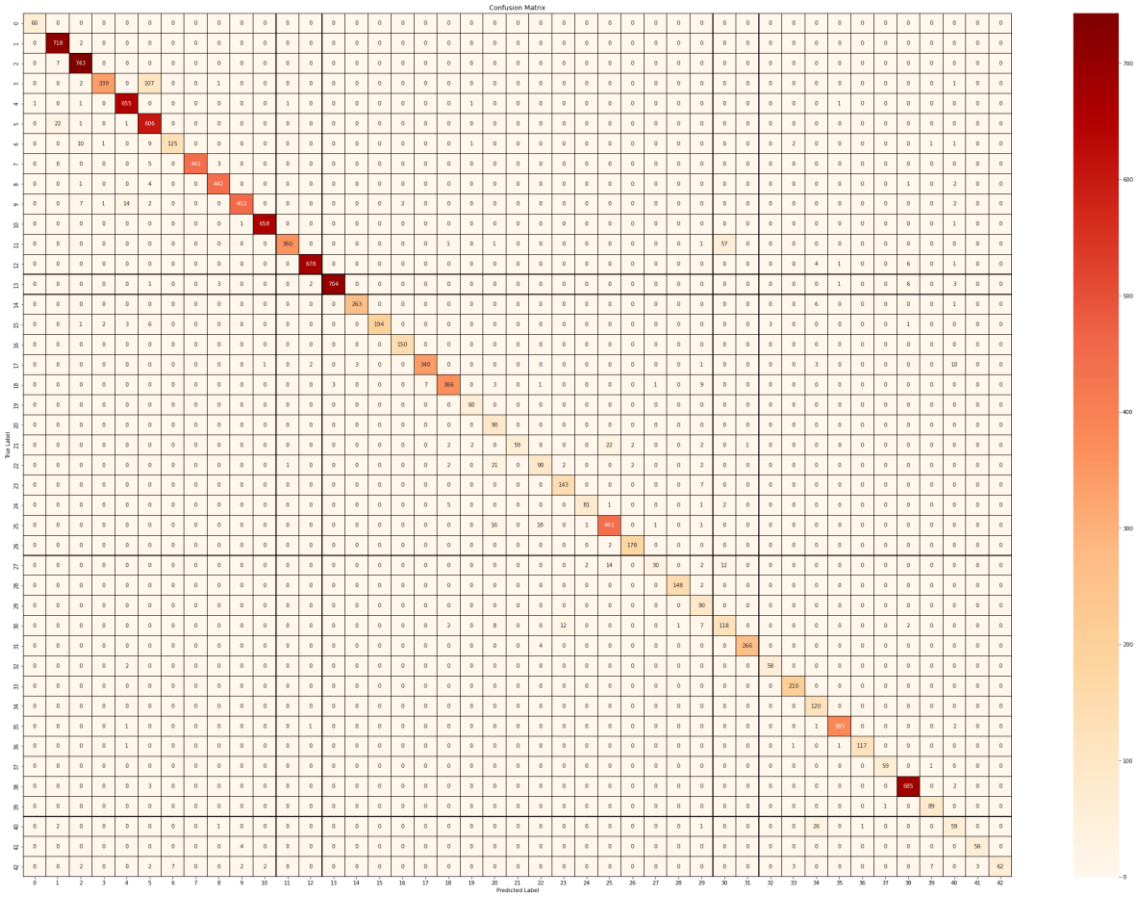
自動駕駛實務

交通號誌辨識作業

一、交通號誌分類結果

分類結果學生先以 Confusion Matrix 做視覺化的呈現， x 座標為 Predicted Label，y 座標為 True Label ，也就是說，在對角線上的元素越紅代表辨識結果越佳。

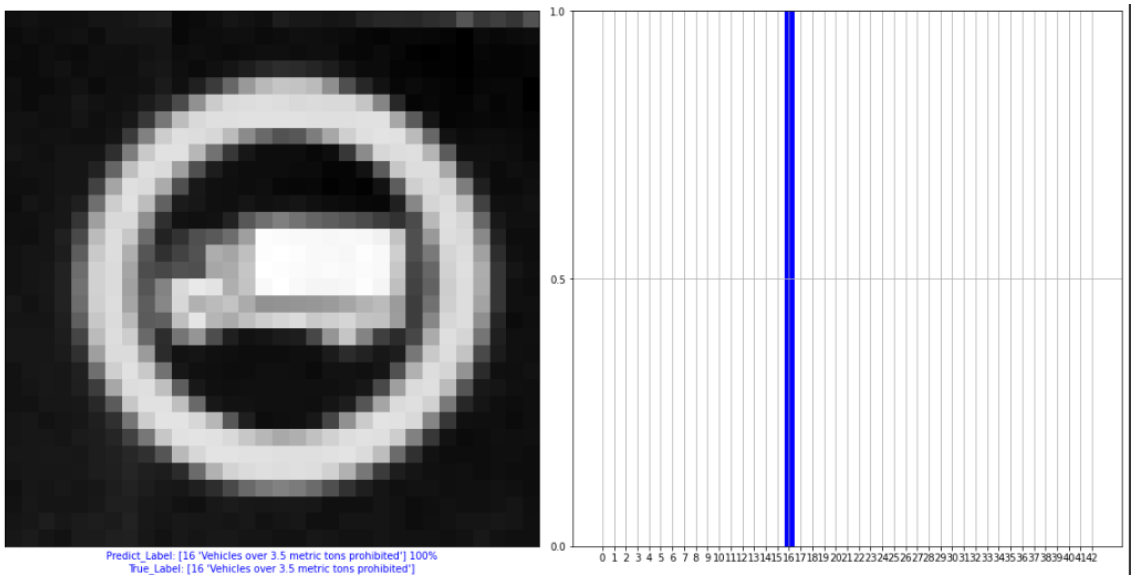
```
#Plot confusion_matrix
import seaborn as sn
pred = cnn.predict(grayscale_test)
pred_classes = np.argmax(pred, axis = 1)
true = np.argmax(y_test, axis = 1)
confusion_mtx = confusion_matrix(true, pred_classes)

f,ax = plt.subplots(figsize = (40,30))
sn.heatmap(confusion_mtx, annot=True, linewidths=0.1, cmap = 'OrRd', linecolor="black", fmt='g', ax=ax)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```

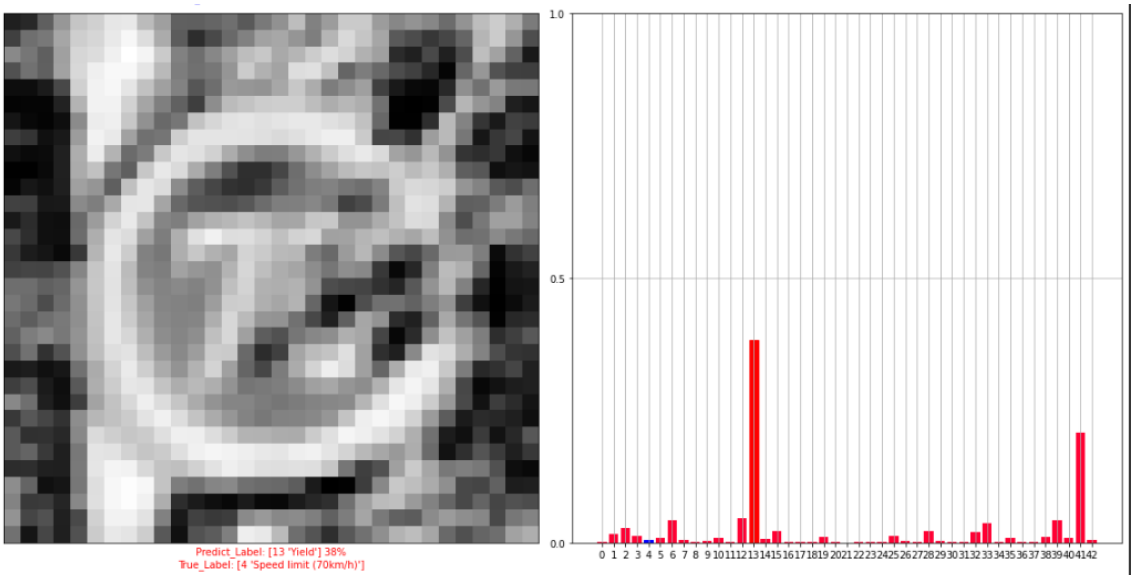


再來是 Test Accuracy 的部分，可以發現整體辨識準確度達到 0.964

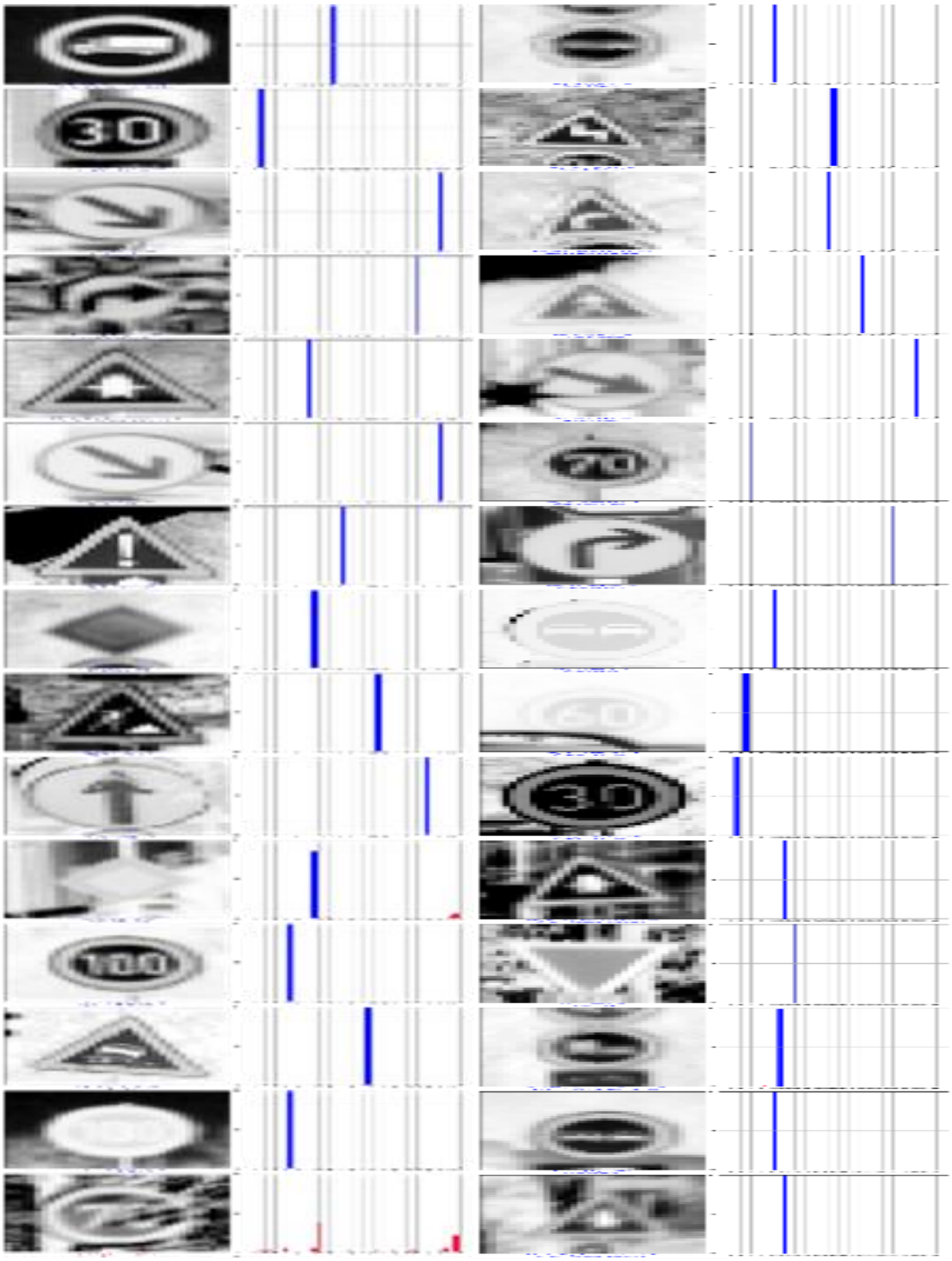
```
[21] #輸入數據(data)和黃金標準(label),然後將預測結果與黃金標準相比較,得到兩者誤差並輸出
     score = cnn.evaluate(grayscale_test, y_test, verbose=0)
     print("Test Loss:", score[0])
     print("Test Accuracy:", score[1])

     Test Loss: 0.17059145867824554
     Test Accuracy: 0.9638954997062683
```

以下是另一種視覺化的呈現方式，如果預測結果跟實際結果一致，左下角會呈現藍色字體，否則會標示紅色字體，代表預測錯誤，右邊的長條圖會將正確的類別顯示為藍色，其餘類別預測顯示為紅色，大小則為 0 至 1 的機率分布，以下圖為例，16 為正確類別，故右方的長條圖第 16 類會是藍色長條，而預測結果為 16 類的機率接近 1，故其餘類別幾乎看不到紅色長條，再下方則為預測結果錯誤的情形。





以下是實作的 Code 以及從測試集取 30 張圖的測試結果，從圖中可發現 30 張僅錯一張，跟 Test Accuracy 結果接近。



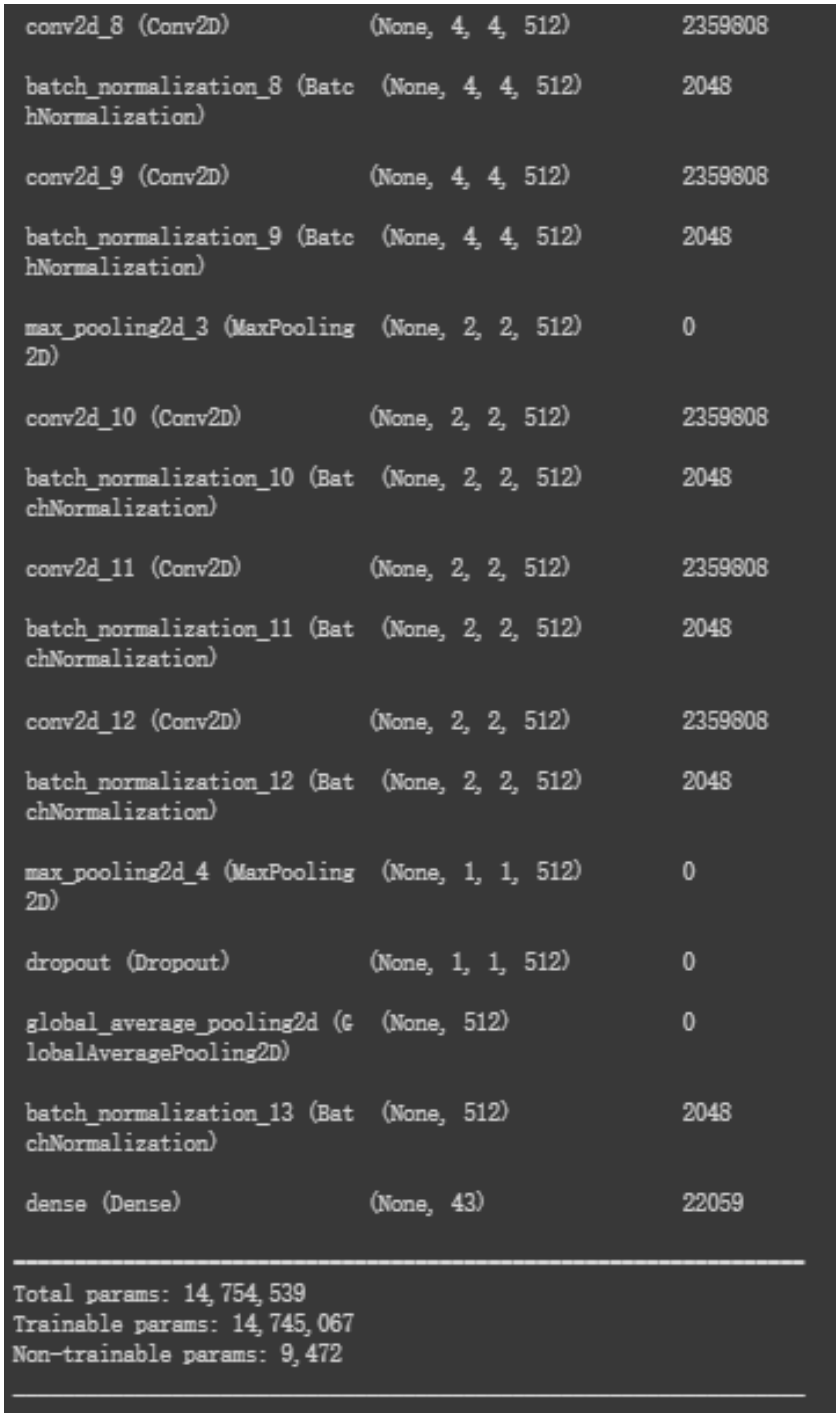
```
#繪製此圖以查看完整的 43 個類別預測集。 亦即 = 繪製predictions的全圖

#定義一個繪圖函數
def plot_image(i, predictions_array, true_label , img):
    true_label, img = true_label[i], img[i,:,:,0] #正確的標籤與圖像

    #定義軸的格線和刻度
    plt.grid(False)
    plt.xticks([]) #傳遞一個空列表會刪除所有X軸刻度。
    plt.yticks([])
    #繪圖
    plt.imshow(img, cmap=plt.cm.binary)
    #預測出來的label
    predicted_label = np.argmax(predictions_array) #array中的最大值

    if predicted_label == true_label:
        color = "blue"
    else:
        color = "red"
    #定義X軸的主題(預測的類別 預測的機率 真正的類別) :2.0f是取到浮點數小數點後2位
    plt.xlabel("Predict_Label: {} {:2.0f}%\nTrue_Label: {}".format(class_names[predicted_label],
                                                                   100*np.max(predictions_array),
                                                                   class_names[true_label]),
                                                                   color=color)

#定義一個可以將值可視化的函數
def plot_value_array(i, predictions_array, true_label):
    true_label = true_label[i] #正確的標籤
    plt.grid(True)
    plt.xticks(range(43)) #設定X軸刻度0-43
    plt.yticks([0,0.5,1]) #設定Y軸刻度
    #繪製長條圖
    thisplot = plt.bar(range(43), predictions_array, color="#ff0033") #深紅色
    #y軸的刻度限制
    plt.ylim([0, 1])
    #預測出來的label array中的最大值
    predicted_label = np.argmax(predictions_array)

    thisplot[predicted_label].set_color('red')
    thisplot[true_label].set_color('blue')
```

```
#讓我們用他們的預測繪製幾個圖像。 請注意，即使非常自信，模型也可能是錯誤的。 繪製prediction全圖
# Plot the first X test images, their predicted labels, and the true labels.
# Color correct predictions in blue and incorrect predictions in red.

#定義列跟行(兩張圖片為一組)
num_rows = 30
num_cols = 1

num_images = num_rows*num_cols #設定圖片張數總共30張圖片

plt.figure(figsize=(16*num_cols, 8*num_rows)) #設定總圖大小

for i in range(num_images):

    plt.subplot(num_rows, 2*num_cols, 2*i+1) #1/3/5/7/9/....
    plot_image(i, pred[i], true, grayscale_test)

    plt.subplot(num_rows, 2*num_cols, 2*i+2) #2/4/6/8/10/......
    plot_value_array(i, pred[i], true)

    plt.tight_layout()


plt.show()
```

二、模型架構和參數調整

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 64)	640
batch_normalization (BatchNormalization)	(None, 32, 32, 64)	256
conv2d_1 (Conv2D)	(None, 32, 32, 64)	36928
batch_normalization_1 (BatchNormalization)	(None, 32, 32, 64)	256
max_pooling2d (MaxPooling2D)	(None, 16, 16, 64)	0
conv2d_2 (Conv2D)	(None, 16, 16, 128)	73856
batch_normalization_2 (BatchNormalization)	(None, 16, 16, 128)	512
conv2d_3 (Conv2D)	(None, 16, 16, 128)	147584
batch_normalization_3 (BatchNormalization)	(None, 16, 16, 128)	512
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 128)	0
conv2d_4 (Conv2D)	(None, 8, 8, 256)	295168
batch_normalization_4 (BatchNormalization)	(None, 8, 8, 256)	1024
conv2d_5 (Conv2D)	(None, 8, 8, 256)	590080
batch_normalization_5 (BatchNormalization)	(None, 8, 8, 256)	1024
conv2d_6 (Conv2D)	(None, 8, 8, 256)	590080
batch_normalization_6 (BatchNormalization)	(None, 8, 8, 256)	1024
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 256)	0
conv2d_7 (Conv2D)	(None, 4, 4, 512)	1180160
batch_normalization_7 (BatchNormalization)	(None, 4, 4, 512)	2048



以上是學生的模型架構，基本上是套用 VGG16 的架構去設計，將全連接層的部分改為 GlobalAveragePooling，最後接上 43 類辨識器， Dropout 部分給 0.5，因為是灰階圖片，特徵相對全彩照片沒有那麼多，因次將一半萃取的特徵捨去。架構部分有參考過 VGG19 以上等更深層的網路去做訓練，但並未獲得明顯較佳的訓練結果，反而增加訓練的時間跟運算資源，因此最後選擇以此架構做訓練。

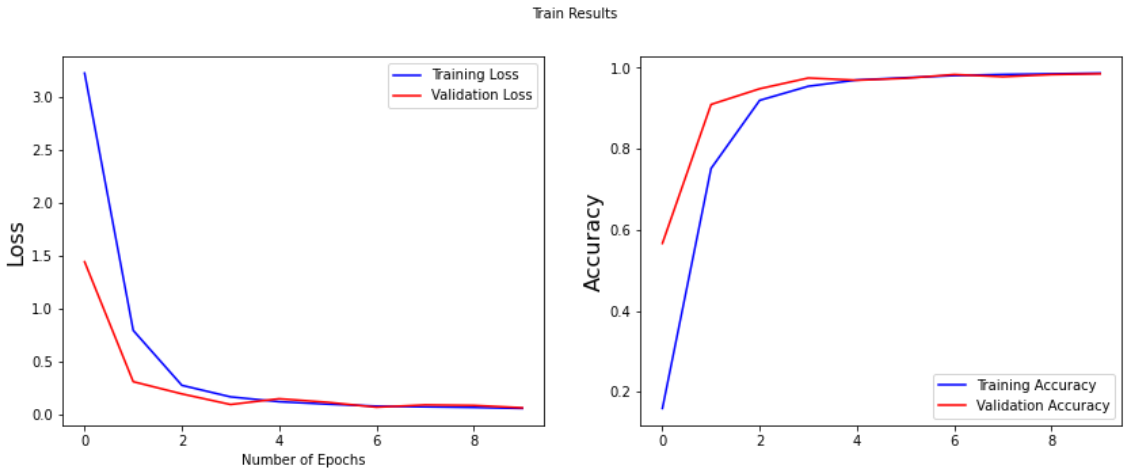
```
[16] # comiple model
cnn.compile(optimizer = tf.keras.optimizers.Adam(learning_rate=0.001),loss=tf.keras.losses.CategoricalCrossentropy(),metrics=['accuracy'])

[17] # Data Augmentation
datagen = ImageDataGenerator(featurewise_center=False,  # 以每一張feature map為單位將平均值設為0
                             samplewise_center=False,  # set each sample mean to 0
                             featurewise_std_normalization=False,  # 以每一張feature map為單位將數值除以其標準差(上述兩步驟就是我們常見的Standardization)
                             samplewise_std_normalization=False,  # 將輸入的每個樣本除以其自身的標準差。
                             zca_whitening=False,  # dimesion reduction
                             rotation_range=0.2,  # 隨機旋轉圖片
                             zoom_range = 0.2, #  隨機縮放範圍
                             width_shift_range=0.2,  #  水平平移，相對總寬度的比例
                             height_shift_range=0.2,  # 垂直平移，相對總高度的比例
                             horizontal_flip=False,  # 一半影象水平翻轉
                             vertical_flip=False)  # 一半影象垂直翻轉
datagen.fit(grayscale_train)

# # 載入最近的檢查點的權重
# cnn.load_weights(filepath='/content/drive/MyDrive/自動駕駛實務/TrafficSign_Classifier-P2/TrafficSign_Classifier.h5')

history = cnn.fit_generator(datagen.flow(grayscale_train, y_train, batch_size=16),shuffle=True,epochs=10,
                                                validation_data = (grayscale_valid, y_valid),
                                                # verbose = 2, #verbose=2過程全顯示
                                                steps_per_epoch=grayscale_train.shape[0] // 16,
                                                )

Epoch 1/10
2174/2174 [==============================] - 53s 19ms/step - loss: 3.2279 - accuracy: 0.1591 - val_loss: 1.4456 - val_accuracy: 0.5667
Epoch 2/10
2174/2174 [==============================] - 40s 18ms/step - loss: 0.7972 - accuracy: 0.7514 - val_loss: 0.3130 - val_accuracy: 0.9095
Epoch 3/10
2174/2174 [==============================] - 40s 19ms/step - loss: 0.2785 - accuracy: 0.9197 - val_loss: 0.1976 - val_accuracy: 0.9483
Epoch 4/10
2174/2174 [==============================] - 40s 18ms/step - loss: 0.1696 - accuracy: 0.9545 - val_loss: 0.0973 - val_accuracy: 0.9748
Epoch 5/10
2174/2174 [==============================] - 40s 19ms/step - loss: 0.1236 - accuracy: 0.9700 - val_loss: 0.1516 - val_accuracy: 0.9694
Epoch 6/10
2174/2174 [==============================] - 40s 18ms/step - loss: 0.1005 - accuracy: 0.9756 - val_loss: 0.1180 - val_accuracy: 0.9741
Epoch 7/10
2174/2174 [==============================] - 40s 19ms/step - loss: 0.0818 - accuracy: 0.9809 - val_loss: 0.0717 - val_accuracy: 0.9834
Epoch 8/10
2174/2174 [==============================] - 40s 19ms/step - loss: 0.0754 - accuracy: 0.9836 - val_loss: 0.0936 - val_accuracy: 0.9780
Epoch 9/10
2174/2174 [==============================] - 40s 19ms/step - loss: 0.0694 - accuracy: 0.9852 - val_loss: 0.0890 - val_accuracy: 0.9832
Epoch 10/10
2174/2174 [==============================] - 40s 18ms/step - loss: 0.0605 - accuracy: 0.9869 - val_loss: 0.0663 - val_accuracy: 0.9853
```

參數設定的部分:

learning_rate=0.001
optimizer = Adam
batch_size=16
epochs=10

Data Augmentation:

rotation_range=0.2, # 隨機旋轉圖片
zoom_range = 0.2, # 隨機縮放範圍
width_shift_range=0.2, # 水平平移，相對總寬度的比例
height_shift_range=0.2, # 垂直平移，相對總高度的比例



上方是我的 Loss 和 Accuracy 做圖曲線(下方是第 10 個 epoch 結果)
loss: 0.0605 - accuracy: 0.9869 - val_loss: 0.0663 - val_accuracy: 0.9853

三、實作過程中遇到的問題與討論延伸

實作過程中，共遇到**兩大難關**
第一個是在**模型訓練**上，原先想透過 Keras 套件做 Transfer learning，但發現 Transfer learning 輸入需為 rgb 的圖片，不僅運算的資源相對高，訓練時間也相對久，而這邊輸入是灰階圖，預訓練好的模型不適用，因此以手動一層一層建模，並透過測試選擇比較好的模型以及參數做訓練。
第二個是在**測試集結果視覺化**的部分花了不少時間，為了讓圖表一目了然(這部分在標題一已呈現)。此外，學生也試著對原圖雖機抽樣幾張做**銳利化(Sharpening)處理**，以下是 Code 和圖表

```
def  Sharpening(image):
    Kernel=np.array([[0,-1,0],[-1,5,-1],[0,-1,0]])
    sharpening_image  =  cv2.filter2D(src=image,  ddepth=-1,  kernel=Kernel)
    return  sharpening_image

#  make  new  images
test_new=np.zeros((20,32,32,3))
test_new=test_new.astype('uint8')
label_test_new=np.zeros((20,43))
fig,axs  =  plt.subplots(10,2,figsize=(20,20))
fig.subplots_adjust(hspace=.3,  wspace=.001)
axs  =axs.ravel()
for  i  in  range(1,11,1):
    index  =  random.randint(0,  n_test)
    title1=  'Test  '+str(index)+'  Label=  '+str(np.argmax(y_test[i-1]))
    title2=  'Sharpening  of  test  '+str(index)+'  Label=  '+str(np.argmax(y_test[i-1]))
    label_test_new[i-1]=y_test[i-1]
    sharpening_image  =  Sharpening(X_test[index])
    test_new[i-1]=sharpening_image
    axs[i*2-2].axis('off')
    axs[i*2-2].imshow(X_test[index])
    axs[i*2-2].set_title(title1)
    axs[i*2-1].axis('off')
    axs[i*2-1].imshow(sharpening_image)
    axs[i*2-1].set_title(title2)
```

Test 5129 Label= 16

Sharpening of test 5129 Label= 16

Test 1098 Label= 1

Sharpening of test 1098 Label= 1

Test 4533 Label= 38

Sharpening of test 4533 Label= 38

Test 10711 Label= 33

Sharpening of test 10711 Label= 33

Test 2094 Label= 11

Sharpening of test 2094 Label= 11

Test 7840 Label= 38

Sharpening of test 7840 Label= 38

Test 12597 Label= 18

Sharpening of test 12597 Label= 18

Test 418 Label= 12

Sharpening of test 418 Label= 12

Test 4562 Label= 25

Sharpening of test 4562 Label= 25

Test 5921 Label= 35

Sharpening of test 5921 Label= 35

當銳利化的結果去做 Top5 預測結果分析，會發現結果慘不忍睹，錯誤率極高，儘管辨識正確，準確率也很差，以下附 Code、Top5 結果和圖表

```
grayscale_test_new = tf.image.rgb_to_grayscale(test_new)
grayscale_test_new=grayscale_test_new.numpy()
trueY=np.zeros((10,1)).astype('uint8')
preds=np.zeros((10,1)).astype('uint8')
preds_prob=np.zeros((10,5)).astype('float')
top5=np.zeros((10,5)).astype("uint8")
print(grayscale_test_new.shape)
for i in range(10):
    trueY[i] = np.argmax(label_test_new[i])
    img = grayscale_test_new[i]
    test_img = img.reshape(1,32,32,1)

    #預測類別
    preds[i]=np.argmax(cnn.predict(test_img))
    # preds = np.argmax((cnn.predict(test_img)>0.5).astype("int32"))
    #預測機率
    prob = cnn.predict(test_img)
    prob_rank=np.argsort(prob)
    top5[i]=prob_rank[:,42:37:-1]
    for j in range(5):#record top5 probability
            preds_prob[i][j]=prob[:,top5[i][j]]

    print("trueY:  ",trueY[i])
    print("preds:  ",preds[i])
    print("top5:  ",top5[i])
    print("top5_Prob:  ",preds_prob[i])
    # print("all_Prob:  ",prob)
    print("**********************************************************************")
```

trueY: [16]

preds: [15]

top5: [15 3 33 32 34]

top5_Prob: [0.58485001 0.28671938 0.04789686 0.02374901 0.01658265]

****

trueY: [1]

preds: [18]

top5: [18 26 6 32 16]

top5_Prob: [9.99879599e-01 5.45942239e-05 3.43719694e-05 4.09529730e-06
 3.31501178e-06]

****

trueY: [38]

preds: [12]

top5: [12 38 4 17 0]

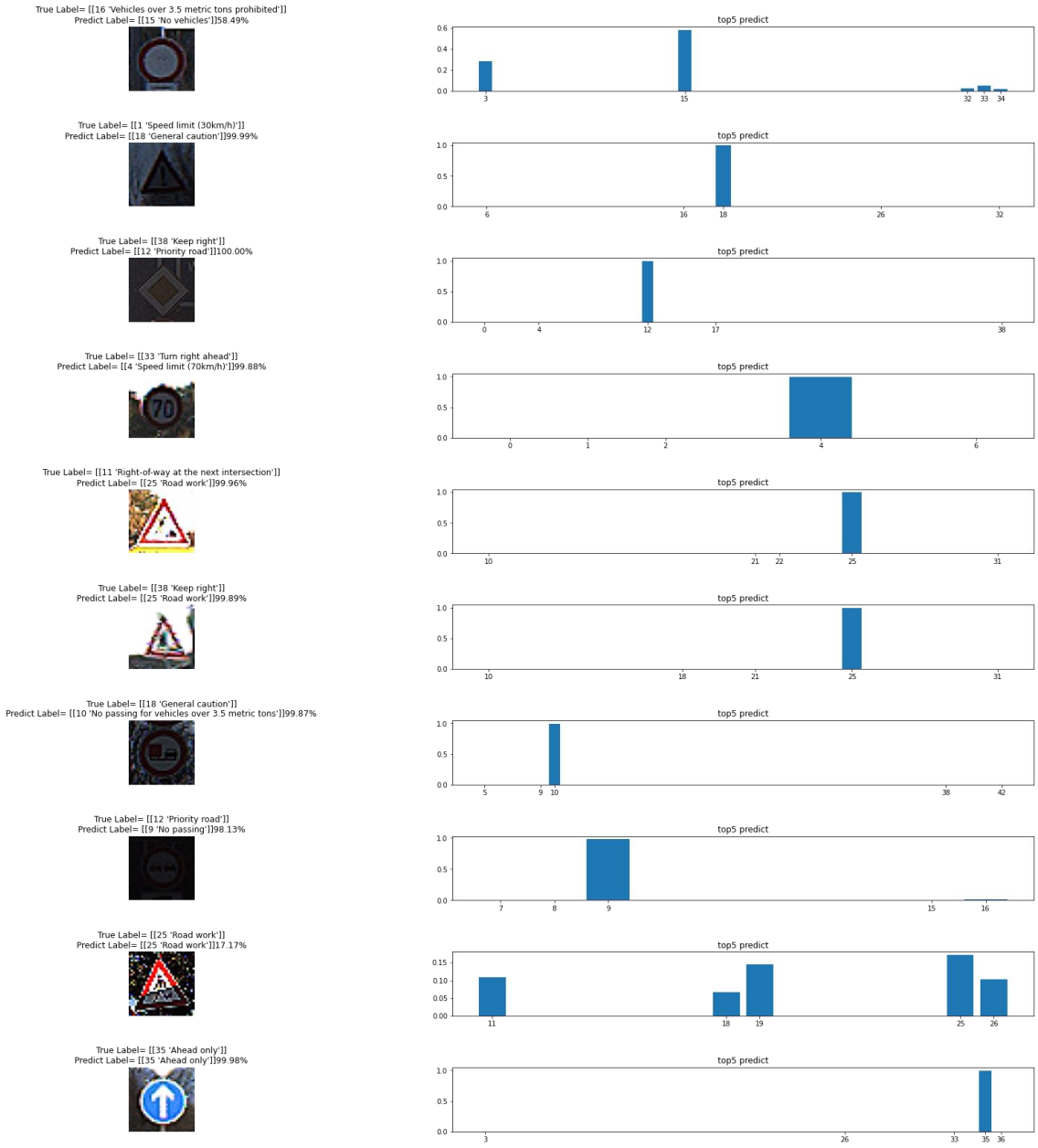
top5_Prob: [9.99994755e-01 3.23425570e-06 4.31867846e-07 3.92449181e-07
 3.24353920e-07]

****
trueY: [33]
preds: [4]
top5: [4 1 0 6 2]
top5_Prob: [9.98841822e-01 3.93480092e-04 1.83984754e-04 1.77136812e-04
5.92742435e-05]
****
trueY: [11]
preds: [25]
top5: [25 10 31 22 21]
top5_Prob: [9.99602973e-01 1.87449128e-04 9.14807824e-05 3.35816985e-05
1.92507559e-05]
****
trueY: [38]
preds: [25]
top5: [25 31 10 21 18]
top5_Prob: [9.98922110e-01 2.02398733e-04 1.89941915e-04 1.08331391e-04
9.68838576e-05]
****
trueY: [18]
preds: [10]
top5: [10 9 42 38 5]
top5_Prob: [9.98701930e-01 2.02633571e-04 1.31825262e-04 1.05802712e-04
8.76015038e-05]
****
trueY: [12]
preds: [9]
top5: [9 16 7 8 15]
top5_Prob: [9.81311142e-01 1.39353443e-02 1.25297101e-03 8.64681962e-04
4.78812319e-04]
****
trueY: [25]
preds: [25]
top5: [25 19 11 26 18]
top5_Prob: [0.17174086 0.14529784 0.10863332 0.10351466 0.0669952]
****
trueY: [35]
preds: [35]

top5: [35 26 3 36 33]

top5_Prob: [9.99796331e-01 1.03682840e-04 2.23626994e-05 1.16378187e-05
 9.97835377e-06]

```
fig, axs  =  plt.subplots(10, 2, figsize=(30, 30))
fig.subplots_adjust(hspace=.8,  wspace=.001)
axs  =axs.ravel()
class_names
for  i  in  range(1, 11, 1):
        percent=preds_prob[i-1][0]*100
        percent=format(percent, '.2f')
        title1  =  'True  Label=  '+  str(class_names[trueY[i-1]])  +'\nPredict  Label=  '+str(class_names[preds[i-1]])+str(percent)+'%'
        title2  ='top5  predict'
        axs[i*2-2].axis('off')
        axs[i*2-2].imshow(test_new[i-1])
        axs[i*2-2].set_title(title1)
        axs[i*2-1].bar(top5[i-1],  preds_prob[i-1])
        axs[i*2-1].set_title(title2)
        axs[i*2-1].set_xticks(list(top5[i-1]))
```



圖的部分上分文字列的是 True_Label、Predict_Label 和 Predict_top1 的百分比(可放大檢視)，表的部分 X 座標為預測的前五大可能 Label，Y 座標則為預測機率，由圖表可知，雖然在測試集的結果得到蠻高的準確度，但只要對圖片稍加處理，便可輕易讓模型預測結果相差十萬八千里

四、工具使用

利用 Google Colab 去做訓練和圖表分析

五、心得

這次 Project 不僅訓練學生對影像辨識的了解和 Keras、Tensorflow 套件的使用，同時也學到一些視覺化分析的技巧，了解到圖表的重要性。不僅僅是將模型訓練完即可，還要能分析訓練結果，並針對缺點做改善。感謝老師在授課的同時不時會展示一些實作，更是針對實作部份加入許多經驗談和解釋，帶學生一步一步去作有效率的學習。