自主學習計畫

利用Python來製作圖像識別AI

一王45廖培君

∴自主學習主題:

利用Python的擴充函式庫--Pytorch來製作一個圖形識別的AI以及將它實際應用。

∴自主學習目標:

目標是捕捉音樂遊戲中的遊玩過程，並將遊玩過程轉換為一個包含時間、滑鼠位置及滑鼠點擊狀態的紀錄檔，或根據紀錄檔回放遊玩過程，最終讓AI自己去操作。

∴自主學習計畫&執行紀錄:

週次	實施內容與進度	自我檢核 (A.依計畫完成B.部分完成C.未完成)	補充(備註)
1	研究大致上要撰寫的方向，選擇要使用甚麼來呈現(應用)圖像識別技術。	A	使用音樂遊戲
2	考慮要使用哪種程式語言來撰寫AI。	A	使用Python
3	研究tensorflow、pytorch等深度學習的函式庫。	A	使用Pytorch
4	撰寫使用者操作介面。	A	有基本的架構了
5	撰寫AI的辨識譜面部分(1/2)	A	
6	撰寫AI的辨識譜面部分(2/2)	A	
7	幫訓練物上標籤(1/2)	A	
8	幫訓練物上標籤(2/2)	A	
9	調整參數，並且引入函式庫	A	

10	調整函式庫	A	
11	修正程式上的錯誤	A	
12	調整存取	A	修好了一個BUG又有一個...
13	調整存取	A	修好了，但是太慢了
14	調整數據	A	開始寫GPU版
15	測試並修正	A	
16	微調	A	GPU版本壞了
17	測試並修正*2	B	GPU版又又又壞了，又好了
18	小結	A	

∴自主學習成果:

經過了無數次的微調，最終我終於成功寫出了一個能夠成功遊玩音樂遊戲的AI。

雖然還有很大的問題存在(執行速度問題)，但是在接下來的幾年內，我會試著將其完善(做成一個可記錄的AI)，儘管會遇到許多困難與挫折，但我仍會勇於試錯，並從失敗之中得到那成功的果實。

∴自主學習歷程&省思

第一週:

開始想著去做一個可以輔助使用者去玩遊戲的AI,一開始其實是想朝著製作一個音樂遊戲的方向去製作的，但是感覺去寫一個破解它的AI較為有趣(和具挑戰性)，所以決定製作可以協助去玩音樂遊戲的AI。

列舉音樂遊戲(電腦)種類:Osu!, Friday Night FnukiRhythm Doctor...等(我腦袋裡只想到這些

最後，使用[Osu!]來當作機器學習的對象，不管在技術上或者是複雜性上都大大的降低。

▲如上圖，這個就是[OSU!]，一個電腦上的音樂遊戲應用程式。 **(照片By自己)**

綠色的是要點擊的目標物，而粉色的是自己的滑鼠游標。

我是分隔線

第二週:

研究要用哪種程式語言來撰寫這個機器學習相關的程式(因為會的語言還蠻多的)。

一開始想試試看用Java，後來看到了Python有Torch的函式庫，所以就改用Python了(比較方便)

(後來也有想過用C#來寫，但是他的語法挺複雜的，不想把自己弄那麼累)。

▲這是Pytorch官網的Available list，上面寫了能用的程式語言和版本。 **(照片By Pytorch官網)**

我使用的是Python的pip來安裝，而不是用conda。(因為我比較習慣用Visual Studio Code)

第三週:

既然決定要使用Python了，就來把一些近些年來比較流行和我發現比較好用的函式庫都學一下，如下:

pyautogui:模擬裝置輸入輸出操作&一些其他的功能(Ex:螢幕截圖，好用 :3)

tenserflow:用於機器學習的平台和函式庫

(可以用Keras來訓練東西.etc,後來查了一下才知道這個原來是由Google研發的)

pillow:利用這個函式庫來讀取、處理和保存圖像

(但我還是用pyautogui來簡單弄一下就好了)

…還有很多，介紹不完。

▲這是剛剛提到的那些函式庫在PYPI(Python Package Index)上的下載指令(pip install XXX)

後記:後來總共在電腦中下載的函式庫竟然高達30-40種。

第四週:

設計一個讓使用者可以選擇模式的介面(目前用簡單一點，先用基本的輸入，進階[像整合成一個Server之類的]看下學期能不能把它完成)，有寫入模式和操作模式，但是詳細的架構(時間有沒有對上之類的)應該之後再把它一併解決吧。

```
10  Song_Name = str(input("Song Name:"))
11  if not os.path.exists(f'{path_song}/{Song_Name}'):
12      os.makedirs(f'{path_song}/{Song_Name}')
13  path = f'{path_song}/{Song_Name}/{Song_Name}.txt'
14  if not os.path.exists(f'{path_song}/{Song_Name}/{Song_Name}test.png'):
15      Screenshot = m.screenshot(f'{path_song}/{Song_Name}/{Song_Name}test.png')
16  image = cv2.imread(f'{path_song}/{Song_Name}/{Song_Name}test.png')
17  i = 0
18
19  with open(path, 'w+') as f:
20          WriteOrPlay = str(input("Writein or Play?"))
21          while Song_Not_End:
22              if WriteOrPlay == "Writein":
23                  if not os.path.exists(f'{path_song}/{Song_Name}/{Song_Name}{i}.png'):
24                      Screenshot = m.screenshot(f'{path_song}/{Song_Name}/{Song_Name}{i}.png')
25                      img = cv2.imread(f'{path_song}/{Song_Name}/{Song_Name}{i}.png')
26                      if img is None:
27                          print(f"Failed to load image {f'{path_song}/{Song_Name}/{Song_Name}{i}.pn
28                      continue
29              if WriteOrPlay == "Play":
30                  break #still developing
```

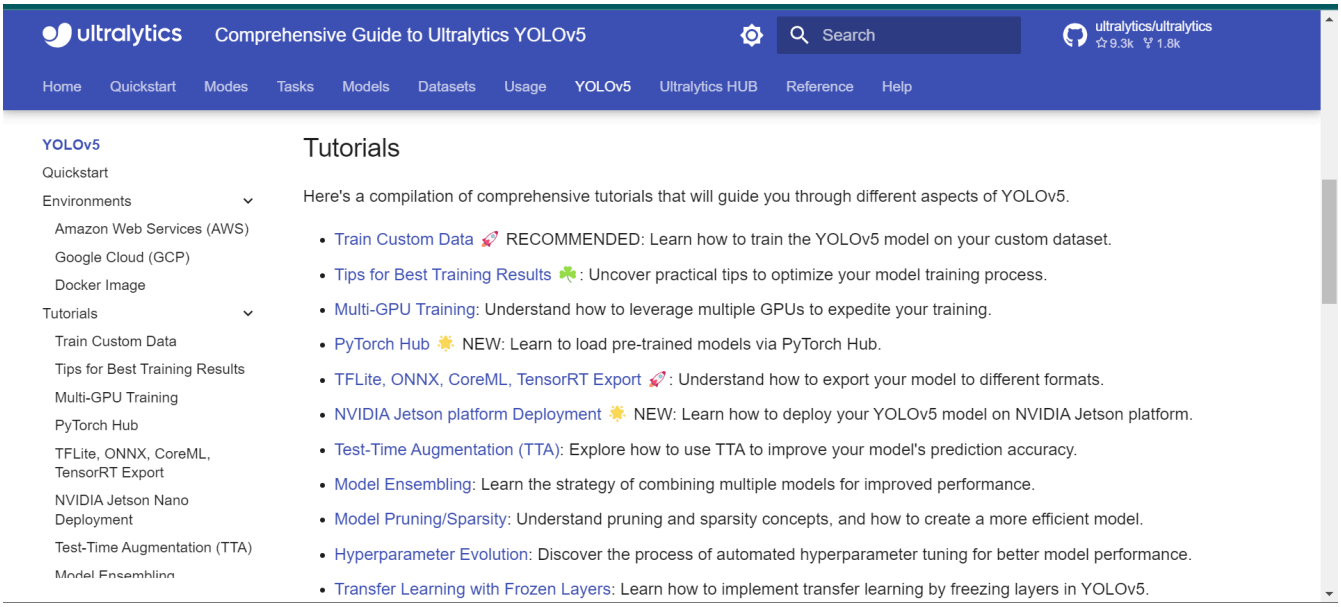
▲開始撰寫基本的框架，發現真的有點簡陋

第五週:

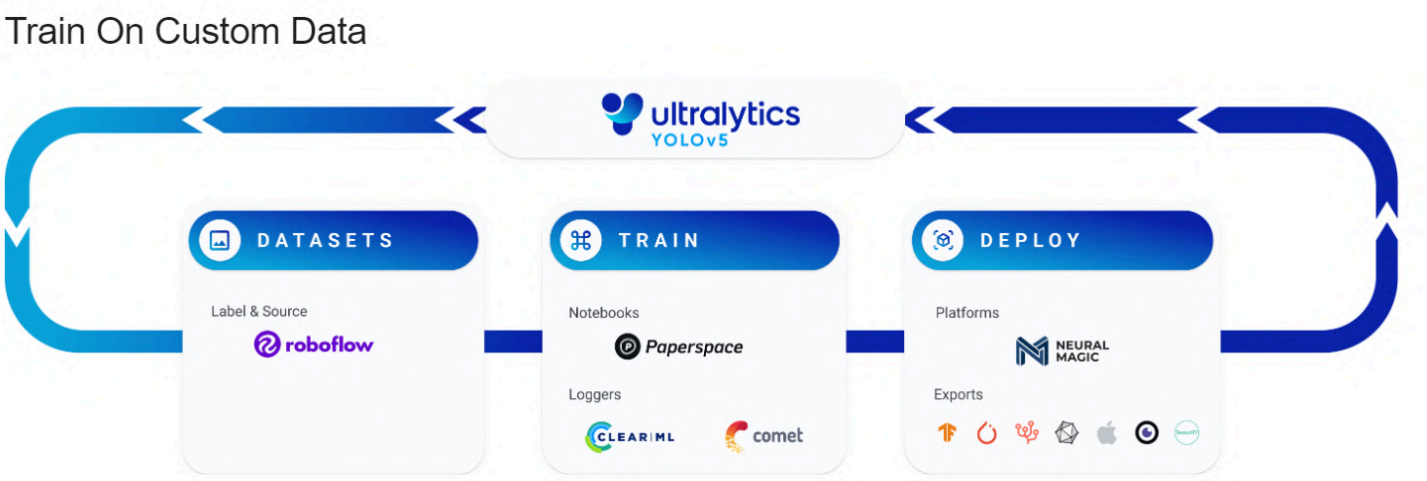
開始撰寫程式來辨識譜面，研究遊戲的譜面格式和相關的遊戲機制，並開始設計一個方法辨識和解析譜面。目前想到的是利用AI來識別螢幕上的圖像來達成檢測的目的。雖然這個的延遲會是一大硬傷，但目前我的技術也只能做那樣了。後面可能會搭配演算法和GPU來加速吧。

第六週:

在這週，我找到了一個神奇的物件模型框架(用於訓練AI)--YOLOV5.利用這個可以讓我自訂想要去訓練的東西。只是物件需要一個一個地去幫它上標籤，很麻煩。



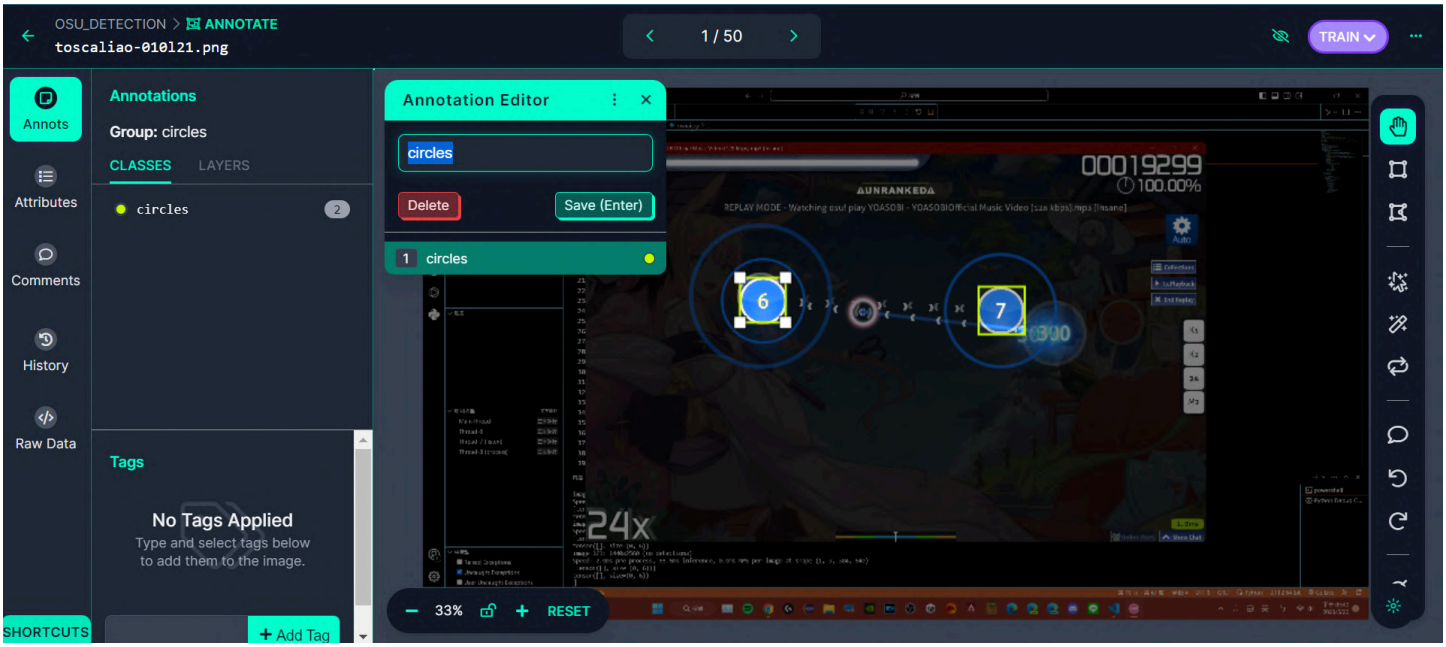
▲這是yolov5的首頁，他的第一項"Train Custom Data"就是我這次所使用的。



▲這是它的運作原理，然後他其中的"Dataset"是我要自己上Labels的部分。

第七週:

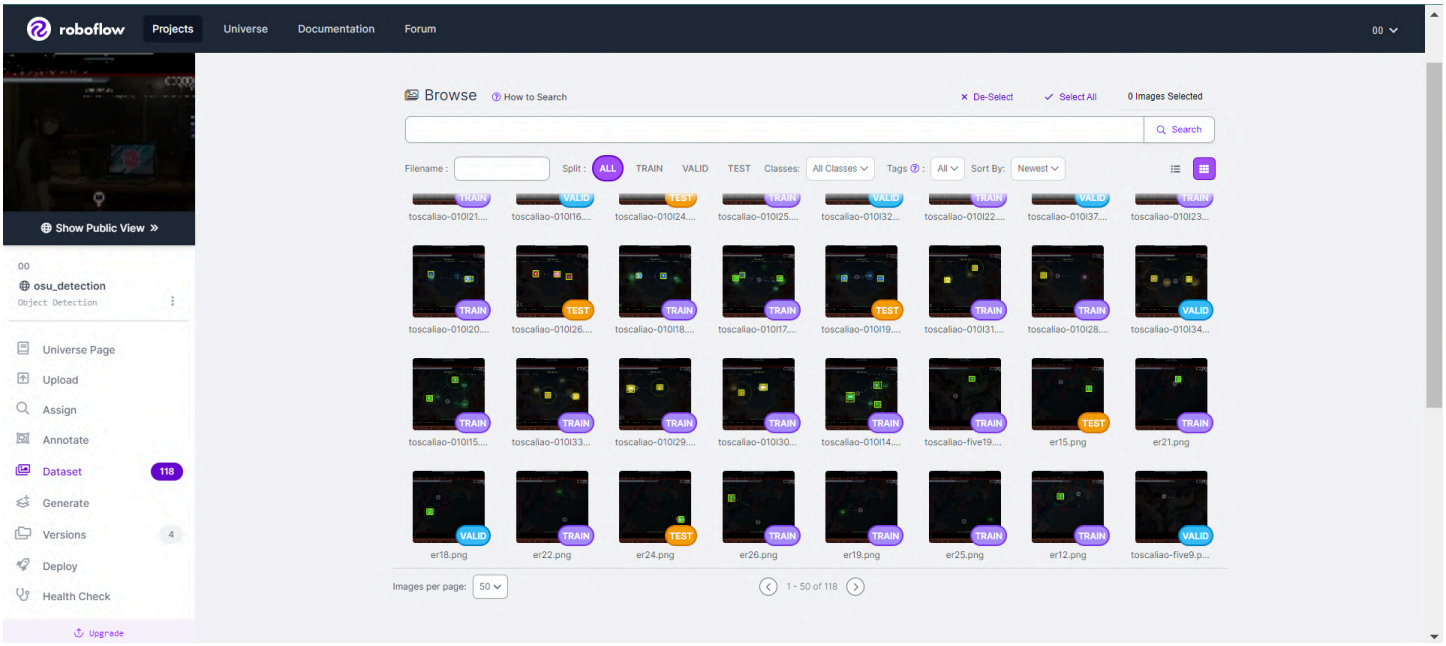
幫識別物體上標籤。我利用了一個名為roboflow的網站來幫物體上標籤。之所以選擇這個網站是因為它輸出檔案時很方便。



▲要上大量的Labels，作業時間繁長。

第八週:

繼續幫物體上標籤。上了共約100個，希望這幾周的努力不會白費。



▲總算是把標籤上的差不多了

第九週:

把輸出的訓練模型丟到程式裡面去執行，發現了一些(或者是說 大量的)Bug，所以決定把整體的邏輯再順一次，看看哪裡有錯誤。後來，發現是因為引入的函式庫版本錯了，所以後來把它重新安裝成正確的版本(先解除安裝再把它安裝成指定的版本)。

▲以[pip uninstall XXX]來解除安裝。

(這週沒有照片，因為真的太忙了)

第十週:

函式庫引入又發生了一些錯誤，所以把它重新安裝並且導入，然後發現有些東西無法執行是因為同樣名稱的東西被重複導入了，所以後來把它用其他的取代掉，並且把舊的刪除。

不小心把兩個都刪了，所以將它重裝

第十一週:

執行-->無法執行，雖然無法執行，但我也同時找到了Bug。原來是我在提取陣列的時候出了問題(超出範圍和有些沒有偵測到的情況)，後來我把它重新改了一下，好像還是有一些錯誤。

```
if len(predictions) > 0:
    x1 = int(predictions[0]['x'])
    y1 = int(predictions[0]['y'])
    circle_list.append({
        'position': (x1, y1),
    })
else:
    x1, y1 = width / 2, height / 2
```

▲直接這樣改，好像就好了...

第十二週:

雖然用Debugger看上周的程式片段是可以成功的執行，但是後面的片段還是無法執行(又卡住了...)，於是我決定用另外一種方式來寫這個code.

用另外一種方式重寫，也就先從初期設置的code開始。

```
Song_Name = input("Song Name:")
if not os.path.exists(f'{path_song}/{Song_Name}'):
    os.makedirs(f'{path_song}/{Song_Name}')
path = f'{path_song}/{Song_Name}/{Song_Name}.txt'
screenx, screeny = m.size()
screenx = int(screenx)
screeny = int(screeny)
if not os.path.exists(f'{path_song}/{Song_Name}/{Song_Name}test.png'):
    Screenshot = m.screenshot(f'{path_song}/{Song_Name}/{Song_Name}test.png')
image = cv2.imread(f'{path_song}/{Song_Name}/{Song_Name}test.png')
height, width = image.shape[:2]
```

▲這是初期設置，又要重新撰寫了

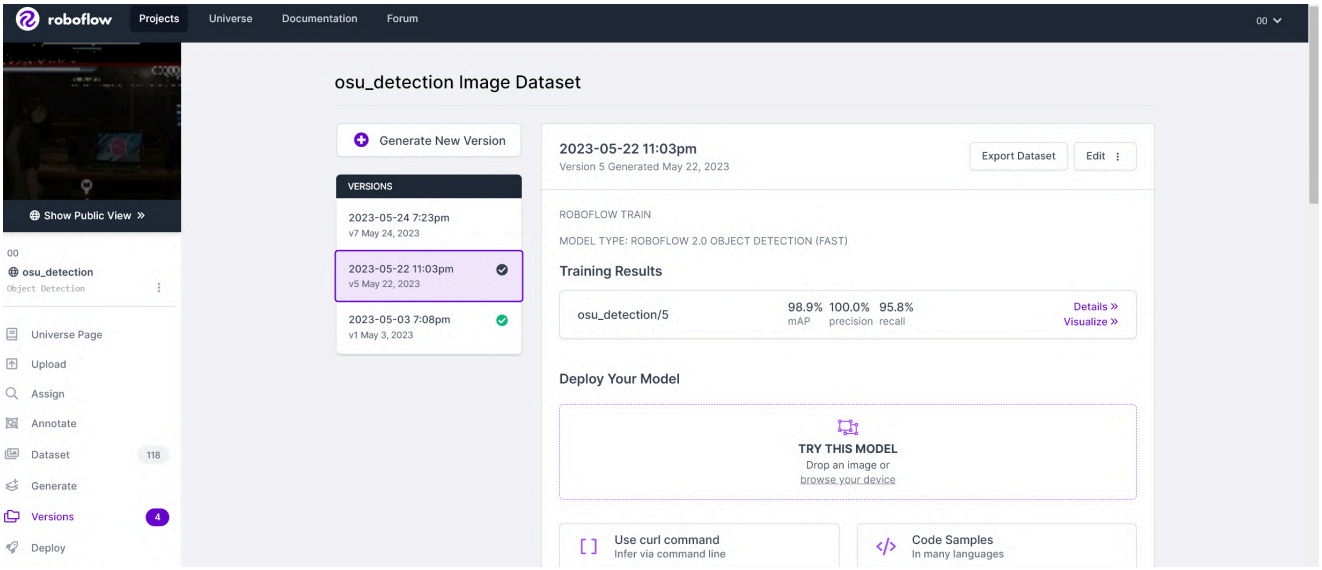
第十三週:

調整過後，終於可以成功執行。但是數據不太準，所以回去把model重新再train一次。

第十四週:

結果有些不準確，所以我改了另外一種機器學習的方式。

後來發現，使用Roboflow內建的機器學習模型(就是把自己的上過的標籤拿去官網上面訓練)的話結果會相當的準確，所以使用了它。其額度只有三次，刪掉的話就不能重新Train了。



```
rf = Roboflow(api_key="i5kLF02XqivLHtTH1wFX")
project = rf.workspace().project("osu_detection")
model = project.version(5).model
```

▲只要這樣就可以用裡面的模型了，API KEY在這裡，希望可以讓有需求的人們自行使用。

第十五週:

用了新的方式之後，我發現效率實在是太慢了。

所以我決定改用Gpu來進行運算，這樣的話會比較快。

```
image_tensor = torch.tensor(image_array).unsqueeze(0).permute(0, 3, 1, 2).float().to(device)

with torch.no_grad():
    image_tensor = image_tensor / 55.0
    predictions = model.predict(image_array)
```

▲Gpu+自己改造過的，把影像化為tensor，好像以目測而言比較快了。

∴結論:

在這次學習中，我學會了去利用Python製作（撰寫）一個基礎AI，並且進一步提升了我的問題解決能力。我同時也學會了許多進階的程式語法以及更多擴充的指令，讓我的知識得到了提升。這樣的學習經驗不僅讓我對AI有了更深入的理解，也更進一步的使我在解決現實問題時學會了以不同的觀點去理解並觀察。

在這次的學習過程中，不只學會了圖像辨識的原理以及程式架構的本身，更從中學到了屢敗屢戰，永不放棄的能力，我想，這才是我在這段學習的過程中，所學習到的，最重要的吧。

∴出處:

- Osu! 官網: 🌐 welcome | osu!
- PYPI 官網: 🌐 PyPI · The Python Package Index
- Pytorch官網: 🌐 PyTorch
- Roboflow 官網: 🌐 Roboflow: Give your software the power to see objects in images and video