MNIST - Categorical Classification

> ## Convolutional Neural Network

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

- import Tensorflow

```
import tensorflow

tensorflow.__version__
```

```
'2.11.0'
```

I. MNIST Data_Set Load

```
from tensorflow.keras.datasets import mnist

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

X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 2s 0us/step
((60000, 28, 28), (60000,), (10000, 28, 28), (10000,))
```

II. Data Preprocessing

1) Normalization

- 범위: 0 ~ 1

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

2) Reshape

- (60000, 28, 28) to (60000, 28, 28, 1)

```
X_train = X_train.reshape((60000, 28, 28, 1))
X_test = X_test.reshape((10000, 28, 28, 1))
```

3) One Hot Encoding

```
from tensorflow.keras.utils import to_categorical

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

4) train_test_split( )

- Train(48,000) vs. Validation(12,000)

```
from sklearn.model_selection import train_test_split

X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train,
                                                      test_size = 0.2,
                                                      random_state = 2045)

X_train.shape, y_train.shape, X_valid.shape, y_valid.shape
```

```
((48000, 28, 28, 1), (48000, 10), (12000, 28, 28, 1), (12000, 10))
```

III. MNIST Keras Modeling

1) Model Define

- Feature Extraction Layer

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

model = models.Sequential()
model.add(layers.Conv2D(filters = 32, kernel_size = (3, 3),
                        strides = (1, 1), padding = 'valid',
                        activation = 'relu', input_shape = (28, 28, 1)))
model.add(layers.MaxPool2D(pool_size = (2, 2)))
model.add(layers.Conv2D(filters = 32, kernel_size = (3, 3),
                        strides = (1, 1), padding = 'valid',
                        activation = 'relu'))
model.add(layers.MaxPool2D(pool_size = (2, 2)))
model.add(layers.Conv2D(filters = 32, kernel_size = (3, 3),
                        strides = (1, 1), padding = 'valid',
                        activation = 'relu'))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 11, 11, 32)        9248      
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 5, 5, 32)         0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (None, 3, 3, 32)          9248      
                                                                 
=================================================================
Total params: 18,816
Trainable params: 18,816
Non-trainable params: 0
_________________________________________________________________
```

- Classification Layer

```
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(units=64, activation='relu'))
model.add(layers.Dense(units=10, activation='softmax'))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 11, 11, 32)        9248      
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 5, 5, 32)         0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (None, 3, 3, 32)          9248      
                                                                 
 flatten (Flatten)           (None, 288)               0         
                                                                 
 dropout (Dropout)           (None, 288)               0         
                                                                 
 dense (Dense)               (None, 64)                18496     
                                                                 
 dense_1 (Dense)             (None, 10)                650       
                                                                 
=================================================================
Total params: 37,962
Trainable params: 37,962
Non-trainable params: 0
_________________________________________________________________
```

2) Model Compile

- 모델 학습방법 설정

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

3) Model Fit

- 약 5분

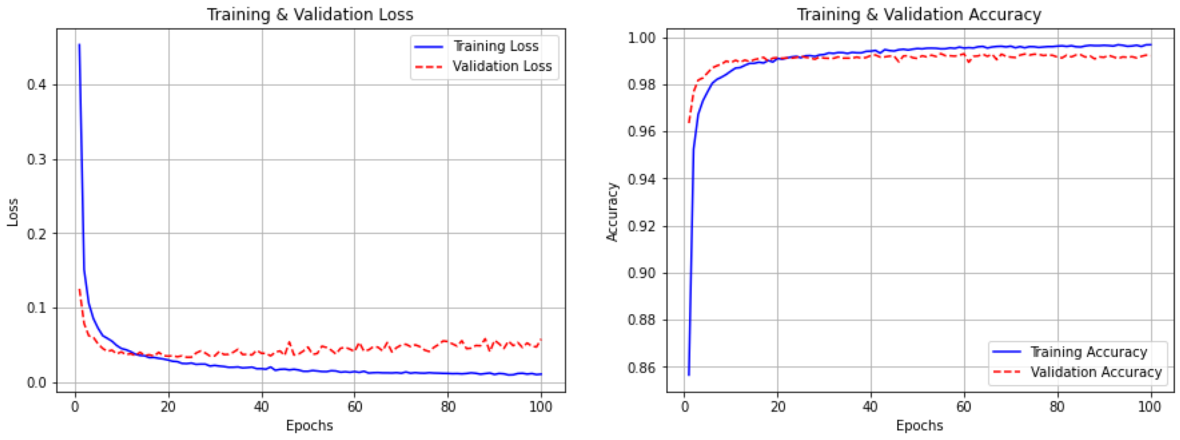
```
%%time

Hist_mnist = model.fit(X_train, y_train,
                       epochs = 100,
                       batch_size = 128,
                       validation_data = (X_valid, y_valid))
```

```
Epoch 1/100
375/375 [==============================] - 16s 8ms/step - loss: 0.4530 - accuracy: 0.8566 - val_loss: 0.1254 - val_accuracy: 0.9635
Epoch 2/100
375/375 [==============================] - 2s 5ms/step - loss: 0.1515 - accuracy: 0.9524 - val_loss: 0.0790 - val_accuracy: 0.9768
Epoch 3/100
375/375 [==============================] - 2s 5ms/step - loss: 0.1062 - accuracy: 0.9672 - val_loss: 0.0625 - val_accuracy: 0.9818
Epoch 4/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0854 - accuracy: 0.9730 - val_loss: 0.0603 - val_accuracy: 0.9827
Epoch 5/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0726 - accuracy: 0.9768 - val_loss: 0.0515 - val_accuracy: 0.9848
Epoch 6/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0626 - accuracy: 0.9804 - val_loss: 0.0454 - val_accuracy: 0.9869
Epoch 7/100
375/375 [==============================] - 3s 7ms/step - loss: 0.0590 - accuracy: 0.9820 - val_loss: 0.0416 - val_accuracy: 0.9878
Epoch 8/100
375/375 [==============================] - 3s 7ms/step - loss: 0.0551 - accuracy: 0.9830 - val_loss: 0.0431 - val_accuracy: 0.9885
Epoch 9/100
375/375 [==============================] - 2s 6ms/step - loss: 0.0494 - accuracy: 0.9841 - val_loss: 0.0381 - val_accuracy: 0.9898
Epoch 10/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0453 - accuracy: 0.9856 - val_loss: 0.0408 - val_accuracy: 0.9892
Epoch 11/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0437 - accuracy: 0.9869 - val_loss: 0.0371 - val_accuracy: 0.9902
Epoch 12/100
375/375 [==============================] - 2s 6ms/step - loss: 0.0411 - accuracy: 0.9871 - val_loss: 0.0384 - val_accuracy: 0.9893
Epoch 13/100
375/375 [==============================] - 2s 6ms/step - loss: 0.0377 - accuracy: 0.9880 - val_loss: 0.0358 - val_accuracy: 0.9902
Epoch 14/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0360 - accuracy: 0.9888 - val_loss: 0.0404 - val_accuracy: 0.9896
Epoch 15/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0356 - accuracy: 0.9889 - val_loss: 0.0360 - val_accuracy: 0.9904
Epoch 16/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0331 - accuracy: 0.9894 - val_loss: 0.0370 - val_accuracy: 0.9908
Epoch 17/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0333 - accuracy: 0.9890 - val_loss: 0.0350 - val_accuracy: 0.9914
Epoch 18/100
375/375 [==============================] - 2s 6ms/step - loss: 0.0321 - accuracy: 0.9901 - val_loss: 0.0403 - val_accuracy: 0.9898
```

```
Epoch 19/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0313 - accuracy: 0.9894 - val_loss: 0.0357 - val_accuracy: 0.9911
Epoch 20/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0299 - accuracy: 0.9909 - val_loss: 0.0351 - val_accuracy: 0.9912
Epoch 21/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0282 - accuracy: 0.9906 - val_loss: 0.0356 - val_accuracy: 0.9910
Epoch 22/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0277 - accuracy: 0.9911 - val_loss: 0.0343 - val_accuracy: 0.9912
Epoch 23/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0254 - accuracy: 0.9916 - val_loss: 0.0355 - val_accuracy: 0.9911
Epoch 24/100
375/375 [==============================] - 3s 7ms/step - loss: 0.0251 - accuracy: 0.9918 - val_loss: 0.0336 - val_accuracy: 0.9913
Epoch 25/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0259 - accuracy: 0.9912 - val_loss: 0.0337 - val_accuracy: 0.9919
Epoch 26/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0241 - accuracy: 0.9921 - val_loss: 0.0392 - val_accuracy: 0.9914
Epoch 27/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0244 - accuracy: 0.9921 - val_loss: 0.0422 - val_accuracy: 0.9912
Epoch 28/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0244 - accuracy: 0.9918 - val_loss: 0.0393 - val_accuracy: 0.9906
Epoch 29/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0218 - accuracy: 0.9924 - val_loss: 0.0352 - val_accuracy: 0.9916
```

4) 학습 결과 시각화

```
import matplotlib.pyplot as plt

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

plt.figure(figsize = (15, 5))

plt.subplot(1, 2, 1)
plt.plot(epochs, Hist_mnist.history['loss'], 'b-')
plt.plot(epochs, Hist_mnist.history['val_loss'], 'r--')
plt.title('Training & Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(['Training Loss', 'Validation Loss'])
plt.grid()

plt.subplot(1, 2, 2)
plt.plot(epochs, Hist_mnist.history['accuracy'], 'b-')
plt.plot(epochs, Hist_mnist.history['val_accuracy'], 'r--')
plt.title('Training & Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend(['Training Accuracy', 'Validation Accuracy'])
plt.grid()

plt.show()
```



5) Model Evaluate

- Loss & Accuracy

```
loss, accuracy = model.evaluate(X_test, y_test, verbose = 0)

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

```
Loss = 0.03491
Accuracy = 0.99410
```

#

#

#

The End

#

#

#