影像處理-期末專題-物件辨識

學號:313512072

姓名:洪亮

執行方式:

在終端機輸入 python3 hw.py

使用模型:

Yolov10x

程式碼:

```python
#!/usr/bin/python3
from ultralytics import YOLO
import cv2
import os

# 設定影片來源路徑和儲存目錄
source = 'new_input3/person_elephants_V3.mp4'
save_dir_p = './output_frames'  # 儲存影格的資料夾
save_dir_v = './output_video'  # 儲存輸出影片的資料夾
os.makedirs(save_dir_p, exist_ok=True)
os.makedirs(save_dir_v, exist_ok=True)

# 載入 YOLOv10 模型
model = YOLO('yolov10x.pt')

# 開啟輸入影片
cap = cv2.VideoCapture(source)

# 取得影片屬性
frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))  # 影片寬度
frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))  # 影片高度
fps = int(cap.get(cv2.CAP_PROP_FPS))  # 每秒影格數
fourcc = cv2.VideoWriter_fourcc(*'XVID')  # 指定影片的編碼格式
out = cv2.VideoWriter(os.path.join(save_dir_v, 'output_video.mp4'), fourcc, fps, (frame_width, frame_height))

# 進行人與大象的檢測
results_person = model.predict(source=source, classes=[0], imgsz=1440, conf=0.1, iou=0.5)  # 檢測人類
results_elephant = model.predict(source=source, classes=[20], imgsz=1280, conf=0.5, iou=0.85)  # 檢測大象

# 遍歷影片影格並合併檢測結果
for i, (result_person, result_elephant) in enumerate(zip(results_person, results_elephant)):
    # 取得原始影像
    orig_image = result_person.orig_img.copy()
```

```python
# 合併人和大象的檢測框
combined_boxes = []
combined_classes = []
combined_confs = []

if result_person.boxes is not None:
    combined_boxes.extend(result_person.boxes.xyxy.cpu().numpy())  # 人類檢測框
    combined_classes.extend(result_person.boxes.cls.cpu().numpy())  # 人類類別
    combined_confs.extend(result_person.boxes.conf.cpu().numpy())  # 人類置信度

if result_elephant.boxes is not None:
    combined_boxes.extend(result_elephant.boxes.xyxy.cpu().numpy())  # 大象檢測框
    combined_classes.extend(result_elephant.boxes.cls.cpu().numpy())  # 大象類別
    combined_confs.extend(result_elephant.boxes.conf.cpu().numpy())  # 大象置信度

# 繪製檢測框
for box, cls, conf in zip(combined_boxes, combined_classes, combined_confs):
    x1, y1, x2, y2 = map(int, box)  # 取得檢測框座標
    class_name = result_person.names[int(cls)]  # 取得類別名稱

    if class_name == "person":
        color = (0, 0, 255)  # 紅色代表人類
    elif class_name == "elephant":
        color = (0, 255, 0)  # 綠色代表大象
    else:
        color = (255, 255, 255)  # 白色代表其他

    # 繪製檢測框及標籤
    cv2.rectangle(orig_image, (x1, y1), (x2, y2), color, 4)
    label = f"{class_name} {conf:.2f}"  # 顯示類別與置信度
    cv2.putText(orig_image, label, (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 1, color, 3)
```

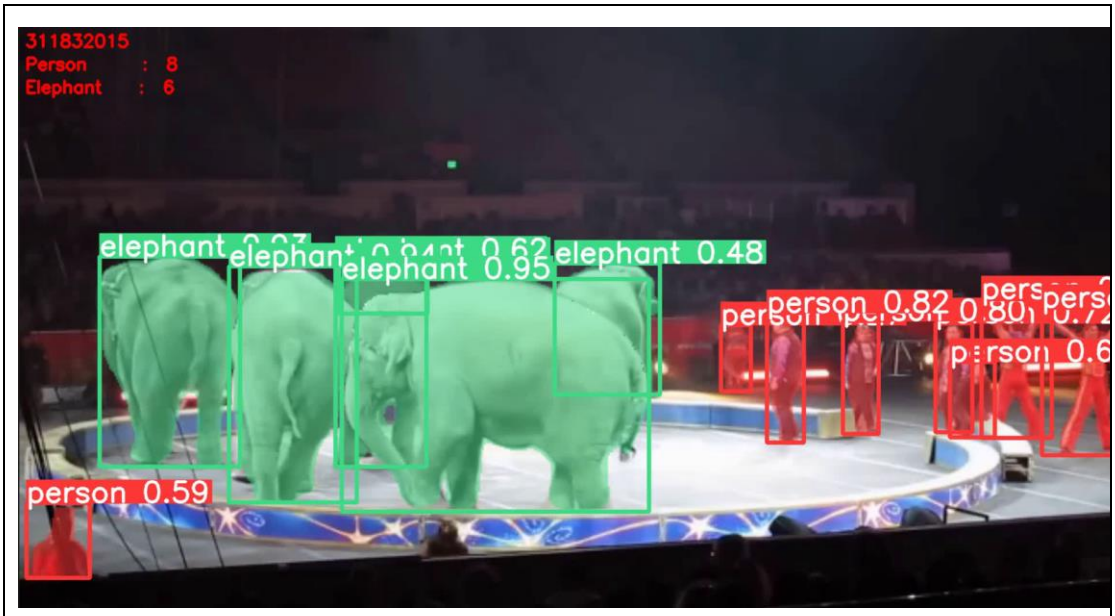
```python
    # 計算人類與大象的數量
    class_human_count = sum(1 for cls in combined_classes if cls == 0)
    class_elephant_count = sum(1 for cls in combined_classes if cls == 20)

    # 在影格上顯示學號與檢測數量
    red_color = (0, 0, 255)  # 紅色
    cv2.putText(orig_image, f"313512072", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, red_color, 2)
    cv2.putText(orig_image, f"person: {class_human_count}", (10, 70), cv2.FONT_HERSHEY_SIMPLEX, 1, red_color, 2)
    cv2.putText(orig_image, f"elephant: {class_elephant_count}", (10, 110), cv2.FONT_HERSHEY_SIMPLEX, 1, red_color, 2)

    # 儲存處理後的影格
    frame_file_path = os.path.join(save_dir_p, f'frame_{i}.jpg')
    cv2.imwrite(frame_file_path, orig_image)
    out.write(orig_image)

# 釋放資源
cap.release()
out.release()
cv2.destroyAllWindows()
```

結果討論:

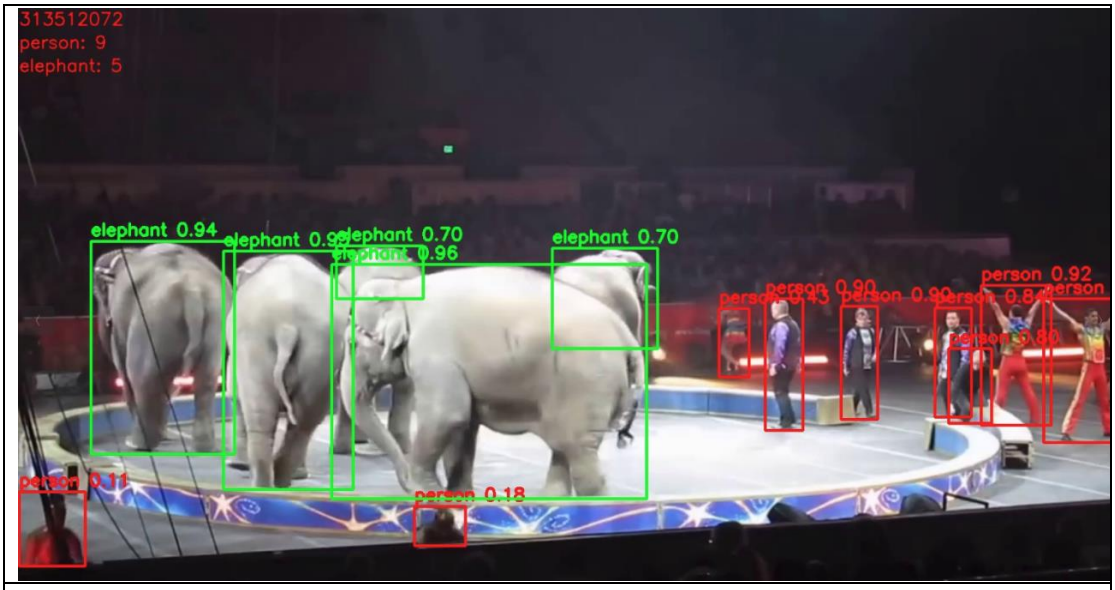
基本上我每幀都比 Baseline 好，以下我分不同情況進行介紹

第 0 幀:

從 baseline 可以看出，大象實際上明明只有 5 隻，但它卻誤判成 6 隻，另外人的部分，它也偵測不太到下排的觀眾。



Baseline

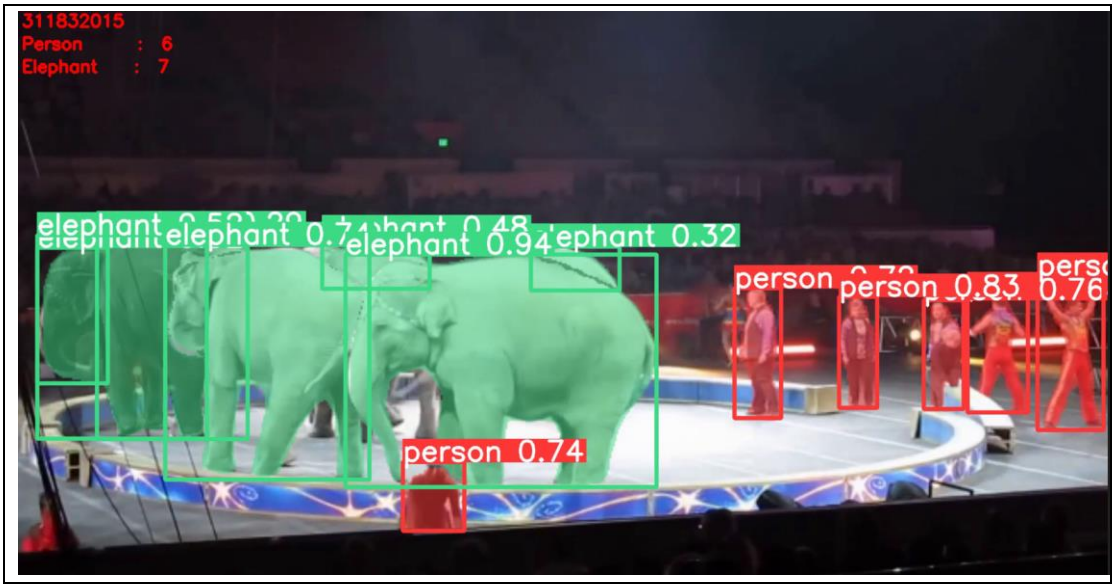


我的

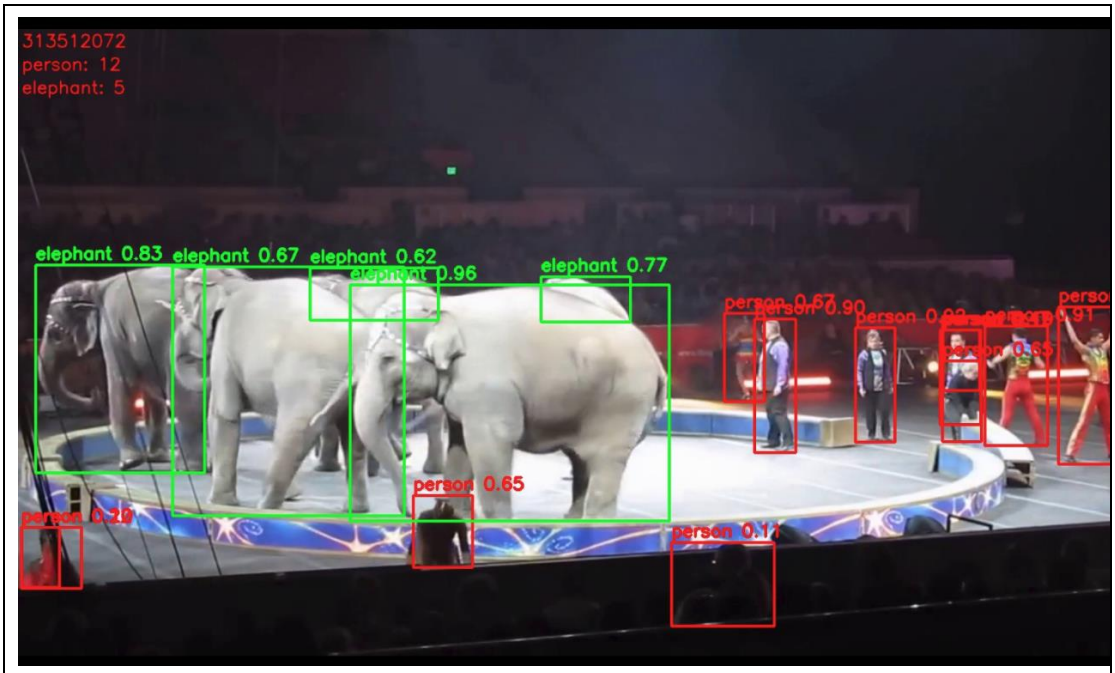
第 11 幀:

從 baseline 可以看出，大象實際上明明只有 5 隻，但它卻誤判成 7

隻，另外人的部分，我比它多偵測到左下和右下以及後排的人。



Baseline

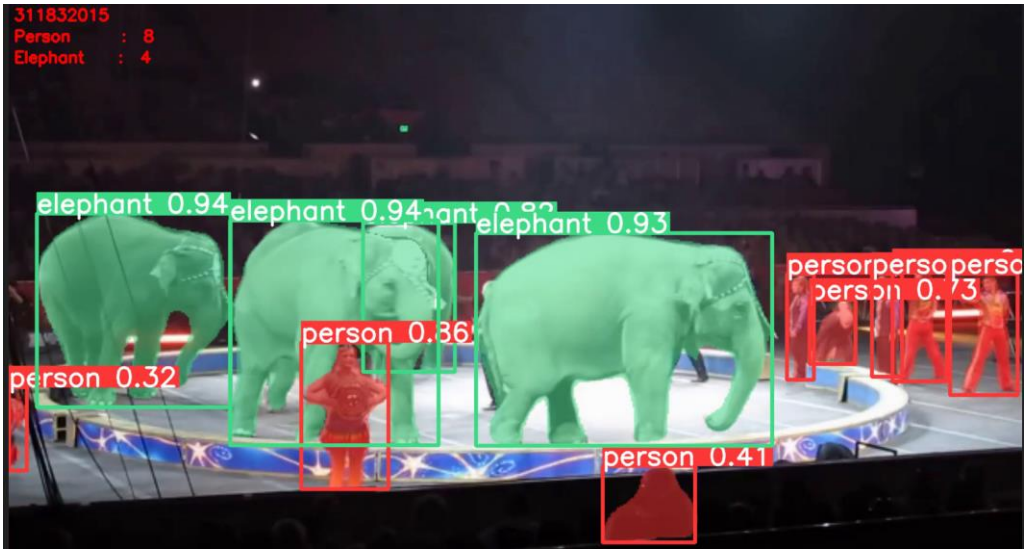


我的

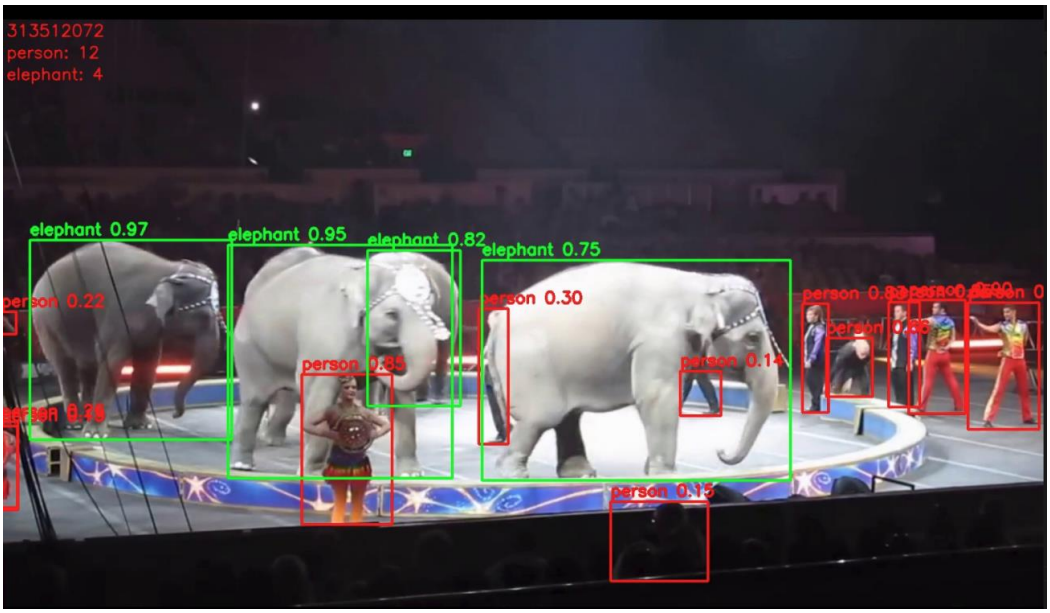
第 70 幀:

從 baseline 可以看出最右邊的大象前後都有人，但沒被偵測出來，

但我的有成功偵測



Baseline



我的

由以上結果可以知道，大象的部分 baseline 經常誤判大象的數量，人的部分也經常漏掉重要的部分。

調整 YOLO 參數的原因說明:

在進行人類和大象的檢測時，因為兩種類別差異極大，所以我分別調整兩種類別的參數:

程式部分:

```
# 進行人與大象的檢測
results_person = model.predict(source=source, classes=[0], imgsz=1440, conf=0.1, iou=0.5)  # 檢測人類
results_elephant = model.predict(source=source, classes=[20], imgsz=1280, conf=0.5, iou=0.85)  # 檢測大象
```

1. **人類檢測參數**

- classes=[0]：只檢測類別索引為 0 的物件，也就是 person，避免處理其他不相關的物件。
- imgsz=1440：選擇較大的影像大小，因為人類目標數量較多且物體細節重要，較大的影像輸入能提升精度。
- conf=0.1：設定較低的置信度閾值，允許檢測更多物件，因為人類數量較多，避免遺漏重要的檢測框。
- iou=0.5：設定較低的 IoU 閾值，允許更多重疊檢測框，因為此場景人群較密集，多數檢測框可能會出現重疊。

2. **大象檢測參數**

- classes=[20]：只檢測類別索引為 20 的物件，也就是 elephant，避免處理其他不相關的物件。

- imgsz=1280：選擇稍小的影像大小，因為大象通常在影像中占據較大的範圍，不需要更高解析度即可進行準確檢測。
- conf=0.5：設定較高的置信度閾值，因為場景中大象的數量只有 5 隻，且其特徵容易辨識。
- iou=0.85：設定較高的 IoU 閾值，用於嚴格篩選重疊框，確保每隻大象只有一個準確的檢測框，避免重複計算。

結論：

人類和大象的檢測需求，我使用不同的參數配置，主要在於人類數量較多，且分布較為密集，因此需要設定較低的置信度閾值和較大的影像尺寸，確保檢測的全面性與細節完整。而大象的數量肉眼可見只有 5 隻，且特徵明顯，因此採用較高的置信度閾值和 IoU 閾值，以提升檢測的準確性並減少重複檢測框。