Multimedia Tools and Applications (2024) 83:25005–25035
https://doi.org/10.1007/s11042-023-16451-1

A performance comparison of YOLOv8 models for traffic sign detection in the Robotaxi-full scale autonomous vehicle competition

Emel Soylu[1] · **Tuncay Soylu**[2]

Received: 16 May 2023 / Revised: 24 July 2023 / Accepted: 6 August 2023 /
Published online: 12 August 2023

Abstract
The ability to recognize traffic signs is a critical skill for safe driving, as traffic signs provide drivers with essential information about the road conditions, potential hazards, speed limits, and other important details that can impact their driving. By recognizing and understanding traffic signs, drivers can react quickly and appropriately to different situations on the road, which helps prevent accidents and ensures the safety of all road users. As part of the Robotaxi-Full Scale Autonomous Vehicle Competition, a project was undertaken to develop a traffic sign recognition system using YOLOv8, a state-of-the-art deep learning model that can detect and classify objects in real-time. The project team trained YOLOv8 on a dataset of traffic sign images to create a model that could accurately recognize and classify different types of traffic signs. This traffic sign recognition system has the potential to significantly improve road safety by helping autonomous vehicles and human drivers to better understand their surroundings and react appropriately to changing road conditions. The system can assist drivers by providing real-time alerts and warnings about potential hazards, speed limits, and other important information. Furthermore, this project demonstrates the power and potential of deep learning and artificial intelligence in improving transportation safety and efficiency. As AI technology continues to advance, we can expect to see more innovative applications in the automotive industry that will help improve the driving experience and make our roads safer for everyone.

Keywords Traffic sign detection · YOLOv8 · Autonomous car · Deep learning

✉ Emel Soylu
emel.soylu@samsun.edu.tr

Tuncay Soylu
tuncay.soylu@samsun.edu.tr

[1] Faculty of Engineering, Department of Software Engineering, Samsun University, Samsun, Turkey

[2] Faculty of Engineering, Department of Electric-Electronics Engineering, Samsun University, Samsun, Turkey

1 Introduction

Autonomous vehicles, also known as self-driving cars, are vehicles that can operate without Human intervention. They are equipped with advanced technologies such as sensors, cameras, and artificial intelligence systems, which allow them to perceive the environment around them and make decisions accordingly. Autonomous vehicles have the potential to revolutionize the transportation industry, as they offer several benefits over traditional vehicles. They can reduce traffic congestion, improve safety, and increase accessibility for individuals who cannot drive due to physical or cognitive impairments [1, 23, 42].

There are several levels of autonomy for vehicles, ranging from Level 0 (no automation) to Level 5 (full automation). Currently, most autonomous vehicles on the market are at Level 2 or 3, which means that they can perform some driving tasks without human intervention but still require a driver to be present and ready to take control if needed [47].

Automatic traffic sign recognition is an important technology for autonomous vehicles and other advanced driver assistance systems. By using computer vision algorithms and machine learning techniques to recognize and interpret traffic signs, vehicles can accurately detect and respond to changing traffic conditions in real-time [17, 41].

The benefits of automatic traffic sign recognition are many. It can help improve road safety by alerting drivers to potential hazards and reducing the risk of accidents caused by human error or distraction. It can also improve traffic flow by providing drivers with real-time information about speed limits, road closures, and other important traffic conditions [2, 46].

In addition, automatic traffic sign recognition can enhance the capabilities of autonomous vehicles by allowing them to navigate roads and intersections more safely and efficiently. This technology is also important for the development of connected and cooperative vehicle systems, which rely on real-time communication and information sharing between vehicles and infrastructure [27, 29, 44, 52].

Artificial intelligence (AI) is being increasingly used in automobiles to enhance their capabilities and improve the driving experience. AI algorithms are used to process large amounts of data from various sources, including sensors, cameras, and connected devices, and to make real-time decisions based on that data. One of the main applications of AI in automobiles is autonomous driving. By using machine learning algorithms to analyze sensor data and generate real-time maps of the surrounding environment, autonomous vehicles can navigate roads and intersections without human intervention. AI can also be used to optimize routes, reduce fuel consumption, and improve overall driving performance. AI is also being used in connected vehicles to provide real-time traffic updates, weather alerts, and other information to drivers. By using advanced analytics and natural language processing, AI can help drivers stay informed and make better decisions on the road. Another important application of AI in automobiles is predictive maintenance. By analyzing data from sensors and other sources, AI algorithms can detect potential issues before they become major problems, allowing for proactive maintenance and reducing the risk of breakdowns and costly repairs [32–34, 40].

TEKNOFEST Technology competitions hold the annual Robotaxi-Full Scale Autonomous Vehicle Competition in Turkey to promote participant competence in autonomous vehicle technologies, original design, algorithms, and reporting. The autonomous vehicle race is an important event for several reasons. Firstly, it promotes innovation and research in the field of autonomous vehicle technology. This competition provides an opportunity for teams to showcase their advancements and technological breakthroughs in autonomous

vehicle design and control systems. Secondly, the autonomous vehicle race helps to accelerate the development of autonomous vehicle technology, which has the potential to revolutionize transportation, making it safer, more efficient, and more accessible. By bringing together experts and enthusiasts from diverse backgrounds, the competition helps to push the boundaries of what is possible in autonomous vehicle technology. Thirdly, the competition helps to build a community of individuals and organizations that are dedicated to advancing autonomous vehicle technology. This community provides a platform for knowledge sharing, collaboration, and networking, which can lead to new partnerships, investment opportunities, and job creation. This helps to build a sense of community and allows individuals to learn from one another. It also encourages individuals to work together to solve complex problems, leading to the development of new ideas and approaches.

Previous studies have investigated various approaches for traffic sign detection using computer vision techniques. For instance, YOLOv3 [30, 36, 56], Faster R-CNN [16, 18, 39, 57], and SSD [15, 51, 54] have been used to detect traffic signs in images and videos. In addition, some studies have explored the use of deep learning algorithms such as CNN [7, 31, 43, 48] and LSTM [3, 10, 12] for traffic sign detection. Other studies have focused on feature extraction and selection methods [4, 13, 24] to improve the accuracy of traffic sign detection. Moreover, some studies have investigated the use of color and shape-based features for traffic sign detection. Despite these efforts, there is still a need for more accurate and efficient traffic sign detection methods, especially for real-time applications.

In our study, under the heading of 5, experimental investigations are presented for Faster R-CNN, SSD, and various versions of YOLO, including YOLOv4, YOLOv4 Tiny, and YOLOv5.

Advanced solutions, such as Deformable DETR (Detection Transformer) and SETR (Set Transformer), have emerged as state-of-the-art (SOTA) approaches for object recognition. Deformable DETR extends the DETR framework by incorporating deformable convolution layers, enabling the model to effectively capture object deformations and handle complex scenes. On the other hand, SETR utilizes a pure transformer architecture for object detection, directly processing tokenized image representations without relying on convolutional layers. These transformer-based algorithms have showcased competitive performance in terms of accuracy and speed compared to traditional convolutional neural network (CNN) methods. They offer several advantages, including improved handling of long-range dependencies, enhanced robustness to object scale variations, and the ability to process samples of different sizes without the need for anchor boxes. The field of computer vision is rapidly evolving, and researchers are continually proposing new SOTA transformer-based algorithms for object detection and segmentation tasks. They are actively exploring various architectural modifications, attention mechanisms, and training strategies to further enhance the performance of transformer-based models in these tasks [9, 26, 49, 50].

The focus of this study is on an application that was created specifically for the purpose of recognizing road signs in the Robotaxi-Full Scale Autonomous Vehicle Competition held in Turkey [38]. The goal of the project is to create a system that could be used in an autonomous vehicle to enable safe and efficient driving. By recognizing and interpreting traffic signs in real-time, the system could help the vehicle navigate complex roadways and make safe driving decisions. The project was successful in developing a robust traffic sign recognition system using YOLOV8, which could be implemented in an autonomous vehicle to improve safety and efficiency on the road.

The novelty of this manuscript lies in the performance comparison of different YOLOv8 models specifically for traffic sign detection in the Robotaxi-Full Scale Autonomous Vehicle Competition. While the proposed method utilizes YOLOv8 as a detector, the

manuscript provides valuable insights by evaluating and comparing the performance of various YOLOv8 models in the context of traffic sign detection. This comparison allows for an assessment of the strengths and weaknesses of different YOLOv8 configurations, shedding light on their effectiveness and suitability for autonomous driving applications. By focusing on traffic sign detection within the specific competition setting, the manuscript offers a targeted analysis that can guide researchers and practitioners in developing optimized YOLOv8 models for similar real-world scenarios.

The application works by using a camera to capture images of the road ahead, and then processing those images in real-time using the YOLOV8 model. When a traffic sign is detected, the application displays information about the sign to the driver, such as its meaning and any relevant speed limits or warnings. This can help drivers stay aware of road conditions and drive more safely. The subsequent sections of this study include the dataset, software, electric vehicle system, comparison results based on varying parameters, and the final outcomes used in the traffic sign recognition application. These sections provide detailed information on the features and specifications of each component, including the comparison results, which help to evaluate the performance of the implemented system. The dataset section covers the image dataset, data preprocessing techniques, and annotation process used in the application. In the software section, we discuss the algorithmic implementation and optimization techniques employed in the system, as well as the programming language and libraries used.

2 Dataset

The traffic sign dataset for the Robotaxi-Full Scale Autonomous Vehicle Competition was created using photographs taken at the competition area. The dataset includes a variety of traffic sign images, such as speed limit signs, stop signs, and pedestrian crossing signs, among others. By using real-world images from the competition site, the dataset provides a realistic representation of the types of traffic signs that an autonomous vehicle would encounter on the road. During the creation of the dataset, an attempt was made to keep the frame wide. Not only the sign itself, but also the area in which the sign is located, was included in the frame. The dataset contains 1106 training images and 123 validation images. Number of images for each traffic sign in dataset is given in Table 1. Data augmentation was not performed on the dataset.

The program called "LabelImg v1.8.6" allows us to label one or multiple areas on the images in the desired way. LabelImg 1.8.6 is an open-source graphical image annotation tool that allows users to manually label and tag objects within an image dataset. It is designed to support a wide variety of image formats, including JPEG, PNG, BMP, and TIFF, and provides a simple and intuitive interface for creating and modifying bounding boxes around objects of interest. As a result of this labeling process, it provides a separate "txt" file for each image. With the help of this file, we can classify our images. Here are a few images from the dataset, along with the annotations Figs. 1, 2 and 3.

Shadows can blur, distort, or render text and other elements on a sign unreadable, posing challenges for signage recognition systems in terms of accuracy and reliability. The impact of shadows on signage recognition systems can be attributed to various factors. For instance, sunlight or shadows cast by structures on the sign can make the text difficult to read. Additionally, shadows can reduce contrast on the sign and darken the image, making it challenging to differentiate the sign. To address this difficulty,

Table 1 Number of images for each traffic sign in dataset

Image	Traffic Sign	Number of Images	Image	Traffic Sign	Number of Images
	No entry	191		Red light	100
	Forward and left	50		Green light	67
	Forward and right	43		Station	106
	Mandatory left	94		Park	37
	Mandatory right	172		No park	31
	No right turn	40		Intersection	129
	No left turn	147		Handicapped parking	22

Fig. 1 Images from the dataset, along with the annotations

image editing tools were used to create shadowed signs. Shadows were artificially added to all images in the dataset to tackle this issue. The added shadows took the form of random geometric shapes, tree branch reflections, or cloud shadows. By doing so, a second dataset was created that included shadows. The data volume was doubled

Fig. 2 Images with added shadows using a photo editing program

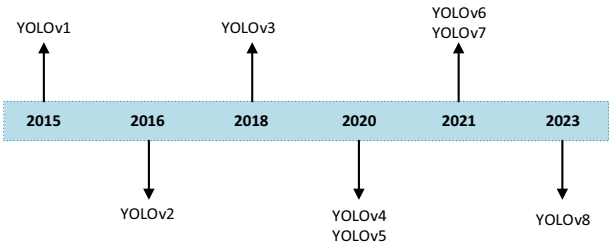

Fig. 3 YOLO Timeline

as a result. The number of images in the dataset containing shadows is twice the values given in Table 2. Examples of images with added shadows are illustrated in Fig. 2.

In this study, when labeling traffic signs, care was taken to ensure their accurate positioning within the image. This involved making sure that the labeled object was fully visible and not cut off or partially obscured. Additionally, the position and size of the object were labeled as accurately as possible. These details are crucial for enabling the YOLO algorithm to detect and locate traffic signs in new images with high accuracy. Therefore, accurate positioning and sizing of the labeled objects were emphasized during the labeling process. During real-time experiments, only nearby traffic signs that autonomous vehicles need to consider were labeled while labeling distant signs was avoided, as they could cause confusion and lead to false detection. In an attempt to mitigate distance-related issues in the table, efforts have been made during the labeling stage. In future versions of the study, better results can be achieved by utilizing the Monocular Depth Estimation (MDE) technique, which estimates depth from a single image [8]. Shadows produce harmful perturbations on the targets [55]. To address shadow-induced errors, shadows were added to the images in the dataset using an image processing program, thereby enriching the dataset.

Table 2 Comparison table for different object detection models

Method	Backbone	Model Size	Accuracy for mAP 0.5	FPS	Average Inference Time	Batch Size	Input Resolution
Faster R-CNN	ResNet101	72.8 MB	100%	19	0.0528 seconds	8	600x600
SSD	Mobilenet-v2	9.2 MB	86%	433	0.00023 seconds	8	320x320
YOLOv4	CSPDarknet53	162.2 MB	96%	1083	0.0009 seconds	64	416x416
YOLOv4-Tiny	CSPDarknet53-Tiny	22.5 MB	97%	1015	0.0009 seconds	64	416x416
YOLOv5 small	yolov5s	14.1 MB	99%	1002	0.001 seconds	16	416x416
YOLOv8 nano	EfficientNet	6.2 MB	100%	1100	0.0009 seconds	16	640x640

3 Method

YOLO (You Only Look Once) is a popular object detection algorithm that can detect objects in real-time from images or videos. There have been eight different versions of YOLO developed so far, each with its own improvements and enhancements [21]:

YOLOv1 is the original version of YOLO introduced in 2015, which can detect objects in real-time but with limited accuracy and struggles with small objects. YOLOv2 is released in 2016, YOLOv2 introduced several improvements over the original version, including better accuracy, faster performance, and the ability to detect smaller objects. YOLOv3 is released in 2018, YOLOv3 further improved the accuracy and speed of object detection, making it one of the most widely used versions of YOLO. It also introduced the concept of feature pyramids to better detect objects of different scales. YOLOv4 is released in 2020, YOLOv4 was a major upgrade over YOLOv3, with even better accuracy and speed. It introduced new features such as scaled-YOLOv4, which can detect even smaller objects. YOLOv5 is released in 2020, YOLOv5 was developed by a different team and uses a different architecture than the previous versions of YOLO. It uses a single-stage detector with a focus on speed and efficiency, but with slightly lower accuracy than YOLOv4. YOLO Nano is a lightweight version of YOLO designed for use on low-power devices. YOLOv7 is an experimental version of YOLO that is not widely used but has shown promise in improving object detection accuracy. YOLOv8 is an experimental version of YOLO that is currently under development, with no official release date yet announced. It is expected to further improve the accuracy and speed of object detection [53].

YOLOV8 is a variant of the YOLO model family that uses a novel backbone architecture based on EfficientNet, a family of convolutional neural networks that are designed to achieve high accuracy while using fewer parameters than traditional models. This makes YOLOV8 more efficient and faster than some other object detection models, while still maintaining high accuracy [11, 28, 45].

Figure 4 provides a summarized version of the detailed YOLOv8 network architecture of the source [6]. The YOLOv8 architecture includes input layer, backbone network, neck, head and output. The input layer of YOLOv8 takes an image as input and preprocesses it by scaling it to a fixed size. The backbone network consists of a series of convolutional layers that extract features from the input image. YOLOv8 uses the CSPDarknet53 backbone network, which is an improved version of the Darknet53 network used in YOLOv7. The neck of the network consists of additional convolutional layers that refine the feature maps produced by the backbone network. YOLOv8 uses a Spatial Attention Module (SAM) to improve feature representation. The head of the network includes several convolutional layers that produce bounding box predictions and class probabilities for each grid cell in the output feature map. YOLOv8 uses a SPP (Spatial Pyramid Pooling) module to better capture object features at different scales. The output layer of YOLOv8 generates the final

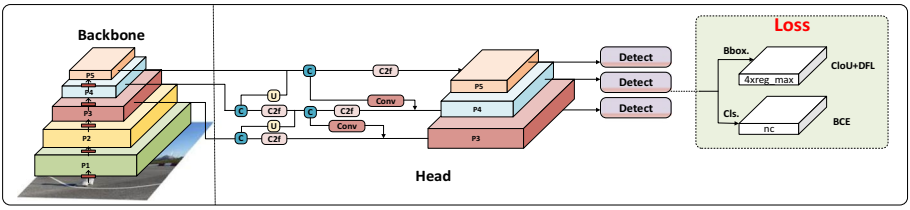

Fig. 4 Abstract architecture of YOLOv8

object detection predictions, including the class labels, bounding box coordinates, and confidence scores.

YOLOV8 can be trained on a variety of object detection tasks, including traffic sign recognition, object tracking, and pedestrian detection, among others. It is often used in applications such as autonomous vehicles, surveillance systems, and robotics, where real-time object detection is essential.

The accuracy of the YOLO algorithm for object detection depends on several parameters, including precision, recall, F1 score, average precision (AP), intersection over union (IoU), non-maximum suppression (NMS) threshold. Precision is the ratio of the number of correctly detected objects to the total number of objects detected by the algorithm. A high precision value indicates that the algorithm produces accurate results. Recall is the ratio of the number of objects detected by the algorithm to the actual number of objects present. A high recall value indicates that the algorithm does not miss real objects. In Eq. 1 and Eq. 2 TP isTrue positive, TN isTrue negative, FP is False positive and FN is False negative. F1 score is the harmonic mean of the precision and recall values. This value indicates that the algorithm produces both accurate results and does not miss real objects. The equation of F1 Score is given in Eq. 3. Average Precision (AP) is a metric that measures how accurately the algorithm detects objects in different classes. AP is calculated by cumulatively computing precision and recall values for a class [17]. And mAP=mean Average Precision. In Eq. 4 AP_k is AP of class k and n is the number of classes [14]. Intersection over Union (IoU) is a metric that measures how much the detected object overlaps with the actual object. This value is used to measure object classification and location accuracy. The equation of IoU is given in Eq. 5. Non-Maximum Suppression (NMS) threshold is a value used to prevent the algorithm from detecting the same object multiple times. This value is determined by measuring the similarity of the same objects in the algorithm's output [19].

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$F1 = 2 \bullet \frac{precision \bullet recall}{precision + recall} \tag{3}$$

$$mAP = \frac{1}{n} \sum_{k=1}^{k=n} AP_k \tag{4}$$

$$IoU = \frac{Intersection\ Area}{Union\ Area} \tag{5}$$

In YOLOv8, the bounding box loss is computed using the CIoU and DFL loss functions, while the classification loss is calculated using binary cross-entropy. Incorporating these loss functions has demonstrated enhanced object detection capabilities, especially when confronted with smaller objects [45].

In evaluating its performance, YOLOv8 incorporates a range of metrics. Box Loss metric gauges the model's ability to accurately fit the true bounding boxes of objects. It quantifies the disparity between the predicted and ground truth bounding boxes, providing a measure of localization accuracy. Class Loss: The class loss metric assesses how

effectively the model predicts object classes. It quantifies the dissimilarity between the predicted class probabilities and the actual object labels, reflecting the model's classification performance. Dual Focal Loss (DFL) is employed to address class imbalance within the dataset. By assigning higher weights to misclassified samples, DFL emphasizes the importance of correctly predicting rare or less frequent classes, helping to alleviate the challenges posed by imbalanced datasets. mAP0.50 & mAP0.50-0.95 metrics, known as Mean Average Precision, serve as widely used evaluation measures in object detection tasks. "mAP0.50" calculates the average precision when considering a detection as correct if the intersection over union (IoU) with the ground truth is above 0.50. On the other hand, "mAP0.50-0.95" computes the average precision over a range of IoU thresholds, specifically from 0.50 to 0.95, providing a broader assessment of detection performance across varying levels of IoU thresholds [35].

Ultralytics is a company that develops and maintains open source software for computer vision tasks, and YOLOv8 is one of their popular projects. We used ultralytics YOLOv8 models in our study as pretrained networks. For the re-training process, we utilized the Python programming language within the Google Colaboratory environment.

4 Training

To detect the objects in our model, the first step we took was to collect data by capturing images from the competition area. Next, we used the LabelImg v1.8.6. freeware software to label the data by annotating each image with bounding boxes that indicate the object locations we want to detect. After labeling our data, we randomly split it into separate sets for training and validation, with a 10% percent validation value. The training set was used to train the model, while the validation set was used to evaluate the model's performance. Following the data split, we created configuration files specifying the model architecture, hyperparameters, and training parameters. This task involved selecting a pretrained YOLOv8 model as a starting point and modifying it to suit our specific application. Finally, we began training the model by feeding it batches of images from the training set and updating its weights. Throughout the training process, we monitored the model's performance on the validation set and made necessary adjustments. We compared the performance of various models and selected the most advantageous one. To train YOLOv8 using a custom dataset, it is necessary to install the ultralytics package which includes the yolo Command Line Interface (CLI). The benefit of using this package is that there is no need to separately clone the repository and install its requirements, which saves a lot of time and effort.

Colab offers several advantages for deep learning training. Firstly, it provides access to powerful GPUs that are required for training deep neural networks, which may not be available on personal computers. Additionally, Colab Pro is a cloud-based service, meaning that users can access their notebooks from anywhere with an internet connection, making it convenient for collaborative work. Another advantage of using Colab Pro is that it comes with pre-installed popular deep learning frameworks such as TensorFlow and PyTorch, eliminating the need for users to set up their own environment. Finally, Colab Pro offers more resources, such as storage and memory, compared to the free version, allowing for more complex and larger-scale deep learning projects Figs. 6, 7, 8, 9, 10, 11, 12 and 13.

Here are a few pointers explaining the hyperparameter choices that we made while training:

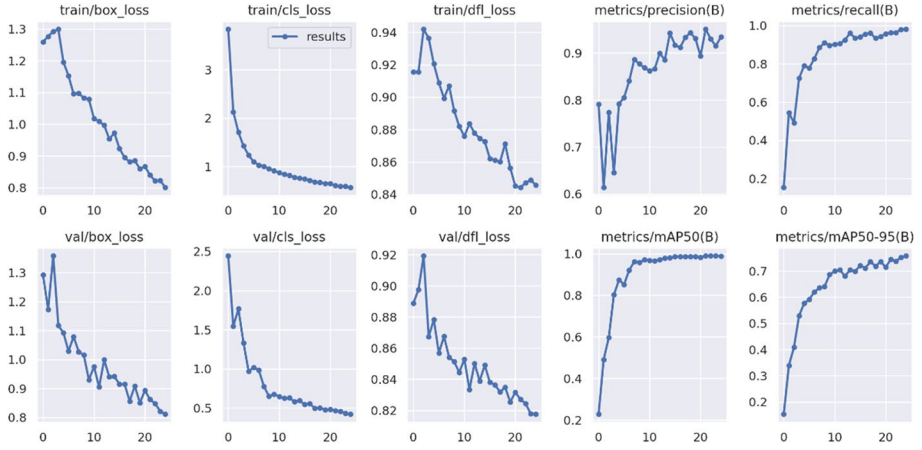

Fig. 5 After training the YOLOv8 Nano model with 224 image resolution, the plots for mAP and loss

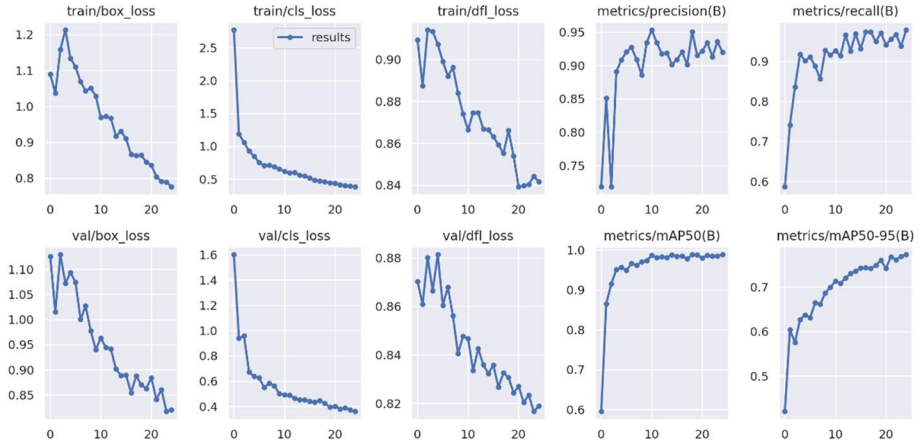

Fig. 6 After training the YOLOv8 Small model with 224 image resolution, the plots for mAP and loss

An epoch represents a complete iteration through the entire image dataset in YOLO. During each epoch, the YOLO model traverses all the bounding boxes in the dataset and updates its model parameters based on the loss function and optimization algorithm mentioned above. The higher the number of epochs, the more times the YOLO model will go through the entire dataset. However, the choice of the epoch number must be made carefully. A low epoch number can lead to underfitting, where the model fails to capture important image features. Conversely, a high epoch number can result in overfitting, where the model becomes overly biased towards the training data and performs poorly on new data. Therefore, selecting an appropriate epoch number often involves a trial-and-error process. During the training process, it was observed that the best epoch values occurred within the range of epochs 16 and 22. Therefore, the epoch value was set to 25. As it was a concept project, we tried to get the best possible results with limited training. Training almost 1106 images for 25 epochs took a considerable amount of time, but we expected decent results. For a fair comparison between models, we kept the

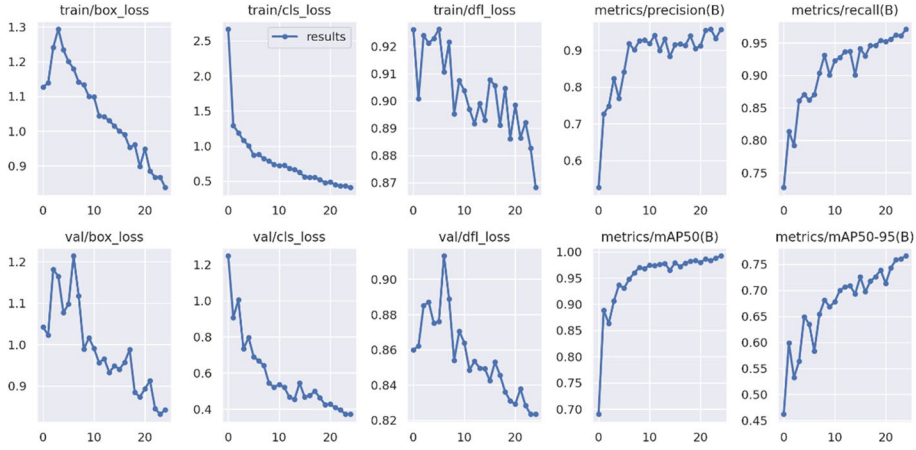

Fig. 7 After training the YOLOv8 Medium model with 224 image resolution, the plots for mAP and loss

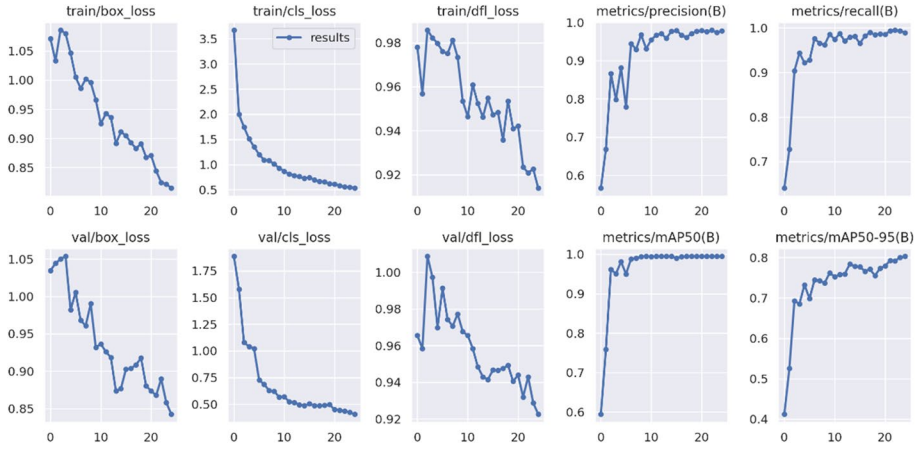

Fig. 8 After training the YOLOv8 Nano model with 640 image resolution, the plots for mAP and loss

batch size constant at 8 for all experiments. We trained each model using three different image resolutions: 224, 640, and 1280. Additionally, we retrained the models using YOLOv8 Nano, YOLOv8 Small, and YOLOv8 Medium architectures. Our goal in conducting this comparative study was to identify the model with high accuracy and performance rate in real-time detection. Performance metrics for the training processes are provided between Figs. 5 and 13.

In preparation for future competitions in the following years, training results obtained from the expanded dataset with added shadows are provided in Figs. 14, 15, 16, 17, 18, 19, 20, 21 and 22. The training process involved approximately 2212 images for training and 246 images for validation over 25 epochs, requiring a significant amount of time, but we had optimistic expectations regarding the results. To ensure a fair comparison across models, we maintained a constant batch size of 8 for all experiments. Each model underwent training at three distinct image resolutions: 224, 640, and 1280. Furthermore, we retrained the models using YOLOv8 Nano, YOLOv8 Small, and YOLOv8 Medium architectures. The primary objective of this

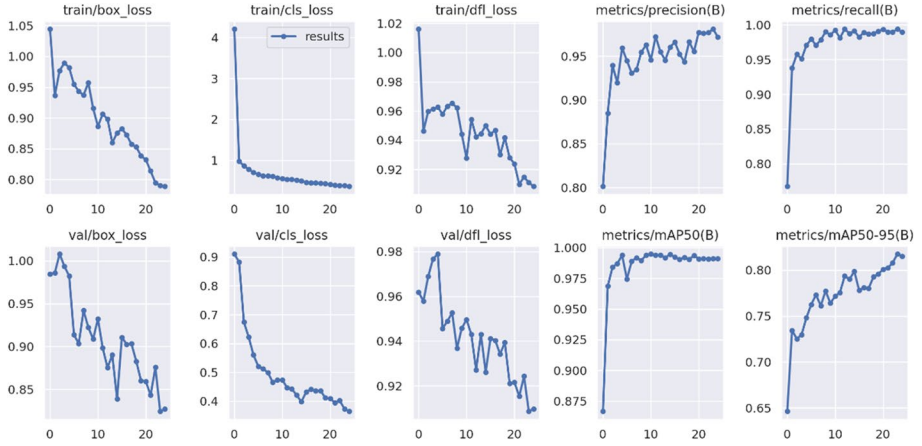

Fig. 9 After training the YOLOv8 Small model with 640 image resolution, the plots for mAP and loss

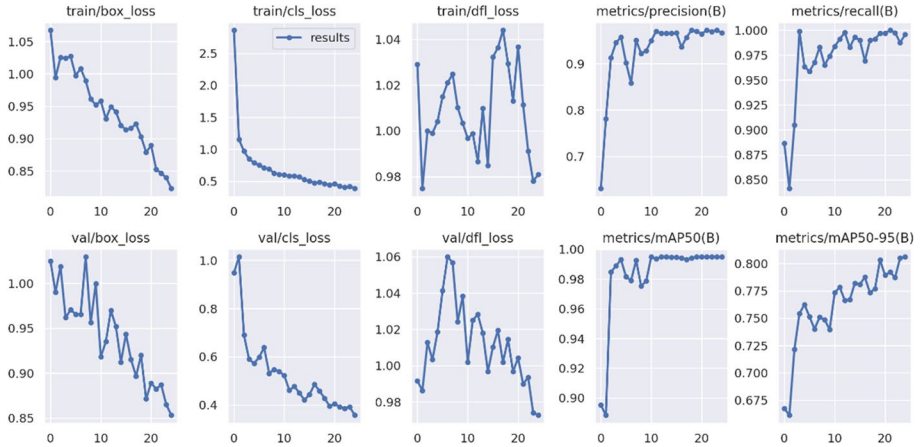

Fig. 10 After training the YOLOv8 Medium model with 640 image resolution, the plots for mAP and loss

comparative study was to pinpoint the model that achieved high accuracy and real-time detection performance.

5 Comparison of other object detection methods

Under this title, we provide information and results for several object detection models, including Faster R-CNN, SSD, YOLOv4, YOLOv4-Tiny, and YOLOv5, by applying an extended dataset in this study.

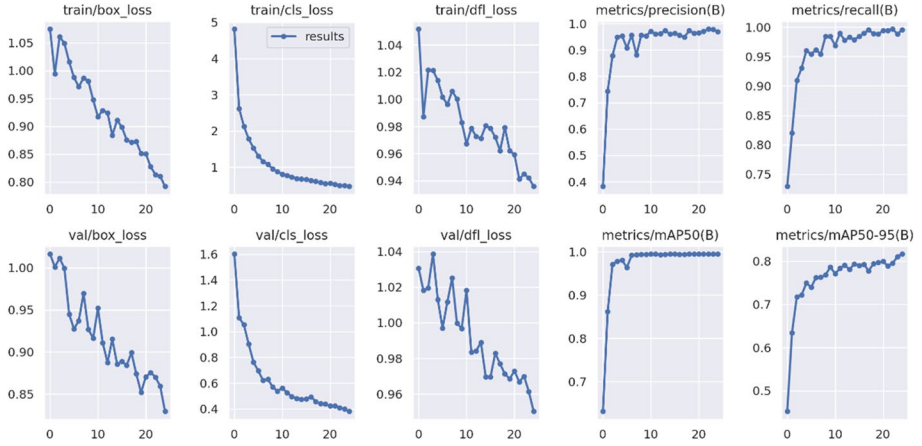

Fig. 11 After training the YOLOv8 Nano model with 1280 image resolution, the plots for mAP and loss

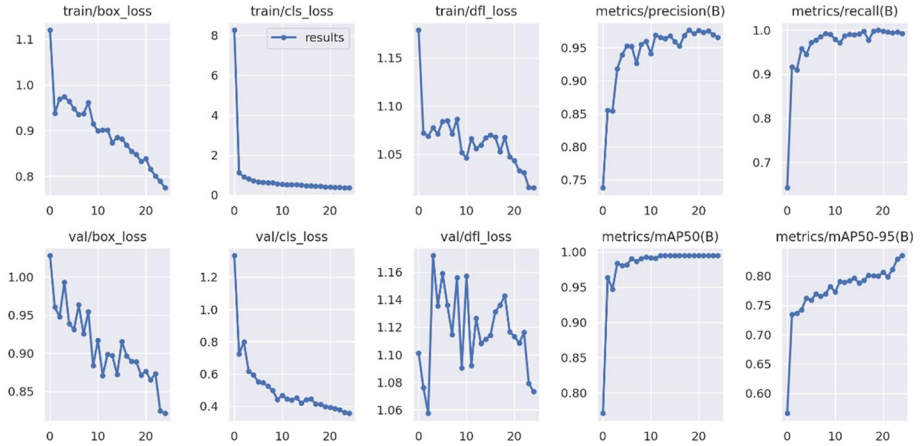

Fig. 12 After training the YOLOv8 Small model with 1280 image resolution, the plots for mAP and loss

5.1 Faster R-CNN

Introduced in 2015 by Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, Faster R-CNN (Region-based Convolutional Neural Networks) is a highly regarded object detection algorithm [37]. It builds upon R-CNN and revolutionizes the detection pipeline by integrating region proposal generation directly into the neural network, resulting in enhanced speed and efficiency. The Region Proposal Network (RPN) is a fully convolutional network responsible for generating region proposals, representing potential bounding boxes with objects of interest. These proposals are scored based on their likelihood of containing objects. Subsequently, the Region of Interest (RoI) pooling layer aligns and warps features within each proposed region to a fixed size, ensuring consistent representation regardless of the original region size. Faster R-CNN utilizes separate branches for classification and regression, predicting object class probabilities and bounding box offsets, respectively, to accurately localize the objects. By

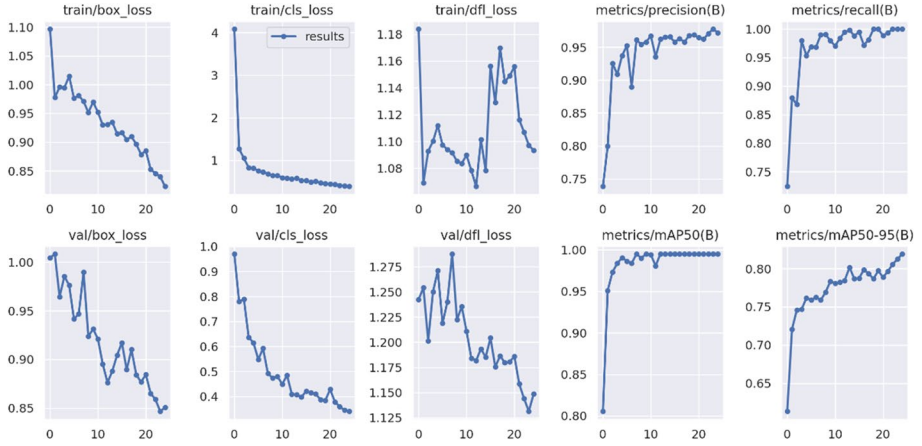

Fig. 13 After training the YOLOv8 Medium model with 1280 image resolution, the plots for mAP and loss

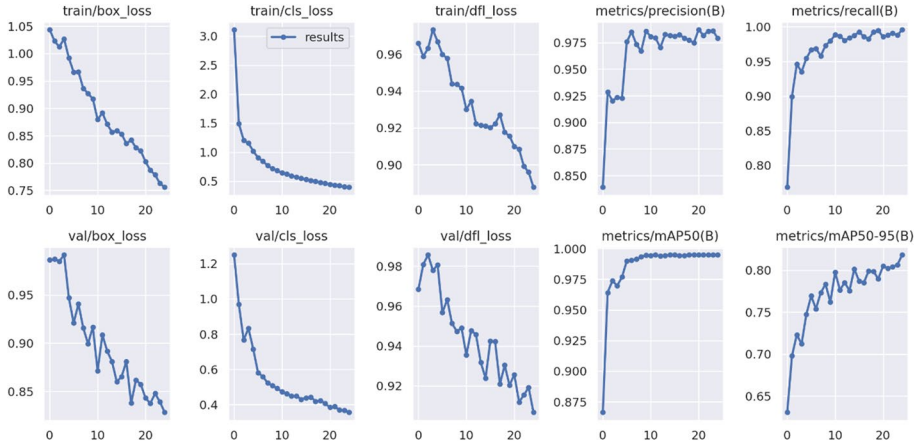

Fig. 14 The plots for mAP and loss values after training the YOLOv8 Nano model with 224 image resolution using the expanded dataset

eliminating the need for external region proposal methods like Selective Search, Faster R-CNN significantly improves detection efficiency, making it one of the most widely used and influential object detection architectures in the realm of deep learning-based algorithms.

In our study, we utilized Resnet101 as the Backbone Feature Extractor in the Faster R-CNN method. The extended dataset was split into 80% for training, 10% for validation, and 10% for testing, totaling 1974, 247, and 247 images, respectively. The images were resized to 600x600 dimensions. After training for 54 epochs, with a learning rate of 0.00001, the loss values were as follows: total loss: 0.295, loss_classifier: 0.084, loss_box: 0.209, loss_rpn_box: 0.00068, loss_object: 0.0011. The training process took approximately 3.91 hours. At the end of the training, the model size is 72.8 MB.

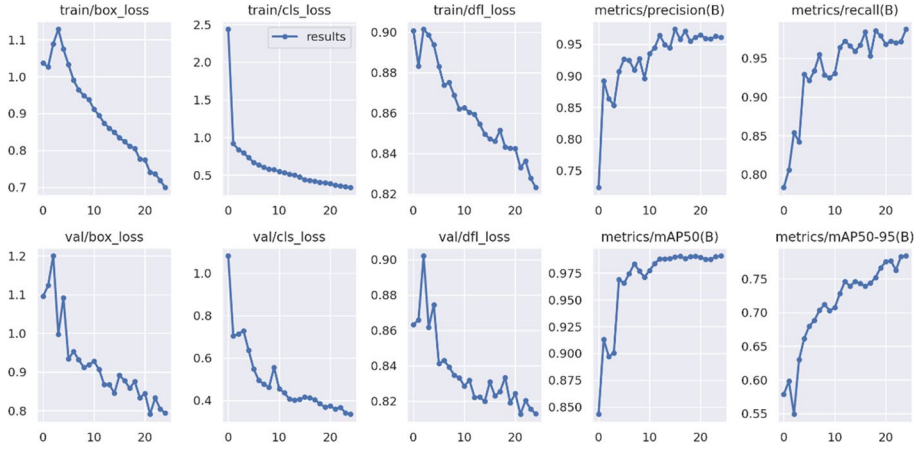

Fig. 15 The plots for mAP and loss values after training the YOLOv8 Small model with 224 image resolution using the expanded dataset

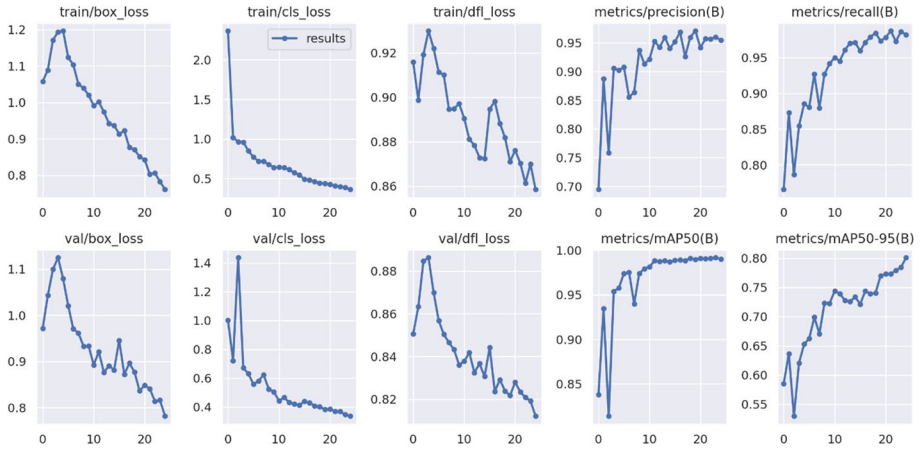

Fig. 16 The plots for mAP and loss values after training the YOLOv8 Medium model with 224 image resolution using the expanded dataset

5.2 SSD

Single Shot Multibox Detection (SSD) is a widely used real-time object detection algorithm in computer vision and deep learning, introduced by Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg in 2016 [25]. SSD's key innovation is performing object detection and localization in a single forward pass through a deep neural network, eliminating the need for multiple stages in previous methods. Leveraging feature maps of various resolutions from multiple neural network layers, SSD captures information at different scales, enabling robust detection of objects in different sizes. The algorithm utilizes predefined anchor boxes with diverse aspect ratios and scales at each location in the feature maps, serving as reference templates for detecting objects of various shapes. At each anchor box location, SSD predicts class

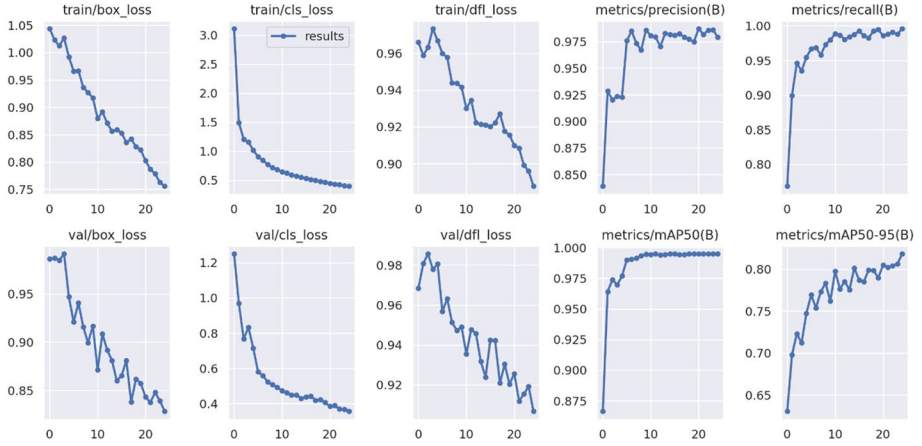

Fig. 17 The plots for mAP and loss values after training the YOLOv8 Nano model with 640 image resolution using the expanded dataset

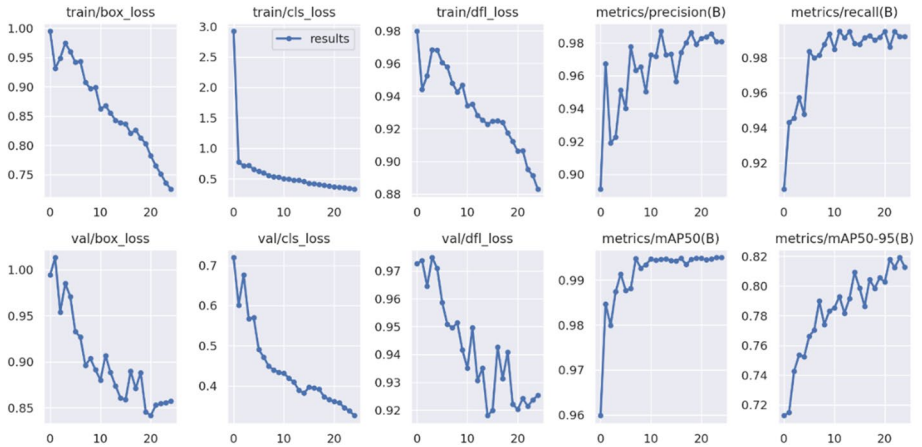

Fig. 18 The plots for mAP and loss values after training the YOLOv8 Small model with 640 image resolution using the expanded dataset

scores and bounding box offsets to identify the object class and refine the anchor box position and size. With a combination of classification and localization loss during training, SSD efficiently detects and locates objects in images, making it highly suitable for real-time applications and scenarios requiring speed. Since its inception, SSD has become a cornerstone in object detection, inspiring further research and advancements in the field, leading to the development of various SSD variants and other single-shot detection architectures aimed at improving both accuracy and efficiency.

Using the extended dataset with added shadows, we trained the SSD (Single Shot Multibox Detection) model. The training process reached step 16800, with each step taking approximately 0.795 seconds. Classification loss is 0.059912365, Localization loss is 0.02045659, Regularization loss is 0.07491396, Total loss is 0.15528291 and Learning rate is 0.061172597 at last step. These results indicate the performance of the model in

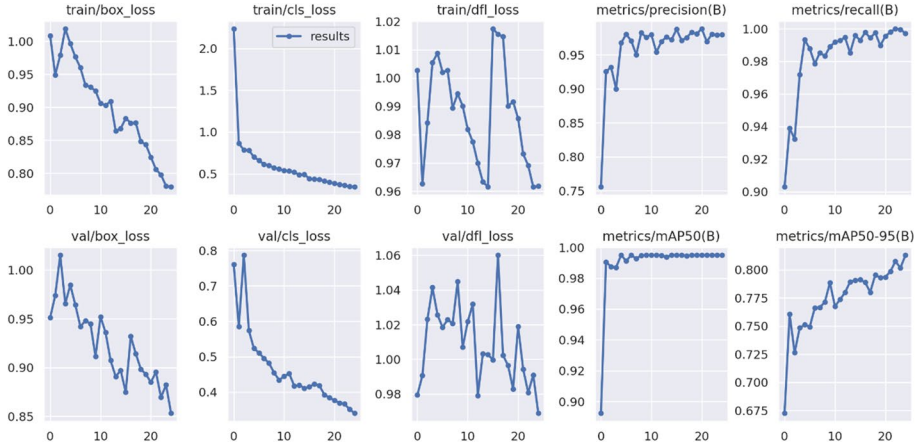

Fig. 19 The plots for mAP and loss values after training the YOLOv8 Medium model with 640 image resolution using the expanded dataset

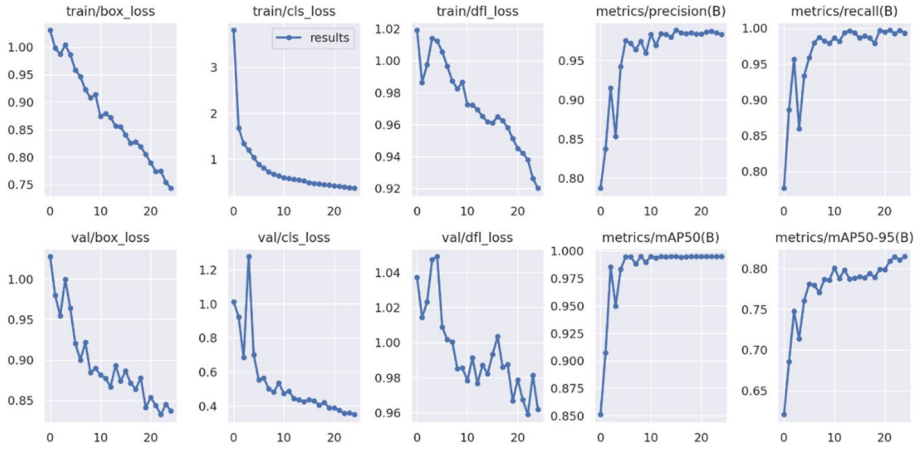

Fig. 20 The plots for mAP and loss values after training the YOLOv8 Nano model with 1280 image resolution using the expanded dataset.

detecting and localizing objects, with the losses providing insights into how well the model is learning from the training data. The learning rate shows the rate at which the model's weights are being updated during training.

5.3 Yolov4

Introduced in 2020 by Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao, YOLOv4 is a cutting-edge real-time object detection algorithm known for its efficiency and accuracy, making it a favored choice across a wide range of computer vision applications [5]. Its backbone architecture, CSPDarknet53, is a more profound and efficient version of Darknet, enabling superior feature extraction capabilities. By utilizing PANet (Path

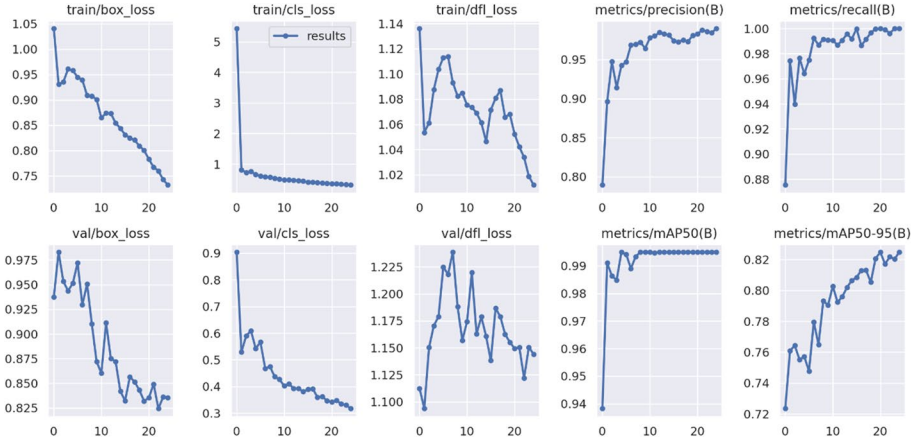

Fig. 21 The plots for mAP and loss values after training the YOLOv8 Small model with 1280 image resolution using the expanded dataset

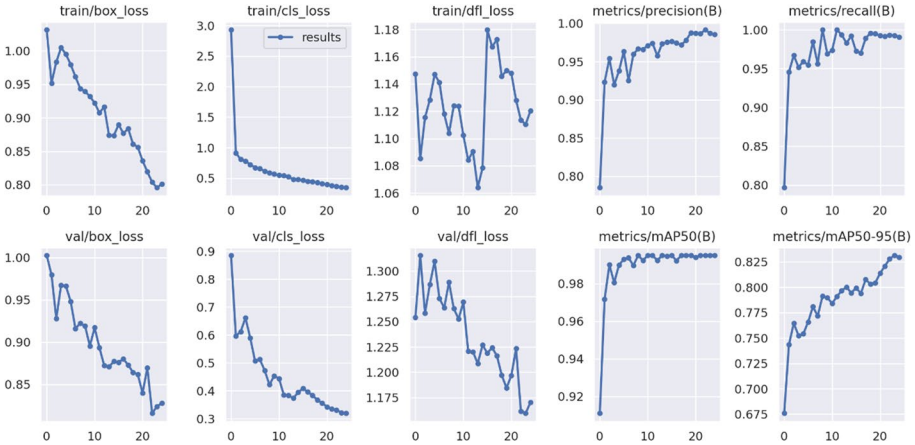

Fig. 22 The plots for mAP and loss values after training the YOLOv8 Medium model with 1280 image resolution using the expanded dataset

Aggregation Network), YOLOv4 creates a feature pyramid, allowing it to effectively detect objects of varying sizes. Incorporating spatial attention further enhances the model's focus on critical image regions, leading to improved detection performance. With CSP connections, YOLOv4 fosters better information flow and gradient propagation during training. The model's loss function combines multiple components, prioritizing object detection, classification, and bounding box regression. Utilizing an ensemble of models trained on different scales and resolutions, YOLOv4 attains heightened accuracy and robustness in detecting diverse objects. Its impressive performance and efficiency have resulted in widespread adoption in applications ranging from surveillance systems to autonomous vehicles and robotics.

The extended dataset, which was augmented with shadows, was split into 80% for training, 10% for validation, and 10% for testing. The results obtained at iteration 28000 of the

YOLOv4 model training are as follows: The loss recorded for the last iteration is 0.547, while the average loss over the training process stands at 0.797368. The learning rate used for this training phase is set to 0.001. The training process took 45.43 hours. At the end of the training, the model size is 162.2 MB. When using the Google Colab Pro+ environment, the obtained results demonstrate impressive performance for the YOLOv4 model. During the inference process on a test dataset containing 122 images, the total inference time was measured to be 0.11 seconds, resulting in an outstanding average inference time per image of just 0.0009 seconds. This remarkable speed translates to an impressive frames-per-second (FPS) rate of approximately 1083, indicating that the model can process more than a thousand images in a single second.

5.4 YOLOv4 Tiny

YOLOv4 Tiny presents a lighter and faster iteration of the YOLOv4 object detection algorithm, tailored to cater to scenarios with limited computational resources. Despite its reduced computational requirements, YOLOv4 Tiny retains real-time object detection capabilities, making it ideal for edge devices and embedded systems [20]. Compared to the original YOLOv4, YOLOv4 Tiny employs a smaller and simplified network architecture, resulting in swifter inference times and lower memory usage. It typically operates with fewer layers, parameters, detection scales, and anchor boxes per scale. Additionally, YOLOv4 Tiny often handles a lower input resolution, ensuring reduced computational demands during inference. As a favored choice for various real-time object detection applications on devices like mobile phones, drones, and other edge devices, YOLOv4 Tiny excels at processing video streams or camera feeds in real-time while maintaining a lightweight model.

We applied YOLOv4 Tiny for the extended dataset, which was augmented with shadows, was split into 80% for training, 10% for validation, and 10% for testing. In the YOLOv4 Tiny model, the final accuracy recorded for the mAP at an Intersection-Over-Unity (IoU) threshold of 0.50 at the 28000th iteration was 92.43%, while the best mAP achieved throughout the training was 97.63%. The training process took 11.01 hours. At the end of the training, the model size is 22.5 MB. The FPS is 1015.32 and average inference time is 0.001 seconds.

5.5 YOLO v5

YOLOv5, released in mid-2020 by Ultralytics as an open-source project, has quickly gained popularity in the computer vision community. It introduces a modern and versatile architecture, departing from the Darknet-based approach of its predecessors, while still drawing inspiration from the YOLO series' core concepts [22]. With a range of pre-built models catering to different resources and performance needs, YOLOv5 offers simplicity and flexibility for customization. Built on PyTorch, it provides accessibility to researchers and developers familiar with the framework. Besides its renowned object detection capabilities, YOLOv5 proves adaptable for tasks like image segmentation and feature extraction. Supporting transfer learning, it can be fine-tuned on specific datasets for real-world applications. YOLOv5's optimized efficiency enables deployment on edge devices and embedded systems, facilitating real-time object detection in resource-constrained environments. Its user-friendly interface, transfer learning capabilities, and efficient performance have made it a preferred choice for various applications, from research to practical implementations.

We applied YOLOv5s model for the extended dataset, which was augmented with shadows, was split into 80% for training, 10% for validation, and 10% for testing. YOLOv5s model has 182 layers and a total of 7281579 parameters. There are 8 classes in the evaluation, and a total of 247 images were used for testing, containing 263 instances of objects across all classes. The model achieved high precision (P) and recall (R) scores, with an average precision (mAP50) of 0.994 and a mean average precision over the range 50-95% (mAP50-95) of 0.797. 100 epochs completed in 1.615 hours and the model size is 14.1 MB. The FPS is 1002 and average inference time is 0.001 seconds.

5.6 Comparison Table for object detection models

Table 2 presents a comparison of various object detection methods along with their respective backbone models, model sizes, accuracy rates (mAP at 0.5), frames per second (FPS), average inference times, batch sizes, and input resolutions. These results were obtained using Google Colab Pro+ with GPU and high speed RAM configuration.

According to Table 2, the most advantageous model is YOLOv8 nano. This is because it not only achieves a high accuracy rate but also operates at a high number of frames per second, making it highly efficient for real-time applications. Additionally, YOLOv8 nano has the advantage of having a small model size, making it more lightweight and suitable for deployment on resource-constrained devices.

6 Autonomous Car and other equipment

This section provides technical details regarding the original vehicle used in the competition, the computer system on which the software was executed, and the camera utilized for capturing images. The appearance of the vehicle is shown in Figure 23, and its technical specifications are provided in Table 3. The table provides information about the specifications of a vehicle, including its dimensions, materials, and technical components. The vehicle has a length of 3500 mm, a width of 1200 mm, and a height of 1100 mm, and is built with an aluminum 6063 T5 and 6082 T6 chassis and a carbon fiber shell. It features a brushless DC motor with a power output of 2kW, 90% motor efficiency, and a weight of 20

Fig. 23 Autonomous vehicle

kg, and is equipped with the manufacturer's own motor driver. The vehicle is powered by a Lityum Ion battery with a nominal voltage of 46.8V, a capacity of 22.75Ah, and a maximum voltage of 54.6V, providing an energy capacity of 1064.7Wh. The brake system is fiber-based and operates at the same nominal voltage as the battery pack. Overall, the specifications suggest that the vehicle is designed for efficiency and lightweight performance, with a focus on using advanced materials and technology to minimize energy consumption and maximize power output.

The system block diagram is illustrated in Fig. 24, where real-time image captured by the camera is transmitted to the computer, and the traffic signs are recognized using the YOLOv8 Nano model, and then displayed on the screen.

The autonomous control of the vehicle is provided by a computer. The computer sends information about steering angle, speed, and brake to the vehicle control system. Based on this information, the battery will control the vehicle. The computer used in this study is the Acer Predator Helios 300 laptop, which operates on the Windows 11 Pro operating system. It is powered by an Intel i7 10750H processor, featuring a clock speed of 2.60 GHz and 6 cores. Additionally, it is equipped with an NVIDIA GeForce RTX3060 graphics card with 6GB memory and has a 16GB RAM capacity. We used Visual Studio Code environment to use trained model in real-time experiments.

In this study, the Intel RealSense D435 depth camera was used for image capture. The Intel RealSense D435 depth camera is a stereo solution with high depth resolution for various applications. Stereo image detection technologies use two cameras to calculate depth. The obtained images in the stereo camera method are evaluated to calculate the 3D position. The design of the human eye is inspired by the 3D vision feature of objects. This module will be used as a camera in the autonomous driving algorithm of the vehicle. This camera is preferred due to its open-source code support, ROS compatibility, and 1920*1080 RGB frame resolution.

7 Experimental Results

In Table 4 a YOLOv8 comparison chart that measures the performance of various versions of a traffic sign detection. The table provides information about how the different versions of the model were trained for different image sizes, the number of parameters they have, their training time, and file size. Additionally, the average precision (mAP) measurements of each model are provided. The YOLOv8 model has three different versions for three different image sizes - 224x224, 640x640, and 1280x1280 - named nano, small, and medium. Each version has different numbers of parameters and training times. The table also includes mAP-50 and mAP-50-95 measurements, which are criteria used to measure the accuracy and precision of the model. mAP-50 gives the detection rate of objects that have at least a 50% accuracy rate, while mAP-50-95 gives the detection rate of objects that have an accuracy rate between 50% and 95%. The different model versions (nano, small,

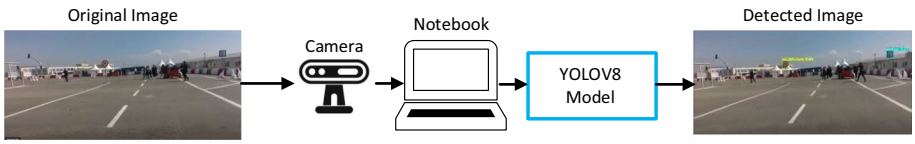

Fig. 24 Block diagram of the system

medium) achieve varying levels of mAP and loss components (box loss, class loss, Dfl loss). Overall, the results show that models with larger image sizes generally have higher mAP measurements. Additionally, models with more parameters and longer training times also tend to have higher mAP measurements. However, it's important to note that larger models also have larger file sizes. Therefore, model selection should be based on the application's requirements and available resources.

When the results were compared, the most advantageous model was found to be the YOLOv8 Nano model with 640 image resolution. This is because of its high mAP-50 value and smaller file size, which result in better real-time performance. The confusion matrix for Yolov8 Nano model with 640 image resolution, showing the differences between the actual and predicted classes, is given in Fig. 25. According to the confusion matrix, the accuracy rates of the results are high, nearly for all classes. However, the accuracy rates for class (park) is relatively lower but still acceptable. Sample screenshots of real-time traffic sign detection results are given in Figs. 26, 27 and 28.

Figure 29 presents two examples of misclassified results. The possible reason for these misclassifications could be attributed to the limited training data.

Table 5 presents a comprehensive comparison of various YOLOv8 models using extended data, categorized by their versions, types, image sizes in pixels, layers, parameters, training times in hours, file sizes in MB, and evaluation metrics. From the table, it can be observed that as the image size increases, the number of parameters, training time, and file size also increase, indicating the greater computational resources required for larger images. Furthermore, there is a general trend of improvement in mAP-50 and mAP-50-95

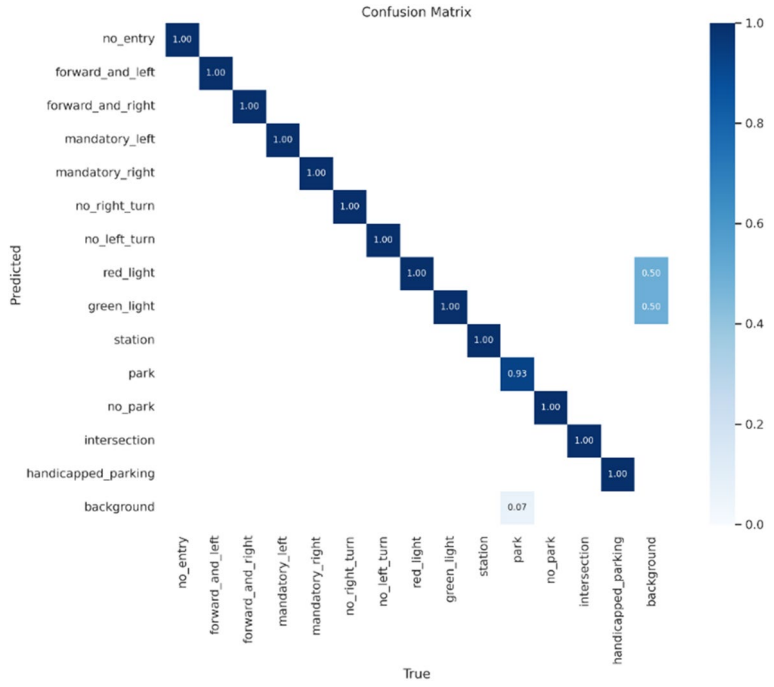

Fig. 25 The confusion matrix of Yolov8 Nano model for traffic sign classification with 640 image resolution

Fig. 26 Real-time traffic sign detection results (a)

Fig. 27 Real-time traffic sign detection results (b)

Fig. 28 Real-time traffic sign detection results (c)

scores as the model size increases, with the medium-sized models achieving higher mAP scores compared to the nano and small-sized models, signifying superior performance on this extended dataset. Regarding the loss components, the box loss decreases as the model size increases, while the class loss and Dfl loss demonstrate varying trends across different model versions and image sizes. Considering computational constraints, the most advantageous model identified from this comparison is YOLOv8 nano. The confusion matrix of

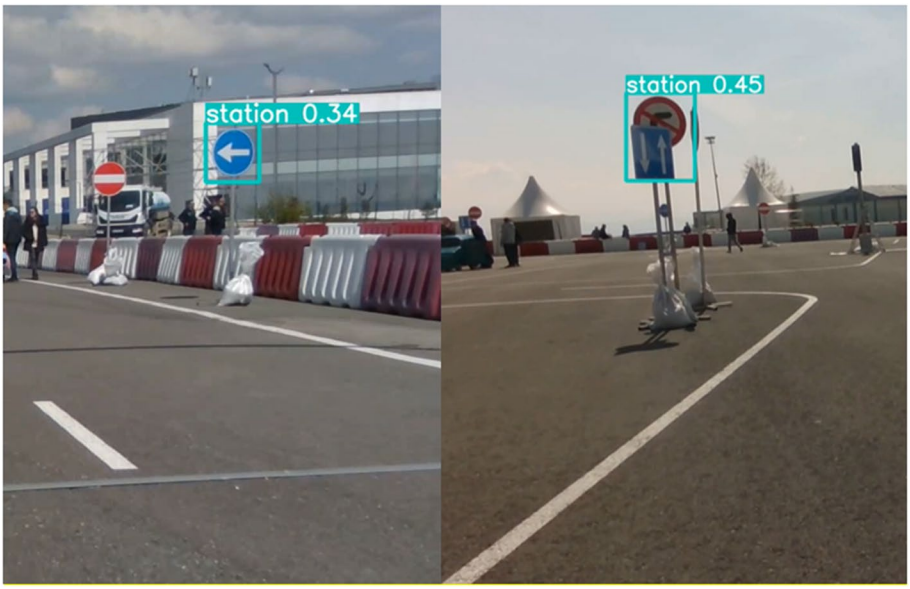

Fig. 29 Misclassified frames

Yolov8 Nano model for traffic sign classification with 640 image resolution for extended data is given in Fig. 30.

8 Results and Conclusion

With the increasing development of autonomous vehicles, the ability to accurately recognize traffic signs is becoming more critical. By using YOLOv8 for traffic sign recognition, these vehicles can navigate the roads more safely and efficiently, making them a more viable transportation option for the future. A successful implementation of YOLOv8 for traffic sign recognition could serve as a benchmark for further advancements in computer vision techniques and their application to real-world problems.

The study made use of a distinctive dataset, which ultimately led to achieving highly accurate results. In our study according to the comparison results, the YOLOv8 model with the "nano" version and trained for 14 types traffic sign recognition using an image size of 640 pixels achieved the highest mAP-50 score of 0.995 and a respectable mAP-50-95 score of 0.802. In addition, this model has a relatively small number of parameters and training time compared to the other versions, and has a relatively small file size. Therefore, this version of YOLOv8 may be the best option for traffic sign recognition tasks that require high accuracy and efficiency. However, it's important to note that the choice of model ultimately depends on the specific requirements and constraints of the task at hand.

In this study, the dataset was expanded by considering the shadow hazard, and shading was applied in the image processing program. The augmented dataset was used for training different object detection algorithms, including Faster R-CNN, SSD, YOLOv4, YOLOv4-Tiny, and YOLOv5 (previous versions). Subsequently, these models were tested using the expanded dataset. Additionally, the YOLOv8 model was trained on the augmented dataset,

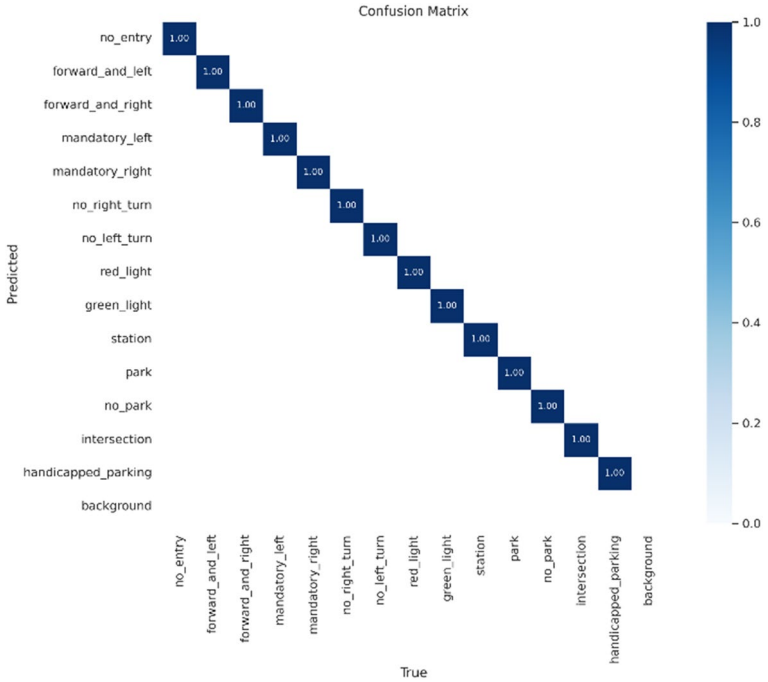

Fig. 30 The confusion matrix of Yolov8 Nano model for traffic sign classification with 640 image resolution for extended data

Table 3 Vehicle properties

Property	Value	Property	Value
Length	3500 mm	Motor power	2kW
Width	1200 mm	Motor efficiency	90%
Height	1100 mm	Electric machine weight	20kg
Chassis	Aluminum 6063 T5, 6082 T6	Battery	Lityum Ion
Shell	Carbon Fiber	Battery pack nominal voltage	46.8V
Brake system	Fiber Battery pack nominal voltage 46.8V	Battery pack capacity	22,75Ah
Motor	Brushless DC Motor	Battery pack maximum voltage	54.6V
Motor driver	Our own design	Battery pack energy	1064.7Wh

considering various model sizes and input image resolutions. Test results were obtained for the YOLOv8 model under different configurations. When taking into account accuracy, speed, and model size in real-time, the most advantageous model configuration found is still YOLOv8-nano.

Integrating multiple cameras with different viewpoints can improve the accuracy and reliability of the model. Techniques like model quantization, pruning, and optimization can be used to optimize the model's performance for real-time applications. This study can be further enriched in the future by applying it to transformer-based models that involve natural language processing.

Table 4 Comparison table for different size of YOLOv8 models

Version	Type	Image Size (pixels)	Layers	Parameters	Training Time (hours)	File Size (Mb)	mAP-50	mAP-50-95	Box Loss	Cls Loss	Dfl Loss
YOLOv8	nano	224	225	3013578	1.269	6.200	0.989	0.758	0.8	0.566	0.845
YOLOv8	small	224	225	11141018	1.263	22.500	0.989	0.773	0.776	0.381	0.841
YOLOv8	medium	224	295	25864426	1.352	52.000	0.992	0.766	0.837	0.409	0.868
YOLOv8	nano	640	225	3013578	1.226	6.200	0.995	0.802	0.814	0.931	0.913
YOLOv8	small	640	225	11141018	1.250	22.500	0.991	0.815	0.788	0.311	0.908
YOLOv8	medium	640	295	25864426	1.358	52.000	0.995	0.807	0.823	0.386	0.981
YOLOv8	nano	1280	225	3013578	1.361	6.300	0.989	0.758	0.792	0.466	0.935
YOLOv8	small	1280	225	11141018	1.475	22.600	0.989	0.772	0.774	0.354	1.015
YOLOv8	medium	1280	295	25864426	1.738	52.100	0.992	0.766	0.823	0.387	1.093

Table 5 Comparison table for different size of YOLOv8 models for extended data

Version	Type	Image Size (pixels)	Layers	Parameters	Training Time (hours)	File Size (Mb)	mAP-50	mAP-50-95	Box Loss	Cls Loss	Dfl Loss
YOLOv8	nano	224	225	3013578	1.223	6.2	0.99	0.776	0.73	0.433	0.832
YOLOv8	small	224	225	11141018	1.226	22.5	0.99	0.78	0.699	0.334	0.823
YOLOv8	medium	224	295	25864426	1.295	52	0.99	0.801	0.761	0.361	0.858
YOLOv8	nano	640	225	3013578	1.202	6.2	0.99	0.81	0.755	0.397	0.887
YOLOv8	small	640	225	11141018	1.2	22.5	0.99	0.81	0.72	0.362	0.883
YOLOv8	medium	640	295	25864426	1.738	52	0.99	0.81	0.77	0.343	0.961
YOLOv8	nano	1280	225	3013578	1.62	6.3	0.99	0.81	0.74	0.36	0.92
YOLOv8	small	1280	225	11141018		22.6	0.99	0.82	0.73	0.31	1.012
YOLOv8	medium	1280	295	25864426	2.3	52.1	0.99	0.83	0.8	0.34	1.12

Acknowledgement This study received financial support from the Scientific and Technological Research Council of Turkey (TUBITAK) in 2023 as a project of the Samrobotik team with ID 895869.

Data availability Data will be made available on reasonable request.

Declarations

Conflict of interests The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Abdallah MKA (2022) Autonomous self-driving car using Raspberry Pi
2. Ansari S, Naghdy F, Du H (2022) Human-machine shared driving: challenges and future directions. IEEE Trans Intell Veh 7:499–519
3. Atif M, Zoppi T, Gharib M, Bondavalli A (2022) Towards enhancing traffic sign recognition through sliding windows. Sensors 22:2683
4. Bahlmann C, Zhu Y, Ramesh V et al (2005) A system for traffic sign detection, tracking, and recognition using color, shape, and motion information. IEEE Proceed Intell Vehicles Symp 2005:255–260
5. Bochkovskiy A, Wang C-Y, Liao H-YM (2020) Yolov4: optimal speed and accuracy of object detection. arXiv Prepr arXiv200410934.
6. (2023) Brief summary of YOLOv8 model structure. https://github.com/ultralytics/ultralytics/issues/189
7. Chen EH, Röthig P, Zeisler J, Burschka D (2019) Investigating low level features in CNN for traffic sign detection and recognition. In: 2019 IEEE intelligent transportation systems conference (ITSC). Pp 325–332.
8. Cheng Z, Liang J, Choi H et al (2022) Physical attack on monocular depth estimation with optimal adversarial patches. European Conference on Computer Vision, In, pp 514–532
9. Cui Y, Yan L, Cao Z, Liu D (2021) TF-blender: temporal feature blender for video object detection. Proc IEEE Int Conf Comput Vis 8118–8127. https://doi.org/10.1109/ICCV48922.2021.00803
10. Dasgupta S, Hollis C, Rahman M et al (2022) An innovative attack modeling and attack detection approach for a waiting time-based adaptive traffic signal controller. Int Conf Transp Dev 2022:72–84
11. Dumitriu A, Tatui F, Miron F, et al Rip Current Segmentation: A Novel Benchmark and YOLOv8 Baseline Results. 1261–1271
12. Eteifa S, Rakha HA, Eldardiry H (2021) Predicting coordinated actuated traffic signal change times using long short-term memory neural networks. Transp Res Rec 2675:127–138
13. Fang C-Y, Chen S-W, Fuh C-S (2003) Road-sign detection and tracking. IEEE Trans Veh Technol 52:1329–1341
14. Gad AF (2023) Evaluating object detection models using mean Average Precision (mAP). https://blog.paperspace.com/mean-average-precision/
15. Gao B, Jiang Z, Zhang J (2019) Traffic sign detection based on ssd. In: Proceedings of the 2019 4th International Conference on Automation, Control and Robotics Engineering. pp 1–6
16. Garg P, Chowdhury DR, More VN (2019) Traffic sign recognition and classification using YOLOv2, faster RCNN and SSD. In: 2019 10th international conference on computing, communication and networking technologies (ICCCNT). Pp 1–5.
17. Gu Y, Si B (2022) A novel lightweight real-time traffic sign detection integration framework based on YOLOv4. Entropy 24:487
18. Han C, Gao G, Zhang Y (2019) Real-time small traffic sign detection with revised faster-RCNN. Multimed Tools Appl 78:13263–13278
19. Hui J (2023) mAP (mean Average Precision) for Object Detection. https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173
20. Jiang Z, Zhao L, Li S, Jia Y (2020) Real-time object detection method based on improved YOLOv4-tiny. arXiv Prepr arXiv201104244.
21. Jiang P, Ergu D, Liu F et al (2022) A review of yolo algorithm developments. Procedia Comput Sci 199:1066–1073
22. Jocher G, Stoken A, Borovec J et al (2020) Ultralytics/yolov5: v3. 0. Zenodo.
23. Karmakar S, Deb PP (2022) Analyzing the future of autonomous vehicle using machine learning. AIJR Abstr 22

24. la Escalera A, Armingol JM, Mata M (2003) Traffic sign recognition and analysis for intelligent vehicles. Image Vis Comput 21:247–258
25. Liu W, Anguelov D, Erhan D, et al (2016) Ssd: Single shot multibox detector. In: Computer Vision--ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11--14, 2016, Proceedings, Part I 14. pp 21–37
26. Liu D, Cui Y, Tan W, Chen Y (2021) SG-net: spatial granularity network for one-stage video instance segmentation. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit 9811–9820. https://doi.org/10.1109/CVPR46437.2021.00969
27. Liu Y, Gao Y, Zhang Q et al (2022) Multi-task safe reinforcement learning for navigating intersections in dense traffic. J Franklin Inst
28. Lou H, Duan X, Guo J et al (2023) DC-YOLOv8: small-size object detection algorithm based on camera sensor. Electron 12:1–14. https://doi.org/10.3390/electronics12102323
29. Malekzadeh M, Manolis D, Papamichail I, Papageorgiou M (2022) Empirical investigation of properties of lane-free automated vehicle traffic. In: 2022 IEEE 25th international conference on intelligent transportation systems (ITSC). Pp 2393–2400.
30. Marques R, Ribeiro T, Lopes G, Ribeiro AF (2022) YOLOv3: traffic signs \& lights detection and recognition for autonomous driving. In: ICAART 3:818–826
31. Mehta S, Paunwala C, Vaidya B (2019) CNN based traffic sign classification using Adam optimizer. In: 2019 international conference on intelligent computing and control systems (ICCS). Pp 1293–1298.
32. Nikitas A, Michalakopoulou K, Njoya ET, Karampatzakis D (2020) Artificial intelligence, transport and the smart city: definitions and dimensions of a new mobility era. Sustainability 12:2789
33. Olugbade S, Ojo S, Imoize AL et al (2022) A review of artificial intelligence and machine learning for incident detectors in road transport systems. Math Comput Appl 27:77
34. Priscila SS, Sharma A, Vanithamani S et al (2022) Risk-based access control mechanism for internet of vehicles using artificial intelligence Secur Commun Networks:2022
35. (2023) Propulsion shafting arrangement modeling from mechanical drawings using deep learning and YOLOv8
36. Rajendran SP, Shine L, Pradeep R, Vijayaraghavan S (2019) Real-time traffic sign recognition using YOLOv3 based detector. In: 2019 10th international conference on computing, communication and networking technologies (ICCCNT). Pp 1–7.
37. Ren S, He K, Girshick R, Sun J (2015) Faster r-cnn: towards real-time object detection with region proposal networks. Adv Neural Inf Process Syst 28
38. (2023) Robotaxi-full scale autonomous vehicle competition. https://teknofest.org/en/competitions/competition/29
39. Satti SK (2023) Recognizing the Indian cautionary traffic signs using GAN, improved mask R-CNN, and grab cut. Concurr Comput Pract Exp 35:e7453
40. Sestino A, Peluso AM, Amatulli C, Guido G (2022) Let me drive you! The effect of change seeking and behavioral control in the artificial intelligence-based self-driving cars. Technol Soc 70:102017
41. Singh K, Malik N (2022) CNN based approach for traffic sign recognition system. Adv J Grad Res 11:23–33
42. Srinivas Rao P, Gudla R, Telidevulapalli VS et al (2022) Review on self-driving cars using neural network architectures.
43. Sun Y, Ge P, Liu D (2019) Traffic sign detection and recognition based on convolutional neural network. In: 2019 Chinese automation congress (CAC). PP 2851–2854.
44. Tengilimoglu O, Carsten O, Wadud Z (2023) Implications of automated vehicles for physical road environment: a comprehensive review. Transp Res part E Logist Transp Rev 169:102989
45. Terven J, Cordova-Esparza D (2023) A comprehensive review of YOLO: from YOLOv1 to YOLOv8 and beyond. 1–33.
46. Torbaghan ME, Sasidharan M, Reardon L, Muchanga-Hvelplund LCW (2022) Understanding the potential of emerging digital technologies for improving road safety. Accid Anal \& Prev 166:106543
47. van der Aalst W (2022) Six levels of autonomous process execution management (APEM). arXiv Prepr arXiv220411328.
48. Vennelakanti A, Shreya S, Rajendran R et al (2019) Traffic sign detection and recognition using a CNN ensemble. In: 2019 IEEE international conference on consumer electronics (ICCE). Pp 1–4.
49. Wang W, Han C, Zhou T, Liu D (2022) Visual recognition with deep nearest centroids.
50. Wang W, Liang J, Liu D (2022) Learning Equivariant segmentation with instance-unique querying.
51. Yan Y, Deng C, Ma J et al (2023) A traffic sign recognition method under complex illumination conditions. IEEE Access
52. Yasmin S, Durrani MY, Gillani S et al (2022) Small obstacles detection on roads scenes using semantic segmentation for the safe navigation of autonomous vehicles. J Electron Imaging 31:61806

53. (2023) YOLO: A Brief History. https://docs.ultralytics.com/
54. You S, Bi Q, Ji Y et al (2020) Traffic sign detection method based on improved SSD. Information 11:475
55. Zhai D, Jiang J, Ji X Shadows can be Dangerous : Stealthy and Effective Physical-world Adversarial Attack by Natural Phenomenon. 15345–15354
56. Zhang H, Qin L, Li J et al (2020) Real-time detection method for small traffic signs based on Yolov3. IEEE Access 8:64145–64156
57. Zuo Z, Yu K, Zhou Q et al (2017) Traffic signs detection based on faster r-cnn. In: 2017 IEEE 37th international conference on distributed computing systems workshops (ICDCSW). Pp 286–288.