Procesado de imagen y visión por computador

DETECCIÓN DE OBJETOS CON YOLO

1. Detección de objetos con YOLO

Uno de los ámbitos en los que las CNN están mostrando todo su potencial es la detección de objetos en tiempo real. Dentro de ese campo una de las implementaciones que mejores y más rápidos resultados está ofreciendo es el detector YOLO[1] ('You Only Look Once'), que es un modelo desarrollado inicialmente por Joseph Redmon en la Universidad de Washington y lanzado en 2015. Desde entonces, la familia de modelos de visión YOLO ha ido ganando gran popularidad por su alta velocidad y seguridad. En esta práctica trabajaremos con la última versión (YOLOv8[2]) lanzada en 2023 y desarrollada por Ultralytics[3]. Esta última versión admite una gama completa de tareas de visión artificial e incluye detección, segmentación, estimación de pose, seguimiento y clasificación. Esta versatilidad permite a los usuarios aprovechar las capacidades de YOLOv8 en diversas aplicaciones y dominios.

En este primer apartado vamos a comprobar el funcionamiento de dicho detector con imágenes reales. Lo primero que vamos hacer es instalar la librería ultralytics. A continuación, abriremos el script de Python facilitado en la práctica *yolov8_detection.py* y leeremos el código para ver cómo funciona y ver qué pasos se siguen para procesar cada imagen.

Una vez hecho eso pasaremos a probarlo. Para ello escribiremos desde el terminal:

python3 yolov8_detection.py --image=horses.jpg

o bien desde Spyder ejecutaremos el programa (*Ejecutar/Configuración de ejecución por archivo/Opciones de línea de comandos*) con los parámetros de entrada: *--image=horses.jpg.* El resultado debe ser una imagen donde se marcarán los caballos detectados y sus probabilidades (fiabilidad del sistema de haber encontrado un caballo).

Cuando hayamos comprobado que funciona, visualizaremos por consola al principio de la ejecución los ochenta clases (tipos de objetos) con los que ha sido entrenado a partir de la base de datos *coco,* la cual es ampliamente utilizada para entrenamiento y evaluación en modelos de deep learning para visión artificial. Cada alumno elegirá tres categorías de estos objetos y para categoría buscará cuatro imágenes que contengan ese objeto en variedad de situaciones (objetos en diferentes poses, oclusiones, agrupamientos de objetos, etc.). En total, se deben reunir 12 imágenes de tres categorías diferentes. Hay que procurar que haya variedad y que algunas de las imágenes sean más difíciles de reconocer. Cuando las tengamos deberemos procesarlas para ver qué resultados obtenemos. Comente las limitaciones encontradas por el detector YOLO. De este ejercicio se entregarán las 12 imágenes procesadas. Conteste a la siguiente pregunta: ¿Qué dos parámetros de entrada incorpora la función de predicción? Justifique cómo afectan ambos parámetros en función de los resultados con las imágenes seleccionadas.

Para comprobar el funcionamiento en tiempo real del detector YOLO, proceda a ejecutarlo a partir de la entrada webcam (para ello, ejecute el script sin introducir ningún argumento de entrada) o bien con un vídeo (para ello ejecute el script de este modo: *python3 yolov8_detection.py –video=video_prueba.avi)*. El vídeo debe incluir varios objetos del dataset con el que ha sido entrenado el modelo. De este ejercicio se entregará también el vídeo de salida. Como puede comprobar en este apartado, uno de los problemas de la implementación es la velocidad de

1 https://yolov8.com
2 https://blog.roboflow.com/whats-new-in-yolov8/
3 https://docs.ultralytics.com/

ejecución del detector en tiempo real. Esto podría mejorarse ostensiblemente si se utilizase un ordenador con unidad de procesamiento gráfico (GPU). Determine el tiempo promedio de ejecución en frames/segundo del detector en su ordenador. Para ello calcule el tiempo de ejecución del vídeo completo utilizando un temporizador en el script[4] y divídalo entre el número de frames del vídeo.

Para finalizar el apartado, probaremos la segmentación. Para ello cambie el modelo de detección de objetos (yolov8n.pt) por el de segmentación (yolov8m-seg.pt) y pruebe a comprobar los resultados.

2. Seguimiento de objetos con YOLO

La última versión de YOLO integra el seguimiento ('tracking') de objetos como una capa adicional de inteligencia sobre la detección. El seguimiento trata de utilizar la información entre frames para conocer el movimiento y trayectoria de los objetos en una secuencia. Para ello, los algoritmos de tracking asignan un identificador (ID) a cada nuevo objeto detectado. Las aplicaciones del seguimiento de objetos son ilimitadas y abarcan desde la videovigilancia hasta el análisis de eventos deportivos.

Ejecute el script yolov8_tracking.py sobre el vídeo de prueba pedestrians.mp4 y responda a las siguientes preguntas:

1) Explique el cometido del siguiente bloque de código y el significado de las variables implicadas (*track_history, track y points*).

```
for box, track_id in zip(boxes, track_ids):
    x, y, w, h = box
    track = track_history[track_id]
    track.append((float(x), float(y)))  # x, y center point
    if len(track) > 30:
        track.pop(0)
    points = np.hstack(track).astype(np.int32).reshape((-1, 1, 2))
    cv2.polylines(annotated_frame, [points], isClosed=False, color=(230, 230, 230), thickness=10)
```

2) ¿Cuál es el objetivo del parámetro fijado en un valor igual a 30? Pruebe con otros valores para ver cómo afecta a la representación.

3) La robustez del algoritmo deberá verificarse comprobando si el sistema es capaz de mantener el mismo ID para un objeto en la secuencia ante una serie de dificultades. Busque o grabe nuevos vídeos con situaciones problemáticas para comprobar el comportamiento del modelo ante algunas de las siguientes situaciones críticas: a) oclusión: un objeto rastreado queda oculto momentáneamente por quedar detrás de otro, b) cruce de trayectorias entre objetos, c) transformación del objeto: el objeto cambia la apariencia durante el recorrido (por ejemplo, una persona camina de frente con una camiseta de un color y al cambiar el rumbo de su trayectoria la camiseta por la espalda es de otro color) y d) efectos visuales: cuando un objeto cambia momentáneamente su apariencia por destello de luz o entra en zona de sombra generando variación

4 https://docs.opencv.org/master/dc/d71/tutorial_py_optimization.html

en sus colores. A partir de este análisis debe comprobar si el sistema mantiene la misma identidad antes y después del cambio o bien lo considera un nuevo objeto después del cambio.

4) YOLOv8 soporta dos algoritmos de tracking: BoT-SORT (utilizado por defecto) y ByteTrack. Investigue sobre el primero de ellos y describa brevemente su funcionamiento.

De este apartado se entregarán las respuestas a las preguntas pedidas, así como los videos de resultados utilizados en el item 3.

3. Estimación de pose (opcional)

En este apartado vamos a comprobar la aplicación de YOLOv8 para estimación de pose de personas, la cual genera la detección de puntos clave (*keypoints* o *landmarks*) a partir de los cuales podemos extraer información de movimientos, patrones, etc. La estimación de pose puede tener un amplia utilidad de aplicaciones en videovigilancia, medicina y entrenamiento deportivo, entre otras. La pose de la persona se visualiza mediante el esqueleto a partir de la conexión de los puntos clave.

En primer lugar, ejecute el script *yolov8_pose.py* mediante el video running.mp4 y analice el código.

Modifique el código para que mediante la función cv2.putText represente en la imagen el índice de cada keypoint con la finalidad de identificarlo. A partir de esta visualización, contente a las siguientes preguntas: ¿Cuántos puntos clave constituyen el modelo y a qué parte del cuerpo están asociados?

Opcionalmente, puede intentar estimar el movimiento del corredor a partir de su patrón de marcha. Para ello deberá calcular el ángulo que forman en cada frame los tres puntos de referencia de cada pierna. Para ello, puede utilizar la función facilitada *compute_angle,* que determina el ángulo entre dos vectores. Como valor a indicar, se deberá al menos representar sobre la imagen el ángulo en cada frame,

De este apartado se entregará el script con el código, así como el vídeo de resultado.