실시간 스타일 변환 어플리케이션 개발과정

생성모델II-GAN

Introduction

앞에서는...

오토 인코더(Auto Encoder), 변이형 오토 인코더(Variational Auto Encoder)에 대해 다루었습니다. 이번 장에서는 GAN의 특징에 대해 다루어 보겠습니다.

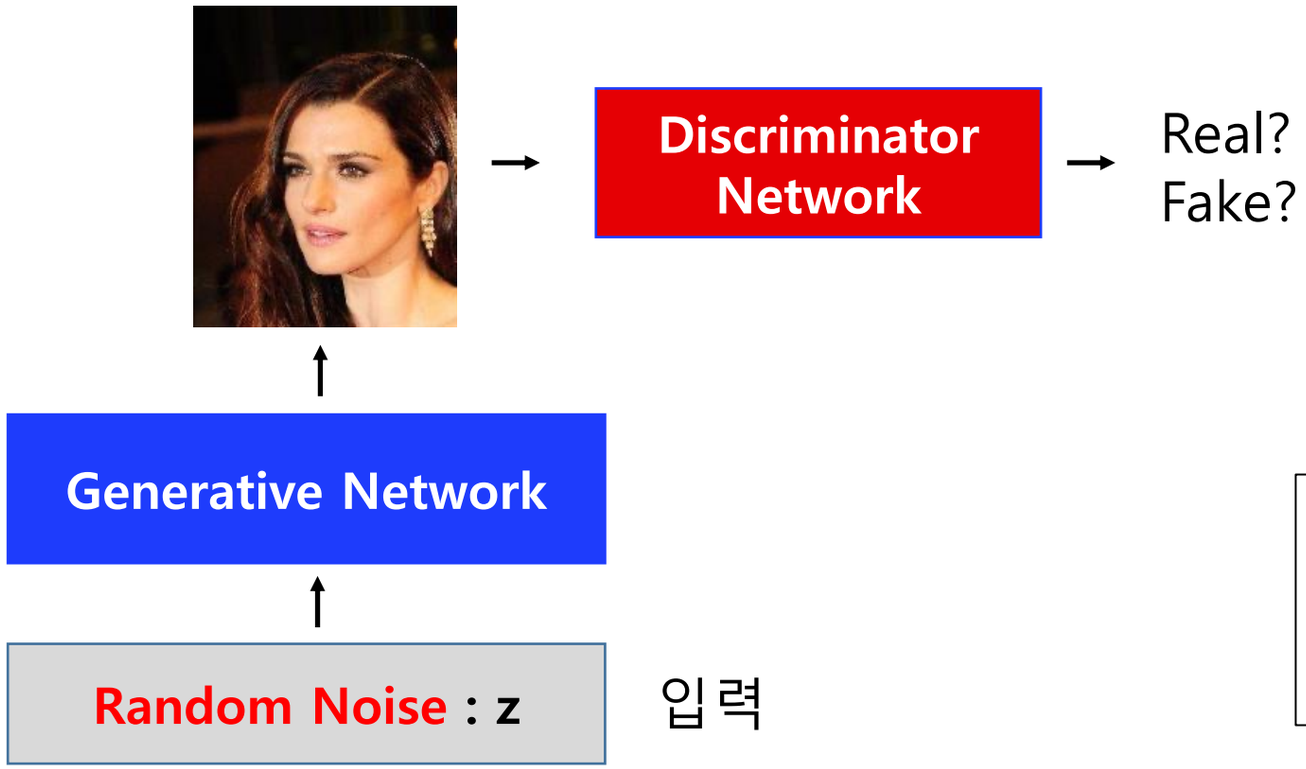
GAN

Generative Adversarial Networks(GANs)

복잡한 고차원 학습 데이터의 분포로부터 데이터를 샘플링 하고 싶으나, 이를 직접적으로 하는 것이 불가능하다. 그러기에, 데이터를 쉽게 샘플링하기 위해 간단한 분포(random noise)를 이용하는 것에서 출발해본다. 그리고 **이 간단한 분포를 학습 분포로 변형하는 법**을 학습한다.

↓

신경망을 사용



각 샘플 z가 어떤 이미지로 맵핑 되는지 알 수 없으나, Discriminator Network을 사용하면 생성된 이미지가 우리가 원하는 데이터 분포 내에 속하는지 판단할 수 있다.

GAN

Generative Adversarial Networks(GANs)

GAN은 위조지폐범(Generator)과 경찰(Discriminator)의 관계로 표현되기도 한다.

From Trainset : 진짜

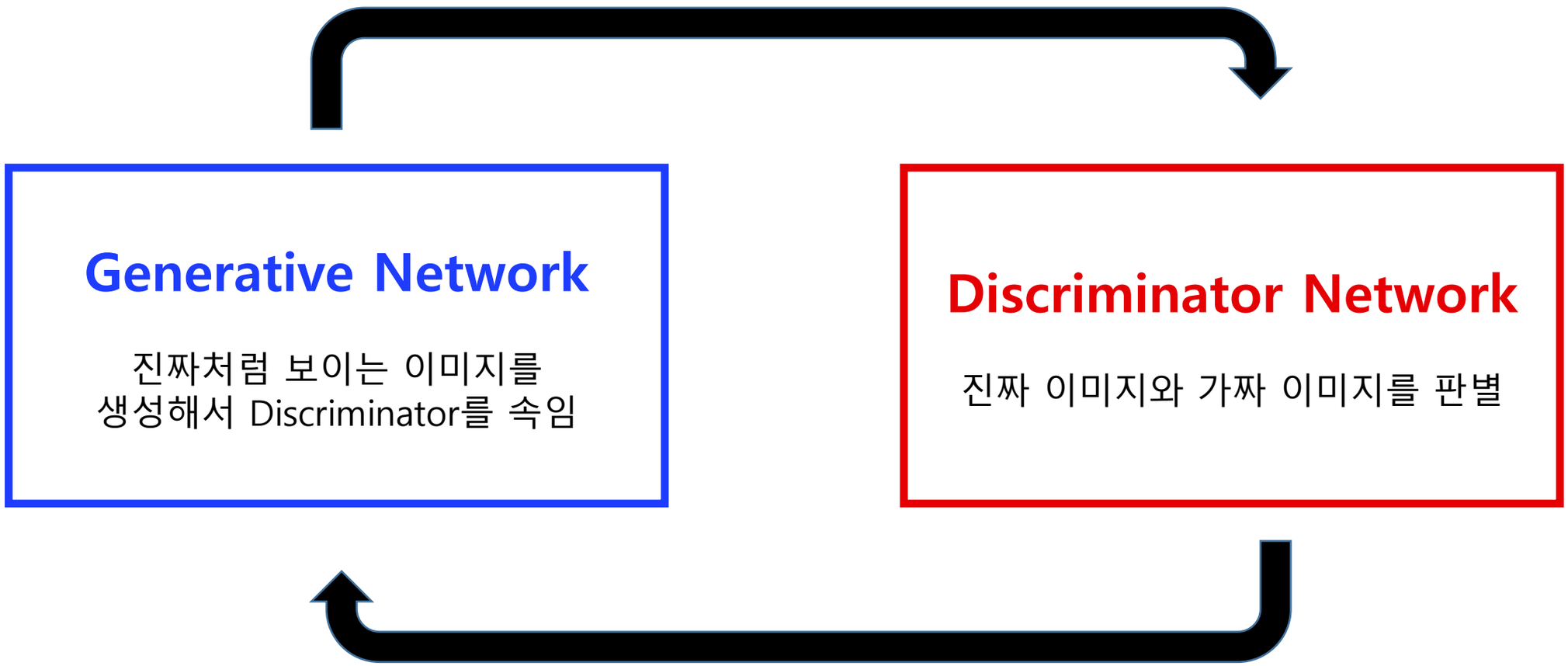
From Generator : 가짜

Discriminator Network → 진짜? 가짜?

Generative Network

GAN

GAN의 학습 : Two player game



GAN

GAN의 학습 : Two player game

Sample from
training distribution

출력

Discriminator Network

Real?
Fake?

Generator Network

학습이 끝나면 Generator Network을 이용하여 새로운 이미지들을 생성한다.

Random Noise : z

입력

GAN

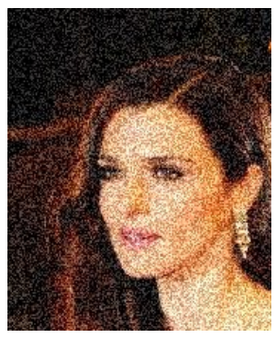
Discriminator에 진짜 이미지가 입력된 경우

x  → **Discriminator Network** → $D(x)$

$D(x)$는 sigmoid를 사용하여
0~1사이의 값을 출력

진짜 이미지인 경우 큰 값 출력

Discriminator에 생성 이미지가 입력된 경우

$G(z)$  → **Discriminator Network** → $D(G(z))$

$D(x)$는 sigmoid를 사용하여
0~1사이의 값을 출력

가짜 이미지인 경우 작은 값 출력

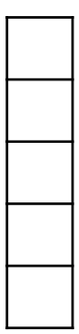
↑

Generator Network

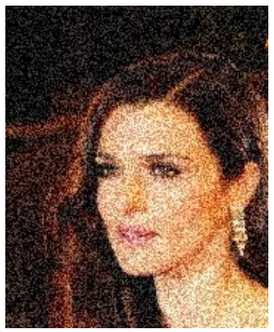
↑

Random Noise : z 입력

GAN



입력
z



$G(z)$

$D(G(z))$

$D(x)$는 sigmoid를 사용하여
0~1사이의 값을 출력

GAN의 목적 함수

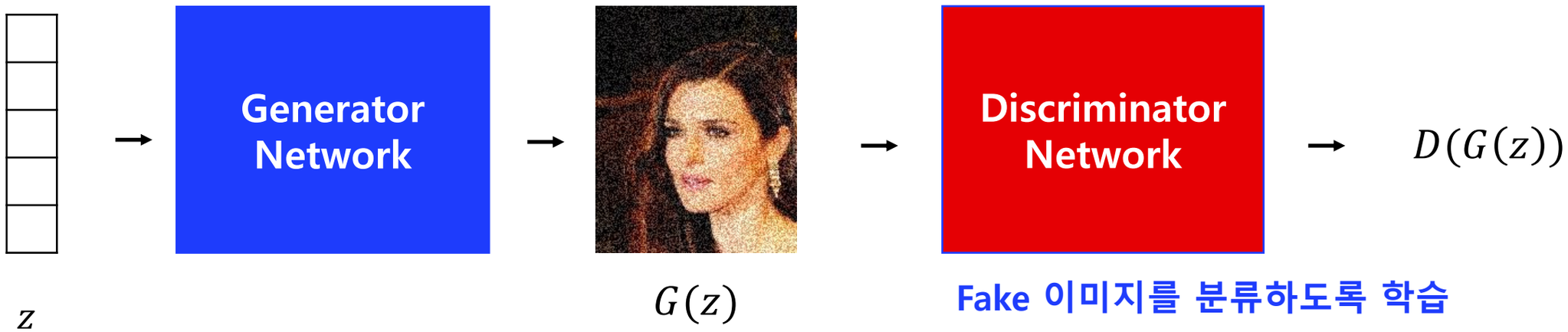
D(Discriminator)는 아래 수식이 최소가 되도록 학습한다.

실제 데이터 분포로부터 x를 샘플링 — 정규 분포로부터 latent code z를 샘플링

$$\max_G \min_D L(D,G) = E_{x\sim p_{data}(x)}[-\log D(x)] + E_{z\sim p_z(z)}[-\log(1 - D(G(z)))]$$

D는 $L(D,G)$를 최소화해야 함 — $D(x) = 1$일 때 최소가 됨 — $D(G(z)) = 0$일 때 최소가 됨

Negative log likelihood = binary cross entropy



Fake 이미지를 분류하도록 학습

$D(x)$는 sigmoid를 사용하여
0~1사이의 값을 출력

GAN의 목적 함수

G(Generator)는 아래 수식이 최소가 되도록 학습한다.

Generator는 관련이 없는 항

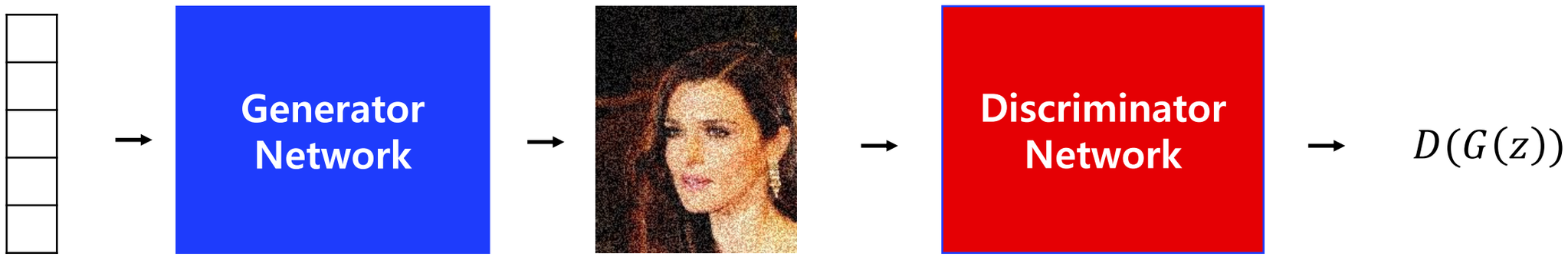
정규 분포로부터 latent code z를 샘플링

$$\max_G \min_D L(D,G) = E_{x\sim p_{data}(x)}[-\log D(x)] + E_{z\sim p_z(z)}[-\log(1 - D(G(z)))]$$

G는 $L(D,G)$를 최대화 해야 함

$D(G(z)$는 1일때 최대가 됨

Negative log likelihood = binary cross entropy



z

D를 속이도록 G를 학습

$G(z)$

$D(x)$는 sigmoid를 사용하여
0~1사이의 값을 출력

Mini-max optimization

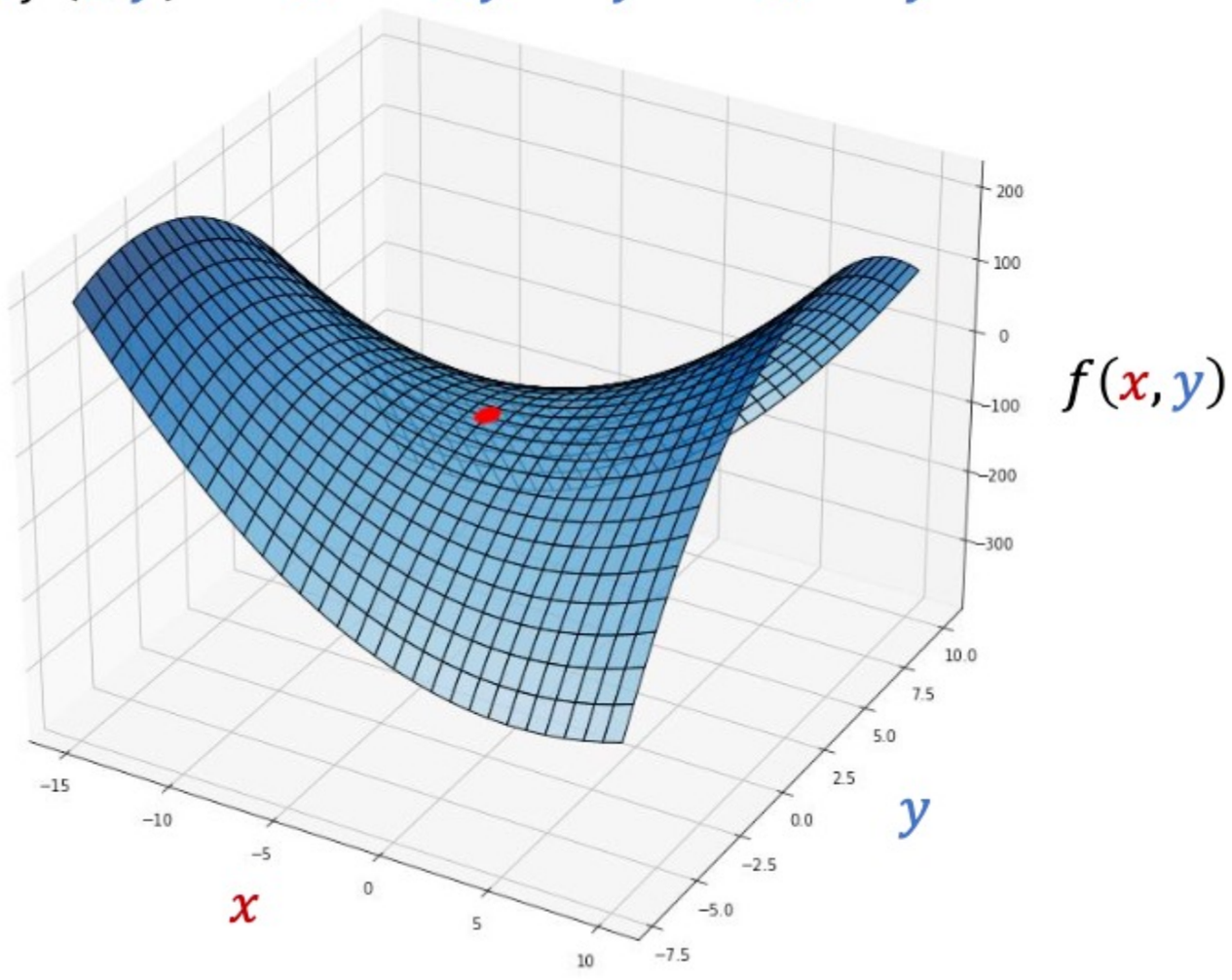
두가지 목적함수로 Generator와 Discriminator를 함께 학습

$$\max_G \min_D L(D,G) = E_{x\sim p_{data}(x)}[-\log D(x)] + E_{z\sim p_z(z)}[-\log(1 - D(G(z)))]$$

Generator의 목적함수

Discriminator의 목적함수

$$\min_x \max_y f(x,y) = x^2 + 2xy - 3y^2 + 4x + 5y + 6$$



$f(x,y)$

Mini-max optimization

$$\min_{x}\max_{y} f(x,y) = x^2 + 2xy - 3y^2 + 4x + 5y + 6$$

(랜덤) 초기값 설정: $x = -1, y = -0.5$

$y = -0.5$ 로 고정할 때, x만의 함수:

$$f(x, y = -0.5) = x^2 - x - 0.75 + 4x - 2.5 + 6 = x^2 + 3x + 2.75$$

$x = -1$ 로 고정할 때, y만의 함수:

$$f(x = -1, y) = 1 - 2y - 3y^2 - 4 + 5y + 6 = -3y^2 + 3y + 3$$

x 만의 함수에 대해, x 에 대한 편미분을 구하면,

$$\frac{\partial f(x, y = -0.5)}{\partial x} = 2x + 3$$

x 만의 함수에 대해, x 에 대한 편미분을 구하면,

$$\frac{\partial f(x = -1, y)}{\partial y} = -6y + 3$$

Mini-max optimization

$$\frac{\partial f(x,y=-0.5)}{\partial x} = 2x + 3$$
$$\frac{\partial f(x=-1,y)}{\partial y} = -6y + 3$$

x는 $f(x, y)$를 최소화하는 방향으로 gradient descent를 수행:

$$x = -1 - \underbrace{0.1}_{\text{learning rate}} \times \left.\frac{\partial f(x, y=-0.5)}{\partial x}\right|_{x=-1} = -1 - 0.1 \times 1 = -1.1$$

y는 $f(x, y)$를 최대화하는 방향으로 gradient ascent를 수행:

$$y = -0.5 + \underbrace{0.1}_{\text{learning rate}} \times \left.\frac{\partial f(x=-1, y)}{\partial y}\right|_{y=-0.5} = -0.5 + 0.1 \times 6 = 0.1$$

Mini-max optimization

$$\max_G \min_D L_{GAN}(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[-logD(x)] + \mathbb{E}_{z\sim p_z(z)}[-\log\left(1-D\left(G(z)\right)\right)]$$

Minimax 목적 함수:

$$\max_G \min_D L(D,G) = E_{x\sim p_{data}(x)}[-\log D(x)] + E_{z\sim p_z(z)}[-\log(1-D(G(z)))]$$

아래의 두 과정을 번갈아 진행:

1. **Gradient descent** on discriminator

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

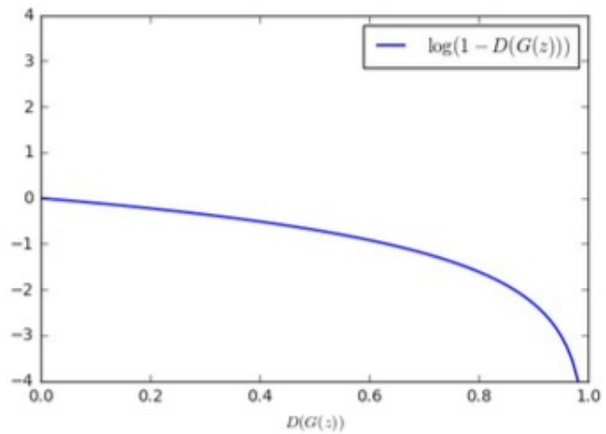
2. **Gradient ascent** on generator

$$\max_G L(D,G) = E_{z\sim p_z(z)}[-\log(1-D(G(z)))]$$

하지만 실제 상황에서는 이러한 Generator 목적 함수가 잘 학습되지 않음

학습 초반에 기울기가 작다는 것이 가장 큰 문제다.

샘플이 가짜처럼 보일 때,
이 샘플을 통해 Generator를 학습하려고 함
(X 축의 오른쪽으로 옮기려 함)



Mini-max optimization

Minimax 목적 함수:

$$\max_G \min_D L(D,G) = E_{x\sim p_{data}(x)}[-\log D(x)] + E_{z\sim p_z(z)}[-\log(1 - D(G(z)))]$$

아래의 두 과정을 번갈아 진행:

1. **Gradient descent** on discriminator

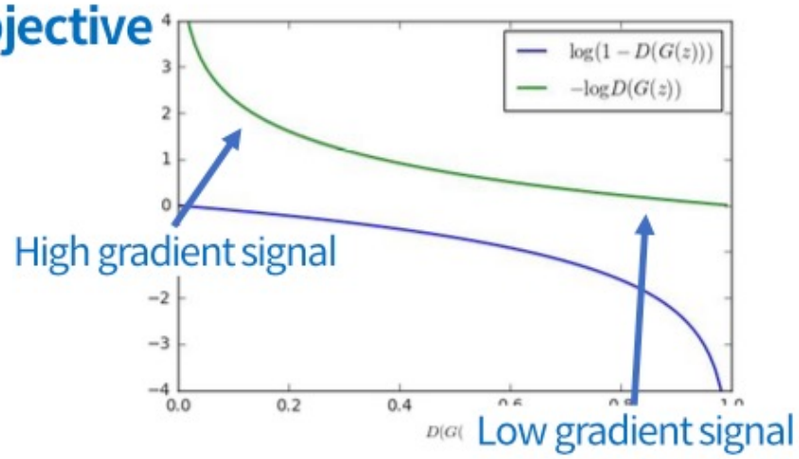
$$\min_D L(D,G) = E_{x\sim p_{data}(x)}[-\log D(x)] + E_{z\sim p_z(z)}[-\log(1 - D(G(z)))]$$

2. **Instead: Gradient descent** on generator using **different objective**

$$\max_G L(D,G) = E_{z\sim p_z(z)}[\log(D(G(z)))]$$
$$\leftrightarrow \min_G L(D,G) = E_{z\sim p_z(z)}[-\log(D(G(z)))]$$

기존과 동일하게 Discriminator를 속이기 위한 목적 함수이지만 가짜 같이 보이는 샘플들에 대한 그라디언트가 커짐.
그 결과, 실제 학습에서 잘 작동함.

Ian Goodfellow et al., "Generative Adversarial Nets", NeurIPS 2014



Instead : Gradient Descent를 사용하면 Fake이미지일때
기울기가 크게 나오므로 학습이 원활하게 이루어진다.

GAN 이미지 생성 모델

- MNIST 데이터 생성
- CIFAR10 데이터 생성

GAN 특징 정리

- 장점
  - ✓뛰어난 품질의 데이터 샘플을 생성한다
- 단점
  - ✓더 까다롭고 안정적이지 못한 학습 과정
- 추가 학습 포인트
  - ✓더 안정적인 학습과 더 좋은 목적함수(LSGAN, Wasserstein GAN)
  - ✓다양한 조건을 수용할 수 있는 Conditional GANs 기법

실시간 스타일 변환 어플리케이션 개발과정

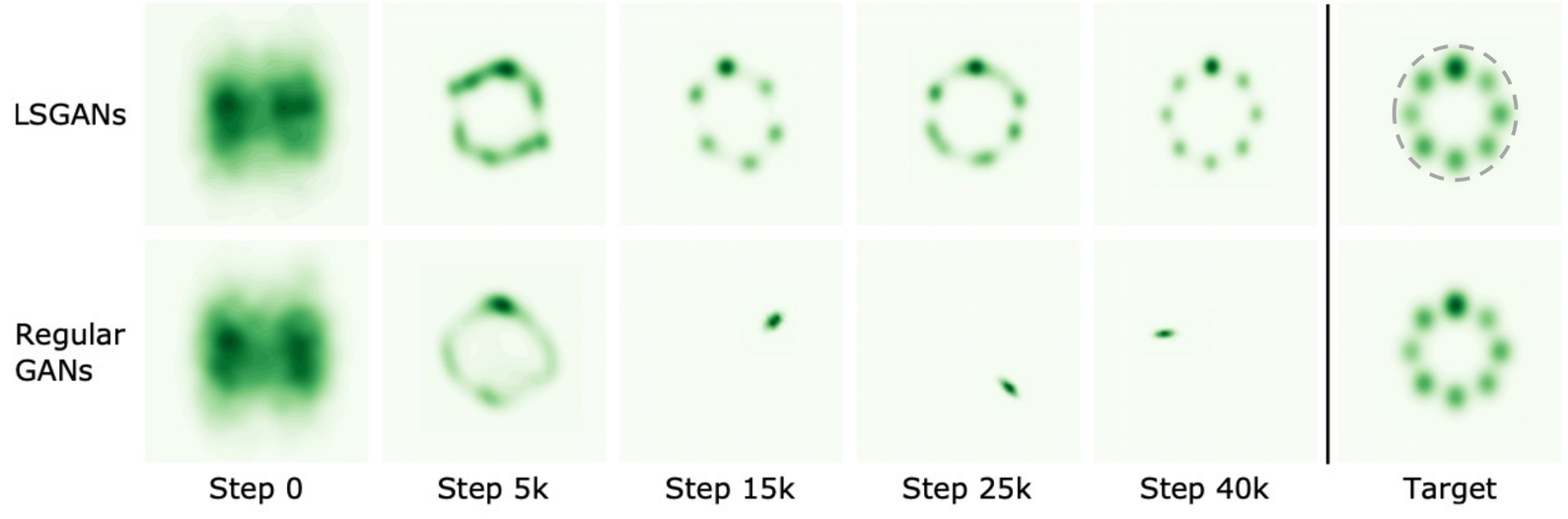
생성모델II – 학습 안정화 기법

LSGAN

LSGAN

Least Squares Generative Adversarial Networks

Xudong Mao[1] Qing Li[1] Haoran Xie[2]
Raymond Y.K. Lau[3] Zhen Wang[4] Stephen Paul Smolley[5]
[1]Department of Computer Science, City University of Hong Kong
[2]Department of Mathematics and Information Technology, The Education University of Hong Kong
[3]Department of Information Systems, City University of Hong Kong
[4]Center for Optical Imagery Analysis and Learning, Northwestern Polytechnical University
[5]CodeHatch Corp.
xudonmao@gmail.com, itqli@cityu.edu.hk, hrxie2@gmail.com
raylau@cityu.edu.hk, zhenwang0@gmail.com, steve@codehatch.com



LSGAN은 기존의 GAN보다 mode-collapsing 없이 더 안정적으로 학습 가능

모든 분포의 데이터를 균등하게 안정적으로 학습이 가능하다.

LSGAN

(a) LSGANs: without BN in G using Adam.

(b) Regular GANs: without BN in G using Adam.

(c) LSGANs: without BN in G and D using RMSProp.

(d) Regular GANs: without BN in G and D using RMSProp.

Figure 6. Comparison experiments by excluding batch normalization (BN).

LSGAN

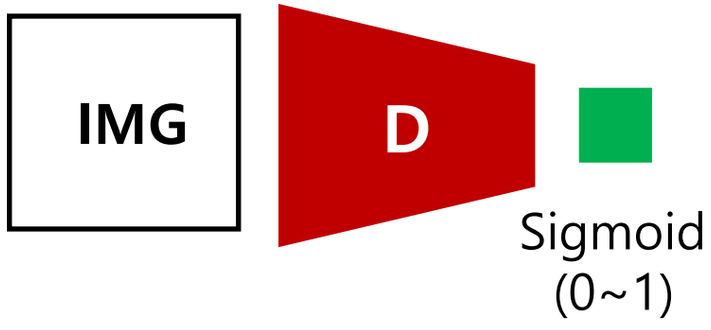
- Least Squares Generative Adversarial Networks(LSGANs)

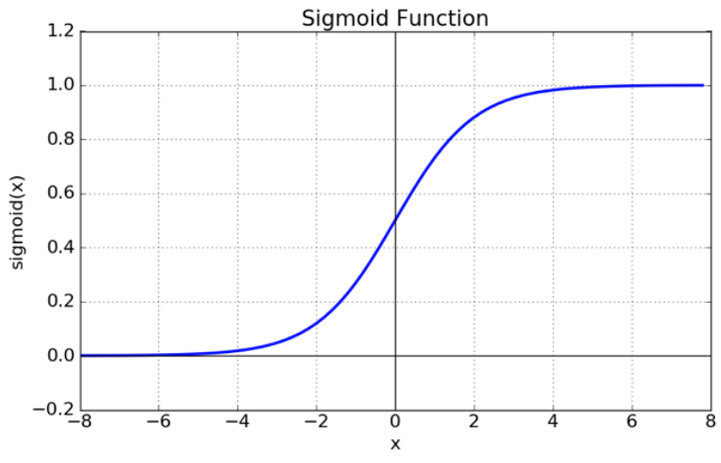
  특징 : Discriminator학습에 binary cross entropy대신 least squares loss를 사용

- **Binary cross entropy loss**

$$\max_G \min_D L_{GAN}(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[-logD(x)] + \mathbb{E}_{z\sim p_z(z)}[-\log\left(1-D\left(G(z)\right)\right)]$$

Vanishing gradients 문제가 발생, 그래디언트 값은 작은데 이미지는 여전히 진짜 이미지 같아 보이지 않는다.





$-\infty$에 가까울수록 $Fake$

$+\infty$에 가까울수록 $Real$

Sigmoid는 양쪽 끝으로 갈수록 기울기가 소실된다.

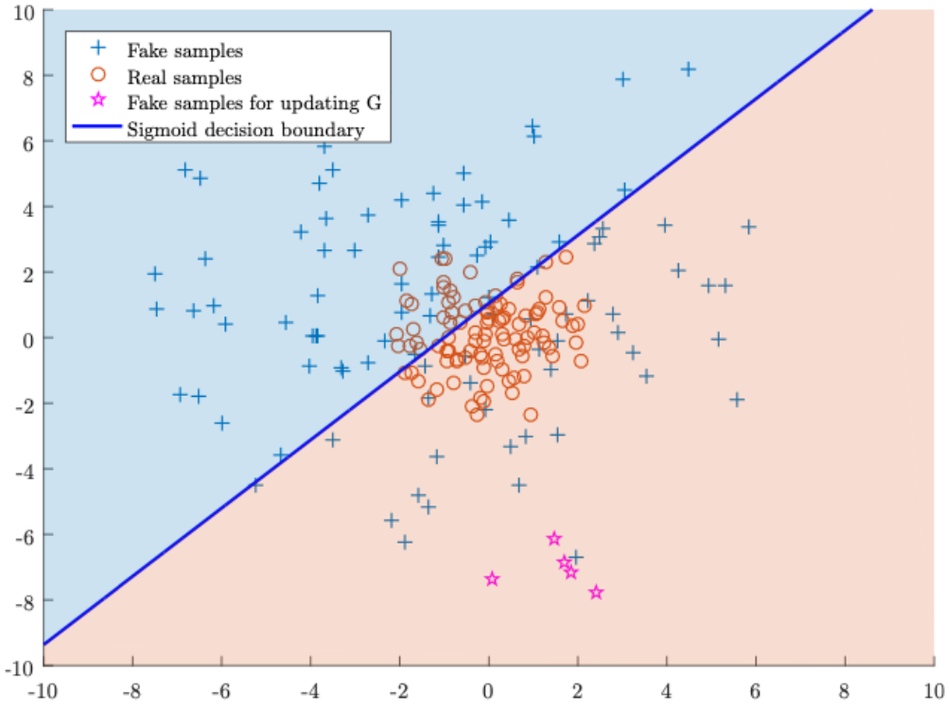
LSGAN

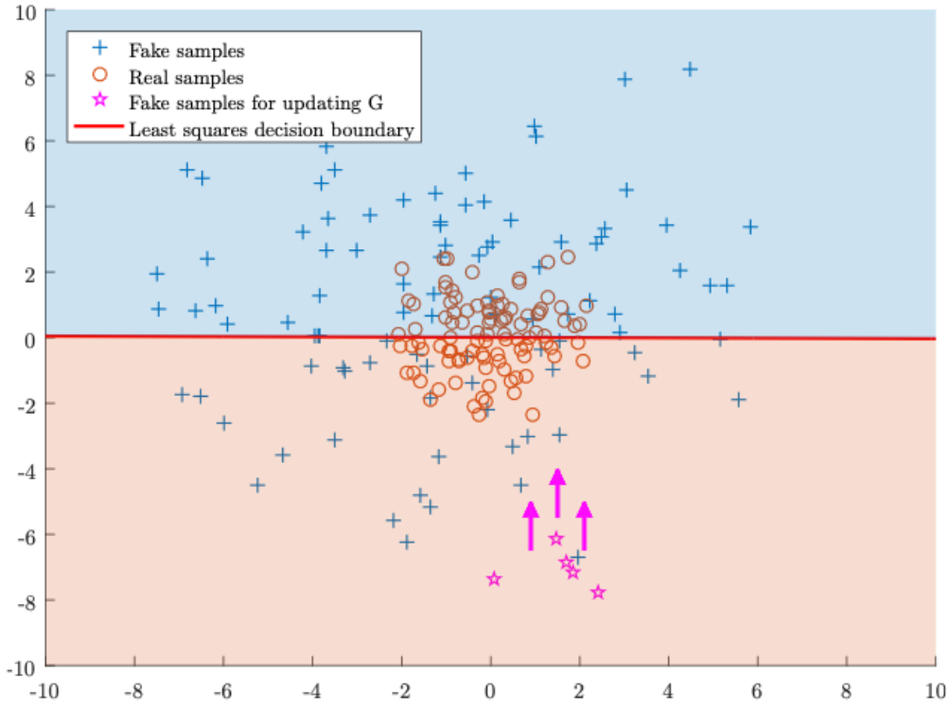
- **Least squares loss**

$$\min_D L_{LSGAN}(D) = \frac{1}{2}\mathbb{E}_{x\sim p_{data}(x)}\left[(D(x)-b)^2\right] + \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}\left[-\log\left(D(G(z)-a)^2\right)\right]$$

$$\min_G L_{LSGAN}(G) = \frac{1}{2}\mathbb{E}_{z\sim p_z(z)}\left[D(G(z)-c)^2\right]$$

(a=0, b=1, c=1)





LSGAN

- **Least squares loss**
  - ✓ 진짜 이미지라고 분류된 Fake 이미지도 Real 이미지와 거리가 멀면 진짜 이미지들과 비슷한 예측 값을 갖도록 학습됨
  - ✓ 기존 GAN에 비해 이미지들의 품질이 향상됨
  - ✓ Vanishing Gradient 문제를 완화해서 더 안정적으로 학습이 가능

인공지능 심화 과정

생성모델II – 학습 안정화 기법

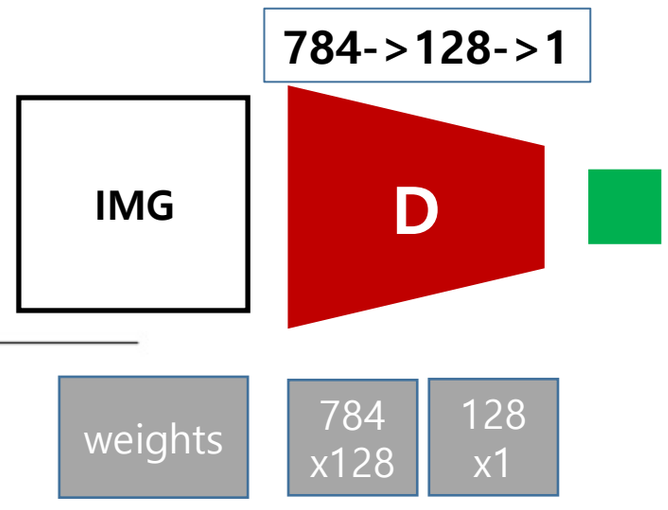
Wasserstein GAN-GP

Wasserstein GAN – GP(Gradient Penalty)

Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

Require: The gradient penalty coefficient λ, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2.
Require: initial critic parameters w_0, initial generator parameters θ_0.

while θ has not converged **do**
  for $t = 1, ..., n_{\text{critic}}$ **do**
    for $i = 1, ..., m$ **do**
      Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0, 1]$.
      $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
      $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
      $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$
    end for
    $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2)$
  end for
  Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^{m} \sim p(\boldsymbol{z})$.
  $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
end while



Discriminator의 Loss함수에 규제항을 추가한다.

iteration당 기울기의 변화율에 규제항을 추가하여 안정적으로 학습이 되게 한다.

Pytorch 구현 코드 참조 : https://github.com/eriklindernoren/PyTorch-GAN

GAN의 학습 안정화 기법

- Generator와 Discriminator의 2 player game을 통해 학습하는 GAN 모델은 어느 한 쪽이 과도하게 학습되고 다른 쪽은 전혀 학습이 되지 않아 학습이 불안정해 질 수 있다.
- LSGAN, WGAN-GP와 같이 목적 함수를 새롭게 정의하여 추가적인 안정화를 얻을 수 있다.