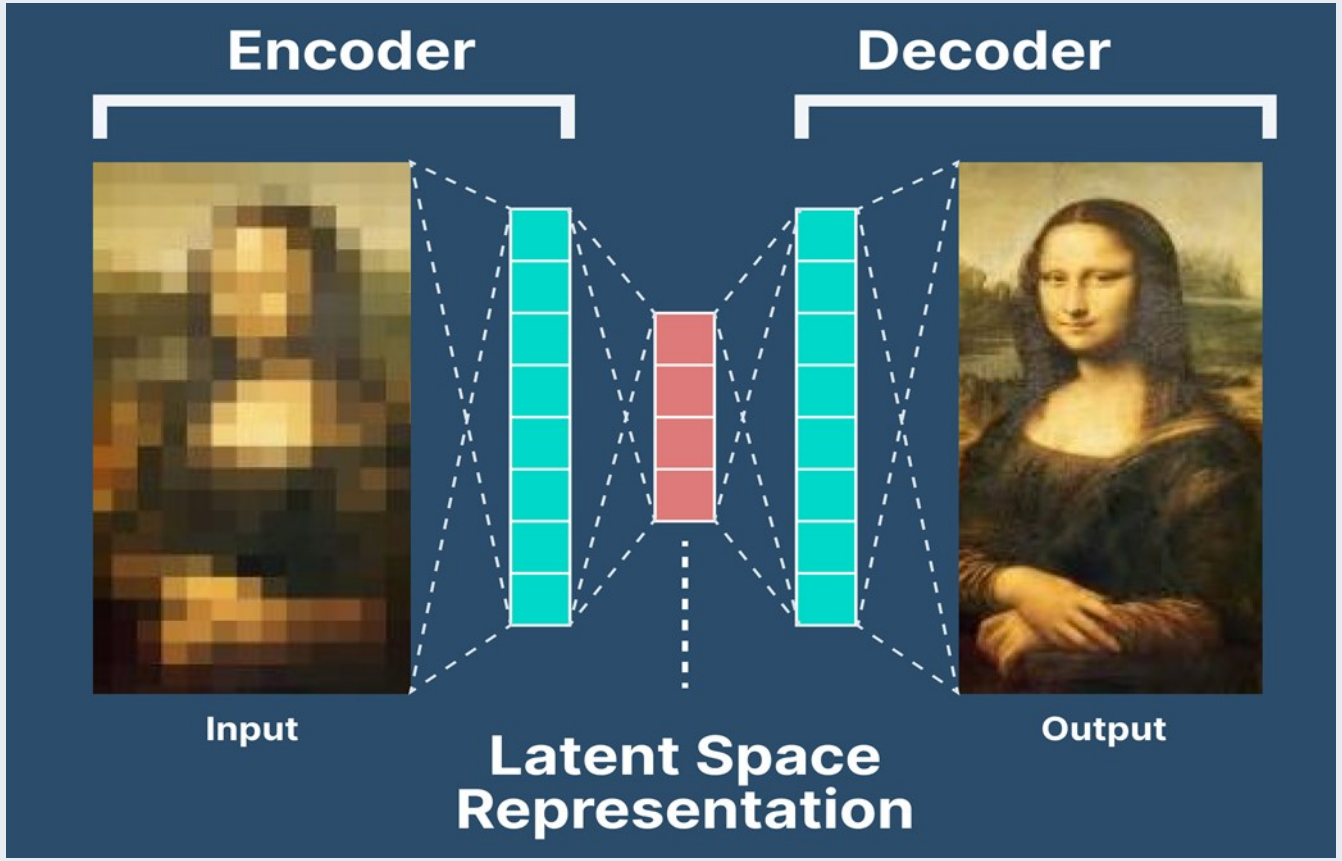
오토인코더(Auto Encoder)



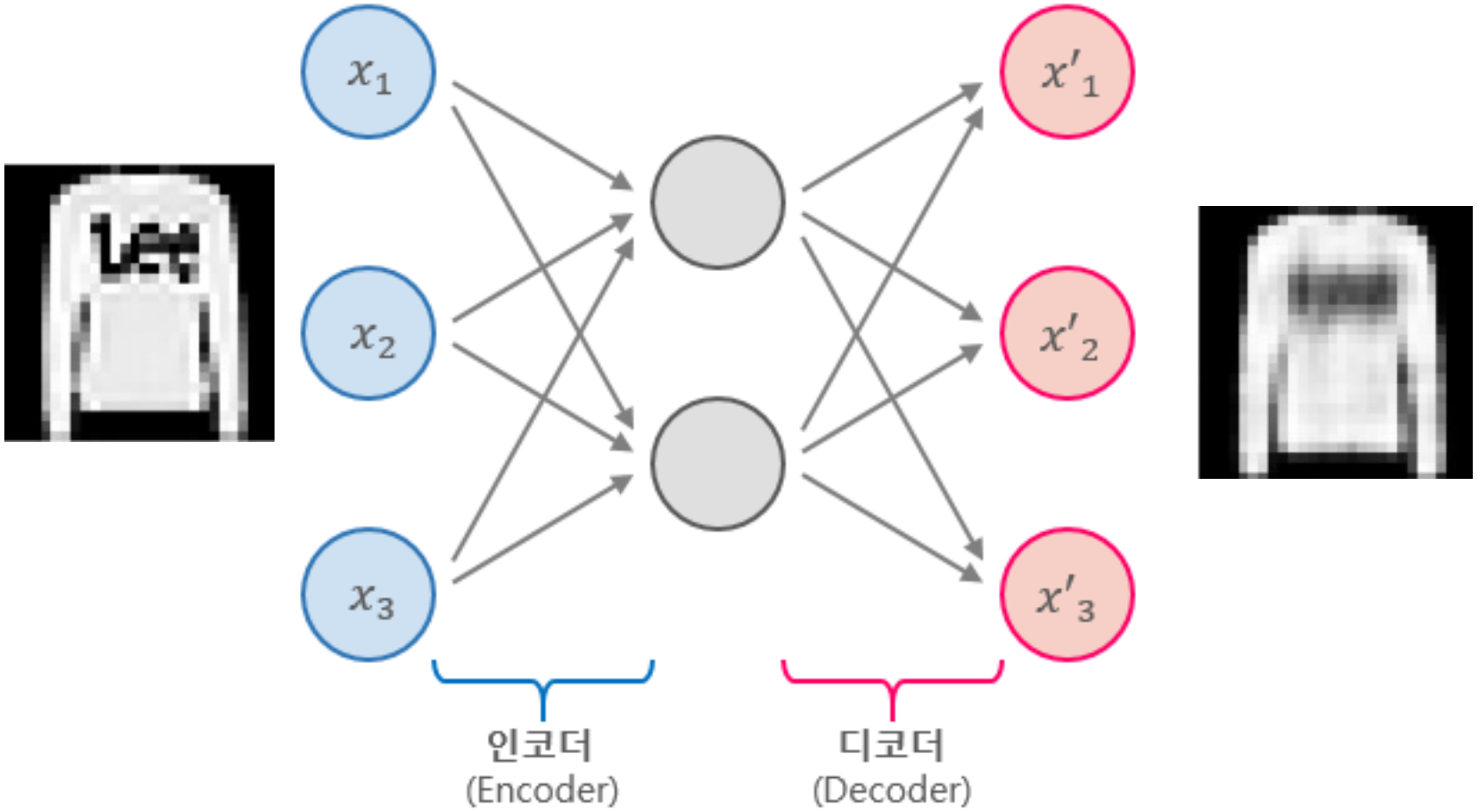
박경규

오토인코더(Auto Encoder)

오토인코더는 원본 이미지 벡터의 차원을 축소하여 은닉 레이어에 주요 특징을 추출합니다.
그리고, 은닉 레이어에 추출된 핵심 정보만을 학습하여 원본 이미지와 비슷하게 복원한 이미지를 출력합니다.

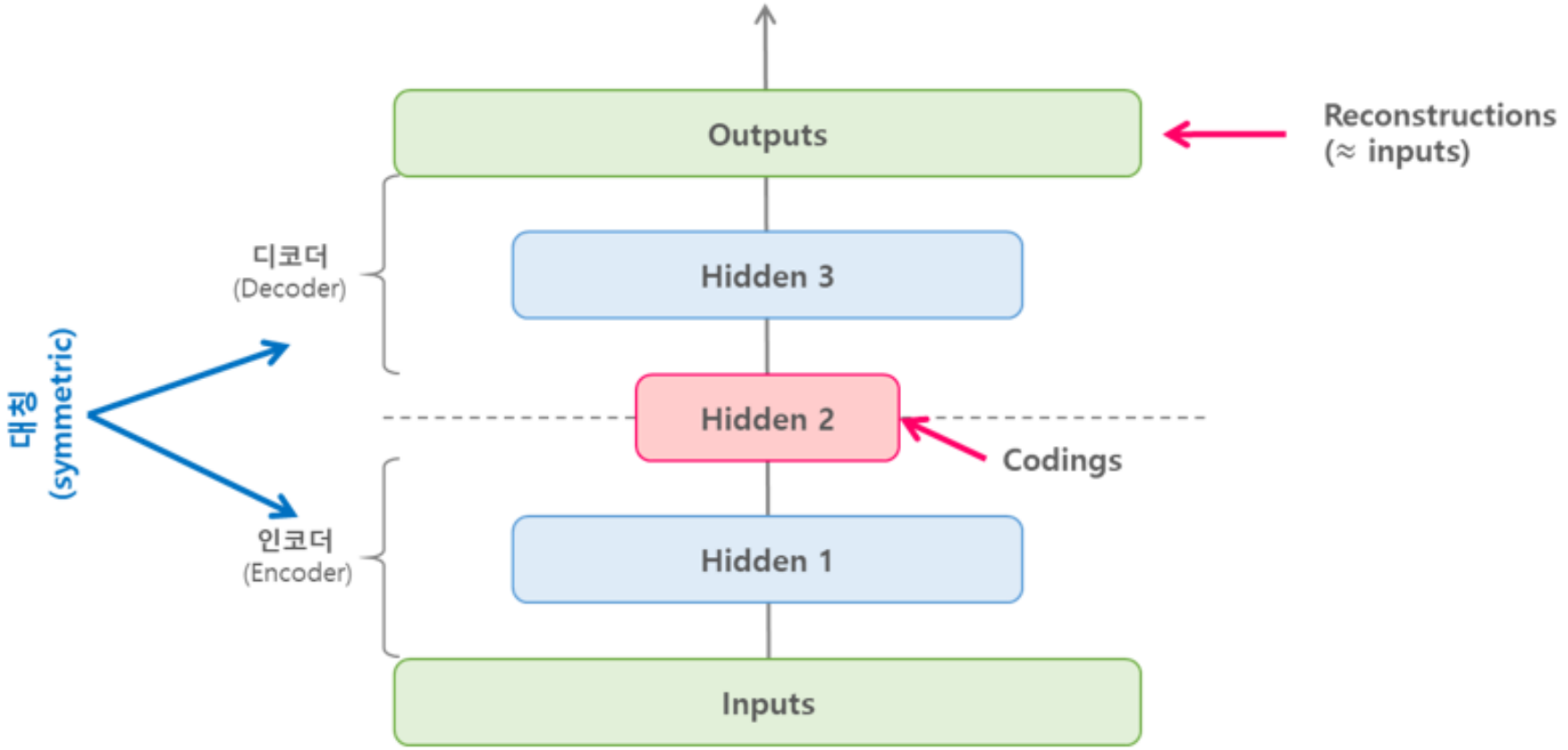
AutoEncoder

- Unsupervised Learning
- Representation Learning
- Dimensionality Reduction
- Generative Model Learning



Stacked AutoEncoder

**Stacked AutoEncoder는 여러개의 히든 레이어를 가지는 Auto Encoder이며,
레이어를 추가할수록 AutoEncoder가 더 복잡한 코딩을 학습할 수 있습니다.**



기본 오토인코더

두 개의 Dense 레이어로 autoencoder를 정의합니다. 이미지를 64차원 잠재 벡터로 압축하는 encoder와 잠재 공간에서 원본 이미지를 재구성하는 decoder입니다.

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import tensorflow as tf

from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.model_selection import train_test_split
from tensorflow.keras import layers, losses
from tensorflow.keras.datasets import fashion_mnist
from tensorflow.keras.models import Model
```

```
(x_train, _), (x_test, _) = fashion_mnist.load_data()

x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.

print (x_train.shape)
print (x_test.shape)
```

기본 오토인코더

```
latent_dim = 64

class Autoencoder(Model):
  def __init__(self, encoding_dim):
    super(Autoencoder, self).__init__()
    self.latent_dim = latent_dim
    self.encoder = tf.keras.Sequential([
      layers.Flatten(),
      layers.Dense(latent_dim, activation='relu'),
    ])
    self.decoder = tf.keras.Sequential([
      layers.Dense(784, activation='sigmoid'),
      layers.Reshape((28, 28))
    ])

  def call(self, x):
    encoded = self.encoder(x)
    decoded = self.decoder(encoded)
    return decoded

autoencoder = Autoencoder(latent_dim)
```

기본 오토인코더

```
autoencoder.compile(optimizer='adam', loss=losses.MeanSquaredError())
```

`x_train`을 입력과 대상으로 사용하여 모델을 훈련합니다.

`encoder`는 데이터세트를 784차원에서 잠재 공간으로 압축하는 방법을 배우고,

`decoder`는 원본 이미지를 재구성하는 방법을 배웁니다..

```
autoencoder.fit(x_train, x_train,
                epochs=10,
                shuffle=True,
                validation_data=(x_test, x_test))
```

```
Epoch 1/10
1875/1875 [==============================] - 7s 2ms/step - loss: 0.0236 - val_loss: 0.0130
Epoch 2/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.0114 - val_loss: 0.0104
Epoch 3/10
```

기본 오토인코더

```
encoded_imgs = autoencoder.encoder(x_test).numpy()
decoded_imgs = autoencoder.decoder(encoded_imgs).numpy()
```

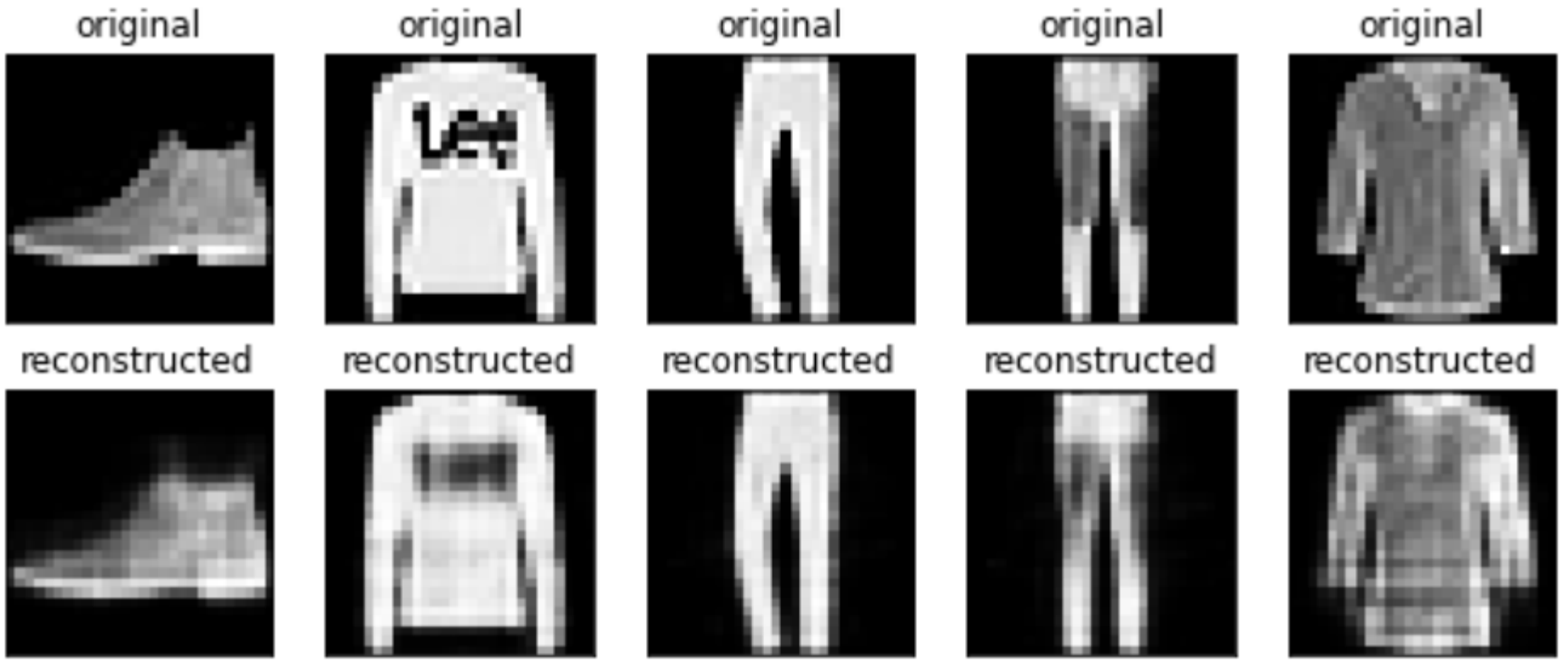
```
n = 10
plt.figure(figsize=(20, 4))
for i in range(n):
  # display original
  ax = plt.subplot(2, n, i + 1)
  plt.imshow(x_test[i])
  plt.title("original")
  plt.gray()
  ax.get_xaxis().set_visible(False)
  ax.get_yaxis().set_visible(False)

  # display reconstruction
  ax = plt.subplot(2, n, i + 1 + n)
  plt.imshow(decoded_imgs[i])
  plt.title("reconstructed")
  plt.gray()
  ax.get_xaxis().set_visible(False)
  ax.get_yaxis().set_visible(False)
plt.show()
```

기본 오토인코더

```
images = model.predict(X_test)
```

```
num = 5
plt.figure(figsize=(20, 8))
for i in range(num):
    # 원본 이미지
    ax = plt.subplot(2, num, i + 1)
    plt.imshow(X_test[i].reshape((28, 28)), cmap='gray')
    plt.title("Original %s" % str(i))
    plt.axis('off')
    # 복원 이미지
    ax = plt.subplot(2, num, i + num + 1)
    plt.imshow(images[i], cmap='gray')
    plt.title("Auto-encoded %s" % str(i))
    plt.axis('off')
plt.show()
```

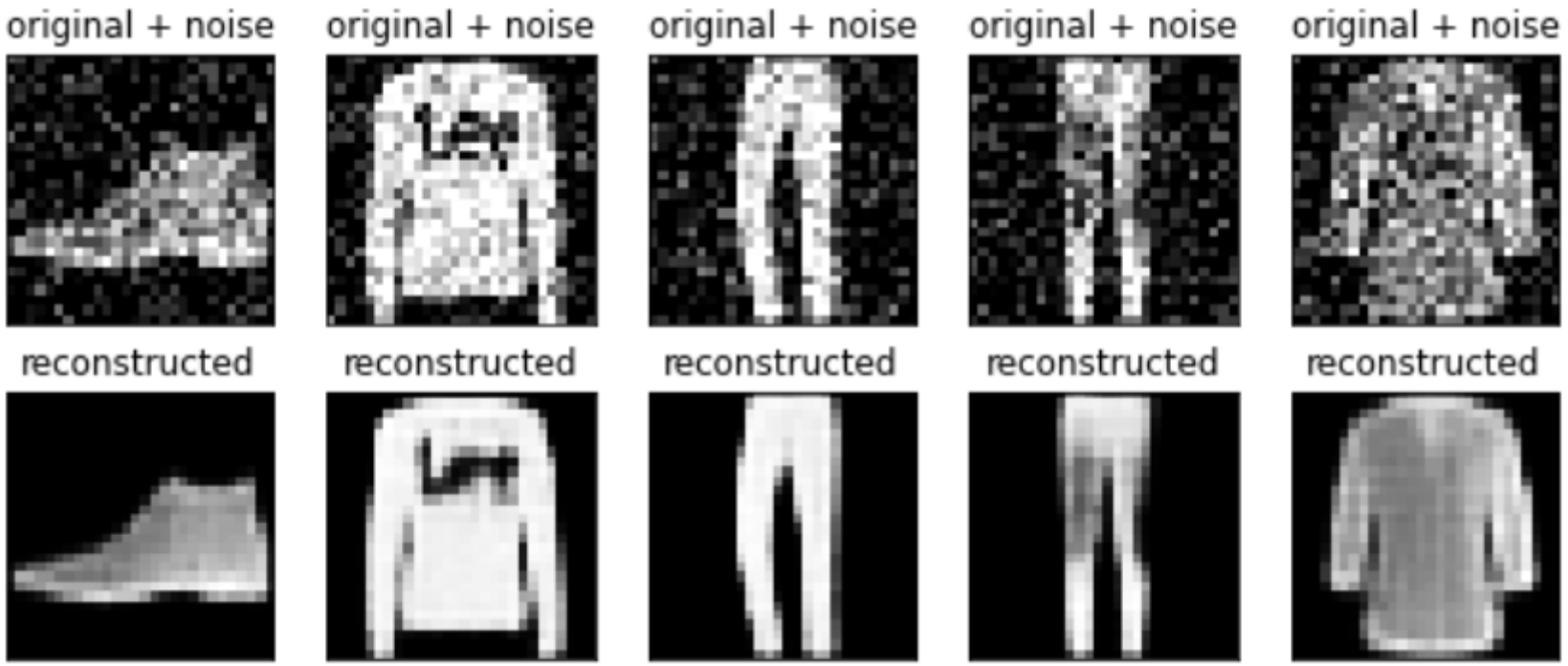
기본 오토인코더


original
original
original
original
original
reconstructed
reconstructed
reconstructed
reconstructed
reconstructed

이미지 노이즈 제거

```
class Denoise(Model):
  def __init__(self):
    super(Denoise, self).__init__()
    self.encoder = tf.keras.Sequential([
      layers.Input(shape=(28, 28, 1)),
      layers.Conv2D(16, (3,3), activation='relu', padding='same', strides=2),
      layers.Conv2D(8, (3,3), activation='relu', padding='same', strides=2)])

    self.decoder = tf.keras.Sequential([
      layers.Conv2DTranspose(8, kernel_size=3, strides=2, activation='relu', padding='same'),
      layers.Conv2DTranspose(16, kernel_size=3, strides=2, activation='relu', padding='same'),
      layers.Conv2D(1, kernel_size=(3,3), activation='sigmoid', padding='same')])

  def call(self, x):
    encoded = self.encoder(x)
    decoded = self.decoder(encoded)
    return decoded
```

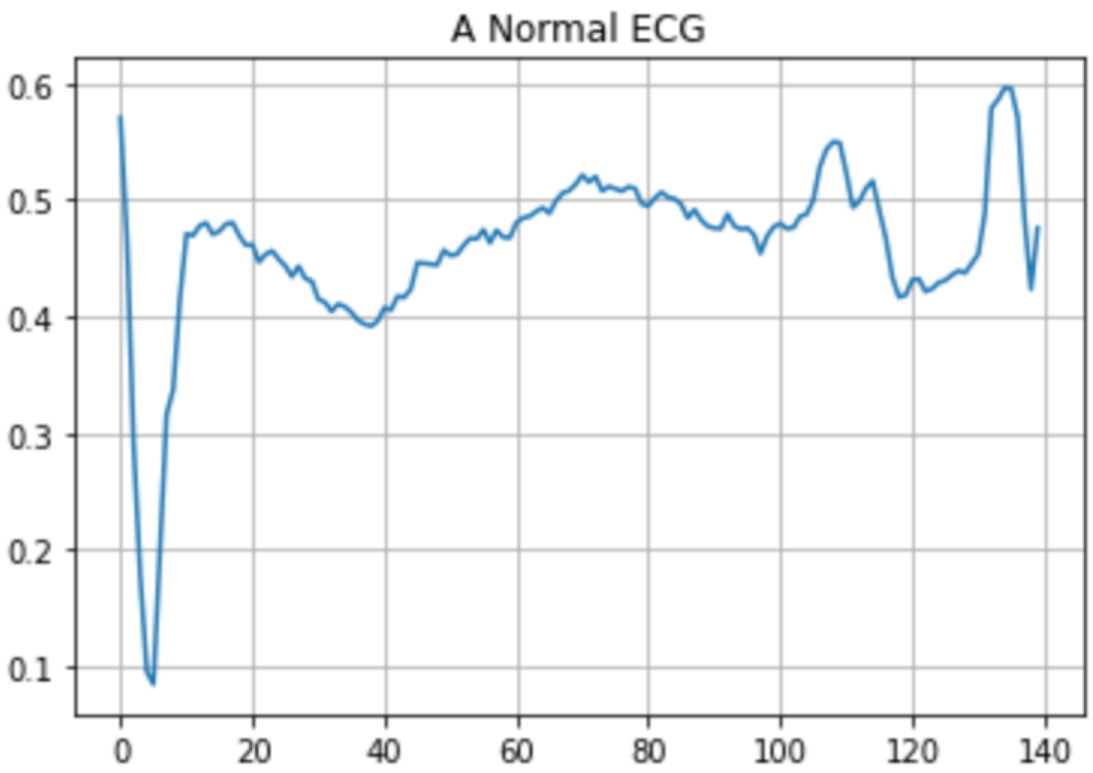
이미지 노이즈 제거

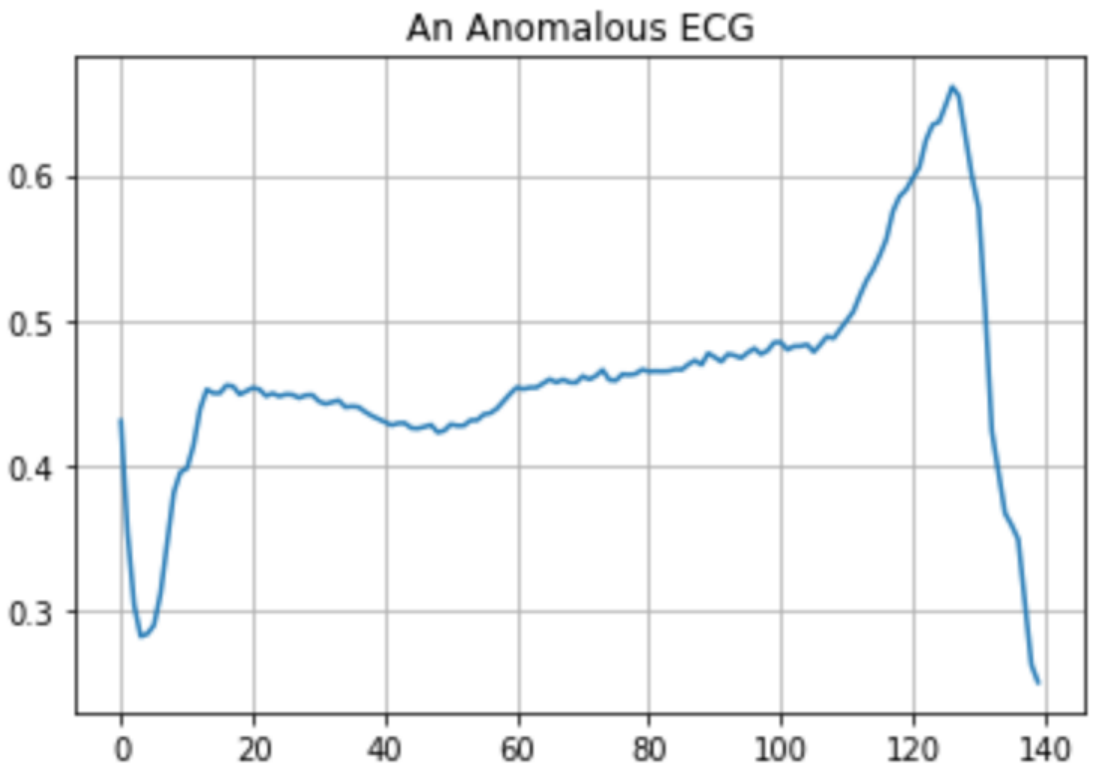

original + noise
original + noise
original + noise
original + noise
original + noise
reconstructed
reconstructed
reconstructed
reconstructed
reconstructed

이상치 탐지

```
class AnomalyDetector(Model):
  def __init__(self):
    super(AnomalyDetector, self).__init__()
    self.encoder = tf.keras.Sequential([
      layers.Dense(32, activation="relu"),
      layers.Dense(16, activation="relu"),
      layers.Dense(8, activation="relu")])

    self.decoder = tf.keras.Sequential([
      layers.Dense(16, activation="relu"),
      layers.Dense(32, activation="relu"),
      layers.Dense(140, activation="sigmoid")])

  def call(self, x):
    encoded = self.encoder(x)
    decoded = self.decoder(encoded)
    return decoded
```

이상치 탐지


A Normal ECG
0.6
0.5
0.4
0.3
0.2
0.1
0
20
40
60
80
100
120
140

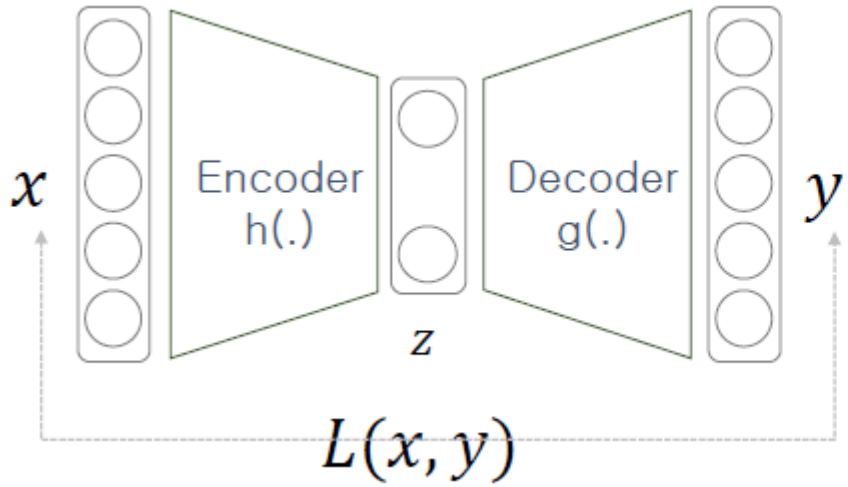

An Anomalous ECG
0.6
0.5
0.4
0.3
0
20
40
60
80
100
120
140

오토인코더 구현 실습

https://www.tensorflow.org/tutorials/generative/autoencoder?hl=ko

Further Study

https://deepinsight.tistory.com/126



$$z = h(x) \in \mathbb{R}^{d_z}$$

$$y = g(z) = g(h(x))$$

$$L_{AE} = \sum_{x \in D} L(x, y)$$

- Make output layer same size as input layer

$$x, y \in \mathbb{R}^d$$

- Loss encourages output to be close to input

$$L(x, y)$$

입출력이 동일한 네트워크

- Unsupervised Learning → Supervised Learning

비교사 학습 문제를 교사 학습 문제로 바꾸어서 해결

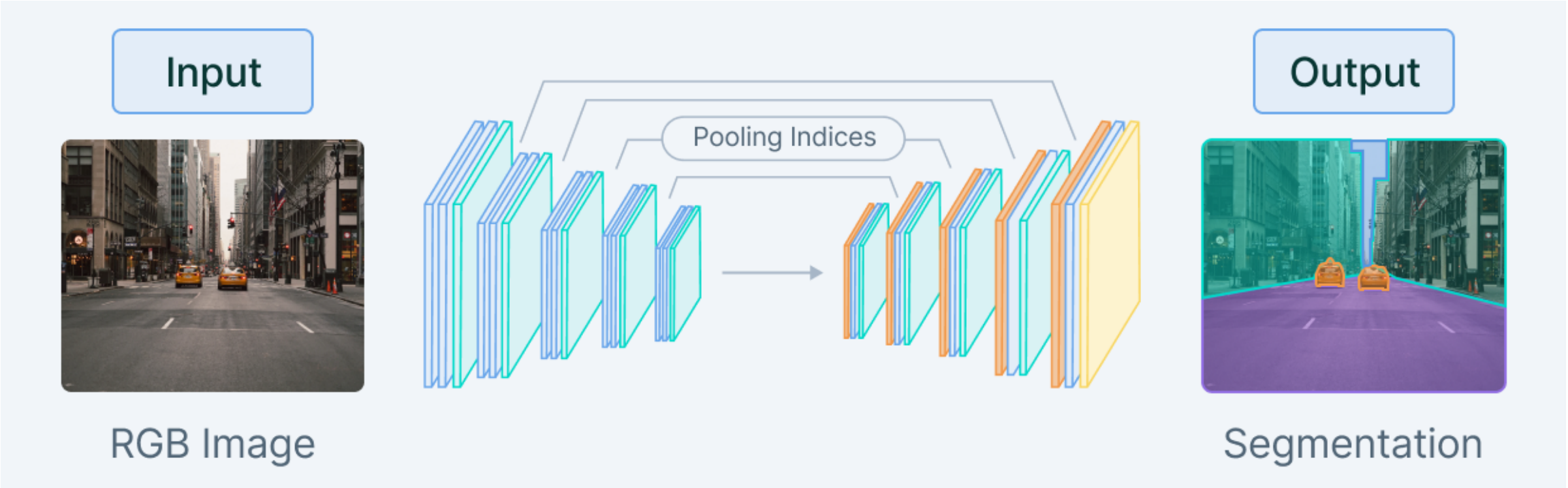
Decoder가 최소한 학습 데이터는 생성해 낼 수 있게 된다.
→ 생성된 데이터가 학습 데이터 좀 닮아 있다.

Encoder가 최소한 학습 데이터는 잘 latent vector로 표현할 수 있게 된다.
→ 데이터의 추상화를 위해 많이 사용된다.

http://videolectures.net/deeplearning2015_vincent_autoencoders/?q=vincent%20autoencoder

Further Study

https://www.v7labs.com/blog/autoencoders-guide



THANK YOU

kgpark88@gmail.com