1 Introducción

El presente trabajo tiene como objetivo principal aplicar técnicas avanzadas de procesamiento de video para la detección de vehículos, utilizando el algoritmo YOLO (*You Only Look Once*), una red neuronal convolucional ampliamente reconocida por su desempeño en tareas de detección y localización de objetos en imágenes y videos.
YOLO se destaca en el estado del arte de la detección de objetos debido a su rapidez y precisión. Este algoritmo evita el uso de *pipelines* complejas, lo que le permite procesar imágenes a una velocidad de 45 cuadros por segundo (FPS). Además, logra un promedio de precisión (mAP) significativamente superior al de otros sistemas en tiempo real, consolidándose como una herramienta ideal para aplicaciones que requieren procesamiento eficiente en tiempo real.
Entre las ventajas más relevantes de YOLO se encuentran su alta precisión en la detección de objetos, su capacidad de generalización en dominios diversos y su naturaleza de código abierto, que ha permitido una evolución constante gracias a la contribución de la comunidad investigadora. Estos factores lo posicionan como una solución robusta y versátil para tareas complejas de visión por computadora.
El propósito específico de este trabajo es emplear YOLO para determinar la velocidad de los vehículos detectados en videos, con el fin de aplicar estos resultados a análisis posteriores en contextos reales. Dichos contextos incluyen, entre otros, el estudio del tráfico en vías altamente transitadas y el análisis de carreras automovilísticas, permitiendo una evaluación detallada y precisa.

2 Estado de la cuestión

La visión por computadora ha experimentado un crecimiento exponencial en los últimos años en el campo de la detección y el seguimiento de objetos en imágenes y videos. Este desarrollo se ha dado debido a grandes avances en el hardware de procesamiento, la disponibilidad de conjuntos grandes de datos y la evolución de algoritmos de aprendizaje automático, como las redes neuronales convolucionales (CNNs). El modelo YOLO (You Only Look Once) ha surgido como una solución innovadora que ha impactado mucho en el campo de reconocimiento de objetos en tiempo real. A continuación se analizará el contexto general de la detección de objetos y el modelo YOLO, así como su relevancia y aplicaciones.

2.1 Evolución en la Detección de Objetos

La detección de objetos es una de las tareas más relevantes de la visión por computador que se basa en la clasificación de objetos y su localización en una imagen o video. Esto se realiza mediante cuadros delimitadores (bounding boxes). Últimamente se han estudiado varios enfoques para abordar esta tarea, que pueden clasificarse en métodos tradicionales y basados en aprendizaje automático:

1. Métodos Tradicionales: Existen métodos clásicos para detectar objetos como los modelos basados en descriptores de características (SIFT, HOG) y clasificadores como SVMs. Estos predominaban hace unos años y requieren de introducir manualmente las especificaciones. Están limitados en términos de precisión y capacidad de generalización, sobre todo en problemas muy complejos y dinámicos.

2. Métodos Basados en Aprendizaje Automático Las CNNs fueron revolucionarias.

2.2 Modelo YOLO: Nacimiento y Evolución

YOLO fue introducido por Joseph Redmon en 2016 y fue una herramienta revolucionaria en el ámbito de la detección de objetos. Este modelo presenta un enfoque unificado y en un solo paso, lo que le permite procesar imágenes completas en un único paso, a diferencia de los métodos basados en regiones.

1. Principales Innovaciones YOLO: El procesamiento se hace en una sola etapa, pues YOLO divide la imagen en una cuadrícula y precide a la vez todos los bounding boxes, las clases y la precisión en cada objeto, eliminando la necesidad de pasos intermedios y separados que generen la clasificación. YOLO tiene un diseño muy eficiente, permitiéndolo así actuar con gran velocidad. Es capad de procesar imágenes muy rápido, a velocidades de hasta 45 FPS, por lo que es ideal para aplicaciones en tiempo real. Además, YOLO destaca por su generalización, ya que trabaja bien con imágenes no vistas anteriormente, por eso es una herramienta muy versátil.

2. Versiones de YOLO: Desde que salió, YOLO ha evolucionado bastante. De la versión 1 a la 3, se centraron en mejorar la precisión y velocidad del modelo ajustando su arquitectura. De las versión 4 a la 5, se incluyeron técnicas avanzadas como los mecanismos de atención para mejorar el rendimiento en escenarios más complejos. De la versión 7 a la 8, las mas recientes, se ha trabajado más en el equilibrio entre velocidad

y precisión, metiendo redes neuronales ligeras y más eficientes. Se usa una versión personalizada, yolo11n.pt, que está adaptada a escenarios específicos, como la detección de vehículos.

2.3 Aplicaciones de YOLO

YOLO es un modelo muy versátil por lo que se puede aplicar a muchas industrias diferentes, entre ellos:

1. Seguridad y Vigilancia: Identificar personas en tiempo real en el campo de la videovigilancia o detección de objetos raros o comportamientos extraños en espacios públicos.

2. Tráfico y Transporte: Detectar vehículos para estudiar el flujo en carretera, análisis de la velocidad/comportamiento de vehículos en autopistas o identificar infracciones, como exceso de velocidad.

3. Vehículos Autónomos: Detección de objetos y obtáculos o señales de tráfico. Seguimiento de objetos para tareas.

4. Agricultura: identificación de plagas o problemas en cultivos.

5. Salud: detección de anomalías o lesiones en imágenes de pacientes médicos.

2.4 Desafíos y Limitaciones de YOLO

1. Compromiso entre Velocidad y Precisión: YOLO es rápido pero su precisión puede verse afectada al compararla con otros métodos con más detalle como Faster R-CNN, sobre todo en escenarios con objetos demasiado pequeños o complejos de analizar.

2. Escala: si los objetos son demasiado pequeños/grandes, esto se convierte en un desafío.

3. Datos de entrenamiento: la detección de objetos puede ser menos precisa si los datos tienen poca calidad y son demasiado diversos.

2.5 Por qué YOLO en este Proyecto

El empleo de YOLO se justifica por lo siguiente:

1. Procesamiento Vídeo: el análisis de tráfico de coches necesita unos resultados rápidos y en tiempo real de vídeos a altas velocidades.

2. Precisión Alta: YOLO es un modelo ligero pero logra un gran acierto al reconocer objetos.

3. Implementación Sencilla: es un modelo de código abierto con mucha documentación.

4. Versátil: es un modelo flexible que puede ser adaptado a muchos contextos muy específicos.

3 Desarrollo

La metodología del presente proyecto se estructuró en diferentes fases para garantizar la correcta implementación del sistema de detección y análisis de vehículos, empleando el modelo YOLO y técnicas avanzadas de visión por computadora. A continuación, se describen cada una de las etapas:

3.1 Selección del Modelo de Detección de Objetos

Para este proyecto, se seleccionó el modelo YOLO para la detección de objetos, concretamente de coches. Se ha escogido este algoritmo por su eficacia a la hora de ejecutar su tarea en tiempo real manteniendo una alta precisión en la identificación de objetos. Específicamente, se utilizó una versión preentrenada del modelo denominada YOLOv11n (yolo11n.pt), ya que está optimizada para su uso en sistemas que requieren analizar velocidad y precisión. El modelo es capaz de identificar múltiples clases de objetos en un solo pase de red neuronal, pero en este caso solo se quiere reconocer automóviles, entonces se utilizarán las clases [2, 7].

El modelo es conocido por robustez y fácil integración en aplicaciones reales, lo que lo hace adecuado para analizar contextos como el tráfico en tiempo real.

3.2 Entorno y Datos

El entorno de trabajo está basado en Python, utilizando bibliotecas especializadas como Ultralytics para trabajar con YOLO y OpenCV para el procesamiento de vídeos. El dato de entrada es un vídeo de tráfico de coches real. Este vídeo fue almacenado en un directorio estructurado llamado ../data. Se creó también una clase personalizada (Video) para extraer información relevante del vídeo como la resolución, los FPS o fotogramas por segundo y la duración total. El modelo YOLO se cargó desde un directorio preconfigurado.

3.3 Definición de la Región de Interés (ROI)

Para simplificar el proyecto y así optimizar los recursos, es preciso centrarse exclusivamente en los objetos que sean interesantes (los coches). Para ello, se define una ROI dentro de cada frame del vídeo. Esta región se delimitó mediante coordenadas específicas (x1, y1, x2, y2), que son acordes al área del video donde se esperaba encontrar vehículos en movimiento. La ROI permitió reducir el área que se va analizar y procesar, descartando elementos irrelevantes como el puede ser el cielo o la delimitación del medio de la carretera.

3.4 Algoritmo de Detección y Seguimiento

La base de esta práctica ha sido la integración del modelo YOLO con el procesamiento de vídeo. Para leer los fotogramas se diseñó un generador de frames a través de la clase Video para así poder iterar sobre cada uno de ellos. Cada frame es recortado según la ROI antes de ser procesado. Para detectar los vehículos, se configuró el modelo YOLO para que tan solo se centrara en coches y camiones. A cada frame se le aplica el método 'track', el cual asigna a cada objeto un identificador único para hacer seguimiento de él a lo largo del tiempo. Los objetos que se detectan se representan con un cuadro delimitador o bounding box y salen en el vídeo final.

3.5 Estimación Velocidades

Una de las partes claves de la práctica fue calcular la velocidad de cada coche utilizando píxeles por frame. Para ello, se desarrolló lo siguiente:

1. Posición del objeto: Para cada vehículo que se reconoce, se registraron sus coordenadas de posición en los distintos frames.

2. Cálculo de Velocidad: En base a las diferencias de posición, en píxeles, y el tiempo transcurrido entre los distintos frames (por tasa de FPS), se calcula la velocidad en términos de píxeles por frame. Estos valores se imprimen directamente en el vídeo procesado junto con el identificador de cada vehículo.

3. Visualización: Se incluyeron puntos centrales de los vehículos para ilustrar la posición actual y las trayectorias detectadas.

4. Tráfico: Se ha considerado que si se cuentan más de ciertos coches seguidos, entonces se imprimirá en el vídeo que hay tráfico.

3.6 Visualización de Resultados

Durante el procesamiento se muestra una ventana emergente que presenta los resultados en tiempo real: la detección de los vehículos, la ROI y las velocidades. Todos los frames procesados se guardan en un archivo de salida con OpenCV. Esto lo que permite es generar un vídeo final para ver en análisis resultante de todos los cálculos realizados.

3.7 Evaluación y Validación del Sistema

Para evaluar el modelo se han hecho dos pruebas: de precisión - se compararon las detecciones realizadas con YOLO con observaciones manuales para evaluar qué tan bien se identificaron los vehículos -, y validación de velocidades para ver si son coherentes y así ajustarlos.

3.8 Otras Consideraciones

Este proyecto sienta las bases para futuras implementaciones más avanzadas como puede ser la conversión de las velocidades de píxeles por frame a km/h calibrando el sistema con distancias reales. Se podría usar en aplicaciones como análisis de tráfico en tiempo real o monitoreo de carreteras para estudios de seguridad en las carreteras.

Dataset: el vídeo se encuentra en el enlace siguiente: "$https://www.youtube.com/watch?v = wqctLW0Hb_0$"

3.9 Exploración del dataset

Para esta práctica se ha elegido un vídeo en vista aérea de coches que van circulando por una autopista en dos sentidos contrarios. Este dura unos 34 minutos y tiene una resolución de 640x360 píxeles. El vídeo tiene buena iluminación, contraste y claridad.

4 Conclusión

En esta práctica se ha implementado un sistema de detección de vehículos en tiempo real mediante el modelo YOLOv8, un algoritmo muy avanzado y eficiente para esta tarea. Se han logrado procesar videos frame por frame, identificar los vehículos y sus posiciones mediante bounding boxes. El desarrollo incluyó varias estapas:

1. Preparar entorno y herrramientas: hubo que elegir el entorno de desarrollo e instalar librerías como ultralytics y openCV.

2. Implementaciónon: se empleó YOLO para realizar las detecciones con precisión y rapidez. Se escribió un pipeline eficiente para procesar y guardar el vídeo. Se filtró por clases específicas (coches y camiones).

3. Visualización y Análisis: se incorporaron bounding boxes y etiquetas con la precisión para que fuera lo más visual y claro posible. Se generó un archivo de salida con el vídeo procesado.

En definitiva, se ha obtenido una detección confiable de la posición de los coches y es un modelo muy útil que podría mejorarse para ser aplicado a la vida real.