
IKEA鯊鯊
防盜機器影像辨識訓練

動機

因為鯊鯊很可愛
喜歡不需要理由

IKEA鯊鯊一炮而紅後

市面上盜版鯊鯊猖獗

尤其網路上販賣的商品又未能親自確認過

希望能藉由AI辨識網路上販賣之鯊鯊

是否為正版的IKEA鯊鯊

避免消費者受害

目標

能精準分辨出圖片中的

IKEA鯊鯊

終極目標是可以辨識出網路上圖片內鯊鯊

是否為正版IKEA鯊鯊

分工

陳欣渝
訓練集蒐集、簡報製作、講稿

蔡承翰
訓練集蒐集、主程式、資料分析、講稿

涂宜伶
訓練集蒐集、簡報製作、講稿、報告

蕭郁涵
訓練集蒐集、主程式、資料分析、講稿

訓練預期成果

第一階段

能分辨出圖片中的物件

"IKEA鯊鯊"

"真的大白鯊"

"其他物件"

第二階段

提高機器辨識之準確度

第三階段

能詳細精準辨識出真、偽IKEA鯊鯊

流程

資料集蒐集 → 程式撰寫 → 機器訓練參數測試 → 資料分析 → 成果結論

資料集蒐集 ➡ 程式撰寫 ➡ 資料分析 ➡ 成果與結論 ➡ ppt製作

IKEA_shark

約12,000張

Real_shark

約10,272張

Other(其他動物)

約10,000張

訓練 & 測試集

數量比 9 : 1

```
IKEA _shark_train   9000
IKEA _shark_test    1000
-
Real_shark_train   9000
Real_shark_test    1000
-
Other_train   9000
Other_test    1000
```

程式撰寫

```
import cv2 import os import numpy as np import
matplotlib.pyplot as plt import random from
keras.utils import np_utils
np.empty((9000,3,32,32),dtype="uint8")
realshark_train_data =
np.empty((9000,3,32,32),dtype="uint8")
others_train_data =
np.empty((9000,3,32,32),dtype="uint8")
ikeashark_train_label =
np.empty((9000,),dtype="uint8") in_label =
np.empty((9000,),dtype="uint8")
ikeashark_test_data =
np.empty((1000,3,32,32),dtype="uint8")
realshark_test_data =
np.empty((1000,3,32,32),dtype="uint8")
others_test_data =
np.empty((1000,3,32,32),dtype="uint8")
ikeashark_test_label =
np.empty((1000,),dtype="uint8")
realshark_test_label =
np.empty((1000,),dtype="uint8") others_test_label
= np.empty((1000,),dtype="uint8") # 1-1.
```

```
model.add(Flatten()) model.add(Dropout(rate=0
model.add(Dense(128, activation='relu'))
model.add(Dropout(rate=0.25))
model.add(Dense(3, activation='softmax'))
print(model.summary())
model.compile(loss='categorical_crossentropy
am', metrics=['accuracy'])
train_history=model.fit(x_train_normalize, y_
validation_split=0.3, epochs=200, batch_size=
```

```
import matplotlib.pyplot as plt def show_train_history(train_history):
plt.plot(train_history.history['accuracy'])
plt.plot(train_history.history['val_accuracy']) plt.title('Train History')
plt.ylabel('Accuracy') plt.xlabel('Epoch') plt.legend(['train', 'test'],
```

程式撰寫

-- 圖片轉檔

利用cv2.resize()

將訓練集及測試集的圖片

強制轉成32*32的大小

避免訓練時間過長

```
import cv2
import os
import numpy as np
data_train = np.empty((100000,3,32,32),dtype="uint8")
train_imgs = os.listdir('drive/Shareddrives/鯊鯊/其他 fish dog cat/fish')
for i in range(0,len(train_imgs)):
    img_1 = cv2.imread('drive/Shareddrives/鯊鯊/其他 fish dog cat/fish/'+train_imgs[i])
    if img_1 is None:#如果圖片數據是空的，就跳過
        continue
    new_img_1 = cv2.resize(img_1,(32,32),interpolation=cv2.INTER_LINEAR)
    data_train[i,:,:,:] = [new_img_1[:,:,0],new_img_1[:,:,1],new_img_1[:,:,2]]
    cv2.imwrite('drive/Shareddrives/鯊鯊/Others_train/new_'+train_imgs[i], new_img_1)
```

原圖片

新圖片
(32x32)

程式撰寫
-- 模型訓練

1. 資料前處理

2. 建立模型

3. 建立神經網路

4. 訓練模型

5. 為訓練集作圖

6. 評估模型準確率

7. 進行預測

資料前處理

np.empty空陣列,

儲存圖片data以及圖片label

IKEAshark、Realshark、Others

train 9000 ； test 1000

```
# 1. 資料前處理

'''
訓練集分為三大類：IKEA鯊魚、真鯊魚、其他各9000張，測試集各1000張
利用np.empty()去做出所需的空陣列，用來儲存圖片data以及其圖片label
所有圖片大小皆已轉為32x32大小
'''

ikeashark_train_data = np.empty((9000,3,32,32),dtype="uint8")
realshark_train_data = np.empty((9000,3,32,32),dtype="uint8")
others_train_data = np.empty((9000,3,32,32),dtype="uint8")

ikeashark_train_label = np.empty((9000,),dtype="uint8")
realshark_train_label = np.empty((9000,),dtype="uint8")
others_train_label = np.empty((9000,),dtype="uint8")

ikeashark_test_data = np.empty((1000,3,32,32),dtype="uint8")
realshark_test_data = np.empty((1000,3,32,32),dtype="uint8")
others_test_data = np.empty((1000,3,32,32),dtype="uint8")

ikeashark_test_label = np.empty((1000,),dtype="uint8")
realshark_test_label = np.empty((1000,),dtype="uint8")
others_test_label = np.empty((1000,),dtype="uint8")
```

os.listdir 將所有的檔名存成一串列

for 迴圈將圖片讀進程式中

圖片轉為 RGB 影像

判斷式防止迴圈 bug

讀取進來的圖片存入 data 中

Ikeashark_train_label ＝ 0

Realshark_train _label＝ 1

Other_train_label ＝ 2

```
# 1-1. 訓練集檔案輸入 ('@@'為路徑)

'''
先使用os.listdir()將資料夾中所有的檔名存成一串列，再利用不同資料夾去進行三種訓練集圖片資料的讀
取，並分別存在三個data陣列中
'''

ikeashark_train_names = os.listdir('./train/IKEA_shark_train')
for i in range(0,len(ikeashark_train_names)):
    img = cv2.imread('./train/IKEA_shark_train/'+ikeashark_train_names[i])
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) # 將圖片顏色轉為正常
    if img is None: # 讀取不到圖片則跳過此次迴圈，防bug用
        continue
    ikeashark_train_data[i,:,:,:] = [img[:,:,0] , img[:,:,1] , img[:,:,2]]
    ikeashark_train_label[i] = 0 # label:0 代表此張圖片為IKEA鯊魚

realshark_train_names = os.listdir('./train/Real_shark_train')
for i in range(0,len(realshark_train_names)):
    img = cv2.imread('./train/Real_shark_train/'+realshark_train_names[i])
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    if img is None:
        continue
    realshark_train_data[i,:,:,:] = [img[:,:,0] , img[:,:,1] , img[:,:,2]]
    realshark_train_label[i] = 1 # label:1 代表此張圖片為real鯊魚

others_train_names = os.listdir('./train/Others_train')
for i in range(0,len(others_train_names)):
    img = cv2.imread('./train/Others_train/'+others_train_names[i])
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    if img is None:
        continue
    others_train_data[i,:,:,:] = [img[:,:,0] , img[:,:,1] , img[:,:,2]]
    others_train_label[i] =2 # label:2 代表此張圖片屬於others
```

而測試集也和訓練集一樣

我們將三類別分別讀取並存入data中

```
# 1-2. 測試集檔案輸入

'''
基本上與訓練集的蒐集方式相同
'''

ikeashark_test_names = os.listdir('./test/IKEA_shark_test')
for i in range(0,len(ikeashark_test_names)):
    img = cv2.imread('./test/IKEA_shark_test/'+ikeashark_test_names[i])
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    if img is None:
        continue
    ikeashark_test_data[i,:,:,:] = [img[:,:,0] , img[:,:,1] , img[:,:,2]]
    ikeashark_test_label[i] = 0

realshark_test_names = os.listdir('./test/Real_shark_test')
for i in range(0,len(realshark_test_names)):
    img = cv2.imread('./test/Real_shark_test/'+realshark_test_names[i])
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    if img is None:
        continue
    realshark_test_data[i,:,:,:] = [img[:,:,0] , img[:,:,1] , img[:,:,2]]
    realshark_test_label[i] = 1

others_test_names = os.listdir('./test/Others_test')
for i in range(0,len(others_test_names)):
    img = cv2.imread('./test/Others_test/'+others_test_names[i])
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    if img is None:
        continue
    others_test_data[i,:,:,:] = [img[:,:,0] , img[:,:,1] , img[:,:,2]]
    others_test_label[i] = 2
```

vstack縱向合併將三類

訓練集合併為 x_train

測試集合併為 x_test

訓練集label合併為 y_train

測試集label合併為 y_test

```
# 1-3. 資料合併與打亂

'''
先進行三個data陣列與三個label陣列的合併，再利用shuffle()將其打亂 (x為資料，y為label，也就是答案)
'''

x_train =
np.vstack((ikeashark_train_data,realshark_train_data,others_train_data))
x_test = np.vstack((ikeashark_test_data,realshark_test_data,others_test_data))
y_train =
np.concatenate((ikeashark_train_label,realshark_train_label,others_train_label))
y_test =
np.concatenate((ikeashark_test_label,realshark_test_label,others_test_label))
```

transpose

將原本第二維的資料移至最後一維

三、四維的資料往前補

利用shuffle

將train、test的所有元素隨機排序

```
x_train = x_train.transpose(0,2,3,1)
x_test = x_test.transpose(0,2,3,1)

index_1 = [i for i in range(len(x_train))]
random.shuffle(index_1)
x_train = x_train[index_1]
y_train = y_train[index_1]

index_2 = [i for i in range(len(x_test))]
random.shuffle(index_2)
x_test = x_test[index_2]
y_test = y_test[index_2]
```

訓練集與測試集做正規化

label 做 onehot 編碼

有三個類別，後面這裡我們將其設為 3

```
# 1-4. 將資料做正規化，且將label做onehot編碼

x_train_normalize = x_train.astype('float32') / 255.0
x_test_normalize = x_test.astype('float32') / 255.0

y_train_OneHot = np_utils.to_categorical(y_train,3) # 3->三種分類
y_test_OneHot = np_utils.to_categorical(y_test,3)
```

建立模型

卷積層與池化層各三層

filters 分別設為 64，128, 256

kernal 設為 3×3, 激活函數皆為 relu

圖片大小 32×32, Dropout 設為 0.25

最後設定了幾個 maxpooling 防止 overfitting

```
# 2. 建立模型

from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D, ZeroPadding2D

model = Sequential()

# 卷積層1與池化層1

model.add(Conv2D(filters=64,kernel_size=(3,3),
                 input_shape=(32,32,3),
                 activation='relu',
                 padding='same'))

model.add(Dropout(rate=0.25))

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

# 卷積層2與池化層2

model.add(Conv2D(filters=128, kernel_size=(3, 3),
                 activation='relu', padding='same'))

model.add(Dropout(0.25))

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

# 卷積層3與池化層3

model.add(Conv2D(filters=256, kernel_size=(3, 3),
                 activation='relu', padding='same'))

model.add(Dropout(0.25))

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

建立神經網路

使用 flatten 將數據壓成只有一個維度

建立一大小為128的隱藏層, 激活函數relu

最後輸出層大小設為 3, 激活函數softmax

```
# 3. 建立神經網路(平坦層、隱藏層、輸出層)

model.add(Flatten())
model.add(Dropout(rate=0.25))

model.add(Dense(128, activation='relu'))
model.add(Dropout(rate=0.25))

model.add(Dense(3, activation='softmax'))

print(model.summary())
```

訓練模型

模型參數設定

```
# 4. 訓練模型

model.compile(loss='categorical_crossentropy',optimizer='adam', metrics=
['accuracy'])
train_history=model.fit(x_train_normalize, y_train_OneHot,
                        validation_split=0.3,
                        epochs=200, batch_size=512, verbose=1)
```

模型參數整理

預設參數

正規化	有
shuffle	無
卷積層數	2層
kernel_size	(3,3)
池化層數	2層
pool_size	(2,2)
隱藏層數	1層
dense	128
dropout rate	0.25
batch_size	512
optimizer	RMSprop
epochs	12

修改後參數

正規化	有
shuffle	有
卷積層數	3層
kernel_size	(3,3)
池化層數	3層
pool_size	(2,2)
隱藏層數	1層
dense	128
dropout rate	0.25
batch_size	512
optimizer	adam
epochs	200

訓練集作圖與評估

```
# 5. 為訓練集作圖

import matplotlib.pyplot as plt
def show_train_history(train_history):
    plt.plot(train_history.history['accuracy'])
    plt.plot(train_history.history['val_accuracy'])
    plt.title('Train History')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend(['train', 'test'], loc='upper left')
    plt.show()

show_train_history(train_history)
```

```
# 6. 評估模型準確率

scores = model.evaluate(x_test_normalize,y_test_OneHot,verbose=0)
print(scores[:10])
```

預測結果

挑選出測試集中的

20張圖片來做預測

```
# 7-1. 查看預測結果

label_dict={0:"ikeashark",1:"realshark",2:"other"}

print(label_dict)

import matplotlib.pyplot as plt
def plot_images_labels_prediction(images,labels,prediction,idx,num=3):
    fig = plt.gcf()
    fig.set_size_inches(12,14)
    if num>25: num=25
    for i in range(0, num):
        ax=plt.subplot(5,5, 1+i)
        ax.imshow(images[idx],cmap='binary')

        title=str(i)+','+label_dict[labels[i]]
        if len(prediction)>0:
            title+='=>'+label_dict[prediction[i]]

        ax.set_title(title,fontsize=10)
        ax.set_xticks([]);ax.set_yticks([])
        idx+=1
    plt.show()

plot_images_labels_prediction(x_test_normalize,y_test,prediction,0,20)
```

預測機率

```
# 7-2. 查看預測機率

Predicted_Probability=model.predict(x_test_normalize)

def show_Predicted_Probability(y,prediction,x_img,Predicted_Probability,i):
    print('label:',label_dict[y[i]],
          'predict:',label_dict[prediction[i]])
    plt.figure(figsize=(2,2))
    plt.imshow(np.reshape(x_test[i],(32,32,3)))
    plt.show()
    for j in range(3):
        print(label_dict[j]+ ' Probability:%1.9f'%(Predicted_Probability[i][j]))

for i in range(0,4):

 show_Predicted_Probability(y_test,prediction,x_test_normalize,Predicted_Probabi
lity,i)
```

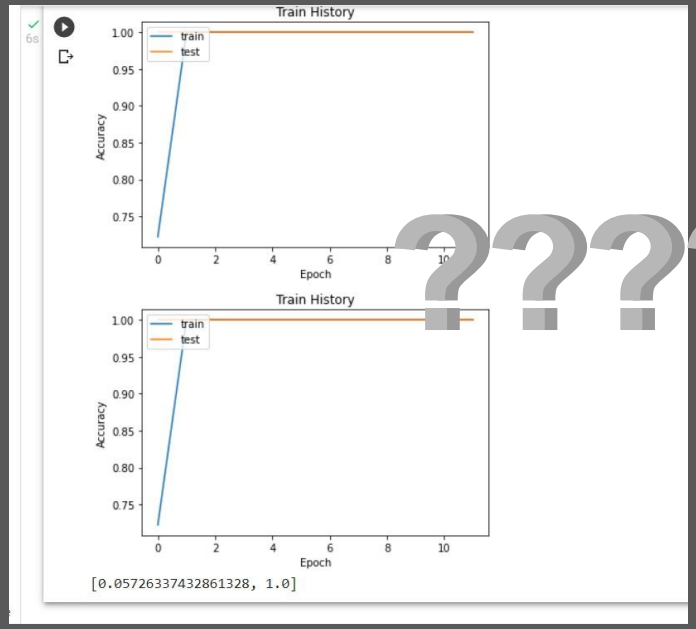
訓練過程

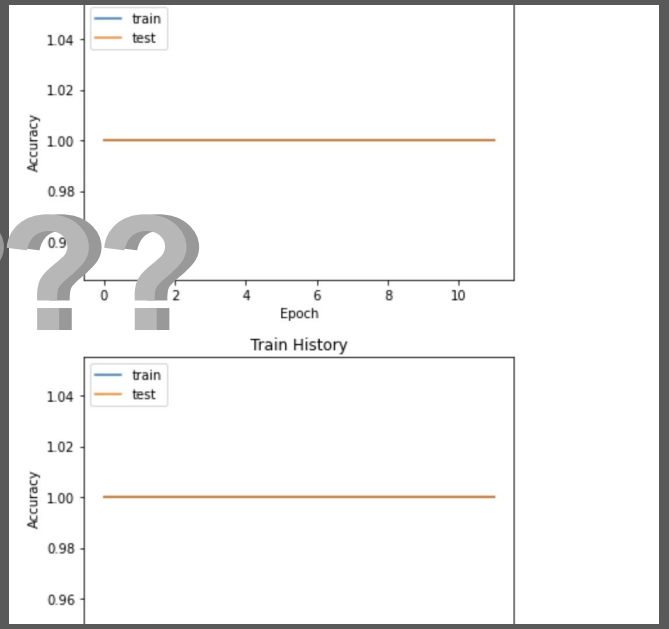
是漫長的路程.....

大概三小時吧.....

✓ 2 小時 51 分鐘 30 秒

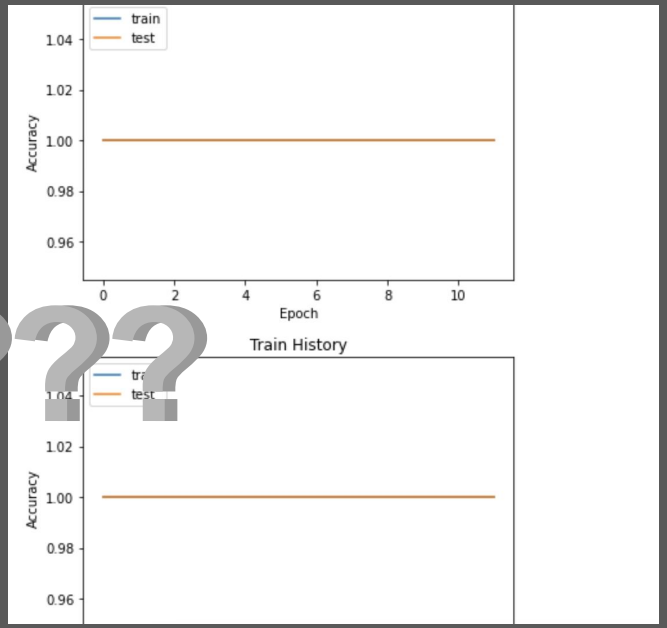
trouble滿滿的一開始.....





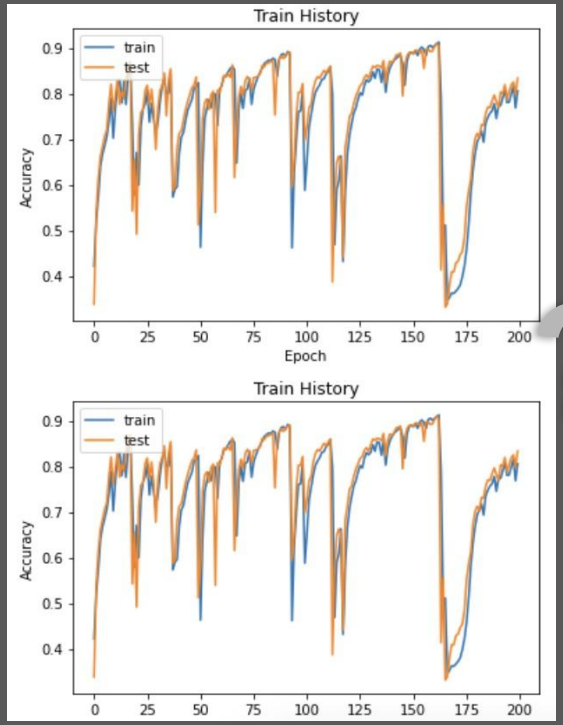
trouble滿滿的一開始.....

((3萬張要跑3小時，花轟

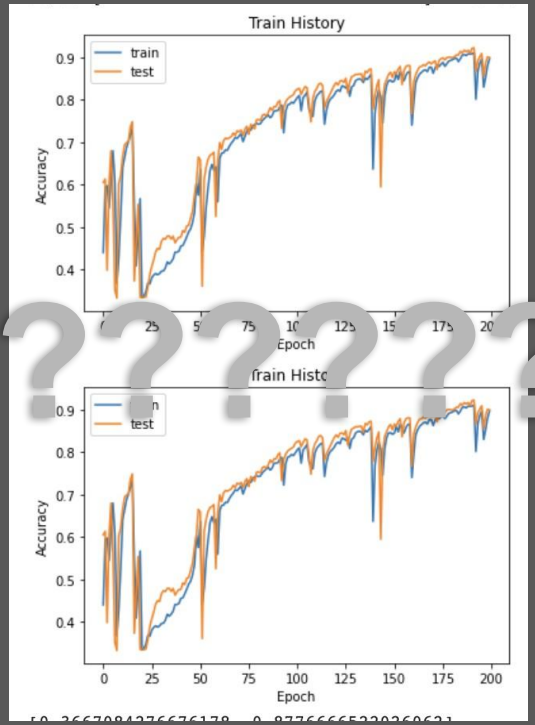


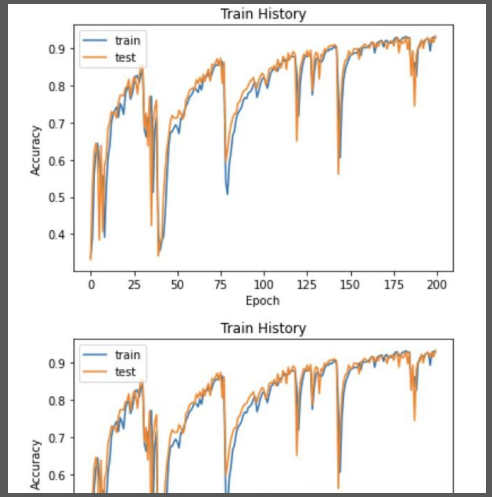
原來是我們只選了一個分類去訓練

訓練過程


Train History
train
test
Accuracy
Epoch
Train History
train
test
Accuracy

??????

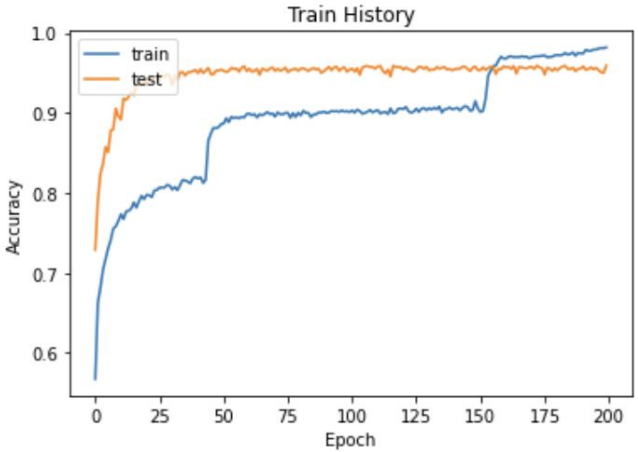

Train History
train
test
Accuracy
Epoch
Train Histo
test
Accuracy
Epoch

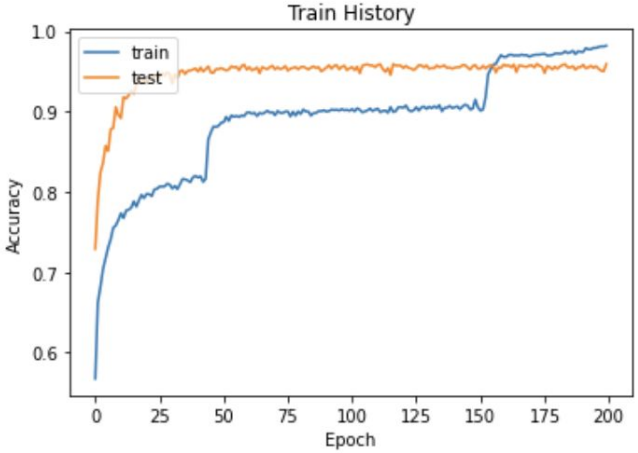

Train History
train
test
Accuracy
Epoch
Train History
train
test
Accuracy

成果展示

Accuracy: 0.9807 、Val_Accuracy: 0.9590

預測準確度:(19張 / 20張) * 100% = 95%





[0.21889756619930267, 0.9380000233650208]

訓練成果

目前最佳狀態

Accuracy: 0.9807
Val_Accuracy: 0.9590

```
s 58ms/step - loss: 0.1024 - accuracy: 0.9753 - val_loss: 0.1511 - val_accuracy: 0.9581
s 59ms/step - loss: 0.0956 - accuracy: 0.9775 - val_loss: 0.1546 - val_accuracy: 0.9537
s 58ms/step - loss: 0.0946 - accuracy: 0.9772 - val_loss: 0.1531 - val_accuracy: 0.9548
s 57ms/step - loss: 0.0952 - accuracy: 0.9771 - val_loss: 0.1463 - val_accuracy: 0.9569
s 59ms/step - loss: 0.0899 - accuracy: 0.9783 - val_loss: 0.1573 - val_accuracy: 0.9543
s 59ms/step - loss: 0.0875 - accuracy: 0.9793 - val_loss: 0.1532 - val_accuracy: 0.9564
s 59ms/step - loss: 0.0900 - accuracy: 0.9810 - val_loss: 0.1564 - val_accuracy: 0.9532
s 57ms/step - loss: 0.0901 - accuracy: 0.9816 - val_loss: 0.1692 - val_accuracy: 0.9507
s 58ms/step - loss: 0.0860 - accuracy: 0.9821 - val_loss: 0.1790 - val_accuracy: 0.9499
s 59ms/step - loss: 0.0868 - accuracy: 0.9807 - val_loss: 0.1434 - val_accuracy: 0.9590
```

```python
# 卷積層1與池化層1

model.add(Conv2D(filters=64,kernel_size=(3,3),
                 input_shape=(32,32,3),
                 activation='relu',
                 padding='same'))

model.add(Dropout(rate=0.25))

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

# 卷積層2與池化層2

model.add(Conv2D(filters=128, kernel_size=(3, 3),
                 activation='relu', padding='same'))

model.add(Dropout(0.25))

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


# 卷積層3與池化層3

model.add(Conv2D(filters=256, kernel_size=(3, 3),
                 activation='relu', padding='same'))

model.add(Dropout(0.25))

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

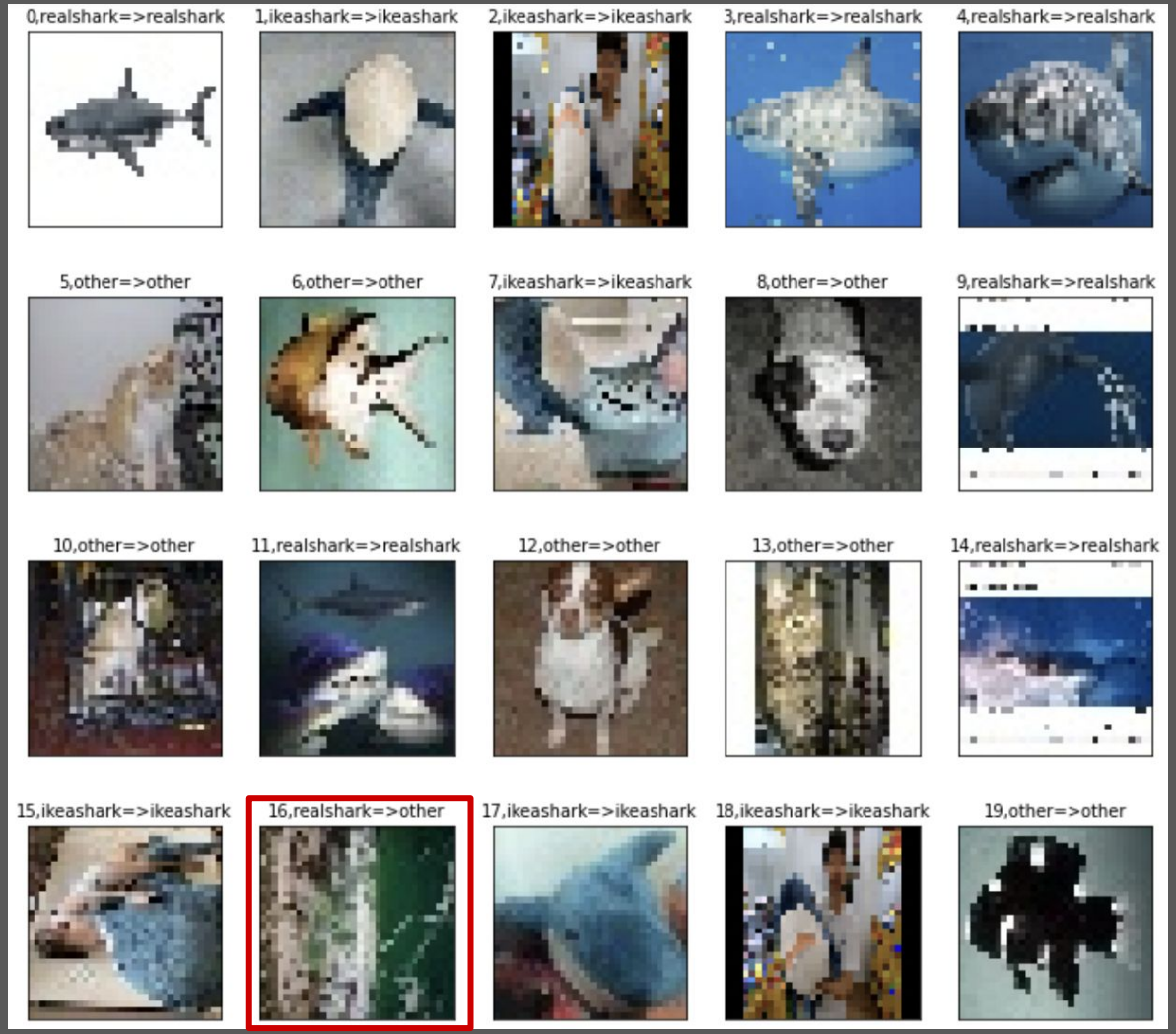
訓練成果

目前最佳狀態

Accuracy: 0.9807
Val_Accuracy: 0.9590

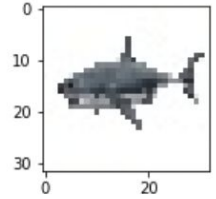
```python
# 4. 訓練模型

model.compile(loss='categorical_crossentropy',optimizer='adam', metrics=['accuracy'])
train_history=model.fit(x_train_normalize, y_train_OneHot,
                        validation_split=0.3,
                        epochs=200, batch_size=512, verbose=1)
```


0,realshark=>realshark
1,ikeashark=>ikeashark
2,ikeashark=>ikeashark
3,realshark=>realshark
4,realshark=>realshark
5,other=>other
6,other=>other
7,ikeashark=>ikeashark
8,other=>other
9,realshark=>realshark
10,other=>other
11,realshark=>realshark
12,other=>other
13,other=>other
14,realshark=>realshark
15,ikeashark=>ikeashark
16,realshark=>other
17,ikeashark=>ikeashark
18,ikeashark=>ikeashark
19,other=>other

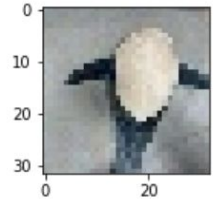
辨識成果

label: realshark predict: realshark



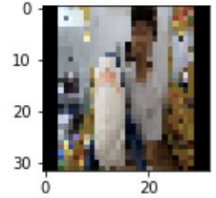
ikeashark Probability:0.000166480
realshark Probability:0.999581993
other Probability:0.000251550

label: ikeashark predict: ikeashark



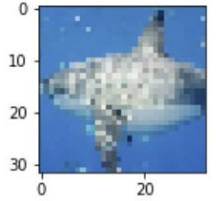
ikeashark Probability:0.993012130
realshark Probability:0.002192087
other Probability:0.004795820

label: ikeashark predict: ikeashark



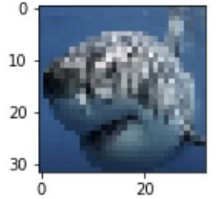
ikeashark Probability:0.993015766
realshark Probability:0.002190553
other Probability:0.004793626

label: realshark predict: realshark



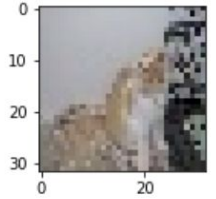
ikeashark Probability:0.000236941
realshark Probability:0.999412417
other Probability:0.000350581

label: realshark predict: realshark



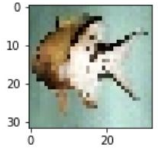
ikeashark Probability:0.000085007
realshark Probability:0.999788582
other Probability:0.000126450

label: other predict: other



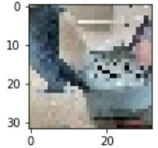
ikeashark Probability:0.002913872
realshark Probability:0.004524930
other Probability:0.992561162

label: other predict: other



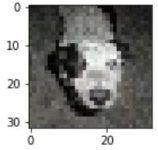
ikeashark Probability:0.000570936
realshark Probability:0.000442071
other Probability:0.998987019

label: ikeashark predict: ikeashark



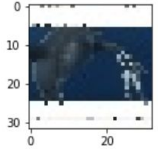
ikeashark Probability:0.993015528
realshark Probability:0.002190634
other Probability:0.004793833

label: other predict: other



ikeashark Probability:0.000536342
realshark Probability:0.000420203
other Probability:0.999043405

label: realshark predict: realshark



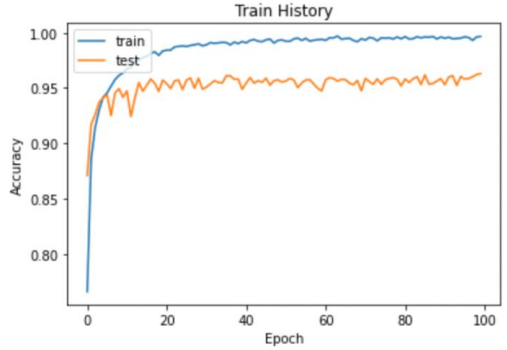
ikeashark Probability:0.000186445
realshark Probability:0.999565899
other Probability:0.000247598

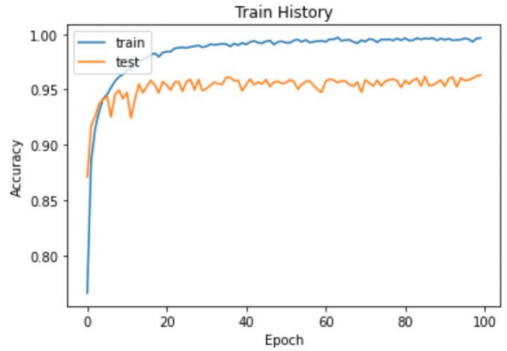
就在今日稍早，我們達到了準確度新高！！！

accuracy: 0.9962，val_accuracy: 0.9630

只可惜 overfitting了.... ಥ_ಥ

```
Epoch 100/100
169/169 [==============================] - 3s 17ms/step - loss: 0.0106 - accuracy: 0.9962 - val_loss: 0.1596 - val_accuracy: 0.9630
```






```python
# 3. 建立神經網路(平坦層、隱藏層、輸出層)

model.add(Flatten())
model.add(Dropout(rate=0.25))

model.add(Dense(512, activation='relu'))
model.add(Dropout(rate=0.25))

model.add(Dense(3, activation='softmax'))

print(model.summary())

# 4. 訓練模型

model.compile(loss='categorical_crossentropy',optimizer='adam', metrics=['accuracy'])
train_history=model.fit(x_train_normalize, y_train_OneHot,
                        validation_split=0.2,
                        epochs=100, batch_size=128, verbose=1)
```

修改部分

討論

提升準確率的可能??

- 可能不是最好的模型
- 圖片畫質可能可以從32x32提高
- 訓練集數量不足
- 圖片雜訊太多
- bounding box

bounding box

鯊鯊

- Thanks for listening -