



Universidad Autónoma de San Luis Potosí
Facultad de ingeniería
Inteligencia Artificial Aplicada
Practica 10
Aplicaciones de detección de objetos
Ana Sofía Medina Martínez



Fecha 16/11/2024

Objetivo

Que el alumno conozca las aplicaciones que se pueden implementar utilizando modelos de redes neuronales para detección de objetos.


Procedimiento

10.1.- Sigue las instrucciones del archivo “yolo_app.ipynb” para implementar seguimiento y conteo de objetos.

Resultados

```
[2] from ultralytics import YOLO
```

```
#Cargar el modelo pre-entrenado  
model = YOLO("yolo11m.pt")
```

Creating new Ultralytics Settings v0.0.6 file 
View Ultralytics Settings with 'yolo settings' or at '/root/.config/Ultralytics/settings.json'
Update Settings with 'yolo settings key=value', i.e. 'yolo settings runs_dir=path/to/dir'. For help see
Downloading <https://github.com/ultralytics/assets/releases/download/v8.3.0/yolo11m.pt> to 'yolo11m.pt'...
100%|██████████| 38.8M/38.8M [00:00<00:00, 122MB/s]

El modelo YOLO11 ya es capaz de detectar maletas, ya que esta clase está incluida en el conjunto de datos COCO, por práctica a la detección de maletas en un video.

```
from google.colab import files
```

```
uploaded = files.upload() #Selecciona el archivo de video (Solo 1)
```

Choose Files video1.mp4
• **video1.mp4**(video/mp4) - 53272415 bytes, last modified: 11/14/2024 - 100% done
Saving video1.mp4 to video1.mp4

```
[4] video_path = list(uploaded.keys())[0]
```

```
[5] suitcase_class = 28 #Clase de la maleta en el dataset de COCO
```

```
[6] import cv2

cap = cv2.VideoCapture(video_path)

assert cap.isOpened(), f"Failed to open {video_path}"

width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
fps = cap.get(cv2.CAP_PROP_FPS)

#Creamos un video de salida para guardar el video procesado
output_path = "output.mp4"

fourcc = cv2.VideoWriter_fourcc(*'mp4v')
writer = cv2.VideoWriter(output_path, fourcc, fps, (width, height))

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    #Procesar el frame
    results = model.predict(frame, classes=[suitcase_class]) #Solo detectar maletas

    #Dibujar las detecciones
    annotated_frame = results[0].plot()

    writer.write(annotated_frame)
    cv2.waitKey(1)

cap.release()
writer.release()
```



```
0: 352x640 2 suitcases, 24.1ms
Speed: 3.4ms preprocess, 24.1ms inference, 2.0ms postprocess per image at shape (1, 3, 35

0: 352x640 2 suitcases, 24.1ms
Speed: 4.3ms preprocess, 24.1ms inference, 2.9ms postprocess per image at shape (1, 3, 35

0: 352x640 2 suitcases, 24.0ms
Speed: 3.9ms preprocess, 24.0ms inference, 2.0ms postprocess per image at shape (1, 3, 35
```

```

cap = cv2.VideoCapture(video_path)

assert cap.isOpened(), f"Failed to open {video_path}"

width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
fps = cap.get(cv2.CAP_PROP_FPS)

#Creamos un video de salida para guardar el video procesado
output_path = "output_track.mp4"

fourcc = cv2.VideoWriter_fourcc(*'mp4v')
writer = cv2.VideoWriter(output_path, fourcc, fps, (width, height))

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    #Procesar el frame
    results = model.track(
        frame,
        persist=True, #Seguir las detecciones entre frames
        classes=[suitcase_class], #Solo detectar maletas
    )

    #Dibujar las detecciones
    annotated_frame = results[0].plot()

    writer.write(annotated_frame)
    cv2.waitKey(1)

cap.release()
writer.release()

```

```

0: 352x640 2 suitcases, 24.1ms
Speed: 3.4ms preprocess, 24.1ms inference, 1.9ms postprocess per image at shape (1, 3, 352, 640)

0: 352x640 2 suitcases, 36.2ms
Speed: 3.6ms preprocess, 36.2ms inference, 2.1ms postprocess per image at shape (1, 3, 352, 640)

0: 352x640 2 suitcases, 39.4ms

```

```
[8] print(f"Ancho: {width}, Alto: {height}")
```

↗ Ancho: 4096, Alto: 2160

Colocaremos una línea en la parte derecha del video, 800 píxeles antes del borde derecho, para contar las maletas que pasan por esa línea.

Los valores escogidos para la línea son arbitrarios, puedes cambiarlos según tus necesidades.

Vamos a comprobar la posición de la línea en el video.

```
[9] from matplotlib import pyplot as plt

line_pt1 = (width - 200, 100)
line_pt2 = (width - 200, height - 100)

video = cv2.VideoCapture(video_path)

assert video.isOpened(), f"Failed to open {video_path}"

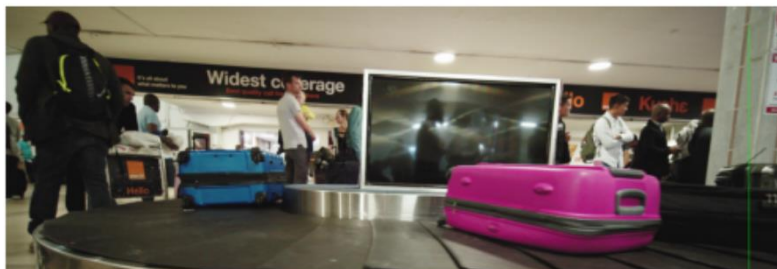
ret, frame = video.read()

cv2.line(frame, line_pt1, line_pt2, (0, 255, 0), 2)

plt.imshow(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
plt.axis("off")
plt.show()

cap.release()
```

↗



```

[10] from ultralytics import solutions

counter = solutions.ObjectCounter( #Objeto para contar objetos
    classes=[suitcase_class], #Clase a contar, puede ser una lista de clases
    region=[line_pt1, line_pt2], #Region de interes en este caso dos puntos que forman una
    show=True, #Mostrar el video en tiempo real
)

cap = cv2.VideoCapture(video_path)

assert cap.isOpened(), f"Failed to open {video_path}"

width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
fps = cap.get(cv2.CAP_PROP_FPS)

#Creamos un video de salida para guardar el video procesado
output_path = "output_counter.mp4"

fourcc = cv2.VideoWriter_fourcc(*'mp4v')
writer = cv2.VideoWriter(output_path, fourcc, fps, (width, height))

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

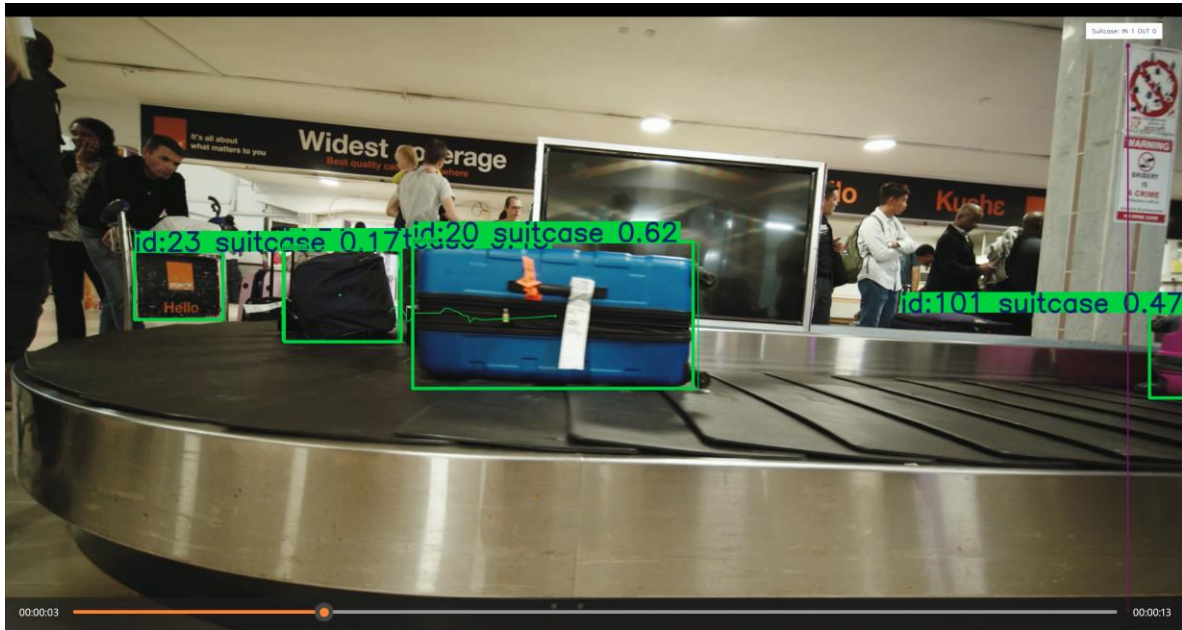
    #Procesar el frame
    results = model.track(
        frame,
        persist=True, #Seguir las detecciones entre frames
        classes=[suitcase_class], #Solo detectar maletas
    )

    annotated_frame = results[0].plot()

    #Contar las maletas
    annotated_frame = counter.count(annotated_frame)

    writer.write(annotated_frame)
    cv2.waitKey(1)

```



Realiza el conteo de maletas en el segundo video proporcionado.



```
[11] from google.colab import files
```

```
    uploaded = files.upload() #Selecciona el archivo de video (Solo 1)
```



Choose Files video2.mp4

- **video2.mp4**(video/mp4) - 29499193 bytes, last modified: 11/14/2024 - 100% done
Saving video2.mp4 to video2.mp4

```
[12] video_path = list(uploaded.keys())[0]
```

```
[13] suitcase_class = 28 #Clase de la maleta en el dataset de COCO
```

```
[18] import cv2
```

```
    cap = cv2.VideoCapture(video_path)
```

```
    assert cap.isOpened(), f"Failed to open {video_path}"
```

```
    width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
```

```
    height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
```

```
    fps = cap.get(cv2.CAP_PROP_FPS)
```

```
    #Creamos un video de salida para guardar el video procesado
```

```
    output_path = "output.mp4"
```

```
    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
```

```
    writer = cv2.VideoWriter(output_path, fourcc, fps, (width, height))
```

```
    prev_frame=None
```

```
    while cap.isOpened():
```

```
        ret, frame = cap.read()
```

```
        if not ret:
```

```
            break
```

```
        #Validar que el fotograma no es NONE y tiene el mismo tamaño
```

```
        if prev_frame is not None and (prev_frame.shape != frame.shape):
```



```
if prev_frame is not None and (prev_frame.shape != frame.shape):
    #Procesar el frame
    print("Error")
    prev_frame=frame
    continue

#Procesar el frame
results = model.predict(frame, classes=[suitcase_class]) #Solo detectar maletas

#Dibujar las detecciones
annotated_frame = results[0].plot()

writer.write(annotated_frame)
cv2.waitKey(1)

cap.release()
writer.release()
```

```
-----
error                                Traceback (most recent call last)
<ipython-input-18-8e7ad60da82b> in <cell line: 19>()
    30
    31     #Procesar el frame
--> 32     results = model.predict(frame, classes=[suitcase_class]) #Solo detectar maletas
    33
    34     #Dibujar las detecciones

----- 8 frames -----
/usr/local/lib/python3.10/dist-packages/ultralytics/trackers/utils/gmc.py in
applySparseOptFlow(self, raw_frame)
    341
    342     # Find correspondences
--> 343     matchedKeypoints, status, _ = cv2.calcOpticalFlowPyrLK(self.prevFrame,
frame, self.prevKeyPoints, None)
    344
    345     # Leave good correspondences only

error: OpenCV(4.10.0) /io/opencv/modules/video/src/lkpyramid.cpp:1394: error:
(-215:Assertion failed) prevPyr[level * lvlStep1].size() == nextPyr[level * lvlStep2].size()
in function 'calc'
```

Comprensión

1. ¿Qué es Ultralytics Solutions?

Es una empresa de tecnología en el área inteligencia artificial y visión por computadora. Es conocida por desarrollar herramientas como YOLOv5 y YOLOv8, que se utilizan para tareas de detección de objetos, segmentación de imágenes y clasificación.

2. ¿Qué aplicaciones de detección de objetos se pueden desarrollar utilizando el módulo de Ultralytics Solutions?

Se pueden desarrollar aplicaciones para la seguridad (vigilancia y detección de intrusos), control del tráfico (detección de vehículos y peatones), agricultura (monitoreo de cultivos), manufactura (inspección de calidad) y salud (detección de anomalías en imágenes médicas).

Conclusiones

Ultralytics Solutions es una empresa clave en el ámbito de la inteligencia artificial aplicada a la visión por computadora, ofreciendo herramientas avanzadas como YOLO que facilitan el desarrollo de aplicaciones innovadoras en diversos sectores. Gracias a su tecnología, es posible abordar problemas complejos, optimizar procesos y mejorar la eficiencia en áreas como seguridad, salud, agricultura y manufactura, demostrando el impacto transformador de la detección de objetos en la vida cotidiana y en la industria.