資料探勘研究與實務期末專題-

中文手寫數字辨識

第三組

0753431 吳伯揚　0753425 李嘉晨

0753440 吳肇堉　0753423 劉奕辰

目錄

一、主題與動機

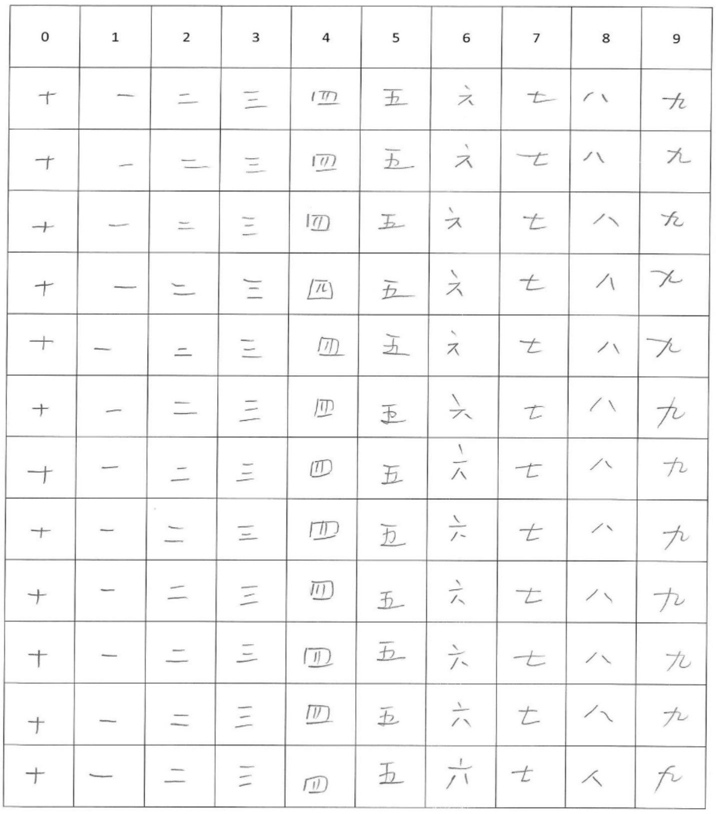
- 主題：中文手寫數字辨識。
- 動機：現今數字圖像辨識技術已相當普及，然而大部分開發出的都是以辨識阿拉伯、羅馬數字等為主；為比較適用於西方國家的文字辨識系統。

  因此，我們決定使用 Keras 框架做出一款以辨識中文(國字)數字的模型。

二、資料集敘述

透過與同學、朋友進行合作提供手寫數字資料，匯集製作而成，資料範例如下：

訓練資料約 12383 個圖片檔(小格子)，驗證資料為 5308 個，為 70%:30%的比例。



■　訓練資料集

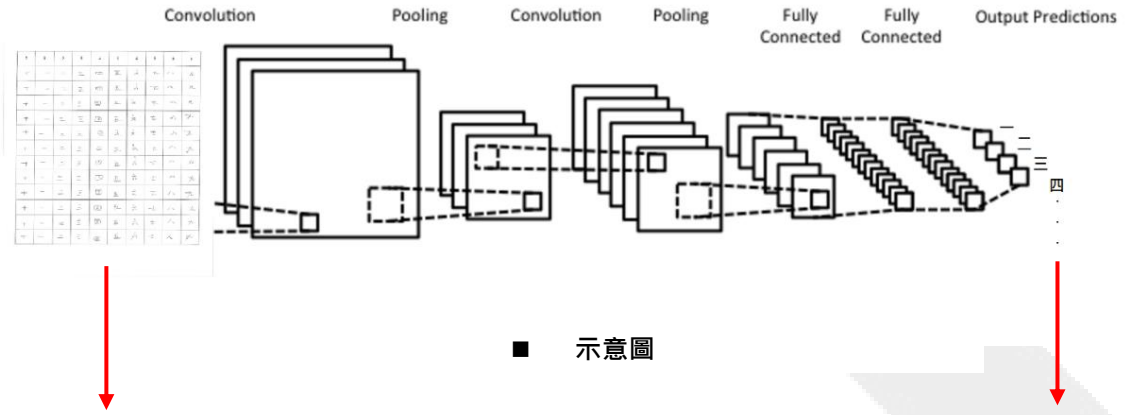
```
驗證檔案 12383
訓練檔案 5308
x_train_image: (12383, 190, 190)
y_train_label: (12383,)
x_img_test: (5308, 190, 190)
y_label_test: (5308,)
```

三、 分析工具

- 開發工具:Spyder。
- 開發環境:Python3.6。
- 使用技術:Keras、Pandas、Matplotlib 套件。

四、 實作與評估方法

- 實作模型:使用 CNN 模型。
- 實作方法:建立模型後，首先使用約 12400 筆訓練資料進行模型的訓練後，再對約 5300 筆的測試資料進行預測。
- 評估方法:使用 Accuracy、Loss、Confusion Matrix 等…對預測的結果進行評估，並將評估結果以視覺化的圖表呈現。



■ 示意圖

傳入手寫數字圖片

辨識結果(國字數字)

五、 流程

- 建立 CNN 模型，並設定相關參數。

1. **卷積層(Convolution)**:輸入的矩陣大小為 190x190，並產生 16 個 Filter，每個 Filter 大小為 5x5，使用 ReLU 當作激活函數。
2. **池化層(Pooling)**: 縮小尺寸，將萃取出的特徵矩陣降維成 5x5。
3. **第二個卷積層(Convolution)**:輸入的矩陣大小為 5x5，並產生 36 個 Filter，每個 Filter 大小為 5x5，使用 ReLU 當作激活函數。
4. **第二個池化層(Pooling)**:縮小尺寸，將萃取出的特徵矩陣降維成 2x2。
5. **Dropout 層**:隨機捨棄 25% 特徵，避免預測結果 Overfitting。
6. **平坦層(Flatten)**:將矩陣轉為一維陣列當作輸入。
7. **隱藏層**:隨機產生神經元。
8. 印出預測結果。

```
model.add(MaxPooling2D(pool_size=(5,5)))

model.add(Conv2D(filters=36,kernel_size=(5,5)
                 ,padding='same',activation='relu'))

model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(128,activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(10,activation='softmax'))

print(model.summary())

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

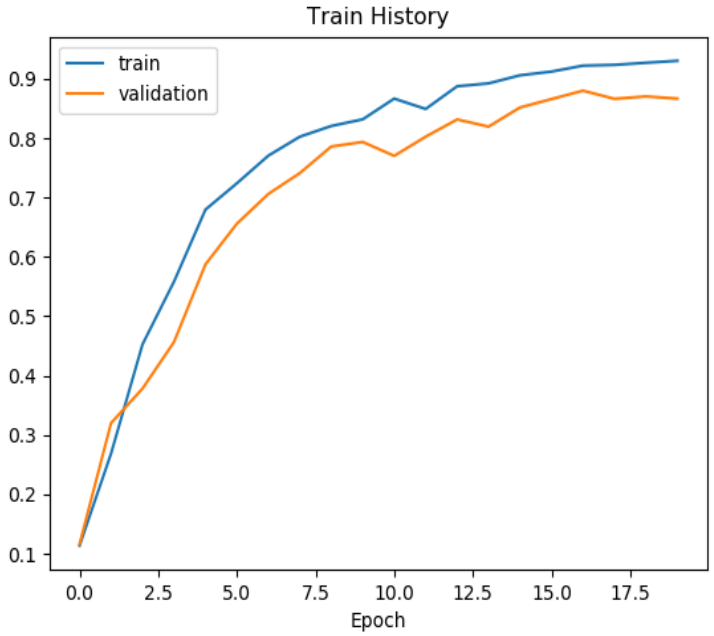
■　模型架構

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 190, 190, 16)      416
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 38, 38, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 38, 38, 36)        14436
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 19, 19, 36)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 19, 19, 36)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 12996)             0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               1663616
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
=================================================================
Total params: 1,679,758
Trainable params: 1,679,758
Non-trainable params: 0
```

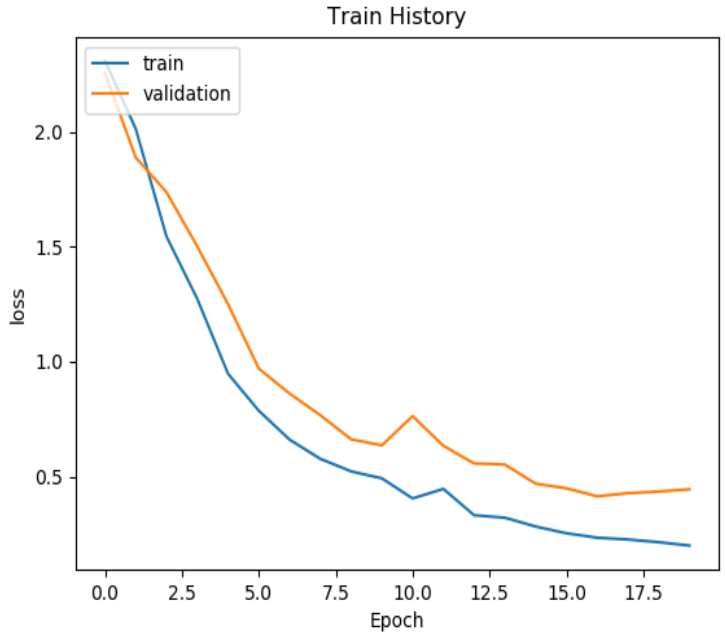
■　模型架構總覽

六、 分析結果與結論

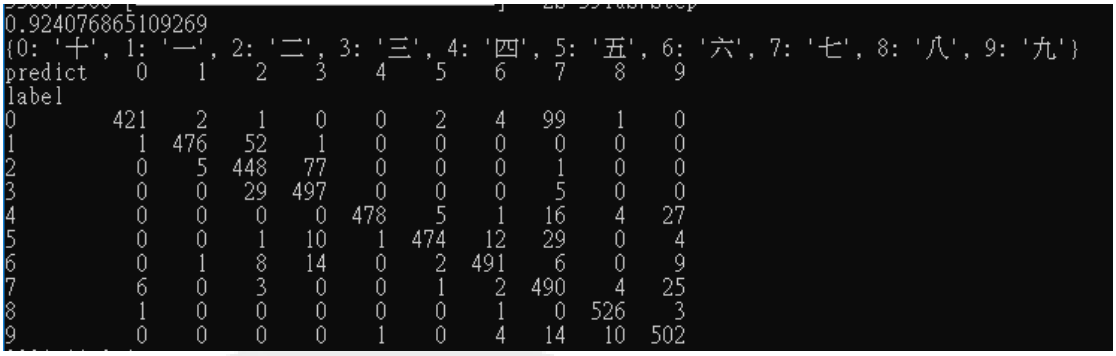
➢ 視覺化分析結果



■ Accuracy



■ Loss



0.924076865109269

{0: '十', 1: '一', 2: '二', 3: '三', 4: '四', 5: '五', 6: '六', 7: '七', 8: '八', 9: '九'}

predict label	0	1	2	3	4	5	6	7	8	9
0	421	2	1	0	0	2	4	99	1	0
1	1	476	52	1	0	0	0	0	0	0
2	0	5	448	77	0	0	0	1	0	0
3	0	0	29	497	0	0	0	5	0	0
4	0	0	0	0	478	5	1	16	4	27
5	0	0	1	10	1	474	12	29	0	4
6	0	1	8	14	0	2	491	6	0	9
7	6	0	3	0	0	1	2	490	4	25
8	1	0	0	0	0	0	1	0	526	3
9	0	0	0	0	1	0	4	14	10	502

■ Confusion Matrix

➢ 結論

- Accuracy 的數據圖中，可以發現 Epoch 越高的時候，不論是 train 或是 validation 都會有更高的 Accuracy。
- Loss 的數據圖中，隨著 Epoch 越高，train 跟 validation 都會有越來越低的 Loss。
- 最後，Confusion Matrix 中有比較特別的現象，我們可以發現 Model 最容易辨識錯的三個情形是：

  (1) 把十誤認成七

  (2) 把二誤認成三

  (3) 把一誤認成二

  以一般的觀察來說，這幾個錯誤的判斷都在於字體間本身就有較高的相似度，算是在我們可以理解的範疇之中，未來會再調整各種參數以增進我們的 Model 品質。