CIFAR10-cnn

一、 程式碼

1. 導入 TensorFlow 庫中 CIFAR10 資料集，並分為測試集與訓練集。

```
### Import TensorFlow

import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
import numpy as np

### Download and prepare the CIFAR10 dataset

(train_images, train_labels), (test_images, test_labels) =
datasets.cifar10.load_data()
```

- 新增混淆矩陣

2. 進行標準化

```
# Normalize pixel values to be between 0 and 1
train_images, test_images = train_images / 255.0, test_images
/ 255.0
```

3. 檢驗資料庫數據，並有 6*6 個的訓練圖像。

```
### Verify the data

class_names = ['airplane', 'automobile', 'bird', 'cat',
'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']

plt.figure(figsize=(10,10))
for i in range(36):
    plt.subplot(6,6,i+1)
```

```
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i])
    plt.xlabel(class_names[train_labels[i][0]])
plt.show()
```

- **原本只有顯示 25 個，改為 36 個**

4. 建立 2 層卷積，提取特徵

```
### Create the convolutional base

model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',
input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.summary()
```

5. 查看模型，輸出形狀、參數等等

```
### Add Dense layers on top

model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))

model.summary()
```

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d_2 (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_4 (Conv2D)	(None, 13, 13, 64)	18496
max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_5 (Conv2D)	(None, 4, 4, 64)	36928
flatten_1 (Flatten)	(None, 1024)	0
dense_2 (Dense)	(None, 64)	65600
dense_3 (Dense)	(None, 10)	650

6. 編譯及訓練模型，訓練過程中的損失值和準確率會儲存在 history 中，以便分析和可視覺化，並評估模型性能。

```
### Compile and train the model

model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentrop
y(from_logits=True),
              metrics=['accuracy'])

history = model.fit(train_images, train_labels, epochs=12,
                    validation_data=(test_images, test_labels))
    ### Evaluate the model


plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label =
'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.1, 1])
plt.legend(loc='lower right')

test_loss, test_acc =
model.evaluate(test_images,  test_labels, verbose=2)
print(test_acc)
```

7. 生成混淆矩陣，原本沒有

```
### Generate confusion matrix and plot it

# Predict the labels for the test set
predictions = model.predict(test_images)
predicted_labels = np.argmax(predictions, axis=1)

# Create confusion matrix
cm = confusion_matrix(test_labels, predicted_labels)

# Plot the confusion matrix
plt.figure(figsize=(10, 8))
```

```
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
xticklabels=class_names, yticklabels=class_names)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```