Deep Generative Modelling

Introduced models

- EBM
- VAE
- GAN
- Autoregressive model
- Normalize flows
- Diffusion model

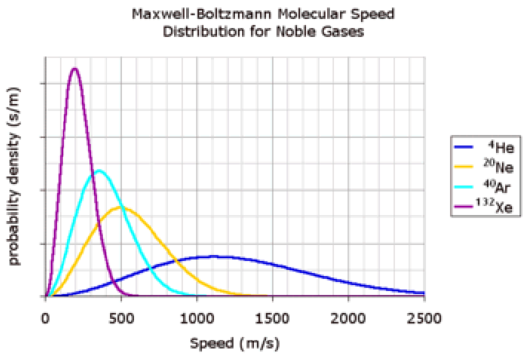
Energy-Based Models

- Input space의 각 data를 single scalar값인 energy로 mapping하는 함수를 만드는 것

-scalar=energy

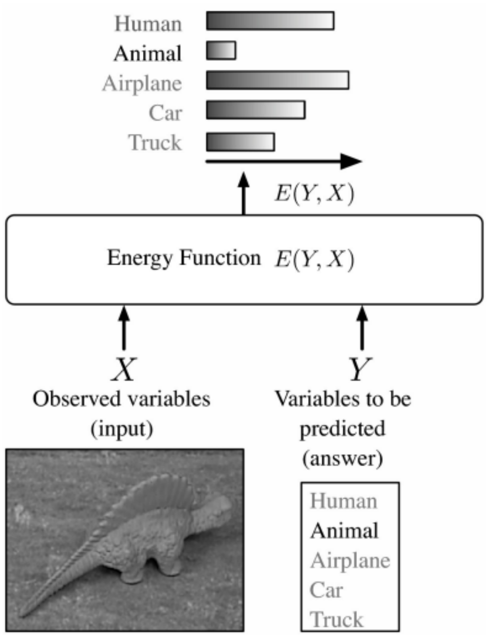
- E(x)로 표현되는 pdf p(x)에 대한 observation에 기초됨 $p(\boldsymbol{x}) = \frac{e^{-E(\boldsymbol{x})}}{\int_{\tilde{\boldsymbol{x}} \in \mathcal{X}} e^{-E(\tilde{\boldsymbol{x}})}}.$

-E(x): $\mathbb{R}^D \rightarrow \mathbb{R}$ which associates realistic points with low values and unrealistic points with high values



Energy-Based Models

Energy-Based Model for Decision-Making

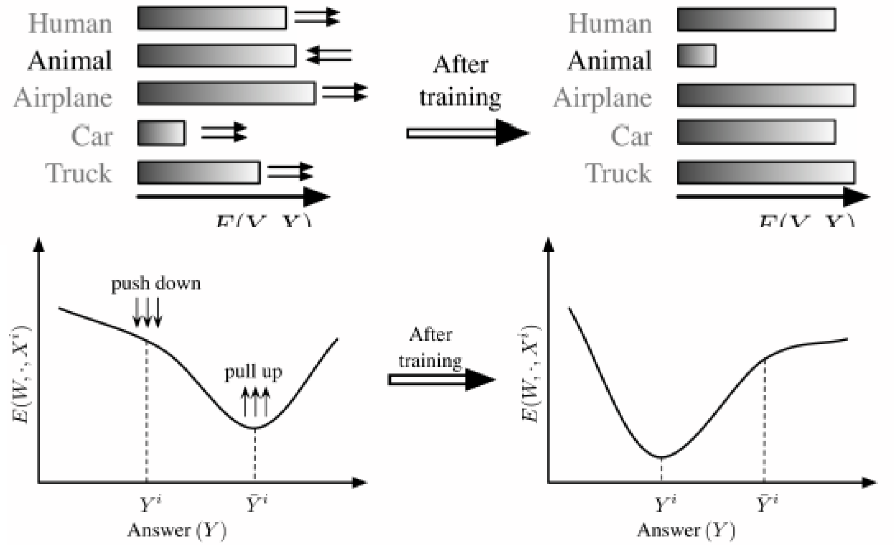


- **Model:** **Measures the compatibility between an observed variable X and a variable to be predicted Y through an energy function E(Y,X).**

$$Y^* = \operatorname{argmin}_{Y \in \mathcal{Y}} E(Y, X).$$

- **Inference:** **Search for the Y that minimizes the energy within a set** $\mathcal{Y}$
- **If the set has low cardinality, we can use exhaustive search.**

Designing a Loss Functional



- **Correct answer has the lowest energy -> LOW LOSS**
- **Lowest energy is not for the correct answer -> HIGH LOSS**

Energy-Based Models

$$FreeEnergy(\mathbf{x}) = -log \sum_{\mathbf{h}} e^{-Energy(\mathbf{x},\mathbf{h})}$$

$$\frac{\partial log P(\mathbf{x})}{\partial \theta} = -\frac{\partial FreeEnergy(\mathbf{x})}{\partial \theta} + \sum_{\tilde{\mathbf{x}}} P(\tilde{\mathbf{x}}) \frac{\partial FreeEnergy(\tilde{\mathbf{x}})}{\partial \theta}$$

$$E_{\hat{P}}[\frac{\partial \log P(\mathbf{x})}{\partial \theta}] = -E_{\hat{P}}[\frac{\partial FreeEnergy(\mathbf{x})}{\partial \theta}] + E_{P}[\frac{\partial FreeEnergy(\mathbf{x})}{\partial \theta}]$$

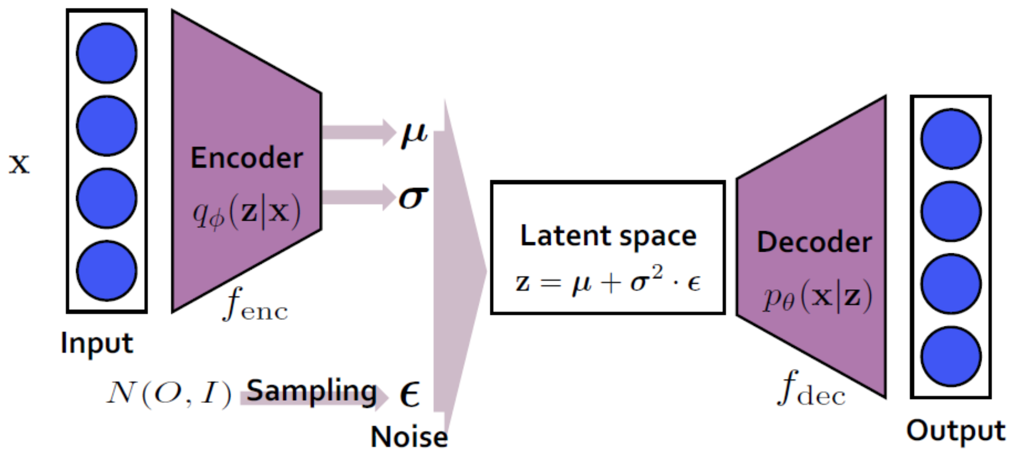
$$\frac{\partial \log P(\mathbf{x})}{\partial \theta} = - < \frac{\partial FreeEnergy(\mathbf{x})}{\partial \theta} >_{data} + \boxed{< \frac{\partial FreeEnergy(\mathbf{x})}{\partial \theta} >_{model}}$$

Variational Auto Encoder

- Input image X를 잘 설명하는 feature를 추출하여 Latent vector z에 담고, 이 Latent vector z를 통해 X와 유사하지만 완전히 새로운 데이터를 생성하는 것

- Auto Encoder와 유사한 구조

- Decoder를 학습시키는 것이 목적



Variational Auto Encoder

- $p_\theta(x|z)$, $p_\theta(z)$가 있음

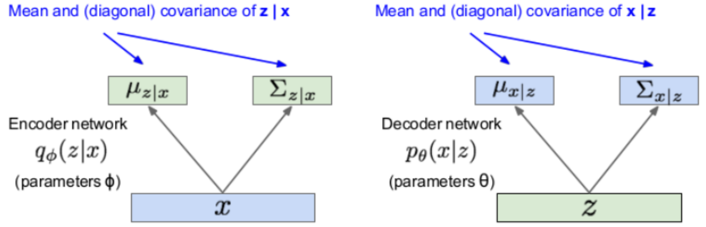
이때 $p_\theta(x) = \int p_\theta(z)\, p_\theta(x|z)dz$ 를 구하기는 어렵다

$$\begin{aligned}
\log(p(x)) &= \int \log(p(x))q_\phi(z|x)dz \quad \leftarrow \int q_\phi(z|x)dz = 1 \\
&= \int \log\left(\frac{p(x,z)}{p(z|x)}\right)q_\phi(z|x)dz \quad \leftarrow p(x) = \frac{p(x,z)}{p(z|x)} \\
&= \int \log\left(\frac{p(x,z)}{q_\phi(z|x)} \cdot \frac{q_\phi(z|x)}{p(z|x)}\right)q_\phi(z|x)dz \\
&= \underbrace{\int \log\left(\frac{p(x,z)}{q_\phi(z|x)}\right)q_\phi(z|x)dz}_{\boldsymbol{ELBO(\emptyset)}} + \underbrace{\int \log\left(\frac{q_\phi(z|x)}{p(z|x)}\right)q_\phi(z|x)dz}_{\boldsymbol{KL(q_\emptyset(z|x)\,||\,p(z|x))}}
\end{aligned}$$

$$\log(p(x)) = ELBO(\phi) + KL(q_\phi(z|x)|p(z|x))$$

$$q_{\phi^*}(z|x) = \underset{\phi}{\operatorname{argmax}}\, ELBO(\phi)$$

$$\boxed{\begin{aligned}
ELBO(\phi) &= \int \log\left(\frac{p(x,z)}{q_\phi(z|x)}\right)q_\phi(z|x)dz \\
&= \int \log\left(\frac{p(x|z)p(z)}{q_\phi(z|x)}\right)q_\phi(z|x)dz \\
&= \int \log(p(x|z))q_\phi(z|x)dz - \int \log\left(\frac{q_\phi(z|x)}{p(z)}\right)q_\phi(z|x)dz \\
&= \mathbb{E}_{q_\phi(z|x)}[\log(p(x|z))] - KL(q_\phi(z|x)||p(z))
\end{aligned}}$$



Variational Auto Encoder

$$= -D_{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p_\theta(\boldsymbol{z})) + \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\ln p_\theta(\boldsymbol{x}|\boldsymbol{z})]$$
$$\equiv \mathcal{L}(\theta, \phi; \boldsymbol{x}),$$

이 값을 올리는 방향으로 training을 진행

$$\arg\min_{\phi,\theta} \sum_i \underbrace{-\mathbb{E}_{q_\phi(z|x_i)}[\log(p(x_i|g_\theta(z)))] + KL(q_\phi(z|x_i)||p(z))}_{L_i(\phi,\theta,x_i)}$$

원 데이터에 대한 likelihood

Variational inference를 위한
approximation class 중 선택

다루기 쉬운 확률 분포 중 선택

$$L_i(\phi,\theta,x_i) = \underbrace{-\mathbb{E}_{q_\phi(z|x_i)}[\log(p(x_i|g_\theta(z)))]}_{\text{Reconstruction Error}} + \underbrace{KL(q_\phi(z|x_i)||p(z))}_{\text{Regularization}}$$

Reconstruction Error

- 현재 샘플링 용 함수에 대한 negative log likelihood
- x_i에 대한 복원 오차 (AutoEncoder 관점)

Regularization

- 현재 샘플링 용 함수에 대한 추가 조건
- 샘플링의 용의성/생성 데이터에 대한 통제성을 위한 조건을 prior에 부여 하고 이와 유사해야 한다는 조건을 부여

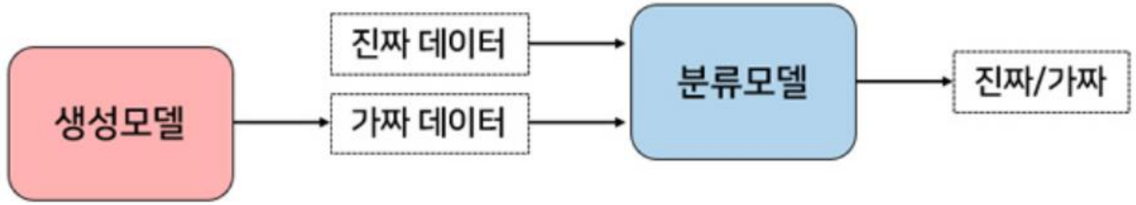
이후 Backpropagation을 통해 L을 높임

Generative Adversarial Networks

- Discriminator 와 Generator로 이루어져 있음

-Generator: 진짜 같은 데이터를 만들기

-Discriminator: 판별을 잘하기



$$\min_G \max_D V(D,G) = \boxed{E_{x\sim p_{data}(x)}[\log D(x)]} + \boxed{E_{z\sim p_z(z)}[\log(1 - D(G(z)))]}$$

X: real data
Z: fake data

Real data를 discriminator에 넣고 이 값을 log를 취했을 때 얻는 기댓값

Fake data를 generator에 넣고 이를 discriminator에 넣는다. 이 결과를 log(1-결과값)를 취했을 때 얻는 기댓값

이 값이 최대가 되도록 D를 학습, 이 값이 최소가 되도록 G를 학습시키는 것이 목표

Autoregressive Model

- 자기 회귀 모델: 이전 관측 값이 이후 관측 값에 영향을 주는 모델

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t,$$

$\epsilon_t : white\ noise$

Chain rule of probability에 기초한 모델

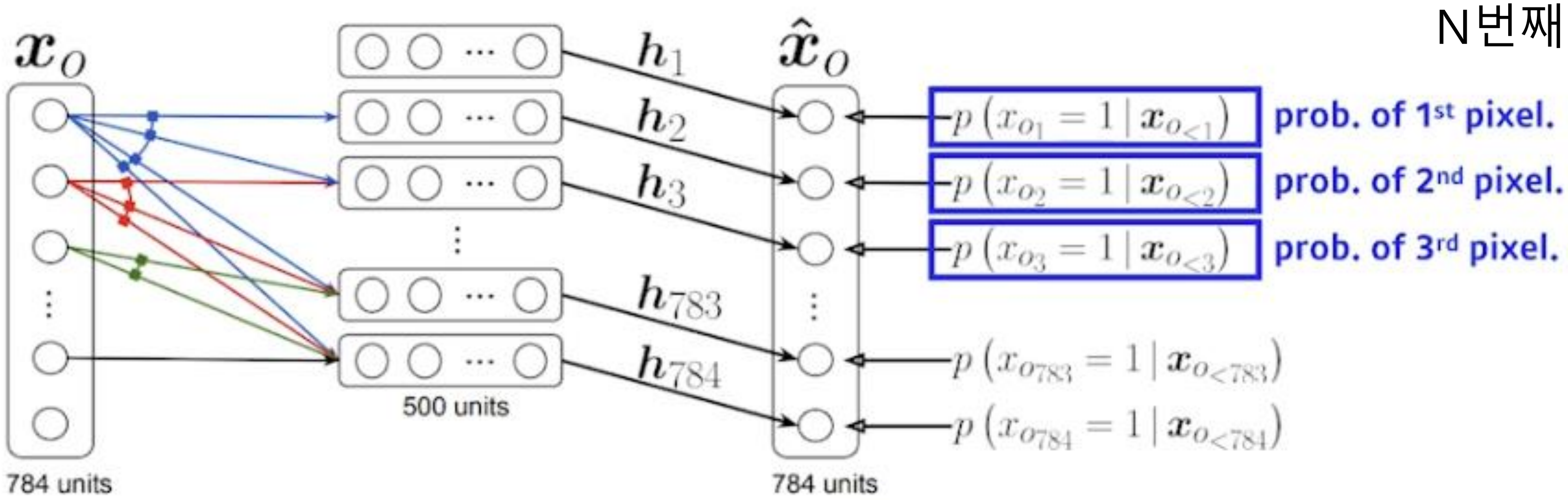
$$p(\boldsymbol{x}) = p(x_1, \ldots, x_n) = \prod_{i=1}^{n} p(x_i | x_1, \ldots, x_{i-1}).$$

$$\begin{aligned}
\mathbb{P}(A_1 \cap A_2 \cap \ldots \cap A_n) &= \mathbb{P}(A_n \mid A_1 \cap \ldots \cap A_{n-1})\, \mathbb{P}(A_1 \cap \ldots \cap A_{n-1}) \\
&= \mathbb{P}(A_n \mid A_1 \cap \ldots \cap A_{n-1})\, \mathbb{P}(A_{n-1} \mid A_1 \cap \ldots \cap A_{n-2})\, \mathbb{P}(A_1 \cap \ldots \cap A_{n-2}) \\
&= \mathbb{P}(A_n \mid A_1 \cap \ldots \cap A_{n-1})\, \mathbb{P}(A_{n-1} \mid A_1 \cap \ldots \cap A_{n-2}) \cdot \ldots \cdot \mathbb{P}(A_3 \mid A_1 \cap A_2)\mathbb{P}(A_2 \mid A_1)\mathbb{P}(A_1) \\
&= \mathbb{P}(A_1)\mathbb{P}(A_2 \mid A_1)\mathbb{P}(A_3 \mid A_1 \cap A_2) \cdot \ldots \cdot \mathbb{P}(A_n \mid A_1 \cap \cdots \cap A_{n-1}) \\
&= \prod_{k=1}^{n} \mathbb{P}(A_k \mid A_1 \cap \cdots \cap A_{k-1}) \\
&= \prod_{k=1}^{n} \mathbb{P}\left(A_k \,\middle|\, \bigcap_{j=1}^{k-1} A_j\right).
\end{aligned}$$

GANs나 energy model과 다르게 데이터의 가능도를
직접적으로 최대화 시킬 수 있다.

Autoregressive Model

N번째 pixel이 1~n-1번째 pixel들에 의존

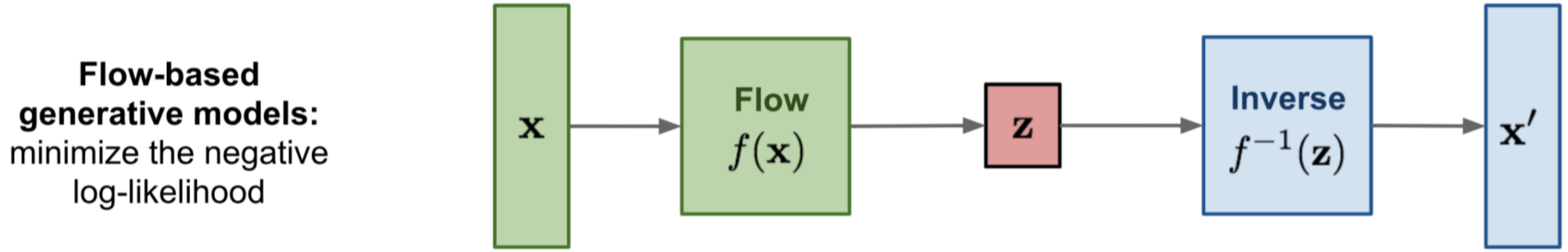


$$p(x_1, \dots, x_{784}) = p(x_1)p(x_2 \mid x_1) \cdots p(x_{784} \mid x_{1:783})$$

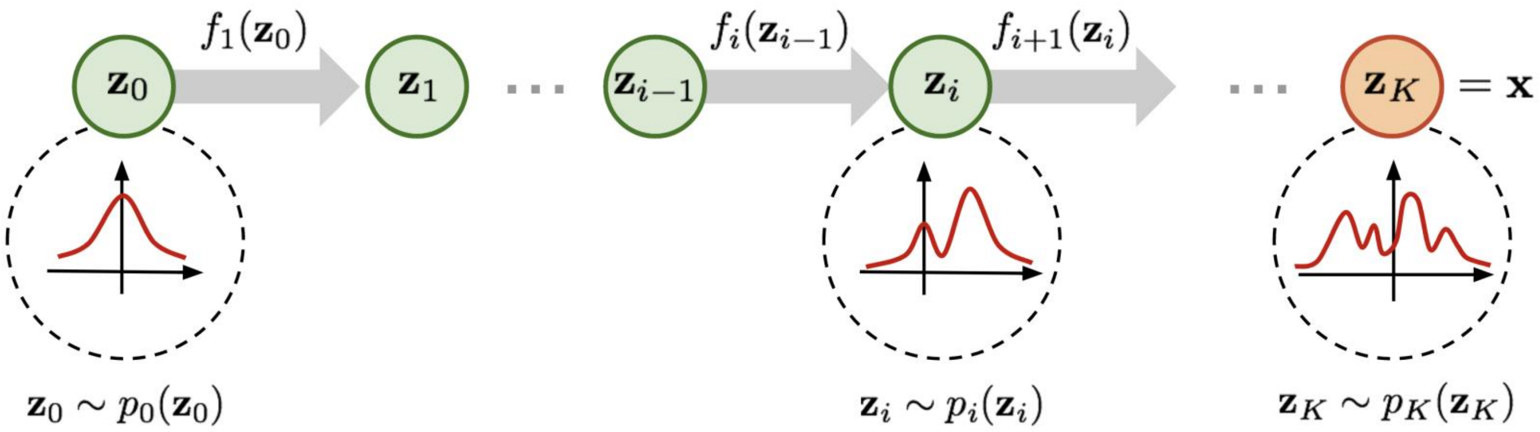
where each conditional probability $p(x_i \mid x_{1:i-1})$ is computed independently.

Neural autoregressive density estimator

Normalize flows



$z = f(x)$ 를 학습하면서, f^{-1}을 통해서 다시 x를 계산하는 것을 목표로 한다.



Normalize flows

다음과 같은 확률 밀도 함수가 있다

$$x \sim p(x)$$
$$z \sim \pi(z)$$

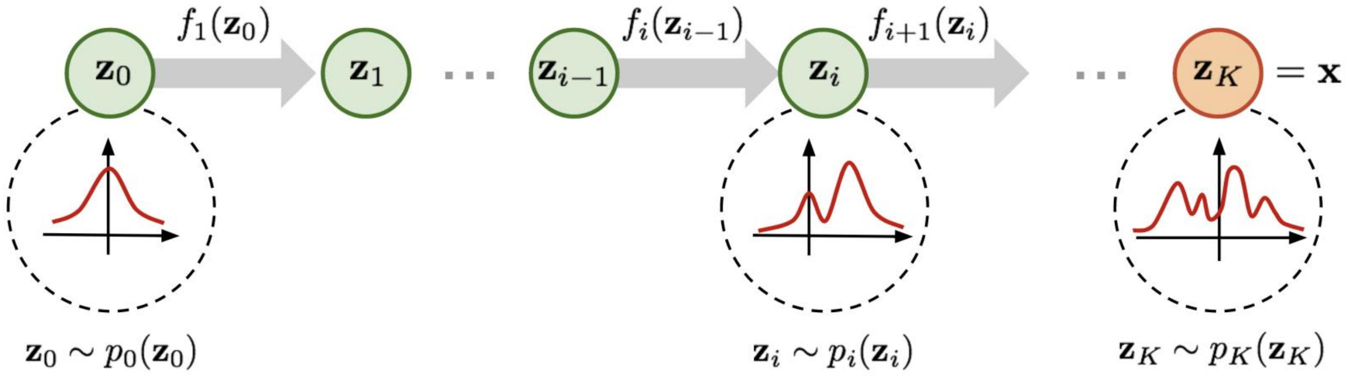
이 때, $x = f(z)$라고 할 수 있다면, 새로운 랜덤 변수를 구할 수 있을 것

$$\int p(x)dx = \int \pi(z)d\pi = 1$$

$$\int p(x)dx = \int \pi\left(f^{-1}(x)\right)df^{-1}(x) = 1$$

$$p(\mathbf{x}) = \pi(\mathbf{z})\left|\det\frac{d\mathbf{z}}{d\mathbf{x}}\right| = \pi(f^{-1}(\mathbf{x}))\left|\det\frac{df^{-1}}{d\mathbf{x}}\right|$$

Normalize flows



$$\mathbf{z}_{i-1} \sim p_{i-1}(\mathbf{z}_{i-1})$$

$$\mathbf{z}_i = f_i(\mathbf{z}_{i-1}), \text{ thus } \mathbf{z}_{i-1} = f_i^{-1}(\mathbf{z}_i)$$

$$p_i(\mathbf{z}_i) = p_{i-1}(f_i^{-1}(\mathbf{z}_i)) \left| \det \frac{d f_i^{-1}}{d\mathbf{z}_i} \right|$$

$$\begin{aligned} p_i(\mathbf{z}_i) &= p_{i-1}(f_i^{-1}(\mathbf{z}_i)) \left| \det \frac{d f_i^{-1}}{d\mathbf{z}_i} \right| \\ &= p_{i-1}(\mathbf{z}_{i-1}) \left| \det \left(\frac{d f_i}{d\mathbf{z}_{i-1}} \right)^{-1} \right| \\ &= p_{i-1}(\mathbf{z}_{i-1}) \left| \det \frac{d f_i}{d\mathbf{z}_{i-1}} \right|^{-1} \end{aligned}$$

$$\log p_i(\mathbf{z}_i) = \log p_{i-1}(\mathbf{z}_{i-1}) - \log \left| \det \frac{d f_i}{d\mathbf{z}_{i-1}} \right|$$

Normalize flows

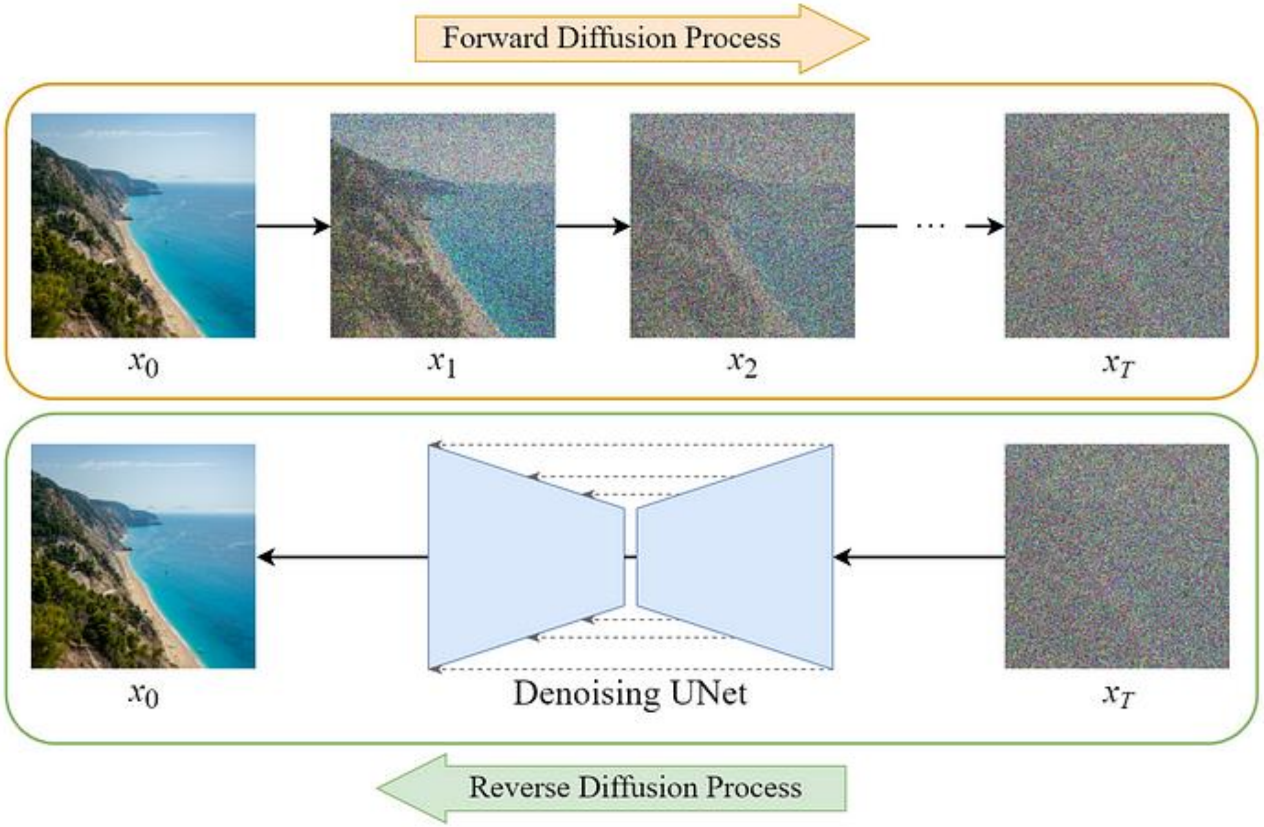
$$
\begin{aligned}
\mathbf{x} = \mathbf{z}_K &= f_K \circ f_{K-1} \circ \cdots \circ f_1(\mathbf{z}_0) \\
\log p(\mathbf{x}) = \log \pi_K(\mathbf{z}_K) &= \log \pi_{K-1}(\mathbf{z}_{K-1}) - \log \left| \det \frac{d f_K}{d \mathbf{z}_{K-1}} \right| \\
&= \log \pi_{K-2}(\mathbf{z}_{K-2}) - \log \left| \det \frac{d f_{K-1}}{d \mathbf{z}_{K-2}} \right| - \log \left| \det \frac{d f_K}{d \mathbf{z}_{K-1}} \right| \\
&= \dots \\
&= \log \pi_0(\mathbf{z}_0) - \sum_{i=1}^{K} \log \left| \det \frac{d f_i}{d \mathbf{z}_{i-1}} \right|
\end{aligned}
$$

Negative log likelihood로 만들어서 model에게 train

$$
\mathcal{L}(\mathcal{D}) = -\frac{1}{|\mathcal{D}|} \sum_{\mathbf{x} \in \mathcal{D}} \log p(\mathbf{x})
$$

Diffusion Model

- 원본 이미지에 gaussian noise를 순차적으로 추가하여 random noise로 만드는 과정 (diffusion process)에서 이 과정의 역변환을 학습하고 이를 사용하여 random noise에서 이미지를 생성함



Diffusion Model

- Diffusion process

$$q(z_t|z_{t-1}) = N(\sqrt{1-\beta_t}z_{t-1}, \beta_t I)$$

위의 식의 의미는 이전 step의 latent variable z_{t-1}에서 $\sqrt{1-\beta_t}$ 만큼의 signal을 가져오고, β_t만큼 noise를 추가해서 z_t를 계산하는 것입니다. 여기서 β_t는 noise 양으로 직접 지정하는 값입니다. Time step이 커질수록 noise를 더 큰 값으로 정합니다. $\beta_0 < \beta_1 < \cdots < \beta_T$

- Reverse diffusion process

$$p_\theta(z_{0:T}) = p(z_T)\prod_{t=1}^{T} p_\theta(z_{t-1}|z_t)$$

$p_\theta(z_{t-1}|z_t)$는 반대 방향 분포를 모델이 estimate한 것입니다. 이 분포도 Gaussian distribution입니다. 모델이 Gaussian distribution의 parameter인 평균을 학습하는 것입니다. 수식은 아래와 같습니다.

$$p_\theta(z_{t-1}|z_t) = N(z_{t-1}; \mu_\theta(z_t, t), \Sigma_\theta(z_t, t))$$

Diffusion Model

$$\mathbb{E}_q[-\log p_\theta(\mathbf{x}_0)]$$

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

위 수식이 0이 되도록 $p(\theta)$ 업데이트

참고자료

- EBM

-https://jaejunyoo.blogspot.com/2018/02/energy-based-generative-adversarial-nets-1.html

-https://www.cs.toronto.edu/~vnair/ciar/lecun1.pdf

-https://www.slideshare.net/blaswan/energy-based-models-and-boltzmann-machines

-https://www.edwith.org/deeplearningchoi/lecture/15304?isDesc=false

- VAE

-https://velog.io/@hong_journey/VAEVaritional-Auto-Encoder%EB%A5%BC-%EC%95%8C%EC%95%84%EB%B3%B4%EC%9E%90

-https://velog.io/@jochedda/%EB%94%A5%EB%9F%AC%EB%8B%9D-Autoencoder-%EA%B0%9C%EB%85%90-%EB%B0%8F-%EC%A2%85%EB%A5%98

-https://taeu.github.io/paper/deeplearning-paper-vae/

-https://velog.io/@tobigs-gm1/Variational-Autoencoder

- GAN

-https://www.samsungsds.com/kr/insights/generative-adversarial-network-ai-2.html

-https://velog.io/@hyebbly/Deep-Learning-Loss-%EC%A0%95%EB%A6%AC-1-GAN-loss

참고자료

- Autoregressive model

-https://thebook.io/080289/0450/

-https://otexts.com/fppkr/AR.html

-https://velog.io/@dldydldy75/Generative-Models-%EA%B8%B0%EB%B3%B8

- Normalizing flow

-https://devkihyun.github.io/study/Flow-based-Generative-Models-1-Normalizing-Flow/

- Diffusion model

-https://velog.io/@dongdori/Denoising-Diffusion-Probabilistic-Model2020

-https://www.lgresearch.ai/blog/view?seq=190&page=1&pageSize=12