Comparison of Vehicle Detection Techniques

1st Pranav Raj Sowrirajan Balaji
Khoury College of Computer Sciences
Northeastern University
Boston, USA
sowrirajanbalaji.p@northeastern.edu

***Abstract*—In the rapidly advancing domain of computer vision, the detection of moving objects within video feeds plays a critical role across a spectrum of applications including security surveillance, traffic monitoring, and automated vehicular systems. This report presents a comparative analysis of multiple methodologies for vehicle detection, focusing on both traditional unsupervised learning techniques and cutting-edge supervised deep learning models. Initially, we explore unsupervised methods using OpenCV, specifically contour detection and frame differencing techniques to identify moving vehicles without the need for labeled data. Subsequently, we delve into the realm of deep learning with an examination of YOLOv4 and YOLOv8 models highlighting their architectures and innovations such as the integration of advanced backbone structures. The system aims to explain the effectiveness, computational efficiency, and application suitability of each method, providing a comprehensive guide for users selecting the optimal approach for various operational environments.**

I. Introduction

The field of computer vision has witnessed significant developments in object detection technologies, driven by both theoretical innovations and practical demands. Among these applications, vehicle detection in video streams is particularly vital due to its implications for safety and traffic management in urban environments, as well as its integration within autonomous driving systems. The challenge lies not only in accurately detecting vehicles across diverse and dynamic conditions but also in doing so in real-time with minimal computational resources.

Traditional unsupervised techniques for vehicle detection leverage the capabilities of OpenCV, utilizing methods such as frame differencing, image thresholding, contour detection and image dilation. These techniques are advantageous due to their low dependency on high-quality labeled datasets and their relative computational simplicity. However, they might struggle with high levels of accuracy in complex scenarios involving variable lighting, weather conditions, and occlusions.

On the other hand, supervised deep learning approaches, particularly the You Only Look Once (YOLO) series, have set new standards in both speed and accuracy for object detection tasks. The evolution from YOLOv3 to YOLOv4[5] and the recent YOLOv8[6] has seen continuous improvements in model architecture, training strategies and inference efficiency. This project aims to provide a detailed exploration of these methodologies, assessing their practicality across different scenarios and their integration within larger systems. Through comparative analysis, we seek to offer insights into selecting appropriate detection strategies based on specific requirements and constraints, thus aiding researchers and practitioners in the field of computer vision and automated systems.

However, to develop such applications, we need a large dataset with well-defined labels. Hence, in this project we have to chosen to train our models on the COCO and COCO8 dataset. The COCO dataset contains around 330,000 images with 80 object catergories including common objects like cars, bicycles, and animals, as well as more specific categories such as umbrellas, handbags and sports equipment. It is designed to encourage research on a wide variety of object categories and is commonly used for benchmarking computer vision models. It is an essential dataset for researchers and developers working on object detection, segmentation, and pose estimation tasks.

A. RELATED WORK

The YOLO (You Only Look Once) architecture, first revolutionized by Redmon and Farhadi[5] with the introduction of YOLOv3 in 2018, marked a pivotal shift towards faster and more accurate object detection frameworks. The subsequent iterations including YOLOv8, have continued this trajectory with Telaumbanua et al. (2023)[6] demonstrating YOLOv8's effectiveness in vehicle detection and its practical applications in traffic management. Further real-world applications were explored by Ali et al. (2023)[1], who detailed an AI-based vehicle detection system for traffic surveillance using the YOLO architecture. Their work showcases the adaptability of YOLO models in various traffic conditions. The ongoing refinement of the YOLO models for vehicle detection is captured in the 2024 study by Maleki et al.[3], which discusses the model's utility in accurately identifying vehicular objects. The YOLO models operate on the backbone of Darknet, an open-source neural network framework developed by Joseph Redmon[4], which has been pivotal to the architecture's evolution. The advancement and accessibility of the YOLO framework particularly YOLOv8, are largely attributed to the development work by Jocher et al. (2023)[2], who released it as part of the Ultralytics project.

II. Methods

A. Vehicle Detection using Classic Techniques

Our objective is to capture the coordinates of the moving object and highlight that object in the video. Consider this frame in Fig.1, We would want our model to detect the moving

Fig. 1. Example of a moving car

object in a video as illustrated in the image above. The moving car is detected and a bounding box is created surrounding the car. There are multiple techniques to solve this problem, a deep learning model can be trained for object detection or we can pick a pre-trained model and fine-tune it on our data. However, these are supervised learning approaches and they require labeled data to train the object detection model. we will focus on the unsupervised way of object detection in videos, i.e. object detection without using any labeled data. The open-source OpenCV library, known for its comprehensive set of tools for computer vision, provides robust solutions to the detection of moving objects. In this project, we examine a combination of Contour Detection and Frame Differencing that can be used to detect moving objects. Contour Detection, a method used to identify and outline objects, paired with Frame differencing, which computes absolute differences between frames to identify moving objects, creates a powerful duo for detecting moving objects in real time. This approach is practical and computationally efficient, making it suitable for applications that require quick and accurate object detection. These are the steps:

1) Frame Differencing: A video is a set of frames stacked together in the right sequence. So, when we see an object moving in a video, it means that the object is at a different location at every consecutive frame. If we assume that apart from that object nothing else moved in a pair of consecutive frames, then the pixel difference of the first frame from the second frame will highlight the pixels of the moving object, from which we can estimate the pixels and the coordinates of the moving object. This is broadly how the frame differencing method works as seen in the figure down below.

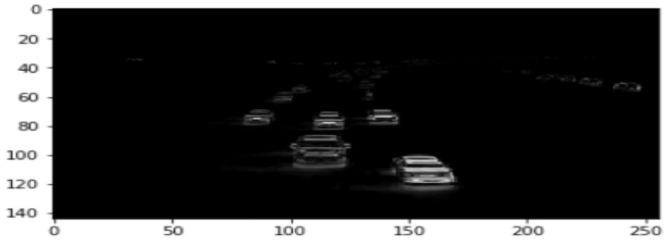

Fig. 2. Frame Differencing in OpenCV

2) Image Thresholding: In this step, the pixel values of a grayscale image are assigned one of the two values representing black and white colors based on a threshold. So, if the value of a pixel is greater than a threshold value, it is assigned one value, else it is assigned the other value. In our case, we will apply image thresholding on the output image of the frame differencing in the previous step. The resultant image can also be called as a binary image as there are only two colors in it. In the next step, we will see how to capture these highlighted regions.

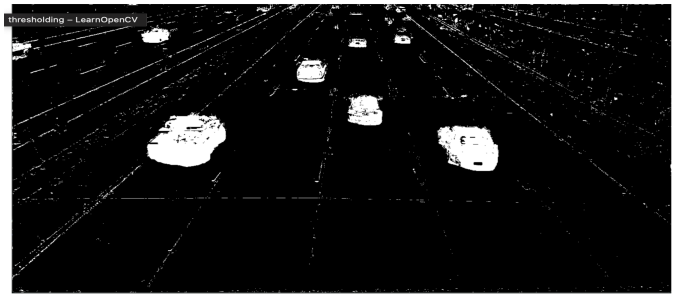

Fig. 3. Thresholding in OpenCV

3) Contour Detection: The contours are used to identify the shape of an area in the image having the same color or intensity. Contours are like boundaries around regions of interest. So, if we apply contours on the image after the thresholding step, we would get white regions that have been surrounded by grayish boundaries which are nothing but contours, from which we can easily get the coordinates of these contours. There can be multiple highlighted regions and each region will be encircled by a contour. In our case, the contour having the maximum area is the desired region. Hence, it is better to have as few contours as possible. Despite this, there will be some unnecessary fragments of the white region. To improve this, we can merge the nearby white regions to have fewer contours and for that, we can use another technique known as image dilation.

Fig. 4. Contour Detection in OpenCV

4) Image Dilation: This is a convolution operation on an image wherein a kernel (a matrix) is passed over the entire image. Here, we have only four candidate contours from which we would select the one with the largest area.

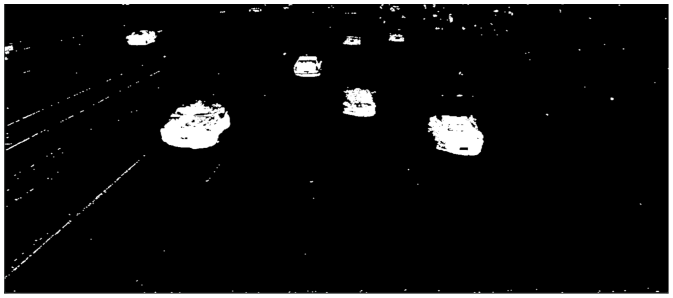

Fig. 5. Image Dilation in OpenCV

B. Vehicle Detection using YOLOv4

YOLOv4 is a real-time object detection model developed to address the limitations of previous YOLO versions like YOLOv3 and other object detection models. Unlike other convolutional neural network (CNN) based object detectors, YOLOv4 is not only applicable for recommendation systems but also for standalone process management and human input reduction. Its operation on conventional graphics processing units (GPUs) allows for mass usage at an affordable price, and it is designed to work in real-time on a conventional GPU while requiring only one such GPU for training. YOLOv4 is designed to provide the optimal balance between speed and accuracy, making it an excellent choice for many applications.

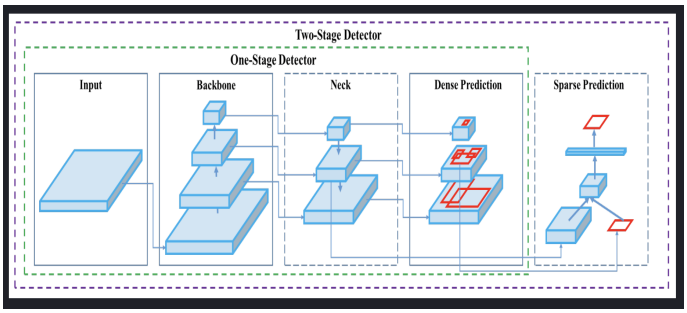

Fig. 6. YOLOv4 Architecture Diagram

YOLOv4 architecture diagram, showcasing the intricate network design of YOLOv4 including the backbone, neck, and head components, and their interconnected layers for optimal real-time object detection. YOLOv4 is a powerful and efficient object detection model that strikes a balance between speed and accuracy. YOLOv4 can be trained and used by anyone with a conventional GPU, making it accessible and practical for a wide range of applications.

1) System Architecture:

- Object Detection: The system utilizes YOLOv4-tiny, a lightweight version of the full YOLOv4 model, configured to detect multiple object classes. It operates on a neural network architecture optimized for speed and efficiency, suitable for real-time applications. The model reads from a video source, resizes the input frames and detects objects by predicting bounding boxes and class probabilities.
- Correlation Tracking: Post-detection, Dlib's correlation tracker is employed to follow the objects identified by YOLOv4-tiny across frames. The tracker updates its model based on the appearance of the object within the predicted bounding box, allowing it to handle some degree of object appearance variation due to rotation, scaling, and occlusion.
- Integration Mechanism: Each detected object is tracked by a separate correlation tracker instance. The system checks if the detected object is already being tracked by comparing the spatial overlap of newly detected bounding boxes with existing trackers using a defined threshold. If an object is not currently tracked, a new tracker is initialized.

2) Steps: : The system processes video frames in a sequential manner:

- Frame Acquisition: Frames are captured from a video source, resized, and cropped to focus on a region of interest.
- Detection: Each frame is converted into a blob and passed through the YOLOv4-tiny network to detect objects. Detected objects are filtered based on confidence scores and non-maximum suppression.
- Tracking: For each detected object, the system checks whether it matches the location of any existing trackers. If matched, the existing tracker is updated; otherwise, a new tracker is initiated.
- Tracking Quality Assessment: Trackers are periodically assessed for their tracking quality. Poor performing trackers (based on a quality threshold) are removed.
- Visualization and Output: The system overlays bounding boxes and tracking information on the original frame for display

The proposed system effectively integrates the rapid detection capabilities of YOLOv4-tiny with the robust tracking provided by Dlib's correlation tracker. This hybrid approach ensures that the system maintains high performance in tracking objects in dynamic environments where object appearances can change drastically. Future work may focus on enhancing the system's ability to handle more complex scenarios, including crowded scenes and faster object movements.

C. Vehicle Detection using YOLOv8

YOLOv8 is the latest iteration in the YOLO series of real-time object detectors, offering cutting-edge performance in terms of accuracy and speed. Building upon the advancements of previous YOLO versions, YOLOv8 introduces new features and optimizations that make it an ideal choice for various object detection tasks in a wide range of applications

1) Features:

- Advanced Backbone and Neck Architectures: YOLOv8 employs state-of-the-art backbone and neck architectures, resulting in improved feature extraction and object detection performance.
- Anchor-free Split Ultralytics Head: YOLOv8 adopts an anchor-free split Ultralytics head, which contributes to

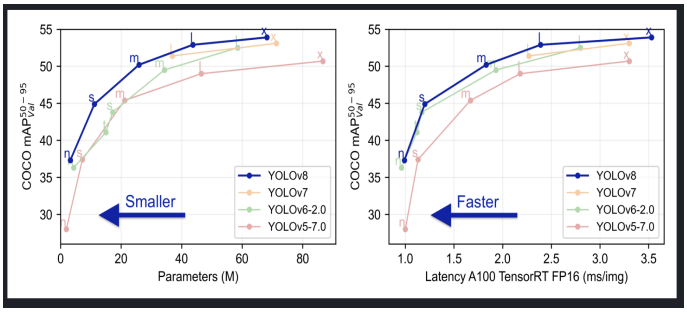

Fig. 7. YOLOv8 Performance Diagram

better accuracy and a more efficient detection process compared to anchor-based approaches.

- Optimized Accuracy-Speed Tradeoff: With a focus on maintaining an optimal balance between accuracy and speed as seen in Fig. 7, YOLOv8 is suitable for real-time object detection tasks in diverse application areas.
- Variety of Pre-trained Models: YOLOv8 offers a range of pre-trained models to cater to various tasks and performance requirements, making it easier to find the right model for your specific use case.

III. RESULTS

These are the system specifications and metrics used to evaluate the performance.

Experimenal Setup:

- Operating System: Mac OS Ventura
- IDE: Visual Studio Code
- Compiler: Cmake files with VS Code
- CPU: Apple M1 Chip with integrated GPU
- RAM: 16BG

Performance Metrics:

- Accuracy
- Precision
- Recall
- F-1 Score
- Processing Time

A. Results for Classic Techniques

The following results were obtained for classic computer vision techniques (Image Differencing, Image Thresholding, Contour Detection & Image Dilation). The system overall performed well, despite being an un-supervised approach. The system was able to detect moving objects and drawing bounding boxes over them. It achived an average frame rate of 16 FPS and a average processing time of 0.03s per frame as seen Table 1.

TABLE I
PERFORMANCE OF CLASSIC COMPUTER VISION TECHNIQUES

Average Frame Rate(fps)	Average Processing Time (s)
16.00	0.03

From Fig. 8 we can see that the approach works relatively well in real-time. But there were some problems, particularly when the object is moving fast, multiple bounding boxes are drawn for the same object due to improper segmentation. This can further be improved with the introduction of more sophisticated approaches.

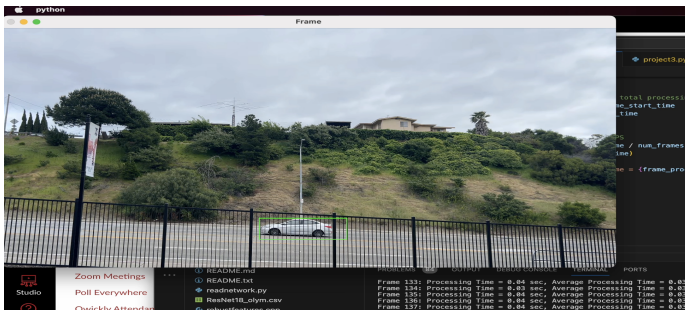

Fig. 8. Realtime Detection of Classic Techniques

B. Results for YOLOv4

This approach of using a pre-trained YOLOv4 model performed much better than the classic computer vision techniques, being able to detect and track objects across multiple frames. We can use the YOLOv4 in command line as seen in Fig. 9 & Fig. 10. The YOLOv4 model is able to get the accurate prediction with a very high confidence level.

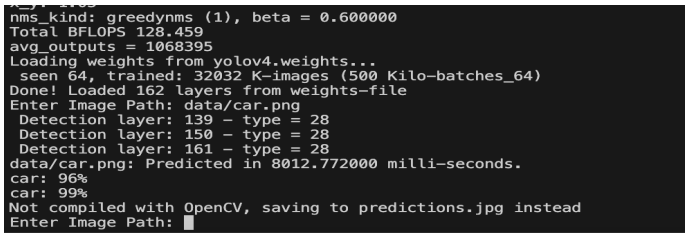

Fig. 9. Command Line Detection for YOLOv4

Fig. 10. Predicted Detection of YOLOv4

The system also implements a python script of performing live vehicle detection which performs quite well in dynamic environments as seen in Fig. 11. Few drawbacks of this model is that the model also picks up phantom objects (objects that have left the screen) and draws bounding boxes around them. This can be improved with implementing even stronger models such as YOLOv8.

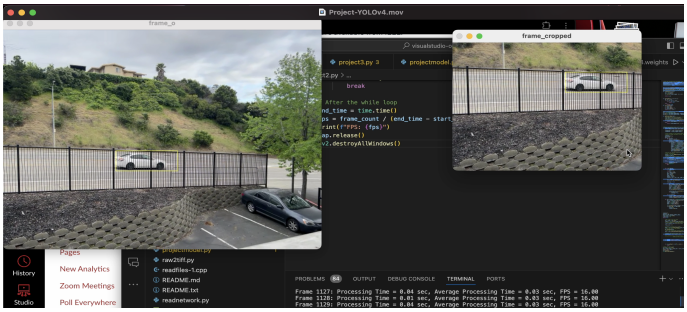

Fig. 11. Realtime Detection of YOLOv4

C. Results for YOLOv8

The YOLOv8 model has by far the best-results out of all the approaches we have seen in this project. This model is able to capture moving objects in real-time in a robust and precise manner. The system uses the pre-trained YOLOv8 segmentation model to detect the vehicles in the frame as seen in Fig. 12.

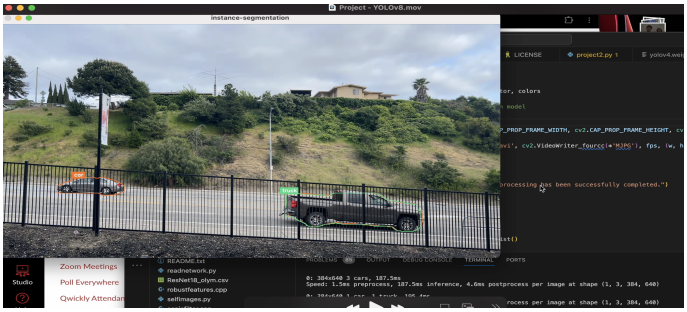

Fig. 12. Real-time Perfomance of YOLOv8

The model can be validated using a simple command-line argument, which will instantly generate the charts for F-1 score, Precision & Recall charts. We can see that in the following Figures 13, 14 & 15 respectively.

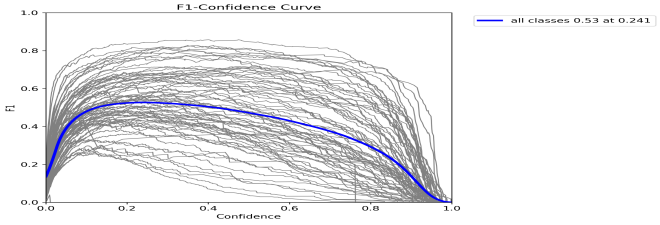

Fig. 13. F-1 Score Chart for YOLOv8

In Fig. 13, the peak of the blue line indicates the highest F1 score that the model achieves across all classes. The label on the graph, ”all classes 0.53 at 0.241,” suggests that the highest average F1 score is 0.53, achieved at a confidence threshold of approximately 0.241. This means that if the model predicts a class with a confidence level above 0.241, it is optimizing for the best balance between precision and recall across all classes.

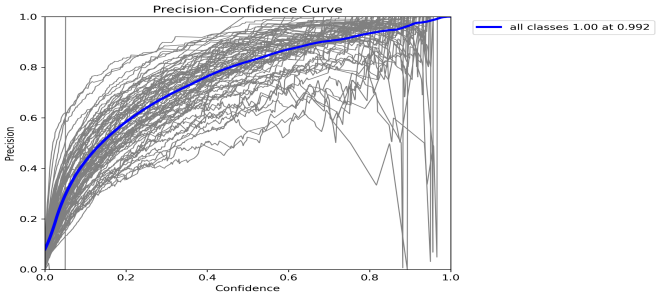

Fig. 14. Precision Score Chart for YOLOv8

In Fig. 14, Each of the numerous light lines on the chart signifies the precision at different confidence levels for individual classes the model can identify. The bold blue line represents the precision across all classes. The annotation ”all classes 1.00 at 0.992” signifies that the model achieves perfect precision (1.00) when the confidence threshold is set to 0.992. This means that almost every prediction the model makes with greater than or equal to 99.2 confidence is correct. However, this might also imply that the model is very selective, potentially at the cost of missing some true positives (lower recall), because it’s only this precise at such a high confidence level.

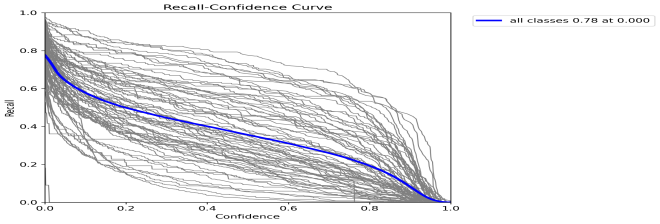

Fig. 15. Recall Score Chart for YOLOv8

In Fig. 15, each line corresponds to the recall at various confidence thresholds for a specific class that the model detects. The label ”all classes 0.78 at 0.000” suggests that the model, when not using any confidence threshold (0.000), can identify 78 percent of all positive cases across all classes. This is an indication of the model’s ability to find true positives, but without any threshold, the model is likely also capturing many false positives, which is not indicated by recall alone but would affect precision.

IV. CONCLUSION

In conclusion, this project provides an extensive comparative analysis of various methodologies for vehicle detection within video streams, a critical component in the development of advanced computer vision applications. Through rigorous examination of both traditional unsupervised techniques and cutting-edge supervised deep learning models, particularly the YOLO series, this research elucidates the strengths and limitations of each approach in real-world scenarios. Traditional methods, such as contour detection and frame differencing, offer computational simplicity and low dependency

on labeled data, making them suitable for scenarios where rapid deployment is necessary. However, their effectiveness is limited under complex environmental conditions. In contrast, the YOLO models, especially YOLOv4 and YOLOv8, demonstrate superior accuracy and speed, benefiting from ongoing advancements in model architecture and training strategies. This report not only highlights the technological progression in vehicle detection but also serves as a guide for selecting the appropriate method based on specific operational needs and constraints.

V. FUTURE WORK

The scope of this research opens several pathways for future investigations and developments within the field of vehicle detection in computer vision. Future work should focus on enhancing object recognition under variable conditions such as different weather and lighting, integrating the strengths of both unsupervised and supervised learning to create hybrid models that combine the resource efficiency of unsupervised techniques with the accuracy of supervised models like YOLOv8. There is also a pressing need for optimizing computational efficiency to facilitate real-time processing and adapting these models for deployment on edge devices to support real-time applications in IoT and smart city infrastructure. Extending detection capabilities to include a broader range of vehicle types and other related objects, improving tracking algorithms to handle high-speed movements or densely packed environments, and tailoring models to specific applications such as automated toll collection or traffic rule enforcement could significantly enhance the practical utility and effectiveness of vehicle detection systems. These advancements will not only expand the capabilities of computer vision but also open new avenues for innovation in automated systems.

ACKNOWLEDGMENT

I would like to extend my sincere thanks to Dr. Bruce Maxwell for his exceptional guidance throughout the course CS5330: Pattern Recognition and Computer Vision. His clear explanations and expert insight have greatly demystified the complexities of this subject for me. A heartfelt thank you also goes to the Teaching Assistants for their steadfast support and dedication. Their readiness to clarify doubts and provide thoughtful feedback has been crucial to my learning process and the successful completion of this report. Engaging with such intricate and fascinating topics has reshaped my approach to research. For their encouragement and for fostering a truly enriching educational experience, my gratitude to Dr. Maxwell and the TA's is profound.

REFERENCES

[1] Asad Ali, Manal Imran Sheikh, Awais Ashraf, and Adil Ali Raja. Development of real time ai-based vehicle detection system for traffic surveillance. In *2023 International Conference on Communication, Computing and Digital Systems (C-CODE)*, pages 1–6, 2023.

[2] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics yolov8, 2023.

[3] Pouria Maleki, Abbas Ramazani, Hassan Khotanlou, Sina Ojaghi, Seyed Milad Mousavi, Alexey Kalinin, and Amir Mosavi. Object detection for vehicles with yolo. pages 000343–000350, 01 2024.

[4] Joseph Redmon. Darknet: Open source neural networks in c. http://pjreddie.com/darknet/, 2013–2016.

[5] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018.

[6] Agustritus Telaumbanua, Tri Larosa, Panji Pratama, Ra'uf Fauza, and Amir Husein. Vehicle detection and identification using computer vision technology with the utilization of the yolov8 deep learning method. *sinkron*, 8:2150–2157, 10 2023.