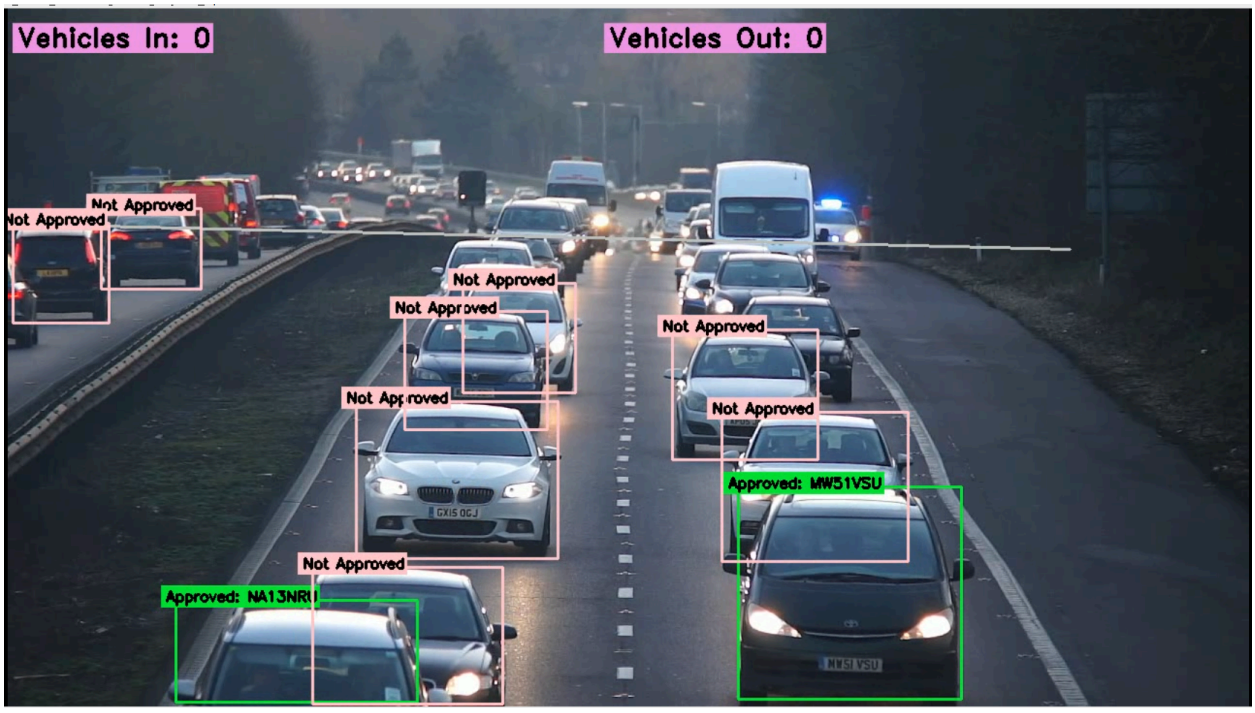
Intel® Unnati Industrial Training 2024

PS-13: Vehicle Movement Analysis and Insight Generation in a College Campus using Edge AI



Team: Encoders

Sayan Banerjee

Prasanna Dhungana

Pushkar Niroula

Arvind Kaphley

Mentor: Mr. Kunal Anand

College: Kalinga Institute Of Industrial Technology

Introduction

College campuses face a great deal of difficulty in effectively managing vehicular activity because they frequently see heavy traffic and complicated parking needs. Modern monitoring and analysis tools are needed to guarantee the safety and efficient operation of automobiles. Inadequate real-time traffic insights and inefficient use of parking spaces are among the problems caused by the inadequacy of traditional manual surveillance and parking management system methods.

Our project makes use of cutting-edge edge AI and computer vision technologies to solve these issues. This project's two main goals are as follows:

- **Real-time parking lot monitoring:** By detecting and displaying occupancy status using video feeds from parking lot cameras, real-time parking lot monitoring makes it possible to make effective use of available spaces and quickly identify any that are vacant.
- **Vehicle Movement Tracking and Authorization:** The system enhances campus security by identifying and validating license plates against a pre-approved database, monitoring vehicle movements on campus, entry and exit points, and confirming authorization through the use of video feeds.

By integrating custom-trained license plate, YOLO nano, and PaddleOCR models, the system improves vehicle detection and character recognition, opening the door to more intelligent, safe, and effective vehicle management.

Dataset

- License Plate dataset: The dataset was publicly available in Roboflow datasets [1]. This dataset consists of 24K+ annotated images divided 88%, 8%, and 4% respectively between train, valid and test sets.
- COCO dataset: Common Objects in Context(COCO) is a large-scale object detection, segmentation dataset. It consists of more than 330K+ annotated images for 80 different object categories. It is frequently utilized for YOLOv8 and other deep

learning models for object detection training and evaluation. It is used to pretrained the YOLOv8 nano, small, medium, large, and very large models.

Methodology

1. Requirement analysis

We focused on the objective of the problem statement. Based on that we were able to identify what were the steps we have to take. First, we decided that mostly parking area spaces and the vehicle in and out movements are mostly monitored by separate CCTV cameras. Hence we decided to perform inference separately on two video streams.

For inference on parking space video footage, we estimated the following tasks: marking the parking lot spaces, and predicting vehicle's presence inside it. Similarly for inference on road video footage or vehicle movement footage, we estimated marking a line where vehicle in and out take place (we assume vehicle in and out happen from a same place), counting and displaying vehicle in and out counts in realtime and comparing detected plates with plates in the database. So the list of major tools we required are:

- Python for ease of use and wide support of libraries
- OpenCV for image/frame processing as well as reading and writing video files
- Ultralytics for processing and using YOLO models
- SQLite3 for easy database interaction with python scripts
- PaddleOCR for license plate characters recognition
- OpenVINO for conversion of YOLO models to OpenVINO format

2. Data preparation and preprocessing

We did not require any preprocessing or augmentation on the COCO dataset which we used for vehicle detection tasks. This is because the model we used was already pre trained on it. According to Ultralytics (which introduced YOLOv8 models), mosaicing techniques were used in training batches to enhance model

generalization across various object sizes and contexts. Eventually the information about dataset configuration would be stored in a YAML file which is important during model training.

However, for the license plate detection model, we used a custom dataset found in Roboflow public datasets. Preprocessing applied during dataset export are: auto-oriented, and resized to 640 x 640 dimensions. Following are the augmentations applied to the dataset:

- Horizontal Flip, and rotations between -10° and +10°
- Crop: 0% Minimum Zoom, 15% Maximum Zoom
- Shear: ±2° Horizontal, ±2° Vertical
- Grayscale: Apply to 10% of images
- Hue: Between -15° and +15°
- Saturation: Between -15% and +15%
- Brightness: Between -15% and +15%
- Exposure: Between -15% and +15%
- Blur: Up to 0.5px
- Cutout: 5 boxes with 2% size each

These changes were already applied using roboflow tools during model export.

3. Model Selection and Training

When comparing this model's mAP values with those of its predecessors, including YOLOv7, YOLOv6, and others, we were impressed with its simplicity, accuracy, and efficiency. Since YOLO NAS would be effective and helpful in identifying smaller objects within a frame, we gave it some thought. Better models, like the recently released YOLOv9 and YOLOv10, were also available. But we went with YOLOv8 because of its great community support—bug fixes would have been simpler—and because it was simple to use and export when switching to the OpenVINO model for efficiency.

The pretrained model was used for vehicle detection and the license plate model was custom trained, in kaggle notebook as per the instruction given in the official documentation of YOLOv8.

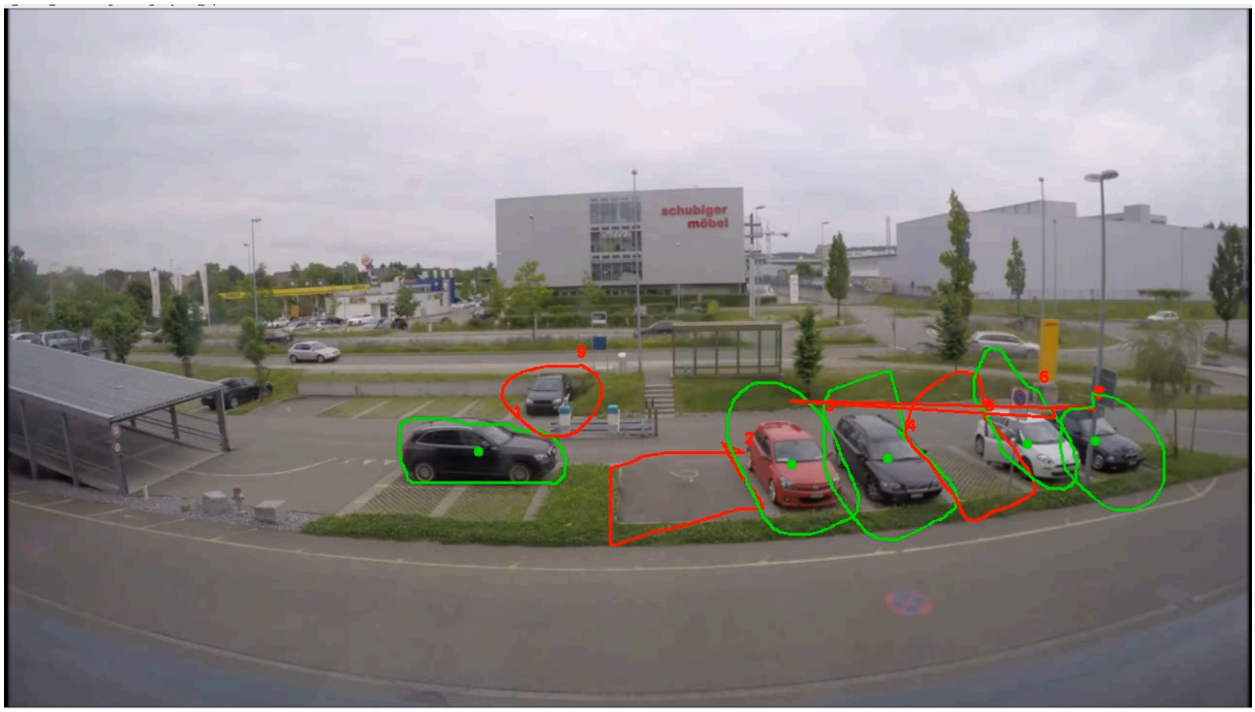
4. Model optimization

The YOLOv8 models can achieve up to a 3x speedup on CPU inference by converting to OpenVINO, utilizing Intel's optimization and acceleration capabilities. OpenVINO also allows models to be used on Intel GPUs and NPUs, reducing memory footprint and computational demands. This optimization is crucial for real-time applications and constrained processing power settings like embedded systems, mobile devices, and Internet of Things applications, ensuring high-performance and low-latency object detection.

We optimize the original FP32 precision encoding by converting it to half or into FP16 precision encoding after exporting to OpenVINO. It reduces the memory footprint and computational requirements of the models, making them more efficient to run on resource-constrained CPU and edge devices.

5. Parking Video inference

The project allows the user to mark the parking spaces as well as assign ids to each lot. These parking spaces were stored as polygons in a pickle file as a numpy array. The frame was processed for any vehicle detection and point polygon test was done using OpenCV to check if the center of bounding boxes calculated from the detected vehicles data is present inside that polygon or not. This would display in real time if any parking space is available or not.



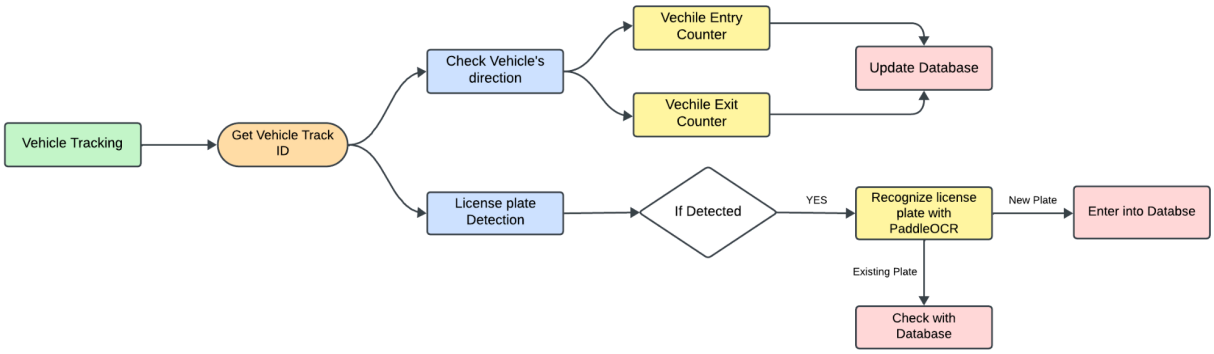
6. Vehicle Movement Inference

If the user wants to perform inference to get real time information of vehicle entry and exit counts, the user is allowed to select a line in a frame where possible vehicle in and out can be counted properly assuming there is only one way of vehicle in and out.

a. To check if any vehicle's bounding box's center point has crossed through the line or not, we store two previous center points and use *perp dot product.* This is calculated between the line and each center point respectively and the results are compared. If the results are of different magnitude the center point has crossed the line.
b. Similarly by comparing the y-coordinates of these two center points, we can get the direction in which the vehicle is traveling. Currently, the program assumes the vehicles' movements are from north to south direction or vice versa only.

Another part of inference done on road video/vehicle movement video is to check its license plate and if detected, recognize it and display "Approved" if it's

entry is present in the database. We utilize PaddleOCR to recognize license plates after detection.



7. Database utilization

We use sqlite3. We used this database, as it is easy to set up and can be easy to interact with Python. We manually save plates in the database, and check if any plate detected is the same as present in the approved plates database.

Results and Discussion

The described datasets were used to evaluate the system's performance across a range of scenarios.

- **Parking Lot Occupancy Detection**

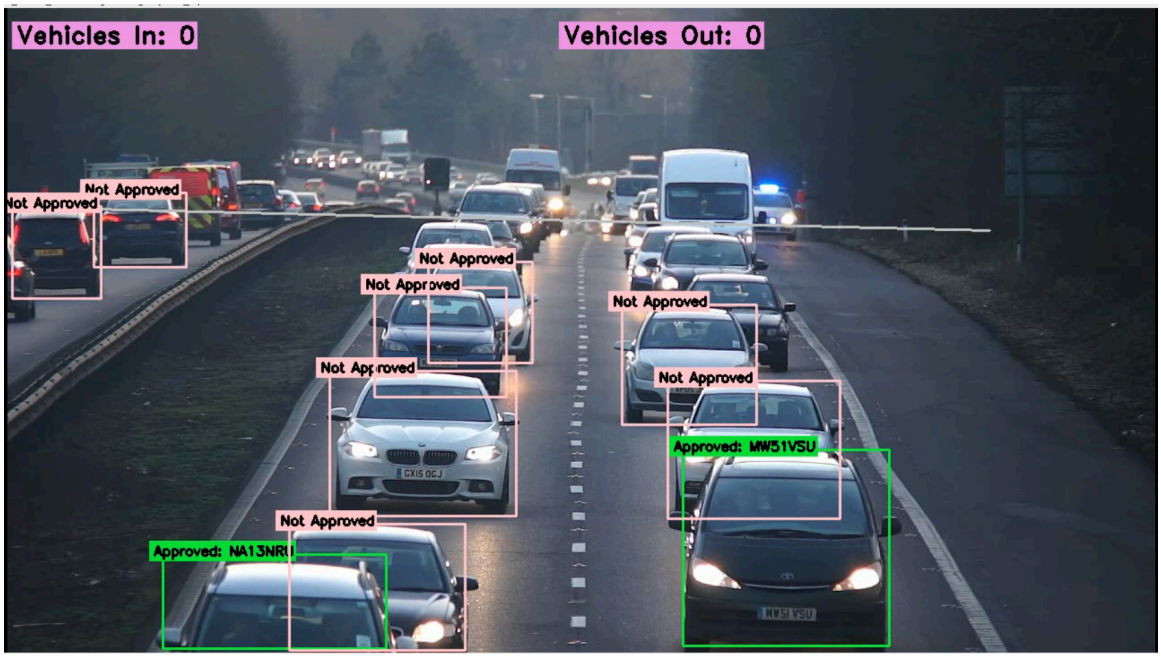
  The parking lot occupancy detection system displayed a high degree of accuracy, accurately determining in real time whether the spots were occupied or vacant. Even in difficult circumstances like dim lighting or occlusions, the YOLO nano model was able to detect cars with accuracy. With very little processing latency, the real-time performance was effective.

- **Vehicle Entry and Exit Tracking**

  The system accurately tracked vehicle movements across marked entry and exit points. The entry and exit events were logged correctly, providing valuable data for traffic analysis and security monitoring. The user-defined marking of the line proved to be flexible and easy to use, allowing for precise tracking in different video scenarios.

- **License Plate Recognition and Verification**

  The custom license plate detection model and PaddleOCR combination achieved high accuracy in recognizing license plate characters. The real-time verification against the approved license plates database was successful, with the system promptly displaying "Approved" or "Not Approved" based on the recognition results. This functionality enhances campus security by ensuring only authorized vehicles access restricted areas.



- **Inference Efficiency**

  Converting the models to OpenVINO format significantly improved inference speed and efficiency. The FP16 precision provided a good balance between model accuracy and computational efficiency, making the system suitable for deployment on edge devices with limited resources.

Conclusion

Real-time insights into vehicular activity on a college campus are provided by this project, which successfully combines edge computing and advanced computer vision techniques. Enhancing campus security are the precise detection, tracking, and verification capabilities of the system. One thing our project currently lacks is generating meaningful insights after video processing. In the future, efforts will be directed towards refining the models even more, growing the database of authorized license plates, and investigating new uses like anomaly detection and predictive analytics for school security.

References

[1]
https://universe.roboflow.com/roboflow-universe-projects/license-plate-recognition-rxg4e