

Denoising Diffusion Probabilistic Models (DDPM)

Ho et al., 2020

Kenichiro Goto
西川研究室 B3
2025-11-14

Agenda

1. 導入
2. メカニズムの理解
3. 理論と定式化
4. 実験とアーキテクチャ
5. 議論とまとめ
6. Appendix

1. 導入

1.1. 背景

従来の生成モデルには以下のような弱点があった

GAN

- Pros: 高品質な生成
- Cons: モード崩壊, 学習が不安定 (D, Gのバランスが難しい)

VAE

- Pros: 尤度ベースで学習が安定
- Cons: ELBOという下界を最大化するため生成がぼやけやすい

Flow-based Model

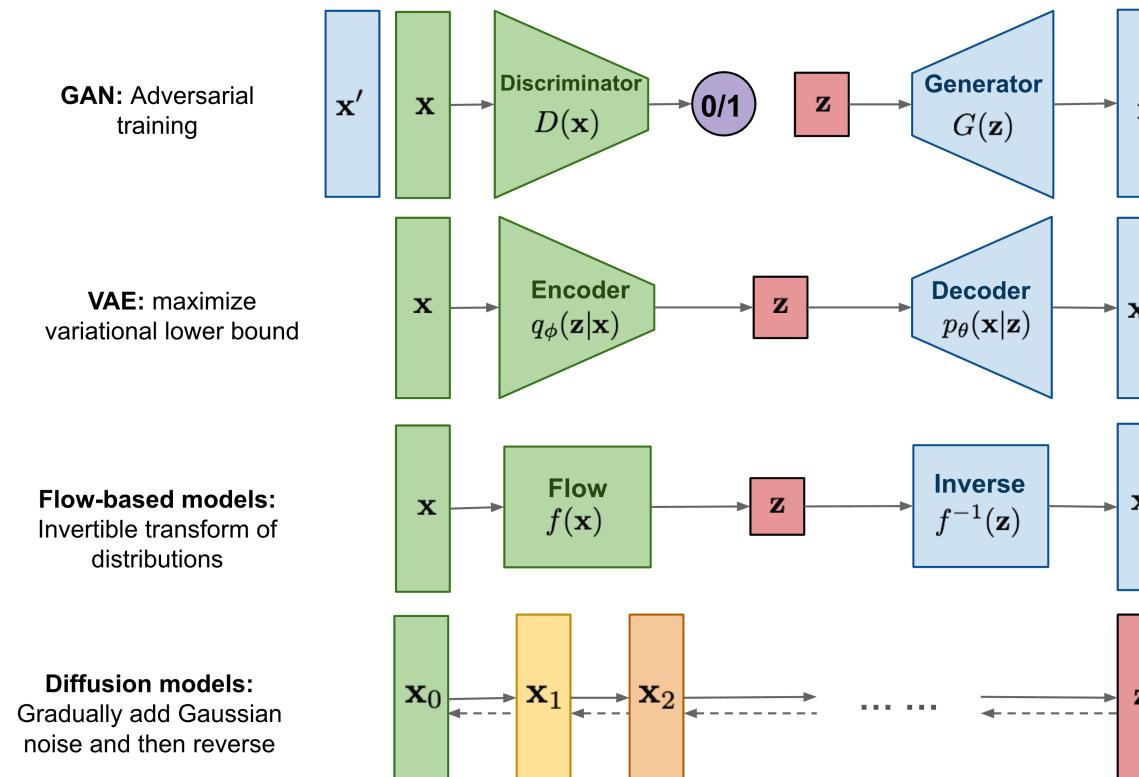
- Pros: 尤度を厳密に計算可能
- Cons: 可逆変換の制約がありアーキテクチャ設計が難しい

1.2. DDPMの位置付け

DDPMは高品質かつ安定した学習（尤度ベース）の両立を目標とする

2つのプロセスで構成される

1. Forward Process (拡散過程) : データを徐々にノイズにしていく (固定プロセスでアルゴリズム的)
2. Reverse Process (逆拡散過程) : 学習対象。ノイズから画像空間上のデータへ復元する



1.3. 論理展開流れ

DDPMでは、仕組みとしてはシンプルだが、随所に簡略化や性能向上のための工夫が凝らされている

拡散過程

- 元画像 \mathbf{x}_0 にノイズを加算する
- 完全なノイズ (\mathbf{x}_T , 標準正規分布) になるまで続ける。(ノイズから画像を生成する逆のプロセスとしたいから)
- 数学的に扱いやすいことが重要 (閉形式)

→ 逆拡散過程

- 逆過程の真の分布は \mathbf{x}_0 が必要。だがそれは知ることができないのでNNでの近似をここで使う

→ NNによる近似

計算できない $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ の代わりに p_θ を推定する

- 逆過程の分布 $p_\theta(x_{t-1}|x_t)$ は、平均 μ_θ と分散 Σ_θ を持つ正規分布であると仮定
- NNは、 μ_θ (分布の平均) を直接予測するのではなく、時刻 t の画像 x_t に加えられたノイズ ϵ を予測するように学習させる
 - なぜなら、加えられたノイズ ϵ が予測できれば、それを使って x_{t-1} の平均 μ_θ を (間接的に) 計算できることが分かっているから

補足：Langevin Dynamics や Flow との関係

Langevin Dynamics

- これは「スコアベースモデル（Score-Based Models）」と呼ばれる、DDPMの兄弟のようなモデル群で、サンプリング（生成）時によく使われる手法
- DDPMのサンプリングステップ（ x_t から x_{t-1} を生成する）は、数学的に見ると Langevin Dynamics の1ステップと非常によく似た形になっている

Flow-based Models

- これはDDPMとは別の生成モデルのカテゴリ
- しかし、DDPMを連続時間で考えると（SDEやODE）、その生成プロセスは Flow-based Models の理論と深く関連することが示されている

2. メカニズムの理解

2.1. Forward Process (拡散過程)

$q(\mathbf{x}_t | \mathbf{x}_{t-1})$: データ→ノイズ方向への変換

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

- β_t は分散スケジュール (ハイパーパラメータ)

この正規分布からのサンプリングは、 **Reparameterization Trick** により、
以下のように「ノイズの加算」として表現できます。

$$\mathbf{x}_t = \sqrt{1 - \beta_t} \mathbf{x}_{t-1} + \sqrt{\beta_t} \boldsymbol{\epsilon} \quad (\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})) \quad (2)$$

- 直感的な理解
 - $\sqrt{1 - \beta_t} \mathbf{x}_{t-1}$: 元の画像を少し縮小 (スケールダウン)
 - $\sqrt{\beta_t} \boldsymbol{\epsilon}$: 小さなノイズを加算
- t が大きくなるにつれて q はガウシアンノイズ $\mathcal{N}(\mathbf{0}, \mathbf{I})$ に漸近する

2.1. Forward Process (拡散過程)

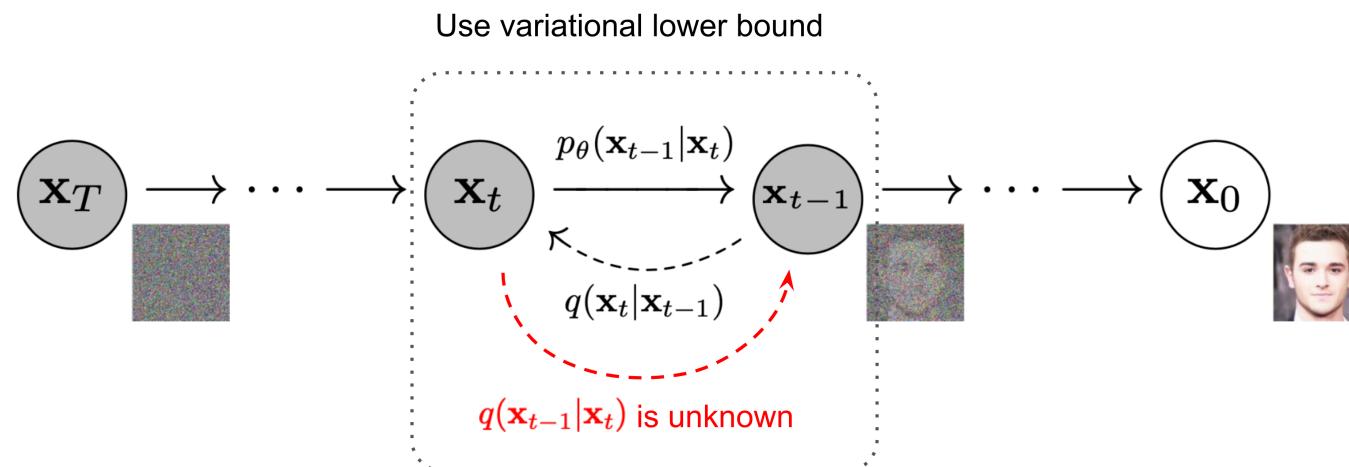
マルコフ性の利点： T ステップの反復計算は不要。

$\alpha_t := 1 - \beta_t$, $\bar{\alpha}_t := \prod_{s=1}^t \alpha_s$ とおくと、

\mathbf{x}_0 から任意の \mathbf{x}_t を一発でサンプリング可能 (Reparameterization Trick)

$$q(\mathbf{x}_t | \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t} \mathbf{x}_0, (1 - \bar{\alpha}_t) \mathbf{I}) \quad \mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \boldsymbol{\epsilon} \quad (\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})) \quad (4)$$

これは 訓練時に極めて重要 となる。



2.2. Reverse Process (逆拡散過程)

$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$: ノイズからデータを復元する

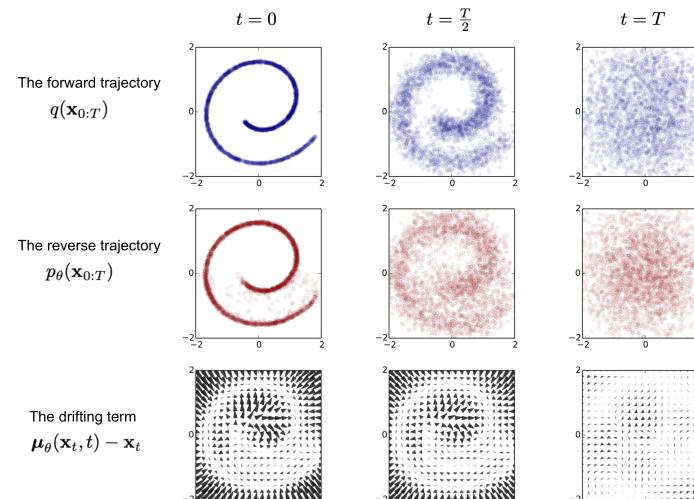
Forward Processの逆をたどる。

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ からスタートし、ノイズを除去 (Denoising) しながら \mathbf{x}_0 を復元する。

この逆過程 p_θ をニューラルネットワーク (NN) で近似する。

$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)) \quad (5)$$

- NN $\boldsymbol{\mu}_\theta$ が真の逆過程の平均 $\tilde{\boldsymbol{\mu}}$ を予測するように学習する (のちに ϵ の推定に置き換えられる)
- $\boldsymbol{\Sigma}_\theta$ はDDPMでは固定値 ($\tilde{\beta}_t \mathbf{I}$ or $\beta_t \mathbf{I}$)



3. 理論と定式化

3.1. 損失関数の導出

どうやって学習するか

目標：尤度 $p_\theta(\mathbf{x}_t)$ を最大化したい。

→ 真の逆過程 q とNNによる近似 p_θ のKL divergenceを最小化したい

変分下界 (ELBO) を用いて損失 $L_{\text{VLB}} \geq L$ を定義する (Appendix)

$$L_{\text{VLB}} = \mathbb{E}_q[-\log p_\theta(\mathbf{x}_{0:T}) + \log q(\mathbf{x}_{1:T}|\mathbf{x}_0)] \quad (6)$$

これを整理すると各ステップのKL divergenceの和になる

$$L_{\text{VLB}} = \mathbb{E}_q[\underbrace{D_{KL}(q(\mathbf{x}_T)||p(\mathbf{x}_T))}_{L_T} + \sum_{t>1} \underbrace{D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} - \underbrace{\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0}] \quad (7)$$

3.1. 損失関数の導出

L_{t-1} 項にある $q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)$ のように \mathbf{x}_0 で条件付けると解析的に計算可能になる (Appendix)

$$q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}) \quad (8)$$

ここで

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_t \right) \quad (9)$$

($\boldsymbol{\epsilon}_t$ は \mathbf{x}_t の生成に使われたノイズ)

L_{t-1} は、この 真の平均 $\tilde{\boldsymbol{\mu}}_t$ と NN の予測 $\boldsymbol{\mu}_\theta$ の差を測る項になる

3.2 損失関数の簡略化

L_{simple} : ϵ -prediction

L_{t-1} は p_θ と q の平均 μ のL2距離 (MSE) として計算できる。

$$L_{t-1} = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2} \|\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)\|^2 \right] + C \quad (10)$$

さらに、 μ を直接予測するのではなく、
 $\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$ の関係を使って、ノイズ ϵ を予測する 問題に置き換える。
(ϵ -prediction (Appendix))

NNで $\epsilon_\theta(\mathbf{x}_t, t)$ が、真のノイズ ϵ を予測するように学習する。

3.2 損失関数の簡略化

L_{simple} : 最終的な損失関数

論文では重み係数を無視した以下の単純な損失関数の方が安定しており性能も良かったと報告している

$$L_t^{\text{simple}} = \mathbb{E}_{t \sim [1, T], \epsilon_t} \left[\|\epsilon_t - \epsilon_\theta(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon_t, t)\|^2 \right] \quad (11)$$

L_T は定数なので最終的な損失関数は式6である

ここまでをまとめると

ニューラルネット ϵ_θ は、入力されたノイズ画像 \mathbf{x}_t と時刻 t から、そこに含まれるノイズ成分 ϵ を予測するように学習すればよい。

Algorithm 1 Training

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

Algorithm 2 Sampling

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

3.3 Score Matching との関連性 (発展)

損失関数 L_{simple} は、Denoising Score Matching[^1]の目的関数と密接に関連している。

データ分布の勾配 $\nabla_{\mathbf{x}} \log q(\mathbf{x})$ を「Score」と呼ぶ。

Langevin Dynamics (スコアベースのサンプリング手法) において、Scoreの推定が重要。

DDPMのノイズ予測 ϵ_θ は、

このScore s_θ と以下の関係にあることが示されている。

(ノイズ予測は、実質的にデータの勾配 (Score) を推定している)

$$s_\theta(\mathbf{x}_t, t) = -\frac{\epsilon_\theta(\mathbf{x}_t, t)}{\sqrt{1 - \bar{\alpha}_t}} \quad (12)$$

[^1] <https://ieeexplore.ieee.org/abstract/document/6795935>

4. 実験とアーキテクチャ

4.1. アーキテクチャ (U-Net)

ノイズ予測器 ϵ_θ の実装

ノイズ予測 $\epsilon_\theta(\mathbf{x}_t, t)$ のNNアーキテクチャ

- 入力 (\mathbf{x}_t) と 出力 (ϵ) の解像度が同じである必要がある。
- 論文では U-Net (ResNetブロック + Attention) を採用。

時刻 t の入力方法 (Time Embedding)

- t は離散値だが、そのままNNに入力しない。
- TransformerのPositional Encodingと同様の手法で高次元ベクトルに変換し、
- U-Netの各ResNetブロックに加算する。

4.2. サンプリング（生成）

学習とは逆に、Reverse Process p_θ を T ステップ実行する。

1. $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ (ガウスノイズ) からスタート

2. $t = T, \dots, 1$ について以下を反復

- $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
- NN ϵ_θ を使って μ_θ を計算
- $\mathbf{x}_{t-1} = \mu_\theta(\mathbf{x}_t, t) + \sigma_t \mathbf{z}$
 $(\mu_\theta = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(\mathbf{x}_t, t) \right))$

3. \mathbf{x}_0 が生成画像

4.3. 生成結果

生成結果 (CIFAR-10, CelebA)

- 評価指標: FID (Fréchet Inception Distance), Inception Score
 - 当時のSOTA (特にGAN) に匹敵、あるいは凌駕するスコアを達成。
- Ablation Study:
 - μ 予測より ϵ 予測の方が性能が良いことを確認。
 - 損失の重み付けを無視した L_{simple} の方が性能が良いことを確認。

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

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

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

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

5. 議論とまとめ

5.1. 議論とまとめ

本研究の貢献

- 高品質な画像生成: GANに匹敵する高忠実度な画像生成を達成。
- 安定した学習: 敵対的学習が不要で、安定した尤度ベースの学習が可能。
- 理論的背景: 变分推論とScore Matchingに裏打ちされた堅牢な理論。
- 単純な実装: 最終的な損失関数は「ノイズ予測のMSE」というシンプルな形。

課題

サンプリング速度: T ステップ (e.g., 1000~4000) の反復計算が必要なため、推論（生成）が非常に遅い

影響

- DDPMの成功と課題（速度）が、爆発的な後続研究を生み出した。
 - DDIM (2020): サンプリングを高速化 (e.g., 50ステップ)
 - Latent Diffusion (2021): 高解像度化 (\rightarrow Stable Diffusionの基礎)

参考文献

- [1] Ho, J., Jain, A., & Abbeel, P. (2020). "Denoising Diffusion Probabilistic Models". Advances in Neural Information Processing Systems (NeurIPS), 33. (Also available as arXiv:2006.11239).
- [2] Kingma, D. P., & Welling, M. (2013). "Auto-Encoding Variational Bayes". arXiv preprint arXiv:1312.6114.
- [3] Song, Y., & Ermon, S. (2020). "Score-Based Generative Modeling through Stochastic Differential Equations". International Conference on Learning Representations (ICLR). (Also available as arXiv:2011.13456).
- [4] Weng, L. (2021, July 11). "What are Diffusion Models?". Lil'Log (Blog). Retrieved November 14, 2025, from <https://lilianweng.github.io/posts/2021-07-11-diffusion-models/>

Appendix

- $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$ は、考えられる全ての \mathbf{x}_0 について周辺化（積分）する必要があり、計算が困難
- しかし、 \mathbf{x}_0 を（訓練データとして）観測している前提で条件付けを行うと、ベイズの定理

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)}$$
 が使える
- Forward Processの定義により、右辺の3つの項 ($q(\mathbf{x}_t|\mathbf{x}_{t-1})$, $q(\mathbf{x}_{t-1}|\mathbf{x}_0)$, $q(\mathbf{x}_t|\mathbf{x}_0)$) は全て既知のガウス分布
- ガウス分布の積・商は解析的に計算可能であり、結果として $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ もまたガウス分布 $\mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t, \tilde{\boldsymbol{\beta}}_t \mathbf{I})$ となることが導出できる

ϵ -prediction

- $\tilde{\mu}_t$ と μ_θ は、それぞれ ϵ と ϵ_θ を使って（線形変換で）表現できるため、 $\|\tilde{\mu}_t - \mu_\theta\|^2$ は $\|\epsilon - \epsilon_\theta\|^2$ の定数倍として書き直せる

$$\begin{aligned}
 \|\tilde{\mu}_t - \mu_\theta\|^2 &= \left\| \left(\frac{1}{\sqrt{\alpha_t}} \mathbf{x}_t - \frac{\beta_t}{\sqrt{\alpha_t} \sqrt{1 - \bar{\alpha}_t}} \epsilon \right) - \left(\frac{1}{\sqrt{\alpha_t}} \mathbf{x}_t - \frac{\beta_t}{\sqrt{\alpha_t} \sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta \right) \right\|^2 \\
 &= \left\| -\frac{\beta_t}{\sqrt{\alpha_t} \sqrt{1 - \bar{\alpha}_t}} \epsilon + \frac{\beta_t}{\sqrt{\alpha_t} \sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta \right\|^2 \\
 &= \left\| \frac{\beta_t}{\sqrt{\alpha_t} \sqrt{1 - \bar{\alpha}_t}} (\epsilon_\theta - \epsilon) \right\|^2 \\
 &= \left(\frac{\beta_t^2}{\alpha_t (1 - \bar{\alpha}_t)} \right) \|\epsilon - \epsilon_\theta\|^2
 \end{aligned} \tag{13}$$

- したがって、損失 L_{t-1} は ϵ 予測の「重み付き」MSEとして表現できる
- しかし、DDPM論文の重要な発見として、この重み付けを全て無視し、単純なMSE $\|\epsilon - \epsilon_\theta\|^2$ を使った L_{simple} の方が、学習が安定し性能も向上したと報告されている

ELBOを使った L の導出

ELBO : 普通に求めるのは難しい尤度 $-\log p_\theta$ を求めるためにそれを常に上回る値のこと

$$\begin{aligned} -\log p_\theta(\mathbf{x}_0) &\leq -\log p_\theta(\mathbf{x}_0) + D_{KL}(q(\mathbf{x}_{1:T}|\mathbf{x}_0)||p_\theta(\mathbf{x}_{1:T}|\mathbf{x}_0)) \quad ; \text{KL is non-negative} \\ &= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_{\mathbf{x}_{1:T} \sim q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})/p_\theta(\mathbf{x}_0)} \right] \\ &= -\log p_\theta(\mathbf{x}_0) + \mathbb{E}_q \left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} + \log p_\theta(\mathbf{x}_0) \right] \\ &= \mathbb{E}_q \left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} \right] \\ \text{Let } L_{VLB} &= \mathbb{E}_{q(\mathbf{x}_{0:T})} \left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})} \right] \geq -\mathbb{E}_{q(\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0) \end{aligned} \tag{14}$$