CNN으로 MNIST 데이터 분류

IT융합공학부 윤세영

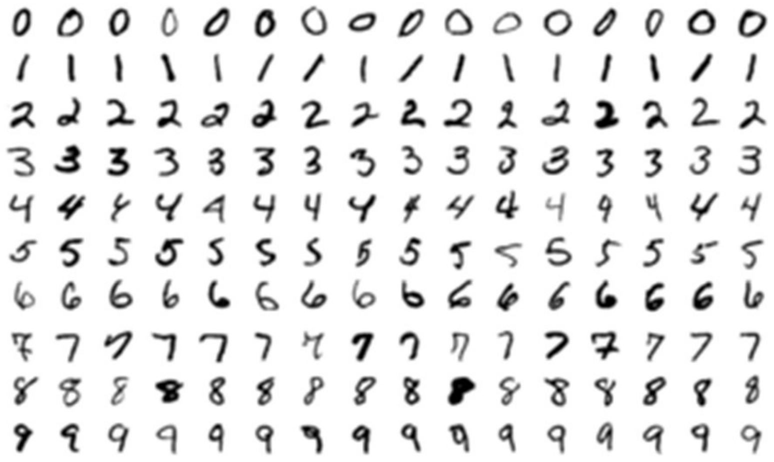
유투브 주소: https://youtu.be/x3FfdF7aAnU

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB

MNIST란?

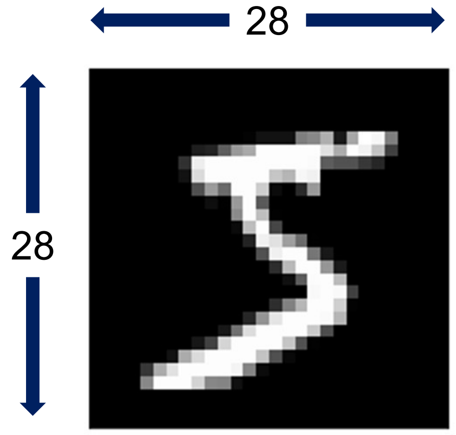
- 미국 우체국에서 기계로 우편번호를 자동으로 구분하기 위해 만든 손글씨 표



MNIST Dataset

MNIST란?

- Train set: 60000, Test set: 10000
- 28x28x1 이미지로 되어있음
- 픽셀: 28x28 (2D) → 784 (1D)
- 이미지: 흑백 (0 ~ 255)



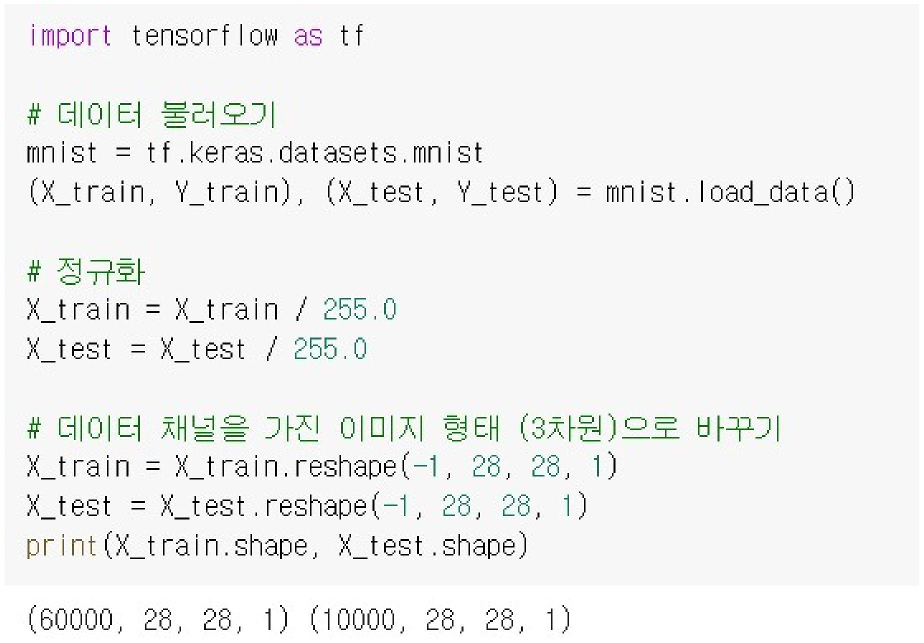
28 x 28 x 1 image

진행 순서

0. 데이터 불러오기 및 정리
1. CNN 모델 구성
2. FIT
3. 그래프로 보기
4. 테스트
5. 데이터 예측
6. 틀린 데이터 추출

0. 데이터 불러오기 및 정리

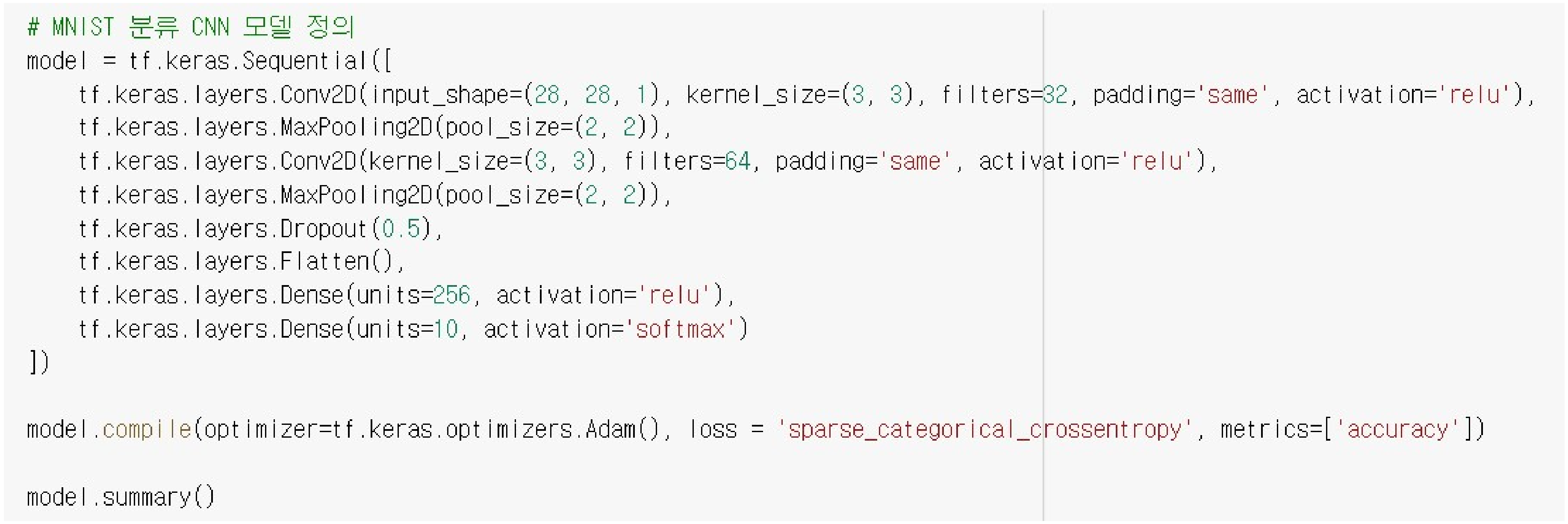
https://hmkim312.github.io/posts/MNIST_%EB%8D%B0%EC%9D%B4%ED%84%B0%EB%A1%9C_%ED%95%B4%EB%B3%B4%EB%8A%94_CNN(Convolution_Neral_Network)/



```
import tensorflow as tf

# 데이터 불러오기
mnist = tf.keras.datasets.mnist
(X_train, Y_train), (X_test, Y_test) = mnist.load_data()

# 정규화
X_train = X_train / 255.0
X_test = X_test / 255.0

# 데이터 채널을 가진 이미지 형태 (3차원)으로 바꾸기
X_train = X_train.reshape(-1, 28, 28, 1)
X_test = X_test.reshape(-1, 28, 28, 1)
print(X_train.shape, X_test.shape)
```

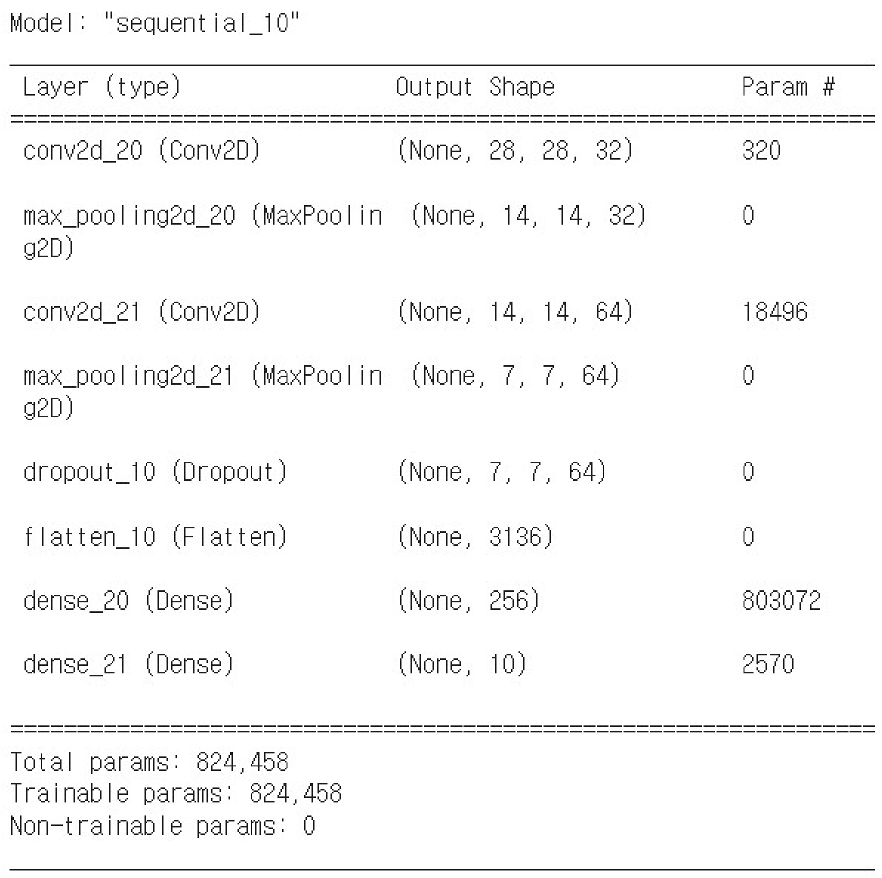
```
(60000, 28, 28, 1) (10000, 28, 28, 1)
```

1. CNN 모델 구성



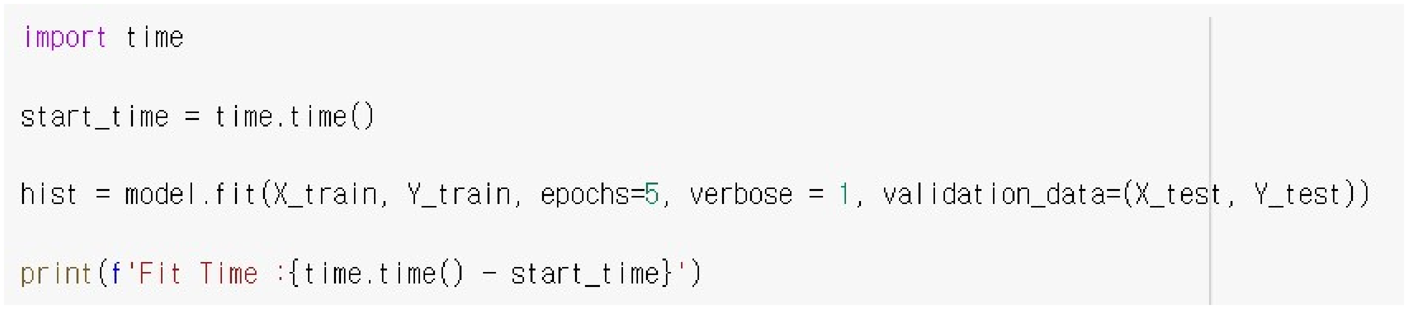
```
# MNIST 분류 CNN 모델 정의
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(input_shape=(28, 28, 1), kernel_size=(3, 3), filters=32, padding='same', activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
    tf.keras.layers.Conv2D(kernel_size=(3, 3), filters=64, padding='same', activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(units=256, activation='relu'),
    tf.keras.layers.Dense(units=10, activation='softmax')
])

model.compile(optimizer=tf.keras.optimizers.Adam(), loss = 'sparse_categorical_crossentropy', metrics=['accuracy'])

model.summary()
```

1. CNN 모델 구성

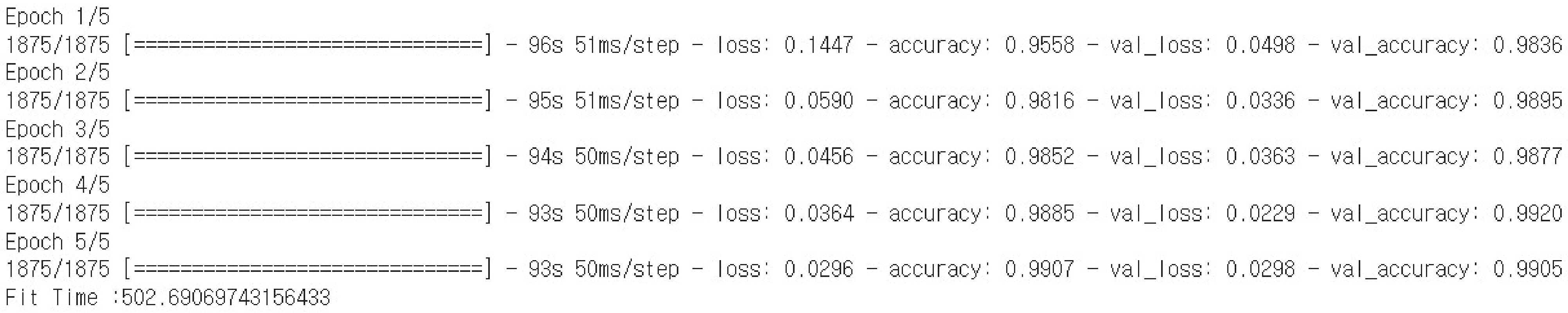


```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_20 (Conv2D)          (None, 28, 28, 32)        320

 max_pooling2d_20 (MaxPoolin  (None, 14, 14, 32)       0
 g2D)

 conv2d_21 (Conv2D)          (None, 14, 14, 64)        18496

 max_pooling2d_21 (MaxPoolin  (None, 7, 7, 64)         0
 g2D)

 dropout_10 (Dropout)        (None, 7, 7, 64)          0

 flatten_10 (Flatten)        (None, 3136)              0

 dense_20 (Dense)            (None, 256)               803072

 dense_21 (Dense)            (None, 10)                2570

=================================================================
Total params: 824,458
Trainable params: 824,458
Non-trainable params: 0
_________________________________________________________________
```

2. FIT

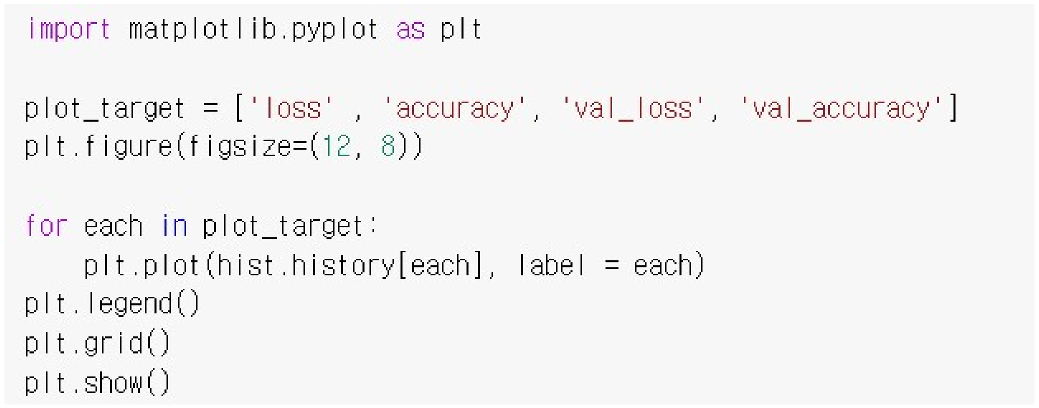


```
import time

start_time = time.time()

hist = model.fit(X_train, Y_train, epochs=5, verbose = 1, validation_data=(X_test, Y_test))

print(f'Fit Time :{time.time() - start_time}')
```



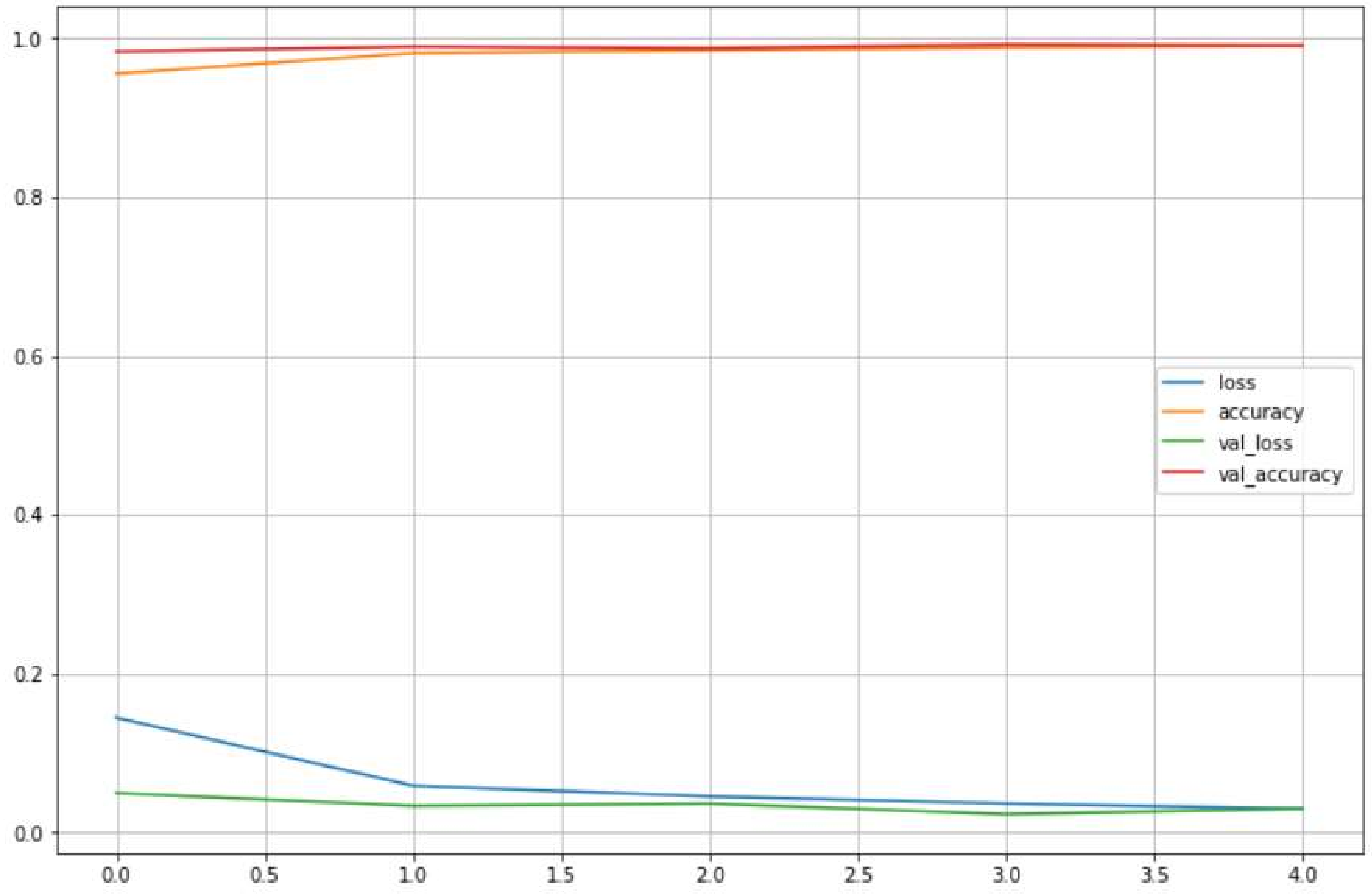
```
Epoch 1/5
1875/1875 [==============================] - 96s 51ms/step - loss: 0.1447 - accuracy: 0.9558 - val_loss: 0.0498 - val_accuracy: 0.9836
Epoch 2/5
1875/1875 [==============================] - 95s 51ms/step - loss: 0.0590 - accuracy: 0.9816 - val_loss: 0.0336 - val_accuracy: 0.9895
Epoch 3/5
1875/1875 [==============================] - 94s 50ms/step - loss: 0.0456 - accuracy: 0.9852 - val_loss: 0.0363 - val_accuracy: 0.9877
Epoch 4/5
1875/1875 [==============================] - 93s 50ms/step - loss: 0.0364 - accuracy: 0.9885 - val_loss: 0.0229 - val_accuracy: 0.9920
Epoch 5/5
1875/1875 [==============================] - 93s 50ms/step - loss: 0.0296 - accuracy: 0.9907 - val_loss: 0.0298 - val_accuracy: 0.9905
Fit Time :502.69069743156433
```

3. 그래프로 보기



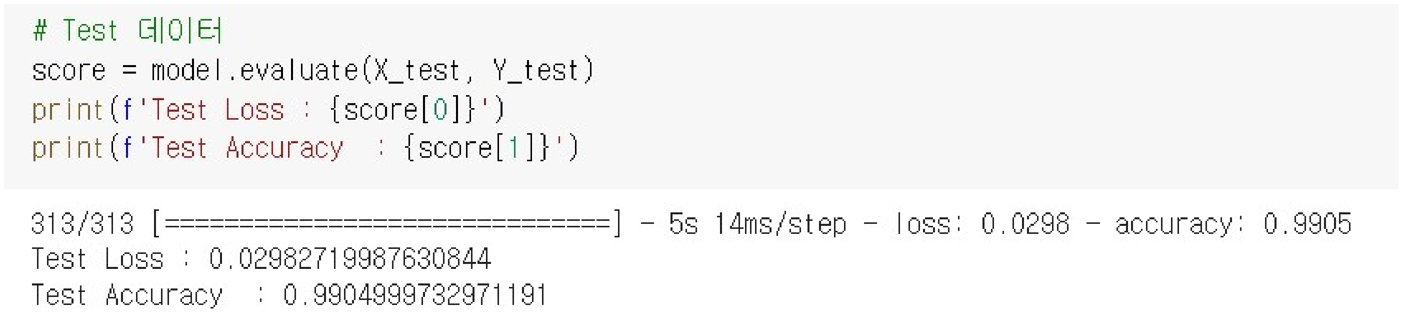
```python
import matplotlib.pyplot as plt

plot_target = ['loss' , 'accuracy', 'val_loss', 'val_accuracy']
plt.figure(figsize=(12, 8))

for each in plot_target:
    plt.plot(hist.history[each], label = each)
plt.legend()
plt.grid()
plt.show()
```

3. 그래프로 보기



1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
loss
accuracy
val_loss
val_accuracy

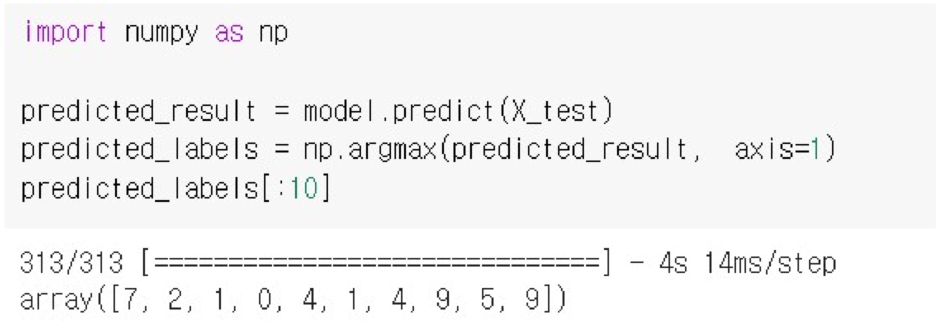
4. 테스트



```
# Test 데이터
score = model.evaluate(X_test, Y_test)
print(f'Test Loss : {score[0]}')
print(f'Test Accuracy  : {score[1]}')
```

```
313/313 [==============================] - 5s 14ms/step - loss: 0.0298 - accuracy: 0.9905
Test Loss : 0.029827199876308441
Test Accuracy  : 0.9904999732971191
```

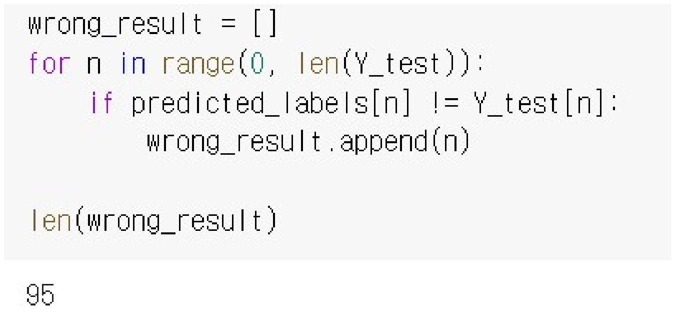
5. 데이터 예측



```python
import numpy as np

predicted_result = model.predict(X_test)
predicted_labels = np.argmax(predicted_result,  axis=1)
predicted_labels[:10]
```

```
313/313 [==============================] - 4s 14ms/step
array([7, 2, 1, 0, 4, 1, 4, 9, 5, 9])
```

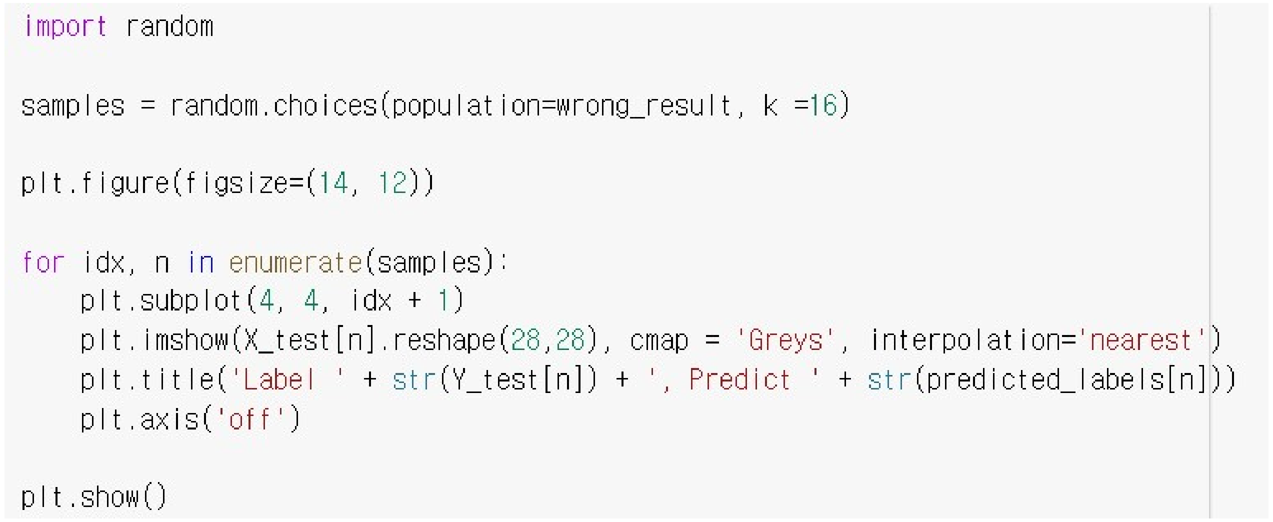
5. 틀린 데이터 추출



```
wrong_result = []
for n in range(0, len(Y_test)):
    if predicted_labels[n] != Y_test[n]:
        wrong_result.append(n)

len(wrong_result)
```

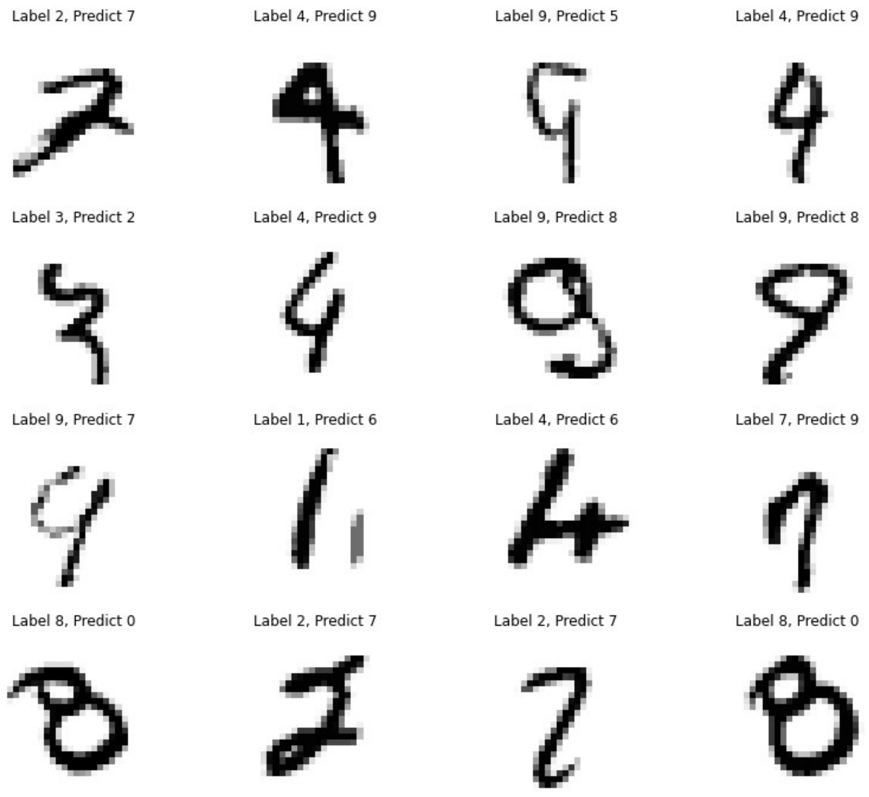
```
95
```

5. 틀린 데이터 추출



```python
import random

samples = random.choices(population=wrong_result, k =16)

plt.figure(figsize=(14, 12))

for idx, n in enumerate(samples):
    plt.subplot(4, 4, idx + 1)
    plt.imshow(X_test[n].reshape(28,28), cmap = 'Greys', interpolation='nearest')
    plt.title('Label ' + str(Y_test[n]) + ', Predict ' + str(predicted_labels[n]))
    plt.axis('off')

plt.show()
```

5. 틀린 데이터 추출



Label 2, Predict 7
Label 4, Predict 9
Label 9, Predict 5
Label 4, Predict 9
Label 3, Predict 2
Label 4, Predict 9
Label 9, Predict 8
Label 9, Predict 8
Label 9, Predict 7
Label 1, Predict 6
Label 4, Predict 6
Label 7, Predict 9
Label 8, Predict 0
Label 2, Predict 7
Label 2, Predict 7
Label 8, Predict 0

Q & A