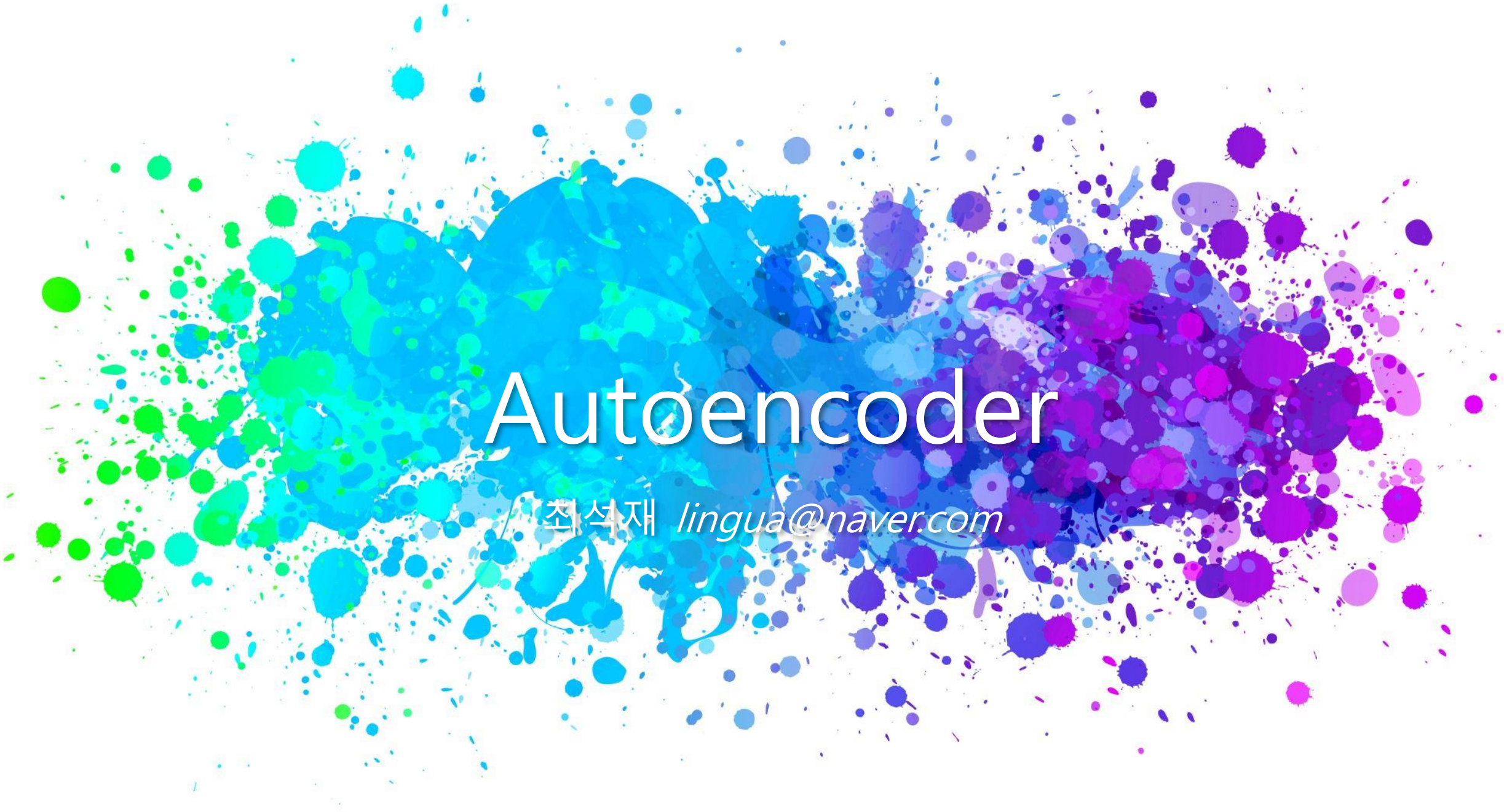


Autoencoder

최석재 *lingua@naver.com*

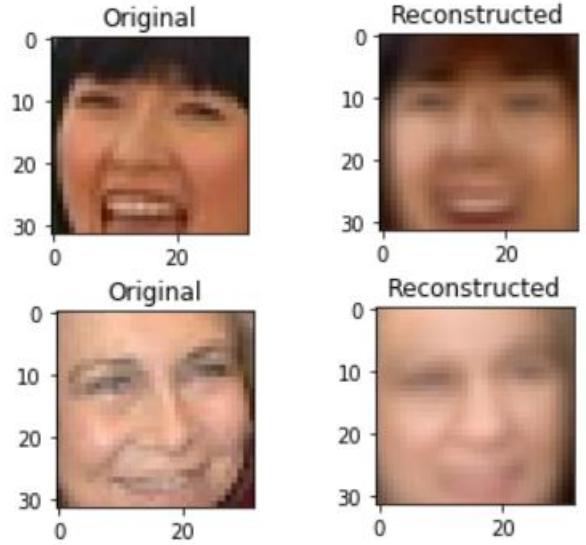
Autoencoder

Auto-Encoder

오토인코더

- GAN은 세상에 존재하지 않는 가상의 이미지를 만듦
- 따라서 GAN은 진짜 같아 보여도 실제로는 존재하지 않는 이미지를 생성

- 오토인코더는 입력 데이터의 특징을 담아낸 이미지를 만듦
- 따라서 만들어낸 것임을 알 수 있으나 원본의 특징이 남아 있는 이미지를 생성



활용분야

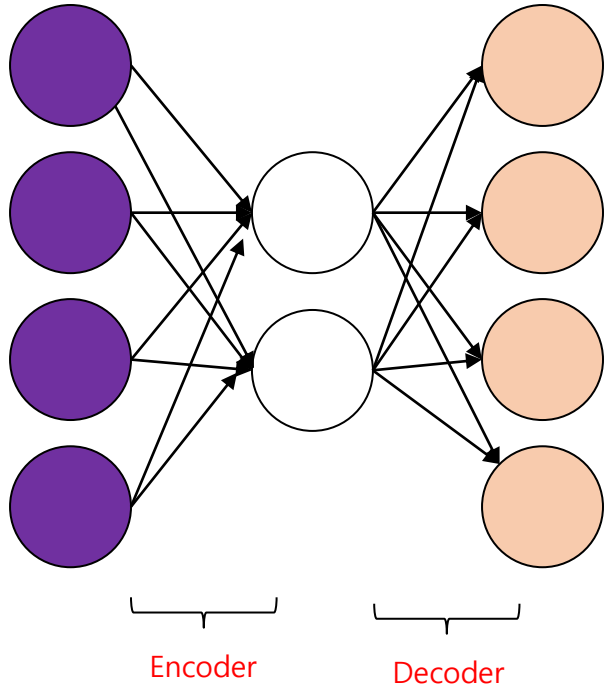
- 오토인코더는 영상, 의학 분야 등 아직 데이터 수가 충분하지 않은 곳에서 활용 가능

- 학습 데이터는 현실 세계의 정보를 담고 있어야 하므로, 완전한 가상의 이미지를 넣으면 잘못된 결과를 낼 수 있음

- 한편 오토인코더는 원본 이미지의 특징을 잘 담아내므로, 부족한 학습 데이터의 수를 올바르게 늘리는 효과를 기대할 수 있음

오토인코더의 원리

- 입력 이미지와 출력 이미지는 동일한 크기게 되게 함
- 중간의 은닉층을 입력층보다 작게 만들어 차원을 줄여줌
- 출력층은 은닉층보다 크고, 입력층과 같으므로 소실된 부분을 복원하기 위해 학습을 시작
- 은닉층에 압축된 입력층의 특징을 기반으로 하여 새로운 이미지가 생성됨



구글 드라이브와 연결

- from google.colab import drive
- drive.mount('/content/gdrive')

관련 패키지 로딩

- from tensorflow.keras.datasets import mnist
- from tensorflow.keras.models import Sequential, Model
- from tensorflow.keras.layers import Input, Dense, Conv2D, MaxPooling2D, UpSampling 2D, Flatten, Reshape
- import os
- import matplotlib.pyplot as plt
- import numpy as np

저장 경로 생성

_generated_images 폴더 밑에 MNIST_AE 라는 폴더를 만든다

- save_path = '/content/gdrive/MyDrive/pytest_img/_generated_images'
- if not os.path.exists(os.path.join(save_path, 'MNIST_AE/')):
    os.makedirs(os.path.join(save_path, "MNIST_AE/"))

데이터 로딩

독립변수 부분만 필요하다

- (X_train, _), (X_test, _) = mnist.load_data()
- X_train = X_train.reshape(X_train.shape[0], 28, 28, 1).astype('float32') / 255
- X_test = X_test.reshape(X_test.shape[0], 28, 28, 1).astype('float32') / 255

인코더

- autoencoder = Sequential()

(3, 3) 사이즈의 마스크 16개 적용
- autoencoder.add(Conv2D(16, kernel_size=3, padding='same', input_shape=(28, 28, 1), activation='relu'))

이미지의 절반을 취하는 MaxPooling. (14, 14, 16)
- autoencoder.add(MaxPooling2D(pool_size=2))

(3, 3) 사이즈의 마스크 8개 적용
- autoencoder.add(Conv2D(8, kernel_size=3, activation='relu', padding='same'))

이미지의 절반을 취하는 MaxPooling. (7, 7, 8)
- autoencoder.add(MaxPooling2D(pool_size=2))

(3, 3) 사이즈의 마스크 8개 적용. 여기서는 strides=2로 형상이 절반으로 줄어듦 (4, 4, 8)
- autoencoder.add(Conv2D(8, kernel_size=3, strides=2, padding='same', activation='relu'))

마지막 Conv2D에 입력되는 값은 (7, 7, 8)로서 strides=2가 제대로 적용될 수 없으므로
Conv2D()에 사용한 padding='same'이 모자라는 자리를 0으로 채워 가능하게 한다

디코더

앞에서 3번의 이미지 축소가 있었으므로 여기서는 UpSampling을 3번한다
가로와 세로의 픽셀수를 2배씩 늘리므로 처음 입력값 28 x 28 로 내보내게 된다
인코더에서와 같이 Conv2D(kernel_size=3)를 같이 이용한다

- autoencoder.add(Conv2D(8, kernel_size=3, padding='same', activation='relu'))
- autoencoder.add(UpSampling2D())
- autoencoder.add(Conv2D(8, kernel_size=3, padding='same', activation='relu'))
- autoencoder.add(UpSampling2D())
- autoencoder.add(Conv2D(16, kernel_size=3, activation='relu')) # padding='same' 하지 않음
- autoencoder.add(UpSampling2D())
- autoencoder.add(Conv2D(1, kernel_size=3, padding='same', activation='sigmoid'))

모델 요약

- autoencoder.summary()

인코더

디코더

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 16)        160

max_pooling2d (MaxPooling2D  (None, 14, 14, 16)        0
)

conv2d_1 (Conv2D)            (None, 14, 14, 8)         1160

max_pooling2d_1 (MaxPooling  (None, 7, 7, 8)           0
2D)

conv2d_2 (Conv2D)            (None, 4, 4, 8)           584

conv2d_3 (Conv2D)            (None, 4, 4, 8)           584

up_sampling2d (UpSampling2D  (None, 8, 8, 8)           0
)

conv2d_4 (Conv2D)            (None, 8, 8, 8)           584

up_sampling2d_1 (UpSampling  (None, 16, 16, 8)         0
2D)

conv2d_5 (Conv2D)            (None, 14, 14, 16)        1168

up_sampling2d_2 (UpSampling  (None, 28, 28, 16)        0
2D)

conv2d_6 (Conv2D)            (None, 28, 28, 1)         145

=================================================================
Total params: 4,385
Trainable params: 4,385
Non-trainable params: 0
_________________________________________________________________
```

컴파일 및 학습

오토인코더에서는 입력되는 내용이 출력되는 내용과 유사하게 나와야 하므로
독립변수와 종속변수가 같다

- autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
- autoencoder.fit(X_train, X_train, epochs=10, batch_size=128, validation_data=(X_test, X_test))

이미지 출력 1

5개의 숫자 선택

- random_test = np.random.randint(X_test.shape[0], size=5)

테스트 데이터를 오토인코더 모델에 넣어 생성된 이미지를 만듦

- ae_imgs = autoencoder.predict(X_test)

출력 이미지 size 정하기 (가로, 세로)

- plt.figure(figsize=(5, 2))

이미지 출력 2

- fig, axs = plt.subplots(2, 5) # 2행 5열로 출력
- count = 0

- for k in range(5):

```
    axs[0, k].imshow(X_test[count].reshape(28, 28))
    axs[0, k].axis('off')
```

윗쪽 행에는 원본 이미지 출력
1채널을 가지고 있던 X_test[0]의 (28, 28, 1)의 형상을 단순하게 (28, 28)로 바꿈

```
    axs[1, k].imshow(ae_imgs[count].reshape(28, 28))
    axs[1, k].axis('off')
    count += 1
    fig.savefig(os.path.join(save_path, "MNIST_AE/")+"ae_mninst.png")
```

출력 결과

- 위쪽이 원본, 아래쪽이 생성된 이미지
- 10 epochs만으로도 좋은 결과가 나왔다