# 時護(石虎)系統-石虎與虎斑貓影像辨識

類神經系統-期末報告
報告時間：2021/6/29

學生:郭郁鴻
指導教授:許超雲

# Outline 大綱


01 背景動機

02 研究目標

03 訓練資料

04 實驗過程

05 實驗成果

06 結論與未來展望

# 背景動機-1

- 生活在淺山地區的石虎(海拔1500公尺以下)也是人類活動頻繁的區域，人類棄養或放養犬貓到淺山生態系裡，也可能對石虎造成競爭、傷害和傳播疾病[1]
- 道路開入淺山地區，使得石虎有機會進入路面並遭到車輛撞擊，石虎路死事件層出不窮，如今數量可能已少於500隻 [2]
- 先前公路單位已在高速公路特定路段設置隔離網和生物通道[3]，新創公司開發出一款辨識石虎的人工智慧模型，偵測到石虎靠近道路時警告牠們遠離馬路，減少直接穿越馬路的機會[4]
- 然而道路以外的地方仍然會有石虎出沒，這些地方沒有科技的幫助，儼然成為保育死角。

[1] 台灣石虎的分布數量與生存威脅(2021年)。https://ahutw.info/status.html

[2] 台灣石虎的生存危機(2021年)。http://leopardcat.net/endangered.html

[3] 阿虎加油：石虎保育大使 - 常見問題(2021年)。https://ahutw.info/qa.html

[4] MELODY TU(2019年9月3日)。【台灣石虎靠 AI 來保護】遠離路殺！這套 AI 系統創下首個阻擋石虎過馬路的紀錄。https://buzzorange.com/techorange/2019/09/03/leopard-cat-ai-conservation/

# 背景動機-2

- 現階段守護石虎還是需要靠大眾的力量，在發現受傷、死亡石虎或是撿到石虎小寶寶進行通報並拍照，讓石虎在安全、有效率的救援下，提高獲救的機會[5]
- 然而石虎的體型和外貌和家貓相當類似，其中又以虎斑貓最為相似，因此常被誤認，甚至有人以為是一般貓咪而想帶回家飼養，因而有觸法之虞[6]，民眾對於石虎的認知以及石虎與貓的分辨能力不足。
- 因此如果能有一個**民間使用的石虎影像辨識系統**，讓民眾也能輕易快速分辨石虎與貓，提高通報的正確率與效率，將會是一個解決方法。

[5] 阿虎加油：石虎保育大使 - 事件通報(2021年)。https://ahutw.info/report.php

[6] 消失中的台灣石虎：辨別石虎：(2021年)。http://leopardcat.net/identify.html

# 研究目標

先前新創公司所開發的石虎影像辨識模型都是在特定場域針對石虎進行影像辨識，可應用的範圍較狹窄。

因此考量三個因素：

1. 石虎容易誤認為貓
2. 虎斑貓與石虎不易分辨
3. 不易將虎斑貓以外的品種誤認成石虎

本研究除了石虎資料集之外，加入虎斑貓資料集，進行石虎與虎斑貓影像辨識

# 為什麼選擇虎斑貓？

## 1. 石虎與虎斑貓難分辨





石虎

虎斑貓

圖片來源：
[7] 【插畫】石虎、貓咪，傻傻分不清楚？(2015年9月21日)。
https://www.thenewslens.com/article/25071
[8] 阿虎加油：石虎保育大使-石虎身家調查-衣(2021年)。
https://ahutw.info/facts-2.html

# 為什麼選擇虎斑貓？

## 2. 台灣虎斑貓數量多

台灣最常見的貓俗稱米克斯(MIX)[9]，米克斯貓是所有混血貓咪的統稱，人們依據貓咪常見的毛色將米克斯貓大致分成七大類：白貓、黑貓、橘貓、賓士貓、三花貓、虎斑貓、玳瑁貓[10]

白貓

黑貓

橘貓

賓士貓

三花貓

虎斑貓

玳瑁貓

[9] 認識貓咪-米克斯 MIX(2021年)。https://www.istoshare.com/intro/educationInfoItem/%E8%B2%93/1/

[10] 米克斯獨領風騷！米克斯貓花色、性格總整理(2021年2月27日)。https://www.hotpets.com.tw/mix-cats-introduction/

# 訓練資料-資料來源

1. 相關單位及新創公司並未提供石虎影像資料集開放使用

2. open data上也沒有台灣虎斑貓相關影像資料集

3. 資料來源使用網路爬蟲抓取google上面相關圖片

4. 過濾掉與石虎、虎斑貓無關的圖片


# Github：Scraping-Google-Images-using-Python專案[11]

使用[11]所提供的程式碼，透過網路爬蟲抓取Google上特定關鍵字的圖片

## Image-Scraping-from-Google-Images

Now scrape as many images as you want, from google images using Python, Chromdriver and Selenium

**Dependencies needed**

1. `Selenium` Install as `pip install selenium`
2. Python 3+ - Pyhton 3.6+ verion
3. Download `chromedriver.exe`
4. Place your `chromedriver.exe` and `google_image_scraping_script.py` file in the same folder
5. Open your terminal (Command Prompt for Windows) from that location and execute the script by typing `python googe_image_scraping_script.py`

**目標網址：**

https://www.google.com.tw/search?q={關鍵字}&tbm=isch
https://www.google.com.tw/search?q=虎斑貓&tbm=isch

**程式運作流程：**

程式裡寫好要抓取的關鍵字、抓取數量並儲存，打開Windows命令提示字元（cmd.exe）執行程式，開始抓取所有圖片的網址，再一次下載所有圖片

**改良程式：**

1. 在命令提示字元直接輸入關鍵字和抓取的圖片數量，省去每次從程式當中修改參數的麻煩
   python google_image_scraping_script_for_arg.py 石虎 1000
2. 排除開發者程式當中的BUG：實際圖片數量 < 所需圖片數量，導致陷入無窮迴圈



[11] GitHub - debadridtt/Scraping-Google-Images-using-Python(2021年)。https://github.com/debadridtt/Scraping-Google-Images-using-Python

# 訓練資料-資料處理

去除不相干的圖片

資料處理後

Google圖片爬蟲

| | 石虎 | 虎斑貓 |
|---|---|---|
| 應抓取數量 | 1000張 | 1000張 |
| 實際抓取數量 | 990張 | 996張 |

手動資料處理

| | 石虎 | 虎斑貓 |
|---|---|---|
| 原有圖片數量 | 990張 | 996張 |
| 資料處理後<br>圖片數量 | 339張 | 631張 |

# 實驗過程-1

**模型訓練測試資料切分：**

cat_train：504張
cat_test：127張
shihu_train：271張
shihu_test：68張

**卷積神經網路 CNN：**

使用GitHub上影像辨識專案的程式碼[12]
兩層各128個神經元
輸出：sigmoid
Loss function：binary_crossentropy
Target_size：64*64
Batch_size：32
Epochs：15

```
# Step 4 - Full connection
classifier.add(Dense(units = 128, activation = 'relu'))
classifier.add(Dense(units = 128, activation = 'relu'))
classifier.add(Dense(units = 1, activation = 'sigmoid'))

# Compiling the CNN
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

# Part 2 - Fitting the CNN to the images
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255)

test_datagen = ImageDataGenerator(rescale = 1./255)

trainset = train_datagen.flow_from_directory('body_dataset/trainset',
                                             target_size = (64, 64),
                                             batch_size = 32,
                                             class_mode = 'binary')

testset = test_datagen.flow_from_directory('body_dataset/testset',
                                           target_size = (64, 64),
                                           batch_size = 32,
                                           class_mode = 'binary')

classifier.fit_generator(trainset,
                         steps_per_epoch= 40, validation_steps= 20,
                         epochs = 15,
                         validation_data = testset
                         )
```

[12] GitHub-ywchiu/digiplus_ai(2019年7月23日)。https://github.com/ywchiu/digiplus_ai

# 實驗成果-1

圖片的辨識率落在80.83%~90.78%之間

```
TensorFlow binary was not compiled to use: AVX2
40/40 [==============================] - 9s 232ms/step - loss: 0.6066 - acc: 0.6889 - val_loss: 0.5944 - val_acc: 0.6667
Epoch 2/15
40/40 [==============================] - 4s 110ms/step - loss: 0.4941 - acc: 0.7776 - val_loss: 0.4411 - val_acc: 0.8083
Epoch 3/15
40/40 [==============================] - 5s 116ms/step - loss: 0.2962 - acc: 0.8800 - val_loss: 0.4100 - val_acc: 0.8318
Epoch 4/15
40/40 [==============================] - 4s 112ms/step - loss: 0.2087 - acc: 0.9347 - val_loss: 0.3210 - val_acc: 0.8662
Epoch 5/15
40/40 [==============================] - 4s 112ms/step - loss: 0.0911 - acc: 0.9851 - val_loss: 0.3216 - val_acc: 0.8825
Epoch 6/15
40/40 [==============================] - 4s 108ms/step - loss: 0.0326 - acc: 0.9945 - val_loss: 0.3195 - val_acc: 0.8843
Epoch 7/15
40/40 [==============================] - 4s 111ms/step - loss: 0.0151 - acc: 0.9984 - val_loss: 0.3619 - val_acc: 0.9078
Epoch 8/15
40/40 [==============================] - 5s 116ms/step - loss: 0.0142 - acc: 0.9977 - val_loss: 0.3307 - val_acc: 0.9021
Epoch 9/15
40/40 [==============================] - 5s 117ms/step - loss: 0.2181 - acc: 0.9402 - val_loss: 1.0760 - val_acc: 0.4304
Epoch 10/15
40/40 [==============================] - 5s 117ms/step - loss: 0.2137 - acc: 0.9437 - val_loss: 0.3681 - val_acc: 0.8571
Epoch 11/15
40/40 [==============================] - 5s 116ms/step - loss: 0.0219 - acc: 0.9984 - val_loss: 0.3708 - val_acc: 0.8861
Epoch 12/15
40/40 [==============================] - 5s 116ms/step - loss: 0.0088 - acc: 0.9984 - val_loss: 0.4273 - val_acc: 0.8933
Epoch 13/15
40/40 [==============================] - 5s 115ms/step - loss: 0.0075 - acc: 0.9977 - val_loss: 0.3622 - val_acc: 0.8788
Epoch 14/15
40/40 [==============================] - 5s 121ms/step - loss: 0.0045 - acc: 0.9984 - val_loss: 0.4017 - val_acc: 0.8987
Epoch 15/15
40/40 [==============================] - 5s 116ms/step - loss: 0.0141 - acc: 0.9957 - val_loss: 0.4470 - val_acc: 0.8883
```

# 臉部辨識-1

為了更精進模型準確率，增加臉部的辨識模型，額外識別出石虎與貓的臉，觀看效果


優點：

1. 不容易受背景影響，且強調出頭部紋路的特徵
2. 一張照片不會有兩隻貓

# OpenCV-Haar cascade分類器-1

使用OpenCV[13]所提供的Haar cascade分類器[14]進行臉部偵測

```
img = Image.open(filename)
imgary = cv.imread(filename)
face_cascade = cv.CascadeClassifier('haarcascade_frontalcatface_extended.xml')
faces = face_cascade.detectMultiScale(imgary, scaleFactor=1.3, minNeighbors=1)
```

導入指定的Haar cascade分類器

ScaleFactor：濾波器每次搜尋時減少的比例

minNeighbers：每個目標至少檢測到幾次以上，才可被認定是真數據

master ▾ opencv / data / haarcascades /

alalek fix files permissions

..

- haarcascade_eye.xml
- haarcascade_eye_tree_eyeglasses.xml
- haarcascade_frontalcatface.xml
- haarcascade_frontalcatface_extended.xml
- haarcascade_frontalface_alt.xml
- haarcascade_frontalface_alt2.xml
- haarcascade_frontalface_alt_tree.xml
- haarcascade_frontalface_default.xml
- haarcascade_fullbody.xml


[13] OpenCV - 維基百科，自由的百科全書(2021年)。https://zh.wikipedia.org/wiki/OpenCV

[14] opencv/data/haarcascades at master · opencv/opencv · GitHub(2020年4月13日)。https://github.com/opencv/opencv/tree/master/data/haarcascades

# OpenCV-Haar cascade分類器-2

使用Haar cascade分類器前後圖片數量的差異

| | 石虎 | 虎斑貓 |
|---|---|---|
| 原有圖片數量 | 339張 | 631張 |
| 可識別出臉部的圖片數量 scaleFactor=1.3, minNeighbors=1 | 54張 | 205張 |
| 可識別出臉部的圖片數量 scaleFactor=1.3, minNeighbors=3 | 15張 | 120張 |
| 可識別出臉部的圖片數量 scaleFactor=1.3, minNeighbors=5 | 2張 | 79張 |

從分割出石虎與貓臉的資料量差異，可看出Haar cascade分類器只適合分割貓的臉不適合分割石虎的臉，該分類器對於辨識出石虎臉的能力較差

# 實驗結果-2

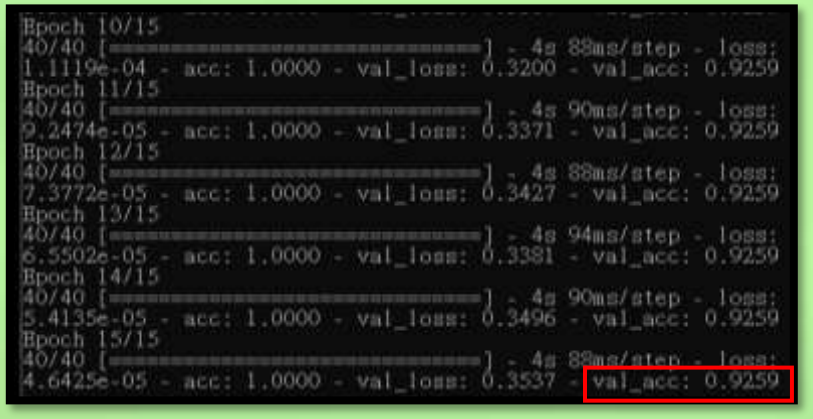
## 模型訓練測試資料切分：

scaleFactor=1.3,
minNeighbors=1

| | 石虎 | 虎斑貓 |
|---|---|---|
| 訓練資料 | 43張 | 164張 |
| 測試資料 | 11張 | 41張 |


scaleFactor=1.3,
minNeighbors=3

| | 石虎 | 虎斑貓 |
|---|---|---|
| 訓練資料 | 12張 | 96張 |
| 測試資料 | 3張 | 24張 |




scaleFactor=1.3,
minNeighbors=5

| | 石虎 | 虎斑貓 |
|---|---|---|
| 訓練資料 | 1張 | 63張 |
| 測試資料 | 1張 | 16張 |


臉部的辨識率落在88.46%~94.12%之間

分割出石虎與貓臉的資料量太少，模型的準確率較無意義，不足以呈現影像辨識效果

# 臉部辨識-2

自行手動分割臉部特徵，擴充訓練模型使用的資料量

**自行手動分割前後圖片數量的差異**

| | 石虎 | 虎斑貓 |
|---|---|---|
| 原有圖片數量 | 339張 | 631張 |
| Haar cascade分類器分割圖片數量 | 54張 | 205張 |
| 自行手動分割後圖片數量 | 242張 | 504張 |

- 有些圖片只有拍到石虎或虎斑貓側身、背部，而沒有臉部的圖像，故捨棄之
- 使用臉部辨識分類器與自行分割，得出圖片的數量差異大

# 實驗成果-3

臉部的辨識率落在86%~92%之間

## 模型訓練測試資料切分

| | 石虎 | 虎斑貓 |
|---|---|---|
| 訓練資料 | 403張 | 193張 |
| 測試資料 | 101張 | 49張 |

```
40/40 [==============================] - 5s 121ms/step - loss: 0.6545 - acc: 0.6696 - val_loss: 0.5010 - val_acc: 0.7667
Epoch 2/15
40/40 [==============================] - 4s 104ms/step - loss: 0.3305 - acc: 0.8641 - val_loss: 0.3217 - val_acc: 0.8600
Epoch 3/15
40/40 [==============================] - 4s 101ms/step - loss: 0.1587 - acc: 0.9519 - val_loss: 0.2703 - val_acc: 0.8933
Epoch 4/15
40/40 [==============================] - 4s 101ms/step - loss: 0.0488 - acc: 0.9961 - val_loss: 0.2686 - val_acc: 0.8733
Epoch 5/15
40/40 [==============================] - 4s 108ms/step - loss: 0.0191 - acc: 1.0000 - val_loss: 0.2260 - val_acc: 0.9133
Epoch 6/15
40/40 [==============================] - 4s 106ms/step - loss: 0.0089 - acc: 1.0000 - val_loss: 0.2211 - val_acc: 0.9067
Epoch 7/15
40/40 [==============================] - 5s 124ms/step - loss: 0.0052 - acc: 1.0000 - val_loss: 0.2374 - val_acc: 0.9133
Epoch 8/15
40/40 [==============================] - 4s 104ms/step - loss: 0.0032 - acc: 1.0000 - val_loss: 0.2250 - val_acc: 0.9133
Epoch 9/15
40/40 [==============================] - 4s 102ms/step - loss: 0.0024 - acc: 1.0000 - val_loss: 0.2296 - val_acc: 0.9200
Epoch 10/15
40/40 [==============================] - 4s 103ms/step - loss: 0.0017 - acc: 1.0000 - val_loss: 0.2293 - val_acc: 0.9133
Epoch 11/15
40/40 [==============================] - 4s 104ms/step - loss: 0.0013 - acc: 1.0000 - val_loss: 0.2306 - val_acc: 0.9200
Epoch 12/15
40/40 [==============================] - 4s 107ms/step - loss: 0.0011 - acc: 1.0000 - val_loss: 0.2449 - val_acc: 0.8933
Epoch 13/15
40/40 [==============================] - 4s 105ms/step - loss: 8.9286e-04 - acc: 1.0000 - val_loss: 0.2348 - val_acc: 0.9200
Epoch 14/15
40/40 [==============================] - 4s 104ms/step - loss: 7.1773e-04 - acc: 1.0000 - val_loss: 0.2351 - val_acc: 0.9133
Epoch 15/15
40/40 [==============================] - 4s 103ms/step - loss: 6.2042e-04 - acc: 1.0000 - val_loss: 0.2430 - val_acc: 0.9133
```

# 實驗成果-4

## 模型調整參數前後兩種模型準確率比較

| | 全身辨識率 | 臉部辨識率(自行切割) |
|---|---|---|
| (1)原始參數 | 80.83%~90.78% | 86%~92% |
| (2)第二層神經元數量改為256個(原為128個) | 75.41%~90.24% | 84.67%~93.33% |
| (3)修改參數batch_size= 64 (原為32)<br>steps_per_epoch= 60 (原為40)<br>validation_steps= 30 (原為20) | 82.96%~90.71% | 88.67%~92% |
| 結合(2)、(3)修改參數方式 | 85.41%~88.75% | 90%~92% |

# 結論與未來展望

結論：

1.在沒有現成的石虎與虎斑貓資料集的情況下，使用Google抓取的圖片製作模型，對於石虎與虎斑貓的辨識準確率都超過八成

2.透過Haar cascad分類器分割臉部圖片，從得到的圖片數量差異發現該分類器對於虎斑貓、石虎臉部辨識效果有限

3. 石虎與虎斑貓在臉部差異上為眼周圍白紋圈、白條紋，從準確率看出此臉部辨識模型更能夠分類兩者(石虎、虎斑貓)的不同

4. 從全身模型改為使用臉部模型後，準確率略微提升，未來準確率是否還有成長空間？需要更多影像資料加入模型訓練才能得知

未來展望：

1.能從其他來源擴充更多影像資料集改善模型準確率

2.開發成手機APP應用，掃描就知道眼前的是石虎還是虎斑貓，回傳影像擴充影像資料集