MNIST - Categorical Classification

> ## Overfitting Issue

Import Tensorflow

```
import warnings
warnings.filterwarnings('ignore')
```

- import TensorFlow

```
import tensorflow as tf

tf.__version__
```

```
'2.6.0'
```

- GPU 설정 확인

```
tf.test.gpu_device_name()
```

```
'/device:GPU:0'
```

I. MNIST Data_Set Load & Review

1) Load MNIST Data_Set

```
from tensorflow.keras.datasets import mnist

(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
11501568/11490434 [==============================] - 0s 0us/step
```

- Train_Data Information

```
print(len(X_train))
print(X_train.shape)

print(len(y_train))
print(y_train[0:5])
```

```
60000
(60000, 28, 28)
60000
[5 0 4 1 9]
```

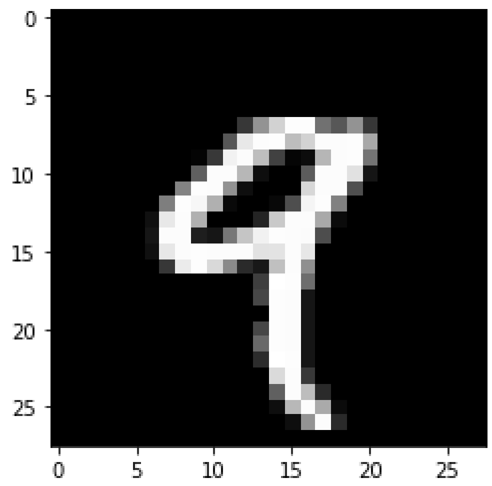
- Test_Data Information

```
print(len(X_test))
print(X_test.shape)

print(len(y_test))
print(y_test[0:5])
```

```
10000
(10000, 28, 28)
10000
[7 2 1 0 4]
```

2) Visualization

```
import matplotlib.pyplot as plt

digit = X_train[4]
plt.imshow(digit, cmap = 'gray')
plt.show()
```



```
import numpy as np
np.set_printoptions(linewidth = 150)

print(X_train[4])
```

```
[[  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   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  55 148 210 253 253 113  87 148  55   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  87 232 252 253 189 210 252 252 253 168   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   4  57 242 252 190  65   5  12 182 252 253 116   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  96 252 252 183  14   0   0  92 252 252 225  21   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0 132 253 252 146  14   0   0   0 215 252 252  79   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0 126 253 247 176   9   0   0   8  78 245 253 129   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  16 232 252 176   0   0   0  36 201 252 252 169  11   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  22 252 252  30  22 119 197 241 253 252 251  77   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  16 231 252 253 252 252 252 226 227 252 231   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0  55 235 253 217 138  42  24 192 252 143   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  62 255 253 109   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  71 253 252  21   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0 253 252  21   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  71 253 252  21   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0 106 253 252  21   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  45 255 253  21   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0 218 252  56   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  96 252 189  42   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  14 184 252 170  11   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  14 147 252  42   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]]
```

II. Data Preprocessing

1) Reshape and Normalization

- reshape
  - (60000, 28, 28) to (60000, 784)

```
X_train = X_train.reshape((60000, 28 * 28))
X_test = X_test.reshape((10000, 28 * 28))

X_train.shape, X_test.shape
```

```
((60000, 784), (10000, 784))
```

- Normalization

```
X_train = X_train.astype(float) / 255
X_test = X_test.astype(float) / 255
```

```
print(X_train[4])
```

```
[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.         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.
 0.21568627 0.58039216 0.82352941 0.99215686 0.99215686 0.44313725 0.34117647 0.58039216 0.21568627 0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.34117647 0.90980392 0.98823529 0.99215686 0.74117647 0.82352941 0.98823529 0.98823529 0.99215686 0.65882353 0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.01568627 0.22352941 0.94901961 0.98823529 0.74509804 0.25490196 0.01960784 0.04705882 0.71372549 0.98823529 0.99215686 0.45
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.37647059 0.98823529 0.98823529 0.71764706 0.05490196 0.         0.         0.36078431 0.98823529 0.98
 0.88235294 0.08235294 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.51764706 0.99215686 0.98823529 0.57254902 0.05490196 0.         0.         0.         0.84
 0.98823529 0.98823529 0.30980392 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.49411765 0.99215686 0.96862745 0.69019608 0.03529412 0.         0.         0.03
 0.30588235 0.96078431 0.99215686 0.50588235 0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.0627451  0.90980392 0.98823529 0.69019608 0.         0.         0.
 0.14117647 0.78823529 0.98823529 0.98823529 0.6627451  0.04313725 0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.08627451 0.98823529 0.98823529 0.11764706 0.08
 0.46666667 0.77254902 0.94509804 0.99215686 0.98823529 0.98431373 0.30196078 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.0627451  0.90588235 0.98
 0.99215686 0.98823529 0.98823529 0.98823529 0.88627451 0.89019608 0.98823529 0.90588235 0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.21568627 0.92156863 0.99215686 0.85098039 0.54117647 0.16470588 0.09411765 0.75294118 0.98823529 0.56078431 0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.24313725 1.         0.99215686 0.42745098 0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.27843137 0.99215686 0.98
 0.08235294 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.
 0.99215686 0.98823529 0.08235294 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.27843137 0.99215686 0.98823529 0.08235294 0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.41568627 0.99215686 0.98823529 0.08235294 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.17647059 1.         0.99215686 0.08235294 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.         0.85490196 0.98823529 0.21960784 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.         0.37647059 0.98823529 0.74
 0.16470588 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.05
 0.72156863 0.98823529 0.66666667 0.04313725 0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.05490196 0.57647059 0.98823529 0.16470588 0.         0.         0.         0.         0.         0.         0.
```

2) One Hot Encoding

```
from tensorflow.keras.utils import to_categorical

y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

```
print(y_train[:5])
```

```
[[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
```

```
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]
```

III. MNIST Keras Modeling

1) Model Define

- 모델 신경망 구조 정의
  - 2개의 Hidden Layers & 768개의 Nodes
  - 복잡한 Model Capacity로 인한 Overfitting

```
from tensorflow.keras import models
from tensorflow.keras import layers

mnist = models.Sequential()
mnist.add(layers.Dense(512, activation = 'relu', input_shape = (28 * 28,)))
mnist.add(layers.Dense(256, activation = 'relu'))
mnist.add(layers.Dense(10, activation = 'softmax'))
```

- 모델 구조 확인

```
mnist.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 512)               401920
_________________________________________________________________
dense_1 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_2 (Dense)              (None, 10)                2570
=================================================================
Total params: 535,818
Trainable params: 535,818
Non-trainable params: 0
_________________________________________________________________
```

2) Model Compile

- 모델 학습방법 설정

```
mnist.compile(loss = 'categorical_crossentropy',
              optimizer = 'rmsprop',
              metrics = ['accuracy'])
```

3) Model Fit

- 약 3분

```
%%time

Hist_mnist = mnist.fit(X_train, y_train,
                       epochs = 100,
                       batch_size = 128,
                       validation_split = 0.2)
```

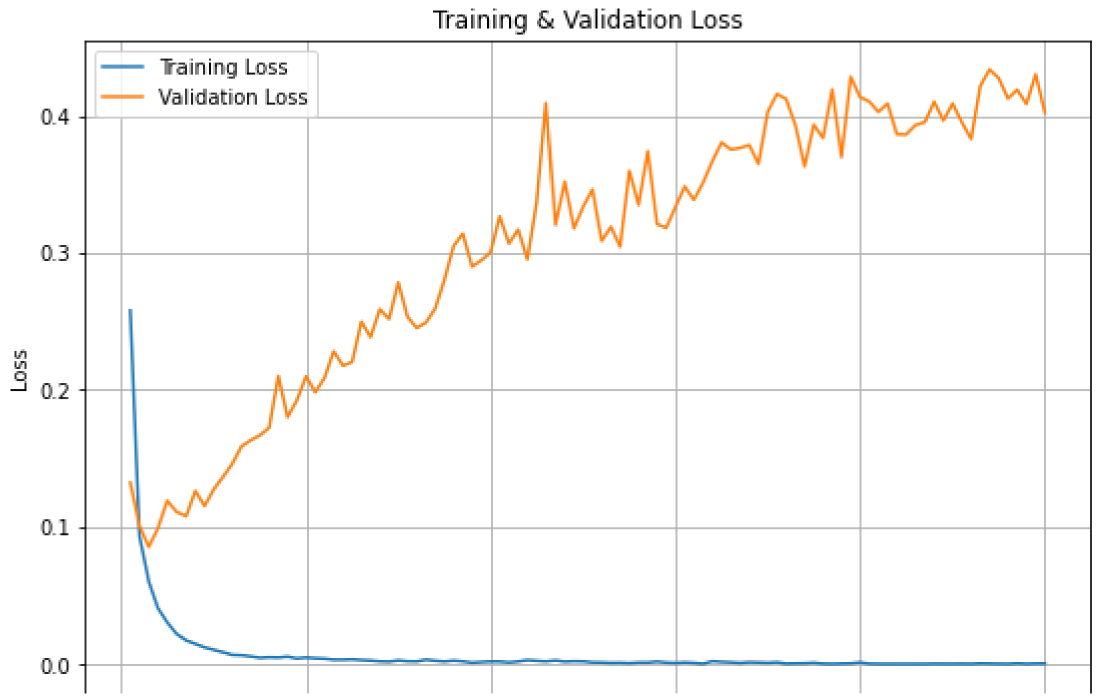
```
Epoch 1/100
375/375 [==============================] - 5s 6ms/step - loss: 0.2577 - accuracy: 0.9216 - val_loss: 0.1320 - val_accuracy: 0.9590
```

```
Epoch 2/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0928 - accuracy: 0.9709 - val_loss: 0.1005 - val_accuracy: 0.9707
Epoch 3/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0604 - accuracy: 0.9811 - val_loss: 0.0850 - val_accuracy: 0.9753
Epoch 4/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0408 - accuracy: 0.9872 - val_loss: 0.0986 - val_accuracy: 0.9743
Epoch 5/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0305 - accuracy: 0.9902 - val_loss: 0.1189 - val_accuracy: 0.9718
Epoch 6/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0222 - accuracy: 0.9927 - val_loss: 0.1108 - val_accuracy: 0.9756
Epoch 7/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0172 - accuracy: 0.9945 - val_loss: 0.1075 - val_accuracy: 0.9783
Epoch 8/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0147 - accuracy: 0.9951 - val_loss: 0.1259 - val_accuracy: 0.9768
Epoch 9/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0121 - accuracy: 0.9957 - val_loss: 0.1150 - val_accuracy: 0.9791
Epoch 10/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0103 - accuracy: 0.9967 - val_loss: 0.1272 - val_accuracy: 0.9782
Epoch 11/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0086 - accuracy: 0.9972 - val_loss: 0.1362 - val_accuracy: 0.9781
Epoch 12/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0066 - accuracy: 0.9980 - val_loss: 0.1458 - val_accuracy: 0.9795
Epoch 13/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0064 - accuracy: 0.9979 - val_loss: 0.1585 - val_accuracy: 0.9799
Epoch 14/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0056 - accuracy: 0.9983 - val_loss: 0.1629 - val_accuracy: 0.9799
Epoch 15/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0044 - accuracy: 0.9985 - val_loss: 0.1666 - val_accuracy: 0.9811
Epoch 16/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0049 - accuracy: 0.9983 - val_loss: 0.1721 - val_accuracy: 0.9804
Epoch 17/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0047 - accuracy: 0.9985 - val_loss: 0.2099 - val_accuracy: 0.9790
Epoch 18/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0054 - accuracy: 0.9985 - val_loss: 0.1799 - val_accuracy: 0.9805
Epoch 19/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0040 - accuracy: 0.9987 - val_loss: 0.1921 - val_accuracy: 0.9806
Epoch 20/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0046 - accuracy: 0.9986 - val_loss: 0.2096 - val_accuracy: 0.9788
Epoch 21/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0041 - accuracy: 0.9989 - val_loss: 0.1981 - val_accuracy: 0.9801
Epoch 22/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0039 - accuracy: 0.9989 - val_loss: 0.2085 - val_accuracy: 0.9789
Epoch 23/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0031 - accuracy: 0.9991 - val_loss: 0.2279 - val_accuracy: 0.9791
Epoch 24/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0031 - accuracy: 0.9990 - val_loss: 0.2175 - val_accuracy: 0.9811
Epoch 25/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0033 - accuracy: 0.9991 - val_loss: 0.2201 - val_accuracy: 0.9791
Epoch 26/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0028 - accuracy: 0.9993 - val_loss: 0.2496 - val_accuracy: 0.9791
Epoch 27/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0025 - accuracy: 0.9993 - val_loss: 0.2383 - val_accuracy: 0.9809
Epoch 28/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0019 - accuracy: 0.9996 - val_loss: 0.2588 - val_accuracy: 0.9801
Epoch 29/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0016 - accuracy: 0.9995 - val_loss: 0.2515 - val_accuracy: 0.9805
Epoch 30/100
```

4) 학습 결과 시각화 - Overfitting

- Loss Visualization

```
import matplotlib.pyplot as plt

epochs = range(1, len(Hist_mnist.history['loss']) + 1)

plt.figure(figsize = (9, 6))
plt.plot(epochs, Hist_mnist.history['loss'])
plt.plot(epochs, Hist_mnist.history['val_loss'])
# plt.ylim(0, 0.25)
plt.title('Training & Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(['Training Loss', 'Validation Loss'])
plt.grid()
plt.show()
```



5) Model Evaluate

- Loss & Accuracy

```
loss, accuracy = mnist.evaluate(X_test, y_test)

print('Loss = {:.5f}'.format(loss))
print('Accuracy = {:.5f}'.format(accuracy))
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.3150 - accuracy: 0.9833
Loss = 0.31498
Accuracy = 0.98330
```

6) Model Predict

- Probability

```
np.set_printoptions(suppress = True, precision = 9)

print(mnist.predict(X_test[:1,:]))
```

```
[[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]]
```

- Class

```
print(np.argmax(mnist.predict(X_test[:1,:])))
```

```
7
```

#

#

#

The End

#

#

#