2021

人工智慧期末專題

C N N

陳 芮 甯

CONTENTS

1 資料蒐集處理
資料的蒐集及人工前處理

2 程式前處理
在程式中的讀檔及前處理

3 程式訓練
70%的model訓練及成果展現(切出20%驗證)

4 程式測試
30%的測試結果去跑另一個測試程式

01

資料蒐集處理

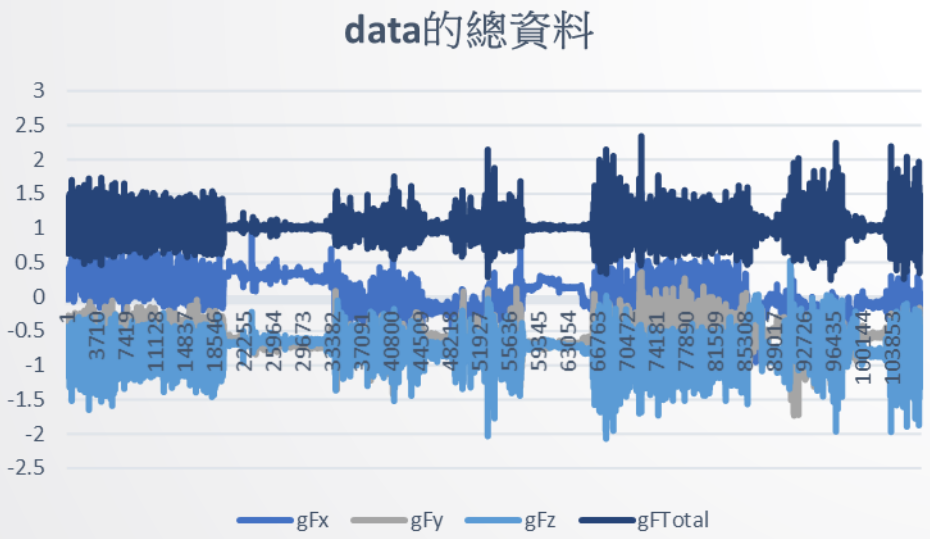
資料蒐集處理

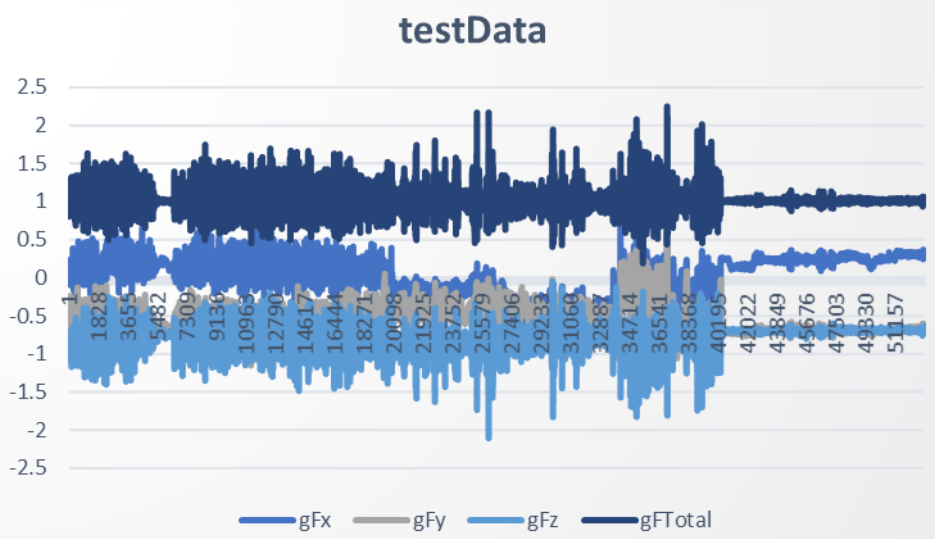
我先將label項新增進我們的csv檔中(如右圖所示)，照著之前上課交的步驟。

並將兩天的資料丟入data(沒有按照順序)，以及一天的資料丟進testData(如下兩圖所示)。

我並沒有平均，我使用原資料，其餘前處理皆於程式內完成。

	A	B	C	D	E	F
1	time	gFx	gFy	gFz	gFTotal	label
2	0.003508	-0.003	-0.32	-0.738	0.804	walk
3	0.011283	-0.004	-0.336	-0.731	0.805	walk
4	0.021508	0.008	-0.361	-0.725	0.81	walk
5	0.031771	0.038	-0.393	-0.727	0.827	walk
6	0.041599	0.067	-0.415	-0.754	0.863	walk
7	0.05201	0.082	-0.43	-0.781	0.895	walk
8	0.062266	0.098	-0.459	-0.783	0.913	walk
9	0.094684	0.073	-0.431	-0.8	0.912	walk
10	0.095028	0.073	-0.431	-0.8	0.912	walk





02

程 式 前 處 理

程 式 前 處 理

```
processedList = []
tempi = 0
# 這邊下面在進行讀檔。
with open("data.csv", "r") as f:
    for line in f:
        sepline = line.split(",")
        sepline[5] = sepline[5].replace("\n","")
        # 第一行標籤忽略
        if tempi == 0:
            tempi = 1
            continue
        temp = [sepline[0], sepline[1], sepline[2], sepline[3],sepline[4],sepline[5]]
        processedList.append(temp)
# 這邊再將資料分成time , x , y, z ,total ,label的標籤
columns = ['time','x', 'y', 'z','total','label']
data = pd.DataFrame(data = processedList, columns = columns)
```

1

讀檔

這邊是把檔案全部讀取進來，
放進processedList裡面。

程 式 前 處 理

```
44  data['x'] = data['x'].astype('float')
45  data['y'] = data['y'].astype('float')
46  data['z'] = data['z'].astype('float')
47  Fs = 100 # Hz的數字。
48  frame_size = Fs*2 # 200 # 乘以二是為了兩秒。
49  hop_size = Fs*1 # 100 # 跳躍點，為了重疊資料而生。
50  activities = data['label'].value_counts().index
51  # 把三類的資料放進activites
```

2

轉換格式，放進變數

將Object型態轉換成float

把label放進 activities

3

設定參數

Fs為Hz，

frame_size為頻率，

hop_size為跳躍點，讓資料重疊。

程 式 前 處 理

```
def plot_activity(activity, data):
    fig, (ax0, ax1, ax2) = plt.subplots(nrows=3, figsize=(10, 6), sharex=True)
    plot_axis(ax0, data['time'], data['x'], 'X-Axis')
    plot_axis(ax1, data['time'], data['y'], 'Y-Axis')
    plot_axis(ax2, data['time'], data['z'], 'Z-Axis')
    plt.subplots_adjust(hspace=0.2)
    fig.suptitle(activity)
    plt.subplots_adjust(top=0.90)
    plt.show()

def plot_axis(ax, x, y, title):
    ax.plot(x, y, 'g')
    ax.set_title(title)
    ax.xaxis.set_visible(False)
    ax.set_ylim([min(y) - np.std(y), max(y) + np.std(y)])
    ax.set_xlim([min(x), max(x)])
    ax.grid(True)
# 這邊上面都在做畫圖的function，不贅述。

for activity in activities:
    data_for_plot = data[(data['label'] == activity)][:Fs*10]
    plot_activity(activity, data_for_plot)
# 這邊再畫圖
```

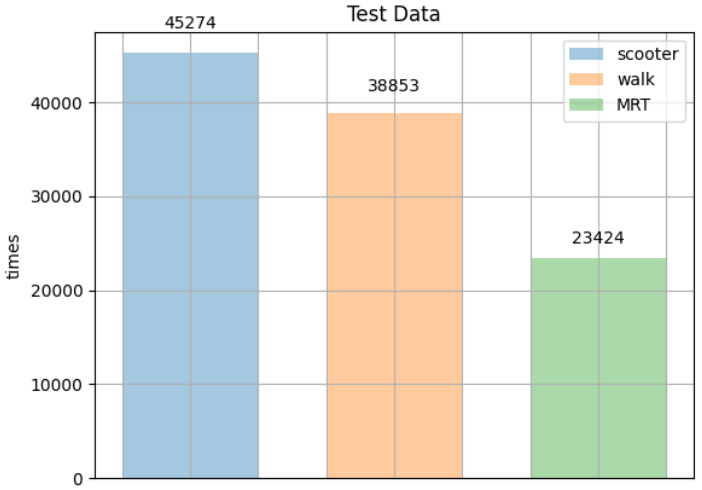
4

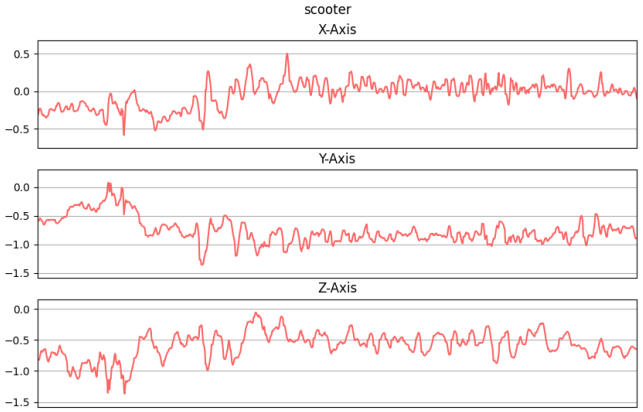
畫圖

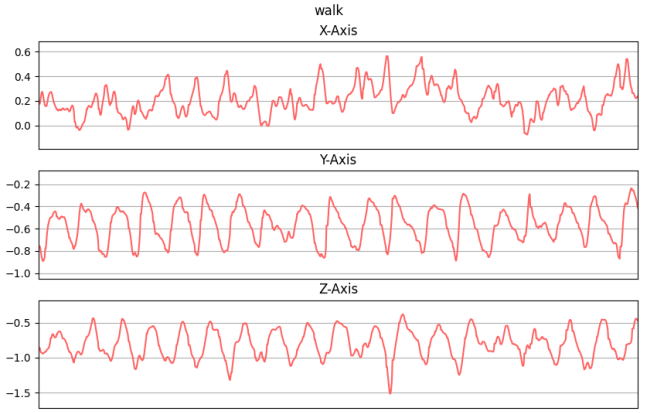
畫類似於前面的圖表去確認，

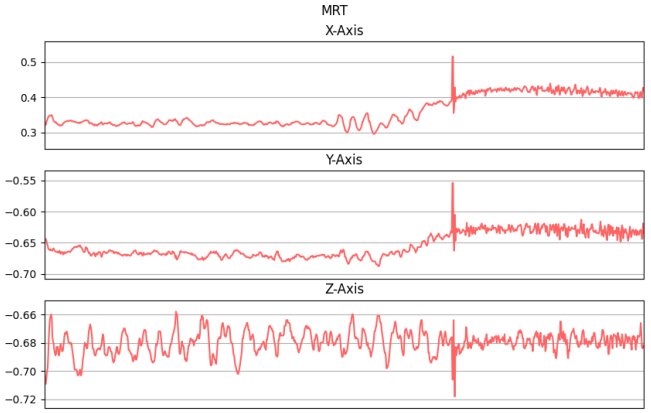
下一頁為圖表。

Data圖表化


Test Data
45274
38853
23424
scooter
walk
MRT
times
40000
30000
20000
10000
0


scooter
X-Axis
Y-Axis
Z-Axis


walk
X-Axis
Y-Axis
Z-Axis


MRT
X-Axis
Y-Axis
Z-Axis

程 式 前 處 理

```
78  df = data.drop(['total','time'], axis = 1).copy()
79  # 把不需要的值丟掉
80  walk = df[df['label']=='walk'].copy()
81  scooter = df[df['label']=='scooter'].copy()
82  MRTt = df[df['label']=='MRT'].copy()
83  # 把df的參數載入
84
85  balanced_data = pd.DataFrame()
86  balanced_data = balanced_data.append([walk, scooter, MRTt])
87  # 放進一個變數中
88
89  label = LabelEncoder()
90  balanced_data['label'] = label.fit_transform(balanced_data['label'])
91  # 把label變成 0,1,2
92
93  X = balanced_data[['x', 'y', 'z']]
94  y = balanced_data['label']
95  # 把值存入X,y
96  scaler = StandardScaler()
97  X = scaler.fit_transform(X)
98  # 計算均值何方差，然後標準化。
```

4

丟不要的值

把total跟time丟棄

5

放入標準化變數

用 StandarScaler.fit_transform() 去進行標準化的行為。(程式下方註解有寫)

程 式 前 處 理

```
def get_frames(df, frame_size, hop_size):

    N_FEATURES = 3

    frames = []
    labels = []
    for i in range(0, len(df) - frame_size, hop_size):
        x = df['x'].values[i: i + frame_size]
        y = df['y'].values[i: i + frame_size]
        z = df['z'].values[i: i + frame_size]
        # 這行是做把兩秒兩秒的資料放在一個array

        label = stats.mode(df['label'][i: i + frame_size])[0][0]
        # 這行是為了符合前一項，就取label中出現最多次的，去當作我們的label。(相接處需要用到)

        frames.append([x, y, z])
        labels.append(label)
        # 一起append是為了讓資料量相同

    frames = np.asarray(frames).reshape(-1, frame_size, N_FEATURES)
    # 這行是先把上面的array變成一行，然後做成200x3的陣列。
    labels = np.asarray(labels)
    # 這行是為了讓label跟上面的資料型態一樣。

    return frames, labels

X, y = get_frames(scaled_X, frame_size, hop_size)
#把這兩個丟進X,y。
```

6

一組一組捆起來

以前面設定的頻率跟跳躍點去把一組一組捆起來。

程 式 前 處 理

```
#資料總數/Hz*秒數 = 處理後的資料總數(也就是下方的859+215)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0, stratify = y)
# 這邊是將程式分成train跟validation

print(X_train.shape, X_test.shape)
print(X_train[0].shape, X_test[0].shape)

# 藉由上面的print去看出怎麼分的，並且把X_train跟X_test reshape，為了讓數據符合CNN的model以及符合他的內容值，放入3維的值
X_train = X_train.reshape(859, 200, 3, 1)
X_test = X_test.reshape(215, 200, 3, 1)

# print(X_train[0].shape, X_test[0].shape)
# 檢查後會變成(200,3,1)
```

7

將訓練資料切出

將訓練及測試資料切開(82分)，並且把shape重新設定。

03

程　式　訓　練

程　式　訓　練

```
159  model = Sequential()
160  # 建立一個順序模型，是最簡單的模型，從頭到尾的結構順序。
161  model.add(Conv2D(16, kernel_size=(2, 2), activation = 'relu', input_shape = X_train[0].shape))
162  # 這是input層
163  # 使用Relu函數去掉負值，更能淬煉出物體的形狀
164  model.add(Dropout(0.1))
165  # Dropout就是在不同的訓練過程中隨機扔掉一部分神經元，也就是暫時不更新權重。
166  # Dropout是用來避免過度配適（Overfitting）的。
167
168  model.add(Conv2D(32, kernel_size=(2, 2), activation='relu'))
169  model.add(Dropout(0.2))
170  # 第二層Convolution 層（32 個神經元）
171
172  model.add(Flatten())
173  # 將特徵值平攤
174
175  model.add(Dense(64, activation = 'relu'))
176  model.add(Dropout(0.5))
177  # 第三層Dense 層（64 個神經元）
178
179  model.add(Dense(3, activation='softmax'))
180  # output 3種結果
181
182  model.compile(optimizer=Adam(learning_rate = 0.001), loss = 'sparse_categorical_crossentropy', metrics = ['accuracy'])
183  # 以compile函數定義損失函數(loss)、優化函數(optimizer)及成效衡量指標(mertrics)
184
185  # 進行訓練, 訓練過程會存在 train_history 變數中
186  history = model.fit(X_train, y_train, batch_size = 32, epochs = 30, validation_data= (X_test, y_test), verbose=1)
```

程式碼展示

程　式　訓　練

- Dense param 計算方法

total_params = (input_image_channels +1)) * number_of_filters

- Conv2D param 計算方法

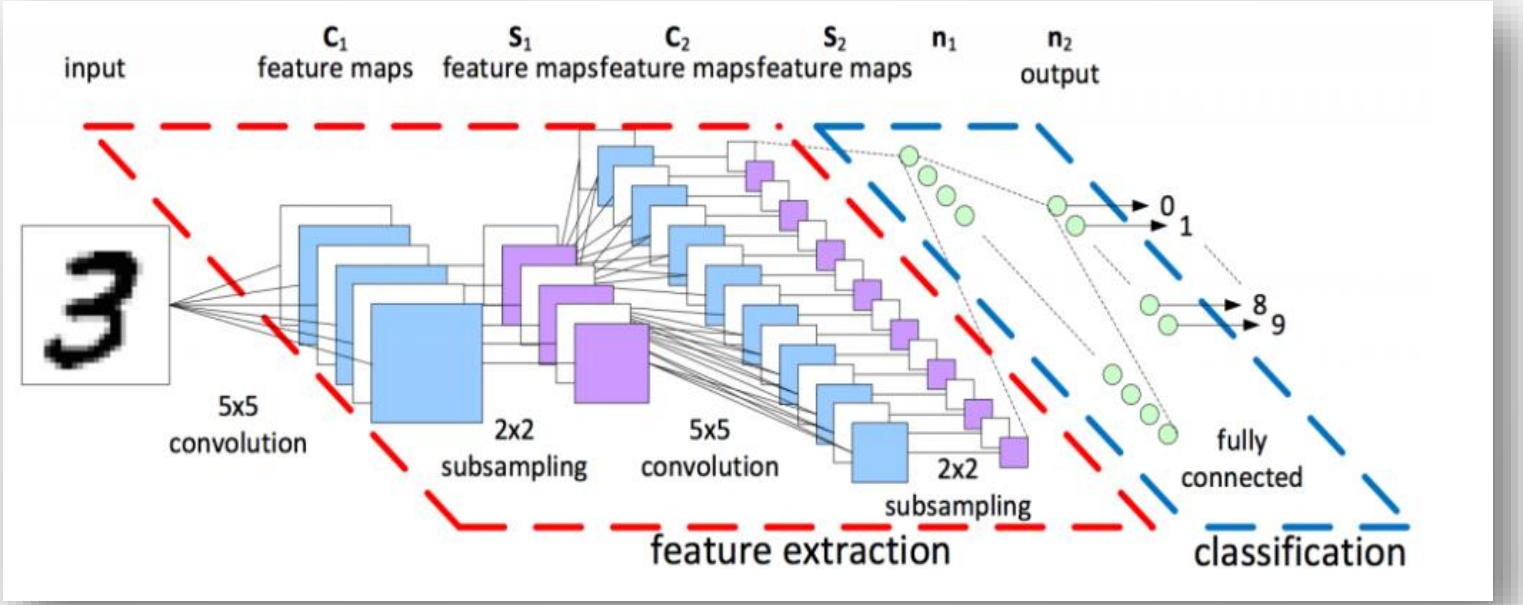
total_params = (filter_height * filter_width * input_image_channels + 1) * number_of_filters

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 199, 2, 16)        80
_________________________________________________________________
dropout (Dropout)            (None, 199, 2, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 198, 1, 32)        2080
_________________________________________________________________
dropout_1 (Dropout)          (None, 198, 1, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 6336)              0
_________________________________________________________________
dense (Dense)                (None, 64)                405568
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 195
=================================================================
Total params: 407,923
Trainable params: 407,923
Non-trainable params: 0
```

```
第一個param算法
Conv2D(16, kernel_size=(2, 2) ... )
# (2*2*1+1)*16 = 80
第二個param算法
Conv2D(32, kernel_size=(2, 2) ... )
 # (2*2*16+1)*32 = 2080
先攤平
Flatten()
# 198*1*32 param輸出0
第三個param算法
Dense(64... )
# (6336+1)*64 = 405568
第四個param算法
Dense(64... )
# (64+1)*3 = 195
```

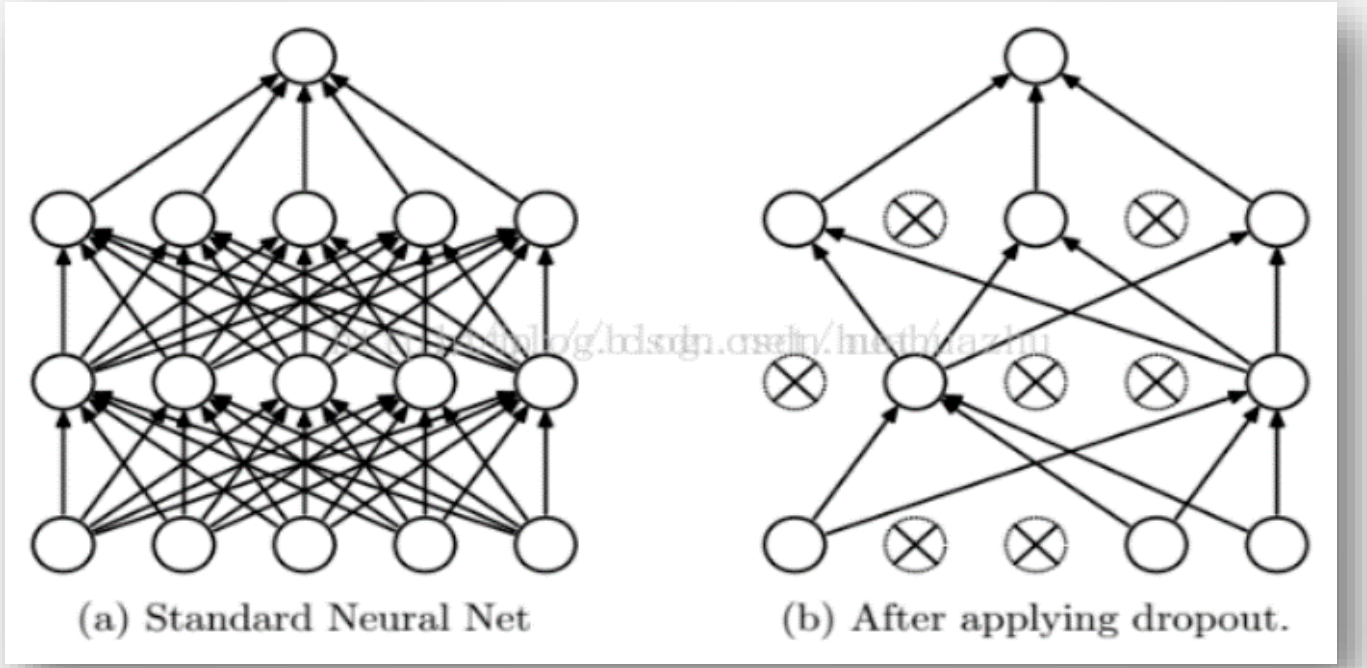
程　式　訓　練

什麼是CNN？
為什麼要用Flatten()攤平？
CNN是先用卷積把特徵抽取，進行MaxPooling或AvgPooling，最後再把特徵攤平，因此我們要用Flatten()。
輸入圖像→卷積組合→攤平(Flatten)→全連接層(Fully Connected layers)→分類



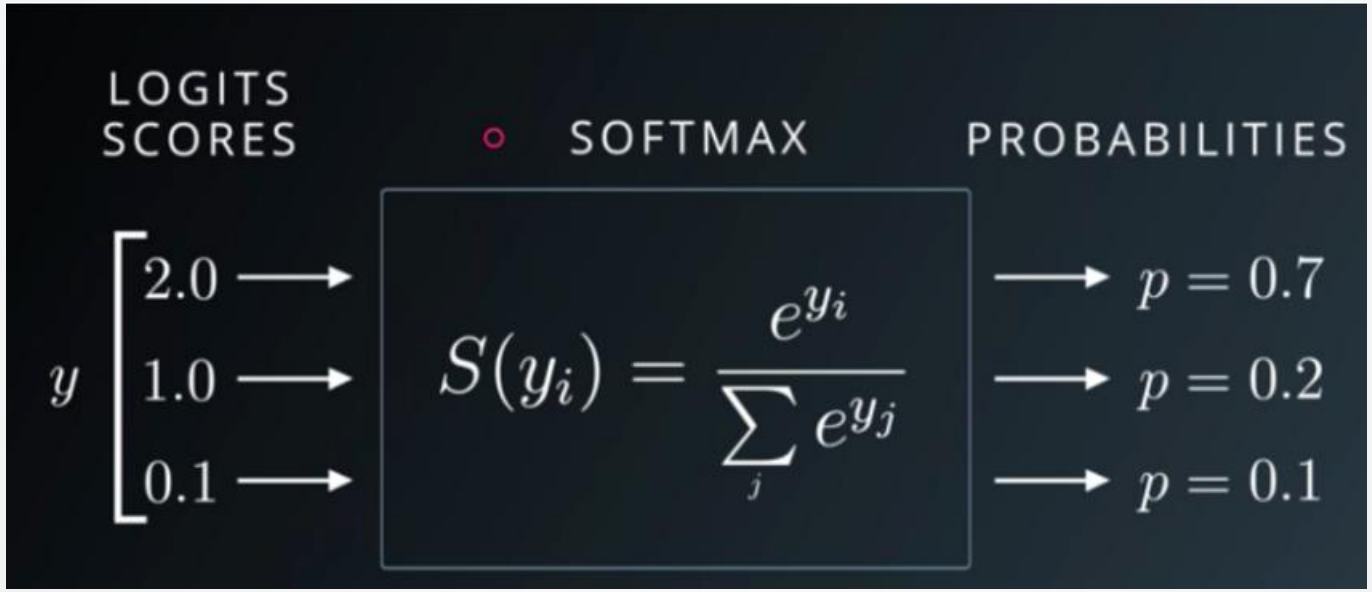
程　式　訓　練

Dropout是什麼？
Dropout就是在不同的訓練過程中隨機扔掉一部分神經元，也就是暫時不更新權重。用來**避免過度配適（Overfitting）**的。



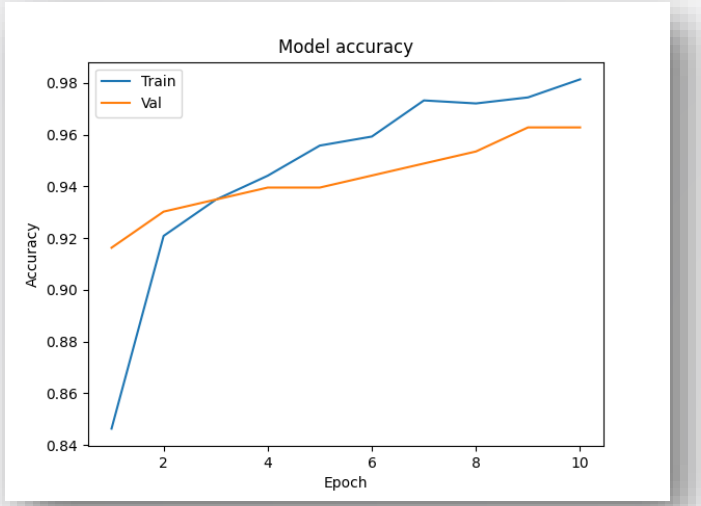
程　式　訓　練

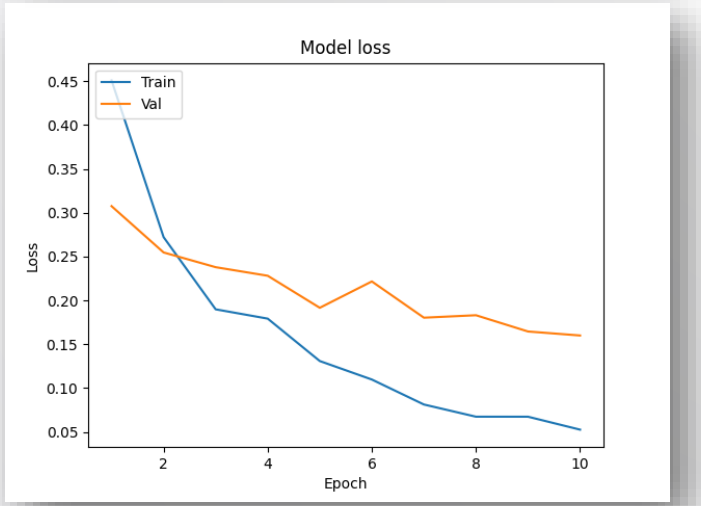
全連接層主要在做最後的特徵提取，並且利用最後一層全連接層當作分類器。
全連階層(Dense)，它的運算就是 output = activation(dot(input, kernel) + bias)，即前面提到的 y = g(x * W + b)
最後一層全連接層會使用SoftMax去進行分類。
Softmax原理，數值經過softmax function後，其加總值會變為1，因此，我們可以把各項的輸出值當作機率，而目標就是要縮小預測機率值與實際Label值，也就是老師之前ptt上課提到過的。

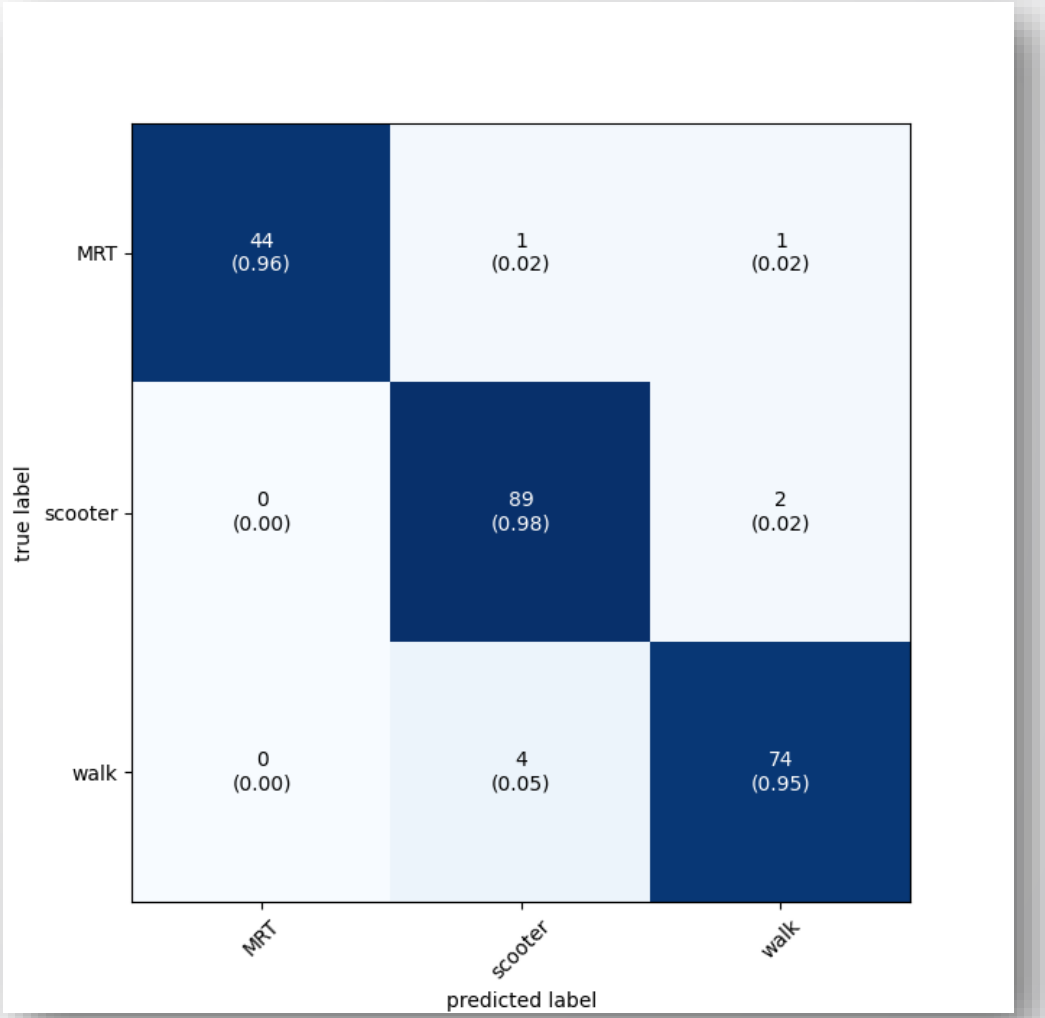


程　式　訓　練

10的圖表

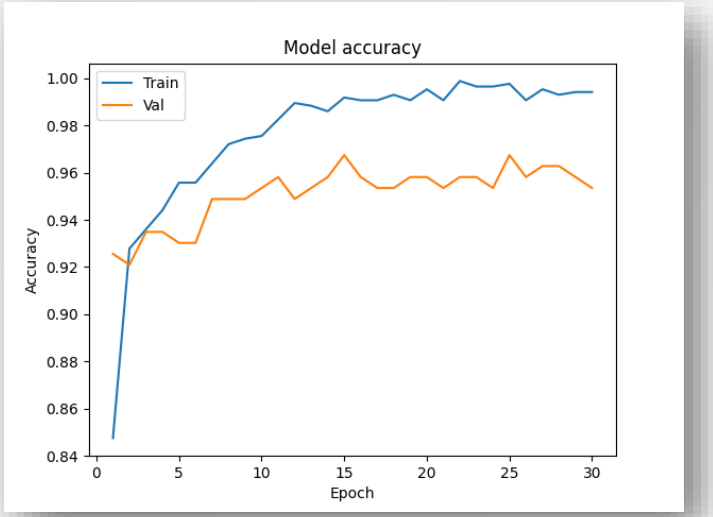

Model accuracy
Train
Val
Accuracy
Epoch
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.84
2
4
6
8
10

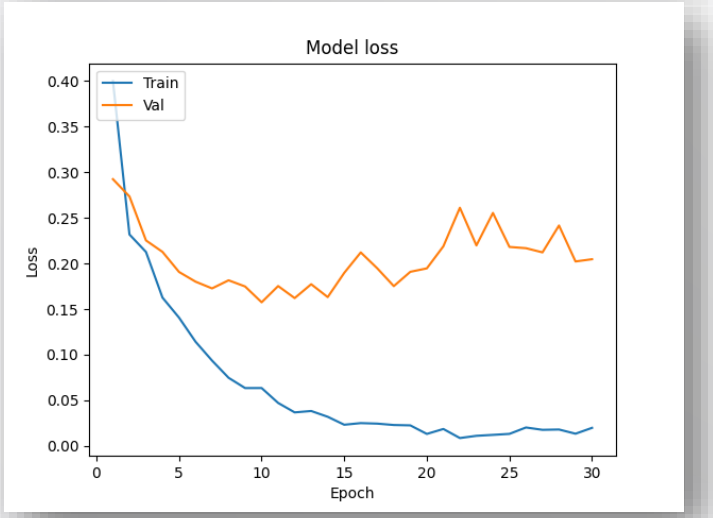

Model loss
Train
Val
Loss
Epoch
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
2
4
6
8
10

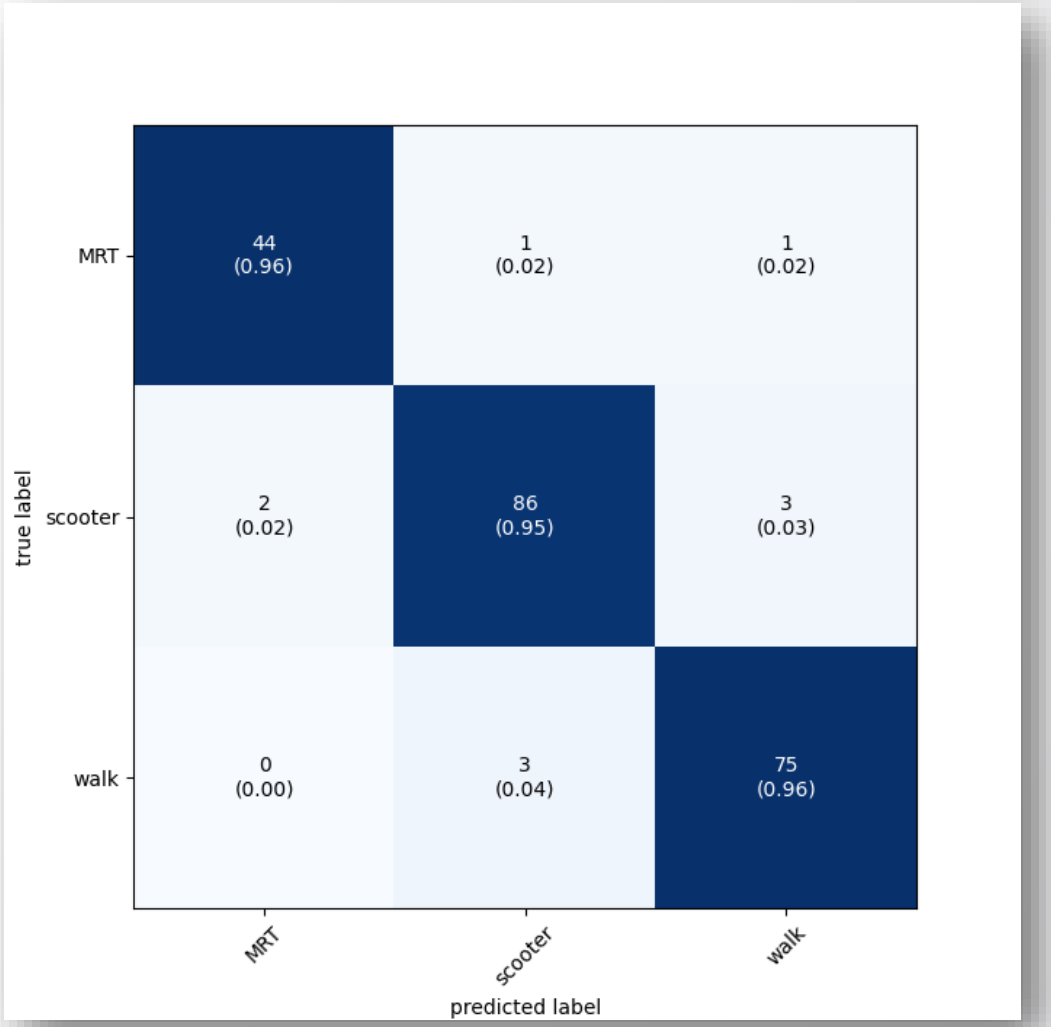

MRT
scooter
walk
true label
predicted label
44
(0.96)
1
(0.02)
1
(0.02)
0
(0.00)
89
(0.98)
2
(0.02)
0
(0.00)
4
(0.05)
74
(0.95)

程　式　訓　練

30的圖表

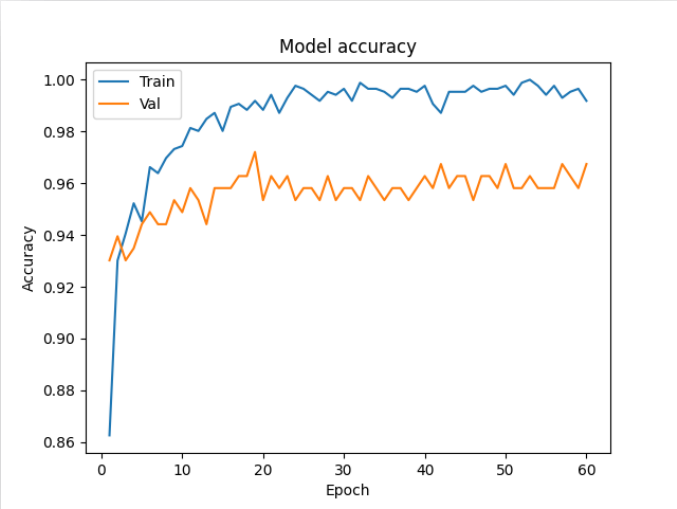

Model accuracy
Train
Val
Accuracy
Epoch
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0.84
0
5
10
15
20
25
30

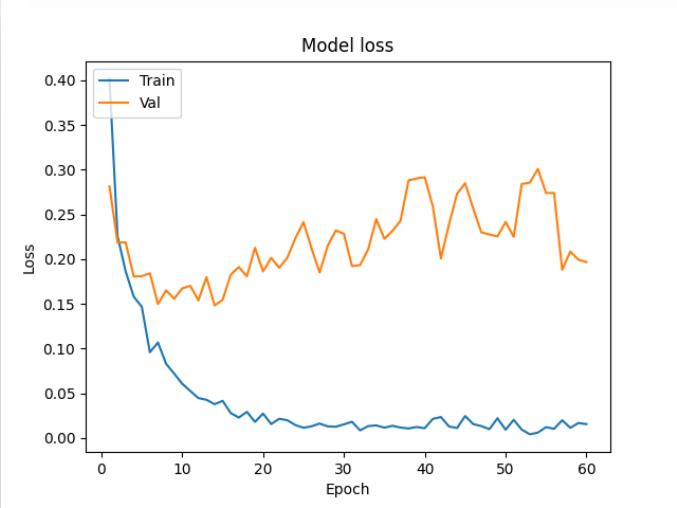

Model loss
Train
Val
Loss
Epoch
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
5
10
15
20
25
30

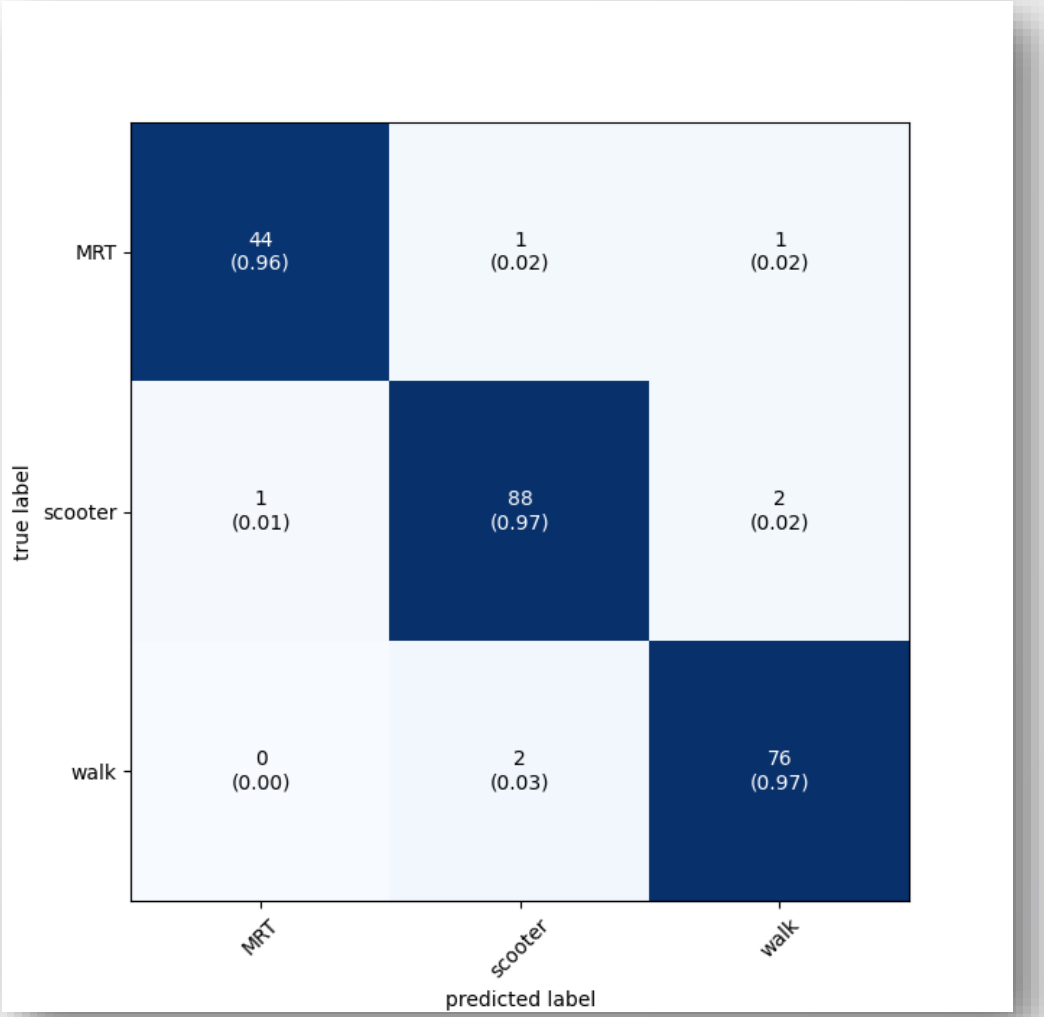

MRT
scooter
walk
true label
44
(0.96)
1
(0.02)
1
(0.02)
2
(0.02)
86
(0.95)
3
(0.03)
0
(0.00)
3
(0.04)
75
(0.96)
MRT
scooter
walk
predicted label

程　式　訓　練

60的圖表(train出來的最好範例)

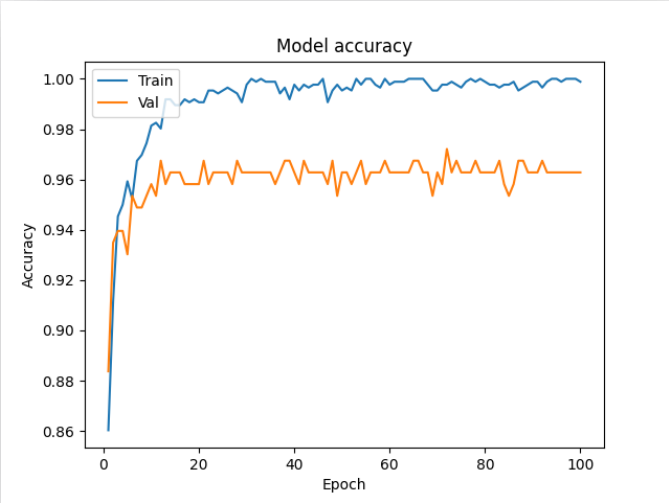


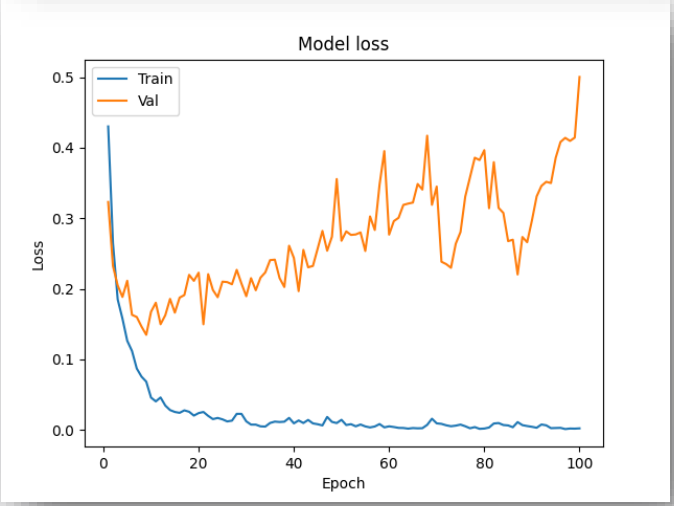


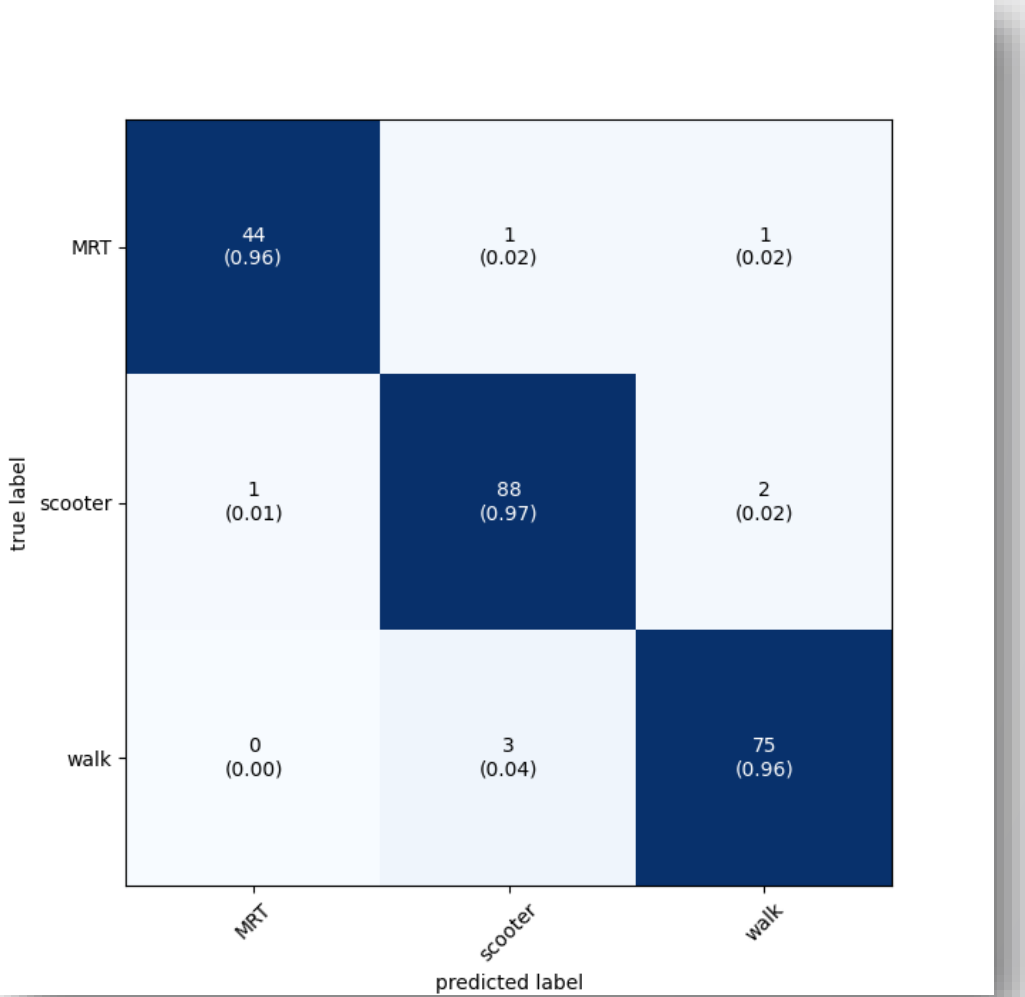


程　式　訓　練

100的圖表(看起來會overfitting)

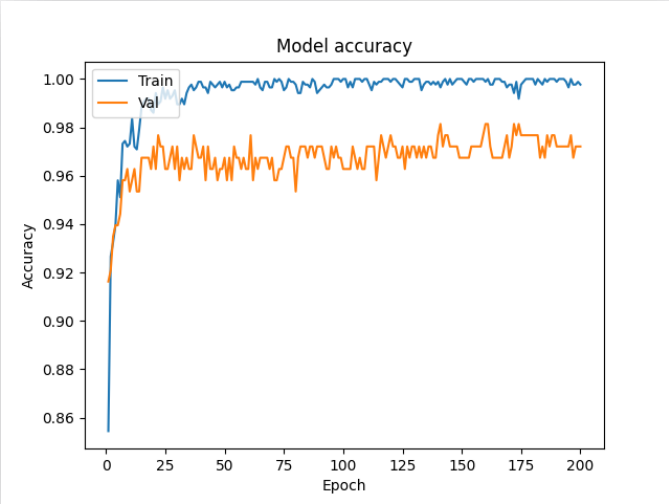


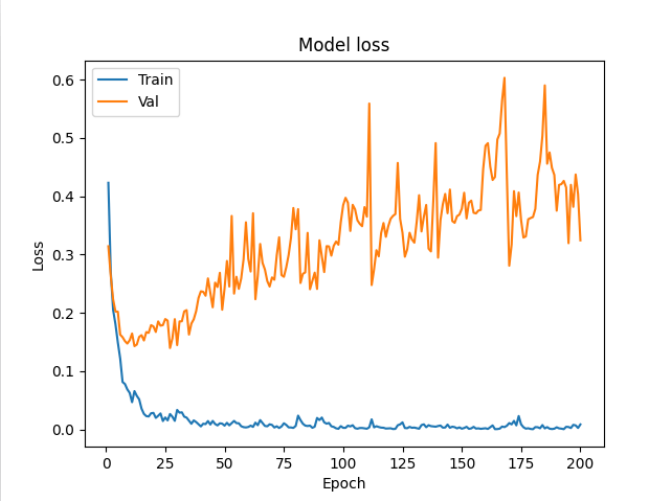


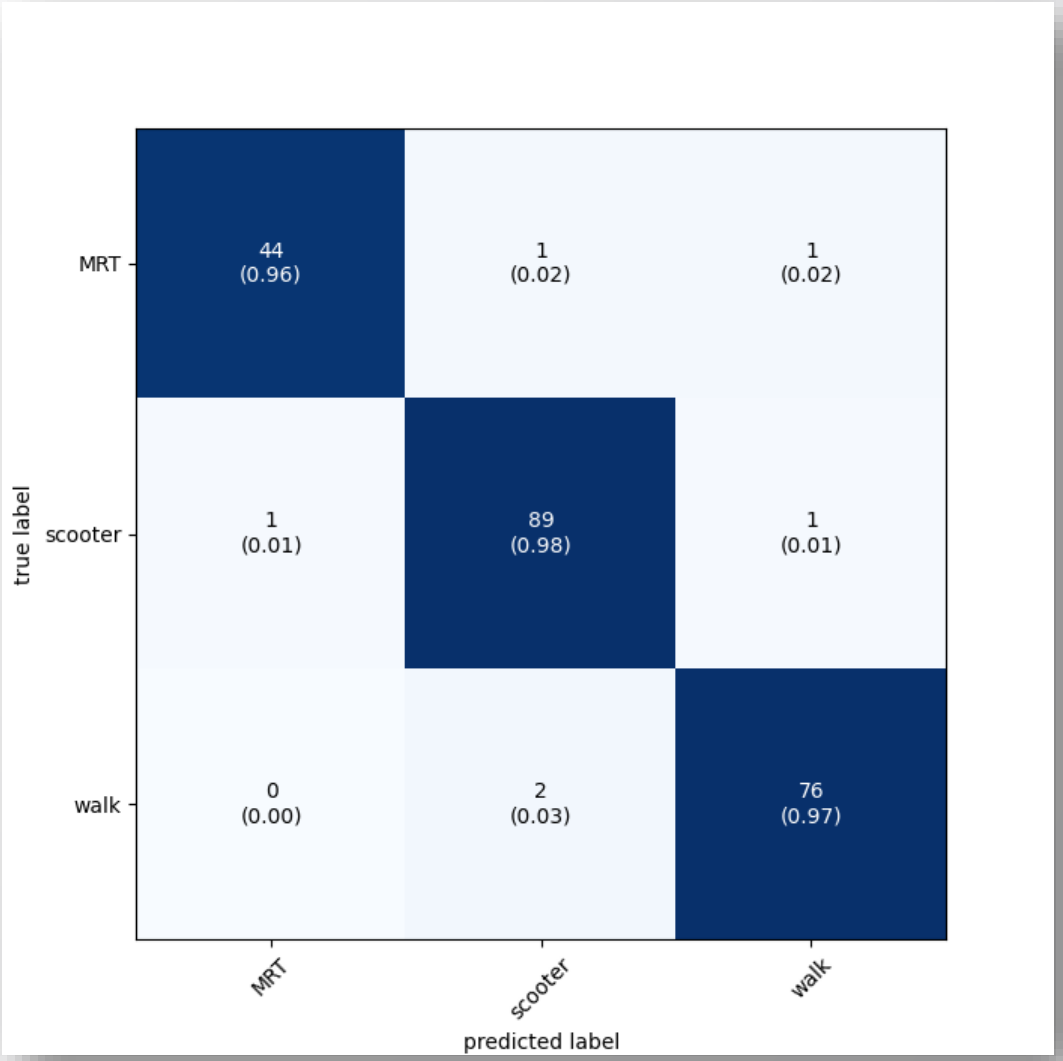


程　式　訓　練

200的圖表(看起來會overfitting)







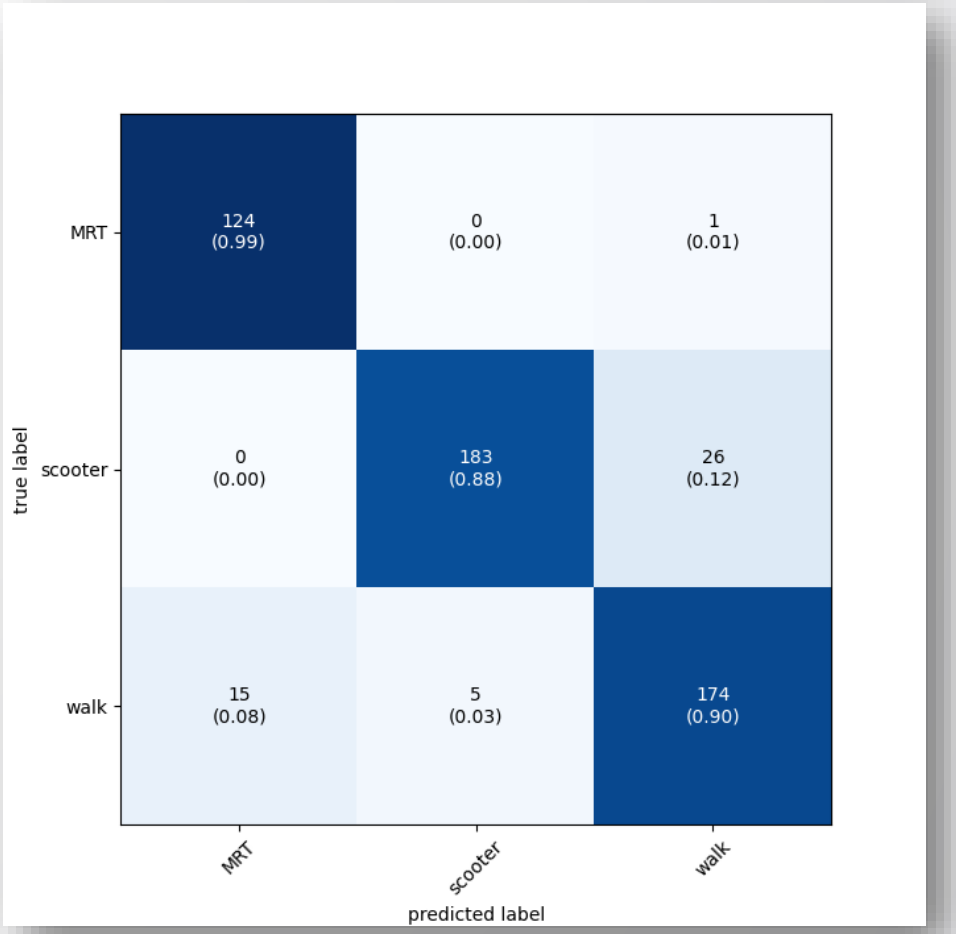
04

程　式　測　試

程　式　訓　練

epoch = 10的結果。

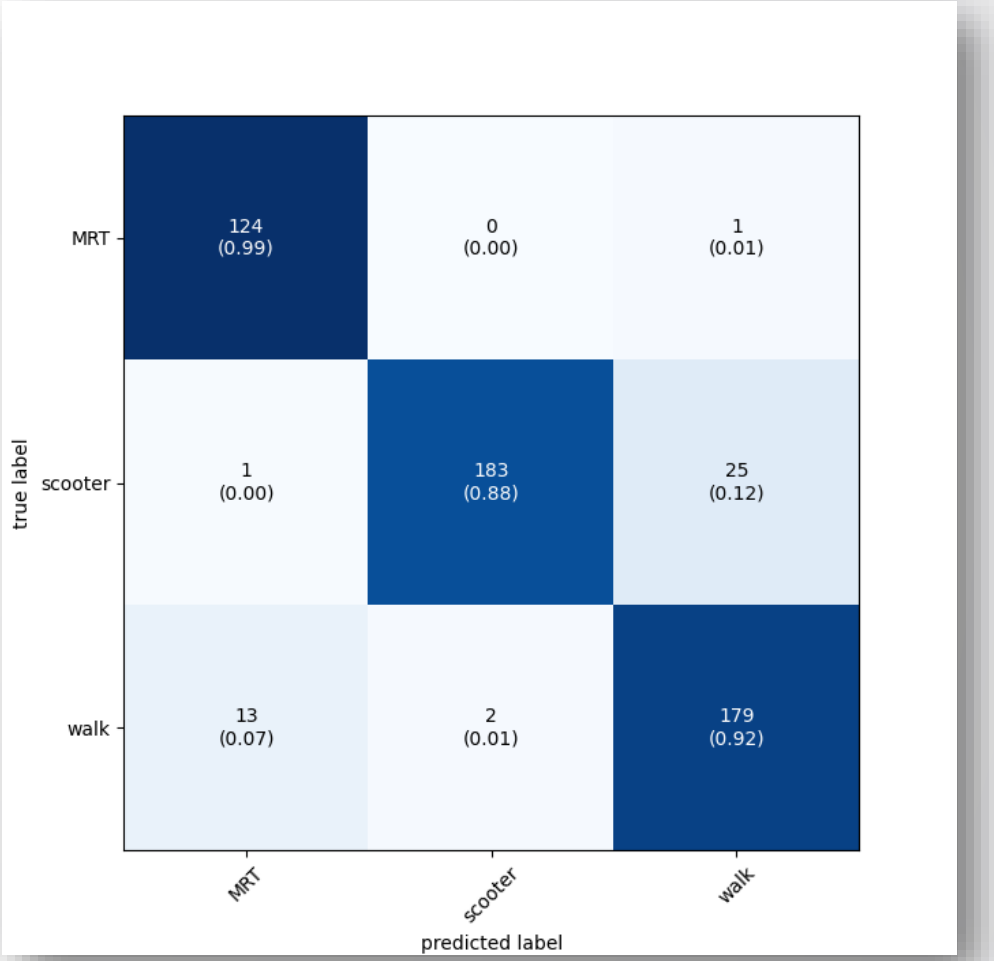
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 199, 2, 16)        80
_________________________________________________________________
dropout (Dropout)            (None, 199, 2, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 198, 1, 32)        2080
_________________________________________________________________
dropout_1 (Dropout)          (None, 198, 1, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 6336)              0
_________________________________________________________________
dense (Dense)                (None, 64)                405568
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 195
=================================================================
Total params: 407,923
Trainable params: 407,923
Non-trainable params: 0
_________________________________________________________________
mission success rate = 91.10%
```



程　式　訓　練

epoch = 30的結果。

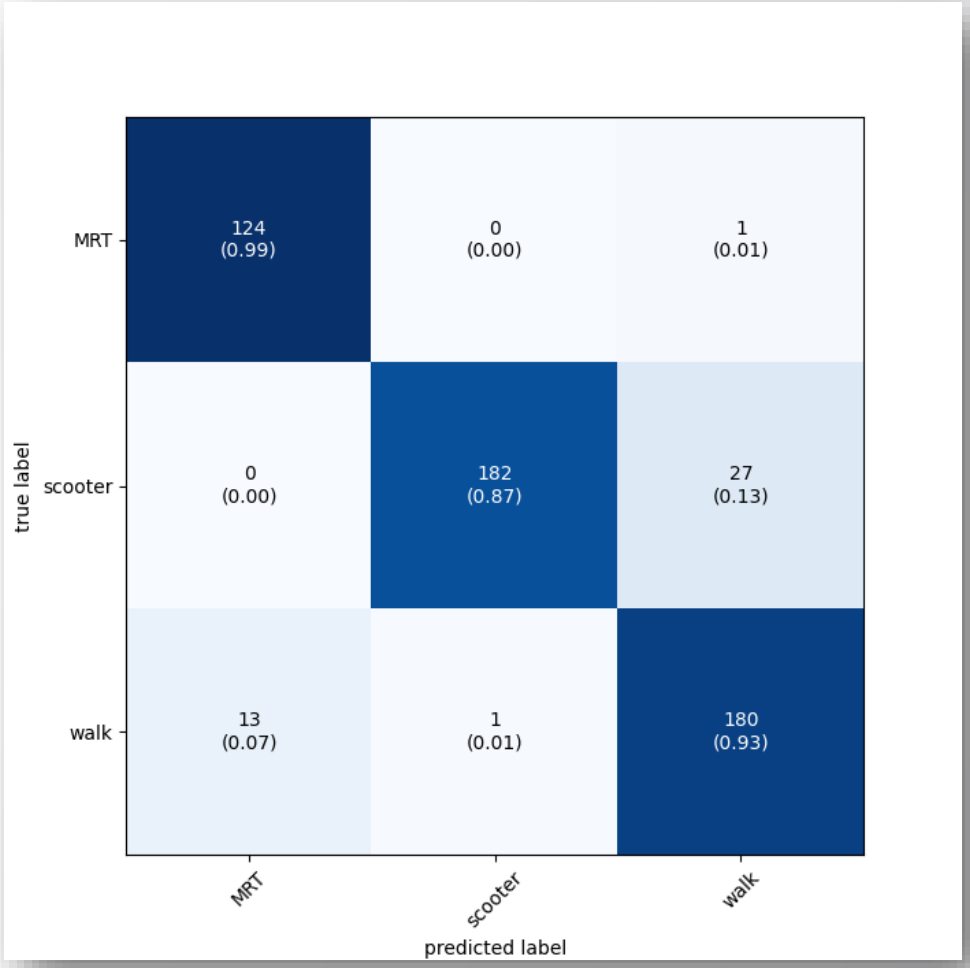
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 199, 2, 16)        80
_________________________________________________________________
dropout (Dropout)            (None, 199, 2, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 198, 1, 32)        2080
_________________________________________________________________
dropout_1 (Dropout)          (None, 198, 1, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 6336)              0
_________________________________________________________________
dense (Dense)                (None, 64)                405568
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 195
=================================================================
Total params: 407,923
Trainable params: 407,923
Non-trainable params: 0
_________________________________________________________________
mission success rate = 92.05%
```



程　式　訓　練

epoch = 60的結果。

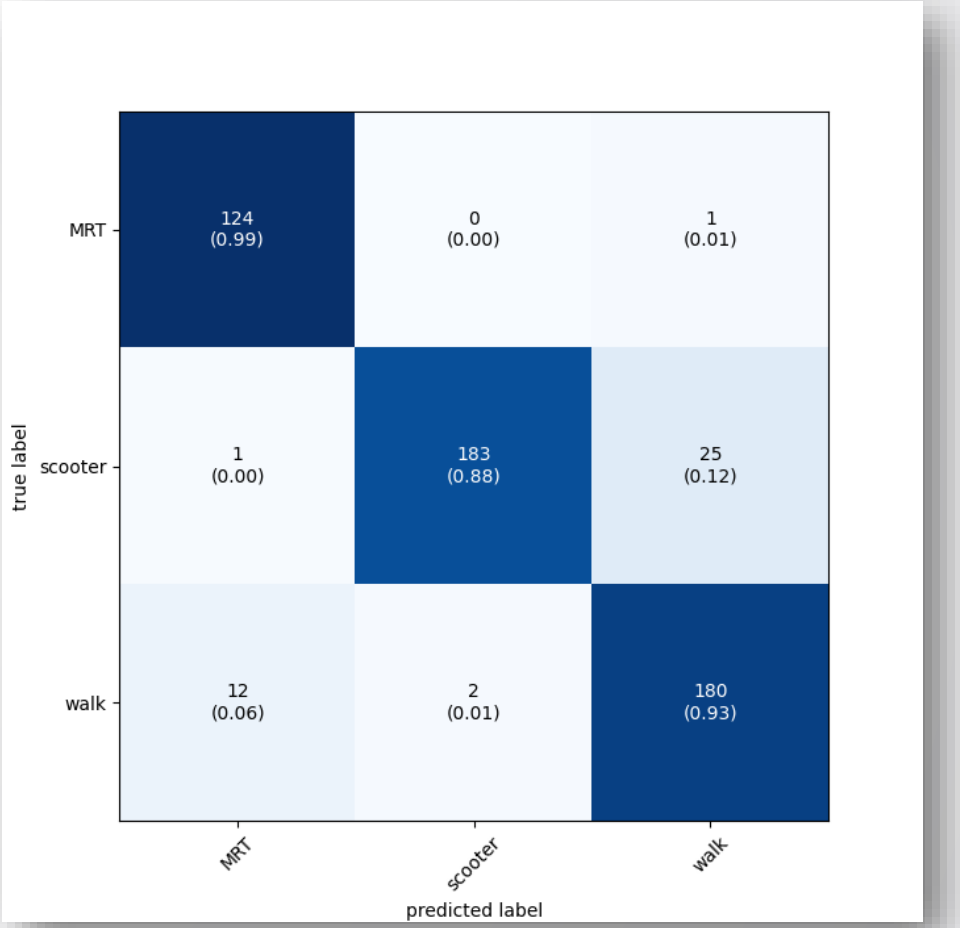
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 199, 2, 16)        80
_________________________________________________________________
dropout (Dropout)            (None, 199, 2, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 198, 1, 32)        2080
_________________________________________________________________
dropout_1 (Dropout)          (None, 198, 1, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 6336)              0
_________________________________________________________________
dense (Dense)                (None, 64)                405568
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 195
=================================================================
Total params: 407,923
Trainable params: 407,923
Non-trainable params: 0
_________________________________________________________________
mission success rate = 92.05%
```



程　式　訓　練

epoch = 100的結果。

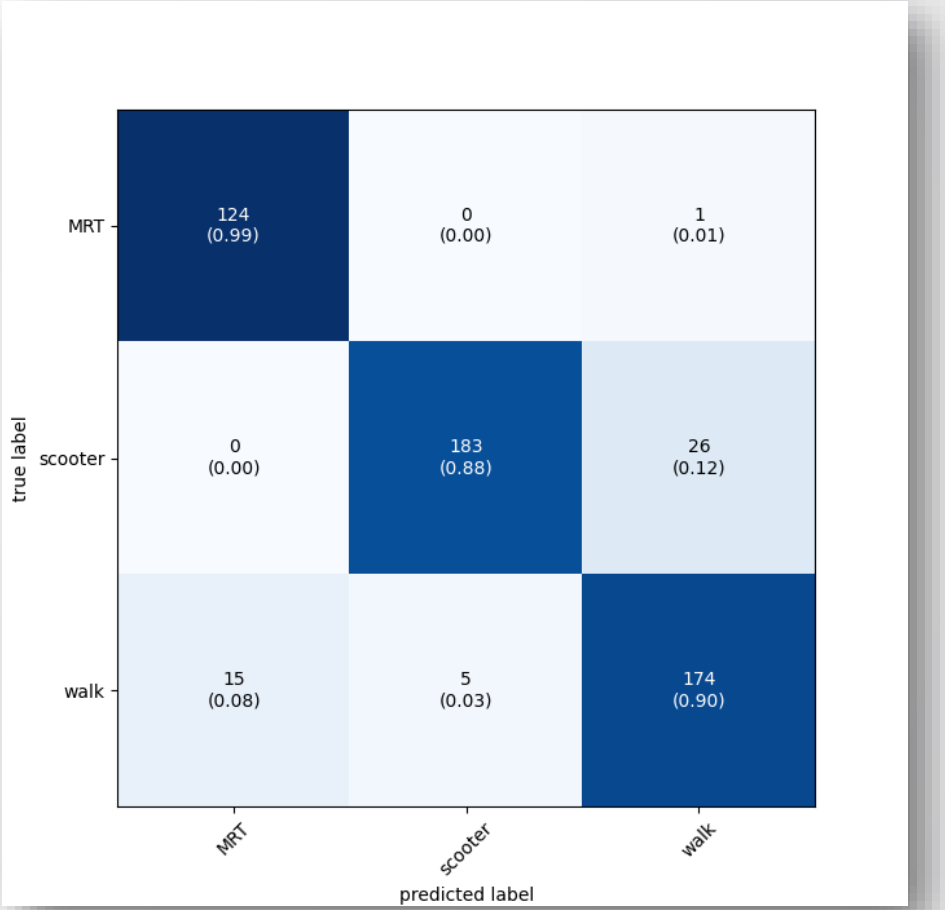
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 199, 2, 16)        80
_________________________________________________________________
dropout (Dropout)            (None, 199, 2, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 198, 1, 32)        2080
_________________________________________________________________
dropout_1 (Dropout)          (None, 198, 1, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 6336)              0
_________________________________________________________________
dense (Dense)                (None, 64)                405568
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 195
=================================================================
Total params: 407,923
Trainable params: 407,923
Non-trainable params: 0
_________________________________________________________________
mission success rate = 92.23%
```



程　式　訓　練

epoch = 200的結果。(overfitting)

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 199, 2, 16)        80
_________________________________________________________________
dropout (Dropout)            (None, 199, 2, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 198, 1, 32)        2080
_________________________________________________________________
dropout_1 (Dropout)          (None, 198, 1, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 6336)              0
_________________________________________________________________
dense (Dense)                (None, 64)                405568
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 195
=================================================================
Total params: 407,923
Trainable params: 407,923
Non-trainable params: 0
_________________________________________________________________
mission success rate = 91.10%
```



程　式　訓　練

問題與討論:

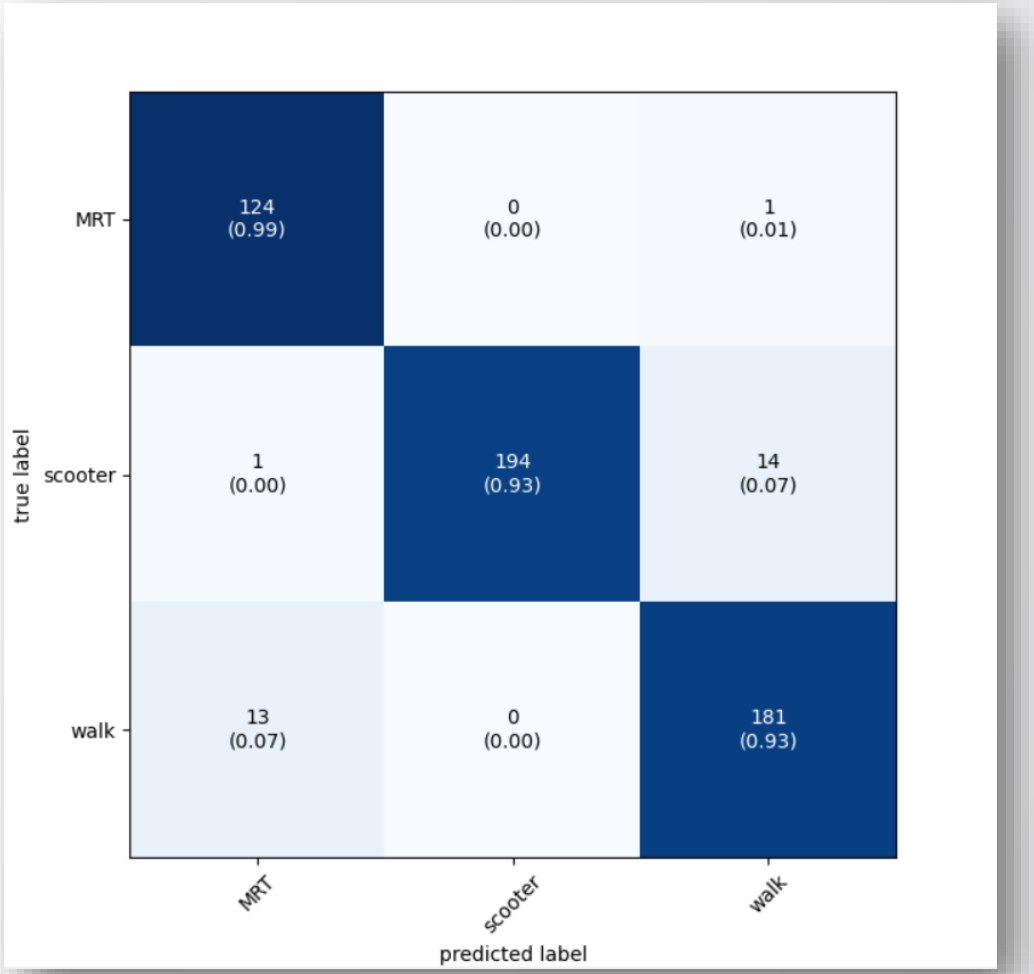
前面認為會overfitting的事情，結果實際test發現反而結果更好，
令人詫異，值得拿出來分享一下。

分別為
10epochs = 91.10%
30epochs = 92.05%
60epochs = 92.05%
100epochs = 92.23% (best)
200epochs = 91.10%

程　式　訓　練

我把dropout值改低成 0.6 後，大約在94%，但有時候是89%左右，我覺得很奇怪，於是想用別的值測測看。

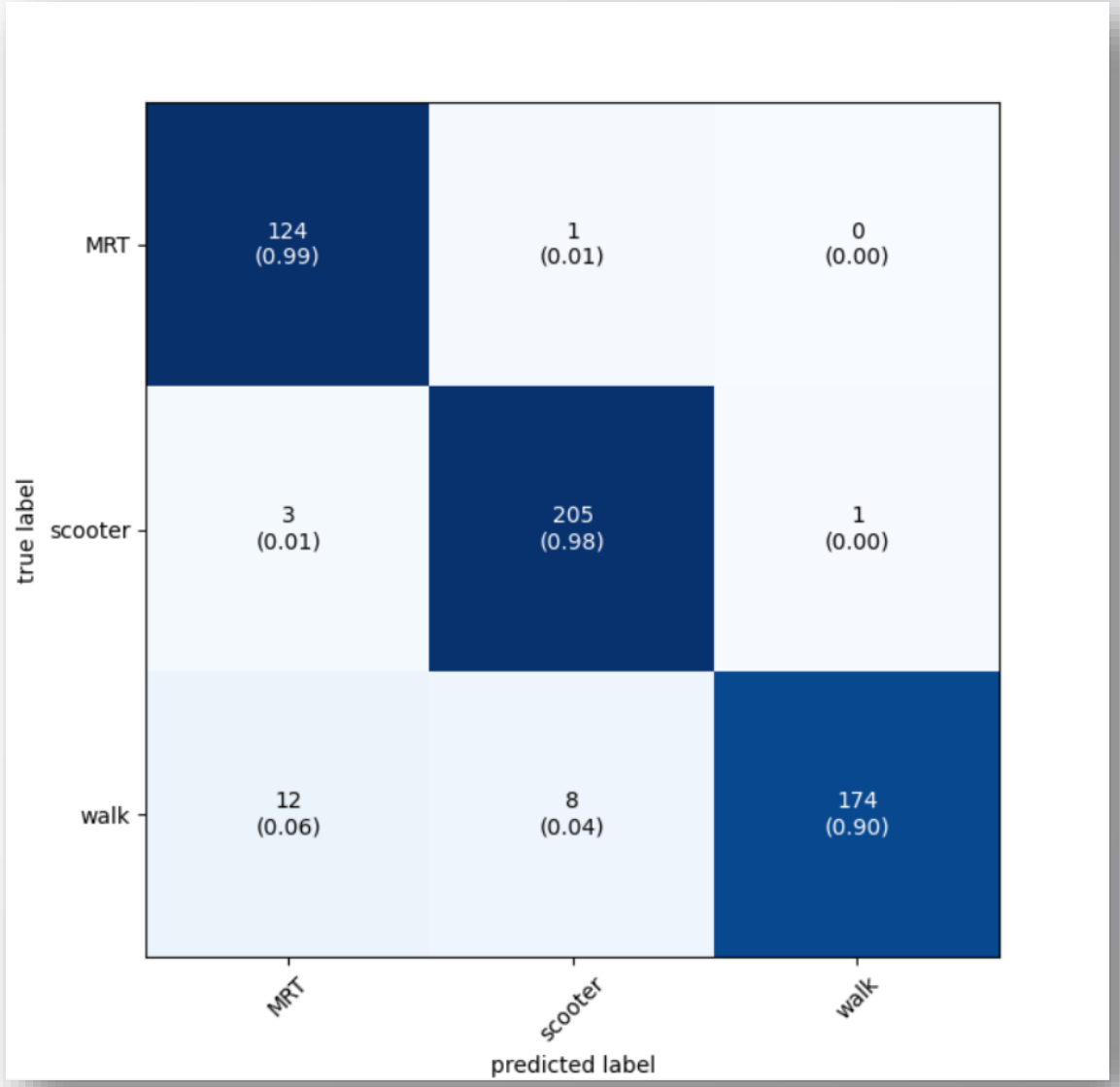
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 199, 2, 16)        80
_________________________________________________________________
dropout (Dropout)            (None, 199, 2, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 198, 1, 32)        2080
_________________________________________________________________
dropout_1 (Dropout)          (None, 198, 1, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 6336)              0
_________________________________________________________________
dense (Dense)                (None, 64)                405568
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 195
=================================================================
Total params: 407,923
Trainable params: 407,923
Non-trainable params: 0
_________________________________________________________________
mission success rate = 94.51%
```



程　式　訓　練

我把dropout值改低成 0.3 後，我就成功上升成95.08%，但我覺得這可能只是運氣，多嘗試幾次後發現會上下起伏，也有測到90%過。
只是剛好loss的部分是對的，他丟掉了卻學對了。

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 199, 2, 16)        80
_________________________________________________________________
dropout (Dropout)            (None, 199, 2, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 198, 1, 32)        2080
_________________________________________________________________
dropout_1 (Dropout)          (None, 198, 1, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 6336)              0
_________________________________________________________________
dense (Dense)                (None, 64)                405568
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 195
=================================================================
Total params: 407,923
Trainable params: 407,923
Non-trainable params: 0
_________________________________________________________________
mission success rate = 95.27%
```

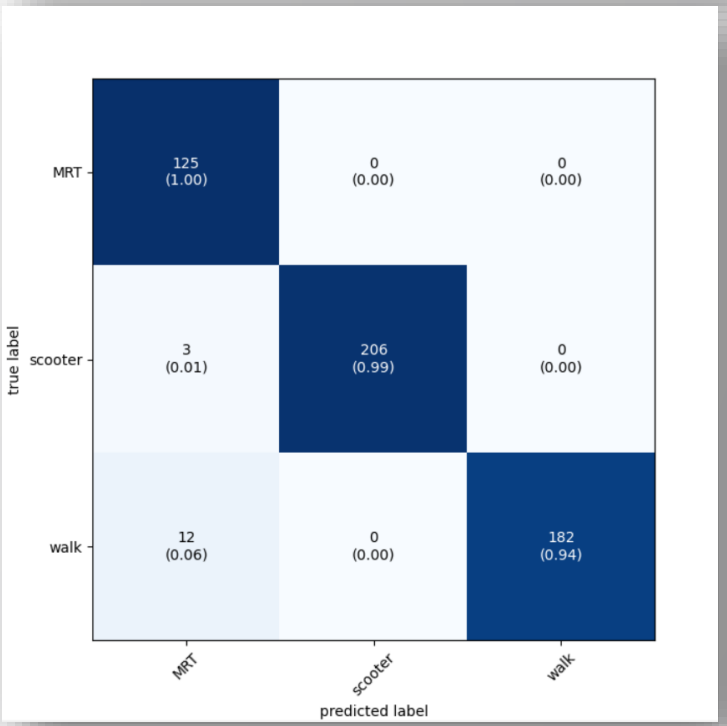


程　式　訓　練

上網查了average_pooling2d&global_average_pooling2d用起來覺得效果很好，成功率上升到97.16%，並且我的loss值也符合了期待。

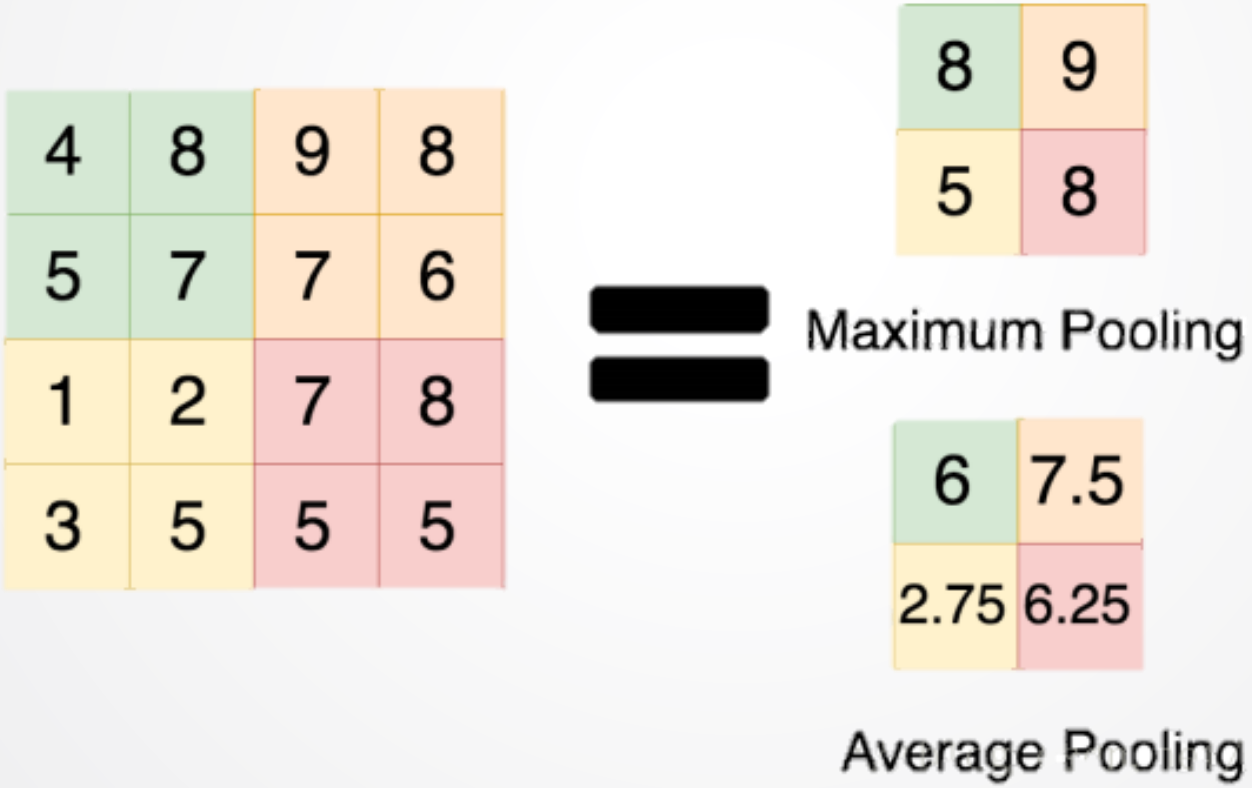
```
model = Sequential()
# 建立一個順序模型，是最簡單的模型，從頭到尾的結構順序。
model.add(Conv2D(filters=16,kernel_size=(3,3),strides=(1,1),padding="same", activation = 'relu', input_shape = X_train[0].shape))
# 這是input層
# 使用Relu函數去掉負值，更能淬煉出物體的形狀
model.add(AveragePooling2D(pool_size=(2,2)))
#AveragePooling2D是為了平均整體值，減少誤差。
model.add(Dropout(0.5))
# Dropout就是在不同的訓練過程中隨機扔掉一部分神經元，也就是暫時不更新權重。
# Dropout是用來避免過度配適（Overfitting）的。

model.add(Conv2D(filters=32,kernel_size=(3,3),strides=(1,1),padding="same", activation='relu'))
model.add(Dropout(0.5))
# 第二層Convolution 層（32 個神經元）
model.add(GlobalAveragePooling2D())
#model.add(Flatten())
# 將特徵值平攤，GlobalAveragePooling2D去做。

model.add(Dense(64, activation = 'relu'))
model.add(Dropout(0.5))
# 第三層Dense 層（64 個神經元）

model.add(Dense(3, activation='softmax'))
# output 3種結果

model.compile(optimizer=Adam(learning_rate = 0.001), loss = 'sparse_categorical_crossentropy', metrics = ['accuracy'])
# 以compile函數定義損失函數(loss)、優化函數(optimizer)及成效衡量指標(mertrics)
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 200, 3, 16)        160
_________________________________________________________________
average_pooling2d (AveragePo (None, 100, 1, 16)        0
_________________________________________________________________
dropout (Dropout)            (None, 100, 1, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 100, 1, 32)        4640
_________________________________________________________________
dropout_1 (Dropout)          (None, 100, 1, 32)        0
_________________________________________________________________
global_average_pooling2d (Gl (None, 32)                0
_________________________________________________________________
dense (Dense)                (None, 64)                2112
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 195
=================================================================
Total params: 7,107
Trainable params: 7,107
Non-trainable params: 0
_________________________________________________________________
mission success rate = 97.16%
```



Average Pooling

亦稱Mean pooling，也就是Average pooling，做法也很簡單，就是將滑動視窗框選到的矩陣值加總，然後計算平均即可。

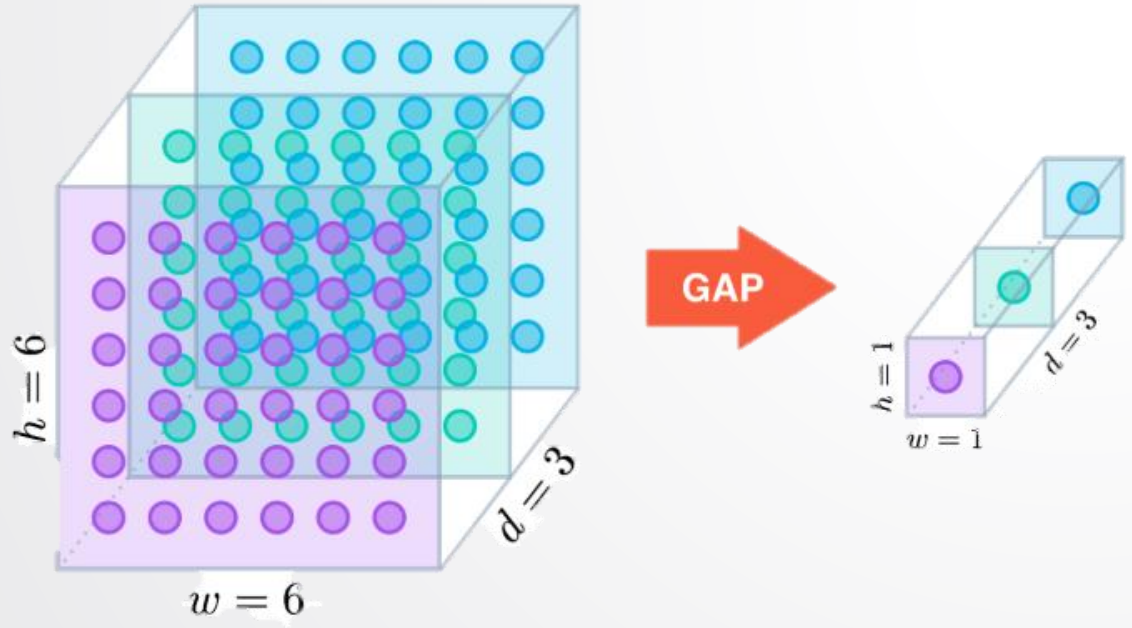


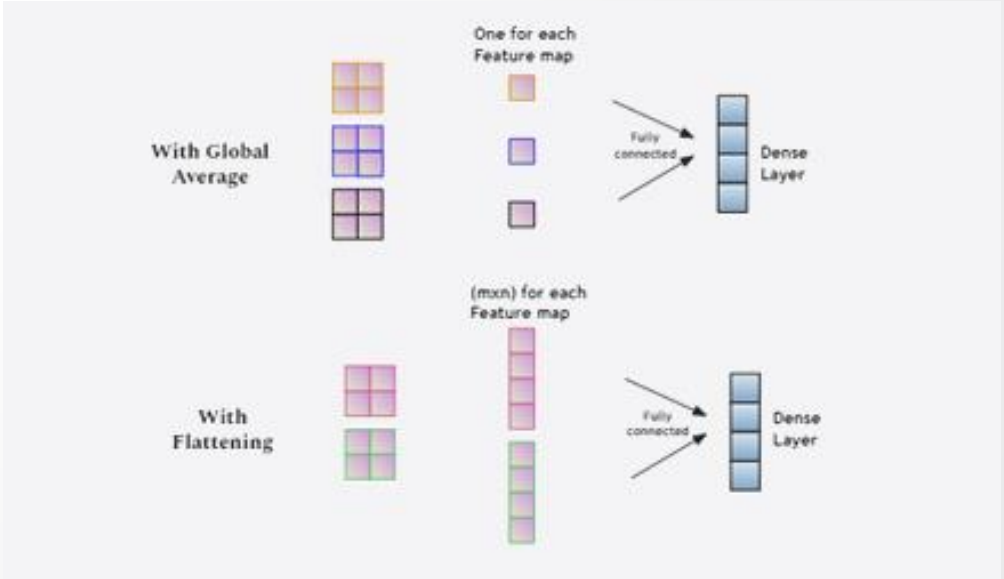
Global Average Pooling

簡稱GAP，簡單說，就是卷積層後，一種可以取代全連接層（FC）的作法。

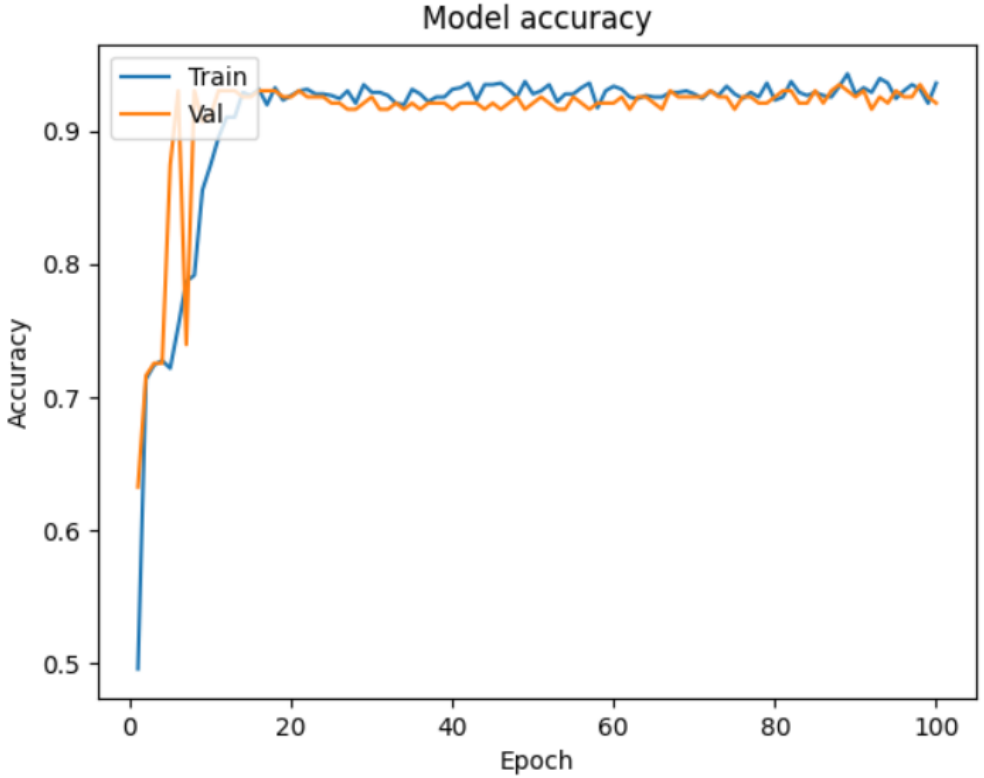
主要就是可以減少參數，降低overfit。

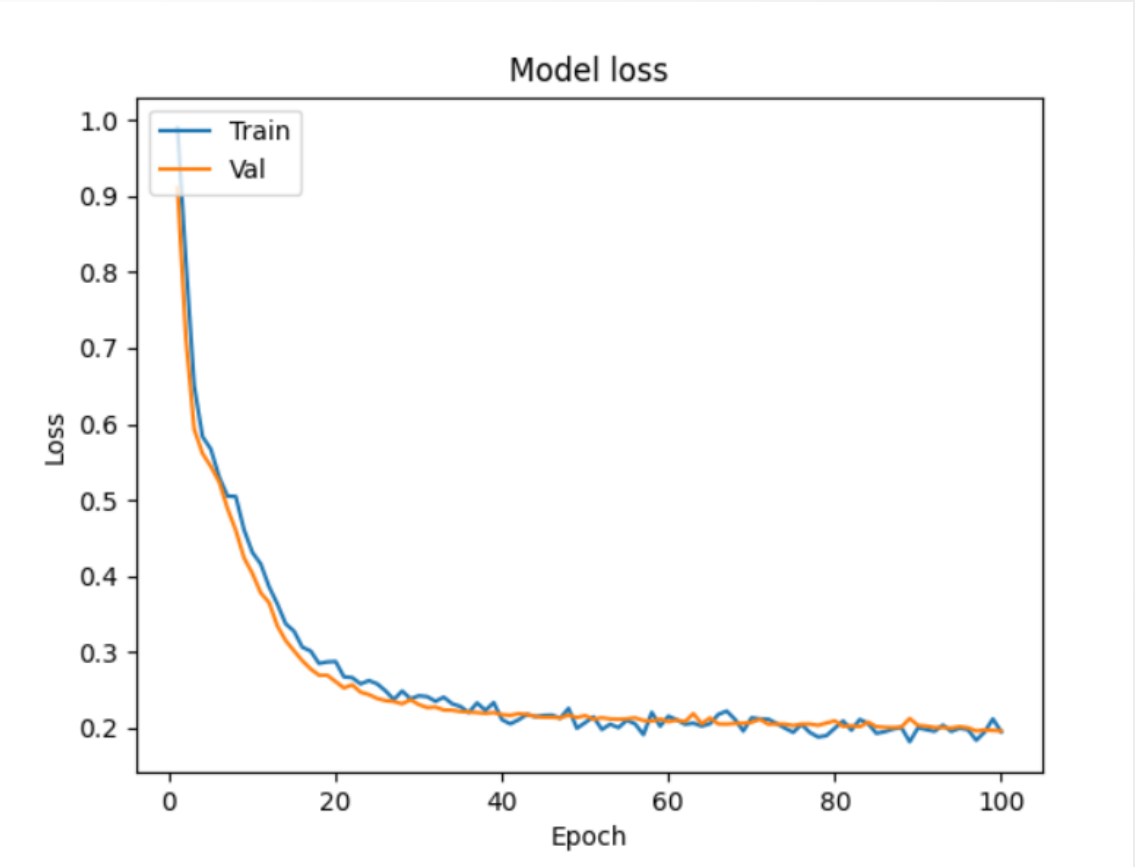
假設卷積層的最後輸出是h x w x d 的三維特徵圖，具體大小為6 x 6 x 3，經過GAP轉換後，變成了大小為 1 x 1 x 3 的輸出值，也就是每一層 h x w 會被平均化成一個值。





程　式　訓　練


Model accuracy
Train
Val
Accuracy
Epoch
0.5
0.6
0.7
0.8
0.9
0
20
40
60
80
100


Model loss
Train
Val
Loss
Epoch
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
0
20
40
60
80
100

2021

Thanks for watching

C N N

陳 芮 甯