Diffussion Model

- jensen's inequality:함수 안에 넣기
- intractability: 다루기 힘듦 Tractability:다루기 쉬움
- 생성모델 종류&특징: diffusion model은 간단히는 는 좀 느리다(cf.삼각형)(why 1000번과정이라-마르코프 체인(전단계에 얽히는)) vs.DDIM:마르코프 체인 굳이 안써서 굳이 막 전단계에 얽히지 않아.
- 생성모델 구분

-명백:VAE, Diffussion Model

-명백x:GAN

- GAN의 이해: 명백x, 확실x,어쩌다 보니 잡혔다(생성자,판별자)..(확률적으로 확률이 적다)
- VAE의 이해: 명백한 모델, px와 z로 하려다가 안돼서 qz인코더를 달았다. 그래서 qz와 z와 왔다갔다하면 px를 근사할 수 있을거다

- 가우시안의 중요도:분포를 가정할때는 가우시안이라하기 Diffusion할때
- Forward: 이미지에서 노이즈 더함 가우시안의 형태로 (학습x, 그냥 노이즈 더하는 것)
- Reverse: 노이즈를 덜어내는 것(학습o) 가우시안 형태로
- 세타 있으면 학습하는 모델표시 reverse는 학습하니까 세타 있는거!

Unet 쓰는 이유 노이즈가 더는 과정인데 이때 UNet이 이걸 잘한다..근데 요즘에는 안씀

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

- 아래는 DDPM loss 중요부분만 필기
- $= \mathbb{E}_{1:\tau \sim q(x_{1:\tau}|x_0)}\left[-\,log\; p_\theta(x_\tau) - \sum_{t=1}^{T} log\, \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})} \right]$

노란색 부분에 t=1 를 대입하고 그 부분만 따로 뺐다. 그러니 두번째 텀은 t=2 로 바뀌고, 세번째 텀이 생겼다.(왜 굳이 이러냐 전개 하고싶으니)

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

:결국 구한 것에서 초록색 부분만 사용한다 우리는

$\sum_{t=2} D_{KL}(q(x_{t-1}|x_t,x_0)||p_\theta(x_{t-1}|x_t))$ 여기서 모델에게 맡기지 않는 loss term 즉,

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

이 부분만 구하면 되는데 이는 가우시안이니까 정규화 시키면 평균(뮤)를 구하면 된다.

수식을 통해 계산해서 구하면 됨 아래는 일부만 보여줌

신기하게도 PDF 과 같음!!!

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

$\frac{1}{\sqrt{2\pi\beta_t(\frac{1-\bar\alpha_{t-1}}{1-\bar\alpha_t})}}exp(-\frac{1}{2\beta_t(\frac{1-\bar\alpha_{t-1}}{1-\bar\alpha_t})}[x_{t-1}-(\frac{\sqrt{\bar\alpha_{t-1}}\beta_t}{1-\bar\alpha_t}x_0+\frac{\sqrt{\alpha_t}(1-\bar\alpha_{t-1})}{1-\bar\alpha_t}x_t)]^2)$

$mean = a = \frac{\sqrt{\bar\alpha_{t-1}}\beta_t}{1-\bar\alpha_t}x_0 + \frac{\sqrt{\alpha_t}(1-\bar\alpha_{t-1})}{1-\bar\alpha_t}x_t$

$var = b = \beta_t(\frac{1-\bar\alpha_{t-1}}{1-\bar\alpha_t})$

최종:

$L_{simple}(\theta) := \mathbb{E}_{t,x_0,\epsilon}\left[\left\|\epsilon - \epsilon_\theta(\sqrt{\bar\alpha_t}\ x_0 + \sqrt{1-\bar\alpha_t}\ \epsilon)\right\|^2\right]$

$$\epsilon$$

이게 즉 노이즈 실제 노이즈랑 예측한 노이즈 빼는 것이다.

결국 최종 정리

- $\mathbf{x}_t = \sqrt{\bar\alpha_t}\mathbf{x}_0 + \sqrt{1-\bar\alpha_t}\epsilon,\ \epsilon \sim \mathcal{N}(0,\mathbf{I})$ `(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

여기서 앱실론은 포워드 과정에서 완성된 전체 노이즈고 입실론 세타는 현재 단계 진행중인 단계의 노이즈

- $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right) + \sqrt{\tilde{\beta}_t}\epsilon,\ \epsilon \sim \mathcal{N}(0, \mathbf{I})$ `(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

++사람이 받아들이는 이미지와 인공지능이 받아들이는 이미지의 차이는 큼 기계는 노이즈가 더해지면 아예 뭔가 인식을 못함

++앱실론은 계속 랜덤으로(포워드 과정에서)

참고 링크

stabilityai/stable-diffusion-xl-base-1.0 · Hugging Face
We're on a journey to advance and democratize artificial intelligence through open source and open science.
🤗 https://huggingface.co/stabilityai/stable-diffusion-xl-base-1.0