作業三 實作keras 模式解MINST 手寫數字辨識問題

以類神經網路實作與測試一個手寫數字(0-9)辨識系統

以網路資料為學習訓練資料，以自己學號數字部分自己繪圖依所需格式做為測試資料，輸出學號每一個數字的辨識結果。

範例系統參考網址

https://ithelp.ithome.com.tw/articles/10191404

作業繳交時需附上每一個學號圖檔的驗證結果。並附上自己的學號圖檔，以供助教驗證。

範例程式

# 導入函式庫

import numpy as np

from keras.models import Sequential

from keras.datasets import mnist

from keras.layers import Dense, Dropout, Activation, Flatten

from keras.utils import np\_utils # 用來後續將 label 標籤轉為 one-hot-encoding

from matplotlib import pyplot as plt

# 載入 MNIST 資料庫的訓練資料，並自動分為『訓練組』及『測試組』

(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()

# 建立簡單的線性執行的模型

model = Sequential()

# Add Input layer, 隱藏層(hidden layer) 有 256個輸出變數

model.add(Dense(units=256, input\_dim=784, kernel\_initializer='normal', activation='relu'))

# Add output layer

model.add(Dense(units=10, kernel\_initializer='normal', activation='softmax'))

# 編譯: 選擇損失函數、優化方法及成效衡量方式

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

# 將 training 的 label 進行 one-hot encoding，例如數字 7 經過 One-hot encoding 轉換後是 0000001000，即第7個值為 1

y\_TrainOneHot = np\_utils.to\_categorical(y\_train)

y\_TestOneHot = np\_utils.to\_categorical(y\_test)

# 將 training 的 input 資料轉為2維

X\_train\_2D = X\_train.reshape(60000, 28\*28).astype('float32')

X\_test\_2D = X\_test.reshape(10000, 28\*28).astype('float32')

x\_Train\_norm = X\_train\_2D/255

x\_Test\_norm = X\_test\_2D/255

# 進行訓練, 訓練過程會存在 train\_history 變數中

train\_history = model.fit(x=x\_Train\_norm, y=y\_TrainOneHot, validation\_split=0.2, epochs=10, batch\_size=800, verbose=2)

# 顯示訓練成果(分數)

scores = model.evaluate(x\_Test\_norm, y\_TestOneHot)

print()

print("\t[Info] Accuracy of testing data = {:2.1f}%".format(scores[1]\*100.0))

# 預測(prediction)

X = x\_Test\_norm[0:10,:]

predictions = np.argmax(model.predict(X), axis=-1)

# get prediction result

print(predictions)

# 顯示 第一筆訓練資料的圖形，確認是否正確

plt.imshow(X\_test[0])

plt.show()

plt.plot(train\_history.history['loss'])

plt.plot(train\_history.history['val\_loss'])

plt.title('Train History')

plt.ylabel('loss')

plt.xlabel('Epoch')

plt.legend(['loss', 'val\_loss'], loc='upper left')

plt.show()