ISSN: 0067-2904

Traffic Sign Detection Using You Only Look Once (YOLOv3) Technique

Dhorgam Fadhel Abdulabass *, Matheel Emaduldeen Abdulmunim
Department of Computer Science, University of Technology, Baghdad, Iraq

Received: 17/12/2022 Accepted: 18/7/2023 Published: 30/10/2024

Abstract

Although deep learning-based object detection has produced excellent performance, there are still many issues with images from real-world capture, including rotating jitter, blurring, and noise deletion. The impact of these issues on object detection is significant. The main goal of this paper is to develop a real-time "You Only Look Once" (YOLOv3) algorithm to detect traffic signs. Compared to all other object detection algorithms, the YOLO method has a number of benefits. In contrast to different algorithms, YOLO looks at the image entirely by making predictions of the bounding boxes utilizing a convolutional neural network (CNN), determining the probability of each class for these boxes, and detecting the image more quickly. The proposed method applies a single neural network to the entire image. Then this network divides that image into regions, which provide the bounding boxes and also predict probabilities for each region. These generated bounding boxes are weighted by the predicted probabilities. The proposed method achieves 99% accuracy in the detection process.

Keywords: Neural Network, YOLO, Object Detection, Deep Learning, CNN.

الكشف عن إشارات المرور باستخدام YOLOv3

ضرغام فاضل عبد العباس ، مثيل عماد الدين عبد المنعم
قسم علوم الحاسوب، الجامعة التكنولوجية، بغداد، العراق

الخلاصة

على الرغم من أن اكتشاف الكائن المستند إلى التعلم العميق قد أنتج أداءً جيدًا للغاية، إلا أنه لا تزال هناك العديد من المشكلات المتعلقة بالصور من الالتقاط في العالم الحقيقي، بما في ذلك الاهتزاز المتناوب والتشويش والضوضاء وما إلى ذلك. تأثير هذه المشكلات على اكتشاف الكائن كبير. الهدف الرئيسي هو العثور على الأشياء باستعمال إستراتيجية .You Only Look Once (YOLO) بالمقارنة مع جميع خوارزميات اكتشاف الكائنات الأخرى، فإن طريقة YOLO لها عدد من الفوائد. على عكس الخوارزميات المختلفة، ينظر YOLO إلى الصورة بالكامل من خلال التنبؤ بالمربعات المحيطة باستعمال شبكة عصبية تلافيفيه (CNN) تحدد احتمالية كل فئة لهذه الصناديق واكتشاف الصورة بسرعة أكبر. الخوارزميات الأخرى، مثل CNN و Fast–

*Email: cs.20.24@grad.uotechnology.edu.iq

CNNلا تفعل ذلك. في هذا البحث، تم تطوير خوارزمية yolov3 في الوقت الفعلي لاكتشاف إشارات المرور.
يحقق النظام المقترح دقة تصل إلى 99٪.

1. Introduction

Computers are used in computer vision (CV) to process and comprehend images and videos. Among the many tasks that fall under the category of computer vision is object detection, which is utilized to identify objects in images or videos [1], [2], and [3]. Self-driving cars, face recognition, car plate recognition, and applications that help the blind and visually impaired identify objects in their environment are just a few examples of the many uses for object detection [4]. YOLO, CNN, region-based convolutional neural networks (R-CNN), and other algorithms are among those used for object detection [5].

The algorithm, YOLOv3, is a quick and precise method to find objects. A neural network output processing algorithm and a CNN make up YOLOv3 [6]. Deep neural networks, such as the CNN, are comparable to the human visual cortex. In the CNN, there is a single input layer, possibly more hidden layers, and a single outcome layer [7]. The images are initially fed into the neural network through the CNN's input layer. The input layer has exactly as many neurons as there are features. Convolutional, activation, pooling, and fully connected layers make up the hidden layer. At least a feature map is produced by the CNN's one convolution layer by computing the dot product between the connected region of the input and the weights. Based on the convolutional layer and the activation layer together, the training process is sped up by eliminating the negative values. The feature map is then simplified by utilizing the pooling layer to down-sample (or pool) the activation layer's output [8]. A fully connected layer is used to connect these layers' output. The fully connected layer, which has all the labels that need to be classified, is a one-dimensional layer that generates scores for each classification label. The output layer, which is the final layer of the CNN, has an equal number of classes and neurons [9], [10], [11], and [12]. The CNN structure is shown in Figure 1.

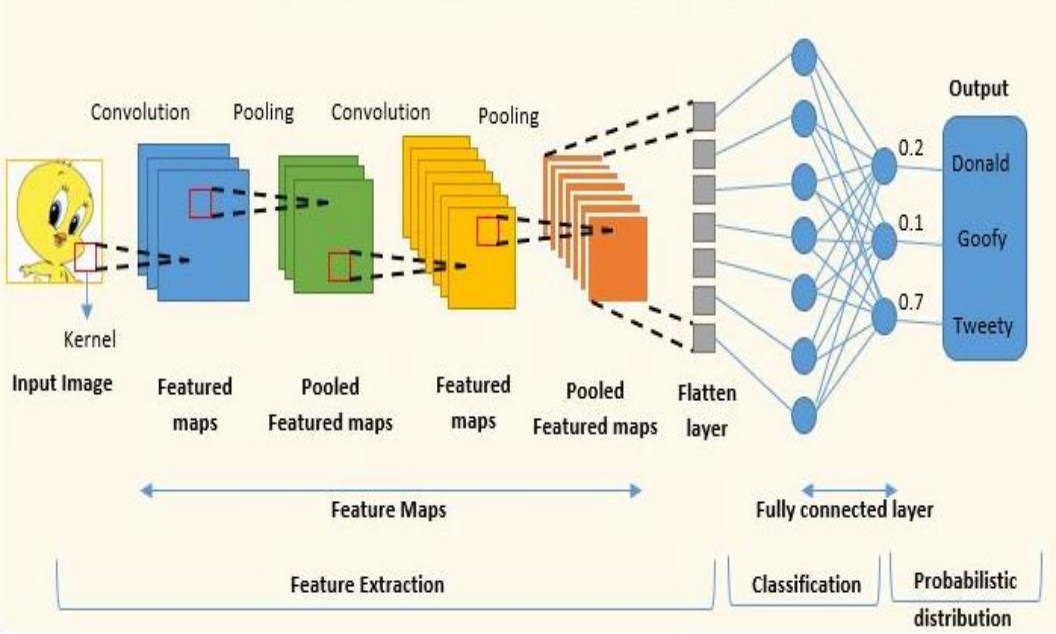

Figure 1: The CNN structure [10].

The remaining portion of this article is organized as follows: A brief history of the studies related to the suggested strategy is given in Section 2. The proposed object detection system is

described in Section 3. The deep neural network's training procedure is depicted in Section 4. Section 5 presents the experimental findings and discusses the suggested approach. Finally, the conclusions are provided in Section 6.

2. Literature Review

There have been numerous attempts to use deep learning algorithms like YOLO, RCNN, and CNN, to name a few, to detect and recognize objects. In this study, a research review is carried out to comprehend a few of these algorithms [13].

Using CNN and the Custom dataset, a model for showing diabetic retinopathy from images of the retina was developed by R. Parmar et al. (2017). Accuracy was 85% [14]. YOLOv3 with the Berkley Deep Drive dataset was used in the implementation of a system by Aleksa Corovi et al. (2018) for detecting traffic participants. The system can recognize five different object classes in various driving situations, including trucks, cars, traffic signs, pedestrians, and traffic lights, in constant lighting conditions, achieving an accuracy of 63% [15]. The Geetha Neravati et al. 2022 accuracy achieved was 98%, and they discovered that eight seconds were needed to detect the objects for each frame [16].

A system for sign language translation was developed by Azher Atallah et al. in 2020. A custom dataset and CNN with TensorFlow were employed. In the proposed system, a voice is created from the sign language. Forty hand gestures were classified, achieving an accuracy of 98% [17]. To assist blind people, Sunit Vaidya et al. 2020 improved an Android application and a web application for object detection. In the proposed system, the YOLOv3 and Coco datasets were employed. The maximum accuracy, according to the authors, was 89% for web applications and 85.5% for mobile devices. Two seconds were needed to detect each object, and this time grew longer as the number of objects increased [18]. An object detection model was put into practice in optical remote sensing images by A. S. Mahmoud et al. in 2020. Datasets from the NWPU-VHR-10 and Mask RCNN were used. Ten different object types can be detected by the model. The maximum accuracy was 95%, and 7.1 seconds were needed for detection [19]. Raspberry Pi, YOLOv3, and the Coco dataset were used by Shifa Shaikh et al. (2020) to implement an object detection system. For a person, a chair, a clock, and a cell phone, the accuracy was 100%, and the general performance accuracy was 95% [20].

In previous studies, several CNN algorithms were used for object detection and classification. The speed and accuracy were not sufficient for a real-time procedure for traffic sign detection and classification, and a dataset was used in non-multiple lighting conditions. In this regard, the custom dataset in different lighting conditions is used with YOLOv3 in our demo, which achieved higher accuracy compared to previous works, achieving an accuracy rate of 99% for detecting and identifying objects.

3. The Proposed Method

In this article, a suggested object detection method is presented. In this article, the proposed object detection method is introduced. For object detection and identification, the proposed method is based on YOLOv3, depending on the custom dataset of traffic signs that are divided into four categories: speed limit, yield, mandatory, and others. Objects within each image are identified and defined by YOLOv3. The proposed object detection method steps are depicted in Figure 2.

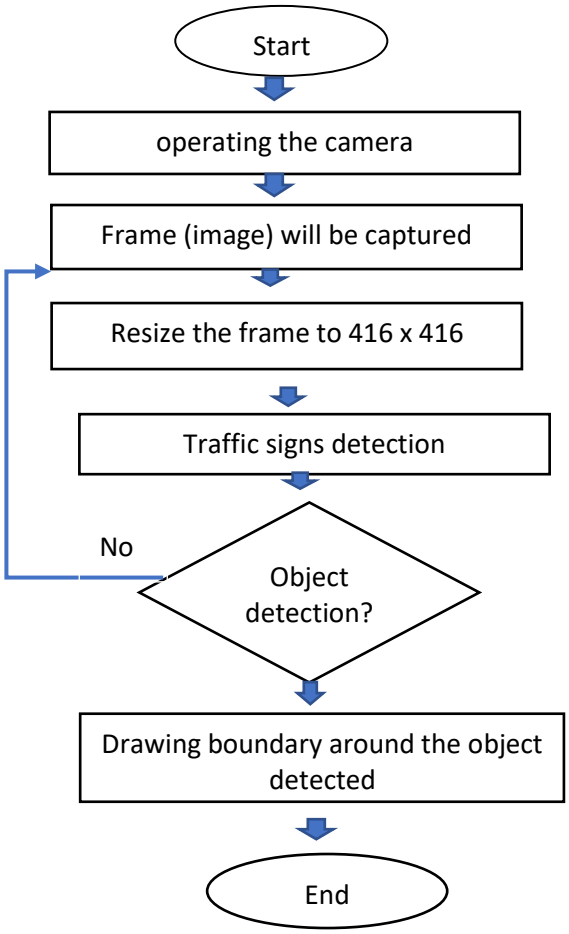

Figure 2: The proposed method flowchart

The proposed method uses the following steps to detect the objects:

- The webcam is operated in the first step in order to take the frames (images) .
- In the second step, each frame is resized to 416 x 416 .
- The third step is to determine whether there are any objects in the image. YOLOv3 will identify objects based on the weight file. The next frame is selected if the current frame is empty of objects. The weight file is produced during the training operation .
- In the fourth step, when YOLOv3 detects an object, it will create a boundary box around it.

The algorithm for the proposed object detection is shown in Algorithm (1).

Algorithm (1): The proposed object detection

Input: Video sequence.
Output: Object surrounded by boundary box.
Process:
Step 1: Open camera and frame extraction.
Step 2: Resize frames to 416x416.
Step 3: IF (object is detected)

YOLOv3 identify the objects and surround by a bounding box.
Else
The next frame is selected.
End IF
Step5: End.

3.1. The Using of You Only Look Once (YOLOv3)

CNN and an algorithm for handling network outputs make up YOLOv3, which is a real-time, multiple-object, and quick method. It performs better than the other algorithms due to its high processing capacity. In particular, a single CNN is applied by YOLOv3 to the entire image, which is then divided into an S × S grid. The bounding boxes are predicted, and the likelihoods for these boxes are discovered. The Yolov3 has 106 layers in total. At three different scales, the objects are detectable (large, medium, and small). Figure 3 presents the stages of YOLOv3.

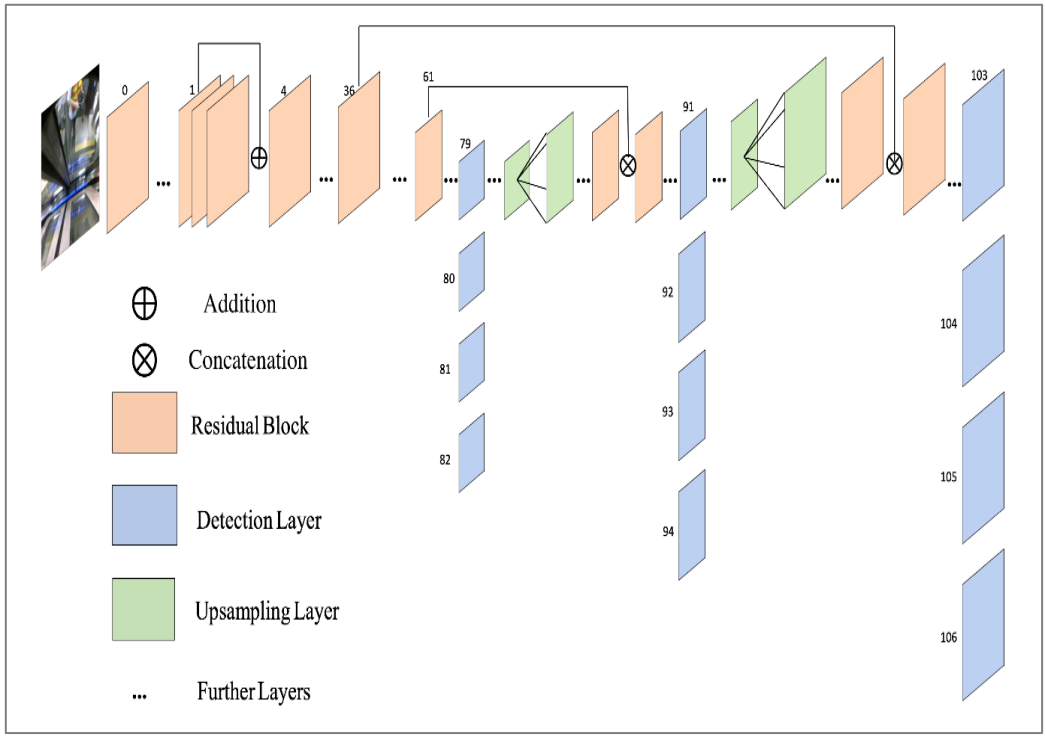

Figure 3: The Yolov3 structure

The following is how the YOLOv3 algorithm operates:

The YOLOv3 first takes a frame from the webcam, resizes it to 416 x 416, and examines it to look for any objects. The image is then divided into several grids, each of which is S x S in size, and each grid may contain one or more objects. Within each grid, a bounding box must

surround these objects. As shown in Figure 4, there could be a number of boundary boxes for every grid.

Figure 4: Anchor Boxes

- There are five values for each bounding box prediction: x, y, w, h, and confidence. The center of the box is depicted as follows: (x, y). W stands for the box's width, while h stands for its height. The range of x, y, w, and h values is zero or one. Although there are three class probabilities for each cell, only one is predicted for each cell. The last prediction has the form S * S * (B × 5 + C). B is the number of bounding boxes a cell on the feature map can predict, and C is the number of classes. Figure 5 illustrates that just one object can be found in each grid cell.
- Depending on the predicted bounding boxes versus the probability of C classes, a confidence score (40% or higher) is provided. No projections are made for the predictions with invalid confidence scores.

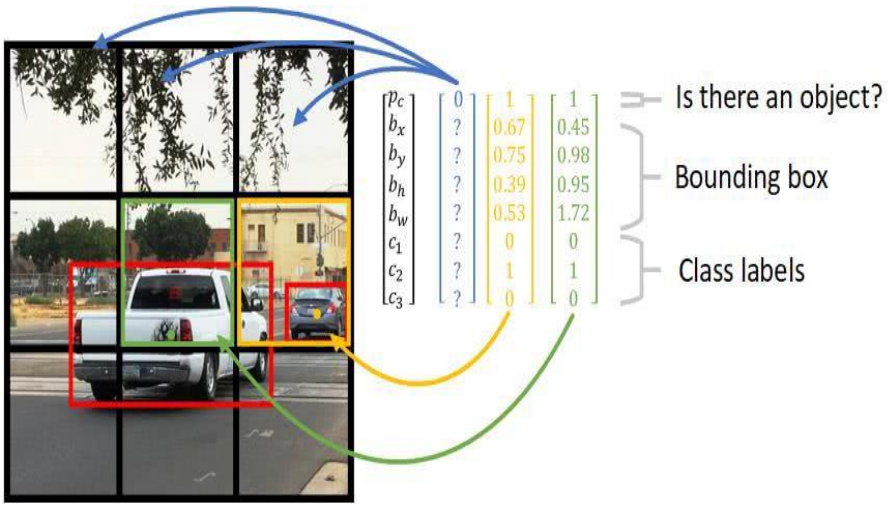

Figure 5: Anchor Boxes

- Each grid cell can only recognize one object. As a result, YOLOv3 makes use of an anchor region to find a variety of things. Take a look at the image in Figure 4. In this picture, the midpoints of the person and the vehicle are inside the same grid cell. Therefore, an anchor frame was employed. The two anchor boxes for these items are indicated by the purple grid cells. One image can be used to detect numerous objects using any number of anchor boxes. There are two anchor circles in this image.
- The grid containing the object's center point is chosen if the same object appears in two or more grid cells. Non-Max Suppression (NMS) and Intersection over Union (IoU) are two

strategies to resolve the issue of multiple bounding boxes generated around the objects. When the intersection over union value is equal to or higher than a threshold value, the prediction made using the IoU method is accurate. The threshold value is the lowest similarity ratio between the expected bounding box and the real bounding box in the proposed method. According to the experiments, the threshold value was 0.90. As the threshold value is raised, the accuracy will rise. The NMS takes the boxes with a good (high) probability and suppresses the boxes with a large IoU. This procedure is followed until the box is chosen, at which point it serves as the object's bounding box; see Figure 6.

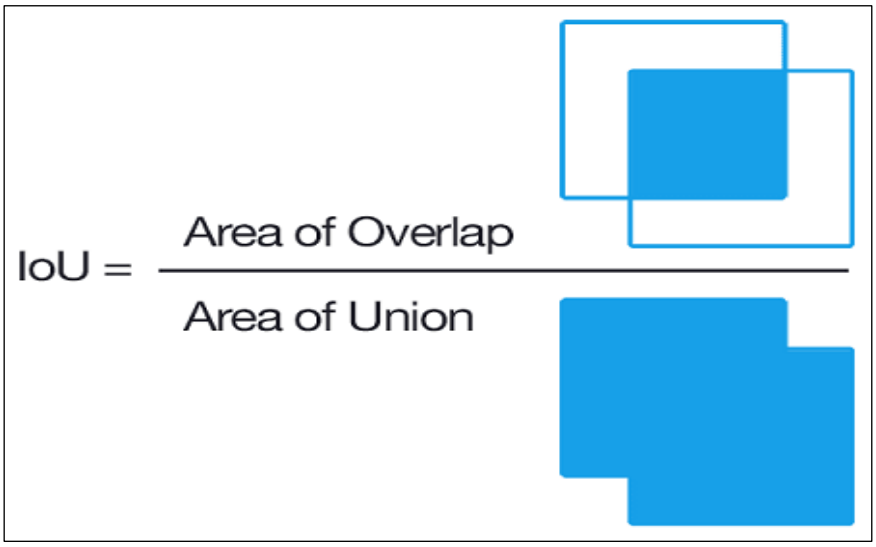

Figure 6: The intersection over union (IoU)

Each object detected in the image has a bounding box after applying the NMS method. The five values for all bounding boxes are x, y, w, h, and confidence. The algorithm for the proposed traffic sign prediction is shown in Algorithm (2).

Algorithm (2): yolov3 algorithm to detect traffic signs.

Input: MP4 Video.
Output: Traffic signs detected.
Process:
Step 1: Open camera.
Step 2: Single frame extraction.
Step 3: Scale frame to 416 x 416.
Step 4: The convolutional neural network is used to detect objects.
Step 5: Outputs' filtration (non max suppression).
Step 6: Determining the locations of objects.
Step 7: End.

3.2. The Training Process

On the GPU of Google Colab, the proposed neural network's training method was carried out. The steps of the training process are listed below.

Step One: A collection of high-resolution, various-sized RGB images is collected.

Step Two: Each object in the image was given a label using LabelImg. An application called LabelImg is used to give a label to each item in an image.

Step Three: The prepared dataset is prepared for neural network training. 250 color images make up the dataset, which was divided into two groups. 90% of the total images make up the

first group, which serves as training images, and the remaining 10%, or the testing images, make up the second group. On Google Colab, the training process was carried out, and it took about four hours. The neural network underwent 8,000 iterations of training.

Step Four: As the final result of the training process, the weights file is created.

The figure below presents the flowchart of the training steps.

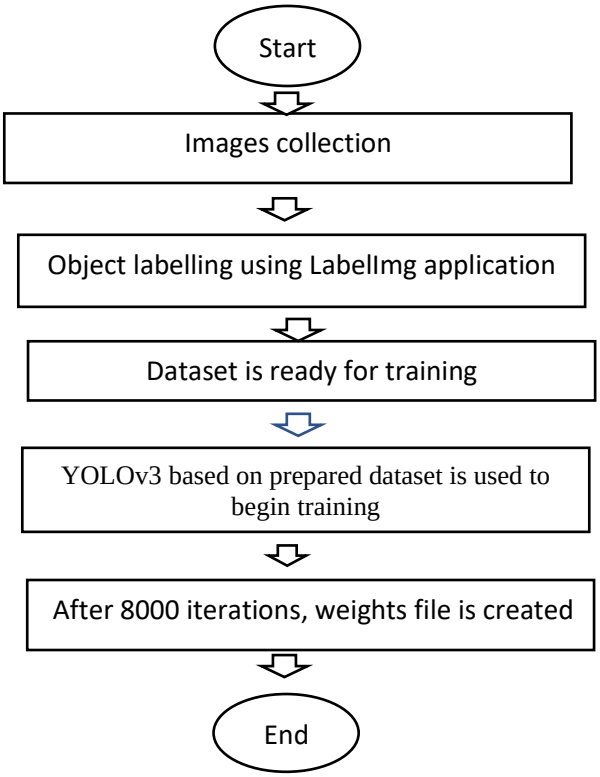

Figure 7: Training process in the proposed method

4. Results and Analysis

YOLOv3 is a fast and accurate algorithm used to achieve good accuracy and high speed. The proposed method can be applied to the traffic sign and can detect multiple objects. In addition, it can detect objects even if the distance between them and the webcam is greater than ten meters. Python version 10 was used to implement the code on a HP laptop with an Intel Core i7 and 8565U clocked at 1.99 GHz. The detection time on this computer was 1.5 seconds. The method can detect four traffic sign categories. Performance was evaluated on the basis of intersection over Union, the mean average precision (mAP), recall (R), precision (P), and F-score (F) measures. These measures can be defined as follows [21] [22]:

True positive (TP) is the number of correctly detected objects. False positive (FP) is the number of incorrect detections. False negative (FN) is the number of missed detections.

- **Precision**

Precision measures how accurate your predictions are. It is calculated as:

The number of true positives (TP) divided by the sum of true positives (TP) and false positives (FP), as given in Eq. (1),

$$precision = \frac{TP}{TP+FN} \quad (1)$$

The precision value obtained was 1.00.

- **Recall**

Recall is used to calculate the true predictions from all correctly predicted data. It is calculated as the number of true positives (TP) divided by the sum of true positives (TP) and false negatives (FN), as illustrated in Eq. (2).

$$\text{Recall} = \frac{TP}{\text{TP+FN}} \quad (2)$$

The recall value obtained was 1.00.

- **F1-score**

The F1-score is the HM (harmonic mean) of precision and recall. The value of the F1-score obtained was 1.00.

- **Average Intersection over Union (IoU)**

The area of overlapping (intersection) is divided by the area of union between the ground truth bounding box and the detection bounding box for a certain threshold. Eq. (3) explains the average IoU.

$$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}} \times 100\% \quad (3)$$

The average IoU for the proposed system was 87.42%.

- **Mean Average Precision (mAP)**

The mAP is the average of the average precision (AP) calculated for all the classes, as indicated in Eq. (4).

$$\text{mAP} = \frac{\text{sum of AP for the total classes}}{\text{no.of total classes}} \times 100\% \quad (4)$$

In particular, the mAP for the proposed system was 100%.

The mean average precision (mAP), which was used to evaluate the performance of the proposed system, was 100%. Figure 8 explains the loss and mAP for the proposed method.
In addition to these metrics, the proposed method's performance was assessed using intersection over union, F1-score, recall, and precision. The performance metrics obtained from the proposed method are declared in Table 1 and Figure 9.

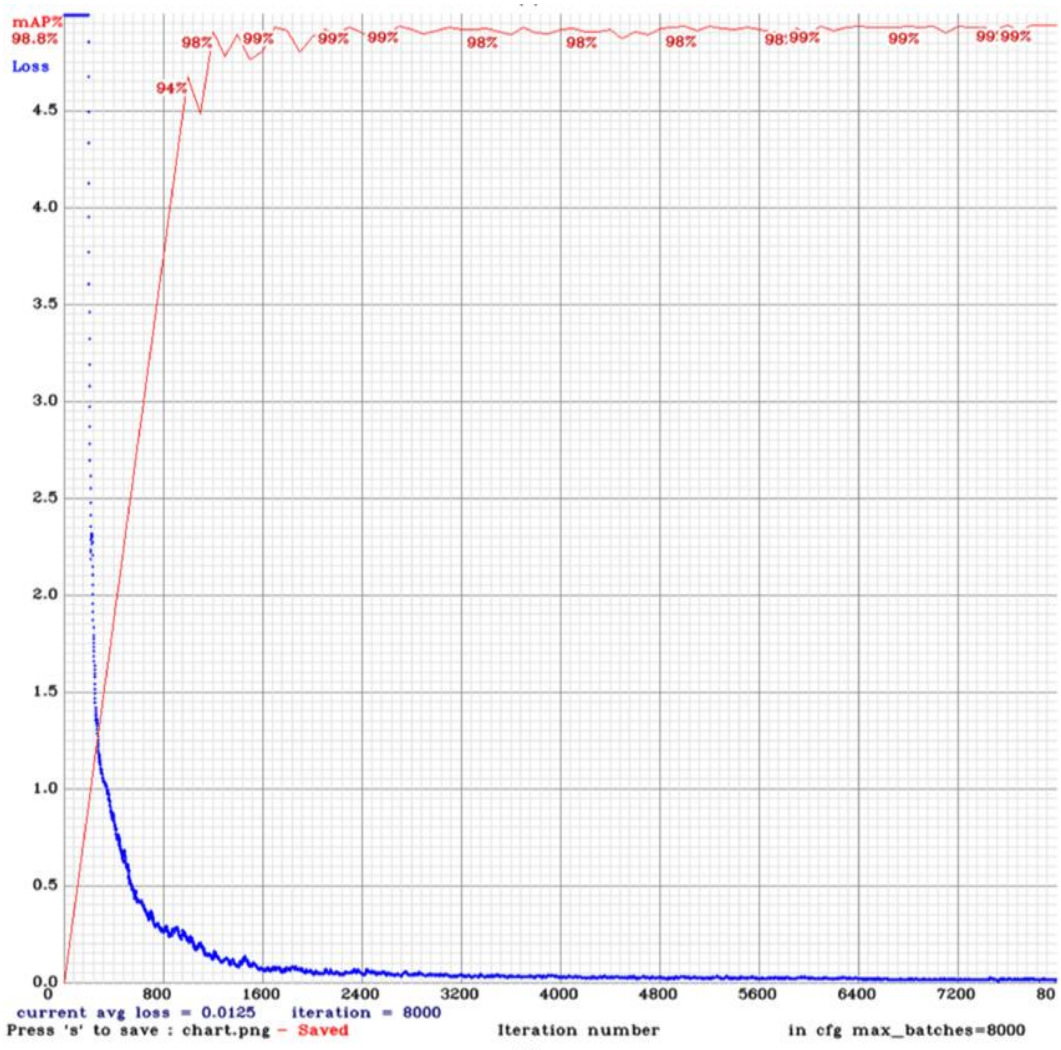

Figure 8: mAP and Loss.

Class_id = 0, name = speed limit, ap = 100.00% (TP = 23, FP = 0)

Class_id = 1, name = yield, ap = 100.00% (TP = 27, FP = 0)

Class_id = 2, name = mandatory, ap = 100.00% (TP = 19, FP = 0)

Class_id = 3, name = others, ap = 100.00% (TP = 15, FP = 0)

for conf_thresh 0.25%, precision = 100.00, recall =100.00, F1-score = 100.00

for conf_thresh 0.25%, TP = 28, FP= 0, FN = 0, average IoU= 88.12%

IoU threshold = 50 % used-Area-Under-Curve for each unique Recall mean average precision (mAP@0.50) =1.000000, or 100.00%

Figure 9: Performance metrics obtained

- **The Confusion Matrix**

The confusion matrix is a summary that gives the results of the prediction on the classification problem. The numbers of correct and incorrect predictions are summarized with counted values and broken down class by class. The confusion matrix shows how your model is confused when it makes predictions. It gives us insight not only into the errors being made by a classifier but also, more importantly, into the types of errors that are being made, as shown in Figure 10.

		Actual traffic sign			
		Speed limit	yield	mandatory	other
Prediction traffic sign	Speed limit	23	0	0	0
	yield	0	27	0	0
	mandatory	0	0	19	0
	other	0	0	0	15

Figure 10: The Confusion Matrix of the Proposed Method

As shown in Table 1, the classification metrics for all tests are classified with accuracy for each type.

Table 1: Classification metric for the yolo model

Name Of Class	Accuracy	Precision	Recall	IoU	MAP
limit_speed	1.00	1.00	1.00		
yield	1.00	1.00	1.00		
mandatory	1.00	1.00	1.00	85%	50%
other	1.00	1.00	1.00		
total accuracy	100%	100%	100%		

A comparison between the proposed method and other existing methods was performed to reinforce the capability of the proposed method. The comparison was made with methods presented in [14], [15], [16], [17], [18], and [19], as shown in Table 2.

Table 2: A comparison between the proposed method and previous works

Paper	Method	Accuracy	Detection time
R. Parmar et al. 2017 [14]	Detecting diabetic retinopathy from retinal images using CUDA deep neural network	85 %	Not Reported
Aleksa Corovi, et al. 2018 [15]	YOLOv3 based on Berkley Deep Drive dataset	63 %	Not Reported
Geetha Neravati et al. 2022 [16]	YOLOv3 based on custom dataset	98 %	8 sec
Azher Atallah et al. 2020 [17]	CNN based on custom dataset	96 %	5 sec
Sunit Vaidya, et al. 2020 [18]	YOLOv3 based coco dataset	89 %	2 sec
A. S. Mahmoud et al. 2020 [19]	Mask RCNN and NWPU-VHR-10 dataset are used	95 %	7.1 sec
Our proposed system	Yolov3 based on custom dataset	99 %	2 sec

5. Conclusion

Many algorithms are used for object detection and recognition, such as YOLO, CNN, Fast R-CNN, and R-CNN. For example, in a self-driving car, when it is moving, it needs a very fast algorithm that is commensurate with the real time to detect and classify traffic lights. In the proposed method, YOLOv3 was used because it is an accurate and fast method that can detect and locate a traffic light in real time. The precision of the algorithm improved with training on more diverse datasets that cover different weather and lighting conditions. The accuracy achieved by the proposed system is 99%. The detection time on the computer was two seconds. In the future, the system can be applied to identify the residence plate for cars to release traffic violations, as well as identify potholes in the road, and it can also be used to make smart glass for the blind.

6. References

[1] V. Kharchenko and I. Chyrka, "Detection of Airplanes on the Ground Using YOLO Neural Network," *Int. Conf. Math. Methods Electromagn. Theory, MMET*, vol. 2018-July, pp. 294–297, 2018. Doi: 10.1109/MMET.2018.8460392.

[2] A. E. Hussain Ali, N., Abdulmunem, M. E., Ali, "Learning Evolution: a Survey," *Iraqi Journal of Science* , vol. 62, no. 12, pp. 4978–4987, 2021.

[3] A. E. Hussain Ali, N., Abdulmunem, M. E., Ali, "Constructed model for micro-content recognition in lip reading based deep learning," *Bull. Electr. Eng. Informatics.*, vol. 10, no. 5, pp. 2557–2565, 2021.

[4] S. N. Srivatsa, "Object Detection using Deep Learning with OpenCV and Python," *Int. Res. J. Eng. Technol.*, pp. 227–230, 2021, [Online]. Available: www.irjet.net

[5] S. Geethapriya, N. Duraimurugan, and S. P. Chokkalingam, "Real time object detection with yolo," *Int. J. Eng. Adv. Technol.*, vol. 8, no. 3 Special Issue, pp. 578–581, 2019.

[6] A. Corovic, V. Ilic, S. Duric, M. Marijan, and B. Pavkovic, "The Real-Time Detection of Traffic Participants Using YOLO Algorithm," *2018 26th Telecommun. Forum, TELFOR 2018 - Proc.*, no. 1, pp. 1–4, 2018. Doi: 10.1109/TELFOR.2018.8611986.

[7] M. S. H. A.-T. Arwa Sahib Abd-Alzhra, "Image Compression Using Deep Learning: Methods and Techniques," *Iraqi Journal of Science*, vol. 63, no. 3, pp. 1299–1312, Mar. 2022.

[8] S. J. Shahbaz, A. A. D. Al-Zuky, and F. E. M. Al-Obaidi, "Real-Night-time Road Sign Detection by the Use of Cascade Object Detector", *Iraqi Journal of Science*, vol. 64, no. 6, pp. 4064–4075, Jun. 2023.

[9] K. Potdar, C. D. Pai, and S. Akolkar, "A Convolutional Neural Network based Live Object Recognition System as Blind Aid," 2018, [Online]. Available: http://arxiv.org/abs/1811.10399

[10] Q. Zou, H. Jiang, Q. Dai, Y. Yue, L. Chen, and Q. Wang, "Robust lane detection from continuous driving scenes using deep neural networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 41–54, 2020. Doi: 10.1109/TVT.2019.2949603.

[11] T. S. Gunawan *et al.*, "Development of video-based emotion recognition using deep learning with Google Colab," *Telkomnika (Telecommunication Comput. Electron. Control.*, vol. 18, no. 5, pp. 2463–2471, 2020. Doi: 10.12928/TELKOMNIKA.v18i5.16717.

[12] N. Mittal, A. Vaidya, and A. Shreya Kapoor, "Object Detection and Classification Using Yolo," *Int. J. Sci. Res. Eng. Trends*, vol. 5, no. 2, 2019.

[13] A. Abdurrasyid, I. Indrianto, and R. Arianto, "Detection of immovable objects on visually impaired people walking aids," *Telkomnika (Telecommunication Comput. Electron. Control.*, vol. 17, no. 2, pp. 580–585, 2019. Doi: 10.12928/TELKOMNIKA.V17I2.9933.

[14] R. Parmar and R. Lakshmanan, "Detecting diabetic retinopathy from retinal images using CUDA deep neural network," *Int. J. Intell. Eng. Syst.*, vol. 10, no. 4, pp. 284–292, 2017. Doi: 10.22266/ijies2017.0831.30.

[15] A. Corovic, V. Ilic, S. Duric, M. Marijan, and B. Pavkovic, "The Real-Time Detection of Traffic Participants Using YOLO Algorithm," 2018 26th Telecommun. Forum, TELFOR 2018 - Proc., 2018. Doi: 10.1109/TELFOR.2018.8611986.

[16] O. Masurekar, O. Jadhav, P. Kulkarni, and S. Patil, "Real Time Object Detection Using YOLOv3," *Int. Res. J. Eng. Technol.*, vol. 07, no. 03, pp. 3764–3768, 2020.

[17] A. A. Fahad, H. J. Hassan, and S. H. Abdullah, "Deep Learning-based Deaf & Mute Gesture Translation System," I*nternational Journal of Science and Research (IJSR)*, vol. 9, no.5 , pp. 288–292, May 2020. Doi: 10.21275/SR20503031800.

[18] S. Vaidya, N. Shah, N. Shah, and R. Shankarmani, "Real-Time Object Detection for Visually Challenged People," *Proc. Int. Conf. Intell. Comput. Control Syst. ICICCS 2020*, no. Iciccs, pp. 311–316, 2020. Doi: 10.1109/ICICCS48265.2020.9121085.

[19] A. S. Mahmoud, S. A. Mohamed, R. A. El-Khoribi, and H. M. AbdelSalam, "Object detection using adaptive mask RCNN in optical remote sensing images," *Int. J. Intell. Eng. Syst.*, vol. 13, no. 1, pp. 65–76, 2020. Doi: 10.22266/ijies2020.0229.07.

[21] S. S. Hussein, S. S. Altyar, and I. A. Abdulmunem, "Improve The Fully Convolutional Network Accuracy by Levelset and The Deep Prior Method," *Iraqi Journal of Science*, vol. 64, no. 5, pp. 2575–2588, May 2023.

[22] E. Beauxis-Aussalet and L. Hardman, "Simplifying the visualization of confusion matrix," *Belgian/Netherlands Artif. Intell. Conf.*, pp. 133–134, 2014.