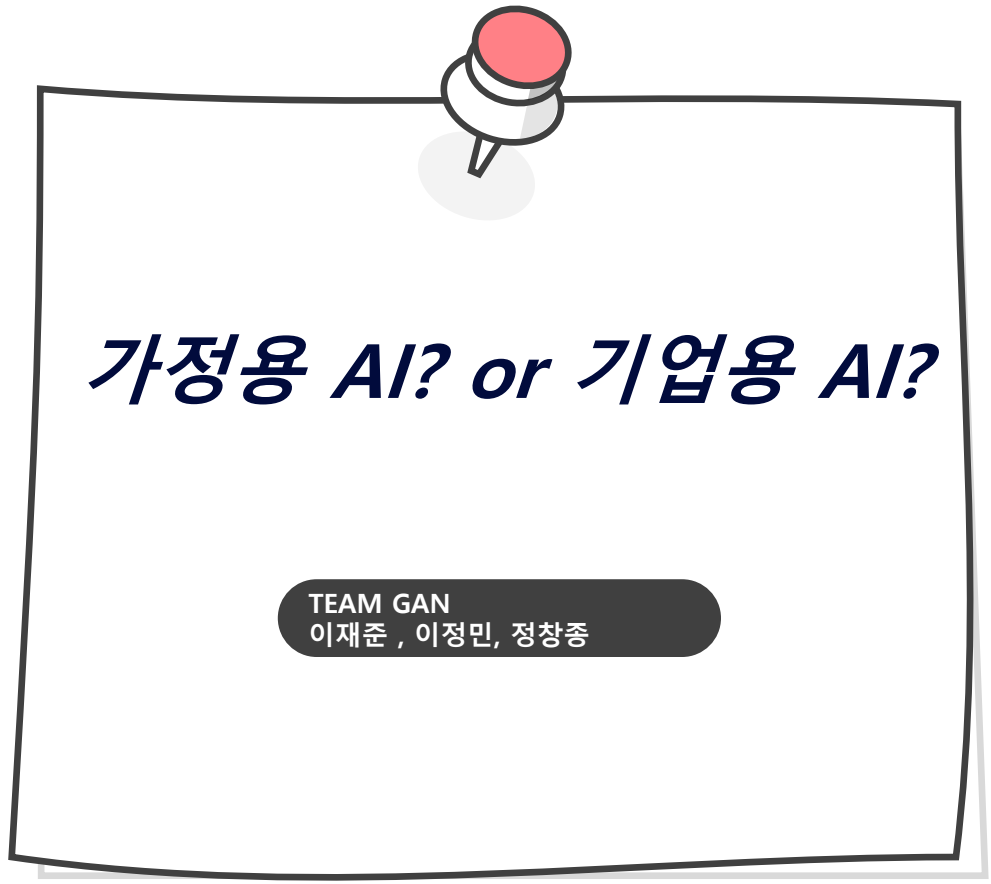


가정용 AI? or 기업용 AI?

TEAM GAN
이재준 , 이정민, 정창종

PPT PRESENTATION

-딥러닝(Deep Learning)이란?

딥러닝(Deep Learning)이란 여러 층을 가진 인공신경망(Artificial Neural Network, ANN)을 사용하여 머신러닝 학습을 수행하는 것으로 심층학습이라고도 부릅니다. 따라서 딥러닝은 머신러닝과 전혀 다른 개념보단 머신러닝의 한 종류라고 할 수 있습니다.

-딥러닝 (Deep Learning) vs 머신러닝 (Machine Learning)

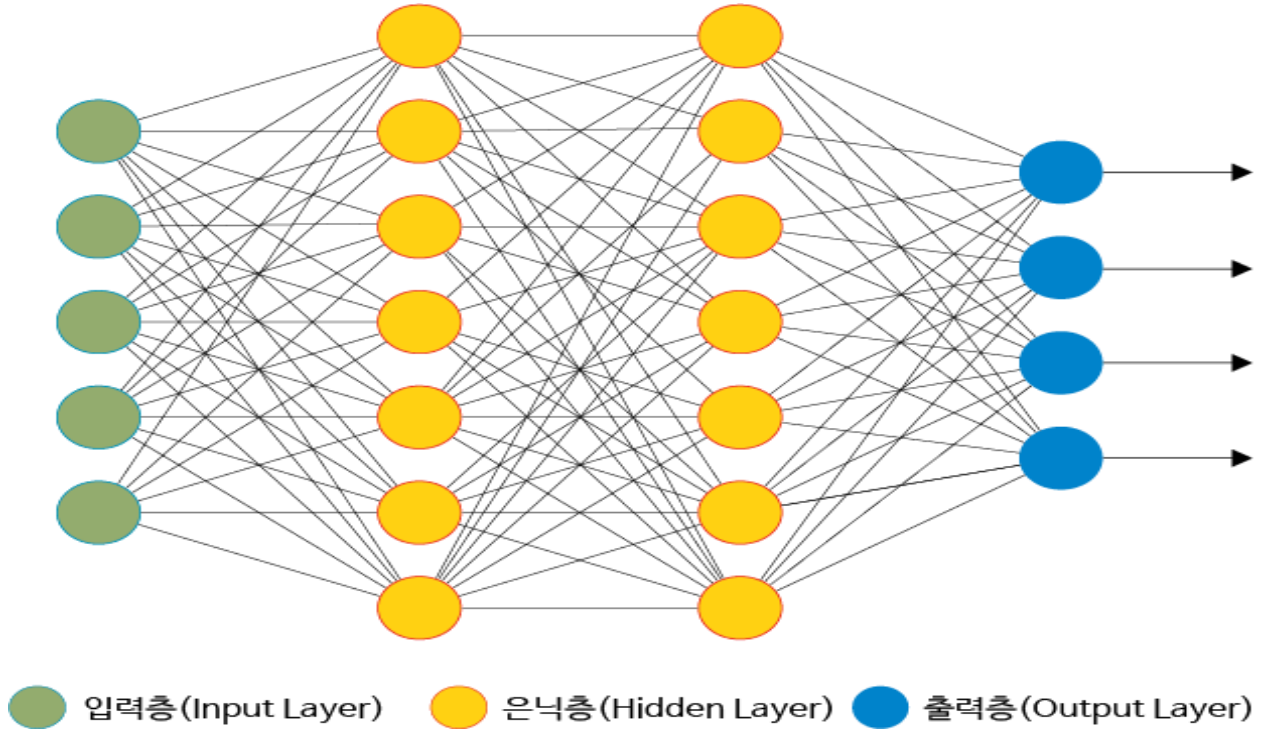
딥러닝과 머신러닝의 가장 큰 차이점은 바로 기계의 자가 학습 여부로 볼 수 있습니다.

따라서 딥러닝이란 기계가 자동으로 대규모 데이터에서 중요한 패턴 및 규칙을 학습하고, 이를 토대로 의사결정이나 예측 등을 수행하는 기술로 정의내릴 수 있습니다.

PPT PRESENTATION

-인공신경망(Artificial Neural Network, ANN)

딥러닝에서 가장 기본이 되는 개념은 바로 신경망(Neural Network)입니다.
신경망이란 인간의 뇌에서 뉴런의 연결 구조를 본떠 만든 네트워크 구조를 인공신경망(Artificial Neural Network, ANN)이라고 부릅니다.



▶딥러닝 기본구조

1)컨볼루션넷(CNN)

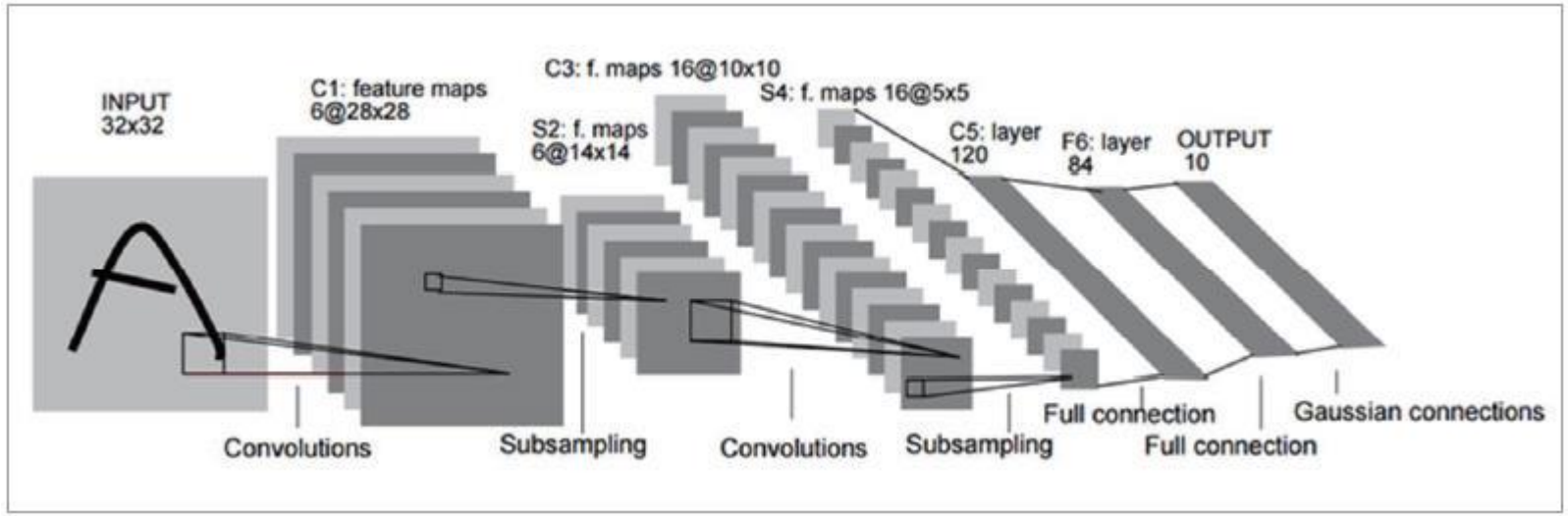
2)딥빌리프넷(BBN)

3)딥하이퍼넷(BHN)

PPT PRESENTATION

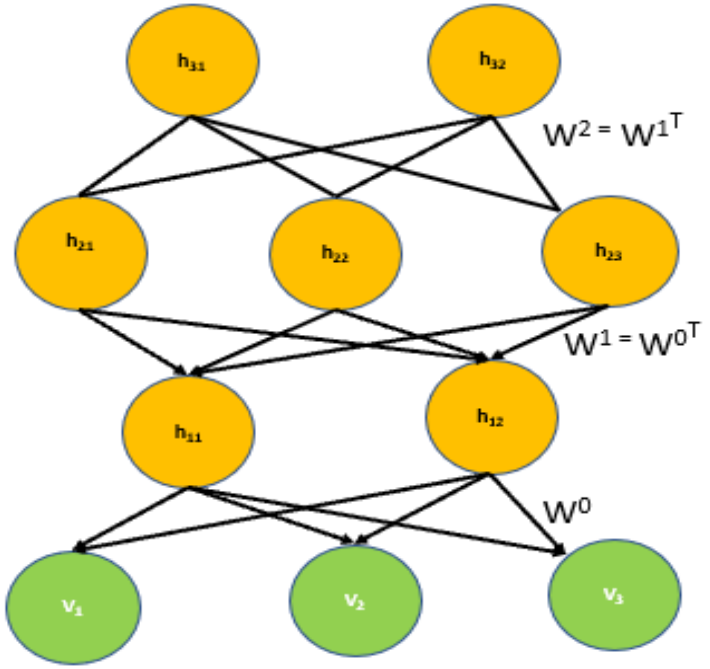
1.컨볼루션넷(**CNN**)
감독학습, 입력데이터를 변별하기 위한 변별적 학습

장점: 패턴 분류 성능 우수
단점: 모델로부터 샘플 생성 불가



2.딥빌리프넷(**DBN**)
무감독학습, 입력데이터를 재생성 하는 생성적 학습

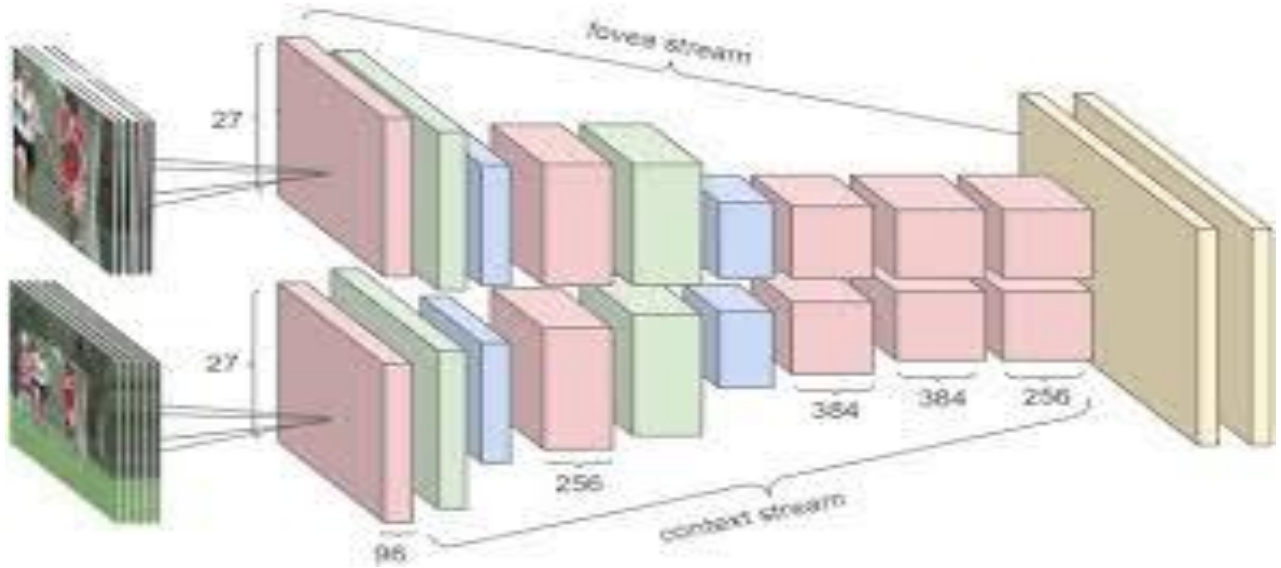
장점: 모델로 부터 새로운 샘플 생성, 데이터 압축
단점: 층간 완전 연결구조로 패턴 분류 적용시 분류 성능 낮음



PPT PRESENTATION

3.딥하이퍼넷(**DHN**)
무감독학습 생성모델, 층간 다양한 부분 연결하는 커널들을 자동으로 자기조직 원리에 의해 찾는다

장점: 모델로 부터 새로운 샘플 생성, 유용한 빌딩블록 찾음
단점:많은 메모리 및 컴퓨팅 파워 요구



-No-Free-Lunch Theorem

특정 계열의 문제를 잘 푸는 알고리즘은 다른 문제는 잘 풀지 못할 수 있음
다양한 딥러닝 모델이 필요한 이유

	Deep Belief Net (DBN)	Convolutional Neural Net (CNN)	Deep Hypernet (DHN)
감독/무감독	감독/무감독	감독	감독/무감독
변별/생성 모델	생성	변별	생성
예측/모듈 이해	예측++/모듈−	예측+++/모듈+	예측+/모듈+++
추론 가능성	추론++	추론+	추론++++
연결성	Full/Compact	Partial/Convolved	Partial/Sparse
깊이	깊이+++	깊이++++	깊이++
배치/온라인 학습	배치	배치	온라인

▶복합 딥러닝 구조의 종류

1)딥강화학습모델(DQN)

2)순환신경망모델(RNN)

3)종단학습모델(N2N)

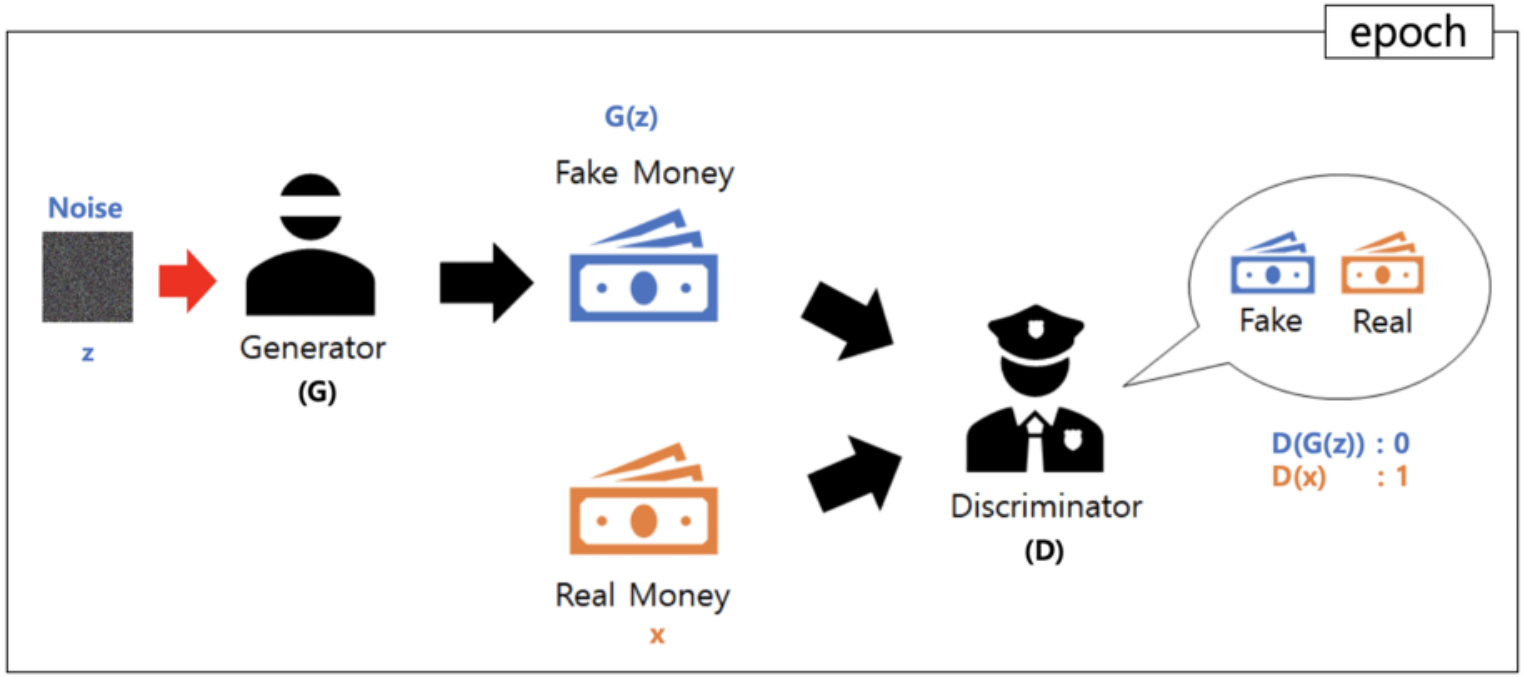
4)대립학습모델(GAN)

5)잔차학습모델(ResNet)

6)메모리넷모델(MemNet)

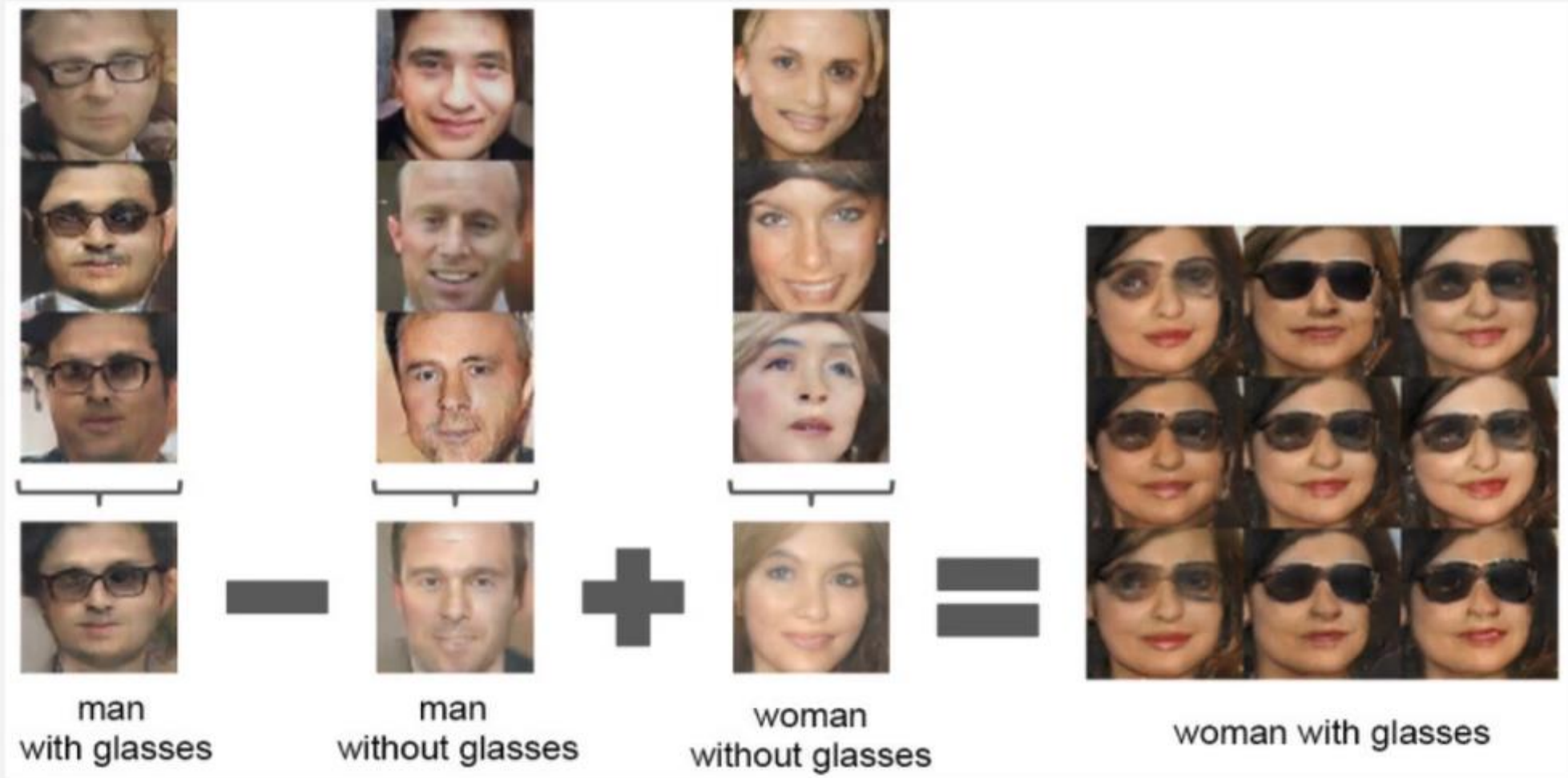
PPT PRESENTATION

▶대립학습모델, **GAN**(Generative Adversarial Network)이란?
게임 이론에 기반, 가짜 이미지를 만드는 Generator와 가짜와 진짜를 구별하는 Discriminator가 마치 게임과 같이 서로 경쟁하며 학습하는 모델



위조지폐 게임

대립학습 모델 (GAN) 응용

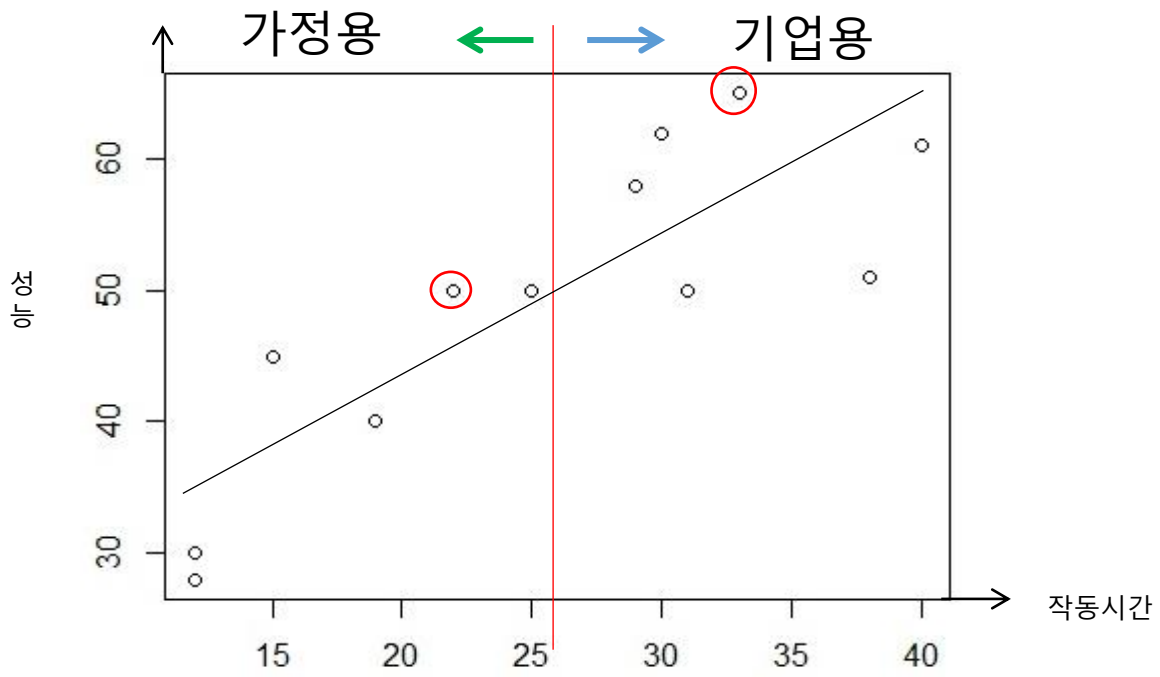


GAN의 가치

PPT PRESENTATION

▶프로젝트 목표

-GAN 각 구성요소를 변경하며 작동 시간 및 성능(정확도)을 비교하여 시각화
-가정용 AI를 겨냥한 의도적인 성능 다운으로 기대해볼 수 있는 컴퓨팅 파워 부족 해소 능력이 가장 좋은 모델 추출
-기업용 AI를 겨냥한 가장 높은 수준의 정확도 혹은 성능을 보이는 모델 추출



PPT PRESENTATION

손실함수(목표함수, 수식변경)

$$\min_{G}\max_{D} V(D,G) = \mathbb{E}_{x\sim Pdata(x)}[logD(x)] + \mathbb{E}_{z\sim Pz(z)}[\log(1 - D(G(z)))]$$

Generator는 D(G(z))를 최대화. Discriminator는 D(X) 최대화, D(G(z)) 최소화

PPT PRESENTATION

▶ LSGAN

$$\min_D V_{LSGAN}(D) = E_{x\sim p_{data}(x)}\left[(D(x)-b)^2\right] + E_{z\sim p_z(z)}\left[(D(G(z))-a)^2\right],$$

$$\min_G V_{LSGAN}(G) = E_{z\sim p_z(z)}\left[(D(G(z))-c)^2\right]$$

▶ Hinge loss based GAN

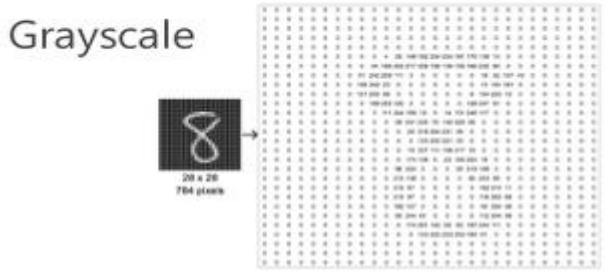
$$V_D\left(\hat{G}, D\right) = E_{x\sim p_{data}(x)}\left[\min(0, -1 + D(x))\right] + E_{z\sim p_z(z)}\left[\min(0, -1 - D\left(\hat{G}(z)\right))\right].$$

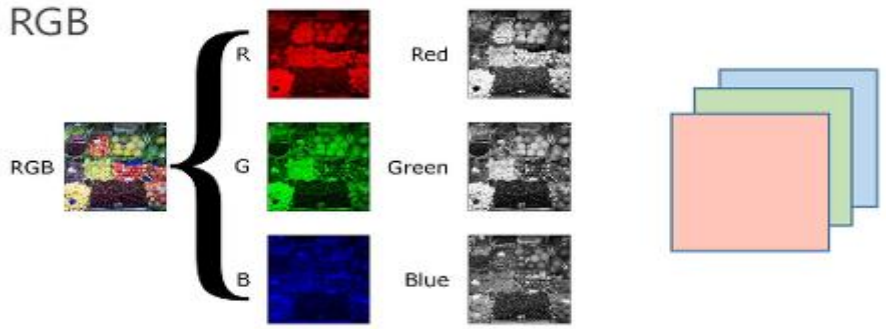
$$V_D\left(G, \hat{D}\right) = -E_{z\sim p_z(z)}\left[\hat{D}(G(z))\right].$$

PPT PRESENTATION

-텐서플로우(Tensorflow) 란?

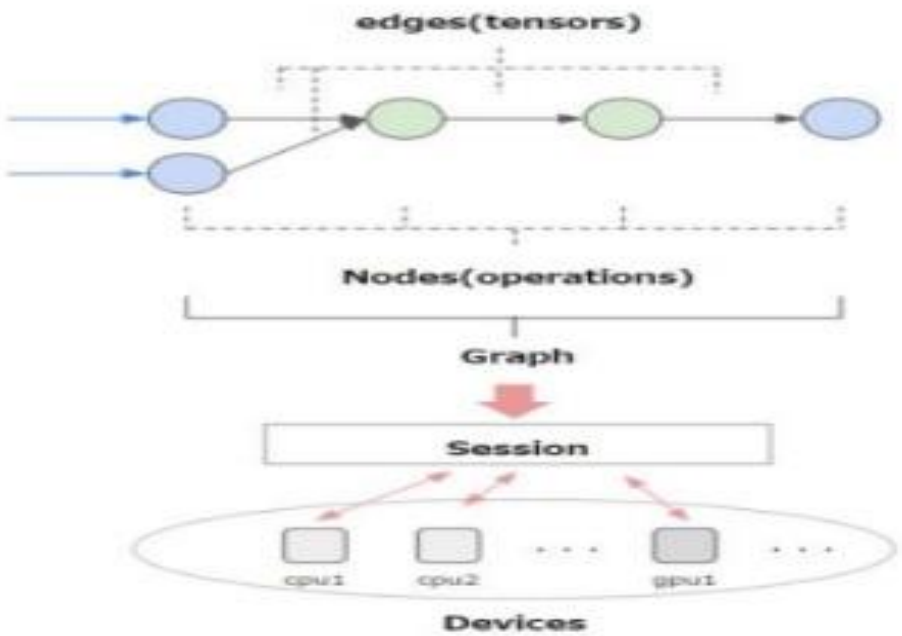
-구글(Google)에서 만든 딥러닝 프로그램을 쉽게 구현할 수 있도록 다양한 기능을 제공해주는 라이브러리
-Tensor(텐서)란 딥러닝에서 데이터를 표현하는 방식
-행렬로 표현할 수 있는 2차원 형태의 배열을 높은 차원으로 확장한 다차원 배열
-텐서 형태의 데이터들이 딥러닝 모델을 구성하는 연산들의 그래프를 따라 흐르면서 연산 진행





Tensorflow 구조

텐서플로우는 edges와 nodes로 구조화된 graph로 프로그램이 구성




```
import tensorflow as tf

a = tf.constant(1)
print(a)
with tf.Session() as sess:
    print(a.eval())
```

```
Tensor("Const_170:0", shape=(), dtype=int32)
1
```

PPT PRESENTATION

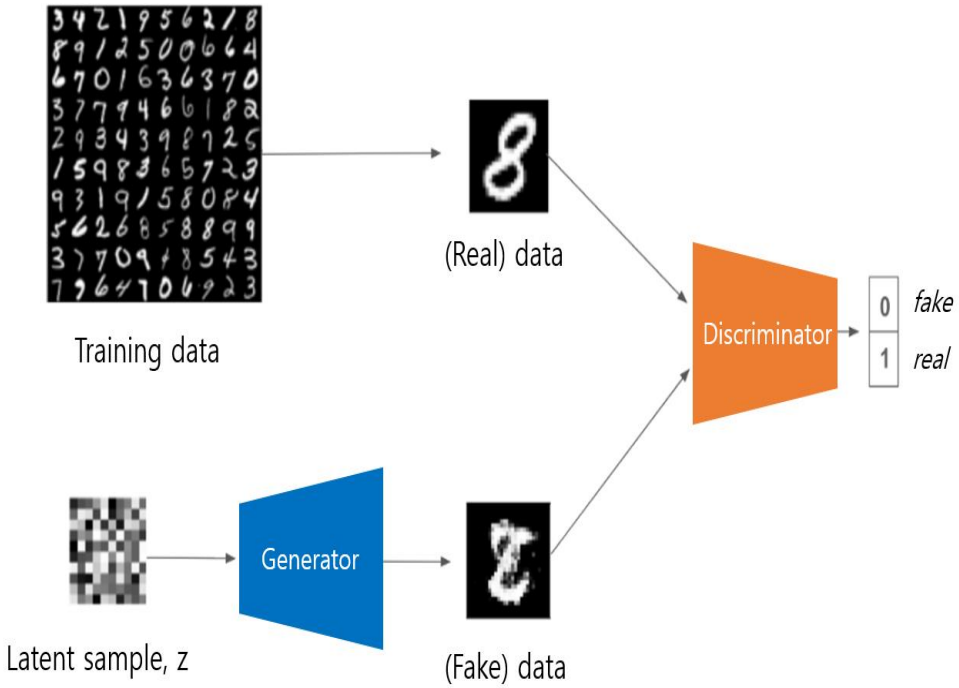
```
In [4]: (train_images, train_labels), (_, _) = tf.keras.datasets.mnist.load_data()
        train_images = train_images.reshape(train_images.shape[0], 28, 28, 1).astype('float32')
        train_images = (train_images - 127.5) / 127.5
        BUFFER_SIZE = 60000
        BATCH_SIZE = 256
        train_dataset = tf.data.Dataset.from_tensor_slices(train_images).shuffle(BUFFER_SIZE).batch(BATCH_SIZE)
```

▶tf.keras.dataset
케라스에서 불러온 데이터셋을 [-1, 1]로 normalize

▶Train_dataset
60000개 단위로 섞고 256개의 샘플 단위마다 모델의 가중치를 업데이트

```
In [5]: def make_generator_model():
            model = tf.keras.Sequential()  #여러개의 레이어
            model.add(layers.Dense(7*7*256, use_bias=False, input_shape=(100,)))
            model.add(layers.BatchNormalization())
            model.add(layers.LeakyReLU())
```

```
In [7]: def make_discriminator_model():
            model = tf.keras.Sequential() #여러개의 레이어
            model.add(layers.Conv2D(64, (5, 5), strides=(2, 2), padding='same',
                                    input_shape=[28, 28, 1]))
            model.add(layers.LeakyReLU())
            model.add(layers.Dropout(0.3))
```

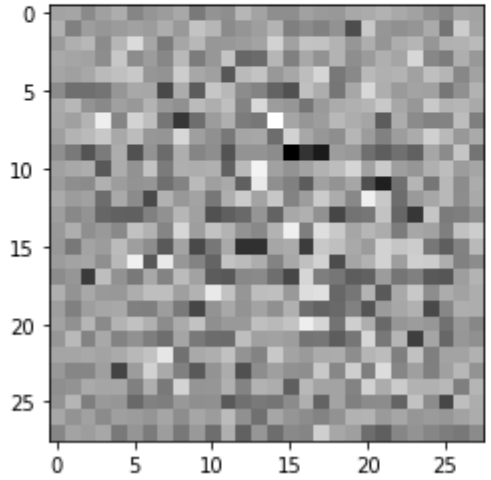


▶tf.keras.Sequential
여러 개의 레이어의 목록을 Sequential 생성자에 전달, Sequential 모델 만들기

PPT PRESENTATION

In [6]:
```
generator = make_generator_model()
noise = tf.random.normal([1, 100])
generated_image = generator(noise, training=False)
plt.imshow(generated_image[0, :, :, 0], cmap='gray')
```

Out[6]: <matplotlib.image.AxesImage at 0x298d6380820>



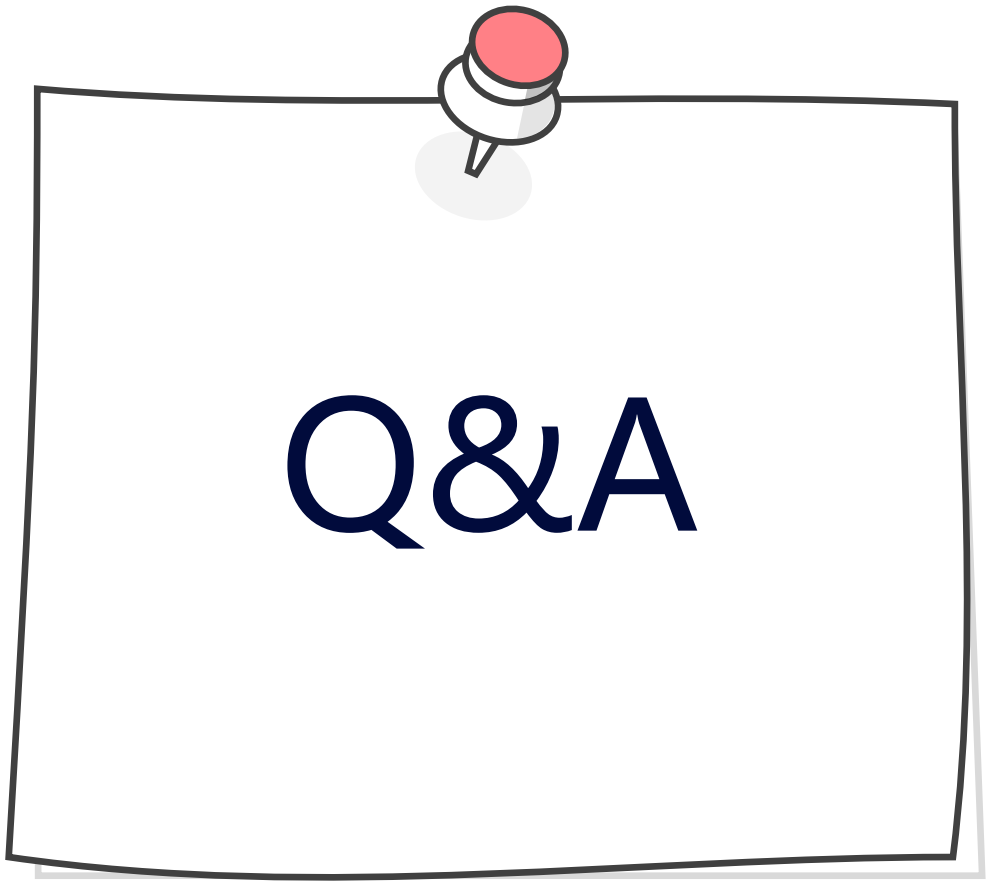
▶tf.random.normal
정규 분포를 통한 난수를 노이즈로 두고
Generator 함수 통한 이미지 생성

▶tf.keras.losses.Binarycrossentropy
이미지 분류가 가능하도록 loss 함수를 이용한
새로운 Discriminator를 정의

In [8]:
```
discriminator = make_discriminator_model()
decision = discriminator(generated_image)
print (decision)
cross_entropy = tf.keras.losses.BinaryCrossentropy(from_logits=True)
```

```
tf.Tensor([[0.00091641]], shape=(1, 1), dtype=float32)
```

Discriminator는 해당 생성 이미지를 보고 판단!


Q&A

감사합니다.