{t=1}^{T}\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right]$ ∵ *separating to summation in logarithm*

$$p_\theta(\boldsymbol{x_{0:T}}) \coloneqq p_\theta(\boldsymbol{x_T})\prod_{t=1}^{T}p_\theta(\boldsymbol{x_{t-1}}|\boldsymbol{x_t}) \qquad q(\boldsymbol{x_{1:T}}|\boldsymbol{x_0}) \coloneqq \prod_{t=1}^{T}q(\boldsymbol{x_t}|\boldsymbol{x_{t-1}})$$

$$\mathbb{E}_{x_{1:T}\sim q(x_{1:T}|x_0)}[-\log p_\theta(x_0)]$$

⑦ $\leq \mathbb{E}_{x_{1:T}\sim q(x_{1:T}|x_0)}\left[-\log p_\theta(\boldsymbol{x_T})-\sum_{t=1}^{T}\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right]$

⑧ $= \mathbb{E}_{x_{1:T}\sim q(x_{1:T}|x_0)}\left[-\log p_\theta(\boldsymbol{x_T})-\sum_{t=2}^{T}\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}-\log\frac{p_\theta(x_0|x_1)}{q(x_1|x_0)}\right]$

⑨ $= \mathbb{E}_{x_{1:T}\sim q(x_{1:T}|x_0)}\left[-\log p_\theta(\boldsymbol{x_T})-\sum_{t=2}^{T}\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t,x_0)}\cdot\frac{q(x_{t-1}|x_0)}{q(x_t|x_0)}-\log\frac{p_\theta(x_0|x_1)}{q(x_1|x_0)}\right]$ ∵*

⑩ $= \mathbb{E}_{x_{1:T}\sim q(x_{1:T}|x_0)}\left[-\log p_\theta(\boldsymbol{x_T})-\sum_{t=2}^{T}\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t,x_0)}-\sum_{t=2}^{T}\log\frac{q(x_{t-1}|x_0)}{q(x_t|x_0)}-\log\frac{p_\theta(x_0|x_1)}{q(x_1|x_0)}\right]$

⑪ $= \mathbb{E}_{x_{1:T}\sim q(x_{1:T}|x_0)}\left[-\log p_\theta(\boldsymbol{x_T})-\sum_{t=2}^{T}\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t,x_0)}-\log\frac{q(x_1|x_0)}{q(x_T|x_0)}-\log\frac{p_\theta(x_0|x_1)}{q(x_1|x_0)}\right]$

⑫ $= \mathbb{E}_{x_{1:T}\sim q(x_{1:T}|x_0)}\left[-\log\frac{p_\theta(\boldsymbol{x_T})}{q(x_T|x_0)}-\sum_{t=2}^{T}\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t,x_0)}-\log p_\theta(x_0|x_1)\right]$

* $q(x_t|x_{t-1})$

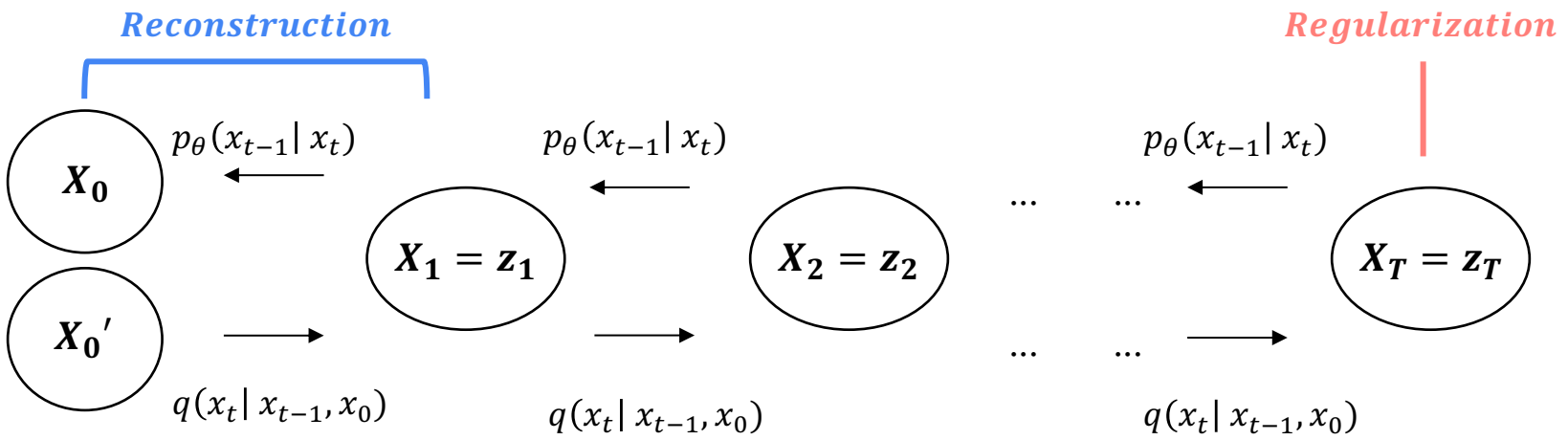
$= q(x_t|x_{t-1},x_0)$ ∵ Markov chain property

$= \frac{q(x_t,x_{t-1},x_0)}{q(x_{t-1},x_0)}$ ∵ bayes rule

$= \frac{q(x_{t-1},x_t,x_0)}{q(x_{t-1},x_0)}\cdot\frac{q(x_t,x_0)}{q(x_t,x_0)}$

$= q(x_{t-1}|x_t,x_0)\cdot\frac{q(x_t,x_0)}{q(x_{t-1},x_0)}$

✓ **Diffusion**

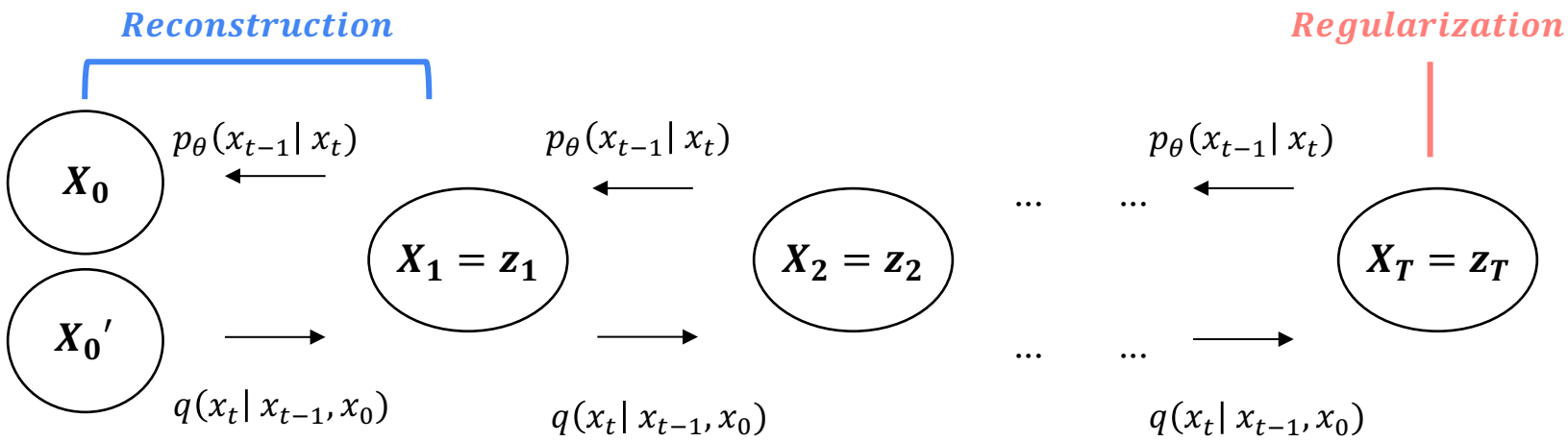


Regularization Reconstruction

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

NAVER AI LAB

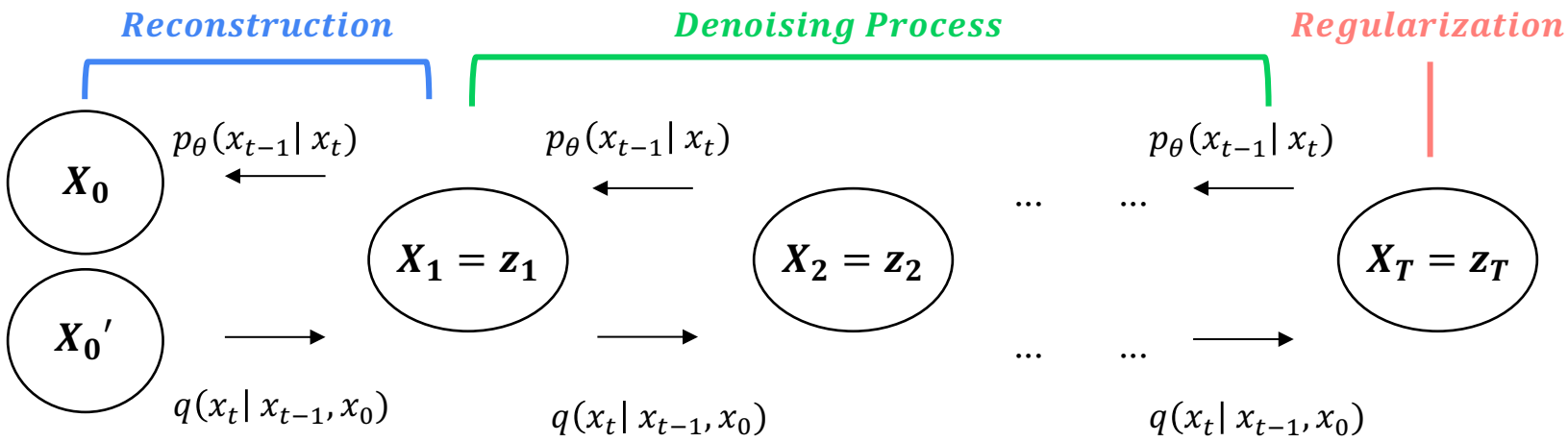
✓ **Diffusion**



$$Loss_{VAE} = \underbrace{D_{KL}\big(q(z|x) \parallel p_\theta(z)\big)}_{\text{Regularization}} - \underbrace{E_{z\sim q(z|x)} \log P_\theta(x|z)}_{\text{Reconstruction}}$$

$$Loss_{Diffusion} = \underbrace{D_{KL}\big(q(x_T|x_0) \parallel p_\theta(x_T)\big)}_{\text{Regularization}} + \sum_{t=2} D_{KL}\big(q(x_{t-1}|x_t, x_0) \parallel P_\theta(x_{t-1}|x_t)\big) - \underbrace{E_q \log P_\theta(x_0|x_1)}_{\text{Reconstruction}}$$

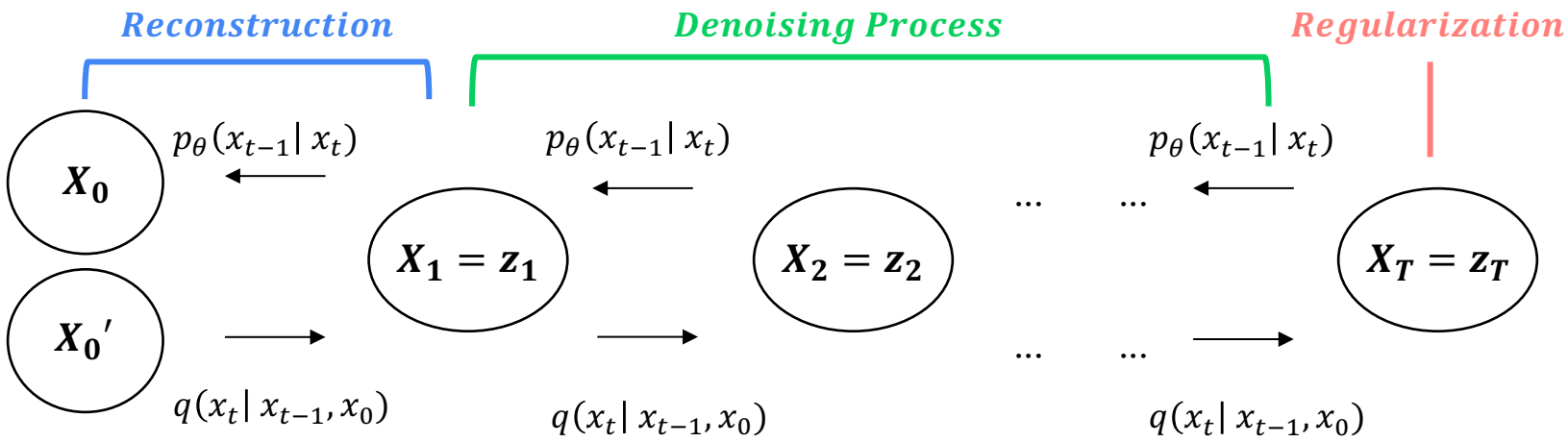
✓ **Diffusion**



$$Loss_{VAE} = \underbrace{D_{KL}\big(q(z|x) \parallel p_\theta(z)\big)}_{\text{Regularization}} - \underbrace{E_{z\sim q(z|x)} \log P_\theta(x|z)}_{\text{Reconstruction}}$$

$$Loss_{Diffusion} = \underbrace{D_{KL}\big(q(x_T|x_0) \parallel p_\theta(x_T)\big)}_{\text{Regularization}} + \underbrace{\sum_{t=2} D_{KL}\big(q(x_{t-1}|x_t, x_0) \parallel P_\theta(x_{t-1}|x_t)\big)}_{\text{Denoising Process}} - \underbrace{E_q \log P_\theta(x_0|x_1)}_{\text{Reconstruction}}$$

✓ **Diffusion**



Regularization Reconstruction

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

Denoising Process

$$Loss_{Diffusion} = D_{KL}\big(q(x_T|x_0) \parallel p_\theta(x_T)\big) + \sum_{t=2} D_{KL}\big(q(x_{t-1}|x_t, x_0) \parallel P_\theta(x_{t-1}|x_t)\big) - E_q \log P_\theta(x_0|x_1)$$

www.youtube.com/watch?v=_JQSMhqXw-4

NAVER AI LAB

DDPM

Denoising diffusion probabilistic models
NeurIPS 2020

- Overview
  - $q(X_{t-1} \mid X_t) \approx p_\theta(X_{t-1} \mid X_t)$
    - $N\left(X_{t-1};\ \mu_\theta(X_t, t),\ \Sigma_\theta(X_t, t)\right)$

DDPM | Reverse process

NAVER AI LAB

- Overview
  - $q(X_{t-1} \mid X_t) \approx p_\theta(X_{t-1} \mid X_t)$
    - $N\big(X_{t-1};\ \mu_\theta(X_t, t),\ \Sigma_\theta(X_t, t)\big)$
  - $q(X_{t-1} \mid X_t) \ \rightarrow q(X_{t-1} \mid X_t, X_0)$
    - $N\left(X_{t-1};\ \tilde{\mu}(X_t, X_0), \tilde{\Sigma}(X_t, X_0)\right)$

DDPM | Reverse process

NAVER AI LAB

- Overview
  - $q(X_{t-1} \mid X_t) \approx p_\theta(X_{t-1} \mid X_t)$
    - $N\left(X_{t-1};\ \mu_\theta(X_t, t),\ \Sigma_\theta(X_t, t)\right)$
  - $q(X_{t-1} \mid X_t) \rightarrow q(X_{t-1} \mid X_t, X_0)$
    - $N\left(X_{t-1};\ \tilde{\mu}(X_t, X_0), \tilde{\Sigma}(X_t, X_0)\right)$

Note

$$P(B \mid A) = P(A \mid B)\frac{P(B)}{P(A)}$$

Note

$$q(x_{t-1} \mid x_t, x_0) = q(x_t \mid x_{t-1})\frac{q(x_{t-1} \mid x_0)}{q(x_t \mid x_0)}$$

DDPM | Reverse process

NAVER AI LAB

- Overview
  - $q(X_{t-1} \mid X_t) \approx p_\theta(X_{t-1} \mid X_t)$
    - $N\big(X_{t-1};\ \mu_\theta(X_t, t),\ \Sigma_\theta(X_t, t)\big)$
  - $q(X_{t-1} \mid X_t) \ \rightarrow q(X_{t-1} \mid X_t, X_0)$
    - $N\left(X_{t-1};\ \tilde{\mu}(X_t, X_0), \tilde{\Sigma}(X_t, X_0)\right)$

Note

$$\boldsymbol{q(x_{t-1} \mid x_t, x_0) = q(x_t \mid x_{t-1}) \frac{q(x_{t-1} \mid x_0)}{q(x_t \mid x_0)}}$$

$$q(x_t|x_{t-1}) = \frac{1}{\sqrt{2\pi\beta_t}} \exp\left(-\frac{(x_t - \sqrt{1-\beta_t}x_{t-1})^2}{2\beta_t}\right)$$

$$q(x_t|x_0) = \frac{1}{\sqrt{2\pi(1-\bar{\alpha}_t)}} \exp\left(-\frac{(x_t - \sqrt{\bar{\alpha}_t}x_0)^2}{2(1-\bar{\alpha}_t)}\right)$$

$$q(x_{t-1}|x_0) = \frac{1}{\sqrt{2\pi(1-\bar{\alpha}_{t-1})}} \exp\left(-\frac{(x_{t-1} - \sqrt{\bar{\alpha}_{t-1}}x_0)^2}{2(1-\bar{\alpha}_{t-1})}\right)$$

Note

$$\boldsymbol{P(B \mid A) = P(A \mid B) \frac{P(B)}{P(A)}}$$

$$\boldsymbol{N\big(x;\ \mu, \sigma^2\big) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}}$$

- $q(x_t \mid x_{t-1}) = N\big(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t * I\big)$
- $q(x_t \mid x_0) = N\big(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t) * I\big)$
  - $\alpha_t = 1 - \beta_t$
  - $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$

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

DDPM | Reverse process

NAVER AI LAB

- Overview
  - $q(X_{t-1} \mid X_t) \approx p_\theta(X_{t-1} \mid X_t)$
    - $N\big(X_{t-1};\ \mu_\theta(X_t, t),\ \Sigma_\theta(X_t, t)\big)$
  - $q(X_{t-1} \mid X_t) \ \rightarrow q(X_{t-1} \mid X_t, X_0)$
    - $N\left(X_{t-1};\ \tilde{\mu}(X_t, X_0), \tilde{\Sigma}(X_t, X_0)\right)$

Note

$$\boldsymbol{P(B \mid A) = P(A \mid B)\frac{P(B)}{P(A)}}$$

$$\boldsymbol{N\big(x;\ \mu, \sigma^2\big) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}}$$

- $q(x_t \mid x_{t-1}) = N\big(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t * I\big)$
- $q(x_t \mid x_0) = N\big(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t) * I\big)$
  - $\alpha_t = 1 - \beta_t$
  - $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$

Note

$$\begin{aligned}
\therefore q(x_{t-1}|x_t, x_0) &= \frac{1}{\sqrt{2\pi\beta_t(\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t})}} \exp\left(-\frac{(x_t - \sqrt{1-\beta_t}x_{t-1})^2}{2\beta_t} - \frac{(x_{t-1} - \sqrt{\bar{\alpha}_{t-1}}x_0)^2}{2(1-\bar{\alpha}_{t-1})} + \frac{(x_t - \sqrt{\bar{\alpha}_t}x_0)^2}{2(1-\bar{\alpha}_t)}\right) \\
&= \frac{1}{\sqrt{2\pi\beta_t(\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t})}} \exp\left(-\left(\left[\frac{1}{2(1-\bar{\alpha}_{t-1})} + \frac{1-\beta_t}{2\beta_t}\right]x_{t-1}^2 - \left[\frac{2\sqrt{1-\beta_t}}{2\beta_t}x_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}}{2(1-\bar{\alpha}_{t-1})}x_0\right]x_{t-1} + C\right)\right) \\
&= \frac{1}{\sqrt{2\pi\beta_t(\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t})}} \exp\left(-\frac{1}{2\beta_t(\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t})}\left[x_{t-1}^2 - \left(\frac{2\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\alpha_t}x_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0\right)x_{t-1} + C\right]\right) \\
&\approx \frac{1}{\sqrt{2\pi\beta_t(\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t})}} \exp\left(-\frac{1}{2\beta_t(\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t})}\left[x_{t-1} - \left(\frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_{t-1}}x_t\right)\right]^2\right)
\end{aligned}$$

DDPM | Reverse process

NAVER AI LAB

- Overview
  - $q(X_{t-1} \mid X_t) \approx p_\theta(X_{t-1} \mid X_t)$
    - $N\big(X_{t-1};\ \mu_\theta(X_t, t),\ \Sigma_\theta(X_t, t)\big)$
  - $q(X_{t-1} \mid X_t) \ \rightarrow q(X_{t-1} \mid X_t, X_0)$
    - $N\left(X_{t-1};\ \tilde{\mu}(X_t, X_0), \tilde{\Sigma}(X_t, X_0)\right)$
    - $N\big(X_{t-1}; \boldsymbol{\mu}, \boldsymbol{\sigma^2}\big)$

Note

$$\boldsymbol{P(B \mid A) = P(A \mid B)\frac{P(B)}{P(A)}}$$

$$\boldsymbol{N\big(x;\ \mu, \sigma^2\big) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}}$$

- $q(x_t \mid x_{t-1}) = N\big(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t * I\big)$
- $q(x_t \mid x_0) = N\big(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t) * I\big)$
  - $\alpha_t = 1 - \beta_t$
  - $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$

Note

$$\therefore q(x_{t-1}|x_t, x_0) = \frac{1}{\sqrt{2\pi\beta_t(\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t})}} \exp\left(-\frac{(x_t - \sqrt{1-\beta_t}x_{t-1})^2}{2\beta_t} - \frac{(x_{t-1} - \sqrt{\bar{\alpha}_{t-1}}x_0)^2}{2(1-\bar{\alpha}_{t-1})} + \frac{(x_t - \sqrt{\bar{\alpha}_t}x_0)^2}{2(1-\bar{\alpha}_t)}\right)$$

$$= \frac{1}{\sqrt{2\pi\beta_t(\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t})}} \exp\left(-\left(\left[\frac{1}{2(1-\bar{\alpha}_{t-1})} + \frac{1-\beta_t}{2\beta_t}\right]x_{t-1}^2 - \left[\frac{2\sqrt{1-\beta_t}}{2\beta_t}x_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}}{2(1-\bar{\alpha}_{t-1})}x_0\right]x_{t-1} + C\right)\right)$$

$$= \frac{1}{\sqrt{2\pi\beta_t(\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t})}} \exp\left(-\frac{1}{2\beta_t(\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t})}\left[x_{t-1}^2 - \left(\frac{2\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\alpha_t}x_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0\right)x_{t-1} + C\right]\right)$$

$$\approx \frac{1}{\sqrt{2\pi\beta_t(\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t})}} \exp\left(-\frac{1}{2\beta_t(\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t})}\left[x_{t-1} - \left(\frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_{t-1}}x_t\right)\right]^2\right)$$

$\boldsymbol{\sigma^2}$ $\qquad$ $\boldsymbol{\mu}$

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

DDPM | Training

NAVER AI LAB

- Overview
  - $q(X_{t-1} \mid X_t) \rightarrow q(X_{t-1} \mid X_t, X_0)$
    - $N\left(X_{t-1};\ \tilde{\mu}(X_t, X_0), \tilde{\Sigma}(X_t, X_0)\right)$
    - $N(X_{t-1}; \boldsymbol{\mu}, \boldsymbol{\sigma^2}) = N\left(X_{t-1}; \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_{t-1}} x_t\ , \beta_t \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\right)$

DDPM | Training

NAVER AI LAB

- Overview
  - $q(X_{t-1} \mid X_t) \rightarrow q(X_{t-1} \mid X_t, X_0)$
    - $N\left(X_{t-1};\ \tilde{\mu}(X_t, X_0), \tilde{\Sigma}(X_t, X_0)\right)$
    - $N\left(X_{t-1}; \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_{t-1}} x_t\ , \beta_t \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\right)$
  - $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$ Forward process

Note

- $q(x_t \mid x_{t-1}) = N(x_t; \sqrt{1-\beta_t} x_{t-1}, \beta_t * I)$
- $q(x_t \mid x_0) = N(x_t; \sqrt{\bar{\alpha}_t} x_0, (1-\bar{\alpha}_t) * I)$
  - $\alpha_t = 1 - \beta_t$
  - $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$

DDPM | Training

NAVER AI LAB

- Overview
  - $q(X_{t-1} \mid X_t) \rightarrow q(X_{t-1} \mid X_t, X_0)$
    - $N\left(X_{t-1};\ \tilde{\mu}(X_t, X_0), \tilde{\Sigma}(X_t, X_0)\right)$
    - $N\left(X_{t-1}; \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_{t-1}} x_t, \beta_t \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\right)$
  - $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$ Forward process

Note

- $q(x_t \mid x_{t-1}) = N(x_t; \sqrt{1-\beta_t} x_{t-1}, \beta_t * I)$
- $q(x_t \mid x_0) = N(x_t; \sqrt{\bar{\alpha}_t} x_0, (1-\bar{\alpha}_t) * I)$
  - $\alpha_t = 1 - \beta_t$
  - $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$

Note

$$\tilde{\mu}_t = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \sqrt{1-\bar{\alpha}_t}\epsilon) + \frac{\sqrt{\bar{\alpha}_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x_t$$

$$= \left(\frac{\beta_t}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}} + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\right) x_t - \frac{\sqrt{1-\bar{\alpha}_t}\beta_t}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}}\epsilon$$

$$= \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right)$$

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

DDPM | Training

NAVER AI LAB

- Overview
  - $q(X_{t-1} \mid X_t) \rightarrow q(X_{t-1} \mid X_t, X_0)$
    - $N\left(X_{t-1};\ \tilde{\mu}(X_t, X_0), \tilde{\Sigma}(X_t, X_0)\right)$
    - $N\left(X_{t-1}; \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_{t-1}} x_t, \beta_t \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\right)$
  - $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$ Forward process

Note

- $q(x_t \mid x_{t-1}) = N(x_t; \sqrt{1-\beta_t} x_{t-1}, \beta_t * I)$
- $q(x_t \mid x_0) = N(x_t; \sqrt{\bar{\alpha}_t} x_0, (1-\bar{\alpha}_t) * I)$
  - $\alpha_t = 1 - \beta_t$
  - $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$

Note

$$\tilde{\mu}_t = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\frac{1}{\sqrt{\bar{\alpha}_t}}(x_t - \sqrt{1-\bar{\alpha}_t}\epsilon) + \frac{\sqrt{\bar{\alpha}_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} x_t$$

$$= \left(\frac{\beta_t}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}} + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\right) x_t - \frac{\sqrt{1-\bar{\alpha}_t}\beta_t}{(1-\bar{\alpha}_t)\sqrt{\alpha_t}}\epsilon$$

$$= \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right)$$

$$\mu_\theta = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta\right)$$

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

DDPM | Training

NAVER AI LAB

- Overview
  - $q(X_{t-1} \mid X_t) \rightarrow q(X_{t-1} \mid X_t, X_0)$
    - $N\left(X_{t-1};\ \tilde{\mu}(X_t, X_0), \tilde{\Sigma}(X_t, X_0)\right)$
    - $N\left(X_{t-1}; \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_{t-1}} x_t\ , \beta_t \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\right)$
    - $N\left(X_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon\right), \tilde{\beta}_t\right)$

Note

- $q(x_t \mid x_{t-1}) = N(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t * I)$
- $q(x_t \mid x_0) = N(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t) * I)$
  - $\alpha_t = 1 - \beta_t$
  - $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$

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

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

DDPM | Training

NAVER AI LAB

- Overview
  - $q(X_{t-1} \mid X_t) \rightarrow q(X_{t-1} \mid X_t, X_0)$
    - $N\left(X_{t-1};\ \tilde{\mu}(X_t, X_0), \tilde{\Sigma}(X_t, X_0)\right)$
    - $N\left(X_{t-1}; \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_{t-1}} x_t\ , \beta_t \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\right)$
    - $N\left(X_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon\right), \tilde{\beta}_t\right)$
    - $N\left(X_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t)\right), \tilde{\beta}_t\right)$

Note

- $q(x_t \mid x_{t-1}) = N\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t * I\right)$
- $q(x_t \mid x_0) = N\left(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t) * I\right)$
  - $\alpha_t = 1 - \beta_t$
  - $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$

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

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

DDPM | Training

NAVER AI LAB

- Overview
  - $q(X_{t-1} \mid X_t) \rightarrow q(X_{t-1} \mid X_t, X_0)$
    - $N\left(X_{t-1};\ \tilde{\mu}(X_t, X_0), \tilde{\Sigma}(X_t, X_0)\right)$
    - $N\left(X_{t-1}; \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_{t-1}} x_t\ , \beta_t \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\right)$
    - $N\left(X_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon\right), \tilde{\beta}_t\right)$
    - $N\left(X_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t)\right), \tilde{\beta}_t\right)$

Note

$$\boldsymbol{Loss = \varepsilon - \varepsilon_\theta(x_t)}$$

Note

- $q(x_t \mid x_{t-1}) = N\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t * I\right)$
- $q(x_t \mid x_0) = N\left(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t) * I\right)$
  - $\alpha_t = 1 - \beta_t$
  - $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$

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

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

DDPM | Training

NAVER AI LAB

- Overview
  - $q(X_{t-1} \mid X_t) \rightarrow q(X_{t-1} \mid X_t, X_0)$
    - $N\left(X_{t-1};\ \tilde{\mu}(X_t, X_0), \tilde{\Sigma}(X_t, X_0)\right)$
    - $N\left(X_{t-1}; \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_{t-1}} x_t\ , \beta_t \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\right)$
    - $N\left(X_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon\right), \tilde{\beta}_t\right)$
    - $N\left(X_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t)\right), \tilde{\beta}_t\right)$

Note

- $q(x_t \mid x_{t-1}) = N\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t * I\right)$
- $q(x_t \mid x_0) = N\left(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t) * I\right)$
  - $\alpha_t = 1 - \beta_t$
  - $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$

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

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

Note

$$\boldsymbol{Loss = \varepsilon - \varepsilon_\theta(x_t)}$$

$$Loss_{Diffusion} = \underbrace{D_{KL}\left(q(x_T|x_0) \parallel p_\theta(x_T)\right)}_{\textit{Regularization}} + \underbrace{\sum_{t=2} D_{KL}\left(q(x_{t-1}|x_t, x_0) \parallel P_\theta(x_{t-1}|x_t)\right)}_{\textit{Denoising Process}} - \underbrace{E_q \log P_\theta(x_0|x_1)}_{\textit{Reconstruction}}$$

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

DDPM | Training

NAVER AI LAB

- Overview
  - $q(X_{t-1} \mid X_t) \rightarrow q(X_{t-1} \mid X_t, X_0)$
    - $N\left(X_{t-1};\ \tilde{\mu}(X_t, X_0), \tilde{\Sigma}(X_t, X_0)\right)$
    - $N\left(X_{t-1}; \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_{t-1}} x_t\ , \beta_t \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\right)$
    - $N\left(X_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon\right), \tilde{\beta}_t\right)$
    - $N\left(X_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t)\right), \tilde{\beta}_t\right)$

Note

$$\boldsymbol{Loss} = \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon_\theta}(\boldsymbol{x_t})$$

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

Note

- $q(x_t \mid x_{t-1}) = N\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t * I\right)$
- $q(x_t \mid x_0) = N\left(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t) * I\right)$
  - $\alpha_t = 1 - \beta_t$
  - $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$

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

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

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

DDPM | Training

NAVER AI LAB

- Overview
  - $q(X_{t-1} \mid X_t) \rightarrow q(X_{t-1} \mid X_t, X_0)$
    - $N\left(X_{t-1};\ \tilde{\mu}(X_t, X_0), \tilde{\Sigma}(X_t, X_0)\right)$
    - $N\left(X_{t-1}; \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_{t-1}} x_t\ , \beta_t \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\right)$
    - $N\left(X_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon\right), \tilde{\beta}_t\right)$
    - $N\left(X_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t)\right), \tilde{\beta}_t\right)$

Note

- $q(x_t \mid x_{t-1}) = N\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t * I\right)$
- $q(x_t \mid x_0) = N\left(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t) * I\right)$
  - $\alpha_t = 1 - \beta_t$
  - $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$

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

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

Note

$$\boldsymbol{Loss = \varepsilon - \varepsilon_\theta(x_t)}$$

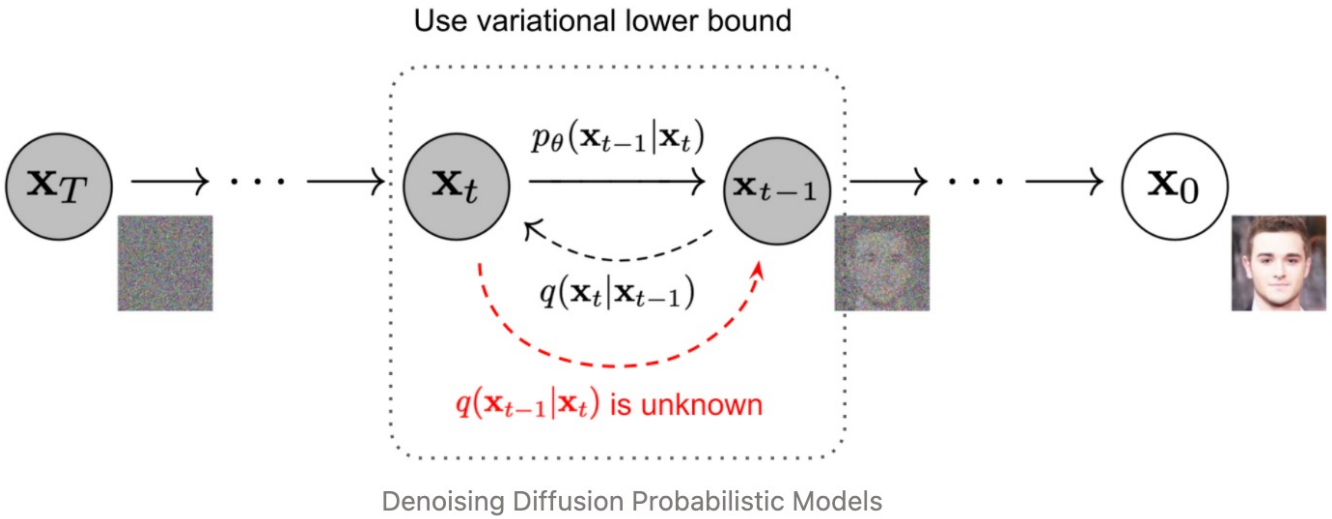
$$\mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t\right)\right\|^2\right]$$

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

t가 커질수록, 값이 작아져서.

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

DDPM | Summary



- $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon,\ \epsilon \sim \mathcal{N}(0, \mathbf{I})$ **(Forward)**
  - $\beta_1 = 10^{-4}, \beta_T = 0.02$
  - $\alpha_t := 1 - \beta_t,\ \bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$
- $\epsilon - \epsilon_\theta(\mathbf{x}_t) = \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon)$ **(Loss)**
  - ϵ_θ = prediction network
- $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right) + \sqrt{\tilde{\beta}_t}\epsilon,\ \epsilon \sim \mathcal{N}(0, \mathbf{I})$ **(Reverse)**
  - $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$
    - $\tilde{\beta}_t = \beta_t$로 해도 성능차이 없음

NAVER AI LAB

DDIM

Denoising diffusion implicit models
ICLR 2021

DDIM | Forward process

NAVER AI LAB

- Overview
  - DDPM
    - $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\varepsilon$ (Forward)
    - $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t)\right) + \sqrt{\tilde{\beta}_t}\varepsilon$ (Reverse)

DDIM | Reverse process

NAVER AI LAB

- Overview
  - DDPM
    - $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\varepsilon$ (Forward)
    - $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t)\right) + \sqrt{\tilde{\beta}_t}\varepsilon$ (Reverse)

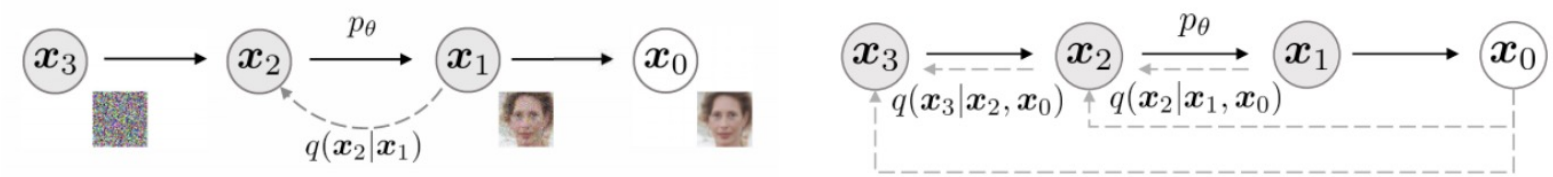


Figure 1: Graphical models for diffusion (left) and non-Markovian (right) inference models.

Denoising Diffusion Implicit Models

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

DDIM | Reverse process

NAVER AI LAB

- Overview
  - DDPM
    - $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$ (Forward)
    - $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t)\right) + \sqrt{\tilde{\beta}_t}\varepsilon$ (Reverse)
  - DDIM
    - $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$ (Forward)
    - $x_{t-1} = \sqrt{\bar{\alpha}_{t-1}} f_\theta(x_t) + \sqrt{1-\bar{\alpha}_{t-1}-\tilde{\beta}_t}\varepsilon_\theta(x_t) + \sqrt{\tilde{\beta}_t}\epsilon$ (Reverse)

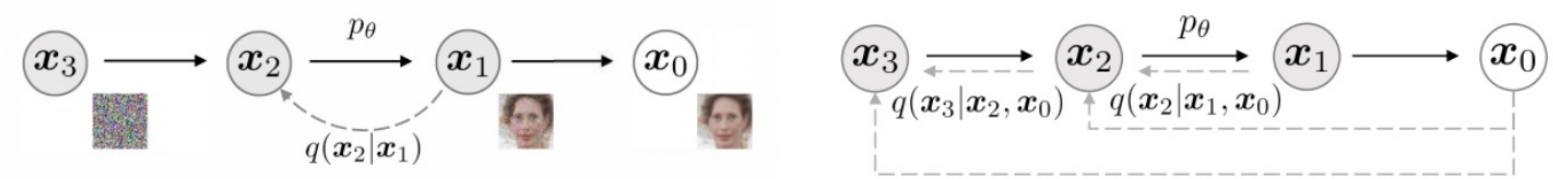


Figure 1: Graphical models for diffusion (left) and non-Markovian (right) inference models.

Denoising Diffusion Implicit Models

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

DDIM | Reverse process

NAVER AI LAB

- Overview
  - DDPM
    - $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$ (Forward)
    - $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t)\right) + \sqrt{\tilde{\beta}_t}\varepsilon$ (Reverse)
  - DDIM
    - $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$ (Forward)
    - $x_{t-1} = \underbrace{\sqrt{\bar{\alpha}_{t-1}} f_\theta(x_t) + \sqrt{1-\bar{\alpha}_{t-1}-\tilde{\beta}_t}\varepsilon_\theta(x_t)}_{\text{mean}} + \underbrace{\sqrt{\tilde{\beta}_t}\epsilon}_{\text{std}}$ (Reverse)

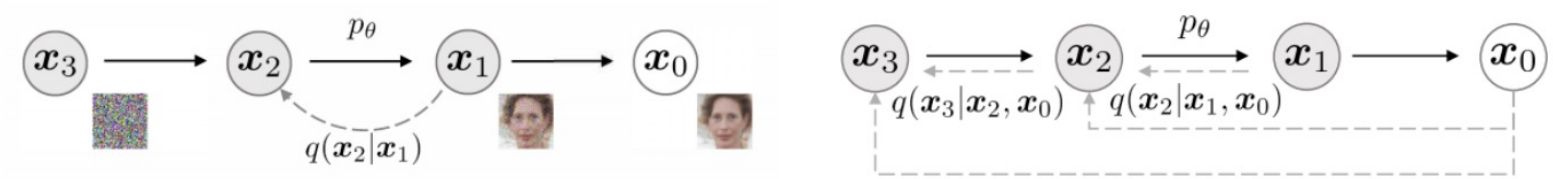


Figure 1: Graphical models for diffusion (left) and non-Markovian (right) inference models.

Denoising Diffusion Implicit Models

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

DDIM | Reverse process

NAVER AI LAB

- Overview
  - DDIM
    - $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\varepsilon$ (Forward)
    - $x_{t-1} = \sqrt{\bar{\alpha}_{t-1}} f_\theta(x_t) + \sqrt{1 - \bar{\alpha}_{t-1} - \tilde{\beta}_t}\varepsilon_\theta(x_t) + \sqrt{\tilde{\beta}_t}\epsilon$ (Reverse)

Note

- DDPM
  - $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\varepsilon$ (Forward)
  - $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t)\right) + \sqrt{\tilde{\beta}_t}\varepsilon$ (Reverse)

$$\tilde{\beta}_t = \sigma_t^2 = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \cdot \beta_t$$

DDIM | Reverse process

NAVER AI LAB

- Overview
  - DDIM
    - $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$ (Forward)
    - $x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}f_\theta(x_t) + \sqrt{1-\bar{\alpha}_{t-1}-\tilde{\beta}_t}\varepsilon_\theta(x_t) + \sqrt{\tilde{\beta}_t}\epsilon$ (Reverse)
      - $= \sqrt{\bar{\alpha}_{t-1}}\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\varepsilon_\theta(x_t) + \sigma_t\epsilon$
      - $N(x_{t-1}; \sqrt{\bar{\alpha}_{t-1}}x_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\frac{x_t-\sqrt{\bar{\alpha}_t}x_0}{\sqrt{1-\bar{\alpha}_t}}, \sigma_t^2)$

Note

- DDPM
  - $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$ (Forward)
  - $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t)\right) + \sqrt{\tilde{\beta}_t}\varepsilon$ (Reverse)

$$\tilde{\beta}_t = \sigma_t^2 = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\cdot\beta_t$$

$$\sigma_t^2 = \eta\cdot\tilde{\beta}_t$$

DDIM | Reverse process

NAVER AI LAB

- Overview
  - DDIM
    - $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$ (Forward)
    - $x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}f_\theta(x_t) + \sqrt{1-\bar{\alpha}_{t-1}-\tilde{\beta}_t}\varepsilon_\theta(x_t) + \sqrt{\tilde{\beta}_t}\epsilon$ (Reverse)
      - $= \sqrt{\bar{\alpha}_{t-1}}\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\varepsilon_\theta(x_t) + \sigma_t\epsilon$
      - $N(x_{t-1}; \sqrt{\bar{\alpha}_{t-1}}x_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\frac{x_t-\sqrt{\bar{\alpha}_t}x_0}{\sqrt{1-\bar{\alpha}_t}}, \sigma_t^2)$

Note

- DDPM
  - $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$ (Forward)
  - $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t)\right) + \sqrt{\tilde{\beta}_t}\varepsilon$ (Reverse)

$$\tilde{\beta}_t = \sigma_t^2 = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\cdot\beta_t$$

$$\sigma_t^2 = \eta\cdot\tilde{\beta}_t$$

Note

- DDPM
  - $N\left(X_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon\right), \tilde{\beta}_t\right)$
  - $N\left(X_{t-1}; \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_{t-1}}x_t, \tilde{\beta}_t\right)$

x_0를 x_t로 정리해서 대입해서 풀었었음

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

DDIM | Reverse process

NAVER AI LAB

- Overview
  - DDIM
    - $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$ (Forward)
    - $x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}f_\theta(x_t) + \sqrt{1-\bar{\alpha}_{t-1}-\tilde{\beta}_t}\varepsilon_\theta(x_t) + \sqrt{\tilde{\beta}_t}\epsilon$ (Reverse)
      - $= \sqrt{\bar{\alpha}_{t-1}}\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\varepsilon_\theta(x_t) + \sigma_t\epsilon$
      - $N(x_{t-1}; \sqrt{\bar{\alpha}_{t-1}}x_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\frac{x_t-\sqrt{\bar{\alpha}_t}x_0}{\sqrt{1-\bar{\alpha}_t}}, \sigma_t^2)$

Note

- DDPM
  - $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$ (Forward)
  - $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t)\right) + \sqrt{\tilde{\beta}_t}\varepsilon$ (Reverse)

$$\tilde{\beta}_t = \sigma_t^2 = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\cdot\beta_t$$

$$\sigma_t^2 = \eta\cdot\tilde{\beta}_t$$

Note

- DDPM
  - $N\left(X_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon\right), \tilde{\beta}_t\right)$
  - $N\left(X_{t-1}; \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_{t-1}}x_t, \tilde{\beta}_t\right)$

x_0를 x_t로 정리해서 대입해서 풀었었음

Use ChatGPT

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

DDIM | Reverse process

NAVER AI LAB

- Overview
  - DDIM
    - $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$ (Forward)
    - $x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}f_\theta(x_t) + \sqrt{1-\bar{\alpha}_{t-1}-\tilde{\beta}_t}\varepsilon_\theta(x_t) + \sqrt{\tilde{\beta}_t}\epsilon$ (Reverse)
      - $= \sqrt{\bar{\alpha}_{t-1}}\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\varepsilon_\theta(x_t) + \sigma_t\epsilon$
      - $N(x_{t-1}; \sqrt{\bar{\alpha}_{t-1}}x_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\frac{x_t-\sqrt{\bar{\alpha}_t}x_0}{\sqrt{1-\bar{\alpha}_t}}, \sigma_t^2)$

Note

- DDPM
  - $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$ (Forward)
  - $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t)\right) + \sqrt{\tilde{\beta}_t}\varepsilon$ (Reverse)

$$\tilde{\beta}_t = \sigma_t^2 = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\cdot\beta_t$$

$$\sigma_t^2 = \eta\cdot\tilde{\beta}_t$$

Note

- DDPM
  - $N\left(X_{t-1}; \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon\right), \tilde{\beta}_t\right)$
  - $N\left(X_{t-1}; \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_{t-1}}x_t, \tilde{\beta}_t\right)$

x_0를 x_t로 정리해서 대입해서 풀었었음

Use ChatGPT

$$\begin{aligned}
\mathbf{x}_{t-1} &= \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}}\boldsymbol{\epsilon}_{t-1} \\
&= \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\boldsymbol{\epsilon}_t + \sigma_t\boldsymbol{\epsilon} \\
&= \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\frac{\mathbf{x}_t-\sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}} + \sigma_t\boldsymbol{\epsilon} \\
q_\sigma(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0) &= \mathcal{N}(\mathbf{x}_{t-1}; \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\frac{\mathbf{x}_t-\sqrt{\bar{\alpha}_t}\mathbf{x}_0}{\sqrt{1-\bar{\alpha}_t}}, \sigma_t^2\mathbf{I})
\end{aligned}$$

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} && \text{;where } \boldsymbol{\epsilon}_{t-1}, \boldsymbol{\epsilon}_{t-2}, \cdots \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\
&= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\bar{\boldsymbol{\epsilon}}_{t-2} && \text{;where } \bar{\boldsymbol{\epsilon}}_{t-2} \text{ merges two Gaussians (*).} \\
&= \cdots \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} \\
q(\mathbf{x}_t|\mathbf{x}_0) &= \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})
\end{aligned}$$

(*) Recall that when we merge two Gaussians with different variance, $\mathcal{N}(\mathbf{0}, \sigma_1^2\mathbf{I})$ and $\mathcal{N}(\mathbf{0}, \sigma_2^2\mathbf{I})$, the new distribution is $\mathcal{N}(\mathbf{0}, (\sigma_1^2+\sigma_2^2)\mathbf{I})$. Here the merged standard deviation is $\sqrt{(1-\alpha_t)+\alpha_t(1-\alpha_{t-1})} = \sqrt{1-\alpha_t\alpha_{t-1}}$.

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

DDIM | Reverse process

NAVER AI LAB

- Overview
  - DDIM
    - $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$ (Forward)
    - $x_{t-1} = \sqrt{\bar{\alpha}_{t-1}} f_\theta(x_t) + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\varepsilon_\theta(x_t) + \sigma_t \epsilon$ (Reverse)

Note

- DDPM
  - $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$ (Forward)
  - $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t)\right) + \sqrt{\tilde{\beta}_t}\varepsilon$ (Reverse)

$$\tilde{\beta}_t = \sigma_t^2 = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t$$

$$\sigma_t^2 = \eta \cdot \tilde{\beta}_t$$

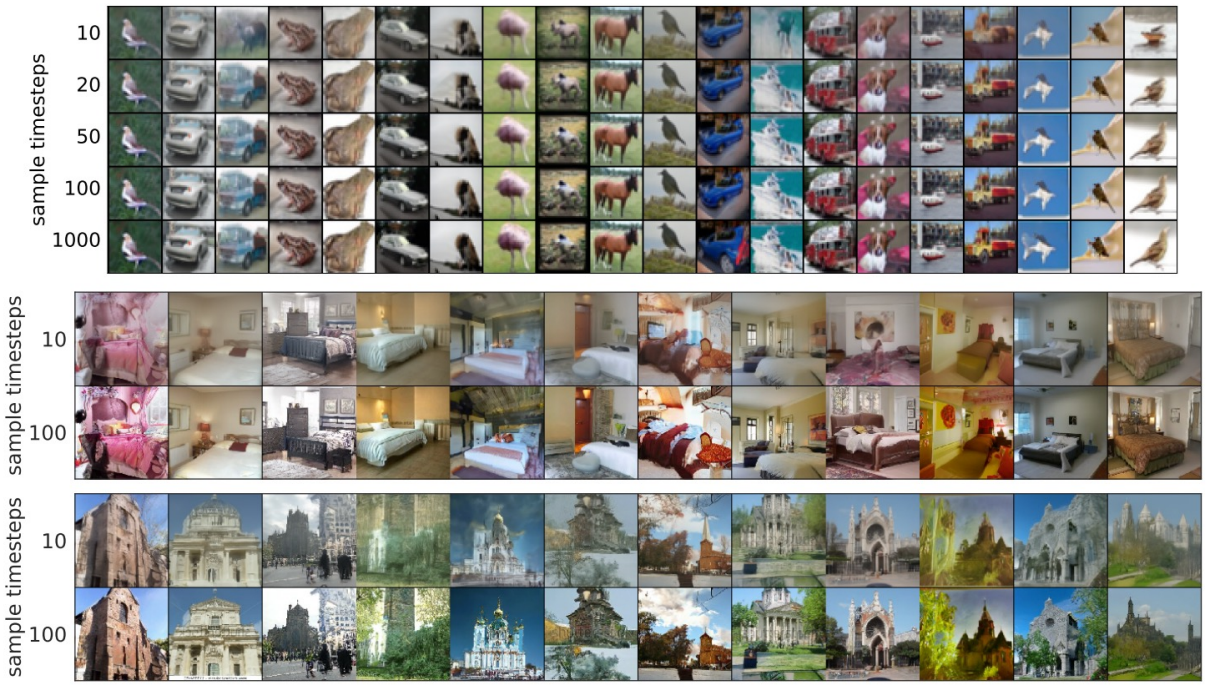


Figure 5: Samples from DDIM with the same random x_T and different number of steps.

DDIM | Reverse process

NAVER AI LAB

- Overview
  - DDIM
    - $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$ (Forward)
    - $x_{t-1} = \sqrt{\bar{\alpha}_{t-1}} f_\theta(x_t) + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\varepsilon_\theta(x_t) + \sigma_t \epsilon$ (Reverse)
    - $x_{t-1} = \sqrt{\bar{\alpha}_{t-1}} f_\theta(x_t) + \sqrt{1-\bar{\alpha}_{t-1}}\varepsilon_\theta(x_t)$

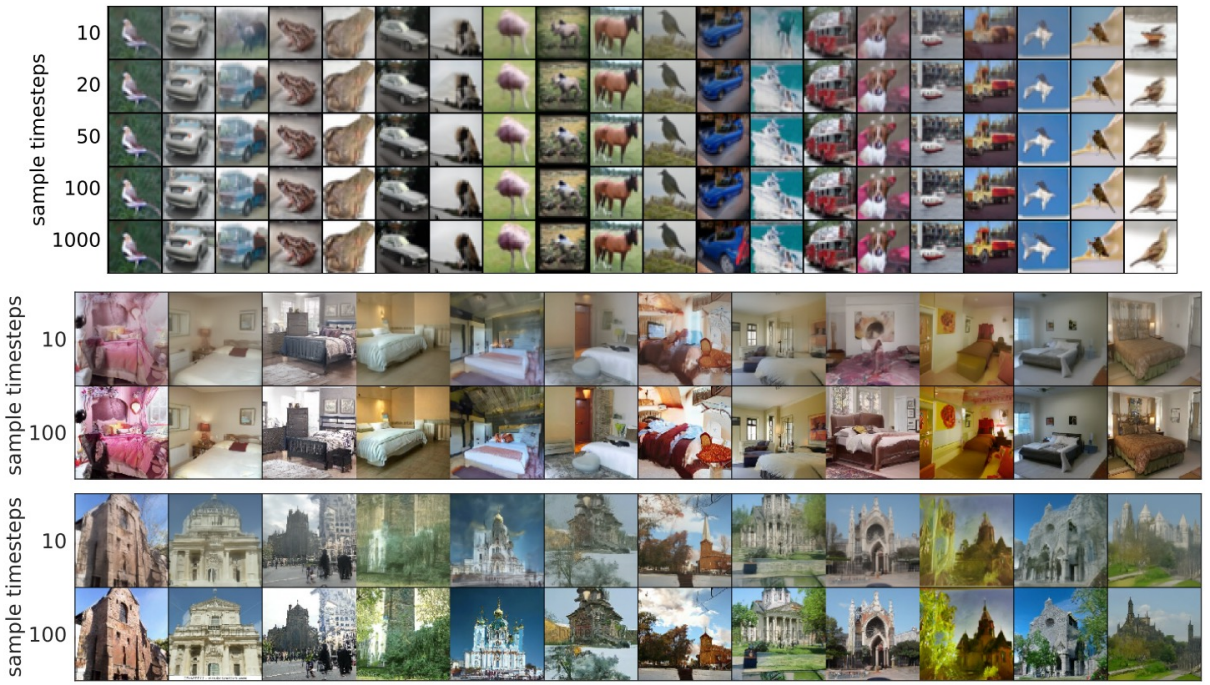


Figure 5: Samples from DDIM with the same random x_T and different number of steps.

Note

- DDPM
  - $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$ (Forward)
  - $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta(x_t)\right) + \sqrt{\tilde{\beta}_t}\varepsilon$ (Reverse)

$$\tilde{\beta}_t = \sigma_t^2 = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \cdot \beta_t$$

$$\sigma_t^2 = \eta \cdot \tilde{\beta}_t$$

Diffusion | Summary

NAVER AI LAB

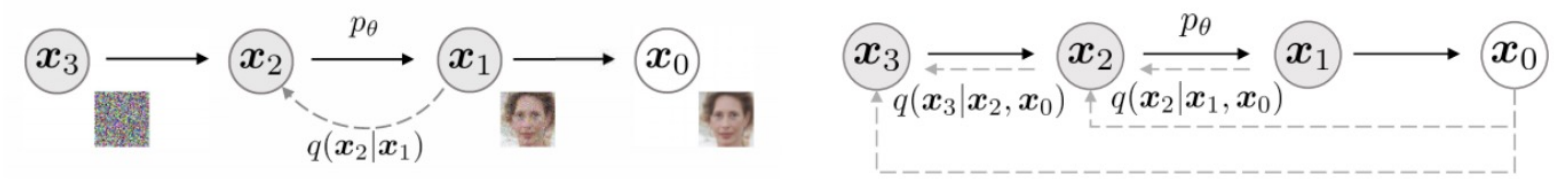


Figure 1: Graphical models for diffusion (left) and non-Markovian (right) inference models.

Denoising Diffusion Implicit Models

DDPM

- $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon,\ \epsilon \sim \mathcal{N}(0, \mathbf{I})$ `(Forward)`
  - $\beta_1 = 10^{-4}, \beta_T = 0.02$
  - $\alpha_t := 1 - \beta_t,\ \bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$
- $\epsilon - \epsilon_\theta(\mathbf{x}_t) = \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon)$ `(Loss)`
  - ϵ_θ = prediction network
- $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right) + \sqrt{\tilde{\beta}_t}\epsilon,\ \epsilon \sim \mathcal{N}(0, \mathbf{I})$ `(Reverse)`
  - $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$
    - $\tilde{\beta}_t = \beta_t$로 해도 성능차이 없음

DDIM

- `In paper,` DDIM α = DDPM $\bar{\alpha}$
- $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$ `(Forward)`
- $\epsilon - \epsilon_\theta(\mathbf{x}_t) = \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon)$ `(Loss)`
  - ϵ_θ = prediction network
- $\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\underbrace{\left(\frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(\mathbf{x}_t)}{\sqrt{\bar{\alpha}_t}}\right)}_{\text{predicted } \mathbf{x}_0 = f_\theta(\mathbf{x}_t)} + \underbrace{\sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\cdot\epsilon_\theta(\mathbf{x}_t)}_{\text{direction pointing to } \mathbf{x}_t} + \underbrace{\sigma_t\epsilon}_{\text{noise}}$

  `(Reverse)`
  - deterministic when $\sigma_t = 0$ → consistency (DDIM)
  - stochastic when $\sigma_t = 1$ → inconsistency (DDPM)

Diffusion | Summary

NAVER AI LAB

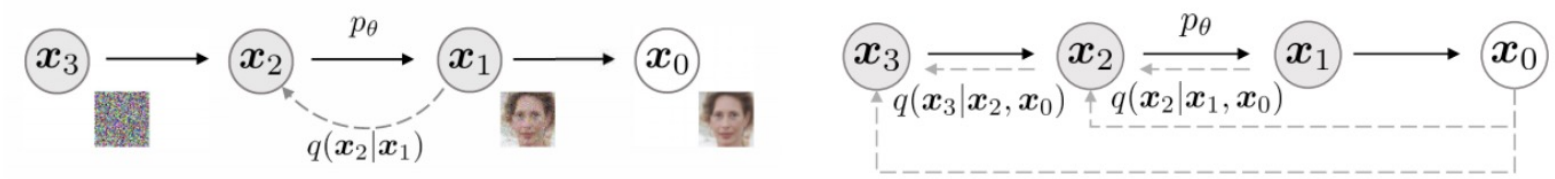


Figure 1: Graphical models for diffusion (left) and non-Markovian (right) inference models.

Denoising Diffusion Implicit Models

최근 Trend

$$x_{t-1} = \hat{\mu}_t(x_t) + \sigma_t \epsilon_t$$

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}} P(f_t(x_t)) + D(f_t(x_t)) + \sigma_t \epsilon_t$$

$$* \; P(f_t(x_t)) = \frac{x_t - \sqrt{1-\bar{\alpha}_t} f_t(x_t)}{\sqrt{\bar{\alpha}_t}}$$

$$* \; D(f_t(x_t)) = \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2} f_t(x_t)$$

DDPM

- $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon,\ \epsilon \sim \mathcal{N}(0, \mathbf{I})$ `(Forward)`
  - $\beta_1 = 10^{-4}, \beta_T = 0.02$
  - $\alpha_t := 1 - \beta_t,\ \bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$
- $\epsilon - \epsilon_\theta(\mathbf{x}_t) = \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon)$ `(Loss)`
  - ϵ_θ = prediction network
- $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right) + \sqrt{\tilde{\beta}_t}\epsilon,\ \epsilon \sim \mathcal{N}(0, \mathbf{I})$ `(Reverse)`
  - $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$
    - $\tilde{\beta}_t = \beta_t$로 해도 성능차이 없음

DDIM

- `In paper,` DDIM α = DDPM $\bar{\alpha}$
- $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$ `(Forward)`
- $\epsilon - \epsilon_\theta(\mathbf{x}_t) = \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon)$ `(Loss)`
  - ϵ_θ = prediction network
- $\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\underbrace{\left(\frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(\mathbf{x}_t)}{\sqrt{\bar{\alpha}_t}}\right)}_{\text{predicted } \mathbf{x}_0 = f_\theta(\mathbf{x}_t)} + \underbrace{\sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\cdot\epsilon_\theta(\mathbf{x}_t)}_{\text{direction pointing to } \mathbf{x}_t} + \underbrace{\sigma_t\epsilon}_{\text{noise}}$

  `(Reverse)`
  - deterministic when $\sigma_t = 0$ → consistency (DDIM)
  - stochastic when $\sigma_t = 1$ → inconsistency (DDPM)

NAVER AI LAB

Diffusion Models Beat GANs on Image Synthesis

NeurIPS 2021

Guided diffusion | Overview

NAVER AI LAB

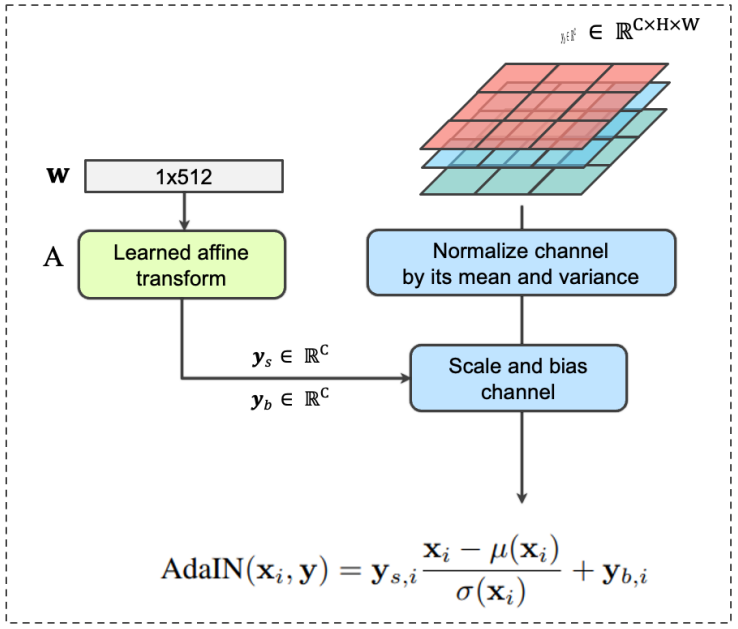
- Overview
  - Architecture improvements
    - Multi head attention
    - Multi resolution attention
    - Adaptive Group Normalization (AdaGN)
  - Truncation trick (fidelity & diversity)
    - Classifier guidance

Which do you like better, coffee or tea? - 문장 타입에 집중하는 어텐션

Which do you like better, coffee or tea? - 명사에 집중하는 어텐션

Which do you like better, coffee or tea? - 관계에 집중하는 어텐션

Which do you like better, coffee or tea? - 감조에 집중하는 어텐션



torch.mean(dim), torch.std(dim)

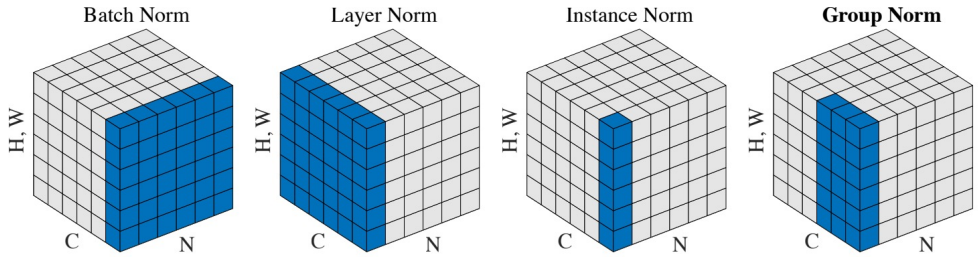


Figure 2. **Normalization methods**. Each subplot shows a feature map tensor, with N as the batch axis, C as the channel axis, and (H, W) as the spatial axes. The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels.

[0] [1, 2, 3] [2, 3]

https://www.blossominkyung.com/deeplearning/transformer-mha

Guided diffusion | Classifier guidance

NAVER AI LAB

- Overview
  - Truncation trick
    - Classifier guidance
      - $x_{t-1} \leftarrow N(\mu + s\Sigma\nabla \log p_\phi(y|x_t), \Sigma)$

Note

BigGAN

- Training = Gaussian normal distribution
- Inference = Truncated gaussian normal distribution

$\psi = 2$ $\psi = 1$ $\psi = 0.5$ $\psi = 0.04$

StyleGAN

- $w' = \bar{w} + \psi(w - \bar{w})$
  - $\bar{w} = \mathbf{mean}(w_1, \ldots, w_{4096})$
  - $\psi = 0 \rightarrow \mathbf{mean}\ of\ latent\ codes$
  - $\psi = 1 \rightarrow latent\ code$

$\psi = 1$ $\psi = 0.7$ $\psi = 0.5$ $\psi = 0$ $\psi = -0.5$ $\psi = -1$

Guided diffusion | Classifier guidance

NAVER AI LAB

- Overview
  - Truncation trick
    - Classifier guidance
      - $x_{t-1} \leftarrow N\left(\mu + s\Sigma\nabla \log p_\phi(y|x_t), \Sigma\right)$

Note

Note

$$p_{\theta,\phi}(x_t|x_{t+1}, y) = Z p_\theta(x_t|x_{t+1}) p_\phi(y|x_t)$$

Guided diffusion | Classifier guidance

NAVER AI LAB

- Overview
  - Truncation trick
    - Classifier guidance
      - $x_{t-1} \leftarrow N(\mu + s\Sigma\nabla \log p_\phi(y|x_t), \Sigma)$

Note

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Note

$$p_{\theta,\phi}(x_t|x_{t+1}, y) = Z p_\theta(x_t|x_{t+1}) p_\phi(y|x_t)$$

$$p_\theta(x_t|x_{t+1}) = \mathcal{N}(\mu, \Sigma)$$

$$\log p_\theta(x_t|x_{t+1}) = -\frac{1}{2}(x_t - \mu)^T \Sigma^{-1}(x_t - \mu) + C$$

Guided diffusion | Classifier guidance

NAVER AI LAB

- Overview
  - Truncation trick
    - Classifier guidance
      - $x_{t-1} \leftarrow N(\mu + s\Sigma\nabla \log p_\phi(y|x_t), \Sigma)$

Note

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$T_f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n = f(a) + f'(a)(x-a)$$

Note

$$p_{\theta,\phi}(x_t|x_{t+1}, y) = Z p_\theta(x_t|x_{t+1}) p_\phi(y|x_t)$$

$$p_\theta(x_t|x_{t+1}) = \mathcal{N}(\mu, \Sigma)$$

$$\log p_\theta(x_t|x_{t+1}) = -\frac{1}{2}(x_t - \mu)^T \Sigma^{-1}(x_t - \mu) + C$$

$$\begin{aligned}\log p_\phi(y|x_t) &\approx \log p_\phi(y|x_t)|_{x_t=\mu} + (x_t - \mu)\nabla_{x_t} \log p_\phi(y|x_t)|_{x_t=\mu} \\ &= (x_t - \mu)g + C_1\end{aligned}$$

Guided diffusion | Classifier guidance

NAVER AI LAB

- Overview
  - Truncation trick
    - Classifier guidance
      - $x_{t-1} \leftarrow N(\mu + s\Sigma\nabla \log p_\phi(y|x_t), \Sigma)$

Note

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$T_f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n = f(a) + f'(a)(x-a)$$

Note

$$p_{\theta,\phi}(x_t|x_{t+1}, y) = Z p_\theta(x_t|x_{t+1}) p_\phi(y|x_t)$$

$$\begin{aligned} p_\theta(x_t|x_{t+1}) &= \mathcal{N}(\mu, \Sigma) \\ \log p_\theta(x_t|x_{t+1}) &= -\frac{1}{2}(x_t-\mu)^T\Sigma^{-1}(x_t-\mu) + C \end{aligned}$$

$$\begin{aligned} \log p_\phi(y|x_t) &\approx \log p_\phi(y|x_t)|_{x_t=\mu} + (x_t-\mu)\nabla_{x_t}\log p_\phi(y|x_t)|_{x_t=\mu} \\ &= (x_t-\mu)g + C_1 \end{aligned}$$

$$\begin{aligned} \log(p_\theta(x_t|x_{t+1})p_\phi(y|x_t)) &\approx -\frac{1}{2}(x_t-\mu)^T\Sigma^{-1}(x_t-\mu) + (x_t-\mu)g + C_2 \\ &= -\frac{1}{2}(x_t-\mu-\Sigma g)^T\Sigma^{-1}(x_t-\mu-\Sigma g) + \frac{1}{2}g^T\Sigma g + C_2 \\ &= -\frac{1}{2}(x_t-\mu-\Sigma g)^T\Sigma^{-1}(x_t-\mu-\Sigma g) + C_3 \\ &= \log p(z) + C_4, z \sim \mathcal{N}(\mu+\Sigma g, \Sigma) \end{aligned}$$

Guided diffusion | Classifier guidance

NAVER AI LAB

- Overview
  - Truncation trick
    - Classifier guidance
      - $x_{t-1} \leftarrow N(\mu + s\Sigma\nabla \log p_\phi(y|x_t), \Sigma)$

Note

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$T_f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n = f(a) + f'(a)(x-a)$$

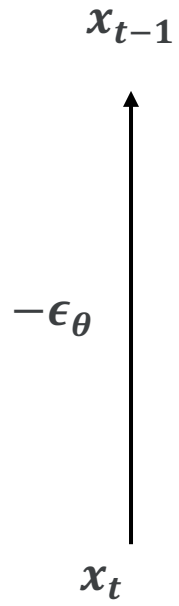
Note

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

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

Algorithm 2 Classifier guided DDIM sampling, given a diffusion model $\epsilon_\theta(x_t)$, classifier $p_\phi(y|x_t)$, and gradient scale s.

Input: class label y, gradient scale s
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for all t from T to 1 **do**
  $\hat{\epsilon} \leftarrow \epsilon_\theta(x_t) - \sqrt{1-\bar{\alpha}_t}\,\nabla_{x_t} \log p_\phi(y|x_t)$
  $x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}}\left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}}\right) + \sqrt{1-\bar{\alpha}_{t-1}}\hat{\epsilon}$
end for
return x_0



Guided diffusion | Classifier guidance

NAVER AI LAB

- Overview
  - Truncation trick
    - Classifier guidance
      - $x_{t-1} \leftarrow N(\mu + s\Sigma\nabla \log p_\phi(y|x_t), \Sigma)$

Note

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$T_f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n = f(a) + f'(a)(x-a)$$

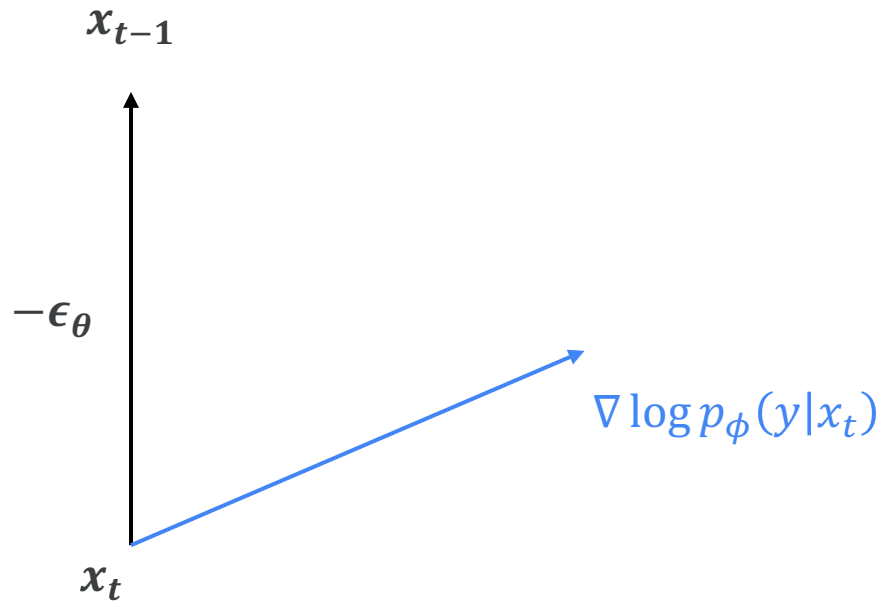
Note

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

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

Algorithm 2 Classifier guided DDIM sampling, given a diffusion model $\epsilon_\theta(x_t)$, classifier $p_\phi(y|x_t)$, and gradient scale s.

Input: class label y, gradient scale s
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for all t from T to 1 **do**
  $\hat{\epsilon} \leftarrow \epsilon_\theta(x_t) - \sqrt{1-\bar{\alpha}_t}\,\nabla_{x_t} \log p_\phi(y|x_t)$
  $x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}}\left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}}\right) + \sqrt{1-\bar{\alpha}_{t-1}}\hat{\epsilon}$
end for
return x_0



Guided diffusion | Classifier guidance

NAVER AI LAB

- Overview
  - Truncation trick
    - Classifier guidance
      - $x_{t-1} \leftarrow N(\mu + s\Sigma\nabla \log p_\phi(y|x_t), \Sigma)$

Note

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$T_f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n = f(a) + f'(a)(x-a)$$

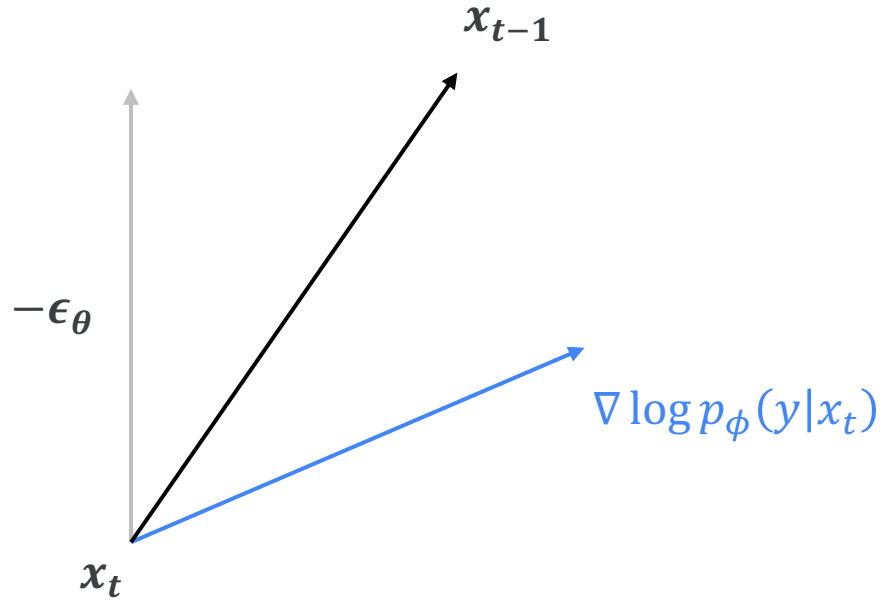
Note

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

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

Algorithm 2 Classifier guided DDIM sampling, given a diffusion model $\epsilon_\theta(x_t)$, classifier $p_\phi(y|x_t)$, and gradient scale s.

Input: class label y, gradient scale s
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for all t from T to 1 **do**
  $\hat{\epsilon} \leftarrow \epsilon_\theta(x_t) - \sqrt{1-\bar{\alpha}_t}\,\nabla_{x_t} \log p_\phi(y|x_t)$
  $x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}}\left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}}\right) + \sqrt{1-\bar{\alpha}_{t-1}}\hat{\epsilon}$
end for
return x_0



Guided diffusion | Classifier guidance

NAVER AI LAB

- Overview
  - Truncation trick
    - Classifier guidance
      - $x_{t-1} \leftarrow N(\mu + s\Sigma\nabla \log p_\phi(y|x_t), \Sigma)$

Note

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$T_f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n = f(a) + f'(a)(x-a)$$

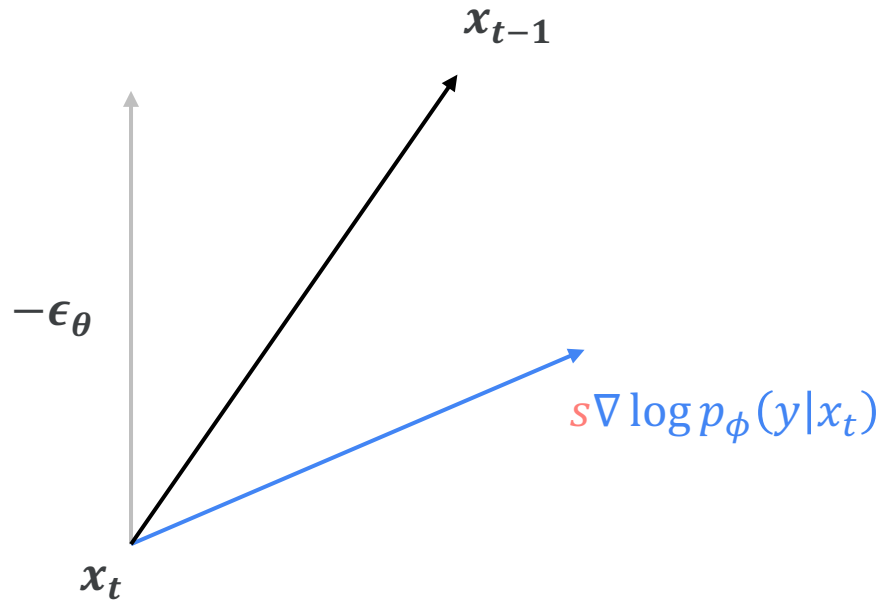
Note

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

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

Algorithm 2 Classifier guided DDIM sampling, given a diffusion model $\epsilon_\theta(x_t)$, classifier $p_\phi(y|x_t)$, and gradient scale s.

Input: class label y, gradient scale s
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for all t from T to 1 **do**
  $\hat{\epsilon} \leftarrow \epsilon_\theta(x_t) - \sqrt{1-\bar{\alpha}_t}\,\nabla_{x_t} \log p_\phi(y|x_t)$
  $x_{t-1} \leftarrow \sqrt{\bar{\alpha}_{t-1}}\left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}}{\sqrt{\bar{\alpha}_t}}\right) + \sqrt{1-\bar{\alpha}_{t-1}}\hat{\epsilon}$
end for
return x_0



Guided diffusion | Classifier guidance

NAVER AI LAB

- Overview
  - Truncation trick
    - Classifier guidance
      - $x_{t-1} \leftarrow N(\mu + s\Sigma\nabla \log p_\phi(y|x_t), \Sigma)$

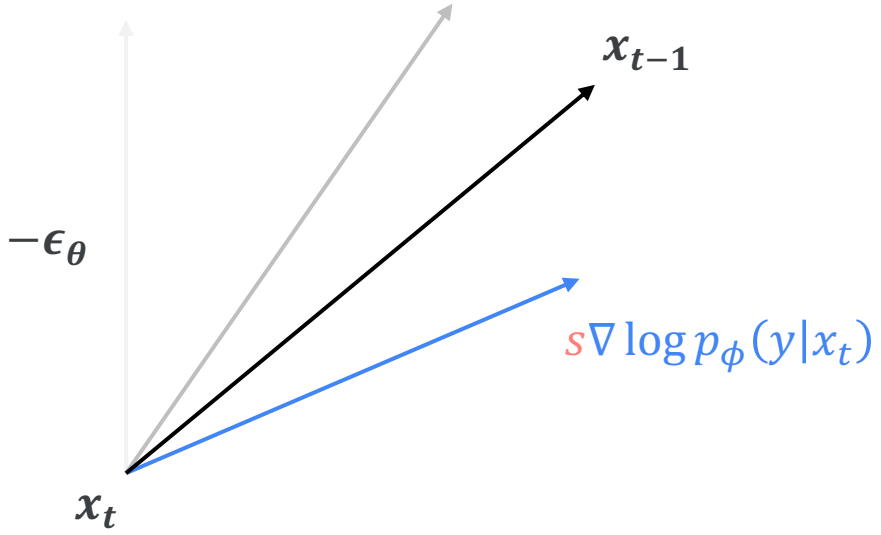
Note

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$T_f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n = f(a) + f'(a)(x-a)$$

Note

Figure 3: Samples from an unconditional diffusion model with classifier guidance to condition on the class "Pembroke Welsh corgi". Using classifier scale 1.0 (left; FID: 33.0) does not produce convincing samples in this class, whereas classifier scale 10.0 (right; FID: 12.0) produces much more class-consistent images.



Guided diffusion | Classifier guidance

NAVER AI LAB

- Overview
  - Truncation trick
    - Classifier guidance
      - $x_{t-1} \leftarrow N\left(\mu + s\Sigma\nabla \log p_\phi(y|x_t), \Sigma\right)$

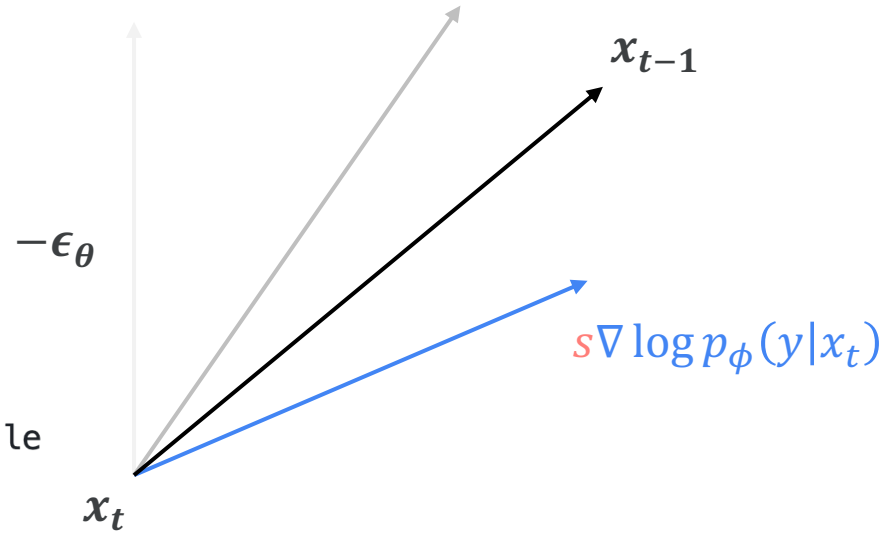
Note

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$T_f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n = f(a) + f'(a)(x-a)$$

Note

```
def cond_fn(x, t, y=None):
    assert y is not None
    with th.enable_grad():
        x_in = x.detach().requires_grad_(True)
        logits = classifier(x_in, t)
        log_probs = F.log_softmax(logits, dim=-1)
        selected = log_probs[range(len(logits)), y.view(-1)]
        return th.autograd.grad(selected.sum(), x_in)[0] * args.classifier_scale
```



NAVER AI LAB

Tackling the Generative Learning Trilemma with Denoising Diffusion GANs

ICLR 2022 Spotlight

NAVER AI LAB

- Overview
  - DDPM + GANs
  - $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\varepsilon$ (Forward)
  - $N\left(X_{t-1}; \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x_0{}' + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_{t-1}} x_t , \beta_t \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\right)$ (Reverse)

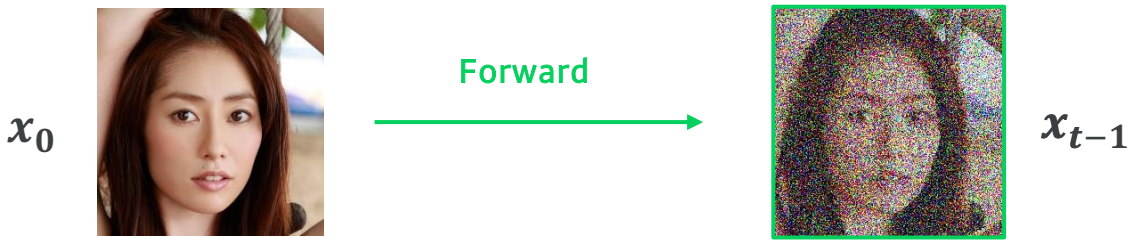
DDGAN | Framework

NAVER AI LAB

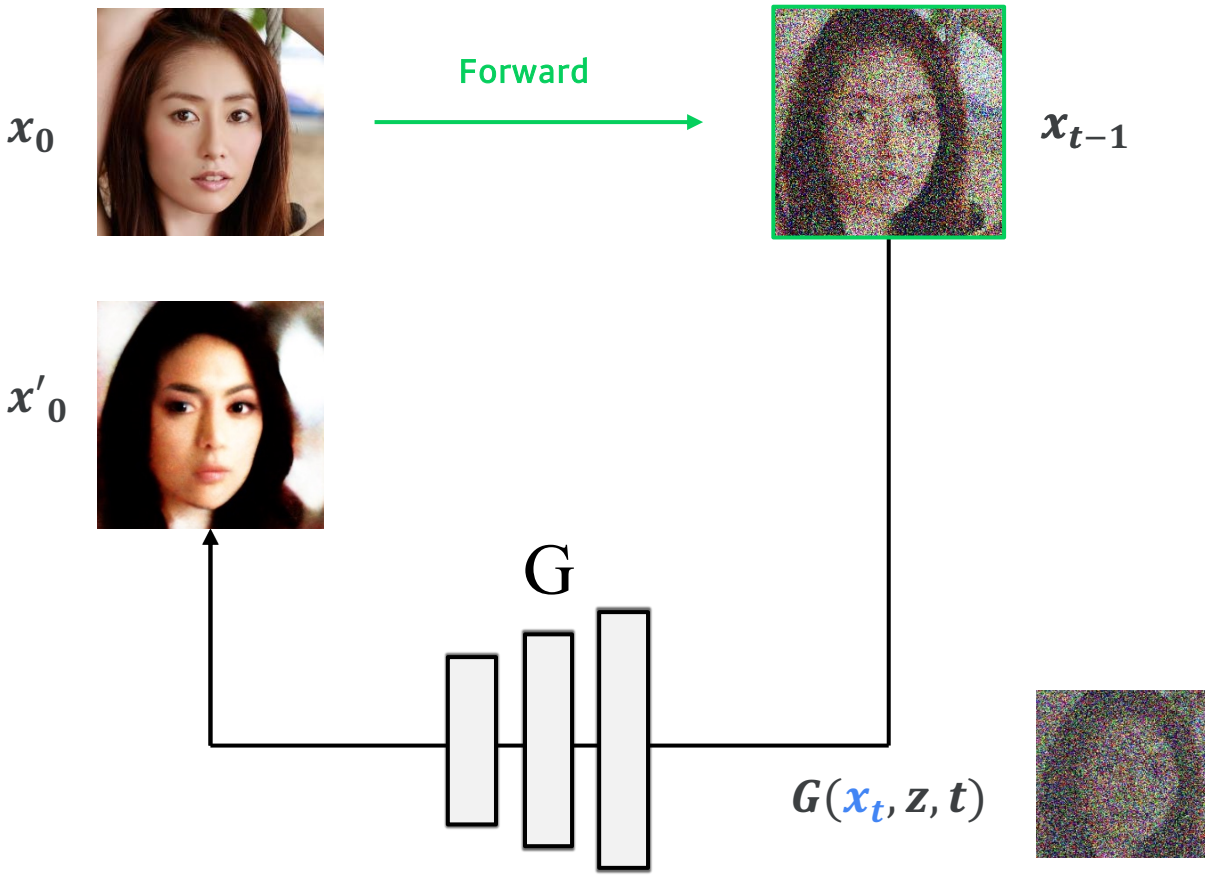
- Overview
  - DDPM + GANs
  - $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\varepsilon$ (Forward)
  - $N\left(X_{t-1}; \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x_0{}' + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_{t-1}} x_t , \beta_t \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\right)$ (Reverse)

x_0

- Overview
  - DDPM + GANs
  - $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\varepsilon$ (Forward)
  - $N\left(X_{t-1}; \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x_0{}' + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_{t-1}} x_t , \beta_t \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\right)$ (Reverse)



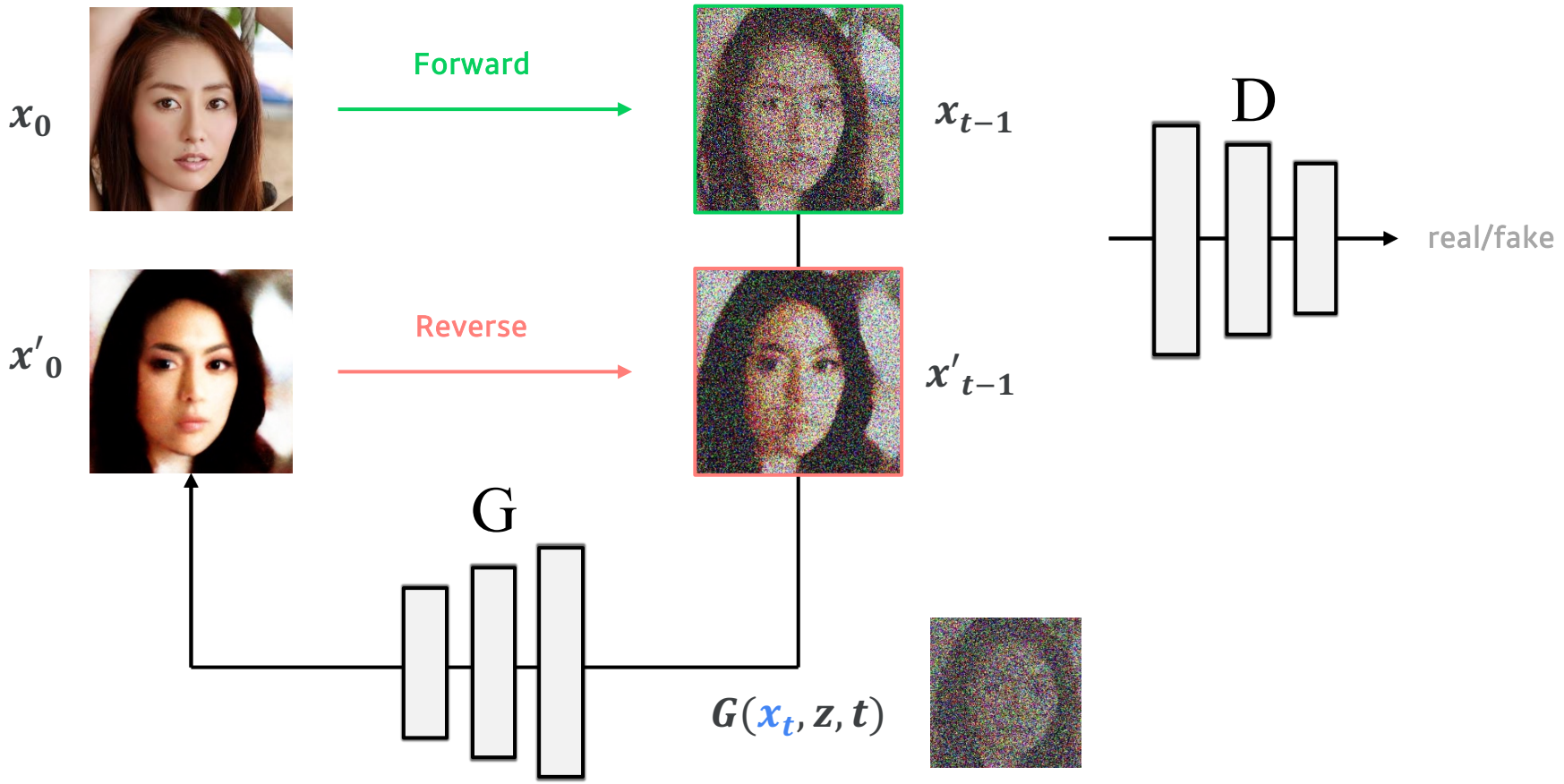
- Overview
  - DDPM + GANs
  - $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$ (Forward)
  - $N\left(X_{t-1}; \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x_0{}' + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_{t-1}} x_t , \beta_t \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\right)$ (Reverse)



DDGAN | Framework

NAVER AI LAB

- Overview
  - DDPM + GANs
  - $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$ (Forward)
  - $N\left(X_{t-1}; \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x_0{}' + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_{t-1}} x_t , \beta_t \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\right)$ (Reverse)



DDGAN | Results

NAVER AI LAB

- Overview
  - DDPM + GANs
  - $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$ (Forward)
  - $N\left(X_{t-1}; \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x_0' + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_{t-1}} x_t, \beta_t \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\right)$ (Reverse)

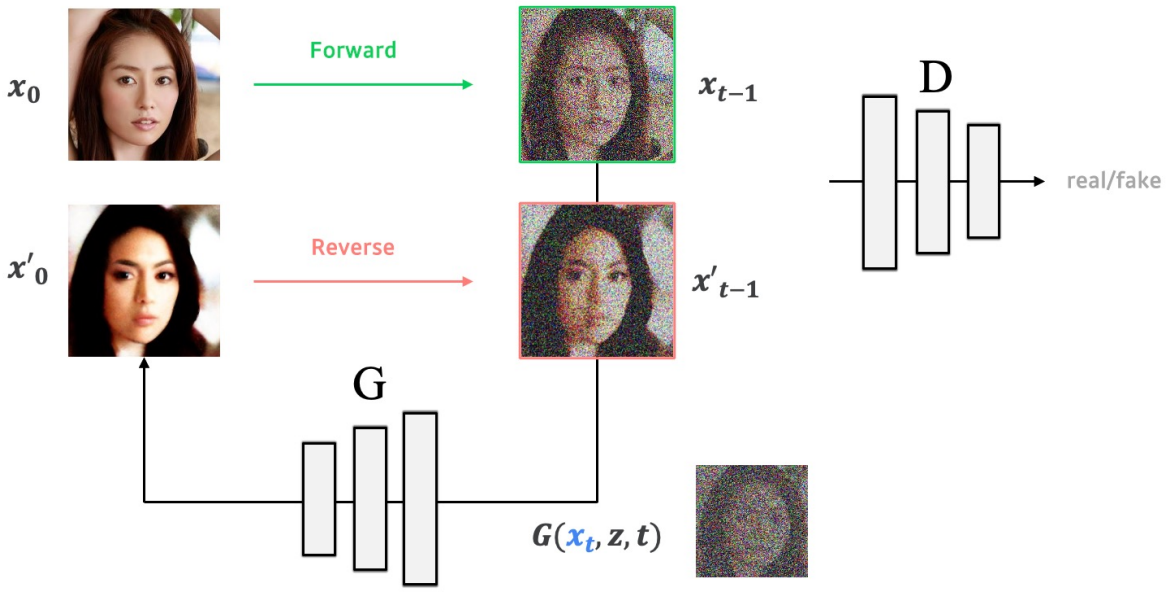


Table 1: Results for unconditional generation on CIFAR-10.

Model	IS↑	FID↓	Recall↑	NFE ↓	Time (s) ↓
Denoising Diffusion GAN (ours), T=4	9.63	3.75	0.57	4	0.21
DDPM (Ho et al., 2020)	9.46	3.21	0.57	1000	80.5
NCSN (Song & Ermon, 2019)	8.87	25.3	-	1000	107.9
Adversarial DSM (Jolicoeur-Martineau et al., 2021b)	-	6.10	-	1000	-
Likelihood SDE (Song et al., 2021b)	-	2.87	-	-	-
Score SDE (VE) (Song et al., 2021c)	9.89	2.20	0.59	2000	423.2
Score SDE (VP) (Song et al., 2021c)	9.68	2.41	0.59	2000	421.5
Probability Flow (VP) (Song et al., 2021c)	9.83	3.08	0.57	140	50.9
LSGM (Vahdat et al., 2021)	9.87	2.10	0.61	147	44.5
DDIM, T=50 (Song et al., 2021a)	8.78	4.67	0.53	50	4.01
FastDDPM, T=50 (Kong & Ping, 2021)	8.98	3.41	0.56	50	4.01
Recovery EBM (Gao et al., 2021)	8.30	9.58	-	180	-
Improved DDPM (Nichol & Dhariwal, 2021)	-	2.90	-	4000	-
VDM (Kingma et al., 2021)	-	4.00	-	1000	-
UDM (Kim et al., 2021)	10.1	2.33	-	2000	-
D3PMs (Austin et al., 2021)	8.56	7.34	-	1000	-
Gotta Go Fast (Jolicoeur-Martineau et al., 2021a)	-	2.44	-	180	-
DDPM Distillation (Luhman & Luhman, 2021)	8.36	9.36	0.51	1	-
SNGAN (Miyato et al., 2018)	8.22	21.7	0.44	1	-
SNGAN+DGflow (Ansari et al., 2021)	9.35	9.62	0.48	25	1.98
AutoGAN (Gong et al., 2019)	8.60	12.4	0.46	1	-
TransGAN (Jiang et al., 2021)	9.02	9.26	-	1	-
StyleGAN2 w/o ADA (Karras et al., 2020a)	9.18	8.32	0.41	1	0.04
StyleGAN2 w/ ADA (Karras et al., 2020a)	9.83	2.92	0.49	1	0.04
StyleGAN2 w/ Diffaug (Zhao et al., 2020)	9.40	5.79	0.42	1	0.04
Glow (Kingma & Dhariwal, 2018)	3.92	48.9	-	1	-
PixelCNN (Oord et al., 2016b)	4.60	65.9	-	1024	-
NVAE (Vahdat & Kautz, 2020)	7.18	23.5	0.51	1	0.36
IGEBM (Du & Mordatch, 2019)	6.02	40.6	-	60	-
VAEBM (Xiao et al., 2021)	8.43	12.2	0.53	16	8.79

3.1 DDPM Sampling with Manifold Constraint

In DDPMs (Ho et al., 2020), starting from a clean image $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$, a forward diffusion process $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ is described as a Markov chain that gradually adds Gaussian noise at every time steps t:

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

where $\{\beta\}_{t=0}^{T}$ is a variance schedule. By denoting $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$, the forward diffused sample at t, i.e. $\boldsymbol{x}_t$, can be sampled in one step as:

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \quad \text{where} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}). \tag{2}$$

As the reverse of the forward step $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is intractable, DDPM learns to maximize the variational lowerbound through a parameterized Gaussian transitions $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ with the parameter θ. Accordingly, the reverse process is approximated as Markov chain with learned mean and fixed variance, starting from $p(\boldsymbol{x}_T) = \mathcal{N}(\boldsymbol{x}_T; \boldsymbol{0}, \boldsymbol{I})$:

$$p_\theta(\boldsymbol{x}_{0:T}) := p_\theta(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t), \quad \text{where} \quad p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) := \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \sigma_t^2 \boldsymbol{I}). \tag{3}$$

where

$$\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) := \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right), \tag{4}$$

Here, $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$ is the diffusion model trained by optimizing the objective:

$$\min_\theta L(\theta), \quad \text{where} \quad L(\theta) := \mathbb{E}_{t, \boldsymbol{x}_0, \boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\|^2\right]. \tag{5}$$

After the optimization, by plugging learned score function into the generative (or reverse) diffusion process, one can simply sample from $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ by

$$\boldsymbol{x}_{t-1} = \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) + \sigma_t \boldsymbol{\epsilon} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right) + \sigma_t \boldsymbol{\epsilon} \tag{6}$$

In image translation using *conditional* diffusion models (Saharia et al., 2022a; Sasaki et al., 2021), the diffusion model $\boldsymbol{\epsilon}_\theta$ in (5) and (6) should be replaced with $\boldsymbol{\epsilon}_\theta(\boldsymbol{y}, \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)$ where $\boldsymbol{y}$ denotes the matched target image. Accordingly, the sample generation is tightly controlled by the matched target in a supervised manner, so that the image content change rarely happen. Unfortunately, the requirement of the *matched* targets for the training makes this approach impractical.

To address this, Dhariwal & Nichol (2021) proposed classifier-guided image translation using the unconditional diffusion model training as in (5) and a pre-trained classifier $p_\phi(\boldsymbol{y}|\boldsymbol{x}_t)$. Specifically, $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)$ in (4) and (6) are supplemented with the gradient of the classifier, i.e. $\hat{\mu}_\theta(\boldsymbol{x}_t, t) := \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) + \sigma_t \nabla_{\boldsymbol{x}_t} \log p_\phi(\boldsymbol{y}|\boldsymbol{x}_t)$. However, most of the classifiers, which should be separately trained, are not usually sufficient to control the content of the samples from the reverse diffusion process.

3.1 DDPM Sampling with Manifold Constraint

In DDPMs (Ho et al., 2020), starting from a clean image $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$, a forward diffusion process $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ is described as a Markov chain that gradually adds Gaussian noise at every time steps t:

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

where $\{\beta\}_{t=0}^{T}$ is a variance schedule. By denoting $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$, the forward diffused sample at t, i.e. $\boldsymbol{x}_t$, can be sampled in one step as:

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \quad \text{where} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}). \tag{2}$$

As the reverse of the forward step $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is intractable, DDPM learns to maximize the variational lowerbound through a parameterized Gaussian transitions $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ with the parameter θ. Accordingly, the reverse process is approximated as Markov chain with learned mean and fixed variance, starting from $p(\boldsymbol{x}_T) = \mathcal{N}(\boldsymbol{x}_T; \boldsymbol{0}, \boldsymbol{I})$:

$$p_\theta(\boldsymbol{x}_{0:T}) := p_\theta(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t), \quad \text{where} \quad p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) := \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \sigma_t^2 \boldsymbol{I}). \tag{3}$$

where

$$\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) := \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right), \tag{4}$$

Here, $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$ is the diffusion model trained by optimizing the objective:

$$\min_\theta L(\theta), \quad \text{where} \quad L(\theta) := \mathbb{E}_{t, \boldsymbol{x}_0, \boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\|^2\right]. \tag{5}$$

After the optimization, by plugging learned score function into the generative (or reverse) diffusion process, one can simply sample from $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ by

$$\boldsymbol{x}_{t-1} = \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) + \sigma_t \boldsymbol{\epsilon} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right) + \sigma_t \boldsymbol{\epsilon} \tag{6}$$

In image translation using *conditional* diffusion models (Saharia et al., 2022a; Sasaki et al., 2021), the diffusion model $\boldsymbol{\epsilon}_\theta$ in (5) and (6) should be replaced with $\boldsymbol{\epsilon}_\theta(\boldsymbol{y}, \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)$ where $\boldsymbol{y}$ denotes the matched target image. Accordingly, the sample generation is tightly controlled by the matched target in a supervised manner, so that the image content change rarely happen. Unfortunately, the requirement of the *matched* targets for the training makes this approach impractical.

To address this, Dhariwal & Nichol (2021) proposed classifier-guided image translation using the unconditional diffusion model training as in (5) and a pre-trained classifier $p_\phi(\boldsymbol{y}|\boldsymbol{x}_t)$. Specifically, $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)$ in (4) and (6) are supplemented with the gradient of the classifier, i.e. $\hat{\mu}_\theta(\boldsymbol{x}_t, t) := \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) + \sigma_t \nabla_{\boldsymbol{x}_t} \log p_\phi(\boldsymbol{y}|\boldsymbol{x}_t)$. However, most of the classifiers, which should be separately trained, are not usually sufficient to control the content of the samples from the reverse diffusion process.

x에다 noise를 더하는 과정이네

3.1 DDPM Sampling with Manifold Constraint

In DDPMs (Ho et al., 2020), starting from a clean image $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$, a forward diffusion process $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ is described as a Markov chain that gradually adds Gaussian noise at every time steps t:

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

where $\{\beta\}_{t=0}^{T}$ is a variance schedule. By denoting $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$, the forward diffused sample at t, i.e. $\boldsymbol{x}_t$, can be sampled in one step as:

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \quad \text{where} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}). \tag{2}$$

As the reverse of the forward step $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is intractable, DDPM learns to maximize the variational lowerbound through a parameterized Gaussian transitions $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ with the parameter θ. Accordingly, the reverse process is approximated as Markov chain with learned mean and fixed variance, starting from $p(\boldsymbol{x}_T) = \mathcal{N}(\boldsymbol{x}_T; \boldsymbol{0}, \boldsymbol{I})$:

$$p_\theta(\boldsymbol{x}_{0:T}) := p_\theta(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t), \quad \text{where} \quad p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) := \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \sigma_t^2 \boldsymbol{I}). \tag{3}$$

where

$$\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) := \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right), \tag{4}$$

Here, $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$ is the diffusion model trained by optimizing the objective:

$$\min_\theta L(\theta), \quad \text{where} \quad L(\theta) := \mathbb{E}_{t, \boldsymbol{x}_0, \boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\|^2\right]. \tag{5}$$

After the optimization, by plugging learned score function into the generative (or reverse) diffusion process, one can simply sample from $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ by

$$\boldsymbol{x}_{t-1} = \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) + \sigma_t \boldsymbol{\epsilon} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right) + \sigma_t \boldsymbol{\epsilon} \tag{6}$$

In image translation using *conditional* diffusion models (Saharia et al., 2022a; Sasaki et al., 2021), the diffusion model $\boldsymbol{\epsilon}_\theta$ in (5) and (6) should be replaced with $\boldsymbol{\epsilon}_\theta(\boldsymbol{y}, \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)$ where $\boldsymbol{y}$ denotes the matched target image. Accordingly, the sample generation is tightly controlled by the matched target in a supervised manner, so that the image content change rarely happen. Unfortunately, the requirement of the *matched* targets for the training makes this approach impractical.

To address this, Dhariwal & Nichol (2021) proposed classifier-guided image translation using the unconditional diffusion model training as in (5) and a pre-trained classifier $p_\phi(\boldsymbol{y}|\boldsymbol{x}_t)$. Specifically, $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)$ in (4) and (6) are supplemented with the gradient of the classifier, i.e. $\hat{\mu}_\theta(\boldsymbol{x}_t, t) := \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) + \sigma_t \nabla_{\boldsymbol{x}_t} \log p_\phi(\boldsymbol{y}|\boldsymbol{x}_t)$. However, most of the classifiers, which should be separately trained, are not usually sufficient to control the content of the samples from the reverse diffusion process.

그걸 일반식으로 표현했네
(forward)

3.1 DDPM Sampling with Manifold Constraint

In DDPMs (Ho et al., 2020), starting from a clean image $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$, a forward diffusion process $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ is described as a Markov chain that gradually adds Gaussian noise at every time steps t:

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

where $\{\beta\}_{t=0}^{T}$ is a variance schedule. By denoting $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$, the forward diffused sample at t, i.e. $\boldsymbol{x}_t$, can be sampled in one step as:

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \quad \text{where} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}). \tag{2}$$

As the reverse of the forward step $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is intractable, DDPM learns to maximize the variational lowerbound through a parameterized Gaussian transitions $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ with the parameter θ. Accordingly, the reverse process is approximated as Markov chain with learned mean and fixed variance, starting from $p(\boldsymbol{x}_T) = \mathcal{N}(\boldsymbol{x}_T; \boldsymbol{0}, \boldsymbol{I})$:

$$p_\theta(\boldsymbol{x}_{0:T}) := p_\theta(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t), \quad \text{where} \quad p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) := \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \sigma_t^2 \boldsymbol{I}). \tag{3}$$

where

$$\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) := \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right), \tag{4}$$

Here, $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$ is the diffusion model trained by optimizing the objective:

$$\min_\theta L(\theta), \quad \text{where} \quad L(\theta) := \mathbb{E}_{t, \boldsymbol{x}_0, \boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\|^2\right]. \tag{5}$$

After the optimization, by plugging learned score function into the generative (or reverse) diffusion process, one can simply sample from $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ by

$$\boldsymbol{x}_{t-1} = \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) + \sigma_t \boldsymbol{\epsilon} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right) + \sigma_t \boldsymbol{\epsilon} \tag{6}$$

In image translation using *conditional* diffusion models (Saharia et al., 2022a; Sasaki et al., 2021), the diffusion model $\boldsymbol{\epsilon}_\theta$ in (5) and (6) should be replaced with $\boldsymbol{\epsilon}_\theta(\boldsymbol{y}, \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)$ where $\boldsymbol{y}$ denotes the matched target image. Accordingly, the sample generation is tightly controlled by the matched target in a supervised manner, so that the image content change rarely happen. Unfortunately, the requirement of the *matched* targets for the training makes this approach impractical.

To address this, Dhariwal & Nichol (2021) proposed classifier-guided image translation using the unconditional diffusion model training as in (5) and a pre-trained classifier $p_\phi(\boldsymbol{y}|\boldsymbol{x}_t)$. Specifically, $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)$ in (4) and (6) are supplemented with the gradient of the classifier, i.e. $\hat{\mu}_\theta(\boldsymbol{x}_t, t) := \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) + \sigma_t \nabla_{\boldsymbol{x}_t} \log p_\phi(\boldsymbol{y}|\boldsymbol{x}_t)$. However, most of the classifiers, which should be separately trained, are not usually sufficient to control the content of the samples from the reverse diffusion process.

x_t에서 noise를 뺀것을 표현했네
분산은 상수니까, 평균만 구하면 되겠구나

3.1 DDPM Sampling with Manifold Constraint

In DDPMs (Ho et al., 2020), starting from a clean image $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$, a forward diffusion process $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ is described as a Markov chain that gradually adds Gaussian noise at every time steps t:

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

where $\{\beta\}_{t=0}^{T}$ is a variance schedule. By denoting $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$, the forward diffused sample at t, i.e. $\boldsymbol{x}_t$, can be sampled in one step as:

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \quad \text{where} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}). \tag{2}$$

As the reverse of the forward step $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is intractable, DDPM learns to maximize the variational lowerbound through a parameterized Gaussian transitions $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ with the parameter θ. Accordingly, the reverse process is approximated as Markov chain with learned mean and fixed variance, starting from $p(\boldsymbol{x}_T) = \mathcal{N}(\boldsymbol{x}_T; \boldsymbol{0}, \boldsymbol{I})$:

$$p_\theta(\boldsymbol{x}_{0:T}) := p_\theta(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t), \quad \text{where} \quad p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) := \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \sigma_t^2 \boldsymbol{I}). \tag{3}$$

where

$$\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) := \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right), \tag{4}$$

Here, $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$ is the diffusion model trained by optimizing the objective:

$$\min_\theta L(\theta), \quad \text{where} \quad L(\theta) := \mathbb{E}_{t, \boldsymbol{x}_0, \boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\|^2\right]. \tag{5}$$

After the optimization, by plugging learned score function into the generative (or reverse) diffusion process, one can simply sample from $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ by

$$\boldsymbol{x}_{t-1} = \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) + \sigma_t \boldsymbol{\epsilon} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right) + \sigma_t \boldsymbol{\epsilon} \tag{6}$$

In image translation using *conditional* diffusion models (Saharia et al., 2022a; Sasaki et al., 2021), the diffusion model $\boldsymbol{\epsilon}_\theta$ in (5) and (6) should be replaced with $\boldsymbol{\epsilon}_\theta(\boldsymbol{y}, \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)$ where $\boldsymbol{y}$ denotes the matched target image. Accordingly, the sample generation is tightly controlled by the matched target in a supervised manner, so that the image content change rarely happen. Unfortunately, the requirement of the *matched* targets for the training makes this approach impractical.

To address this, Dhariwal & Nichol (2021) proposed classifier-guided image translation using the unconditional diffusion model training as in (5) and a pre-trained classifier $p_\phi(\boldsymbol{y}|\boldsymbol{x}_t)$. Specifically, $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)$ in (4) and (6) are supplemented with the gradient of the classifier, i.e. $\hat{\mu}_\theta(\boldsymbol{x}_t, t) := \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) + \sigma_t \nabla_{\boldsymbol{x}_t} \log p_\phi(\boldsymbol{y}|\boldsymbol{x}_t)$. However, most of the classifiers, which should be separately trained, are not usually sufficient to control the content of the samples from the reverse diffusion process.

mu에 대한 수식표현이네

Tests | Read a paper

NAVER AI LAB

3.1 DDPM Sampling with Manifold Constraint

In DDPMs (Ho et al., 2020), starting from a clean image $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$, a forward diffusion process $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ is described as a Markov chain that gradually adds Gaussian noise at every time steps t:

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

where $\{\beta\}_{t=0}^{T}$ is a variance schedule. By denoting $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$, the forward diffused sample at t, i.e. $\boldsymbol{x}_t$, can be sampled in one step as:

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \quad \text{where} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}). \tag{2}$$

As the reverse of the forward step $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is intractable, DDPM learns to maximize the variational lowerbound through a parameterized Gaussian transitions $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ with the parameter θ. Accordingly, the reverse process is approximated as Markov chain with learned mean and fixed variance, starting from $p(\boldsymbol{x}_T) = \mathcal{N}(\boldsymbol{x}_T; \boldsymbol{0}, \boldsymbol{I})$:

$$p_\theta(\boldsymbol{x}_{0:T}) := p_\theta(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t), \quad \text{where} \quad p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) := \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \sigma_t^2 \boldsymbol{I}). \tag{3}$$

where

$$\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) := \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right), \tag{4}$$

Here, $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$ is the diffusion model trained by optimizing the objective:

$$\min_\theta L(\theta), \quad \text{where} \quad L(\theta) := \mathbb{E}_{t, \boldsymbol{x}_0, \boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\|^2\right]. \tag{5}$$

After the optimization, by plugging learned score function into the generative (or reverse) diffusion process, one can simply sample from $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ by

$$\boldsymbol{x}_{t-1} = \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) + \sigma_t \boldsymbol{\epsilon} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right) + \sigma_t \boldsymbol{\epsilon} \tag{6}$$

In image translation using *conditional* diffusion models (Saharia et al., 2022a; Sasaki et al., 2021), the diffusion model $\boldsymbol{\epsilon}_\theta$ in (5) and (6) should be replaced with $\boldsymbol{\epsilon}_\theta(\boldsymbol{y}, \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)$ where $\boldsymbol{y}$ denotes the matched target image. Accordingly, the sample generation is tightly controlled by the matched target in a supervised manner, so that the image content change rarely happen. Unfortunately, the requirement of the *matched* targets for the training makes this approach impractical.

To address this, Dhariwal & Nichol (2021) proposed classifier-guided image translation using the unconditional diffusion model training as in (5) and a pre-trained classifier $p_\phi(\boldsymbol{y}|\boldsymbol{x}_t)$. Specifically, $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)$ in (4) and (6) are supplemented with the gradient of the classifier, i.e. $\hat{\mu}_\theta(\boldsymbol{x}_t, t) := \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) + \sigma_t \nabla_{\boldsymbol{x}_t} \log p_\phi(\boldsymbol{y}|\boldsymbol{x}_t)$. However, most of the classifiers, which should be separately trained, are not usually sufficient to control the content of the samples from the reverse diffusion process.

더한 noise만 예측하면 되겠군

3.1 DDPM Sampling with Manifold Constraint

In DDPMs (Ho et al., 2020), starting from a clean image $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$, a forward diffusion process $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ is described as a Markov chain that gradually adds Gaussian noise at every time steps t:

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

where $\{\beta\}_{t=0}^{T}$ is a variance schedule. By denoting $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$, the forward diffused sample at t, i.e. $\boldsymbol{x}_t$, can be sampled in one step as:

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \quad \text{where} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}). \tag{2}$$

As the reverse of the forward step $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is intractable, DDPM learns to maximize the variational lowerbound through a parameterized Gaussian transitions $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ with the parameter θ. Accordingly, the reverse process is approximated as Markov chain with learned mean and fixed variance, starting from $p(\boldsymbol{x}_T) = \mathcal{N}(\boldsymbol{x}_T; \boldsymbol{0}, \boldsymbol{I})$:

$$p_\theta(\boldsymbol{x}_{0:T}) := p_\theta(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t), \quad \text{where} \quad p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) := \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \sigma_t^2 \boldsymbol{I}). \tag{3}$$

where

$$\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) := \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right), \tag{4}$$

Here, $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$ is the diffusion model trained by optimizing the objective:

$$\min_\theta L(\theta), \quad \text{where} \quad L(\theta) := \mathbb{E}_{t, \boldsymbol{x}_0, \boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\|^2\right]. \tag{5}$$

After the optimization, by plugging learned score function into the generative (or reverse) diffusion process, one can simply sample from $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ by

$$\boldsymbol{x}_{t-1} = \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) + \sigma_t \boldsymbol{\epsilon} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right) + \sigma_t \boldsymbol{\epsilon} \tag{6}$$

In image translation using *conditional* diffusion models (Saharia et al., 2022a; Sasaki et al., 2021), the diffusion model $\boldsymbol{\epsilon}_\theta$ in (5) and (6) should be replaced with $\boldsymbol{\epsilon}_\theta(\boldsymbol{y}, \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)$ where $\boldsymbol{y}$ denotes the matched target image. Accordingly, the sample generation is tightly controlled by the matched target in a supervised manner, so that the image content change rarely happen. Unfortunately, the requirement of the *matched* targets for the training makes this approach impractical.

To address this, Dhariwal & Nichol (2021) proposed classifier-guided image translation using the unconditional diffusion model training as in (5) and a pre-trained classifier $p_\phi(\boldsymbol{y}|\boldsymbol{x}_t)$. Specifically, $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)$ in (4) and (6) are supplemented with the gradient of the classifier, i.e. $\hat{\mu}_\theta(\boldsymbol{x}_t, t) := \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) + \sigma_t \nabla_{\boldsymbol{x}_t} \log p_\phi(\boldsymbol{y}|\boldsymbol{x}_t)$. However, most of the classifiers, which should be separately trained, are not usually sufficient to control the content of the samples from the reverse diffusion process.

x_(t-1)를 일반식으로 표현했네 (reverse)

NAVER AI LAB

3.1 DDPM Sampling with Manifold Constraint

In DDPMs (Ho et al., 2020), starting from a clean image $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$, a forward diffusion process $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ is described as a Markov chain that gradually adds Gaussian noise at every time steps t:

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

where $\{\beta\}_{t=0}^{T}$ is a variance schedule. By denoting $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$, the forward diffused sample at t, i.e. $\boldsymbol{x}_t$, can be sampled in one step as:

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \quad \text{where} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}). \tag{2}$$

As the reverse of the forward step $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is intractable, DDPM learns to maximize the variational lowerbound through a parameterized Gaussian transitions $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ with the parameter θ. Accordingly, the reverse process is approximated as Markov chain with learned mean and fixed variance, starting from $p(\boldsymbol{x}_T) = \mathcal{N}(\boldsymbol{x}_T; \boldsymbol{0}, \boldsymbol{I})$:

$$p_\theta(\boldsymbol{x}_{0:T}) := p_\theta(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t), \quad \text{where} \quad p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) := \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \sigma_t^2 \boldsymbol{I}). \tag{3}$$

where

$$\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) := \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right), \tag{4}$$

Here, $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$ is the diffusion model trained by optimizing the objective:

$$\min_\theta L(\theta), \quad \text{where} \quad L(\theta) := \mathbb{E}_{t, \boldsymbol{x}_0, \boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\|^2\right]. \tag{5}$$

After the optimization, by plugging learned score function into the generative (or reverse) diffusion process, one can simply sample from $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ by

$$\boldsymbol{x}_{t-1} = \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) + \sigma_t \boldsymbol{\epsilon} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right) + \sigma_t \boldsymbol{\epsilon} \tag{6}$$

In image translation using *conditional* diffusion models (Saharia et al., 2022a; Sasaki et al., 2021), the diffusion model $\boldsymbol{\epsilon}_\theta$ in (5) and (6) should be replaced with $\boldsymbol{\epsilon}_\theta(\boldsymbol{y}, \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)$ where $\boldsymbol{y}$ denotes the matched target image. Accordingly, the sample generation is tightly controlled by the matched target in a supervised manner, so that the image content change rarely happen. Unfortunately, the requirement of the *matched* targets for the training makes this approach impractical.

To address this, Dhariwal & Nichol (2021) proposed classifier-guided image translation using the unconditional diffusion model training as in (5) and a pre-trained classifier $p_\phi(\boldsymbol{y}|\boldsymbol{x}_t)$. Specifically, $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)$ in (4) and (6) are supplemented with the gradient of the classifier, i.e. $\hat{\mu}_\theta(\boldsymbol{x}_t, t) := \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) + \sigma_t \nabla_{\boldsymbol{x}_t} \log p_\phi(\boldsymbol{y}|\boldsymbol{x}_t)$. However, most of the classifiers, which should be separately trained, are not usually sufficient to control the content of the samples from the reverse diffusion process.

classifier guidance

NAVER AI LAB

3.1 DDPM Sampling with Manifold Constraint

In DDPMs (Ho et al., 2020), starting from a clean image $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$, a forward diffusion process $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ is described as a Markov chain that gradually adds Gaussian noise at every time steps t:

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

where $\{\beta\}_{t=0}^{T}$ is a variance schedule. By denoting $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$, the forward diffused sample at t, i.e. $\boldsymbol{x}_t$, can be sampled in one step as:

$$\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \quad \text{where} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}). \tag{2}$$

As the reverse of the forward step $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ is intractable, DDPM learns to maximize the variational lowerbound through a parameterized Gaussian transitions $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ with the parameter θ. Accordingly, the reverse process is approximated as Markov chain with learned mean and fixed variance, starting from $p(\boldsymbol{x}_T) = \mathcal{N}(\boldsymbol{x}_T; \boldsymbol{0}, \boldsymbol{I})$:

$$p_\theta(\boldsymbol{x}_{0:T}) := p_\theta(\boldsymbol{x}_T) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t), \quad \text{where} \quad p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) := \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \sigma_t^2 \boldsymbol{I}). \tag{3}$$

where

$$\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) := \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right), \tag{4}$$

Here, $\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)$ is the diffusion model trained by optimizing the objective:

$$\min_\theta L(\theta), \quad \text{where} \quad L(\theta) := \mathbb{E}_{t, \boldsymbol{x}_0, \boldsymbol{\epsilon}}\left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\|^2\right]. \tag{5}$$

After the optimization, by plugging learned score function into the generative (or reverse) diffusion process, one can simply sample from $p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ by

$$\boldsymbol{x}_{t-1} = \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) + \sigma_t \boldsymbol{\epsilon} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, t)\right) + \sigma_t \boldsymbol{\epsilon} \tag{6}$$

In image translation using *conditional* diffusion models (Saharia et al., 2022a; Sasaki et al., 2021), the diffusion model $\boldsymbol{\epsilon}_\theta$ in (5) and (6) should be replaced with $\boldsymbol{\epsilon}_\theta(\boldsymbol{y}, \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)$ where $\boldsymbol{y}$ denotes the matched target image. Accordingly, the sample generation is tightly controlled by the matched target in a supervised manner, so that the image content change rarely happen. Unfortunately, the requirement of the *matched* targets for the training makes this approach impractical.

To address this, Dhariwal & Nichol (2021) proposed classifier-guided image translation using the unconditional diffusion model training as in (5) and a pre-trained classifier $p_\phi(\boldsymbol{y}|\boldsymbol{x}_t)$. Specifically, $\boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t)$ in (4) and (6) are supplemented with the gradient of the classifier, i.e. $\hat{\mu}_\theta(\boldsymbol{x}_t, t) := \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t) + \sigma_t \nabla_{\boldsymbol{x}_t} \log p_\phi(\boldsymbol{y}|\boldsymbol{x}_t)$. However, most of the classifiers, which should be separately trained, are not usually sufficient to control the content of the samples from the reverse diffusion process.

Summary

NAVER AI LAB

Summary

NAVER AI LAB

Denoising diffusion probabilistic models (DDPM)

- $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon,\ \epsilon \sim \mathcal{N}(0, \mathbf{I})$ (Forward)
  - $\beta_1 = 10^{-4}, \beta_T = 0.02$
  - $\alpha_t := 1 - \beta_t,\ \bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$
- $\epsilon - \epsilon_\theta(\mathbf{x}_t) = \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon)$ (Loss)
  - ϵ_θ = prediction network
- $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right) + \sqrt{\tilde{\beta}_t}\epsilon,\ \epsilon \sim \mathcal{N}(0, \mathbf{I})$ (Reverse)
  - $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$
    - $\tilde{\beta}_t = \beta_t$로 해도 성능차이 없음

Diffusion Models Beat GANs on Image Synthesis

Summary

NAVER AI LAB

- DDIM α = DDPM $\bar{\alpha}$
- $\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\epsilon$ (Forward)
- $\epsilon - \epsilon_\theta(\mathbf{x}_t) = \epsilon - \epsilon_\theta(\sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\epsilon)$ (Loss)
  - ϵ_θ = prediction network
- $\mathbf{x}_{t-1} = \sqrt{\alpha_{t-1}}\underbrace{\left(\frac{\mathbf{x}_t - \sqrt{1-\alpha_t}\epsilon_\theta(\mathbf{x}_t)}{\sqrt{\alpha_t}}\right)}_{\text{predicted } \mathbf{x}_0 = f_\theta(\mathbf{x}_t)} + \underbrace{\sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\epsilon_\theta(\mathbf{x}_t)}_{\text{direction pointing to } \mathbf{x}_t} + \underbrace{\sigma_t\epsilon}_{\text{noise}}$ (Reverse)
  - deterministic when $\sigma_t = 0$ → consistency

Denoising diffusion implicit models (DDIM)

ICLR 2021

Tackling the Generative Learning Trilemma with Denoising Diffusion GANs

ICLR 2022

2020

NeurIPS 2020

NeurIPS 2021

2022

Denoising diffusion probabilistic models (DDPM)

Diffusion Models Beat GANs on Image Synthesis

- $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon,\ \epsilon \sim \mathcal{N}(0, \mathbf{I})$ (Forward)
  - $\beta_1 = 10^{-4}, \beta_T = 0.02$
  - $\alpha_t := 1 - \beta_t,\ \bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$
- $\epsilon - \epsilon_\theta(\mathbf{x}_t) = \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon)$ (Loss)
  - ϵ_θ = prediction network
- $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right) + \sqrt{\tilde{\beta}_t}\epsilon,\ \epsilon \sim \mathcal{N}(0, \mathbf{I})$ (Reverse)
  - $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$
    - $\tilde{\beta}_t = \beta_t$로 해도 성능차이 없음

Summary

NAVER AI LAB

- DDIM α = DDPM $\bar{\alpha}$
- $\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\epsilon$ (Forward)
- $\epsilon - \epsilon_\theta(\mathbf{x}_t) = \epsilon - \epsilon_\theta(\sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\epsilon)$ (Loss)
  - ϵ_θ = prediction network
- $\mathbf{x}_{t-1} = \sqrt{\alpha_{t-1}}\underbrace{\left(\frac{\mathbf{x}_t - \sqrt{1-\alpha_t}\epsilon_\theta(\mathbf{x}_t)}{\sqrt{\alpha_t}}\right)}_{\text{predicted } \mathbf{x}_0 = f_\theta(\mathbf{x}_t)} + \underbrace{\sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\epsilon_\theta(\mathbf{x}_t)}_{\text{direction pointing to } \mathbf{x}_t} + \underbrace{\sigma_t\epsilon}_{\text{noise}}$ (Reverse)
  - deterministic when $\sigma_t = 0$ → consistency

Denoising diffusion implicit models (DDIM)

ICLR 2021

Tackling the Generative Learning Trilemma with Denoising Diffusion GANs

ICLR 2022

2020

NeurIPS 2020

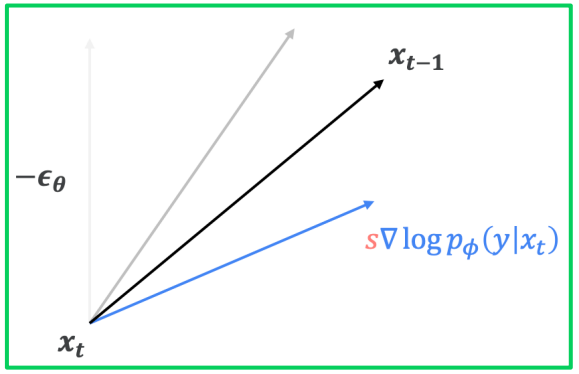
NeurIPS 2021

2022

Denoising diffusion probabilistic models (DDPM)

- $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon,\ \epsilon \sim \mathcal{N}(0, \mathbf{I})$ (Forward)
  - $\beta_1 = 10^{-4}, \beta_T = 0.02$
  - $\alpha_t := 1 - \beta_t,\ \bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$
- $\epsilon - \epsilon_\theta(\mathbf{x}_t) = \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon)$ (Loss)
  - ϵ_θ = prediction network
- $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right) + \sqrt{\tilde{\beta}_t}\epsilon,\ \epsilon \sim \mathcal{N}(0, \mathbf{I})$ (Reverse)
  - $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$
    - $\tilde{\beta}_t = \beta_t$로 해도 성능차이 없음

Diffusion Models Beat GANs on Image Synthesis



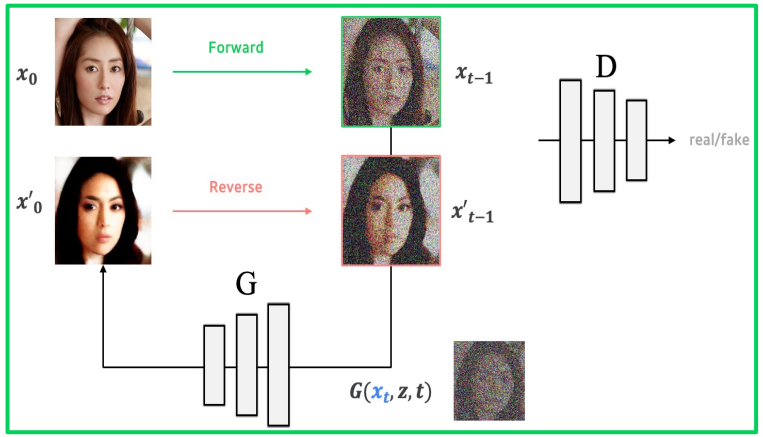
Summary

NAVER AI LAB

- DDIM α = DDPM $\bar{\alpha}$
- $\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\epsilon$ (Forward)
- $\epsilon - \epsilon_\theta(\mathbf{x}_t) = \epsilon - \epsilon_\theta(\sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\epsilon)$ (Loss)
  - ϵ_θ = prediction network
- $\mathbf{x}_{t-1} = \sqrt{\alpha_{t-1}}\underbrace{\left(\frac{\mathbf{x}_t - \sqrt{1-\alpha_t}\epsilon_\theta(\mathbf{x}_t)}{\sqrt{\alpha_t}}\right)}_{\text{predicted } \mathbf{x}_0 = f_\theta(\mathbf{x}_t)} + \underbrace{\sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\epsilon_\theta(\mathbf{x}_t)}_{\text{direction pointing to } \mathbf{x}_t} + \underbrace{\sigma_t\epsilon}_{\text{noise}}$ (Reverse)
  - deterministic when $\sigma_t = 0$ → consistency

Denoising diffusion implicit models (DDIM)

ICLR 2021



Tackling the Generative Learning Trilemma with Denoising Diffusion GANs

ICLR 2022

2020

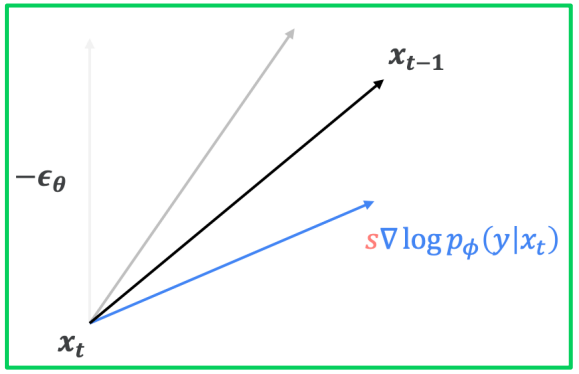
NeurIPS 2020

Denoising diffusion probabilistic models (DDPM)

- $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon,\ \epsilon \sim \mathcal{N}(0, \mathbf{I})$ (Forward)
  - $\beta_1 = 10^{-4}, \beta_T = 0.02$
  - $\alpha_t := 1 - \beta_t,\ \bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$
- $\epsilon - \epsilon_\theta(\mathbf{x}_t) = \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon)$ (Loss)
  - ϵ_θ = prediction network
- $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right) + \sqrt{\tilde{\beta}_t}\epsilon,\ \epsilon \sim \mathcal{N}(0, \mathbf{I})$ (Reverse)
  - $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$
    - $\tilde{\beta}_t = \beta_t$로 해도 성능차이 없음

NeurIPS 2021

Diffusion Models Beat GANs on Image Synthesis



END !

2022

NAVER AI LAB

Thank you !

jhkim.ai@navercorp.com

Junho Kim

https://github.com/taki0112