MNIST - Categorical Classification

> # Convolutional Neural Network

Import Keras

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

- import Keras

```
import keras

keras.__version__
```

```
'2.4.3'
```

I. MNIST Data_Set Load

```
from 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
```

II. Data Preprocessing

> ## 1) Reshape and Normalization

- reshape

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

- Normalization

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

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

2) One Hot Encoding

```
from keras.utils import to_categorical

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

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), activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPool2D(pool_size=(2,2)))
model.add(layers.Conv2D(filters=64, kernel_size=(3,3), activation='relu'))
model.add(layers.MaxPool2D(pool_size=(2,2)))
model.add(layers.Conv2D(filters=64, kernel_size=(3,3), 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, 64)        18496     
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 3, 3, 64)          36928     
=================================================================
Total params: 55,744
Trainable params: 55,744
Non-trainable params: 0
_________________________________________________________________
```

- Classification Layer

```
model.add(layers.Flatten())
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, 64)        18496     
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 3, 3, 64)          36928     
_________________________________________________________________
flatten (Flatten)            (None, 576)               0         
_________________________________________________________________
dense (Dense)                (None, 64)                36928     
_________________________________________________________________
dense_1 (Dense)              (None, 10)                650       
=================================================================
Total params: 93,322
Trainable params: 93,322
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_split = 0.2)
```

```
Epoch 1/100
375/375 [==============================] - 35s 7ms/step - loss: 0.6107 - accuracy: 0.8074
Epoch 2/100
375/375 [==============================] - 2s 6ms/step - loss: 0.0686 - accuracy: 0.9786 -
Epoch 3/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0394 - accuracy: 0.9871 -
Epoch 4/100
375/375 [==============================] - 2s 6ms/step - loss: 0.0286 - accuracy: 0.9908 -
Epoch 5/100
375/375 [==============================] - 2s 6ms/step - loss: 0.0222 - accuracy: 0.9928 -
Epoch 6/100
375/375 [==============================] - 2s 6ms/step - loss: 0.0171 - accuracy: 0.9945 -
Epoch 7/100
375/375 [==============================] - 2s 6ms/step - loss: 0.0124 - accuracy: 0.9958 -
Epoch 8/100
375/375 [==============================] - 2s 6ms/step - loss: 0.0097 - accuracy: 0.9970 -
Epoch 9/100
375/375 [==============================] - 2s 6ms/step - loss: 0.0082 - accuracy: 0.9973 -
Epoch 10/100
375/375 [==============================] - 2s 6ms/step - loss: 0.0061 - accuracy: 0.9979 -
Epoch 11/100
375/375 [==============================] - 2s 6ms/step - loss: 0.0058 - accuracy: 0.9979 -
Epoch 12/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0048 - accuracy: 0.9981 -
Epoch 13/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0040 - accuracy: 0.9988 -
Epoch 14/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0031 - accuracy: 0.9988 -
Epoch 15/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0027 - accuracy: 0.9991 -
Epoch 16/100
375/375 [==============================] - 2s 6ms/step - loss: 0.0031 - accuracy: 0.9991 -
Epoch 17/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0031 - accuracy: 0.9992 -
Epoch 18/100
375/375 [==============================] - 2s 6ms/step - loss: 0.0029 - accuracy: 0.9990 -
Epoch 19/100
375/375 [==============================] - 2s 5ms/step - loss: 0.0026 - accuracy: 0.9991 -
Epoch 20/100
375/375 [==============================] - 2s 6ms/step - loss: 0.0030 - accuracy: 0.9992 -
Epoch 21/100
375/375 [==============================] - 2s 6ms/step - loss: 0.0015 - accuracy: 0.9994 -
Epoch 22/100
375/375 [==============================] - 2s 6ms/step - loss: 0.0019 - accuracy: 0.9992 -
Epoch 23/100
375/375 [==============================] - 2s 6ms/step - loss: 0.0020 - accuracy: 0.9994 -
Epoch 24/100
375/375 [==============================] - 2s 6ms/step - loss: 0.0024 - accuracy: 0.9992 -
Epoch 25/100
375/375 [==============================] - 2s 6ms/step - loss: 0.0013 - accuracy: 0.9996 -
Epoch 26/100
375/375 [==============================] - 2s 6ms/step - loss: 0.0016 - accuracy: 0.9996 -
Epoch 27/100
375/375 [==============================] - 2s 6ms/step - loss: 0.0022 - accuracy: 0.9994 -
Epoch 28/100
375/375 [==============================] - 2s 6ms/step - loss: 0.0014 - accuracy: 0.9996 -
Epoch 29/100
375/375 [==============================] - 2s 6ms/step - loss: 0.0023 - accuracy: 0.9995 -
```

4) 학습 결과 시각화

- 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, 0.4)
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 = model.evaluate(X_test, y_test)

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

```
313/313 [==============================] - 1s 2ms/step - loss: 0.1608 - accuracy: 0.9919
Loss = 0.16085
Accuracy = 0.99190
```

#

#

#

The End

#

#

#