CIFAR 10 - Categorical Classification

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

Import Tensorflow & Keras

- import Keras

```
import keras

keras.__version__
```

```
'2.4.3'
```

I. CIFAR 10 Data_Set Load & Review

1) Load CIFAR 10 Data_Set

```
from keras.datasets import cifar10

(X_train, y_train), (X_test, y_test) = cifar10.load_data()
```

- Train_Data Information

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

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

```
50000
(50000, 32, 32, 3)
50000
[[6]
 [9]
 [9]
 [4]
 [1]]
```

- Test_Data Information

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

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

```
10000
(10000, 32, 32, 3)
10000
[[3]
 [8]
 [8]
 [0]
 [6]]
```

2) Visualization

```
import matplotlib.pyplot as plt

digit = X_train[0]
plt.imshow(digit)
plt.show()
```

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

print(X_train[0][0])
```

```
[[ 59  62  63]
 [ 43  46  45]
 [ 50  48  43]
 [ 68  54  42]
 [ 98  73  52]
 [119  91  63]
 [139 107  75]
 [145 110  80]
 [149 117  89]
 [149 120  93]
 [131 103  77]
 [125  99  76]
 [142 115  91]
 [144 112  86]
 [137 105  79]
 [129  97  71]
 [137 106  79]
 [134 106  76]
 [124  97  64]
 [139 113  78]
 [139 112  75]
 [133 105  69]
 [136 105  74]
 [139 108  77]
```

```
  [152 120  89]
  [163 131 100]
  [168 136 108]
  [159 129 102]
  [158 130 104]
  [158 132 108]
  [152 125 102]
  [148 124 103]]
```

II. Data Preprocessing

1) Reshape and Normalization

- reshape
  - (50000, 32, 32, 3) to (50000, 3072)

```
X_train = X_train.reshape((50000, 32 * 32 * 3))
X_test = X_test.reshape((10000, 32 * 32 * 3))

X_train.shape, X_test.shape
```

```
((50000, 3072), (10000, 3072))
```

- Normalization

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

```
print(X_train[0])
```

```
[0.231372549 0.243137255 0.247058824 ... 0.482352941 0.360784314 0.282352941]
```

2) One Hot Encoding

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

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

III. Keras Modeling

1) Model Define

- 모델 신경망 구조 정의
  - 2개의 Hidden Layers & 3968개의 Nodes

```
from keras import models
from keras import layers

CIFAR = models.Sequential()
CIFAR.add(layers.Dense(2048, activation = 'relu', input_shape = (32 * 32 * 3,)))
CIFAR.add(layers.Dense(1024, activation = 'relu'))
CIFAR.add(layers.Dense(512, activation = 'relu'))
CIFAR.add(layers.Dense(256, activation = 'relu'))
CIFAR.add(layers.Dense(128, activation = 'relu'))
CIFAR.add(layers.Dense(10, activation = 'softmax'))
```

- 모델 구조 확인

```
mnist.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 512)               1573376
_________________________________________________________________
dense_1 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_2 (Dense)              (None, 10)                2570
=================================================================
Total params: 1,707,274
Trainable params: 1,707,274
Non-trainable params: 0
_________________________________________________________________
```

2) Model Compile

- 모델 학습방법 설정

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

3) Model Fit

- 약 4분

```
%%time

Hist_CIFAR = CIFAR.fit(X_train, y_train,
                       epochs = 100,
                       batch_size = 128,
                       validation_split = 0.2)
```

```
Epoch 1/100
313/313 [==============================] - 3s 6ms/step - loss: 3.2322 - accuracy: 0.1612 -
Epoch 2/100
313/313 [==============================] - 2s 6ms/step - loss: 1.9415 - accuracy: 0.2963 -
Epoch 3/100
313/313 [==============================] - 2s 6ms/step - loss: 1.8189 - accuracy: 0.3471 -
Epoch 4/100
313/313 [==============================] - 2s 6ms/step - loss: 1.7436 - accuracy: 0.3775 -
Epoch 5/100
313/313 [==============================] - 2s 6ms/step - loss: 1.6842 - accuracy: 0.3962 -
Epoch 6/100
313/313 [==============================] - 2s 6ms/step - loss: 1.6302 - accuracy: 0.4167 -
Epoch 7/100
313/313 [==============================] - 2s 6ms/step - loss: 1.5919 - accuracy: 0.4292 -
Epoch 8/100
313/313 [==============================] - 2s 6ms/step - loss: 1.5757 - accuracy: 0.4368 -
Epoch 9/100
313/313 [==============================] - 2s 6ms/step - loss: 1.5214 - accuracy: 0.4590 -
Epoch 10/100
313/313 [==============================] - 2s 6ms/step - loss: 1.4999 - accuracy: 0.4582 -
Epoch 11/100
313/313 [==============================] - 2s 6ms/step - loss: 1.4752 - accuracy: 0.4693 -
Epoch 12/100
313/313 [==============================] - 2s 6ms/step - loss: 1.4441 - accuracy: 0.4839 -
Epoch 13/100
313/313 [==============================] - 2s 6ms/step - loss: 1.4236 - accuracy: 0.4901 -
Epoch 14/100
313/313 [==============================] - 2s 6ms/step - loss: 1.4020 - accuracy: 0.4986 -
Epoch 15/100
313/313 [==============================] - 2s 6ms/step - loss: 1.3738 - accuracy: 0.5114 -
Epoch 16/100
313/313 [==============================] - 2s 6ms/step - loss: 1.3461 - accuracy: 0.5191 -
Epoch 17/100
313/313 [==============================] - 2s 6ms/step - loss: 1.3167 - accuracy: 0.5295 -
Epoch 18/100
313/313 [==============================] - 2s 6ms/step - loss: 1.3205 - accuracy: 0.5250 -
Epoch 19/100
313/313 [==============================] - 2s 6ms/step - loss: 1.2927 - accuracy: 0.5363 -
Epoch 20/100
313/313 [==============================] - 2s 6ms/step - loss: 1.2826 - accuracy: 0.5484 -
```

```
Epoch 21/100
313/313 [==============================] - 2s 6ms/step - loss: 1.2687 - accuracy: 0.5456 -
Epoch 22/100
313/313 [==============================] - 2s 6ms/step - loss: 1.2464 - accuracy: 0.5596 -
Epoch 23/100
313/313 [==============================] - 2s 6ms/step - loss: 1.2073 - accuracy: 0.5702 -
Epoch 24/100
313/313 [==============================] - 2s 6ms/step - loss: 1.2068 - accuracy: 0.5669 -
Epoch 25/100
313/313 [==============================] - 2s 6ms/step - loss: 1.2320 - accuracy: 0.5743 -
Epoch 26/100
313/313 [==============================] - 2s 6ms/step - loss: 1.1758 - accuracy: 0.5819 -
Epoch 27/100
313/313 [==============================] - 2s 6ms/step - loss: 1.1709 - accuracy: 0.5872 -
Epoch 28/100
313/313 [==============================] - 2s 6ms/step - loss: 1.1438 - accuracy: 0.5967 -
Epoch 29/100
313/313 [==============================] - 2s 6ms/step - loss: 1.1147 - accuracy: 0.6004 -
```

4) 학습 결과 시각화

- Loss Visualization

```
import matplotlib.pyplot as plt

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

plt.figure(figsize = (9, 6))
plt.plot(epochs, Hist_CIFAR.history['loss'])
plt.plot(epochs, Hist_CIFAR.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 = CIFAR.evaluate(X_test, y_test)

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

```
313/313 [==============================] - 1s 2ms/step - loss: 3.7357 - accuracy: 0.4688
Loss = 3.73567
```

```
Accuracy = 0.46880
```

6) Model Predict

- Probability

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

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

```
[[0.014115457 0.000029894 0.004261574 0.7763388   0.001592926 0.18596697  0.00000009  0.00000
```

- Class

```
print(CIFAR.predict_classes(X_test[:1,:]))
```

```
[4]
```

#

#

#

The End

#

#

#