一、 中文手寫辨識準確率及損失率

Test loss ,Test accuracy 截圖:

```
Test loss: 0.2370680868625641
Test accuracy: 0.9682353138923645
```

二、 Source code 之逐行解釋

第一步：import OS、keras、numpy 等模組：

```
import os
import random
import numpy as np
import keras
from PIL import Image
from keras.models import Sequential
from keras.layers import Dense,Dropout,Flatten
from keras.layers import Conv2D,MaxPooling2D
from keras.models import load_model
from keras.utils import np_utils
```

第二步：資料處理前函式：

```
def data_x_y_preprocess(datapath):
    img_row,img_col = 28,28  定義圖片大小
    datapath=datapath  訓練資料路徑
    data_x=np.zeros((28,28)).reshape(1,28,28) 儲存圖片
    pictureCount=0 紀錄圖片張數，初始化為 0
    data_y=[]  紀錄 label
    num_class=10  數字種類有 10
    for root,dirs,files in os.walk(datapath):讀取所有檔案
        for f in files:
            label=int(root.split("\\")[2]) 取得 label
            data_y.append(label)
            fullpath=os.path.join(root,f) 取得檔案路徑
            img=Image.open(fullpath) 開啟 img
            img=(np.array(img)/255).reshape(1,28,28) 讀取資料並做正規劃跟 reshape
            data_x=np.vstack((data_x,img))將資料堆疊起來。
            pictureCount+=1
    data_x=np.delete(data_x,[0],0)刪除一開始宣告的 np.zeros
    data_x=data_x.reshape(pictureCount,img_row,img_col,1) 調整資料格式
                                                          （圖片張數,row,col,1）
    data_y=np_utils.to_categorical(data_y,num_class)
    return data_x,data_y   將 label 改成 one-hot encoding
```

第三步：建立 CNN 模型：

```
model=Sequential() 建立模型
model.add(Conv2D(32,kernel_size=(3,3),activation='relu',input_shape=(28,28,1)))建立卷積層
model.add(MaxPooling2D(pool_size=(2,2))) 建立池化層
model.add(Conv2D(64,(3,3),activation='relu')) 建立卷積層
model.add(MaxPooling2D(pool_size=(2,2))) 建立池化層
model.add(Dropout(0.1)) Dropout 層斷開輸入神經元，用來防止過度擬合。斷開比例 0.1。
model.add(Flatten()) 把多維輸入一維化。
model.add(Dropout(0.1)) Dropout 層斷開輸入神經元，用來防止過度擬合。斷開比例 0.1。
model.add(Dense(128,activation='relu'))全連接層，128 個 output。
model.add(Dropout(0.25)) Dropout 層斷開輸入神經元，用來防止過度擬合。斷開比例 0.25。
model.add(Dense(units=10,activation='softmax'))將結果分成 10 類。
```

第四步：輸入訓練與測試資料：

```
train_x, train_y = data_x_y_preprocess("train_image")
test_x , test_y = data_x_y_preprocess("test_image")
```

第五步：模組編譯與訓練：

model.compile 進行編譯，選擇損失函數、優化方法及成效衡量方式。

進行訓練，訓練過程中會存在 train_history 變數中。

顯示 Test Loss 跟 Test Accuracy。

```
model.compile(loss="categorical_crossentropy",
              optimizer="adam",
              metrics=['accuracy'])
train_history=model.fit(train_x,train_y,
                        batch_size=32,
                        epochs=150,
                        verbose=1,
                        validation_split=0.1)
score=model.evaluate(test_x,test_y,verbose=0)
print("Test loss:",score[0])
print("Test accuracy:",score[1])
```