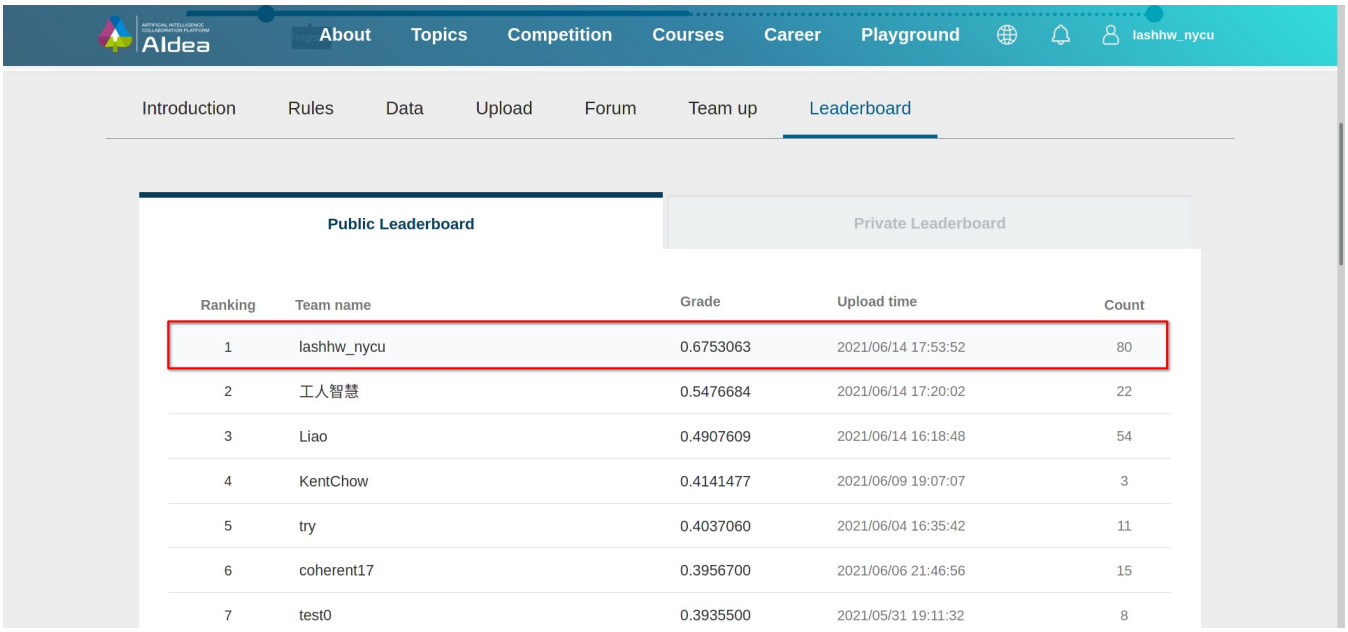
* 為了方便閱讀，程式碼以 Colab 呈現，放在最後一頁。

1. Leaderboard 的競賽成績截圖



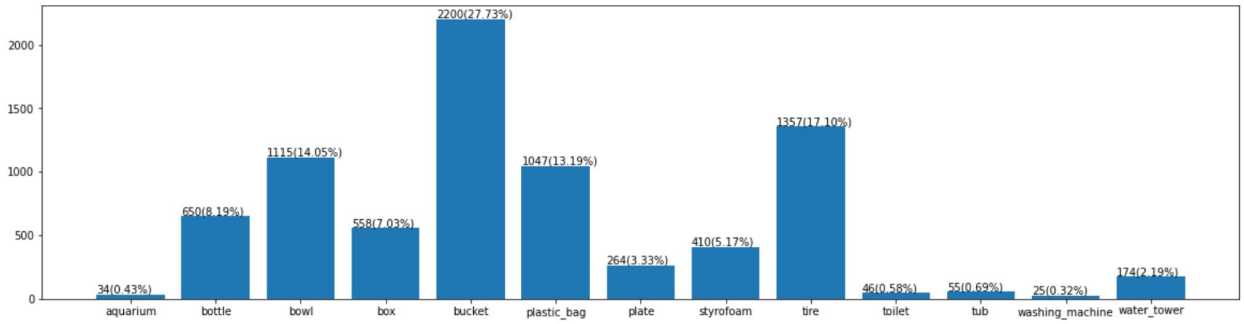
Ranking	Team name	Grade	Upload time	Count
1	lashhw_nycu	0.6753063	2021/06/14 17:53:52	80
2	工人智慧	0.5476684	2021/06/14 17:20:02	22
3	Liao	0.4907609	2021/06/14 16:18:48	54
4	KentChow	0.4141477	2021/06/09 19:07:07	3
5	try	0.4037060	2021/06/04 16:35:42	11
6	coherent17	0.3956700	2021/06/06 21:46:56	15
7	test0	0.3935500	2021/05/31 19:11:32	8

2. 在競賽過程中，做了哪些實驗與測試

在這個競賽中，我選擇使用 yolov5 作為物件偵測的模型。我之所以會選用 yolov5 訓練的主要原因是因為 yolov5 集成許多機器學習常用的技巧，例如我們可以輕易地調整 augmentation 的參數，與修改程式碼作額外的 augmentation。以下我會條列式說明我在這次競賽中做的實驗與測試：

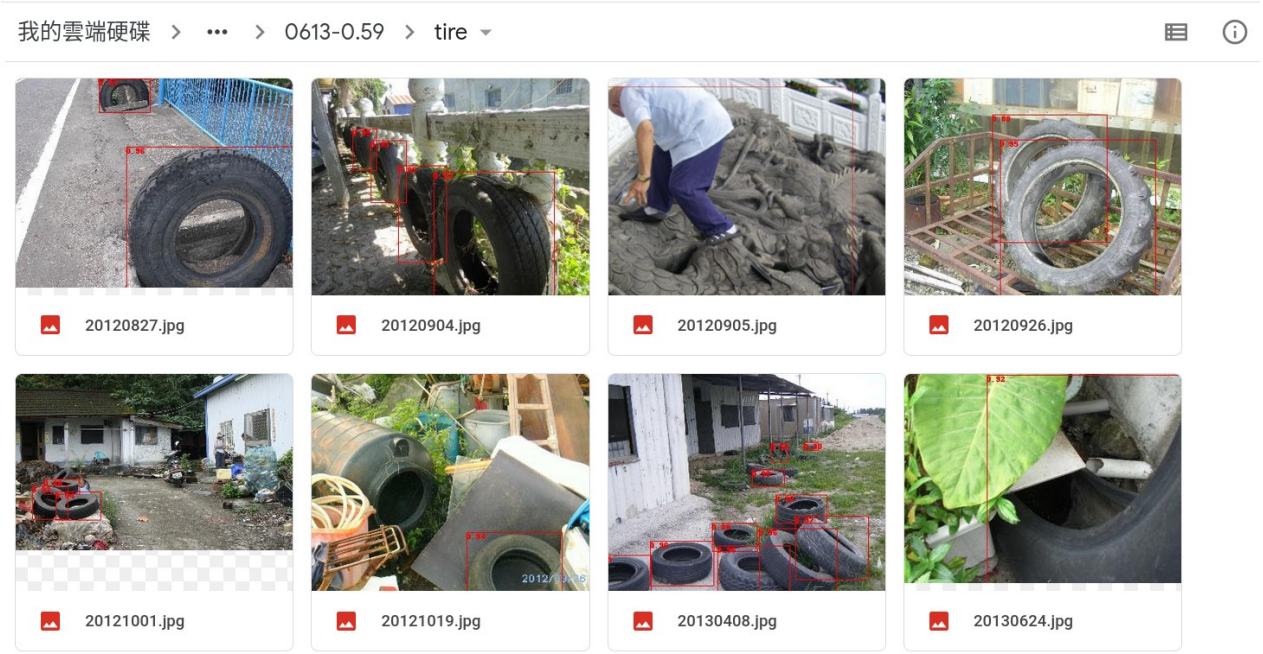
- **使用 COCO 資料集的 pre-trained weight 進行 transfer learning**
  由於 Google Colab 使用時數相當有限，無法使用自己帳號先行訓練 COCO 資料集作 transfer learning，因此我使用官方提供的 COCO pre-trained weight 進行 transfer learning。
  yolov5 提供了四種架構的模型，分別為 Small、Medium、Large 及 XLarge。受限於 Google Colab 只有一個 GPU，難以使用最大型的模型架構，我在這次競賽中主要選用 Medium 及 Large 訓練。

- **分析資料集 instances 分佈**

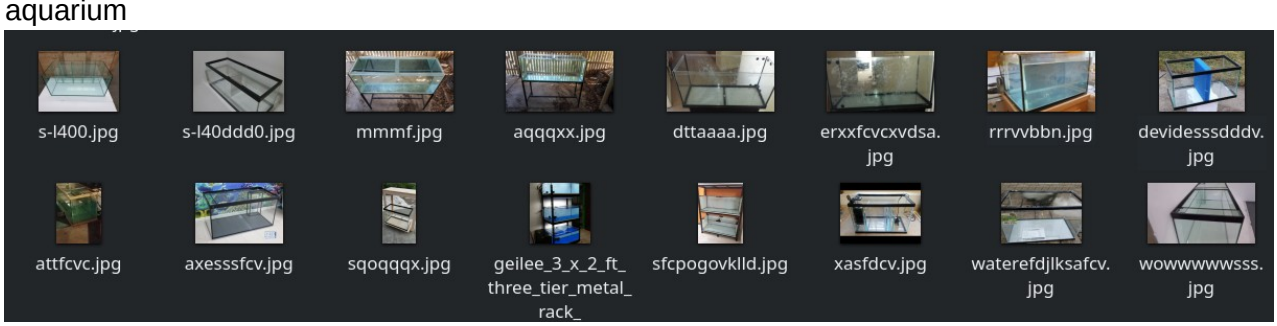


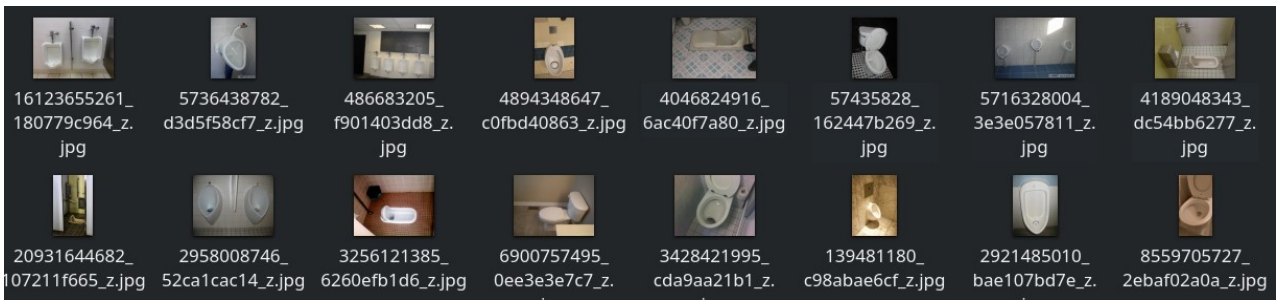
  我使用 matplotlib.pyplot 分析 train_images 中的 instances 分佈，發現 aquarium、toilet、tub、washing_machine 資料極度不足，如此少數的資料幾乎不可能訓練出能準確辨別這四種物件的模型，因此我之後進行的 augmentation 都是著手於增加這四種物件的數量。
  此外，我也得到 train_images 中的照片的長邊平均為 438 個 pixel，這也是我之後訓練模型的重要參考資料。

- **將預測結果分門別類，易於檢視**
  我額外寫了一些 code 將同個物件的偵測結果放到同個資料夾，並將物件以紅框標示。舉例來說，tire 資料夾存放所有含有輪胎的照片，並將輪胎以紅框包圍：



用這種方式可以找出模型常犯的錯誤，例如我找到模型常將「空調外機」辨識成「洗衣機」，因此我之後加入許多「空調外機」的照片使模型認識「空調外機」的長相。

- **增加 aquarium、toilet、tub、washing_machine 的資料個數**
  由於 Open Images 中有已標注的 Toilet、Bathtub、Washing machine、Plastic bag 可拿來訓練模型以辨識登革熱競賽的相應物件，我在 Open Images 中的 190 萬張照片挑選出約略 2000 張包含上述物件的照片及 1000 張不含上述物件的照片訓練 yolov5 模型。我使用此模型辨識 toilet、tub、plastic_bag、washing_machine，其餘類別則使用 AIdea 提供的 train_images 辨識。可惜的是，使用此方法成果不如預期，甚至比全部使用 AIdea 提供的 train_images 訓練表現還差。因此，我決定上網搜尋 aquarium、toilet、tub、washing_machine 手動標記。
  我在 Flicker、Google 上搜尋以上四種類別標記 300 餘張照片（其中包含 50 餘張不含任何辨識物件的照片用以降低 false positive 的機率），我主要選擇清晰可辨、背景單純的照片。以下為一些我找到的照片：
  aquarium
  

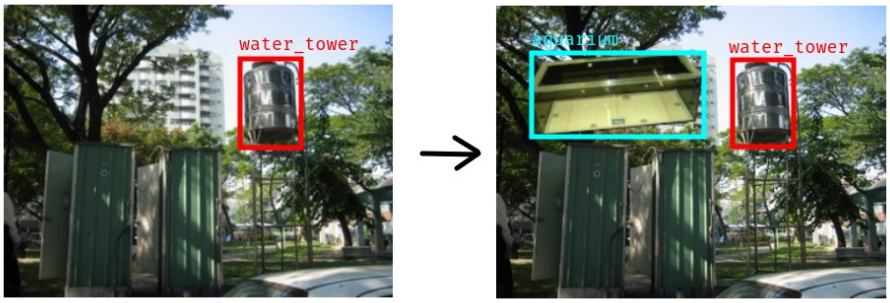
  toilet
  

tub

washing_machine

加上我自己標注的資料集之後，mAP 有上升，代表成效不錯。

- **修改 yolov5 程式碼作額外 augmentation**
  除了上述手動標注的資料集之外，我也額外製作許多 sticker（自己發明的名字，我想不到更好的了 XD）。sticker 是我將某些易於辨識的物件去除背景另外儲存，訓練時的 augmentation 會將這些 sticker 隨意貼到訓練照片不含其他辨識物件的地方。舉例來說，我的程式會隨機選一個 sticker 貼到圖片上：



這是我看到 [Simple Copy-Paste is a Strong Data Augmentation Method for Instance Segmentation] 這篇論文而想到的作法，我製作了 66 個 sticker，這些 sticker 在訓練時有 5% 的機率會貼到訓練圖片上（這些 sticker 貼到圖片前會作 scale、rotate 及 shear，以增加多樣性）。
aquarium 的 sticker：

washing_machine 的 sticker：

- **使用 yolov5 內建的 augmentation 來訓練**
  yolov5 有個很酷的 augmentation 方式叫做 mosaic，它會將每張照片多加三張其他照片形成類似「馬

賽克拼貼畫」的效果，舉例來說以下是經 augmentation 後的照片：

以上就是電腦在每次訓練見到的圖片長相。我認為這是相當聰明的作法，因為這樣能讓模型更能認識物件在不同環境下的長相。
此外，yolov5 還有許多可調整的 augmentation 參數，例如色彩調校、momentum 及 learning rate，不過這部份我沒有很了解，所以都是依照預設的參數作訓練。

- **使用 Ensemble Learning 提升預測準確率**
  由於 GPU 記憶體限制的關係，在大尺寸照片下 batch size 勢必要減低，因此當 Large 模型設定訓練尺寸為 832*832 時，batch size 最多只能為 7。如此一來會造成模型不穩定，從而使預測結果不如預期。因此，我決定使用多種參數訓練不同的模型，最後再用 Ensemble Learning 的方式得到預測結果。我最終的預測結果是整合五種模型所做出的預測，這五種模型的特徵分別為：

  1. Large 模型，img-size=640*640，batch-size=13，epochs=50，75% 資料用作訓練。獨立預測時 mAP=0.613。
  2. Large 模型，img-size=832*832，batch-size=7，epochs=55，99% 資料用作訓練，使用較低 learning rate。獨立預測時 mAP=0.599。
  3. Medium 模型，img-size=832*832，batch-size=13，epochs=50，99% 資料用作訓練。獨立預測時 mAP=0.598。
  4. 同 3，但加上更多手動標注的資料。獨立預測時 mAP=0.590。
  5. 同 1，但 epochs 改為 70、使用較低的 learning rate。獨立預測時 mAP=0.585。

  將上述 5 個模型作 Ensemble Learning，得到 mAP=0.675。我推測預測結果大幅進步的原因是，不同模型各有長處，例如 img-size=832*832 的 Medium 模型，較能準確預測小物件（如 bottle、bowl、plate），且 batch-size 大較具穩定性；而 img-size=640*640 的 Large 模型則較能準確預測形狀特殊的物件（如 plastic_bag）。

3. 競賽期間遇到的困難與解決方式

我認為我遇到最大的困難就是將 sticker 貼到適當的位置（亦即貼到沒有其他 bounding box 的位置），因為要寫出效率高的演算法需要一些思考，否則執行速度會很慢。最後我用線段樹來完成「尋找適當位置」的任務，花了不少時間。
此外，要將這個功能整合到 yolov5 裡面，勢必要修改 yolov5 的程式碼。所幸 yolov5 註解標示清楚，因此我才能成功地完成貼上 sticker 的功能。
最後我想小小抱怨的一點是，AIdea 提供的 train_images 分佈太過不平均，像是 aquarium、toilet、tub、washing_machine 的個數都不到 60 個，那麼少資料根本訓練不出來好的模型。而 train_images 也不包含「沒有任何物件的照片」，這會使 False Positive 的結果增加。

4. 競賽的心得

我花了不少時間在這個競賽上，不過我認為我花的這些時間都是值得的，也學到許多物件偵測的技巧與知識。在多次實驗與測試下，終於找到合適的訓練方案，得到不錯的成果使我得到成就感。

5. 程式碼

主要：

yolov5-pytorch.ipynb 所有訓練、預測會用到的程式碼
https://colab.research.google.com/drive/1Yv3Rsp5ThPrmCwZrVr3yLVXMVAjiI9w2?usp=sharing
statistics.ipynb 統計資料分佈
https://colab.research.google.com/drive/1ijLHgPaho7r8sCTVhPgJyKERvYC0K53q?usp=sharing
我 fork 原版 yolov5，增加 sticker 的功能
https://github.com/lashhw/yolov5

其他：

open_images_detection.ipynb 使用 Open Images 預測某些物件的程式碼
https://colab.research.google.com/drive/1my7slYv_usJgg4MbVFsY3WS-CkPyBPuc?usp=sharing
split_submission.ipynb 將 submission.csv 以 class 劃分，輸出每個 class 的預測結果，用以評估各個模型預測每個 class 的表現
https://colab.research.google.com/drive/1GgSlIZgjDESqDEI2DoHjh5Y5aH0bwEb9?usp=sharing