Ⅰ. Denoising Diffusion Probabilistic Model

1. Denoising Diffusion Probabilistic Model

Diffusion Overview

- Diffusion model은 backward process를 통해서 explicit(가우시안)한 분포에서 sampling한 random noise로부터 denoising을 통해 데이터를 만들어내는 생성 방식.
- Forward process(=diffusion process)와 backward process(denoising)으로 구성.
- 이미지에 noise를 주입하는 forward process는 쉽게 정의할 수 있지만, backward process는 정의하기 어려움. 따라서 DDPM은 backward process를 diffusion probabilistic model로 학습하는 것을 의미함.

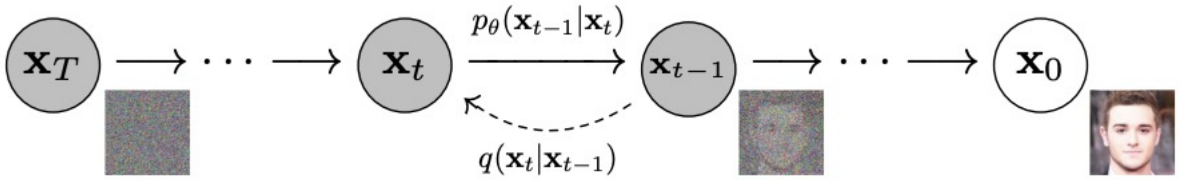

Figure 2: The directed graphical model considered in this work.

- 위 그림에서 q는 noise를 더하는 forward process(=diffusion process)이고,
- P는 gaussian noise를 원래 이미지로 되돌리는 reverse process를 나타냄.

1. Denoising Diffusion Probabilistic Model

Forward process

- Forward process q는 미세한 Gaussian noise를 원본 이미지 x_0 에 점차적으로 추가하는 과정을 의미함.

- Noise를 주입하고 제거하는 과정은 큰 변화이기 때문에 한번에 적용하기 어려움. 따라서 T=1000 step으로 변화를 매우 작게 하여 학습을 쉽게 만듦. 이때 Markov Chain으로 표현됨.

1. Markov Chain

정의 : Markov 성질을 갖는 이산 확률과정

Markov 성질 : 특정 상태의 확률(t+1)은 오직 현재t)의 상태에 의존한다.

이산 확률과정 : 이산적인 시간(0초, 1초, 2초 …) 속에서의 확률적인 현상

0시점에서 T시점으로 noise를 주입하고 제거하는 과정에서 1000번의 step으로 나누어 수행하고 이때 Markov chain을 적용

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

1. Denoising Diffusion Probabilistic Model

Forward process

- Gaussian noise를 서서히 주입하는 과정
- Forward process는 조건부 가우시안 $q(\mathbf{x}_{1:T}|\mathbf{x}_0)$ 의 Markov chain으로 구성됨.

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

- 주입되는 gaussian noise의 크기는 사전에 정의되고, β_t 로 표현됨.
- 따라서 Forward process는 따로 학습할 필요가 없음.

  - *Code example*

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

1. Denoising Diffusion Probabilistic Model

Forward process

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

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

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

$$\alpha_t := 1 - \beta_t \text{ and } \bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$$

증명 : https://jang-inspiration.com/ddpm-1#cd89a0a7eec4470996db10628c9bf556

- *Code example*

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

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

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

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

    return x_seq
```

1. Denoising Diffusion Probabilistic Model

Backward process

- Noise로부터 이미지를 복원하는 과정.
- q와 반대로 noise를 점진적으로 걷어내는 denoising process를 의미함.
- 따라서 조건부 확률 q와 time이 반대로 뒤바뀐 p로 표현됨.

$$q(x_t|x_{t-1}) \qquad p(x_{t-1}|x_t)$$

- 사전에 정의된 q와는 달리 p는 알기 어렵기 때문에 p_θ 로 근사하여 구하고 p_θ 는 학습된 모델을 통해 구함.

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

reverse process의 수식 표현

- 위 식에서 평균 μ_θ 와 표준편차 $\sum_\theta$ 는 학습되어야 하는 parameter임.

Backward process

- Noise로부터 이미지를 복원하는 과정.
- q와 반대로 noise를 점진적으로 걷어내는 denoising process를 의미함.
- 따라서 조건부 확률 q와 time이 반대로 뒤바뀐 p로 표현됨.

$$q(x_t|x_{t-1}) \qquad\qquad p(x_{t-1}|x_t)$$

- 사전에 정의된 q와는 달리 p는 알기 어렵기 때문에 p_θ 로 근사하여 구하고 p_θ 는 학습된 모델을 통해 구함.

    - *Code example*

```
def p_mean_variance(model, x, t):

    # Make model prediction
    out = model(x, t.to(device))

    # Extract the mean and variance
    mean, log_var = torch.split(out, 2, dim=-1)
    var = torch.exp(log_var)
    return mean, log_var
```

1. Denoising Diffusion Probabilistic Model

Diffusion Objective Function

- Maximize log likelihood

- 증명 : https://jang-inspiration.com/ddpm-1#cd89a0a7eec4470996db10628c9bf556

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

최종 Loss Function 증명

1. Denoising Diffusion Probabilistic Model

Objective Function

DDPM은 Loss를 굉장히 간단하게 정의하여 성능을 개선함.

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

Diffusion Model의 최종적인 Loss Function

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

1. Denoising Diffusion Probabilistic Model

DDPM Loss

- L_T는 p가 generate하는 noise와 q가 x_0 가 주어졌을 때 generate하는 T시점의 noise간의 분포 차이
- DDPM에서는 β.를 학습 가능한 파라미터가 아니라 사전에 정의한 상수로 fix하기 때문에 고려하지 않아도 됨.

- 이유

1000 step으로 나누고 noise를 linear하게 증가시키도록 linear noise scheduling을 설정할 경우

최종적인 X_T는 isotropic gaussian(원형 대칭) 분포와 매우 유사하게 형성되기 때문에 굳이 학습 필요 X

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

Diffusion Model의 최종적인 Loss Function

DDPM Loss

- L_{t-1} 는 p와 q의 reverse / forward process의 분포 차이를 의미함. Forward대로 Denoising하기 때문에 두 값이 최대한 비슷해지는 방향으로 학습을 진행.

(1) $\Sigma_\theta(X_t, t)$ 의 상수화

- P의 표준편차는 $\sigma_t^2 I$ 상수 행렬에 대응함. 따라서 학습 대상이 하나 줄어듦.

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

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

Diffusion Model의 최종적인 Loss Function

DDPM Loss

- L_{t-1} 는 p와 q의 reverse / forward process의 분포 차이를 의미함. Forward대로 Denoising하기 때문에 두 값이 최대한 비슷해지는 방향으로 학습을 진행.

(2) $q(x_{t-1}|x_t, x_0)$

앞서 정의한 것처럼 표현할 수 있음

- $q(x_{t-1}|x_t, x_0) = N(x_{t-1}; \tilde{\mu}(x_t, x_0), \tilde{\beta}_t \mathrm{I})$
- $where, \tilde{\mu}_t(x_t, x_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t} x_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}t} x_t, \ \tilde{\beta}_t := \frac{(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\beta_t$

1. Denoising Diffusion Probabilistic Model

DDPM Loss

- L_{t-1} 는 p와 q의 reverse / forward process의 분포 차이를 의미함. Forward대로 Denoising하기 때문에 두 값이 최대한 비슷해지는 방향으로 학습을 진행.

(3) $p_\theta(x_{t-1}|x_t)$

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

1. Denoising Diffusion Probabilistic Model

DDPM Loss

- L_{t-1} 는 p와 q의 reverse / forward process의 분포 차이를 의미함. Forward대로 Denoising하기 때문에 두 값이 최대한 비슷해지는 방향으로 학습을 진행.

(4) $\mu_\theta(x_t, t)$

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \tilde{\boldsymbol{\mu}}_t\left(\mathbf{x}_t, \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t))\right) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) \tag{11}$$

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

증명 : https://jang-inspiration.com/ddpm-2

1. Denoising Diffusion Probabilistic Model

DDPM Loss

- L_{t-1} 는 p와 q의 reverse / forward process의 분포 차이를 의미함. Forward대로 Denoising하기 때문에 두 값이 최대한 비슷해지는 방향으로 학습을 진행.

(5) sampling

$p_\theta(x_{t-1}|x_t)$ 로 x_{t-1} 을 추출할 수 있고 반복적으로 수행하여 $\mathbf{x}_0$ 을 리턴

Algorithm 2 Sampling

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

DDPM Loss

- L_{t-1} 는 p와 q의 reverse / forward process의 분포 차이를 의미함. Forward대로 Denoising하기 때문에 두 값이 최대한 비슷해지는 방향으로 학습을 진행.

(6) L_{t-1}

앞의 값들과 KL Divergence로 계산 시 아래와 같은 값을 출력할 수 있음.

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

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

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

DDPM Loss

- L_{t-1} 는 p와 q의 reverse / forward process의 분포 차이를 의미함. Forward대로 Denoising하기 때문에 두 값이 최대한 비슷해지는 방향으로 학습을 진행.

(7) $L_{simple}(\theta)$

Loss function을 epsilon에 대한 식의 형태로 다시 한 번 표현할 수 있음.

이러한 형태를 simplified objective function이라 부르고 더 효율적인 학습이 가능함.

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

Algorithm 1 Training

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