Giorgio Mendoza
CS549-E24-E01
Dr. Alexandros Lioulemes

Week 9 Assignment

Objective: The objective of this project is to develop a video processing system that accurately detects and counts pedestrians, bicycles, and cars within a defined area of interest (pedestrian lines) using a pre-trained YOLOv5 model. The system aims to process real-time video footage, applying constraints to ignore objects outside specified zones and only counting objects that meet the required overlap criteria within the bounded area.

Code Description: This program is designed to process video footage for object detection and counting, utilizing the YOLOv5 model. The YOLOv5 model is pre-trained and loaded using the PyTorch framework to detect objects in each frame of the video. The main function, `detect_objects`, converts the video frames from BGR to RGB color space and passes them to the YOLOv5 model, which returns the coordinates and labels of detected objects.

The script defines a bounded area within the video where objects of interest, such as persons, bicycles, and cars, are detected and counted. A function, `is_inside_bounded_area`, calculates the percentage of the bounding box that falls within the bounded area, helping to determine whether an object should be counted. The video processing loop reads and processes each frame, skipping a few frames at a time to reduce computational load. It draws horizontal and vertical lines to outline the area of interest and checks whether detected objects cross these lines.

The objects are classified as persons, bicycles, or cars based on their labels and confidence levels. The script counts objects that cross the defined entry and exit lines and increments their respective counters by specified amounts. The confidence levels of detected objects, particularly for cars, are displayed as white labels on the video frames. Finally, the script writes the processed frames with annotated labels and counts to an output video file, providing a visual representation of the detection and counting process.

Limitations: Implementing this task required a thorough understanding of the limitations of the YOLO model used and the processing power available. Google Colab was utilized for the entire implementation, and processing time was a critical consideration. Initially, processing every video frame proved too time-consuming, so a frame skip of 5 was introduced, allowing every fifth frame to be processed. This approach balanced the need for speed with the retention of essential features.

Another important aspect was identifying the area of interest, specifically the boundary area within the crossing lines. Initially, segmentation was considered to define this area, but given the relatively static environment, where only bicycles, persons, and some cars exhibited movement, a simpler approach was adopted.

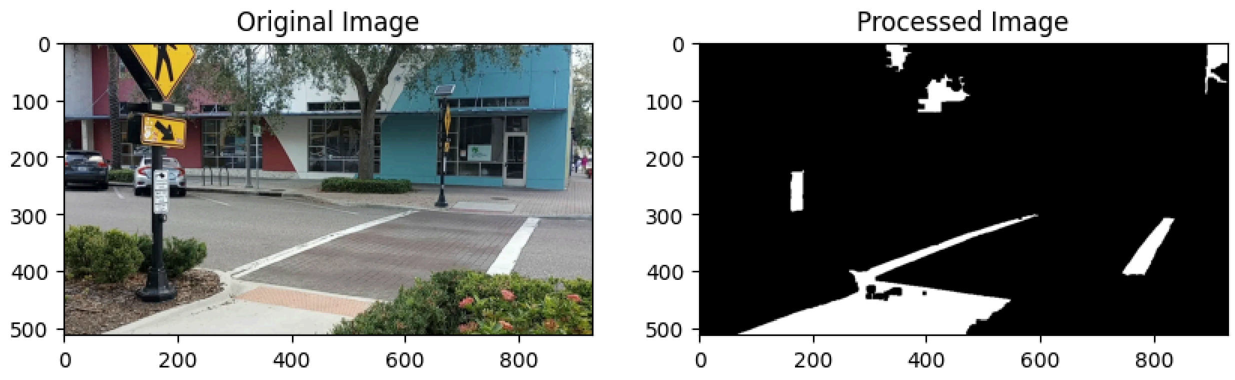

Figure 1 Area of interest using Segmentation

The coordinates of the area of interest were manually identified, streamlining processing and ensuring accuracy. To further speed up processing, the resolution of the video was reduced.

However, several limitations of the YOLO model became evident after using it for the first time. For instance, while the model could successfully identify persons and cars, it lacked the ability to discern the direction of movement.

To address this, blue lines were added as top and bottom checkpoints, and magenta lines as left and right checkpoints. These lines served as markers to update counters when an object crossed both lines. Nevertheless, this method introduced its own challenges; skipping frames occasionally led to missed updates in the state flags, resulting in inaccurate counts.

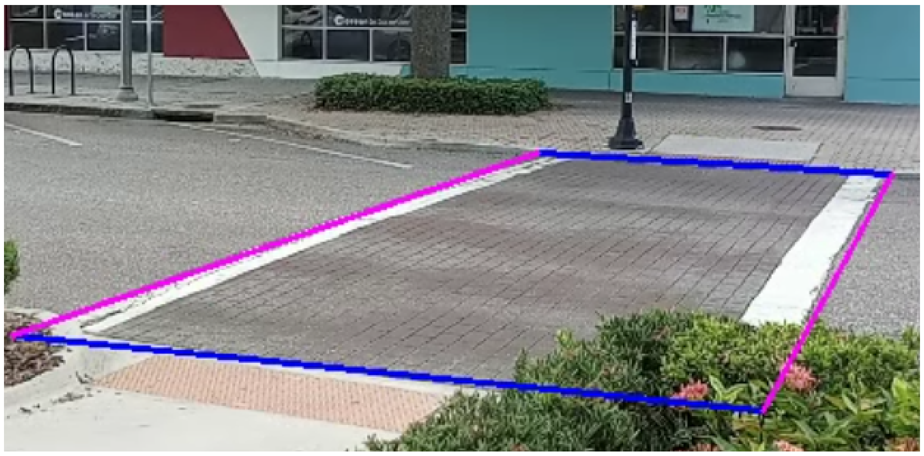

Figure 2. Area of Interest Identified

Another limitation was the way the model assigned IDs to detected persons. Ideally, the model would track a person throughout their appearance in the video, maintaining a consistent ID. However, the model assigned a new ID to the person in each frame, which created inconsistencies.

This behavior, though problematic, was leveraged to identify when a person first entered and then left the area of interest. The ID updates were used to increment counters by dividing the number of ID changes by the total frames processed. This logic was similarly applied to bicycles moving from right to left.

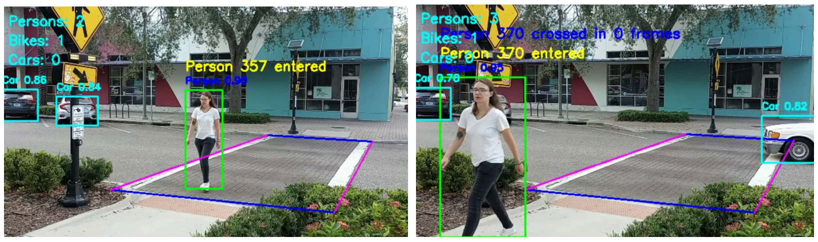

Figure 3. YOLO Model In Action

Cars presented a more complex challenge, as they moved both from right to left and left to right. Counting cars was based on the percentage of the car's bounding box inside the area of interest. The percentage was divided over the number of frames to determine whether a car should be counted. While this method successfully counted most cars, it did miss a few due to the set thresholds.

Conclusions: The submitted video demonstrates the program operating with optimized parameters, successfully detecting and counting three persons and one biker. For the cars, the program accurately counted 9 out of 12 due to the thresholds applied. Despite the challenges encountered, the program performed well overall.

However, it is important to note that the environment in this task was largely static, with only bikes, persons, and cars in motion. Consequently, the program may not perform as effectively in more dynamic, real-world scenarios.

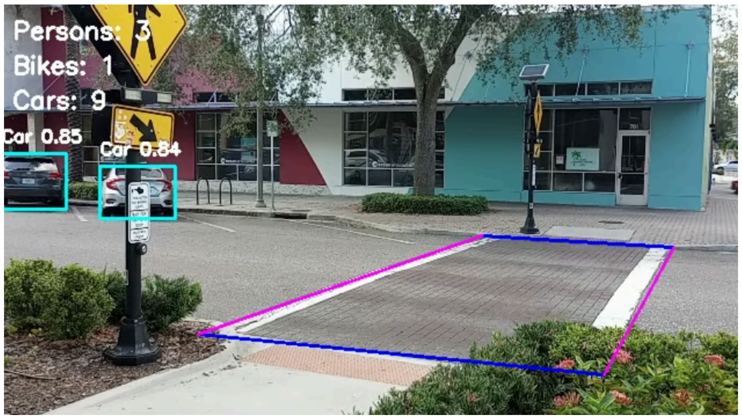

Figure 4. Final Frame with Counters Updated