

# CV팀

CV팀  
강서진  
송다은  
최종혁  
이나연  
이민호

# INDEX

---

1. Neural Style Transfer
2. VAE
3. GAN
4. Conditional GAN
5. Image-to-Image Translation

# 1

## Neural Style Transfer

# 1

# Neural Style Transfer

## Neural Style Transfer

### Neural Style Transfer

딥러닝을 사용하여 **이미지의 스타일을 변경**하는 방법



Content image



Style image

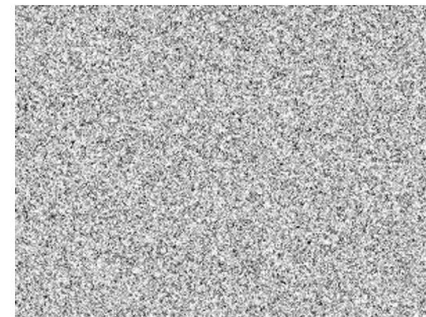
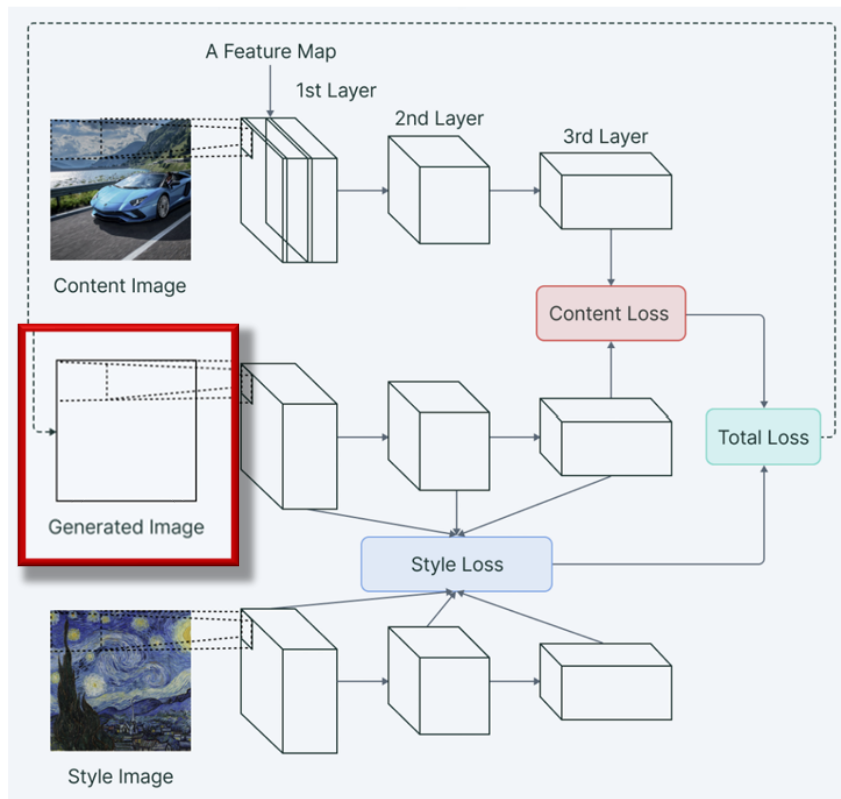


Content와 Style의 결합

## 1

# Neural Style Transfer

## Neural Style Transfer | 구조



**Random Noise**

시작되는 이미지를 나타냄



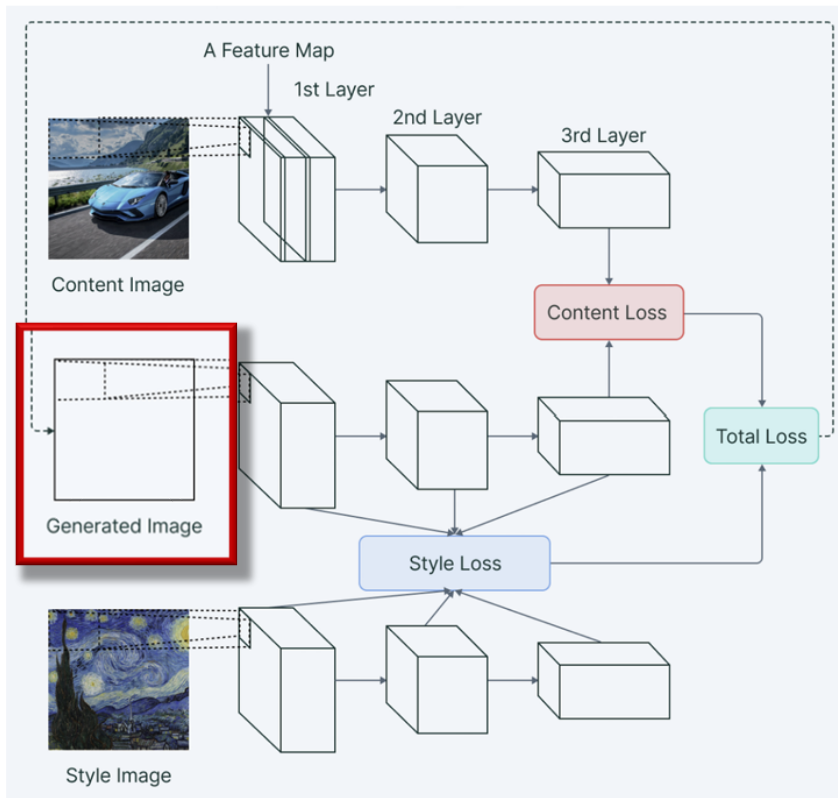
**Generated Image**

랜덤 노이즈가 학습을 거쳐  
Generated Image로 업데이트

## 1

# Neural Style Transfer

## Neural Style Transfer | 구조



랜덤 노이즈를 업데이트하면  
점차 그리고자 하는 그림이 되어 감

업데이트를 위해서는

Loss Function이 필요!

Generated Image로

업데이트 됨

## 1

# Neural Style Transfer

## Neural Style Transfer | Loss function

### Loss Function

$$L(G) = \alpha L_{Content}(C, G) + \beta L_{style}(S, G)$$

⋮

$G$ : generated image

$C$ : content image

$S$ : style image

$\alpha, \beta$ : weighting factor



Style loss와 Content loss의 결합으로 이루어짐

# 1

## Neural Style Transfer

### Extracting Features

이미지의 특징을 뽑기 위해 VGG-19 등  
사전 학습된 모델 사용



모델 안의 여러 층들 중  
Content Feature와 Style Feature를 뽑을 **특정한 층을 선택**하면 됨





## 1

## Neural Style Transfer

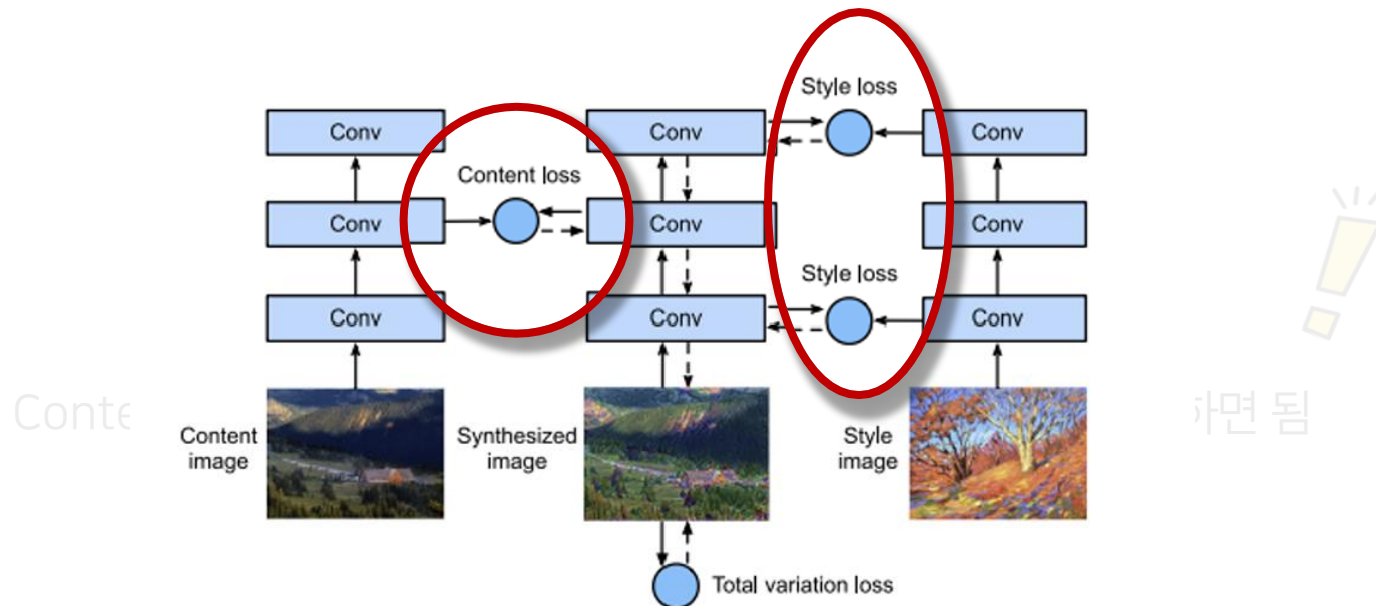
## Extracting Features

## Content Layer

출력층 쪽에 더 가까운  
레이어 선택

## Style Layer

Local, Global Style  
모두 추출하기 위해 서로 다른  
레이어의 Output들 선택



## 1

## Neural Style Transfer

## Content Loss

## Content Loss

$$L_{content}(C, G) = \frac{1}{2} \sum_{i,j} (F_{ij}^l - P_{ij}^l)^2$$

⋮

$F^l$ :  $l$ 층의 Feature map of  $G$

$P^l$ :  $l$ 층의 Feature map of  $C$

$i$ : Channel 수

$j$ : Height X Width (각 Pixel)

## 1

## Neural Style Transfer

## Content Loss

## Content Loss

$$L_{content}(C, G) = \frac{1}{2} \sum_{i,j} (F_{ij}^l - P_{ij}^l)^2$$



$F^l$ :  $l$ 층의 Feature map of  $G$

Content Feature를 뽑겠다고 정해둔  $l$ 번째 layer의 Feature map의 MSE,

즉 각 픽셀의 오차 제곱합을 구하는 식

$j$ : Height X Width (각 Pixel)

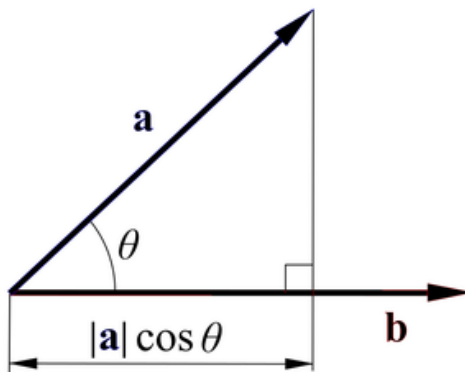
## 1

# Neural Style Transfer

## Style Loss | Gram Matrix

### Gram Matrix

Style 벡터들 간의 **내적**을 통해 그들의 **상관관계**를 나타내는 행렬



벡터의 내적을 직관적으로 생각해보면,  
두 벡터끼리 유사성이 높을수록 내적 값이 커지게 됨

## 1

# Neural Style Transfer

## Style Loss | Gram Matrix

사전학습 모델에 Style image 를 input 으로 넣어 Feature Map 생성



Feature Map으로부터 flattened feature vector 형성



위 벡터를 내적하여 만든 행렬은  
그 Feature들 간의 관계, 즉 유사성에 대한 정보를 제공

## 1

# Neural Style Transfer

## Style Loss | Gram Matrix

### Gram Matrix

$$G = XX^T \in \mathbb{R}^{C \times C}$$

⋮

이 행렬의 element  $x_{ij}$ 는  $i$ 번째 채널과  $j$ 번째 채널 간의 style vector 간 내적이므로, 스타일 상관관계를 의미



즉, Gram matrix 는 스타일 레이어 출력에서  
서로 다른 채널 간의 특징이 얼마나 비슷한 지를 나타내는 방식

## 1

## Neural Style Transfer

## Style Loss | Gram Matrix

## Gram Matrix

$$G = XX^T \in \mathbb{R}^{C \times C}$$

## 상관계수의 정의

$$r_{X,Y} = \frac{1}{N-1} x^T y$$

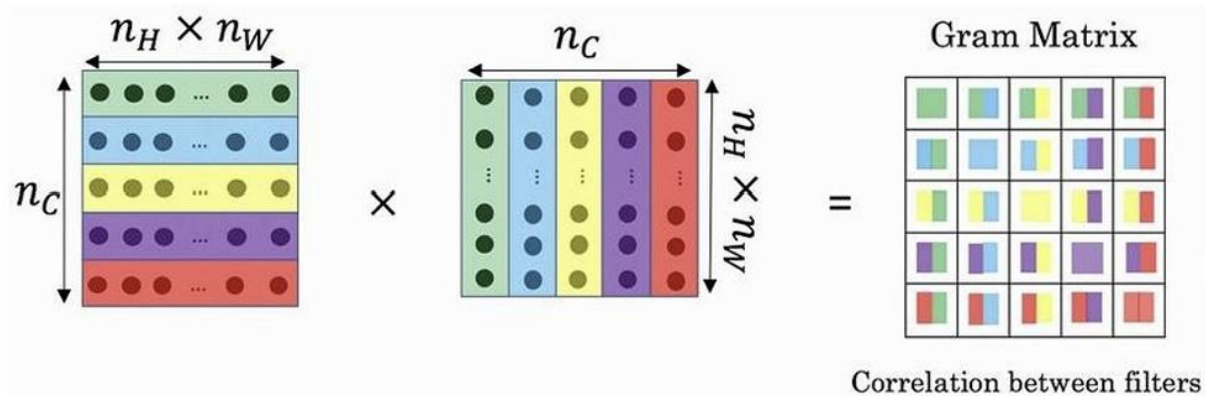


즉, Gram 행렬의 정의와 비교했을 때 형태가 매우 비슷함  
Gram Matrix는 표준화하지 않은 상관행렬의 개념!

## 1

# Neural Style Transfer

## Style Loss | Gram Matrix



Feature map의 채널 하나하나가 결국 이미지의 feature가 됨



Feature들의 상관관계가 바로 Style



## 1

# Neural Style Transfer

## Style Loss | Gram Matrix

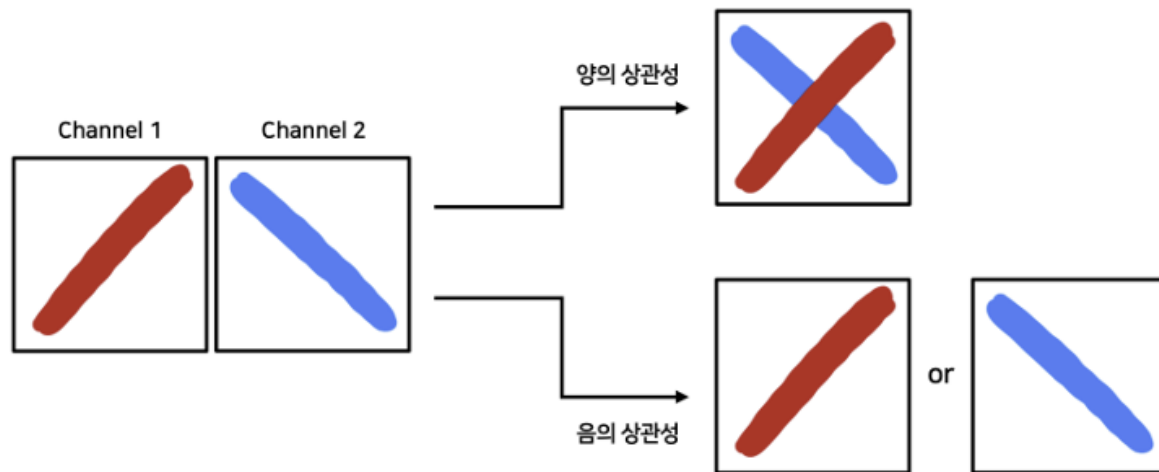


임의의 Layer의 Feature map에서 채널 1과 채널 2를 발췌했다고 가정

## 1

# Neural Style Transfer

## Style Loss | Gram Matrix



특정 스타일의 두 채널이

양의 상관관계를 가지고 있다면 동시에 나타나고,

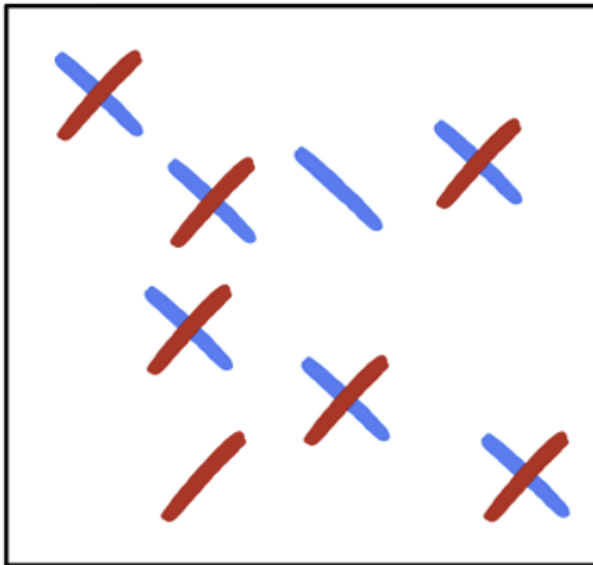
음의 상관관계를 가지고 있다면 둘 중 하나만 등장

## 1

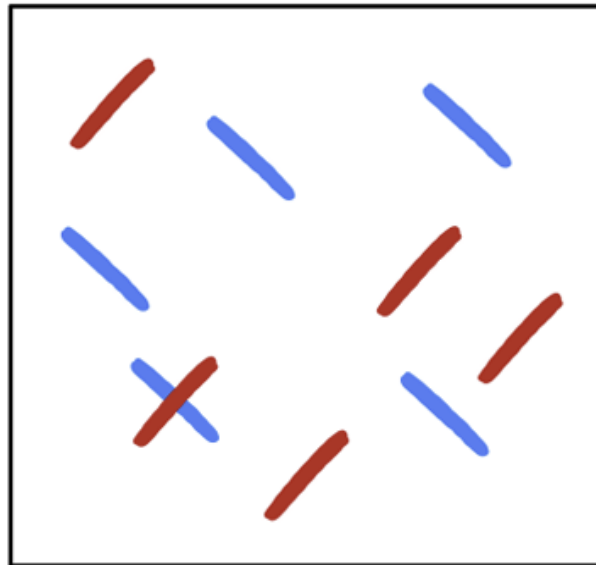
# Neural Style Transfer

## Style Loss | Gram Matrix

양의 상관성



음의 상관성



Gram Matrix로 뽑은 Style 양상!

## 1

## Neural Style Transfer

## Style Loss

Style Loss

$$E_l = \frac{1}{4c^2h^2w^2} \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2$$

⋮

$G_{ij}^l$  : Generated image의  $l$  층의 Style representation

$A_{ij}^l$  : Style image의  $l$  층의 Style representation

$c$  :  $l$  층의 Channel 개수

$h, w$  :  $l$  층의 Height, Weight

→ Gram Matrix의 **Squared Error**를 N, M으로 스케일링 한 식

## 1

## Neural Style Transfer

## Style Loss

Style Loss

$$E_l = \frac{1}{4c^2 h^2 w^2} \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2$$



$$L_{style}(S, G) = \sum_{l=0}^L \omega_l E_l$$

$G_{ij}^l$  : Generated image의  $l$  층의 Style representation

$A_{ij}^l$  : Style image의  $l$  층의 Style representation

$\omega_l$  : weighting factors of layer  $l$

$c$  :  $l$  층의 Channel 개수



저수준의 style부터 고수준의 style까지 모두 의미가 있으므로

모든 layer에 대한 가중합을 사용함

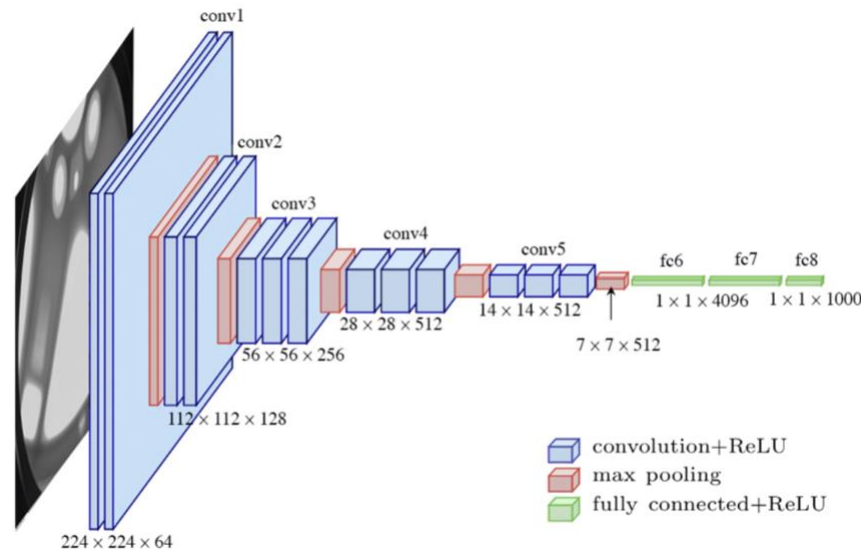


Gram Matrix의 Squared Error를 N, M으로 스케일링 한 식

## 1

# Neural Style Transfer

## Neural Style Transfer | 준비



이미지의 특징을 추출할 수 있는 포괄적인 이미지 모델만이 필요하므로  
Pretrained CNN Model 준비

# 1

## Neural Style Transfer

### Neural Style Transfer | 준비

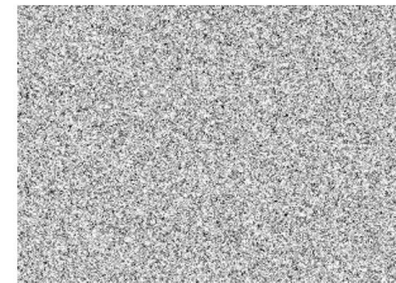
Content image



Style image



Random noise

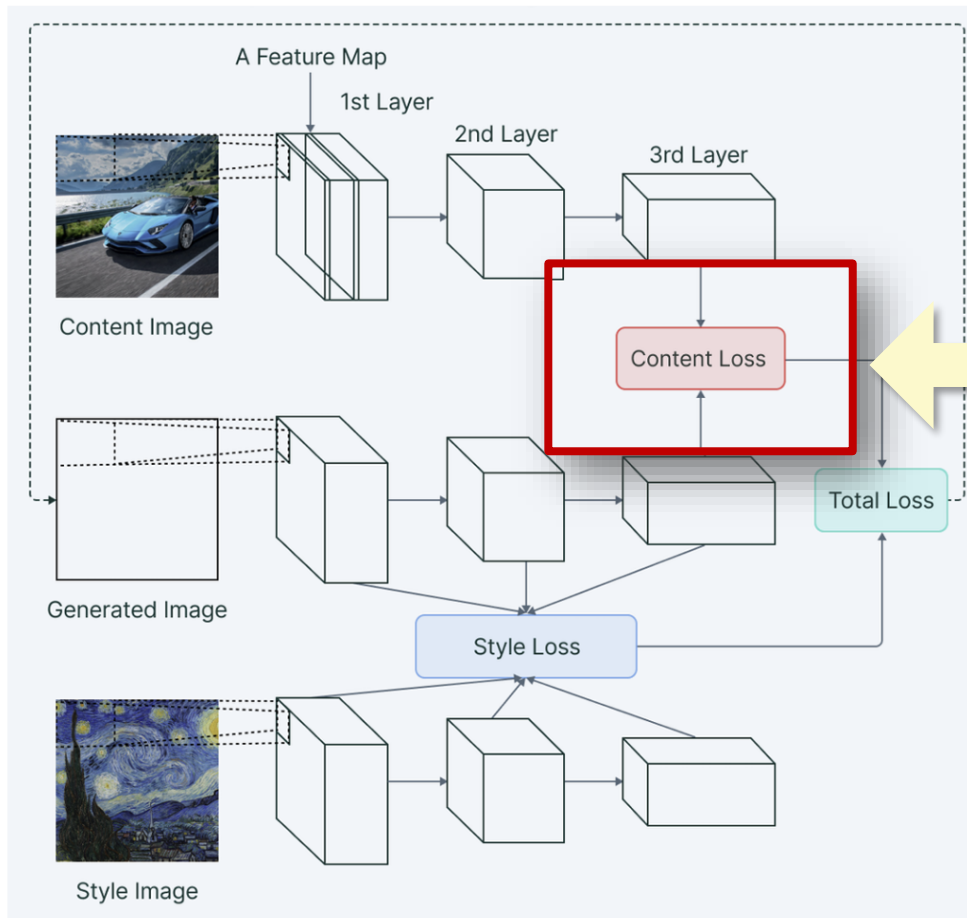


Style image, Content image와 Random noise 준비

## 1

# Neural Style Transfer

## Neural Style Transfer | 학습



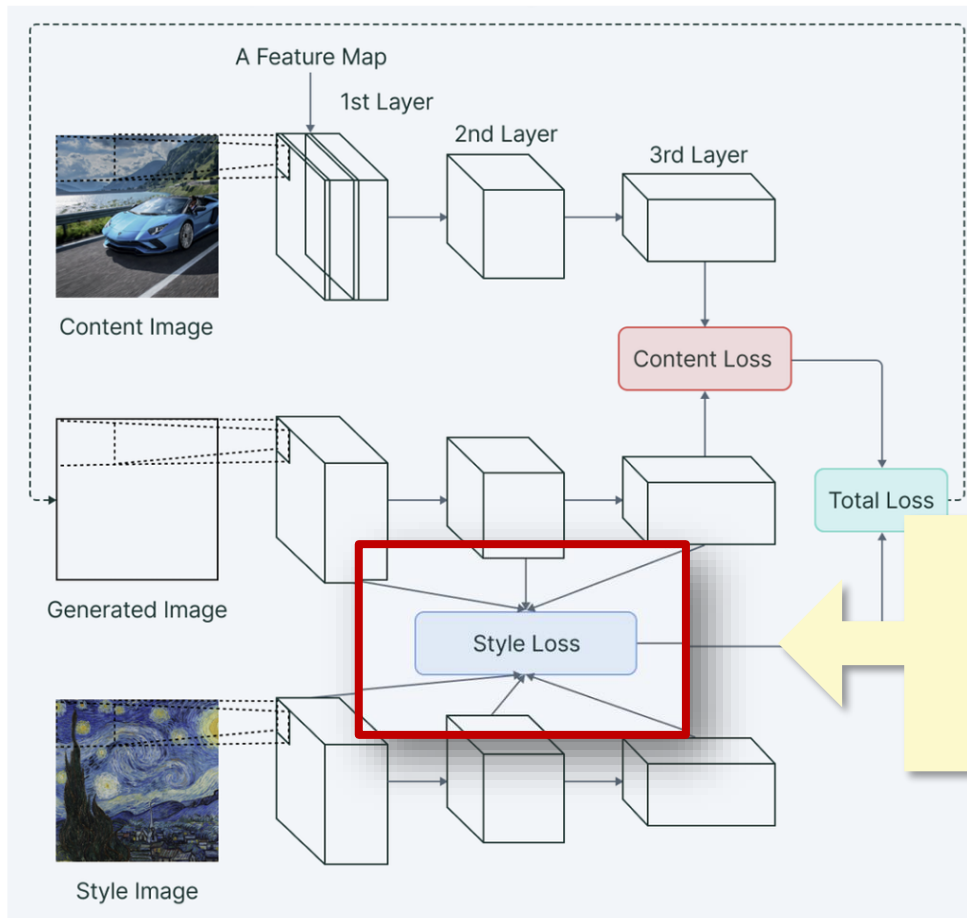
① 선택된 임의의 layer  $l$  에서  
Content Loss 계산



## 1

# Neural Style Transfer

## Neural Style Transfer | 학습

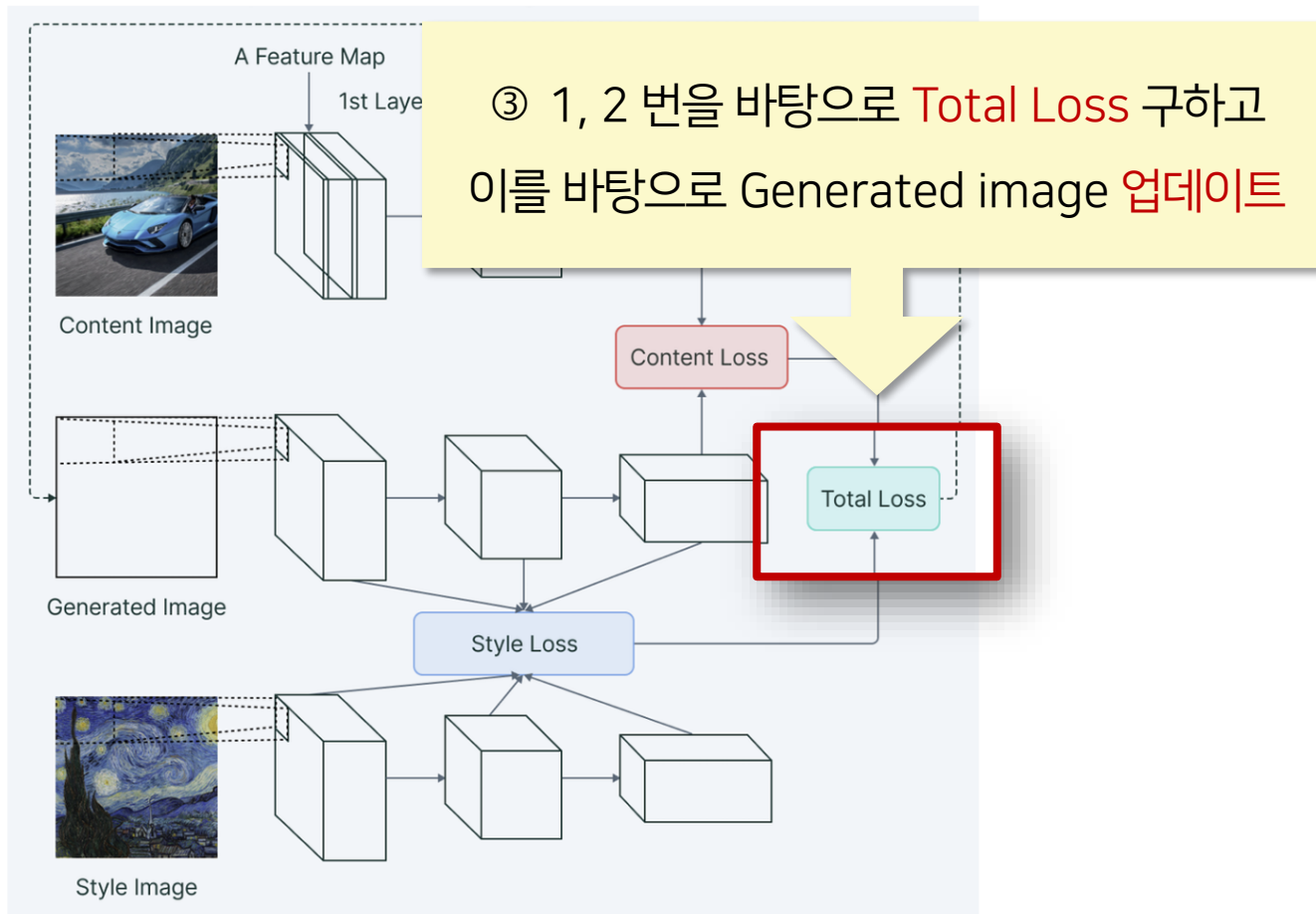


② 전체 layer  $l$  에 대한  
Style Loss 계산

## 1

# Neural Style Transfer

## Neural Style Transfer | 학습



2

VAE

## AE

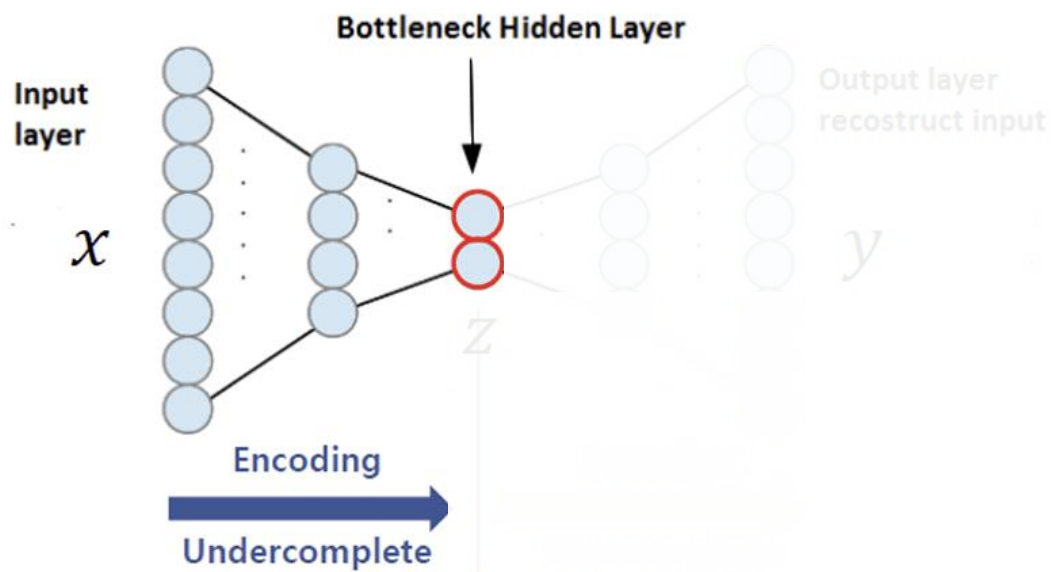
## AE AutoEncoder

입력 데이터의 잠재 표현을 배우는 것을 목적으로 하는 인공 신경망



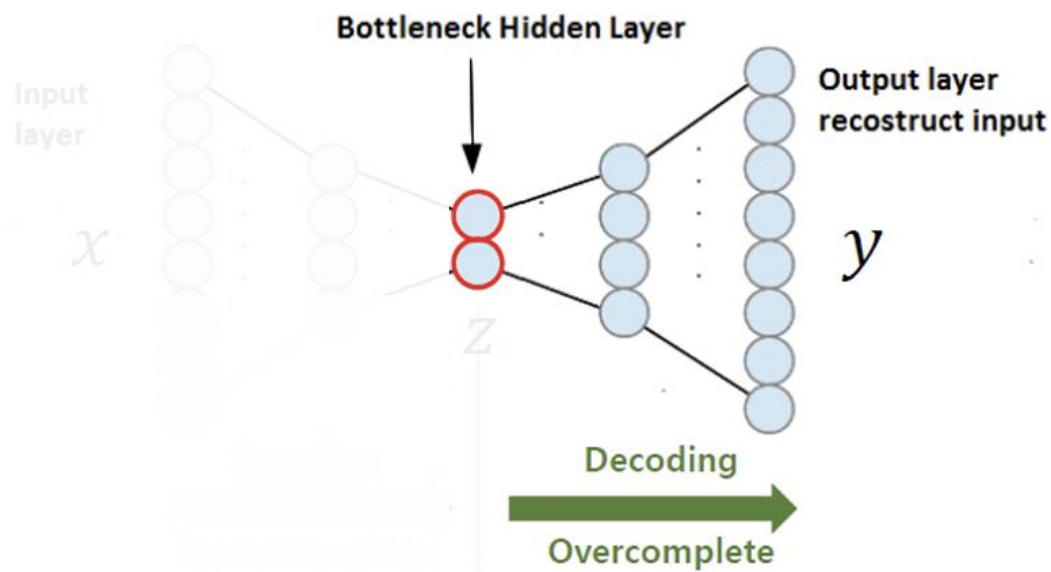
단순히 입력을 출력으로 복사하는 방법을 학습하는데,  
네트워크에 제약을 가하여 작업을 오히려 어렵게 만드는 것이 특징

## AE

**Encoder**

입력을 압축시켜 잠재 표현으로 바꿈

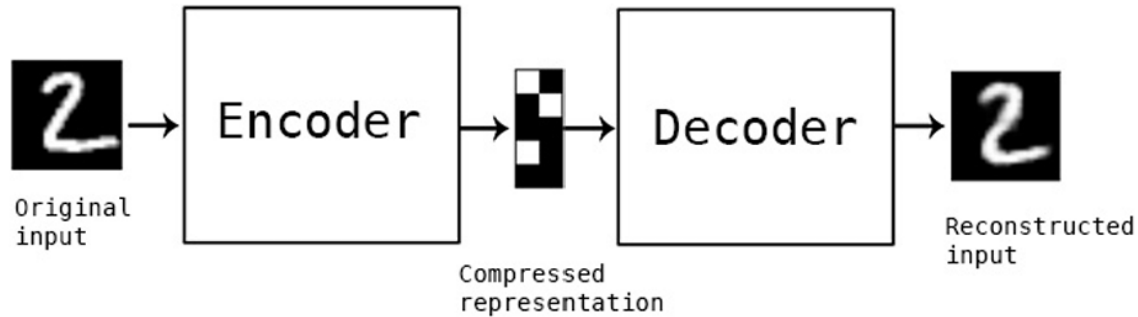
## AE



**Decoder**

그 잠재 표현을 reconstruct(재건)

## AE



핵심은 AE가 **Encoder의 학습, 그리고 차원 축소**에 중점을 두고 있다는 점!  
Decoder는 Encoder의 학습을 위한 수단일 뿐

AE 자체는 생성 모델 X

## VAE

## VAE Variational AutoEncoder

생성에 중점을 둔 AutoEncoder

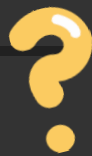


즉 **Decoder**를 학습시켜서,  
특징이 주어지면 최대한 이에 맞는 이미지를 생성해내는 것이 목적



AE와는 어떤 점이 다를까요?





VAE

## AE와 VAE의 차이점

VAE Variational AutoEncoder

AE

생성에 중점을 둔 AutoEncoder

학습 데이터에서  
중요한 Feature들로  
압축된 manifold를 찾는 것

VS

VAE

학습 데이터에 있는  
데이터 포인트  $x$ 에 대해  
likelihood를 최대화하는  
 $p(x|z)$ 를 구하는 것

특징이 주어지면 최대한 이에 맞는 이미지를 생성해내는 것이 목적

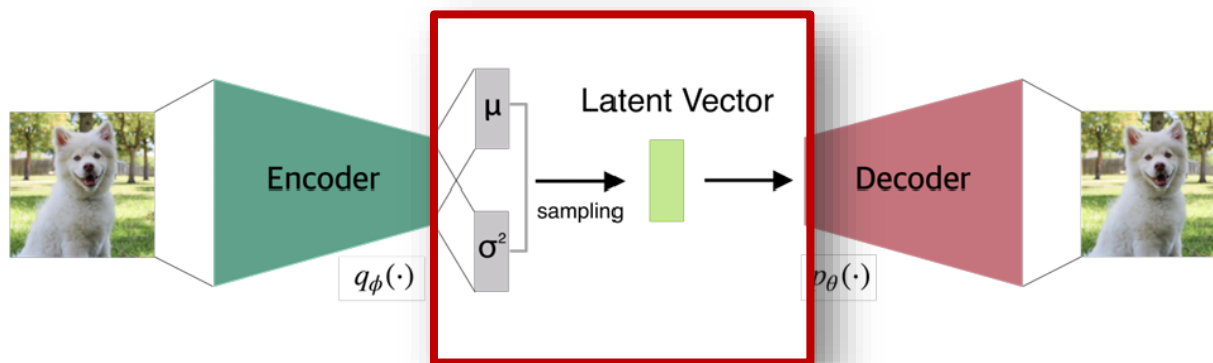
$p(x|z)$ 의 분포를 알면,

$z$ 를 샘플링하여 학습 데이터  $x$ 와 유사한 새로운 데이터  $x_{new}$ 를 생성할 수 있음

## VAE

## 목표

이미지의 latent vector를 학습하는 것이 아니라,  
Feature의 평균  $\mu$ 와 분산  $\sigma^2$  벡터를 나타내는 latent Space  $z$ 를 학습하는 것



## VAE

## 최종 목적

원하는 정답인  $x$ 가 나올 확률 ( $x$ 의 likelihood)인  $p_{\theta}(x)$ 를 최대화

$$p_{\theta}(x) = \frac{p_{\theta}(x|z)p_{\theta}(z)}{p_{\theta}(z|x)}$$

$x$  : input

$z$  : latent vector

## VAE

## 최종 목적

원하는 정답인  $x$ 가 나올 확률 ( $x$ 의 likelihood)인  $p_{\theta}(x)$ 를 최대화

$$p_{\theta}(x) = \frac{p_{\theta}(x|z)p_{\theta}(z)}{p_{\theta}(z|x)}$$

$x$  : input

$z$  : latent vector



$z$ 의 확률분포인  $p_{\theta}(z)$ 는 단순히  $z$ 의 샘플링 공간 분포  
(Standard Multivariate Gaussian Distribution)

## VAE

## 최종 목적

원하는 정답인  $x$ 가 나올 확률 ( $x$ 의 likelihood)인  $p_{\theta}(x)$ 를 최대화

$$p_{\theta}(x) = \frac{p_{\theta}(x|z)p_{\theta}(z)}{p_{\theta}(z|x)}$$

$x$  : input

$z$  : latent vector



$z$ 가 주어질 때의  $x$ 의 확률분포인  $p_{\theta}(x)$ 는

Decoder를 통해 구할 수 있음

## VAE

## 최종 목적

원하는 정답인  $x$ 가 나올 확률 ( $x$ 의 likelihood)인  $p_{\theta}(x)$ 를 최대화

$$p_{\theta}(x) = \frac{p_{\theta}(x|z)p_{\theta}(z)}{p_{\theta}(z|x)}$$

$x$  : input

$z$  : latent vector



$p_{\theta}(z|x)$  는 분포를 모르기에 구할 수 없음

→ Encoder의  $q_{\phi}(z|x)$ 를 통해 근사

## VAE | Loss Function

$$L_{VAE}(x, z) = -E_{z \sim q_{\phi}(z|x)} [\log p_{\theta}(x|z)] + KL(q_{\phi}(z|x) || p_{\theta}(z))$$

## VAE | Loss Function

$$L_{VAE}(x, z) = -E_{z \sim q_{\phi}(z|x)} [\log p_{\theta}(x|z)] + KL(q_{\phi}(z|x) || p_{\theta}(z))$$

$$E_{z \sim q_{\phi}(z|x)} [\log p_{\theta}(x|z)]$$

Reconstruction Loss

Input data와 유사하게  
출력을 재건하기 위해 적용

AE의 관점과 동일한 복원오차로,  
Negative log likelihood라고  
할 수 있음



## VAE | Loss Function

$$L_{VAE}(x, z) = -E_{z \sim q_{\phi}(z|x)} [\log p_{\theta}(x|z)] + KL(q_{\phi}(z|x) || p_{\theta}(z))$$

$$KL(q_{\phi}(z|x) || p_{\theta}(z))$$

Regularization Loss

$p_{\theta}(z)$  와  $q_{\phi}(z|x)$  의 분포 차이를  
KLD를 통해 수치화하고,  
이를 줄이는 방향으로 학습하는 것

Encoder를 통과한 뒤의 분포가  
 $z$  의 분포인 **다변량 정규분포**를  
따르도록 함

## VAE | Loss Function

$$L_{VAE}(x, z) = -E_{z \sim q_{\phi}(z|x)} [\log p_{\theta}(x|z)] + KL(q_{\phi}(z|x) || p_{\theta}(z))$$



위 Loss를 통해 VAE를 학습시키면,  
Encoder를 제거하고 **Decoder만 가지고** 특정 확률분포를 넣었을 때  
학습된 모델을 통해 **이미지를 만들어낼 수 있음**

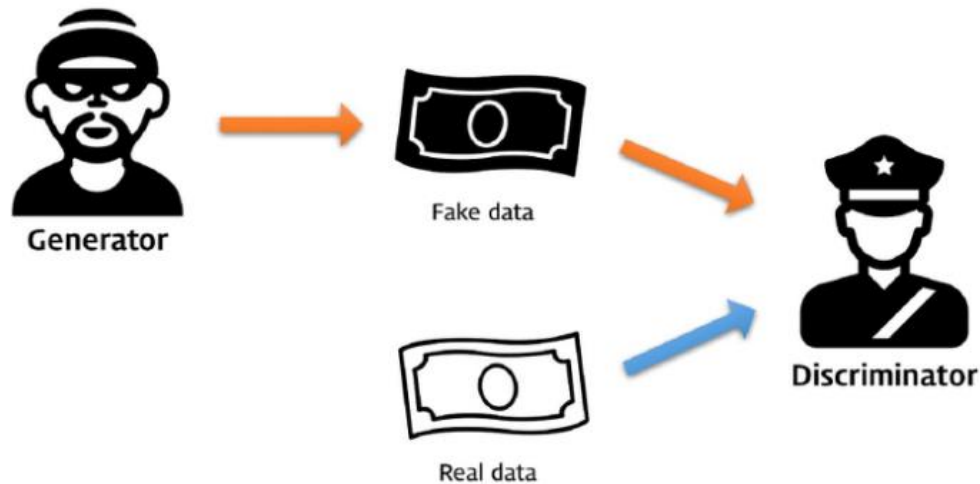
3

GAN

## GAN

## 적대적 생성 신경망 Generative Adversarial Networks

서로 다른 두 개의 네트워크를 **적대적으로 학습**시켜 구현한 생성모델

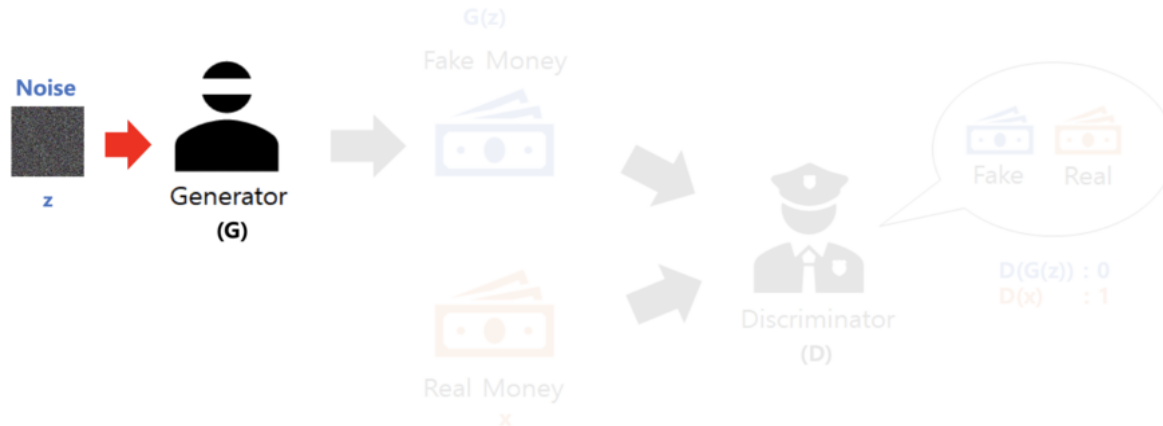


모델은 Generator와 Discriminator 네트워크로 구성되어 있음

## GAN

## 적대적 생성 신경망 Generative Adversarial Networks

Generator G는 실제 데이터와 비슷하게 데이터를 생성하는 것이 목적

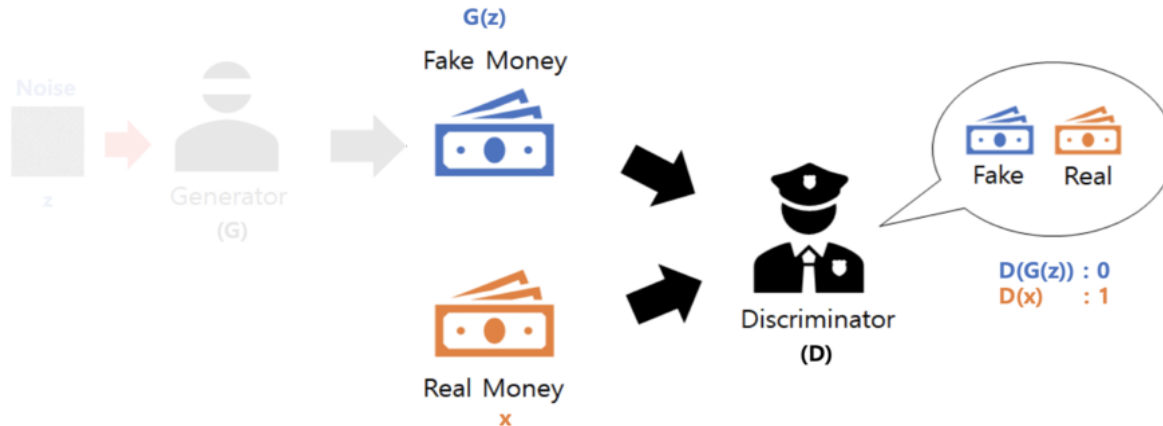


실제 분포와 비슷한 분포를 생성하기 위해 끊임없이 노력!

## GAN

## 적대적 생성 신경망 Generative Adversarial Networks

Discriminator D는 가짜와 실제 데이터를 **정확하게 구별**하는 것이 목적



D는 분포를 더욱 잘 구분하기 위해 노력하지만 잘 구분하지 못하는 경우  
G의 생성능력이 발전하였다고 판단 가능!

## GAN

## 가치함수 구성

$$z \sim p_z(z):$$

노이즈 데이터

$$G(z) \sim p_G:$$

G가 생성한 임의의 데이터

$$x \sim p_{data}:$$

실제 데이터  $x$

$z \sim p_z(z)$ : 정규분포를 사용하는 임의의 노이즈에서 샘플링한 데이터

## GAN

## 가치함수 구성

$$z \sim p_z(z):$$

노이즈 데이터

$$G(z) \sim p_G:$$

G가 생성한 임의의 데이터

$$x \sim p_{data}:$$

실제 데이터  $x$ 

$G$ : Input Noise를 받고 새로운 이미지를 생성,  
이때 생성 이미지 또한 분포를 가지나 분포의 형태는 가정하지 않음



## GAN

## 가치함수 구성

$$z \sim p_z(z):$$

노이즈 데이터

$$G(z) \sim p_G:$$

G가 생성한 임의의 데이터

$$x \sim p_{data}:$$

실제 데이터  $x$ 

$D$ : 가짜라고 판별되는 데이터에 0을 배정

$$D(data) = \begin{cases} 1 & \text{if the data is real} \\ 0 & \text{if the data is fake} \end{cases}$$

## GAN

## 이상적인 네트워크 모델

D는 **V를 최대화**하는 것, G는 **V를 최소화**하는 것이 목표

이상적인 D:  $D(G(z)) = 0, D(x) = 1$

이상적인 G:  $D(G(z)) = 1$

\* 즉, D는  $D(x)$ 가 커지고  $D(G(z))$ 가 감소하길 원하고

G는  $D(G(z))$ 가 증가하길 원함

$$V(D, G) = \log D(x) + \log(1 - D(G(z)))$$

## GAN

## 최종적인 가치함수

$$\min_G \max_D \left( \log D(x) + \log (1 - D(G(z))) \right)$$

\* 고정된 X와 Z에 대한 목적함수

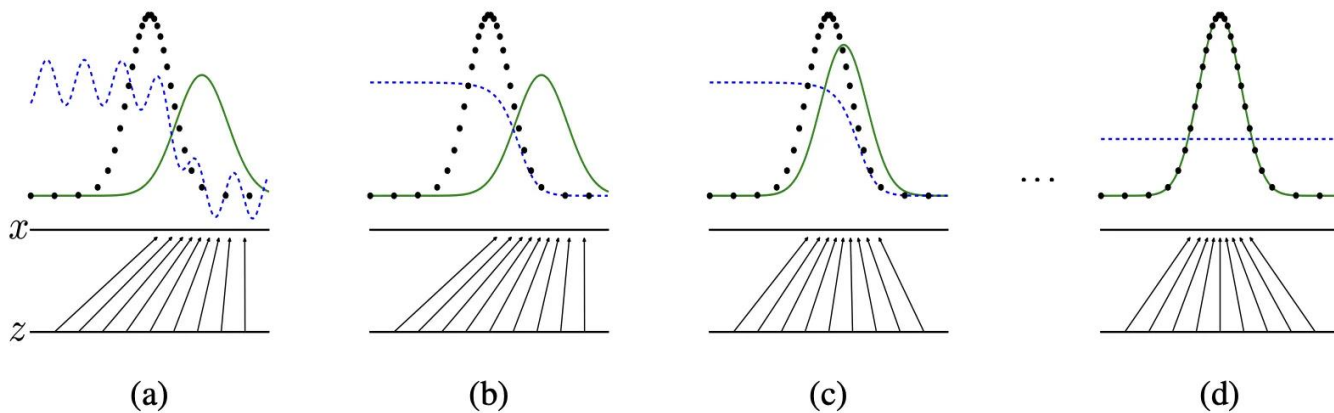
$$\min_G \max_D V(D, G) = E_{x \sim p_{data}(x)} [\log D(x)] + E_{z \sim p_z(z)} [\log (1 - D(G(z)))]$$

\* 기댓값을 구해준 형태의 X와 Z 샘플 전체에 대한 목적함수

결국 G와 D가 목적함수의 값을  
줄이고 키우면서 **경쟁**하는 형태

## GAN

## 직관적인 학습 과정



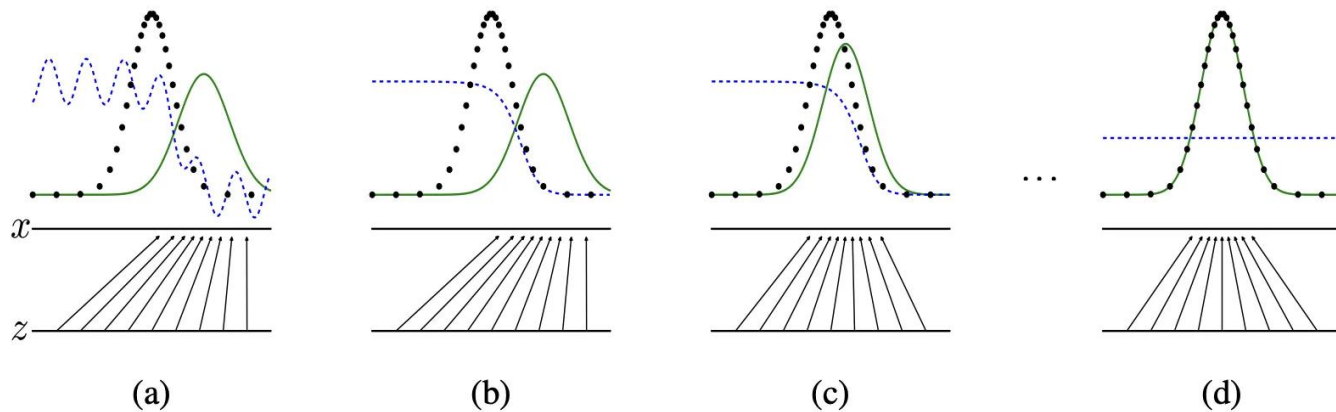
검은색 점  
실제 데이터 분포

초록색 선  
G가 만들어내는  
생성 데이터의 분포

파란색 점  
D의 분포에 따른  
검증 확률

## GAN

## 직관적인 학습 과정



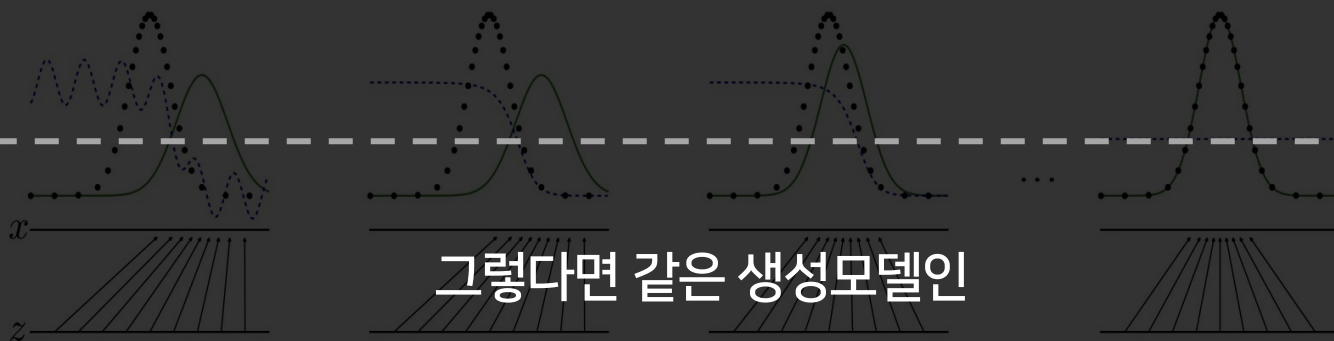
- (a) - 생성 데이터와 실제 데이터의 분포가 매우 다른 초기 상태
- (b) - D가 학습을 잘 진행한 상태, 진짜 데이터를 구별하는 능력이 향상
- (c) - G가 D의 학습(피드백)을 진행하여 더 유사한 데이터를 생성
- (d) - D가 차이를 판별하지 못해 중간 값 0.5만 출력

## GAN

직관적인 학습 과정



## GAN vs VAE



그렇다면 같은 생성모델인

(a) GAN과 VAE의 **차이**는 어디에서 오는지 알아보자!(a) - 생성 데이터와 실제 데이터의 분포가 매우 다른 **초기 상태**(b) - D가 학습을 잘 진행한 상태, 진짜 데이터를 **구별하는 능력**이 향상(c) - G가 D의 **학습(피드백)**을 진행하여 더 유사한 데이터를 생성(d) - **D가 차이를 판별하지 못해** 중간 값 0.5만 출력

## Implicit vs Explicit Generative Models

### Implicit Model

학습 데이터 분포에 대한 모델을 **정의하지 않고**,  
단지 생성된 데이터가 **실제 데이터와 유사한지 여부**로 평가 (GAN)

\* 데이터의 분포를 잘 학습한다면, 그 후 이에 대한 샘플링은 자동으로 수행 가능

### Explicit Model

학습 데이터 **분포에 대한 모델을 확실히 정의**할 수 있고,  
이를 활용하여 생성을 진행하는 생성 모델 (VAE)

\* GAN은 분포를 학습하려 하지 않고, Density Estimation 또한 진행하지 않음

## Implicit vs Explicit Generative Models

### GAN의 학습 원리

G를 고정했을 때, **최적의 D**는 다음과 같이 계산 가능

$$D^* = \operatorname{argmax}_D V(D, G)$$



$$\begin{aligned} V &= E_{x \sim P_{data}} [\log D(x)] + E_{x \sim P_g} [\log(1 - D(x))] \\ &= \int_x P_{data}(x) \log D(x) dx + \int_x P_g(x) \log(1 - D(x)) dx \\ &= \int_x [P_{data}(x) \log D(x) + P_g(x) \log(1 - D(x))] dx \end{aligned}$$



## Implicit vs Explicit Generative Models

식을 간단하게 하기 위해  $a = P_{data}(x)$ ,  $b = P_g(x)$ ,  $x = D(x)$ 로 둔다면,  
아래의 Convex 함수로 나타남



$$f(x) = a \log x + b \log(1 - x)$$



$$\frac{f(x)}{dx} = \frac{a}{x} - \frac{b}{1-x} = 0$$


$$a(x-1) - bx = 0$$

$$(b+a)x - a = 0$$

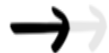
$$\therefore x = \frac{a}{a+b}$$


## Implicit vs Explicit Generative Models

식을 간단하게 하기 위해  $a = P_{data}(x)$ ,  $b = P_g(x)$ ,  $x = D(x)$ 로 둔다면,  
다음과 같은 Convex 함수로 나타남



$$\text{optimal } D^* = \frac{P_{data}(x)}{P_{data}(x) + P_g(x)}$$





$$\begin{aligned} \min_G V(G, D) &= V(G, D^*) \\ &= \int_x [P_{data}(x) \log D^*(x) + P_g(x) \log(1 - D^*(x))] dx \\ &= \int_x \left[ a \log \left( \frac{a}{a+b} \right) + b \log \left( 1 - \frac{a}{a+b} \right) \right] dx \\ &= \int_x \left[ a \log \left( \frac{1}{2} \cdot \frac{a}{(a+b)/2} \right) + b \log \left( \frac{1}{2} \cdot \frac{a}{(a+b)/2} \right) \right] dx \\ &= -2 \log 2 + \int_x a \log \frac{a}{(a+b)/2} dx + \int_x b \log \frac{a}{(a+b)/2} dx \end{aligned}$$

## Implicit vs Explicit Generative Models



### JSD에 대한 이해

식을 간단하게 하기 위해  $a = P_{data}(x)$ ,  $b = P_g(x)$ ,  $x = D(x)$ 로 둔다면,

다음과 같은 Convex 함수로 나타남

정리한 식을 잘 이해하기 위해서는

**JSD에 대한 이해**가 선행되어야 함

$$\text{optimal } D^* = \frac{P_{data}(x)}{P_{data}(x) + P_g(x)}$$



$$\begin{aligned} \max_D V(G, D) &= V(G, D^*) \\ &= \int_x [P_{data}(x) \log D^*(x) + P_g(x) \log(1 - D^*(x))] dx \\ &= \int_x \left[ a \log \left( \frac{a}{a+b} \right) + b \log \left( 1 - \frac{a}{a+b} \right) \right] dx \\ &= \int_x \left[ a \log \left( \frac{1}{2} \cdot \frac{a}{(a+b)/2} \right) + b \log \left( \frac{1}{2} \cdot \frac{a}{(a+b)/2} \right) \right] dx \\ &= -2 \log 2 + \int_x a \log \frac{a}{(a+b)/2} dx + \int_x b \log \frac{a}{(a+b)/2} dx \end{aligned}$$

## JS Divergence

## JSD

비대칭적이라는 한계를 가지고 있던 **KLD의 대칭성 문제를 해결한**  
두 확률 분포 간의 차이를 수치화하는 방법

기존 KLD

$$D_{KL}(P||Q) \neq D_{KL}(Q||P)$$



두 확률 간의 중간 분포 M 추가

$$M = \frac{1}{2}(P + Q)$$

$$JSD(P||Q) = \frac{1}{2}D_{KL}(P||M) + \frac{1}{2}D_{KL}(Q||M)$$



**대칭적인** JSD

$$JSD(P||Q) = JSD(Q||P)$$

## JS Divergence

### GAN의 Loss Function

가치함수는 결국 두 분포 사이의 **거리를 최소화**하는 것이 목적,  
따라서 이를 Loss function으로 사용해도 문제 없음

$$\begin{aligned}
 \min_G V(G, D) &= V(G, D^*) = -2 \log 2 + \int_x a \log \frac{a}{(a+b)/2} dx + \int_x b \log \frac{b}{(a+b)/2} dx \\
 &= -2 \log 2 + \int_x P_{\text{data}}(x) \log \left( \frac{P_{\text{data}}(x)}{(P_{\text{data}}(x) + P_G(x))/2} \right) dx \\
 &\quad + \int_x P_G(x) \log \left( \frac{P_G(x)}{(P_{\text{data}}(x) + P_G(x))/2} \right) dx \\
 &= -2 \log 2 + KL \left( P_{\text{data}} \parallel \frac{P_{\text{data}} + P_G}{2} \right) + KL \left( P_G \parallel \frac{P_{\text{data}} + P_G}{2} \right) = -2 \log 2 + 2 JSD(P_{\text{data}} \parallel P_G)
 \end{aligned}$$

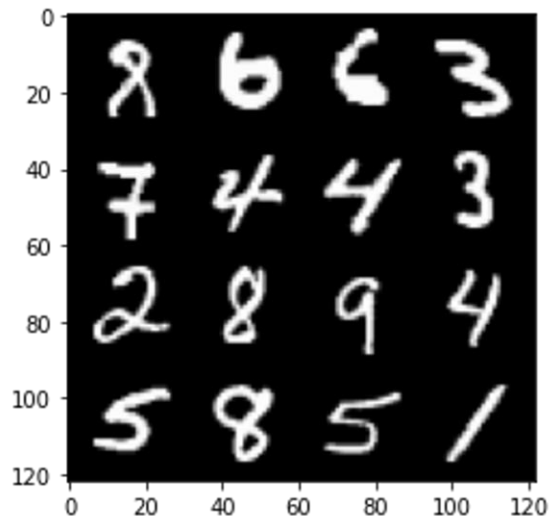
JSD를 활용해 이전의 식을 더 잘 정리해줄 수 있다!

## DCGAN

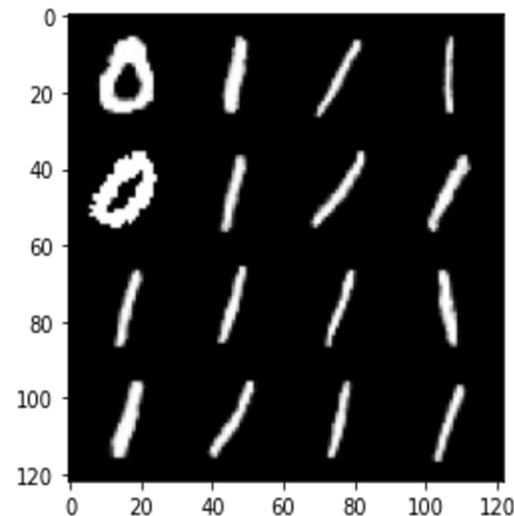
## GAN의 한계

기존 GAN은 구조가 불안정하고 **Mode Collapsing** 현상 발생  
이를 해결하기 위해 CNN을 도입한 DCGAN이 개발

\* Discriminator가 헛갈려하는 동일한 이미지만을 계속 출력하는 모습



Train Data Point

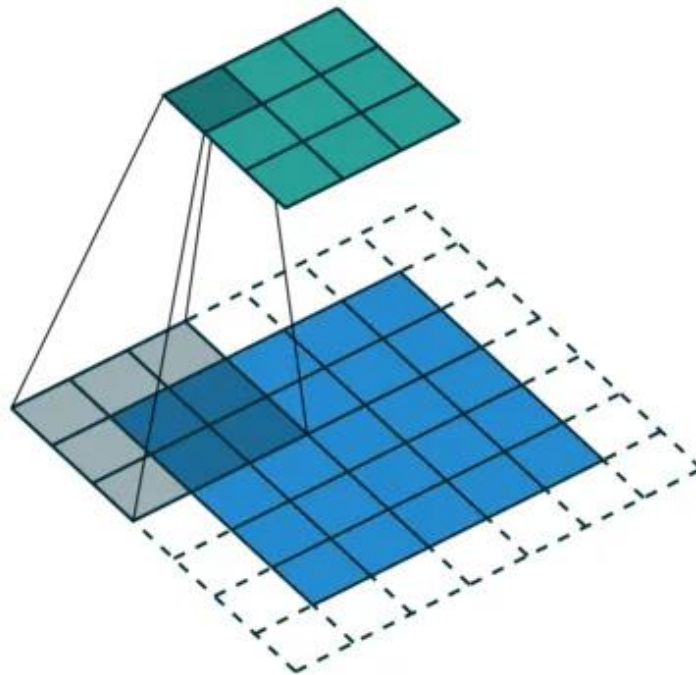


GAN Generated Data Point

## DCGAN

## Strided Convolution

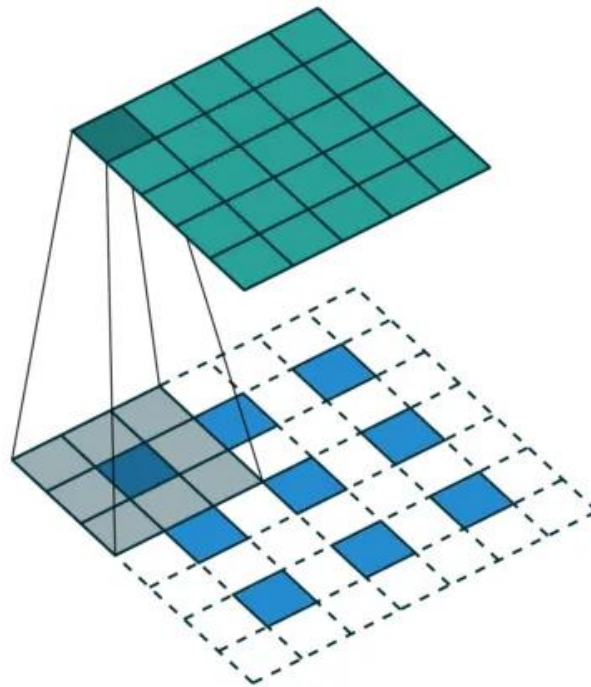
Max Pooling layer에선 미분이 불가능하므로 대신 **Discriminator**에선 **Stride 값을 올려** Feature map의 크기를 줄이며 극복



## DCGAN

## Fractionally-Strided Convolution

Input 픽셀 사이사이에 Padding을 적용시켜 크기를 키우는 것으로  
Size가 계속 커지는 Generator 네트워크에서 사용

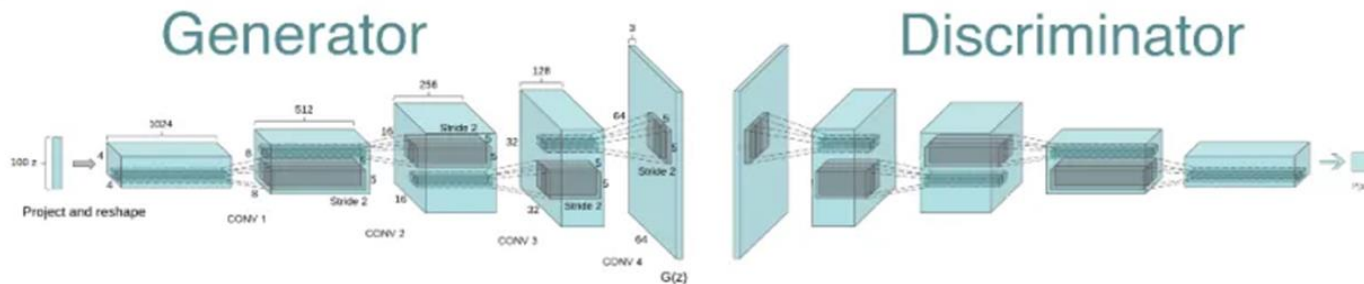




## DCGAN

## Eliminate Fully-Connected Layers

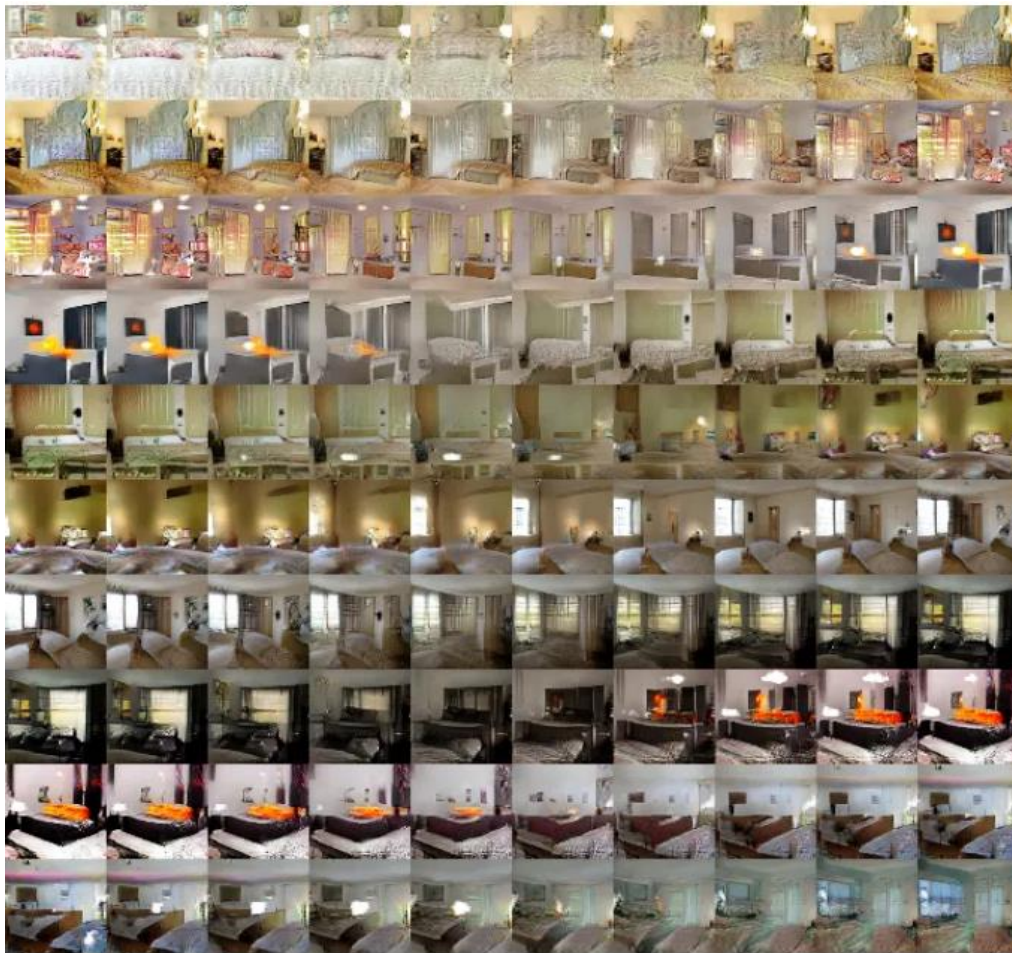
Noise Vector를 넣는 첫 Layer 이후로는  
**Fully-Connected Layer가 존재하지 않음**



FC Layer는 많은 파라미터 수로 인한 **연산 문제**와,  
Feature의 **지역적인 정보가 유실**된다는 단점 존재  
-> 마지막 Classifier Layer도 **Global Average Pooling(GAP)**로 대체

\* GAP을 거친 후 FC Layer를 달아 Softmax로 출력

## DCGAN의 생성 역량



급격하게 변하지 않고

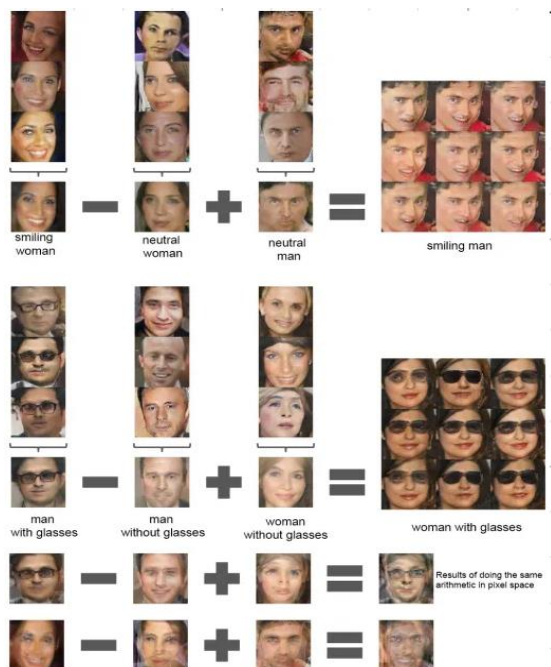
변

좌측에서 우측으로 이동하면서  
**연속적으로 변화**하는 모습

## DCGAN의 생성 역량

## Walking in the Latent Space

GAN의 잠재 공간에서도 유사한 현상이 일어날 수 있는데,  
잠재 공간 벡터들이 각각 특정 **이미지 특징을 제어하는 벡터**들로 해석될 수 있음



잠재 공간의 벡터를

**선형 결합**했더니

원하는 특징을 얻었다!



원하는 특징이 강조된

이미지를 생성 가능...!

## DCGAN의 생성 역량

## Walking in the Latent Space

GAN의 잠재 공간에서도 유사한 현상이 일어날 수 있는데,  
잠재 공간 벡터들이 각각 특정 **이미지 특징을 제어하는 벡터**들로 해석될 수 있음



이런 획기적인 DCGAN에도 단점은 존재! 잠재 공간의 벡터를

**선형 결합**했더니

원하는 특징을 얻었다!



원하는 특징이 강조된

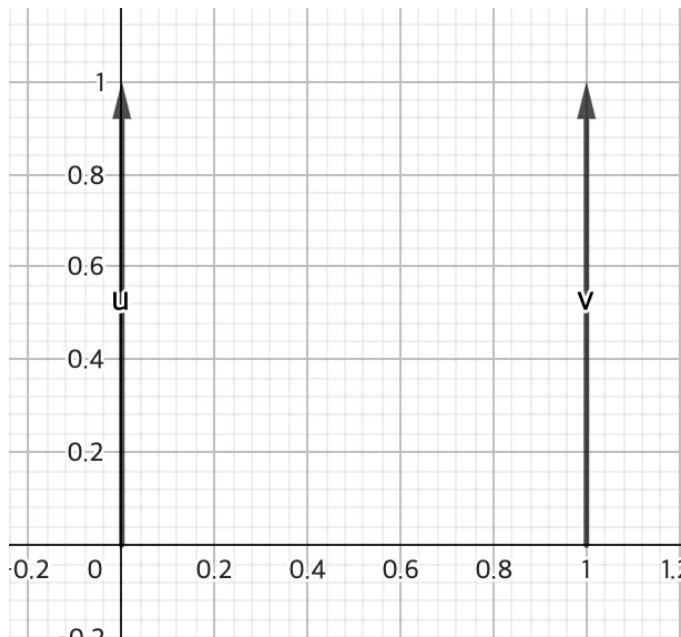
이미지를 생성 가능...!



## WGAN

## 기존 GAN의 문제점

JSD는 분포가 **겹치지 않는 경우에 대해서** 분포가 변해도 차이를 동일한 값으로 계산하기에 **분포 간 거리를 고려하지 못함**



JSD에서의 두 분포( $\delta(x)$ ,  $\delta(x - 1)$ )의 차이 =  $\log 2$

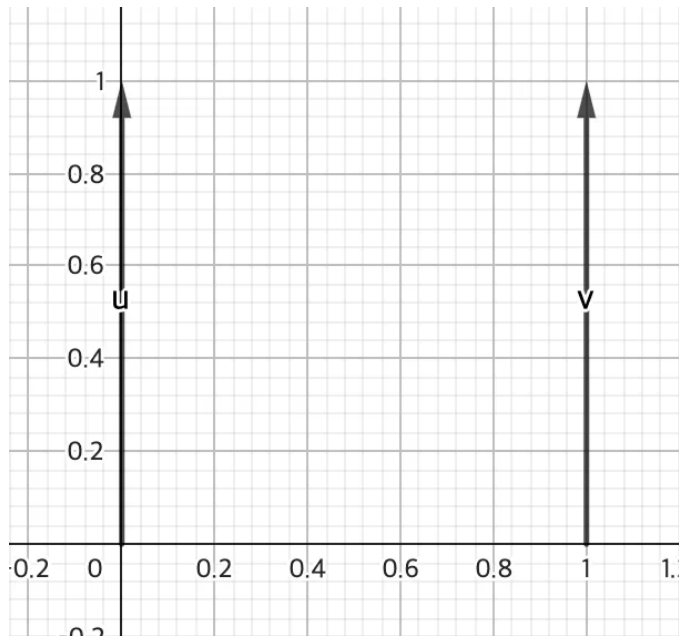
$$\text{JSD}(P_0 \parallel P_m) = \int_{P_0 \neq 0} \log \left( \frac{P_0(x)}{P_0(x)/2} \right) P_0(x) \mu(dx) = \log 2$$

$$\text{JSD}(P_1 \parallel P_m) = \int_{P_1 \neq 0} \log \left( \frac{P_1(x)}{P_1(x)/2} \right) P_1(x) \mu(dx) = \log 2$$

## WGAN

## 기존 GAN의 문제점

JSD는 분포가 **겹치지 않는 경우에 대해서** 분포가 변해도 차이를 동일한 값으로  
계산하기에 **분포 간 거리를 고려하지 못함**



JSD에서의 두 분포  $\delta(x)$ ,  $\delta(x - \theta)$ 의 차이 =  $\log 2$

$$\text{JSD}(P_0 \parallel P_m) = \int_{P_0 \neq 0} \log \left( \frac{P_0(x)}{P_0(x)/2} \right) P_0(x) \mu(dx) = \log 2$$

$$\text{JSD}(P_\theta \parallel P_m) = \int_{P_\theta \neq 0} \log \left( \frac{P_\theta(x)}{P_\theta(x)/2} \right) P_\theta(x) \mu(dx) = \log 2$$

## WGAN

## EM Distance(Wasserstein-1 Distance)

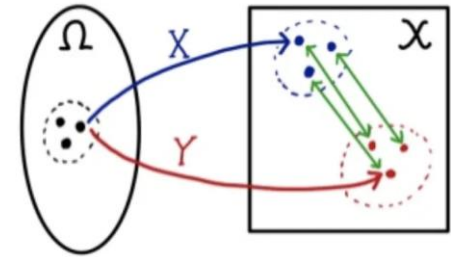
따라서 분포 간 거리를 계산하는 새로운 척도인

**EM Distance** 사용

## Earth Mover's Distance

$$W(P_r, P_\theta) = \inf_{\gamma \in \pi(P_r, P_\theta)} E_{(x,y) \sim \gamma} [||x - y||]$$

\*  $P_r$ 와  $P_\theta$ 의 결합확률분포인  $\gamma \in \pi(P_r, P_\theta)$  중에서  
 $||x - y||$ 의 기댓값을 가장 작게 추정한 값!



## WGAN



# EM Distance (Wasserstein Distance) Earth Mover's Distance

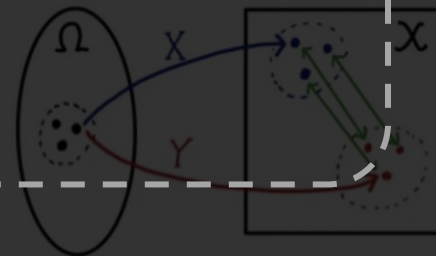
따라서 분포 간 거리를 계산하는 새로운 척도인

EM Distance 사용

직관적으로 **질량을 옮기는** 경우에 대해

살펴보면서 이해하자!

$$W(P_r, P_\theta) = \inf_{\gamma \in \pi(P_r, P_\theta)} E_{(x,y) \sim \gamma} [\|x - y\|]$$



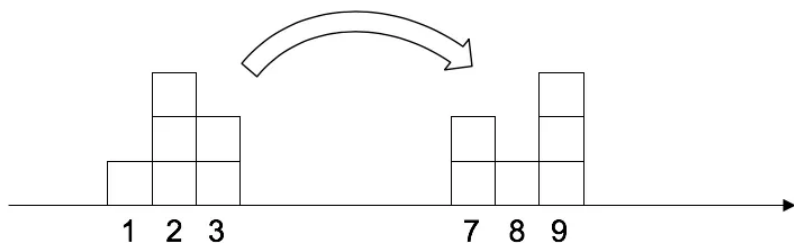
\*  $P_r$ 와  $P_\theta$ 의 결합확률분포인  $\gamma \in \pi(P_r, P_\theta)$  중에서

$\|x - y\|$ 의 기댓값을 가장 작게 추정한 값!

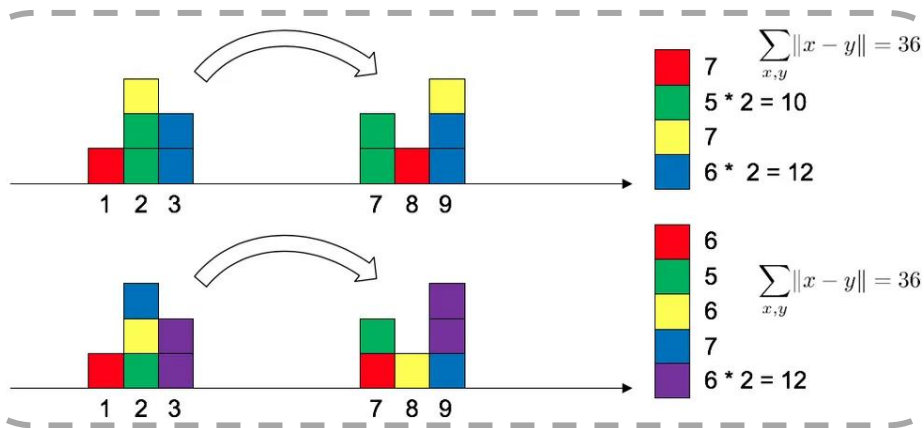


## WGAN

## EM Distance(Wasserstein-1 Distance)



왼쪽 확률분포를 오른쪽 확률분포로  
옮기기 위한 Cost를 계산한다면?



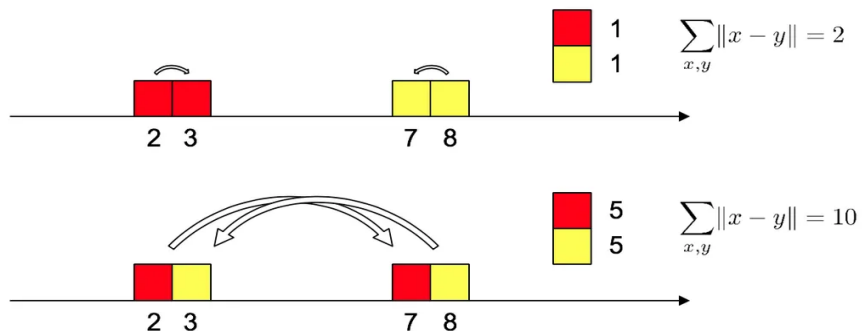
해당 상황에서  
두 방법의 Cost는 동일하게 나타남

## WGAN

## EM Distance(Wasserstein-1 Distance)

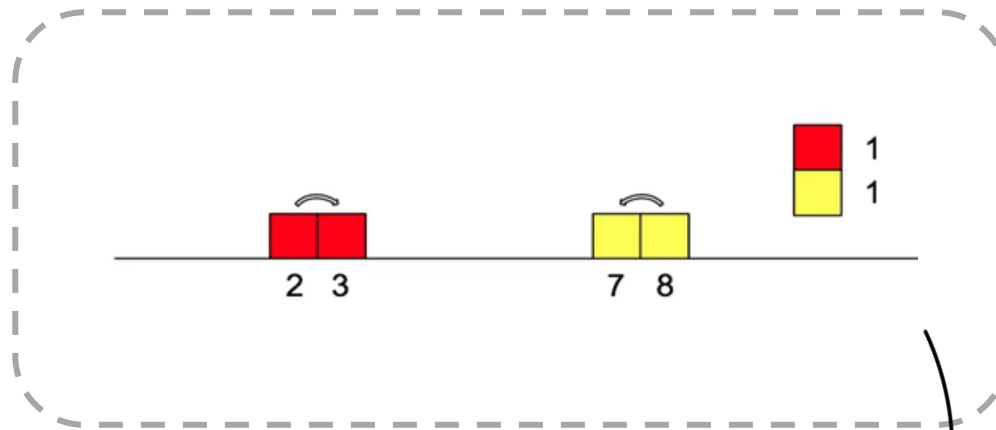


검은색과 회색이 각각 확률분포이며,  
검은 블록을 회색 블록처럼 옮긴다면?



해당 상황에서  
두 방법의 **Cost**는 서로 다름

## WGAN



해당 방법의 EM Distance

$$\begin{aligned} W(P_r, P_\theta) &= \gamma_{(X,Y)}(2,3) \times \|2 - 3\| + \gamma_{(X,Y)}(8,7) \times \|8 - 7\| \\ &= 0.5 \times 1 + 0.5 \times 1 = 1 \end{aligned}$$

## WGAN



해당 방법의 EM Distance =

$$W(P_r, P_\theta) = \gamma(X, Y) (2, 3) \times \|2 - 3\| + \gamma(X, Y) (8, 7) \times \|8 - 7\| = 0.5 \times 1 + 0.5 \times 1 = 1$$

# 3

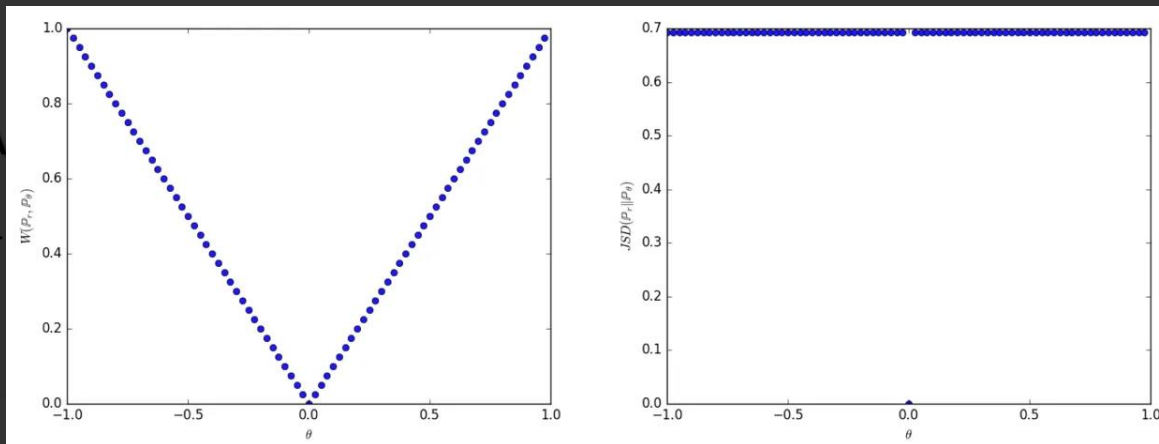
## GAN



### WGAN

기존 GAN

JSD는



앞서 만난 두 분포  $\delta(x), \delta(x-\theta)$ 의 EM Distance를 계산 가능

$$W(P_0 \parallel P_\theta) = \theta \int_{P_0 \neq 0} \log \left( \frac{P_0(x)}{P_0(x)/2} \right) P_0(x) \mu(dx) = \log 2$$

$$JSD(P_\theta \parallel P_\theta) = \int_{P_\theta \neq 0} \log \left( \frac{P_\theta(x)}{P_\theta(x)/2} \right) P_\theta(x) \mu(dx) = \log 2$$

그러나 학습을 진행하기 위해서

Loss function 내에 들어가는 함수들은 모두 미분가능해야 함!

\* 거리가 다르더라도 log 2만 반환하는 모습

## Lipschitz Continuous

### EM Distance의 연속성/미분가능성 증명

EM Distance의 **연속성을 증명**하기 위한 조건들을 명시



$g$ 가  $\theta$ 에 대해 연속적이면,  $W(P_r, P_\theta)$ 도 연속적



$g$ 가 *locally Lipschitz*이고, 특정 조건을 만족한다면,  
 $W(P_r, P_\theta)$ 는 거의 모든 점에서 연속이며 미분가능



이러한 조건들은 JSD와 KLD에서는 만족하지 않음

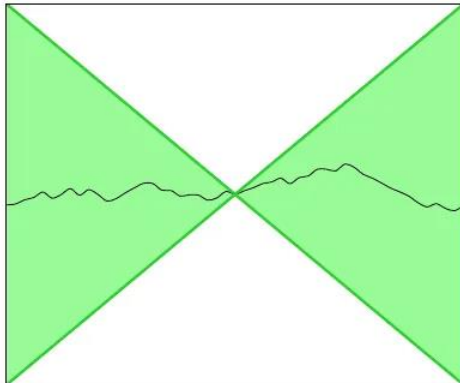
## Lipschitz Continuous

### K-Lipschitz

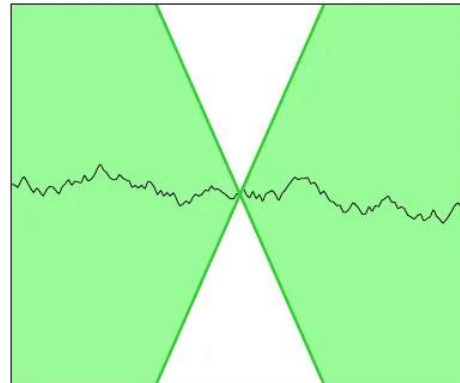
임의의 실수  $x_1, x_2$  에 대하여  $|f(x_1) - f(x_2)| \leq K|x_1 - x_2|$ 를 만족하는 조건으로

**기울기의 크기에 상한이 존재**

50 samples;  $K = 1$ ; seed = 123456789



143 samples;  $K = 2.718$ ; seed = 1729



함수의 기울기가 K 이상으로 넘어가지 않기에

Gradient Exploding과 같은 문제 해결

## Lipschitz Continuous

### K-Lipschitz

임의의 실수  $x_1, x_2$  에 대하여  $|f(x_1) - f(x_2)| \leq K|x_1 - x_2|$ 를 만족하는 조건으로

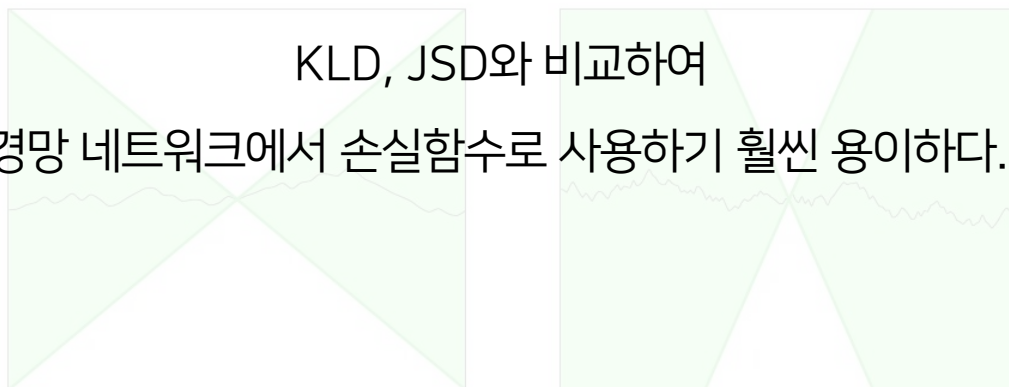
기울기의 크기에 상한이 존재



50 samples;  $K = 1$ ; seed = 1729      결국 EM Distance가       $K = 2.718$ ; seed = 1729

KLD, JSD와 비교하여

신경망 네트워크에서 손실함수로 사용하기 훨씬 용이하다....!



함수의 기울기가 K 이상으로 넘어가지 않기에

Gradient Exploding과 같은 문제 해결



## WGAN

## Kantorovich-Rubenstein Duality

WGAN의 EM distance 식에서,  
결합확률분포의 모든 조합을 생각하여 **하한(inf)**을 구하는 것은 **매우 어려움**

$$W(P_r, P_\theta) = \inf_{\gamma \in \pi(P_r, P_\theta)} E_{(x,y) \sim \gamma} [|x - y|]$$

$$W(P_0, P_\theta) = \sup_{|f|_L \leq 1} (E_{x \sim P_0} [f(x)] - E_{x \sim P_\theta} [f(x)])$$

## WGAN

### Kantorovich-Rubenstein Duality

따라서, Kantorovich-Rubenstein Duality를 이용해 식을 변형

$$W(P_r, P_\theta) = \inf_{f \in \text{Lip}(1)} E_{(x,y) \sim \gamma} [\|x - y\|]$$

이때 함수  $f$ 는 **1-Lipschitz**를 만족해야 함!

$$W(P_0, P_\theta) = \sup_{|f|_L \leq 1} (E_{x \sim P_0} [f(x)] - E_{x \sim P_\theta} [f(x)])$$

## WGAN

이후 파라미터를 추가하여  $P_\theta$ 를  $g_\theta$ 에 대한 식으로 바꿀 수 있음



$$\max_{w \sim \mathcal{W}} E_{x \sim p_r} [f_w(x)] - E_{z \sim p(z)} [f_w(g_\theta(z))]$$

$f = \log D(x)$ 일 때  
기존 GAN과 비슷해지는 것을 확인!

## WGAN

### CRITIC

실제 데이터의 분포 ( $P_r$ )와의 거리를 추정하는 것이  $D$ 의 역할 (WGAN에서는 **Critic**),  
 $G$ 의 Gradient update 시  $\theta$ 에 대해 미분되어 사라지므로 학습 가능



#### Critic의 Gradient

$$\nabla_w W(P_r, P_\theta) = E_{x \sim P_r}[\nabla_w f_w(x)] - E_{z \sim p}[\nabla_w f_w(g_\theta(z))]$$

해당 경우에는 Critic이  $P_r$ 의 역할을 해주기 때문에  $E_{x \sim P_r}[\nabla_w f_w(x)]$ 를 구하는 것이 가능



#### Generator의 Gradient

$$\nabla_\theta W(P_r, P_\theta) = -E_{z \sim p}[\nabla_\theta f_w(g_\theta(z))]$$

해당 경우에는 앞 항이 상수 취급되어 사라짐

## WGAN의 학습

### 기존의 GAN에서의 D

기존 GAN서의 Discriminator는 입력 이미지가 진짜인지 가짜인지를 판별하여,  
그 확률을 출력하는 데 목적



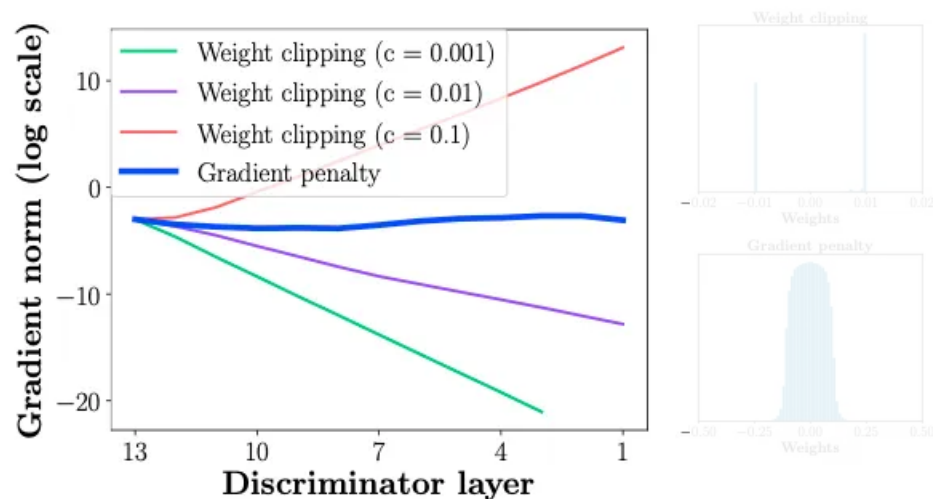
### WGAN에서의 D

WGAN에서의 D는 생성 이미지와 실제 이미지의  
분포 간 **차이를 줄이는 방향의 Gradient**를 제시하는 역할,  
Critic으로서 이미지의 진위성 판별 대신  
실제 이미지와 더 유사한 이미지를 만들어 낼 수 있도록 **피드백**을 제시

## Weight Clipping

### Weight Clipping

1- Lipschitz 조건을 만족시키기 위해 학습 중 Critic의 가중치를 조작하는 기법

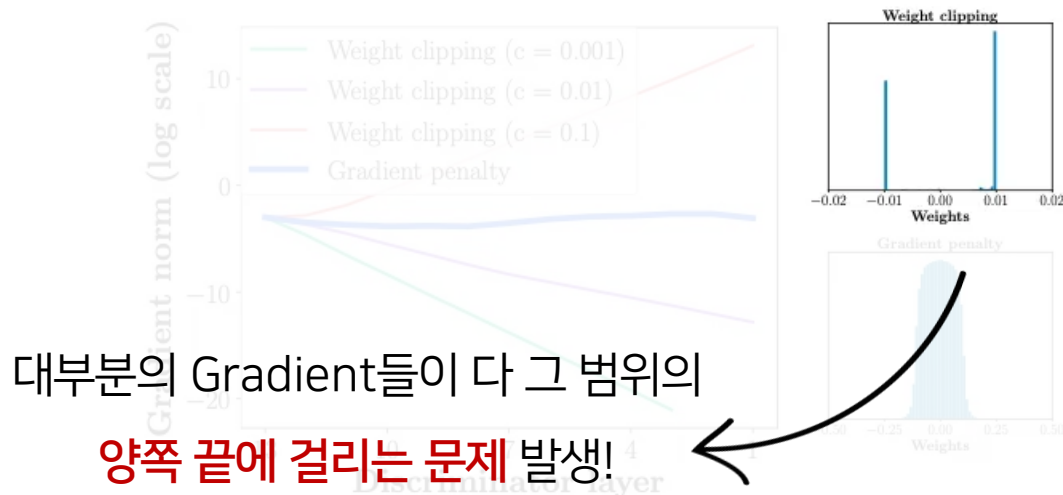


Critic을 학습시킬 때마다  $\text{clip}(-w, -c, c)$ 를 추가하여  $(-c, c)$  공간 내에 들어가도록 강제

## Weight Clipping

### Weight Clipping

1- Lipschitz 조건을 만족시키기 위해 학습 중 Critic의 가중치를 조작하는 기법



Critic을 학습시킬 때마다  $\text{clip}(-w, -c, c)$ 를 추가하여  $(-c, c)$  공간 내에 들어가도록 강제

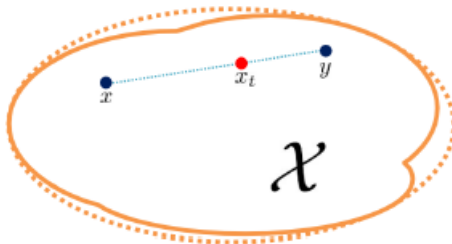
## WGAN-GP

### Gradient Penalty

Clipping 없이 **1-Lipschitz 조건을 만족**하기 위해

Critic Loss 뒤에 더해진 **Penalty**

$$f^* = \arg \max_{|f|_L \leq 1} E_{y \sim P_r} [f(y)] - E_{x \sim P_\theta} [f(x)] + \lambda E_{x_t \sim P_{x_t}} \left[ (|\nabla_{x_t} f(x_t)| - 1)^2 \right]$$



이때  $x_t$ 는 샘플  $x, y$ 의 내분점!



## WGAN-GP

## Gradient Penalty

$f^* = \arg \max_{|f|_L \leq 1} E_{y \sim P_r} [f(y)] - E_{x \sim P_\theta} [f(x)]$  인  $f^*$ 와  $X \sim P_x, Y \sim P_G$ 로부터 각각 얻은 샘플  $x, y$ 의 내분점  $x_t$ 에 대하여  $|\nabla f^*(x)| = 1$ 을 만족



Penalty를 추가함으로써  $f$  는 이미 1- Lipshitz 조건을 만족하는  
Optimal Critic  $f^*$ 에 더 유사하게 추정될 수 있음

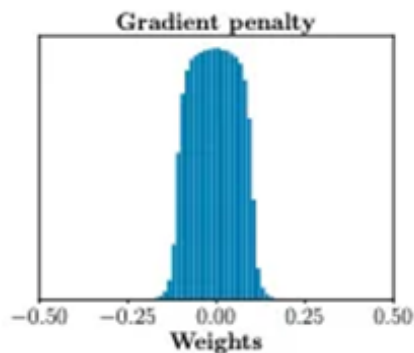
## WGAN-GP

## Gradient Penalty

Clipping 없이 1-Lipschitz 조건을 만족하기 위해

Critic Loss 뒤에 더해진 Penalty

$$f^* = \arg \max_{|f|_L \leq 1} E_{y \sim P_r} [f(y)] - E_{x \sim P_\theta} [f(x)] + \lambda E_{x_t \sim P_{x_t}} \left[ (|\nabla_{x_t} f(x_t)| - 1)^2 \right]$$



가중치가 양 끝 값에 걸리는 대신  
고르게 분포하는 모습을 확인 가능!

# 4

## Conditional GAN

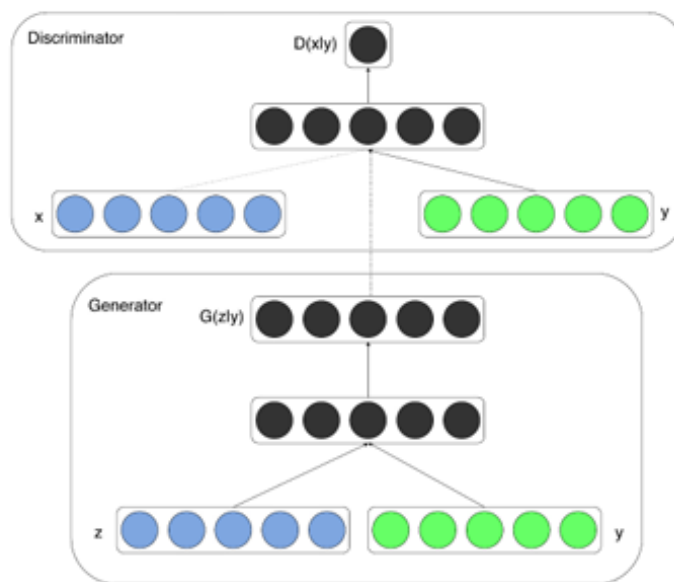
# 4

## Conditional GAN

### cGAN의 정의

#### Conditional GAN

조건부로 데이터를 생성하는 GAN의 변종 알고리즘



## cGAN의 정의

## Conditional GAN

조건부로 데이터를 생성하는 GAN의 변종 알고리즘



GAN의 목적이 일단 데이터를 만드는 것이 목적이었다면  
cGAN의 목적은 무엇을 만들어내는지에 보다 집중!

# 4

## Conditional GAN

cGAN의 가치 함수

가치함수

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

$G$ : Generator

$D$ : Discriminator

$x$ : 실제 input에서 샘플링한 데이터

$z$ : Latent Vector

$y$ : Conditional Vector, cGAN에서 추가된 개념

## 4

## Conditional GAN

## cGAN의 가치 함수

## 가치함수

$$\min_G \max_D V(G, D)$$

$$= E_{x \sim p_{data}(x)} [\log D(x|y)] + E_{z \sim p_z(z)} \left[ \log \left( 1 - D(G(z|y)) \right) \right]$$

## MNIST 데이터셋의 경우

D: Discriminator

G가 학습되어 숫자를 생성하고 있음에도

그 숫자가 어떤 숫자인지 **까보지 않으면** 결코 알 수 없음!

y: Conditional Vector, cGAN에서 추가된 개념

# 4

## Conditional GAN

$y$ 를 붙이는 이유



MNIST에서의 예시  
G와 D의 입력에서 Latent Vector  $z$  옆에

☞ **Condition 변수**  $[1, 0, 0, 0, 0, 0, 0, 0, 0, 0]$ 를 붙여 줌  
G가 MNIST의 데이터를 학습하여 **데이터 샘플**을 생성

☞ 해당 데이터 샘플은 **특정한 숫자 모양**으로 나타남

☞ 하지만 해당 데이터 샘플이 **어떤 Label에 해당하는지**는 Label을 직접  
찾아보지 않는 한 구현한 사람 ↓ 샘플링한 사람도 **알지 못함**

입력 노이즈 옆에 **Conditional 변수**가 붙게 되면서  
**조건부로 우리가 얻고자 하는 이미지를 지정해줄 수 있음!**



## 4

## Conditional GAN

$y$ 를 붙이는 이유



G가



해당



하지



Label을 직접

찾아보지 않는 한 구현한 사람, 샘플링한 사람도 **알지 못함**

**각 row별로 1~9까지 조건을 준 cGAN**

# 5

## Image-to-Image Translation

## Image-to-Image Translation의 정의

### Image-to-Image Translation

어떤 이미지의 특징을 바꾸는 Task

EX) 이미지 속 사람의 표정 변화



기존의 생성에서는 Latent Vector  $z$ 를 넣어주었지만,

Image-to-Image Translation의 input은

실제 input으로 넣어줄 이미지  $x$ 가 입력됨

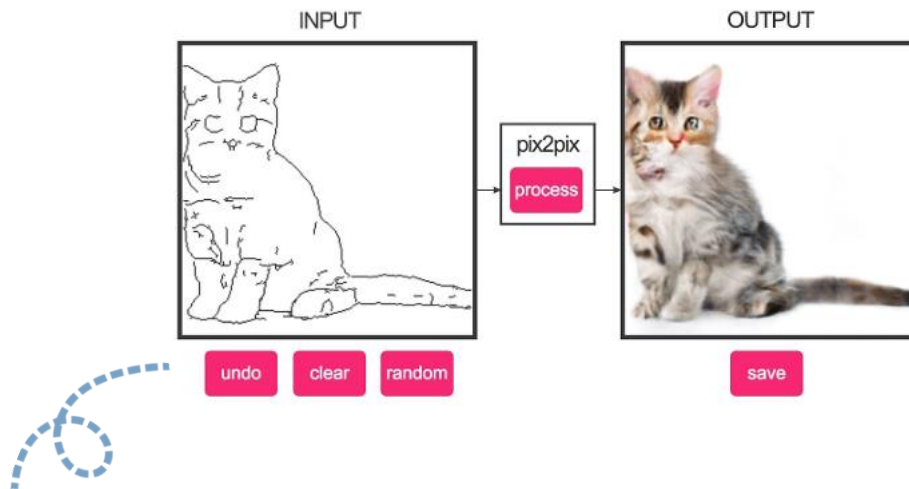
# 5

## Image-to-Image Translation

### Pix2Pix

#### Pix2Pix

**pair image**를 통해 조건에 맞는 **이미지 변환**을 수행하는 기법



특정한 스케치를 주었을 때 이에 맞는 실제 이미지를 출력함!

# 5

## Image-to-Image Translation

### Pix2Pix

Pix2Pix



어떠한 이미지를 condition으로 주고,  
이에 따른 translation을 만들어내는 cGAN 방식을 사용하는 기법

학습 시 Pair image를 가지고 있어야 함



시간 + 경제적으로 비효율적

특정한 스케치를 주었을 때 이에 맞는 실제 이미지를 출력함!

# 5

## Image-to-Image Translation

### CycleGAN

#### Unpair Image-to-Image Translation

Pair에 구매받지 않고 학습할 수 있는

Unpair Image-to-Image Translation의 등장

# 5

## Image-to-Image Translation

### CycleGAN

그러나 Unpaired하게 학습시키고자 하면  
앞서 언급한 **Mode Collapse** 문제가 발생함



#### Mode Collapse 문제



Pair랑 전혀 상관 없는 이미지들이 **결과로써 학습됨**



Input의 특징과 관계 없이 D를 속일 수 있는 **하나의 출력만**을 만드는 모델이 됨

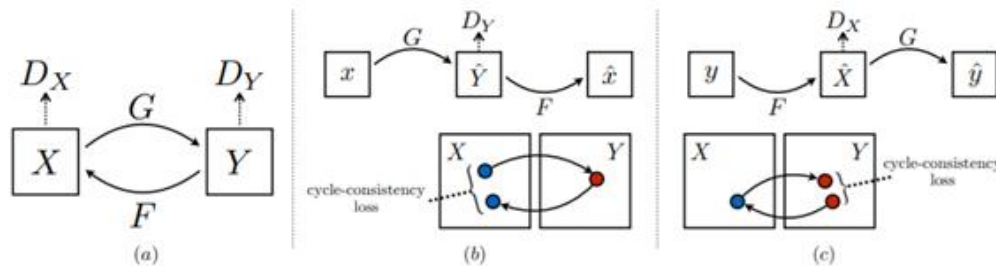
## 5

## Image-to-Image Translation

## CycleGAN의 정의

## CycleGAN

Mode Collapse 문제를 해결 하기 위해서  
두 개의 Generator를 사용하여 순환구조로 구현한 모델





## 5

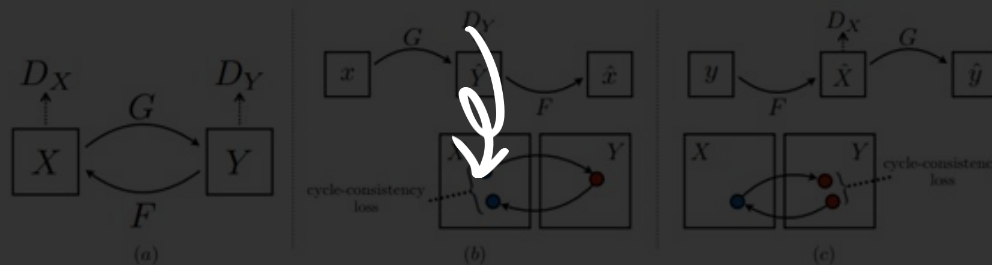
## Image-to-Image Translation

## CycleGAN의 정의



## CycleGAN

model collapse 문제를 해결 하기 위해서  
 Generator  $G$ 는 새로운 이미지를 생성해내며  
 두 개의 Generator를 사용하여 순환구조로 구현한 모델  
 $F$ 는 만들어진 이미지를 다시 원래의 이미지로 돌려냄



결국 Input이  $G$ 와  $F$ 를 거쳐 다시 자기 자신으로 돌아와야 함

이 특징을 Cycle Consistency라고 부름!

## Cycle Consistency

## Forward Cycle Consistency



$x$ 를  $G$ 에 넣어  $G(x)$ 를 출력함

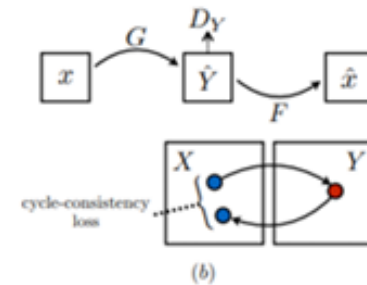


$G(x)$ 를  $D$ 에 제시하여  $D_Y$ 를 출력함



$G(x)$ 를  $F$ 에 넣어  $\hat{x}$ 로 복구함

$$\rightarrow x \mapsto G(x) \mapsto F(G(x)) \approx x$$



## Cycle Consistency

## Backward Cycle Consistency



$y$ 를  $F$ 에 넣어  $F(y)$ 를 출력함

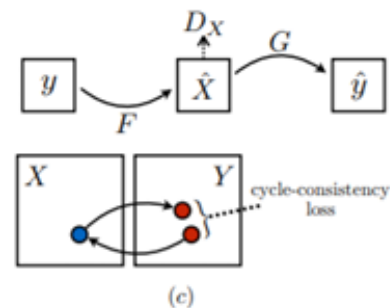


$x$ 와  $F(y)$ 를  $D$ 에 제시하여  $D_X$ 를 출력함



$F(y)$ 를  $G$ 에 넣어  $\hat{y}$ 로 복구함

$$\rightarrow y \mapsto F(y) \mapsto G(F(y)) \approx y$$



## 5

## Image-to-Image Translation

## Cycle Consistency



## Backward Cycle Consistency

Input을 주었을 때 **의도한** 방향으로 바꾸되,

**다시 되돌릴 수 있을 정도로만** 바꾸게 됨

→ **mode collapse**를 해결 할 수 있음!



$y$ 를  $F$ 에 넣어  $F(y)$ 를 출력함

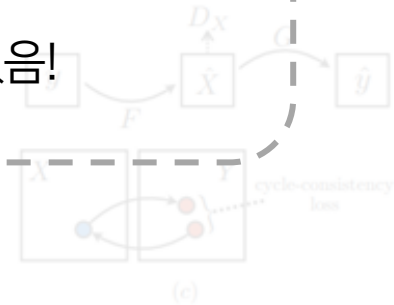


$x$ 와  $F(y)$ 를  $D$ 에 제시하여  $D_x$ 를 출력함



$F(y)$ 를  $G$ 에 넣어  $\hat{y}$ 로 복구합니다

$$\rightarrow y \mapsto F(y) \mapsto G(F(y)) \approx y$$



## 5

## Image-to-Image Translation

## Cycle Consistency



## Backward Cycle Consistency

Input을 주었을 때 **의도한 방향**으로 바꾸되,

**다시 되돌릴 수 있을 정도로만** 바꾸게됨

→ **model collapse**를 해결 할 수 있음!



$y$ 를  $F$ 에 넣어  $F(y)$ 를 출력함



$x$ 와  $F(y)$ 를  $D$ 에 제시하여  $D_x$ 를 출력함



$F(y)$ 를  $G$ 에 넣어  $\hat{y}$ 로 복구합니다



→  $y \mapsto F(y) \mapsto G(F(y)) \approx y$

모델에 들어가는 생성된 데이터와 기존 데이터는

**Pair가 아니라는 점에 주의**해야함

→ 데이터의 **진위여부**를 **구분**하는 용도로만 사용됨

## Cycle GAN

## Loss Function

$$\begin{aligned} & \min_G \max_{D_Y} \mathcal{L}_{GAN}(G, D_Y, X, Y) \\ &= E_{y \sim p_{data}(y)} [\log D_Y(y)] + E_{x \sim p_{data}(x)} [\log (1 - D_Y(G(x)))] \\ & \min_G \max_{D_X} \mathcal{L}_{GAN}(F, D_X, Y, X) \\ &= E_{x \sim p_{data}(x)} [\log D_X(x)] + E_{y \sim p_{data}(y)} [\log (1 - D_X(F(y)))] \end{aligned}$$



두 개의 **Generator**를 잘 학습하는 것이 목표!

## Cycle GAN

## Loss Function

$$\begin{aligned} & \min_G \max_{D_Y} \mathcal{L}_{GAN}(G, D_Y, X, Y) \\ &= E_{y \sim p_{data}(y)} [\log D_Y(y)] + E_{x \sim p_{data}(x)} [\log (1 - D_Y(G(x)))] \\ & \min_G \max_{D_X} \mathcal{L}_{GAN}(F, D_X, Y, X) \\ &= E_{x \sim p_{data}(x)} [\log D_X(x)] + E_{y \sim p_{data}(y)} [\log (1 - D_X(F(y)))] \end{aligned}$$



**Cycle Consistency**가 제대로 작동되고 있는지도 적용해야 함!

## 5

## Image-to-Image Translation

cGAN의 가치 함수



Loss Function

왜 Cycle Consistency가 제대로 작동되고 있는지를 적용해야 할까?

$$\min_G \max_{D_Y} \mathcal{L}_{\mathcal{GN}}(G, D_Y, X, Y)$$

$$= E_{y \sim p_{\text{dlt}}(y)} [\log D_Y(y)] + E_{x \sim p_{\text{dlt}}(x)} [\log (1 - D_Y(G(x)))]$$

$$\min_G \max_{D_X} \mathcal{L}_{\mathcal{GN}}(F, D_X, Y, X)$$

$$= E_{x \sim p_{\text{dlt}}(x)} [\log D_X(x)] + E_{y \sim p_{\text{dlt}}(y)} [\log (1 - D_X(F(y)))]$$

개별적인 모델들이 GAN의 가치함수만을 단독으로 사용해서

제대로 학습을 할 수 있다고 보장하기 어렵기 때문!



Cycle Consistency가 제대로 작동되고 있는지도 적용해야 함!



## Cycle GAN

Cycle-Consistent 보장 함수

$$\mathcal{L}_{cyc}(G, F)$$

$$= E_{x \sim p_{data}(x)} [\|F(G(x)) - x\|_1] + E_{y \sim p_{data}(y)} [\|G(F(y)) - y\|_1]$$



각 변환이 상호연관성을 띠도록

함수의 **Cycle-Consistent**를 보장해야 함→ Cycle-Consistent를 **loss에 추가**하여 보장 할 수 있음

## Cycle GAN

최종 Loss Function

$$\begin{aligned} & \mathcal{L}(G, F, D_X, D_Y) \\ &= \mathcal{L}_{GAN}(G, D_Y, X, Y) + \mathcal{L}_{GAN}(F, D_X, Y, X) + \lambda \mathcal{L}_{cyc}(G, F) \\ & G^*, F^* = \arg \min_{G, F} \max_{D_X, D_Y} \mathcal{L}(G, F, D_X, D_Y) \end{aligned}$$



두 개의 loss Function과  
Cycle-Consistent 보장 함수를 합친  
최종 Loss 함수!

# 5

## Image-to-Image Translation

### Cycle GAN



기존 Image-to-Image Translation 모델의 단점

도메인이 고정되어 있어 **최대 두개의 도메인 간의** Translation만 가능함  
가령 사람표정을 바꾸는 과정에서도 **각 표정 간의 Generator**가 필요함

→ Cross-Domain Translation의 **단점**



StarGAN 통해  
단점 극복

## 5

## Image-to-Image Translation

## StarGAN

## StarGAN

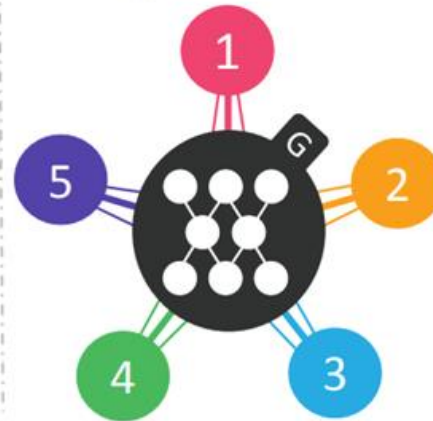
단 하나의 생성만을 이용하여

**Multi-Domain Translation**이 가능하도록 한 모델

(a) Cross-domain models



(b) StarGAN



## StarGAN

## StarGAN

단 하나의 생성만을 이용하여

**Multi-Domain Translation**이 가능하도록 한 모델



(a) Cross-domain models

(b) StarGAN

cGAN과 CycleGAN의 **개념을 차용**하여서

**핵심 개념**을 구축하고 **Multi-Domain Translation**을 실현함

# 5

## Image-to-Image Translation

### StarGAN

#### StarGAN의 생성



우선 세 가지 정보를 바탕으로 Conditional한 조건부 생성 진행



Discriminator가 이미지의 진위 여부만을 판별할 뿐만 아니라 해당 이미지의 class 역시 판별함

$x$ : Input Image

$y$ : Output Image

$z$ : Domain label (one-hot vector)



$$G(x, c) \rightarrow y$$

## StarGAN

## StarGAN의 생성



우선 세 가지 정보를 바탕으로 Conditional한 조건부 생성 진행



Discriminator가 이미지의 진위 여부만을 판별할 뿐만 아니라  
해당 이미지의 class 역시 판별함



$$D: x \rightarrow \{D_{\text{src}}(x), D_{\text{cls}}(x)\}$$

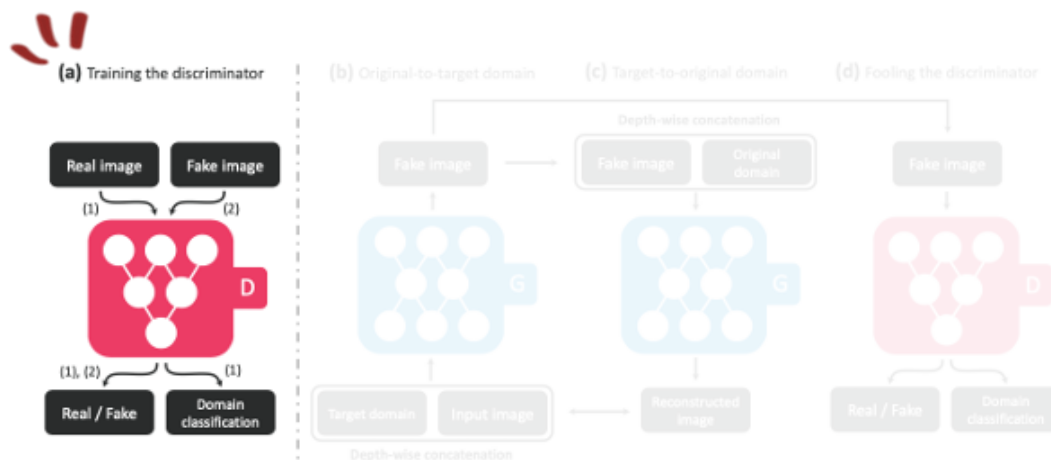
# 5

## Image-to-Image Translation

### StarGAN

#### StarGAN의 학습과정

D가  $G(x)$  와  $y$ 를 받아 **진위 여부를 판단**하고,  
 $y$ 에 대한 **Domain Classification**도 함께 진행





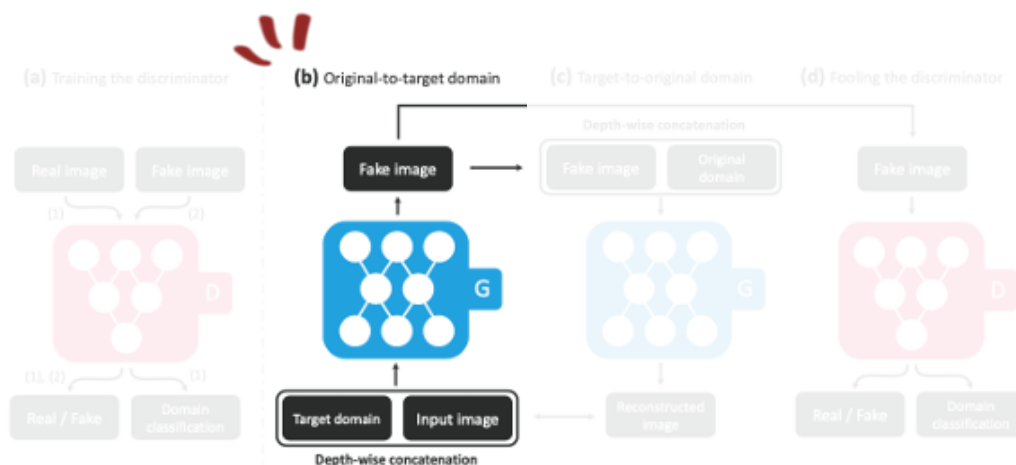
## 5

## Image-to-Image Translation

## StarGAN

## StarGAN의 학습과정

G는 **train domain**인  $c$ 와  
**input image**인  $x$ 를 받아  $G(x)$ 를 **생성함**



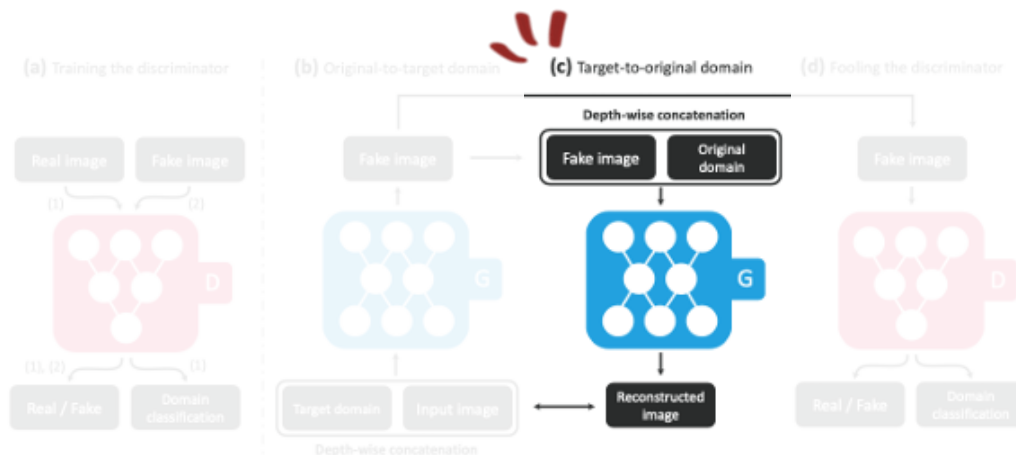
## 5

## Image-to-Image Translation

## StarGAN

## StarGAN의 학습과정

**Cycle consistency**를 위해 만들어진  $G(x)$ 를  
**Reconstruct**하여 원래 이미지로 **잘 되돌아갈** 수 있도록 함



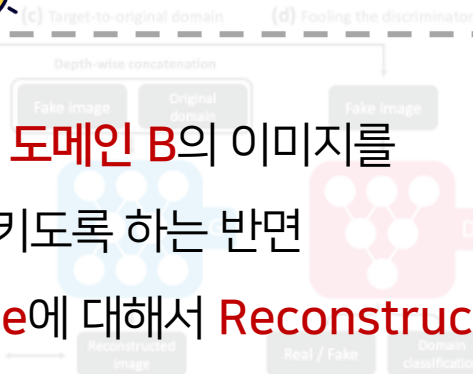
## StarGAN

## StarGAN의 학습과정

Cycle consistency를 위해 만들어진  $G(x)$ 를  
Reconstruct하여 원래 이미지로 잘 되돌아갈 수 있도록 함



이때 cycle GAN은 실제 도메인 B의 이미지를  
도메인 A로 변환시키도록 하는 반면  
StarGAN은 만들어진 Fake Image에 대해서 Reconstruct 진행!



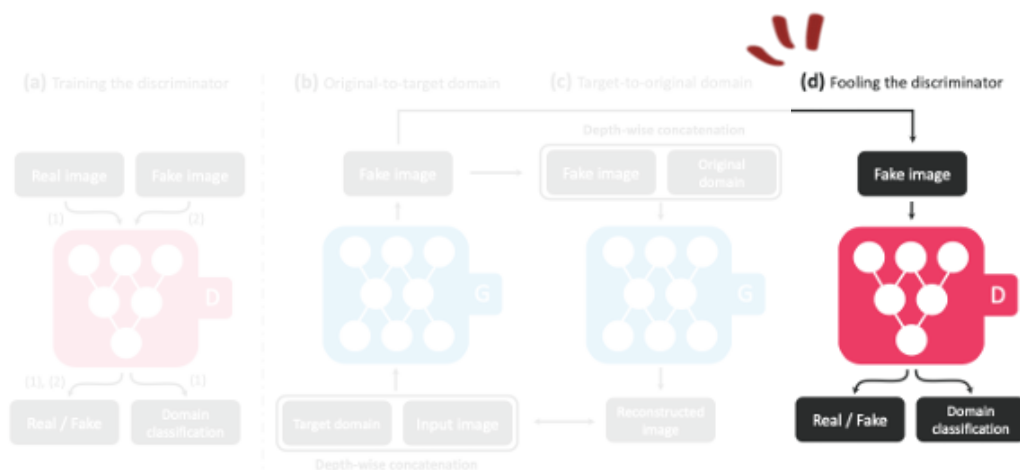
## 5

## Image-to-Image Translation

## StarGAN

## StarGAN의 학습과정

D에 **Fake image**를 넣었을 때 이를 잘 판단하지 못할 때 까지  
**경쟁적 학습**을 진행



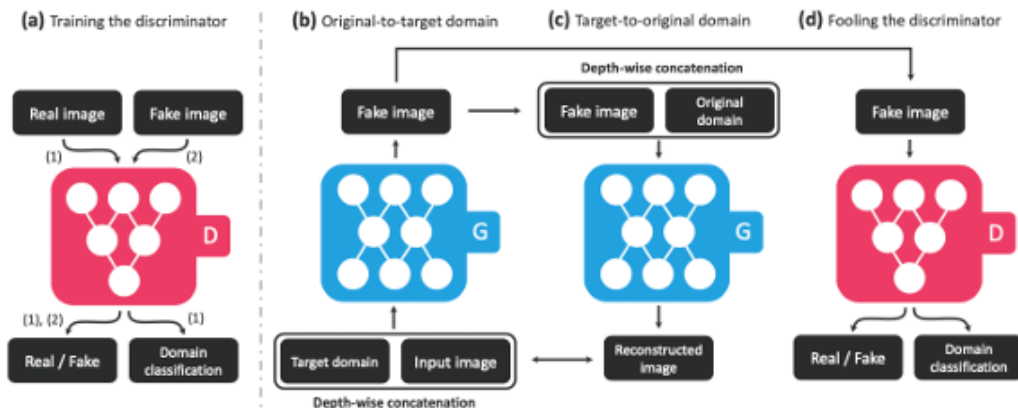
## 5

## Image-to-Image Translation

## StarGAN

StarGAN의 최종 학습 함수

$$L_{\text{adv}} = E_x[\log D_{\text{src}}(x)] + E_{x,c}[1 - \log D_{\text{src}}(x | c)]$$



## 5

## Image-to-Image Translation

## StarGAN



StarGAN의 최종 학습 함수 StarGAN의 목표

$$L_{adv} = E_x[\log D_{src}(x)] + E_{x,c}[1 - \log D_{src}(x | c)]$$

$x$ 와 **target domain label**  $c$ 가 주어졌을 때,

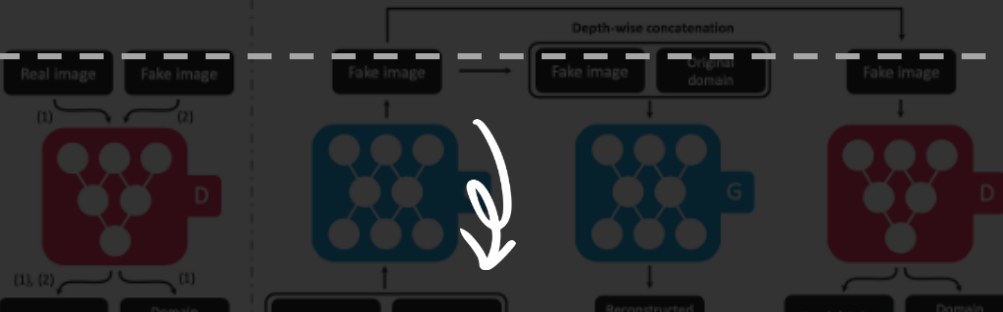
$c$ 로 분류할 수 있는 **output image**  $y$ 를 만드는 것

(a) Training the discriminator

(b) Original-to-target domain

(c) Target-to-original domain

(d) Fooling the discriminator



이를 위해 D에 **class 판별 정보**를 추가하고

이에 따른 **Loss Term**을 Loss 함수에 추가함!

## 5

## Image-to-Image Translation

## StarGAN

## Domain Classification Loss

$$L_{\text{cls}}^r = E_{x,c'}[-\log D_{\text{cls}}(c' | x)]$$

$$L_{\text{cls}}^f = E_{x,c}[-\log D_{\text{cls}}(c | G(x | c))]$$



$D$ 를 **최적화**하기 위하여  
진짜 이미지  $x$ 에 대해  
 $c'$ 로의 **예측할 확률을**  
**높이는 Loss Function**

## 5

## Image-to-Image Translation

## StarGAN

## Domain Classification Loss

$$L_{\text{cls}}^r = E_{x,c'} [-\log D_{\text{cls}}(c' | x)]$$

$$L_{\text{cls}}^f = E_{x,c} [-\log D_{\text{cls}}(c | G(x | c))]$$



가짜 이미지  $G(x | c)$ 가  
 $c$ 로 **예측되기 위한 확률**을  
올리는 것이므로  $G$ 를  
**최적화하는 Function**



## StarGAN

## Reconstruction Loss

$$L_{\text{rec}} = E_{x,c,c'} [|x - G(G(x | c) | c')|]$$



Cycle consistency를 **보존하기 위하여 추가**한 loss로,  
 $c'$ 에서  $c$ 가 되도록 생성한 이미지가 **Reconstruction** 했을 때  
원래 이미지와  $x$ 와의 **차이**를 수식화 한 함수

## StarGAN



WGAN-GP Loss

$$L_D = -L_{\text{adv}} + \lambda_{\text{cls}} L_{\text{cls}}^r$$

$$L_G = L_{\text{adv}} + \lambda_{\text{cls}} L_{\text{cls}}^f + \lambda_{\text{rec}} L_{\text{rec}}$$



Domain Classification Loss와 Reconstruction Loss를  
통합한 형태로 각각 D와 G의 Loss Function을 구성함

## Mask Vector

### Mask Vector

StarGAN에서 특정 도메인에 대한 조건을 명확하게  
정의하기 위해 도입한 One-hot Vector

#### Dataset A

Black, Blond, Brown, Male, Young

#### Dataset B

Angry, Fearful, Happy, Sad, Disgusted

# 5

## Image-to-Image Translation

Mask Vector



Mask Vector

왜 StarGAN은 Mask Vector를 사용할까?

정의하기 위해 도입한 One-hot Vector



Dataset A

Black, Blond, Brown, Male, Young

서로 다른 두 도메인 간의 형태차이로 인한

Dataset B

학습의 어려움을 방지하기 위함!

Angry, Fearful, Happy, Sad, Disgusted

# 5

## Image-to-Image Translation

### Mask Vector

#### Dataset A

Black, Blond, Brown, Male, Young

→ [0,0,1,0,0]

#### Dataset B

Angry, Fearful, Happy, Sad, Disgusted

→ [0,0,1,0,0]



학습을 복잡하게 만드는 요소

A에서 B에 대한 학습을 진행할 때

**Happy로 설정한** 경우와

B에서 A에 대한 학습을 진행할 때

**Brown을 설정한** 경우의

**차이가 없음**

# 5

## Image-to-Image Translation

### Mask Vector

#### Dataset A

Black, Blond, Brown, Male, Young

→ [0,0,1,0,0]

#### Dataset B

Angry, Fearful, Happy, Sad, Disgusted

→ [0,0,1,0,0]



학습을 복잡하게 만드는 요소

Happy한 데이터 B에서  
Brown 머리가 되고 싶은 경우  
이미 [0,0,1,0,0]을 만족하므로

**학습이 제대로 진행되지  
않을 수 있음**

# 5

## Image-to-Image Translation

Mask Vector

Condition Vector

$$\tilde{c} = \{c_1, c_2, \dots, c_n, m\}$$



$c_i$ : Domain에서의 설정 값

$m$ : Mask Vector, 특정 도메인에 대한 **조건을 명확하게 정의**하는 Vector

## 5

## Image-to-Image Translation

## Mask Vector

## Mask Vector 적용 예시



$$\tilde{c} = \{0,0,0,0,0,0,0,1,0,0,0,1\}$$



$$\tilde{c} = \{0,0,0,0,0\}_{c_1} + \{0,0,1,0,0\}_{c_2} + \{0,1\}_m$$



우선 **Happy**한 상태로 이미지를 변환하고 싶다면

**Happy**에 해당하는 **Condition vector**와 **Mask Vector**에 1을 입력함



## 5

## Image-to-Image Translation

## Mask Vector

## Mask Vector 적용 예시



$$\tilde{c} = \{0,0,0,0,0,0,0,0,1,0,0,0,1\}$$



$$\tilde{c} = \{0,0,0,0,0\}_{c_1} + \{0,0,1,0,0\}_{c_2} + \{0,1\}_m$$



이를 하나씩 분해해보면 **Mask Vector**를 통해서  $c_2$ 에 해당하는

**Dataset B**를 설정하고  $c_2$ 에서 **Happy**라는 항목을 설정함

→ 이를 통해 확실하게 Happy를 학습할 수 있음!

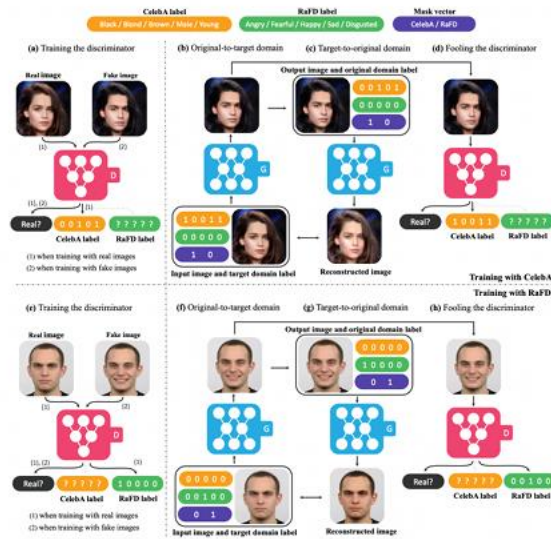
# 5

## Image-to-Image Translation

### Mask Vector

### Multi-Domain Generator

기존 작동방식과 **크게 다르지 않으면서도**  
도메인 별 클래스 정보를 **명확하게 구분** 할 수 있음



# 5

## Image-to-Image Translation

### Mask Vector

#### Multi-Domain Generator

기존 작동방식과 **크게 다르지 않으면서도**  
도메인 별 클래스 정보를 **명확하게 구분** 할 수 있음

이러한 접근 방법의 가장 큰 장점은  
**데이터의 풍부함!**

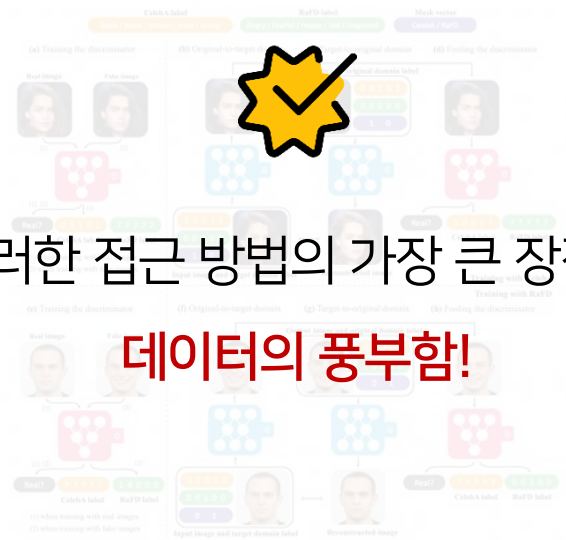


Figure 3: Overview of StarGAN when training with both CelebA and RaFD. (a) ~ (d) shows the training process using CelebA, and (e) ~ (h) shows the missing process using RaFD. (a), (e) The discriminator  $D$  learns to distinguish between real and fake images and minimize the classification error only for the known label. (b), (f), (g) When the mask vector (purple) is  $[1, 0]$ , the generator  $G$  learns to focus on the CelebA label (yellow) and ignore the RaFD label (green) to perform image-to-image translation, and vice versa when the mask vector is  $[0, 1]$ . (d), (h)  $G$  tries to generate images that are both indistinguishable from real images and classifiable by  $D$  as belonging to the target domain.

## StarGAN의 장점

## 기존 Cross-Domain Translation의 단점

각 모델이 특정 데이터로만 Generator를 학습해야 하므로, 데이터의 양이 모델 수 만큼 많아야 함

## Multi-Domain Generator의 적용



모든 도메인의 데이터를 활용하여 진짜 같은 이미지를 생성



동시에 개별 Domain의 데이터를 활용해서 더 현실감 있는 이미지를 생성

# 5

## Image-to-Image Translation

### StarGAN의 장점

#### 기존 Cross-Domain Translation의 단점

각 모델이 특정 데이터로만 Generator를 학습해야 하므로, 데이터의 양이 모델 수 만큼 많아야 함

#### Multi-Domain Generator의 적용



**모든 도메인의 데이터**를 활용하여 진짜 **같은 이미지**를 생성



**동시에 개별 Domain의** 데이터를 활용해서 **더 현실감 있는 이미지**를 생성

## 5

## Image-to-Image Translation

## Mask Vector

## Multi-Domain Generator

다양한 도메인에서 더 많은 데이터로 학습 할 수 있으며,  
이를 통해 이미지의 **질적 향상**을 이룩함

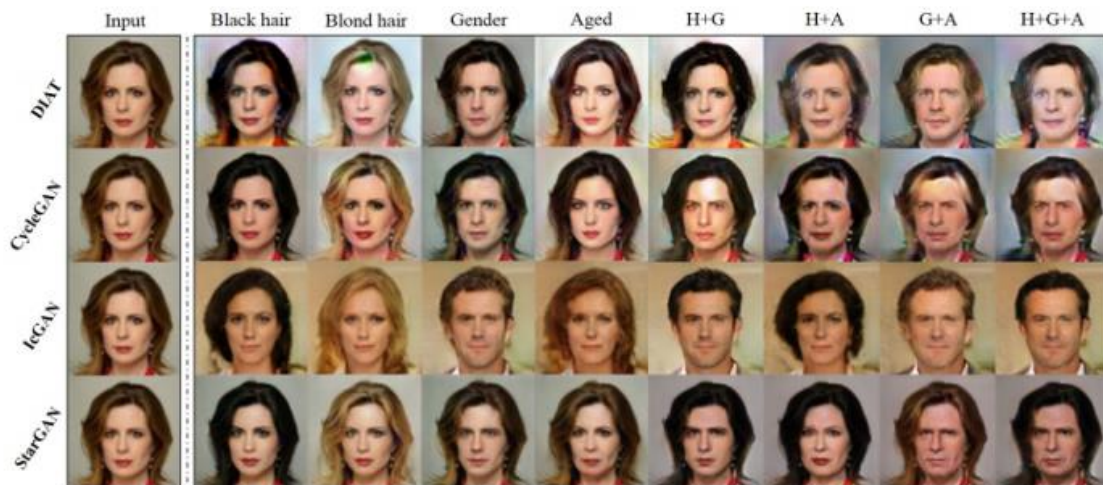


Figure 4. Facial attribute transfer results on the CelebA dataset. The first column shows the input image, next four columns show the single attribute transfer results, and rightmost columns show the multi-attribute transfer results. H: Hair color, G: Gender, A: Aged.

---

**감사합니다**

---