Traffic Sign

第三組
111034542 張元豪
111034554 徐斯敏
110034549 林育模

Contents

04 結論與建議

03 網站

02 模型訓練與績效

01 背景介紹

PART ONE
背景與動機

背景介紹

根據交通部道安資訊查詢網歷年資料顯示，

違反號誌管制或指揮為國人十大死傷人數的肇事主因

交通素養對個人和社會都極為重要

使用CNN model製作一個分類不同類別的交通標誌，

幫助國人**了解各個交通號誌**，以達到降低交通事故死傷人數。

5W1H

WHO 想了解交通號誌的國人

WHAT 藉由本研究開發的網頁來供國人學習交通號誌

WHY 顧客台灣交通事故逐年攀升，違反號誌管制或指揮為國人十大死傷人數的肇事主因

WHEN 隨時使用網站都能查閱相關交通號誌資訊

WHERE 任何有出現交通號誌的地點

HOW 架設網站、深度學習

1-10月逾30萬件交通事故(111年)

必須努力推動道路安全，關於道路設計，也要積極改善

304267
交通事故 較去年同期 +55%

40000
受傷人數

6824
行人路口事故 較去年同期 +111%

8139
酒駕案件導致受傷

人行道的違停、斑馬線的縮減也應該列入修正考量

背景介紹

PART TWO

資料收集

- 交通號誌
- 58個類別
- 每個類別的資料數量差距非常大
  - 最多的類別有324張圖片
  - 最少的僅有2張

PART TWO

資料前處理

先將資料量少於10張的類別剔除
再將各類別資料依比例分為train、test以及validation sets

```
sdir=r'../input/traffic-sign-dataset-classification/traffic_Data/DATA'
csv_path=r'../input/traffic-sign-dataset-classification/labels.csv'
trsplit=.8
vsplit=.1
min_file_count = 10
train_df, test_df, valid_df= preprocess(sdir,csv_path,trsplit, vsplit, min_file_count)
```

```
There are  58 classes but the labels.csv file only has 58 entries

class  19  has been excludes because it has only  4  samples
class  25  has been excludes because it has only  2  samples
class  8  has been excludes because it has only  8  samples
class  53  has been excludes because it has only  2  samples
class  18  has been excludes because it has only  8  samples
class  57  has been excludes because it has only  6  samples
class  9  has been excludes because it has only  2  samples
class  51  has been excludes because it has only  8  samples
class  33  has been excludes because it has only  4  samples
train_df length:  3300   test_df length:  413   valid_df length:  413
[357, 259, 208, 155, 129, 125, 122, 120, 114, 110, 104, 102, 101, 94, 88, 80, 78, 77, 66, 64, 62, 56, 46, 45, 35, 34, 34, 32, 32, 29, 29, 27, 26, 26, 24, 24, 22, 21, 19, 18, 14, 14,
14, 14, 11, 11, 10, 10, 8]
```

PART TWO

資料前處理

```
total=0
gen=ImageDataGenerator(horizontal_flip=False,  rotation_range=20, width_shift_range=.2,
                              height_shift_range=.2, zoom_range=.2)
groups=train_df.groupby('labels') # group by class
for label in train_df['labels'].unique():  # for every class
    group=groups.get_group(label)  # a dataframe holding only rows with the specified label
    sample_count=len(group)   # determine how many samples there are in this class
    if sample_count< max_samples: # if the class has less than target number of images
        aug_img_count=0
        delta=max_samples-sample_count  # number of augmented images to create
        target_dir=os.path.join(aug_dir, label)  # define where to write the images
        aug_gen=gen.flow_from_dataframe( group,  x_col='filepaths', y_col=None, target_size=image_size,
                                        class_mode=None, batch_size=1, shuffle=False,
                                        save_to_dir=target_dir, save_prefix='aug-', color_mode='rgb',
                                        save_format='jpg')
        while aug_img_count<delta:
            images=next(aug_gen)
            aug_img_count += len(images)
        total +=aug_img_count
```

Create Train, Test and Validation Generators


```
channels=3
batch_size=40
img_shape=(img_size[0], img_size[1], channels)
length=len(test_df)
test_batch_size=sorted([int(length/n) for n in range(1,length+1) if length % n ==0 and length/n<=80],reverse=True)[0]
test_steps=int(length/test_batch_size)
print ( 'test batch size: ' ,test_batch_size, '  test steps: ', test_steps)
def scalar(img):
    return img  # EfficientNet expects pixelsin range 0 to 255 so no scaling is required
trgen=ImageDataGenerator(preprocessing_function=scalar, horizontal_flip=True)
tvgen=ImageDataGenerator(preprocessing_function=scalar)
msg='                                                          for the train generator'
print(msg, '\r', end='')
train_gen=trgen.flow_from_dataframe( train_df, x_col='filepaths', y_col='labels', target_size=img_size, class_mode='categorical',
                                    color_mode='rgb', shuffle=True, batch_size=batch_size)
msg='                                                          for the test generator'
print(msg, '\r', end='')
test_gen=tvgen.flow_from_dataframe( test_df, x_col='filepaths', y_col='labels', target_size=img_size, class_mode='categorical',
                                    color_mode='rgb', shuffle=False, batch_size=test_batch_size)
msg='                                                         for the validation generator'
print(msg, '\r', end='')
valid_gen=tvgen.flow_from_dataframe( valid_df, x_col='filepaths', y_col='labels', target_size=img_size, class_mode='categorical',
                                    color_mode='rgb', shuffle=True, batch_size=batch_size)
classes=list(train_gen.class_indices.keys())
class_count=len(classes)
train_steps=int(np.ceil(len(train_gen.labels)/batch_size))
```

```
test batch size:  59   test steps:  7
Found 3300 validated image filenames belonging to 49 classes. for the train generator
Found 413 validated image filenames belonging to 49 classes.  for the test generator
Found 413 validated image filenames belonging to 49 classes. for the validation generator
```

PART TWO

訓練設定

- 最大epochs: 30
- 如果持續1個epoch訓練結果沒有改善就會將learning rate減半
- 如果持續3個epochs訓練結果沒有改善就會直接停止訓練
- 訓練acc小於90%時會以訓練acc作為目標，如果高於90%則改以validation loss作為判斷依據

```
epochs =30
patience= 1 # number of epochs to wait to adjust lr if monitored value does not improve
stop_patience =3 # number of epochs to wait before stopping training if monitored value does not improve
threshold=.9 # if train accuracy is < threshhold adjust monitor accuracy, else monitor validation loss
factor=.5 # factor to reduce lr by
freeze=False # if true free weights of  the base model
batches=train_steps
callbacks=[LRA(model=model,base_model= base_model,patience=patience,stop_patience=stop_patience, threshold=threshold,
                   factor=factor,dwell=dwell, batches=batches,initial_epoch=0,epochs=epochs, ask_epoch=ask_epoch )]
```

PART TWO

MODEL

模擬訓練與績效

```
model_name='EfficientNetB3'
base_model=tf.keras.applications.EfficientNetB3(include_top=False, weights="imagenet",input_shape=img_shape, pooling='max')
x=base_model.output
x=keras.layers.BatchNormalization(axis=-1, momentum=0.99, epsilon=0.001 )(x)
x = Dense(256, kernel_regularizer = regularizers.l2(l = 0.016),activity_regularizer=regularizers.l1(0.006),
                bias_regularizer=regularizers.l1(0.006) ,activation='relu')(x)
x=Dropout(rate=.45, seed=123)(x)
output=Dense(class_count, activation='softmax')(x)
model=Model(inputs=base_model.input, outputs=output)
model.compile(Adamax(learning_rate=.01), loss='categorical_crossentropy', metrics=['accuracy'])
```

PART TWO

實驗設計Design Of Experiment

Model	Optimizer	Begin Learning Rate
EfficientNetB3	Adam	0.01
MobileNetV2	Adamax	0.001
ResNet50	Adadelta	

PART TWO

實驗設計Design Of Experiment

Model	Optimizer	Begin LR	Final LR	Epoch	Train Loss	Train Acc	Test Loss	Test Acc
EfficientNetB3	Adam	0.001	0.00003	30	0.283	99.97	0.25369	100
EfficientNetB3	Adam	0.01	0.00062	30	0.318	97.97	0.3056	97.579
EfficientNetB3	Adamax	0.001	0.00025	30	0.354	100	0.32379	99.758
EfficientNetB3	Adamax	0.01	0.00031	30	0.301	99.848	0.27093	99.758
EfficientNetB3	Adadelta	0.001	0.0005	30	0.314	99.97	0.29347	99.758
EfficientNetB3	Adadelta	0.01	0.00062	30	0.276	99.818	0.23981	99.516
MobileNetV2	Adam	0.001	0.00003	30	0.243	100	0.2192	99.758
MobileNetV2	Adam	0.01	0.00125	17	0.563	93.303	10.52687	3.874
MobileNetV2	Adamax	0.001	0.00025	30	0.273	100	0.25468	99.758
MobileNetV2	Adamax	0.01	0.00062	30	0.391	99.576	0.41859	97.579
MobileNetV2	Adadelta	0.001	0.00025	30	9.883	41.606	9.41456	56.659
MobileNetV2	Adadelta	0.01	0.01	30	5.856	99.03	5.75771	99.031
ResNet50	Adam	0.001	0.00003	30	0.275	100	0.25016	99.758
ResNet50	Adam	0.01	0.005	30	1.117	78.091	1.11618	79.177
ResNet50	Adamax	0.001	0.0005	30	0.316	100	0.28796	99.758
ResNet50	Adamax	0.01	0.005	30	0.771	91.697	3.2055	90.073
ResNet50	Adadelta	0.001	0.001	30	8.95	77.606	8.75035	87.409
ResNet50	Adadelta	0.01	0.01	30	4.729	100	4.635	99.758

模擬訓練結績效

PART TWO

實驗設計Design Of Experiment

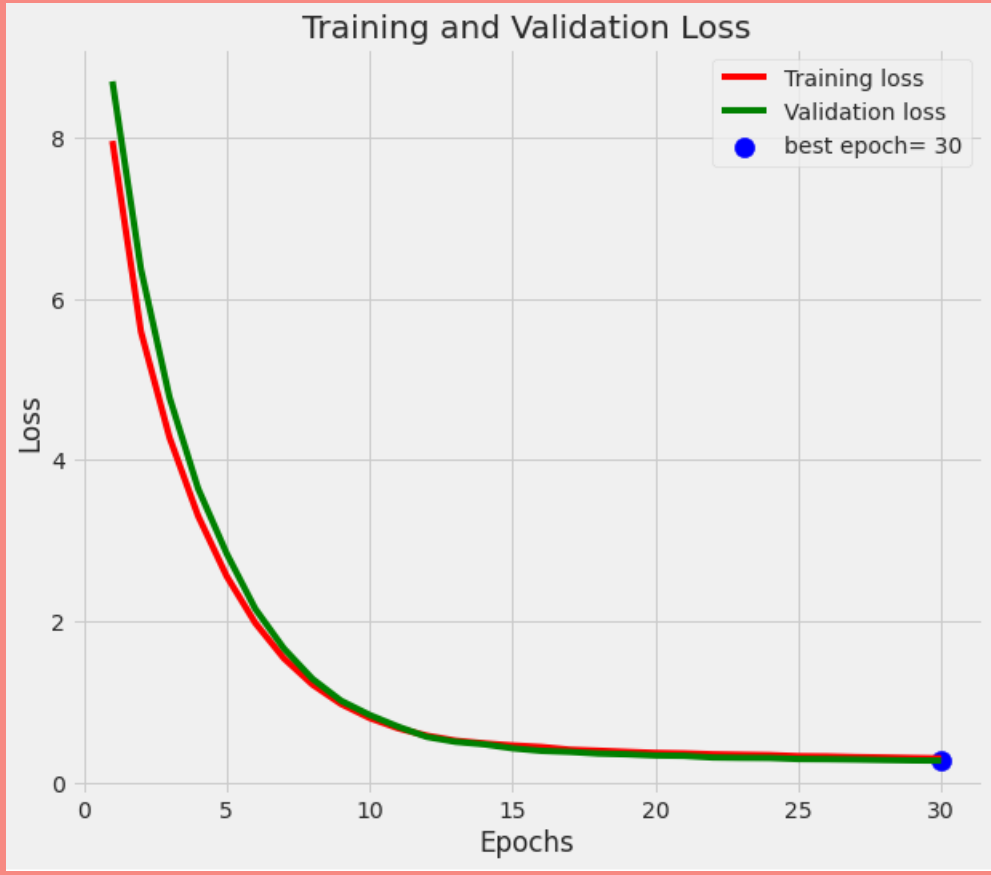
模擬訓練與績效

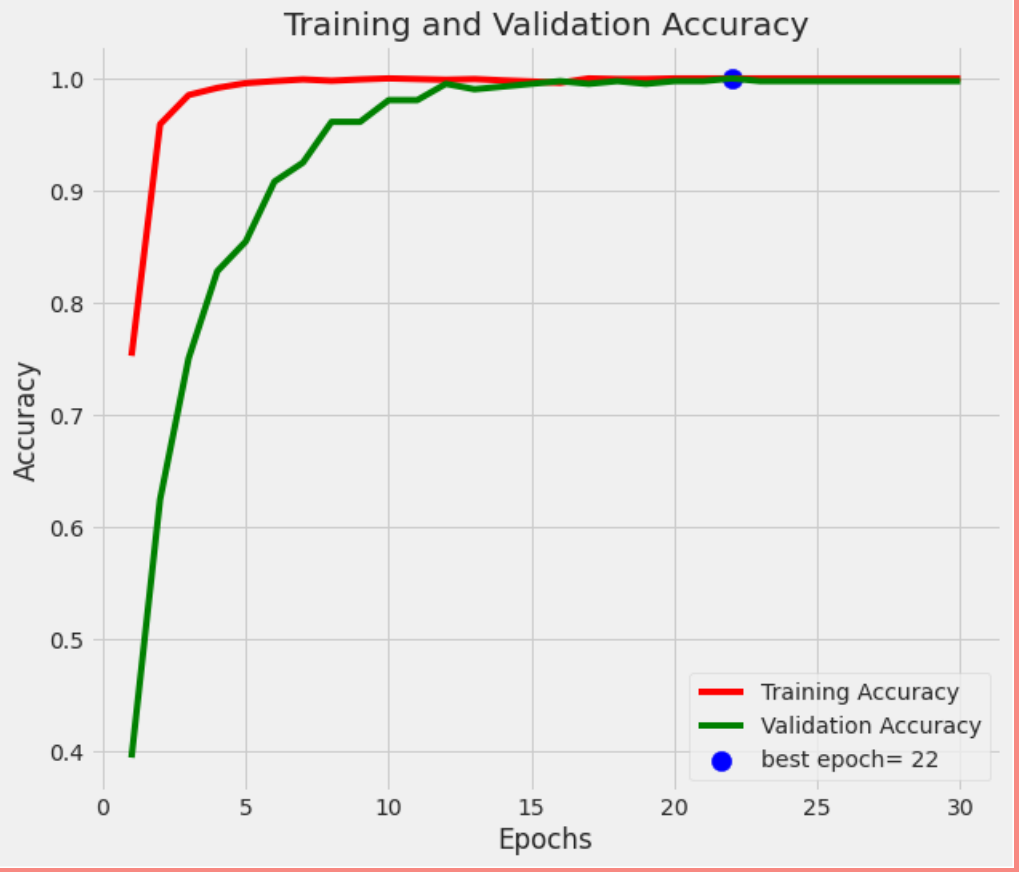
Model	Data on Keras Applications				Practical Implement	
	size(mb)	parameters	depth	Time (ms) per inference step (GPU)	Model Loading Time(s)	Inference Time(s)
EfficientNetB3	48	12.3M	210	8.8	8	4
MobileNetV2	14	3.5M	105	3.8	3	2
ResNet50	98	25.6M	107	4.6	9	2

Model	Optimizer	Begin LR	Final LR	Epoch	Train Loss	Train Acc	Test Loss	Test Acc	Training Time
MobileNetV2	Adam	0.001	0.00003	30	0.243	100	0.2192	99.758	6min 20s
MobileNetV2	Adamax	0.001	0.00025	30	0.273	100	0.25468	99.758	5min 58s

PART TWO

實驗設計Design Of Experiment





模擬訓練與績效

PART THREE

網站架構

網站

主頁

index.php

管理者後台

manager_login.php → checkpwd.php → manager.php → manager_crud.php → logout.php → index.php

後端登入相關資料

Login.php

PART THREE

後端Table

網站

traffic_signal

img_id(PK)

name

url

管理員帳號密碼

id(PK)

account

password

name

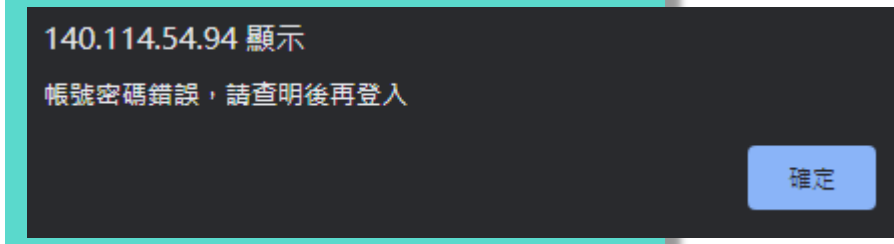
comment

PART THREE

管理者後台

點選index.php中的管理者後台
進入manager_login.php

帳密正確則進入manager.php
帳密錯誤則提示



結束

PART THREE

管理者後台

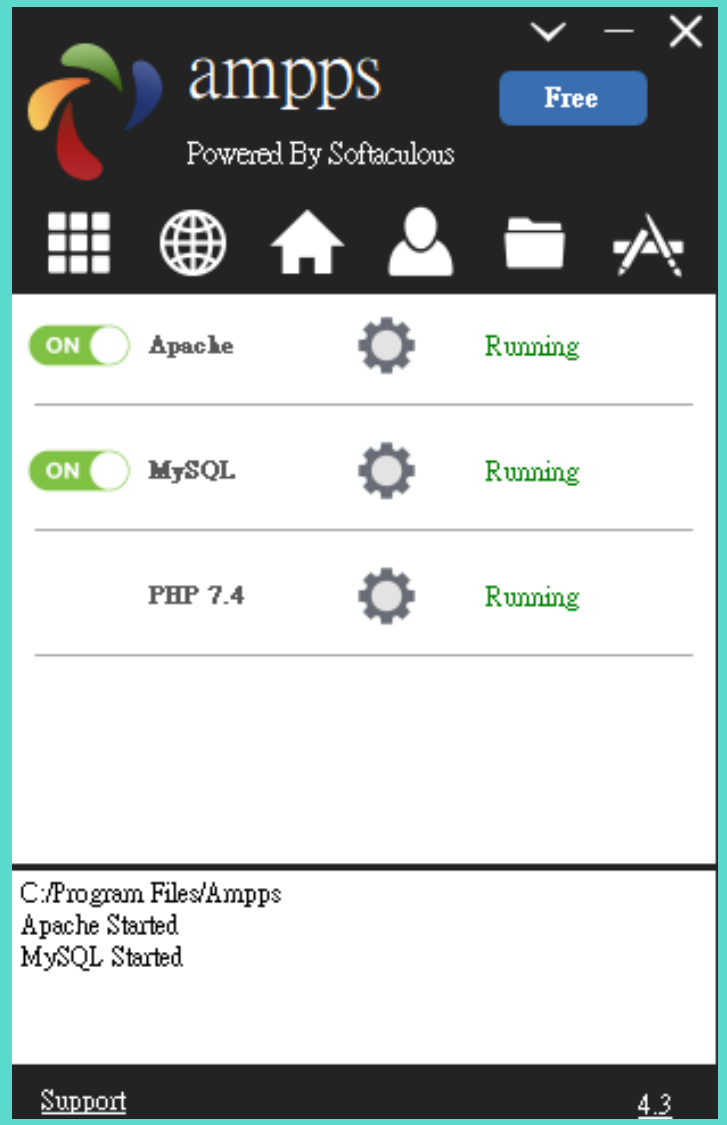
根據不同管理員登入帳號
判斷現在是哪位管理員

管理員 管理員2號 登出

結束

PART THREE

Ampps



因為py檔案中需要Import一些 package，但是沒有Server端的anaconda env詳細位置，因此使用Local端執行圖片辨識的php程式碼

結束

PART THREE

圖像識別呈現

網站

PART FOUR

結論

1.根據Doe參數的設置，本研究已經在測試集上達到了100%的準確率
2.最後依照訓練和推論時間選出的組合是MobileNetV2+Adamax+lr0.001
3.網站架設讓使用者將圖片匯入就能快速辨別交通號誌

建議

1. 本研究僅能幫助用路人提升交通號誌識別能力，無法得知用路人是否實際遵守交通安全法規，僅能透過提升交通符號識別能力以逐漸達成更安全的用路環境
2. 若圖像準確率不夠高或使用者輸入的圖像不在訓練的類別裡面，結果僅將隨機匯出一不正確的答案 ，容易會讓使用者誤導。
3. 之後可以增加資料的種類和數量，以達到更完善並準確的結果
4. 增加會員功能讓使用者可以看見之前的圖片記錄，方便學習
5. 沒有將預測完就刪除圖片功能完成，導致先前預測的結果消失。
改善這點才能確保每一位使用者即使輸入的圖片名稱相同也能成功預測

THANK YOU!