自然圖像辨識

07360532 陳威諭

07360726 劉沛綸

製作動機

探討CNN圖像辨識的深淺層數差異

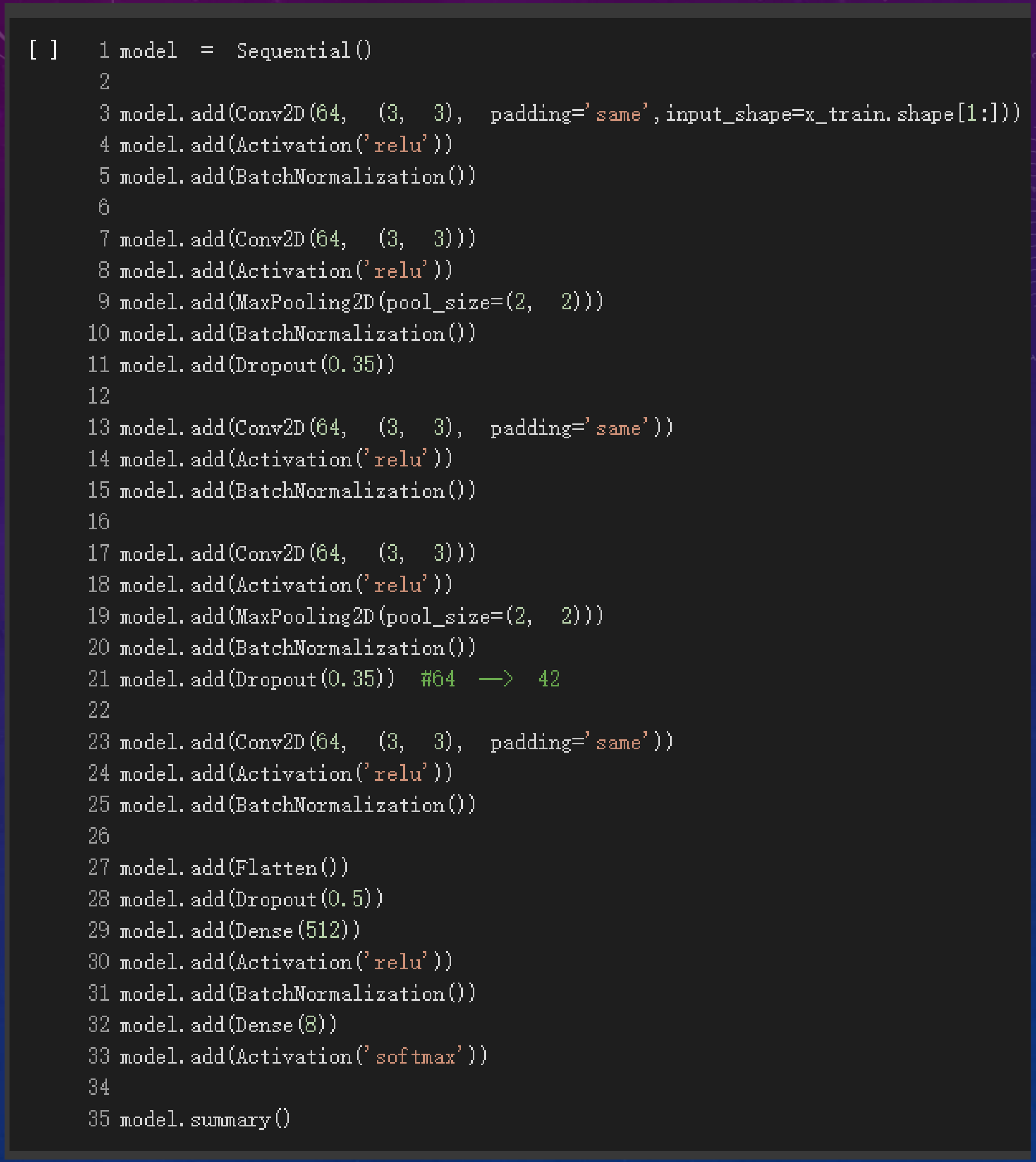
使用工具

- 模型架構採用 CNN
- 使用 Google Colab 實作
- 資料集來自 Kaggle

實作目錄

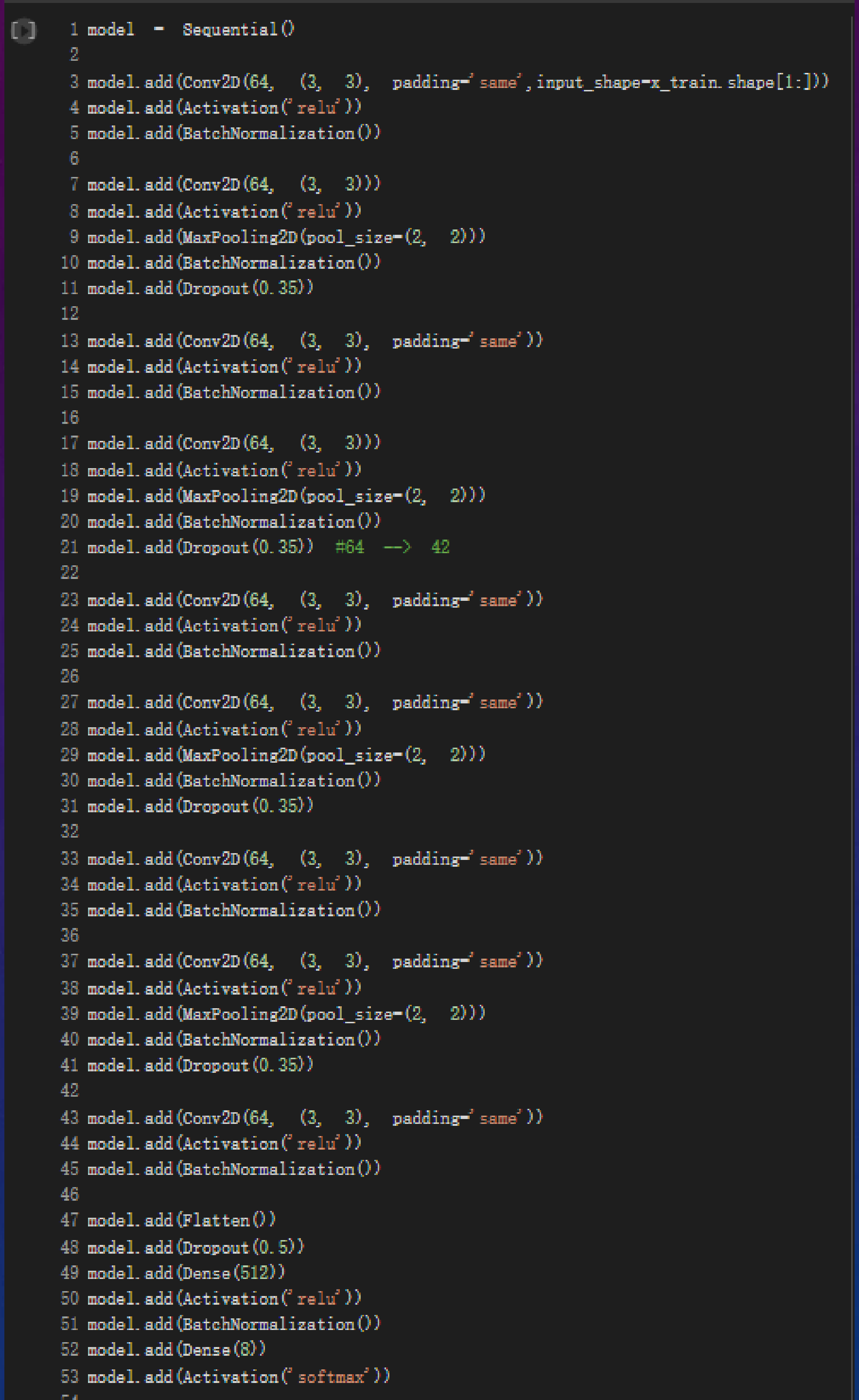
- 模型建立
- 訓練模型
- 訓練曲線
- 準確率與損失率
- 預測結果

模型建立

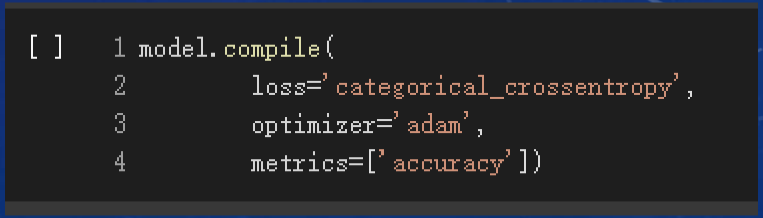


```python
model = Sequential()

model.add(Conv2D(64, (3, 3), padding='same',input_shape=x_train.shape[1:]))
model.add(Activation('relu'))
model.add(BatchNormalization())

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.35))

model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(BatchNormalization())

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.35))  #64 --> 42

model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(BatchNormalization())

model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(512))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(Dense(8))
model.add(Activation('softmax'))

model.summary()
```

模型建立

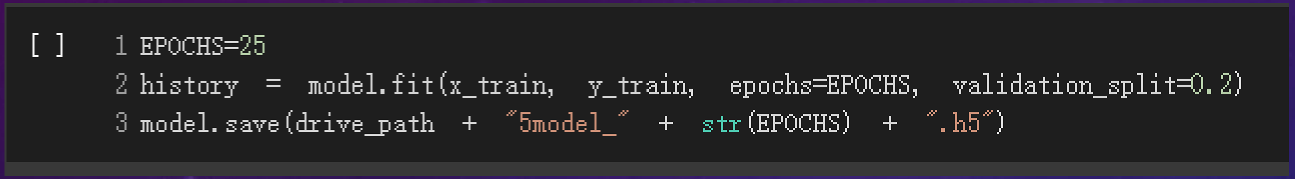


```
model = Sequential()

model.add(Conv2D(64, (3, 3), padding='same',input_shape=x_train.shape[1:]))
model.add(Activation('relu'))
model.add(BatchNormalization())

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.35))

model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(BatchNormalization())

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.35))  #64 --> 42

model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(BatchNormalization())

model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.35))

model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(BatchNormalization())

model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.35))

model.add(Conv2D(64, (3, 3), padding='same'))
model.add(Activation('relu'))
model.add(BatchNormalization())

model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(512))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(Dense(8))
model.add(Activation('softmax'))
```

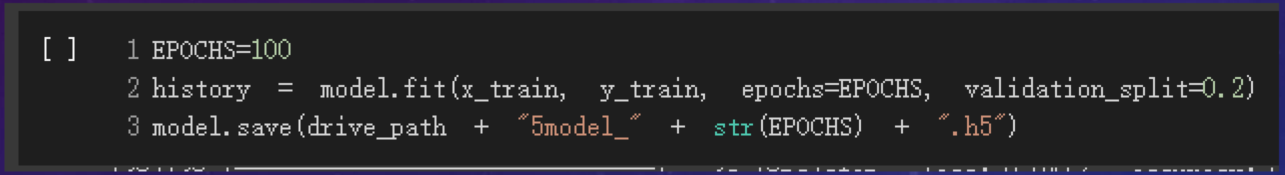


```
model.compile(
        loss='categorical_crossentropy',
        optimizer='adam',
        metrics=['accuracy'])
```

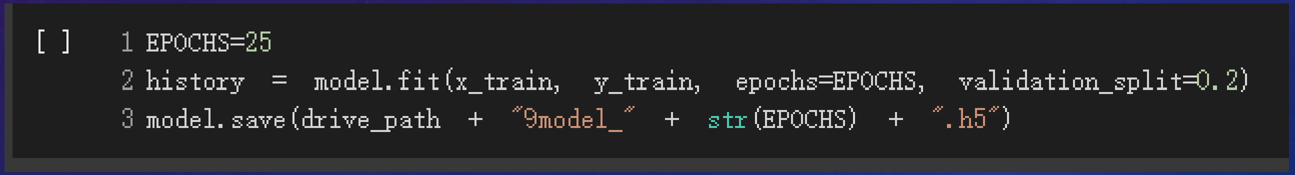
訓練模型



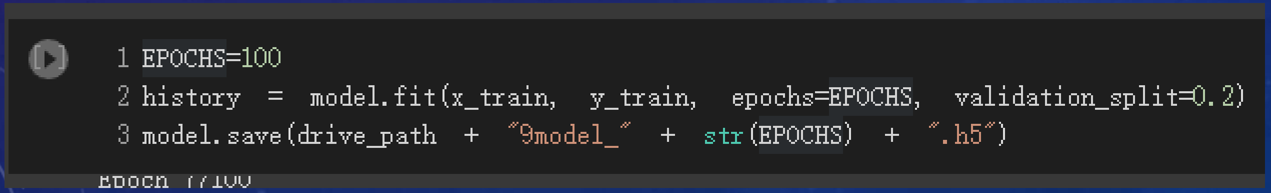
```
EPOCHS=25
history = model.fit(x_train, y_train, epochs=EPOCHS, validation_split=0.2)
model.save(drive_path + "5model_" + str(EPOCHS) + ".h5")
```



```
EPOCHS=100
history = model.fit(x_train, y_train, epochs=EPOCHS, validation_split=0.2)
model.save(drive_path + "5model_" + str(EPOCHS) + ".h5")
```



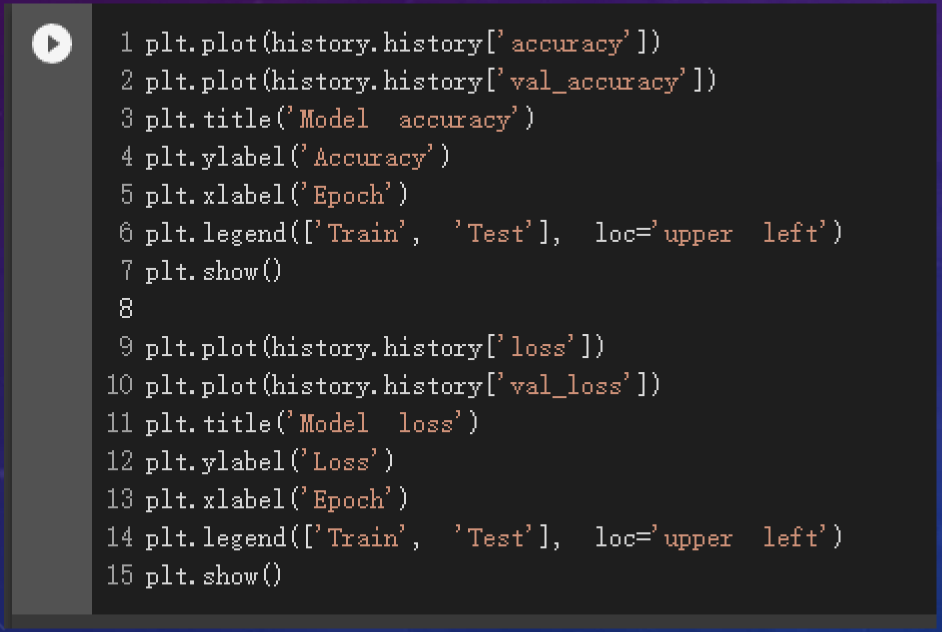
```
EPOCHS=25
history = model.fit(x_train, y_train, epochs=EPOCHS, validation_split=0.2)
model.save(drive_path + "9model_" + str(EPOCHS) + ".h5")
```



```
EPOCHS=100
history = model.fit(x_train, y_train, epochs=EPOCHS, validation_split=0.2)
model.save(drive_path + "9model_" + str(EPOCHS) + ".h5")
```

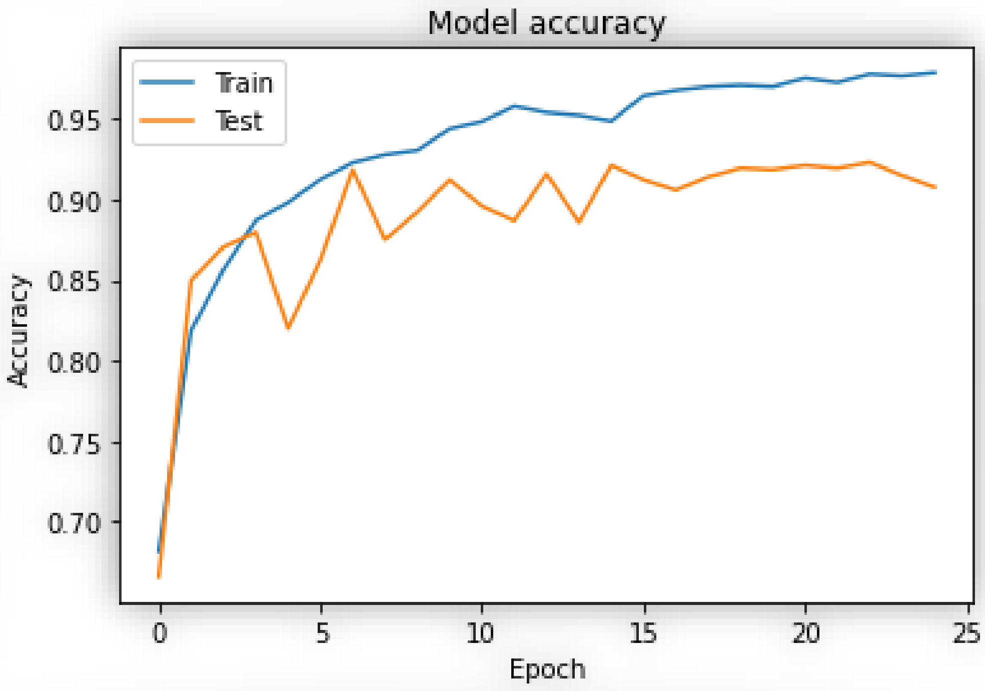
```
Epoch 1/100
```

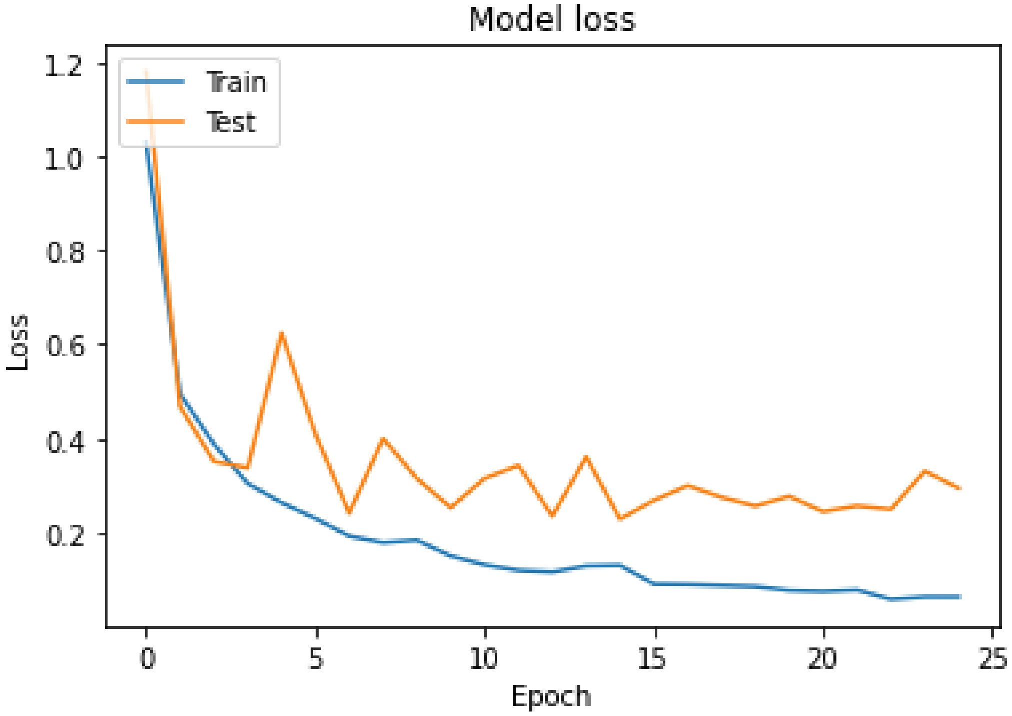
訓練曲線



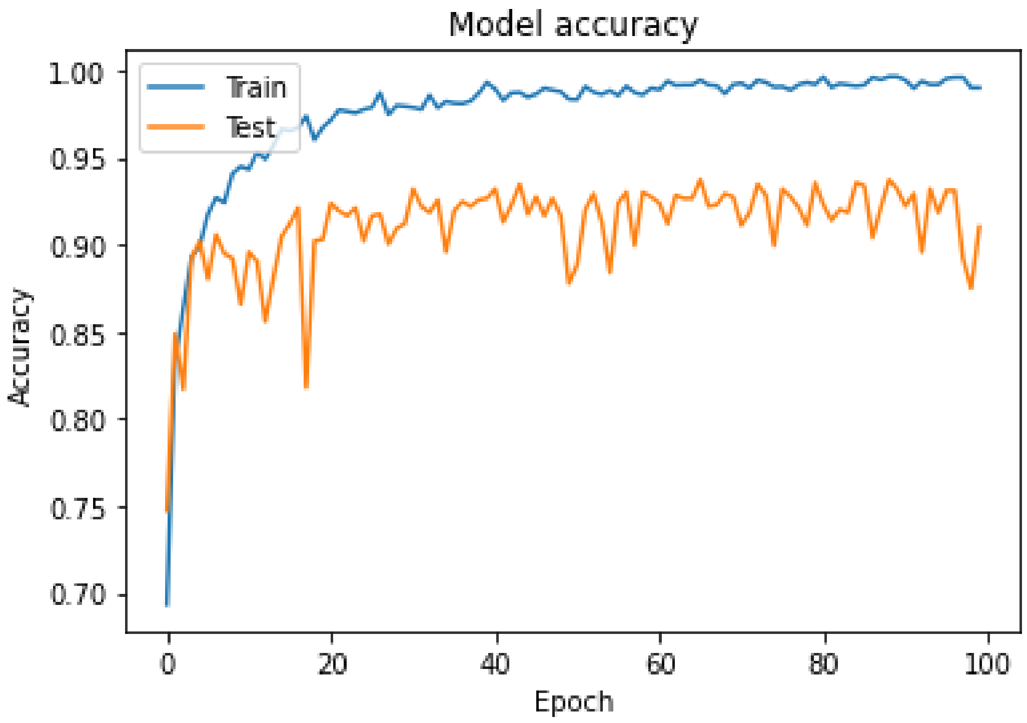
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```

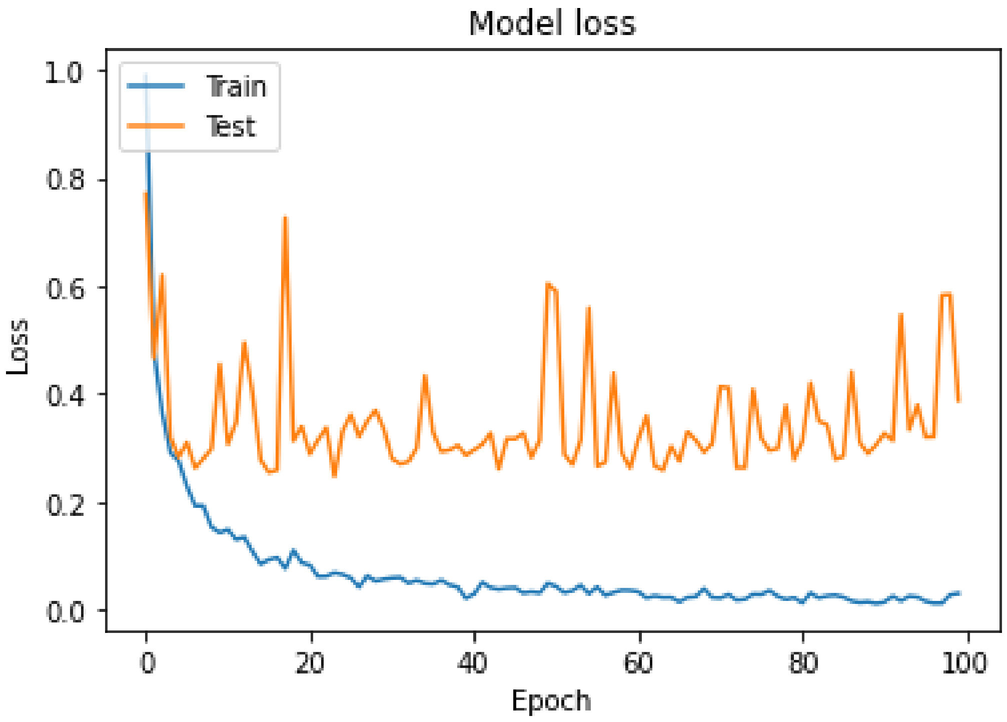
訓練曲線（5層CNN網路 訓練25圈）



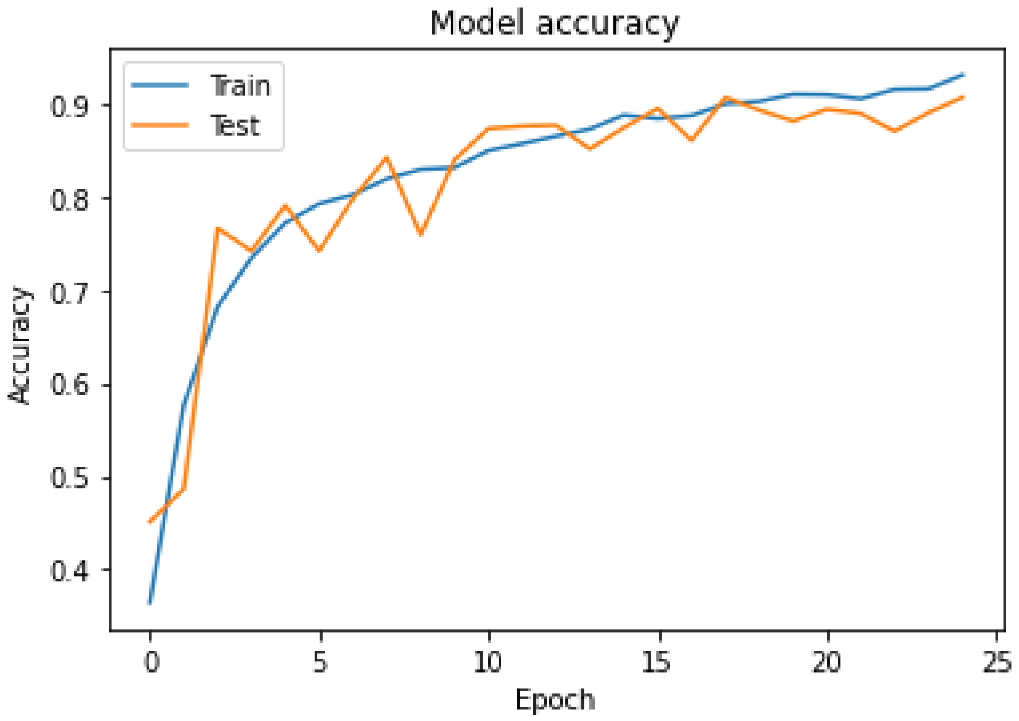


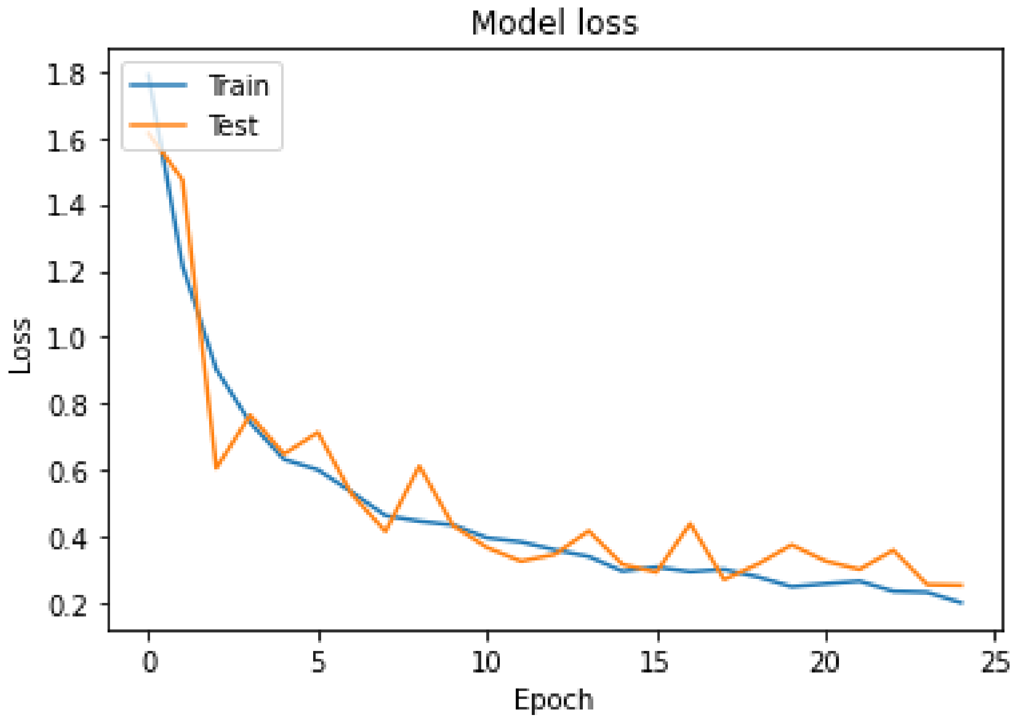
訓練曲線（5層CNN網路 訓練100圈）



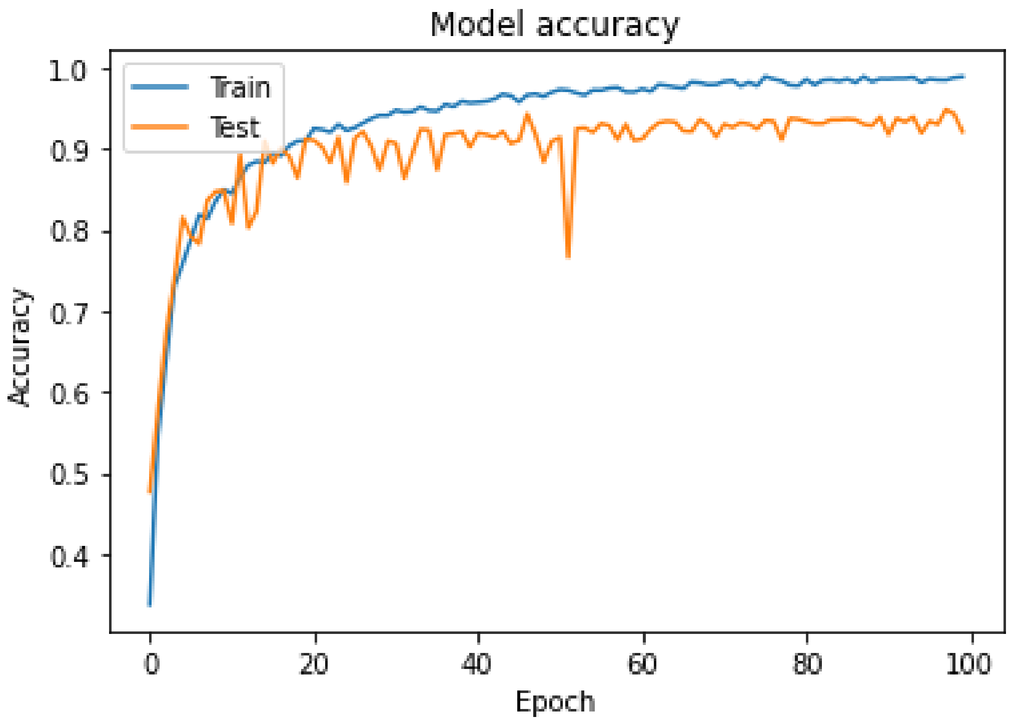


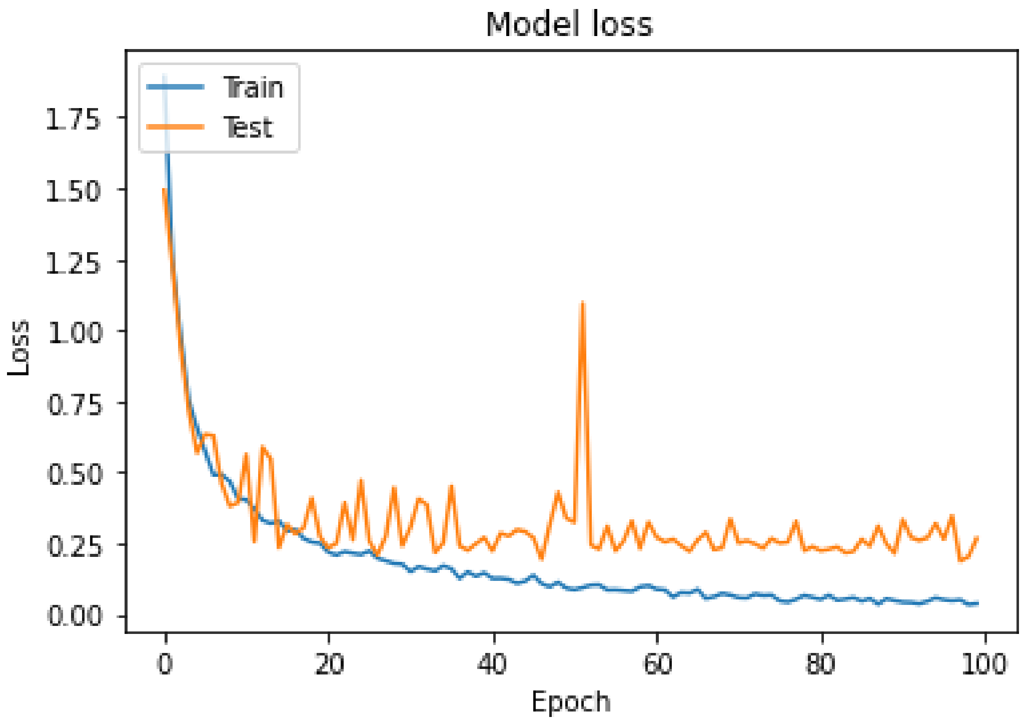
訓練曲線（9層CNN網路 訓練25圈）



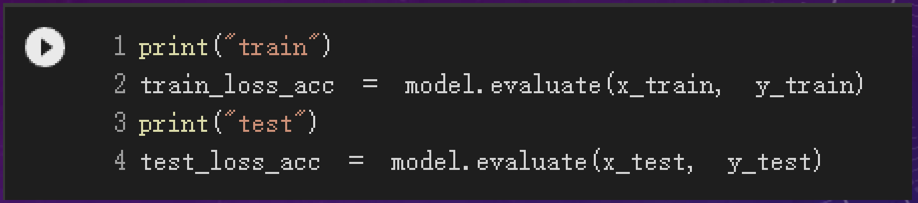


訓練曲線（9層CNN網路 訓練100圈）

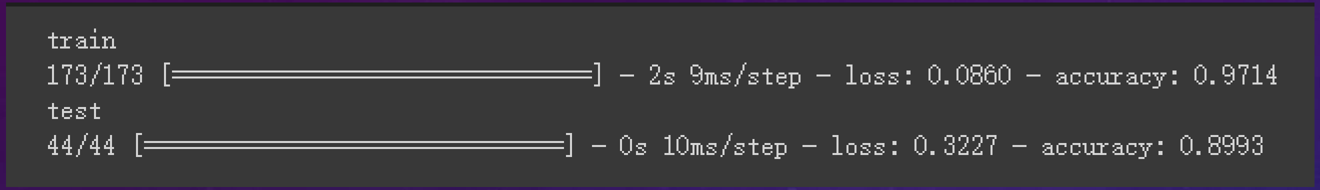




準確率與損失率

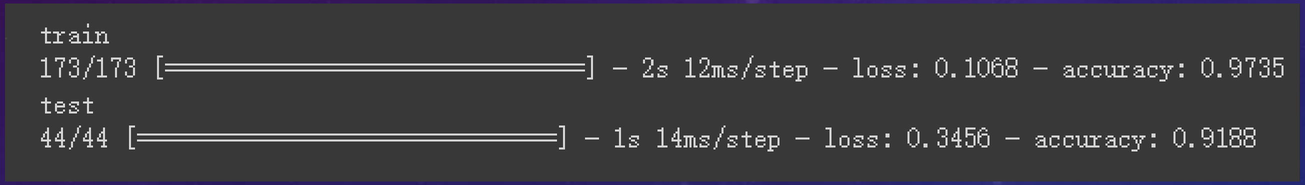


```
print("train")
train_loss_acc = model.evaluate(x_train, y_train)
print("test")
test_loss_acc = model.evaluate(x_test, y_test)
```



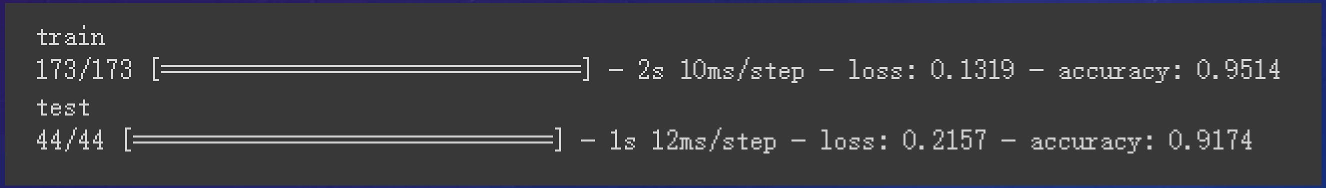
```
train
173/173 [==============================] - 2s 9ms/step - loss: 0.0860 - accuracy: 0.9714
test
44/44 [==============================] - 0s 10ms/step - loss: 0.3227 - accuracy: 0.8993
```

5層CNN網路 訓練25圈



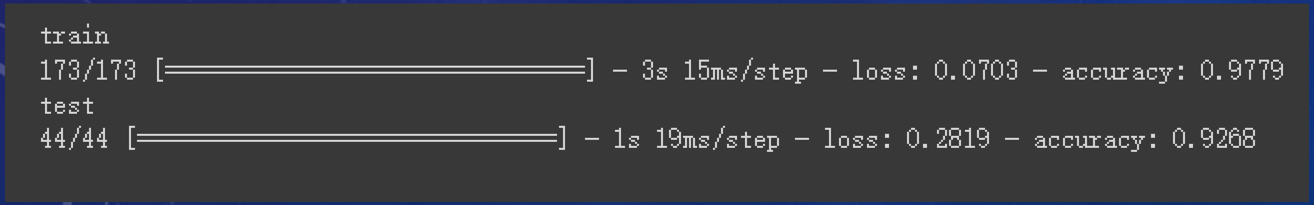
```
train
173/173 [==============================] - 2s 12ms/step - loss: 0.1068 - accuracy: 0.9735
test
44/44 [==============================] - 1s 14ms/step - loss: 0.3456 - accuracy: 0.9188
```

5層CNN網路 訓練100圈



```
train
173/173 [==============================] - 2s 10ms/step - loss: 0.1319 - accuracy: 0.9514
test
44/44 [==============================] - 1s 12ms/step - loss: 0.2157 - accuracy: 0.9174
```

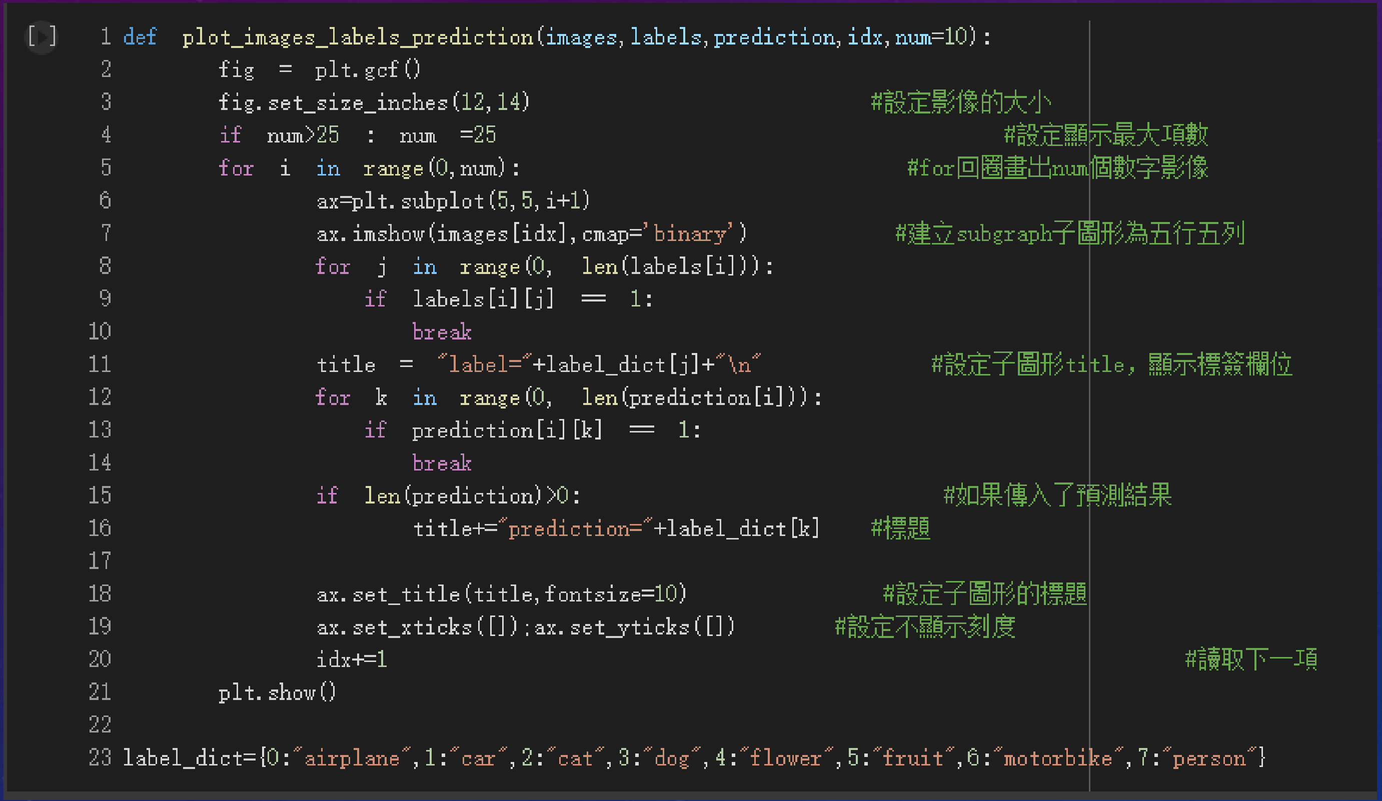
9層CNN網路 訓練25圈



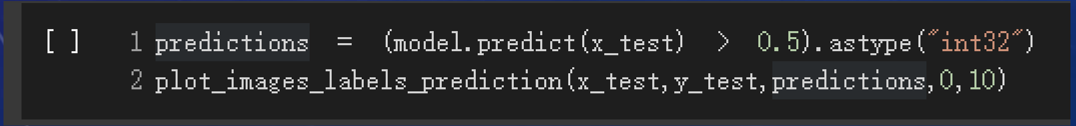
```
train
173/173 [==============================] - 3s 15ms/step - loss: 0.0703 - accuracy: 0.9779
test
44/44 [==============================] - 1s 19ms/step - loss: 0.2819 - accuracy: 0.9268
```

9層CNN網路 訓練100圈

預測結果

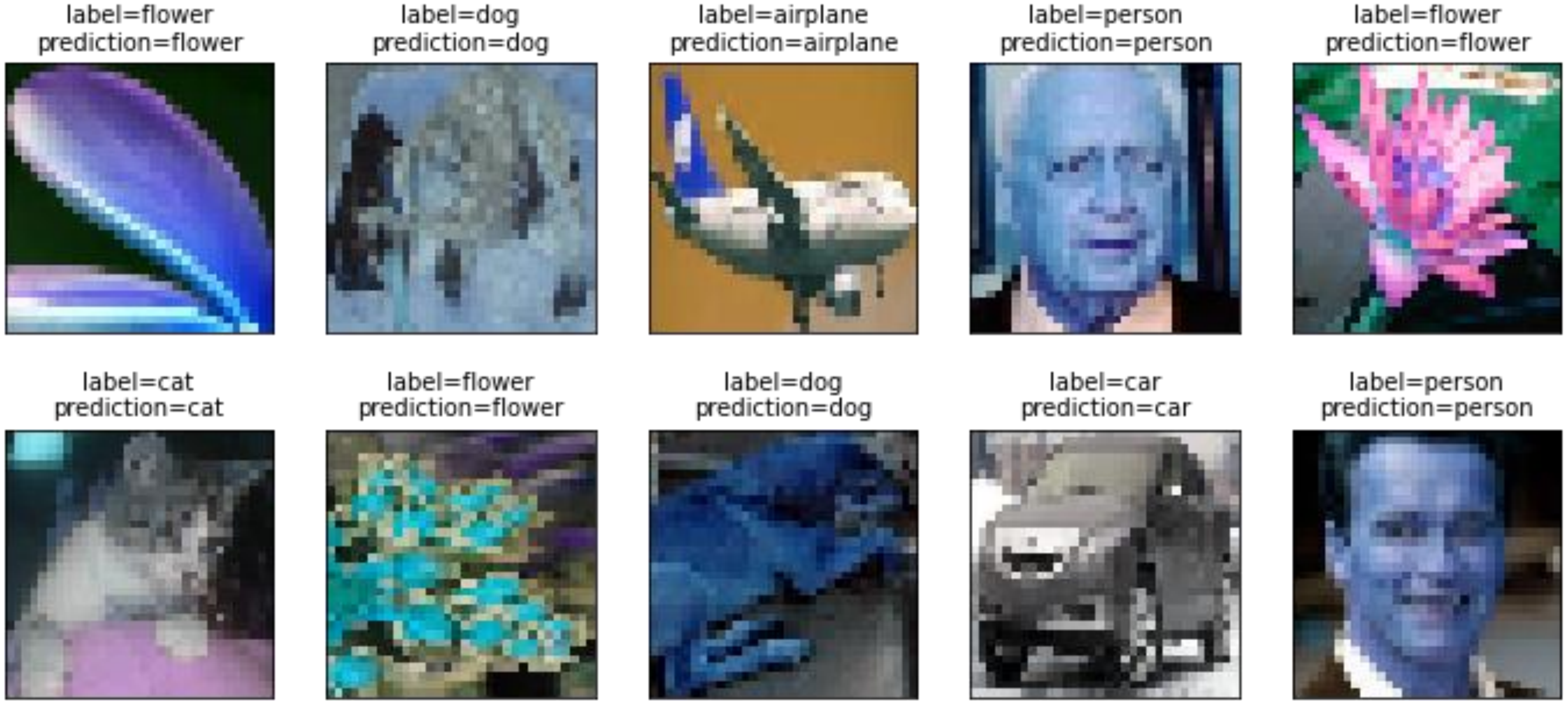


```
[ ]
def  plot_images_labels_prediction(images,labels,prediction,idx,num=10):
    fig  =  plt.gcf()
    fig.set_size_inches(12,14)                              #設定影像的大小
    if  num>25  :  num  =25                                           #設定顯示最大項數
    for  i  in  range(0,num):                               #for回圈畫出num個數字影像
        ax=plt.subplot(5,5,i+1)
        ax.imshow(images[idx],cmap='binary')              #建立subgraph子圖形為五行五列
        for  j  in  range(0,  len(labels[i])):
            if  labels[i][j]  ==  1:
                break
        title  =  "label="+label_dict[j]+"\n"               #設定子圖形title, 顯示標簽欄位
        for  k  in  range(0,  len(prediction[i])):
            if  prediction[i][k]  ==  1:
                break
        if  len(prediction)>0:                              #如果傳入了預測結果
                title+="prediction="+label_dict[k]     #標題

        ax.set_title(title,fontsize=10)                     #設定子圖形的標題
        ax.set_xticks([]);ax.set_yticks([])        #設定不顯示刻度
        idx+=1                                                        #讀取下一項
    plt.show()

label_dict={0:"airplane",1:"car",2:"cat",3:"dog",4:"flower",5:"fruit",6:"motorbike",7:"person"}
```

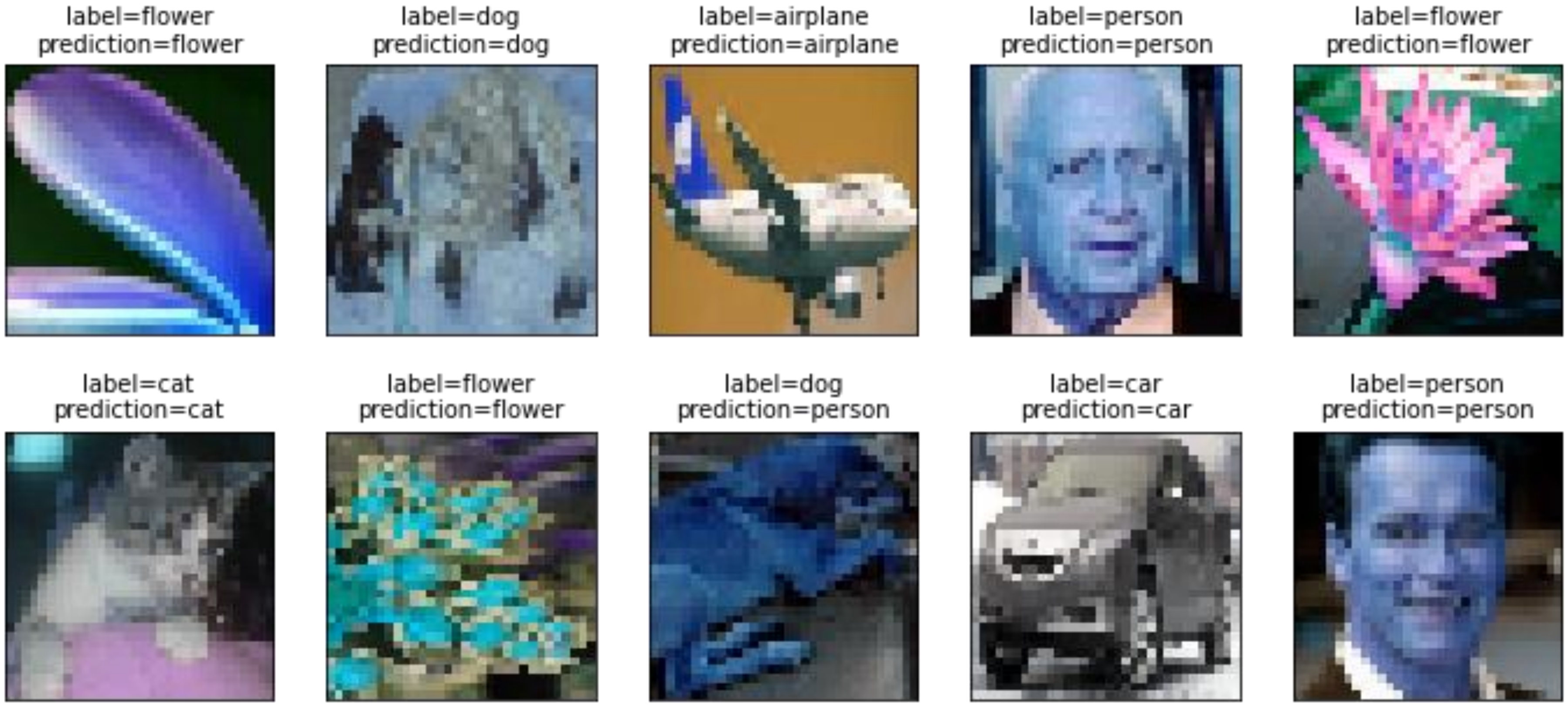


```
[ ]
predictions  =  (model.predict(x_test)  >  0.5).astype("int32")
plot_images_labels_prediction(x_test,y_test,predictions,0,10)
```

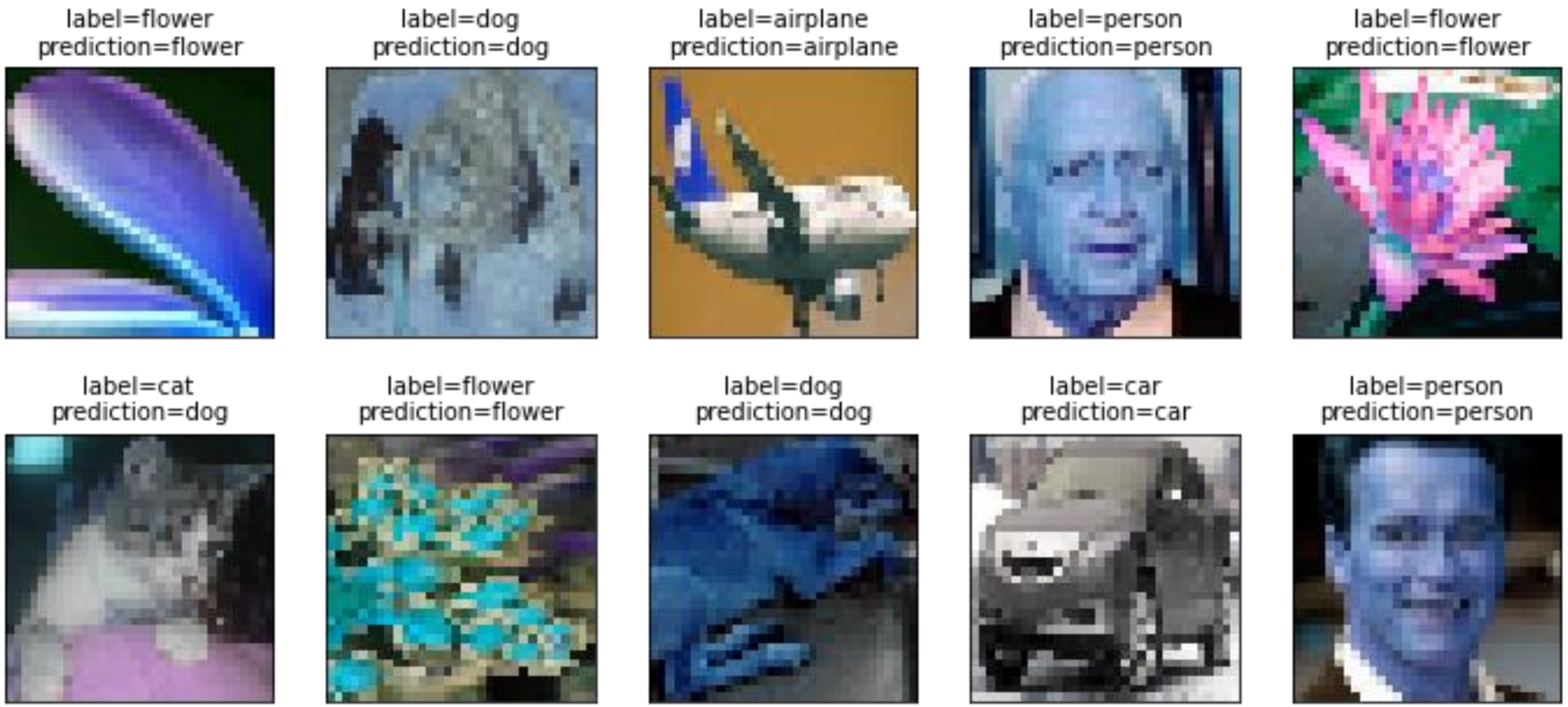
預測結果（5層CNN網路 訓練25圈）



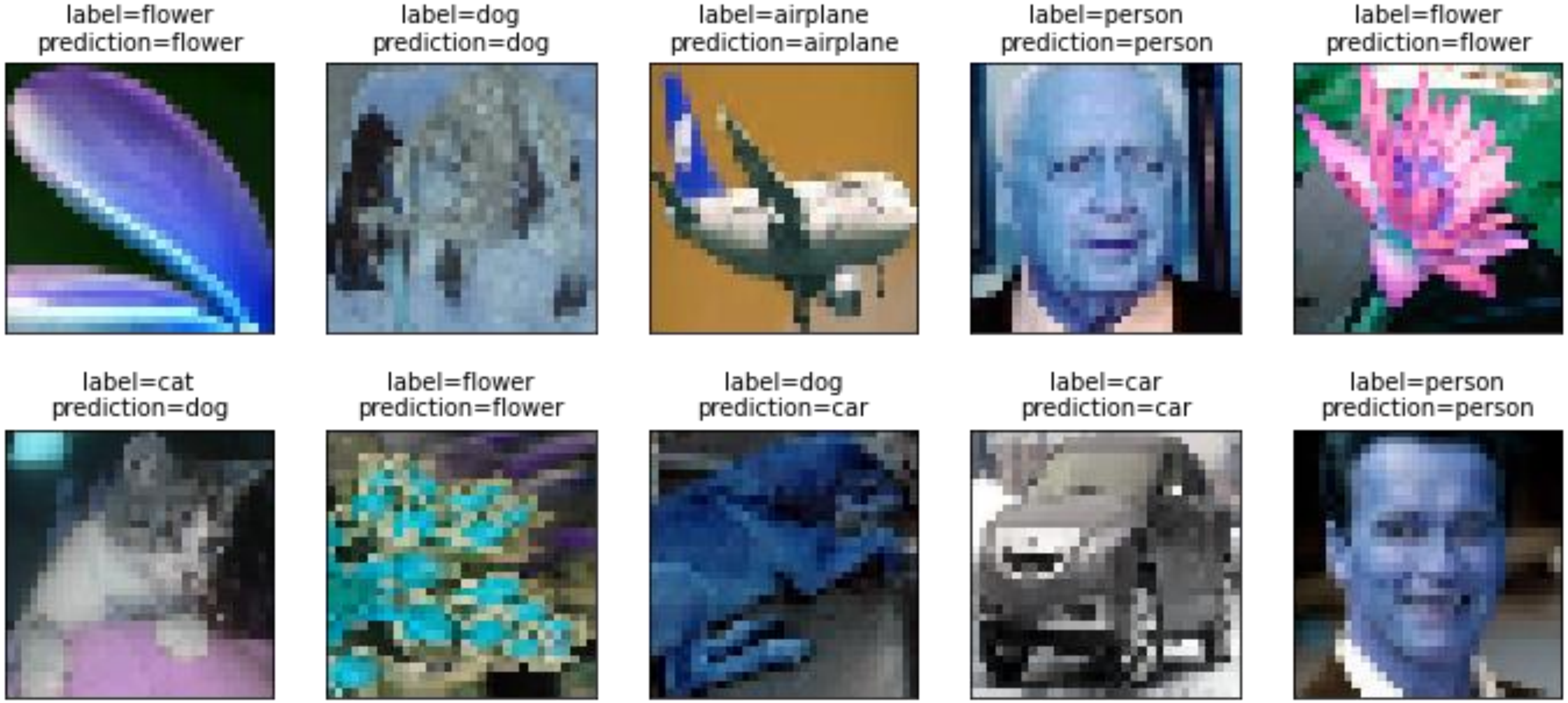
預測結果（5層CNN網路 訓練100圈）



預測結果（9層CNN網路 訓練25圈）



預測結果（9層CNN網路 訓練100圈）



結論

測試過相同圈數與不同層數的比較後，我們認為較多層的CNN模型能夠提高準確率。同時發現到在卷基層較多時，8種類別中僅在貓狗辨識的部分出錯，其餘6種類別的辨識效果良好，我們認為是由於增加了池化層後減少了特徵擷取，因此對於較接近的物件上特徵數不足而造成了辨識的困難。

分工表

大家一起做

07360532 陳威諭

07360726 劉沛綸