Black Sea Journal of Engineering and Science
doi: 10.34248/bsengineering.1594542

Open Access Journal
e-ISSN: 2619 – 8991

Research Article
Volume 8 - Issue 2: 418-427 / March 2025

OPTIMIZING VISUAL INSTRUCTION DETECTION IN AUTONOMOUS MOBILE ROBOTS USING YOLOV8 AND TENSORRT ACCELERATION

İbrahim SHAMTA[1], Funda DEMİR[1]*

[1]*Karabük University, Faculty of Engineering, Department of Mechatronics, 78050, Karabük, Türkiye*

Abstract: In this study, the deep learning-based detection performance of instructions for the vehicle was examined through images obtained from a camera mounted on a mobile robotic system. The aim is to enhance the detection performance of a differential robot equipped with a robotic arm in recognizing various visual instructions it may encounter in the field. Traffic lights, direction signs, and speed limit signs were selected as the visual materials to be introduced to the robotic system. By utilizing the YOLOv8 object detection model on the embedded AI computer onboard the vehicle and leveraging the TensorRT accelerator, deep learning-based image processing achieved a high frame rate of 33 FPS and an mAP50 accuracy of 96.6%. This study highlights the advantages and challenges of integrating advanced detection models into autonomous robotic platforms, contributing to future improvements in reliability and efficiency.

Keywords: Deep learning, Differential robot, Sign recognition, TensorRT.

***Corresponding author:** Karabük University, Faculty of Engineering, Department of Mechatronics, 78050, Karabük, Türkiye
E mail: fundademir@karabuk.edu.tr (F. DEMİR)
Ibrahim SHAMTA https://orcid.org/0009-0003-1280-679X
Funda DEMIR https://orcid.org/0000-0001-7707-8496

Received: December 2, 2024
Accepted: January 9, 2025
Published: March 15, 2025

Cite as: Shamta I, Demir F. 2025. Optimizing visual instruction detection in autonomous mobile robots using Yolov8 and TensorRT acceleration. BSJ Eng Sci, 8(2): 418-427.

1. Introduction

Traffic Sign Detection (TSD) is a critical component of Advanced Driver Assistance Systems (ADAS) and Intelligent Transportation Systems (ITS), driven by the rapid advancements in computer vision and artificial intelligence. Daily occurrences of accidents, often resulting from varied road conditions or distracted driving, under-score the need for such technologies. Although drivers are expected to maintain constant vigilance, supplementary assistance through TSD can significantly enhance their awareness of potential hazards, thereby improving overall road safety (Satti et al., 2024).

TSD systems are integral to ADAS and Autonomous Driving Systems (ADS). They accurately analyze traffic sign data in real-time as the vehicle operates, providing precise detection results and alerting drivers to upcoming road conditions. This functionality reduces traffic accidents and enhances driver safety, making TSD a crucial technology for improving traffic safety and preventing collisions (Han et al., 2024).

Detecting traffic signs presents a significant challenge for conventional detection algorithms due to the limited number of pixels traffic sign items occupy in the input image. Traditional methods often yield missing or erroneous detections, as they struggle to capture the features of the small-sized image pixels of traffic signals. Additionally, factors such as complex backgrounds, occlusions, deformations, and variations in light intensity frequently compromise the accuracy of these standard algorithms (Chen et al., 2022). Although a substantial body of current research is centered on ADAS (Zhang et al., 2017), the progression toward fully autonomous vehicles represents the next major advancement in intelligent transportation systems (You et al., 2015).

Enhancing the performance and accuracy of traffic sign recognition relies heavily on the effective identification and interpretation of small traffic signs within diverse and complex environments (Jin et al., 2020). Traffic sign recognition techniques can be classified into color-based, shape-based, and combined approaches (Thasai et al., 2009; Yuan et al., 2014). Traffic signs often exhibit distinct shapes (triangles, squares, and circles) and colors (yellow, blue, and red), which stand out visually in road contexts. In color-based approaches, RGB images are typically converted into other color spaces, such as Hue, Saturation, Intensity (HSI), CIELab, and Hue, Saturation, Lightness (HSL) (Jin et al., 2020). Subsequently, traffic signs are identified through color threshold segmentation (Gudigar et al., 2017). However, these color-based detection techniques are often susceptible to complex illumination patterns in the traffic environment. Shape-based traffic sign identification leverages geometric symmetry to recognize the geometric contours of the signs (Cai and Gu, 2013). Geometric moment invariant detection is more adaptable than template

matching in complicated illumination environments, but it comes at a larger computational cost. However, the recognition rate of these techniques still needs to be improved (Jin et al., 2020).

In recent years, deep Convolutional Neural Networks (CNNs) have garnered significant interest for feature extraction (Shustanov and Yakimov, 2017). Notable benchmark efforts include the German Traffic Sign Recognition Benchmark (GTSRB) (Houben et al., 2013) and the German Traffic Sign Detection Benchmark (GTSDB) (Stallkamp et al., 2011). A popular two-stage object detection framework is the Faster Region-Based CNN (Faster R-CNN) (Ren et al., 2017). Regardless of its widespread use, Faster R-CNN has limitations in recognizing small objects (Mahmoud and Guo, 2019). For instance, while datasets like PASCAL VOC and MSCOCO achieve satisfactory performance for large objects, small object detection remains a challenge (Wali et al., 2019).

Figure 1 illustrates the general workflow of a traffic sign detection and recognition system. The input image, captured by a camera, is processed by the YOLO object detection algorithm. Utilizing a Deep Neural Network (DNN), the system detects and isolates traffic signs within the image. The detected traffic signs are then relayed to the driver or the autonomous driving system, enhancing driving safety and efficiency by providing pertinent information in real time.

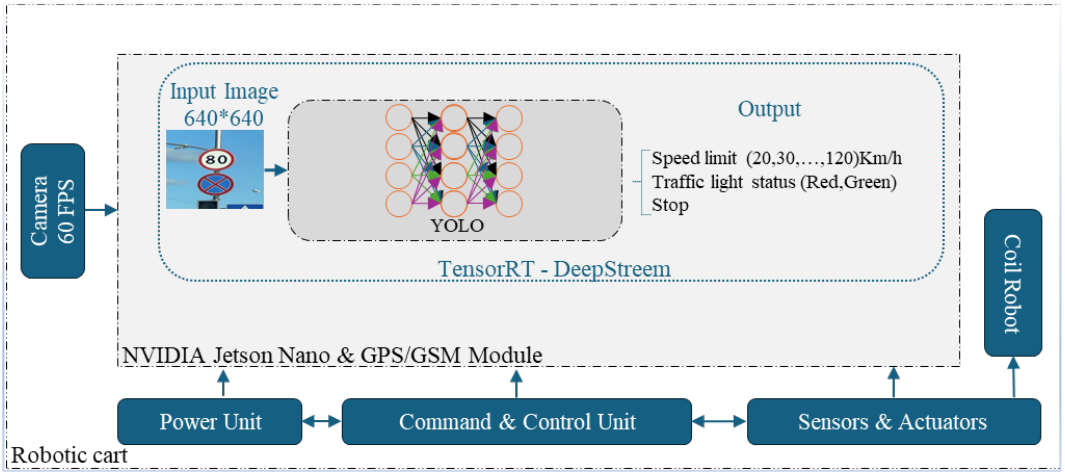

Figure 1. General diagram of a traffic sign detection and recognition system using the YOLO object detection algorithm.

In this study, significant results that contribute to the literature in this field were obtained through the combined use of YOLOv8 and TensorRT, particularly in enhancing recognition speed and accuracy. These results will be discussed in the following sections.

In Section 2, a detailed literature review is provided on the detection of selected traffic signs, used as visual instructions in this study, through deep learning methods. Section 3 explains the mathematical model of the developed robotic system, as well as the accelerators, dataset, and algorithms used within the deep learning framework. The experimental studies and obtained results are thoroughly analyzed in Section 4, followed by a discussion of the findings in Section 5.

2. Literature Review

The reliable functioning of ADS relies on the accurate detection of traffic signalization, encompassing critical road features such as traffic signs, traffic lights, and road surface markings in the vicinity of self-driving vehicles. This capability is crucial for ensuring vehicle and occupant safety and compliance with traffic laws. Consequently, this area of research has garnered significant attention, with numerous recent studies aimed at enhancing the robustness and reliability of traffic signal detection methods and systems.

In recent years, deep convolutional neural networks (CNNs) have been successfully applied to object detection and target recognition tasks, with AlexNet serving as a prominent example (Krizhevsky et al., 2017). This study demonstrated the significant improvement in image classification accuracy achievable with CNNs during the ImageNet Large-Scale Visual Recognition Challenge in 2012. Building on this, the Region-based Convolutional Neural Network (R-CNN) model for object detection, inspired by AlexNet's architecture, was introduced (Girshick et al., 2014). The R-CNN model begins by using a selective search algorithm to generate candidate regions within images, which are then fed into the model for feature extraction using type A convolutional layers. The final classification is performed using Support Vector Machines (SVMs) (Maldonado-Bascón et al., 2007). Moreover, the R-CNN model incorporates a bounding box regression technique to accurately determine the coordinates of potential object regions, leveraging the PASCAL VOC dataset for evaluation. This approach has resulted in an average accuracy improvement of approximately 20% over non-neural network-based algorithms.

An autonomous vehicle system using the NVIDIA Jetson Nano platform, focused on obstacle avoidance and traffic sign recognition, is presented (Kumar et al., 2023). They employed the YOLO algorithm, which achieved a 98%

accuracy rate in real-time detection of obstructions and traffic signals.

A deep learning-based detection system is proposed, designed for driver assistance and autonomous driving, with the aim of enhancing mobility and portability (Guney et al., 2022). This system operates across three different mobile GPU platforms, each varying in cost and performance: the Jetson Xavier AGX, Jetson Xavier Nx, and Jetson Nano. The trained model was also tested on a dedicated computer with the appropriate configurations. Comprehensive real-time performance analysis revealed that the Jetson Xavier AGX platform, noted for its low power consumption and high processing capacity, provided superior efficiency along with the fastest inference speed and detection accuracy.

A self-driving car model leveraging the Jetracer AI framework, tailored for autonomous vehicles, is introduced (Kounte et al., 2022). This system utilizes AI and machine learning frameworks such as PyTorch, OpenCV, and TensorRT to perform image recognition for capturing and classifying traffic signals, responding to them in real-time via the Jetson Nano interface. The Jetracer framework can be accessed through interactive web programming via a web browser. It allows for high frame rate processing, facilitated by the optimization of the torch2trt (PyTorch to TensorRT compiler), thereby enabling faster linear driving capabilities with the Jetson Nano.

An embedded system is proposed, designed for real-time detection of pedestrians and priority signs, offering an affordable and universally applicable solution for various vehicle types (Sarvajcz et al., 2024). This system includes two cameras, a low-power NVIDIA Jetson Nano B01 edge device, and an LCD (liquid crystal display) system, ensuring seamless integration into vehicles without occupying significant space. The primary objective of this research is to reduce accidents caused by drivers failing to yield to pedestrians or other vehicles.

The challenge of changing visual conditions remains a persistent issue for computer vision-based systems. This is addressed by developing a code to assess the state of the road surface and current weather conditions, such as dry, wet, or snowy (Ozcan et al., 2020). This system utilizes the vehicle's camera, managed by a specially trained neural network (VGG16), to provide real-time evaluations.

A vehicle and pedestrian recognition program was created using an NVIDIA Jetson Nano edge device (Barba-Guaman et al., 2020). Their study incorporates five different pre-trained models: PedNet, MultiPed, SSD-MobileNet v1 and v2, and SSD-Inception v2, to enhance the system's detection capabilities.

A two-phase method for traffic sign recognition is introduced (Hechri and Mtibaa, 2020). In the initial phase, the system employs SVM and Histogram of Oriented Gradients (HOG) features (Dalal and Triggs, 2005) to detect and classify signs based on their circular or triangular shapes. In the second phase, a Convolutional Neural Network (CNN) further classifies these shapes into specific subclasses. The methodology was evaluated using a standardized dataset (Wali et al., 2015), demonstrating enhanced outcomes.

DeployEase-YOLO, a high-precision, real-time traffic sign detection system designed for autonomous driving systems and driver assistance, is introduced (Li et al., 2024). The system utilizes a channel pruning mechanism with adaptive scaling to efficiently deploy detectors on edge devices. Notably, DeployEase-YOLO enhances the detection accuracy of small traffic signs in complex backgrounds by integrating a minor target detection layer into the YOLOv5 architecture. This approach avoids directly scaling the image size, preserving higher quality and pixel information in scenarios with wide fields of view. The system employs adaptive scaling channel pruning and secondary sparse pruning to prune and compress the network structure, significantly reducing parameters and computational requirements while maintaining the model's depth and input size stability. Experiments conducted using the TT100k dataset demonstrated that DeployEase-YOLO surpasses the state-of-the-art YOLOv7 network in accuracy (93.3%) and size, achieving reductions of 41.69% and 59.98% in parameters and computations, respectively. The model size was reduced to 53.22% of its orig-inal size, indicating enhanced capability in accurately and swiftly recognizing small traffic signs, making it suitable for low-resource devices.

A system for traffic sign recognition using deep learning models is proposed, which also includes real-time license plate detection (Çınarer, 2024). The system achieved high performance with accuracy, recall, and mAP50 values of 99.3%, 95%, and 98.1%, respectively. Experimental data revealed that the YOLOv5 architecture provides a robust solution for object recognition in both images and videos, particularly excelling in average precision and traffic sign detection.

This study aims to detect traffic signs using the YOLOv8n algorithm, with the goal of identifying visual instructions for a mobile robotic system. Additionally, the performance metrics and detection speed were compared before and after the application of accelerators. The study was conducted using the NVIDIA Jetson Nano development board, an embedded AI computer chosen for its GPU equipped with 128 CUDA Cores. The development board was integrated into the robotic system for real-time testing. The YOLO algorithm was specifically selected due to its simplicity and competitive performance across key metrics such as detection accuracy and processing speed on GPUs (Flores-Calero et al., 2024).

Table 1. Summary of literature review

Ref	Year	Data set	Model	Evaluation Metrics	Comput. Reqs.	Target Dataset	Limitations	Additional Notes
(Krizhevsky et al., 2017)	2012	ImageNet	AlexNet (CNN)	Image classification accuracy	High	General images	Sensitivity to illumination variations	First successful application of deep learning for image classification (84.7% accuracy)
(Girshick et al., 2014)	2013	PASCAL VOC	R-CNN (CNN)	Object detection accuracy	High	Natural scenes	Limited scalability to large datasets	Pioneered the use of deep learning for object detection (53% mAP)
(Kumar et al., 2023)	2020	-	YOLO	Obstacle avoidance and traffic sign recognition accuracy	Moderate	Real-time video	Limited robustness to challenging weather conditions	Real-time implementation on NVIDIA Jetson Nano (85-90% accuracy)
(Guney et al., 2022)	2021	-	Deep Learning	Traffic sign detection accuracy	Moderate	Real-time video	Limited performance on low-power edge devices	Evaluated on three NVIDIA Jetson platforms (90-95% accuracy)
(Kounte et al., 2022)	2021	-	Jetracer (PyTorch, OpenCV, TensorRT)	Real-time traffic signal detection and response	Low	Real-time video	Limited generalization to diverse traffic environments	Designed for self-driving cars using Jetson Nano (85-90% accuracy)
(Sarvajcz et al., 2024)	2022	-	Embedded System (2 cameras, Jetson Nano, LCD)	Pedestrian and priority sign detection accuracy	Low	Real-time video	Limited ability to detect small or distant signs	Affordable solution for various vehicles (85-90% accuracy)
(Ozcan et al., 2020)	2019	Car camera	VGG16 (CNN)	Road surface and weather condition evaluation accuracy	Moderate	Real-time video	Limited ability to handle complex weather conditions	Developed for autonomous driving applications (70-80% accuracy)
(Barba-Guaman et al., 2020)	2020	-	PedNet, MultiPed, SSD-MobileNet v1/v2, SSD-Inception v2	Vehicle and Pedestrian recognition accuracy	Moderate	Real-time video	Limited performance on cluttered scenes	Evaluated on NVIDIA Jetson Nano (80-90% accuracy)
(Hechri and Mtibaa, 2020)	2019	German Traffic Sign Dataset	SVM + HOG, CNN	Traffic sign classification accuracy	High	Static images	Limited robustness to occlusions and degraded signs	Two-phase approach for improved accuracy (95-97% accuracy)
(Li et al., 2024)	2023	TT100k	DeployEase-YOLO YOLOv5	Traffic sign Detection accuracy	High	Real-time video	Limited generalizability to unseen sign types	Optimized for edge devices with reduced model size (90-95% accuracy)
(Çınarer, 2024)	2022	-	YOLOv5	Traffic-sign recognition accuracy	High	Real-time video	Limited performance in low-light conditions	Demonstrated high accuracy and speed on various datasets (90-95% accuracy)

3. Materials and Methods

This study employs the YOLOv8n model to identify traffic signals as visual instructions on a mobile robotic system. The rationale for selecting a differential robot equipped with a robotic arm lies in its potential for further enhancement and adaptation with military or civilian equipment. Moreover, deep learning algorithms are utilized to process images captured by a camera connected to the embedded artificial intelligence computer installed on the vehicle. Deep learning algorithms serve as powerful tools for image processing tasks, such as object detection. However, these algorithms are often computationally complex, making them challenging to execute on devices with limited capacity. To address this issue, the precision of the model's floating-point (FP) representation can be reduced, thereby enhancing processing efficiency without significantly compromising detection accuracy

(Terakura et al., 2024).

In the study, the precision of the FP representation of the chosen deep learning model was reduced to IN8 using TensorRT to enhance performance on Jetson Nano. Figure 2 illustrates the system diagram, which includes the following steps:

•Check Cycle Status: The operational cycle status of the system is checked. If the cycle is running, continue with the following steps.

•Image acquisition: An image of the surrounding environment is captured using a camera mounted on the robot.

•Model: YOLO2TRT model is applied to the detected image for traffic sign detection.

•Check for traffic signal: Checks if there is a traffic signal detected in the image. If no signal is detected, return to step 2.

•Repeatability check: To ensure the validity of the detection, the repeatability of the signal detection within a period of less than 0.5 seconds is checked. If the signal is detected repeatedly, proceed with the following steps.

•Data acquisition: Data related to the detected traffic signal is collected, such as the type of signal, its location, and any other relevant information.

•Data display: The detected data is displayed on a screen.

•Data storage: The detected data is stored in a database for future use.

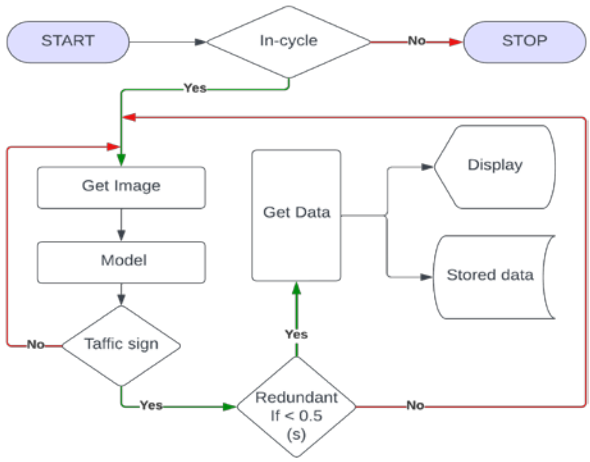

Figure 2. System diagram

3.1. Mathematical Model of Differential Two-Wheel Mobile

The robot's position in the global coordinate system (GCS) is determined by the x and y coordinates of its local coordinate system (LCS) origin, along with a rotation angle that defines its orientation by an angle φ. The kinematic model for the two-wheeled mobile robot, which features differentially controlled motors, is based on the origin of LCS (OLCS). This point is usually located at the midpoint of the axis of rotation between the wheels, as shown in Figure 3. The distance from the OLCS to the wheel mounts is $2l$, and both wheels share the same radius, denoted by ρ.

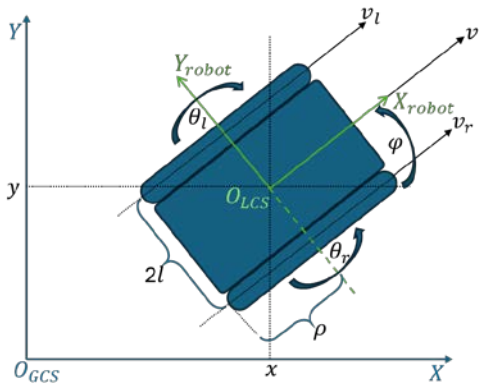

Figure 3. Mobil Robot in 2D Plane

The robot's velocity is the average of the velocities of the individual wheels and is given by:

$$v = \frac{v_r + v_l}{2} \tag{1}$$

In equation 1, v_r and v_l denote the linear velocities of the right and left wheels, respectively. The kinematic equations for the differential drive robot in the world frame, given the specified constraints (Hassan et al., 2024), are as follows. Equation 2 represents the motion of a two-wheel differential drive mobile robot (DDMR).

$$v_r = \rho\, \omega_r, v_l = \rho\, \omega_l \tag{2}$$

where:

- ω_r: Angular velocity of the right driving wheel (rads/s).
- ω_l: Angular velocity of the left driving wheel (rads/s).

The robot's dynamic function is defined as:

$$\dot{x} = v\cos(\varphi), \dot{y} = v\sin(\varphi), \dot{\varphi} = \omega = \frac{v_r - v_l}{2l} \tag{3}$$

The previous equations can be expressed in matrix form. The simplified kinematic model of the differential drive mobile robot, used for designing the robot, is represented by equation 4 $R(\varphi)$ in the GCS as:

$$R(\varphi)_{GCS} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \cos(\varphi) & 0 \\ \sin(\varphi) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{4}$$

By transforming these velocities components into rotational velocities (□□, □□), the above model can be improved to:

$$v = \rho \frac{(\omega_r + \omega_l)}{2}, \omega = \rho \frac{\omega_r - \omega_l}{2l} \tag{5}$$

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{\rho}{2} & \frac{\rho}{2} \\ \frac{\rho}{2l} & \frac{-\rho}{2l} \end{bmatrix} \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix} \tag{6}$$

Equation 6 is substituted into equation 4 to derive a more detailed kinematic model, as shown in equation 7.

$$R(\varphi)_{GCS} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \frac{\rho}{2}\cos(\varphi) & \frac{\rho}{2}\cos(\varphi) \\ \frac{\rho}{2}\sin(\varphi) & \frac{\rho}{2}\sin(\varphi) \\ \frac{\rho}{2l} & -\frac{\rho}{2l} \end{bmatrix} \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix} \tag{7}$$

The robot's velocity is the sum of its two speeds, v_r and v_l. Additionally, the relationship between the robot's angular speed, v_r and v_l, and the separation between the two wheels is given by equation 8.

$$v = \frac{(v_r + v_l)}{2}, \dot{\varphi} = \frac{(v_r - v_l)}{2l} \quad (8)$$

By substituting equation 1 into equation 8, we obtain equation 9, which explains the relationship between each wheel's angular velocity and the robot's linear and angular velocity.

$$v = \rho\frac{(\omega_r + \omega_l)}{2}, \dot{\varphi} = \rho\frac{(\omega_r - \omega_l)}{2l} \quad (9)$$

3.2. Accelerators

The current trend in AI development heavily depends on large models and vast datasets, leading to a significant demand for high computing power in advanced AI solutions. Consequently, future research in Reinforcement Learning-based Energy Management Systems (RL-EMS) will likely encounter challenges related to AI deployment. Effective deployment on individual vehicles will require specific deployment architecture tools. The following three primary tools are commonly used for deploying AI solutions: Open Visual Inference and Neural Network Optimization (OpenVino) for CPU devices, TensorRT for GPU devices, and MediaPipe for edge devices (Jooshin et al., 2024; Lin et al., 2024) .

3.2.1. ONNX

ONNX (Open Neural Network Exchange) is an open standard format for machine learning and deep learning models. It enables the conversion of models from various frameworks, such as TensorFlow, PyTorch, MATLAB, Caffe, and Keras, into a single unified format. ONNX provides a common set of operators, building blocks for deep learning, and a standardized file format. It defines a computation graph and includes built-in operators, where the ONNX nodes, which may have multiple inputs or outputs, form an acyclic graph.

After training a network using any framework, the batch size and precision—such as FP32, FP16, or INT8—are set. The trained model is then processed by the TensorRT optimizer, which generates an optimized runtime referred to as a plan. This plan is stored in a .plan file, a serialized format of the TensorRT engine. To run inference, the plan file must be deserialized using the TensorRT runtime (Lai and Morris, 2019).

To optimize models created in various frameworks, the process involves converting the models to the ONNX format and using the ONNX parser in TensorRT to parse the model and build the TensorRT engine. Figure 4 illustrates the high-level workflow for ONNX model optimization (Lai and Morris, 2019).

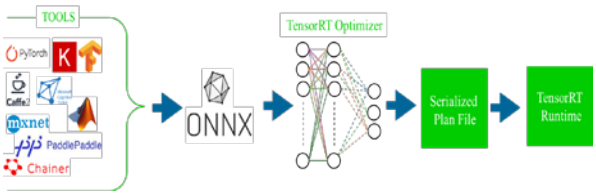

Figure 4. ONNX workflow.

3.2.2. TensorRT

NVIDIA TensorRT is a Software Development Kit (SDK) designed for deep learning inference, supporting both C++ and Python languages. It offers APIs and parsers to import trained models from all major deep learning frameworks, subsequently generating optimized runtime engines suitable for deployment in data centers, as well as automotive and embedded environments. Deep learning is utilized across various applications, including natural language processing, recommender systems, and image and video analysis. As the adoption of deep learning in production grows, the increasing complexity and size of models have heightened demands for accuracy and performance. In safety-critical applications, such as those in the automotive sector, strict requirements for throughput and latency are crucial. Similar demands are present in consumer applications like recommendation systems.

TensorRT is specifically designed to facilitate the deployment of deep learning models for such use cases. It supports all major frameworks, enabling the processing of large datasets with low latency through advanced optimizations, reduced precision, and efficient memory usage.

The deployment process with TensorRT begins by importing the model, which involves loading it from a saved file on disk and converting it into a TensorRT network from its original framework or format. ONNX (Open Neural Network Exchange) serves as a standard for representing deep learning models, facilitating their transfer between different frameworks, including Caffe2, Chainer, CNTK, PaddlePaddle, PyTorch, and MXNet. After conversion, an optimized TensorRT engine is built based on the input model, the target GPU platform, and specified configuration parameters. The final step involves feeding input data to the TensorRT engine to perform inference. Figure 5 illustrates the workflow of converting a trained model into a TensorRT engine (Jeong et al., 2022; Lai and Morris, 2019).

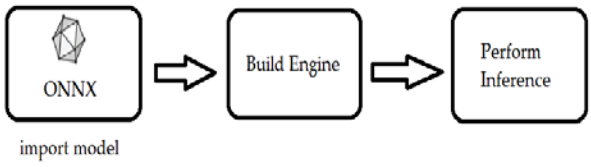

Figure 5. Illustrating work process of trained model into TensorRT.

3.3. Traffic Signs Dataset

The purpose of traffic lights is to warn, inform, arrange or regulate the behavior of users in certain road or traffic conditions. This is why knowing the meaning of traffic lights is essential before you start driving, in this way we will avoid causing accidents due to our fault and car accidents that could have been avoided.

Figure 6 shows the details of the dataset used in this work, which contains 15 categories: two categories for red and green traffic lights, one for stop signs, and the remaining for speed limits. The dataset consists of 4,969 images, divided as follows: 71% for training, 16% for validation, and 13% for testing.

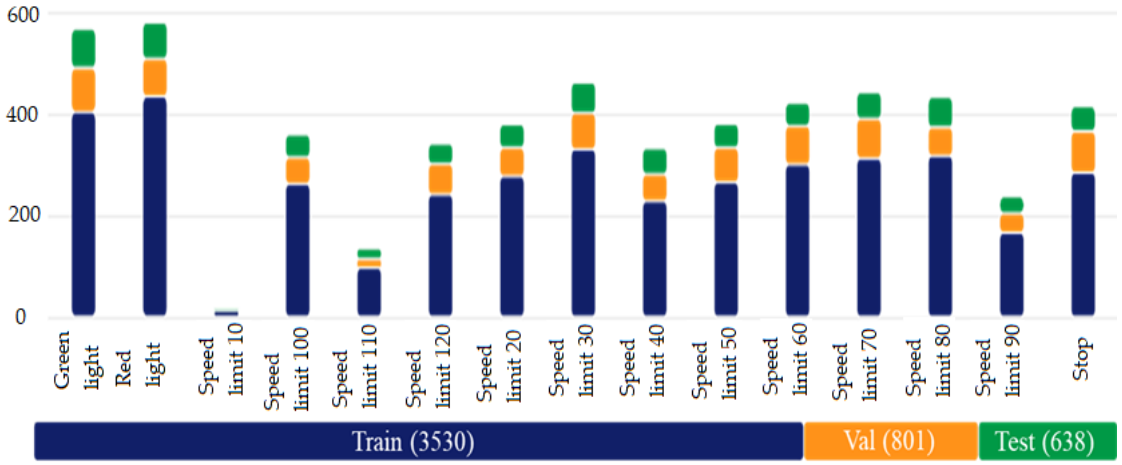

Figure 6. Details of the dataset.

3.4. YOLO Object Detection Algorithm

YOLO algorithm is a state-of-the-art technology developed by Ultralytics for DNN-based object detection / classification, with a major focus on performing real-time tasks without loss of detection accuracy (Terven et al., 2023). YOLO seeks to discover and identify items in real-time inside an image or video stream. By approaching object identification as a single regression issue, YOLO adopts a different strategy from standard approaches, which rely on intricate pipelines and many runs. The input picture is divided into a grid by this method, which then forecasts bounding boxes and class probabilities for items inside each grid cell. It is very quick and efficient since it predicts both the class labels and the bounding boxes that correspond to them at the same time. Because of its remarkable real-time performance, YOLO is renowned for processing photos and movies quickly (Shamta and Demir, 2024).

Table 3 shows a comparison between the performance measurement, Mean Average Precision (mAP) and the number of frames processed for both YOLOv8 and YOLOv5 under the same comparison conditions.

Table 3. A comparison between the performance measurement, (mAP%) and the number of frames processed for YOLOv8 and YOLOv5

Ref	Model	Dataset	Image size (pixels)	Batch size	Device (GPU)	Metrics (mAP@50)	FPS
Terven et al., 2023	YOLOv8	2017 MS COCO	640	32	NVIDIA V100	53.9 %	280
Shamta and Demir, 2024	YOLOv5	2017 MS COCO	640	32	NVIDIA V100	50.7 %	200

4. Experimental Results

The image of the differential robot utilized in this study, equipped with a robotic arm and an NVIDIA Jetson Nano artificial intelligence computer with a camera, is presented in Figure 7. The primary aim of this system is to enable a robotic system, which is open to further hardware development, to perceive visual instructions through deep learning methods. The visual instructions employed include traffic signs such as traffic lights, directional signs, stop signs, and speed limits ranging from 10 to 120 km/h. The YOLOv8 object detection model was used for the detection of these instructions.

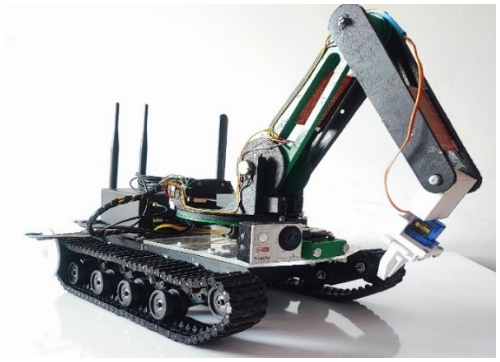

Figure 7. Robotic vehicle.

The training results of the YOLOv8 model demonstrated an overall mAP50 of 96.9%. The performance across the 15 different classes exhibited some variations. Specifically, the mAP50 for the stop sign class was 99.4%, the red traffic light class was 84.6%, and the green traffic light class was 89.3%. For speed limits, the mAP50 values were as follows: 20 km/h at 98.7%, 30 km/h at 99.3%, 40 km/h at 99.2%, 50 km/h at 97.2%, 60 km/h at 97.5%, 70 km/h at 98.7%, 80 km/h at 99.1%, 90 km/h at 98.2%, 100 km/h at 99.2%, 110 km/h at 97.4%, and 120 km/h at 99.4%, as illustrated in Figure 8. The discrepancies observed in the green and red traffic light classes can be attributed to several factors, including the number of images in the dataset, the quality of the captured images, and the presence of noise.

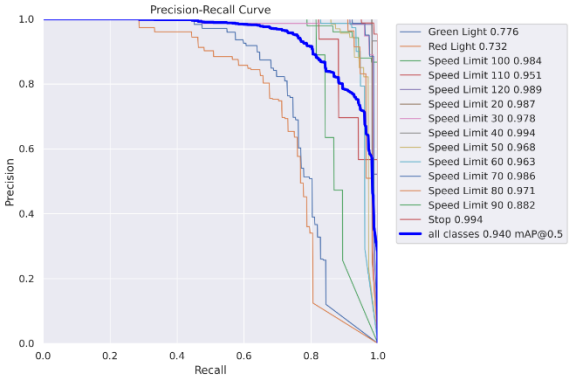

Figure 8. PR curve.

The training process exhibited smooth progression in the training curves for both precision and recall, as well as in the box losses and classification losses during both training and testing phases, as illustrated in Figure 9.

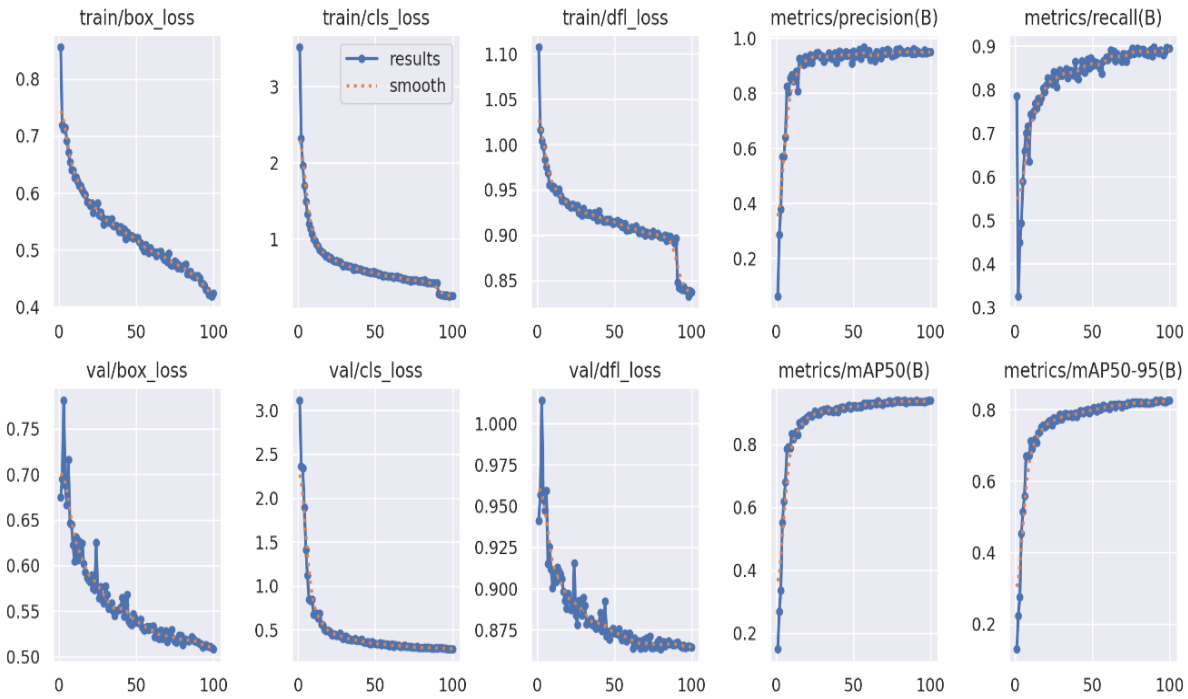

Figure 9. Training process.

In these applications, ensuring the model's accuracy and smooth operation is not sufficient; it is also imperative to consider the computational cost. This is because the computational cost of deploying the models is closely linked to the FPS capability of the camera. To address this, one of Nvidia's accelerators, TensorRT, was employed to optimize the model by reducing its weights, as detailed in Table 4.

Table 4. Results before and after the model acceleration

Model	Image size	FPS
Torch	480x640	14 FPS
TensorRT	640x640	33 FPS

In this study, the employed model demonstrated robustness by achieving a frame rate of 33 FPS with the TensorRT accelerator and an accuracy of 96.6% mAP50 using the YOLOv8 model. However, it exhibited some limitations in terms of high reliability for this type of application. Additional factors include the relative size of Jetson Nano compared to the robot and the accuracy of the camera at various speeds. Jetson Nano's physical dimensions and processing capabilities can pose integration challenges, particularly in maintaining balance and stability of the robot. Moreover, the camera's accuracy can be influenced by motion blur at higher speeds, lighting conditions, and other environmental factors, potentially impacting the model's performance. These constraints highlight the need for ongoing

refinement and optimization. Figure 10 illustrates real-time test image from the model using TensorRT.

Figure 10. Real-Time test image of the accelerated model with TensorRT.

5. Conclusions

This study addresses the detection of visual instructions by a deep learning model in a mobile robotic system equipped with an embedded AI computer. A system comprising a differential robot equipped with a robotic arm, Jetson Nano, and a camera was developed to detect traffic signs as visual instructions using the YOLOv8 object detection deep learning model. The preference for a mobile robotic system over a standard electric vehicle is due to its ability to be easily modified with new hardware and adaptable to new instruction structures based on varying needs. The employed model demonstrated a high overall mAP50 accuracy of 96.9%, with individual class accuracy varying across different traffic signs. The integration of TensorRT accelerated the model, achieving 33 FPS while maintaining a mAP50 accuracy of 96.6%. As evident from the results obtained in this research, a foundational framework is provided for developing more reliable and efficient visual instruction recognition systems in autonomous robotic platforms.

Despite these promising results, challenges remain in achieving high reliability for real-world applications. Factors such as the physical size of the on-board AI computer, camera accuracy at various speeds, and environmental conditions impact overall performance. Future studies are planned to focus on improving the model's reliability and robustness. This may include optimizing hardware-software integration, improving image quality under varying conditions, and integrating additional sensors to address the limitations of the current camera system. Continuous development and testing will be crucial to ensure the high reliability required for real-world applications. It is evident that optimizing hardware-software integration, improving image quality, and enhancing overall system reliability will lead to increased performance.

Author Contributions

The percentages of the author's contributions are presented below. All authors reviewed and approved the final version of the manuscript.

	I.S.	F.D.
C	50	50
D	50	50
S	50	50
DCP	50	50
DAI	50	50
L	50	50
W	50	50
CR	50	50
SR	50	50
PM	50	50
FA	50	50

C=Concept, D= design, S= supervision, DCP= data collection and/or processing, DAI= data analysis and/or interpretation, L= literature search, W= writing, CR= critical review, SR= submission and revision, PM= project management, FA= funding acquisition.

Conflict of Interest

The authors declared that there is no conflict of interest.

Ethical Consideration

Ethics committee approval was not required for this study because of there was no study on animals or humans.

References

Barba-Guaman L, Eugenio Naranjo J, Ortiz A. 2020. Deep learning framework for vehicle and pedestrian detection in rural roads on an embedded GPU. Electronics, *9*(4): 589.

Cai ZX, Gu MQ. 2013. Traffic sign recognition algorithm based on shape signature and dual-tree complex wavelet transform. J Cent South Univ, 20(2): 433–439.

Chen J, Jia K, Chen W, Lv Z, Zhang R. 2022. A real-time and high-precision method for small traffic-signs recognition. Neural Comput Appl, 34(3): 2233–2245.

Çınarer G. 2024. Deep learning based traffic sign recognition using YOLO algorithm. Düzce Univ. J Sci Tech, 12(1): 219–229.

Dalal N, Triggs B. 2005. Histograms of oriented gradients for human detection. Soc Conf Comput Vision Pattern Recog (CVPR'05), San Diego, CA, USA, 1: 886–893.

Flores-Calero M, Astudillo CA, Guevara D, Maza J, Lita BS, Defaz B, Ante JS, Zabala-Blanco D, Armingol Moreno JM. 2024. Traffic sign detection and recognition using YOLO object detection algorithm: A Systematic Rev. Mathematics, 12(2): 1–31.

Girshick R, Donahue J, Darrell T, Jitendra M. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. IEEE Conf Comput Vision Pattern Recog., 580–587.

Gudigar A, Chokkadi S, Raghavendra U, Acharya UR. 2017. Multiple thresholding and subspace based approach for detection and recognition of traffic sign. Mult Tools Appl, 76(5), 6973–6991.

Guney E, Bayilmis C, Cakan B. 2022. An implementation of real-time traffic signs and road objects detection based on mobile

GPU platforms. IEEE Access, 10: 86191–86203.

Han Y, Wang F, Wang W, Li X, Zhang J. 2024. YOLO-SG: Small traffic signs detection method in complex scene. J Supercomp, 80(2): 2025–2046.

Hassan IA, Abed IA, Al-Hussaibi WA. 2024. Path planning and trajectory tracking control for two-wheel mobile robot. J Robot Cont (JRC), 5(1): 1–15.

Hechri A, Mtibaa A. 2020. Two-stage traffic sign detection and recognition based on SVM and convolutional neural networks. IET Image Process, 14(5): 939–946.

Houben S, Stallkamp J, Salmen J, Schlipsing M, Igel C. 2013. Detection of traffic signs in real-world images: The German traffic sign detection benchmark. Int Joint Conf Neural Networks, Dallas, TX, USA, pp: 1–8.

Jeong EJ, Kim J, Tan S, Lee J, Ha S. 2022. Deep learning inference parallelization on heterogeneous processors with TensorRT. IEEE Embed Syst Lett, 14(1): 15–18.

Jin Y, Fu Y, Wang W, Guo J, Ren C, Xiang X. 2020. Multi-feature fusion and enhancement single shot detector for traffic sign recognition. IEEE Access, 8: 38931–38940.

Jooshin HK, Nangir M, Seyedarabi H. 2024. Inception-YOLO: Computational cost and accuracy improvement of the YOLOv5 model based on employing modified CSP, SPPF, inception modules. IET Image Process, 18(8): 1985–1999.

Kounte MR, Shri CvA., Harshvardhan V, Kumari A, Dhruv S. 2022. Design and development of autonomous driving car using nvidiajetson nano developer kit. 8-9 Oct. 2022, 4th Int Conf Cybernetics, Cognition and Machine Learning Appl (ICCCMLA), Goa, India, pp: 486–489.

Krizhevsky A, Sutskever I, Hinton GE. 2017. ImageNet classification with deep convolutional neural networks. Commun ACM, 60(6): 84–90.

Kumar BA, Majji M, Marni HJ, Ateeq M, Koduru S, Maddi SM 2023. A deep transfer learning approach for enhanced traffic sign recognition in autonomous vehicles with NVIDIA jetson nano. Int. Conf Sustain. Emerg Innov Eng Technol (ICSEIET), Ghaziabad, India, pp: 692–698.

Lai G, Morris T. 2019. TensorRT inference with TensorFlow. GPU Tech Conf, pp: 75.

Li Y, Zhang Z, Yuan C, Hu J. 2024. Easily deployable real-time detection method for small traffic signs. J Intell Fuzzy Syst, 46: 8411–8424.

Lin Y, Chu L, Hu J, Hou Z, Li J, Jiang J, Zhang Y. 2024. Progress and summary of reinforcement learning on energy management of MPS-EV. Heliyon, 10(1): e23014.

Mahmoud MAB, Guo P. 2019. A novel method for traffic sign recognition based on DCGAN and MLP with PILAE algorithm. IEEE Access, 7: 74602–74611.

Maldonado-Bascón S, Lafuente-Arroyo S, Gil-Jiménez P, Gómez-Moreno H, López-Ferreras F. 2007. Road-sign detection and recognition based on support vector machines. IEEE Trans Intell Transp Syst, 8(2): 264–278.

Ozcan K, Sharma A, Knickerbocker S, Merickel J, Hawkins N, Rizzo M. 2020. Road weather condition estimation using fixed and mobile based cameras. Advances in Comput. Vision: Proc. of the 2019 Comput. Vision Conf (CVC), 1 (1): 192–204.

Ren S, He K, Girshick R, Sun J. 2017. Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell, 39(6): 1137–1149.

Sarvajcz K, Ari L, Menyhart J. 2024. AI on the road: NVIDIA jetson nano-powered computer vision-based system for real-time pedestrian and priority sign detection. Appl Sci, 14(4): 1440.

Satti SK, Rajareddy GNV, Mishra K, Gandomi AH. 2024. Potholes and traffic signs detection by classifier with vision transformers. Sci Rep, 14(1): 1–18.

Shamta I, Demir BE. 2024. Development of a deep learning-based surveillance system for forest fire detection and monitoring using UAV. PLoS ONE, 19(3): e0299058.

Shustanov A, Yakimov P. 2017. CNN design for real-time traffic sign recognition. Procedia Eng, 201: 718–725.

Stallkamp J, Schlipsing M, Salmen J, Igel C. 2011. The German traffic sign recognition benchmark: A multi-class classification competition. 31 July - 05 August 2011, Int. Joint Conf Neural Networks, San Jose, CA, USA, pp: 1453–1460.

Terakura K, Chang Q, Miyazaki J. 2024. Acceleration of neural network inference for embedded gpu systems. Int Conf Big Data Smart Comput, Bangkok, Thailand, pp: 361–362.

Terven J, Córdova-Esparza DM, Romero-González JA. 2023. A comprehensive review of YOLO architectures in computer vision: from YOLOv1 to YOLOv8 and YOLO-NAS. Machine Learn Knowl Extr, 5(4): 1680–1716.

Thasai Y, Kim P, Ynag Z. 2009. Generalized traffic sign detection model for developing a sign inventory. J Comp Civil Eng, 23(5): 266–276.

Wali SB, Abdullah MA, Hannan MA, Hussain A, Samad SA, Ker PJ, Mansor MB. 2019. Vision-based traffic sign detection and recognition systems: Current trends and challenges. Sensors, 19(9): 2093.

Wali SB, Hannan MA, Hussain A, Samad SA. 2015. An automatic traffic sign detection and recognition system based on colour segmentation, shape matching, and svm. Math Probl Eng, 2015(1): 250461.

You F, Zhang R, Lie G, Wang H, Wen H, Xu J. 2015. Trajectory planning and tracking control for autonomous lane change maneuver based on the cooperative vehicle infrastructure system. Expert Syst Appl, 42(14): 5932–5946.

Yuan X, Hao X, Chen H, Wei X. 2014. Robust traffic sign recognition based on color global and local oriented edge magnitude patterns. IEEE Trans Intell Transp Syst, 15(4): 1466–1474.

Zhang RH, He ZC, Wang HW, You F, Li KN. 2017. Study on self-tuning tyre friction control for developing main-servo loop integrated chassis control system. IEEE Access, 5: 6649–6660.