

Supervised Learning

- Data with labels (x, y)
- Goals: Mapping ($x \rightarrow y$)

Unsupervised Learning

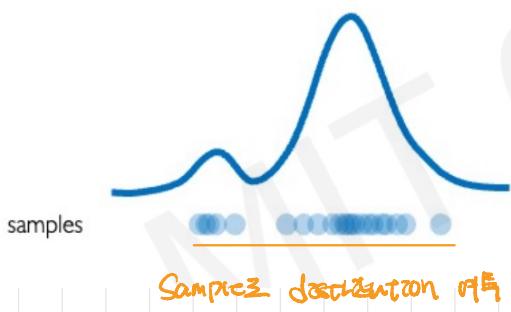
- Data x with no labels
- Goals: Learn the hidden or underlying structure of the data

Generative Modeling

- 특정 distribution의 training sample을 통해 해당 distribution을 학습하는 방법
- Unsupervised Learning의 일종

두 가지 흐름을 지닌다.

1. Density Estimation



2. Sample Generation



Training data $\sim P_{\text{data}}(x)$ Generated $\sim P_{\text{model}}(x)$
Samples distribution 예측, Samples 예측
Instance 예측

Our Goals → 어떤 Data와 같은 Distribution을 예측할까?

Generative Model의 장점

1. Debiasing

→ Training data의 feature를 알아서 학습하기 때문에 Training data의 내재된 Bias를 제거할 수 있다.

2. Outlier detection

→ Training data distribution과는 확률화되어 다른
outlier (Low probability)를 원래하기 쉽다.

Latent Variable

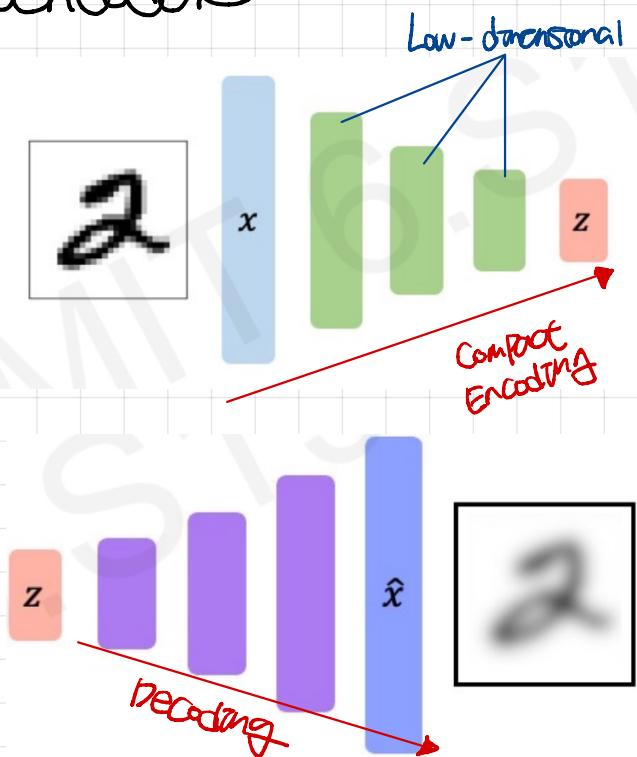
→ 각각 모체를 두는 모체, 데이터를 생성하는데 중요한 역할을 하는 변수

↳ Generative model을 통해 학습 가능하다.

Latent Variable models

1. Autoencoders and Variational Autoencoders (VAEs)
2. Generative Adversarial Networks (GANs)

Autoencoders



Raw data를 low-dimensional latent space로
통해 \rightarrow 로 만든다:

→ More efficient!

- 같은 Data를 저차원으로 만들 수 있다.

- Low memory and low memory space

→ \rightarrow 로
Encoder Model을 활용하여 reconstruct original data → 원래의 data와 유사하게 출력하기 때문에
 \rightarrow 이라고 표시한다.

Can we use fully-connected layers?

→ Reconstruct 및 다른 브론트 classification에 유용
암호화 된 특징을 해석해 class label 결정한다. 또는 이전에는
Decoding이라고 불 수 있다. Generative model이
차이가 있다.

$$\text{Loss: } L(x, \hat{x}) = \|x - \hat{x}\|^2$$

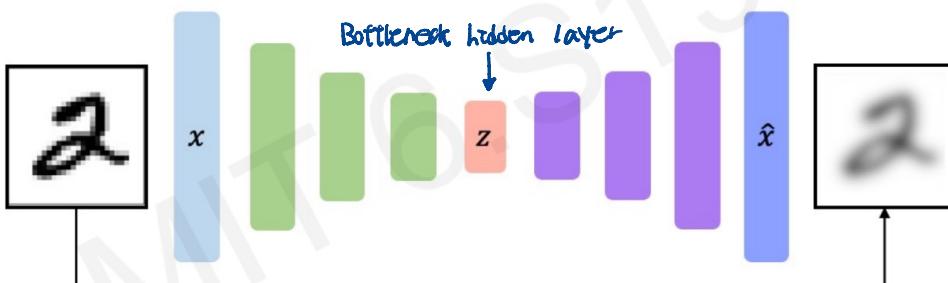
Objective: x 와 \hat{x} 를 최대한 유사하게 만들자!

→ Class label이 필요없다. ← Unsupervised learning.

Reconstruction quality

- Dimension of latent space \uparrow : 암호화 정보 \uparrow \rightarrow 재구성률 \uparrow , overfitting 가능성 \uparrow
- Dimension of latent space \downarrow : 암호화 정보 \downarrow \rightarrow 재구성률 \downarrow , overfitting 가능성 \downarrow

Autoencoding = Automatically encoding data
self-encoding



$$L(x, \hat{x}) = \|x - \hat{x}\|^2$$

Loss function doesn't
use any labels!!

→ To minimize loss, z는 최대한 많은 정보를 저장해야 한다.

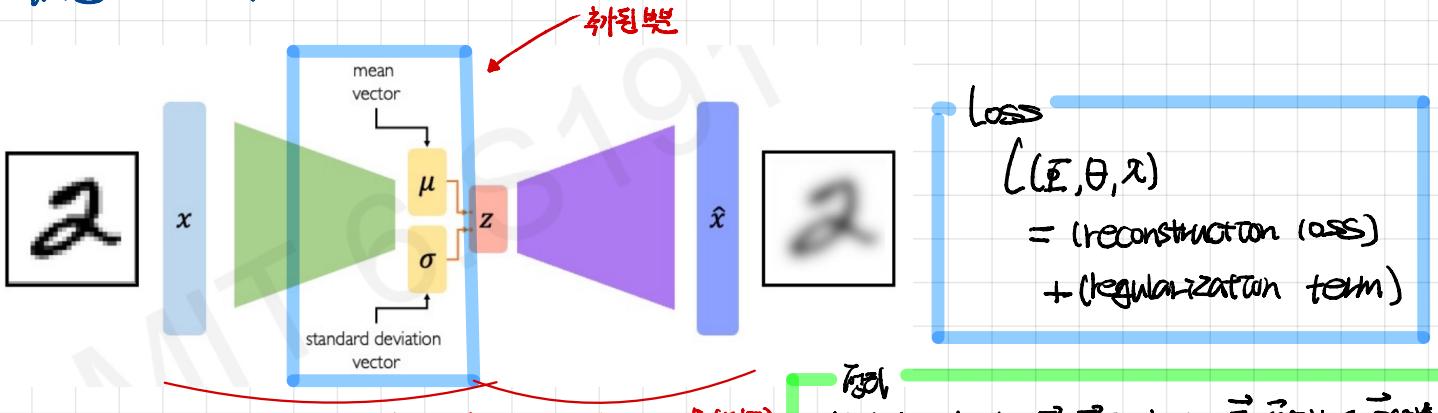
Z : Latent Space ($= P(z|x)$)

→ 차원을 줄여주는 공간들

Variational Autoencoders (VAEs)

→ 기존 Autoencoder에 기반하여, 자주 좋은 성능이 아니라 가끔 GAN처럼

유사한 새로운 Instance를 생성할 수 있다.



Tip!

VAE의 Encoder는 μ, σ 를 출력해 $\vec{z} \sim \mathcal{N}(\mu, \sigma^2)$ 로 random sampling을 통해, 무작위로 랜덤하게 \vec{z} 를抽選해 후보들로 Random Sampling을 한다. 후보들 중에서 prior에 의해 확률이 높은 것만 선택한다.

Reconstruction loss = $\|x - \hat{x}\|^2$ ex. log-likelihood : $\log p(x|\hat{x})$

KL-divergence: 두 분포 사이 거리 / $q_p(z|x)$ 와 $p(z)$ 는 같은 특징

Regularization term = $D(q_p(z|x) \| P(z))$: 입력 x 에 대해 추운 distribution $P(z|x)$ 가
distribution prior $P(z)$ 와 얼마나 다른가

→ Regularization term은 Loss에 추가되면서 $P(z)$ 와 너무 다르지 않도록 강제된다.

→ $P(z)$ 는 Bottleneck hidden layer가 아니라 강제하기로 하는 distribution이다.

일반적으로 $P(z) \sim \mathcal{N}(0, I)$ 사용 : $P(z|x)$ 가 정밀하고 간단하도록 한다.
→ 차이가 인식하기 어렵다. (이미지가 흐릿함)

$$\text{- 이 경우 } D = -\frac{1}{2} \sum_{i=1}^n (\mu_i + \sigma_i^2 - 1 - \log 2\pi)$$

$L(\theta, \hat{x}) = \text{(reconstruction loss)} + \text{(regularization term)}$

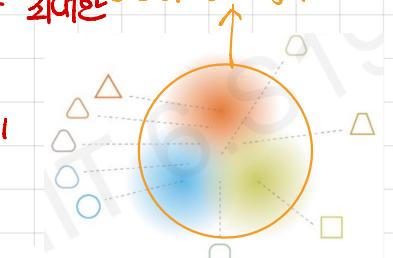
$\hat{x} \in x$ 의 확장집합

구조화된 드롭

$P(z|x)$ 는 x 로 조정된
구조화된 드롭의 의미하는 빈
공간에서 decoding 시 가져온다.

→ Autoencoding 같은 단순히 Reconstruction loss만 있다면 x 와 \hat{x} 가 차이를
나누어 Model이 x 를 암기할 수 있다.

→ Regularization term은 모순 암기가 아니라 구조화된 Latent Variable을
이해 허용하도록 할 수 있다.



Intuition on regularization and the Normal prior

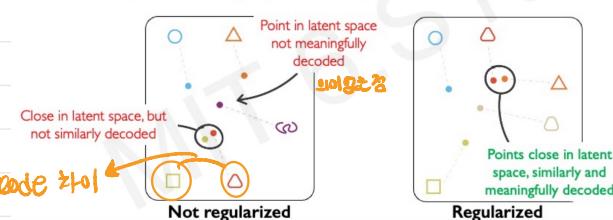
What properties do we want to achieve from regularization? 🤔

Latent space가→ decoding 후로 기여를

1. **Continuity:** points that are close in latent space → similar content after decoding
2. **Completeness:** sampling from latent space → "meaningful" content after decoding

이면 sample이면 → 의미있게 되는 드롭

Latent space: 데이터의 핵심 특성을 추적하여 표현한 공간



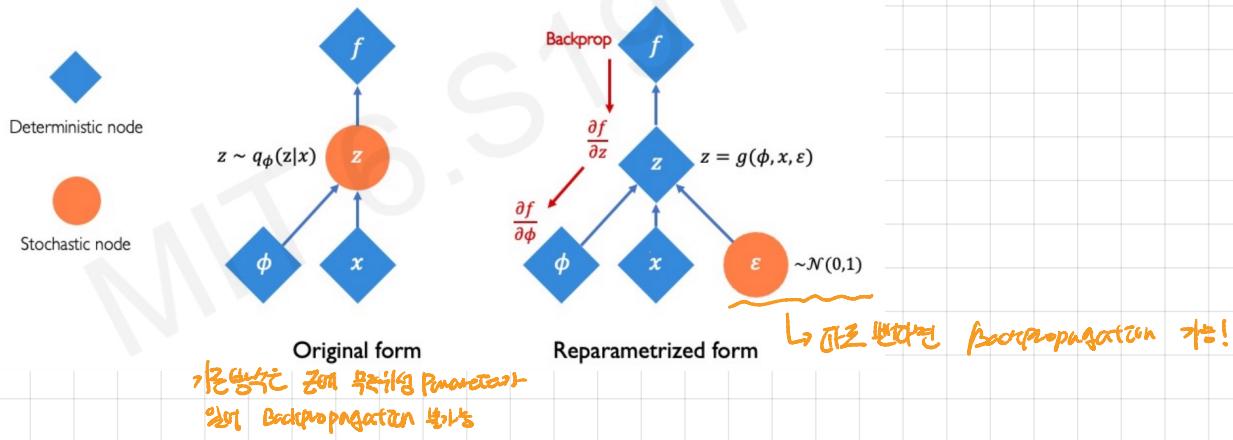
Reparametrizing the Sampling layer

Key Point :

$$z = \underbrace{U \cdot V}_{\text{Fixed by Encoder}} \cdot \varepsilon$$

Random constant from the prior distribution

Backpropagation

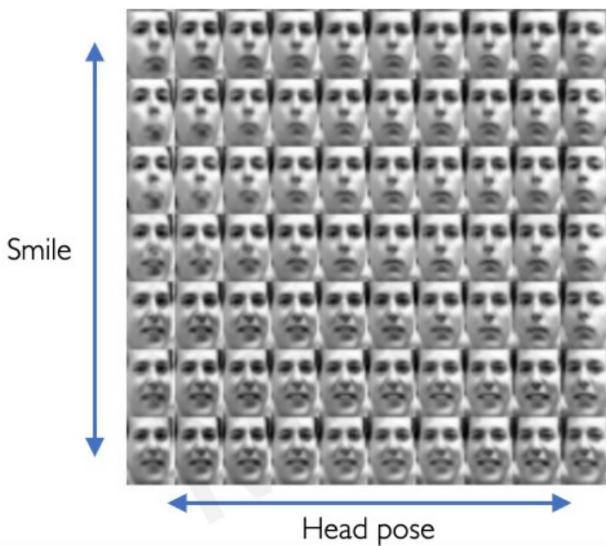


Latent perturbation

Goal: Latent Variables 를 조작한 서로 독립이 되도록 하자

→ 다른 것을 증하고 특정 하나만 바꾸었을 때, 해당 속성 하나만 바뀌도록 하기 위해서!

Example: Independent Latent Variables



For Latent space disentanglement : β -VAE

$$L(\theta, \phi, x, z, \beta) = E_{q_\phi(z|x)} [\log p_\theta(x|z)] - \boxed{\beta} D_{KL}(q_\phi(z|x) || P(z))$$

Loss function Log likelihood $\beta > 1$

Priority $\beta = I(\text{дано} | \text{получено})$ 이라면
 β 를 증가로써 각 조건이 서로
독립이 되도록 할 수 있다.

Ex) $P(z) \sim \mathcal{N}(0, I)$

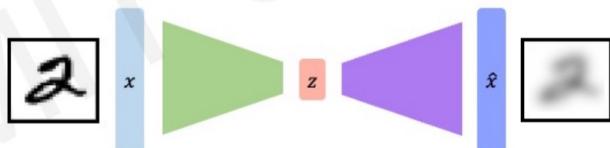
Disentangled latent representation

각 축이 하나의 구체적인 의미를 나타낸다.

- Independence
- Axis alignment : 정렬
- Magnitude : 크기 일치

Summary

1. Compress representation of world to something we can use to learn \rightarrow x 을 학습 가능한 표현으로 압축
2. Reconstruction allows for unsupervised learning (no labels!) $x \rightarrow z \rightarrow x$
3. Reparameterization trick to train end-to-end
4. Interpret hidden latent variables using perturbation \rightarrow z 를 바꾸어보면 그 의미를 알 수 있음
5. Generating new examples \rightarrow ~~latent space가 prior distribution과 같은 형태를 갖다.~~ Latent space가 prior distribution과 같은 형태를 갖다. \rightarrow Prior distribution에서 Sampling하여 새로운 유사한 데이터를 만들 수 있다.

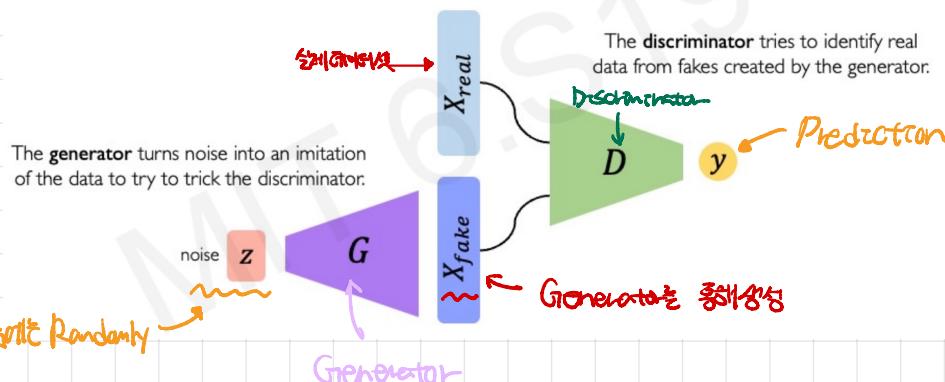


VAE의 문제점: Prior distribution이 놓쳤으면 바로 생성하기 어렵거나 즉시 새로운 이미지를 만들기 어렵다.

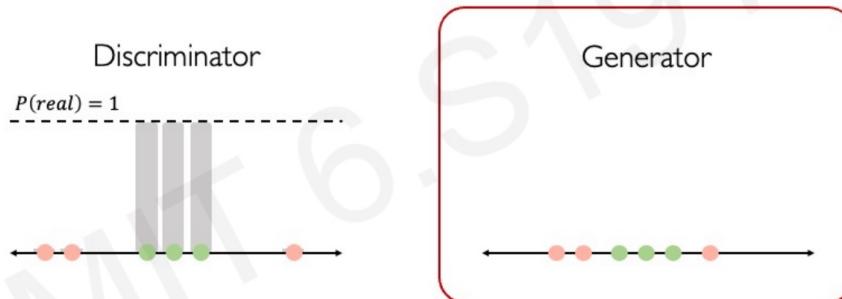
Solution: Generative Adversarial Networks (GANs)

\rightarrow Complex Prior distribution이 있는 흐름을 수 있는 흐름

Generative Adversarial Networks (GANs)



학습 과정



\rightarrow Generator가 자체적인 G가지 X_{real} 의 Distribution을 흡수!
 \rightarrow Generator와 Discriminator 경쟁

1. 초기에는 무작위 분포 (e.g. normal distribution)에서 아무 이미지를 가정한다.
2. Discriminator는 X_{real} 과 X_{fake} 를 무작위로 선택하여 확률적인 판별을 한다.
3. Generator는 Discriminator의 출력을 통해 X_{fake} 를 X_{real} 과 더 잘 가깝게 만든다.
4. (2) ~ (3)의 과정을 반복한다.

* 학습된 분포가 실제 이미지보다 무작위로 희미

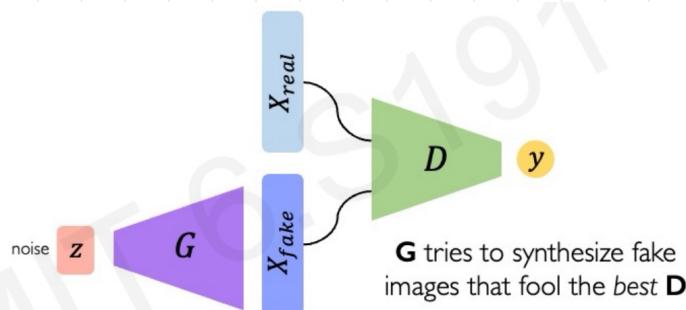
\Rightarrow Random한 값을 통해 유사한 새로운 데이터를 만들 수 있다.

\rightarrow 예상치 못한 X_{fake} 는 실제 이미지랑 굉장히 유사할 것이다.

\rightarrow Generator가 Discriminator를 속이려고 노력하는 과정에서 유사한 이미지가 생겨난다.

= 적대적 경쟁

Training GANs



$D(x)$ x가 진짜라는 확률하면 1
x가 가짜라고 판단하면 0

$$\text{Loss: } \arg \min_G \max_D L(D, G) = E_{x \sim P_{\text{data}(x)}} [\log D(x)] + E_{z \sim P(z)} [\log (1 - D(G(z)))]$$

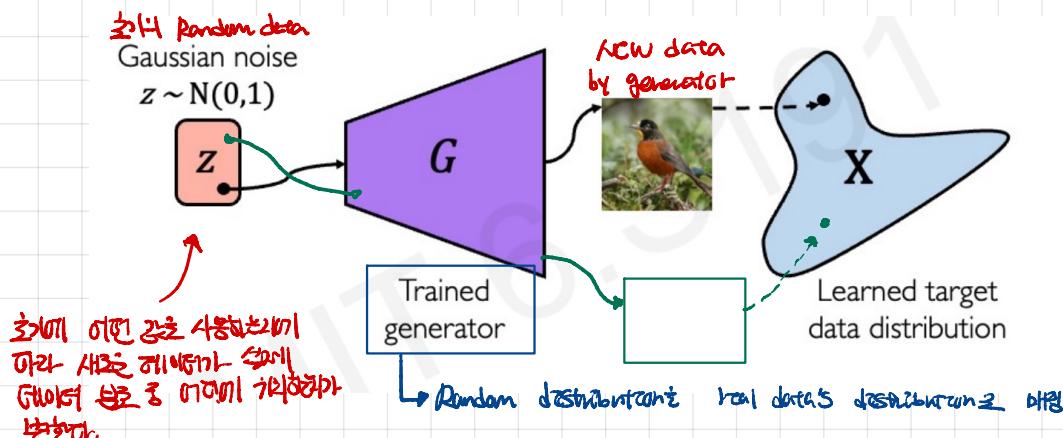
Real data를 판별 Fake data를 판별

1. Max : Discriminator 입장에서 최대의 손실 $D(x)=1$ and $D(G(z))=0 \rightarrow \text{Loss} = 0$ 이다.
2. Min : Generator 입장에서 최대의 손실 $x(\text{real data})$ 의 대비 흔적을 하지 못하도록 유도하고, $D(G(z))=1 \rightarrow \text{Loss} = -\infty$ 로 최대화된다.

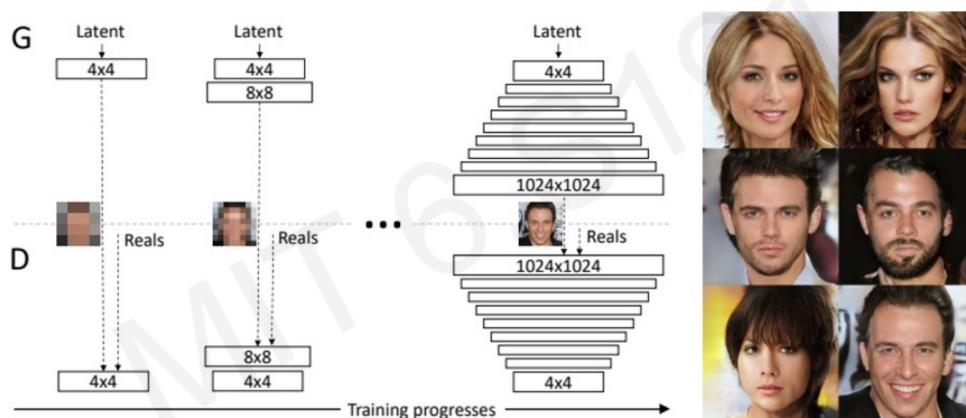
After training

→ Generator 부분만 이용하여 새로운 데이터를 생성한다.

GANs의 응용



Advanced GANs 1: Progressive GANs

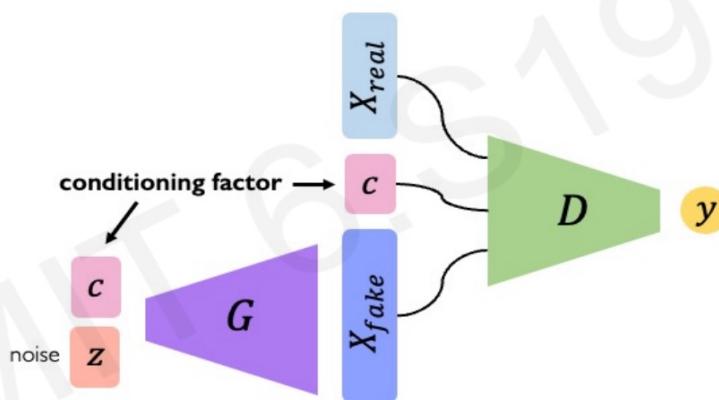


Generated images + real images are used to calculate loss function

전체 학습 과정 ← → 디테일

Advanced GANs 2 : Conditional GANs

- Conditioning factor를 추가하여 Input: 원본 사진, 조건 사진이 다른
Output: 원본 + 내용의 사진 / 조건 사진 등은 볼 수 있다.



↳ Paired data

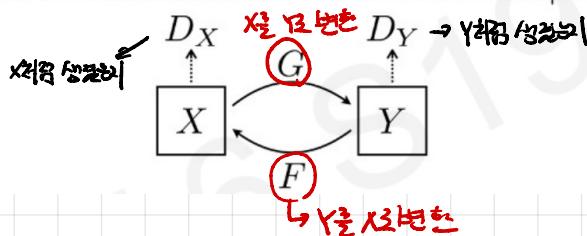
→ 서로 같은 데이터에 대한 조건付け

이후 Transformation!

$$\text{Loss} = E[\log D(x, c)] - E[\log(1 - D(G(z, c), c))]$$

Advanced GANs 3 : CycleGANs

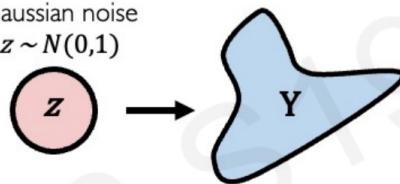
CycleGAN learns transformations across domains with unpaired data.



→ 서로 다른 영역을 바꿀 수 있다!

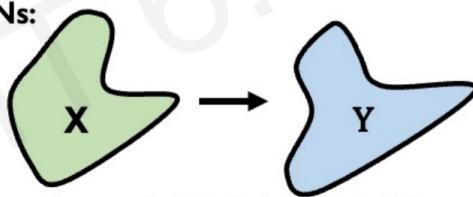
Distribution transformations

GANs: Gaussian noise $z \sim N(0,1)$



Gaussian noise \rightarrow target data manifold

CycleGANs:



data manifold $X \rightarrow$ data manifold Y

한국어: Domain-1 이미지를 Domain-2로 변환

Unpaired data transformation 문제

$$\text{Loss} = L_{\text{GAN}}(G, D_Y) + L_{\text{GAN}}(G, D_X) + \lambda L_{\text{cycle}}$$

Diffusion Model, VAE, GANs 등 가진다

→ In Lecture 17