

# 생성 모델과 시각 지능

Generative Model and Visual Intelligence

04 주차 |

GAN 기초

상명대학교 컴퓨터과학과  
민 경 하

# 학습목차

1. GAN의 기본 원리
2. GAN의 구성 요소 (1): generator
3. GAN의 구성 요소 (2): discriminator
4. GAN의 구성 요소 (3): loss 함수
5. GAN의 구성 요소 (4): 훈련



# 1. GAN의 기본 원리

나의 행운은 적의 불행이요,  
나의 불행은 적의 행운이다.



## 1. GAN의 기본 원리

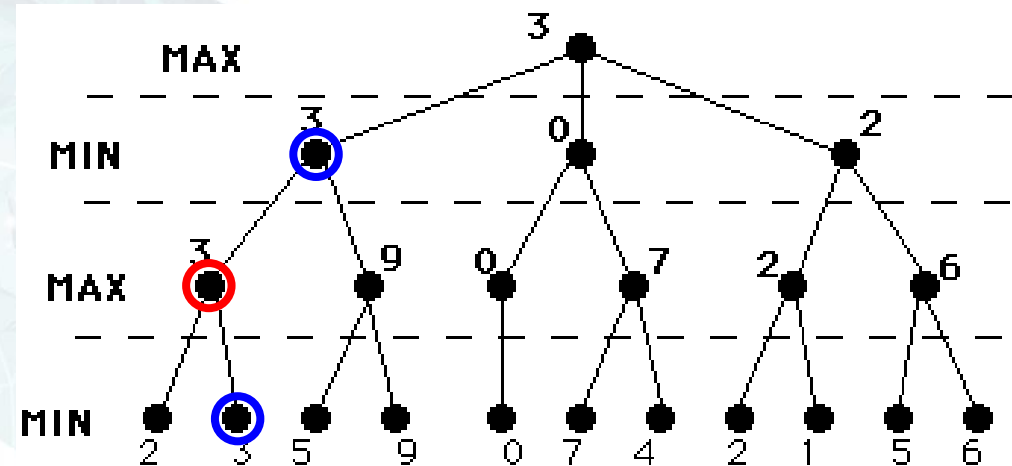
# 배경 1: Adversarial search (적대적 탐색)에 대한 오랜 연구가 존재함

- 게임 이론의 알고리즘으로 서로 반대되는 입장의 두 player가 서로 입장을 바꾸어 가면서 play함으로써 최적의 해를 찾아가는 과정
  - ☑ 예) mini-max search for tic-tac-toe

자식 중에 최대값을 선택 → 3

자식 중에 최소값을 선택 → 3

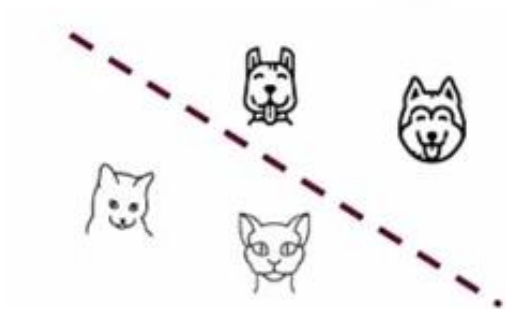
자식 중에 최대값을 선택 → 3



# 배경 2: 심층 학습의 Discriminative model이 중점적으로 발전해 옴

## ◉ Discriminative model (Classifier)

- ✓ 입력 데이터를 미리 정의한 카테고리로 분류하는 도구
- ✓ SVM → CNN
- ✓ 입력: 복잡한 모델 (e.g. 영상)
- ✓ 출력 (predicted probability): 하나의 scalar 값 or n 개의 scalar 값



Features      Class

$$X \rightarrow Y$$
$$P(Y|X)$$

## 1. GAN의 기본 원리

# 배경 3: 심층 학습에서 Generative model에 대한 개념이 떠오름

## Generative model

- ✓ 간단한 정보 (latent vector)로부터 복잡한 모델을 합성하는 도구
- ✓ AE (auto encoder) → VAE (variational auto encoder)
- ✓ 입력: 간단한 정보 (latent vector)
- ✓ 출력: (비교적 정교하고 사실적인) 영상





# 1. GAN의 기본 원리

## Generative Adversarial Network의 시작



### Generative Adversarial Nets

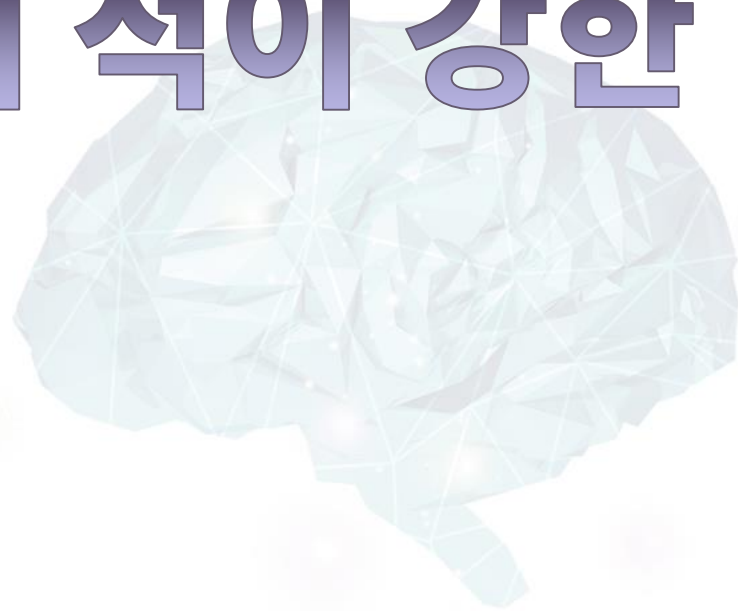
Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio<sup>†</sup>  
Département d'informatique et de recherche opérationnelle  
Université de Montréal  
Montréal, QC H3C 3J7

#### Abstract

We propose a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model  $G$  that captures the data distribution, and a discriminative model  $D$  that estimates the probability that a sample came from the training data rather than  $G$ . The training procedure for  $G$  is to maximize the probability of  $D$  making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions  $G$  and  $D$ , a unique solution exists, with  $G$  recovering the training data distribution and  $D$  equal to  $\frac{1}{2}$  everywhere. In the case where  $G$  and  $D$  are defined by multilayer perceptrons, the entire system can be trained with backpropagation. There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples. Experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples.



나는 강해진다,  
나의 적이 강한 만큼



## 1. GAN의 기본 원리

# 생성자 (Generator)와 판별자 (Discriminator)의 대립과 경쟁을 통해서 모델을 훈련시켜서 사용자가 만족할만한 수준의 결과를 생성하는 생성 모델

생성자 (Generator)  $G: p_z(z) \rightarrow p_g$

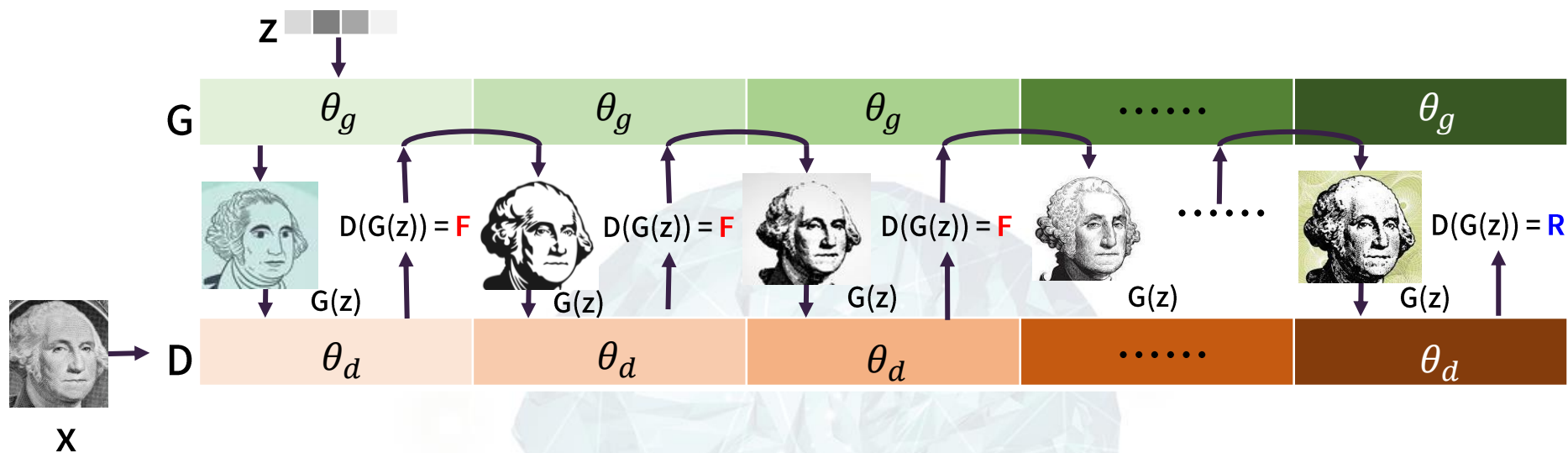
- 대립을 통해서 더 우수한 품질의 결과를 도출하도록 훈련됨
- Trying to produce fake currency

판별자 (Discriminator)  $D: D(x; \theta_d) \rightarrow \text{fake/real}$

- 생성자의 결과가 만족할 수준인지를 판별하여 fake/real의 결과를 도출함
- $x \sim p_g \rightarrow \text{fake}, x \sim \text{data} \rightarrow \text{real}$
- Trying to detect the counterfeit currency

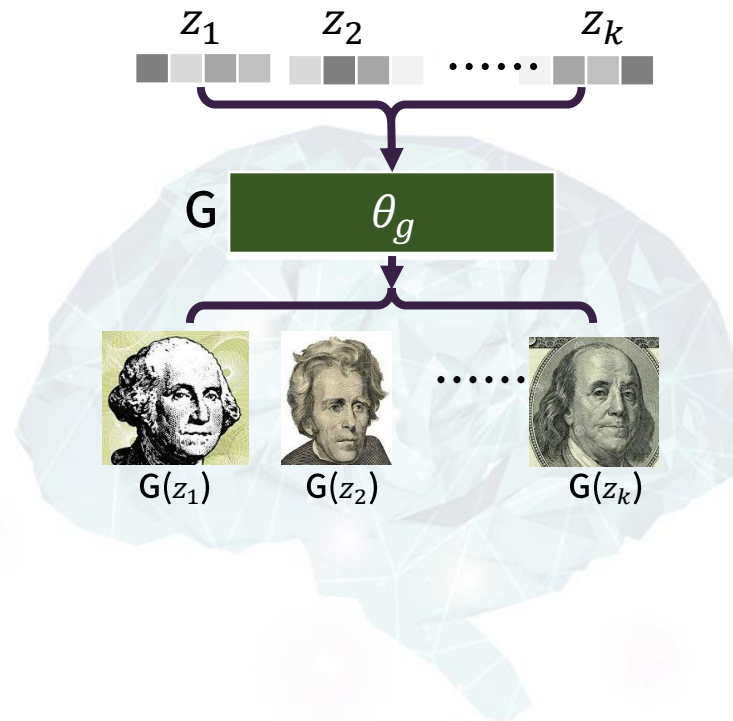
## 1. GAN의 기본 원리

# 생성자 (Generator)와 판별자 (Discriminator)의 대립과 경쟁을 통한 GAN의 훈련



## 1. GAN의 기본 원리

GAN의 훈련이 끝나면 마지막 parameter ( $\theta_g$ )를 저장한 Generator를 이용해서 다양한 샘플을 생성



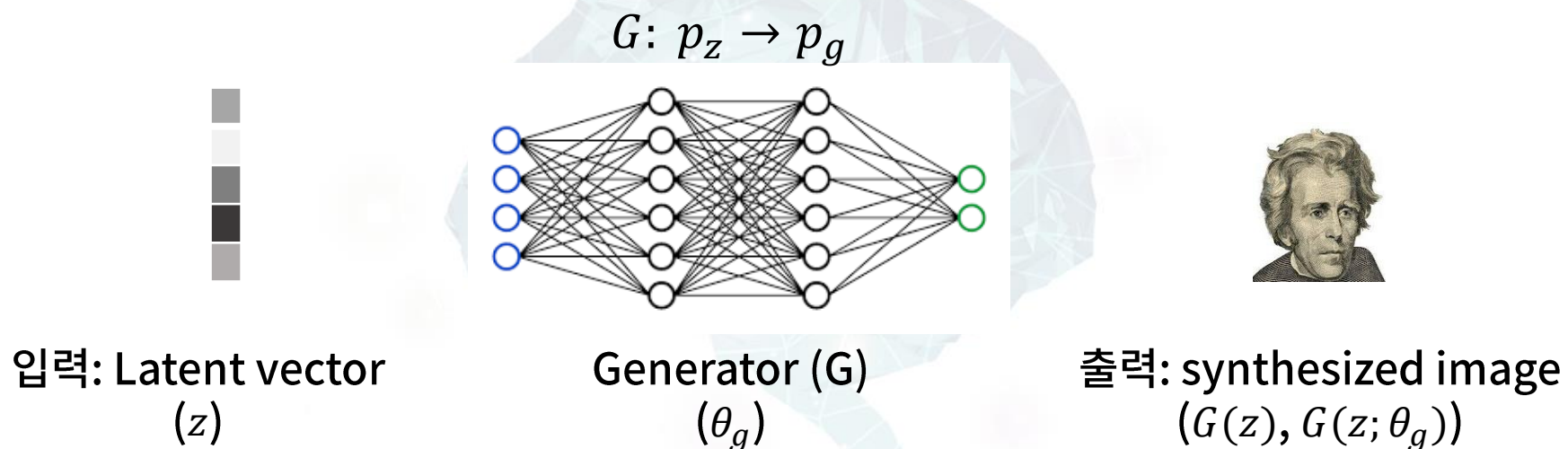


## 2. GAN의 구성 요소 (1): generator

## 2. GAN의 구성 요소 (1): generator

# 생성자 (Generator)

- Decoder와 유사한 구조
- 입력(Latent vector)를 받아서  
결과(Synthesized image)를 생성하는 모듈
- Vanilla GAN에서는 convolution을 사용하지 않음

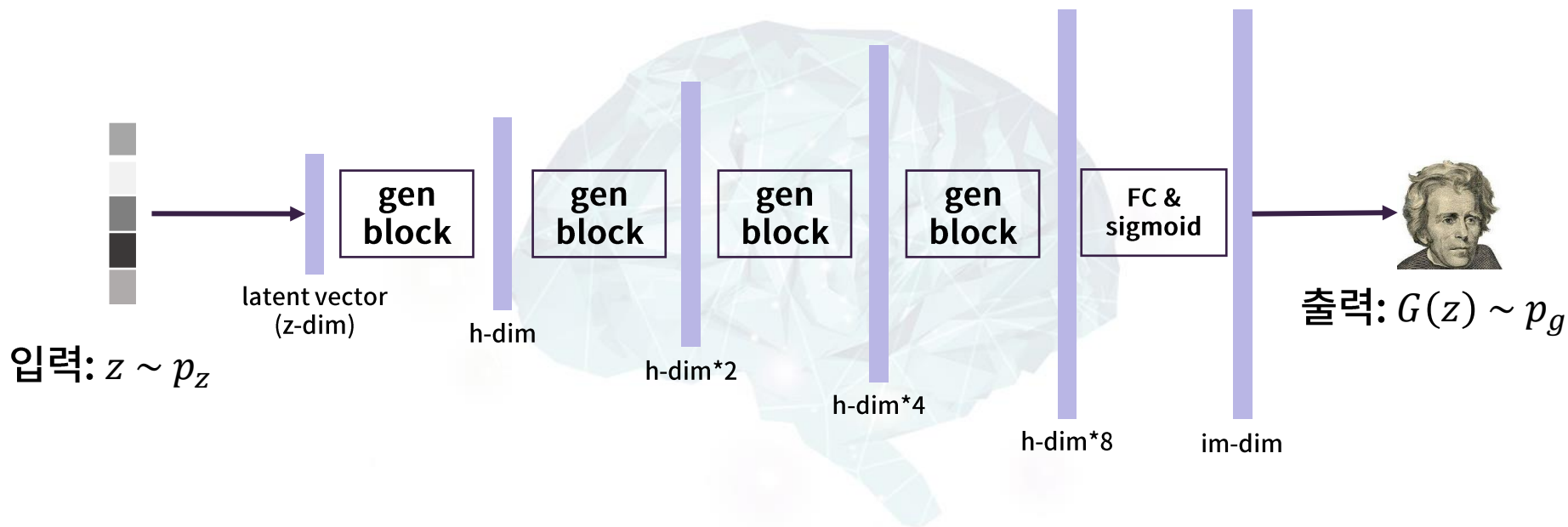




## 2. GAN의 구성 요소 (1): generator

# Generator의 구조 (1): overall

- n 개의 generator block으로 구성됨 ( $n = 4$ )
- generator block은 낮은 해상도의 입력을 받아서 높은 해상도의 출력을 생성



## 2. GAN의 구성 요소 (1): generator

# Generator의 구조 (2): Generator block의 구조

- input\_dim과 output\_dim을 parameter로 받음
- linear layer와 batch norm, ReLU 함수로 구성

```
def generator_block(input_dim, output_dim):  
    return nn.Sequential(  
        nn.Linear(input_dim, output_dim),  
        nn.BatchNorm1d(output_dim),  
        nn.ReLU(inplace=True),  
    )
```

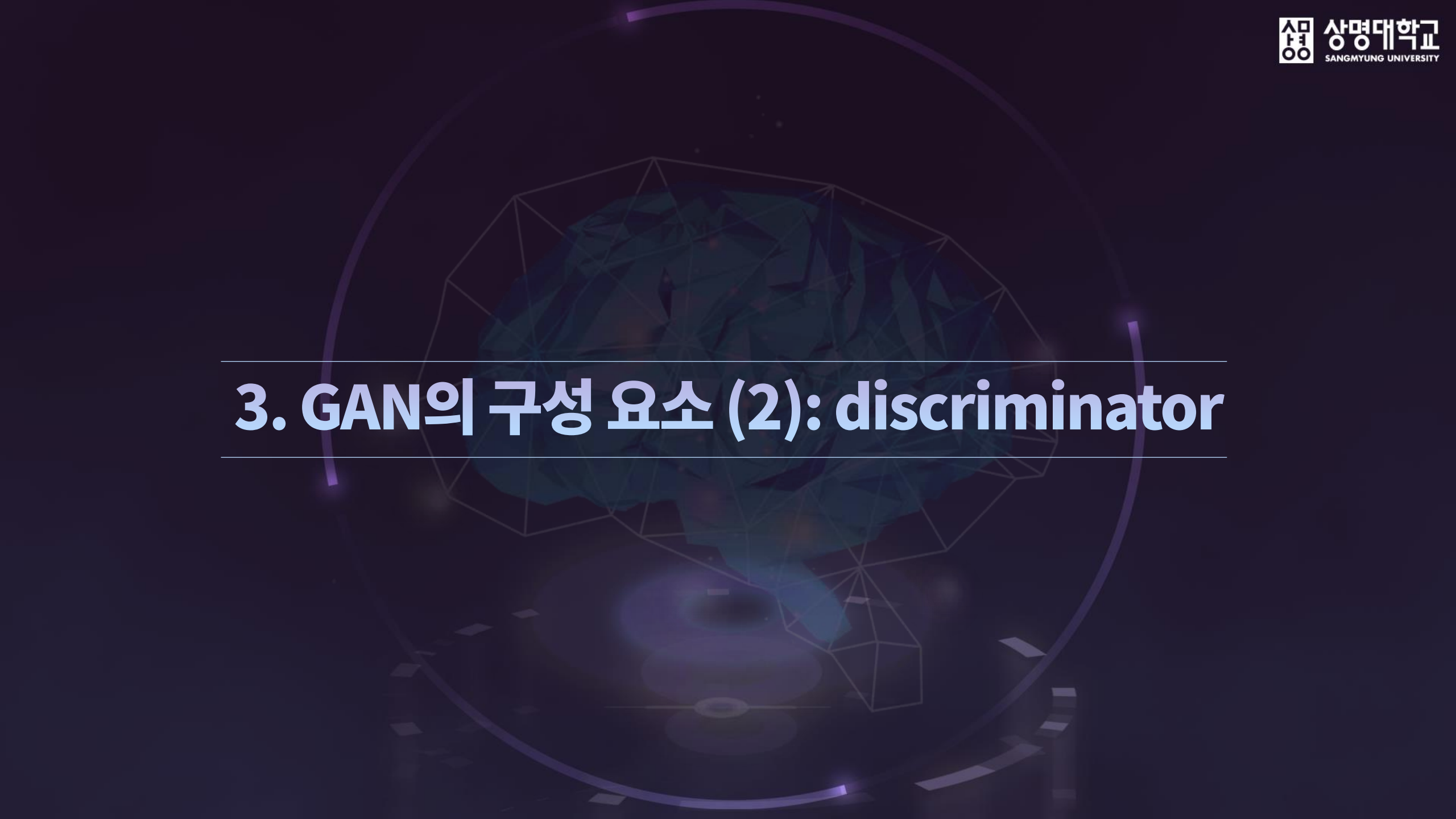


## 2. GAN의 구성 요소 (1): generator

# Generator의 구조 (3): generator의 구조

- 4개의 Generator block와 FC layer, Sigmoid 함수로 구성
- 28x28 해상도의 MNIST dataset 생성
  - 입력:  $z\_dim = 10$
  - 출력:  $im\_dim = 784$

```
class Generator(nn.Module):  
    def __init__(self, z_dim=10, im_dim=784, hidden_dim=128):  
        super(Generator, self).__init__()  
        # Build the neural network  
        self.gen = nn.Sequential(  
            generator_block(z_dim, hidden_dim),  
            generator_block(hidden_dim, hidden_dim * 2),  
            generator_block(hidden_dim * 2, hidden_dim * 4),  
            generator_block(hidden_dim * 4, hidden_dim * 8),  
  
            nn.Linear(hidden_dim * 8, im_dim),  
            nn.Sigmoid()  
        )
```

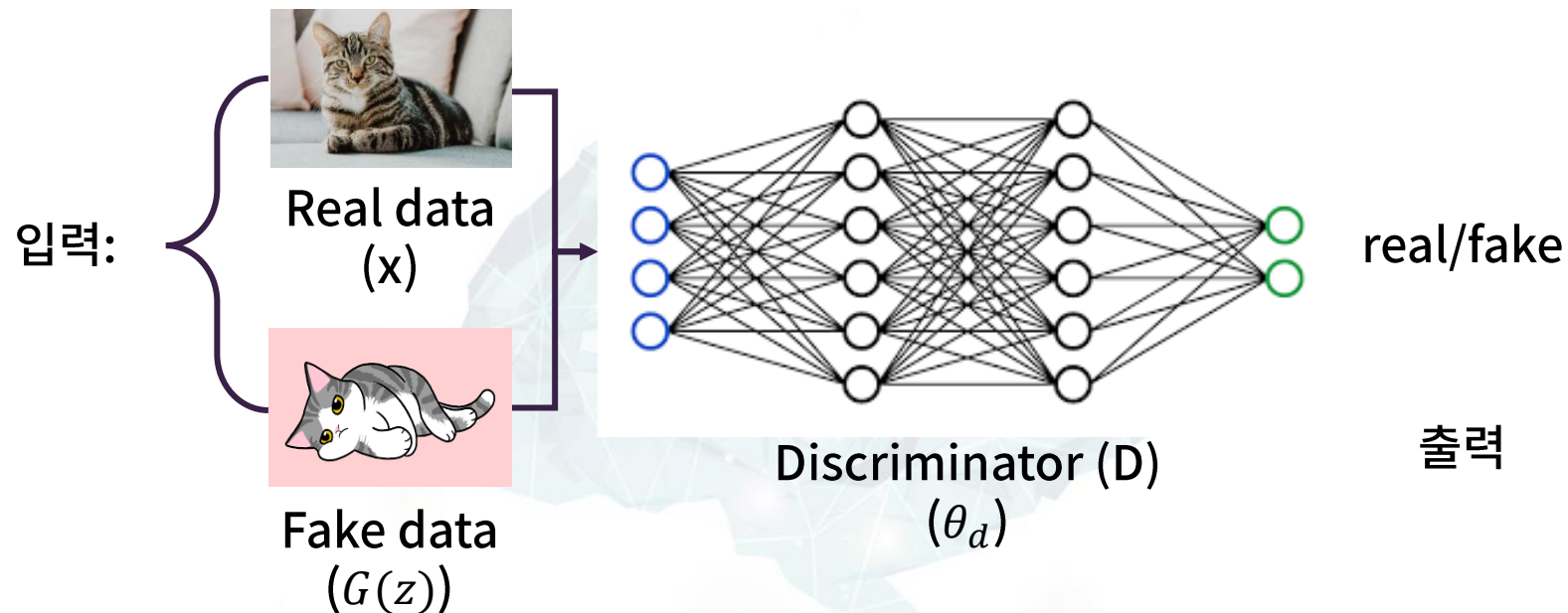


## 3. GAN의 구성 요소 (2): discriminator

## 3. GAN의 구성 요소 (2): discriminator

# 판별자 (discriminator)

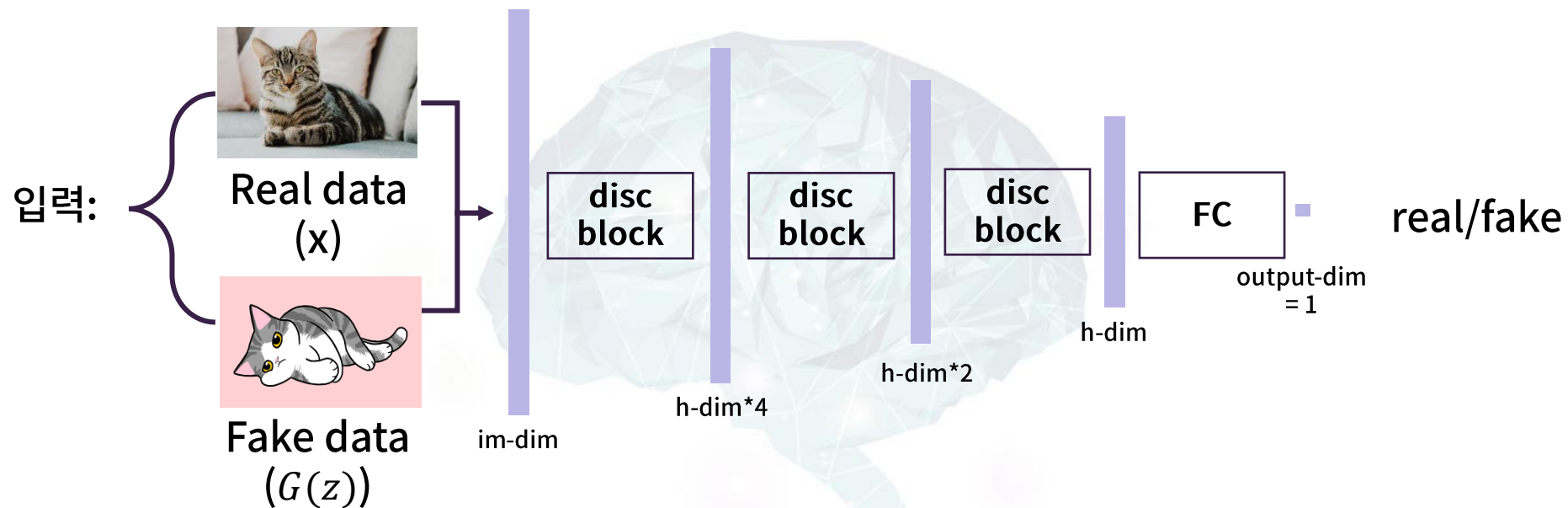
- 입력 (Real data or fake data)을 받아서 real/fake를 판정하는 모듈
- Classifier와 유사한 역할을 수행함



### 3. GAN의 구성 요소 (2): discriminator

## Discriminator의 구조 (1): overall

- $n$  개의 Discriminator block으로 구성됨 ( $n = 3$ )
- Discriminator block은 높은 해상도의 입력을 받아서 낮은 해상도의 출력을 생성





### 3. GAN의 구성 요소 (2): discriminator

## Discriminator의 구조 (2) : Discriminator block의 구조

- input\_dim과 output\_dim을 parameter로 받음
- linear layer와 ReLU 함수로 구성

```
def discriminator_block(input_dim, output_dim):  
    return nn.Sequential(  
        nn.Linear(input_dim, output_dim),  
        nn.LeakyReLU(0.2, inplace=True)  
    )
```



### 3. GAN의 구성 요소 (2): discriminator

## Discriminator의 구조 (3) : Discriminator의 구조

- 3개의 Discriminator block와 FC layer로 구성
- 28x28 해상도의 MNIST dataset를 처리하는 Discriminator
  - ✓ 입력: im\_dim = 784
  - ✓ 출력: 1

```
class Discriminator(nn.Module):  
    def __init__(self, im_dim=784, hidden_dim=128):  
        super(Discriminator, self).__init__()  
        self.disc = nn.Sequential(  
            discriminator_block(im_dim, hidden_dim * 4),  
            discriminator_block(hidden_dim * 4, hidden_dim * 2),  
            discriminator_block(hidden_dim * 2, hidden_dim),  
            nn.Linear(hidden_dim, 1)  
        )
```



## 4. GAN의 구성 요소 (3): loss 함수

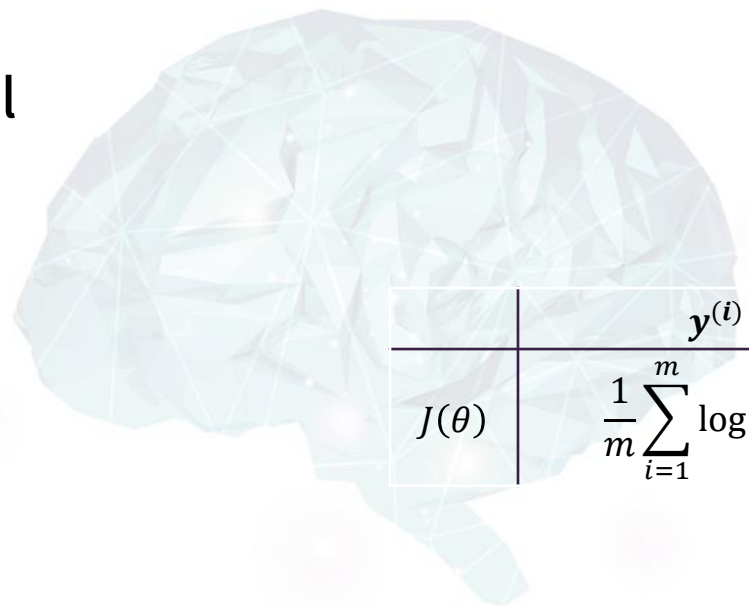
## 4. GAN의 구성 요소 (3): loss 함수

# GAN loss 함수는 BCE에서 도출 (1)

## ● BCE: Binary Cross Entropy

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m [y^{(i)} \log h(x^{(i)}, \theta) + (1 - y^{(i)}) \log(1 - h(x^{(i)}, \theta))]$$

- ✓  $y^{(i)}$ : label
- ✓  $h(\cdot)$ : predicted label
- ✓  $x^{(i)}$ : input
- ✓  $\theta$ : parameter



	$y^{(i)} = 1$	$y^{(i)} = 0$
$J(\theta)$	$\frac{1}{m} \sum_{i=1}^m \log h(x^{(i)}, \theta)$	$\frac{1}{m} \sum_{i=1}^m \log(1 - h(x^{(i)}, \theta))$

## 4. GAN의 구성 요소 (3): loss 함수

# GAN loss 함수는 BCE에서 도출 (2)

## GAN loss 함수 유도

- ✓ 훈련 데이터 (real,  $x$ )  $\rightarrow y^{(i)}=1$ ,
- ✓ 생성 데이터 (fake,  $G(z)$ )  $\rightarrow y^{(i)} = 0, x^{(i)} = G(z^{(i)}, \theta_g)$
- ✓ Discriminator  $\rightarrow h(\cdot, \theta_d)$
- ✓  $\frac{1}{m} \sum_{i=1}^m \cdot \rightarrow \mathbb{E}(\text{expected})$

$$J(\theta) = \underbrace{\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log h(x^{(i)}, \theta)]}_{\mathbb{E}[\log D(x, \theta_d)]} + \underbrace{\frac{1}{m} \sum_{i=1}^m [(1 - y^{(i)}) \log(1 - h(x^{(i)}, \theta))]}_{\mathbb{E}[\log(1 - D(G(z, \theta_g), \theta_d))]}$$

$$V(\theta_d, \theta_g) = \mathbb{E}_{x \sim P_{data}(x)} [\log D(x, \theta_d)] + \mathbb{E}_{z \sim P_z(z)} [\log(1 - D(G(z, \theta_g), \theta_d))]$$

## 4. GAN의 구성 요소 (3): loss 함수

# GAN loss 함수는 BCE에서 도출 (3)

### ➡ BCE와 GAN loss 함수의 차이

- ☑ BCE →  $J(\theta)$ 를 최대화하는  $\theta$ 를 찾을 것

$$\max_{\theta} J(\theta) = \frac{1}{m} \sum_{i=1}^m [y^{(i)} \log h(x^{(i)}, \theta)] + \frac{1}{m} \sum_{i=1}^m [(1 - y^{(i)}) \log(1 - h(x^{(i)}, \theta))]$$

max 방향으로만  
최적화하기 때문에 훈련이  
비교적 쉬움

- ☑ GAN loss 함수 → 역할에 따라서 다른 의미를 추구

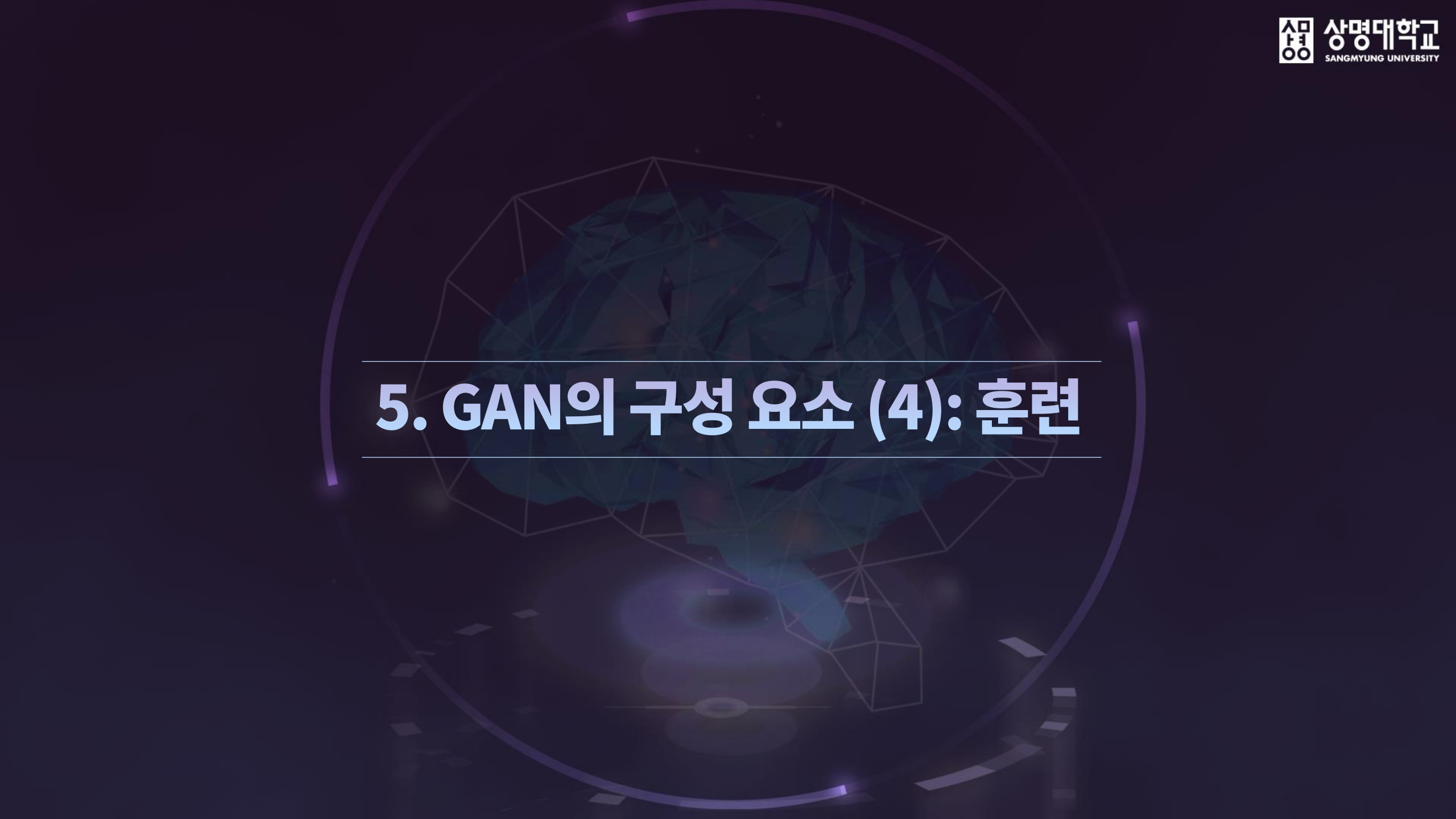
» Discriminator:  $D(x)$ 는 1을,  $D(G(z))$ 는 0을 출력할 것  
→  $\log D(x)$  &  $\log(1 - D(G(z)))$ 가 max

» Generator:  $D(G(z))$ 가 1을 출력할 것 →  $\log(1 - D(G(z)))$ 가 min

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

max와 min 방향으로  
최적화하기 때문에  
훈련이 어려움





## 5. GAN의 구성 요소 (4): 훈련

## 5. GAN의 구성 요소 (4): 훈련

아빠와 엄마가 다 공부하라고 하네?  
그러면 나는 공부할 수 밖에...

아빠는 놀라고 하고, 엄마는 공부하라고 하네?  
그러면 나는 뭘 하죠?



## 5. GAN의 구성 요소 (4): 훈련

# GAN 훈련 목표

- 훈련 데이터  $x$ 와 일치하는  $G(z)$ 를 생성하는 것

$$P_{data} = P_z$$

- $P_{data} = P_z$ 되면  $D(\cdot)$ 는  $x$ 와  $G(z)$ 를 구분하지 못함

$$D(\cdot) = \frac{1}{2}$$

$$V(G, D) = \mathbb{E}[\log D(x)] + \mathbb{E}\left[\log\left(1 - D(G(z))\right)\right] = \log \frac{1}{2} + \log \frac{1}{2} = -\log 4$$

- GAN은  $\min_G \max_D V(G, D)$ 를 추구하기 때문에 수렴이 어려울 수 있지만, 최적의 경우에 수렴되는 값이 있음을 증명함

## 5. GAN의 구성 요소 (4): 훈련

# GAN 훈련의 어려움

- D와 G를 동시에 훈련해야 함
  - ☑ D는 max, G는 min을 추구
- 훈련 초기에 생성되는  $G(z)$ 는 품질이 안 좋음
- ❗ 매번  $D(G(z))$ 이 0에 가까우면,  
 $\log(1-D(G(z)))$ 이 0에 가까운 값을 갖게 됨
- ❗ Gradient descent를 적용하기 힘들

## 5. GAN의 구성 요소 (4): 훈련

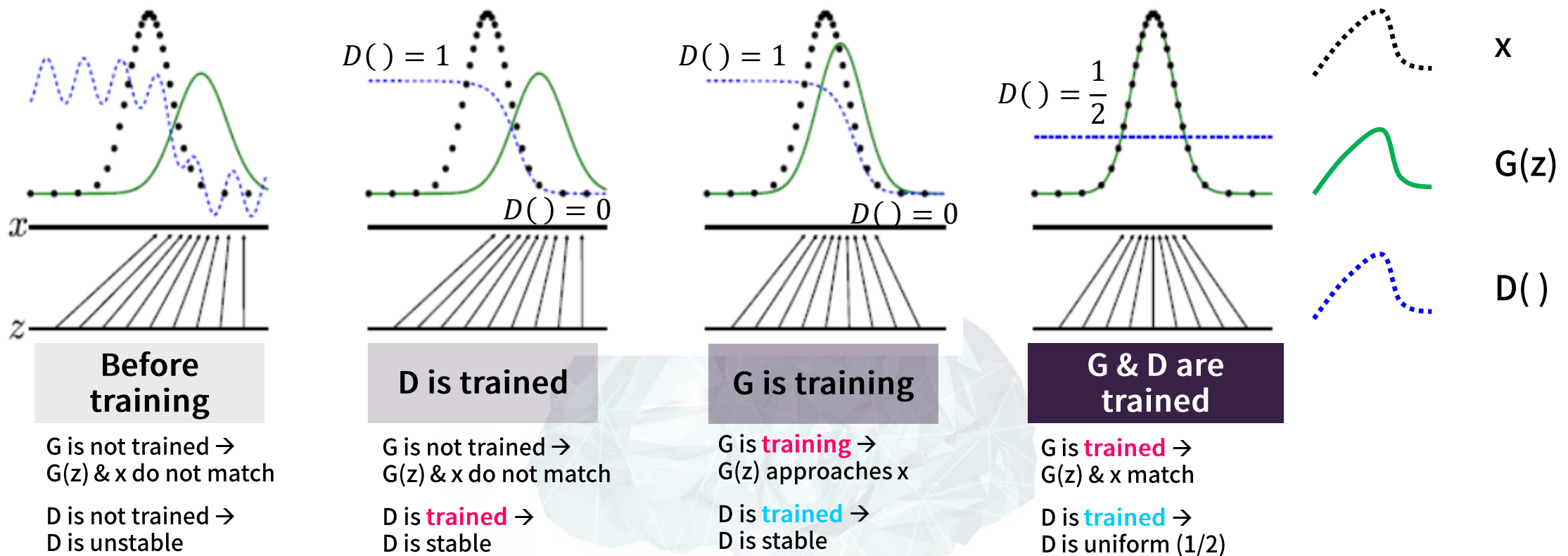
# GAN의 heuristic한 훈련 전략

- 초기에는  $\log(1 - D(G(z)))$ 를 사용하지 말고  $D(G(z))$ 를 최대화시키는 방향으로 훈련시킬 것
- $D(G(z))$ 는  $\log(1 - D(G(z)))$ 보다 값이 더 크기 때문에 Gradient가 소실되는 경우를 피할 수 있음

In practice, equation 1 may not provide sufficient gradient for  $G$  to learn well. Early in learning, when  $G$  is poor,  $D$  can reject samples with high confidence because they are clearly different from the training data. In this case,  $\log(1 - D(G(z)))$  saturates. Rather than training  $G$  to minimize  $\log(1 - D(G(z)))$  we can train  $G$  to maximize  $\log D(G(z))$ . This objective function results in the same fixed point of the dynamics of  $G$  and  $D$  but provides much stronger gradients early in learning.

## 5. GAN의 구성 요소 (4): 훈련

# GAN 훈련 과정





## 5. GAN의 구성 요소 (4): 훈련

# GAN 훈련 알고리즘

Discriminator  
training

for number of training iterations do

for  $k$  steps do

- Sample minibatch of  $m$  noise samples  $\{z^{(1)}, \dots, z^{(m)}\}$  from noise prior  $p_g(z)$ .
- Sample minibatch of  $m$  examples  $\{x^{(1)}, \dots, x^{(m)}\}$  from data generating distribution  $p_{\text{data}}(x)$ .
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[ \log D(x^{(i)}) + \log (1 - D(G(z^{(i)}))) \right].$$

end for

Generator  
training

- Sample minibatch of  $m$  noise samples  $\{z^{(1)}, \dots, z^{(m)}\}$  from noise prior  $p_g(z)$ .
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log (1 - D(G(z^{(i)}))).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

