

# Generative Models 2

실용적( practical)으로 많이 사용되는 생성모델

- VAE (Variational Auto-Encoder)

- Variational inference(VI, 변분추론)

- 목적

posterior distribution(사후분포)과 근사하는 variational distribution을 최적화하는 것

observation 주어짐 => 관심 있는 random variables의 확률 분포, 즉 posterior dist.를 찾는 것

- posterior distribution:  $p_\theta(z|x)$

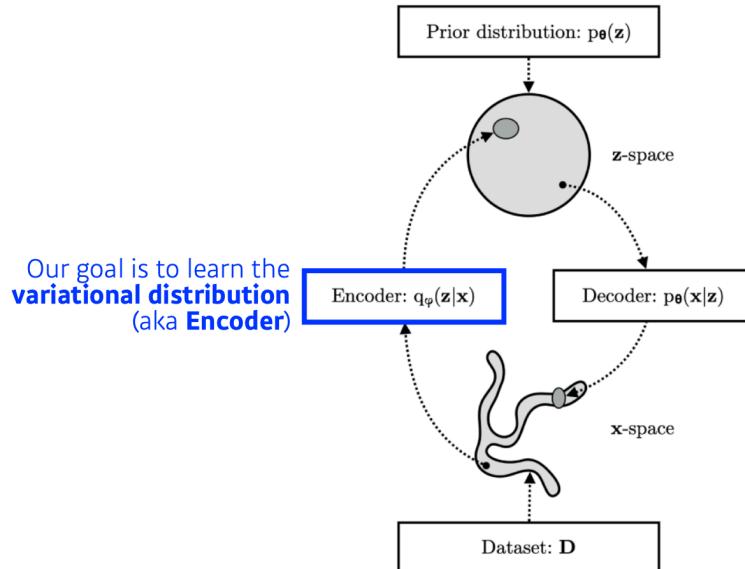
관측값  $x$ 가 주어졌을 때, 특정 확률 변수  $z$ 의 분포

- latent(잠재벡터):  $z$
    - likelihood(가능도):  $p_\theta(x|z)$
    - variational distribution:  $q_\theta(z|x)$

일반적, variational inference 정확히 구하는 것 거의 불가능

=> variational distribution 구해 사후분포에 근사하게 최적화

실제 사후분포와의 KL Divergence(쿨백 라이블러 발산) 최소화되는 variational distribution 찾기



- ELBO (Evidence Lower BOund)

$$\begin{aligned}
\ln p_\theta(D) &= \mathbb{E}_{q_\phi(z|x)} [\ln p_\theta(x)] \\
&= \mathbb{E}_{q_\phi(z|x)} \left[ \ln \frac{p_\theta(x, z)}{p_\theta(z|x)} \right] \\
&= \mathbb{E}_{q_\phi(z|x)} \left[ \ln \frac{p_\theta(x, z) q_\phi(z|x)}{q_\phi(z|x) p_\theta(z|x)} \right] \\
&= \mathbb{E}_{q_\phi(z|x)} \left[ \ln \frac{p_\theta(x, z)}{q_\phi(z|x)} \right] + \mathbb{E}_{q_\phi(z|x)} \left[ \ln \frac{q_\phi(z|x)}{p_\theta(z|x)} \right]
\end{aligned}$$

Importantly, ELBO is a tractable quantity. ↑      ↓ VI minimizes the (intractable) objective via maximizing ELBO.

$$p_\theta(D) = \text{ELBO} + D_{KL}$$

variational inference는 ELBO항과 KL Divergence 항의 합으로 유도

**ELBO법**: KL Divergence를 최소화하기 위해 반대로 ELBO항을 최대화시키는 것

=> ELBO항은 tractable(구할 수 있음) 위 방법 가능

=> Sandwitch method라도 부름

$$\begin{aligned}
\underbrace{\mathbb{E}_{q_\phi(z|x)} \left[ \ln \frac{p_\theta(x, z)}{q_\phi(z|x)} \right]}_{\text{ELBO } \uparrow} &= \int \ln \frac{p_\theta(x|z)p(z)}{q_\phi(z|x)} q_\phi(z|x) dz \\
&= \underbrace{\mathbb{E}_{q_\phi(z|x)} [p_\theta(x|z)]}_{\text{Reconstruction Term}} - \underbrace{D_{KL} (q_\phi(z|x) || p(z))}_{\text{Prior Fitting Term}}
\end{aligned}$$

This term minimizes the reconstruction loss of an auto-encoder.      This term enforces the latent distribution to be similar to the prior distribution.

#### ■ Reconstruction Term

오토인코더의 reconstruction loss 최소화

- reconstructio loss

입력값  $x$ 가 인코더를 통해 latent space(잠재공간)으로 이동, 다시 디코더 돌아오는 과정에서 일어나는 loss

=> 이상적인 샘플링 함수로부터 얼마나 잘 복원했는지 나타냄

#### ■ Prior Fitting Term

입력값  $x$ 를 latent space에 점으로 표현 시

=> 점들이 이루는 분포, latent distribution(잠재분포)로 하여금 prior distribution 비슷해지도록 강제함

이상적인 샘플링 함수의 값이 **최대한 prior(사전분포)**와 유사한 값을 만들어내도록 **condition** 부여

### • VAE 정리

- 목적

가지고 있는 이미지 입력값(관측 데이터)  $x$ 를 잘 표현할 수 있는 latent space 찾는 것

- posterior distribution 정확히 예측 못함  
=> latent space를 posterior distribution에 근사시키기 위해 variational inference 테크닉 사용  
=> 오토인코더 모델: **Variational Auto-Encoder(VAE)**

**VAE: prior distribution과 비슷한 latent space 이루도록 새로운 이미지  $x_{new}$  만듬**

=> Generative model 될 수 있음

AutoEncoder: 단순히 입력값 인코딩, latent space 보냈다가 다시 디코딩, 어떤 출력값 얻어내는 모델

=> Generative model X

## ● VAE 한계

- **intractable model**

어떤 입력 주어짐 => likelihood 측정 판별 X

=> explicit 모델 X

- **prior fitting term 미분가능**

KL Divergence는 그 자체로 적분 들어있음, 적분이 intractable 경우 많음

=> Gaussian prior distribution 제외하고, closed form 잘 안 나옴

- closed form: 수학적으로 유한한 term, 표현할 수 있는 항(식)

But, SGD/Adam 같은 Optimizer 사용하려면 KL Divergence 포함한 prior fitting term 미분가능

=> 대부분 VAE: closed form 나오는 분포, Gaussian prior distribution 사용

- 일반적으로 **isotropic Gaussian** 사용

모든 output dimension이 independent한 Guassian Distribution

Gaussian prior distribution일 경우,

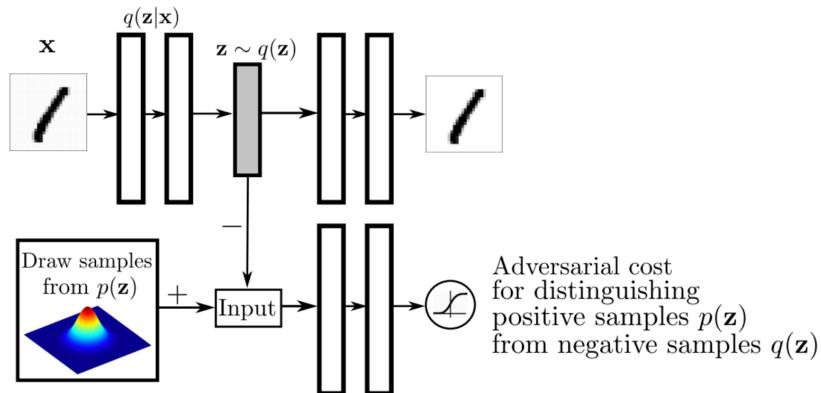
$$D_{KL}(q_\phi(Z|x)||N(0, 1)) = \frac{1}{2} \sum_{i=1}^D (\sigma_{z_i}^2 + \mu_{z_i}^2 - \ln(\sigma_{z_i}^2) - 1)$$

loss function에 집어넣어 학습, 원하는 결과 나옴

## ● Adversarial auto-encoder(AAE)

Gaussian Distribution이 아닌 다른 prior distribution 활용

VAE의 prior fitting term을 GAN objective으로 바꾼 것



- GAN을 사용해 latent 분포들 사이의 분포 맞춰줌  
○ VAE보다 대부분 성능 잘 나옴

## ● GAN(Generative Adversarial Network)

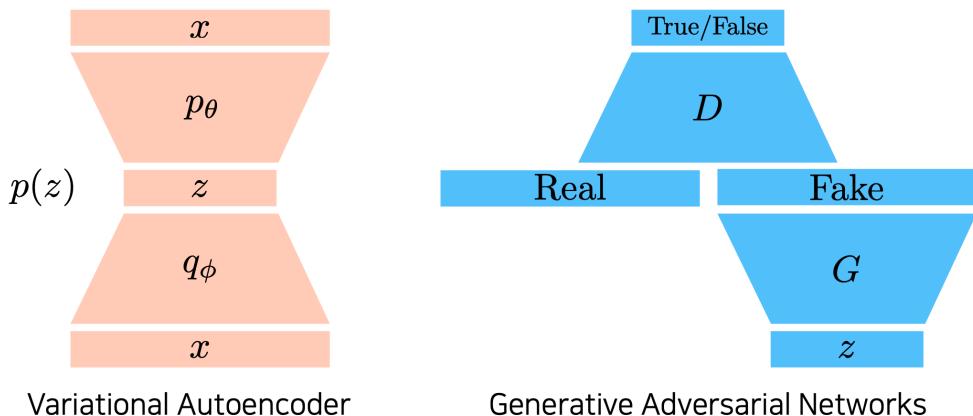
Generator 성능 높이는 것

Generator 학습시키는 Discriminator 점점 좋아짐

$$\min_{G} \max_{D} V(D, G) = \mathbb{E}_{x \sim p_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$$

- 우변: loss, 생성값과 실제값의 오차

Generator: loss 최소화 / Discriminator: loss 최대화



- VAE

- 학습과정

입력값  $x$ 를 인코더를 통해 latent vector  $z$ 로 만듬

=>  $z$ 를 다시 디코딩, 원래  $x$ 로 복원시키도록 인코더와 디코더의 파라미터 학습

- 생성과정

latent distribution  $p_\theta(z)$  sampling한 뒤, 디코딩하여  $x_{new}$  출력

- GAN

- 학습과정

latent distribution  $z$ 에서 출발, generator 통해 Fake 만듬

discriminator 기준의 레이블과 Fake 이미지 비교 판독, 분류기 학습

Generator는 discriminator가 Fake 이미지에 대해 True 나오도록 위조기 update하여 학습

- **GAN 목표**

discriminator의 알고리즘 수식

$$\max_{D} V(G, D) = \mathbb{E}_{x \sim p_{data}} [\log D(x)] + \mathbb{E}_{x \sim p_G} [\log(1 - D(x))]$$

Generator가 고정, Discriminator를 optimization(최적화)시키는 form

$$D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_G(x)}$$

Generator 알고리즘: 동일한 loss에 대해, discriminator 최대화, generator 최소화

$$\min_{G} V(G, D) = \mathbb{E}_{x \sim p_{data}} [\log D(x)] + \mathbb{E}_{x \sim p_G} [\log(1 - D(x))]$$

optimal discriminator를 generator 수식에 적용 시,

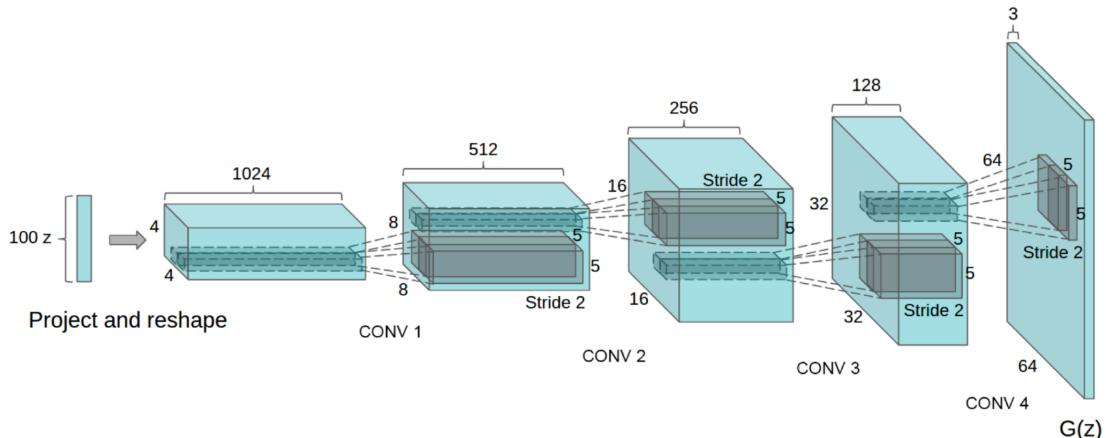
$$\begin{aligned}
V(G, D_G^*(x)) &= E_{x \sim p_{\text{data}}} \left[ \log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)} \right] + E_{x \sim p_G} \left[ \log \frac{p_G(x)}{p_{\text{data}}(x) + p_G(x)} \right] \\
&= E_{x \sim p_{\text{data}}} \left[ \log \frac{p_{\text{data}}(x)}{\frac{p_{\text{data}}(x) + p_G(x)}{2}} \right] + E_{x \sim p_G} \left[ \log \frac{p_G(x)}{\frac{p_{\text{data}}(x) + p_G(x)}{2}} \right] - \log 4 \\
&= \underbrace{D_{KL} \left[ p_{\text{data}}, \frac{p_{\text{data}} + p_G}{2} \right] + D_{KL} \left[ p_G, \frac{p_{\text{data}} + p_G}{2} \right]}_{2 \times \text{Jenson-Shannon Divergence (JSD)}} - \log 4 \\
&= 2D_{\text{JSD}}[p_{\text{data}}, p_G] - \log 4
\end{aligned}$$

=> GAN 목적: 실제 데이터의 분포와 학습한 데이터의 분포 사이에 jenson-Shannon Divergence(JSD) 최소화

=> Discriminator가 optimal하다는 가정에 generator 알고리즘 대입

- discriminator가 optimal discriminator에 수렴 보장 어려움, generator 위 식처럼 전개 못함  
=> 이론적 타당, 현실적 JSD 줄이는 방식 사용 어려움
- DCGAN(Deep Convolutional GAN)**

GAN 모델을 이미지 도메인으로 개량

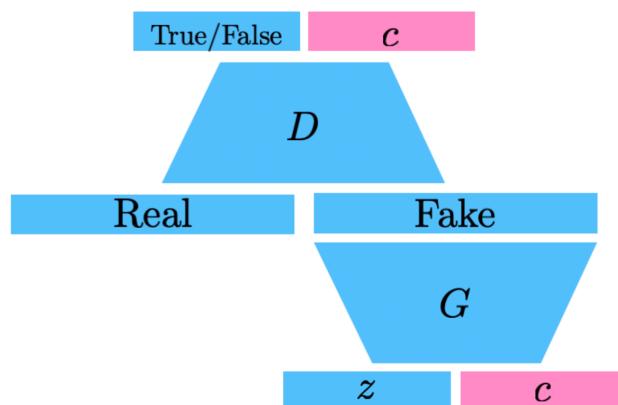


generator에 deconvolution 활용, discriminator에 convolution 연산 수행

=> 벡터 늘림

알고리즘 개량 X, 하이퍼파라미터/테크닉에 대한 실험적 결과 수록

- Info-GAN**



단순히 True/False 뿐만 아니라, class를 랜덤하게 같이 넣어줘서 학습

단순하게 z를 이용해 이미지뿐만 아니라, c라는 클래스 집어넣은 방식

- $c$ 라는 원-핫 벡터 제공, generator의 multi-model distribution 학습 도움

- **Text2Image**

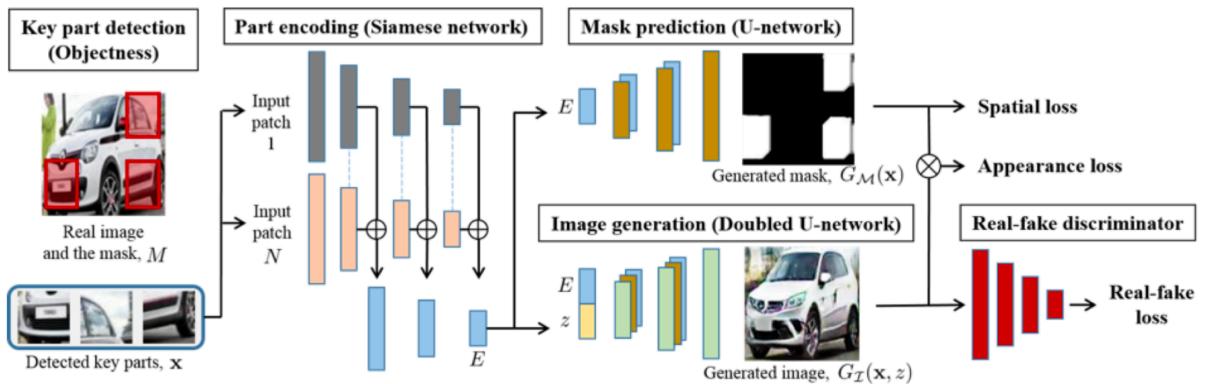


문장 => 이미지 만드는 모델

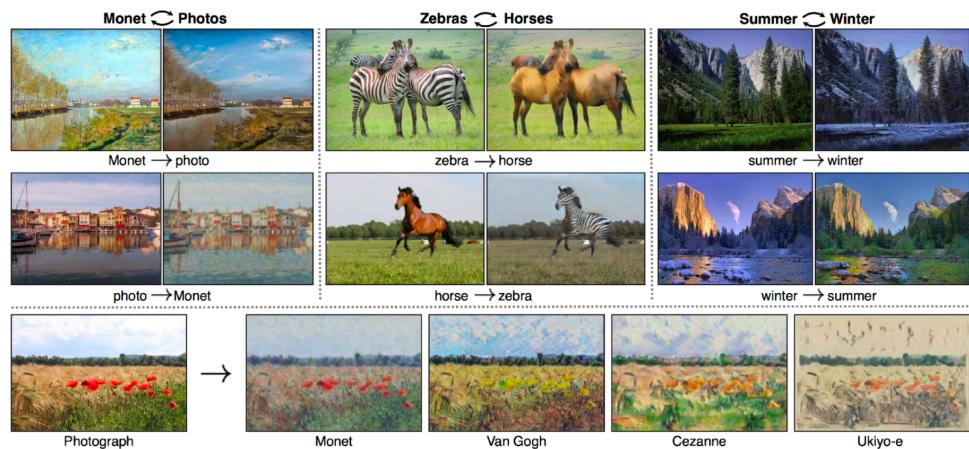
문장 입력으로 받아, Conditional GAN 통해 이미지 출력

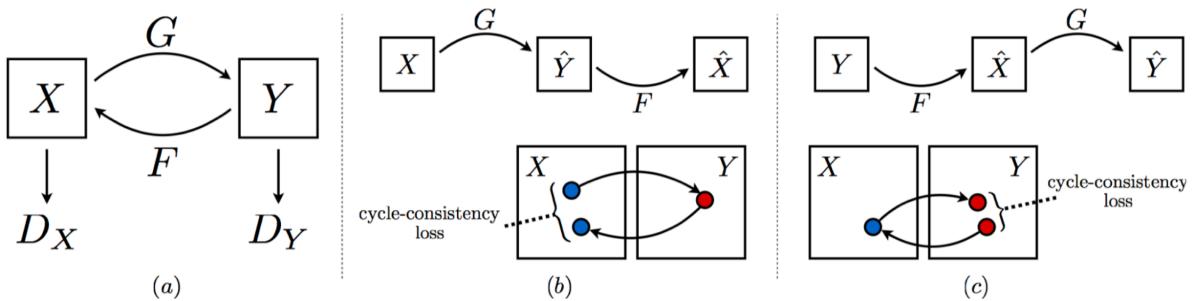
OpenAI에서 제공하는 DALL-E의 원조 격

- **Puzzle-GAN**



- **CycleGAN**





이미지 사이의 도메인 바꿀 수 있는 모델

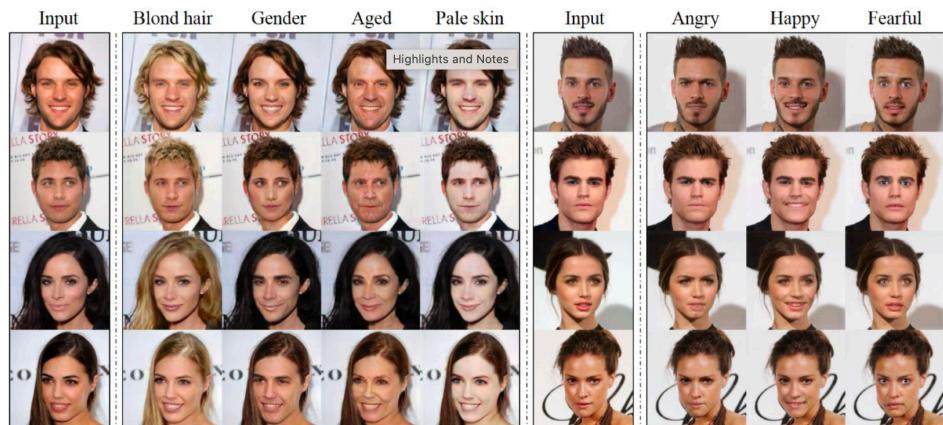
이미지 주어지면, 이미지를 주어진 조건에 알맞은 형태로 변형

### Cycle-consistency loss 알고리즘 활용

- 임의의 말 이미지들과 임의의 얼룩말 이미지 잔뜩 모아둠 => 이를 이용해 하나의 이미지를 다른 도메인 이미지로 그냥 변형시켜줌

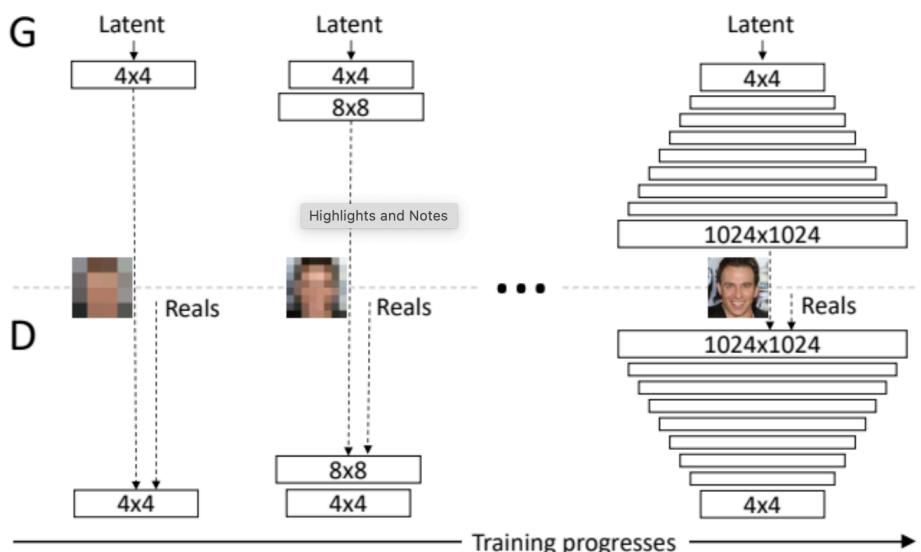
=> GAN 모델 2개 사용

- Star-GAN



단순한 이미지를 원하는 형태로 control할 수 있게 만들어주는 모델

- Progressive-GAN



고해상도 이미지 잘 만들 수 있는 GAN

$4 \times 4$  같이 작은 사이즈에서부터 시작, HD size까지 픽셀을 키워가며 해상도 점진적으로 상승(progressive)

