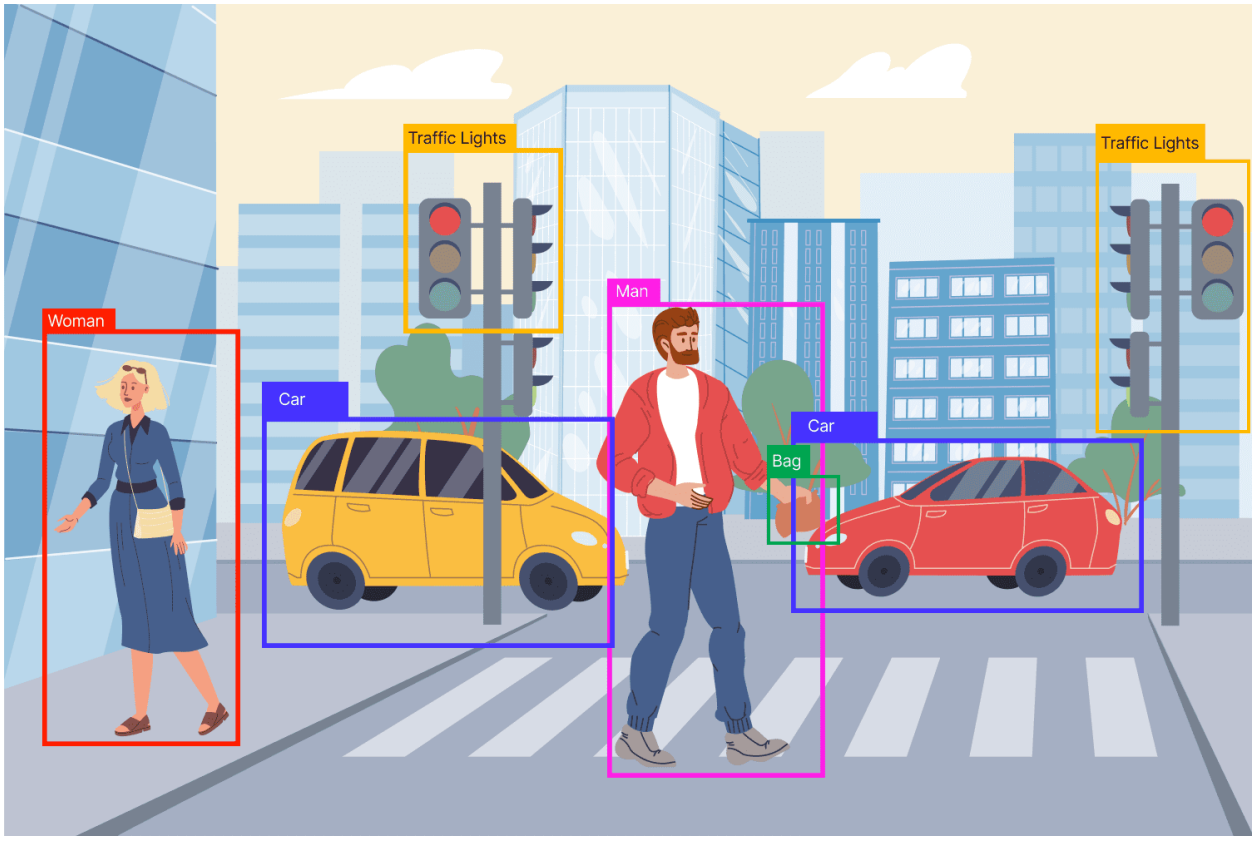


Task 2: Recognizing Objects in Video Sequences

Course name – Project: Computer Vision (DLBAIPCV01)

A course of Study – Bachelor of Science in Applied Artificial Intelligence

Author Name – Fotimakhon Gulamova

Matriculation Number – 92116230

Tutor's Name – Konstantinos Amplianitis

Table of Content

Introduction

This written assignment aims to develop a computer vision system that accurately detects and segments objects in video. I will use state-of-the-art object detection and segmentation algorithms. Firstly, I implement three SoA approaches: YOLO, SSD, and Faster R-CNN and evaluate their performance. Then, I will analyse the results visually and highlight their strengths and weaknesses. Next, I will determine the best-performing approach based on evaluation and analysis. To showcase the best approach in a video I will drop a link.

State-of-the-Art Approaches

1. YOLO (You Only Look Once)

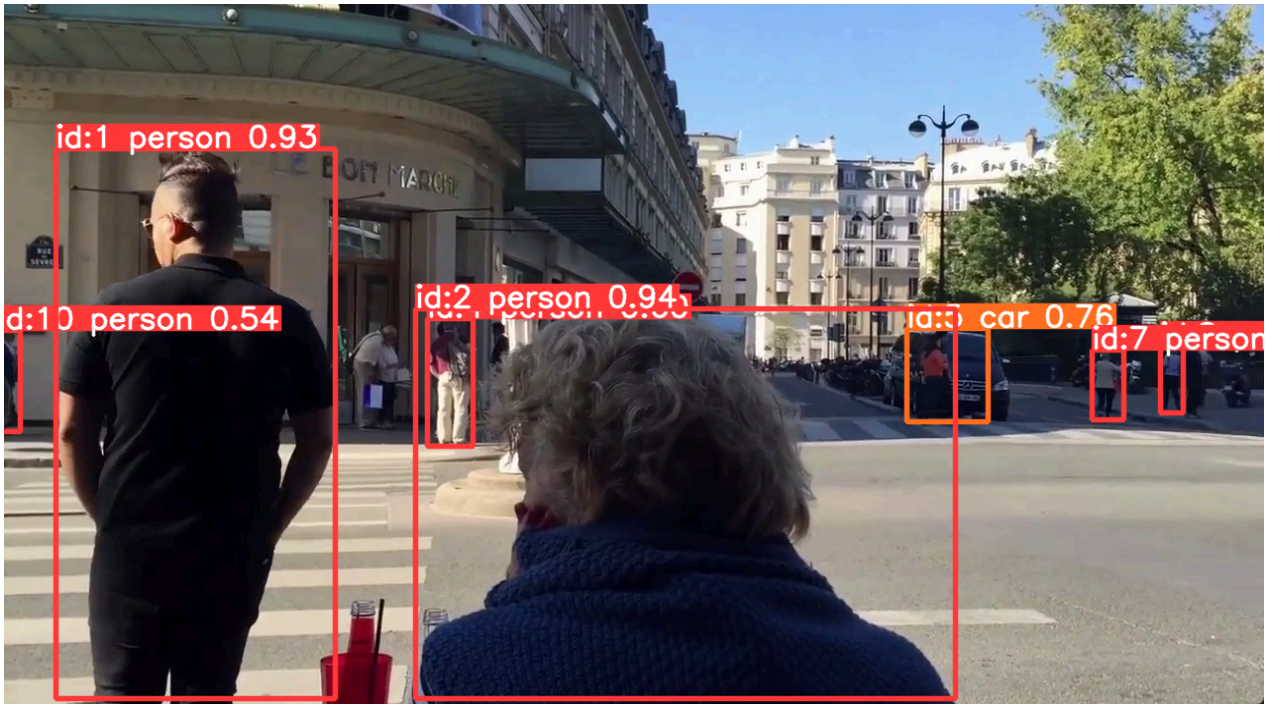
Description

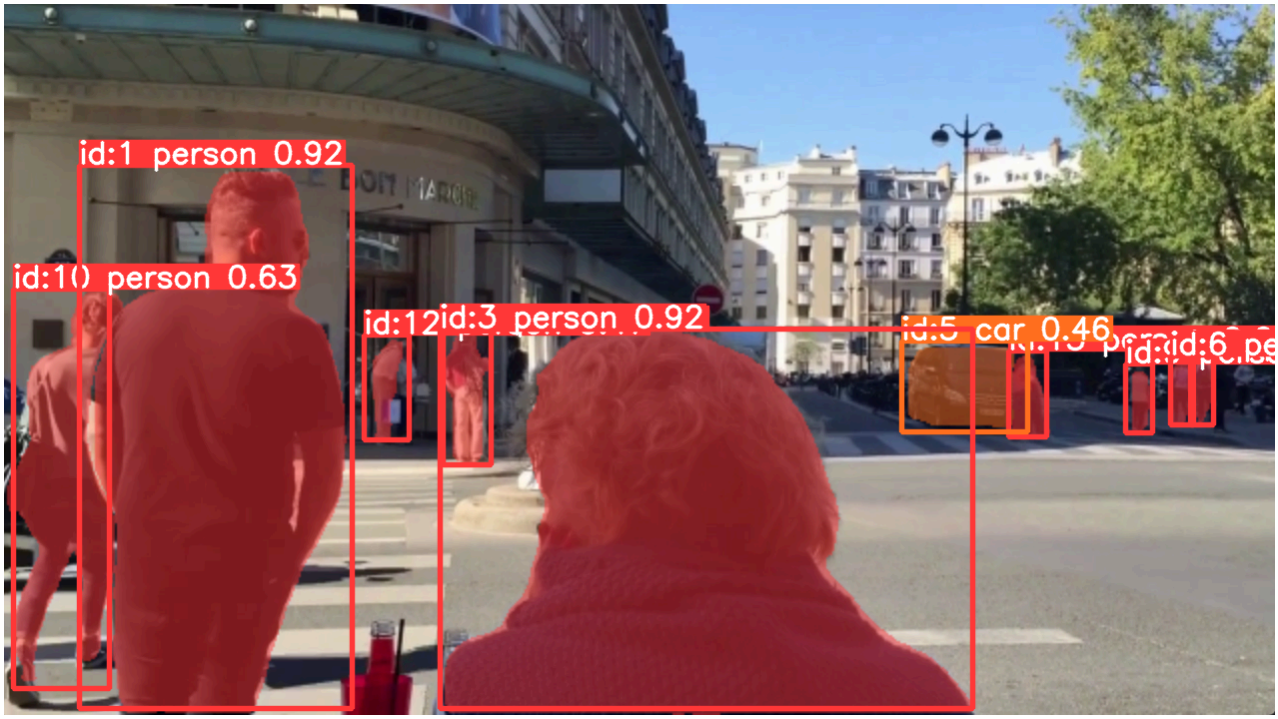
Deep learning models such as YOLOv8 have become essential in various industries, including robotics, autonomous driving, and video surveillance. These models detect objects in real-time and impact safety and decision-making processes. YOLOv8 (You Only Look Once) uses computer vision techniques and machine learning algorithms to identify objects in images and videos at high speed and accuracy. This allows for efficient and accurate object detection, which is crucial in many applications (Keylabs, 2023).

Implementation Details

I created a `run_model` function to implement **object detection** and **segmentation**. This function takes three parameters as input: ***a model***, ***an input video***, and ***an output video***. It reads each frame one by one and visualises the results of the input video onto the frame. Then, the annotated frames are saved in the output video file until all frames have been processed or the user stops the process by pressing "**q**".

I used the YOLO model (`yolov8n.pt, "v8"`) that shows videos with detected bounding boxes for object detection. Similarly, for object segmentation, the YOLO model with segmentation-specific weights (`yolov8n-seg.pt`) generates videos with segmented objects.





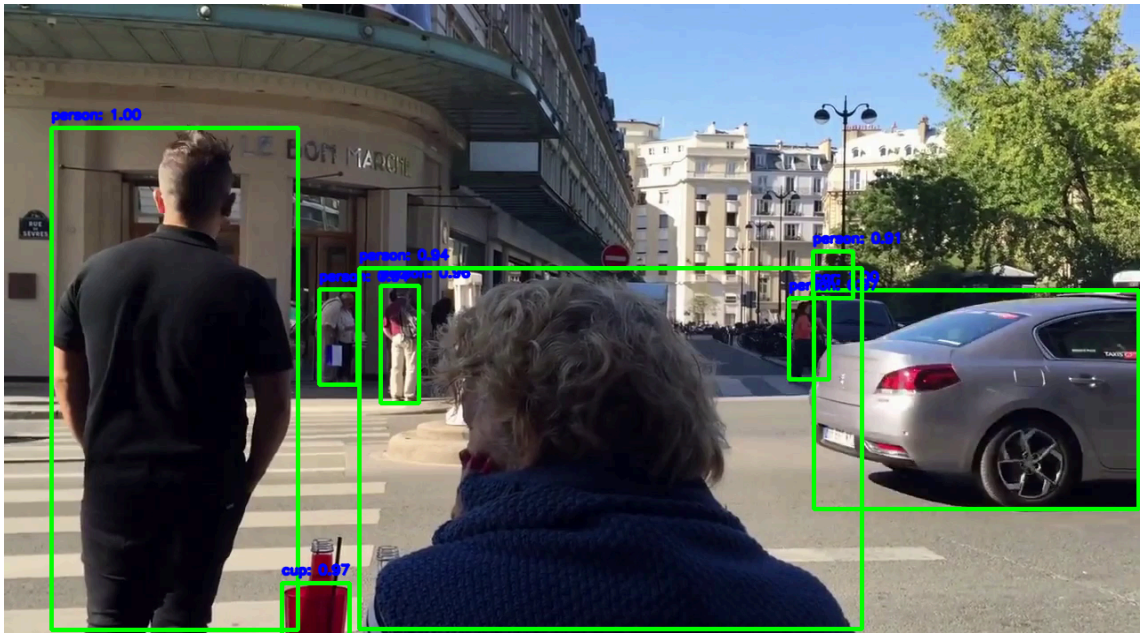
2. Faster R-CNN (Region-based Convolutional Neural Network)

Description

Faster R-CNN is a state-of-the-art object detection model. It has two main components: a deep fully convolutional region proposal network and a Fast R-CNN object detector. It uses a region proposal network (RPN) that shares full-image convolutional features with a detection network (Ren et al., 2015). The RPN is a fully convolutional neural network that generates high-quality proposals. Then, it is used by the Fast R-CNN for object detection. The two models are combined into a single network, with the RPN guiding where to look for objects (Ren et al., 2015).

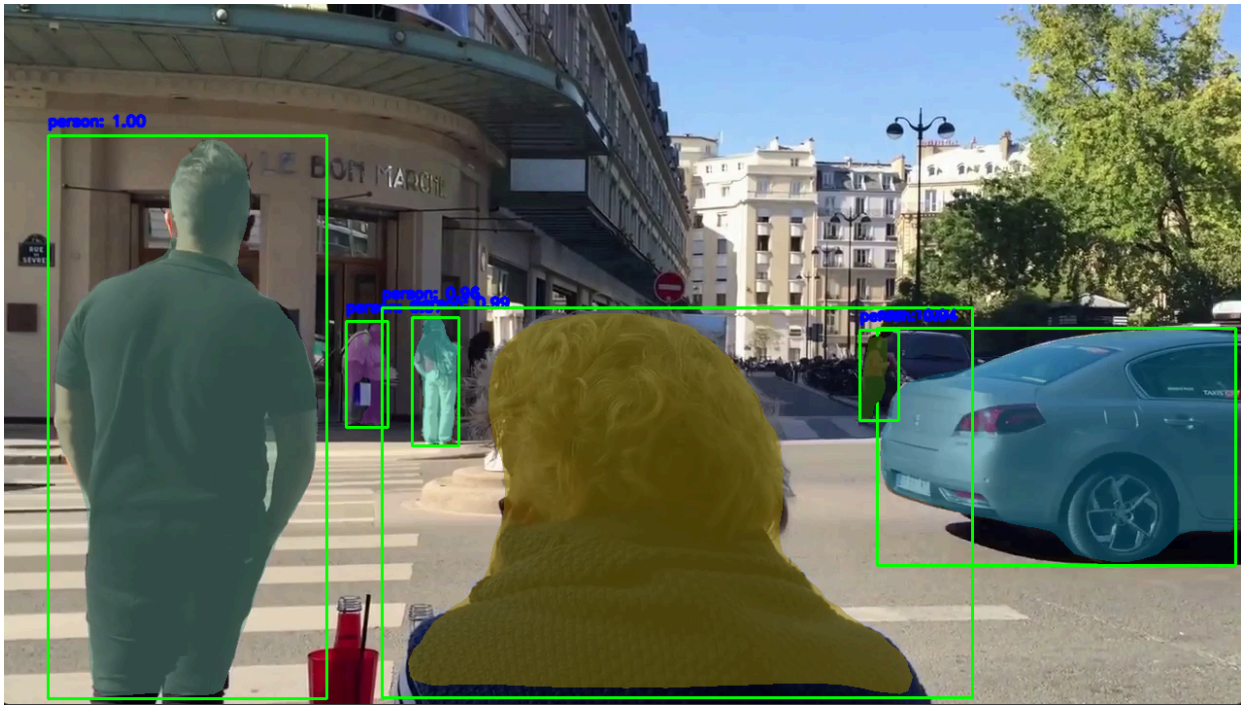
Implementation Details

To implement **object detection**, I created two functions: `get_model` and `detect_and_draw_boxes`. The `get_model` function loads a pre-trained **Faster R-CNN** model that is part of the torchvision library and is pre-trained on the COCO dataset with a **ResNet-50-FPN** backbone. I set the model to evaluation mode. Then**,** the `detect_and_draw_boxes` function performs object detection on a single video frame and draws bounding boxes around detected objects. It transforms the frame into a tensor and passes it to the model. This model returns predictions, including bounding boxes, labels, and scores for detected objects. Bounding boxes with a confidence score above **0.9**, and labels indicating the class and confidence score are added.



To implement **object segmentation**, I created functions to load a pre-trained **Mask R-CNN** model, preprocess video frames, apply segmentation, and overlay masks on the frames. Firstly, I used a pre-trained **Mask R-CNN** model with a **ResNet-50-FPN** backbone loaded from the

torchvision library and set it to evaluation mode. I trained the model on the COCO dataset. Then, a `preprocess_frame` function preprocesses each video frame and transforms it into a tensor. Next, an `apply_segmentation` function applies the segmentation process to the preprocessed frame and an `overlay_masks` function overlays segmentation masks on the frame, draws bounding boxes, and adds labels for confident detections. This involves filtering detections by a confidence threshold, overlaying masks, drawing rectangles, and adding text labels.



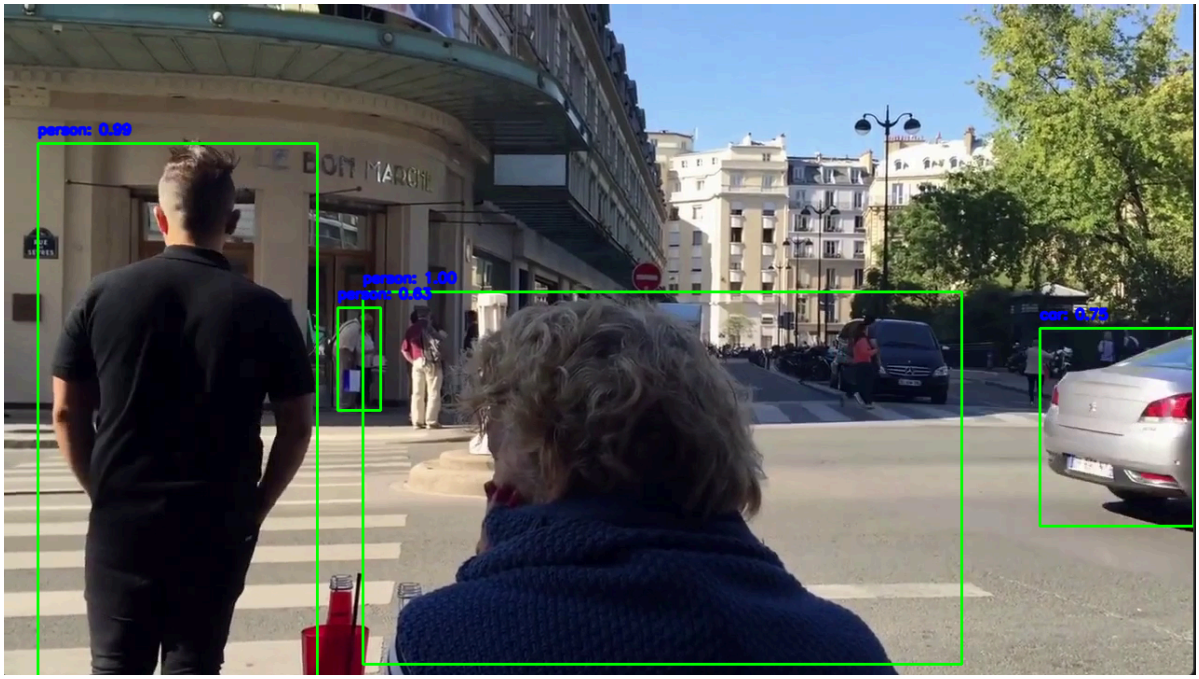
3. SSD (Single Shot Multibox Detector)

Description

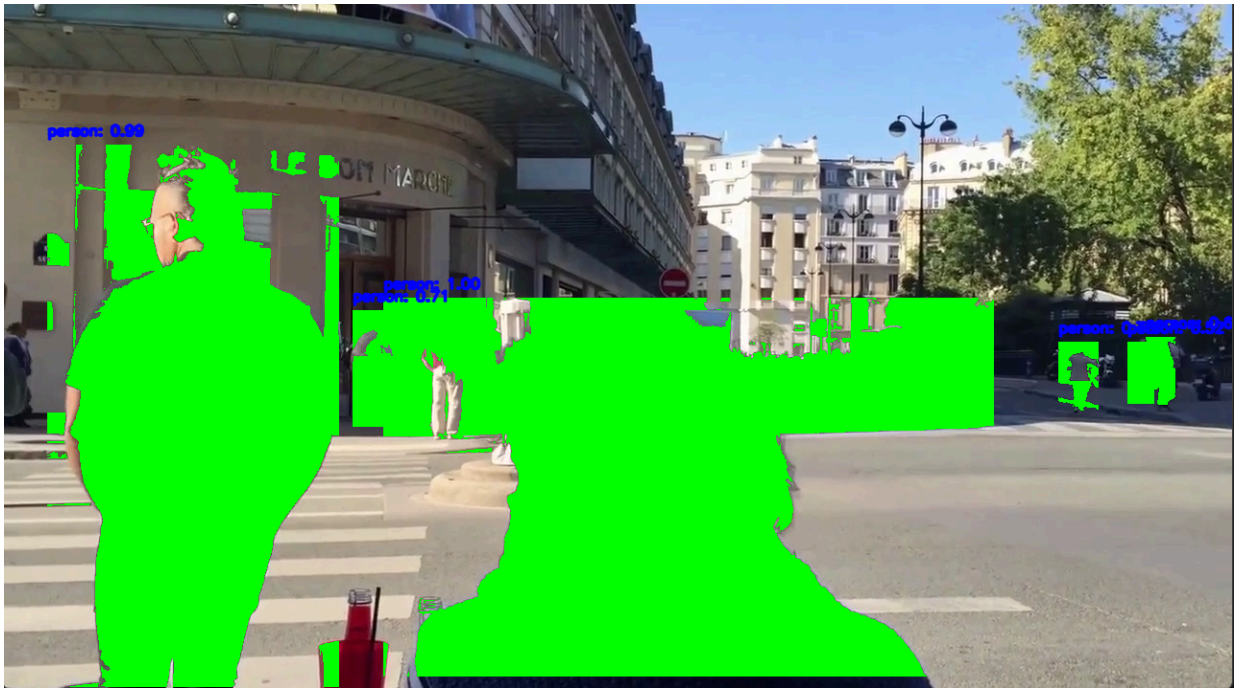
SSD, or Single Shot MultiBox Detector, is a method for object detection in images using a single deep neural network. It discretises the output space of bounding boxes into a set of default boxes with various aspect ratios and scales per feature map location. During prediction, the network generates scores for the presence of each object category in each default box and refines the boxes to match the object shapes better. SSD combines predictions from multiple feature maps at different resolutions to effectively handle objects of various sizes, eliminating the need for proposal generation and resampling stages, thereby simplifying the training process and integration into detection systems (Liu et al., 2016).

Implementation Details

I created a `ssd_object_detection` function that uses a pre-trained SSD model, processes video frames, applies detection, and draws bounding boxes around detected objects to implement **object detection** using an **SSD (Single Shot Multibox Detector)** model.



Similarly, I created the `ssd_object_segmentation` function that loads the pre-trained model, processes video frames, applies segmentation, and draws masks and labels on detected objects to implement **object segmentation**.



Evaluation

In this section, I will evaluate and compare three popular object detection models: YOLO (You Only Look Once), Faster R-CNN (Region-based Convolutional Neural Network), and SSD (Single Shot MultiBox Detector). I worked on a CPU device, not CUDA.

The evaluation stage includes:

- **Frames per second (FPS):** FPS measures the number of frames processed by each model per second.
- **Inference time:** Inference time represents the time taken by each model to detect objects in a frame.
- **Model size**: Model size indicates the disk space occupied by each model.

	Inference Time (s)	FPS (frame per second)	Model Size (MB)
YOLOv8	0.11	8.32	12.48
Faster R-CNN	2.73	0.35	169.89
SSD	0.41	2.26	135.99

Discussion on Performance Differences

From the evaluation results, I observed the following:

- **Speed:** YOLO outperforms both Faster R-CNN and SSD in terms of FPS and inference time. It demonstrates its suitability for real-time applications.
- **Accuracy:** Faster R-CNN tends to achieve higher accuracy compared to YOLO and SSD, suggesting better accuracy in object detection tasks.
- **Model Size:** YOLO exhibits the smallest model size, making it advantageous for deployment on devices with limited storage capacity.

Best Performing Approach

Based on the evaluation results and qualitative analysis, YOLO8v is the best-performing SoA approach for object detection and segmentation in video sequences. Its exceptional speed, compact model size, and robust performance make it an ideal choice for real-world applications where accuracy and efficiency are vital.

Video Demonstration

To view the performance of each model, please follow the provided link. Link:
🗀 Computer Vision Project

Conclusion

In conclusion, this assignment aims to create a computer vision system for accurate object detection and segmentation in videos using YOLO, SSD, and Faster R-CNN. After implementing and evaluating these approaches, I analysed their results, determined the best-performing method, and showcased it in a video demonstration.

Reference

- Keylabs. (2023, December 20). *Under the hood: YOLOV8 Architecture explained*. Keylabs: Latest News and Updates. https://keylabs.ai/blog/under-the-hood-yolov8-architecture-explained/#:~:text=YOLOv8%20is%20a%20state%2Dof,objects%20in%20real%2Dtime%20scenarios.
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. E., Fu, C., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In *Lecture notes in computer science* (pp. 21–37). https://doi.org/10.1007/978-3-319-46448-0_2
- Ren, S., He, K., Girshick, R., & Sun, J. (2015, June 4). *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. arXiv.org. https://arxiv.org/abs/1506.01497v3
- Ultralytics. (2024, March 2). *YOLO Performance Metrics*. Ultralytics YOLOv8 Docs. https://docs.ultralytics.com/guides/yolo-performance-metrics/#object-detection-metrics