

# **IES Pere Maria Orts**

## **Modelos de Inteligencia Artificial**

---

### **Actividad 2\_1: Smart City en Benidorm**

---

**Autor:**

Kenny Berrones

**Profesor:**

David Campoy Miñarro



**iesperemariaorts**



**GENERALITAT  
VALENCIANA**

# **Índice**

<b>1. Introducción</b>	<b>2</b>
<b>2. Experimentos</b>	<b>2</b>
<b>3. Conclusiones</b>	<b>5</b>

# 1. Introducción

En esta práctica vamos a emplear el algoritmo YOLO para detectar objetos en una imagen. YOLO, que significa You Only Look Once, es un modelo de visión por computadora que se utiliza para detectar objetos en imágenes o videos en tiempo real. A diferencia de otros enfoques que analizan las imágenes en varias etapas, YOLO lo hace todo en un solo paso, de ahí su nombre. Esto lo hace muy rápido y eficiente. Básicamente, divide la imagen en una cuadrícula y predice qué objetos hay y dónde están, todo al mismo tiempo. Es como si YOLO tuviera una "visión completa" de la imagen de una sola vez, lo que lo hace ideal para aplicaciones donde la velocidad es crucial, como la seguridad o los autos autónomos.

## 2. Experimentos

En primer lugar tendremos que instalar YOLO, para ello seguiremos estos pasos:

```
!git clone https://github.com/ultralytics/yolov5  
%cd yolov5  
%pip install -qr requirements.txt comet_ml
```

Figura 1: Pasos Instalación YoloV5

Luego tendremos que comprobar que tenemos la cámara disponible, para ello emplearemos el siguiente código:

```
for i in range(10):  
    cap = cv2.VideoCapture(i)  
    if not cap.isOpened():  
        break  
    cap.release()  
    print(f'Camera ID {i} is available')
```

Una vez que sepamos que cámara tenemos disponibles podemos lanzar ya el detector de objetos con el siguiente comando:

```
!python ./detect.py --source 0
```

### 2.1. Reconocer imágenes en tiempo real

Cuando ejecutamos el comando anterior se ejecuta el detector de objetos, obtenemos el resultado siguiente:

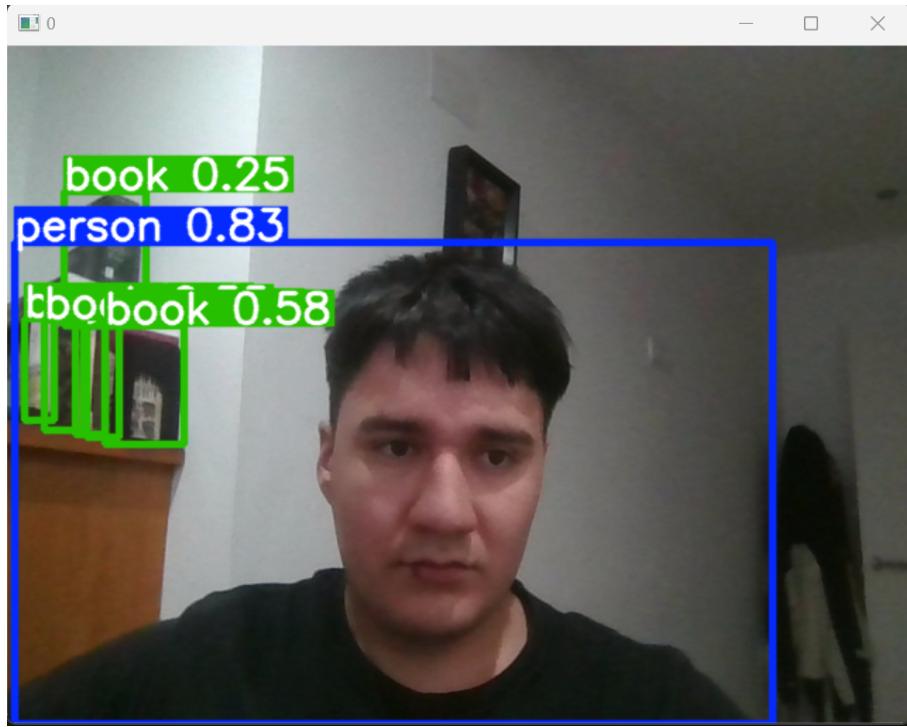


Figura 2: Resultado del detector en tiempo real

Como se puede apreciar en la imagen anterior vemos que me detecta correctamente, además de los libros que tengo detrás de mi. Además, en la terminal nos aparece información relevante como el número de objetos que reconoce entre otros:

```
Fusing layers...
YOLOv5s summary: 213 layers, 7225885 parameters, 0 gradients, 16.4 GFLOPs
1/1: 0... Success (inf frames 640x480 at 30.00 FPS)

0: 480x640 1 person, 5 books, 138.7ms
0: 480x640 1 person, 1 knife, 5 books, 158.7ms
0: 480x640 1 person, 1 cell phone, 5 books, 116.6ms
0: 480x640 1 person, 5 books, 88.0ms
0: 480x640 1 person, 6 books, 85.9ms
0: 480x640 1 person, 5 books, 94.4ms
0: 480x640 1 person, 6 books, 86.4ms
0: 480x640 1 person, 1 knife, 5 books, 97.8ms
0: 480x640 1 person, 6 books, 92.1ms
```

Figura 3: Información sobre el detector en tiempo real

## 2.2. Reconocer imágenes estáticas

Ahora vamos a probar a reconocer imágenes estáticas, para ello pondremos una imagen de nuestra elección dentro del directorio de Yolo, y lanzaremos el siguiente comando:

```
!python ./detect.py --source familia.jpg
```

Obtendremos el siguiente resultado:

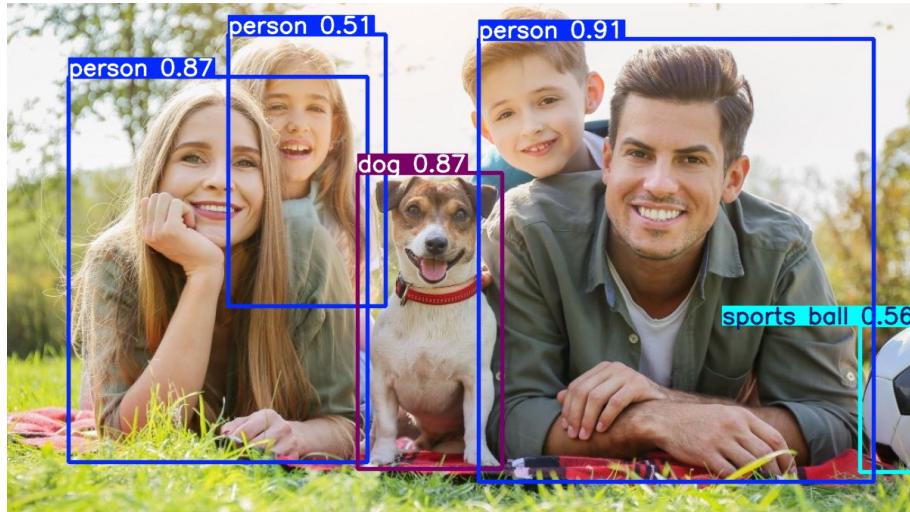


Figura 4: Resultados de detección sobre una imagen estática

También, sobre la misma imagen he querido probar otro modelo para ver como se comporta, he empleado el siguiente comando:

```
!python ./detect.py --weights yolov5x.pt --img 640 --conf 0.25 --source familia.jpg
```

En el comando anterior se especifica el tamaño de la imagen de entrada, los pesos y la confianza. Los pesos se corresponden con el modelo de YOLO que usará para la detección, al correr este modelo en un entorno en local ha tardado mucho en realizar la detección. Por otro lado, la confianza se trata de un valor de límite (threshold), es decir, que solo se mostrarán los objetos que superen ese valor de confianza. Obtenemos el siguiente resultado:

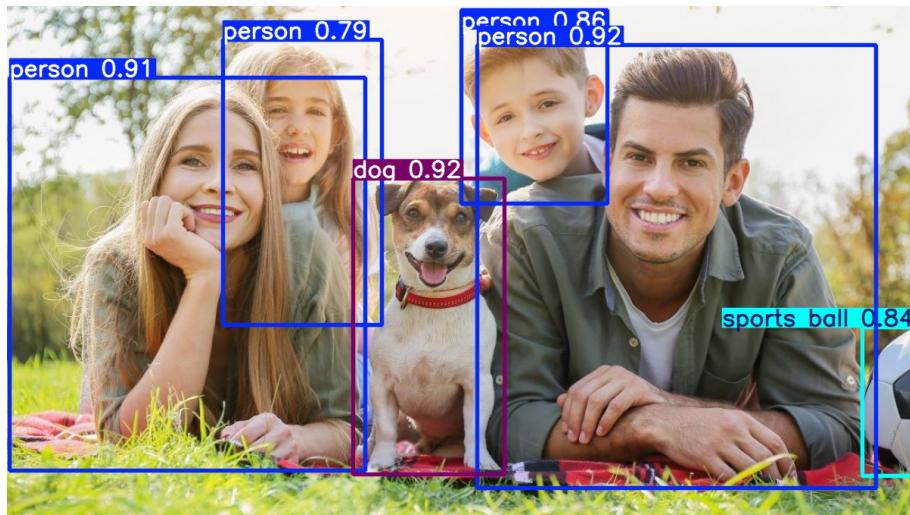


Figura 5: Resultados de imágenes estáticas con el modelo Yolo5X6

Como vemos, la detección con este modelo es mucho mejor que con el modelo por defecto, ya que la niña por ejemplo la detecta con mayor precisión, así como a los demás objetos. Pero el gran problema de esto es que tarda mucho en realizar la detección, aunque esto se deberá más al hardware del que dispongo, ya que no cuento con una tarjeta gráfica de NVidia por lo que todas las operaciones se realizarán con el CPU.

### **2.3. Reconocer imágenes de videos**

También podemos especificar videos para que YOLO reconozca los objetos de los mismos, además, anteriormente se podía realizar detección de objetos en videos de Youtube, pero esa funcionalidad se ha visto capada debido a un cambio de formato en los videos de Youtube.

## **3. Conclusiones**

A lo largo de esta práctica, hemos podido comprobar la efectividad y versatilidad del algoritmo YOLO para la detección de objetos en diferentes escenarios. Su capacidad para procesar imágenes en tiempo real destaca como una de sus principales ventajas, ofreciendo resultados rápidos y precisos. Esto lo convierte en una herramienta valiosa para aplicaciones que requieren respuestas inmediatas, como la vigilancia y los vehículos autónomos.

Durante los experimentos, tanto en la detección en tiempo real como en imágenes estáticas, YOLO ha mostrado un buen rendimiento al identificar correctamente varios objetos con diferentes niveles de confianza. La posibilidad de ajustar parámetros como la resolución de la imagen y el umbral de confianza permite personalizar el modelo según las necesidades específicas de cada proyecto.

Además, hemos comprobado la diferencia en velocidad y precisión entre diferentes modelos de YOLO, siendo los modelos más complejos como YOLOv5X6 más precisos, aunque a costa de tiempos de procesamiento más largos. Esto subraya la importancia de elegir el modelo adecuado dependiendo del entorno de ejecución y los requisitos de la aplicación.

Finalmente, aunque la funcionalidad de detección en videos de YouTube ya no esté disponible debido a cambios en los formatos de video, YOLO sigue siendo una opción robusta para la detección en videos locales. Esto demuestra su flexibilidad y aplicabilidad en diferentes contextos.