CNN Final Project

第一組

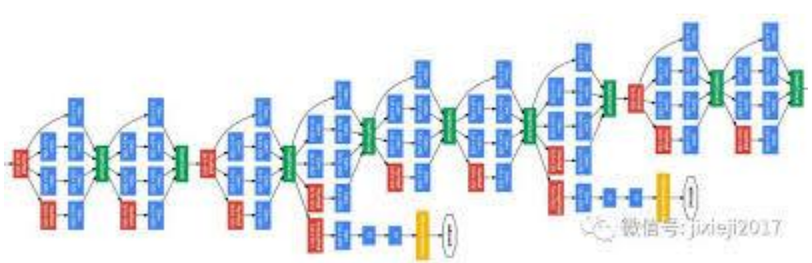
組員:張聚陽、莊宗縉、許智堯

大綱

- 動機
- 4 Steps
  - EDA
  - Data Preprocessing
  - Model Establishment
  - Model Evaluation
- 遇到的問題
- 結論

動機

- 首先是，為什麼我們選擇這個主題?
- 我們試著理解手機上的手寫辨識背後的原理。想先以一個經典的題目來測試一般卷積神經網路的精確度跟強度
- 我們預計先使用一般卷積層作訓練，若無法精確辨識再使用GoogleNet下去做，最後的備案是ResNet。



4 Steps

Step1 - EDA

1.Data Visualization
2.看資料分布

Step2 - Preprocessing

1.OneHot-Encoding
2.增加維度
3.Normalization

Step3 - Model

1.Convolutional Layer
2.Max Pooling
3.Fully Connected Layer

Step4 - Evaluation

1.Confusion Matrix
2.Visualization

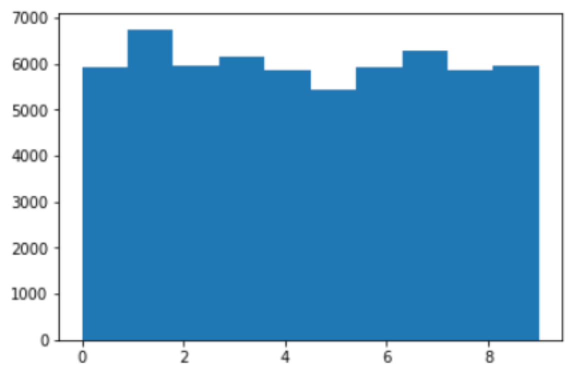
4 Steps-Steps 1

EDA

- Data visualization：看資料分布的平不平均，好不好。若資料不好，可能需要做刪除空值或augmentation。我們是利用直方圖去看，發現我們的資料分布很好，很平均，漂亮！

```
plt.hist(y_Train)

(array([5923., 6742., 5958., 6131., 5842., 5421., 5918., 6265., 5851.,
        5949.]),
 array([0. , 0.9, 1.8, 2.7, 3.6, 4.5, 5.4, 6.3, 7.2, 8.1, 9. ]),
 <a list of 10 Patch objects>)
```



4 Steps-Steps 2

Data Preprocessing

- x_train:60000筆、x_test:10000筆；再者

我們Validation的split比例為0.2

- 我們先將 y label 使用 One hot Encoding 用成維度型使之不要有類別關係而導致很難訓練
- 我們將資料增加一個維度，使我們可以做convnet。
- 爾後為了讓資料訓練速度更快，我們決定讓數據收斂，使用正規化 (將其標準化)，像我們下面這張圖

4 Steps-Steps 2

Data Preprocessing

```
# 多加一個顏色的維度
x_Train4D=x_Train.reshape(x_Train.shape[0],28,28,1).astype('float32')
x_Test4D=x_Test.reshape(x_Test.shape[0],28,28,1).astype('float32')
```

```
print('x_train_image:',x_Train.shape)
print('y_train_label:',y_Train.shape)
```

```
x_train_image: (60000, 28, 28)
y_train_label: (60000,)
```

```
print('x_test_image:',x_Test.shape)
print('y_test_label:',y_Test.shape)
```

```
x_test_image: (10000, 28, 28)
y_test_label: (10000,)
```

```
# 將數值縮小到0~1 #標準化
x_Train4D_normalize = x_Train4D / 255
x_Test4D_normalize = x_Test4D / 255
```

4 Steps-Steps 3

Model Establishment

```
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 16)        416
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 36)        14436
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 36)          0
_________________________________________________________________
dropout (Dropout)            (None, 7, 7, 36)          0
_________________________________________________________________
flatten (Flatten)            (None, 1764)              0
_________________________________________________________________
dense (Dense)                (None, 128)               225920
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                1290
=================================================================
Total params: 242,062
Trainable params: 242,062
Non-trainable params: 0
_________________________________________________________________
```

4 Steps-Steps 3

Model Establishment–Loss Function

- 我們使用的是Categorical crossentropy，並將optimizer 設成 Adam
- Batch Size = 300,Epoch = 20,Iterations = 160

```
train_history=model.fit(x=x_Train4D_normalize,
                        y=y_TrainOneHot,validation_split=0.2,
                        epochs=20, batch_size=300,verbose=2)
```

```
Epoch 1/20
160/160 - 48s - loss: 0.4781 - accuracy: 0.8496 - val_loss: 0.0973 - val_accuracy: 0.9730
Epoch 2/20
160/160 - 47s - loss: 0.1278 - accuracy: 0.9621 - val_loss: 0.0609 - val_accuracy: 0.9818
Epoch 3/20
160/160 - 47s - loss: 0.0974 - accuracy: 0.9710 - val_loss: 0.0532 - val_accuracy: 0.9851
Epoch 4/20
160/160 - 47s - loss: 0.0787 - accuracy: 0.9762 - val_loss: 0.0436 - val_accuracy: 0.9873
Epoch 5/20
160/160 - 47s - loss: 0.0663 - accuracy: 0.9799 - val_loss: 0.0417 - val_accuracy: 0.9878
```

4 Steps-Steps 4

Model Evaluation

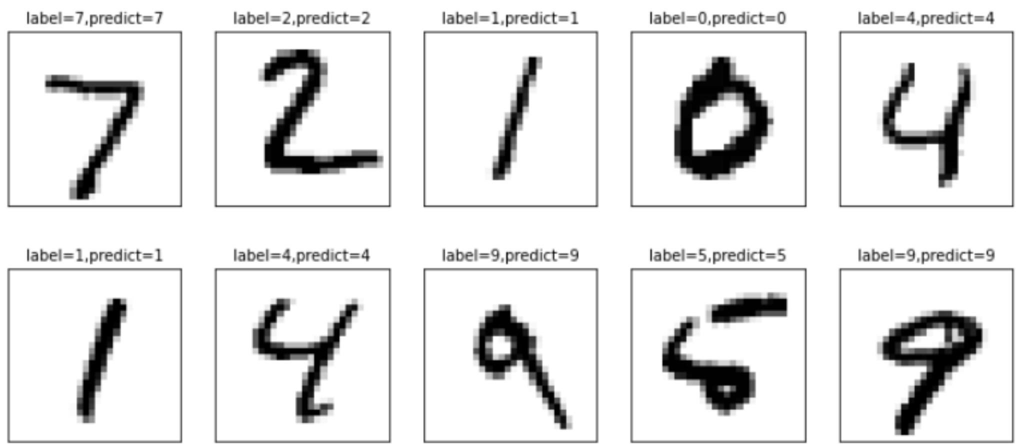
- 觀察Accuracy
- Confusion Matrix 查看各類別資料的辨識情況
  - X Precision
  - X Recall
  - X F1 score

4 Steps-Steps 4

```python
import pandas as pd
pd.crosstab(y_Test,prediction,
            rownames=['label'],colnames=['predict'])
```

predict	0	1	2	3	4	5	6	7	8	9
label										
0	978	1	0	0	0	0	0	1	0	0
1	0	1135	0	0	0	0	0	0	0	0
2	1	1	1028	0	0	0	0	2	0	0
3	0	0	0	1006	0	2	0	0	2	0
4	0	0	0	0	979	0	0	0	1	2
5	1	0	0	6	0	884	1	0	0	0
6	2	2	1	0	1	2	950	0	0	0
7	0	3	1	0	0	0	0	1022	1	1
8	1	1	1	1	0	1	0	1	967	1
9	0	1	0	0	5	3	0	0	1	999

```python
plot_images_labels_prediction(x_Test,y_Test,prediction,idx=0)
```



訓練過程所遭遇的問題

- Training Model Accuracy低:
  一開始訓練結果與網路上他人結果相比還略顯低，
  因此開始推斷以下原因 =>

Feature Collision

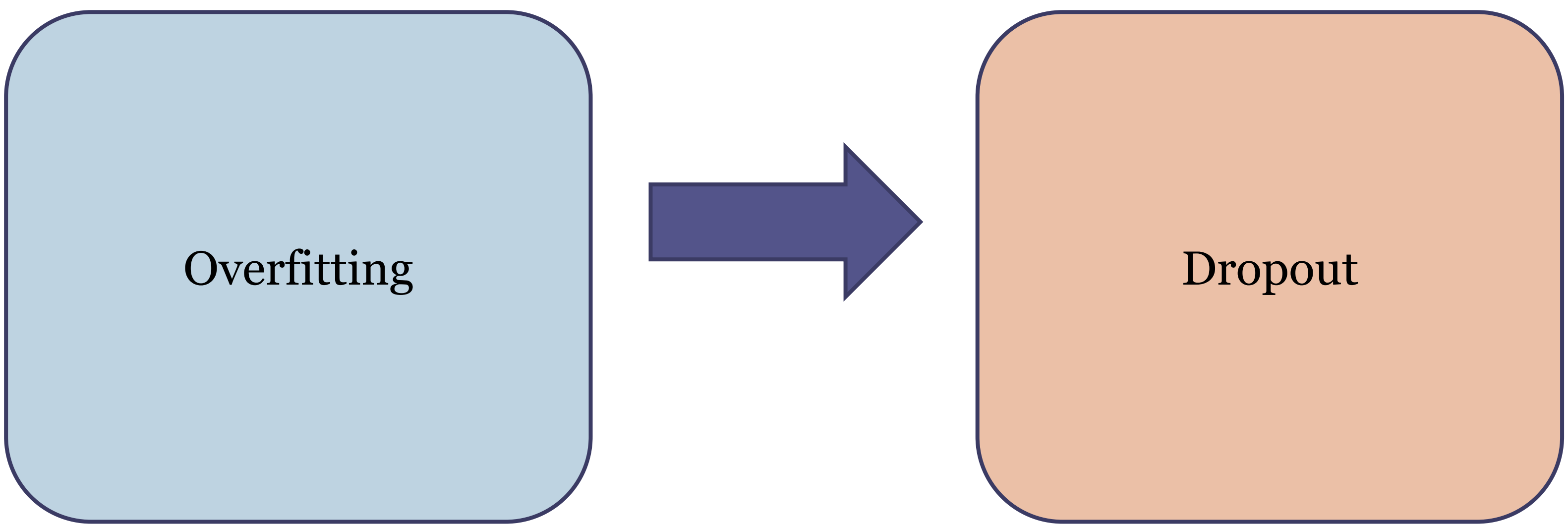
資料分布有問題

模型不夠複雜

訓練過程所遭遇的問題

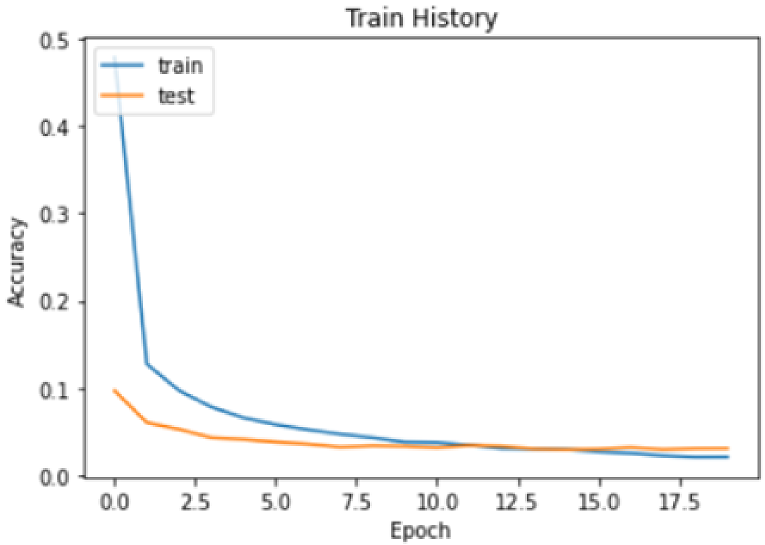
- **Test Model Accuracy**小低於**Training Model Accuracy:**
  將訓練模型的準確度有效提升後，又發現 **Test** 出來的準確度居然相較訓練的還來的低 =>

```
# Drop掉部分神經元避免overfitting
model.add(Dropout(0.25))
```

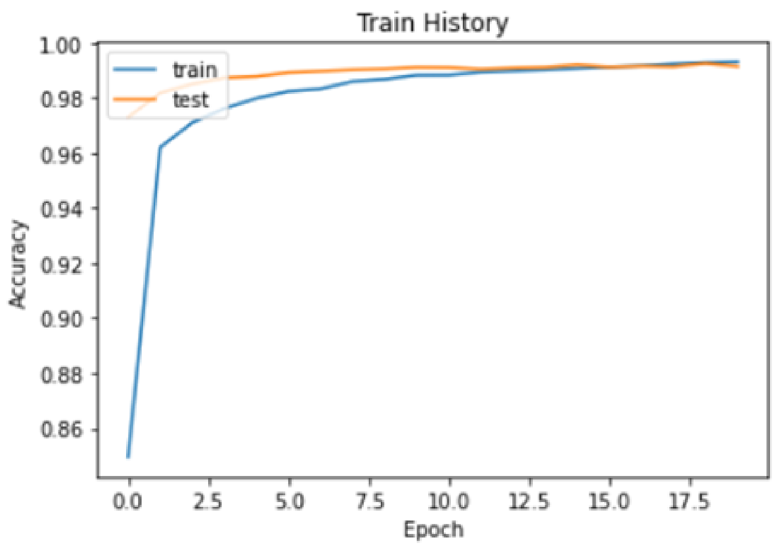


訓練成果

```
show_train_history('loss','val_loss')
```




```
show_train_history('accuracy','val_accuracy')
```



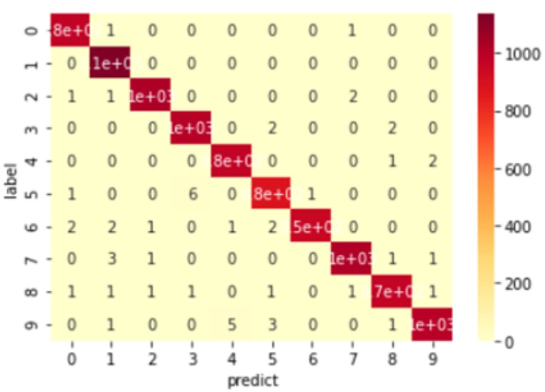

```
scores = model.evaluate(x_Test4D_normalize , y_TestOneHot)
scores[1]
```

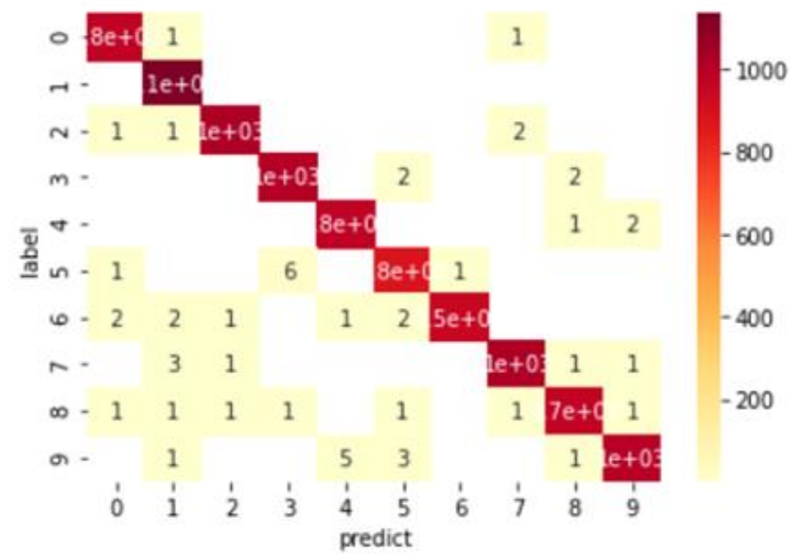
```
313/313 [==============================] - 4s 12ms/step - loss: 0.0197 - accuracy: 0.9948
```

Model Evaluation結果

```
import seaborn as sns
sns.heatmap(pd.crosstab(y_Test,prediction,
            rownames=['label'],colnames=['predict']), cmap='YlOrRd', annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f674b122048>
```



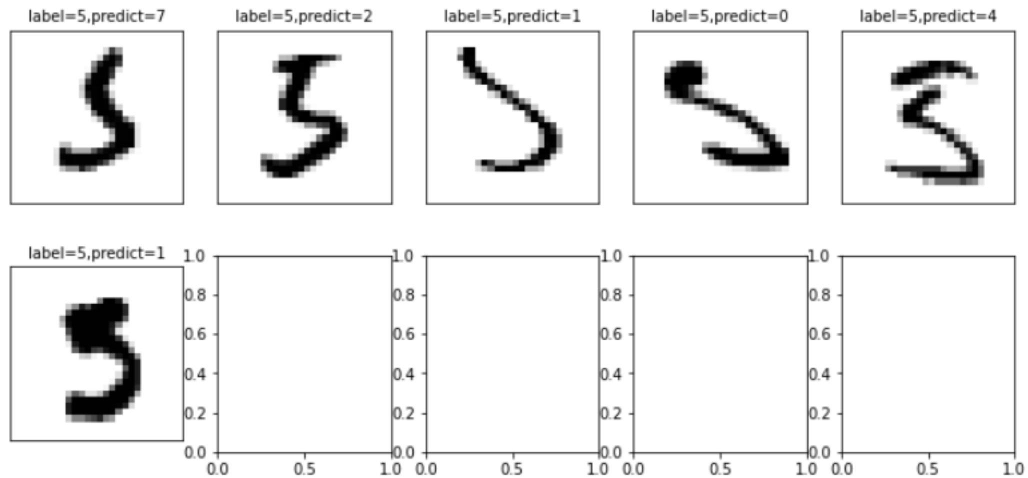


我們發現對角線(辨識正確數)非常高，因此判定訓練結果佳，而且只有零星錯誤

辨識錯誤癥結點

```
df[(df.label==5)&(df.predict==3)]
```

	label	predict
340	5	3
674	5	3
1393	5	3
1737	5	3
2597	5	3
5937	5	3




```
In [ ]: df = pd.DataFrame({'label':y_Test, 'predict':prediction})

In [ ]: df[(df.label==5)&(df.predict==3)]
```

Out[86]:

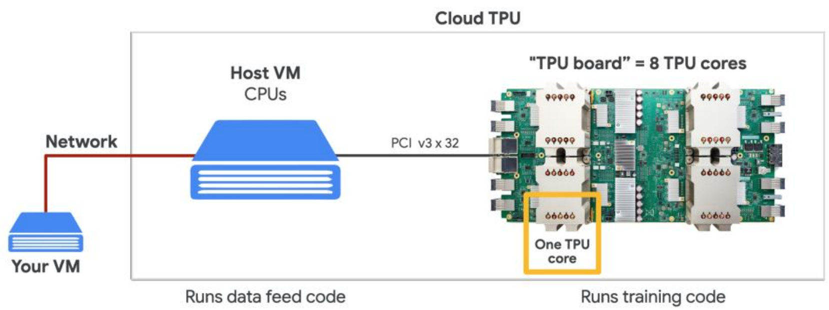
	label	predict
340	5	3
674	5	3
1393	5	3
1737	5	3
2597	5	3
5937	5	3

```
In [ ]: df[(df.label==5)&(df.predict==3)].index

Out[87]: Int64Index([340, 674, 1393, 1737, 2597, 5937], dtype='int64')

In [ ]: plot_images_labels_prediction([x_Test[i] for i in df[(df.label==5)&(df.predict==3)].index],[y_Test[i] for i in df[(df.label==5)&(
```

結論



- 執行環境:Google Colab TPU

- 使用Module:

1. Matplotlib
2. Keras
3. Numpy
4. Pandas
5. **tensorflow**

結論

- 我們在各步驟裡所使用的技術
  - Input X:Image;Output Y:Label(數字)
  - Datasets：MNIST Handwritten Digit Classification Dataset
  - Data Visualization
  - Data Preprocessing(OneHot-Encoding、Normalization)

結論

- Convolutional Layer(Activation Function: Relu)
- Max Pooling
- Fully Connected (Activation Function : Softmax)
- Loss Function(Categorical cross entropy)
- Confusion Matrix