OAC-110-015（研究報告）

AI救你衣命

正式報告

海洋委員會補助研究

中華民國 110年 9月

OAC-110-015（研究報告）

AI救你衣命

正式報告

學校：國立臺北大學

指導教授：吳信龍

學生：曹鶴騰

研究期程：中華民國110年4月至110年11月

研究經費：新臺幣六萬元

海洋委員會補助研究

中華民國 110年 9月

目次

圖次 4
表次 5
摘要 6
第一章 前言 7
第一節 研究緣起 7
第二節 問題背景 7
第三節 現況分析 7
第四節 研究目的 7
第五節 研究重點 8
第六節 預期目標 8
第二章 研究方法及過程 11
第一節 可行方案的選擇 11
第二節 標注方法 13
第三節 資料收集 14
第四節 模型訓練 15
第五節 訓練集以及辨識結果 19
第六節 驗證訓練結果 20
第七節 系統功能 22
第三章 結果與討論 28
第一節 準確度的提升 28
第二節 系統整合 28
第三節 GUI的增加 28
第四章 結論 30
參考資料 31

圖次

圖一 系統架構圖之一.....8
圖二 系統架構圖之二.....9
圖三 系統架構圖之三.....9
圖四 系統架構圖之四.....10
圖五 client-server架構示意圖.....11
圖六 本地端運算示意圖.....12
圖七 系統欲達成任務.....13
圖八 LabelImg.....13
圖九　標示後匯出的xml檔.....14
圖十　使用PixelLib函式庫.....15
圖十一 合成圖像.....15
圖十二 使用PIL和OpenCV套件.....16
圖十三 翻轉部分程式碼.....16
圖十四 翻轉後與原圖比較.....17
圖十五 使用OpenCV調整HSV色彩空間.....17
圖十六 調整HSV參數後與原圖比較.....18
圖十七 loss曲線圖.....18
圖十八 訓練集原圖以及模型辨識結果.....20
圖十九 測試集原圖及模型辨識結果.....22
圖二十 攝影機範例.....23
圖二十一 Jetson TX2上的警告程式.....23
圖二十二 client端的程式碼.....24
圖二十三 server的程式碼.....25
圖二十四 server端顯示訊息的GUI界面.....25
圖二十五 server端播放通知音效的程式.....26
圖二十六 儲存資料程式碼.....26
圖二十七 用Excel開啟csv檔.....27

表次

表一 三種模型比較 12
表二 最終模型準確率 20
表三 期中報告模型準確度 28
表四 最終模型準確率 28

摘要

關鍵詞:救生衣偵測、深度學習、YOLOv4、邊緣計算

救生協會曾調查，釣客落水若穿著救生衣，九成以上可獲救，不穿救生衣者致死率七成；他們也曾向海巡調申請統計資料，十多年之間落海死亡六千多人。如今政府只能透過不斷提高釣客未穿救生衣的罰鍰及加強人員的巡邏勸導，來約束釣客們穿上救生衣，但顯然效果還有待加強，因此如何加強防範此項海釣安全問題，便是我們這次的研究動機，而我們所想到的應對方法，就是做出第一個能自動偵測釣客是否有穿著救生衣的機器，我們的做法，主要是透過深度學習的方法去達成此項目標，而為了達到實用的目的，我們須透過YOLOv4來偵測在進行海釣的人們亦或是搭船的乘客們是否有正確的穿上救生衣，如此一來不僅能降低溺水的死亡率，也能減輕消防與海巡人員的負擔與減少救援產生的社會成本。

「穿著救生衣的重要性」比我們想像中還更需要得到重視。經查閱過去的所有研究成果報告與文獻探討，並沒有使用深度學習來提升海上救難效率的計劃與產品，因此本次研究計畫之偵測是否有穿戴救生衣系統可說是首次提出之創新概念，就我們目前所蒐集的文獻可知，沒有相關的研究報告是關於此系統概念的提出與分析可行性。本研究計畫可視為利用科技方法提升海域安全之先趨研究。

除了警告遊客以外，在系統多次勸阻遊客無效後，系統會利用網路socket傳送當地地址給管理機構，請管理機構的人員前來處理。最後，我們還加入了統計釣魚環境裡，遊客有無穿著救生衣的資料數，以方便相關當局評估當地狀況及調整政策之所需。

第一章 前言

第一節 研究緣起

由於看到遊客海釣落水死亡的新聞([4])，因為這種事件時有耳聞，所以想要找出能夠解決這項問題的方法，我們開始嘗試透過人工智慧中深度學習的方式，開發一套針對加強海域安全的偵測系統，來降低社會成本，同時希望製造拋磚引玉的效果，讓更多資深的研究人員投入這領域的研究。

第二節 問題背景

統計資料顯示，遊客發生溺水案件，其原因主要為週休假期間，民眾從事水上活動如釣魚、潛水等日益增多，其中還有因海釣而溺水死亡的事件。

據消防局108年溺水事件報告顯示(第96頁，[3])，本市各地水域之危險潛勢特性均不相同，且鑒於釣客穿著救生衣可以提升落海獲救機會。不過仍發生不少海釣客因為穿著救生衣令人不適，抱著僥倖的心態不穿而有屢勸不聽的情形。近年來，因為政府機關設立相關規定，這種情形確實有因罰金的提升而逐年下降的趨勢，但是要達成此結果，政府必須派遣不少海巡人員適時地在海邊巡邏取締。

第三節 現況分析

現今為了將海域安全的概念傳播到民眾的想法，僅以透過媒體來宣導、教育，並用罰款輔助方法以達目標成效，儘管如此，至今仍發生許多海浪捲走遊客的憾事發生，若能實現一套系統可以全時段且全區域的去偵測遊客是否有穿救生衣，那不僅可以達到預防之效果，且可以降低大量人力成本的浪費。

查閱過去的研究成果報告與文獻探討，並沒有類似的系統來提昇海上救生救難的效率，因此本次研究計畫之偵測是否有穿戴救生衣系統可以說是首次提出之創新概念，就我們目前所知，沒有相關的研究報告是關於此系統概念的提出與分析可行性。本研究計畫可視為利用科技方法提升海域安全之先趨研究，以及減少政府機關派遣人員所需要的經費以及時間。

第四節 研究目的

本次研究人員針對系統可行性加以探討，試圖研發一套AI系統，達成如下目標：

1.降低人員開銷

2.即時知道地方遊客穿著救生衣之情形

3.針對沒有穿著救生衣之遊客進行警告

4.每隔一段時間紀錄當地遊客有無穿著救生衣的數量，以供相關機構做出政策調整或是用於大數據的研究

5.在多次警告無果後，傳送通知給當地主管機關，請他們派遣人員到遊客所在地進行處理

第五節 研究重點

1.解決方案是利用深度學習去達成

2.減少工作人員的派遣次數，省下人力成本

3.使用從來沒有人想到的創新概念，利用深度學習提昇海上救生救難的效率

4.打造這項研究領域的基石

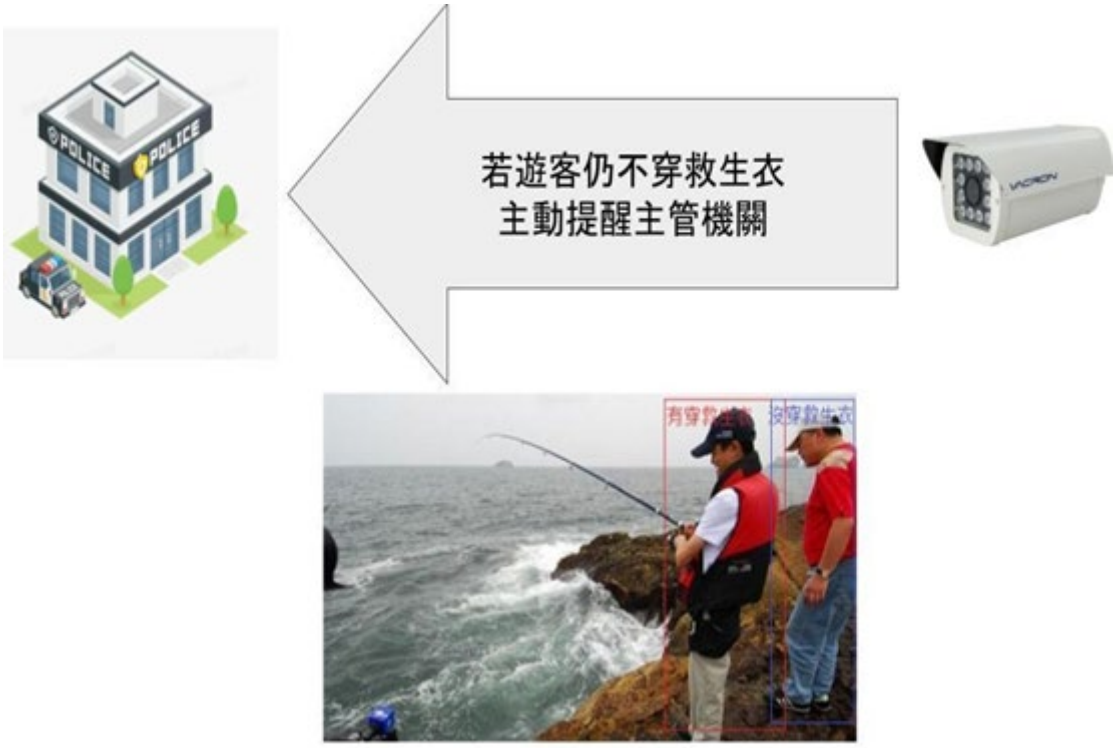
第六節 預期目標

我們的目標是打造一個AI偵測系統，利用監控設備監視遊客狀況(如圖一)，判斷出遊客有無穿著救生衣(如圖二)，並在發現沒有時進行提醒的動作，直到所有遊客都穿上救生衣(如圖三)。而在數次警告無效後，能透過網路傳送遊客所在地址的訊息通知管理機構，請他們派遣人員到當地處理(如圖四)，而且我們想要使接收訊息的程式是就算沒接觸過程式學習的人也能輕鬆使用以及看懂上面訊息。除此之外，AI系統每隔一段時間會紀錄當地遊客的情況，可透過的這些記錄做政策的調整或大數據統計的應用。

圖一 系統架構圖之一

圖二 系統架構圖之二

圖三 系統架構圖之三

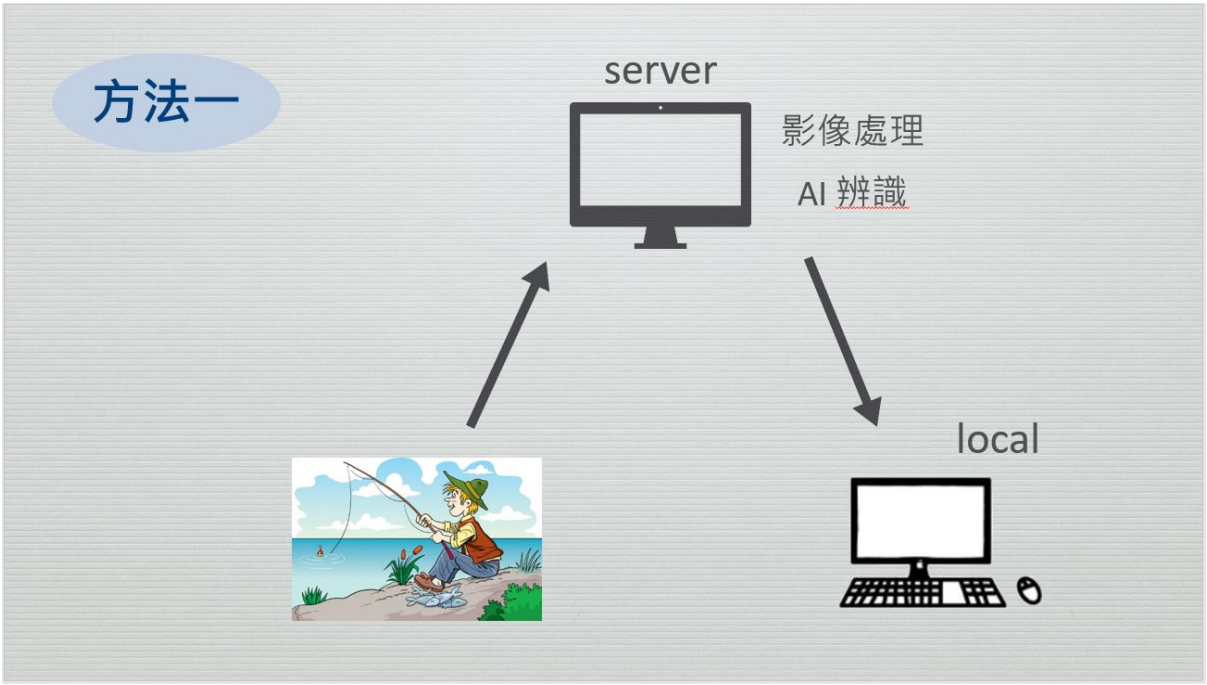


圖四 系統架構圖之四

第二章 研究方法及過程

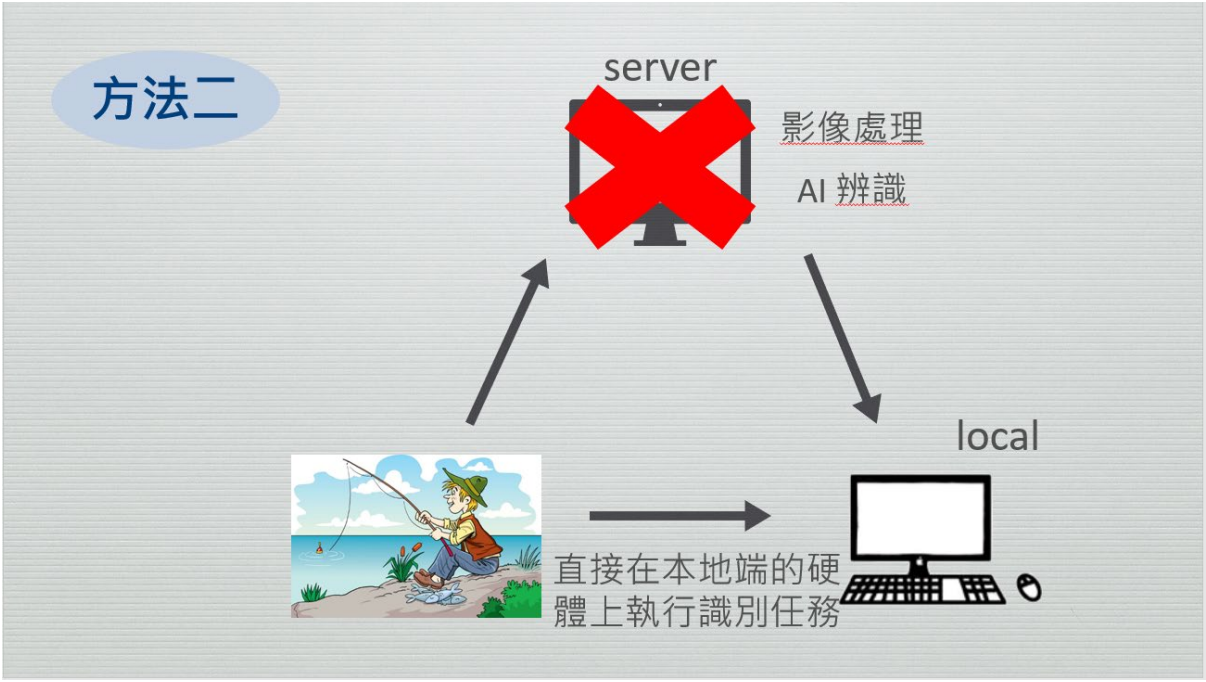
第一節 可行方案的選擇

在系統建置方面，我們有兩種選擇。第一種是利用client和server的方式，於遊客所在的client端拍下畫面，然後傳到server，讓server運作模型辨識遊客有無穿救生衣，再將結果回傳client(如圖五)。這種方法有一個好處，server端的電腦可以選擇使用性能很好的硬體，提高運算的速度，但是網路的問題就是我們會受限的瓶頸，如果網路狀況不順，client端會耗費許多時間在傳送影像，最糟糕的情況是client端甚至有可能沒辦法把影像傳回server端，又或者是server端無法將辨識結果傳回client端，導致無法辨識。



圖五 client-server架構示意圖

第二種方法是從拍攝影像，到透過模型計算出結果都是在同一台硬體上執行，並不需要將影像傳回server(如圖六)。這種方法在模型辨識上就不會有網路方面的問題，除此之外，因為不需要將影像回傳到server，遊客隱私不會有被外洩的問題。但是裝在戶外的電子設備並不能像第一種方法那樣，選用性能很好的電腦，只能是體積小、性能較差的邊緣計算嵌入式系統。如此一來也能降低架設的成本。



圖六 本地端運算示意圖

為了不讓模型辨識因為網路問題而成為運算速度的瓶頸甚至是無法進行辨識，因此我們選擇第二種方法，但是使用第二種方法的話，必須選用在準確率可以達到保證的情況下，運作速度快的模型。我們有Faster R-CNN、YOLOv1、YOLOv4可以選擇，經過比較，我們選擇最適合在邊緣計算硬體的運算效能下運行YOLOv4(如表一)。

表一 三種模型比較

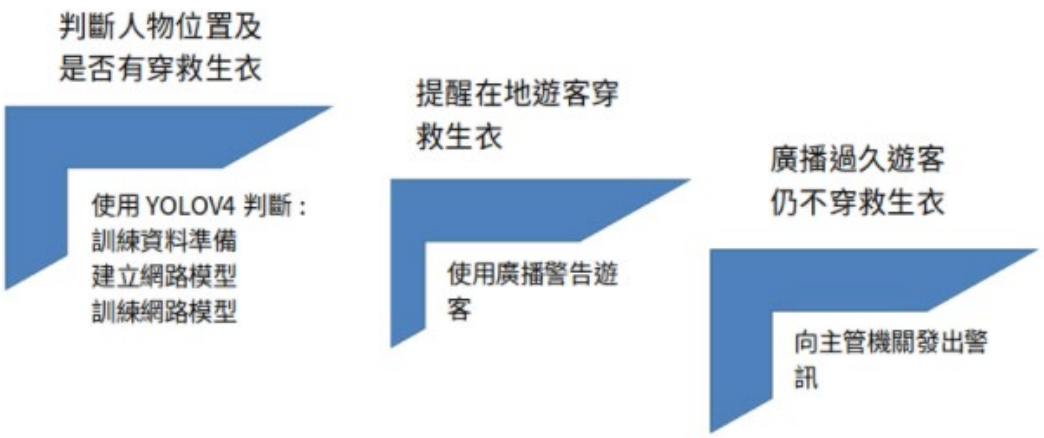
Model	Faster R-CNN([6])	YOLOv1([9])	YOLOv4([10])
Advantage	準確率最高	準確率最低	準確率中等
Disadvantage	運行速度最慢	運行速度中等	運行速度最快

最後總結系統的主要任務(如圖七)：

1.偵測遊客位置並辨識出有無穿著救生衣

2.當發現沒有時，系統會發出警告

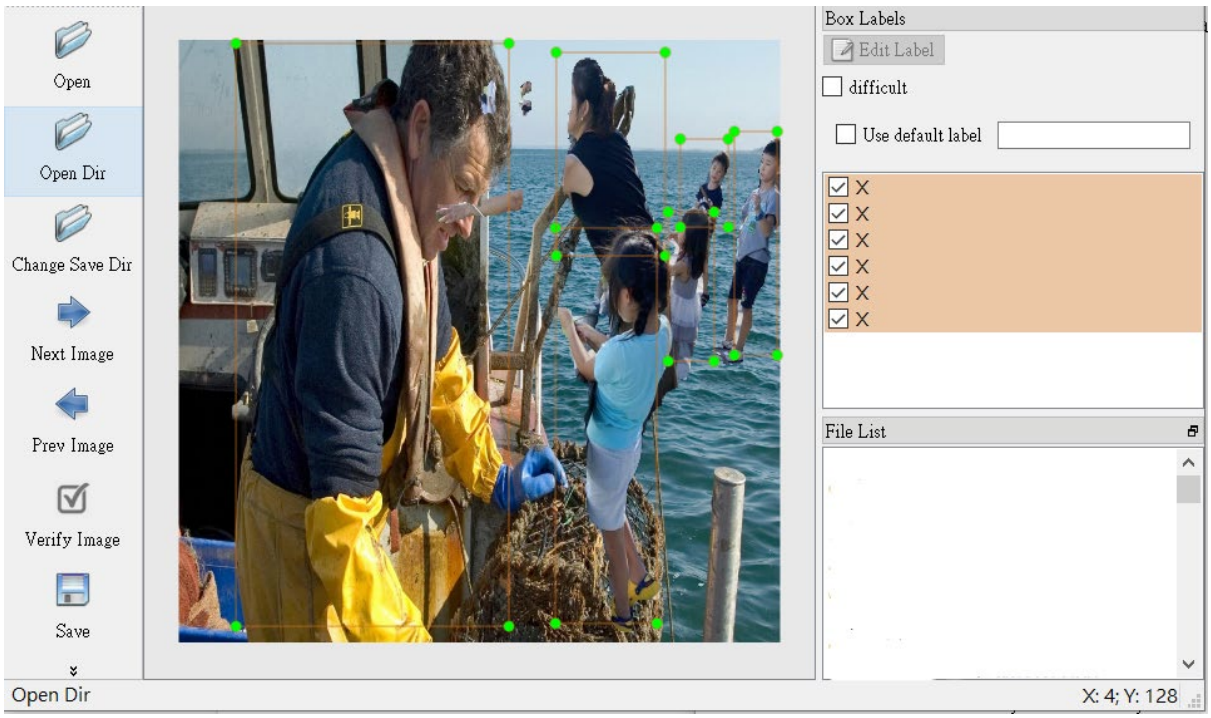
3.如果過一段時間後遊客還是沒有穿上救生衣，將會向主管機關發出通知，請他們派遣工作人員前來處理



圖七 系統欲達成任務

第二節 標注方法

在訓練集標注方面，我們使用labelImg來作為標注工具([7])。在標注目標位置時及名稱時，labelImg會將這些資訊儲存在xml檔裡，方便日後訓練模型時讀取。以圖八為例，當中綠色方框為人工畫出的標記框，而右半側則顯示該方框被標記為何種類型資料，此圖片中有6個沒有穿救生衣的人員，所以有6個類型為X的標記框，然後將這些資訊轉換為如圖九一樣的xml檔，供訓練模型讀取。



圖八 LabelImg

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明

```
<annotation>
        <folder>syn2-3</folder>
        <filename>syn2242.jpg</filename>
        <path>C:\曹\專題\syn2-3\syn2242.jpg</path>
        <source>
                <database>Unknown</database>
        </source>
        <size>
                <width>800</width>
                <height>800</height>
                <depth>3</depth>
        </size>
        <segmented>0</segmented>
        <object>
                <name>X</name>
                <pose>Unspecified</pose>
                <truncated>0</truncated>
                <difficult>0</difficult>
                <bndbox>
                        <xmin>53</xmin>
                        <ymin>57</ymin>
                        <xmax>219</xmax>
                        <ymax>733</ymax>
                </bndbox>
        </object>
        <object>
                <name>X</name>
                <pose>Unspecified</pose>
                <truncated>0</truncated>
                <difficult>0</difficult>
                <bndbox>
                        <xmin>313</xmin>
                        <ymin>63</ymin>
                        <xmax>434</xmax>
                        <ymax>617</ymax>
                </bndbox>
        </object>
        <object>
                <name>X</name>
                <pose>Unspecified</pose>
                <truncated>0</truncated>
                <difficult>0</difficult>
                <bndbox>
                        <xmin>491</xmin>
```

圖九　標示後匯出的xml檔

第三節 資料收集

資料集由三種來源組成:

1. 自行拍攝：約150張

2. 網路蒐集：約650張

3. 合成圖片(擇優挑選)：約1300張

其中合成圖片使用PixelLib函式庫([8])，利用Deeplabv3+([8])模型偵測出人的樣子(不論有無穿上救生衣)，再將他們從原本的背景中分割出來，並且貼在其他的照片之中，其中，我們實作後發現，將沒穿救生衣的人分割出來的效果在大部分的例子中表現得較好(模型在分割有穿救生衣的人時，經常沒辦法完好地切割出救生衣)，使用了這種方法後，我們就可以再次從不同的方法中，增加每一張照片在不同標籤數量上的平衡也增加資料集的豐富度(後續會講到用別的方法來增加照片的數量)。圖十是製造合成圖片的片段程式碼，可以看出我們這邊是將人體物件從背景中擷取出來，圖十一為程式製作出來的合成圖，背後原理是將白衣女子從原照片擷取出來貼到其他照片上，增加資料集的豐富度。

```
import pixellib
from pixellib.tune_bg import alter_bg

change_bg = alter_bg(model_type = "pb")
change_bg.load_pascalvoc_model("xception_pascalvoc.pb")
change_bg.change_bg_img(f_image_path = img_name,
                        b_image_path = bg_name,
                        output_image_name=output,
                        detect = "person")
```

圖十　使用PixelLib函式庫

圖十一 合成圖像

第四節 模型訓練

訓練環境：

作業系統：Ubuntu 18.04.2 LTS

GPU：Geforce RTX 2080 ti

CUDA version：10.2

在訓練模型時，為了方便日後擴充功能，我們選用更加靈活的程式語言來進行系統的架構，因此選擇使用Python版本的YOLOv4，深度學習框架選用Pytorch。

為了防止因為資料集數量較少，造成過擬合(over-fitting)的現象發生，我們在將資料集送入模型前會進行資料擴增(data augment)，常用的資料擴增方法有旋轉、調整大小比例或尺寸、改變色溫、翻轉等等處理，而選用甚麼樣的方法來進行資料擴增也是相當重要的，若選用的方法不好，有可能會導致訓練結果不理想，因此我們選擇把隨機地將幾張照片水平翻轉(如圖十三及圖十四)、在HSV色彩空間下來調整各項參數的影像處理方式來做資料擴增(如圖十五及圖十六)，以增加訓練集的數量。可以從圖十二看出這兩個部份我們分別加入PIL和OpenCV這兩個Python常見的影像處理函式庫，並在後續的程式碼片段中使用。

```
import cv2
import numpy as np
from PIL import Image
```

圖十二 使用PIL和OpenCV套件

```
# 是否翻轉图片
flip = self.rand() < .5
if flip:
    image = image.transpose(Image.FLIP_LEFT_RIGHT)
```

圖十三 翻轉部分程式碼

圖十四 翻轉後與原圖比較

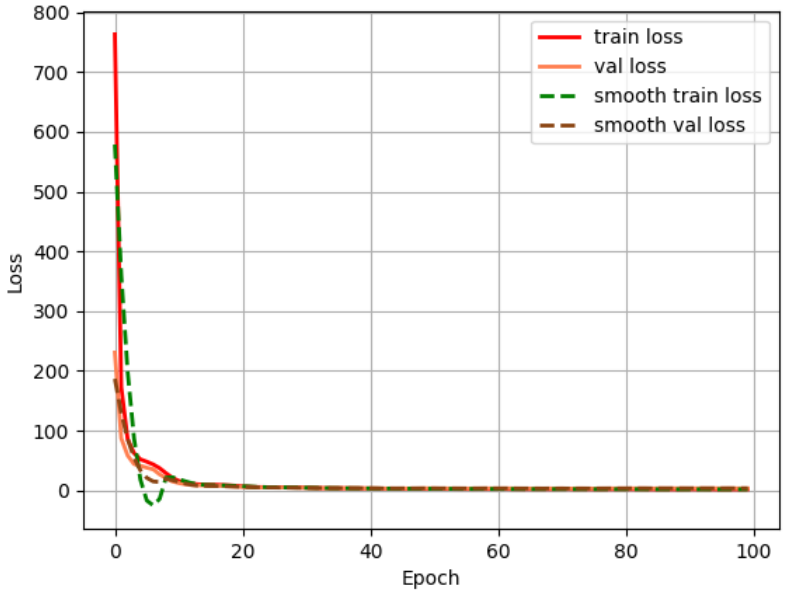
```
# 色域變換
hue = self.rand(-hue, hue)
sat = self.rand(1, sat) if self.rand() < .5 else 1 / self.rand(1, sat)
val = self.rand(1, val) if self.rand() < .5 else 1 / self.rand(1, val)
x = cv2.cvtColor(np.array(image,np.float32)/255, cv2.COLOR_RGB2HSV)
x[..., 0] += hue*360
x[..., 0][x[..., 0]>1] -= 1
x[..., 0][x[..., 0]<0] += 1
x[..., 1] *= sat
x[..., 2] *= val
x[x[:,:, 0]>360, 0] = 360
x[:, :, 1:][x[:, :, 1:]>1] = 1
x[x<0] = 0
image_data = cv2.cvtColor(x, cv2.COLOR_HSV2RGB)*255
```

圖十五 使用OpenCV調整HSV色彩空間

圖十六 調整HSV參數後與原圖比較

參數方面，每張訓練圖片在送入模型之前，因為GPU記憶體限制，解析度會調整成416*416，並把batch size調成4，learning rate根據batch size調整成1e-4，總共訓練100 epochs。

圖十七是每一個epoch最後計算出來的loss，可以看出loss在訓練的後半收斂，這也是我們要訓練100 epochs的原因，在訓練前我們並不知道loss會在哪裡開始收斂。

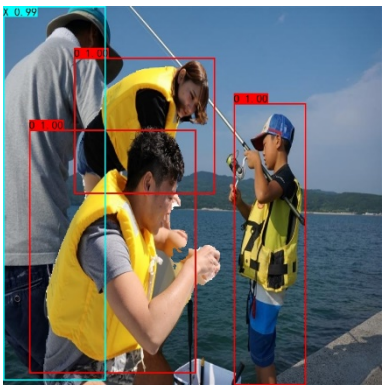


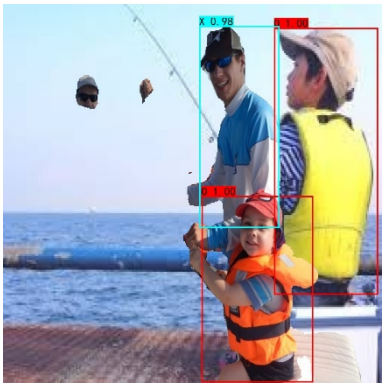
圖十七 loss曲線圖

第五節 訓練集以及辨識結果

訓練集的訓練圖和模型辨識的結果圖如圖十八所示，左邊為原圖，右邊為模型辨識的結果圖。在模型設定裡，我們將有穿救生衣的標籤設為0，並把偵測框顏色設成紅色，沒穿救生衣和沒穿好救生衣的標籤設為X，並把偵測框顏色設成藍色。





圖十八 訓練集原圖以及模型辨識結果

第六節 驗證訓練結果

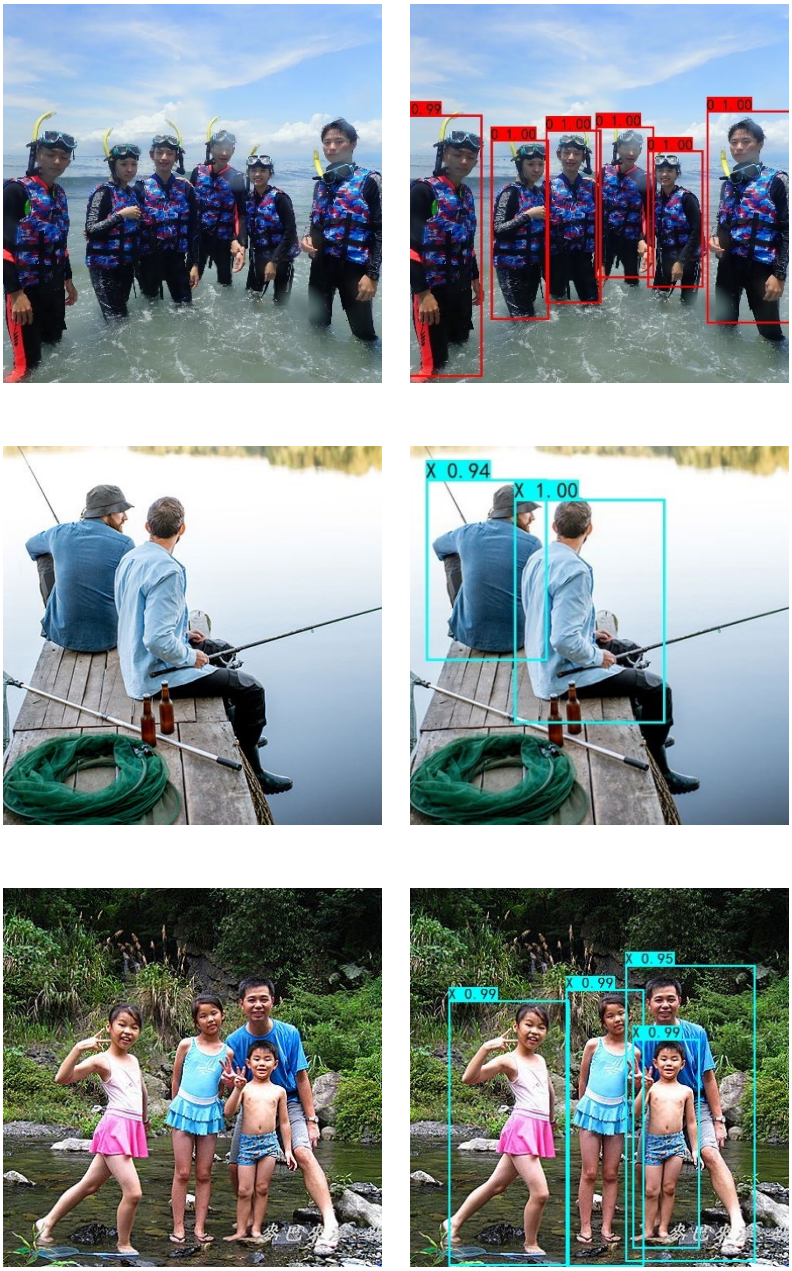
為了驗證訓練結果，我們事前將收集到的資料集(不包含合成圖)隨機分出二成來當作測試集，並計算出Recall、Precision、AP([2])，如表二所示。

表二 最終模型準確率

	Recall	Precision	AP
有穿救生衣	83.00%	83.00%	83.24%
沒穿救生衣	81.61%	82.73%	84.81%

其中Recall代表我們框出ground truth的比例，舉例來說，若圖片中有十個沒穿救生衣的人，而我們的模型框出了其中的8個，則代表Recall(召回率)為80%，Precision代表偵測框正確的比例，若我們的模型在一張圖片給出10個框，而其中有八個框是正確的，則代表我們的Precision(準確率)為八成，以Recall為X軸，Precision為Y軸，記錄不同confidence threshold狀態下的Recall和Precision，計算這條線下的面積則為AP。

圖十九是測試集原圖以及模型辨識的結果圖，左邊是原圖，右邊是模型辨識結果。



圖十九 測試集原圖及模型辨識結果

第七節 系統功能

接下來要做的就是建構AI系統的其他功能，但是在那之前，我們要先選擇邊緣計算的裝置。根據運算能力以及架設成本，我們選擇使用Nvidia的Jetson TX2([1])。Jetson TX2的體積小，但是有足夠的運算能力去應付模型的需求，缺點是本身沒有達成功能的其他硬體，例如音效卡、喇叭、攝影鏡頭等。如果要讓機器拍攝影像，還有發出聲音，都需要在額外加裝這些配件。

以下為系統實現的功能：

實際程式較圖片複雜，為容易說明，以下使用我們程式較簡單的雛型說明。

1.畫面輸入:

圖十八是攝影機拍攝影像的程式，程式會利用攝影鏡頭去捕捉畫面，將擷取出來的每一幀當作一張圖片 ，向模型輸入。這裡是用到OpenCV去擷取鏡頭影像，並送入模型辨識。

```python
import cv2

cap = cv2.VideoCapture(0)
while(cap.isOpened()):
    ret, frame = cap.read()
    if ret:
        cv2.imshow('frame', frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break
cap.release()
out.release()
cv2.destroyAllWindows()
```

圖二十 攝影機範例

從圖中可以看出，當我們不再使用此程式時可以輸入Q鍵來中止程式，且在中止前會將顯示的視窗消除。

2.警告當地遊客:

我們將使用喇叭警告遊客的部分寫成一個函式，當達成偵測到有人沒有穿救生衣這個條件時，我們將呼叫這個函式讓喇叭發出聲響警告遊客。

```python
import os

def alarm():
    os.system("mpg321 tools/alarm.mp3")
```

圖二十一 Jetson TX2上的警告程式

從圖十九中看出我們的作法是將警告作為一個函式，當用於實際程式碼後，就可以很方便地在想要的地方呼叫函式即可，例如某些條件過後或者是某些特定的段落。

3.傳遞訊息：

如果多次勸告無果，將所在地的地址利用傳送到管理機構，請管理機構派遣工作人員前來處理。這裡使用Python自帶的socket撰寫，程式會從一個叫address.txt的文字檔讀出地址，並傳送地址到管理機構，所以在安裝裝置時只要在文字檔更改文字內容即可，不需動到程式碼本身。(圖二十為加載在我們裝置上的client端程式)

```python
from socket import *

def client():
    #實驗室電腦
    # serverip='120.126.151.182'
    # serverport=8887

    #在自己電腦測試
    serverip='127.0.0.1'
    serverport=8888
    client=socket(AF_INET,SOCK_STREAM)
    client.connect((serverip,serverport))
    address_file = open('tools/address.txt', 'r')
    address = address_file.read()
    client.send(address.encode())
    print(client.recv(1024).decode())
```

圖二十二 client端的程式碼

要注意到，作為server的IP須為固定IP。若如圖中設置server的IP為127.0.0.1則是連接到本機上，若要變動server的裝置，則必須確認其IP是正確的，同時因為我們在第十三行中有用到address.txt這份文檔，所以每當變動檔案的位置(不論是在同一台電腦上變動或者是將檔案從某個裝置移動到另一個裝置)，都要一起移動這份文檔。

4.接收訊息：

這裡的作法是讓相關機構的電腦運行server的程式(如圖二十一)，但是會將程式界面做成清楚直白的圖形使用者界面，讓不熟悉軟體操作的人也能簡單使用，而且在收到訊息時，程式會發出通知音效，告訴使用者要去檢查事情發生的位置（最新到最舊的訊息分別由上到下排列）。因為我們只是要做一個功能簡單追求效率的GUI(Graphical User Interface，圖形使用者界面)界面(結果如圖二十二)，所以這部份是用Python自帶的GUI套件Tkinter。若日後需要有介面切換，或者是其他功能，則另外有QtPy這個選項。另外，播放音效的程式則是使用playsound這個套件撰寫。

```python
def server():
    # HOST = '120.126.151.182' #實驗室電腦
    # PORT = 8887

    #在自己電腦測試的話
    HOST = '127.0.0.1'
    PORT = 8888
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    s.bind((HOST, PORT))
    s.listen(5)

    print('server start at: %s:%s' % (HOST, PORT))
    print('wait for connection...')

    while True:
        conn, addr = s.accept()
        print('connected by ' + str(addr))

        while True:
            indata = conn.recv(1024)
            if len(indata) == 0: # connection closed
                conn.close()
                print('client closed connection.')
                break
            message = indata.decode()
            print('recv: ' + message)

            # 將傳來的訊息顯示在GUI視窗上
            show_message(message=message)

            outdata = 'echo ' + indata.decode()
            conn.send(outdata.encode())
```

圖二十三 server的程式碼

圖二十四 server端顯示訊息的GUI界面

```
def notify_sound():
    playsound('tools/notify.mp3')
```

圖二十五 server端播放通知音效的程式

如圖二十三所示，跟Jetson TX2上的警告程式相同，我們同樣將通知程式寫成一個函式，方便我們隨時去呼叫它。

5.記錄：

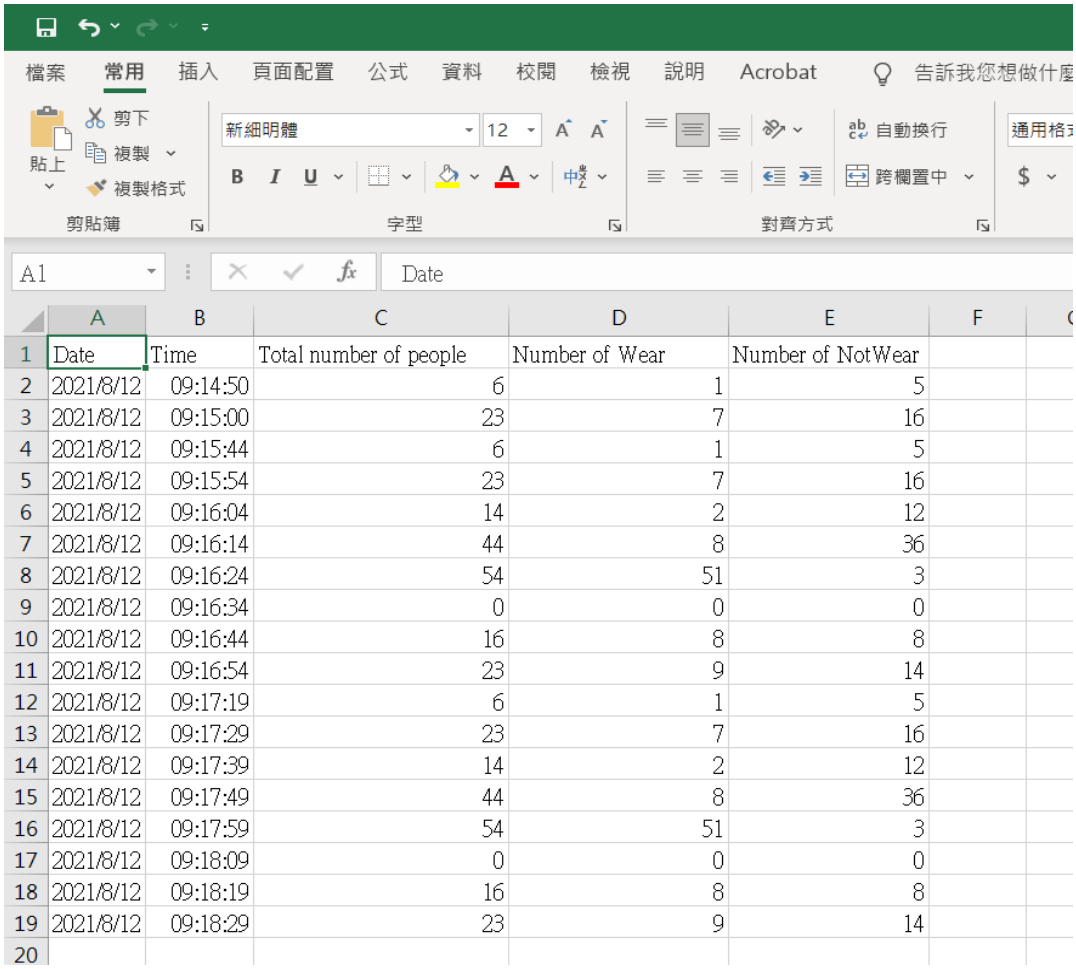
每隔一段時間記錄當地有無穿著救生衣的人的數量，並將資訊儲存在csv檔裡(csv檔可以使用Excel來進行閱覽、處理等等操作)，以供後續相關當局能方便評估當地狀況及調整政策之所需，或是研究人員用於大數據統計。

```
import time
import csv

def save_record(count):
    # count.shape = (1, 2)
    # 裡面個別紀錄有穿和沒穿的數量
    # count[0][0]為有穿的數量，count[0][1]為沒穿的數量
    date_record = str(time.strftime("%Y-%m-%d", time.localtime()))
    time_record = str(time.strftime("%H:%M:%S", time.localtime()))
    total_num = count.sum()

    with open('tools/record.csv', 'a', newline='') as csvfile:
        writer = csv.writer(csvfile)
        writer.writerow([date_record, time_record, total_num, count[0][0], count[0][1]])
```

圖二十六 儲存資料程式碼

我們以圖二十四來做說明，首先先加入time及csv套件，同上面幾個的理由一樣，我們將這個寫成一個函式。

再來藉由time套件內的工具將目前日期以及時間記下來，還有目前記到第幾筆資料，然後再開啟record.csv檔，寫入有分成好幾個模式，其中較為相像的有’w’(write)、’a’(append)模式，若使用write模式則是將文件內所有內容清空然後寫入，而append模式則是從文件內容的末端開始寫入，因此我們在這邊使用append模式。



	A	B	C	D	E
1	Date	Time	Total number of people	Number of Wear	Number of NotWear
2	2021/8/12	09:14:50	6	1	5
3	2021/8/12	09:15:00	23	7	16
4	2021/8/12	09:15:44	6	1	5
5	2021/8/12	09:15:54	23	7	16
6	2021/8/12	09:16:04	14	2	12
7	2021/8/12	09:16:14	44	8	36
8	2021/8/12	09:16:24	54	51	3
9	2021/8/12	09:16:34	0	0	0
10	2021/8/12	09:16:44	16	8	8
11	2021/8/12	09:16:54	23	9	14
12	2021/8/12	09:17:19	6	1	5
13	2021/8/12	09:17:29	23	7	16
14	2021/8/12	09:17:39	14	2	12
15	2021/8/12	09:17:49	44	8	36
16	2021/8/12	09:17:59	54	51	3
17	2021/8/12	09:18:09	0	0	0
18	2021/8/12	09:18:19	16	8	8
19	2021/8/12	09:18:29	23	9	14
20					

圖二十七 用Excel開啟csv檔

如圖二十五所示，記錄資料包含日期、時間、當時總共有幾人、多少人有穿救生衣、多少人沒穿救生衣。

第三章 結果與討論

第一節 準確度的提升

我們藉由製造合成圖的方式，增加了至少一倍的資料，也藉此讓模型準確率提昇。從期中報告中召回率六成提升到現在的八成(表四)，而準確率則更是由三成提升至目前的八成(表三為期中報告時的模型準確度)。

表三 期中報告模型準確度

	Recall	Precision
有穿救生衣	67.8%	32.0%
沒穿救生衣	56.3%	28.1%

表四 最終模型準確率

	Recall	Precision	AP
有穿救生衣	83.00%	83.00%	83.24%
沒穿救生衣	81.61%	82.73%	84.81%

第二節 系統整合

目前系統架構整合以下功能：

1.機器偵測遊客有無穿著救生衣

2.在發現沒有時發出警告

3.如果多次警告無果的情況下通知主管機關，由主管機關自行決定執行相對應的行動

4.主管機關接收訊息的server端界面設計成GUI

5.每隔一段時間在本地端進行資料的儲存，供日後研究人員或主管機關進行統計

第三節 GUI的增加

考慮到一般人員較不熟悉終端機的視窗以及其想表達的資訊，因此我們在這些系統的基礎下，建立了一個較為清晰的介面，包含地址及接收時間，並且一次容納10條

歷史資訊，使其並不會因為眾多地區都有人沒有穿著救生衣而導致後面的資訊掩蓋掉前面的資訊，而造成不必要的憾事發生。

第四章 結論

這次研究，我們做出一套AI偵測系統，目的是要藉由機器去取代人力，令其能夠辨識遊客的位置以及判斷有無穿著救生衣，並在發現沒有時發出警告，來達到減少人力成本的效果，不需要主管機關時時刻刻派遣工作人員到遊客所在地巡邏。

系統採用本地端運算的形式，偵測模型能在當地即時運行，不需將監控畫面上傳網路，交由server端的電腦去做模型運算。

另外，系統還能在多次勸告無果後，透過網路封包傳送所在位置給主管機關，請主管機關派遣工作人員前來處理。最後，我們在系統上添加統計當地遊客穿著救生衣狀況的功能，讓這些統計資料能夠用於政策的調整或大數據的應用。

參考資料

1.NVIDIA，〈高效能的邊緣端人工智慧I NVIDIA Jetson TX2〉，（2017/03/14）<https://www.nvidia.com/zh-tw/autonomous-machines/embedded-systems/jetson-tx2/>(2 Sep. 2021)

2. Medium，〈深度學習系列: 什麼是AP/mAP?〉，(2018/09/14)。<https://chih-sheng-huang821.medium.com/%E6%B7%B1%E5%BA%A6%E5%AD%B8%E7%BF%92%E7%B3%BB%E5%88%97-%E4%BB%80%E9%BA%BC%E6%98%AFap-map-aaf089920848>(2 Sep. 2021)

3.新北市政府消防局，〈新北市消防統計年報96頁〉，(2019/06)，<https://www.fire.ntpc.gov.tw/uploaddowndoc?file=firestatic/202006231146350.pdf&filedisplay=108%E5%B9%B4%E7%B5%B1%E8%A8%88%E5%B9%B4%E5%A0%B1.pdf&flag=doc>(2021/09/12)

4.聯合新聞網，〈瘋狗浪捲釣客:釣魚也要穿救生衣?意外落海的求援策略〉，(2020/09/15)，<https://opinion.udn.com/opinion/story/121058/4858532>(2021/09/12).

5.”Deeplabv3+: Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation”,Cornell University arXiv website, 22 Aug 2018,<https://arxiv.org/abs/1802.02611>

6.“Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”,Cornell University arXiv website, 4 Jun 2015,<https://arxiv.org/abs/1506.01497>

7.”LabelImg,” Github,3 Dec 2018 ,<https://github.com/tzutalin/labelImg>(2 Sep. 2021).

8.”PixelLib,” Github, 13 Apr 2020,<https://github.com/ayoolaolafenwa/PixelLib>(2 Sep. 2021).

9.”You Only Look Once:Unified,Real-Time Object Detection”,Cornell University arXiv website,8 Jun 2015,<https://arxiv.org/abs/1506.02640>

10.“YOLOv4: Optimal Speed and Accuracy of Object Detection”,Cornell University arXiv website, 23 Apr 2020 ,< https://arxiv.org/abs/2004.10934>