計算機視覺作業

干皓丞，2101212850, 信息工程學院

2021 年 11 月 26 日

1　作業目標與章節摘要

下載運行 YOLOv4 and YOLOv5 的 Python 程式碼，測試 5 幅圖，並於文檔中說明跟之前版本的具體改進和不同。

2　文章與作業狀況

作業可以從 GitHub 下的 kancheng/kan-cs-report-in-2021 專案找到，作業程式碼與文件目錄為 kan-cs-report-in-2021/CV/yolo-init。實際執行的環境與實驗設備為 Google 的 Colab 、MacBook Pro (Retina, 15-inch, Mid 2014) 、Acer Aspire R7 與 HP Victus (Nvidia GeForce RTX 3060)。

3　YOLO 歷史

最初由 Joseph Redmon 在 Darknet 中發表，其 YOLO 已經走過了漫長的道路。以下是使 YOLO 的首個版本在 R-CNN 和 DPM 的競爭的過程中脫穎而出的特色：

1. 45 fps 的實時幀處理
2. 減少背景誤報
3. 更高的檢測精度（雖然定位的準確度偏低）

自研究者在 2016 年首次發布以來，該演算法一直在不斷發展。YOLOv2 和 YOLOv3 均由 Joseph Redmon 編寫。在 YOLOv3 之後，出現了新的作者，他們將自己的目標錨定在其他 YOLO 版本中。YOLOv2 在 2017 年發布，由於錨框和分辨率的顯著改進，該版本在 CVPR 2017 上獲得了榮譽獎。YOLOv3 則在 2018 年版本對邊界框預測和與骨幹網絡層的連接有一個額外的客觀性分數。由於能夠在三個不同的粒度級別運行預測，它還提高了對微小對象的性能。而 YOLOv4 於 2020 年 4 月發布的論文成為第一篇非 Joseph Redmon 撰寫的論文。在這裡由 Alexey Bochkovski 介紹了新穎的改進，包括思維激活、改進的特徵聚合等。

最後則是最近期的 YOLOv5，則是由 Glenn Jocher 在 2020 年 6 月的版本中繼續進行進一步改進，其重點放在架構本身，整體用在 Pytorch 上，但他們的重心放在 YOLOv3 的移植跟 YOLOv5 的發展。

而 YOLOv4 和 YOLOv5 之間的性能差異則有 Joseph Nelson, Jacob Solawetz 兩位於 2020 年 1 月 10 號發布的技術文章中有詳細說明，該文章為'YOLOv5 is Here: State-of-the-Art Object Detection at 140 FPS'，而更詳細的測試則於兩天後 2020 年 1 月 12 號發佈名為'YOLOv5 is Here: State-of-the-Art Object Detection at 140 FPS'。詳細的作業閱讀紀錄連結可以於 init.md 中找到，而當中關鍵的原文摘要如下，同時進行原文翻譯。

"Running a Tesla P100, we saw inference times up to 0.007 seconds per image, meaning 140 frames per second (FPS)! By contrast, YOLO v4 achieved 50 FPS after having been converted to the same Ultralytics PyTorch library."

「在運行 Tesla P100，我們看到每張圖像的推理時間高達 0.007 秒，這意味著每秒 140 幀 (FPS)！相比之下，YOLOv4 在轉換為相同的 Ultralytics PyTorch 庫後達到了 50 FPS。」

"YOLO v5 is small. Specifically, a weights file for YOLO v5 is 27 megabytes. Our weights file for YOLO v4 (with Darknet architecture) is 244 megabytes. YOLO v5 is nearly 90 percent smaller than YOLO v4."

「YOLOv5 比較小。具體來說，YOLOv5 的權重文件為 27 兆字節。我們的 YOLOv4（具有暗網架構）的權重文件為 244 兆字節。YOLOv5 比 YOLOv4 小了近 90 % 。這意味著 YOLOv5 可以更輕鬆地部署到嵌入式設備。」

"YOLOv5 achieves 140 frames per second in batch, which the YOLOv5 implementation tested utilizes by default. When batch size is set to 1, YOLOv4 achieves 30 FPS while YOLOv5 outputs 10 FPS."

「YOLOv5 達到每秒 140 幀在批次，其 YOLOv5 執行默認測試利用。當批量大小設置為 1 時，YOLOv4 達到 30FPS，而 YOLOv5 輸出 10 FPS。」

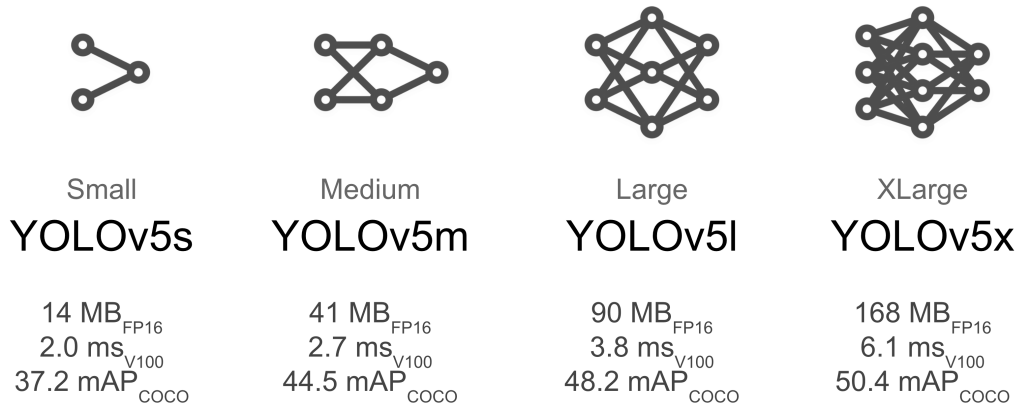


Fig. 1. YOLOv5 模型說明

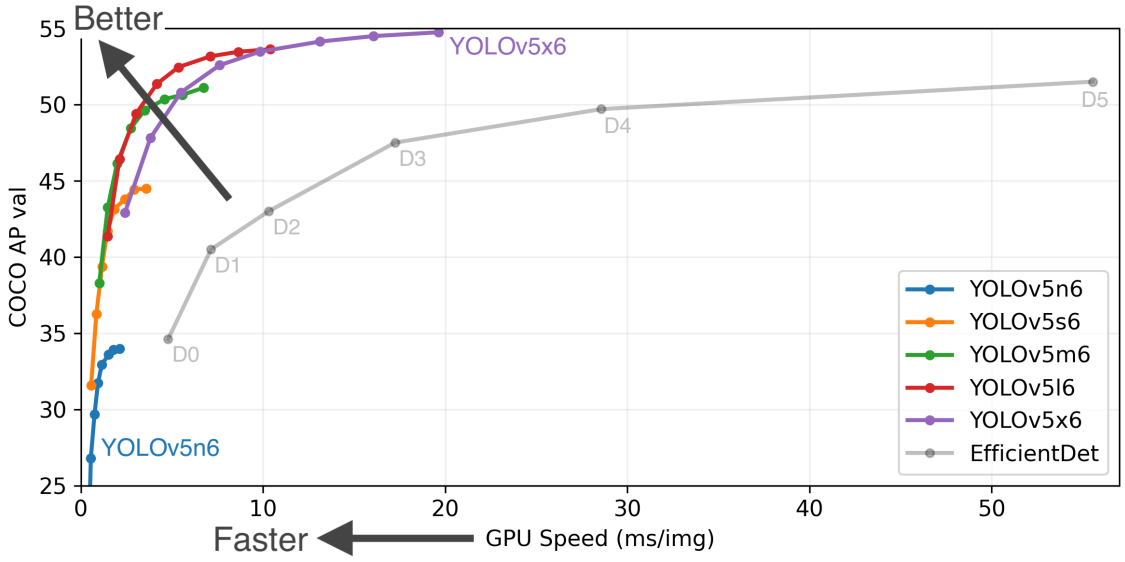


Fig. 2. 效能指標

4 Experiment with YOLOv3 and YOLOv4 on TensorFlow

可以從專案中 code 目錄下，找到 tranleanh/darkeras-yolov4 專案的 YOLOv3 和 YOLOv4 的 TensorFlow 版本。其檔案為 darkeras-yolov3.ipynb 和 darkeras-yolov4.ipynb 。還原環境與模型後可以得到指定的結果。

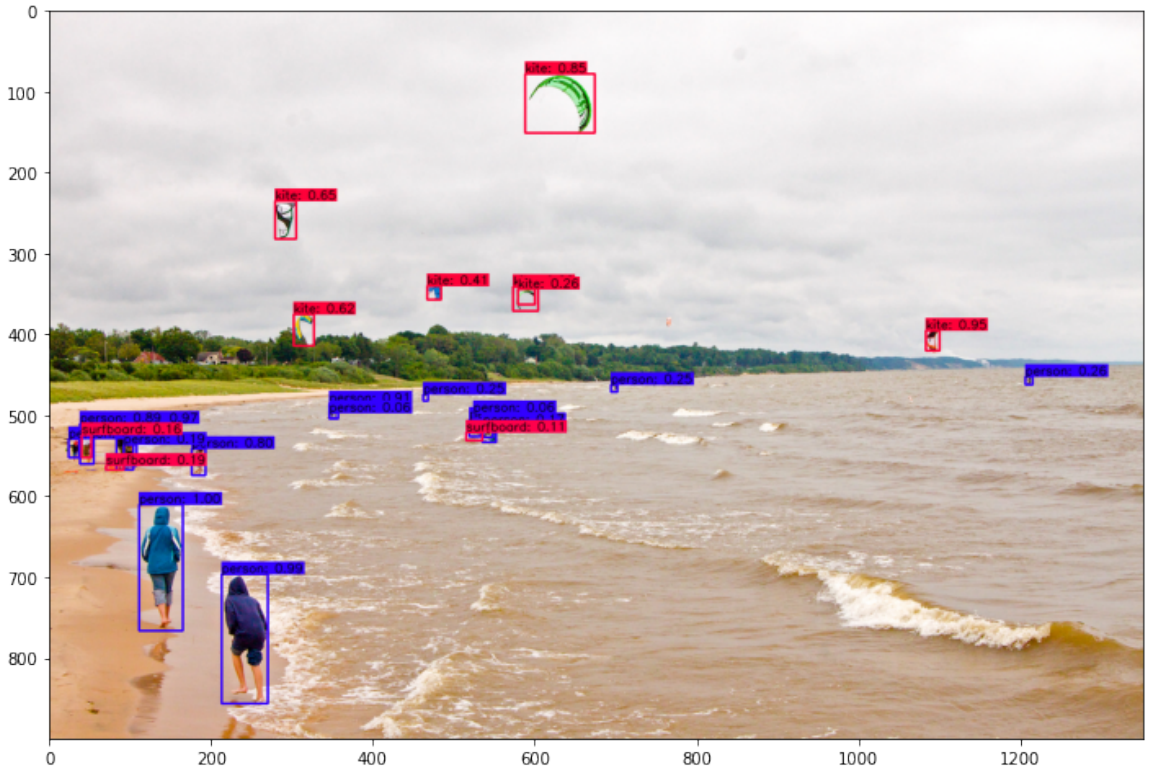


Fig. 3. YOLOv3 結果呈現

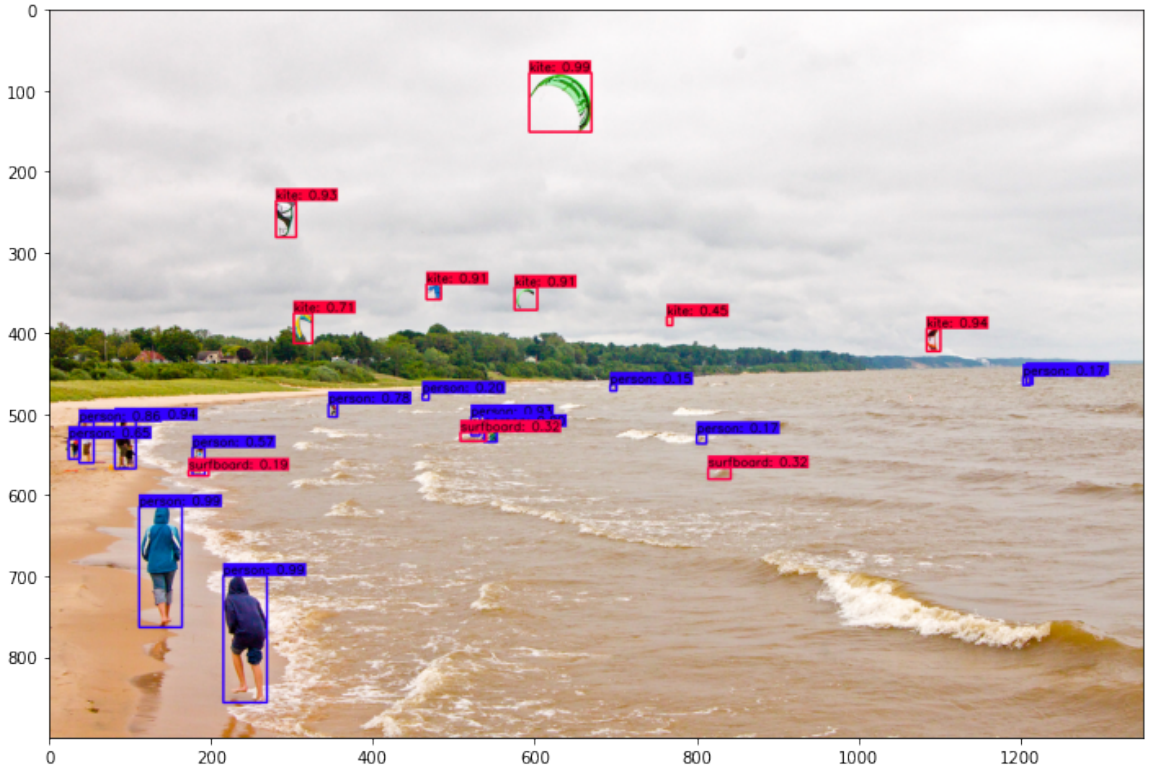


Fig. 4. YOLOv4 結果呈現

5 Experiment with YOLOv5 on Pytorch

YOLOv5 則分為兩大部分，其一為 Pytorch 官方的 Google 的 Colab 官方案例，另外為自己還原 ultralytics/yolov5 官方專案後，使用自己資料執行的結果，誠然官方也有給 Docker 的環境方案，但由於設備、環境跟時間因素，經研究後放棄。其指令說明如下:

```
# 建立虛擬環境
conda create -n test-volo python=3.8

# 進入虛擬環境
conda activate test-volo

# 官方專案
git clone https://github.com/ultralytics/yolov5.git

# Pytorch
conda install pytorch torchvision torchaudio cudatoolkit=11.3 -c pytorch

# 官方套件指定安裝
pip install -r requirements.txt

# 中國大陸地區網路環境建議使用
pip install -r requirements.txt -i https://pypi.tuna.tsinghua.edu.cn/simple

# 模型
python detect.py --weights yolov5s.pt

python detect.py --weights yolov5m.pt

python detect.py --weights yolov5s6.pt

python detect.py --weights yolov5m6.pt

# 測試
python detect.py --source data/images --weights yolov5s.pt --conf 0.25
```

其實際過程與截圖說明如下。

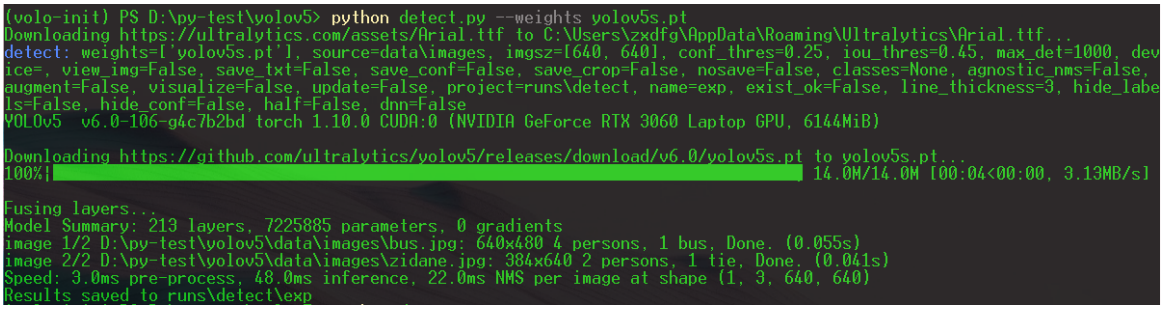


Fig. 5. Weights yolov5s.pt

```
(yolo-init) PS D:\py-test\yolov5> python detect.py --weights yolov5m.pt
detect: weights=['yolov5m.pt'], source=data\images, imgsz=[640, 640], conf_thres=0.25, iou_thres=0.45, max_det=1000, dev
ice=, view_img=False, save_txt=False, save_conf=False, save_crop=False, nosave=False, classes=None, agnostic_nms=False,
augment=False, visualize=False, update=False, project=runs\detect, name=exp, exist_ok=False, line_thickness=3, hide_labe
ls=False, hide_conf=False, half=False, dnn=False
YOLOv5  v6.0-106-g4c7b2bd torch 1.10.0 CUDA:0 (NVIDIA GeForce RTX 3060 Laptop GPU, 6144MiB)

Downloading https://github.com/ultralytics/yolov5/releases/download/v6.0/yolov5m.pt to yolov5m.pt...
100%|                                                                                | 40.7M/40.7M [00:07<00:00, 5.95MB/s]

Fusing layers...
Model Summary: 290 layers, 21172173 parameters, 0 gradients
image 1/2 D:\py-test\yolov5\data\images\bus.jpg: 640x480 4 persons, 1 bus, Done. (0.063s)
image 2/2 D:\py-test\yolov5\data\images\zidane.jpg: 384x640 3 persons, 2 ties, Done. (0.047s)
Speed: 0.0ms pre-process, 55.1ms inference, 0.0ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs\detect\exp3
(yolo-init) PS D:\py-test\yolov5> python detect.py --weights yolov5s6.pt
detect: weights=['yolov5s6.pt'], source=data\images, imgsz=[640, 640], conf_thres=0.25, iou_thres=0.45, max_det=1000, de
vice=, view_img=False, save_txt=False, save_conf=False, save_crop=False, nosave=False, classes=None, agnostic_nms=False,
 augment=False, visualize=False, update=False, project=runs\detect, name=exp, exist_ok=False, line_thickness=3, hide_lab
els=False, hide_conf=False, half=False, dnn=False
YOLOv5  v6.0-106-g4c7b2bd torch 1.10.0 CUDA:0 (NVIDIA GeForce RTX 3060 Laptop GPU, 6144MiB)

Downloading https://github.com/ultralytics/yolov5/releases/download/v6.0/yolov5s6.pt to yolov5s6.pt...
100%|                                                                                | 24.5M/24.5M [00:05<00:00, 5.04MB/s]

Fusing layers...
Model Summary: 280 layers, 12612508 parameters, 0 gradients
image 1/2 D:\py-test\yolov5\data\images\bus.jpg: 640x512 4 persons, 1 bus, Done. (0.047s)
image 2/2 D:\py-test\yolov5\data\images\zidane.jpg: 384x640 2 persons, 2 ties, Done. (0.032s)
Speed: 0.0ms pre-process, 39.3ms inference, 0.0ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs\detect\exp4
(yolo-init) PS D:\py-test\yolov5>
```

Fig. 6. Weights yolov5m.pt and yolov5s6.pt

```
(yolo-init) PS D:\py-test\yolov5> python detect.py --weights yolov5m6.pt
detect: weights=['yolov5m6.pt'], source=data\images, imgsz=[640, 640], conf_thres=0.25, iou_thres=0.45, max_det=1000, de
vice=, view_img=False, save_txt=False, save_conf=False, save_crop=False, nosave=False, classes=None, agnostic_nms=False,
 augment=False, visualize=False, update=False, project=runs\detect, name=exp, exist_ok=False, line_thickness=3, hide_lab
els=False, hide_conf=False, half=False, dnn=False
YOLOv5  v6.0-106-g4c7b2bd torch 1.10.0 CUDA:0 (NVIDIA GeForce RTX 3060 Laptop GPU, 6144MiB)

Downloading https://github.com/ultralytics/yolov5/releases/download/v6.0/yolov5m6.pt to yolov5m6.pt...
100%|                                                                                | 68.7M/68.7M [00:10<00:00, 6.97MB/s]

Fusing layers...
Model Summary: 378 layers, 35704908 parameters, 0 gradients
image 1/2 D:\py-test\yolov5\data\images\bus.jpg: 640x512 4 persons, 1 bus, Done. (0.079s)
image 2/2 D:\py-test\yolov5\data\images\zidane.jpg: 384x640 2 persons, 2 ties, Done. (0.016s)
Speed: 0.0ms pre-process, 47.0ms inference, 0.0ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs\detect\exp5
(yolo-init) PS D:\py-test\yolov5>
```

Fig. 7. Weights yolov5m6.pt

接下來進行第一輪測試的實驗從官方與自己相簿跟新聞的圖做為測試資料。第一輪與第二輪分別用的是 yolov5s.pt 與 yolov5m.pt，其過程截圖如下所示。

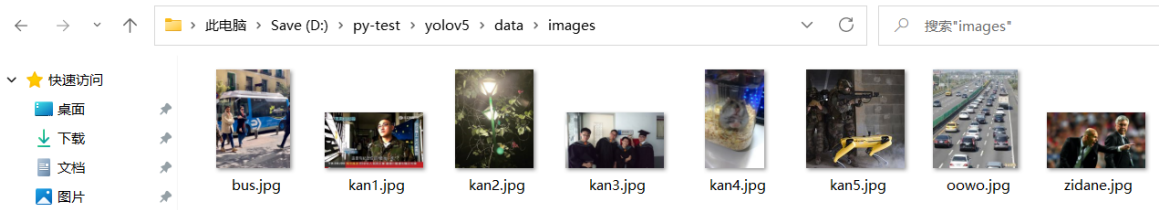


Fig. 8. 第一輪測試資料

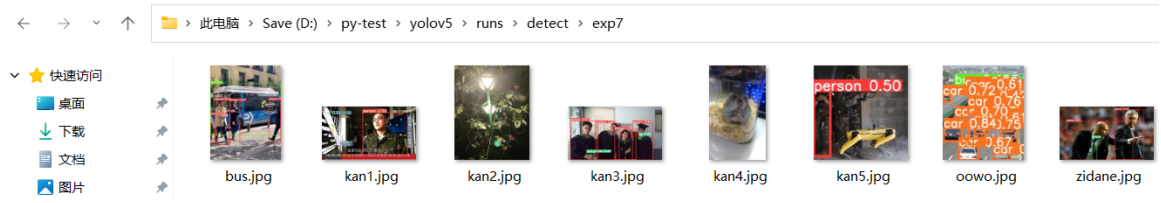


Fig. 9. 第一輪測試結果

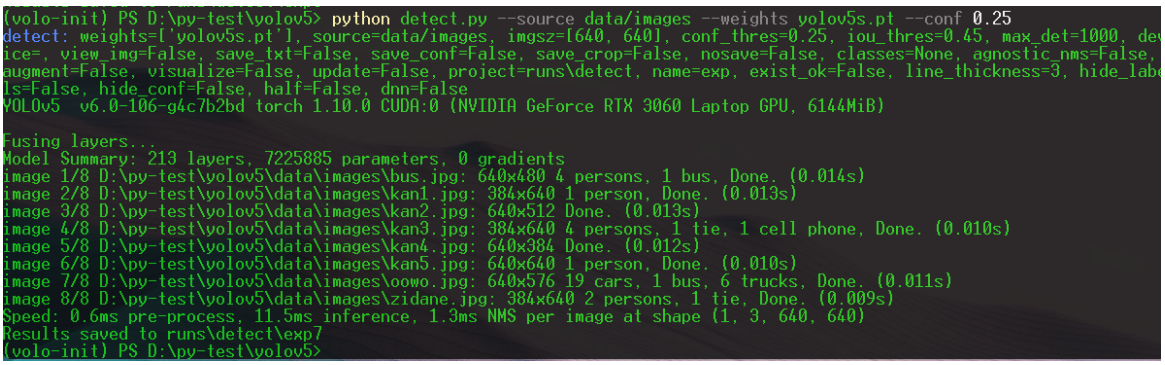


Fig. 10. 第一輪測試狀況

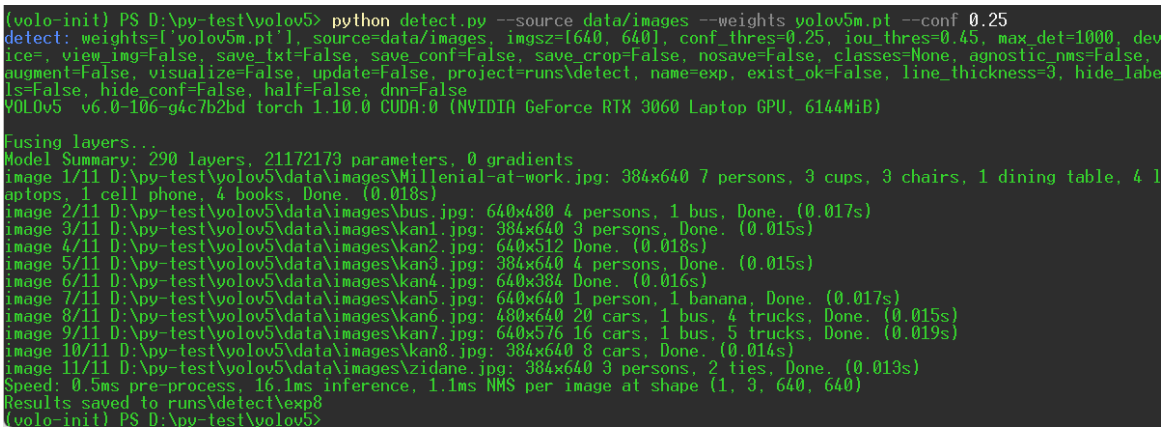


Fig. 11. 第二輪測試截圖

接下來列出原始資料進行說明，同時說明出處，過程中發現倉鼠照片跟北大深研院的路燈照片並沒有正確顯示。當中官方測試圖有三張，交通新聞有三張，人工智慧與機器人新聞一張，軍事新聞一張，生活照一張。

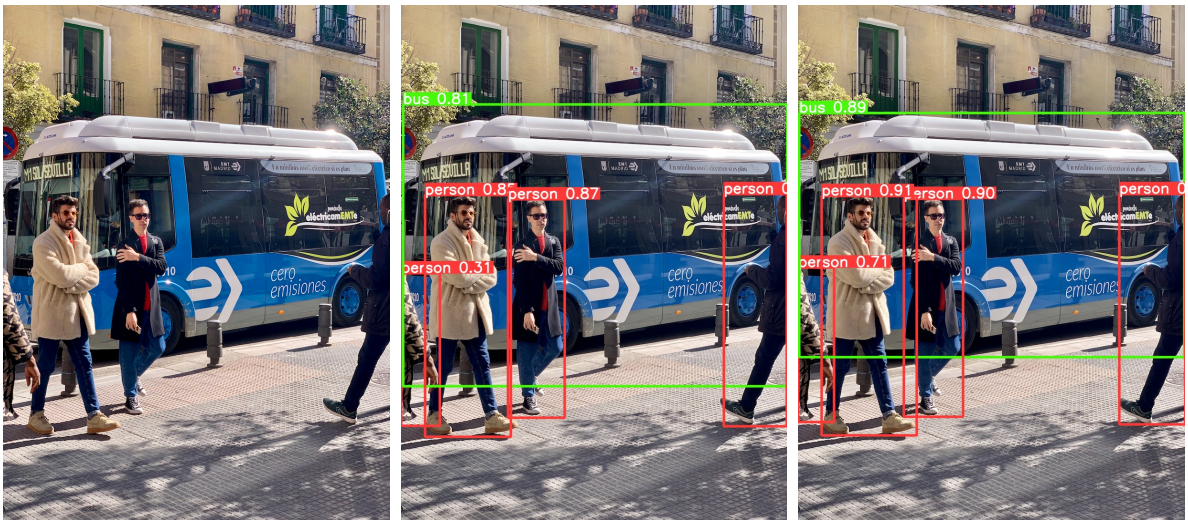
(i) 測試 - 官方　(ii) 測試 - 軍事新聞　(iii) 測試 - 軍事新聞

(iv) 測試 - 新聞　(v) 測試 - 自有照片　(vi) 測試 - 新聞

(vii) 測試 - 新聞　(viii) 測試 - 官方　(ix) 測試 - 官方

Fig. 12. 最終自有測試資料集

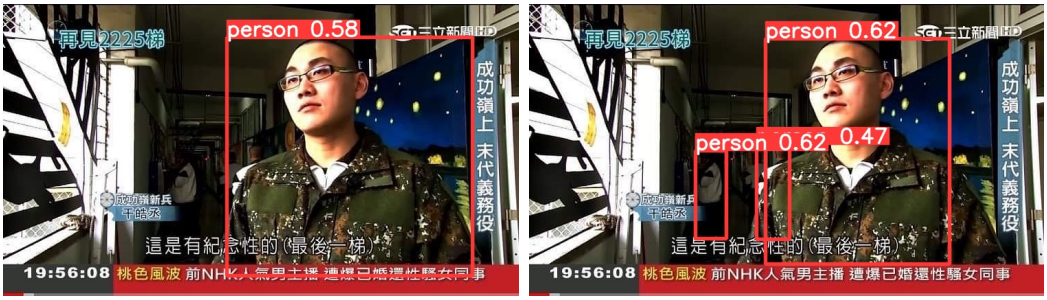
在此列出將第一輪測試的 yolov5s.pt 與 yolov5m.pt 的結果與原始資料進行比較。可以在過程中看到差異。



(i) 原始相片 (ii) yolov5s.pt (iii) yolov5m.pt

Fig. 13. 測試 1 - 官方

(i) 原始相片



(ii) yolov5s.pt (iii) yolov5m.pt

Fig. 14. 測試 2 - 新聞採訪

(i) 原始相片 (ii) yolov5s.pt (iii) yolov5m.pt

Fig. 15. 測試 3 - 自有照片

(i) 原始相片 (ii) yolov5s.pt (iii) yolov5m.pt

Fig. 16. 測試 4 - 軍事新聞

(i) 原始相片 (ii) yolov5m.pt

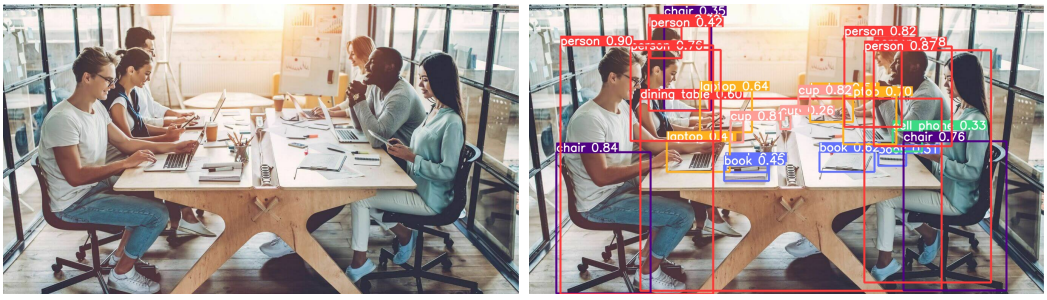
Fig. 17. 測試 5 - 新聞

(i) 原始相片 (ii) yolov5s.pt (iii) yolov5m.pt

Fig. 18. 測試 6 - 新聞

(i) 原始相片 (ii) yolov5m.pt

Fig. 19. 測試 7 - 新聞



(i) 原始相片 (ii) yolov5m.pt

Fig. 20. 測試 8 - 官方

(i) 原始相片 (ii) yolov5s.pt (iii) yolov5m.pt

Fig. 21. 測試 9 - 官方

此部分為 YOLOv5 的 Pytorch 官方 Google Colab 範例結果執行後，輸出的呈現。一共分為 ext1 與 ext2 的部分。

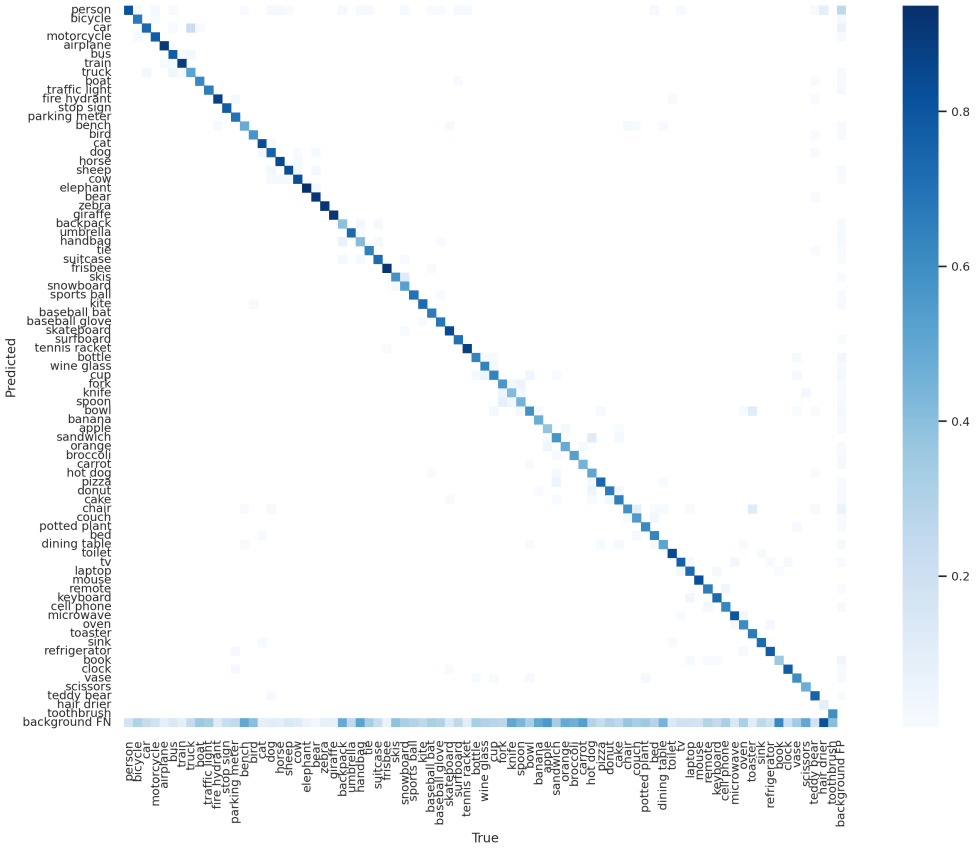


Fig. 22. ext1 - 1

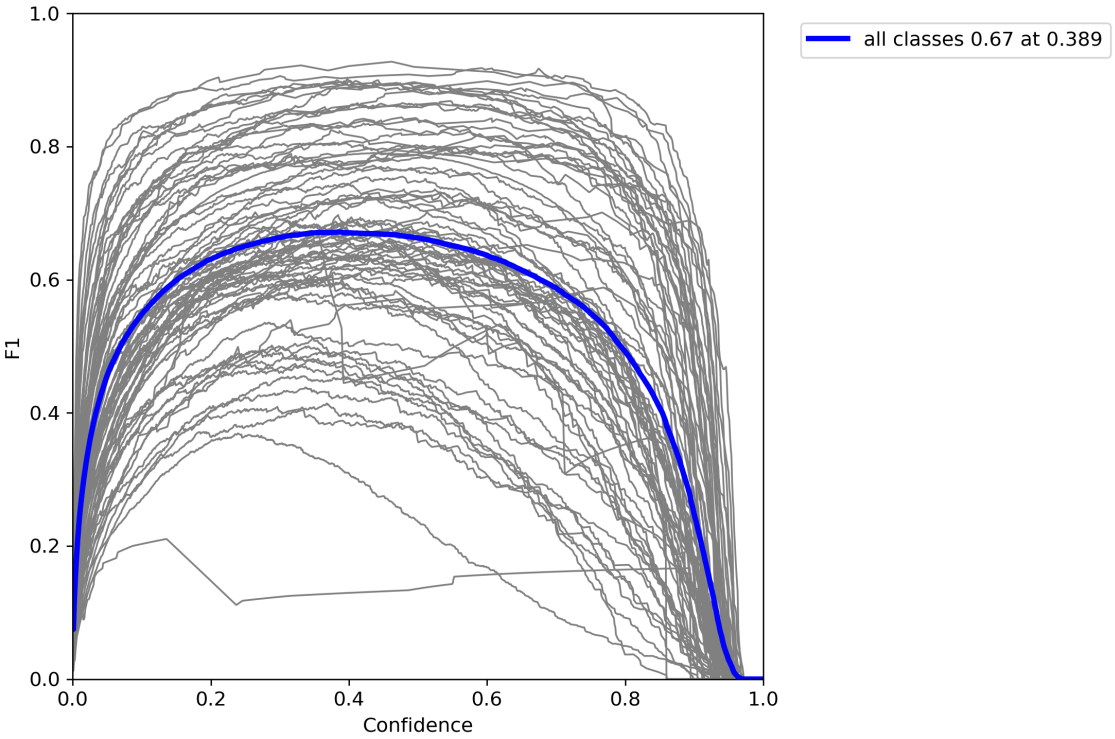


Fig. 23. ext1 - 2

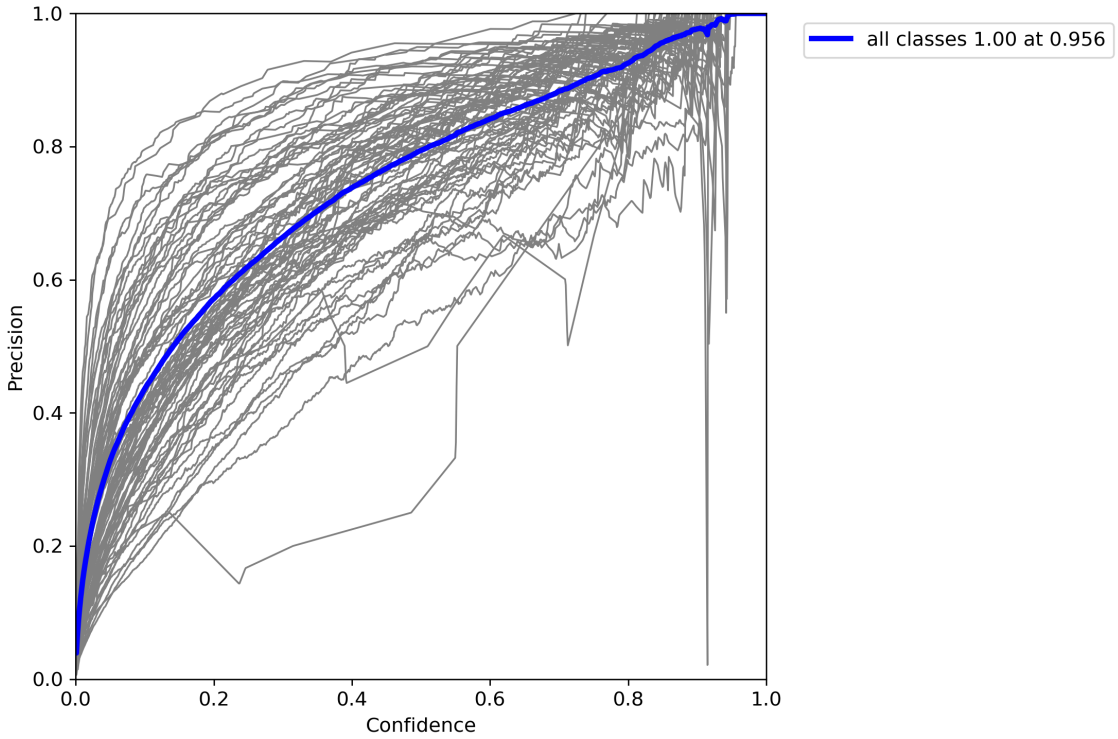


Fig. 24. ext1 - 3

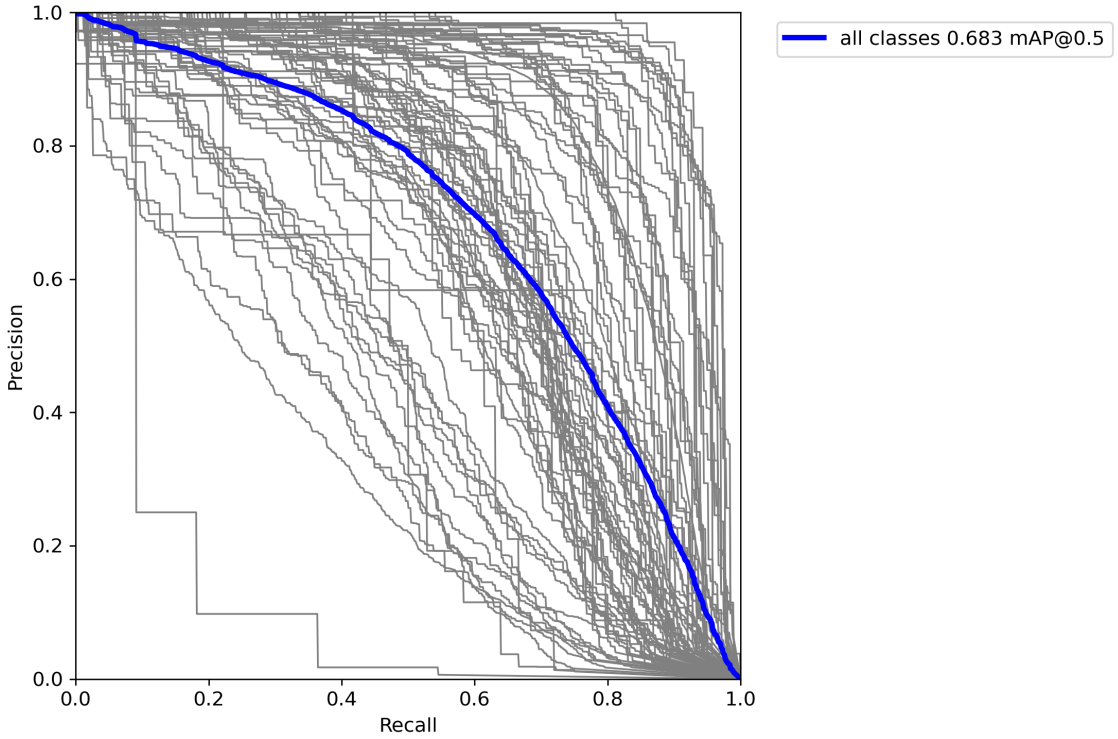


Fig. 25. ext1 - 4

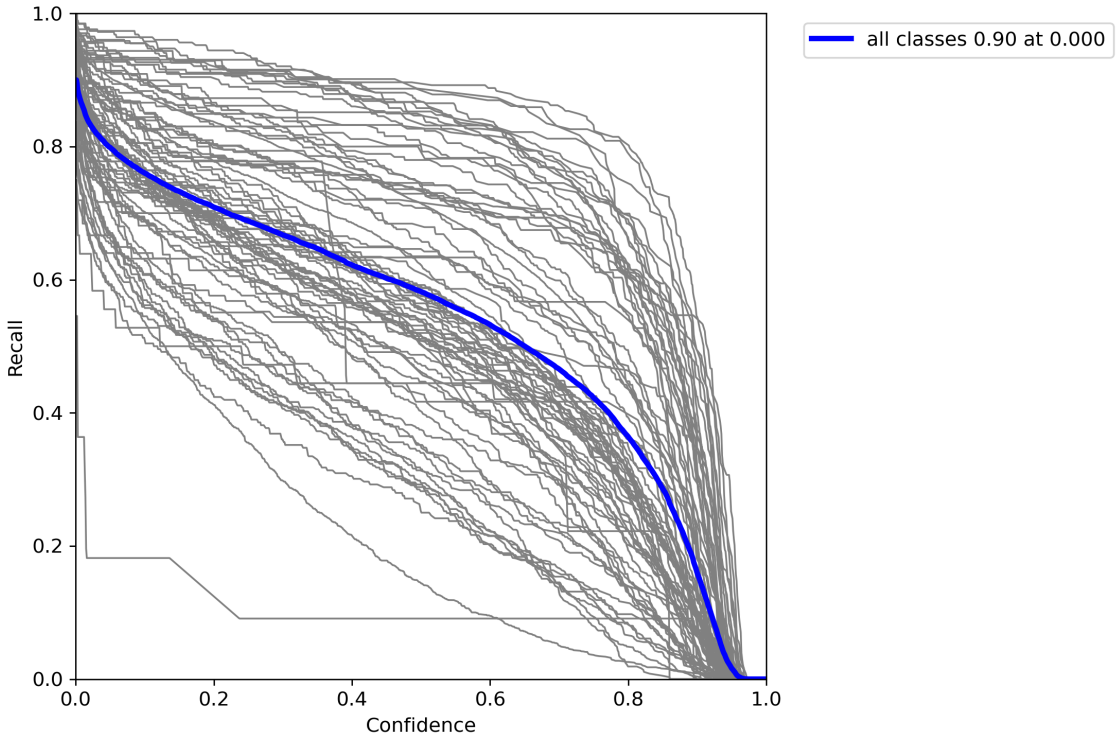


Fig. 26. ext1 - 5

Fig. 27. ext1 - 6

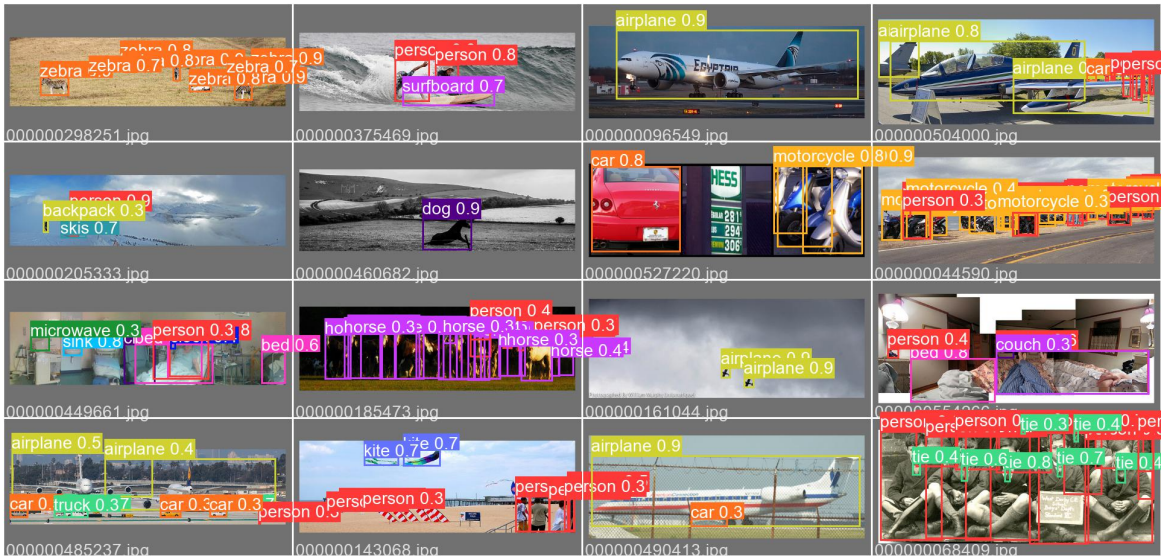


Fig. 28. ext1 - 7

Fig. 29. ext1 - 8

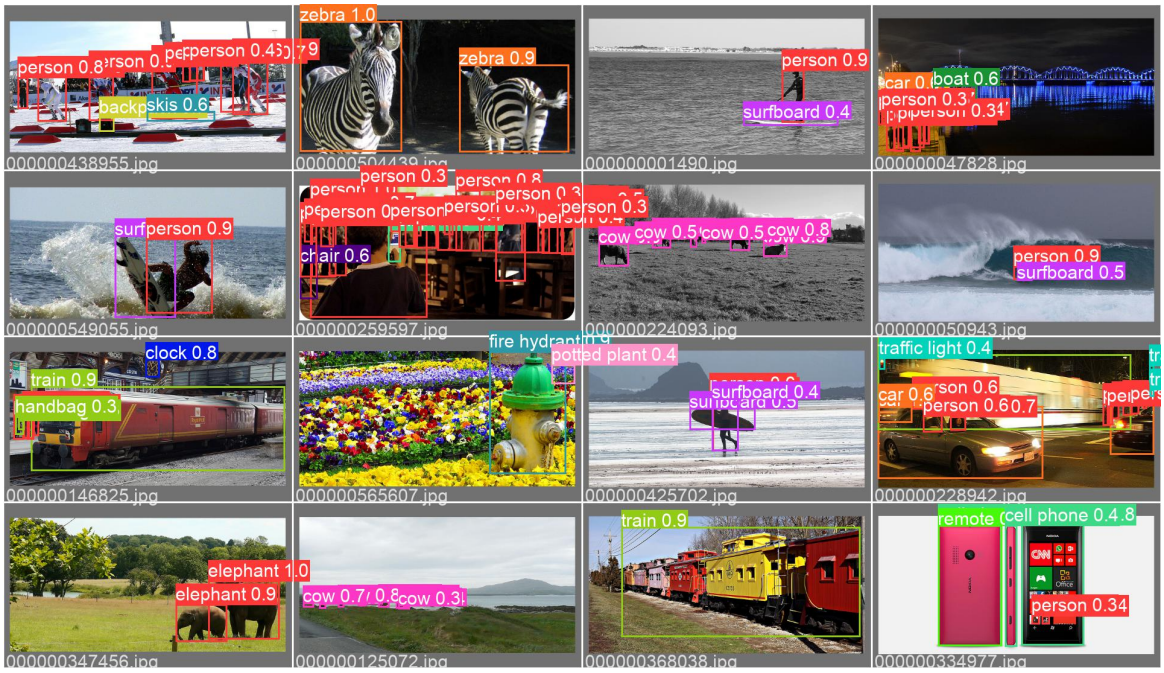


Fig. 30. ext1 - 9

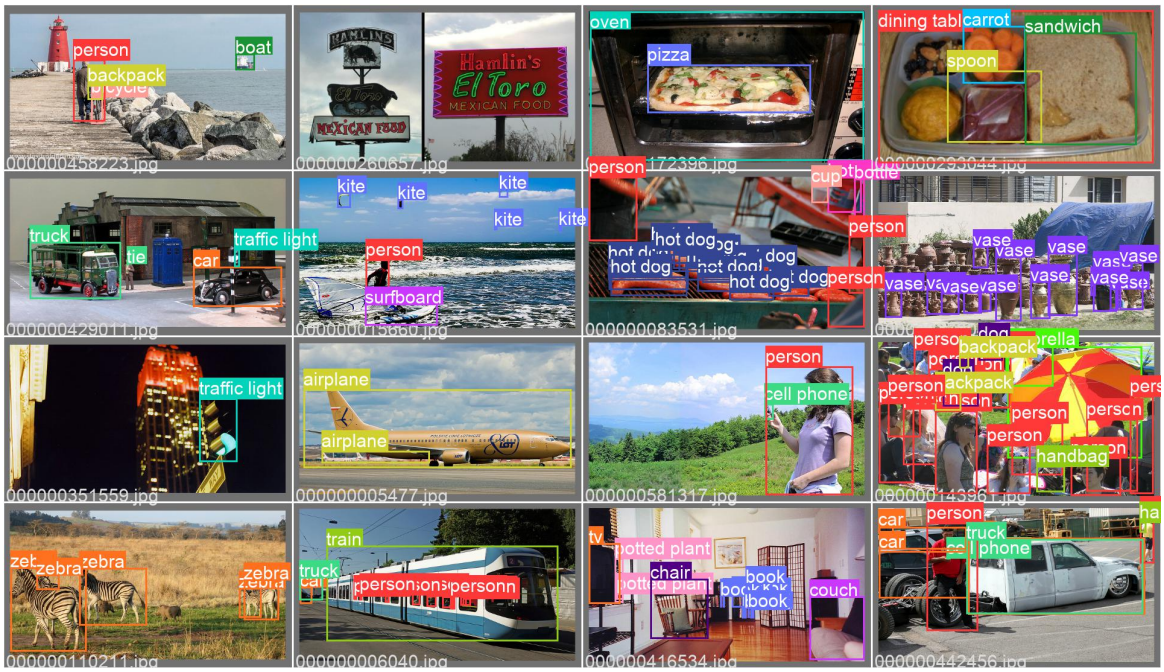


Fig. 31. ext1 - 9

Fig. 32. ext1 - 9

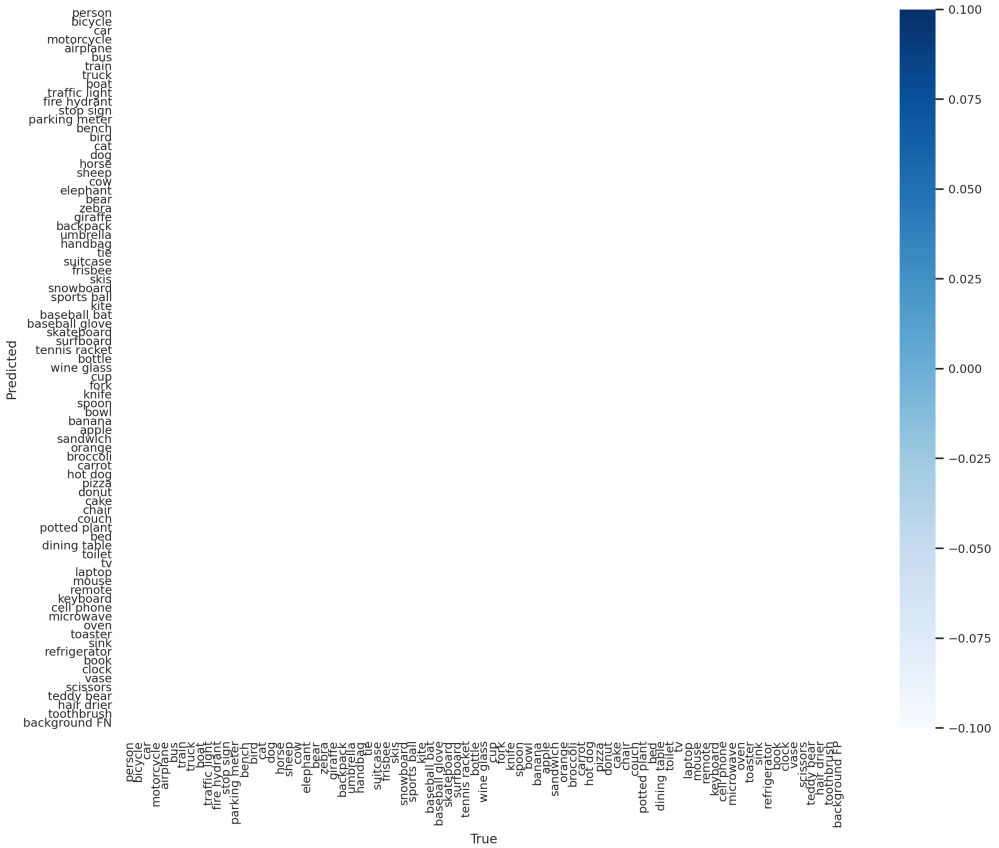


Fig. 33. ext2 - 1

Fig. 34. ext2 - 2

Fig. 35. ext2 - 3

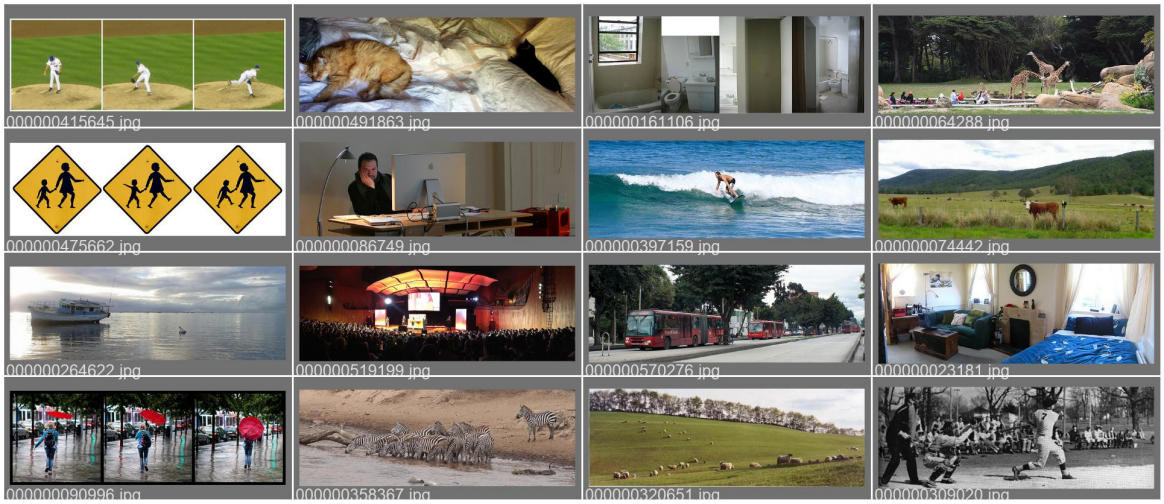


Fig. 36. ext2 - 4

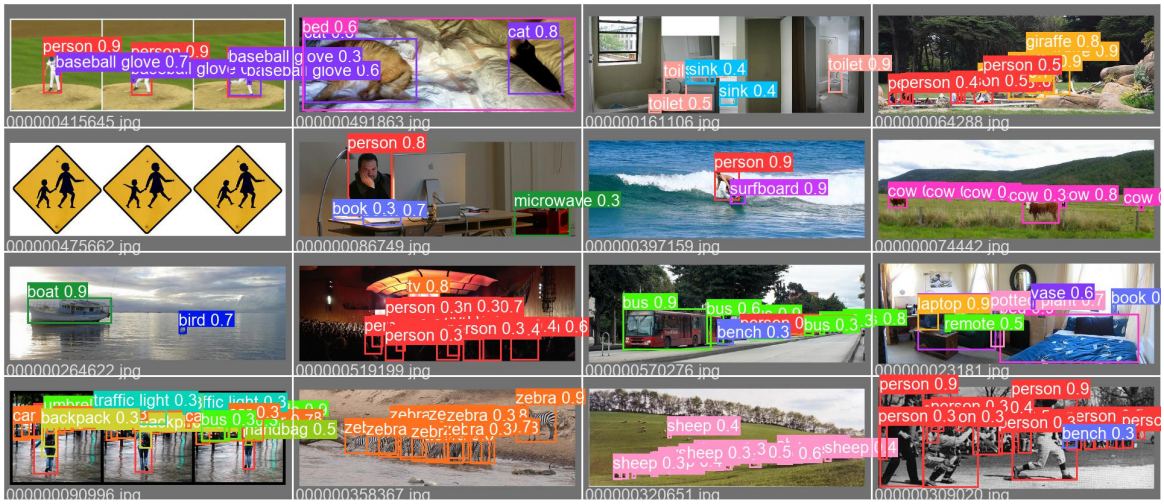


Fig. 37. ext2 - 5

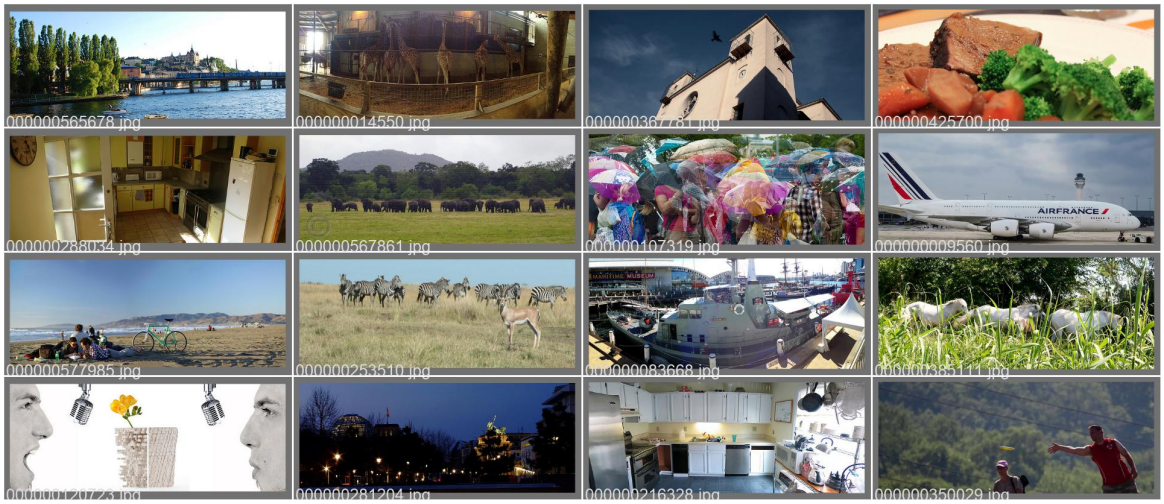


Fig. 38. ext2 - 6

Fig. 39. ext2 - 7