Advanced Sciences and Technology Journal

**ASTJ vol. 2, no. 1 (2025), P 1047**
**10.21608/astj.2025.347656.1047**
https://astj.journals.ekb.eg/

# Design and Analysis of an Autonomous Road Sign Recognition System

Adham Hassan, Athanasious Nashaat, Eithar Reda, Gannah Mohamed, Mohamed Salah, Mohamed Thabet, Omar Mohamed, Veronia Medhat, Youssef Emad, A.M. Elhady*

*Mechatronics Engineering Department, Faculty of Engineering and Technology, Egyptian Chinese University, Cairo, Egypt.*
**abobakr.elhadu@ecu.edu.eg*

A R T I C L E I N F O

**Article history:**
Received 26 December 2024
Revised 7 January 2025
Accepted 15 January 2025
Available online 17 January 2025

**Handling Editor:**
**Prof. Dr. Mohamed Talaat Moustafa**

**Keywords:**
Road sign detection
YOLO V8, GTSDB Dataset, Convolution Neural Network, Deep learning

A B S T R A C T

To ensure efficient and safe traffic flow, it is essential that all drivers should follow the road sign strictly. Car industrial technology is heading towards autonomous driving. And to make auto driving safe, it should be flawless and fault-free. Therefore, the system must read all road signs correctly. There must be an awareness competent that makes the system able to comprehend and interpret those road signs. The problem of our matter is that if we want to ensure safe flow effectively, our system must read and interpret all road signs correctly. Our current state acknowledges that there is a driver who has lost concentration while driving. So, we create a system that recognizes road signs and interprets them into voice alerting the driver. The system completely depends on the You Only Look Once V8 NANO and German Traffic Sign Recognition Benchmark Dataset. We have chosen the German Traffic Sign Recognition Benchmark because it's the most similar to what is actually on Egyptian roads.

## 1. Introduction

Traffic signs are essential for maintaining proper traffic flow and improving driver safety on city streets and motorways. Because these signs are critical to environmental awareness and the operation of self-driving cars, their exact and timely detection and recognition is an important field of self-learning vehicle study. In order to regulate traffic, create intelligent transportation systems, and enable cutting-edge driver-assistance and autopilot technology, traffic signs are essential components. Road safety cannot be ensured by driver education alone, especially as autonomous vehicles with self-driving features proliferate. As a result, detection models for these systems need to show that they can quickly and effectively identify traffic signs in real time. Existing visualization and recognition methods, however, have some safety flaws that highlight the need for more study in this area. Traffic Sign Recognition and Traffic Sign Detection are two separate processes involved in traffic sign identification and detection. Traditional Traffic Sign Detection techniques mostly rely on manually removing characteristics of the original dataset. For feature extraction based on geometric patterns, perimeter detection, and shade information, manual techniques used color centric approaches separate zones with traffic signs utilizing sophisticated color spaces such as Hue-Chroma-Luminance (HCL) [2] and Hue-Saturation-Intensity (HSI) [1]. Fortunately, these systems were laborious and expensive because they necessitated manually editing a vast volume of pictures. Although they were also commonly used, shape and color-oriented approaches [3] have limitations such as sensitivity to illumination changes, occlusions, scale alternatives, rotations, and translations. Machine learning requires large, annotated datasets, but it can help with some of these problems. Deep Learning has developed as a potent tool for Traffic Sign Recognition in recent years, yielding amazing results. The purpose of traffic signs is to swiftly grab people's attention. Techniques that rely on features created by hand [1] [4]–[7] Use these signs' visual characteristics during the feature extraction procedure. Although many things mimic traffic signs, these methods are not very reliable in differentiating real signs from fake ones world-wide. It is difficult to accurately depict the distinctive qualities that set traffic signs apart when using low-level handcrafted features. Applications for Convolutional Neural Networks (CNNs) include You Only Look Once (YOLO) [8], Single Shot multi-box Detector SSD [9], Cascade R-CNN [10], Faster R-CNN [11], and Fast R-CNN [13] have shown to operate exceptionally well and have been routinely used as detectors.

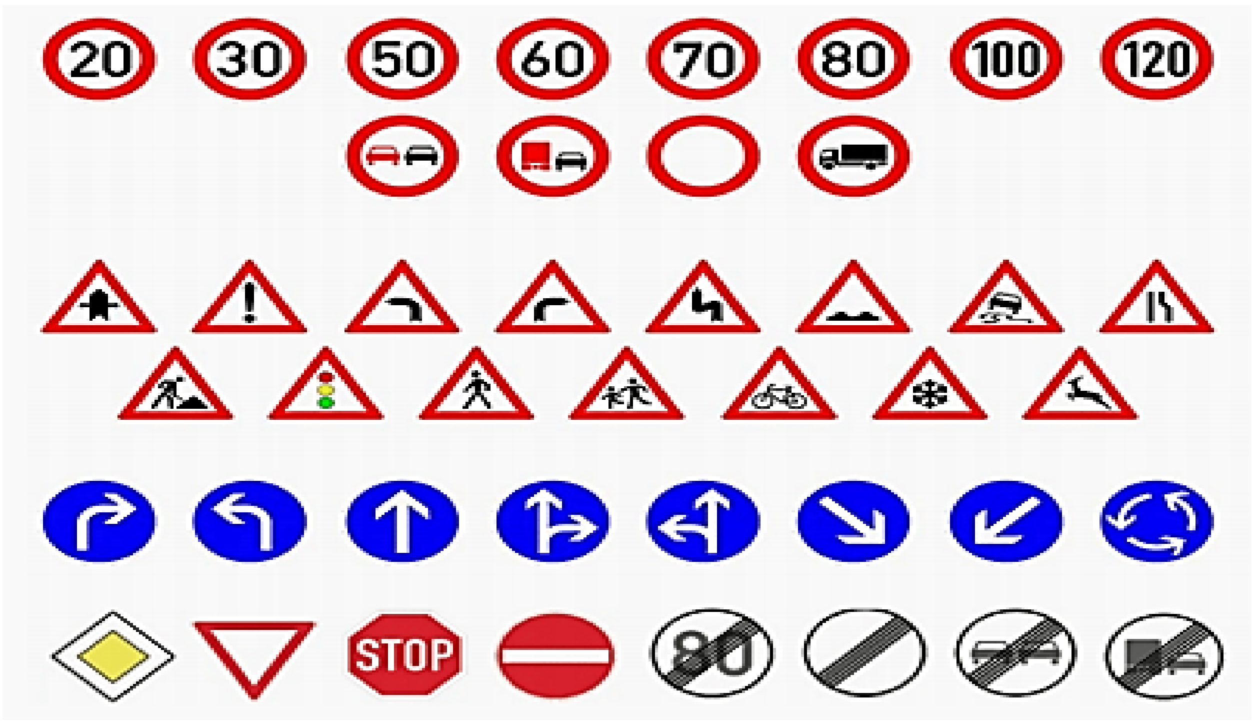


**Fig.1** Categories of GTSRB dataset [14]

Figure 1 depicts the categories the German Traffic Signs Recognition Benchmark (GTSRB) dataset: forbidden, mandatory, dangerous, and other. These classes are further split down into 43 sub-classes. Conventional accuracy measurements, such as Mean Average Precision (MAP), must be examined alongside additional characteristics, like as memory consumption and computational times, to identify the optimum instrument for a certain application. For example, autonomous vehicles must be able to recognize things reliably and in real time, whereas self-driving cars can benefit from less complex model designs that use fewer RAM chips. The present research evaluates techniques for detection for classifying traffic indicators. These signs are separated into various groups, each having subclasses that share similar shapes and proportions but convey different meanings. Traffic sign categorization normally involves two steps: establishing the sign's location and assigning it to the appropriate category. During detects the sing's enclosing box, and the classification stage divides it accordingly. assigning it to the appropriate category.

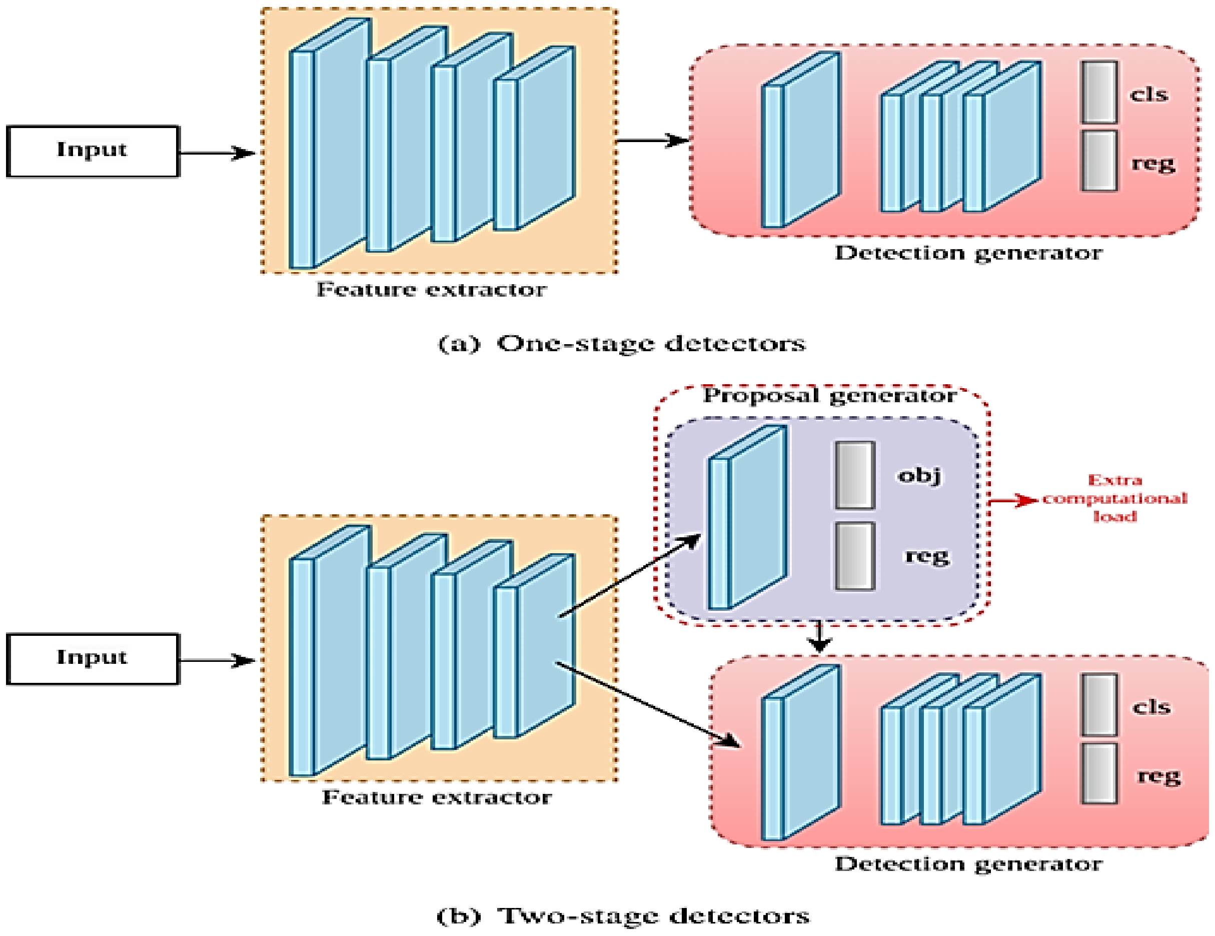


**Fig 2.** Meta-architectures for deep-learning-based detection models [45]

Figure 2 illustrates how advanced detectors are divided into two groups: two-stage detectors and one-stage detectors [8], [9], which are capable of predicting the full image in a single pass [10]–[12] that use a sequence of detectors in a multistage detection process where each stage improves on the findings of the one before it. The results of current two-phase traffic sign analysers [13],[15],[16] have proven promising. Currently, vehicle-mounted monitors collect photos of traffic signs, which are subsequently analysed using computer vision and pattern recognition techniques. As algorithms for deep learning progress, object identification methods such as YOLO become increasingly important for traffic sign detection. Regarding these, one-stage diagnostic approaches are preferred for their efficiency. The YOLO algorithm, introduced by Redmon [17], has expanded rapidly, winning plaudits for its exceptional speed and accuracy. YOLO proposes using an end-to-end neural network that makes predictions of bounding boxes and class probabilities all at once. It differs from the approach taken by previous object detection algorithms, which repurposed classifiers to perform detection. Following a fundamentally different approach to object detection, YOLO achieved state-of-the-art results, beating other real-time object detection algorithms by a large margin.

While algorithms like Faster RCNN work by detecting possible regions of interest using the Region Proposal Network and then performing recognition on those regions separately, YOLO performs all of its predictions with the help of a single fully connected layer. YOLO has many versions starting from 2016 YOLO v2 till 2023 YOLO v8. YOLOv8, in particular, is widely used due to its balance of speed and precision, and it has five model revisions tailored to a variety of computing needs. This analysis is based on the YOLOv8 and GTSRB databases. YOLOv8 is the newest state-of-the-art YOLO model that can be used for object detection, image classification, and instance segmentation tasks. YOLOv8 was developed by Ultralytics, who also created the influential and industry-defining YOLOv5 model. YOLOv8 includes numerous architecture, and developers experience changes and improvements over YOLOv5.

The following are this article's main contributions:

We will use YOLO v8 nano, a lightweight and efficient version of the YOLO architecture, can be used in autonomous sign recognition systems to quickly detect and classify traffic signs in real-time. YOLOv8 nano enables reliable Traffic Sign Identification and Knowledge for self-navigating vehicles in a variety of road layouts and shining scenarios. Additionally, YOLO v8 nano considers all GTSRB categories.

## 2. Related work

### 2.1. CNN

Citlalli Gamez Serna, Yassine Ruichek (IEEE) [18] CNN with the GTSRB divides signs into three categories: danger, prohibitory, and mandatory, consisting of 82,476 signs across 164 classes. A 70-30% split is used for training and testing datasets. sign's location and assigning it to the appropriate category. During detection, the system detects the sing’s enclosing box, and the classification stage divides it accordingly.

Jingwei Cao, Chuanxue Song, Silun Peng, Feng Xiao and Shixin Song [19] and Gangyi Wang, Guanghui Ren, Lihui Jiang, Taifan Quan [20] LE-NET CNN, Hue Saturation Value (HSV): H denotes the hue or color type, S specifies the colors saturation (quality), and V measures illumination with a value of 1 indicating white and 0 Representing black. Region of Interest (ROI) focuses on filling processes. The dataset utilized for both training and testing consists of 39,209 images for training and 12,630 images for testing, accounting for approximately 75% and

25% of the overall dataset, respectively.

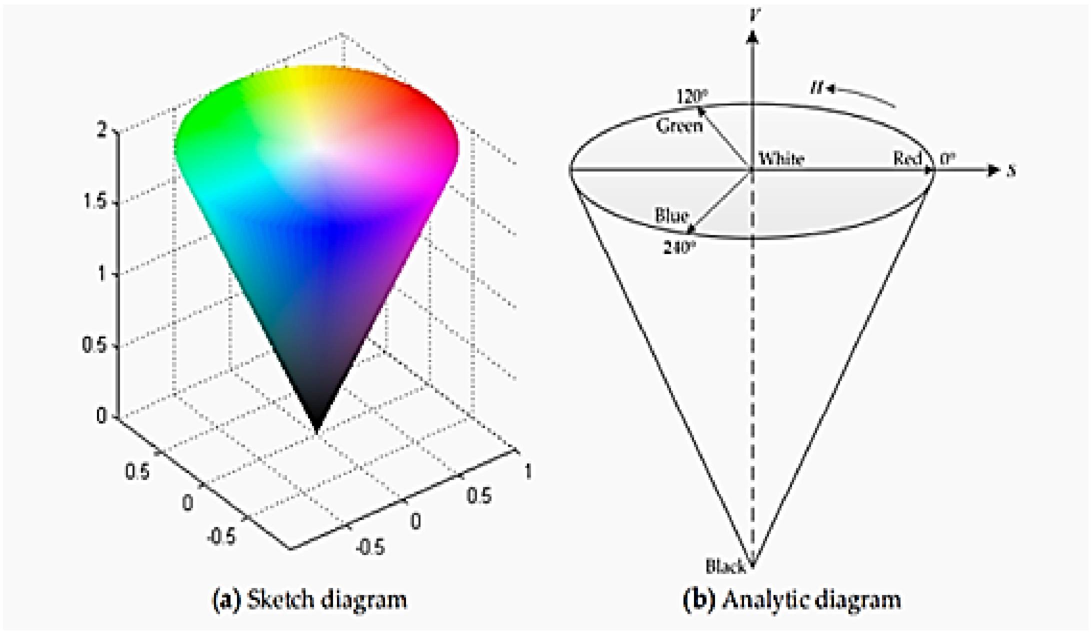


**Fig.3**. The HSV color space [19]

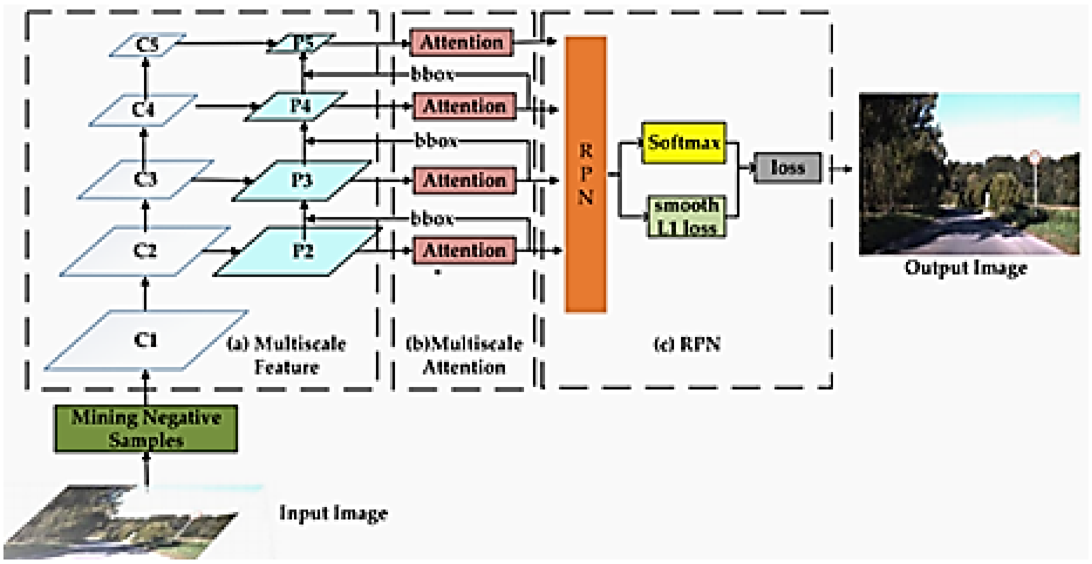


**Fig 4.** Cascaded R-CNN with multiscale attention and imbalanced samples [21].

Jianming Zhang, Zhipeng Xie, Juan Sun, Xin Zou, Jin Wang [21] and Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun and R. B. Girshick. [22] Use R-CNN with the multiscale cascaded object detection network relies on ResNet50 as its backbone network. Five scales are derived using top-down convolution to reduce degradation as the depth of convolutional layers increases {c1, c2, c3, c4, c5}. Multiscale features in pyramids (MFP) include both spatial and semantic data at both high and low resolutions {p1, p2, p3, p4, p5}. Multi-scale attention: Improves detection accuracy in real-world complicated environments and identifies hard negative samples comparable to traffic signals. Region Proposal Network (RPN) generates region proposals, extracts characteristics, performs classification, and provides precise object location. It significantly enhances both speed and accuracy. The testing and training of our method are 98.7% and 90.5% in GTSDB, 99.7% and 83.62% in Chinese City Traffic Sign Detection Benchmark (CCTSDB) and 98.9% and 85.6% in Lisa dataset respectively.

U. Syed Abudhagir, N. Ashok [23] Use the GTSRB dataset that comprises over 50,000 photos split into 39,209 training images and 12630 testing images. The le -Net CNN model with Open Computer Vision (Open CV) obtained an outstanding 98.50% accuracy, proving its ability to classify traffic signs with great precision.

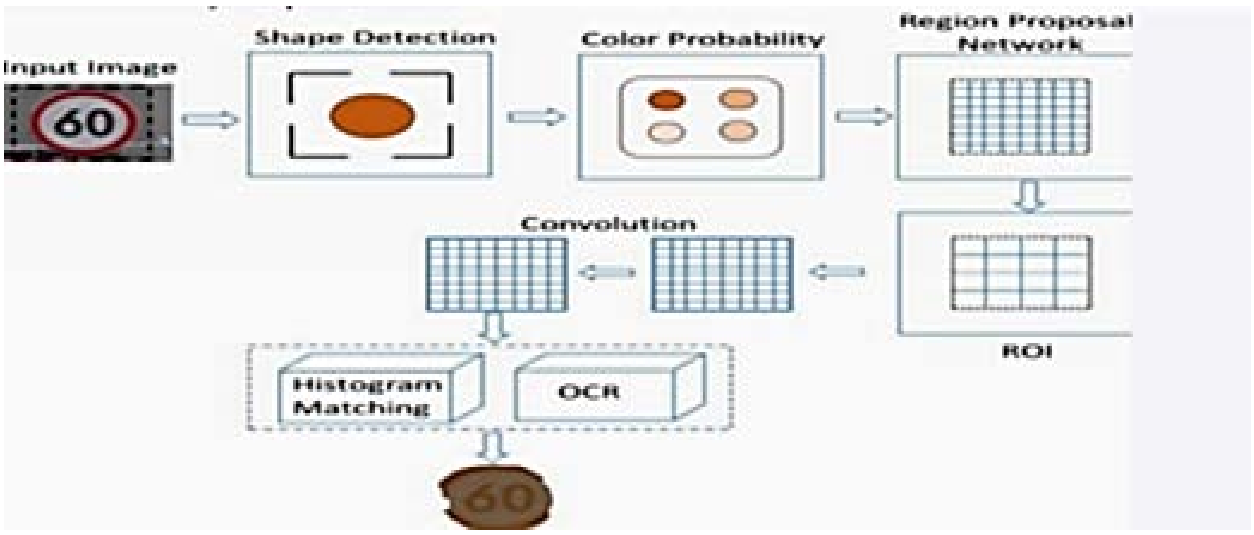


**Fig.5.** CNN Architecture [24].

Pratush Jadoun, Nikita Wanve, Namrata Khandagale, Megha Shinde, Preeti Yadav [24] and Rahul Chauhan, R. C. Joshi [25] CNNs, TensorFlow, and OpenCV were utilized for real-time road sign classification. Input Image: Taking the image from camera using OpenCV, TensorFlow can be employed to process video feeds from cameras mounted on vehicles or at intersections.as shown in Figure5, Pre-Processing allows to eliminate unwanted distortions and improve specific qualities. Its purpose is to raise the images quality. TensorFlow can be used for data preprocessing tasks, such as resizing, normalizing, and augmenting the traffic sign images. Color Segmentation is a method used in machine learning to detect traffic signs. It uses color information from high resolution images to select regions of interest (ROIs). Shape Detection: Based on the area of the counters, the shape of the traffic sign is detected with the threshold parameters defined by the user. Feature Extraction: Feature extraction in TSD using CNN involves automatically learning and capturing relevant visual patterns, such as edges, textures, and shapes from input images.

Model Evaluation: After training, TensorFlow is used to evaluate the performance of the trained models on validation and test datasets. Model Deployment: TensorFlow provides various options for deploying models, including TensorFlow Serving for serving models in a production environment. Output: If Traffic sign matches with the given dataset then we display Detected Sign. To detect traffic sign patterns, the system used a Multi-Layer Perceptron (MLP) combined with a backpropagation learning technique. This architecture was useful for categorizing different sorts of traffic signs using labelled data. Furthermore, a CNN model with 4 layers of convolution was used for traffic sign categorization, by utilizing the GTSDB dataset, a commonly recognized benchmark for testing detection of traffic signs systems. It uses color information from high-resolution images to select ROIs.

### 2.2. You Only Look Once v2

Joseph Redmon, Ali Farhadi [26] Use YOLO v2 that create. Every grid creates boundaries based on the previous box anchors. When the bounding box covers the object, YOLO additional parameter which increases the amount lost from bounding box coordinate forecasts a parameter to reduce the loss resulting from confidence estimates that the provided box does not contain the item. the Mean Average Precision (MAP) 0.5: ~80%-85%, MAP50:95: Not widely reported by GTSRB.

Ross Girshick [27] Use YOLO v2 with GTSDB has 900 images: 600 for training and 300 for testing. The CCTSD has 1100 images 700 for training, 400 for testing. When the bounding box covers the object, YOLO additional parameter, which increases the amount lost from bounding box coordinate forecasts a parameter to reduce the loss resulting from confidence estimates that the provided box does not contain the item. Using GTSDB has 900 images: 600 for training and 300 for testing. The CCTSD has 1100 images, 700 for training, and 400 for testing.

Jianming Zhang, Manting Huang, Xiaokang Jin, Xudong Li [28] GTSDB divides traffic signs into three separate groups: required signs (blue color and circular shape), risk signs (red color and triangular shape), and impossible signs (red color and circular shape). There are 900 images: 600 for training and 300 for testing. The CCTSD has 1100 images (700 for training, 400 for testing). It also categorizes CCTSD into three categories: necessary signs (blue color, circular or rectangular shape), risk indicators (yellow color, triangles shape), and restricting signals (red color, circular shape). The most rapid detection speed recorded was 0.017 seconds per image. To accomplish real-time CCTSDB, we build DCNN based on the end-to-end detection technique. The technique is illustrated using the CCTSDB.

Every grid creates boundaries based on the previous box anchors. When the bounding box covers the object, YOLO additional parameter, which increases the amount lost from bounding box coordinate forecasts the optimal bounding box is determined by the value of IOU. YOLO additionally includes a parameter to reduce the loss resulting from confidence estimates that the provided box does not contain the item.

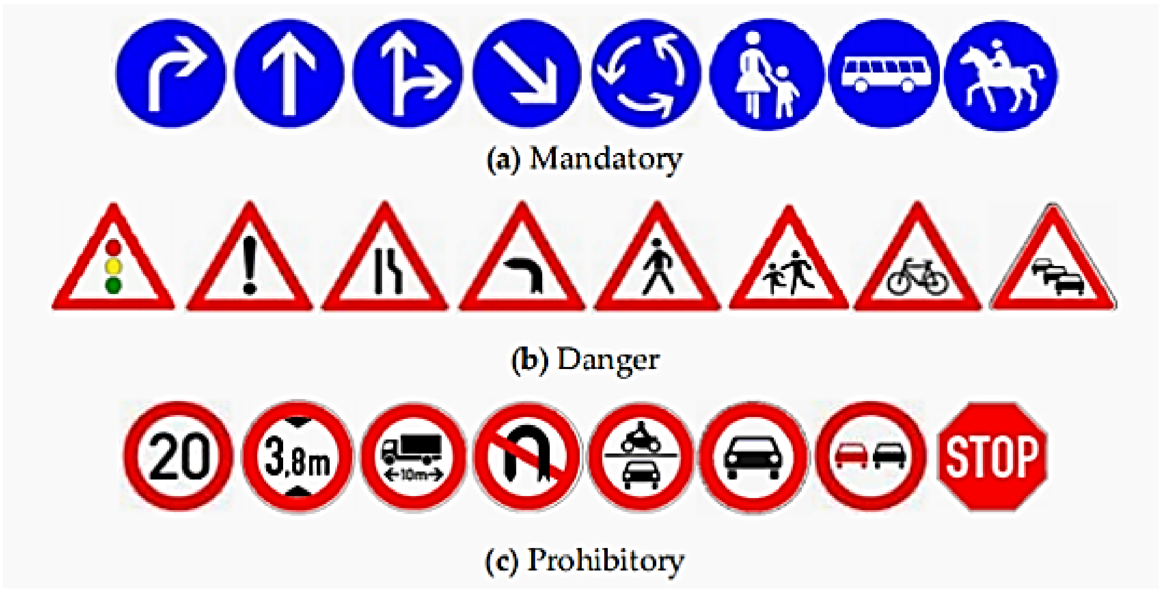


**Fig.6.** Samples of the three super categories in Germany [28]

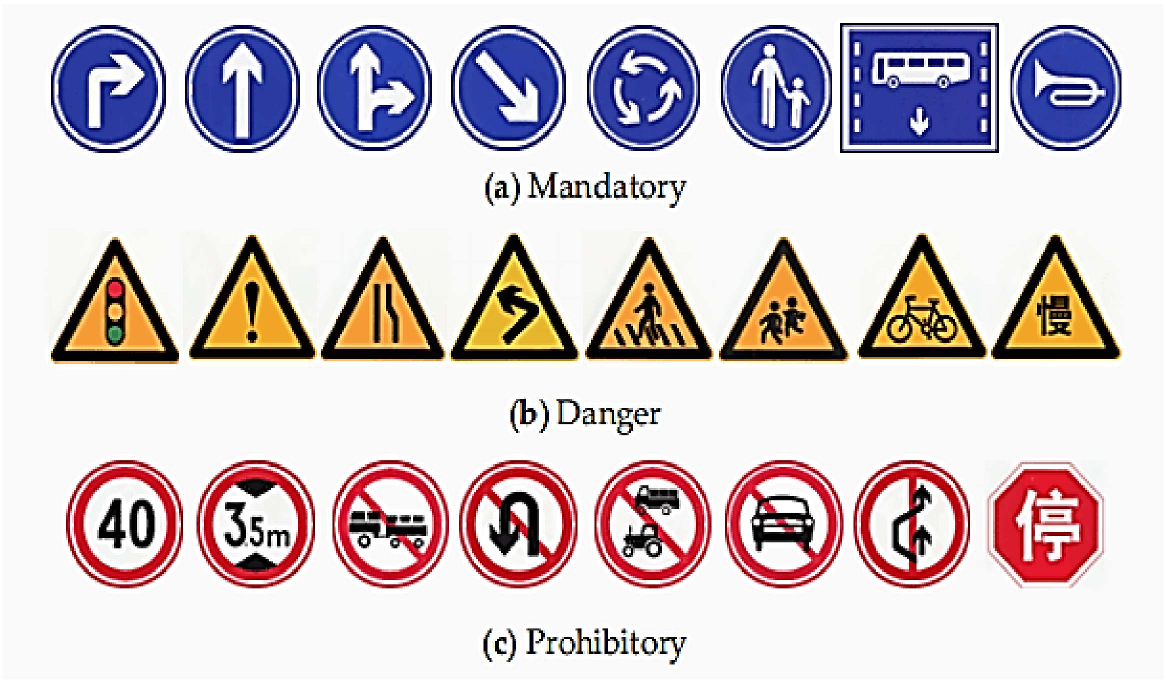


**Fig.7.** Samples of the three super categories in China [28].

## 2.3. YOLO v3

Huibing Zhang, Longfei Qin, Jui Li, Yunchuan Guo, Ya Zhou, Jingwei Zhang, Zhi Xu [29] and Giang Son Tran, Damien Gratadour, Cuong Nguyen[30] YOLO v3, Darknet53 and the feature pyramid structure. Darknet53, the feature maps that are outputted by many consecutive layers are of the same size. The feature pyramid structure is created to improve the locating accuracy on small objects in high-resolution images. Multi-Scale Attention (MSA) YOLO v3 used to improve the detection ability of YOLO v3 on small objects. MSA YOLO v3 with multiscale spatial pyramid pooling module is introduced to Darknet53 to spatial pyramid pooling module is introduced to Darknet53 to improve the detection accuracy of small targets with rich local region features. MAP reaches up to 86%.

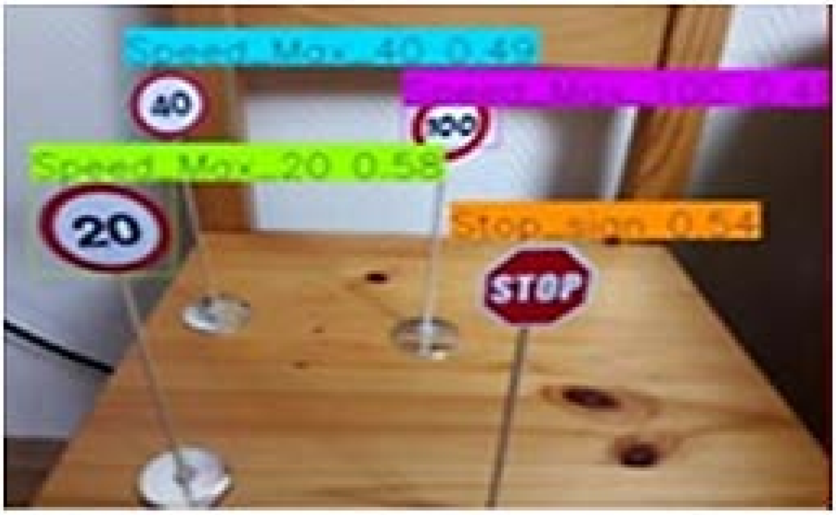


**Fig.8.** The model in light [31]

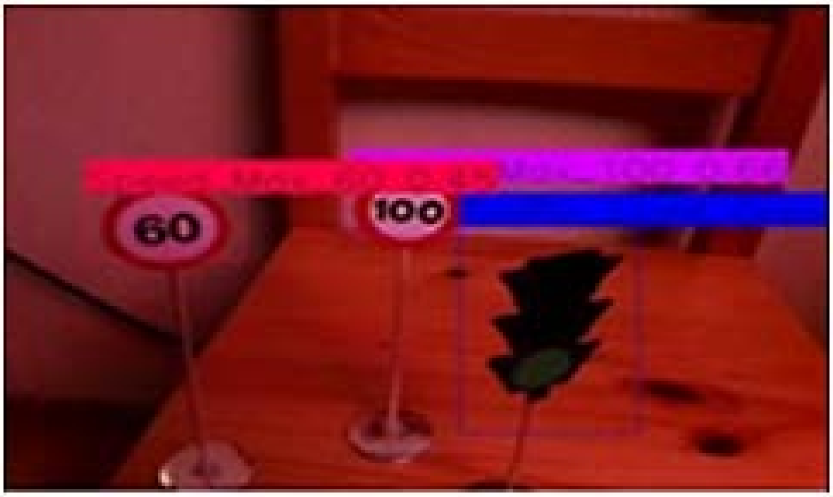
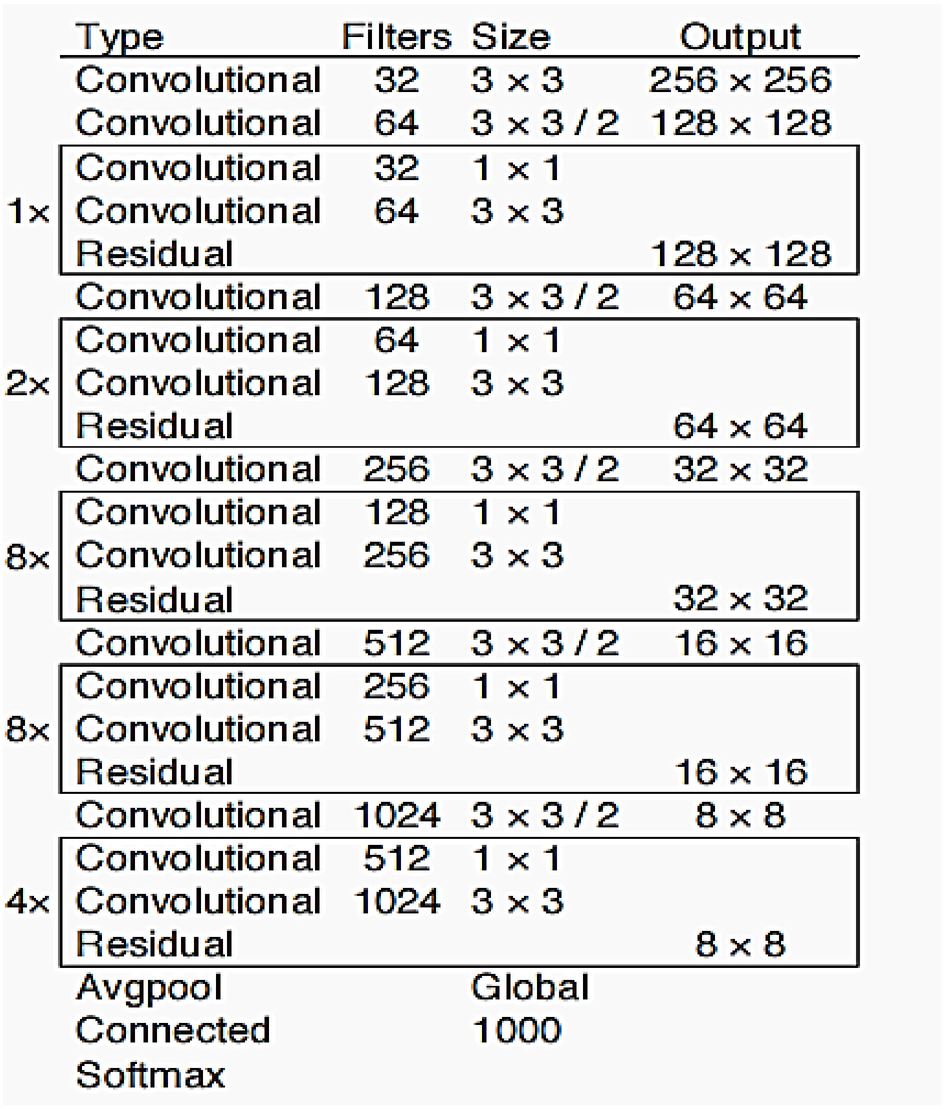


**Fig. 9.** The model in low light [31]



| | Type | Filters | Size | Output |
|---|---|---|---|---|
| | Convolutional | 32 | 3 × 3 | 256 × 256 |
| | Convolutional | 64 | 3 × 3 / 2 | 128 × 128 |
| | Convolutional | 32 | 1 × 1 | |
| 1× | Convolutional | 64 | 3 × 3 | |
| | Residual | | | 128 × 128 |
| | Convolutional | 128 | 3 × 3 / 2 | 64 × 64 |
| | Convolutional | 64 | 1 × 1 | |
| 2× | Convolutional | 128 | 3 × 3 | |
| | Residual | | | 64 × 64 |
| | Convolutional | 256 | 3 × 3 / 2 | 32 × 32 |
| | Convolutional | 128 | 1 × 1 | |
| 8× | Convolutional | 256 | 3 × 3 | |
| | Residual | | | 32 × 32 |
| | Convolutional | 512 | 3 × 3 / 2 | 16 × 16 |
| | Convolutional | 256 | 1 × 1 | |
| 8× | Convolutional | 512 | 3 × 3 | |
| | Residual | | | 16 × 16 |
| | Convolutional | 1024 | 3 × 3 / 2 | 8 × 8 |
| | Convolutional | 512 | 1 × 1 | |
| 4× | Convolutional | 1024 | 3 × 3 | |
| | Residual | | | 8 × 8 |
| | Avgpool | | Global | |
| | Connected | | 1000 | |
| | Softmax | | | |

**Fig. 10.** Darknet-53 [32]

YOLO v3, a Python-based picture processing algorithm with the help of an NVIDIA Compute Unified Device Architecture (CUDA) enabled Graphical Processing Unit (GPU), the system can achieve a frame rate of 40. The images were taken with a camera linked to a PC under a variety of situations, including diverse angles, various backgrounds, conditions of light, and visibility.

Joseph Redmon, Ali Farhadi [32] Use YOLO v3 improved upon YOLOv2 by adopting Darknet-53 as the backbone, offering better feature extraction. It also incorporated multi-label classification and better handling of small objects. And as shown in Figure 10. That explain the Darknet-53 that used with YOLO v3 uses successive 3 × 3 and 1 × 1 convolution layers but now has some shortcut connections as well and is significantly larger. It has 53 convolution layers. MAP 50: ~85%-90%, MAP 50:95: ~40%-50%.

## 2.4. YOLO v4

Mr.Rishabh Singh, Mr.Momin Danish, Mr.Vipul Purohit, Prof. Ashraf Siddiqui [33] and Alexey Bochkovskiy Chien-Yao Wang, Hong-Yuan Mark Liao [34] YOLO v4 scored a MAP of 91.96% after 3,000 iterations on the popular traffic sign recognition benchmark, GTSDB.So the MAP50: ~90%-92% MAP50:95: ~50%-55%.

The CCTSD Dataset includes four sorts of signs: crosswalks, stops, traffic lights, and speed limits. On this data set, the model got a remarkable MAP of 94.78% after 2,000 iterations.

Njayou Youssouf [35] Two advanced models, YOLO v4 and faster R-CNN models, were tested for traffic sign detection. The Faster R-CNN model employs a two-phase detection system: the first stage generates region proposals, while the second stage does classification and bounding box regression. The architecture of the first stage consists of four components: The image is processed as input. Backbone: The feature extract network is utilized. Neck: A layer that enhances feature integration. Dense Prediction: The final prediction is based on revised bounding boxes and classifications. The GTSDB dataset was utilized for training, with traffic signs classified as prohibitive, mandatory, or dangerous. Testing achieved a very high accuracy of 99.20%.

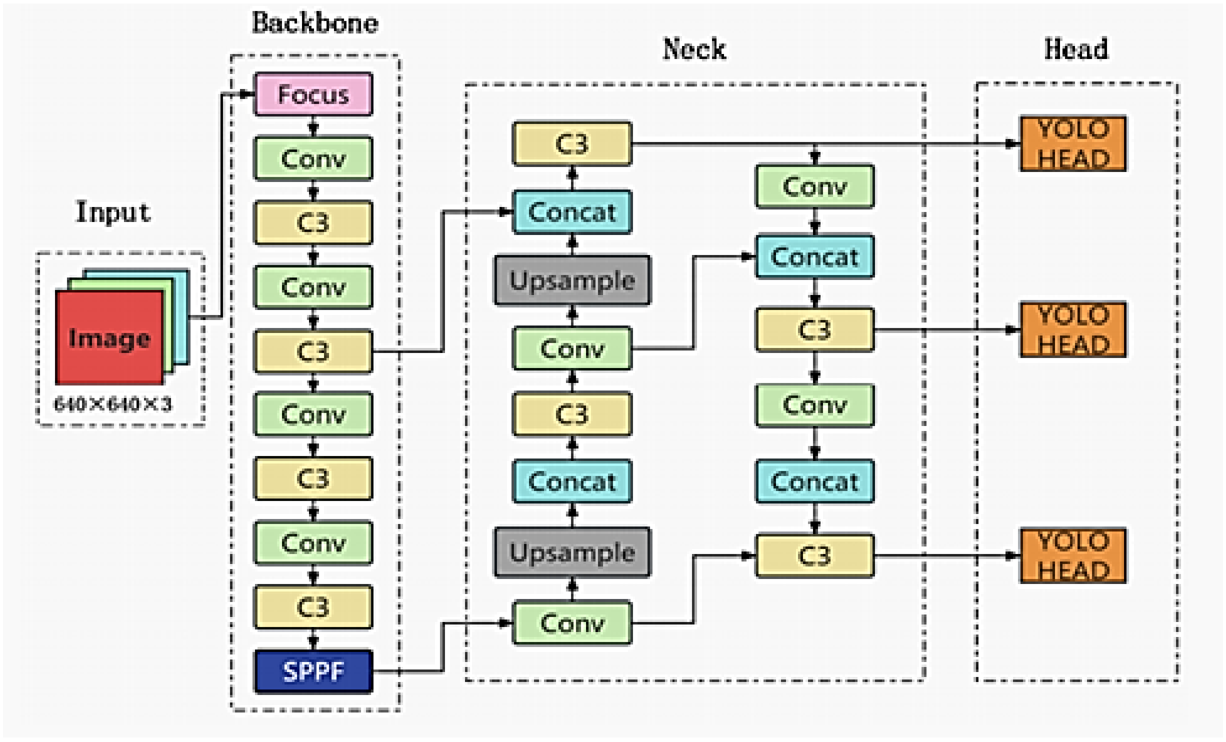


**Fig. 11.** YOLO v5 network architecture [38].

## 2.5. YOLO v5

Yanzhao Zhu, Wei Qi Yan [36] Shubham Gore, Manan Bhasin, Suchitra S [37] YOLO v5 was used for detection of traffic signs and compared with the SSD model. YOLO v5, a CNN developed for optical recognition of objects using deep neural net-works, attained a rate of processing of 30 FPS, while SSD performed at a slower 3.49 FPS. YOLO v5 is divided into four primary parts: input, backbone, neck, and prediction layer. The model performed well, with a MAP50 of 97.7% on the information provided. The dataset was custom dataset separated into two portions for SSD model training and validation: 80% for training-validation (64% trained and 16% validated) and 20% for testing. The MAP of YOLO v5 with GTSRB data set MAP50: ~93%-95% MAP50: 95: ~55%-60%.

Tianxin Han, Lina Sun, Qing Dong [38] and Junfan Wang, Yi Chen, Zhekang Dong, Mingyu Gao [39] For recognizing small traffic sign objects, we exclusively used YOLO v5

Backbone: The Focus module takes an innovative way to down sampling input photos, enabling it to efficiently collect relevant information while decreasing computing complexity. Conv layers provide powerful feature extraction capabilities. The Bottleneck Cross Stage Partial (CSP) module increases the network's capacity while minimizing memory usage. This Spatial Pyramid Pooling (SPP) module is intended for effectively obtaining multi-scale features by pooling features at various resolutions in a pyramid-like pattern. Neck: The Upsample layers increase the clarity of feature maps. Head: The recognition head module detects multi-scale objects on feature maps collected by the backbone system. The object recognition precision in-creased to 95.0%, with a recall rate of 91.6% and a mean precision of 95.4%. The resulting 11,575 photos were separated into three sets: a testing set of 2315 images, a training set of 6945 images, and a validation set of 2315 images, in a 2:6:2 ratio.

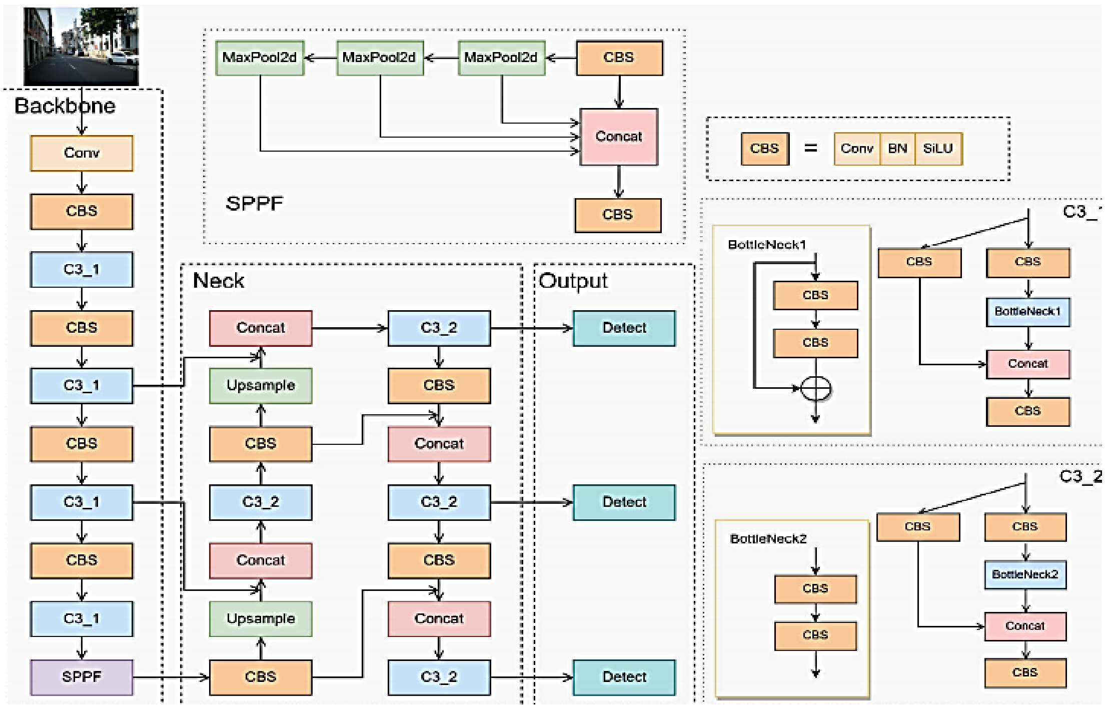


**Fig. 12.** The construction of YOLO v5s [41].

Shenming Qu, XinyuYang, Huafei Zhou , Yuan Xie[40] Qiannying Wang, Xiangyu Li, Ming lu [41] Use YOLO v5s with last version 6.0. consists of four parts: input, back-bone, neck and output.
As shown in Figure 12, Backbone: convolution layer replaces the previous Focus module on the first layer of the network for down-sampling operation. (CBS) module encapsulates the Convolution layer, Batch processing layer and SiLU Activation function. Neck: The Up-sample layers increase the clarity of feature maps. The output layer uses Complete Intersection over Union (CIOU) to compute regression loss of bounding box and predict for the three images' feature with different sizes. CCTSDB data set, the precision of our model reached 98.1%, the recall of our model reached 97.6% and the MAP 98.8% with a speed of 91 FPS.

## 2.6. YOLO v6

Chuyi Li, Lulu Li, Hongliang Jiang, Kai-heng Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, Yiduo Li, Bo Zhang [42] Juan R. Terven, Di-ana M. Cordova-Esparaza [43] Xiaohan Ding, Xiangyu Zhang , Ningning Ma, Jungong Han, Guiguang Ding, Jian Sun [44] Haoyang Zhang, Ying Wang, Feras Dayoub, Niko S¨ underhauf [45] YOLO v6 equivalent to YOLO v4 and YOLO v5, it offers a variety of variants in varying sizes for use in industry. YOLO v6 used an anchor-free scanner. Efficient Rep, a novel backbone based on Representative Visual Geometry Group (Rep VGG), uses greater parallelism than prior YOLO backbones. YOLO v6-L obtained a MAP of 52.5% and a MAP 50 of 70% at roughly 50 FPS on an NVIDIA Tesla.

## 2.7. YOLO v7

Ruturaj Mahadsheeti, Jinsul Kim and Tai-won UM [46] used Sign-yolo based on YOLO v7 [47] which is combined with a squeeze-and-excitation (SE) block and a specific attention mechanism in the Sign-YOLO technique. YOLO v7 is a single-stage detector that is ideal for real-time applications because it can produce predictions for the full image in a single pass. By specifically capturing the connections between the channels in its convolutional features, the SE block improves the caliber of representations produced by a network. By using global information to selectively highlight significant traits and downplay less significant ones, it allows the network to recalibrate features. CNN's also play a vital role in Traffic Sign Detection. Nowadays, the CNN algorithm, a well-liked Deep Learning technique, is widely used in a variety of fields, such as computer vision, natural language processing, and visual alignments of semantics [48-50]. One-stage detection and two-stage detection are the two forms that can be distinguished based on the region proposal's necessity. Because of its quick performance, one-stage detection is frequently used in situations like traffic detection. Shao et al. [51], [52] demonstrated an improved version of Faster R-CNN intended for traffic sign identification. They improved the network's recognition speed by optimizing the Gabor wavelet by putting in place a regional recommendation mechanism. Data augmentation is essential for network optimization and has demonstrated efficacy in a range of vision tasks [2], [11] and [53]. It protects against overfitting and improves CNN performance [54] and provides a simple implementation [55]. There are two primary categories of data augmentation techniques: geometric transformation (which includes zoom, translation, random cropping, and rotation) and color transformation (which includes contrast, blur, noise, and color casting) [56].

These methods use oversampling or data warping to artificially increase the size of the training dataset. And for the Baseline, In the field of computer vision, YOLO is a popular and effective object detection approach. Its exceptional accuracy in real-time object detection is what makes it so effective. In contrast to the most recent YOLO model, YOLO v7 [47] proves to be the best option for high-resolution inference, although having a little slower speed. Because of its portability and simplicity, YOLO v7 is appropriate for real-world applications requiring great accuracy and little memory. For the dataset, Italy [57], China [58], Germany [15], Belgium [59], the United States [60], Croatia [61], Sweden [5], and many more nations have assembled publicly available databases with traffic sign data. The GTSDB dataset [15], which is frequently used to compare Traffic Sign Detection methods, was employed in the experiments conducted for this investigation, see Table 1.

**Table.1.** Compare between YOLO's Versions

| **YOLO Version** | **MAP 50** | **MAP 50-95** | **References** |
|---|---|---|---|
| YOLO v2 | 80%-85% | Not widely reported | [26] |
| YOLO v3 | 85%-90% | 40%-50% | [32] |
| YOLO v4 | 90%-92% | 50%-55% | [34] |
| YOLO v5 | 93%-95% | 55%-60% | [37] |
| YOLO v6 | 92%-94% | 58%-62% | [42] |
| YOLO v7 | 93%-96% | 60%-65% | [47] |

## 3. Problems and Solve

Bin Ji, Jiafeng Xu, Yang Liu, Pengxiang Fan, Mengli Wang [62], Rahul Kumar, Aniket Gupta, Rajeswari D [63], Sumit S Shevtekar[64] Yashank Singh Dev Singh  and Gufran Ali [65] Zhongjie Huang, Lintao Li, Gerd Christian Krizek, Linhao Sun [66] Unreliable Traffic Sign Recognition: Current models typically fail to correctly identify road signs, particularly in inclement weather conditions such as mist, rain, or darkness. Poor Effectiveness in Adverse Weather: Current models are not resistant to weather variations, resulting in missed identifications or incorrect classification. Diverse Traffic Instances: Different traffic situations, such as towns, cities, and suburban locales, create unique challenges due to differences in sign types, shapes, and positions. Destroyed or clouded markers: Trees and constructions can obscure or damage signs, making them difficult to interpret. Computing Efficiency: Real-time applications necessitate models that can analyse pictures rapidly and efficiently. Bin Ji, Jiafeng Xu, Yang Liu, Pengxiang Fan, Mengli Wang [62] Building an improved traffic sign detection model based on YOLO v8 able to increase accuracy in traffic sign detection and improved robustness to adverse weather conditions

## 4. YOLO v8

Sumit S Shevtekar [64] and Venkata Rami Reddy Ch [67] Songjie Du, Weiguo Pan, Nuoya Li, Songyin Dai, Bingxin Xu, Hongzhe Liu, Cheng Xu, Xuewei Li [68] YOLO v8 was launched in January 2023 by Ultralytics, the company that created YOLO v5. YOLO v8 has five size options: YOLO v8n (nano), YOLO v8s (tiny), YOLO v8m (medium), YOLO v8l (massive), and YOLO v8x (extra-large). YOLO v8 adds new capabilities and optimizations, making it an excellent solution for a variety of object identification duties in a wide range of projects. Multiple algorithms and strategies are used to train a YOLO v8 model, optimizing its capacity to identify and categorize objects in photos.

### 4.1 Yolo Architecture:

The YOLO v8 architecture consists of two key components: the backbone and the head, both that use a complete CNN.

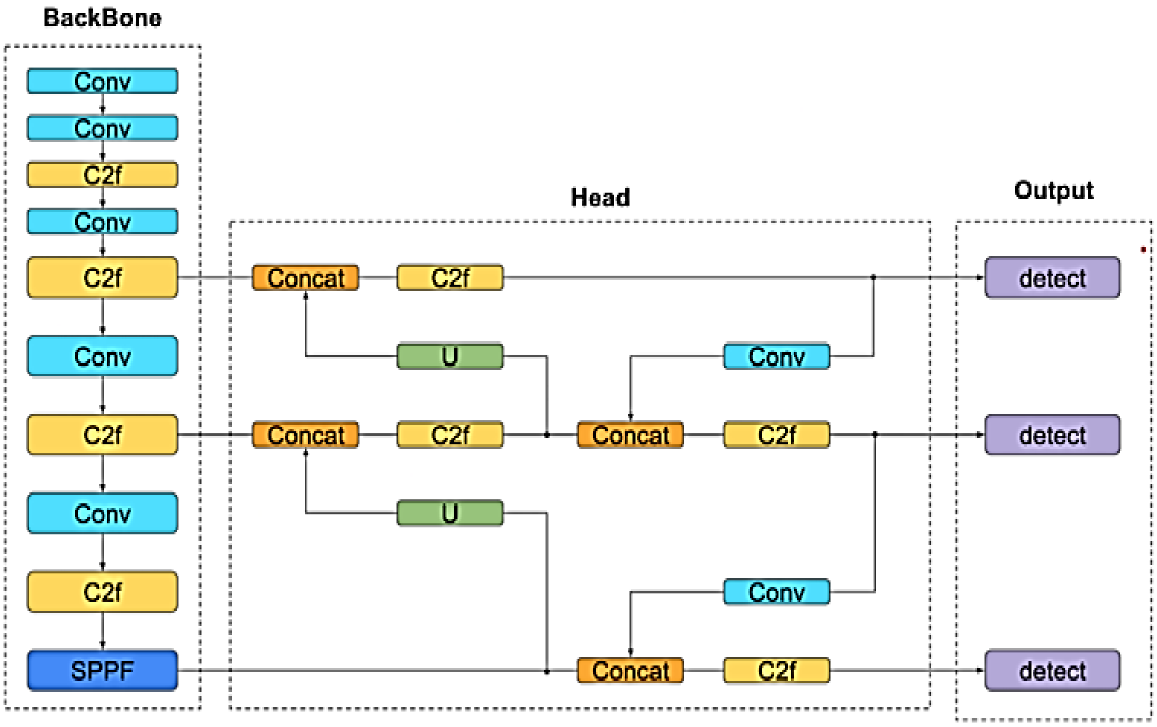


**Fig. 13.** YOLO v8 Architecture [67].

As shown in Figure13.The backbone: system is an upgraded version from the CSPDark-net53 design that employs a technology known as CSP links to increase the flow of data between network layers. The structure includes convolution layer for learning spatial hierarchies. The C2f module combines high-level characteristics with contextual information for improved detection accuracy. Spatial Pyramid Pooling Faster: enhances network performance to handle objects of different sizes by aggregating multi-scale spatial information efficiently. SPPF and convolution layers process features at different sizes. The head: was designed to be removable, which means that it may manage objectless scores, categorization, and regression analysis independently. The Object Recognition Head in YOLO v8 is the model's final stage, where collected and processed properties are converted into predictions.

Consist of:

- The U layers: improve the resolution of feature maps.
- Convolutional layers are used to analyze the feature maps, followed by a linear layer that predicts bounding boxes and class probabilities.

### 4.2. Activation of Sigmoid:

The detection head of YOLO v8 nano uses the sigmoid activation function for:

*4.2.1. Probability of Class Forecast:*

The sigmoid represents the likelihood that an object belongs to a particular class by mapping raw scores (logits) to a range of [0, 1].

The model forecasts probability class for every grid cell for probability class item types. The equation is:

$$\mathrm{p(c)} = \frac{1}{1 + e^{-z}}$$

*4.2.2. Score for objectivity prediction*:

The sigmoid activation makes sure that the objectless score, which indicates the degree of confidence that an object is present in a grid cell, falls between 0 (no object) and 1 (high confidence).

### 4.3 Prediction for the Bounding Box:

Bounding boxes are predicted by YOLO v8 to be offsets from grid cells:

*4.3.1. Regarding every grid cell:*

Estimates the cell's center coordinates, x, y. Estimates the bounding box's width (w) and height (h) in relation to the image's dimensions. Constrains the predictions inside the boundaries of the grid cell using sigmoid activation for x and y.

*4.3.2. Multiple-Scale Forecasts*:

To find objects of different sizes, the detecting head uses feature maps at different scales: High-resolution Maps with high resolution for tiny items. Low-resolution Maps with low resolution for big things. During post-processing, YOLO v8 nano combines the bounding box and class predictions it makes at each scale.

### 4.4. Post-Processing Predictions

After the detection head produces raw predictions, post-processing is used to filter and modify the results:

- Non-Maximum Suppression (NMS): Automatically eliminates overlapping bounding boxes for objects based on confidence scores. Retains the box with utmost assurance.

- Confidence Thresholds: Does not include estimates with low confidence levels (e.g., < 0.5).
- Final outputs include bounding boxes, class labels, and confidence scores.

## 5. Validation Matrix

### 5.1 Training Losses:

*5.1.1Train/box_loss*

Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, Dongwei Ren [69] represent the given equation (CIOU Loss) which is used in YOLO v8 nano to enhance the accuracy of bounding box predictions.

The Formula:

$$\text{CIOU Loss} = 1 - \text{IOU} + \frac{\rho^2(b, b^g)}{c^2} + \alpha v$$

ρ2 (b,bg): Euclidean distance between the centes of the predicted and ground truth boxes.

c2: Diagonal length of the smallest enclosing box.

v: Aspect ratio consistency.

α: Balancing parameter

*5.1.2 Train/cls_loss:*

The given equation represents the Binary Cross-Entropy (BCE) loss, which is commonly used for classification tasks in YOLO v8 nano. In YOLO v8 nano, this loss is applied for class prediction, where the model determines the probability that a detected object belongs to a particular class.

The Formula:

$$cls_{loss} = -\sum_{i=1}^{c}[y_i \log(P_i) + \log(1 - y_i)\log(1 - P_i))$$

C: Number of classes.

$y_i$: Ground truth label (1 for the correct class, 0 otherwise).

$P_i$: Predicted probability for class i.

*5.1.3 Train/dfl_loss:*

Xiang Li, Wenhai Wang, Lijun Wu, Shuo Chen, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang [70] represent the given equation the Distribution Focal Loss (DFL) used in YOLO v8 for bounding box regression, specifically for fine-grained localization of bounding box coordinates. This loss helps improve the precision of bounding box predictions by making them more accurate at the subpixel level.

The Formula:

$$\mathrm{dfl_{loss}} = \sum_{\mathrm{I}} \mathrm{software(z_i).CE(Pred_i, Target_i)}$$

$z_i$: Logits for each bin.

CE: Cross-entropy loss for regression offsets.

## 5.2 Metrics

### *5.2.1 metrics/precision (B)*

Mark Goadrich [71] represents in his paper the Precision formula in YOLOv8 is to evaluate how reliably the model predicts objects that are actually present, avoiding incorrect detections.

The Formula:

$$\mathrm{Precision} = \frac{\mathrm{True\ Positive}}{\mathrm{True\ Positive + False\ positive}}$$

True Positives (TP): These are cases where the model correctly identifies the presence of an object.

False Positives (FP): These are cases where the model predicts an object when there is none or predicts the wrong class for a detected object.

### *5.2.2 metrics/recall (B)*

Mark Goadrich [70] represent in his paper the Recall equation in YOLOv8 is to evaluate the model's ability to detect all actual objects in an image.

The Formula:

$$\mathrm{Recall} = \frac{\mathrm{True\ Positives}}{\mathrm{True\ Positives + False\ negatives}}$$

False Negatives (FN): These are cases where the model fails to detect an object that is present in the image

### *5.2.3 metrics/mAP50 (B)*

Bochkovskiy, A., Yao Wang, C.-, & Yuan Mark Liao, H [34] represent in their paper the aim of the $\mathrm{MAP}_{50}$ equation is to provide a single, easy-to-interpret number that summarizes YOLO v8's detection performance across all classes at an IOU threshold of 0.5.

The Formula:

$$mAP_{50} = \frac{1}{N}\sum_{i=1}^{N} AP_i$$

$AP_i$: Area under the precision-recall curve for class ii.

N: Number of classes.

*5.2.4 metrics/mAP50-95(B)*

MAP averaged over IOU thresholds from 0.50 to 0.95 (in increments of 0.05).

The Formula:

$$mAP_{50-95} = \frac{1}{10}\sum_{j=1}^{10} mAP_{50+0.05j}$$

MAP (50+0.05J): The MAP at a specific IoU threshold.

**5.3 Learning Rates**

*5.3.1 lr/pg0, lr/pg1, lr/pg2*

Ilya Loshchilov and Frank Hutter [72] represent this equation in their paper. This equation is used to dynamically adjust the learning rate over time during training.

The Formula:

$$Ir_t = Ir_t + \frac{1}{2}(Ir_{max} + Ir_{min})(1 + cos(\frac{t}{T}\pi))$$

t: Current epoch.

T: Total training epochs.

$Ir_{min}$ , $Ir_{max}$: Minimum and maximum learning rates.

**6. Training**

Currently, there are two data sets available GTSRB, a large multi-category classification benchmark, and GTSDB a single image detection assessment for researchers. We used the GTSRB to train our YOLOv8 model. The dataset contains 43 classes with various images in each class. With a total of 51839 images, 39209 images for training and 12630 images for testing and validation [73].

We trained our model on 100 epochs, to repeat the training 100 times accumulating each result over the other, to ensure the highest possible accuracy results and precision. Increasing the number of epochs would ensure better results, but with our limited resources and low hardware power it would have taken a long time to complete our training.

## 7. Results

### 7.1 Confusion matrix:

A confusion matrix in YOLO v8 nano training is a visual representation of the model's performance in classifying objects during validation. It shows how well the model predicts each class by comparing its predictions to the ground truth labels [74].

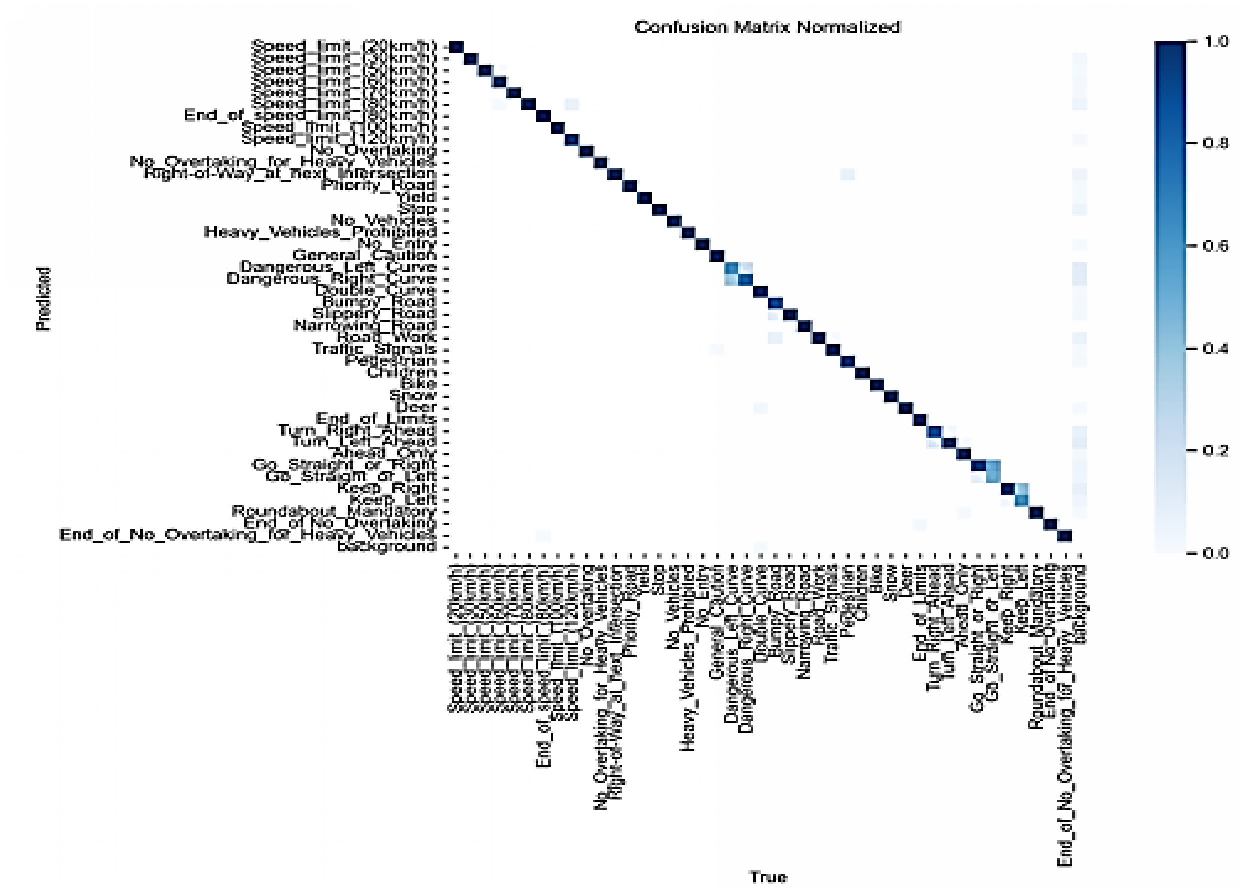


**Fig.14** Confusion Matrix

The matrix is typically structured with: Rows: Representing the actual (ground truth) classes. Columns: Representing the predicted classes. Each cell in the matrix contains a numerical value, which indicates the number of instances where a specific ground truth class (row) was predicted as another class (column). Diagonal cells (top-left to bottom-right): These represent correct predictions where the predicted class matches the ground truth. Higher values in these cells indicate better accuracy for that specific class. Off-diagonal cells: These indicate misclassifications where the model predicted the wrong class. For example, if an actual class left curve is predicted as right curve, this misclassification will appear in the corresponding off-diagonal cell. As shown in Figure 14.

Ideal Trends and Values: Slight inequality in diagonal values is natural, but they should all be close to 1 (high accuracy) for a good model. But our model shows multiple classes that are predicted as false positives. This I due to Class imbalance (some classes have fewer samples). This is caused by the over fitting in the dataset since some classes don't have enough images to be trained with, as shown in Table.2, and Figure15.

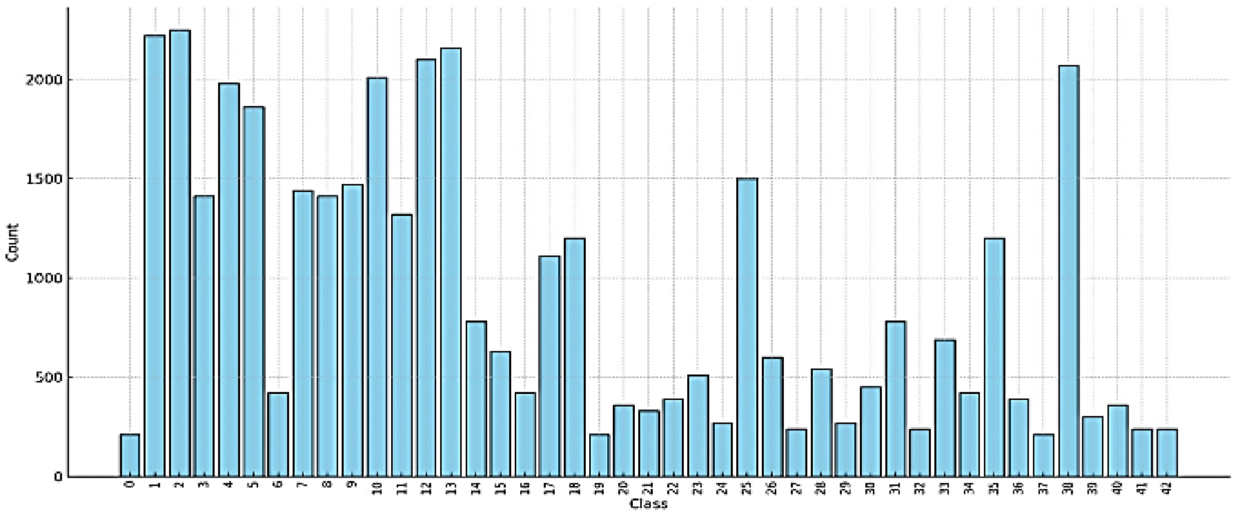


**Fig. 15.** Class counts distribution

**Table 2.** Class ID with Number of labels and Class Name

| **Class ID** | **Class Names** |
|---|---|
| Class 0: 210 labels | Speed limit (20Km/h) |
| Class 1: 2220 labels | Speed limit (30Km/h) |
| Class 2: 2250 labels | Speed limit (50Km/h) |
| Class 3: 1410 labels | Speed limit (60Km/h) |
| Class 4: 1980 labels | Speed limit (70Km/h) |
| Class 5: 1860 labels | Speed limit (80Km/h) |

| | |
|---|---|
| Class 6: 420 labels | End of Speed limit (80Km/h) |
| Class 7: 1440 labels | Speed limit (100Km/h) |
| Class 8: 1410 labels | Speed limit (120Km/h) |
| Class 9: 1470 labels, | No Overtaking |
| Class 10: 2010 labels | No Overtaking for Heavy Vehicles |
| Class 11: 1320 labels | Right-of-way at next intersection |
| Class 12: 2100 labels | Priority Road |
| Class 13: 2160 labels | Yield |
| Class 14: 780 labels | Stop |
| Class 15: 630 labels, | No Vehicles |
| Class 16: 420 labels, | Heavy Vehicles Prohibited |
| Class 17: 1110 labels | No Entry |
| Class 18: 1200 labels, | General Caution |
| Class 19: 210 labels, | Dangerous Left Curve |
| Class 20: 360 labels | Dangerous Right Curve |
| Class 21: 330 labels | Double Curve |
| Class 22: 390 labels, | Bumpy Road |
| Class 23: 510 labels, | Slippery Road |
| Class 24: 270 labels, | Narrowing Road |
| Class 25: 1500 labels | Road Work |
| Class 26: 600 labels, | Traffic Signals |
| Class 27: 240 labels, | Pedestrian |
| Class 28: 540 labels, | Children |
| Class 29: 270 labels, | Bike |
| Class 30: 450 labels, | Snow |
| Class 31: 780 labels, | Deer |
| Class 32: 240 labels | End of Limits |
| Class 33: 689 labels | Turn Right Ahead |
| Class 34: 420 labels | Turn Left Ahead |

| | |
|---|---|
| Class 35: 1200 labels | Ahead Only |
| Class 36: 390 labels | Go Straight or Right |
| Class 37: 210 labels | Go Straight or Left |
| Class 38: 2070 labels | Keep Right |
| Class 39: 300 labels | Keep Left |
| Class 40: 360 labels | Roundabout Mandatory |
| Class 41: 240 labels | End of No Overtaking |
| Class 42: 240 labels | End of No Overtaking for Heavy Vehicles |

## 7.2 Training results over box loss

The train box loss in YOLOv8 nano refers to the error associated with the predicted bounding box coordinates during the training process. It evaluates how accurately the model’s predicted bounding boxes align with the ground truth bounding boxes for the detected objects in the dataset. A lower box loss signifies better alignment between the predictions and the ground truth, indicating improved localization performance by the model. YOLO models represent bounding boxes using centre coordinates (x, y), width w, and height h. During training, the model learns to predict these parameters accurately for each detected object. The train box loss measures the deviation between these predicted values and the corresponding ground truth values. Lower box loss indicates better alignment between predictions and ground truth [74]. As shown in Figure 16.

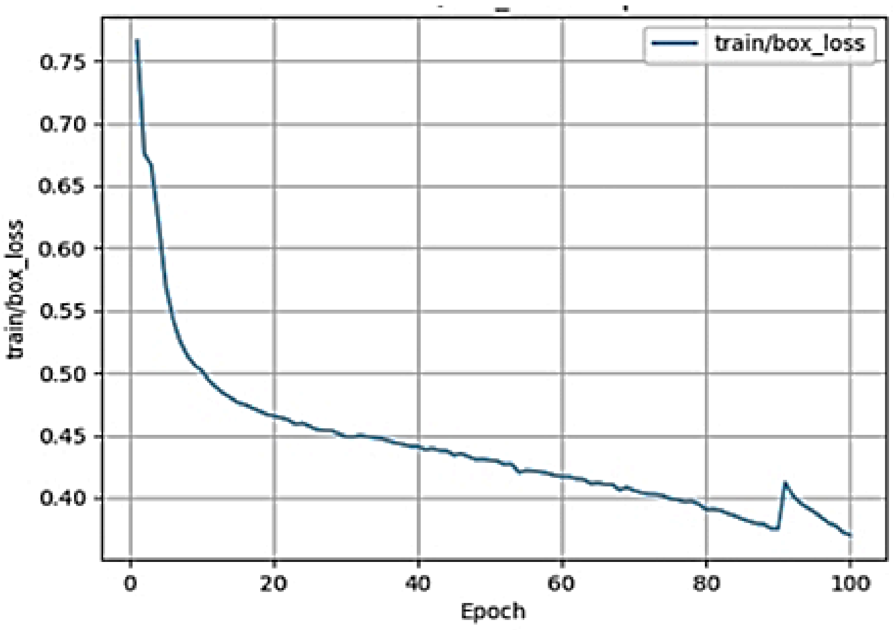


**Fig. 16.** Train/box_loss vs epoch.

At the beginning of training (first few epochs), the train box loss typically starts at higher values, usually around 1.0 to 2.0. This is because the model has not yet learned to align its predictions with the ground truth effectively. and gradually decreases to values 0.01 - 0.1 or lower as training progresses. Train/box_loss: Reduced from 0.76633 to 0.37 over 100 epochs. In a training scenario, the box loss reduced from 0.76633 to 0.37 over 100 epochs. This steady decline suggests the model is learning effectively. The final value of 0.37 indicates that the bounding box predictions have converged reasonably well, though further improvements might be possible with extended training or dataset refinement.

## 7.3 Training results over classification loss

Train classification loss (cls_loss) in YOLO v8 nano measures the error associated with predicting the correct class label for detected objects during training. It evaluates how accurately the model identifies the class of an object within the bounding box. Ensures that the model assigns high probabilities to the correct classes for detected objects and low probabilities to incorrect ones [74]. During training, YOLO models predict a probability distribution over all possible classes for each detected object. The classification loss ensures that High probabilities are assigned to the correct class labels. Low probabilities are assigned to incorrect class labels. As shown in Figure 17. At the beginning of training, the classification loss is usually higher, ranging between 0.5 and 1.5. This reflects the model's initial lack of understanding of class distributions. As the training progresses, the loss decreases steadily. For well-optimized training, the loss can drop to values around 0.01 to 0.1 or lower, indicating that the model has learned to classify objects accurately. The dataset is small, or a class imbalance exists, this might plateau higher. Train/cls_loss has dropped significantly from 2.82310 to 0.16491 over 100 epochs. The classification loss shows a large drop, suggesting the model is learning to classify the traffic signs better.

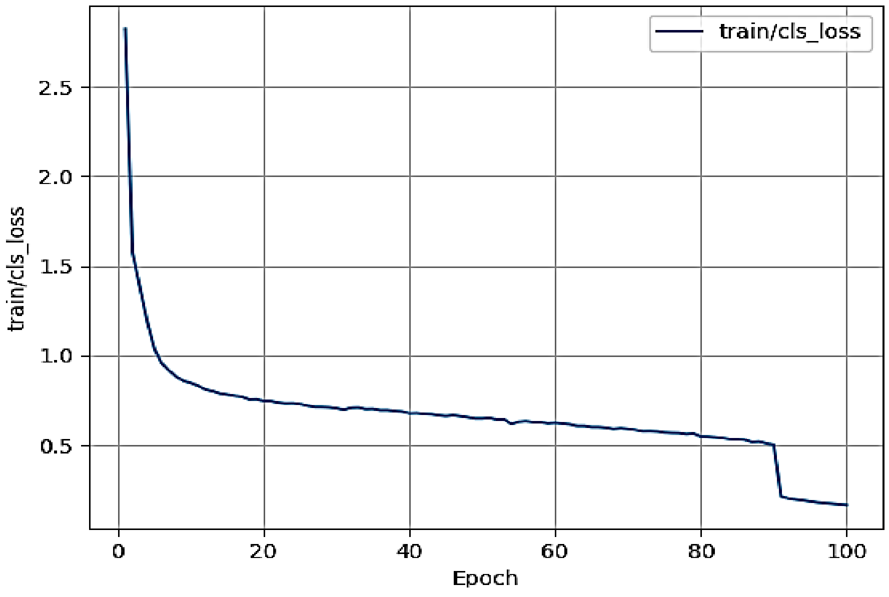


**Fig. 17.** Train/class_loss vs epoch

## 7.4 Training results over distribution focal loss

Train Distribution Focal Loss (DFL Loss) in YOLO v8 nano is related to bounding box regression and improves the precision of the predicted box locations by focusing on fine-grained localization details. This loss is part of the bounding box refinement process. Helps the model predict more accurate bounding box coordinates by learning a smooth probability distribution over the continuous positions of the box [74]. DFL helps the model learn a smooth probability distribution over the continuous positions of bounding box coordinates. Instead of just predicting a single point, it ensures that the predictions align closely with the true location of the object. The loss encourages the model to give more attention to areas where small errors in localization can lead to significant inaccuracies, thus improving overall performance. As show in Figure 18. At the beginning of training, DFL loss values typically range from 0.5 to 1.5, reflecting the model's initial imprecision in localizing objects. As training progresses, DFL loss decreases gradually, typically reaching values between 0.02 and 0.1 or lower. This reduction signifies that the model is improving its ability to predict bounding boxes with fine-grained accuracy. Train/dfl_loss has improved gradually from 1.05844 to 0.94399. A small but steady improvement in localization accuracy. But the model might not have learned to localize well yet, leading to higher DFL values. It might take more epochs for the model to refine its bounding box predictions effectively. And Overfitting to the training data can also result in a high DFL loss because the model may learn to localize objects in ways that are too specific to the training data, making it unable to generalize well to new examples.

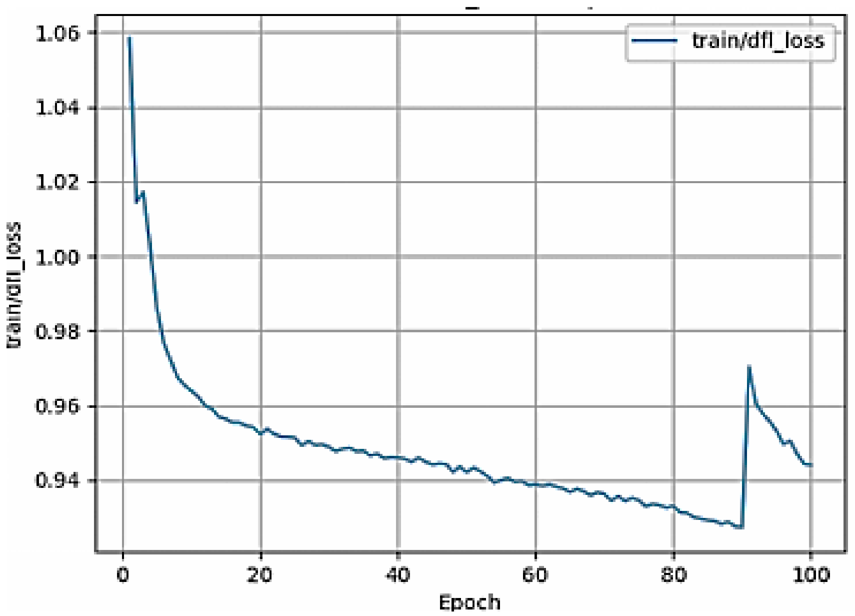


**Fig. 18.** Train/dfl_loss vs epoch.

## 7.5 Metrics results over mean Average Precision 50(B)

Metrics/MAP50 (B) in YOLOv8 nano represents the Mean Average Precision (MAP) at an Intersection over Union (IOU) threshold of 0.5 for bounding box predictions. It is a critical evaluation metric that measures the model's ability to accurately detect objects and place precise bounding boxes around them [74]. The MAP metric combines precision and recall providing a balanced evaluation of the model's object detection performance. A high MAP score indicates that the model is both precise (few false positives) and comprehensive (few false negatives). The IOU compares the overlap between the predicted bounding box and the ground truth bounding box. An IOU threshold of 0.5 means that the predicted bounding box must overlap with the ground truth box by at least 50% to be considered a correct detection. As shown in Figure 19. MAP at IOU=0.5 represents average precision for detecting objects. In the initial stages of training, MAP50 values typically range from 0.4 to 0.6. This reflects the model's early attempts to detect objects with moderate accuracy. As training progresses, the MAP50 score should increase steadily, aiming for a final value of 0.9 or higher, indicating a high level of accuracy in object detection. The MAP50 score improved from 0.44557 to 0.94942. Observation: MAP at IOU = 0.5 has reached a strong level of 94.9%, which shows that the model is accurate in detecting most signs.

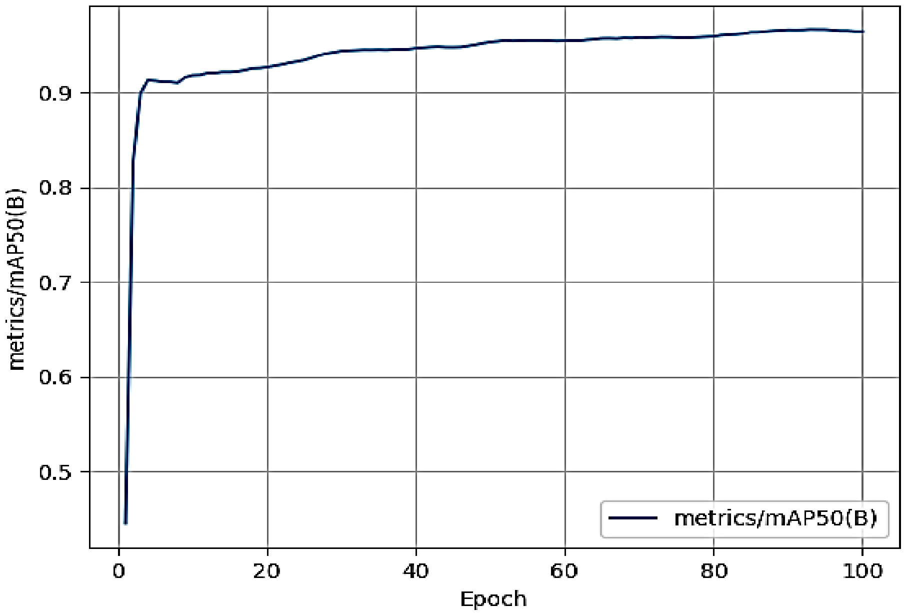


**Fig. 19.** Metrics /map50 (B) vs epoch.

## 7.6 Metrics results over precision (B)

Precision (B) in the Metrics/MAP50 evaluation measures the accuracy of the model's positive predictions, specifically evaluating how many of the bounding boxes predicted as objects are correctly matched to ground truth objects. High precision is critical in applications were reducing false positives (incorrect detections) is a top priority. [74]. High precision means that the model rarely predicts objects where none exist, ensuring fewer false alarms. As shown in Figure 20. Early in training, precision typically starts around 0.5, reflecting the model's initial struggles with distinguishing objects accurately. As training progresses, precision should increase steadily, aiming for values between 0.8 and 0.95. These high values indicate reliable and accurate detections with minimal false positives. High precision means fewer false positives. At 94.399% precision, the YOLOv8 nano model demonstrates a high level of accuracy in its positive predictions. This means that almost all detected bounding boxes are correct, with very few false positives. This level of precision makes the model highly suitable for real-world applications where reliable detections are critical.

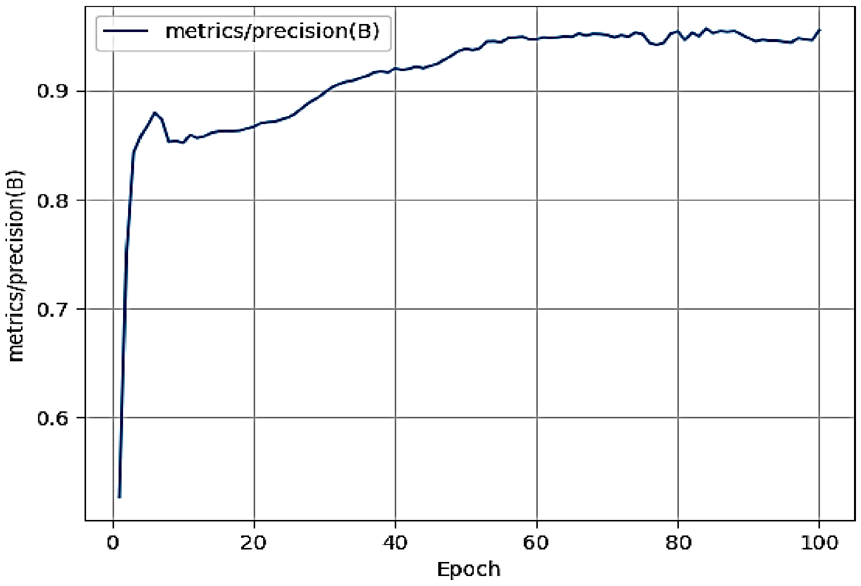


**Fig. 20.** Metrics/ precision(B) vs epoch.

### 7.7. Metrics results over recall (B)

Metrics/recall (B) in YOLO v8 nano measures the model's ability to detect all relevant objects in terms of bounding box predictions. It indicates how many of the ground truth objects the model successfully identifies. High Recall: Indicates the model detects most of the objects present in the image, with few missed detections. Low Recall: Suggests the model misses many objects, leading to a higher number of false negatives [74]. As shown in Figure 21. Recall typically starts around 0.5 in the early stages of training, reflecting the model's initial difficulty in detecting all objects. As the model learns, recall should increase, aiming for values between 0.8 and 0.95. This range indicates that the model reliably detects most objects in the dataset. High recall means fewer missed objects. In a specific scenario, recall improved to 0.9554 (95.54%), indicating the model successfully detected almost all ground truth objects, with very few missed detections. A recall value of 95.54% shows that the model identifies nearly all objects in the images, making it highly effective for tasks where completeness is vital.

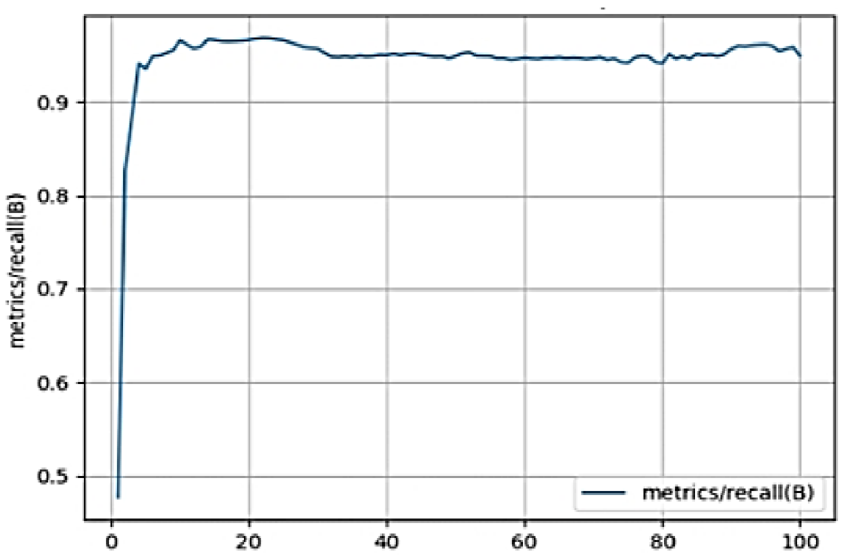


**Fig. 21.** Metrics/recall (B) vs epoch

### 7.8. Metrics results over MAP 50-95(B)

Metrics/MAP50-95 (B) in YOLOv8 nano calculates the Mean Average Precision (MAP) over a range of IOU (Intersection over Union) thresholds, from 0.5 to 0.95. This metric provides a more comprehensive measure of the model’s detection performance by evaluating both detection accuracy and localization precision across various levels of overlap between predicted and ground truth bounding boxes. [74], as shown in Figure 22.

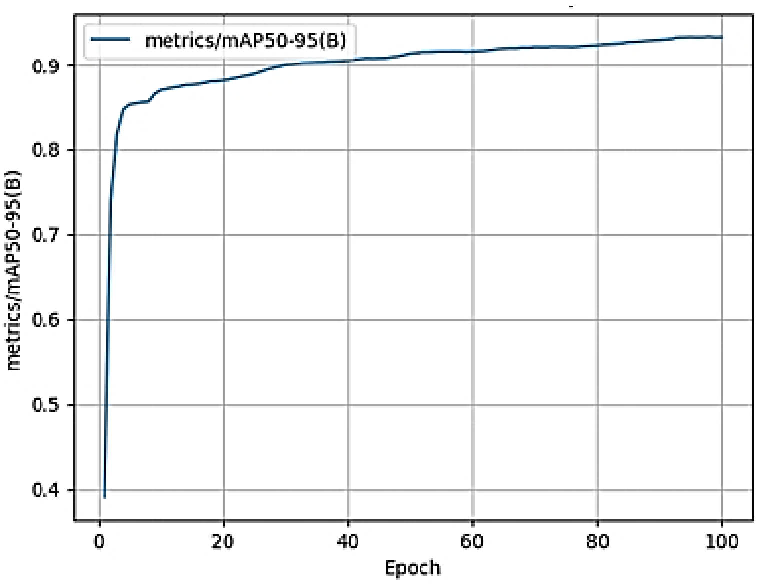


**Fig. 22.** Metrics/mAP50-95(B) vs epoch

MAP50-95 expands on MAP50 by averaging precision values across multiple IOU thresholds, ranging from 0.5 to 0.95, thus requiring both accurate detection and precise localization to achieve a good score. Initially, the mAP50-95 value should be around 0.2 to 0.4 as the model is still learning to accurately localize objects across different thresholds. Over the course of training, mAP50-95 should increase and ideally aim for values between 0.5 and 0.7 or higher, indicating good overall model performance with both accurate detection and precise localization. A high mAP50-95 indicates strong model generalization. The improvement of mAP50-95 from 0.39080 to 0.93294 in our model reflects the model's significant progress in both detecting and accurately localizing objects, even at higher IoU thresholds. A final score of 93.3% indicates that the YOLOv8 nano model is capable of handling more stringent object detection tasks, ensuring reliable and precise performance.

**7.9. Validation results over box loss**

Val/box_loss represents the bounding box regression loss computed during the validation phase of training. It evaluates the alignment between the predicted bounding boxes and the ground truth boxes in the validation dataset. This metric indicates how well the model generalizes to unseen data, as opposed to the training set [68]. YOLO models represent bounding boxes using four parameters: the centre coordinates (x, y), width (w), and height (h). The Val/box_loss measures the error in predicting these parameters for objects in the validation data. A lower Val/box_loss indicates better generalization, meaning the model can accurately predict bounding boxes for unseen data. This is crucial for deploying the model in real-world scenarios where the data distribution might differ from the training dataset. As shown in Figure 23.

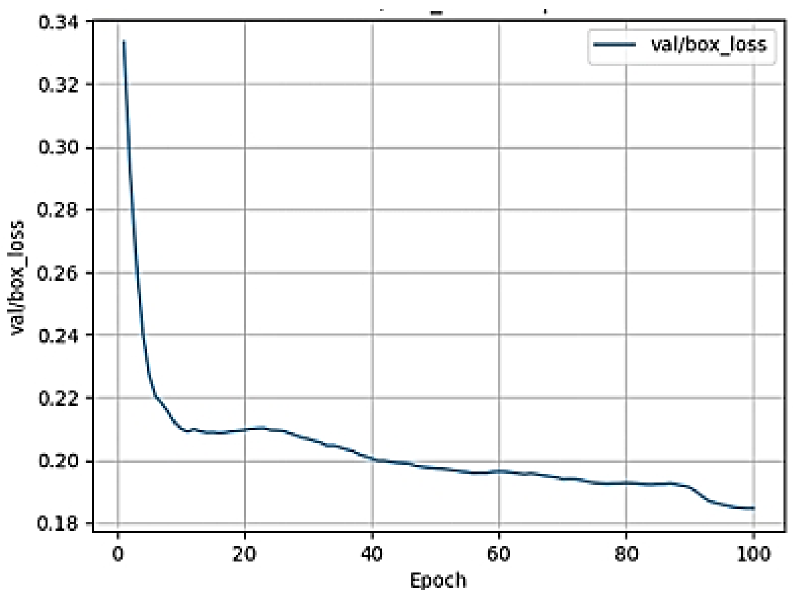


**Fig. 23.** Val/box_loss vs epoch.

At the start of training, Val/box_loss should be close to the train/box_loss values. Early on, the model is still learning to predict bounding boxes and may have relatively high loss values. At convergence, Val/box_loss should fall within a range of 0.02 to 0.1, indicating that the model is accurately localizing objects in the validation dataset. In this specific scenario, Val/box_loss decreased from 0.37 to 0.18. This significant reduction reflects improved generalization, as the model learns to predict bounding boxes consistently across both training and validation datasets.

## 7.10. Validation results over classification loss

Val/cls_loss represents the classification loss computed during the validation phase of training. It evaluates how accurately the model predicts the class labels for objects within the bounding boxes in the validation dataset. This metric helps determine the model's generalization ability in classifying objects not seen during training [74]. Classification loss measures the discrepancy between the predicted class probabilities and the ground truth labels for detected objects. It ensures the model assigns high probabilities to the correct classes while minimizing probabilities for incorrect ones. A lower Val/cls_loss indicates that the model performs well at classifying objects in unseen data. However, validation loss is often slightly higher than training loss due to the model encountering more diverse or challenging examples in the validation dataset. as shown in Figure 24.

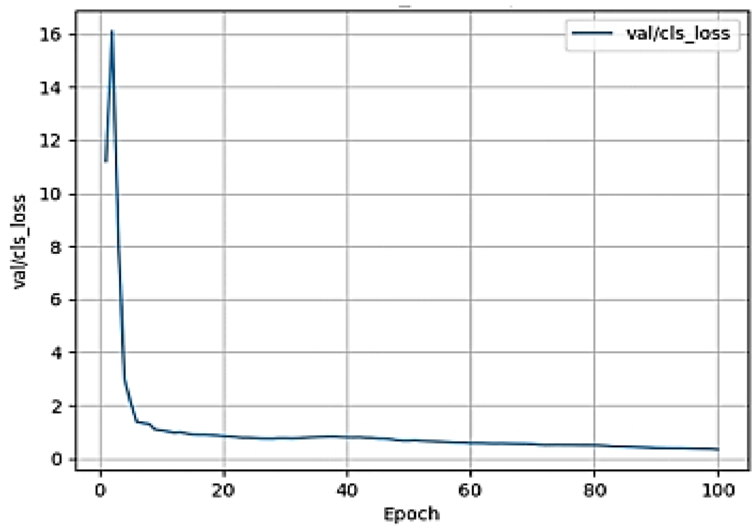


**Fig. 24.** Val/cls_loss vs epoch.

Val/cls_loss typically starts at values similar to train/cls_loss, reflecting the initial state where the model has minimal understanding of object classes. Both train/cls_loss and Val/cls_loss should steadily decrease as the model learns to classify objects more accurately. However, Val/cls_loss might decrease more slowly and stabilize at slightly higher values than train/cls_loss. Val/cls_loss should ideally fall within the range of 0.02 to 0.1. However, slightly higher values can be expected, especially with real-world datasets that include noisy or diverse examples in a specific scenario, Val/cls_loss reached 0.34432, which is slightly higher than the train/cls_loss value of 0.16. This difference is typical for real-world datasets where validation data presents more variability, which is typical for real-world datasets.

### 7.11. Validation results over distribution focal loss

Val/dfl_loss represents the Distribution Focal Loss (DFL) calculated during the validation phase of training. This loss evaluates how accurately the model predicts bounding box coordinates by refining the distribution over possible positions for each bounding box. Unlike typical regression methods, DFL considers the uncertainty around bounding box positions and helps the model focus on precise localization. [74]. DFL measures the model's ability to predict a smooth probability distribution over the potential positions of bounding box coordinates. This is particularly important for achieving fine-grained localization accuracy. Val/dfl_loss assesses the model's performance on unseen data, providing insights into how well the model generalizes its bounding box refinement capabilities beyond the training dataset. As shown in Figure 25. Val/dfl_loss begins at values similar to train/dfl_loss. Early in training, both metrics may be relatively high as the model learns to localize objects. As training progresses, both train/dfl_loss and Val/dfl_loss should gradually decrease, indicating improved bounding box refinement. Val/dfl_loss might remain slightly higher than train/dfl_loss due to the diversity and difficulty of examples in the validation dataset. At convergence, Val/dfl_loss typically falls within the range of 0.02 to 0.1, reflecting precise bounding box localization on validation data. In this scenario, Val/dfl_loss reached 0.54206, which is significantly better than the train/dfl_loss value of 0.94. This suggests strong generalization and better performance on validation data.which is a good sign.

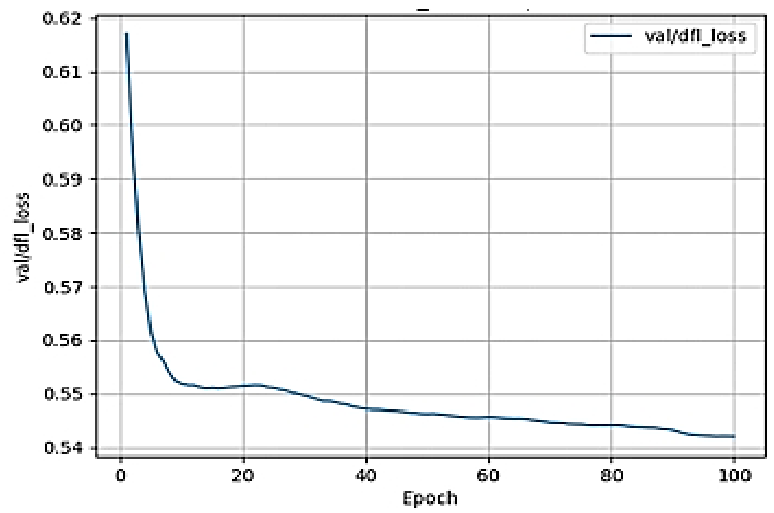


**Fig. 25.** Val/dfl_loss vs epoch.

## 8. Conclusion

To guarantee accurate traffic sign authentication, this work employs YOLO v8 nano to improve recognition and grouping of all categories present in the GTSRB dataset, including forbidden, required, risk, and others. The data set is containing 43 classes with various images in each class with a total of 51, 839 images. 39,209 images for training and 12,630 images for testing and validation. We did an electronic test that shows an almost acceptable result with low confident person because of Class imbalance, Boundary box misplaced, Noisy and bright images. The result provides the way for reliable, effective, and competitively cost standalone sign reading solutions by combining cutting-edge deep learning algorithms with simple architectures. This will ultimately lead to more secure and trustworthy automated public transportation.

## References

[1] Z. Huang, et al."An Efficient Method for Traffic Sign Recognition Based on Extreme Learning Machine," in IEEE Trans actions on Cybernetics, vol. 47, no. 4, pp. 920-933, April 2017.

[2] J. F. Khan, et al., "Image Segmentation and Shape Analysis for Road-Sign Detection," in IEEE Transactions on Intelligent Transportation Systems, vol. 12, no. 1, pp. 83-96, March 2011.

[3] C. Liu, et al. "Rapid Multiclass Traffic Sign Dtection in High-Resolution Images," in IEEE Transactions on Intelligent Transportation Systems, vol. 15, no. 6, pp. 2394-2403, Dec. 2014.

[4] F. Larsson, et al. Correlating Fourier descriptors of local patches for road sign recognition,'' IET Compute Vis, vol. 5, no.4, pp. 244–254, Jul. 2011.

[5] F. Larsson and M. Felsberg, ''Using Fourier descriptors and spatial mod els for traffic sign recognition,'' in Proc. 17th Scandin. Conf. Image Anal.,2011, pp. 238–249

[6] G. Wang, et al. ''A robust, coarse-tofine traffic sign detection method,'' in Proc. Int. Joint Conf. Neural Netw. (IJCNN), Aug. 2013, pp. 1–5.

[7] T. Chen and S. Lu, ''Accurate and efficient traffic sign detection using discriminative AdaBoost and support vector regression,'' IEEE Trans. Veh. Technol., vol. 65, no. 6, pp. 4006–4015, Jun. 2016.

[8] J. Redmon and A. Farhadi, ''YOLOv3: An incremental improvement,'' 2018.

[9] W. Liu, et al. ''SSD: Single shot MultiBox detector,'' in Proc. Eur. Conf. Comput. Vis., 2016..

[10] Z. Cai and N. Vasconcelos, ''Cascade R-CNN: Delving into high quality object detection,'' in Proc. IEEE/CVF Conf. Compute. Vis. Pattern Recognize., Jun. 2018.

[11] S. Ren, et al. ''Faster R-CNN: Towards real-time object detection with region proposal networks,'' IEEETrans. PatternAnal. Mach. Intell., vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[12] R. Girshick, ''Fast R-CNN,'' in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), 2015.

[13] Á. Arcos-García, et al. "Evaluation of deep neural networks for traffic sign detection systems," Neurocomputing, vol. 316, pp. 332–344, Nov. 2018.

[14] Khan, et al. (2023). A Lightweight Convolutional Neural Network (CNN) Architecture for Traffic Sign Recognition in Urban Road Networks. *Electronics*, *12*(8), 1802.

[15] S.Houben, et al. ''Detection of traffic signs in real-world images: The German traffic sign detection benchmark,'' in Proc. Int. Joint Conf. NeuralNetw. (IJCNN), Aug. 2013.

[16] Q.Tang, et al. ''Integrated Feature Pyramid Network With Feature Aggregation for Traffic Sign Detection,'' in IEEE Access, vol. 9, pp. 117784-117794, 2021.

[17] .Redmon, et al.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788 (2016).

[18] Gamez Serna, C., & Ruichek, Y. (2018)." Classification of Traffic Signs: The European Dataset." IEEE(2019).

[19] "Improved Traffic Sign Detection and Recognition Algorithm for Intelligent Vehicles. Sensors", 19(18), 4021

[20] Wang, G., et al. (2013)." Hole-based traffic sign detection method for traffic signs with red rim". The Visual Computer, 30(5), 539–551

[21] Zhang, J., et al. (2020). "A Cascaded R-CNN With Multiscale Attention and Imbalanced Samples for Traffic Sign Detection". IEEE Access, 8, 29742–29754).

[22] Ren, et al. (2016). "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks ".

[23] "Highly sensitive Deep Learning Model for Road Traffic Sign Identification." (2022). Mathematical Statistician and Engineering Applications, 3194–3205.

[24] Jadoun, P. (2023).” Traffic Sign Detection and Recognition System for Autonomous Vehicle”. International Journal for Research in Applied Science and Engineering Technology, 11(11), 1013–1017
[25] Chauhan, et al. (2018). “Convolutional Neural Network (CNN) for Image Detection and Recognition.” In 2018First International Conference on Secure Cyber Computing and Communication (ICSCCC) (pp. 278–282). IEEE.
[26] Redmon, J., & Farhadi, A. (2016). "YOLO9000: Better, Faster, Stronger". University of Washington and Allen Institute for AI. Retrieved
[27] Girshick, R. & Microsoft Research. (2015). “Fast R-CNN.” In arXiv [Report].
[28] Zhang, J., et al. (2017). “A real-time Chinese traffic sign detection algorithm based on modified YOLOv2.” Algorithms, 10(4), 127.
[29] Zhang, H., et al. (2020). "Real-Time Detection Method for Small Traffic Signs Based on YOLOv3". IEEE Access, 8,
[30] Nguyen, et al. (2018). "Towards Real-Time Smile Detection Based on Faster Region Convolutional Neural Network". Towards Real-Time Smile Detection Based on Faster Region Convolutional Neural Network, 1–6.
[31] Dursun, et al. (2020). “Artificial Intelligence Applications in Autonomous Vehicles: Training Algorithm for Traffic Signs Recognition.” IOP Conference Series: Materials Science and Engineering, 898(1), 012035.
[32] Redmon, J., & Farhadi, A. (2018). "YOLOv3: An Incremental Improvement". arXiv preprint arXiv:1804.02767
[33] Singh, et al. & Universal College of Engineering. (2021). "Traffic Sign Detection using YOLOv4. In International Journal of Creative Research Thoughts" (Vol. 9, Issue 5, pp. 891–892)
[34] Bochkovskiy, et al. (2020). "YOLOv4: Optimal Speed and Accuracy of Object Detection". In arXiv(Technical Report 2004.10934v1).
[35] Youssouf, N. (2022). “Traffic sign classification using CNN and detection using faster-RCNN and YOLOV4.” Heliyon, 8(12), e11792.
[36] Zhu, et al. (2022). “Traffic sign recognition based on deep learning. Multimedia Tools and Applications.”
[37] Gore, et al. 2023. Traffic Sign Detection using Yolo v5. International Journal for Research in Applied Science and Engineering Technology, 11(5), pp.2679–2683.
[38] Han, et al. (2023). “An Improved YOLO Model for Traffic Signs Small Target Image Detection. Applied Sciences,” 13(15), 8754–8754.
[39] Wang, et al. (2022). “Improved YOLOv5 network for real-time multi-scale traffic sign detection. Neural Computing and Applications,” 35(10), 7853-7865
[40] Qu, et al.2023). “Improved YOLOv5-based for small traffic sign detection under complex weather. Scientific Reports”, 13(1), 16219.
[41] Wang, et al. (2023). “An Improved Traffic Sign Detection and Recognition Deep Model Based on YOLOv5.” IEEE Access, 11, 54679–54691
[42] Li, et al. & Meituan Inc. (2022). “YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications.” In arXiv (Technical Report 2209.02976v1; p. 1).
[43] Terven, J., & Cordova-Esparza, D. (2023).” A Comprehensive Review of YOLO: From YOLOv1 to YOLOv8 and Beyond.” ArXiv (Cornell University).
[44] Ding, et al. (2021). “RepVGG: Making VGG-style ConvNets great again”. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 13728-13737.
[45] Zhang, H., et al. (2021). “VarifocalNet: An IoU-aware Dense Object Detector. Computer Vision and Pattern Recognition”.
[46] R. Mahadshetti, J. Kim, and T.-W. Um, "Sign-YOLO: Traffic Sign Detection Using Attention-Based YOLOv7," IEEE Access, vol12,pp.132689132700,Sep.2024.
[47] C.Y. Wang, et al. ‘‘YOLOv7: Trainable bag-of freebies sets new state-of-the-art for real-time object detectors.’’ In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2023.
[48] X. Ning, et al. ‘‘Feature Refinement and Filter Network for Person Re-Identification,’’ in IEEE Transactions on Circuits and Systems for Video Technology, vol. 31, no. 9, pp. 3391-3402, Sept. 2021.
[49] X. Ning, et al. ‘‘Real-Time 3D Face Alignment Using an Encoder-Decoder Network With an Efficient Deconvolution Layer,’’ in IEEE Signal Processing Letters, vol. 27, pp. 1944-1948, 2020.
[50] W. Ouyang, et al. ‘‘Deepidnet: Deformable deep convolutional neural networks for object detection.’’ In Proceedings of the IEEE conference on computer vision and pattern recognition, 2015.
[51] F. Shao, et al. ‘‘Real Time Traffic Sign Detection and Recognition Method Based on Simplified Gabor Wavelets and CNN’’ Sensors 18.
[52] F. Shao, et al. ‘‘Improved faster R-CNN traffic sign detection based on a second region of interest and highly possible regions proposal network.’’ Sensors,2019.
[53] J. Deng, et al. "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami
[54] C. Shorten, T.M. Khoshgoftaar, ‘‘A survey on Image Data Augmentation for Deep Learning.’’ J Big Data 6, 60 (2019).
[55] L. Taylor, and G. S. Nitschke. “Improving Deep Learning with Generic Data Augmentation.” IEEE Symposium Series on Computational Intelligence (SSCI) (2018).
[56] H. Zhang and Q. M. J. Wu, "Pattern recognition by affine Leg endre moment invariants," 2011 18th IEEE International Conference on Image Processing, Brussels, Belgium, 2011.
[57] A.Youssef, et al. ‘‘Fast traffic sign recognition using color segmentation and deep convolutional networks,’’ in: Proceedings of International Conference on Advanced Concepts for Intelligent Vision Systems, Springer, 2016.
[58] Z. Zhu, et al. ‘Traffic sign detection and classification in the wild,’’ in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[59] R. Timofte, et al. ‘‘Multi-view traffic sign detection, recognition, and 3D localisation,’’ Mach. Vis. Appl. 25 (3) (2011)

[60] Mogelmose, et al. ‘‘Vision-based traffic sign detection and analysis for intelligent driver assistance systems: per spectives and survey,’’ IEEE Trans. Intell. Transp. Syst. 13 (4) (2012)

[61] F.Jurišic´, et al. ‘‘Multiple-dataset traffic sign classification with on ecnn,’’in:Proceedingsof20153rdIAPRAsian Conference on Pattern Recognition, ACPR, IEEE, 2015.

[62] B. Ji, et al. "Improved YOLOv8 for Small Traffic Sign Detection Under Complex Environmental Conditions," Franklin Open, vol. 8, no. 100167, pp. 1–9, 2024.

[63] R. Kumar, et al. "Traffic Sign Detection Using YOLOv8," TTIC, vol. 7, pp. 60-66, 2023

[64] Shevtekar, S., & Kulkarni, S. (2024). Traffic-sign Recognition and Detection using YOLOv8.

[65] Singh, et al. (2024). Traffic Sign Recognition using YOLOv8 Algorithm extended with CNN. Presented at Sharda University.

[66] Huang, et al. (2023). Research on Traffic Sign Detection Based on Improved YOLOv8

[67] Venkata Rami Reddy Ch A Review on YOLOv8 and Its Advancements(2024)

[68] Du, S. et al., 2024. TSD-YOLO: Small traffic sign detection based on improved YOLO v8. IET Image Processing, 18(11), pp.2884–2898.

[69] Zheng, Z. et al., 2020. Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression. Proceedings of the AAAI Conference on Artificial Intelligence, 34(07), pp.12993–13000.

[70] Li, X. et al., 2020. Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection.

[71] Davis, J. & Goadrich, M., 2006. The relationship between Precision-Recall and ROC curves. Proceedings of the 23rd international conference on Machine learning - ICML '06, pp.233–240.

[72] Loshchilov, I. & Hutter, F., 2016. SGDR: Stochastic Gradient Descent with Warm Restarts.

[73] German Traffic Sign Benchmarks. (n.d.-b). https://benchmark.ini.rub.de/index.html

[74] Ultralytics. "FocalLoss." Ultralytics Documentation. Accessed December14,2024. https://docs.ultralytics.com/reference/utils/loss/#ultralytics.utils.loss.FocalLoss