

DEEP LEARNING

Clean-Up Week 3

1 팀

김원구 강다빈 박서영 전규리 정연준

INDEX

0. 지난주 복습

1. 생성모델 소개

2. Auto Encoder

3. GAN

0

지난 주 복습

Tensor

임의의 차원을 가지는 행렬의
일반화된 모습
즉, 다차원 배열



텐서에서 **차원 = 축 (axis)**

1

scalar
(0D 텐서)

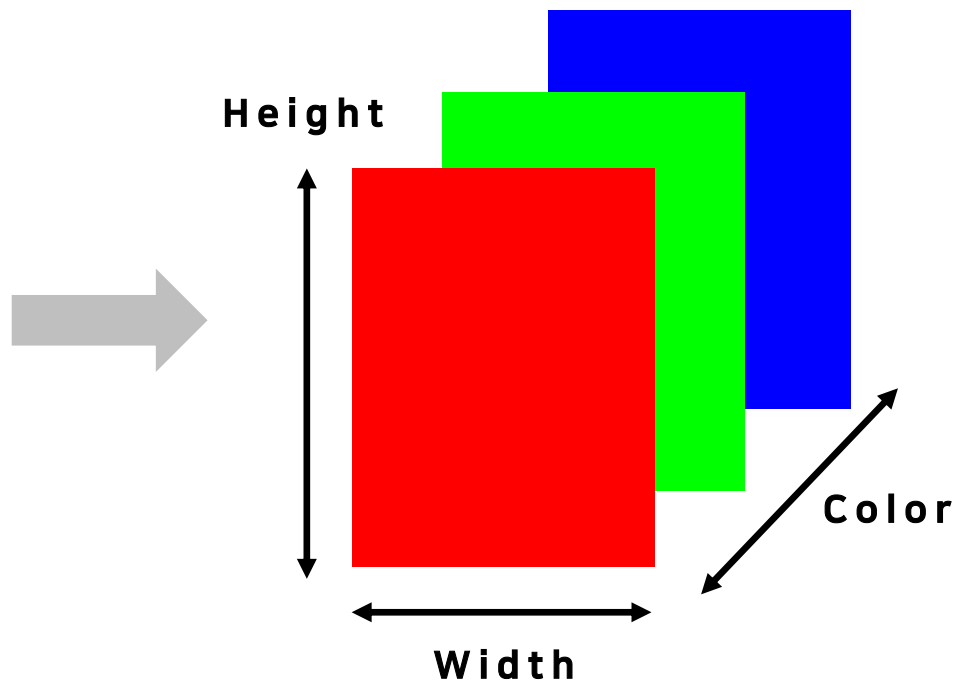
[1,2,3]

Vector
(1D 텐서)

[[1,2],[3,4]]

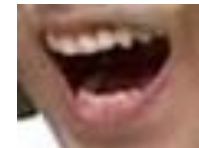
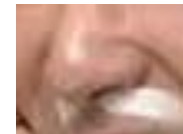
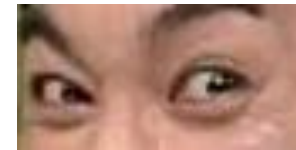
Matrix
(2D 텐서)


Tensor



컬러 이미지 1장은 3차원 텐서
[height 축, width 축, color축]

CNN의 특징



 사진 및 영상의 공간적 특성을 최대한 활용

 가중치 및 편향을 공유하여 학습시간 단축

 사진 및 영상 구별 능력 UP

CNN의 구성요소

Convolutional Layer



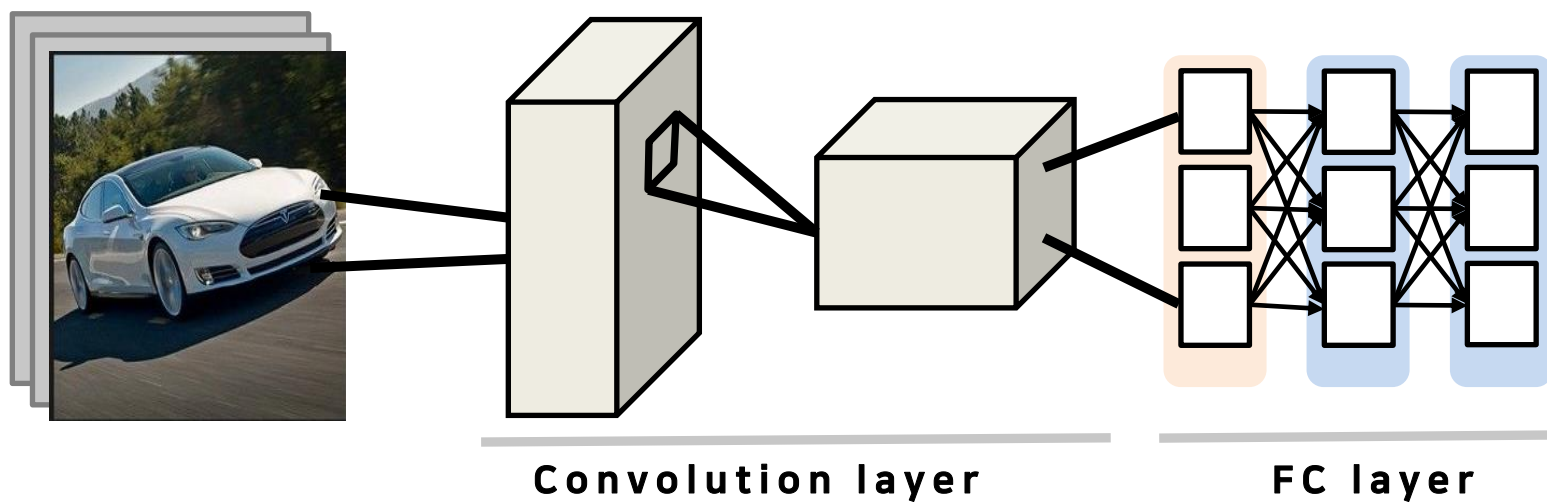
| | | |
|---|---|---|
| 2 | 2 | 2 |
| 3 | 2 | 3 |
| 2 | 3 | 3 |

| | | |
|----|----|----|
| 10 | 11 | 12 |
| 11 | 10 | 12 |
| 12 | 10 | 10 |

공간적으로 가까운 픽셀끼리 비슷한 값 갖거나 color 서로 밀접하게 관련
이런 공간적 정보가 들어있는 3차원 데이터를 처리하기 위해
Convolutional layer 사용

CNN의 구성요
소

Convolutional Layer



입력 받은 이미지에 대한 **특징을 Filter를 통해 추출**
추출 후 평탄화 하여 기존 신경망과 같이 FC Layer를 이용해 분류

Filter (Kernel)

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

**Weights**

학습을 거쳐 업데이트

데이터의 **특징을 추출**하는 기능

Convolution layer에서 원본 데이터에
filter를 적용하여 어떤 특징을 가지는지 판단

Filter의 예시



여러 개의 Convolution layer를 지나면 결과값이 작아져
특징이 유실될 수 있음
Padding으로 해결!

Feature Map

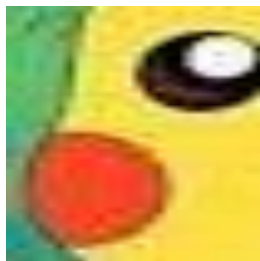
| | |
|---|---|
| 7 | 5 |
| 7 | 8 |

CNN의 구성요
소

Padding



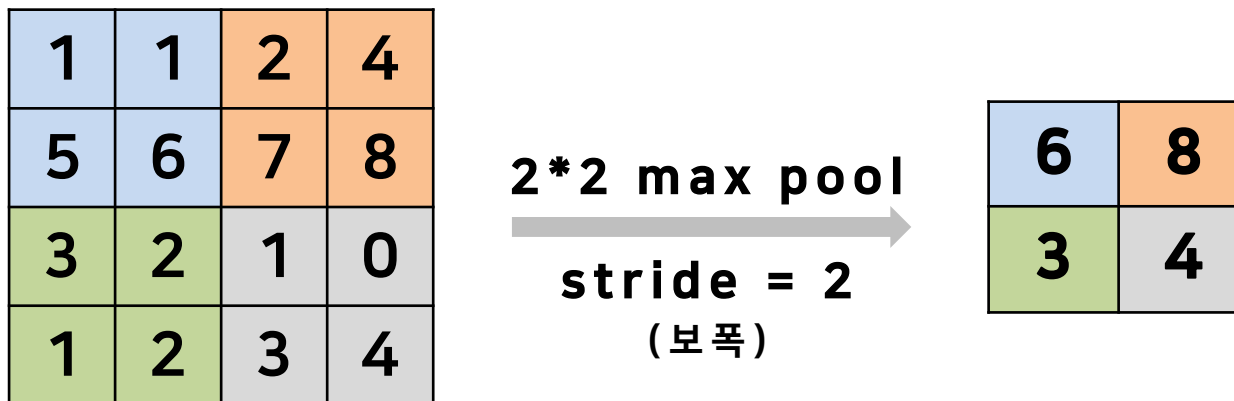
- ✓ 입력 값 주위로 0을 붙여 인위적으로 데이터 크기 키움
- ✓ 결과값이 작아져서 특징이 유실되는 것 방지
- ✓ 과적합 방지



Convolution Layer를 통해 추출된
모든 특징들을 사용할 필요 없음

중요한 정보만을 남겨 차원을 축소하는 Pooling 진행

대표적인 Pooling 기법으로 Max Pooling



Feature Map

Feature map을 자른 후, 각 조각에서 **가장 큰 값**을 추출
일반적인 특징보다 **뚜렷한 특징**을 찾아내기 위해!!

대표적인 Pooling 기법으로 Max Pooling

Pooling은 층마다 적용하는 것이 아닌, **선택적으로 사용**
계산량이 줄고, 과적합을 방지할 수 있음

Feature Map

Feature map을 자른 후, 각 조각에서 **가장 큰 값**을 추출
일반적인 특징보다 **뚜렷한 특징**을 찾아내기 위해!!

RNN이란?



다음 데이터에 이전 데이터에 대한 정보 반영 ㄱ?

FFNN



쓰버ㄱ...





다음 데이터에 이전 데이터에 대한 정보 반영 ㄱㄴ?

RNN



쓰ㄱㄴ!!



How?

Hidden layer의 output을
다음 input과 함께 입력

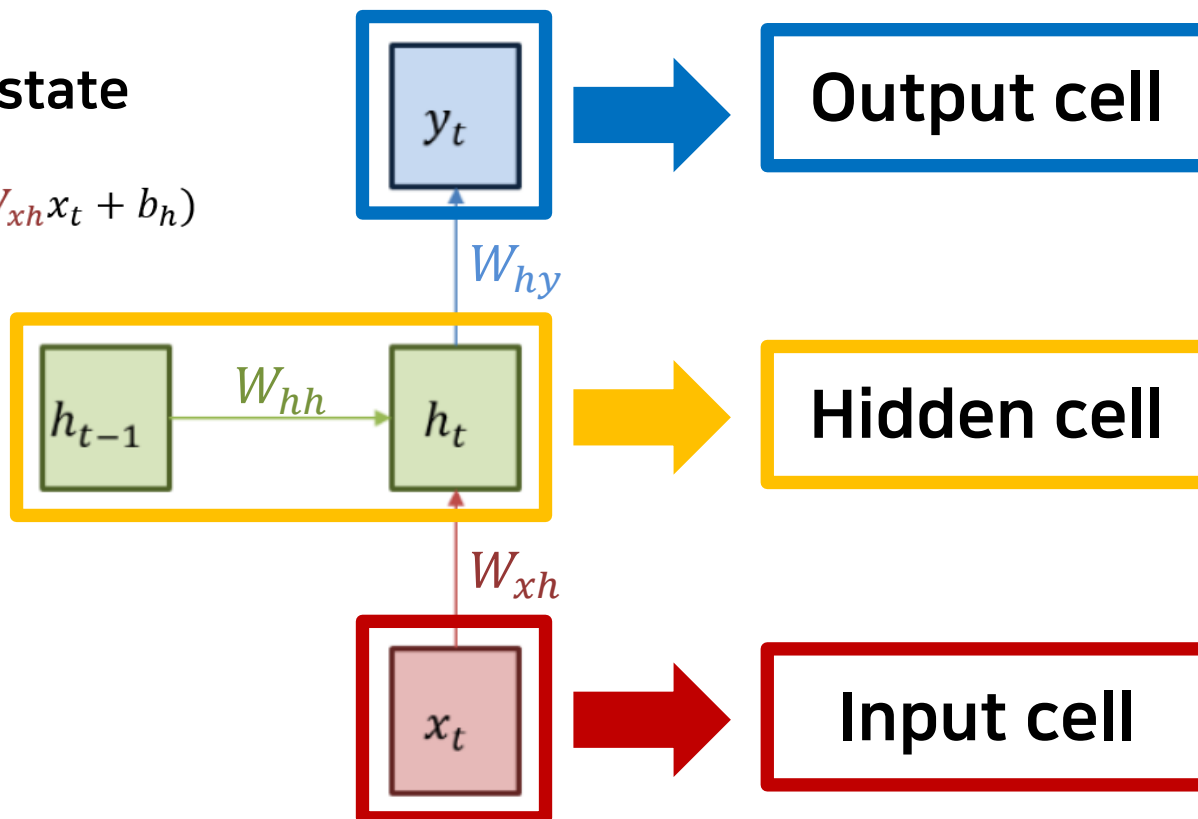
RNN의 구조



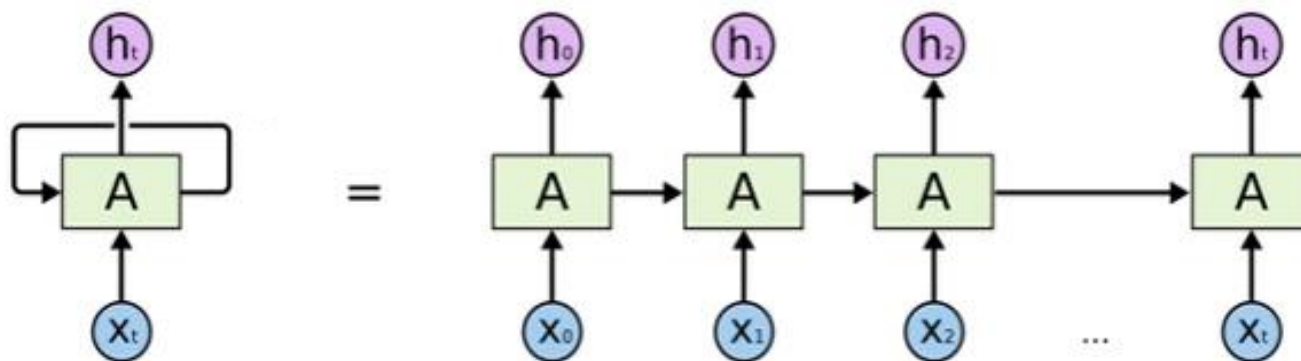
h_{t-1}, h_t : Hidden state



$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$



RNN의 구조



음성, 문자 등 순차성을 가진 비정형 데이터 처리에 적합한 구조

모든 time step마다 가중치들은 모두 같은 값을 가짐

Ex) 여자친구에게 선물을 줬다. 선물을 받은 그녀는 기뻐했다.



‘그녀’를 알기 위해 이전 문장의 정보가 필요

활성화 함수

왜 RNN의 활성화 함수로 \tanh 를 사용할까?

Vanishing gradient가 문제라면

Relu를 사용하면 되지 않을까?



현재 값이 다음 값 계산에 영향
+
모든 time step의 가중치 공유

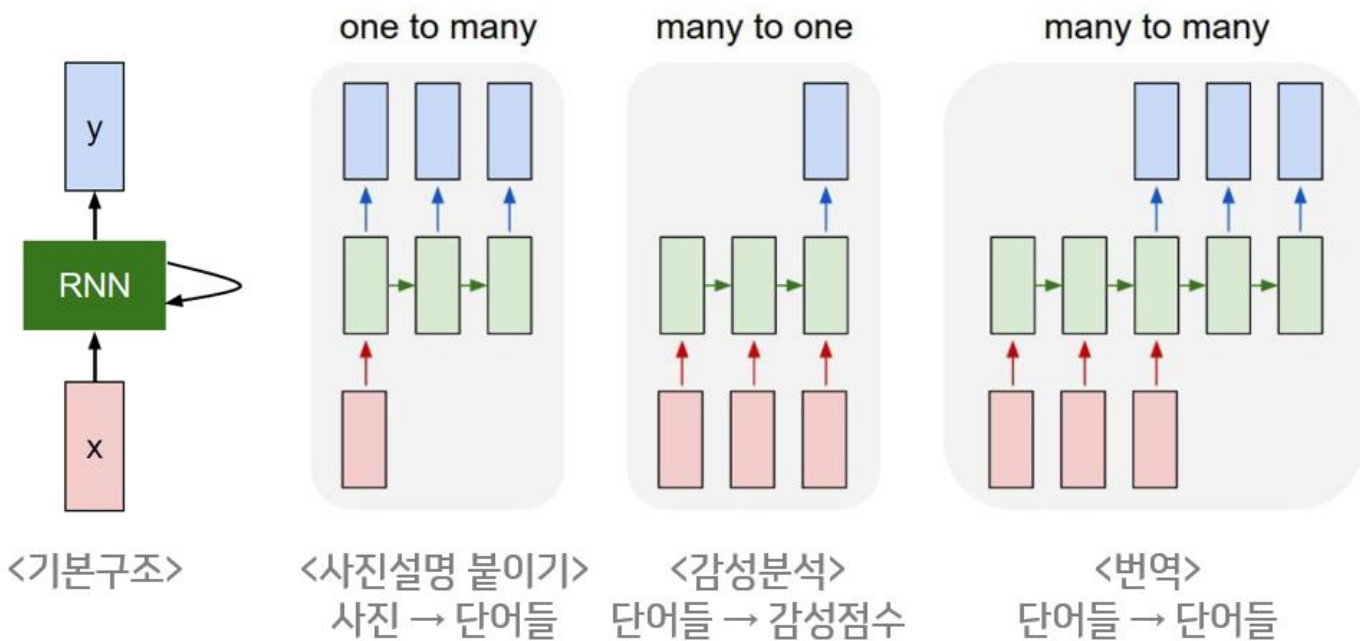
Hidden state의 각 원소에 반복하여 같은 가중치가 곱해짐

→ 값이 기하급수적으로 증가 or 감소

→ Relu 사용시, $x \geq 0$ 이면 출력값이 발산 or exploding gradient 발생

→ Hidden state를 제대로 반영하지 못한 불안정한 학습

RNN의 쓰임새



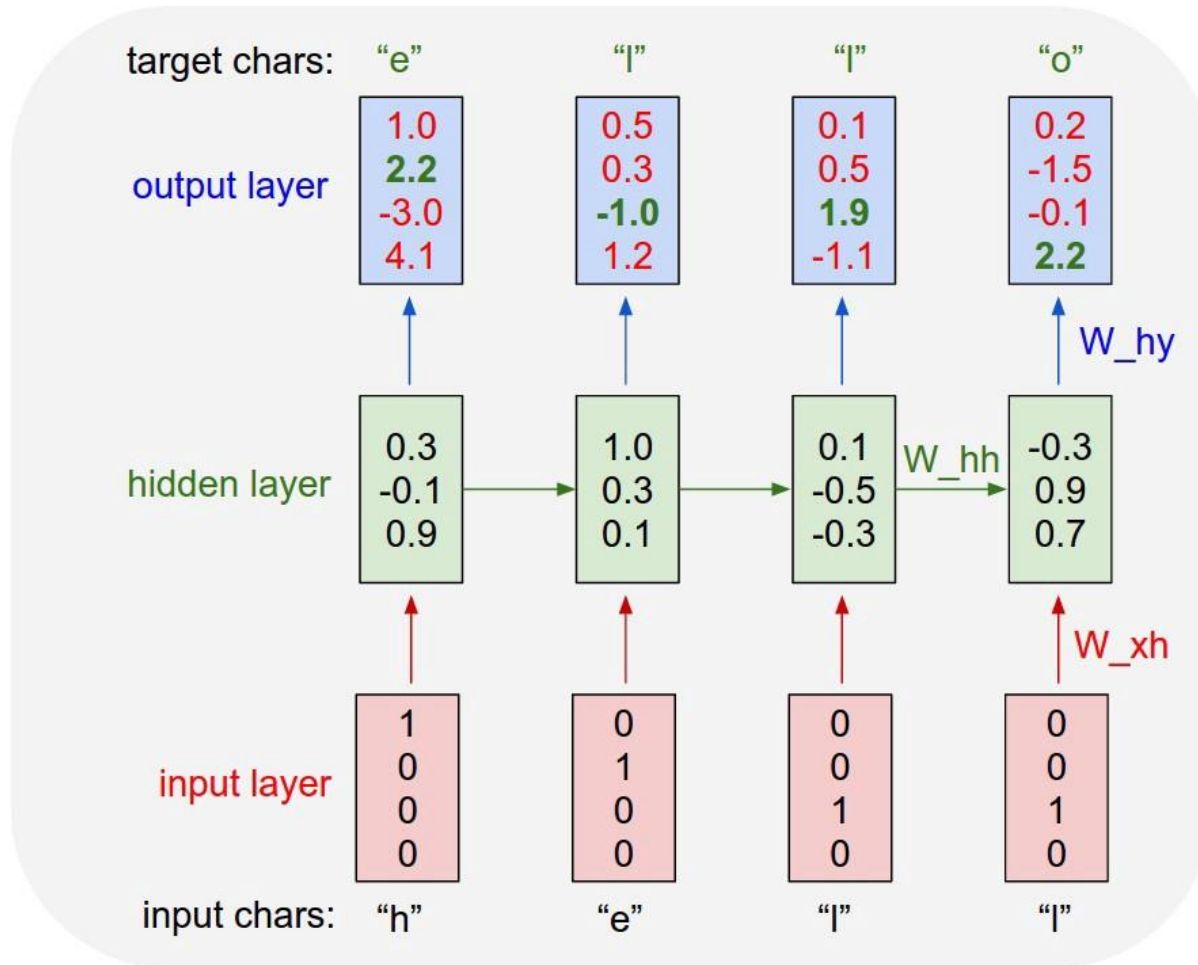
RNN은 입출력이 매우 자유로움



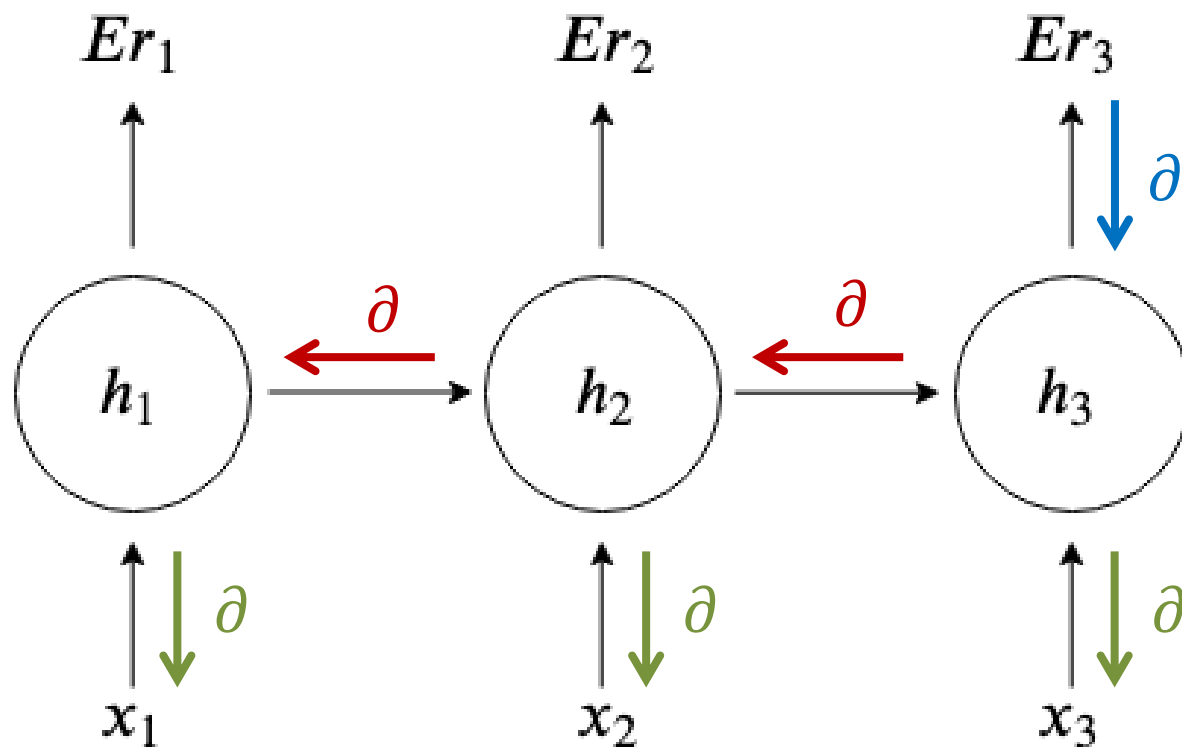
필요에 따라 다양하고 유연한 구조 가능

RNN의 학습과
정

Example



BPTT



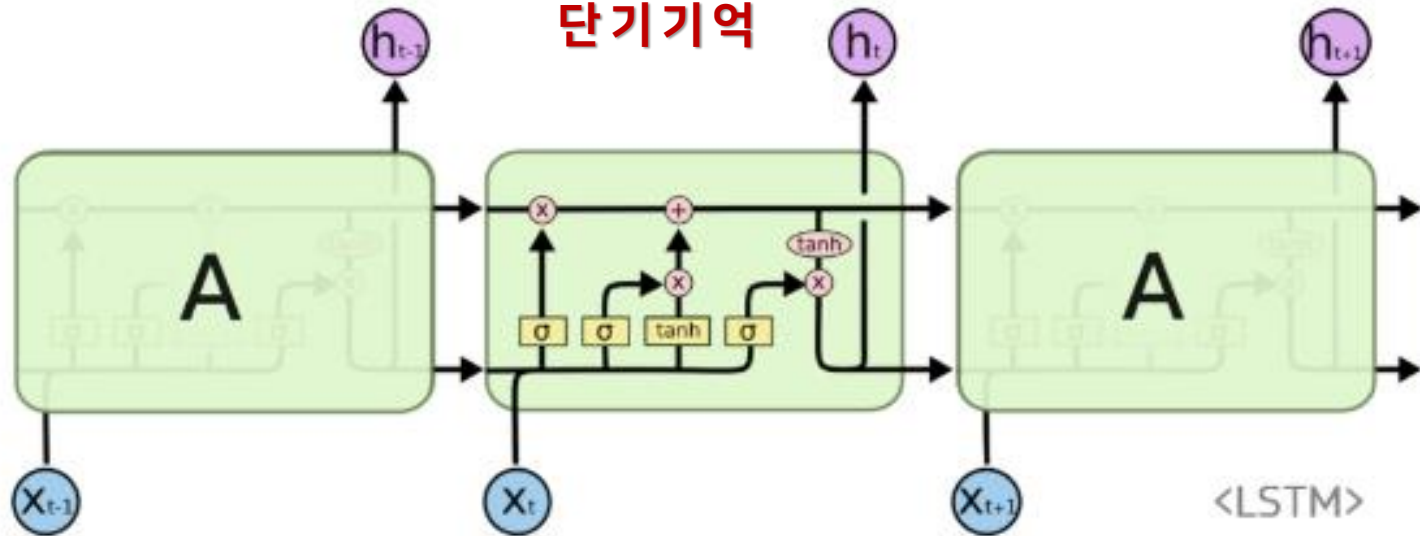
Activation 함수로 \tanh 를 사용하므로 time step이 길어질수록
gradient가 끝까지 전달되지 못함

➡ 결국 **Vanishing gradient** 문제에서 자유롭지 못함

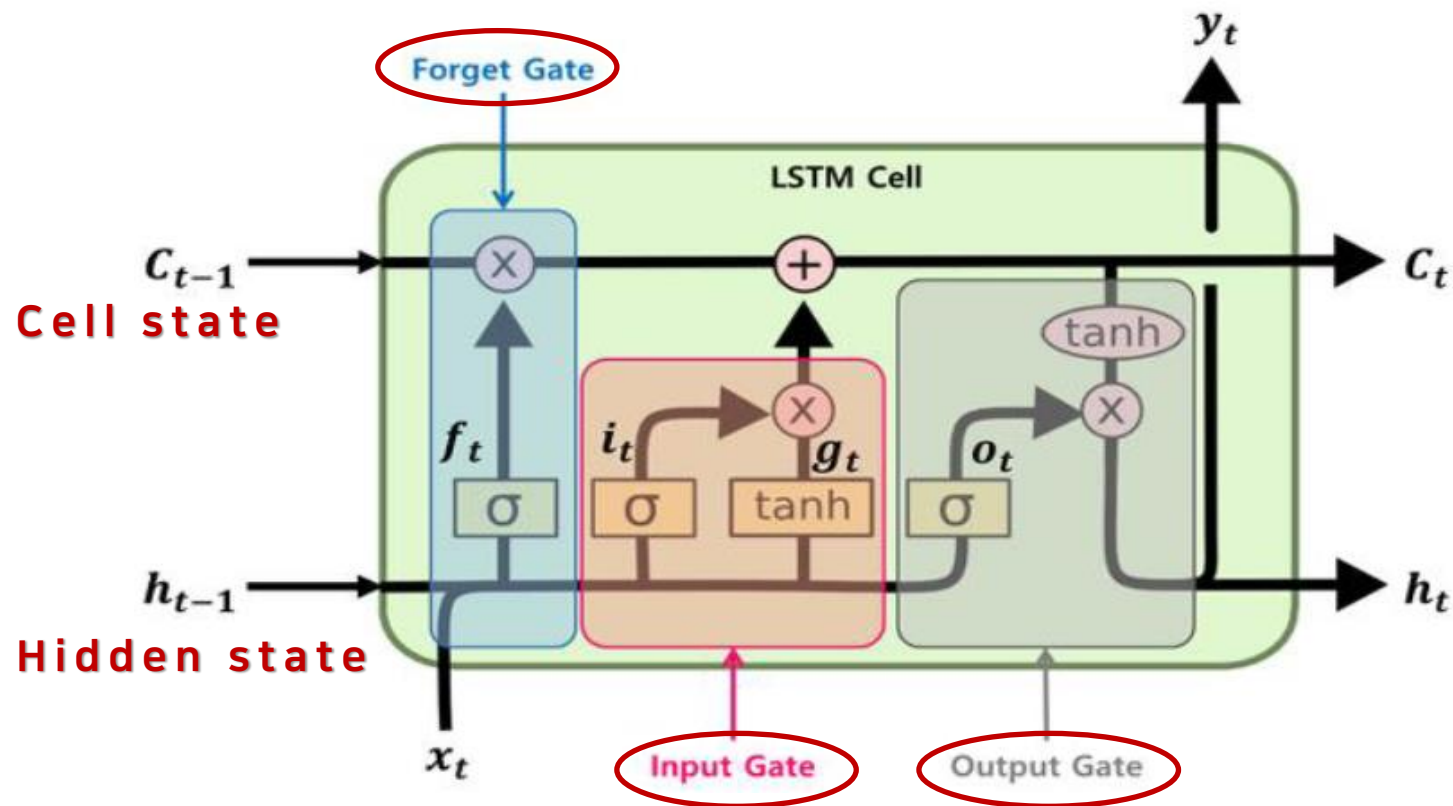
실제 적용상 긴 문맥을 학습할 수 없다

ex) 나는 프랑스에서 자랐습니다. (대략 1억5천8백6십3만개 문장)
나는 000어를 잘합니다.

LSTM이란?

Long Short-Term MemoryCell state +
장기 기억Hidden state
단기 기억

LSTM의 구조



1

Introduction to the Generative Model

1

Introduction to the Generative Model

생성 모델

Output이 Input 데이터와 매우 비슷하도록 새로운 Dataset을 만들어내는 모델

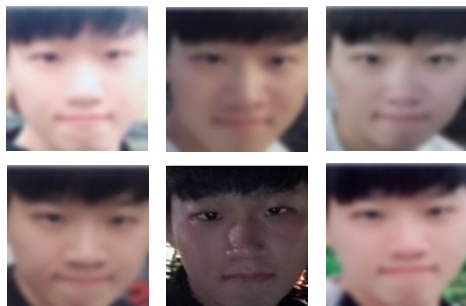
훈련 데이터



생성 모델



생성된 샘플



모델의 분포(P_{model})가
입력 데이터의 분포(P_{data})에
근사해지도록 훈련



1

Introduction to the Generative Model

판별 모델

데이터 X 가 주어졌을 때 레이블 y 의 확률($P(y|x)$)을 추정하는 모델

훈련 데이터



생성 모델



생성된 샘플

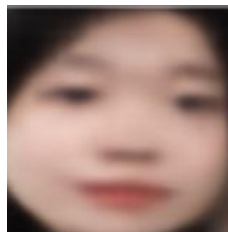


1



0

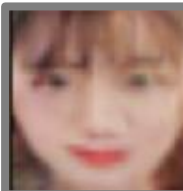
한 개의 샘플



0.83



1



0

샘플 레이블

1

Introduction to the Generative Model

생성 vs 판별

생성 모델

데이터 X 가 주어졌을 때
관측될 확률($P(X)$)을
추정하는 모델

데이터를 발견할 확률을 추정
(레이블에는 관심 없음)

판별 모델

데이터 X 가 주어졌을 때
레이블 y 의 확률($P(y|X)$)을
추정하는 모델

즉, X 가 y 에 속할 확률을 추정

1

Introduction to the Generative Model

생성모델과 딥러닝

머신러닝 생성 모델

변수간 상관관계가 커지고
차원이 높아질수록
제대로 생성 불가
(ex. 이미지)

딥러닝 생성 모델

원본 데이터의 고차원 특징을
저차원 공간으로 투영,
데이터를 잘 설명할 수 있는
특징을 추출(Feature Extraction)
가능

즉, 고차원의 원본 데이터를
저차원의 잠재 공간
(Latent Space)로 차원 축소

1

Introduction to the Generative Model

생성모델과 딥러닝

Manifold (다양체) : **신러닝 생성 모델**

딥러닝 생성 모델

전체적인 형체가 평평하지 않지만 일부분을 평평한 공간에 그릴 수 있는 형체



고차원의 원본 데이터를 저차원의 잠재 공간(Latent Space)로 차원 축소

Introduction to the Generative Model

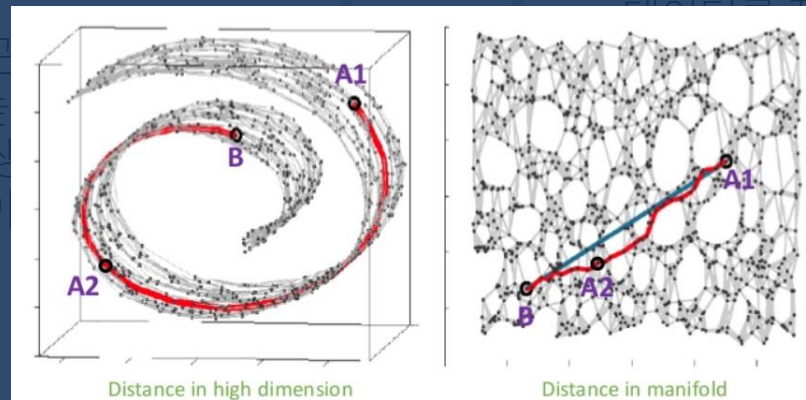
생성모델과 딥러닝

Manifold 가정

신러닝 생성 모델

고차원의 데이터는 낮은 밀도를 갖지만, 이들의 집합을 포함하는 저차원의 manifold가 있음.

즉, 저차원 잠재 공간으로의 축소는 고차원 데이터의 manifold를 찾는 것



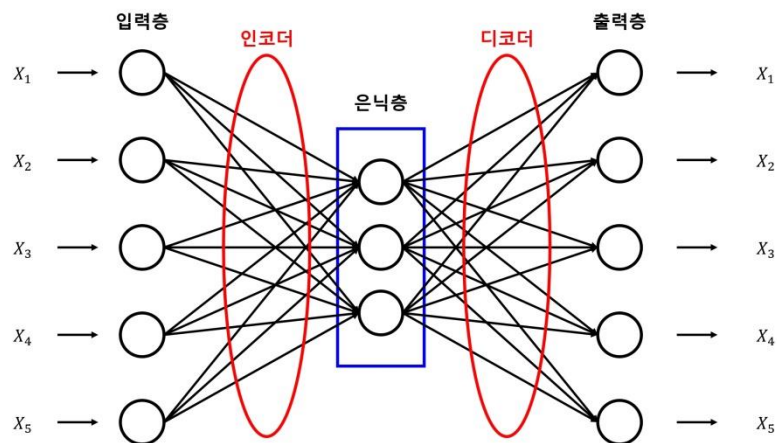
고차원의 원본 데이터를 저차원의 잠재 공간(Latent Space)로 차원 축소

2

Auto Encoder

Auto Encoder

- ✓ 특징 추출을 목적으로 주어진 데이터를 변환하는 비지도 학습 딥러닝 알고리즘
- ✓ 비선형 차원 축소 방법
- ✓ 이미지와 같이 쉽게 특징을 찾기 어려운 데이터를 변환하여 기존의 데이터에서 찾기 어려웠던 단서를 찾기 위해 사용



1 입력층과 출력층이 같은 신경망

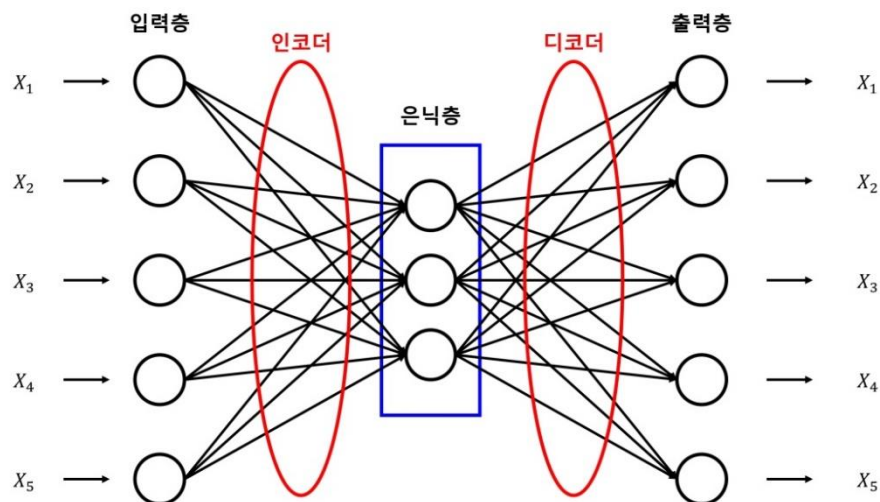
2 은닉층의 유닛 수가 입력층보다 적음

- ✓ 입력 데이터를 압축하는 효과
- ✓ 노이즈 제거에 효과적

2

Auto Encoder

인코더 & 디코더



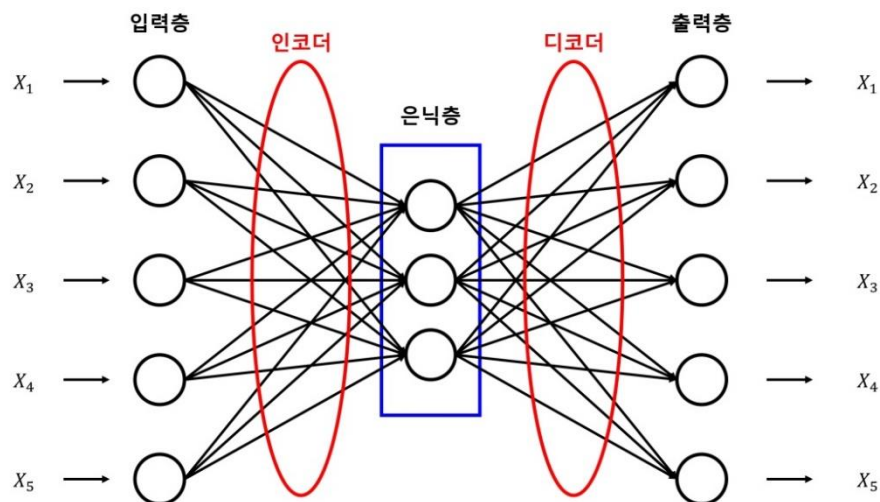
인코더

- ✓ 고차원 입력 데이터를 저차원 표현 벡터로 압축
- ✓ 데이터의 특징만 은닉층에서 학습
- ✓ 나머지 정보는 손실시킴
- ✓ 인코더와 디코더는 대칭적 구조

2

Auto Encoder

인코더 & 디코더



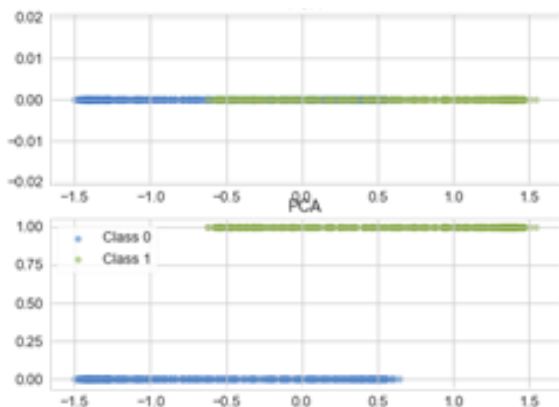
디코더

- ✓ 저차원 표현 벡터를 원본 차원으로 압축해제
- ✓ 출력값이 입력값과 최대한 같아지도록 가중치를 튜닝
- ✓ 특징을 잘 추출할 수 있게 하기 위해

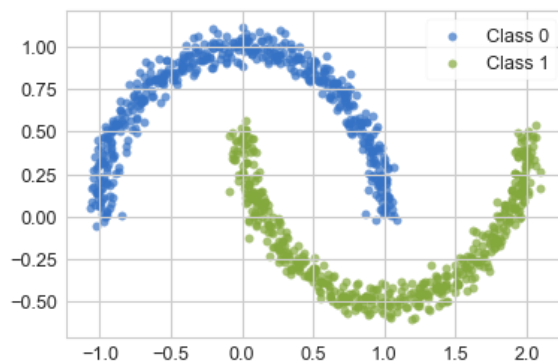
2

Auto Encoder

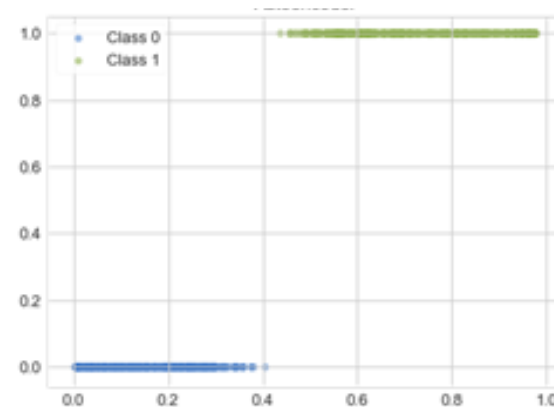
PCA vs AE



PCA



비선형 데이터



AE

선형 차원 축소 방법인 PCA는 차원 축소 시 클래스를 제대로 구분 X

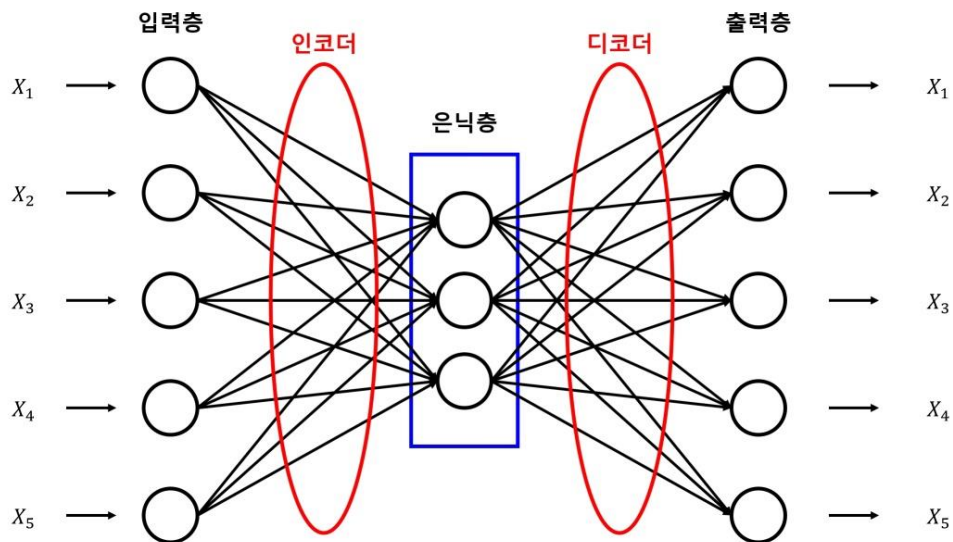
AE는 비선형 차원 축소 방법이라 잘 구분

AE가 더 좋다는 것이 아니라 쓰임새가 다르다는 것!!!

2

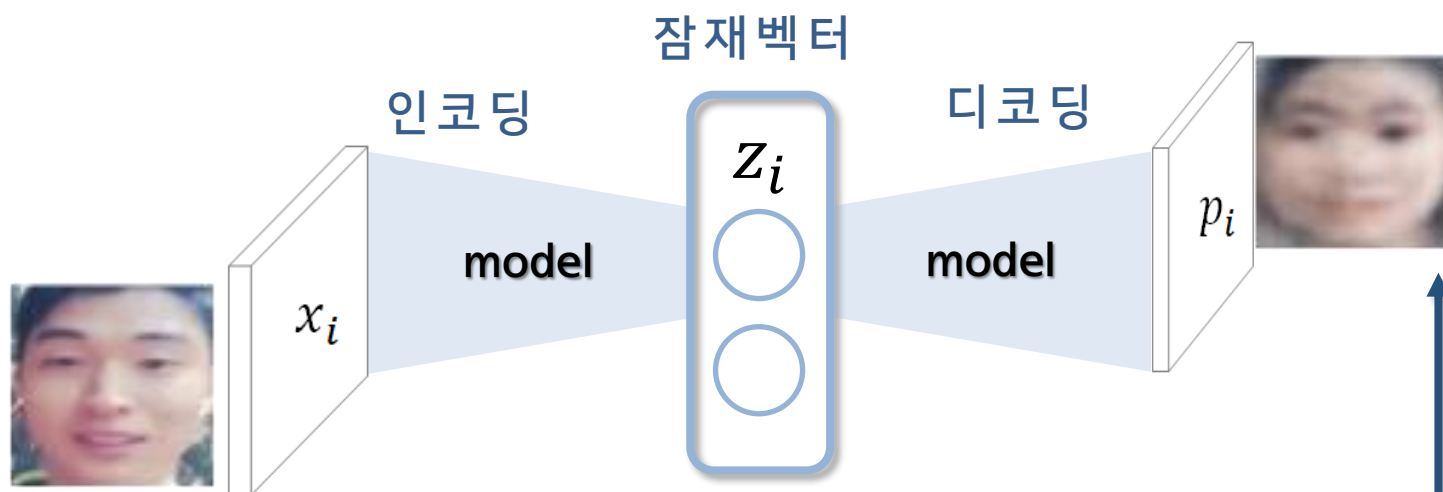
Auto Encoder

AE의 학습과정



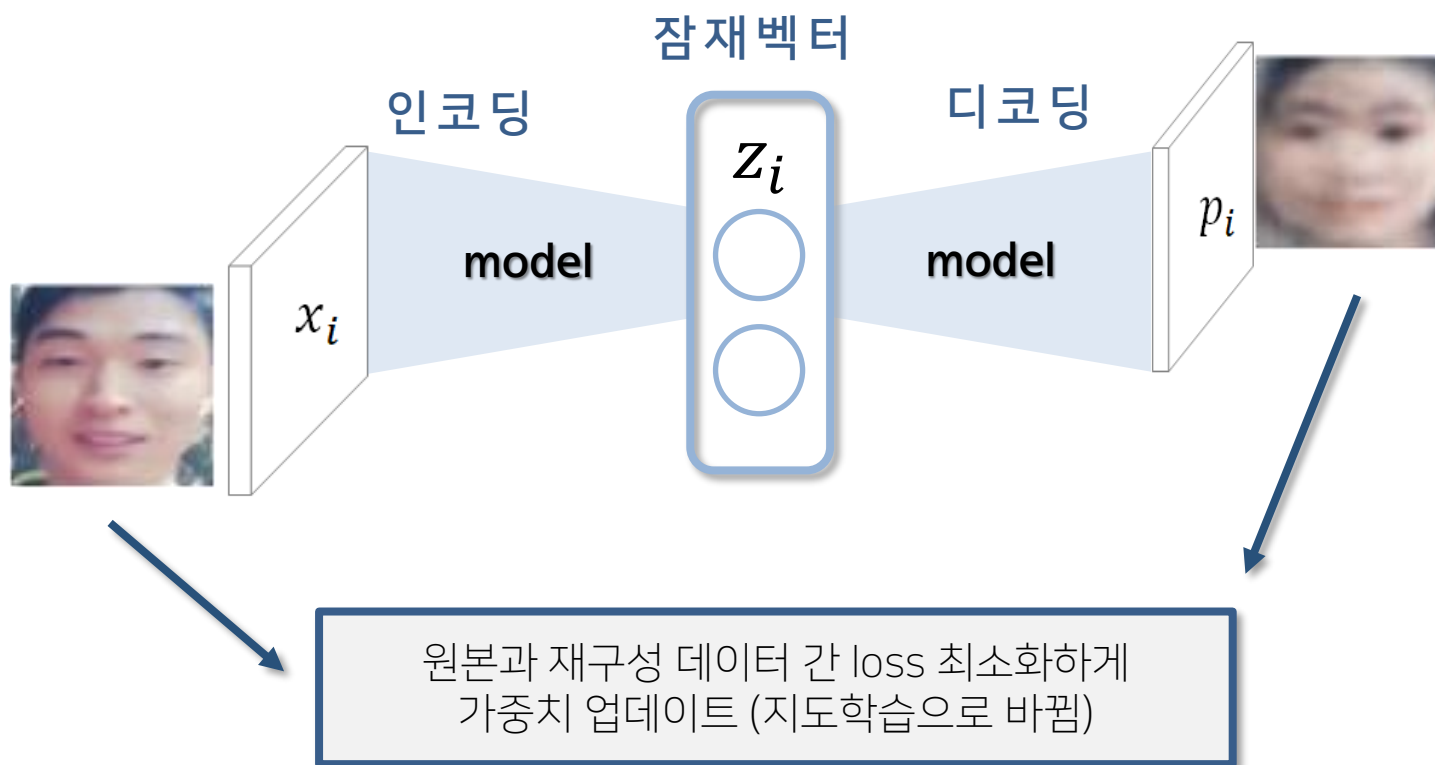
- 1 인코더가 입력층과 은닉층의 가중치를 계산해 활성화 함수 통과
- 2 디코더가 은닉층과 출력층의 가중치를 계산해 Sigmoid 함수 통과
- 3 출력층의 Output을 이용해 원본 데이터와의 loss를 계산 후 SGD로 최적화 및 역전파로 가중치 갱신

생성모델로서의 AE

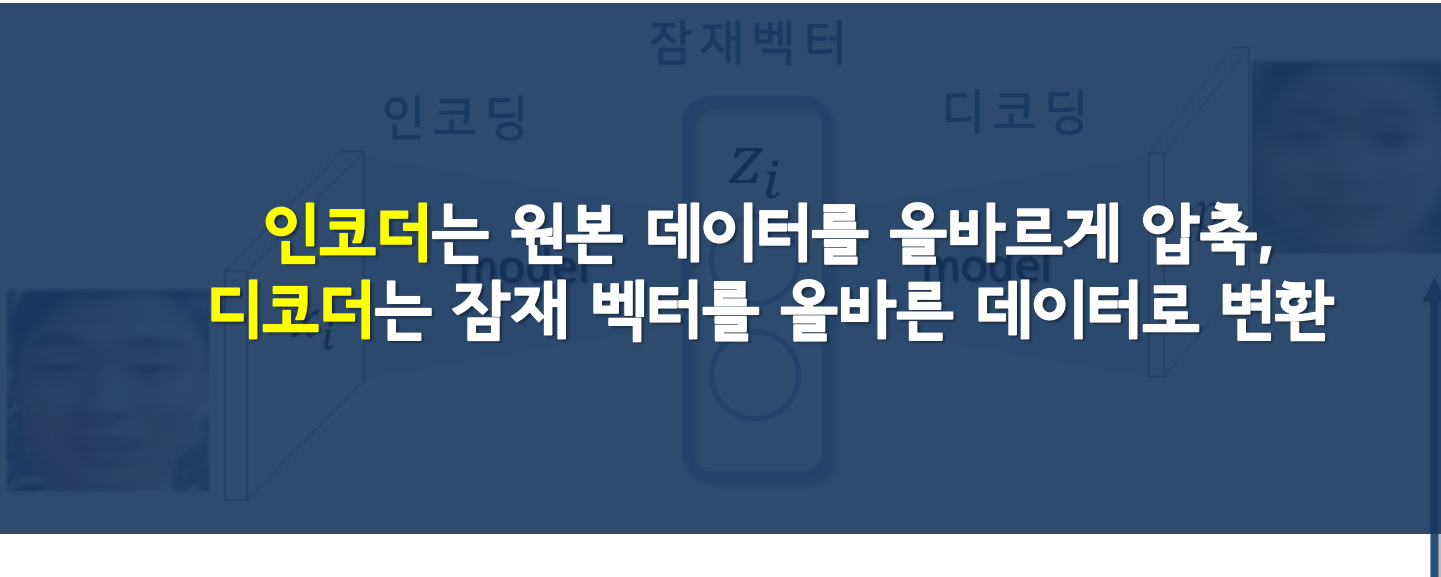


인코딩과 디코딩을 거쳐 재구성 이미지가 됨

생성모델로서의 AE



생성모델로서의 AE



인코더는 원본 데이터를 올바르게 압축,
디코더는 잠재 벡터를 올바른 데이터로 변환

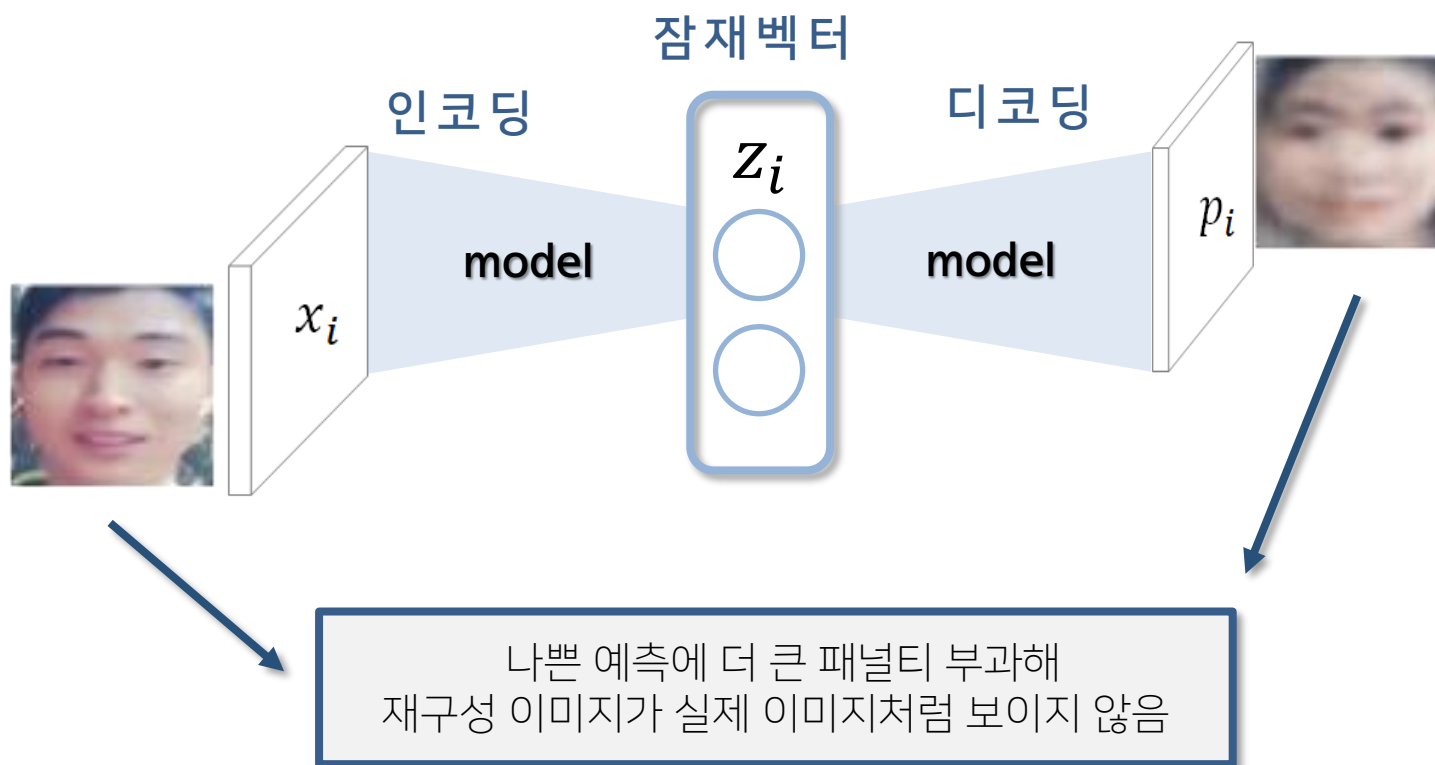
인코딩과 디코딩을 거쳐 재구성 이미지가 됨

생성모델로서의 AE



재구성 이미지 예시 (저번 주분 modeling결과)

Binary Cross Entropy 대신 RMSE 사용하는 이유

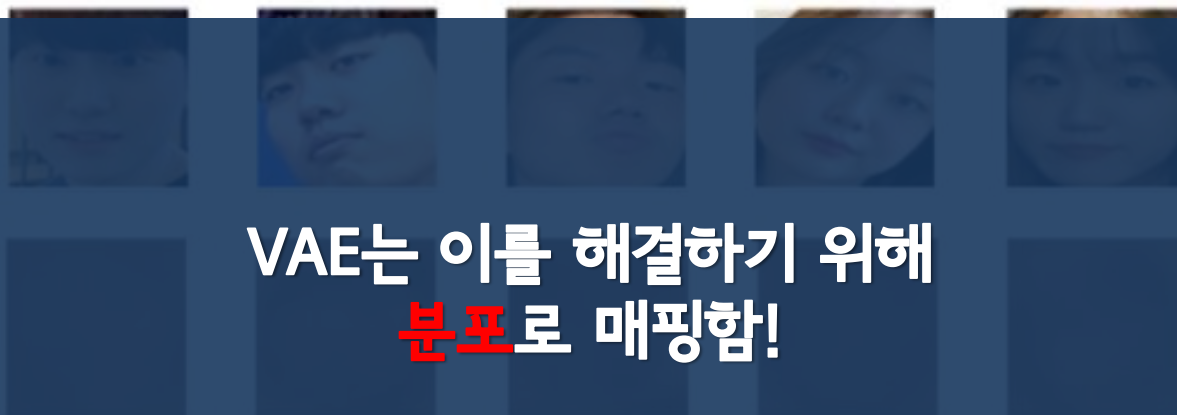


AE의 한계



AE는 한 **포인트**에 매핑 되기에
고차원데이터(이미지)는 뚜렷한 생성이미지를 만들기 힘들

AE의 한계

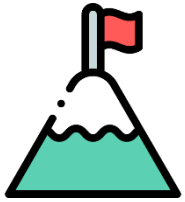


VAE는 이를 해결하기 위해
분포로 매핑함!



AE는 한 **포인트**에 매핑 되기에
고차원데이터(이미지)는 뚜렷한 생성이미지를 만들기 힘들

VAE (Variational Auto Encoder)



인코더는 $P(\mathbf{z}|\mathbf{x})$, 즉 원래의 데이터로 잘 복원할 수 있는 \mathbf{z} 에 대한 이상적인 사후 확률분포를 찾는 게 목표!



이상적인 $P(\mathbf{z}|\mathbf{x})$ 를 모르기 때문에 다루기 쉬운 분포인
정규분포 $q(\cdot)$ 를 가정하고,
이의 파라미터를 바꿔가며 이상적인 $P(\mathbf{z}|\mathbf{x})$ 에 가깝게 함!
(a.k.a 변분추론)

VAE (Variational Auto Encoder)



인코더는 $P(\mathbf{z}|\mathbf{x})$, 즉 원래의 데이터로 잘 복원할 수 있는 \mathbf{z} 에 대한 이상적인 사후 확률분포를 찾는 게 목표!

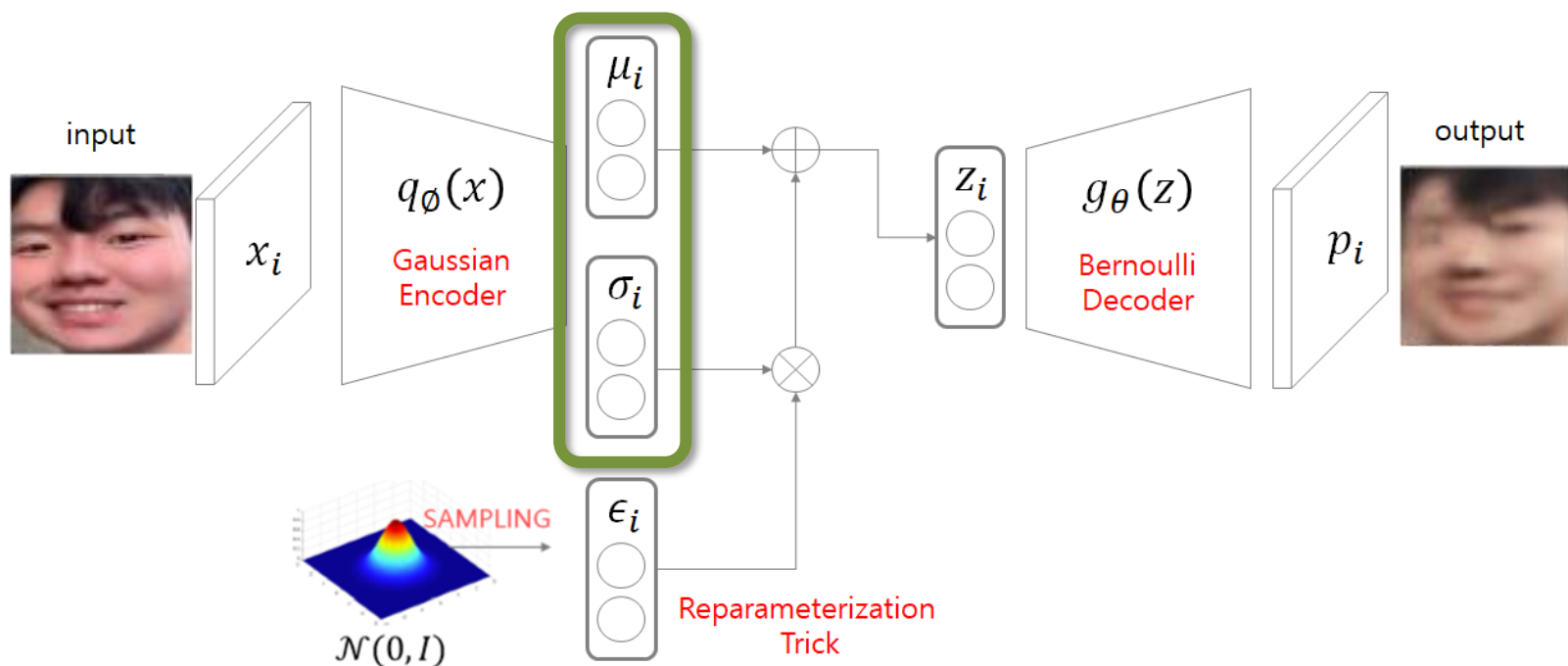
인코더는 $q(\mathbf{z}|\mathbf{x})$ 의 모수인 평균과 분산을 출력함

이상적인 $P(\mathbf{z}|\mathbf{x})$ 를 모르기 때문에 다루기 쉬운 분포인
정규분포 $q(\cdot)$ 를 가정하고,
이의 파라미터를 바꿔가며 이상적인 $P(\mathbf{z}|\mathbf{x})$ 에 가깝게 함!
(a.k.a 변분추론)

2

Auto Encoder

VAE (Variational Auto Encoder)



VAE (Variational Auto Encoder)의 Loss Function

RMSE

*KL Divergence*

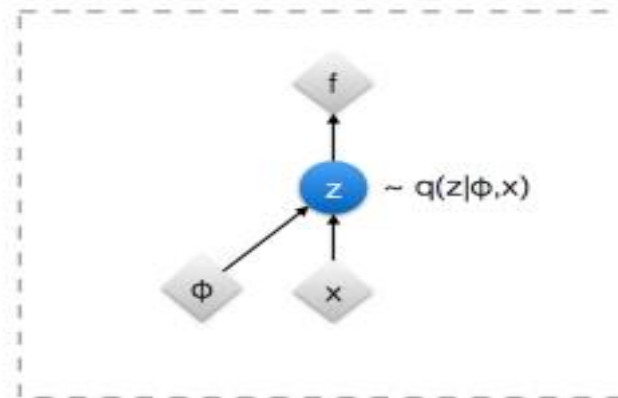
: 확률분포 q 와 p 사이의
정보량 차이를 줄이는 게
목표!



두 값을 동시에 최소화하는 **가중합**을 사용

(참고) 목적함수 : $\mathbb{E}_{\mathcal{D}}[\mathcal{L}(\theta, \phi; x^{(i)})] = \mathbb{E}_{\mathcal{D}} \left[-\mathcal{D}_{KL}(q_{\phi}(z|x^{(i)})||p_{\theta}(z)) + \log p_{\theta}(x^{(i)}|z^{(i)}) \right]$

VAE(Variational Auto Encoder)의 문제점

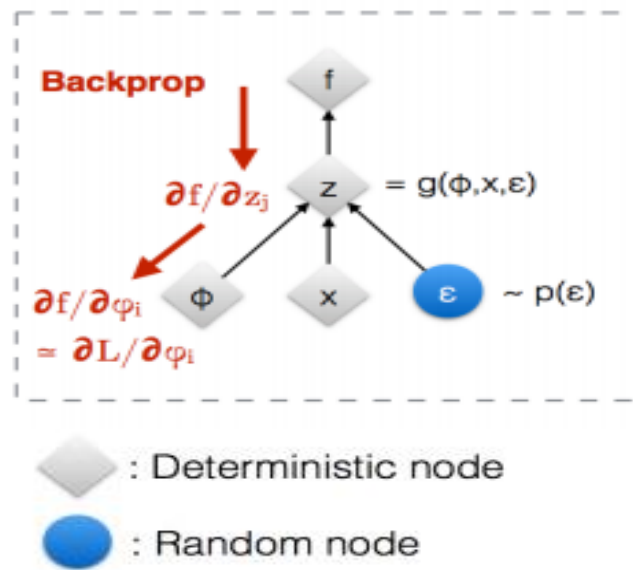


◇ : Deterministic node
● : Random node



$q(z|x)$ 에서 바로 sampling하면 미분 불가능해져
역전파 불가 (q 의 최적화 불가)

VAE(Variational Auto Encoder)의 해결방안



~ $N(0,1)$ 에서 샘플링해 미분 가능하게 함!
즉, Z 를 직접 샘플링 하는 게 아니라, 노이즈를 샘플링 하는 방식

VAE(Variational Auto Encoder)의 해결방안

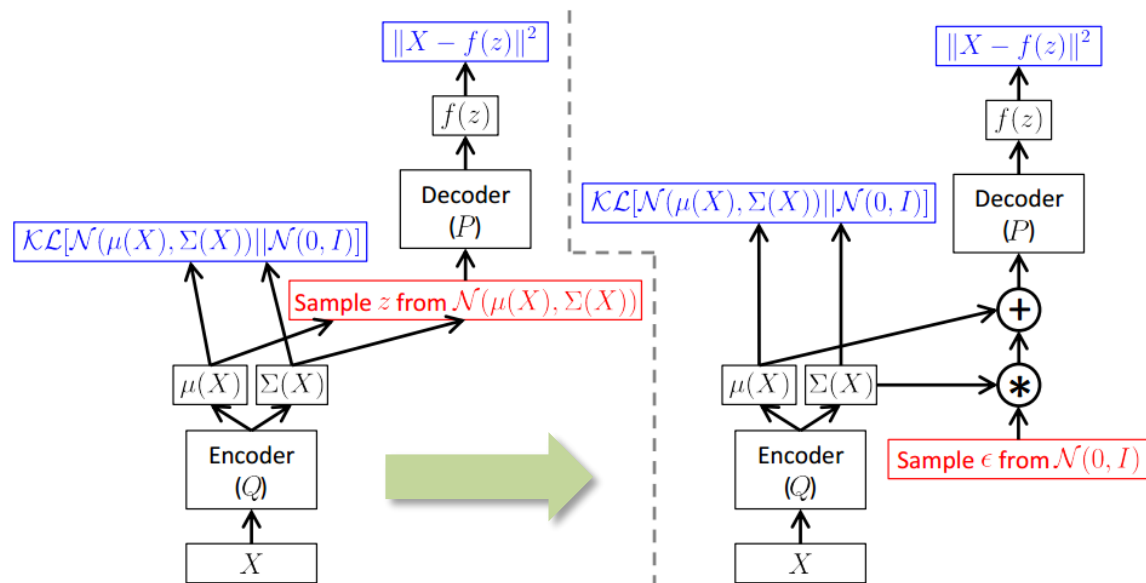
쉽게 말해,
 $q(z|x)$ 에서 샘플링하면 학습마다 다른 분포에서 샘플링 되면서
 최적의 값을 찾는 데에 문제
 → 파라미터가 일정한 ϵ 에서 샘플링하면서 해결!

● : Random node



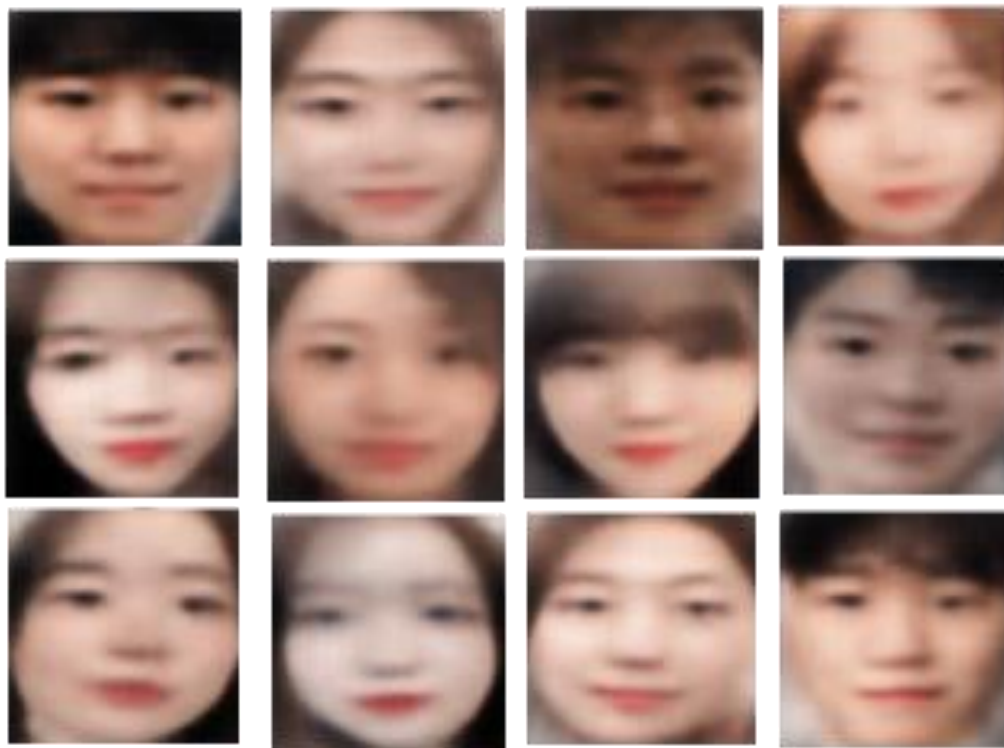
즉, Z 를 직접 샘플링 하는 게 아니라, 노이즈를 샘플링 하는 방식

VAE(Variational Auto Encoder)의 해결방안

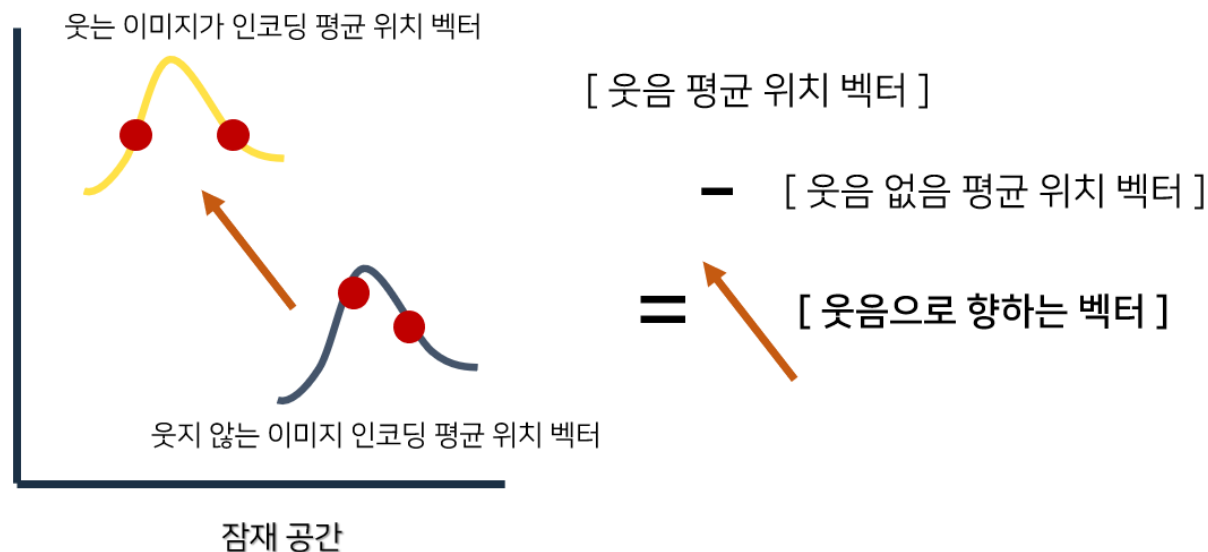


~ $\mathcal{N}(0,1)$ 에서 샘플링해 미분 가능하게 함!
 즉, z 를 직접 샘플링 하는 게 아니라, 노이즈를 샘플링 하는 방식

VAE(Variational Auto Encoder)의 생성모델 예시



VAE(Variational Auto Encoder)의 벡터 연산 (저번 주 주분 복습!)



레이블링 된 데이터를 통해 벡터연산으로 특징을 바꿀 수 있음

2

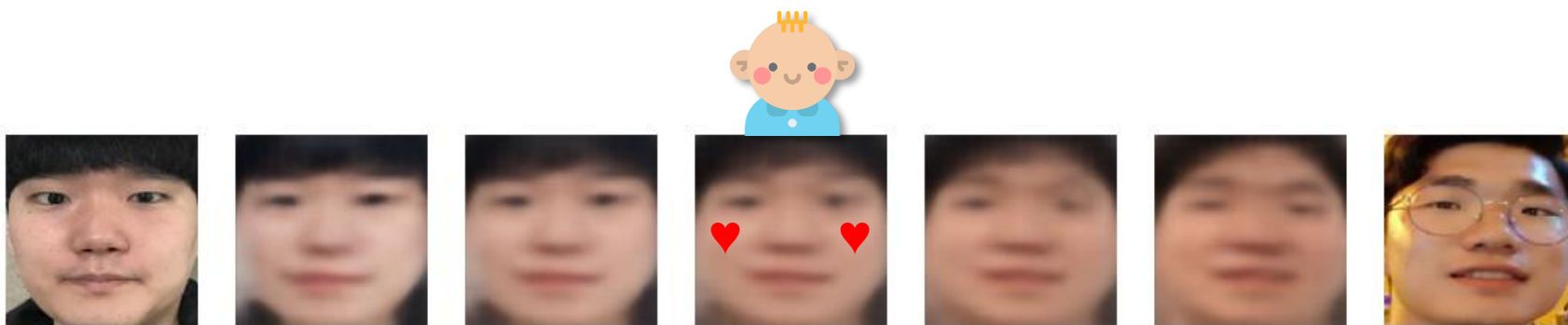
Auto Encoder

VAE(Variational Auto Encoder)의 벡터 연산 예시_남/여 바꾸기



기억나시죠? ㅎㅎ 기억나야합니다..

VAE(Variational Auto Encoder)의 벡터 연산 예시_얼굴합성



아쉬우니 다시보는 원구 ♥ 성민의 2세
24기 가디마ㅏㅏ

-딥러닝을 세상에서 제일 좋아하는 25기 박x영의 진심이 담긴 슬라이드-

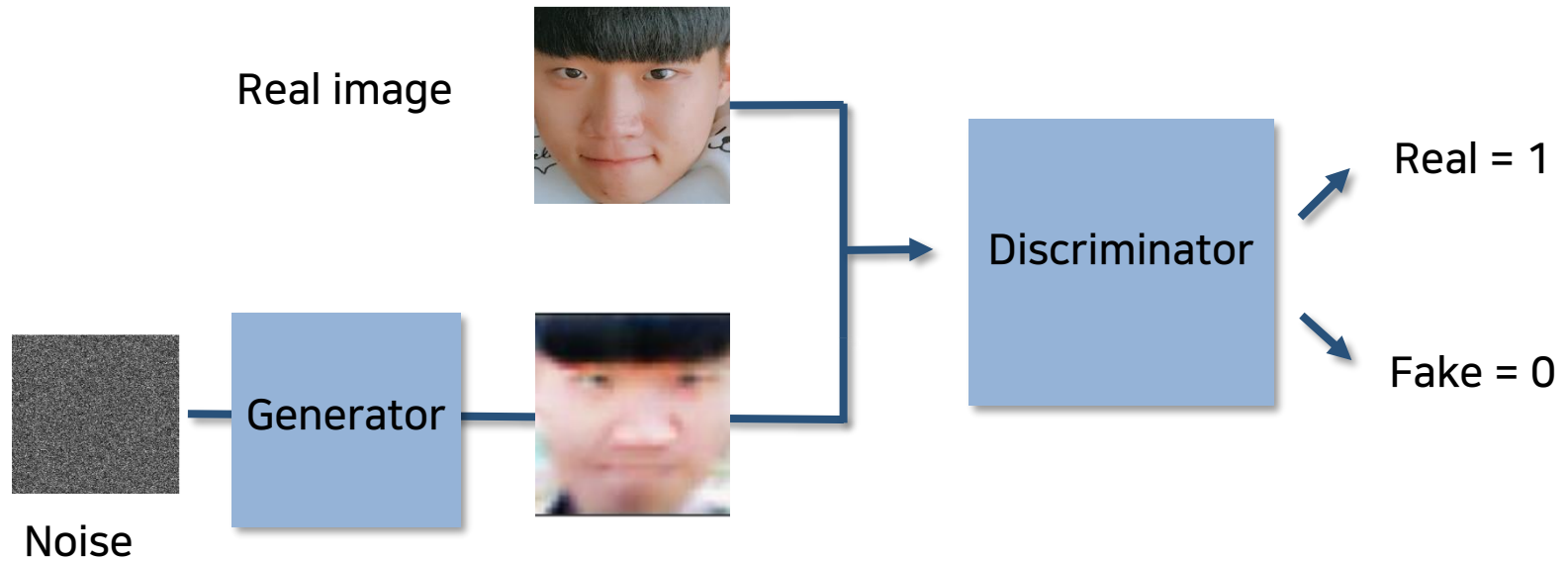
3

Generative Adversarial Network

3

Generative Adversarial Network

GAN



Generator와 Discriminator의 경쟁으로
진짜 같은 가짜 사진을 생성해내는 모델

3

Generative Adversarial Network

GAN

Discriminator

실제를 1로 분류할 기대값 + 가짜를 0으로 판별할 기대값 **MAX**

Generator

가짜를 0으로 판별할 기대값 **MIN**

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

D: 판별자 G: 생성자

실제를 1로 분류할 기대값

가짜를 0로 분류할 기대값

판별자와 생성자는 **MINMAX game**을 통해
손실함수를 최적화하면서 균형점을 찾음

3

Generative Adversarial Network

GAN

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

동시에 최적화 불가능

생성자 고정된 상태에서 판별자 학습

$$\begin{aligned} \max_D V(D) &= E_{x \sim p_{\text{data}}(x)} [\log D(x)] + E_{z \sim p_z(z)} [\log \{1 - D(G(z))\}] \\ &= \frac{1}{m} \sum_{i=1}^m \log D(x^i) + \frac{1}{m} \sum_{i=1}^m \log \{1 - D(G(z^i))\} \end{aligned}$$

판별자 고정된 상태에서 생성자 학습

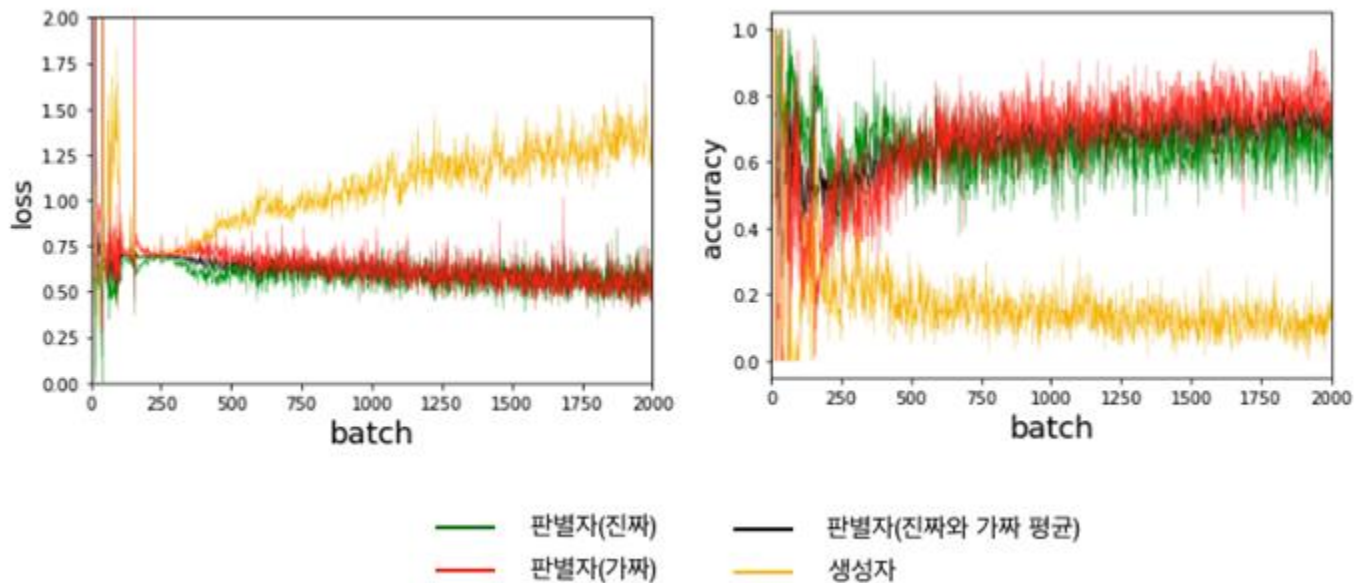
$$\begin{aligned} \min_G V(G) &= E_{z \sim p_z(z)} [\log \{1 - D(G(z))\}] \\ &\Rightarrow -E_{z \sim p_z(z)} [\log D(G(z))] \end{aligned}$$

손실함수를 최적화하면서 판별기를 고정

3

Generative Adversarial Network

GAN

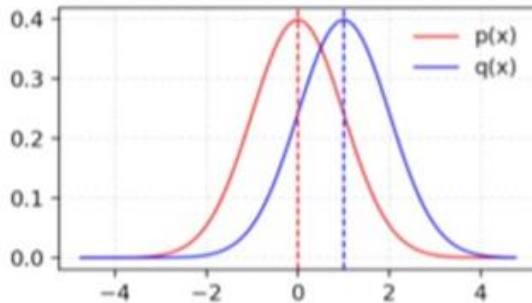


생성자의 로스는 가짜 데이터를 진짜로 판별하도록 하기에 **증가**

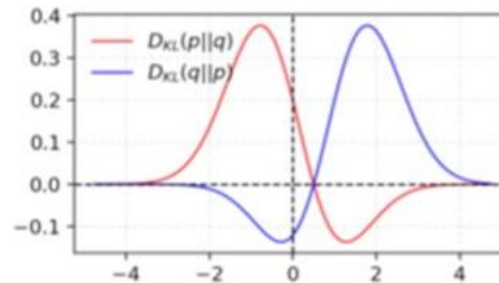
판별자의 로스는 진짜와 가짜를 잘 구별하여 적당히 감소

학습이 진행되면 **평형**을 이뤄야 함, 즉 두 네트워크 간 **균형** 중요

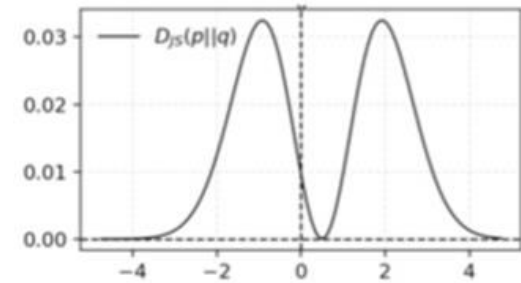
GAN



확률분포



KLD



JSD

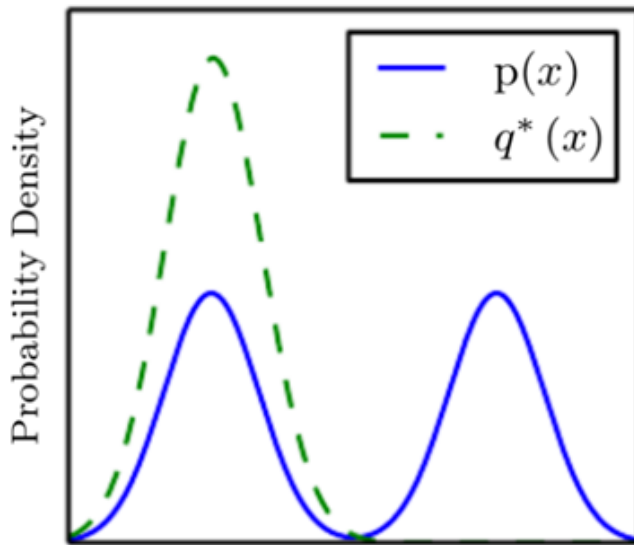
KLD와 JSD 모두 **두 확률분포의 차이**를 나타내는 함수 (자세한 건 부록)

KLD는 비대칭한 특징으로 거리를 나타내지 못함

JSD는 두 확률분포의 **거리**를 나타낼 수 있음

GAN의 목적 함수를 JSD로 나타낼 수 있음!! (부록)

GAN 의 문제점 : Mode Collapse



생성자



품 = 판별자가 이거 구별 못한다

판별자가 잘 구별하지 못하는 mode를 찾으면

그 분포로만 데이터를 생성하게 되는 **Mode Collapse** 문제 발생

Mode Collapse 해결 방안

Feature
matching

가짜와 실제 사이
LSE 고려

Mini-batch
Discrimination

배치 별로 가짜와
실제의 거리 합의 차이 고려

Historical
averaging

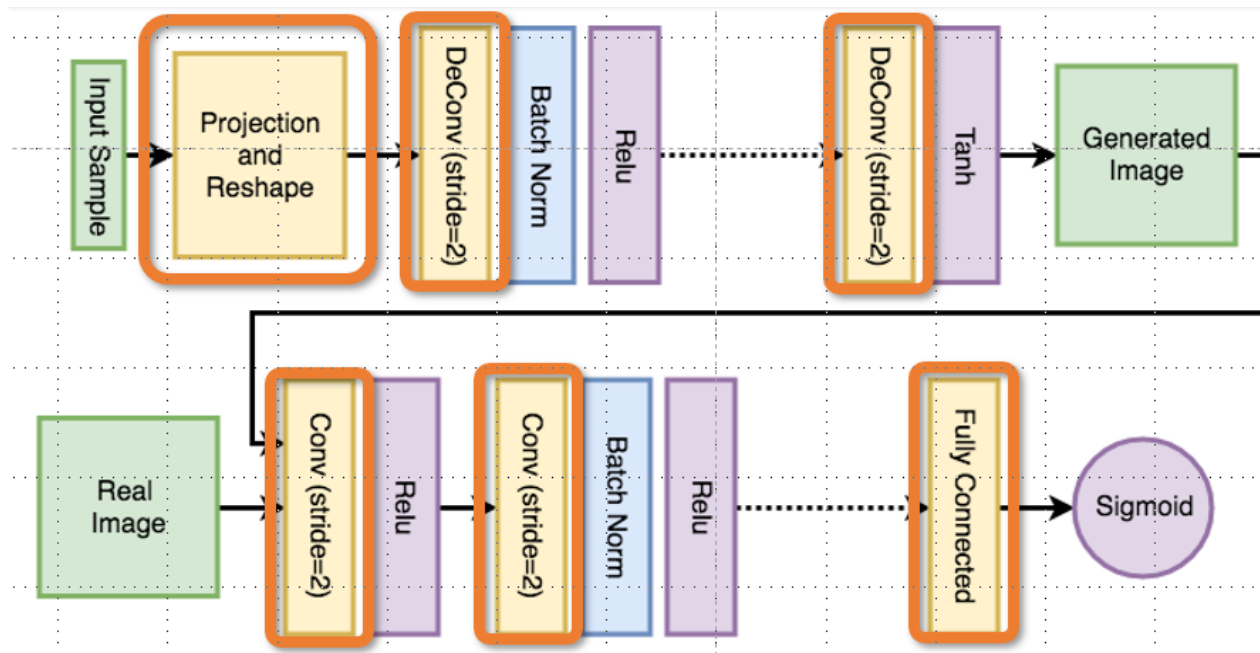
배치 단위로
파라미터 업데이트

전체 데이터 분포를 골고루 학습하고 기억할 수 있게 하여 해결

3

Generative Adversarial Network

Deep Convolutional GAN



기본적인 GAN 구조에 학습 불안정 요소인 FC layer 대신

*Convolutional layer*를 사용하는 모델

Deep Convolutional GAN



안정적인 학습을 위한 DCGAN 가이드

판별자는 convolution 층, 생성자는 fractional convolution 층 사용

판별자와 생성자 모두 배치정규화 적용

FC hidden layer 제거

생성자의 활성화함수는 출력층을 제외하고 ReLU 사용, 출력층은 Tanh

판별자의 활성화함수로 LeakyReLU 사용

Deep Convolutional GAN

판별자는 convolution 층, 생성자는 fractional convolution 층 사용

판별자와 생성자 모두 배치정규화 적용

FC hidden layer 제거
생성자는 왜 Fractional convolution layer를 사용할까?

생성자의 활성화함수는 출력층을 제외하고 ReLU 사용, 출력층은 Tanh

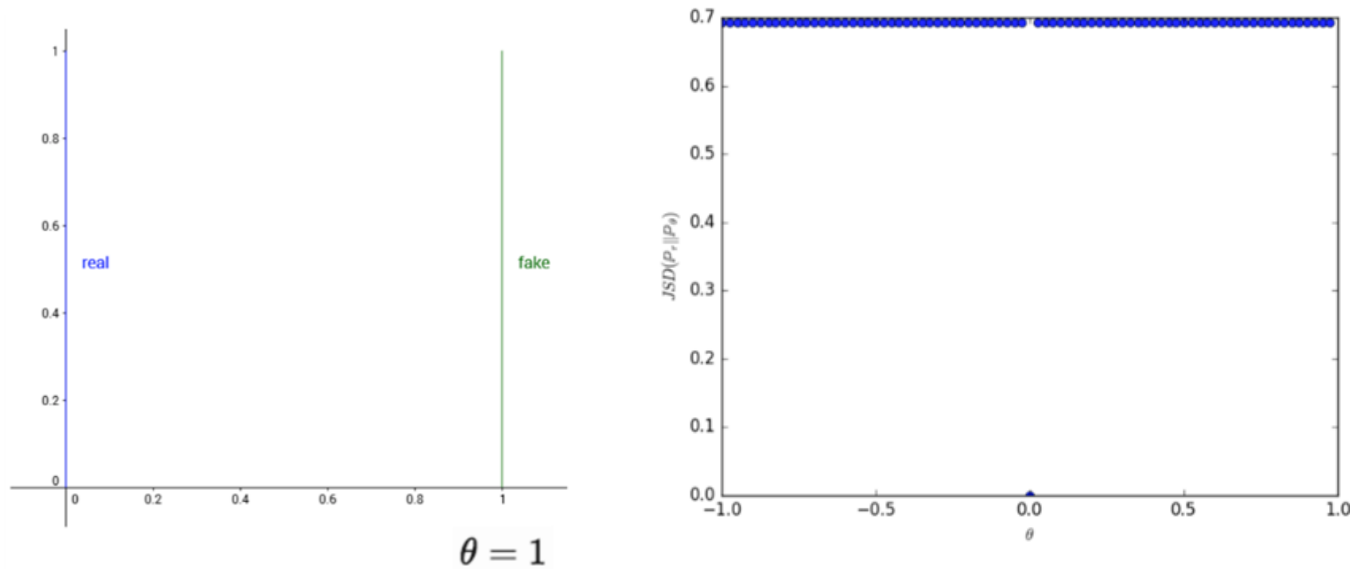
Stride를 1보다 작게 설정하여 feature map의 크기를 키워

이미지 생성을 가능하게 함

3

Generative Adversarial Network

Wasserstein GAN



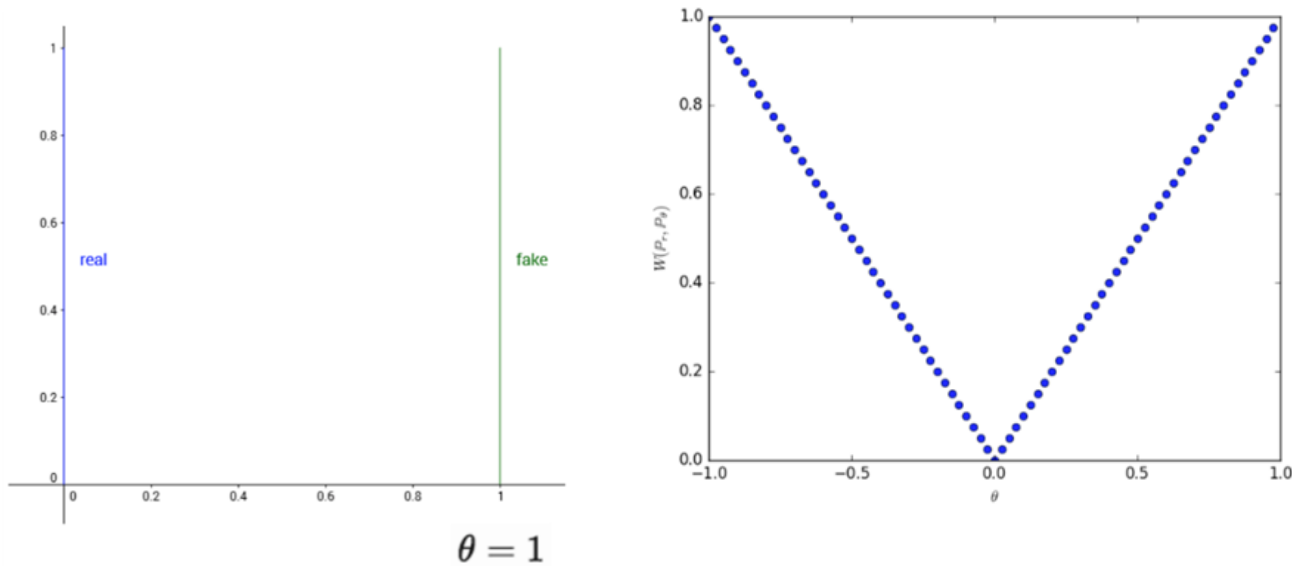
GAN에서 JSD의 최소화를 목적함수로 사용

하지만 JSD가 확률분포의 **거리를 잘 측정하지 못하고 미분이 불가능**한 경우 발생

3

Generative Adversarial Network

Wasserstein GAN



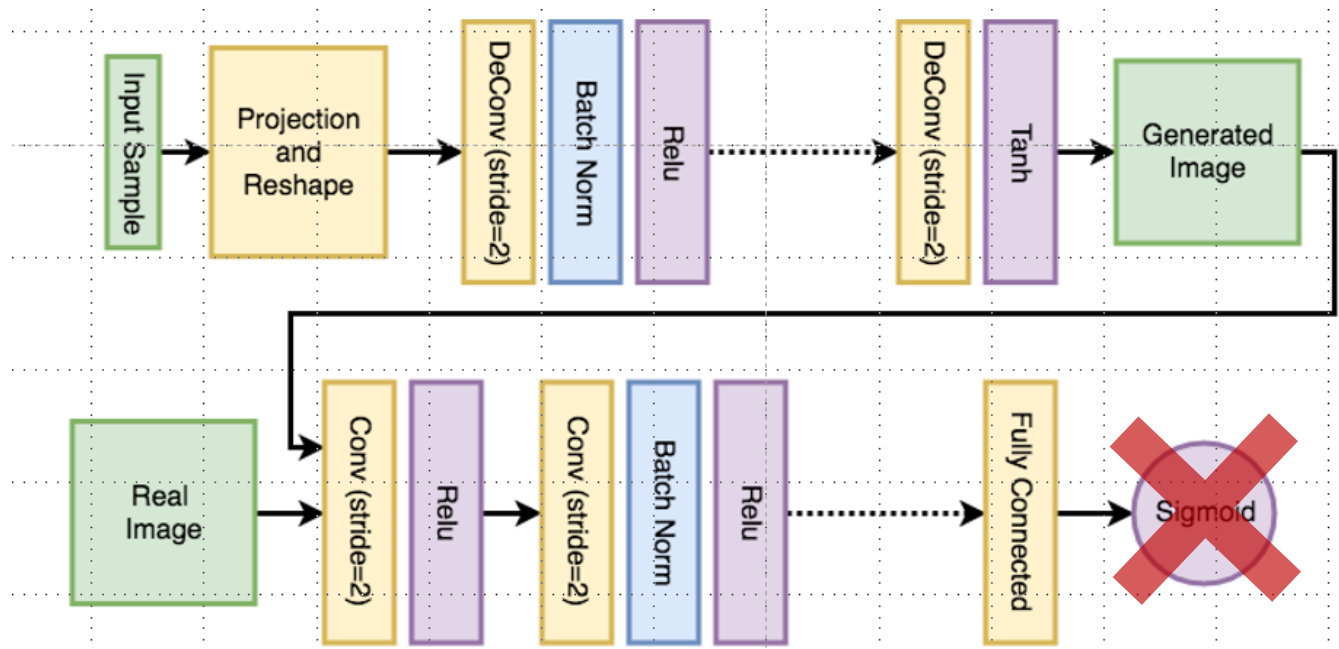
WGAN은 JSD 대신 Wasserstein Distance 사용

WD = 하나의 분포를 다른 분포로 옮길 때 필요한 양과 거리의 곱

3

Generative Adversarial Network

Wasserstein GAN



판별자의 출력층 활성화 함수 Sigmoid를 제거하여
진위 예측을 $[0,1]$ 이 아닌 실수 전체 범위로 넓힘
판별자 대신 비평자!

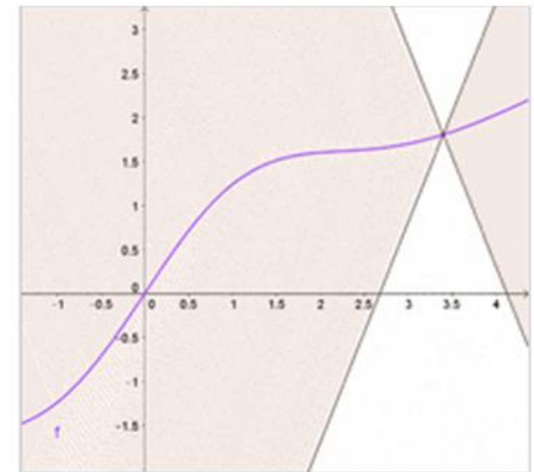
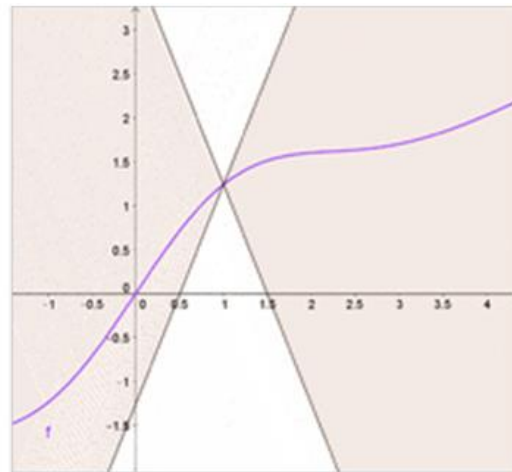
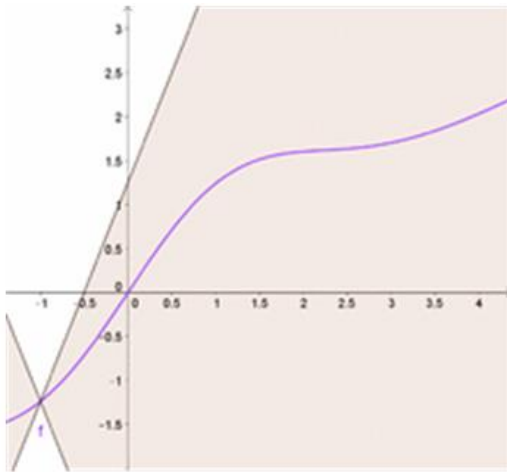
Wasserstein GAN

시그모이드를 제거하여 손실의 출력 범위가 실수 전체가 되면
모델이 발산하는 **Gradient exploding 문제**가 생기지 않을까?

그렇기에 **비평자는 1-립시츠 연속함수**여야 해!

판별자의 출력층 활성화 함수 Sigmoid를 제거하여
진위 예측을 $[0,1]$ 이 아닌 실수 전체 범위로 넓힘
판별자 대신 비평자!

1-립시츠 함수



** 립시츠 함수는 위 그림에서 색칠된 영역에만 존재

임의의 두 지점의 **기울기가 어떤 상수 이상 증가하지 않는** 함수
 상수가 1일때의 립시츠 함수가 1-립시츠

WGAN 어떻게 비평자를 립시츠 함수로 만들까?

가중치 클리핑



비평자의 모든 파라미터의 가중치를
임의의 작은 상수 범위로 고정



고정 범위가 크면 학습이
불안정 or 수렴 속도가 느려지는 문제

고정 범위가 너무 작으면
Gradient Vanishing 발생

3

Generative Adversarial Network

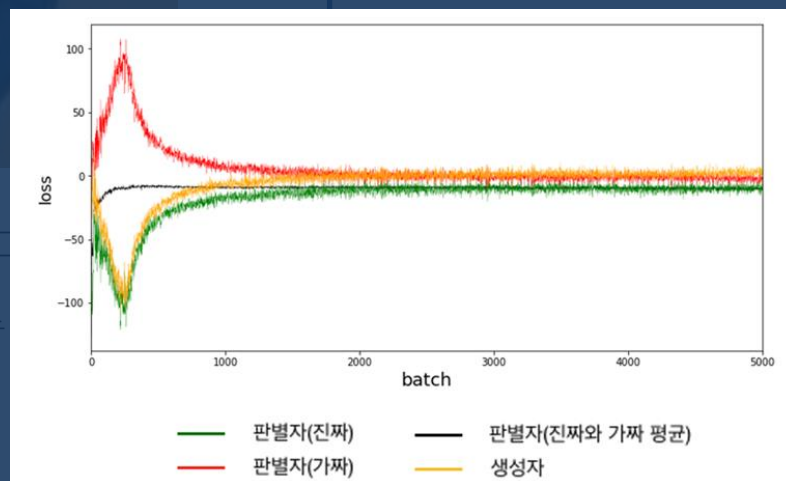
WGAN 어떻게 비평자를 립시츠 함수로 만들까?

이를 보완한 모델이 WGAN-GP

비평자의 손실함수에 **Gradient penalty** 추가

Gradient의 노름이 1보다 큰 정도에 대해 L2 페널티 부과

가중치 클리핑보다 자연스럽게 안정적인 훈련 가능



THANK YOU



We ❤️ Deep Learning

KL Divergence와 JS Divergence

$$H(p, q) = - \sum_i p_i \log q_i$$

$$= - \sum_i p_i \log q_i \quad \overset{=0}{\underbrace{- \sum_i p_i \log p_i + \sum_i p_i \log p_i}}_{= H(p)}$$

$$= H(p) + \sum_i p_i \log p_i - \sum_i p_i \log q_i$$

$$= H(p) + \sum_i p_i \log \frac{p_i}{q_i}$$

p의 엔트로피에 **이만큼** 더해진 것이
cross entropy 가 됩니다.

이만큼 더해지는 것이 무엇일까요?

⇒ 바로 분포 p와 분포 q의 **정보량 차이** 입니다

⇒ 이것이 바로 KL-divergence 입니다

$$JSD(p \| q) = \frac{1}{2} KL(p \| M) + \frac{1}{2} KL(q \| M)$$

$$\text{where, } M = \frac{1}{2}(p + q)$$

Discriminator의 목적 함수와 JS Divergence

$$\begin{aligned}
 \min_G V(D^*, G) &= E_{x \sim p_{data}(x)} [\log D^*(x)] + E_{x \sim p_g(x)} [\log \{1 - D^*(x)\}] \\
 &= E_{x \sim p_{data}(x)} \left[\log \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \right] + E_{x \sim p_g(x)} \left[\log \frac{p_g(x)}{p_{data}(x) + p_g(x)} \right] \\
 &= \int_x p_{data}(x) \log \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} dx + \int_x p_g(x) \log \frac{p_g(x)}{p_{data}(x) + p_g(x)} dx \\
 &= -\log 4 + \int_x p_{data}(x) \log \frac{2 \cdot p_{data}(x)}{p_{data}(x) + p_g(x)} dx + \int_x p_g(x) \log \frac{2 \cdot p_g(x)}{p_{data}(x) + p_g(x)} dx \\
 &= -\log 4 + \int_x p_{data}(x) \log \frac{p_{data}(x)}{\frac{p_{data}(x) + p_g(x)}{2}} dx + \int_x p_g(x) \log \frac{p_g(x)}{\frac{p_{data}(x) + p_g(x)}{2}} dx \\
 &= -\log 4 + KLD \left(p_{data}(x) \parallel \frac{p_{data}(x) + p_g(x)}{2} \right) + KLD \left(p_g(x) \parallel \frac{p_{data}(x) + p_g(x)}{2} \right) \\
 &= -\log 4 + 2 \cdot JSD(p_{data}(x) \parallel p_g(x))
 \end{aligned}$$