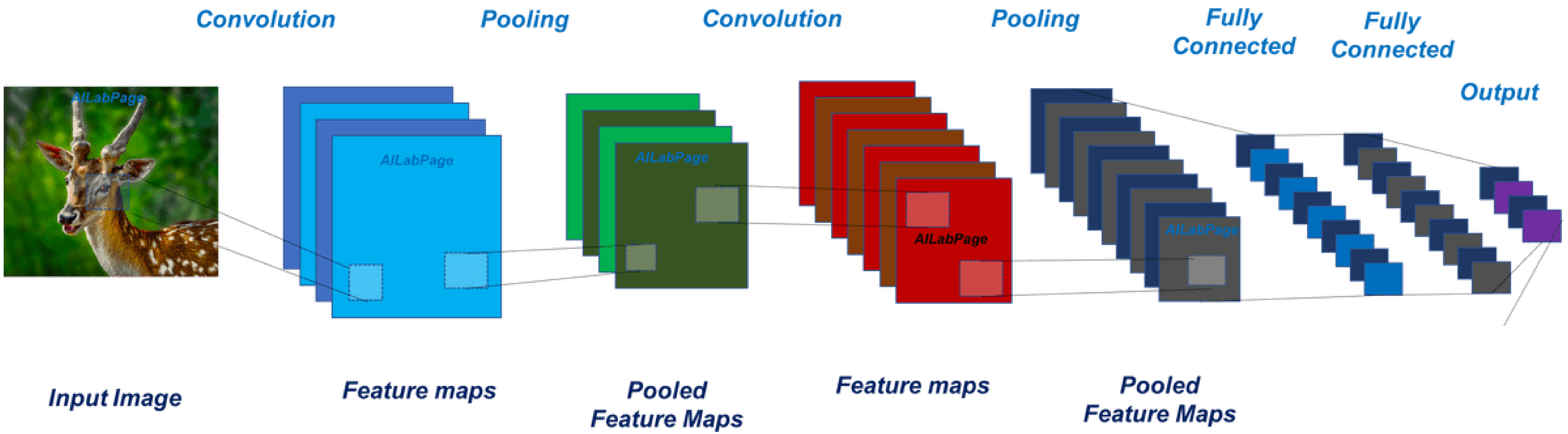
CNN卷積神經網路

2024/10/7 吳品儒

Convolution Neural Network



Convolution

Pooling

Convolution

Pooling

Fully Connected

Fully Connected

Output

Input Image

Feature maps

Pooled Feature Maps

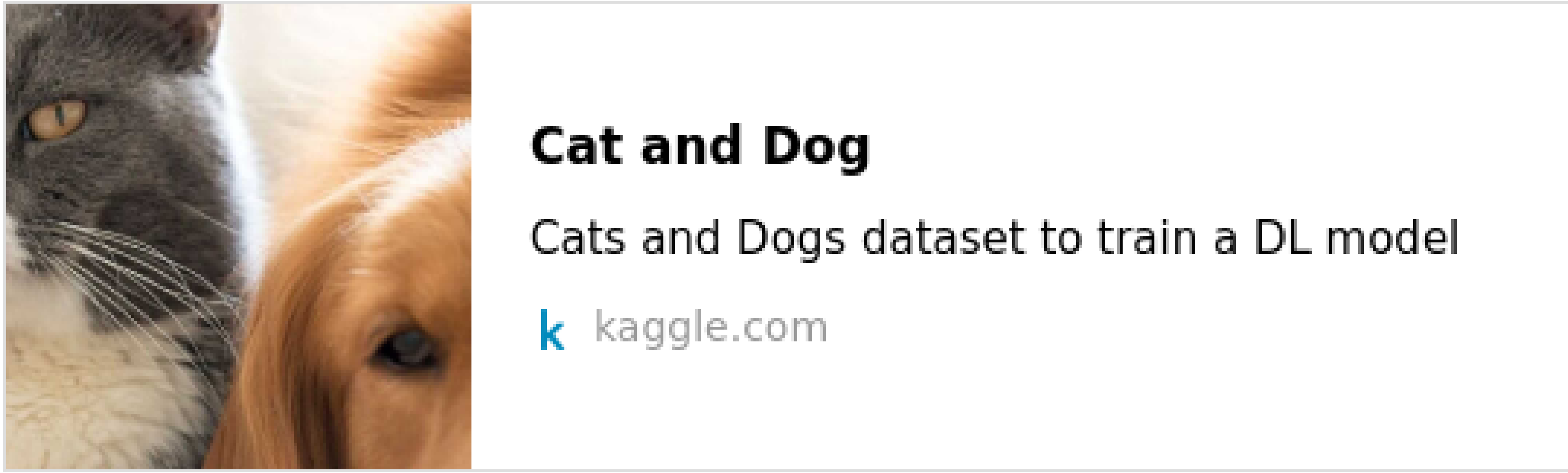
Feature maps

Pooled Feature Maps

Kaggle:Cats and Dogs

Training:cats/1000,dogs/1000

Test:cats/100,dogs/100



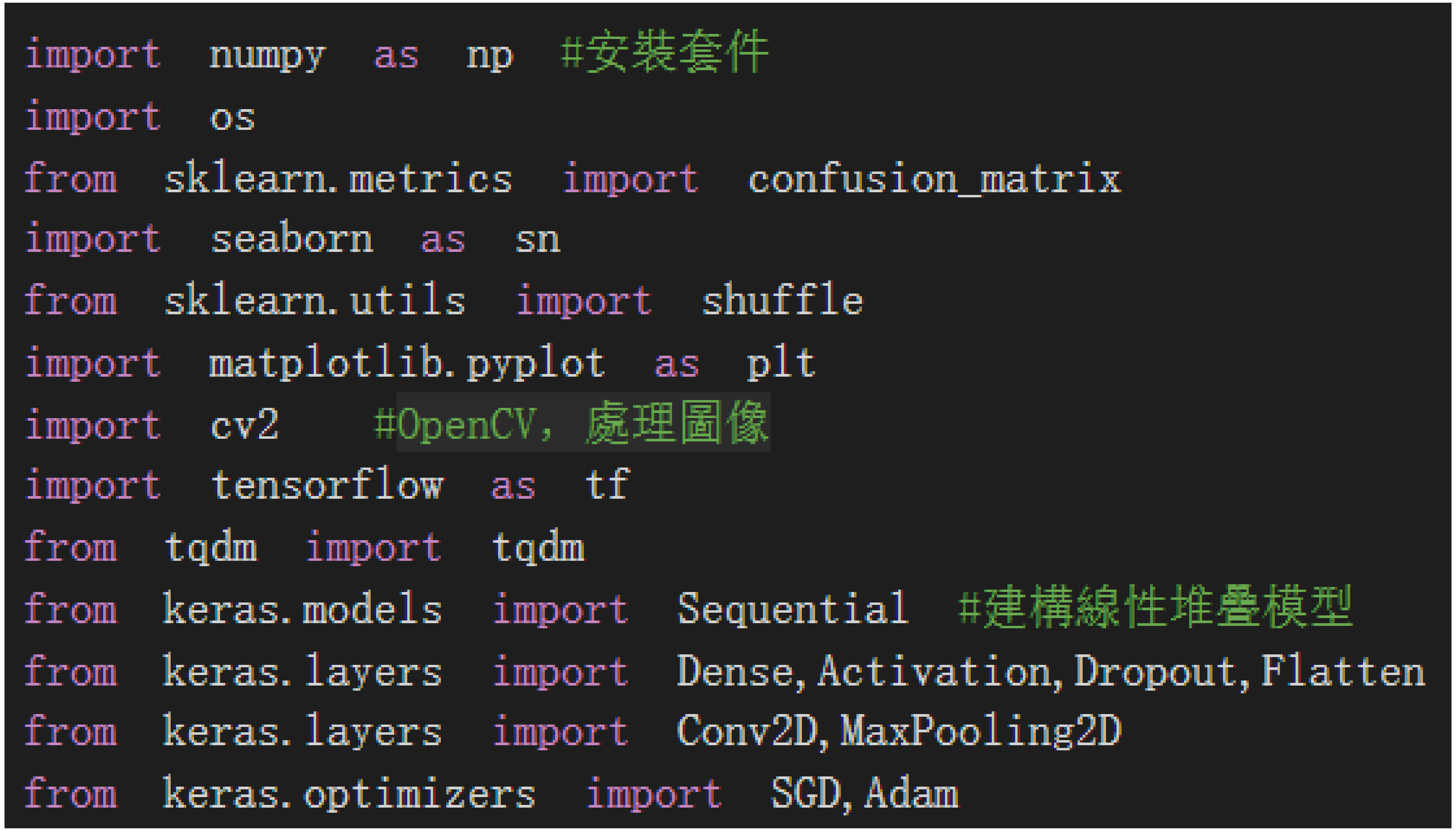
Cat and Dog

Cats and Dogs dataset to train a DL model

k kaggle.com

資料來源:https://www.kaggle.com/datasets/tongpython/cat-and-dog/code

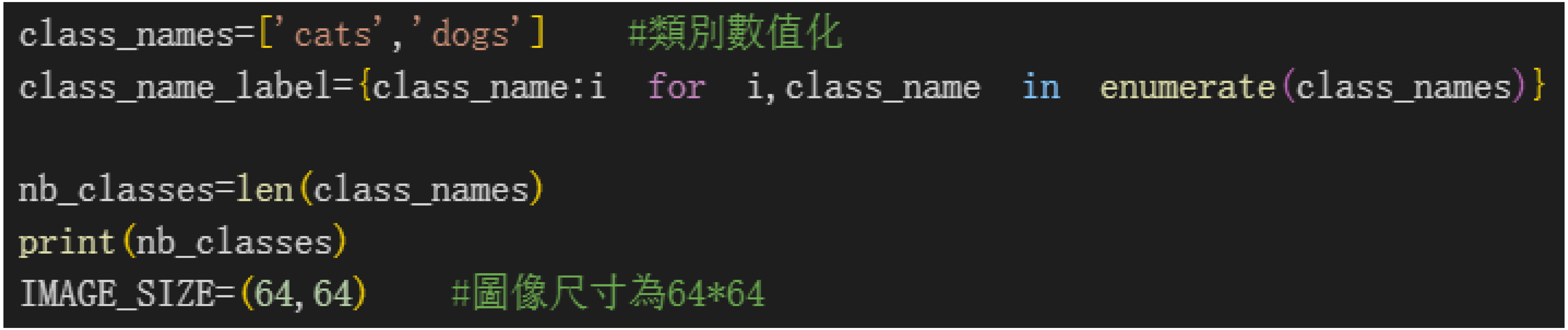
1.安裝套件



```
import numpy as np #安裝套件
import os
from sklearn.metrics import confusion_matrix
import seaborn as sn
from sklearn.utils import shuffle
import matplotlib.pyplot as plt
import cv2   #OpenCV, 處理圖像
import tensorflow as tf
from tqdm import tqdm
from keras.models import Sequential #建構線性堆疊模型
from keras.layers import Dense,Activation,Dropout,Flatten
from keras.layers import Conv2D,MaxPooling2D
from keras.optimizers import SGD,Adam
```

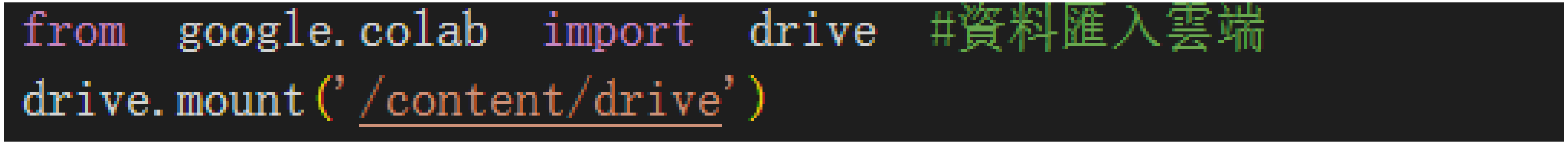
Dense:全連接層
Activation:激活函數
Dropout:丟棄層
Flatten:攤平層
Conv2D:卷積層
MaxPooling2D:池化層
SGD&Adam:優化器

2.把cats&dogs標記成{0,1}，圖像長寬設定為64*64



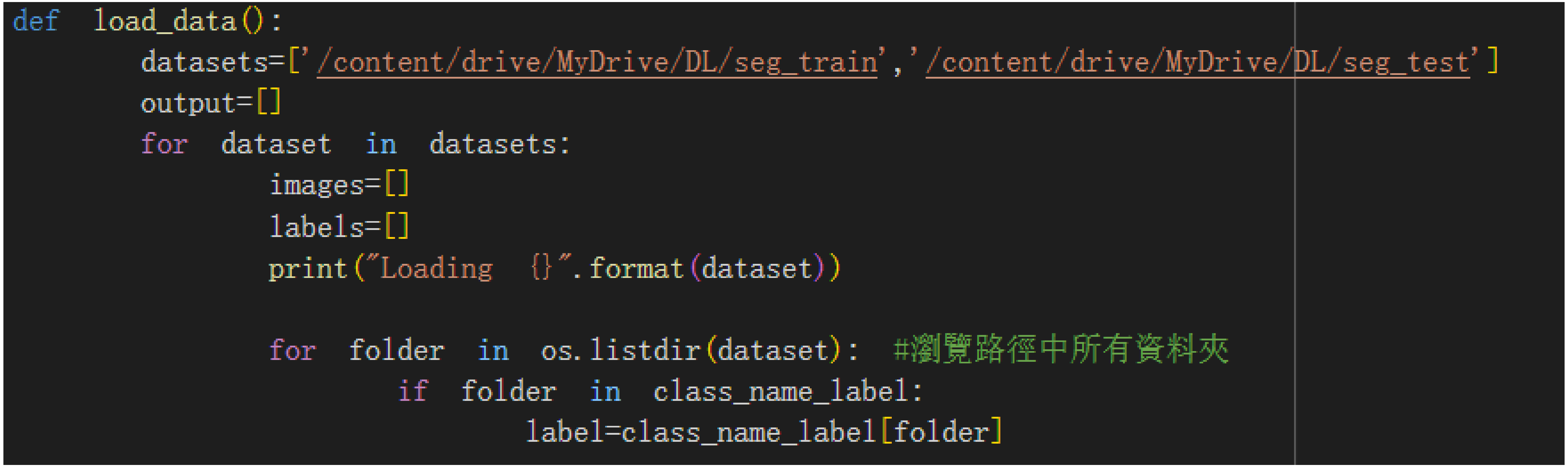
```
class_names=['cats','dogs']    #類別數值化
class_name_label={class_name:i for i,class_name in enumerate(class_names)}

nb_classes=len(class_names)
print(nb_classes)
IMAGE_SIZE=(64,64)    #圖像尺寸為64*64
```

3.將資料集匯入雲端



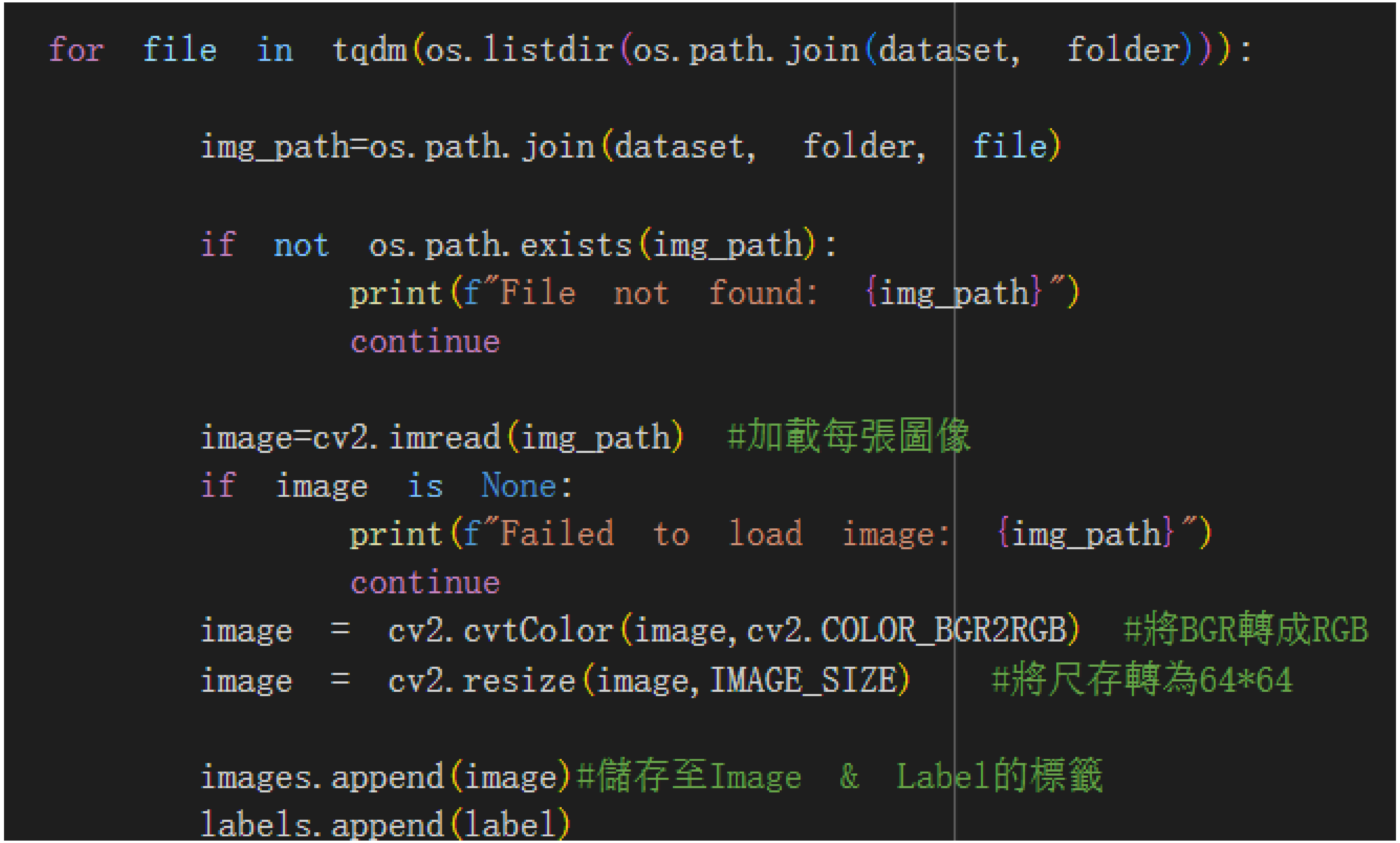
```
from google.colab import drive #資料匯入雲端
drive.mount('/content/drive')
```

4.抓取路徑中含有cats&dogs的資料夾



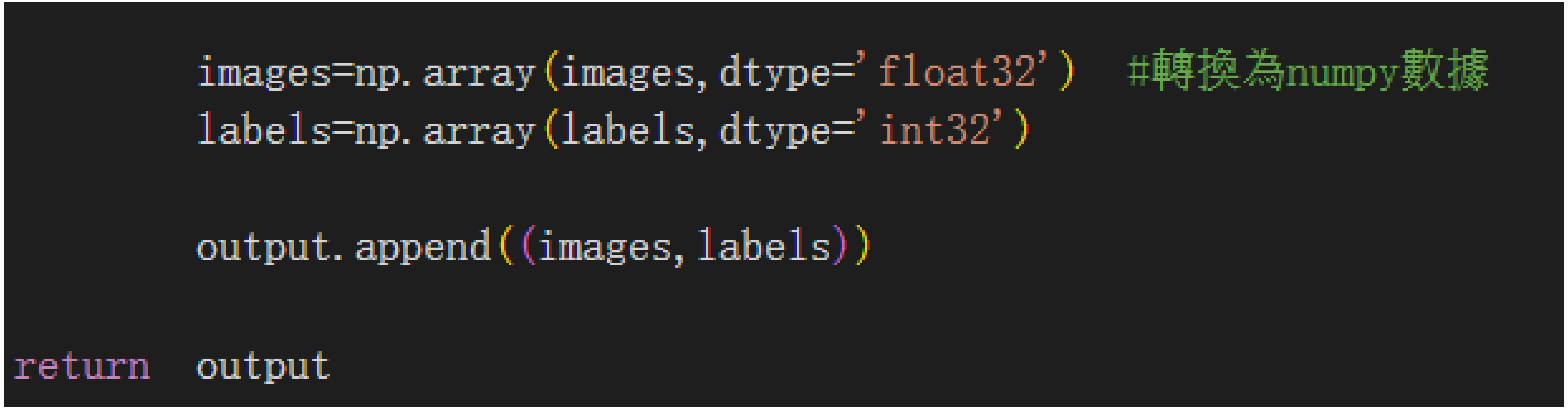
```
def load_data():
    datasets=[’/content/drive/MyDrive/DL/seg_train’,’/content/drive/MyDrive/DL/seg_test’]
    output=[]
    for dataset in datasets:
        images=[]
        labels=[]
        print(”Loading {}”.format(dataset))

        for folder in os.listdir(dataset): #瀏覽路徑中所有資料夾
            if folder in class_name_label:
                label=class_name_label[folder]
```

5.使用OpenCV加載圖像，將BGR轉成RGB



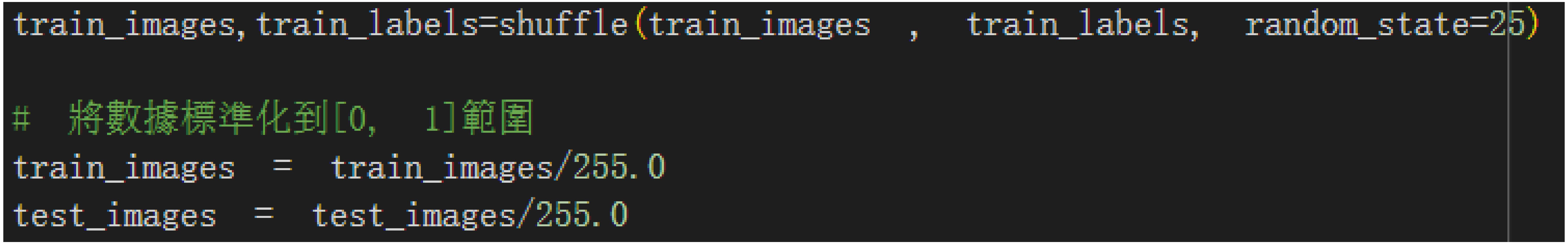
```
for file in tqdm(os.listdir(os.path.join(dataset, folder))):

    img_path=os.path.join(dataset, folder, file)

    if not os.path.exists(img_path):
        print(f"File not found: {img_path}")
        continue

    image=cv2.imread(img_path)  #加載每張圖像
    if image is None:
        print(f"Failed to load image: {img_path}")
        continue
    image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)  #將BGR轉成RGB
    image = cv2.resize(image,IMAGE_SIZE)    #將尺存轉為64*64

    images.append(image)#儲存至Image & Label的標籤
    labels.append(label)
```

6.將images,labels向量化，轉換為numpy數據組



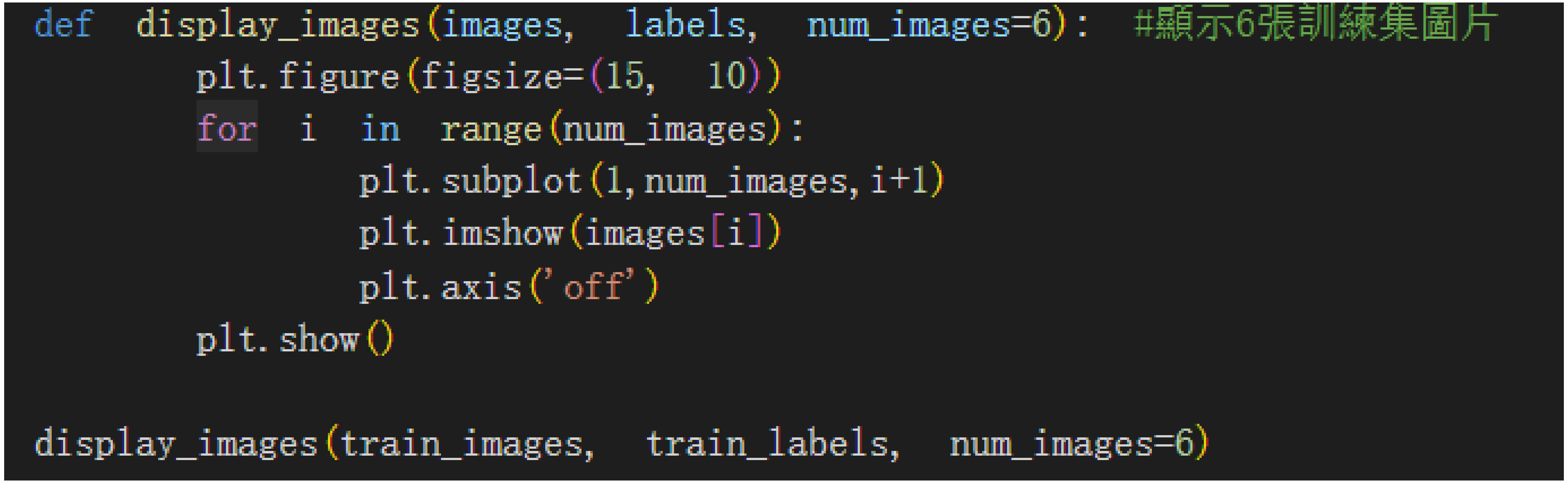
```
        images=np.array(images,dtype=’float32’)  #轉換為numpy數據
        labels=np.array(labels,dtype=’int32’)

        output.append((images,labels))

return  output
```

7.將訓練集數據打亂，並將數據標準化為[0,1]



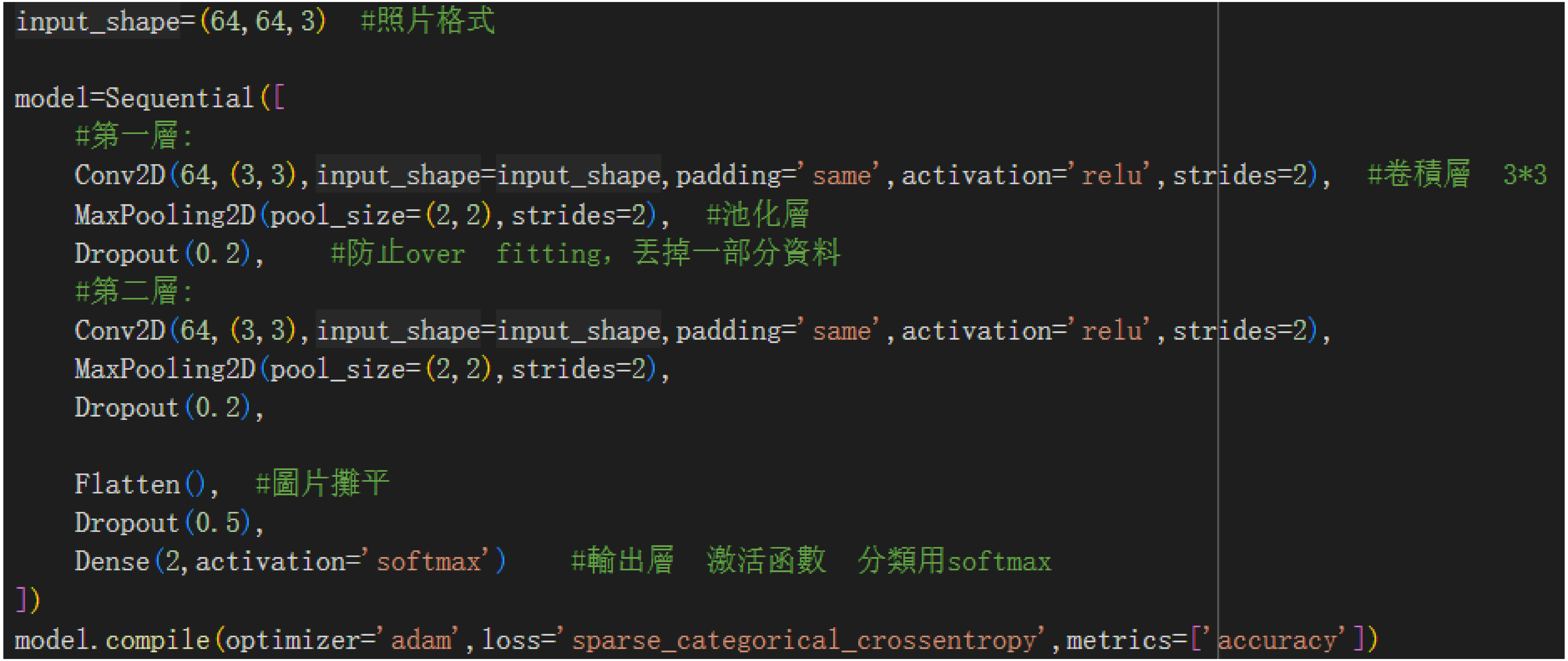
```
train_images,train_labels=shuffle(train_images , train_labels, random_state=25)

# 將數據標準化到[0, 1]範圍
train_images = train_images/255.0
test_images = test_images/255.0
```

8.顯示6張訓練集圖片



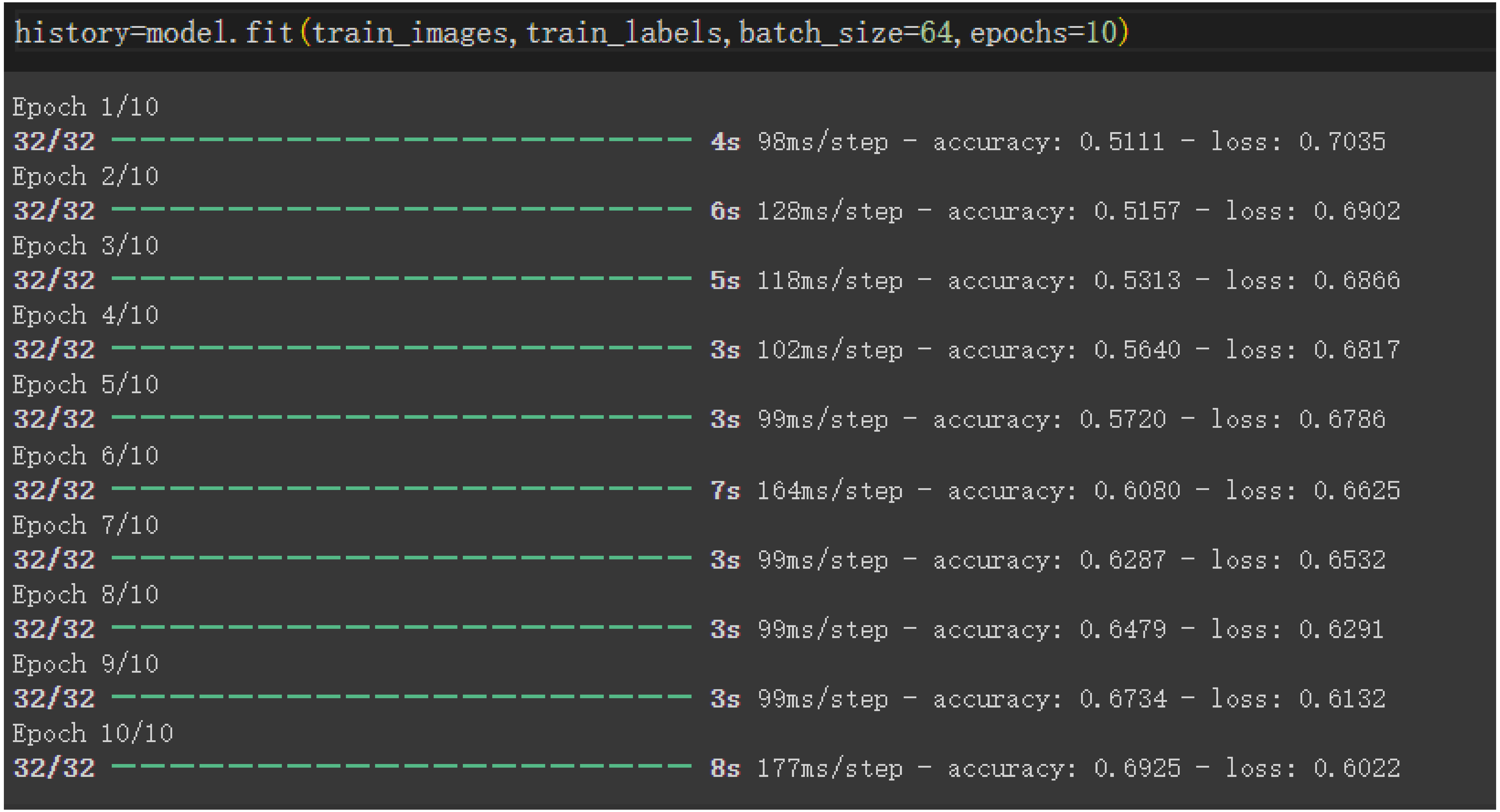
```
def display_images(images, labels, num_images=6):  #顯示6張訓練集圖片
    plt.figure(figsize=(15, 10))
    for i in range(num_images):
        plt.subplot(1,num_images,i+1)
        plt.imshow(images[i])
        plt.axis(’off’)
    plt.show()

display_images(train_images, train_labels, num_images=6)
```

9.建構CNN模型



```
input_shape=(64,64,3)  #照片格式

model=Sequential([
    #第一層:
    Conv2D(64, (3,3),input_shape=input_shape,padding=’same’,activation=’relu’,strides=2),  #卷積層  3*3
    MaxPooling2D(pool_size=(2,2),strides=2),  #池化層
    Dropout(0.2),    #防止over  fitting，丟掉一部分資料
    #第二層:
    Conv2D(64, (3,3),input_shape=input_shape,padding=’same’,activation=’relu’,strides=2),
    MaxPooling2D(pool_size=(2,2),strides=2),
    Dropout(0.2),

    Flatten(),  #圖片攤平
    Dropout(0.5),
    Dense(2,activation=’softmax’)    #輸出層  激活函數  分類用softmax
])
model.compile(optimizer=’adam’,loss=’sparse_categorical_crossentropy’,metrics=[’accuracy’])
```

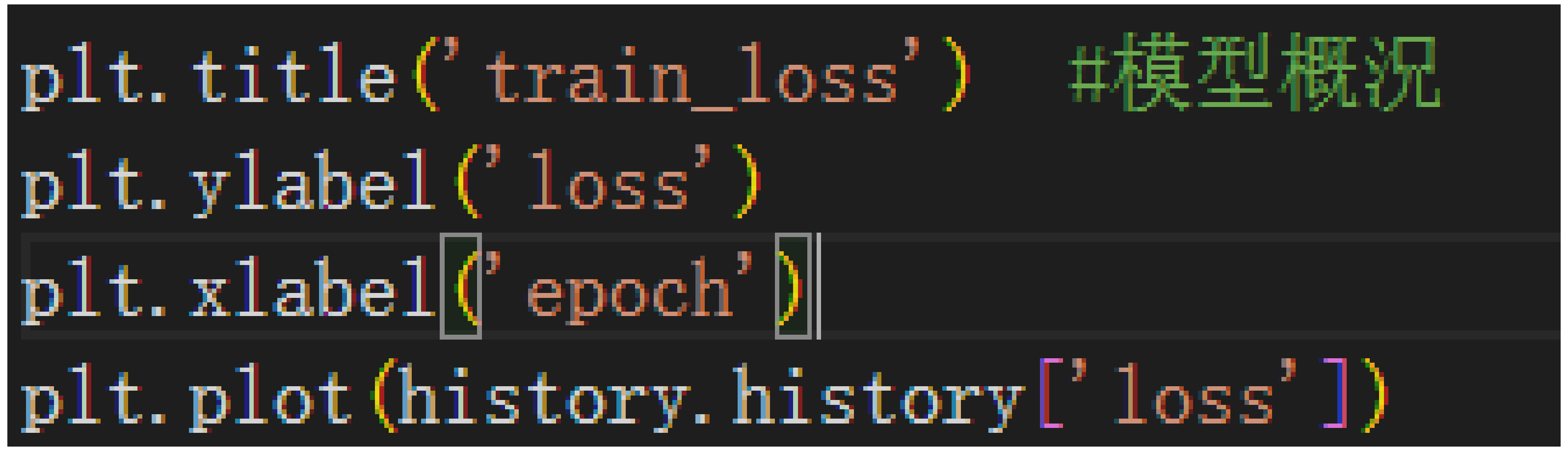
10.訓練CNN模型，batch_size=64,epochs=10



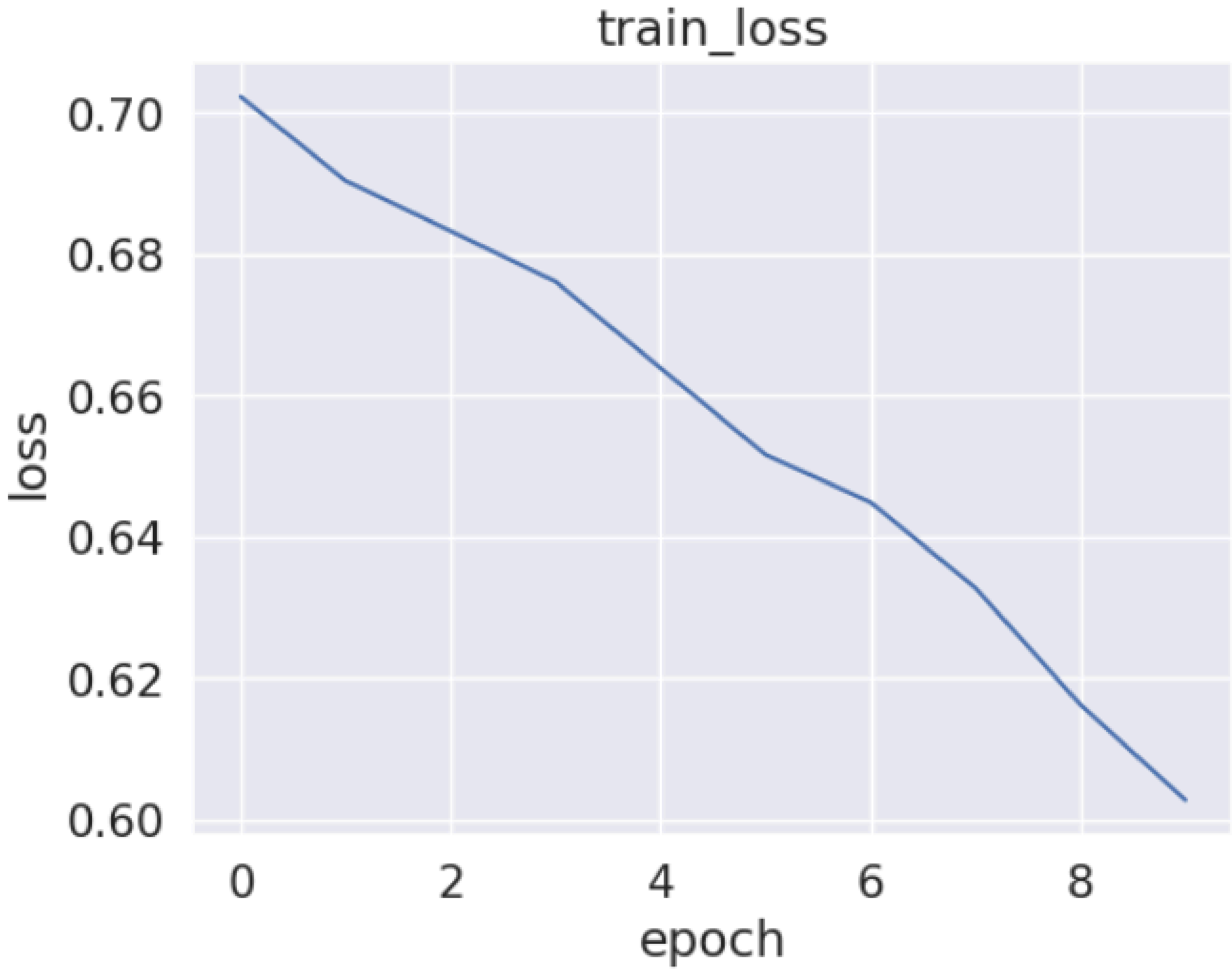
```
history=model.fit(train_images, train_labels, batch_size=64, epochs=10)
```

```
Epoch 1/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 4s 98ms/step - accuracy: 0.5111 - loss: 0.7035
Epoch 2/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 6s 128ms/step - accuracy: 0.5157 - loss: 0.6902
Epoch 3/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 5s 118ms/step - accuracy: 0.5313 - loss: 0.6866
Epoch 4/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 102ms/step - accuracy: 0.5640 - loss: 0.6817
Epoch 5/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 99ms/step - accuracy: 0.5720 - loss: 0.6786
Epoch 6/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 7s 164ms/step - accuracy: 0.6080 - loss: 0.6625
Epoch 7/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 99ms/step - accuracy: 0.6287 - loss: 0.6532
Epoch 8/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 99ms/step - accuracy: 0.6479 - loss: 0.6291
Epoch 9/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 3s 99ms/step - accuracy: 0.6734 - loss: 0.6132
Epoch 10/10
32/32 ━━━━━━━━━━━━━━━━━━━━ 8s 177ms/step - accuracy: 0.6925 - loss: 0.6022
```

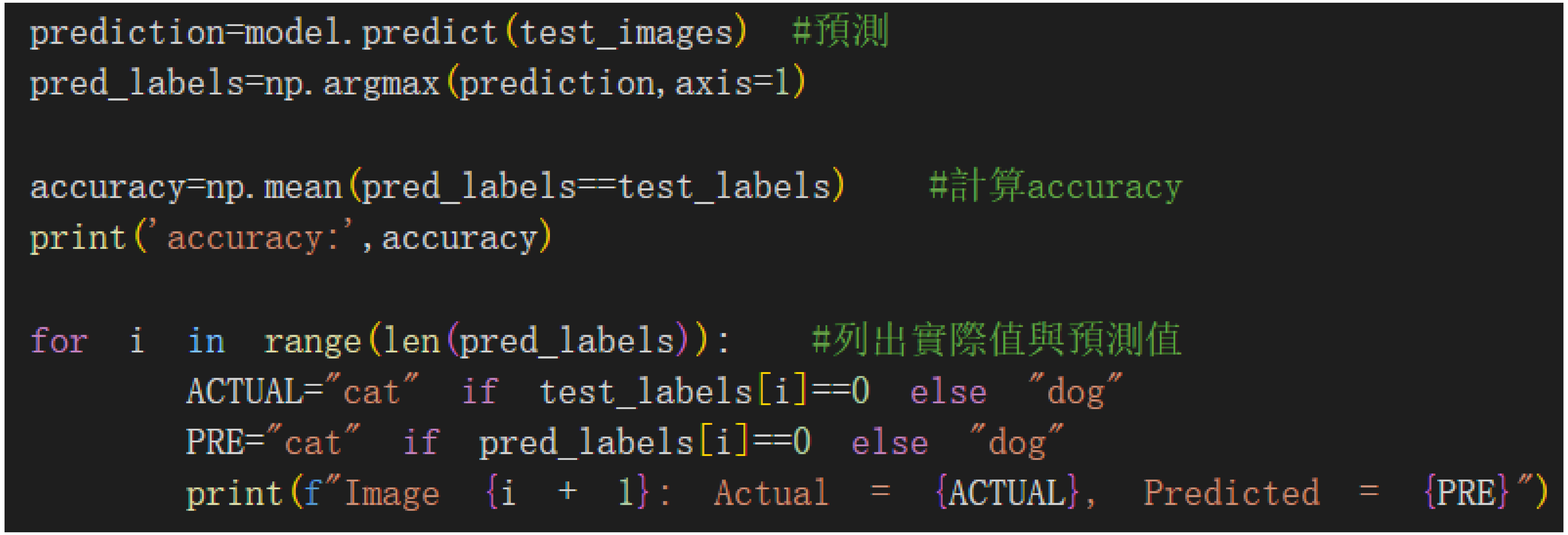
11.將模型訓練過程顯示為圖表



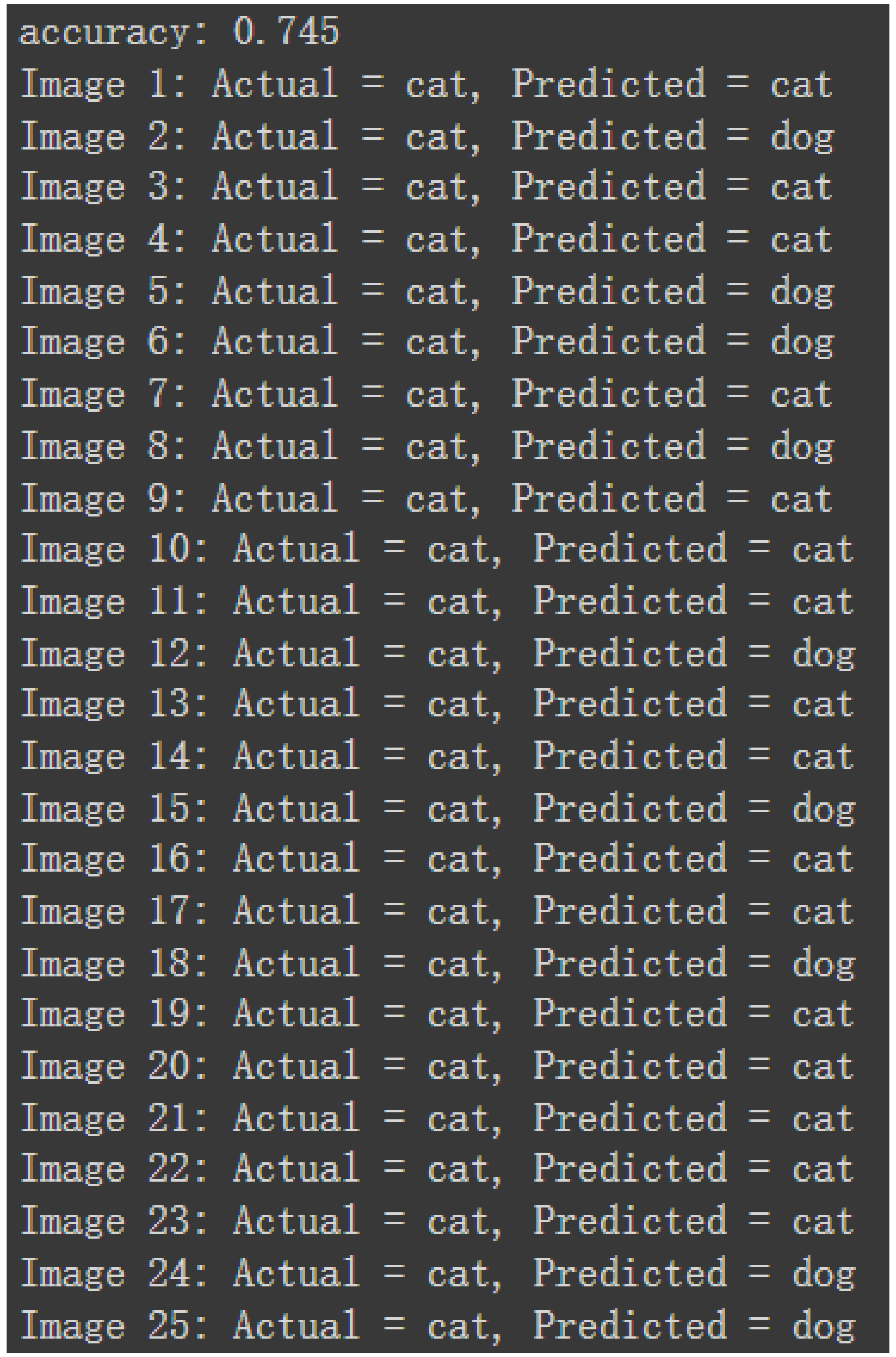
```
plt.title('train_loss')  #模型概況
plt.ylabel('loss')
plt.xlabel('epoch')
plt.plot(history.history['loss'])
```



12.加入測試集預測，並將結果列出

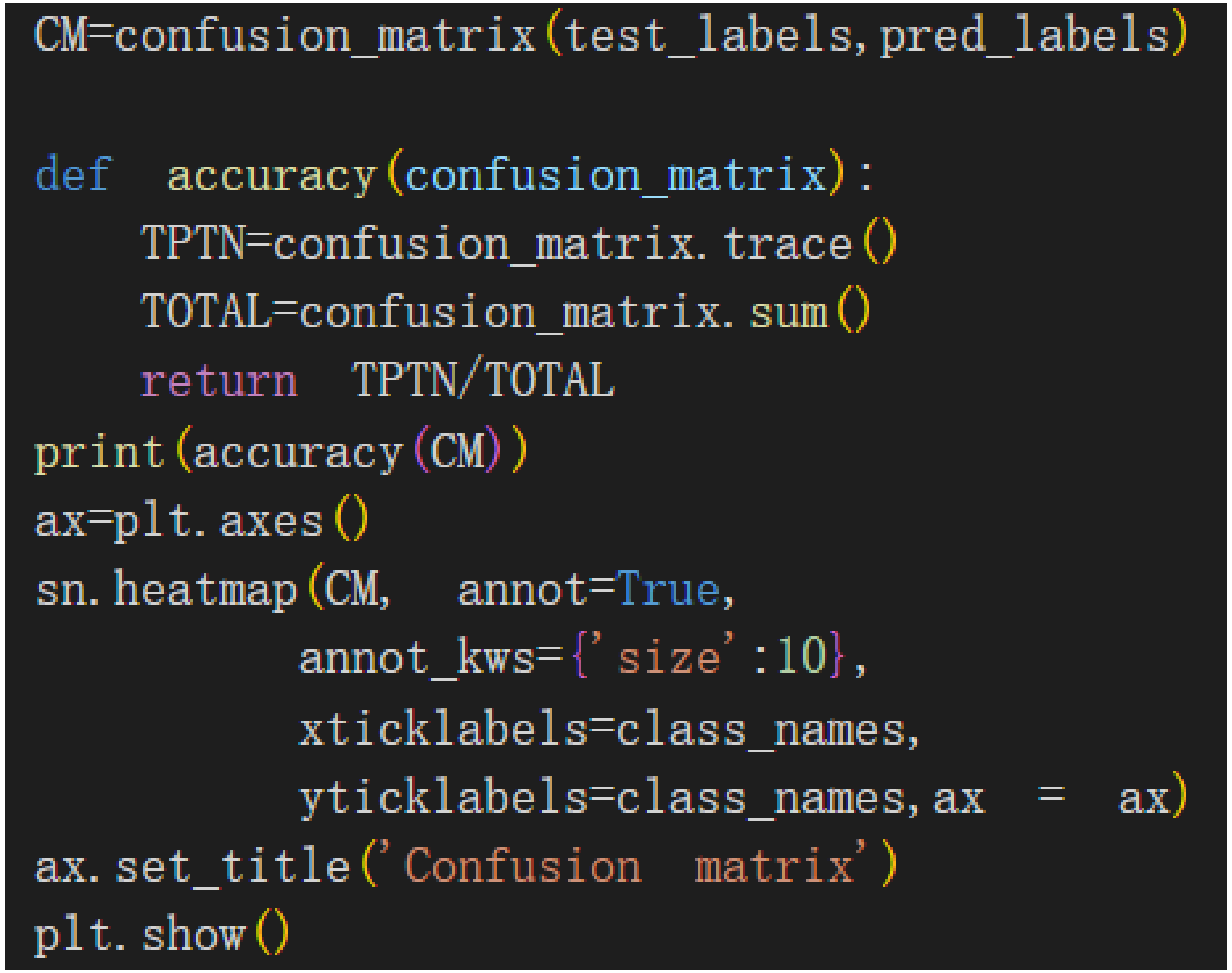


```
prediction=model.predict(test_images)  #預測
pred_labels=np.argmax(prediction,axis=1)

accuracy=np.mean(pred_labels==test_labels)    #計算accuracy
print('accuracy:',accuracy)

for i in range(len(pred_labels)):    #列出實際值與預測值
    ACTUAL="cat" if test_labels[i]==0 else "dog"
    PRE="cat" if pred_labels[i]==0 else "dog"
    print(f"Image {i + 1}: Actual = {ACTUAL}, Predicted = {PRE}")
```

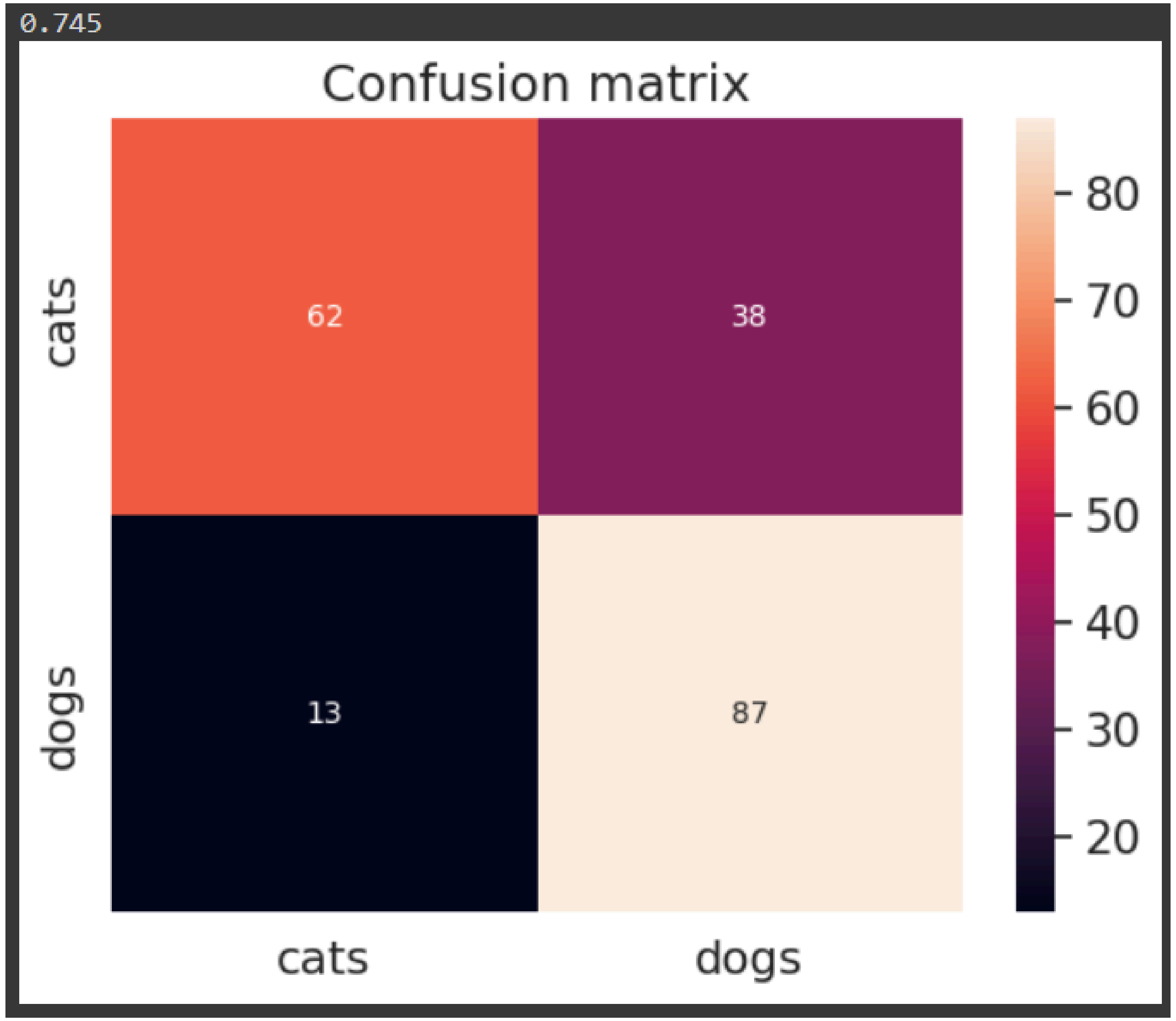


```
accuracy: 0.745
Image 1: Actual = cat, Predicted = cat
Image 2: Actual = cat, Predicted = dog
Image 3: Actual = cat, Predicted = cat
Image 4: Actual = cat, Predicted = cat
Image 5: Actual = cat, Predicted = dog
Image 6: Actual = cat, Predicted = dog
Image 7: Actual = cat, Predicted = cat
Image 8: Actual = cat, Predicted = dog
Image 9: Actual = cat, Predicted = cat
Image 10: Actual = cat, Predicted = cat
Image 11: Actual = cat, Predicted = cat
Image 12: Actual = cat, Predicted = dog
Image 13: Actual = cat, Predicted = cat
Image 14: Actual = cat, Predicted = cat
Image 15: Actual = cat, Predicted = dog
Image 16: Actual = cat, Predicted = cat
Image 17: Actual = cat, Predicted = cat
Image 18: Actual = cat, Predicted = dog
Image 19: Actual = cat, Predicted = cat
Image 20: Actual = cat, Predicted = cat
Image 21: Actual = cat, Predicted = cat
Image 22: Actual = cat, Predicted = cat
Image 23: Actual = cat, Predicted = cat
Image 24: Actual = cat, Predicted = dog
Image 25: Actual = cat, Predicted = dog
```

13.製作混淆矩陣



```
CM=confusion_matrix(test_labels,pred_labels)

def  accuracy(confusion_matrix):
    TPTN=confusion_matrix.trace()
    TOTAL=confusion_matrix.sum()
    return  TPTN/TOTAL
print(accuracy(CM))
ax=plt.axes()
sn.heatmap(CM,  annot=True,
          annot_kws={'size':10},
          xticklabels=class_names,
          yticklabels=class_names,ax  =  ax)
ax.set_title('Confusion  matrix')
plt.show()
```



```
0.745
```

THANKS