Vehicle Number plate detection

S Harish - CB.EN.U4CSE21422

Shreyas Visweshwaran - CB.EN.U4CSE21455

https://github.com/FirefoxSRV/Vehicle-Number-plate-recognition

Reference Papers

Paper Title	Authors	Year	Algorithm used	Key Findings
Vehicle Detection with a Mobile Camera: Spotting Midrange, Distant, and Passing Cars	Marinus B. van Leeuwen, Frans C.A. Groen	2003	Motion Detection, Kalman Filter, Template Matching	The paper discusses a mobile camera-based system for vehicle detection, emphasizing the effectiveness of the Kalman filter in tracking moving objects. This system successfully identifies vehicles at varying distances with reasonable accuracy.
Comparative Study on Real-Time Vehicle Classification	Dilip Kumar Sharma, Rahul Pradhan, Ruchi Agrawal	2022	SVM, YOLO	YOLO outperforms SVM in real-time applications due to higher processing speed.
Vehicle Detection and Classification Using Deep Networks	Shuva Chowdhury, Shithi Chowdhury, Jeba Tahsin Ifty, Riasat Khan	2022	VGG16, VGG19, YOLOv5	YOLOv5 shows best performance among the tested models due to effective real-time processing and high accuracy.

🛠️ Lab Evaluation - 1

Problem statement and its significance

Our project aims to develop an automated system that captures vehicle images through a mobile application to detect vehicle type and gather owner information.

Registered vehicles are recognised instantly on subsequent visits, eliminating the need for re-registration and streamlining access management. This approach of ours
enhances efficiency and security by reducing manual effort and providing real-time monitoring of vehicle entry.
In an organisation, efficient management of vehicle access is crucial for security. The current manual processes of vehicle registration and monitoring are time-consuming and prone to errors, leading to delays and inefficiencies. To address these challenges, we propose the development of an automated vehicle detection and recognition system that uses the mobile-based image capture technology.

Our system will allow for the following:

Initial Registration: New vehicles entering the company premises will be registered by capturing an image of the vehicle using a mobile device. The system will identify the vehicle type and store relevant details, including the owner's information, in a secure database using the vehicle’s number plate.

Automated Recognition: On subsequent visits, the system will automatically detect the vehicle by matching the captured image with the stored template. If the vehicle is already registered, it will be granted access without the need for re-registration.

Time Efficiency and Monitoring: This system significantly reduces the time required for vehicle entry, as registered vehicles are recognised instantly. It

also enhances the security and monitoring of vehicle access, allowing company personnel to efficiently manage and track vehicle movements.

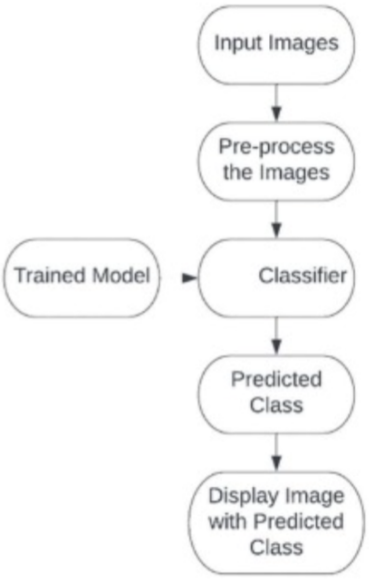
Our system will allow for the following:
Initial Registration: New vehicles entering the company premises will be registered by capturing an image of the vehicle using a mobile device. The system will identify the vehicle type and store relevant details, including the owner's information, in a secure database using the vehicle’s number plate.
Automated Recognition: On subsequent visits, the system will automatically detect the vehicle by matching the captured image with the stored template. If the vehicle is already registered, it will be granted access without the need for re-registration.
Time Efficiency and Monitoring: This system significantly reduces the time required for vehicle entry, as registered vehicles are recognised instantly. It also enhances the security and monitoring of vehicle access, allowing company personnel to efficiently manage and track vehicle movements

Dataset for our project

https://www.kaggle.com/datasets/dataclusterlabs/indian-number-plates-dataset

Existing Models

Fig. 2. VGG16 and VGG19 architecture.



Working sequences of the proposed method.

Experiment: Training and Testing from research paper

VGG16 Results:

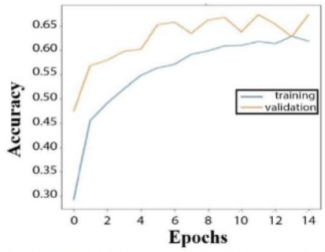


Fig. 5. Training and validation accuracies vs. epochs for the VGG16 model.

VGG19 Results:

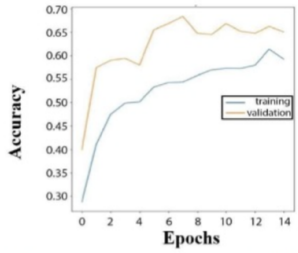


Fig. 8. Training and validation accuracy curves for the VGG19 model.

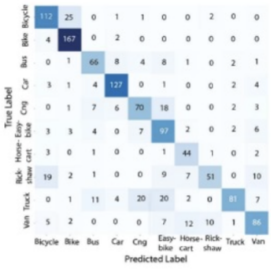


Fig. 7. Confusion matrix of VGG16 model.

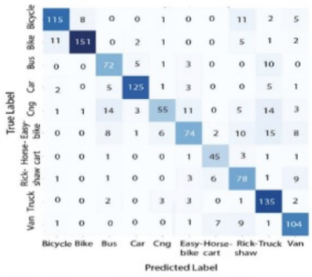


Fig. 10. Confusion matrix of the VGG19 model.

Applied Model	Accuracy	Precision	F1 score
VGG16	75.15%	75.14%	74.54%
VGG19	79.57%	79.56%	78.86%

🛠️ Lab Evaluation 2

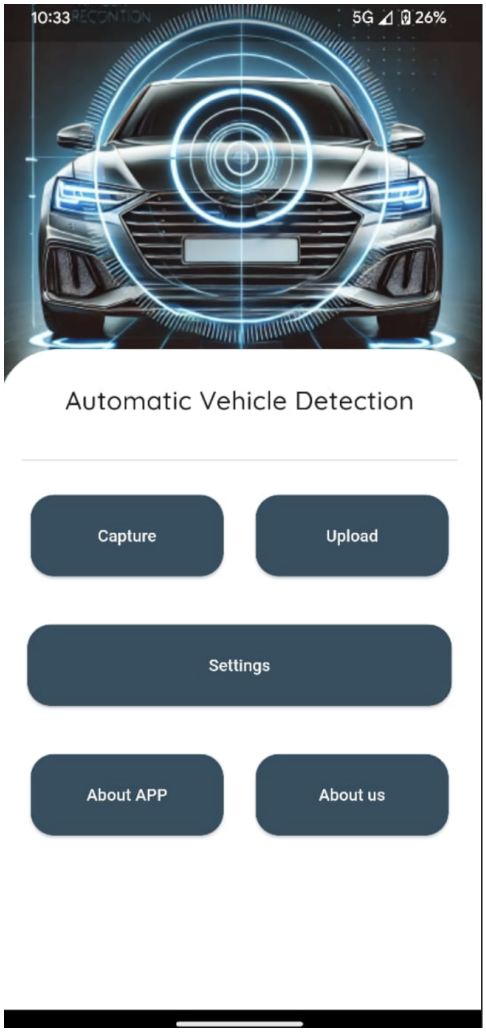
In this lab evaluation, we implemented the UI using flutter and then we used tf lite package to deploy our model into working.

Here we implemented Yolo V5 for detecting the number plate of the vehicle in which the process goes like:

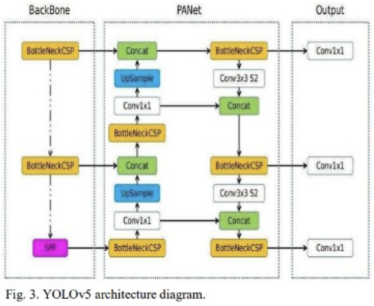
1. Input Image Processing (By capturing through mobile and then process)
2. Bounding Box Prediction
3. Class Prediction
4. Real-Time Detection

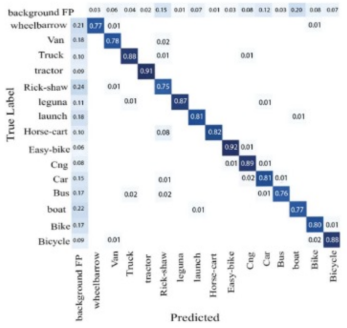
How it helps in number plate detection:
- YOLOv5 processes images in milliseconds, allowing for real-time vehicle detection.
- YOLOv5 can handle varying lighting conditions, weather changes, and occlusions that are common in real-world traffic environments.

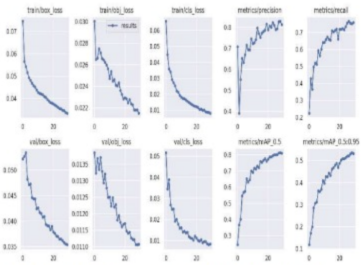
Mobile app and its implementation



New model - Yolov5 and future models



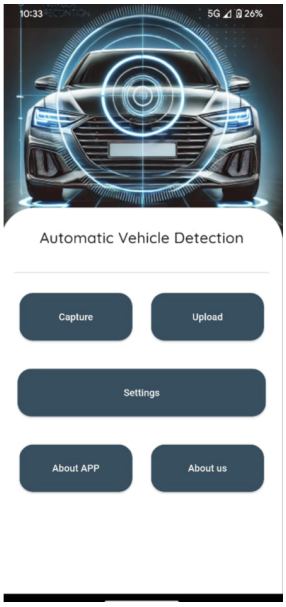


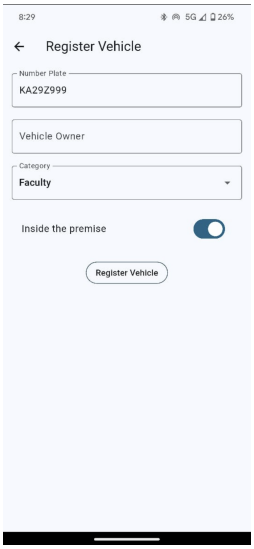


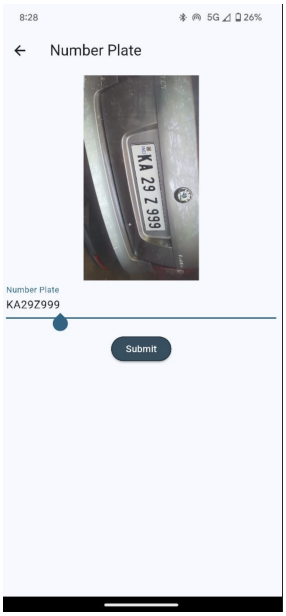
```
Fusing layers...
Model summary: 268 layers, 43614318 parameters, 0 gradients, 164.9 GFLOPs
val: Scanning /content/YOLOv8-DeepSORT-Object-Tracking/ultralytics/yolo/v8/detect/detection-1/valid/labels.cache
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 39/39 [00:27<00:0
                   all        619       2185      0.757      0.687      0.764      0.429
                  boat        619          1      0.472          1      0.995      0.796
           camping car        619         21      0.866      0.667      0.735      0.379
                   car        619       1963      0.882      0.918      0.911      0.422
            motorcycle        619         59      0.859      0.678      0.844      0.326
                 other        619          4       0.32       0.25      0.373      0.232
                pickup        619         67      0.773       0.66      0.723      0.383
                 plane        619          1       0.89          1      0.995      0.697
               tractor        619          6          1      0.482      0.737      0.275
                 truck        619         25      0.673       0.56      0.629      0.384
                   van        619         38      0.831      0.658        0.7      0.393
Speed: 0.5ms pre-process, 36.3ms inference, 0.0ms loss, 1.4ms post-process per image
```

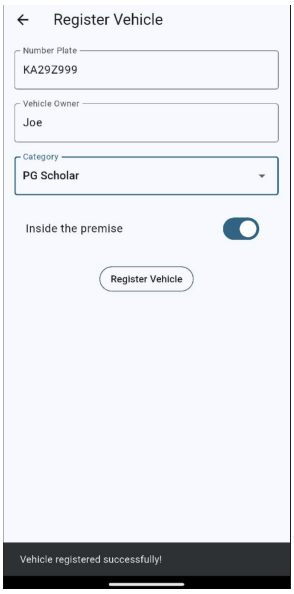
🛠️ Lab Evaluation - 3

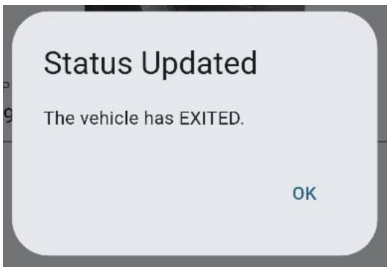
Mobile app and its implementation

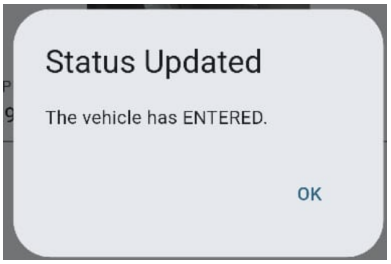




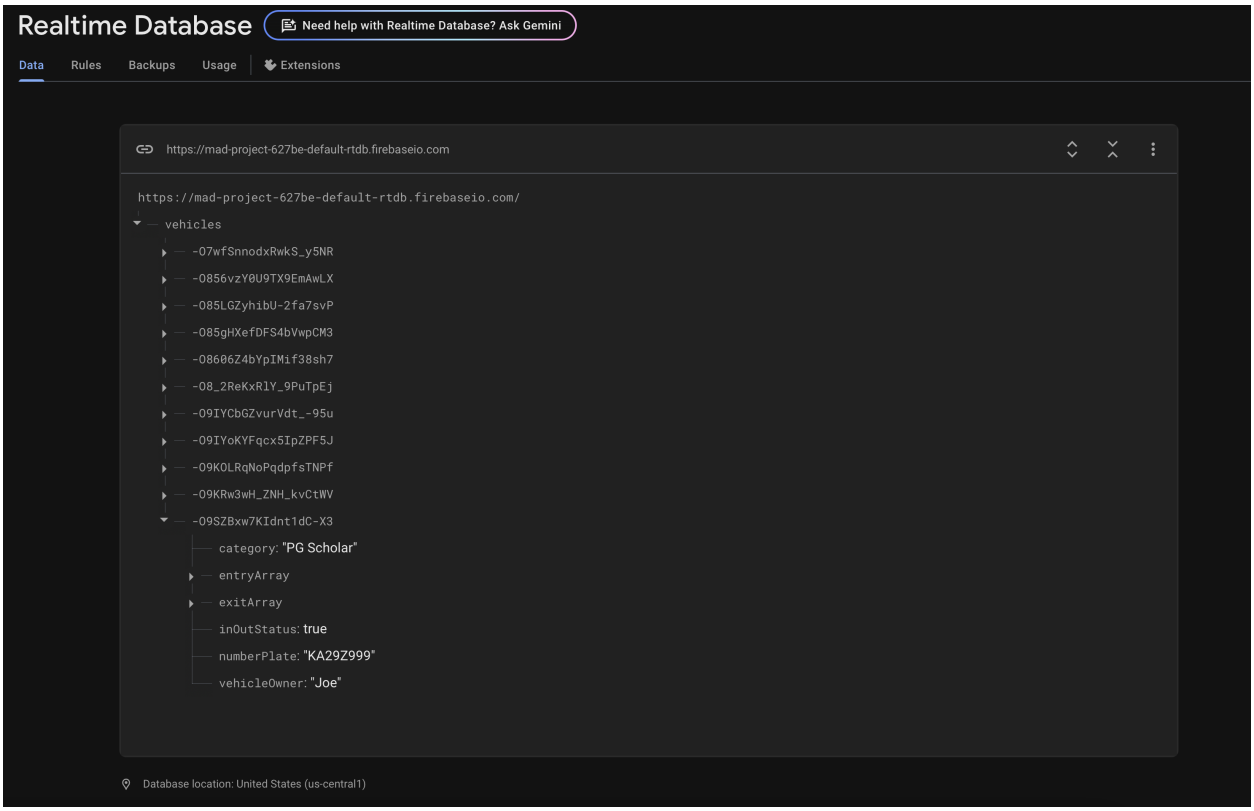




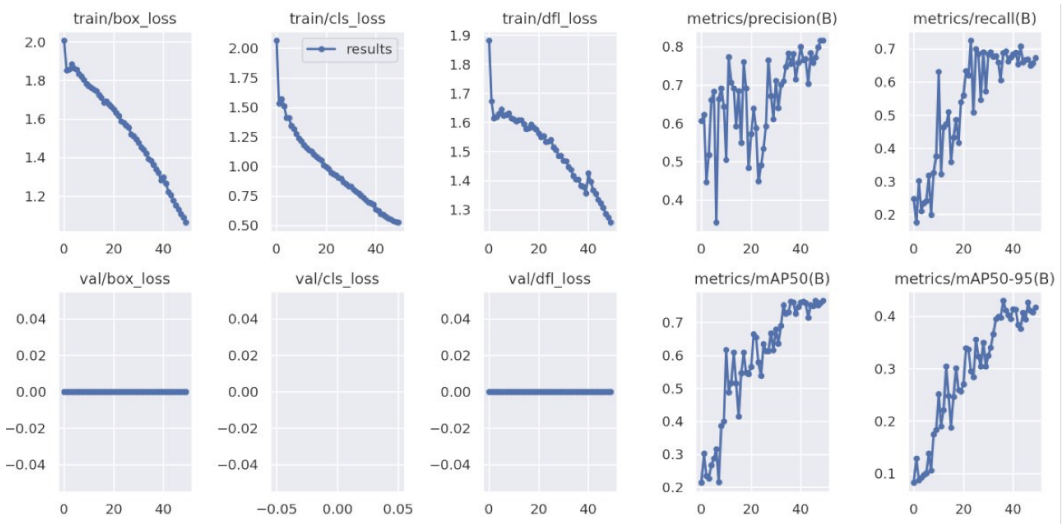


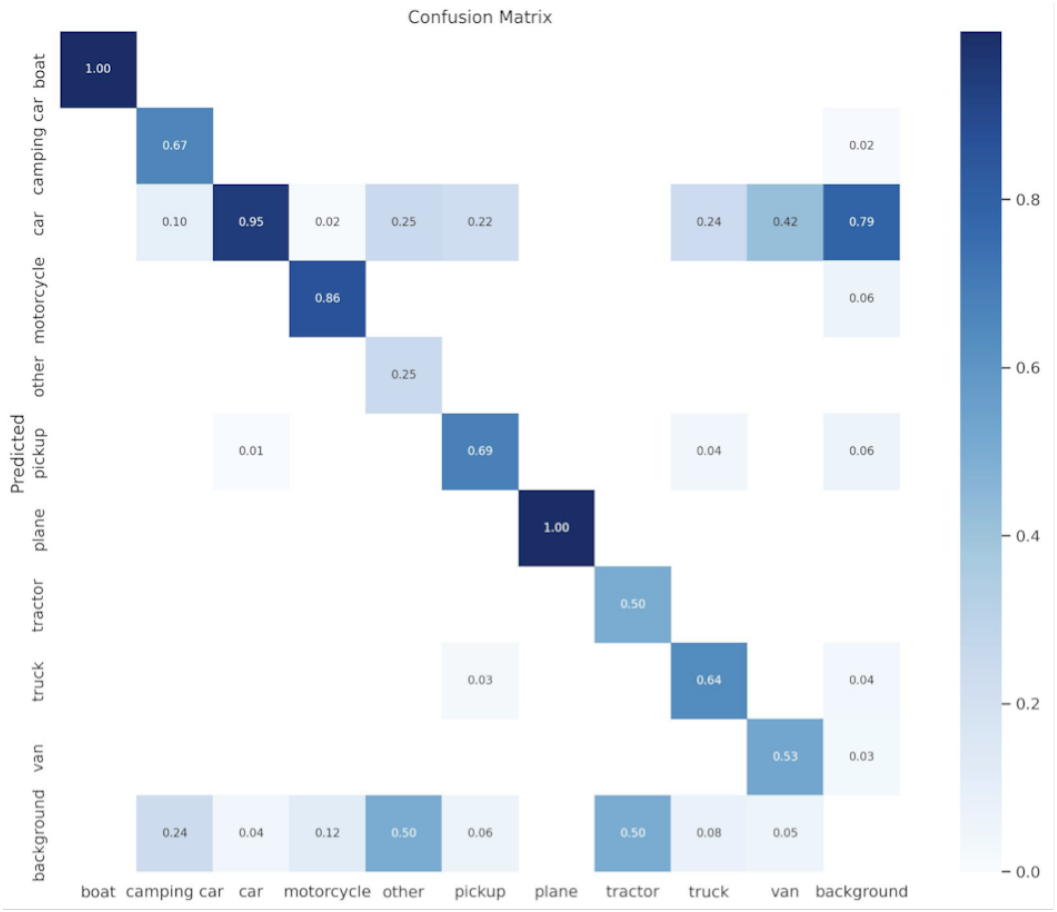


Now, these data are not left alone.....we have a database in the backend. With firebase realtime database, we are able to implement it with ease. Here is the database from firebase.google.com.



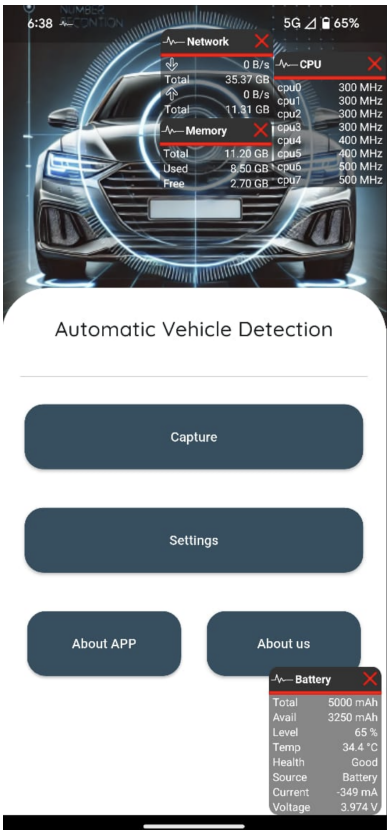
Performance metrics from our ipynb notebooks

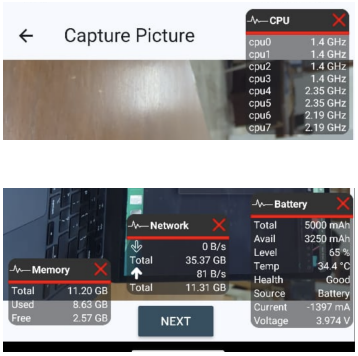


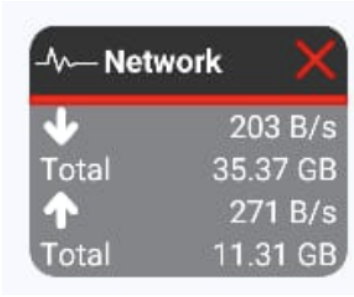


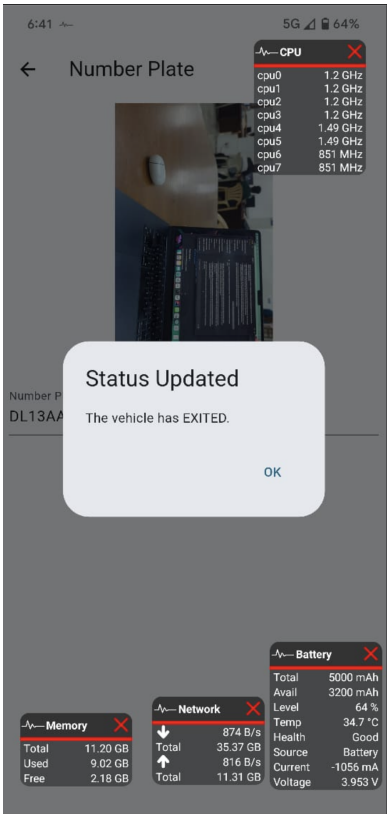
Performance metrics for the product as a whole

We have recorded the performance of the app using the android app “SysFloat”. Here are the screenshots for the same :

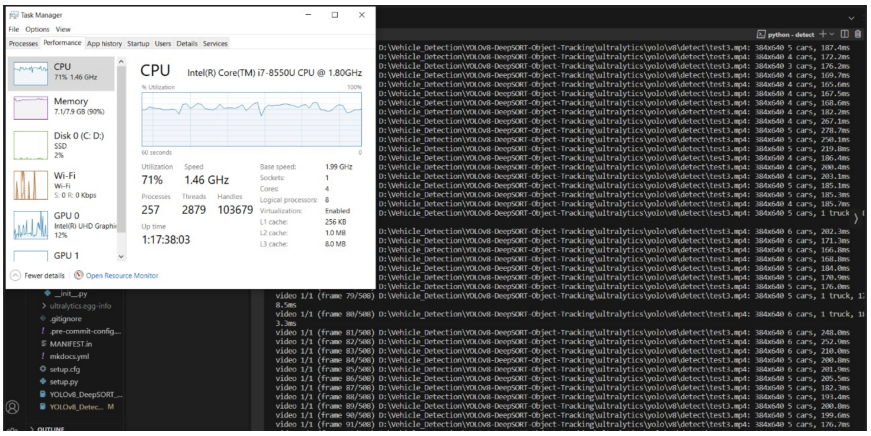


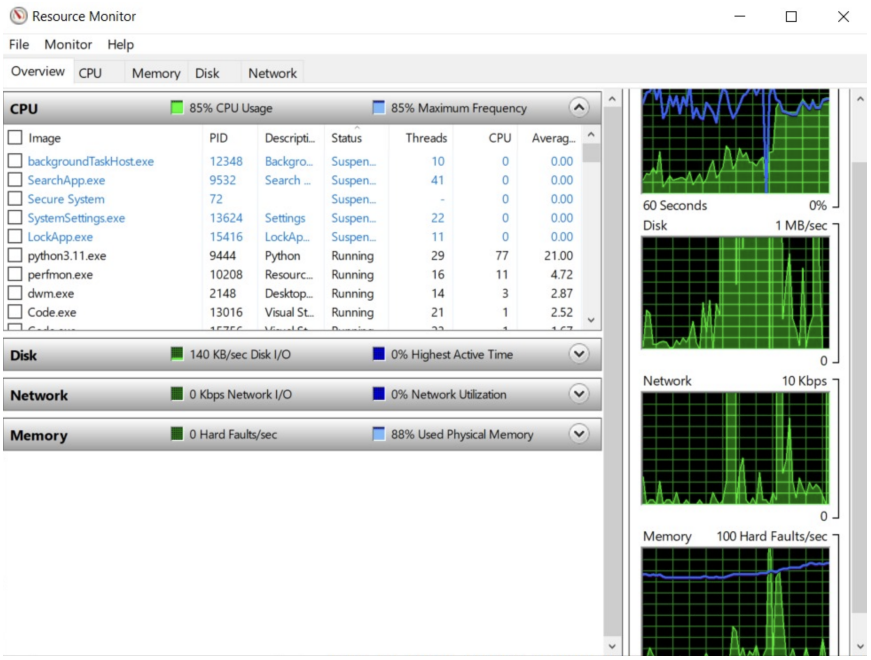






Well, we used a different AI model for our app, as yolo v8 was very CPU and Memory intensive. Hence, running edge applications with deep learning models are quite a difficult task....not for the developer, but for the user. There is also a risk of temp rising when these models run. Anyway, here is the metrics and graphs for the execution of yolov8 model when run on a windows machine.





Conclusion

Hence, with this project, we were able to learn the intricacies of app development and the integration of AI models with pickle and tensorflow lite. We have also a clear understanding of why many applications with deep learning does not exist in

the mobile era currently. With this project, our understanding on the inpainting and object recognition is immensely improved.