Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2023.042675

ARTICLE

Traffic Sign Recognition for Autonomous Vehicle Using Optimized YOLOv7 and Convolutional Block Attention Module

P. Kuppusamy[1,*], M. Sanjay[1], P. V. Deepashree[1] and C. Iwendi[2]

[1]School of Computer Science and Engineering, VIT-AP University, Andhra Pradesh, 522237, India

[2]School of Creative Technology, University of Bolton, Manchester, BL3 5AB, UK

*Corresponding Author: P. Kuppusamy. Email: drpkscse@gmail.com

Received: 07 June 2023 Accepted: 16 August 2023 Published: 31 October 2023

ABSTRACT

The infrastructure and construction of roads are crucial for the economic and social development of a region, but traffic-related challenges like accidents and congestion persist. Artificial Intelligence (AI) and Machine Learning (ML) have been used in road infrastructure and construction, particularly with the Internet of Things (IoT) devices. Object detection in Computer Vision also plays a key role in improving road infrastructure and addressing traffic-related problems. This study aims to use You Only Look Once version 7 (YOLOv7), Convolutional Block Attention Module (CBAM), the most optimized object-detection algorithm, to detect and identify traffic signs, and analyze effective combinations of adaptive optimizers like Adaptive Moment estimation (Adam), Root Mean Squared Propagation (RMSprop) and Stochastic Gradient Descent (SGD) with the YOLOv7. Using a portion of German traffic signs for training, the study investigates the feasibility of adopting smaller datasets while maintaining high accuracy. The model proposed in this study not only improves traffic safety by detecting traffic signs but also has the potential to contribute to the rapid development of autonomous vehicle systems. The study results showed an impressive accuracy of 99.7% when using a batch size of 8 and the Adam optimizer. This high level of accuracy demonstrates the effectiveness of the proposed model for the image classification task of traffic sign recognition.

KEYWORDS

Object detection; traffic sign detection; YOLOv7; convolutional block attention module; road sign detection; Adam

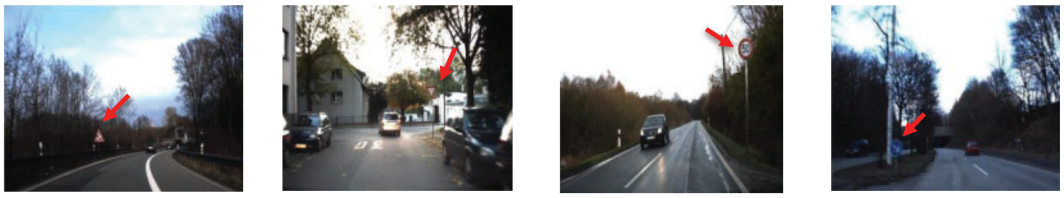
1 Introduction

Infrastructure and construction of roads in any geographical area play a pivotal role in the economic and social development of the region, as it connects people to business and allows the movement of locomotives and services. One of the present-day primary challenges relating to road infrastructure is accidents, and other traffic-related concerns like traffic congestion, restricted infrastructure capacity, low maintenance of roads, etc. [1,2]. Classically, human conception and past experiences have guided the progress of road infrastructure. However, as technology has become ubiquitous, and owing to advancements in automobile-related technologies such as self-parking systems, self-driving cars, fully autonomous systems, etc., all of which are essentially categorized under the umbrella of Autonomous Driving Systems (ADS). There has been a significant increase in the usage of AI, and its sub-domains in accomplishing some cardinal tasks in ADS. An evaluative study on Deep Neural Networks (DNN) for Traffic Sign Detection (TSD), throws some light on how the detection of traffic signs is an indispensable study because these detection systems encompass anchor components required for safety and support in ADS [3]. The IoT devices are utilized to gather the data from the environment, and ML analyses the data to solve the challenges in traffic management systems. The traffic management system contains three layers such as data acquisition, network transmission, and application. The data acquisition is done via sensors, cameras, video monitoring, and online monitoring. The collected data is transmitted over the network using Bluetooth, Wi-Fi (Wireless Fidelity), mobile network, etc. Finally, AI and ML play a major role in the analysis, visualize the analyzed outputs, and derive the systems based on the outputs like ADS [4]. The study using ML showcased the latest and most advanced techniques for monitoring construction progress, including methods for collecting data, retrieving information, estimating progress, and presenting the results visually. Along similar lines, AI and ML are used in many more traffic-related issues [5]. A review of traffic congestion prediction using AI described the probabilistic reasoning models like fuzzy logic, Hidden Markov Model (HMM), the Bayesian network, Support Vector Machine (SVM), Artificial Neural Networks (ANN), Decision Trees, etc., Deep Learning (DL) algorithms like Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM), are used for short-term traffic congestion prediction [6]. AI and ML-based incident detectors in Road Transport Systems (RTS) discussed dire problems and plausible solutions for reducing traffic accidents that enhanced the automatic incident system detectors [7].

The technology development in computer vision plays a key role with goals revolving around improving road infrastructure like road accidents, traffic congestion, etc. Object detection is a sub-field of computer vision that uses various DL architectures for recognizing and classifying objects. A comparative analysis of CNN-based object detection algorithms shows YOLOv3 is the fastest, and performs best overall outperforming Single-Shot Detector (SSD), and Faster Region-based CNN (R-CNN) [8]. However, it is also highlighted that the choice of the algorithm may be dependent upon the specific situation or problem that needs to be solved. For instance, R-CNN works best for small datasets that do not require real-time video outputs, whereas YOLO works best for object detection in the live environment. YOLOv4 runs twice as fast as EfficientNet, with an Average Precision (AP) of 14% more than YOLOv3. The YOLOv7 algorithm surpasses all the well-known real-time object-detection algorithms concerning AP at 56.8%, and speed with a maximum range of 160 FPS [9]. So far, research on the detection of traffic signs has been done using several versions of YOLO, and other object-detection algorithms.

This study aims to use the fairly latest version of the most optimized object-detection algorithm YOLOv7 to detect and identify traffic signs. This study also tries to dive deep into analyzing effective combinations of adaptive optimizers like Adam and SGD along with YOLOv7. SGD has solid theoretical and mathematical support, along with an exhibition of enhanced stability and generality [10]. In most applications, the Adam optimizer is recommended as the default optimization method because it usually generates better results, is faster to compute, and requires fewer tuning parameters than conventional optimization methods [11]. Batch sizes 8 and 16 are used for the task of TSD. A portion of the German traffic signs is used for the training purpose. This study also explores the feasibility of adopting smaller datasets while keeping high accuracy to modify the application domain. Fig. 1 shows the various traffic sign classes that are pointed by red arrows for human reference.

Traffic sign recognition is a primary factor for autonomous cars to make safe travel. However, traffic sign recognition system contains more challenges due to limitations that are shown by recent incidents involving autonomous vehicles [12]. Conventional traffic sign recognition encounters numerous challenges, such as occlusion, lighting conditions, and the existence of several neighboring traffic signs [13].



(a) Dangerous class (b) Other class (c) Prohibitory class (d) Mandatory class

Figure 1: Input images of each class for prediction

1.1 Motivation

The motivation for this research work is as follows:

- Traffic sign recognition is essential for autonomous cars to navigate safely and efficiently. But there are severe worries regarding the limitations of traffic sign recognition systems and the methods they employ, as shown by recent incidents involving autonomous vehicles and research connected to recognition system failure. Therefore, it becomes even more crucial to create powerful algorithms that can get beyond these constraints and provide precise and trustworthy traffic sign detection to improve the performance and safety of autonomous cars. Effective traffic sign identification is essential for maximizing traffic flow and raising overall road safety in addition to lowering the likelihood of accidents.
- Traditional techniques of traffic sign recognition, on the other hand, encounter various problems, such as occlusion, fluctuating lighting conditions, complicated backdrops, and the presence of multiple signs nearby. Due to these challenges, improved approaches must be proposed to manage these situations and provide precise and dependable traffic sign detection and identification.

1.2 Contributions

The contributions of this research work are as follows:

- This research intends to improve the accuracy and speed of traffic sign detection by incorporating the CBAM into the YOLOv7 framework. The CBAM's potent attention mechanism enables the model to effectively acquire, and highlight key spatial and channel-wise information, enabling reliable detection of traffic signs even under difficult conditions like occlusion or complicated backdrops.
- Investigate and compare the effectiveness of the proposed model with different optimizers and batch sizes.
- The enhanced model proposed in this study exhibits improved feature representation, higher detection accuracy, and resilience by combining the characteristics of YOLOv7 and three CBAM modules in a complementary way, advancing the development of autonomous driving technology.
- The significance of this study is to achieve high accuracy on a small-sized real-time dataset thereby applying the model proposed in this study to a larger, and more diverse dataset for real-time applications in autonomous vehicles.

The remainder of this study is organized as follows: Section 2 gives an overview of existing literature on traffic sign recognition for autonomous vehicles, highlighting strengths, limitations, and extensions of current knowledge. Section 3 focuses on the description of the YOLOv7 with the CBAM framework, including its working principle, architecture, and loss function. The dataset used for training and evaluation is described in Section 4. Section 5 describes the evaluation metrics, hyperparameters, and hardware/software configurations used in the experiments. Section 6 presents a detailed analysis of the results, including performance comparisons and visual representations of the model's capabilities. Finally, the conclusion and future scope are discussed in Section 7.

2 Literature Review

There has been a positive trend toward applications in computer vision resulting in a substantial amount of research on TSD using various object-detection algorithms. Relevant to this study, an in-depth inspection and analysis of various machine vision-based traffic detection models divided into 5 categories viz. color, shape, color and shape, ML, and Light Detection and Ranging (LiDAR) based models [14]. A TSD system based on novel DL architectures used the YOLOv3, and Xception models along with Adam and RMSprop optimizers. These models are designed using the dataset with 3 classes such as "Yellow, diamond-shaped pedestrian crossing sign", "Yellow, diamond-shaped other traffic signs" and "others". However, this study is processed with a lower frame rate of 4.5 fps, which could be increased to improve the processing time and performance accuracy [15].

A study specifically focused on the detection of Indian traffic signs using YOLOv3, and CNN over 5 classes, and attained an accuracy of 87%. However, the authors have not used a real-time traffic detection system to predict each frame in a video [16]. A study proposed a cascaded R-CNN to obtain the multi-scale features for TSD that resulted in an accuracy of 99.7%. Additionally, the study also proposed a multi-scale attention mechanism to improve the detection of true traffic signs and reduce false detections [17]. The YOLOv5 model is implemented on 8 classes of datasets viz. "No U-turn", "Road bump", "Road works", "Watch for children crossing", "Crosswalk ahead", "Give way", "Stop", and "No entry", along with a thorough comparison between the YOLOv5 and SSD. The own dataset used for the model displayed an accuracy of 97.70%. The future scope of the study is to expand the existing dataset and apply newly developed models like Mask R-CNN, CapsNet, and Siamese Neural Networks [18]. An improved YOLOv5 model is implemented for real-time multi-scale TSD over a massive size of 182 classes. Data augmentation, and Adaptive Feature Fusion Pyramid Network (AF-FPN) methods were implemented to increase the performance of the standard YOLOv5 model, which indeed increased the accuracy from 60.18% to 62.67%. The performance of the model is low due to the blurring of images captured by the high-speed motion of a vehicle [19]. An indigenous CNN architecture is used for TSD with the dataset having 16 classes viz. "green light", "speed limit", "no parking", "bicycle and pedestrians only", "crossroad 1", "red light", "crosswalk 1", "straight ahead or left turn permitted", "crossroads 2", "traffic division", "no overtaking", "no turns", "stop", "one-way street" and "yellow light". This approach outperforms YOLOv2, and Fast R-CNN, with an average accuracy of 90% in all types of weather conditions. However, authors have developed a model with less training data that could be increased to improve performance in more environments [20]. A combination of Faster R-CNN, and Extreme Learning Machines (ELM) is used over 3 classes. However, the accuracy and performance of the model are not discussed quantitatively, but qualitatively it is stated that combining CNN with ELM increases the accuracy [21].

A study on TSD and classification in the wild constructed a benchmark dataset "Tsinghua—Tencent 100K" covering real-world conditions. The study trained two models CNN and Fast R-CNN which resulted in an accuracy of 88%, and 50%, respectively. The study had been implemented with a minimum number of traffic sign classes that rarely appear in benchmark datasets [22]. Another study presented the YOLOv3 model in detecting temporary traffic control detection for road construction projects. The mentioned study used a dataset containing 8 classes viz. "construction cones", "looper cones", "construction barrels", "construction barricades", "end construction signs", "road construction ahead signs", "right lane reduction signs", "right lane closed ahead signs". The training resulted in a mean Average Precision (mAP) of 90.82%. The proposed model in the mentioned study recognized more than 98% of the temporary traffic signs correctly and approximately 81% of temporary traffic control devices correctly [23]. A design for real-time TSD was implemented with CNN on 50,000 traffic-sign images and reached an accuracy of 97.3%. This model is designed by considering more traffic sign classes, and possible weather conditions affecting the visibility of the signs [24]. The "WAF-LeNet" (an upgraded version of LeNet) is developed to recognize and identify traffic signs for autonomous vehicles. The accuracy attained in the study was 96.4% among 43 classes [25]. Though there is a fairly small amount of research studies revolving around TSD using YOLOv7, research work was carried out to collect, and label the road damage data using Google Street View. The YOLOv7 model is trained with the collected data and results in an F1 score of 81.7% [26]. A study focused on improving the performance of YOLOv5 for the detection of traffic signs in bad weather conditions made use of the Global Context (GC) block, which combined with YOLOv5's results in an accuracy of 79.2% [27]. A study quantitatively demonstrates that the combination of YOLOv7 with a lightweight convolution-based Spatial Pyramidal Pooling Fusion (SPPF) module leads to a significant improvement in model accuracy. The study reports a precise increase of 6.7% in accuracy when incorporating the SPPF module into the YOLOv7 framework [28]. A portable image-based ADS system was developed using the YOLOv5 algorithm and Tesla P100 Graphics Processing Unit (GPU) system. It achieved a remarkable speed of 43.59 frames per second [29]. Multiple studies have utilized a pre-trained model for TSD on large datasets, and have fine-tuned the respective models by using various optimizers [30–32]. Some studies have implemented multi-task learning to simultaneously detect objects like pedestrians and bicycles [33,34]. The use of LiDAR and Radar sensors has come up as one of the ways to increase the accuracy of models for TSD in challenging conditions like low lighting [35]. A unique method is described in a study for analyzing Global Positioning System (GPS) trajectory data to detect vehicle turns, which involves converting the data to image-based data, post-conversion, a personalized CNN model is designed [36,37].

Previous approaches to TSD have used models like YOLOv5 [27], YOLOv7 [28], and CNN [36], which are popular and efficient models. However, these models do not have an attention mechanism, which can limit their performance. The model proposed in this study uses YOLOv7 with CBAM, which is an attention mechanism that helps to improve the model's performance. Specifically, CBAM helps to focus the model's attention on the most important features in an image, which can lead to better object detection, especially in cases where the objects in an image are small or have low contrast.

3 Traffic Sign Detection Using YOLOv7 with CBAM

YOLOv7 is the latest and state-of-the-art object detection model in the family of YOLO single-shot object detection models. YOLOv7 is currently the fastest and best-performing object detection model. YOLOv7 significantly enhances real-time object detection accuracy while lowering inference costs. By cutting around 40% of the parameters and 50% of the processing speed, YOLOv7 effectively beats other well-known object detectors with faster inference speeds, and higher recognition accuracy [38].

3.1 Working Principle

The four components that the YOLO algorithm uses to operate are residual blocks, bounding box regression, Intersection Over Union (IOU), and Non-Maximum Suppression (NMS). The initial component of the residual block divides the original image (A) into N equal-sized grid cells, where N is a hyperparameter. Localizing and determining the object's class using the probability/confidence value is the responsibility of each grid cell. Bounding box regression is the second element that identifies the bounding boxes that correspond to rectangles highlighting all the objects in the image. There can be as many bounding boxes as there are objects within a given image. YOLO uses a single regression module to compute the characteristics of these bounding boxes. Y is the final vector of each bounding box as given in Eq. (1).

$$Y = [P_c, b_x, b_y, b_h, b_w, C_1, C_2, C_3, C_4] \tag{1}$$

where, P_c is the grid's probability score for the cell that contains the object. The bounding box's center's x and y coordinates in relation to the surrounding grid cell are represented by b_x, b_y. The height, and the width of the bounding box are represented by b_h, b_w, respectively. The four classes namely prohibitory, dangerous, mandatory, and others are represented by C_1, C_2, C_3, and C_4, respectively. Despite not all of them being significant, a single object in an image might frequently have many grid box possibilities for prediction. Such grid boxes are to be discarded in order to retain the relevant grid boxes using the third component IOU. IOU always ranges from 0 and 1. The IOU selection threshold is initially set at 0.5. Fig. 2 shows the intersection area divided by the union area which is then calculated for each grid cell by YOLO. Finally, it considers grid cells with an IOU > threshold rather than those predicted to have an IOU $\leq$ threshold.

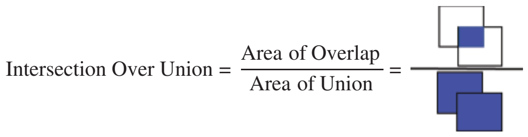


Figure 2: Intersection over union

The final part NMS algorithm is a post-processing technique to remove duplicate and overlapping detections of the same object. When an object is detected, the YOLO algorithm generates multiple bounding boxes with confidence scores indicating the likelihood of an object being present in each box. However, some of these boxes may overlap or contain the same object, resulting in multiple detections for the same object. To address this issue, NMS is used to suppress all but the most confident detection of each object. The algorithm works by first sorting the detected bounding boxes by their confidence scores. Then for each box, it compares its overlap with all other boxes. If the overlap exceeds a certain threshold, the box with the lower confidence score is suppressed. The process is repeated until all boxes have been considered. The generated output helps to improve the overall performance and accuracy of the object detection algorithm. Establishing an IOU threshold is not always adequate since an item may contain several overlapping boxes. Noise might be included if many boxes are overlapped based on an IOU that exceeds the threshold and all those boxes are left unclosed. NMS can be used in these circumstances to keep only the boxes with the highest likelihood of being identified. Hence, the algorithm is designed by initializing the confidence threshold, and IOU threshold values. Then the bounding boxes are organized according to decreasing confidence. If any bounding box contains a confidence threshold 0 that is eliminated. The rest of the bounding boxes are iterated through in a loop beginning with the greatest confidence, and the IOU of the current box with every remaining box that belongs to the same class is calculated. If the IOU of the 2 boxes > IOU_Threshold, then the box with lower confidence is removed from the list of boxes. This operation is repeated until all the boxes are processed in the list. Here is an outline of the code for YOLO, a popular object detection algorithm. Table 1 shows the pseudocode of the steps involved in implementing YOLO.

Table 1: YOLO algorithm

```
Input: Image of 640 × 640 pixels     Output: Image class
∈ {prohibitory, dangerous, mandatory, others}
-----------------------------------------------------------
image = readInputImage()
No_Cells = 7
No_Classes = 4
Th = 0.7
Size_of_step = height(image)/No_Cells
pred_class_array = new_array(size(No_Cells, No_Cells, No_Classes))
pred_bound_box_array = new_array(size(No_Cells, No_Cells, No_Cells, No_Cells))
final_preds = []
for (i < 0; i < No_Cells; i = i + 1):
        for (j < 0; j < No_Cells; j = j + 1):
            cell = image(i: i + Size_of_step, j: j + Size_of_step)
            pred_class_array[i,j] = predict_class(cell)
            pred_bound_box_array[i,j] = predict_bound_box(cell)
            if (pred_bound_box_array[i,j,0, 4] > pred_bound_box_array[i,j,1, 4]):
                    bestfit_bound_box = 1
            else:
                    bestfit_bound_box = 0
            pred_class = max_value_index(pred_class_array[i,j])
        if (pred_bound_box_array[i,j,bestfit_bound_box,4] ∗ max_value(pred_class_array[i,j]) >
        Th):
                    pred = [pred_bound_box_array[i,j,bestfit_bound_box, 0:4], pred_class]
                    final_preds.append(prediction)
print final_preds
```

3.2 Architecture

YOLOv7 can be used in many applications other than object detection, like instance segmentation, pose estimation, etc. In comparison to YOLOv4, YOLOv7 utilizes 36% less processing, reduces the number of parameters by 75%, and generates 1.5% higher AP. When compared to the edge-optimized version, YOLOv4-tiny, and YOLOv7-tiny reduce the number of parameters by 39% and computation by 49% while keeping the same AP. Hence, it can be stated that YOLOv7 is more optimized. A

YOLO architecture is made up of various components, including a head, neck, and backbone. For the inference speed, the effectiveness of the YOLO network's backbone is essential. The full YOLOv7 architecture can be seen in Fig. 3.

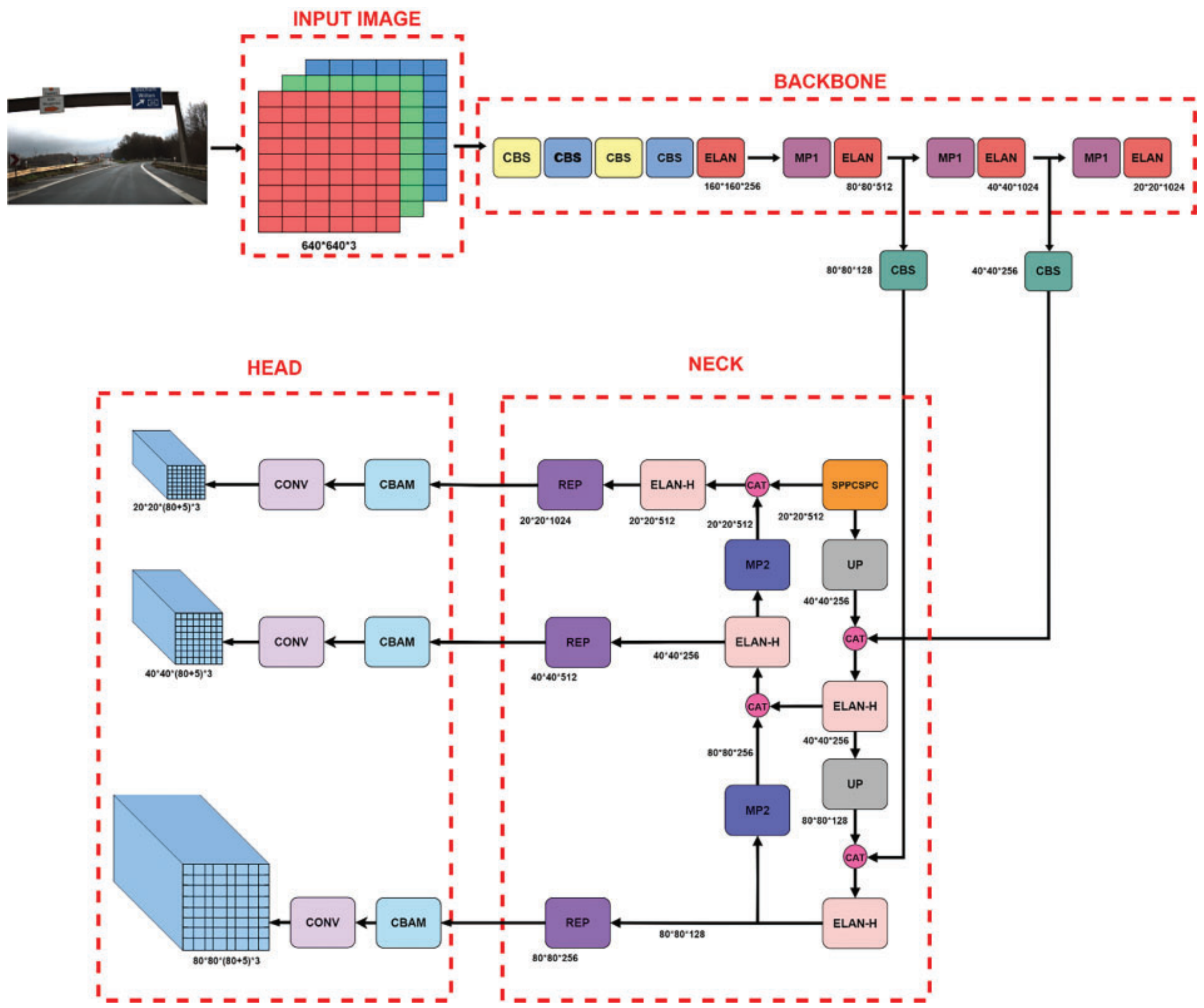


Figure 3: Proposed architecture of YOLOv7 with CBAM

The Extended Efficient Layer Aggregation Network (E-ELAN) helps the model learn better while preserving its original gradient path. To increase the speed and accuracy of the model, E-ELAN considers several variables, including memory cost, input-output channel ratio, element-wise operation, activations, gradient routes, etc. [39]. The CSPDarknet53 serves as the backbone network for the YOLOv7 architecture that makes up the Efficient Layer Aggregation Network (ELAN) model. CSPDarknet53 was created to increase the precision and effectiveness of object detection models. On the other hand, E-ELAN is another YOLOv7 architecture that uses EfficientNet as the backbone network. A series of CNNs called EfficientNet is created to attain cutting-edge accuracy while keeping the model's computing cost to a minimum. The main difference between these two models is the backbone network, i.e., ELAN uses CSPDarknet53, and E-ELAN uses EfficientNet. EfficientNet is more computationally efficient, but it may sacrifice some accuracy compared to CSPDarknet53.

YOLOv7 uses an optimized compound model scaling approach that modifies the characteristics to produce suitable models for various application requirements. For instance, model scaling can improve the resolution of the model, the size of the input image, the depth, or the number of stages, and the width, or the number of channels. The compound scaling technique can keep the model's original design characteristics.

After training, one way to improve the model is by re-parameterizing it. The inference process takes longer, but the outcomes are more substantial. The two forms of ensemble re-parameterizations used to complete models are model level and module level. Model level re-parameterization can be done in two ways. In the first method, distinct sets of data are used to train several models with the same architecture, and then average their weights to get the final model. The second method is to take the average of a model's weight at different epochs. But recently, module-level re-parameterization has been used in a lot of research works. The YOLOv7 contains several heads, including the Lead Head, which is accountable for all the output, and the Auxiliary Head, which helps with training middle layers. To enhance deep network training, a Label Assigner method was created that assigns soft labels after considering ground truth and network prediction results. Reliable soft labels employ optimization techniques to raise the standard and distribution of prediction output in addition to the accuracy of the prediction. However, conventional label assignment generates hard labels based on predetermined norms by directly referencing reality. The YOLOv7 architecture shown above uses kernel sizes such as 3×3, and 1×1 in all its convolution layers with padding of 1 and 2.

Fig. 4 shows a crucial component Cross-Branch Scalability (CBS). It is designed with a convolution layer, Batch Normalization (BN) layer, and a Sigmoid Linear Unit (SILU) activation function to extract images at various scales. Based on the CBS module, which makes up the upper and lower divisions, the MP1 module adds the max-pooling layer. Using max-pooling and the CBS module, the upper division reduces the image's length and width in half. The lower division uses the first CBS module to reduce the image channel in half, the second CBS layer reduces the image's width and length in half, and finally, the Concatenation (CAT) operation is used to combine the features retrieved from the top and lower branches, enhancing the network's ability for feature extraction. The upsampling and CBS modules make up the UP module.

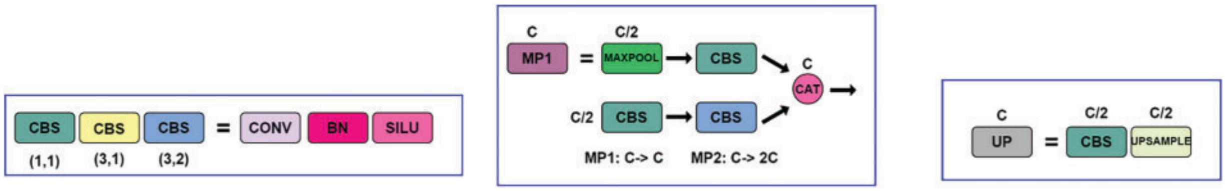


Figure 4: CBS, MP1, UP modules

In Fig. 5, the ELAN module is made up of numerous CBS modules that have been piled on top of one another while maintaining the same input and output feature sizes. The learning capacity of the network is increased without deviating from the initial gradient path by directing the computing units of various feature groups to learn more diverse features. The Spatial Pyramid Pooling Concat Spatial Convolutional (SPPCSPC) module shown in Fig. 6, ELAN-H (Extreme Low-latency Architecture for Network Heads) module, and UP module makes up the majority of the Path Aggregation Feature Pyramid Network (PAFPN) structure that makes up the Neck component of YOLOv7. The bottom-up approach makes it simple to move bottom-level data up to the top level, allowing for the effective fusion of various hierarchical aspects. The CBS module, CAT module, and max-pooling module make up the majority of the SPPCSPC module. SPPCSPC uses different pooling kernel sizes such as 5×5, 9×9, and 13×13. These modules obtain various perception fields through max-pooling. To predict confidence, category, and anchor frame, Head uses Re-parameterization Visual Geometry Group Block (RepVGG) structure to adjust the number of image channels for the output of Neck at three distinct scales and then passes through 1×1 convolution. The model proposed in this study addresses the scale problem in TSD by utilizing the SPPCSPC module in the last layer of the proposed model. Spatial Pyramid Pooling (SPP) allows capturing features at different scales without reducing the input resolution, while Cross Stage Partial (CSP) connections reduce the number of parameters in the proposed model. By incorporating these modules, the model can effectively handle the large variations in object scales commonly encountered in TSD tasks, improving the accuracy of predictions and enhancing the overall performance of the proposed model.

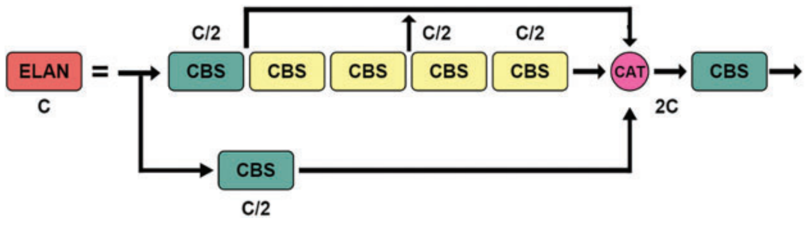


Figure 5: ELAN module

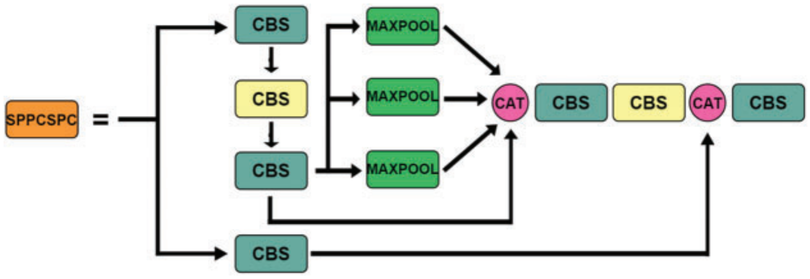


Figure 6: SPPCSPC module

CBAM is a module used to enhance the performance of CNNs by incorporating spatial and channel attention mechanisms. It focuses on capturing both local and global context information from input feature maps, allowing the network to prioritize relevant image regions while suppressing irrelevant ones. The module consists of two components such as spatial attention and channel attention. The spatial attention module captures spatial dependencies among different channels by modeling interdependencies between spatial locations. This enables the network to focus on relevant regions and suppress background regions. The channel attention module captures interdependencies among channels by assessing the importance of each channel in conveying discriminative information. It emphasizes informative channels while suppressing less informative ones. The spatial and channel attention maps are combined to generate an attention map that captures both spatial and channel-wise information. This attention map is used to weigh the feature maps, allowing the network to selectively attend to relevant features. The YOLOv7 model is trained using the sum of the squared error between the predicted bounding boxes and the actual boxes, along with the cross-entropy loss for the class predictions. Its combination of a lightweight backbone network, effective neck, and multi-scale head make it a powerful tool for a variety of computer vision applications.

The technical contribution of this study lies in the integration of three CBAM units before the three outputs of YOLOv7, a model that already detects objects at three different scales. By incorporating the CBAM module, weights are assigned to channel and spatial features of the feature map, which effectively increases the importance of useful features while suppressing irrelevant ones. This attention mechanism enables the proposed model to focus on target regions containing important information that improves accuracy in detecting objects of various sizes.

3.3 Loss Function

The loss function used in YOLOv7 is a mixture of different components, including:

3.3.1 Localization Loss (LL)

This component of the loss measures the difference between the predicted bounding box coordinates and the actual bounding box coordinates. It uses the Mean Squared Error (MSE) loss function to calculate the loss.

$$\mathrm{LL} = \lambda_{\mathrm{cord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} l_{ij}^{obj} \left[(x_i - \widehat{x}_i)^2 + \left(y_i - \widehat{y}_i\right)^2 \right] + \lambda_{\mathrm{cord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} l_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\widehat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_{rmi}} \right)^2 \right] \tag{2}$$

3.3.2 Confidence Loss

This component of the loss measures how confident the model is in its predictions. It calculates the difference between the predicted confidence score and the actual confidence score. The confidence score indicates whether the bounding box contains an object or not. The binary cross-entropy loss function is used to calculate this confidence.

$$\text{Confidence Loss} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} l_{ij}^{obj} (CI_i - \widehat{CI}_j)^2 + \lambda_{no\_obj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} l_{ij}^{no\_obj} (CI_i - \widehat{CI}_j)^2 \tag{3}$$

3.3.3 Classification Loss

This component of the loss measures the difference between the predicted class probabilities and the actual class probabilities. The cross-entropy loss function is used to calculate this classification loss.

$$\text{Classification Loss} = \sum_{i=0}^{S^2} l_{i}^{no\_obj} \sum_{C \in \text{Classes}} (Pr_i\,(C) - \widehat{Pr}_i(C))^2 \tag{4}$$

3.3.4 Total Loss

The overall loss function is a weighted sum of these three components. The weights are hyperparameters that are tuned during training to balance the contributions of the different components. The loss function's ultimate goal is to reduce the difference between predicted, and ground truth bounding boxes, confidence scores, and class probabilities.

$$\text{Total Loss } = \text{LL} + \text{Confidence Loss } + \text{Classification Loss} \tag{5}$$

$$\text{Total Loss} = \lambda_{\text{cord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \left[(x_i - \widehat{x}_i)^2 + \left(y_i - \widehat{y}_i \right)^2 \right] + \lambda_{\text{cord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\widehat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] + \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{obj}} \left(Cl_i - \widehat{Cl}_j \right)^2 + \lambda_{\text{no\_obj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{\text{no\_obj}} \left(CI_i - \widehat{CI}_j \right)^2 + \sum_{i=0}^{S^2} 1_i^{\text{no\_obj}} \sum_{C \in \text{Classes}} (Pr_i(C) - \widehat{Pr}_i(C))^2 \tag{6}$$

where, 1_{ij}^{obj} equals 1 only if box *j*, and cell *i* match, and otherwise it is 0. The variable 1_i^{obj} equals 1 if cell *i* contains an object, and 0 if it dose not. The variable 1_{ij}^{noobj} equals 1 if the box *j* and cell *i* do not match and equals 0 otherwise. The hyperparameters λ_{cord} and $\lambda_{\text{no\_obj}}$ control the weights of each loss term. YOLO makes a prediction in the form a $S * S * (B * 5 + Cl)$ vector. For each grid in the cell, there are *B* bounding box predictions and *Cl* class predictions. The five bounding box outputs for box *j* in cell *i* are the centre coordinates (x_i, y_i), height (h_i) and width (w_i). The confidence index for the bounding box is represented by Cl_{ij}. Pr_i (Cl) is the classification loss.

4 Dataset Description

The images in the dataset are part of the famous German traffic sign dataset and were pre-processed to ensure consistency in size, resolution, and color. The dataset consists of 741 images of traffic signs that are divided into three subsets like training set of 592 images (79.8920%), a validation set of 99 images (13.3606%), and a test set of 50 images (6.7476%) in a stratified method, ensuring that each subset had a proportional representation of each class. Table 2 shows the dataset with four classes such as prohibitory, dangerous, mandatory, and others with a total of 1,213 appearances of traffic signs. The prohibitory class (class 0) has 731 appearances of traffic signs accounting for 45.89% of the dataset. This class includes traffic signs that prohibit certain actions such as no trucks, speed limit, no traffic both ways, and no overtaking. The dangerous class (class 1) has 268 appearances of traffic signs accounting for 18.04% of the dataset. This class includes traffic signs that warn drivers of potential hazards or dangers, such as construction, priority at next intersection, bend left, bend right, bend, uneven road, slippery road, the road narrows, traffic signal, pedestrian crossing, school crossing, dangerous, cycles crossing, animals and snow. The mandatory class (class 2) has 211 appearances of traffic signs accounting for 13.44% of the dataset. This class includes traffic signs that indicate actions that drivers must take, such as a roundabout, go straight, go right, go left, go left or straight, go right or straight, keep right, and keep left. The other class (class 3) has 345 appearances of traffic signs accounting for 22.63% of the dataset. This class includes traffic signs that do not fall into the prohibitory, dangerous, or mandatory categories such as no entry, stop, give way, priority road, and restriction ends.

Table 2: Description of dataset

Class number	Classes	No. of appearances			Total appearances
		Training	Validation	Testing	
0	Prohibitory	577	115	39	731
1	Dangerous	219	40	9	268

(Continued)

Table 2 (continued)

Class number	Classes	No. of appearances			Total appearances
		Training	Validation	Testing	
2	Mandatory	163	33	15	211
3	Other	274	54	17	345

5 Experimental Setup

The objective of this study is to achieve high accuracy while keeping the model size and computational complexity low, making it suitable for deployment on embedded systems. The dataset contains 741 images of traffic signs with varying lighting conditions, occlusions, and backgrounds. The pre-processed version of the dataset is used, where images were cropped and resized to 416 $\times$ 416 $\times$ 3 pixels and annotated properly. The dataset was divided into training (79.89%), validation (13.36%), and test (6.747%) sets. The latest version YOLOv7 object detection model is used. The model has three components that predict the class, location, and confidence of the traffic sign detected. It was first trained using the SGD optimization algorithm with batch sizes 8 and 16. It had the following values of hyperparameter with a learning rate of 0.001, weight decay of 0.0005, and momentum of 0.937. The model was trained for 100 epochs, and the total training time was 1.868, and 1.845 h for batch sizes 8, and 16, respectively. Then the model was trained using the Adam optimization algorithm with batch sizes 8, and 16. It had the following values of hyperparameter with a learning rate of 0.001, weight decay of 0.0005, and momentum of 0.937. The model was trained for 100 epochs, and the total training time was 1.916, and 1.862 h for batch sizes 8, and 16 respectively. At last, the model was trained using the AdamW optimization algorithm with batch sizes 8, and 16. It had the following values of hyperparameter with a learning rate of 0.001, weight decay of 0.0001, and momentum of 0.937. The model was trained for 100 epochs, and the total training time was 1.942, and 1.857 h for batch sizes 8, and 16, respectively.

The evaluation metric mAP is used to measure the accuracy of the model in detecting traffic signs of different sizes, and aspect ratios. Precision, recall, and F1 score are used as secondary evaluation metrics [50,95]. The training was conducted on a single Tesla K80 GPU which is available for free version of Google Colab. The calculation of mAP involves calculating the AP for each class of the detected object and then averaging those AP values across all classes. AP is the area under the precision *vs.* recall curve.

$$\mathrm{mAP} = \frac{1}{\mathrm{N}} * \sum\nolimits_{\mathrm{i=1}}^{\mathrm{N}} \mathrm{AP_i} \tag{7}$$

The precision-recall curve shows how the precision and recall of the algorithm vary with the detection threshold. The precision is the fraction of detected objects that are correct. The recall is the fraction of true positive predictions among all the real positive cases. Precision and recall are calculated as follows:

$$\text{Precision} = \frac{\text{True Positive}}{(\text{True Positive} + \text{False Positive})} \text{ and Recall } = \frac{\text{True Positive}}{(\text{True Positive} + \text{False Negative})} \tag{8}$$

The F1 score is the harmonic mean of precision and recall, calculated as follows:

$$\text{F1 score} = \frac{(\text{Precision} + \text{Recall})}{2 * (\text{Precision} * \text{Recall})} \tag{9}$$

A good model should have a high F1 score, high recall, high accuracy, and high precision. Precision and recall, however, typically trade off against one another. To determine the ideal balance between precision and recall, use the F1 score. The precision-recall curve shows how accuracy and recall are traded off for different thresholds. While a low false negative rate is related to great recall, a low false positive rate is related to superior accuracy. A large area under the precision-recall curve indicates that the precision and recall are high.

6 Results and Discussion

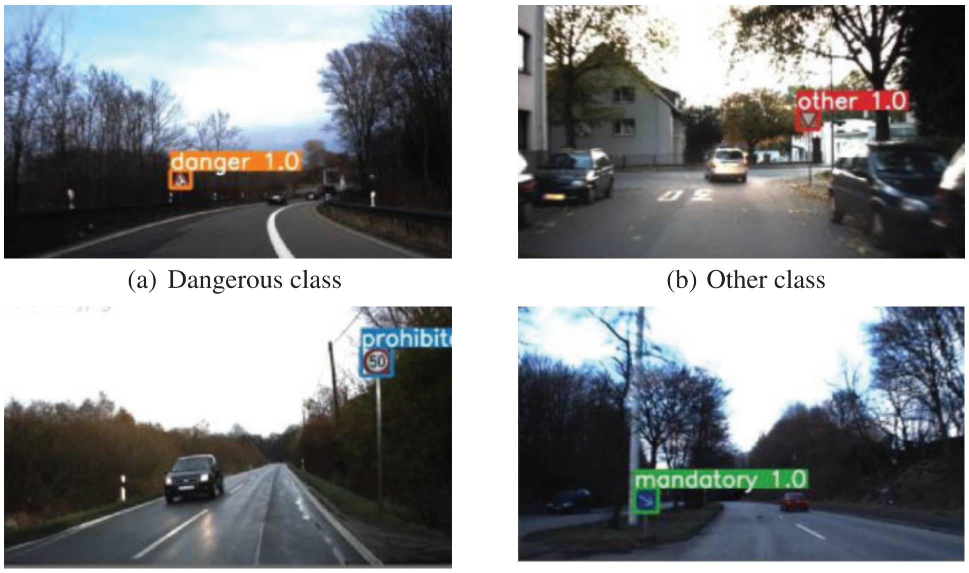
The approach used in this study is evaluated using the standard mAP metric which measures the accuracy of object detection by computing the AP over all possible levels of recall. Table 3 shows the YOLOv7 model's performance under different training configurations using four optimization algorithms such as Adam, SGD, AdamW, and RMSProp with batch sizes of 8, and 16. The outcomes demonstrated that both during training and testing, the YOLOv7 model was able to obtain high precision and recall values. This proved that the model has a low rate of FP and FN when identifying objects in images. The mAP metric measures the model's ability to detect objects at different IOU thresholds. In particular, mAP50 evaluates the AP at an IOU threshold of .5 or 50 percent, while mAP [50,95] measures the AP across all IOU thresholds from .5 to .95, with increments of .05 or 5 percent. The YOLOv7 model achieves high mAP values for both mAP50 and mAP [50,95] indicating that the model can accurately detect objects at different IOU thresholds.

Table 3: Results of all the combinations of YOLOv7 for TSD

Configuration	Training				Testing			
	Precision	Recall	mAP50	mAP [50,95]	Precision	Recall	mAP50	mAP [50,95]
Adam (batch-size 8)	0.998	1	0.997	0.89	1	1	1	0.894
SGD (batch-size 8)	0.978	0.972	0.986	0.858	0.997	0.978	0.981	0.855
AdamW (batch-size 8)	0.98	0.972	0.989	0.849	0.981	0.978	0.981	0.85
Rmsprop (batch-size 8)	0.912	0.975	0.984	0.824	0.975	0.951	0.98	0.825
Adam (batch-size 16)	0.994	1	0.996	0.88	1	1	1	0.884
SGD (batch-size 16)	0.872	0.629	0.723	0.509	0.961	0.55	0.547	0.42
AdamW (batch-size 16)	0.979	1	0.996	0.88	0.986	1	1	0.883
Rmsprop (batch-size 16)	0.967	0.983	0.994	0.83	0.98	0.975	0.985	0.836

The results show that Adam performs consistently better than SGD in terms of precision, recall, and mAP50, regardless of batch size. On the other hand, AdamW is similar to Adam in most cases but shows slightly lower performance in terms of mAP50 for batch size 8. In Table 3, it can be observed that the batch size significantly impacts the performance of the YOLOv7 model. For instance, SGD shows lower precision, recall, and mAP values compared to the other configurations for batch size 16. This suggests that SGD is less effective in handling larger batch sizes, possibly due to its inherent instability in noisy, and high-dimensional optimization spaces.

Fig. 7 shows the predicted images for each class including dangerous, other, prohibitory, and mandatory signs, and each image is accompanied by the corresponding confidence level. Fig. 7a inferred that the model accurately predicted dangerous class signs with high confidence levels, indicating its ability to detect, and classify potentially hazardous situations on the road. Fig. 7b shows that the model was able to predict other class signs with high confidence levels, indicating its ability to accurately classify signs that do not fit into well-defined categories. Fig. 7c tells that the model accurately predicted prohibitory class signs with high confidence levels, indicating its ability to detect and classify signs that restrict certain behaviors on the road. Similarly, Fig. 7d indicates that the model was able to predict mandatory class signs with high confidence levels, indicating its ability to accurately detect and classify signs that require specific actions from drivers.



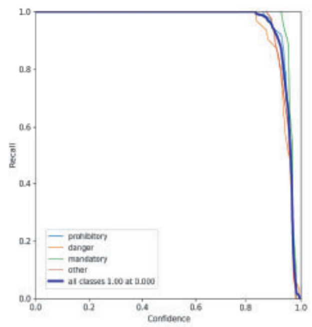
(a) Dangerous class (b) Other class

(c) Prohibitory class (d) Mandatory class

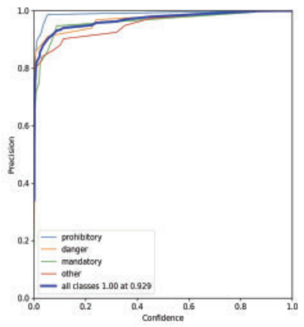
Figure 7: Predicted image for each class

Figs. 8a and 9a illustrate that the YOLOv7 model performs well as it has high recall at low confidence, and as the confidence increases the recall values decrease, and finally confidence becomes 1 when recall becomes zero. Figs. 8b and 9b illustrate that the model is performing well as it has high precision at high confidence, and as the confidence threshold decreases, the precision also decreases. Figs. 8c and 9c prove that the curve near the upper right corner indicates the increase in recall, the reduction in precision is not immediately apparent, and the overall performance of the model is better. Figs. 8d and 9d show that the model has a high F1 score at high confidence, and as the confidence threshold decreases, the F1 score also decreases. It proves that the model becomes less conservative and makes more predictions. Both the models Adam with batch sizes 8, and 16 are performing well in the dataset. However, the model with batch size 8 is performing slightly better than the one with 16. It can be inferred that the precision and the mAP50 of Adam with batch size 8 is more than the Adam with batch size 16.

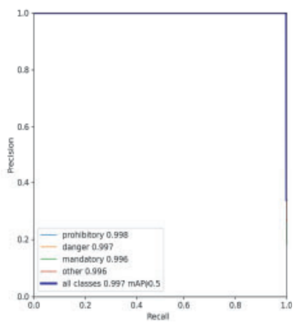
Fig. 10a illustrates that the model performs well as it has high recall at low confidence, and as the confidence increases the recall values decrease, and finally confidence becomes 1 at recall is zero. Fig. 10b shows that the model is performing well but the performance is less than the Adam models as the class-wise prediction is deviating from the overall prediction. By comparing Figs. 10c, 8c and 9c exhibited that the Adam optimizer performs better than the SGD optimizer. Fig. 10d shows that the model has a high F1 score at high confidence, although not as high as the models with Adam, and as the confidence decreases, the F1 score also decreases. It showed that the model becomes less conservative.



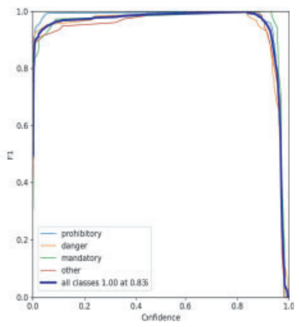
(a) Confidence vs recall



(b) Precision vs confidence

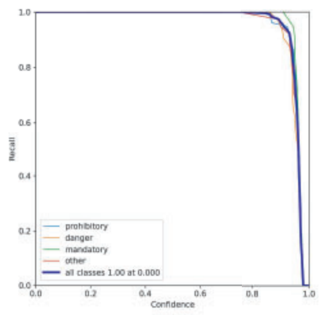


(c) Precision vs recall

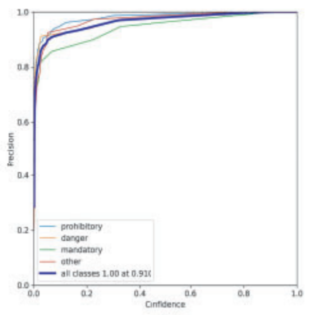


(d) F1 vs confidence

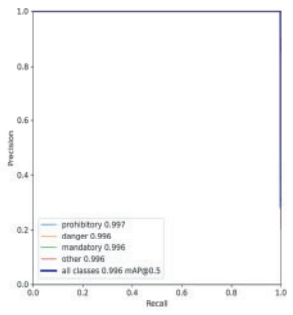
Figure 8: Adam optimizer with batch-size 8



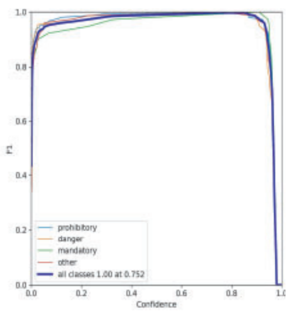
(a) Confidence vs recall



(b) Precision vs confidence

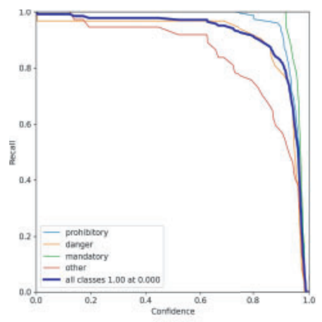


(c) Precision vs recall

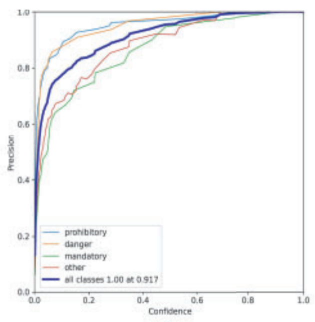


(d) F1 vs confidence

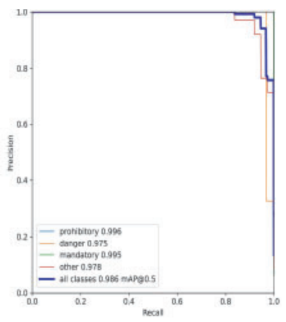
Figure 9: Adam optimizer with batch-size 16



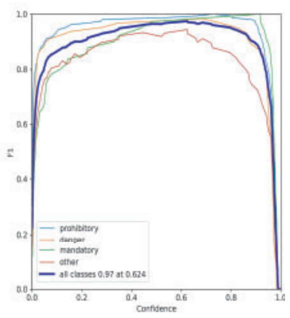
(a) Confidence vs recall



(b) Precision vs confidence



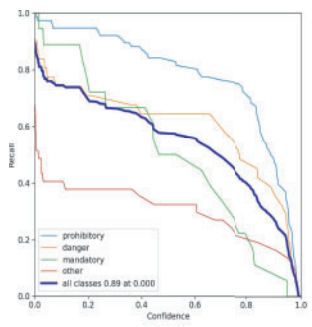
(c) Precision vs recall



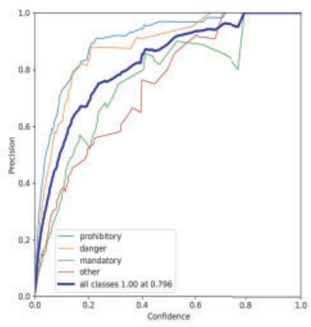
(d) F1 vs confidence

Figure 10: SGD optimizer with batch-size 8

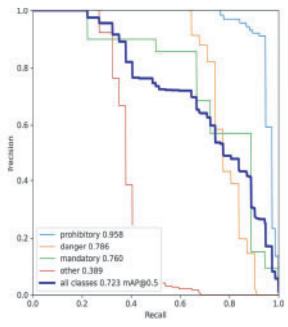
Fig. 11 conveys that the model is performing very poorly compared to the previous Adam models, and SGD with batch size 8. For instance, the precision *vs*. recall curve in Fig. 11c depicts much below at the top corner indicating the model has not learned properly and is struggling to predict the test images and in Fig. 11d the model has a very low F1 score at high confidence, and as the confidence decreases, the F1 score also decreases.



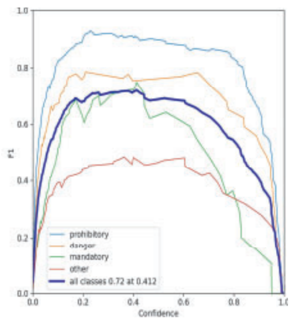
(a) Confidence vs recall



(b) Precision vs confidence



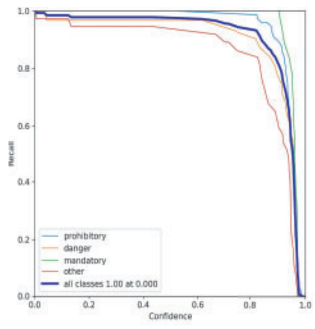
(c) Precision vs recall



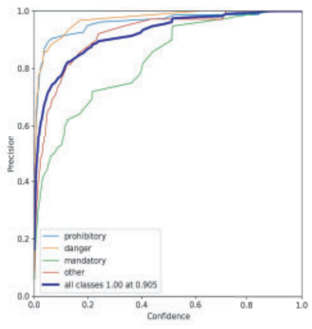
(d) F1 vs confidence

Figure 11: SGD optimizer with batch-size 16

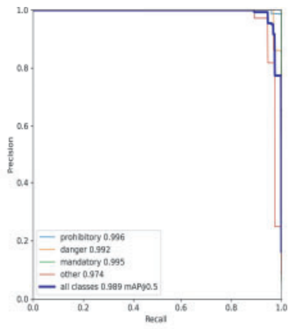
Figs. 12 and 13 indicate that the models AdamW with batch-size 8 and 16 perform well but not as well as models with Adam. Although these models perform better than the SGD optimizer for this particular dataset. The AdamW with batch size 8 performs better than the AdamW with batch size 16. This results that the model with the less batch size performs better for this dataset.



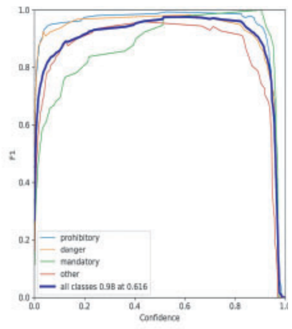
(a) Confidence vs recall



(b) Precision vs confidence

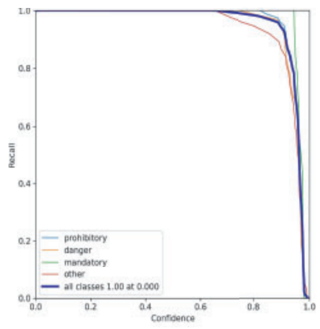


(c) Precision vs recall

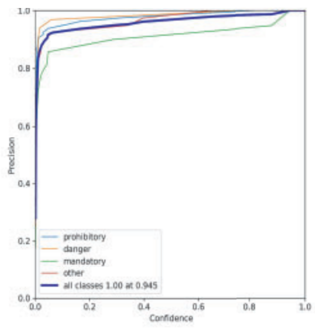


(d) F1 vs confidence

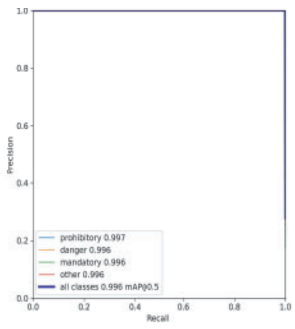
Figure 12: AdamW optimizer with batch-size 8



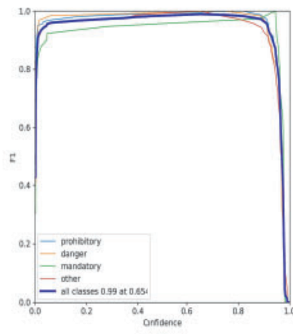
(a) Confidence vs recall



(b) Precision vs confidence



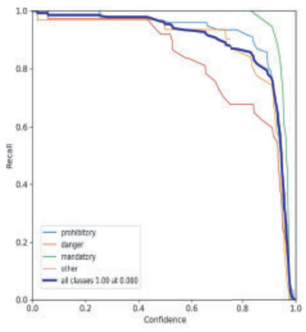
(c) Precision vs recall



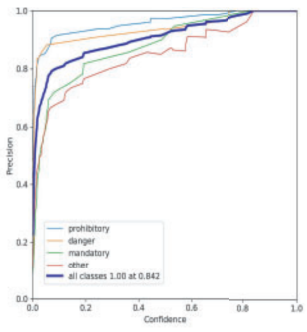
(d) F1 vs confidence

Figure 13: AdamW optimizer with batch-size 16

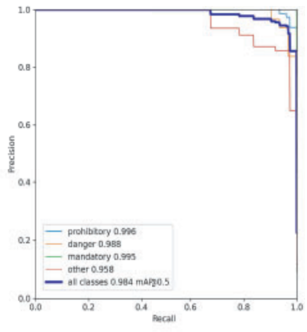
Figs. 14 and 15 illustrate that the models RMSProp with batch-size 8 and 16 perform well but not as well as models with other optimizers. In comparison between the different batch sizes, RMSProp with batch size 16 performs better than the RMSProp with batch size 8. This results that the model with the more batch size performs better for this dataset. However, its performance is lesser than other all optimizers.



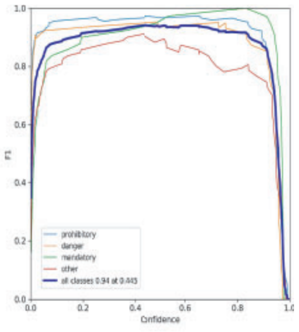
(a) Confidence vs recall



(b) Precision vs confidence

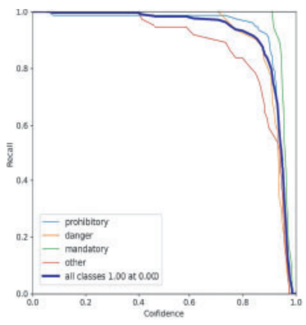


(c) Precision vs recall

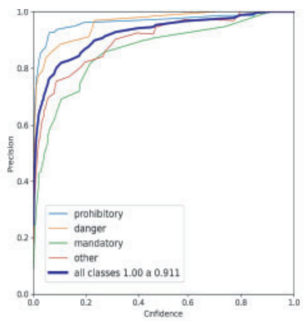


(d) F1 vs confidence

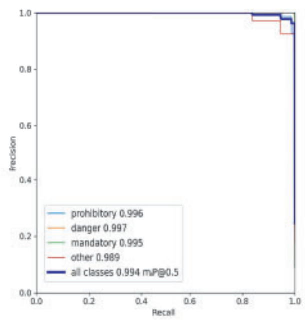
Figure 14: RMSProp optimizer with batch-size 8



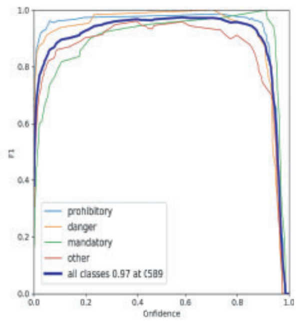
(a) Confidence vs recall



(b) Precision vs confidence



(c) Precision vs recall



(d) F1 vs confidence

Figure 15: RMSProp optimizer with batch-size 16

The results proved that Adam is a more effective optimization algorithm for training the YOLOv7 model and that the performance of the model varies with batch size. This information can be useful for selecting the best configuration for their specific use case. Table 4 shows the comparison of the proposed model with existing models.

Table 4: Comparison of results

Model	Dataset name	Dataset size	mAP50
YOLOv5 with AF-FPN [27]	TT100K	100000	0.6514
YOLOv5 [29]	GTSRB	2500	0.765
YOLOv7 [40]	Europian traffic sign dataset	352	0.975
Ours (YOLOv7 + CBAM)	GTSRB	741	0.997

Overall, the results illustrate the effectiveness of the YOLOv7 model in object detection tasks and provide valuable insights into its performance under different training configurations. This study concluded that Adam with batch size 8 is the most effective of all the combinations above for the TSD use case.

Considering the proposed model's effectiveness, it is important to acknowledge its potential limitations in extreme weather conditions and low-lighting scenarios. Since the model is trained on images captured under specific conditions, its performance may be compromised when exposed to diverse and challenging environments. To enhance the model's generalizability and improve its performance, a more diverse range of input images can be included that encompass varied conditions during the training process. This will enable the model to learn and adapt to different environmental factors, ultimately enhancing its robustness and reliability in real-world applications.

7 Conclusion and Future Scope

The proposed study is implemented with four optimization algorithms, namely Adam, SGD, AdamW, and RMSProp with different batch sizes for TSD using YOLOv7. The integration of CBAM improved the model's performance by focusing on the spatial and channel regions in the input. The evaluation has been performed on both the training, and testing datasets by considering four different metrics namely Precision, recall, mAP50, and mAP [50,95]. The experimental results have shown that the Adam optimizer with batch-size 8 and 16 achieves the highest accuracy in terms of all four metrics for both training and testing datasets. Specifically, the precision and recall rates of the Adam optimizer are very high. It proved that the model can correctly identify traffic signs in the input images with high accuracy. Moreover, the results indicate that the choice of batch size can also have a substantial effect on the accuracy of the model. In general, the smaller batch size can lead to better performance, but it also increases the training time. Hence, the trade-off between accuracy, and training time should be considered while selecting the batch size.

The results obtained in this research provide a strong basis for future work in the field of traffic sign detection. There are several directions in which this work can be extended. One prominent direction for future work is optimizing the performance of YOLOv7 specifically for real-time traffic sign detection on embedded systems like Raspberry Pi and NVIDIA Jetson. By enhancing the efficiency and speed of the model, its applicability in resource-constrained environments can be significantly expanded, enabling practical implementation in various settings. Another crucial aspect that holds great potential is the interpretability of deep learning models. In recent years, there has been a growing interest in understanding and visualizing the decision-making processes of such models. Hence, future research can delve into exploring the interpretability of YOLOv7 for traffic sign detection. This can involve visualizing saliency maps or activation patterns of the model to gain insights into how it makes predictions. These directions hold immense potential for advancing traffic sign detection systems, paving the way for improved road safety, efficient traffic management, and safer autonomous vehicles in the future.

Acknowledgement: The authors would like to thank the editors and reviewers.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: conceptualization, validation, review and editing: Kuppusamy Pothanaicker; supervision: Kuppusamy Pothanaicker and Celestine Iwendi; methodology, formal analysis, result analysis, writing: Sanjay Mythili and Deepashree Pradeep Vaideeswar. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Data used in this study is available from Traffic Signs Dataset in YOLO format, Version 1. Retrieved December 26, 2022 from https://www.kaggle.com/datasets/valentynsichkar/traffic-signs-dataset-in-yolo-format.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] Cotney Attorneys and Consultants, "5 common challenges with roadway construction projects, infrastructure law," 2021. [Online]. Available: https://www.cotneycl.com/5-common-challenges-with-roadway-construction-projects/

[2] I. Ahmed, "Road infrastructure and road safety," *Transport and Communications Bulletin for Asia and the Pacific*, vol. 83, pp. 19–25, 2013.

[3] Á. A. García, J. A. Á. García and L. M. Soria-Morillo, "Evaluation of deep neural networks for traffic sign detection systems," *Neurocomputing*, vol. 316, no. 6, pp. 332–344, 2018.

[4] N. B. Soni and J. Saraswat, "A review of IoT devices for traffic management system," in *Proc. of Int. Conf. on Intelligent Sustainable Systems (ICISS)*, Palladam, India, IEEE, pp. 1052–1055, 2017.

[5] M. Kopsida, I. Brilakis and P. A. Vela, "A review of automated construction progress monitoring and inspection methods," in *Proc. of the 32nd Int. Council for Research and Innovation in Building and Construction (CIB)*, Eindhoven, The Netherlands, pp. 421–431, 2015.

[6] M. Akhtar and S. Moridpour, "A review of traffic congestion prediction using artificial intelligence," *Journal of Advanced Transportation*, vol. 2021, no. 9, pp. 1–18, 2021.

[7] S. Olugbade, S. Ojo, A. L. Imoize, J. Isabona and M. O. Alaba, "A review of artificial intelligence and machine learning for incident detectors in road transport systems," *Mathematical and Computational Applications*, vol. 27, no. 5, pp. 77, 2022.

[8] S. Srivastava, A. V. Divekar, C. Anilkumar, I. Naik, V. Kulkarni *et al.,* "Comparative analysis of deep learning image detection algorithms," *Journal of Big Data*, vol. 8, no. 1, pp. 1–27, 2021.

[9] A. Bochkovskiy, C. Y. Wang and H. Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[10] Y. Tian, Y. Zhang and H. Zhang, "Recent advances in stochastic gradient descent in deep learning," *Mathematics*, vol. 11, no. 3, pp. 682, 2023.

[11] A. Gupta, "A comprehensive guide on deep learning optimizers," 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/

[12] A. F. Magnussen, N. Le, L. Hu and W. E. Wong, "A survey of the inadequacies in traffic sign recognition systems for autonomous vehicles," *International Journal of Performability Engineering*, vol. 16, no. 10, pp. 1588, 2020.

[13] Š. Toth, "Difficulties of traffic sign recognition," in *Proc. of Mathematics Applied to ICT*, Banska Bystrica, Slovakia, 2012.

[14] C. Liu, S. Li, F. Chang and Y. Wang, "Machine vision based traffic sign detection methods: Review, analyses and perspectives," *IEEE Access*, vol. 7, pp. 86578–86596, 2019.

[15] S. Gunasekara, D. Gunarathna, M. B. Dissanayake, S. Aramith and W. Muhammad, "Deep Learning based autonomous real-time traffic sign recognition system for advanced driver assistance," *International Journal of Image, Graphics and Signal Processing*, vol. 14, no. 6, pp. 70–83, 2022.

[16] M. Bichkar, S. Bobhate and C. Sonal, "Traffic sign classification and detection of Indian traffic signs using deep learning," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 7, no. 3, pp. 215–219, 2021.

[17] J. Zhang, Z. Xie, J. Sun, X. Zou and J. Wang, "A cascaded R-CNN with multiscale attention and imbalanced samples for traffic sign detection," *IEEE Access*, vol. 8, pp. 29742–29754, 2020.

[18] Y. Zhu and W. Q. Yan, “Traffic sign recognition based on deep learning,” *Multimedia Tools and Applications*, vol. 81, no. 13, pp. 17779–17791, 2022.

[19] J. Wang, Y. Chen, Z. Dong and M. Gao, “Improved YOLOv5 network for real-time multi-scale traffic sign detection,” *Neural Computing and Applications*, vol. 35, no. 10, pp. 1–13, 2022.

[20] R. Hasegawa, Y. Iwamoto and Y. W. Chen, “Robust Japanese road sign detection and recognition in complex scenes using convolutional neural networks,” *Journal of Image and Graphics*, vol. 8, no. 3, pp. 59–66, 2020.

[21] W. Li, X. Na, P. Su and Q. Zhang, “Traffic sign detection and recognition based on CNN-ELM,” *Journal of Physics: Conference Series*, vol. 1848, no. 1, pp. 012106, 2021.

[22] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li *et al.,* “Traffic-sign detection and classification in the wild,” in *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, pp. 2110–2118, 2016.

[23] S. Seo, D. Chen, K. Kim, K. Kang, D. Koo *et al.,* “Temporary traffic control device detection for road construction projects using deep learning application,” in *Proc. of Construction Research Congress 2022*, Arlington, Virginia, pp. 392–401, 2022.

[24] A. Shustanov and P. Yakimov, “CNN design for real-time traffic sign recognition,” *Procedia Engineering*, vol. 201, no. 4, pp. 718–725, 2017.

[25] W. Farag, “Recognition of traffic signs by convolutional neural nets for self-driving vehicles,” *International Journal of Knowledge-based and Intelligent Engineering Systems*, vol. 22, no. 3, pp. 205–214, 2018.

[26] V. Pham, D. Nguyen and C. Donan, “Road damage detection and classification with YOLOv7,” in *Proc. of 2022 IEEE Int. Conf. on Big Data*, Osaka, Japan, pp. 6416–6423, 2022.

[27] T. P. Dang, N. T. Tran, V. H. To and M. K. T. Thi, “Improved YOLOv5 for real-time traffic signs recognition in bad weather conditions,” *The Journal of Supercomputing*, vol. 79, no. 13, pp. 10706–10724, 2023.

[28] Z. Jia, S. Sun and G. Liu, “Real-time traffic sign detection based on weighted attention and model refinement,” *Neural Processing Letters*, vol. 17, no. 7, pp. 1–17, 2023.

[29] E. Güney, C. Bayilmiş and B. Cakan, “An implementation of real-time traffic signs and road objects detection based on mobile GPU platforms,” *IEEE Access*, vol. 10, pp. 86191–86203, 2022.

[30] H. Luo, Y. Yang, B. Tong, F. Wu and B. Fan, “Traffic sign recognition using a multi-task convolutional neural network,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 4, pp. 1100–1111, 2017.

[31] J. Stallkamp, M. Schlipsing, J. Salmen and C. Igel, “The German traffic sign recognition benchmark: A multi-class classification competition,” in *Proc. of the 2011 Int. Joint Conf. on Neural Networks*, San Jose, CA, USA, pp. 1453–1460, 2011.

[32] Y. Chen, D. Zhao, L. Lv and Q. Zhang, “Multi-task learning for dangerous object detection in autonomous driving,” *Information Sciences*, vol. 432, no. 3, pp. 559–571, 2018.

[33] G. N. Doval, A. Al-Kaff, J. Beltrán, F. G. Fernández and G. F. López, “Traffic sign detection and 3D localization via deep convolutional neural networks and stereo vision,” in *Proc. of IEEE Int. Conf. on Intelligent Transportation Systems (ITSC)*, Auckland, New Zealand, pp. 1411–1416, 2019.

[34] W. Min, R. Liu, D. He, Q. Han, Q. Wei *et al.,* “Traffic sign recognition based on semantic scene understanding and structural traffic sign location,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 15794–15807, 2022.

[35] M. Ikhlayel, A. J. Iswara, A. Kurniawan, A. Zaini and E. M. Yuniarno, “Traffic sign detection for navigation of autonomous car prototype using convolutional neural network,” in *Proc. of Int. Conf. on CENIM*, Surabaya, Indonesia, pp. 205–210, 2020.

[36] C. Y. Wang, A. Bochkovskiy and H. Y. M. Liao, “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors,” in *Proc. of IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, Vancouver, Canada, pp. 7464–7475, 2023.

[37] M. A. Rahim, S. D. Khan, S. Khan, M. Rashid, R. Ullah *et al.,* “A novel spatio-temporal deep learning vehicle turns detection scheme using GPS-only data,” *IEEE Access*, vol. 11, pp. 8727–8733, 2023.

[38] W. Liu, R. G. Lu and X. L. Liu, “Traffic sign detection and recognition via transfer learning,” in *Proc. of Chinese Control and Decision Conf. (CCDC)*, Shenyang, China, pp. 5884–5887, 2018.
[39] Á.A. García, J. A. Á. Garcia and L. M. Soria-Morillo, “Deep neural network for traffic sign recognition systems: An analysis of spatial transformers and stochastic optimization methods,” *Neural Networks*, vol. 99, no. 3, pp. 158–165, 2018.
[40] F. F. Riya, S. Hoque, M. S. H. Onim, E. Michaud and E. Begoli, “Effects of real-life traffic sign alteration on YOLOv7-an object recognition model,” *arXiv preprint arXiv:2305.05499*, 2023.