GAN

목 차

1 이전 발표 질문 답변

- CNN vs RNN
- 음성인식 원리

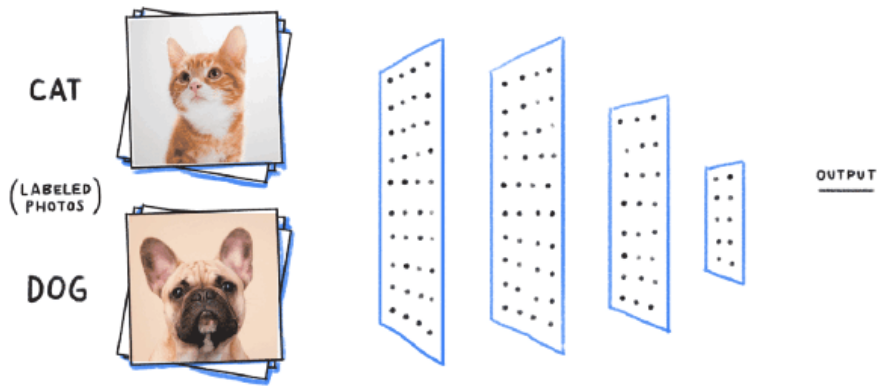
2 GAN

- GAN?
- 생성자
- GAN 모델 생성

CNN vs RNN

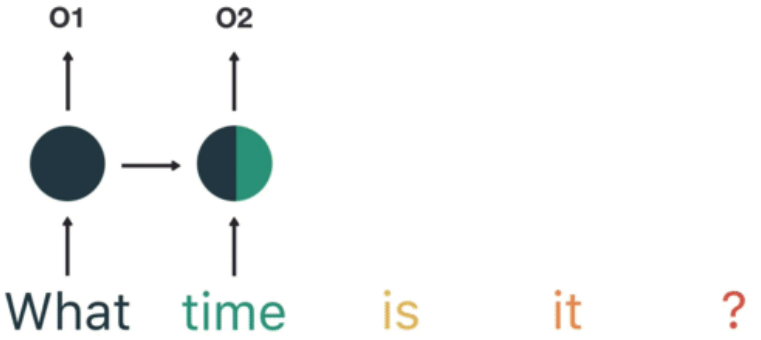
CNN

- 합성곱신경망



RNN

- 순환신경망



CNN vs RNN

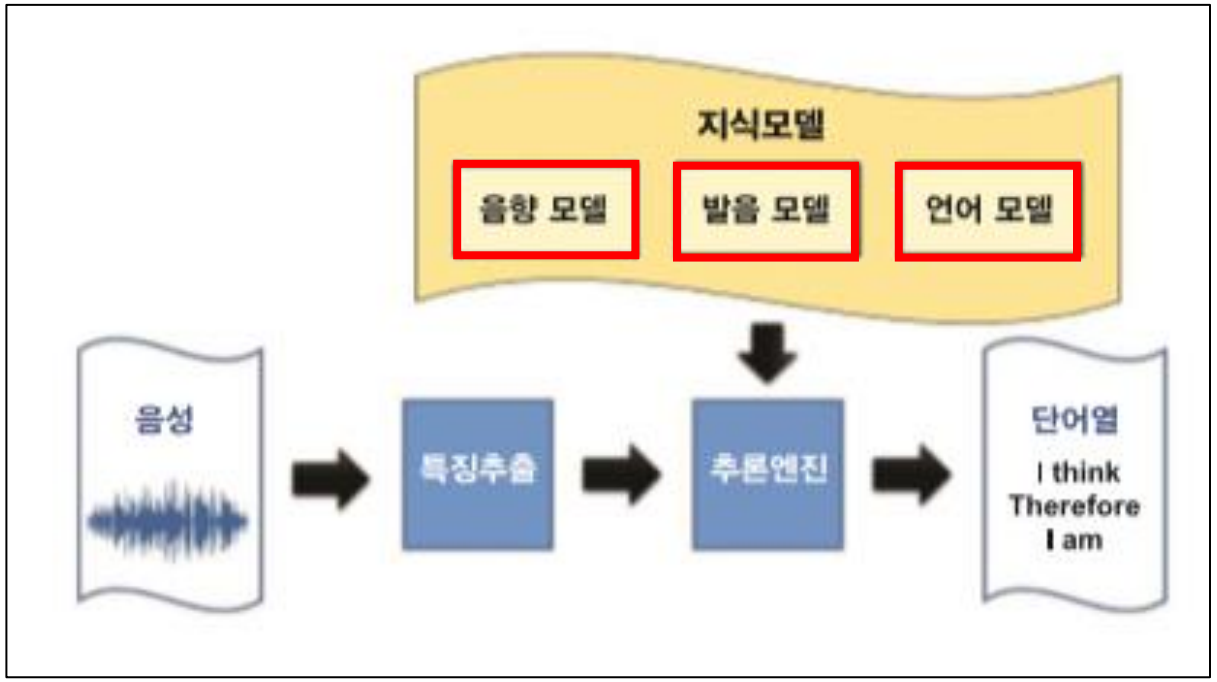
CNN

- 합성곱신경망
- 컨볼루션
- 풀링
- 영상 및 이미지

RNN

- 순환신경망
- 과거 학습을 현재학습에 반영
- 반복적이고 순차적인 데이터에 효과적
- 음성인식, 텍스트

음성인식 원리


지식모델
음향 모델
발음 모델
언어 모델
음성
특징추출
추론엔진
단어열
I think
Therefore
I am


음성신호
"Dog"
1
How Speech is Recognized
In this simplified example, the person begins to dictate a message.
샘플링
2
The PDA's microphone captures the sound waves, and an analog-to-digital converter converts the analog signal into a digital one.
잡음제거
3
The sample is then filtered to remove most, but not all, noise frequencies.
특징추출
1001101110011000
4
The speech sample is chopped up into ~10-ms frames, spectrally significant features are extracted, and the rest of the signal is thrown out.
패턴비교
Dog
5
The speech engine compares the extracted features to its acoustic and language models, searching for what is most likely to have been said.
인터페이스
Dog
6
The speech is recognized and displayed.

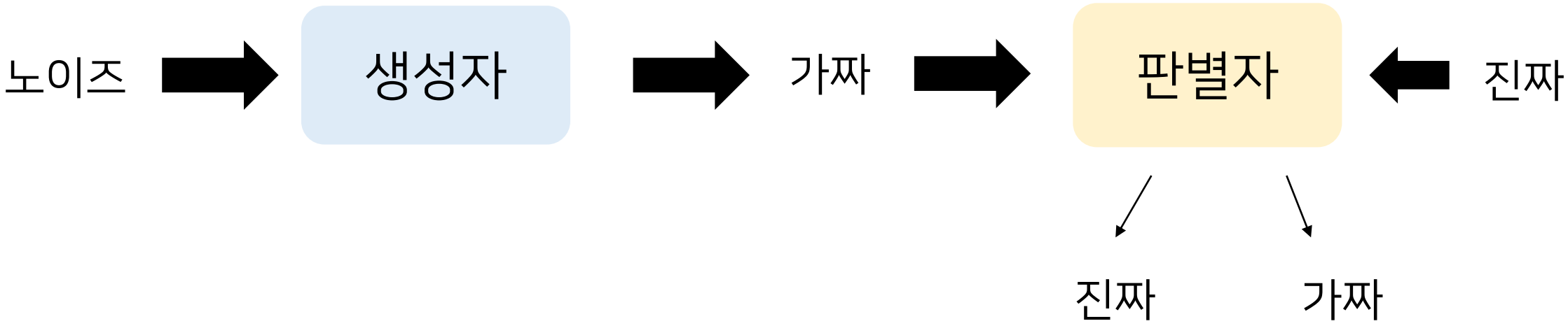
GAN?

Generative Adversarial Network

“딥러닝의 원리를 활용해 가상의 이미지를 생성하는 알고리즘”

생성적 “적대적“ 신경망

“적대적 경합 진행”



DCGAN

컨볼루션 신경망 + GAN

생성자 = 컨볼루션 신경망

CNN	최적화 과정 O, 컴파일 과정 O, 풀링 O
DCGAN	최적화 과정 X, 컴파일 과정 X, 풀링 X, 패딩 O

패딩

1	0	0
0	1	1
1	0	1

+

1	0
0	1

=

2	1
0	2

0	0	0	0	0
0	1	0	0	0
0	0	1	1	0
0	1	0	1	0
0	0	0	0	0

+

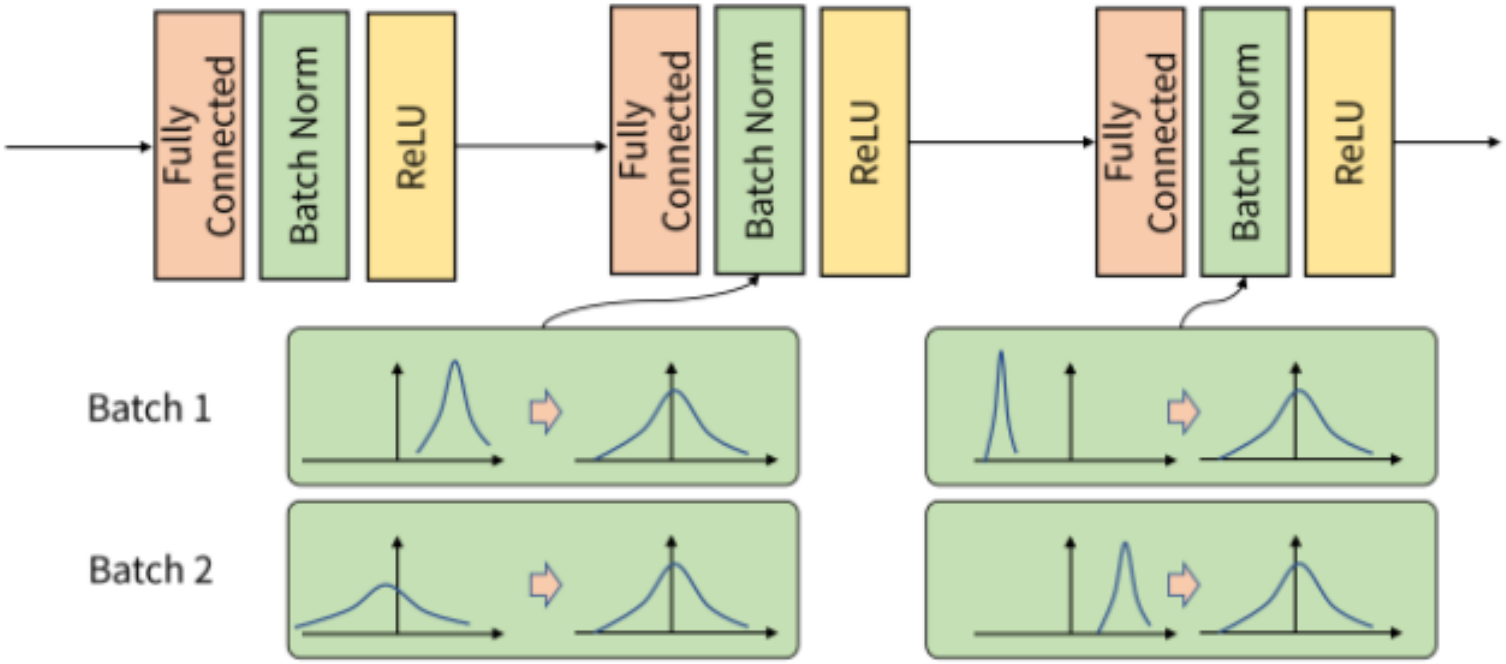
1	1	1
0	0	1
0	0	1

=

1	1	0
2	3	1
1	3	2

배치 정규화

입력데이터 평균 0, 분산 1로 재배치

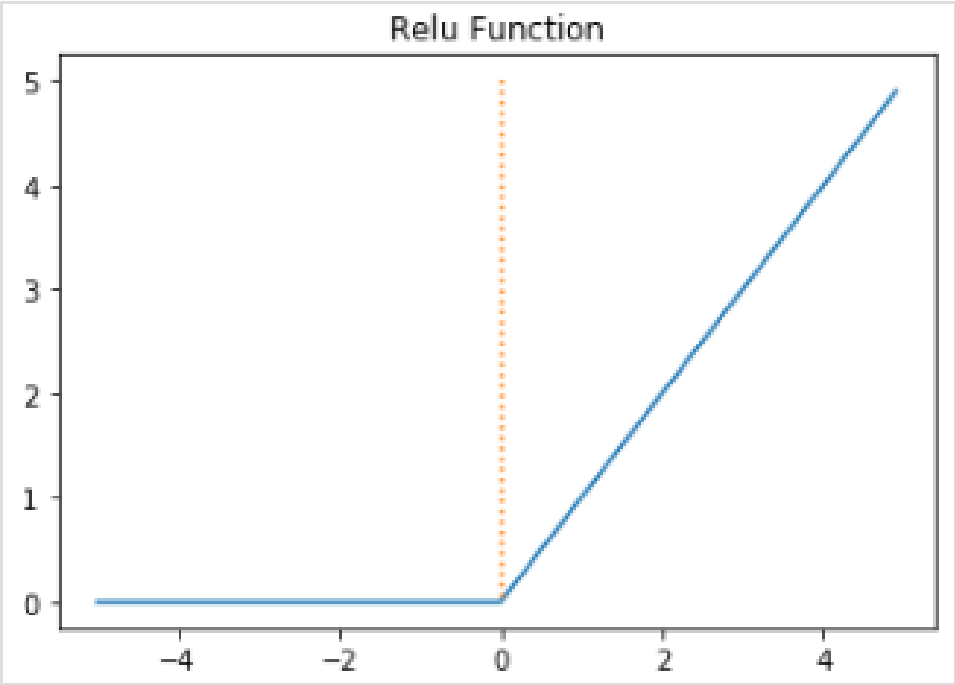


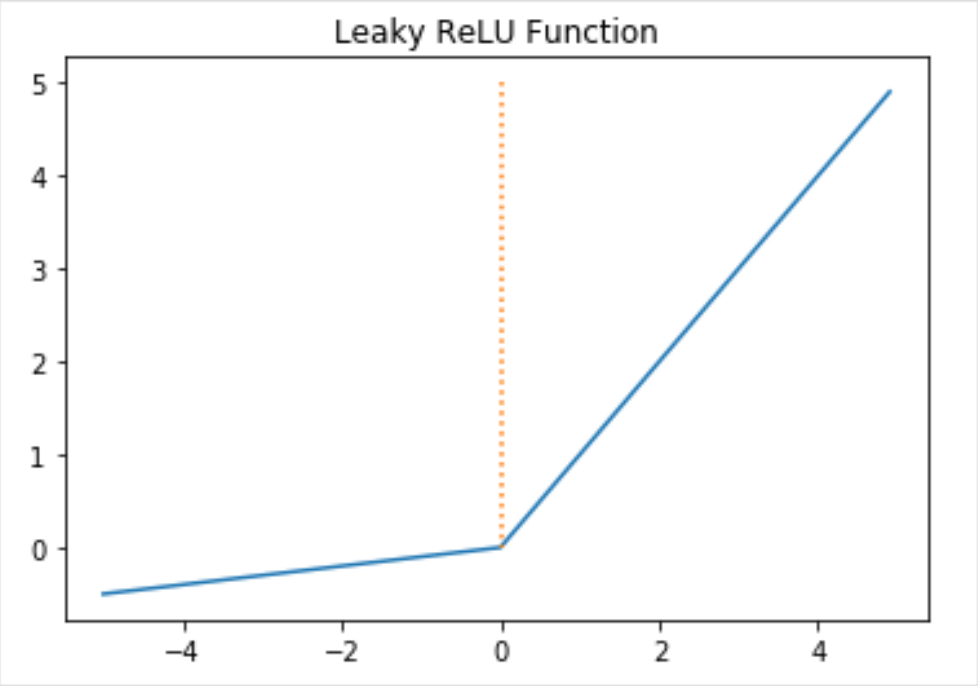
GAN 모델 - 생성자


```
import os
if not os.path.exists("./gan_images"):
    os.makedirs("./gan_images")
```

```
generator = Sequential()
generator.add(Dense(128*7*7, input_dim=100, activation=LeakyReLU(0.2)))
generator.add(BatchNormalization())
generator.add(Reshape((7, 7, 128)))
generator.add(UpSampling2D())
generator.add(Conv2D(64, kernel_size=5, padding='same'))
generator.add(BatchNormalization())
generator.add(Activation(LeakyReLU(0.2)))
generator.add(UpSampling2D())
generator.add(Conv2D(1, kernel_size=5, padding='same', activation='tanh'))
```

Relu / Leaky Relu



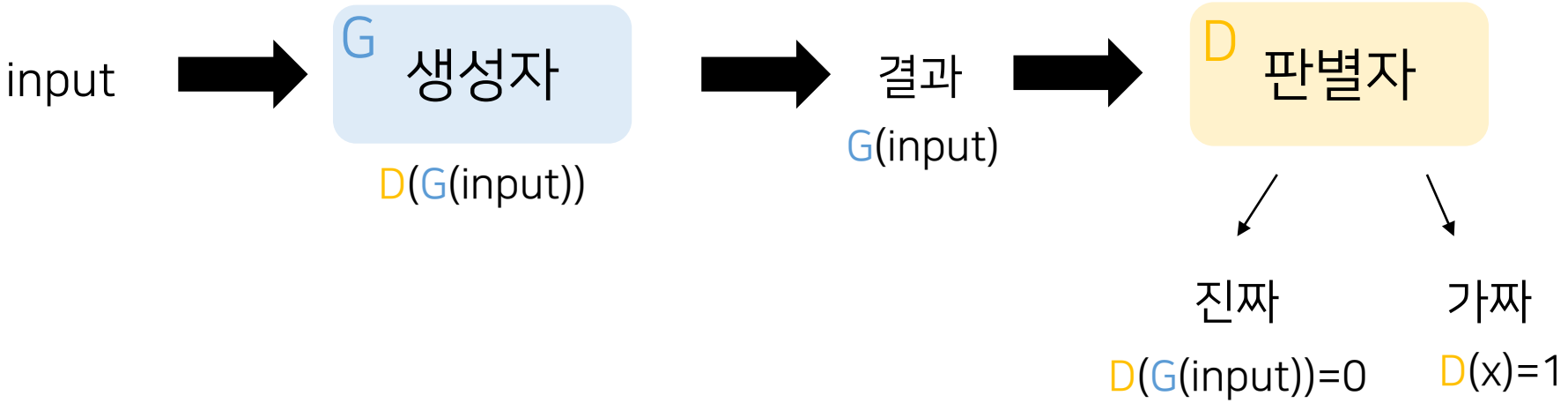


GAN 모델 - 판별자


```
discriminator = Sequential()
discriminator.add(Conv2D(64, kernel_size=5, strides=2, input_shape=(28,28,1), padding="same"))
discriminator.add(Activation(LeakyReLU(0.2)))
discriminator.add(Dropout(0.3))
discriminator.add(Conv2D(128, kernel_size=5, strides=2, padding="same"))
discriminator.add(Activation(LeakyReLU(0.2)))
discriminator.add(Dropout(0.3))
discriminator.add(Flatten())
discriminator.add(Dense(1, activation='sigmoid'))
discriminator.compile(loss='binary_crossentropy', optimizer='adam')
discriminator.trainable = False
```

GAN 모델 - 연결

* 생성자(G), 판별자(D), 실제데이터(x), 입력값(input)




```
ginput = Input(shape=(100,))
dis_output = discriminator(generator(ginput))
gan = Model(ginput, dis_output)
gan.compile(loss='binary_crossentropy', optimizer='adam')
```

GAN 모델 - 실행


```
gan_train(4001, 32)
```

```
def gan_train(epoch, batch_size):


  (X_train, _), (_, _) = mnist.load_data()
  X_train = X_train.reshape(X_train.shape[0], 28, 28, 1).astype('float32')
  X_train = (X_train - 127.5) / 127.5
```

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	3	18	18	18	126	136	175	26	166	255	247	127	0	0	0	0
0	0	0	0	0	0	0	0	30	36	94	154	170	253	253	253	253	253	225	172	253	242	195	64	0	0	0	0
0	0	0	0	0	0	0	49	238	253	253	253	253	253	253	253	253	251	93	82	82	56	39	0	0	0	0	0
0	0	0	0	0	0	0	18	219	253	253	253	253	253	198	182	247	241	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	80	156	107	253	253	205	11	0	43	154	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	14	1	154	253	90	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	139	253	190	2	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	11	190	253	70	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	35	241	225	160	108	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	81	240	253	253	119	25	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	45	186	253	253	150	27	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	93	252	253	187	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	249	253	249	64	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	46	130	183	253	253	207	2	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	39	148	229	253	253	253	250	182	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	24	114	221	253	253	253	253	201	78	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	23	66	213	253	253	253	253	198	81	2	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	18	171	219	253	253	253	253	195	80	9	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	55	172	226	253	253	253	253	244	133	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	136	253	253	253	212	135	132	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

125.5 ÷ 127.5 =

0.9843137254901961

253 - 127.5 =

(x_train) **125.5**

GAN 모델 - 실행


```
true = np.ones((batch_size, 1))
fake = np.zeros((batch_size, 1))

for i in range(epoch):

        idx = np.random.randint(0, X_train.shape[0], batch_size)
        imgs = X_train[idx]
        d_loss_real = discriminator.train_on_batch(imgs, true)


        noise = np.random.normal(0, 1, (batch_size, 100))
        gen_imgs = generator.predict(noise) # G(input)
        d_loss_fake = discriminator.train_on_batch(gen_imgs, fake)
```

```
d_loss = 0.5 * np.add(d_loss_real, d_loss_fake)
g_loss = gan.train_on_batch(noise, true)
```

GAN 모델 - 결과