Automatika

Journal for Control, Measurement, Electronics, Computing and Communications

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/taut20

Video frame feeding approach for validating the performance of an object detection model in real-world conditions

Keerthi Jayan & B. Muruganantham

To cite this article: Keerthi Jayan & B. Muruganantham (2024) Video frame feeding approach for validating the performance of an object detection model in real-world conditions, Automatika, 65:2, 627-640, DOI: 10.1080/00051144.2024.2314928

To link to this article: https://doi.org/10.1080/00051144.2024.2314928

AUTOMATIKA
2024, VOL. 65, NO. 2, 627–640
https://doi.org/10.1080/00051144.2024.2314928

Video frame feeding approach for validating the performance of an object detection model in real-world conditions

Keerthi Jayan and B. Muruganantham

Department of Computing Technologies, School of Computing, SRM Institute of Science and Technology, Kattankulathur, Tamil Nadu, India

ABSTRACT
The challenge of evaluating deep learning-based object detection models in complex traffic scenarios, characterized by changing weather and lighting conditions, is addressed in this study. Real-world testing proves time and cost-intensive, leading to the proposal of a Video Frame Feeding (VFF) approach as a solution. The proposed Video Frame Feeding approach acts as a bridge between object detection models and simulated environments, enabling the generation of realistic scenarios. Leveraging the CarMaker (CM) tool to simulate realistic scenarios, the framework utilizes a virtual camera to capture the simulated environment and feed video frames to an object identification model. The VFF algorithm, with automated validation using simulated ground truth data, enhances detection accuracy to over 95% at 30 frames per second within 130 meters. Employing the You Only Look Once (YOLO) version 4 and the German Traffic Sign Recognition Benchmark dataset, the study assesses a traffic signboard identification model across various climatic conditions. Notably, the VFF algorithm improves accuracy by 2% to 5% in challenging scenarios like foggy days and nights. This innovative approach not only identifies object detection issues efficiently but also offers a versatile solution applicable to any object detection model, promising improved dataset quality and robustness for enhanced model performance.

ARTICLE HISTORY
Received 16 December 2023
Accepted 31 January 2024

KEYWORDS
Traffic sign detection; YOLOv4 CSPDarkNet53; Video Frame Feeding (VFF); CarMaker simulations; Climatic variations

1. Introduction

Ensuring road safety is a critical component of modern vehicular systems, particularly with the increasing integration of computer vision systems in vehicles. These systems often have a virtual camera mounted behind the Inside Rear View Mirror (IRVM), play a crucial role in recognizing and interpreting traffic signs, thereby aiding in safe and informed driving. The efficiency of such systems hinges on the robustness of the underlying object detection algorithms, which must accurately identify, track, and label objects in real-time from video feeds [1]. Despite advancements in this domain, existing systems predominantly rely on statistical-based evaluation methods, utilizing Precision-Recall Curves (PR curves) to assess the performance of object identification models [2]. While these methods provide a baseline for evaluation, they exhibit limitations, particularly in real-world scenarios characterized by dynamic and adverse environmental conditions [3]. Weather phenomena such as rain, fog, and nighttime conditions can significantly impede the model's ability to accurately detect and label traffic signs, leading to reduced precision and reliability. Furthermore, the pre-processing of video feeds, aimed at enhancing visibility, has shown to be insufficient in addressing these challenges, often resulting in a low detection rate and limited robustness in rapidly changing environments. This highlights a critical gap in the current evaluation and validation methodologies, necessitating a more comprehensive and versatile approach to testing object detection models across a spectrum of environmental conditions.

This paper introduces a novel traffic sign detection method, leveraging the capabilities of CM, an advanced environmental simulation tool used for vehicle dynamics and Advanced Driver Assistance Systems (ADAS) [4–6]. Through this tool, we generate photorealistic simulations that encompass a variety of road networks, static and moving objects, and an array of traffic entities. Utilizing IPG Movie for visualization, and a virtual camera configured within CM, we extract video feeds of the simulated environment under various manually adjusted conditions. Central to our approach is the VFF algorithm, a unique automation algorithm designed to identify regions where the object detection model falters. This algorithm facilitates the use of video frames reconstructed from traffic sign data streams, previously trained with the GTSRB dataset [7,8] on the YOLOv4 CSPDarknet53 framework [9,10]. Accompanied by a pre-processing phase incorporating Non-Local Means (NL-Means) estimation [11], and image enhancement techniques for contrast and brightness adjustment, our method ensures a comprehensive and nuanced validation of object detection models. Our study specifically employs YOLOv4 for object detection, focusing

CONTACT Keerthi Jayan kj4134@srmist.edu.in Department of Computing Technologies, School of Computing, SRM Institute of Science and Technology, Kattankulathur, Chennai, Tamil Nadu, 603203, India

on traffic sign detection as a practical application. We assert that by retraining the model with conditions that initially led to failure, we can significantly augment the model's accuracy and reliability, ultimately contributing to safer driving experiences and more robust computer vision systems in vehicles.

Key Features of the VFF Algorithm:

- The VFF algorithm efficiently processes video frames from a simulation environment. These frames are generated based on traffic sign embedded data acquired from pre-trained with the GTSRB dataset using the YOLOv4 CSPDarknet53 framework.
- It incorporates a pre-processing step using NL-Means estimation to smooth irregular object boundaries or edges. This step uses pixel similarity for refinement.
- The visual quality of the video frames is improved by adjusting their contrast and brightness.
- This algorithm is cost-effective, allowing for the simulation of various real-time environmental conditions. These conditions, challenging to capture in reality, can be easily simulated, offering broader perspectives of target objects.
- The study confirmed that around 95% accuracy in traffic signboard detection is achieved. Specifically, an improvement of 2–5% was reached for foggy day, cloudy, foggy night, and nighttime weather conditions from the data observed before applying the proposed VFF algorithm.

The rest of this paper is organized as follows: Section 2 discusses related works, which include a recent study on this topic. Section 3 discusses the methodology, which includes three phases of the model testing approach (i) simulation of video stream using CM Tool, (ii) Conversion of video frames (generation, pre-processing, and contrast and brightness enhancement), and (iii) Validation of object detection model using proposed VFF algorithm. Section 4 presents the results and analysis to demonstrate the efficiency of the proposed method. Section 5 briefly describes the entire work and suggests possible future guidelines.

2. Related works

This section provides an overview of recent articles on the performance evaluation of deep learning-based object detection models in various weather conditions, with a specific focus on traffic signboard detection. It clearly elaborates on the different weather conditions used for validation, as well as the datasets, which include images and video clips – adapted for each paper under discussion.

2.1. Deep learning-based object detection models

The exploration of deep learning-based object detection models has garnered significant attention, addressing various challenges posed by environmental conditions and specific use cases. Sharma et al [12] employed YOLOv5 to recognize entities like cars, traffic lights, and pedestrians, showcasing its robustness in both rainy and normal weather conditions. Al-Haija et al [13] introduced a powerful detection system leveraging transfer learning and Nvidia GPU, with a comparative study across three deep learning models, demonstrating superior performance in diverse weather conditions. Humayum et al [14] utilized the CSPDarkNet53 architecture, aiming at vehicle recognition under low illumination and blurred visibility, resulting in enhanced performance across various challenging conditions. The implementation of a pedestrian detection algorithm by Liu et al [15] demonstrated effective pedestrian identification during rainy conditions, addressing occlusion problems and eliminating rain streaks. Rothmeier and Huber [16] evaluated state-of-the-art object detectors in both normal and foggy conditions, creating dynamic test scenarios to analyze performance degradation under fog. Hnewa and Radha [17] analyzed the efficiency and limitations of de-raining techniques for object detection, providing insights into their impact on performance. Hasirlioglu and Riener [18] compared the performance of object detection algorithms under clear and adverse weather conditions, systematically investigating the effects of rainfall on various sensor data.

2.2. Deep learning model for traffic sign detection

Deep learning models have also been tailored specifically for traffic sign detection, aiming to enhance accuracy and adaptability. Azfar et al. [19] focused on training deep learning models for vehicle detection in complex urban traffic conditions, addressing challenges such as dust and wind sway. Jeon et al. [20] developed a model containing three primary detection modules to detect taillights rapidly and precisely in traffic. Chen et al. [21] integrated visibility complementation modules with YOLOv3, striving to enhance vehicle detection systems in low visibility conditions. Zhu et al. [22] employed RetinaNet for traffic sign recognition and detection, improving detection performance through virtual simulation. Ren et al. [23] used the Recurrent Rolling Convolutional (RRC) model to analyze real-time environmental influences on object detection, providing a systematic evaluation of these impacts. Kuo and Lin [24] investigated CNNs for real-time road sign detection, validating their approach with real-world video data. Lei et al. [25] focused on semantic segmentation, training, and testing labelling models pixel by pixel even in snowy conditions. Jia Li and Zengfu Wang [26] explored Deep Neural Networks (DNN) adapted to traffic sign detection through transfer learning, evaluating their performance across various metrics and conditions. Serna et al. [27] utilized Mask R-CNN for the detection and classification of a wide range of traffic signs, demonstrating successful application across multiple sign categories. These studies collectively highlight the advancements and adaptations made in deep learning models to cater specifically to the challenges posed by traffic sign detection.

Table 1 presents a summary of related works, encompassing research focus, methodology, datasets, evaluation metrics, and inferences. From this table, it becomes evident that current validation methods predominantly emphasize statistical evaluations. Such evaluations entail inputting images or recorded videos into an object identification model, with the PR curve employed to assess the model's output accuracy. In real-world situations, numerous edge cases exist where the model may fail to recognize objects due to various environmental conditions. Practically, it is challenging to test a model across all possible environments simultaneously. To acquire specific environmental conditions, one would need to travel to locations with the requisite climate and capture relevant scenarios. This approach is not only time-consuming and labour-intensive but also incurs substantial expenses.

We have identified a gap in the evaluation of real-time performance of object detection models across diverse environmental conditions and scenarios. To bridge this gap, we propose a novel model validation approach. This approach enables the virtual creation of a wide range of real-world scenarios within a simulation environment, facilitating the comprehensive evaluation of object detection models' real-time performance and the identification of all potential edge cases.

3. Proposed method

This section presents our three-fold methodology for object detection model validation: simulation of driving scenarios using CM tool, conversion of these simulations into enhanced video frames, and model validation using the proposed VFF algorithm. Together, these phases ensure a robust evaluation of the model's performance under varied conditions.

3.1. Overview

Figure 1 depicts the performance evaluation of the YOLOv4 CSPDarknet53 model for detecting traffic signs, utilizing the proposed VFF algorithm. The process unfolds as follows: The CM simulation tool creates a photorealistic simulation of the driving environment, encompassing road networks, traffic signs, static objects (such as buildings and trees), and various traffic participants. Key parameters like vehicle model, driving maneuver, and road profile are set in CM to facilitate this simulation, aligning the scenarios with the GTSRB dataset. In these scenarios, a virtual camera, linked to the VFF algorithm, is mounted behind the IRVM below the vehicle's windshield. As the simulation progresses, this virtual camera captures the frontal road view, inclusive of traffic signs, road markings, and other road users. The captured data is then transformed into a data stream, embedded with simulation time and a list of traffic signs from CM, and sent to a predefined port.

The VFF algorithm connects to this port, retrieving the video streams and separating the embedded data from the video content. The video data is utilized to reconstruct frames to a specific image size, which are subsequently fed into a pre-processing model. This model serves to eliminate noise and enhance image quality. Following pre-processing, the frames are input into the YOLOv4 Darknet framework, loaded with the newly trained GTSDM. The GTSDM then performs traffic sign identification, yielding a list of identified objects corresponding to the simulation time. Concurrently, the ground truth data for traffic signboard identification is extracted from the embedded data provided by CM.

The final step involves comparing the object lists generated by GTSDM and the CM ground truth data, specific to the simulation time. This comparison facilitates a thorough evaluation, highlighting instances where the GTSDM may fail to accurately identify traffic signs across varying environmental conditions. Through this process, the proposed methodology not only identifies potential shortcomings in the model but also opens avenues for model enhancement, particularly in challenging scenarios and conditions.

3.2. Simulation of video stream using CM tool

A virtual camera is mounted behind the IRVM placed below the vehicle's windshield and continuously captures the frontal road simulation environment. This includes oncoming traffic signs, road markings, buildings, and other road users, which are displayed in IPG Movie. To facilitate this process, all intrinsic and extrinsic parameters of the camera must be defined within CM, and the camera configuration parameters must be properly initialized, as detailed in Table 2. The environmental data captured by the virtual camera is then transformed into a data stream. This stream, along with embedded data containing the simulation time and a list of traffic signs from CM, is fed to a pre-defined port.

The provided pseudocode outlines a simplified process for configuring a virtual camera to capture a simulation environment in CM, and it can be divided into three main parts: initialization, configuration file creation, and simulation environment capture.

Initialization: Consider input parameters including the number of camera views (C_{view}), frame size dimensions (W for width and H for height), camera FoV, export format (V_{export}), frame rate (f_r), camera mounting positions (C_{pos_X}, C_{pos_Y}, and C_{pos_Z}), camera rotation angles (ϕ_{x_rot}, ϕ_{y_rot}, and ϕ_{z_rot}), and the

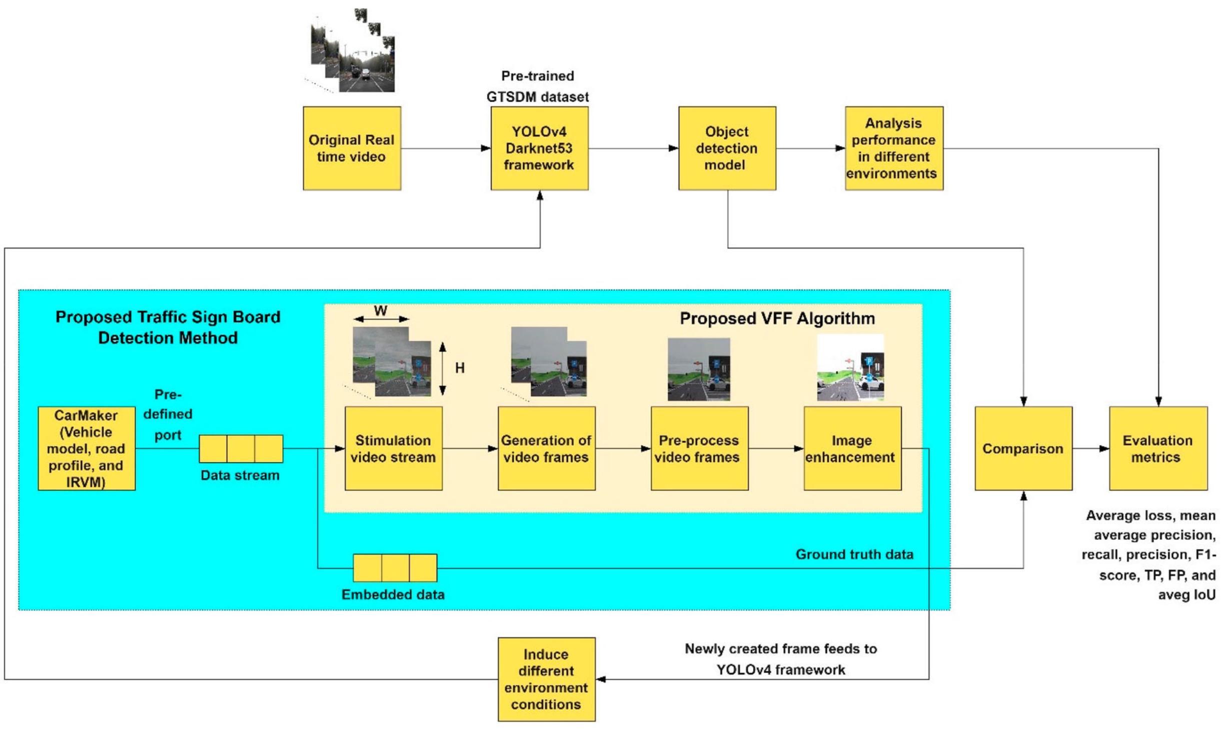
Table 1. Summary of related works.

Author Name	Method	Focus	Dataset	Weather Condition	Evaluation Metrics	Inferences
Sharma et al [12]	Deep learning model (YOLOv5)	Recognizing cars, traffic lights, and pedestrians	Local street level recordings	Rainy and normal	Precision, Recall, F1 Score	YOLOv5 shows robust performance in varied weather conditions.
Al-Haija et al [13]	Deep learning-based detection system (transfer learning, Nvidia GPU)	Comparative analysis of deep learning-based detection systems	DAWN2020, MCWRD2018	Overcast, rainy, snowy, sandy, shiny, sunrise	Accuracy, Precision, Recall, mAP	The Nvidia GPU-accelerated model outperforms others in adverse conditions.
Humayum et al [14]	CSPDarkNet53 (modified SPP layer, fewer batch normalized layers)	Vehicle recognition under various weather conditions, low illumination, and blurry visibility	DAWN dataset (varied image conditions)	Haze, dust, sandstorms, snowy, wet, day and night	mAP, IoU	Improved vehicle recognition in challenging visibility conditions.
Liu et al [15]	Pedestrian detection algorithm (de-raining module)	Pedestrian detection in rainy conditions, rain streak elimination	Urban surveillance datasets	Light, medium, heavy rain	Precision, Recall	Effective rain streak elimination and enhanced pedestrian detection.
Rothmeier and Huber [16]	Object detectors	Evaluate object detectors in normal and foggy conditions	CARISSMA indoor test facility (dense fog, 20 m visibility)	Foggy, normal	IoU, Frame per Second (FPS)	Insights into performance degradation in foggy conditions.
Hnewa and Radha [17]	Deep learning-based adaption and image translation framework	Efficiency and limitations of de-raining techniques for object detection	Recorded images	Clear, rainy	Precision, Recall	Assessment of de-raining techniques' impact on object detection.
Hasirlioglu and Riener [18]	Object detection algorithms	Comparing performance under clear and adverse weather conditions	CARISSMA indoor test facility (rain conditions)	Rainy, clear	Comparative Analysis, Error Rate	Detailed analysis of sensor performance and error rates under rainfall.
Azfar et al [19]	Deep learning models	Training models for vehicle detection in complex urban traffic conditions	Surveillance camera images (urban traffic conditions)	Dust, sway in the wind	Accuracy, Loss	Models exhibit varying degrees of resilience to dust and wind.
Jeon et al [20]	Deep learning model (lane, car, taillight detection)	Precise and rapid taillight detection in traffic	SKKU, Karlsruhe Institute of Technology, Toyota Technological Institute	Various (clear, overcast)	Mean Squared Error, Accuracy	Enhanced detection speed and precision in traffic scenarios.
Chen et al [21]	Deep learning model (YOLOv3) with visibility complementation modules	Enhance vehicle detection in low visibility	Urban and highway driving conditions datasets	Foggy, low-light, twilight	Precision, Recall, F1 Score	Enhanced vehicle detection and reduced false positives in low visibility.
Zhu et al [22]	RetinaNet (traffic sign recognition and detection)	Developing an algorithm for traffic sign recognition and detection	Diverse urban and rural traffic scenarios	Various (clear, cloudy, twilight)	Accuracy, mAP	Successful traffic sign recognition and detection in varied scenarios.
Ren et al [23]	Recurrent rolling convolutional (RRC) model	Analyzing real-time environmental influences on object detection using synthetic images	Synthetic image dataset	Various simulated conditions	Average Precision (AP), Recall	Insights into environmental impacts on object detection accuracy.
Kuo and Lin [24]	Convolutional neural network (CNN)	Investigating CNN for real-time road sign detection	Dataset from two DVRs mounted on a scooter	Various urban and suburban conditions	Accuracy, Precision	Effective road sign detection in real-time scenarios.
Lei et al [25]	ICNet semantic segmentation approach	Semantic labelling of images pixel by pixel	Snowy driving dataset	Snowy	Intersection over Union (IoU), Accuracy	Effective pixel-wise labelling even in snowy conditions.

(*continued*)

Table 1. Continued.

Author Name	Method	Focus	Dataset	Weather Condition	Evaluation Metrics	Inferences
Jia Li and Zengfu Wang [26]	Faster R-CNN, MobileNet, Deep Neural Networks	Traffic sign detection adaptation and performance analysis using transfer learning	GTSRB database	Various (clear, overcast, twilight)	mAP, Memory Allocation, Running Time	Comprehensive analysis and adaptation of traffic sign detection in varied conditions.
Serna et al [27]	Mask R-CNN	Detection and classification of both symbol and text-based traffic signs	GTSDB dataset	Varies (lighting, lane markings)	Accuracy, F1 Score	Successful detection and classification of traffic signs across categories.
Lee and Moon [28]	Vision-based Lane detection algorithm	Lane detection in varied environmental conditions	Dashboard camera videos (US and South Korea)	Varies (lighting, lane markings)	Noise Level, Time Efficiency	Reduction in noise levels and computation time for lane detection.



Original Real time video
Pre-trained GTSDM dataset
YOLOv4 Darknet53 framework
Object detection model
Analysis performance in different environments
Proposed Traffic Sign Board Detection Method
Proposed VFF Algorithm
W
H
CarMaker (Vehicle model, road profile, and IRVM)
Pre-defined port
Data stream
Stimulation video stream
Generation of video frames
Pre-process video frames
Image enhancement
Ground truth data
Embedded data
Comparison
Evaluation metrics
Average loss, mean average precision, recall, precision, F1-score, TP, FP, and aveg IoU
Induce different environment conditions
Newly created frame feeds to YOLOv4 framework

Figure 1. Operational flow of assessing the object detection model with the proposed VFF algorithm. Note: The diverse environmental conditions are generated using the CM tool, aiding in evaluating traffic sign detection performance amidst real-time variations.

port number for CM-VFF algorithm communication (P_{socket}). These parameters are then stored in a dictionary named "params" for easy access and organization.

Configuration File Creation: The pseudocode proceeds to create and configure the virtual camera by writing the parameters to a configuration file, VDS.cfg. The file is opened in write mode, and for each parameter in the params dictionary, a line is written to the file in the format "parameter: value". After writing all the parameters, the configuration file is closed.

Simulation Environment Capture: A connection to CM is established using the specified port number. If the connection is not successful, an error message is logged, and the programme exits. If the connection is successful, the pseudocode enters a loop to continuously capture frames from the CM simulation environment. This loop runs as long as the simulation is running. Each captured frame is then processed.

3.3. *Proposed VFF algorithm*

The VFF algorithm establishes a connection with CM via the same port. It initiates an empty binary string and assigns it to a variable named "data". Subsequently, it calculates the payload size. If the data's length is smaller than the payload size, the VFF algorithm begins to receive video streams from that port. It then segregates the embedded simulation data from the video data

Table 2. Camera configuration in the CM tool.

Parameters	Values
Number of virtual cameras	1
Pixel size	640 × 480
Camera field of view (FoV)	30 degree
Lens type	Direct Lens
Camera output format V_{export}	RGB
Frame rate f_r	30 fps
Mounting position of camera in X-direction C_{pos_X}	3.5 m
Mounting position of camera in Y-direction C_{pos_Y}	0.0 m
Mounting position of camera in Z-direction C_{pos_Z}	1.5 m
Rotation angle X ϕ_{x_rot}	0 degree
Rotation angle Y ϕ_{y_rot}	0 degree
Rotation angle Z ϕ_{z_rot}	0 degree
Distance between camera position and viewing point	0
Distance of first clipping plane	0.1 m
Distance of far clipping plane	500 m
Sensitivity	{(0 1 0),(1 0 0),(0 0 1), and (0 1 0)}
Principle point offset X/Y	0
Sensor noise σ_n	0
Tone mapping	-
SrcHDR16	0

Algorithm 1: Configure VDS in CM

```
Input: Configuration data to create VDS.cfg file
Output: VDS.cfg file
        // Initialize parameters
(1)     params (i) = {$C_{view}, W, H, FoV, V_{export}, f_r, C_{pos\_X}, C_{pos\_Y}, C_{pos\_Z}, \phi_{x\_rot}, \phi_{y\_rot}, \phi_{z\_rot}, P_{socket}$}
        // Create and configure VDS.cfg file
(2)     configFile → OpenFile("CarMakerProjectPath/VDS.cfg", "write");
(3)     For i = 1 to params do
(4)     WriteToFile(configFile, param + ": " " + params[param]);
(5)     End For
(6)     CloseFile(configFile);
        // Establish connection with CM
(7)     connection → EstablishConnection(params[P_socket]);
(8)     If Not connection.Successful:
(9)     Log("Failed to connect to CarMaker on port: " + params[P_socket]);
(10)    Exit;
(11)    End If
        // Capture simulation environment
(12)    While SimulationRunning:
(13)    frame → CaptureFrameFromCarMaker();
(14)    ProcessFrame(frame);
(15)    End While
```

streams based on the payload size. The embedded simulation data includes the number of camera output channels, the actual simulation time, the length of the data, and a list of CM object identifications in bytes.

3.3.1. Generation of video frames

Read the video streams of the simulation environment from the specified port and use the received video stream to regenerate the video frames utilizing the NumPy package. The video data streams provide the frame information in bytes. If the length of the data is less than the image length (frame size; for RGB: height $*$ width $*$ 3), proceed to receive the frame data up to the image length. When the length of the frame data matches the image length, the frame data will contain the information of the first frame in byte format. Utilize the NumPy package to transform the frame data from byte strings to an array, and subsequently recreate the video frame. In instances where the received frame is not in RGB format, employ the OpenCV package to convert the frame data to RGB format. The newly created video frame will encompass simulation environment data from CM, including traffic objects and signboards.

3.3.2. Pre-processing of video frames

Pre-process the video frame to eliminate noise and enhance the frame's quality. Remove noise from the generated frame using the NL-Means denoising approach. The underlying principle of the NL-Means denoising approach is to replace a pixel's colour with an average of the colours of similar pixels. Hence, it scans a large image area to find all pixels closely resembling the pixel targeted for denoising. The similarity is assessed by examining the entire window around each pixel, rather than just the colour alone. To remove noise from a colour image, implement pixel-wise calculations, represented as $x = (x_1, x_2, x_3)$. The filtered image value at position p for a pixel is given in Eq. (1):

$$\widehat{x_i}(p) = \frac{1}{N(p)} \sum_{q \in A} y_i(q) \mathrm{w(p, q)}, \ \forall \ \mathrm{i} = 1,\ 2,\ 3 \qquad (1)$$

Where, $x(p)$ is the filtered image value at p position, y(q) is the unfiltered image value at q point, w(p, q) is the weighting function, and $N(p)$ is a normalizing factor given in Eq. (2):

$$N(p) = \sum_{q \in A} \mathrm{w(p, q)} \qquad (2)$$

The weighting function, based on a normal distribution with a mean $\mu = M(p)$, is given in Eq. (3):

$$\mathrm{w(p, q)} = \mathrm{e}^{-\frac{|\mathrm{M(q)} - \mathrm{M(p)}|^2}{\mathrm{h}^2}} \qquad (3)$$

Where, h is the filtering parameter and $M(p)$ is the local mean value in the region around p in an image, calculated as shown in Eq. (4):

$$M(p) = \frac{1}{|R(p)|} \sum_{i \in R(p)} y(i) \qquad (4)$$

Where, $|R(p)|$ represents the number of pixels in region R, and $R(p) \subseteq A$ is a square region of pixels surrounding p, with dimensions $(2r + 1) \times (2r \times 1)$ pixels. Due to computational constraints, the research zone is confined to a fixed square neighbourhood. For moderate and small values of σ, the window size is set to 21×21. For larger values of σ, the size of the research window increases to 35×35, allowing for the identification of more similar pixels to further minimize noise. In NL-Means, the restoration of each pixel value is achieved by averaging the most similar pixels, with similarity determined from the colour image. Consequently, each channel value for every pixel result from averaging the values of similar pixels.

3.3.3. Enhancement of contrast and brightness

After the initial pre-processing step, the received frames from CM undergo a noise reduction process using the NL-Means de-noising model, resulting in the generation of frames free from noise artifacts. The effectiveness of this noise reduction is quantified using the Peak Signal-to-Noise Ratio (PSNR), which in this case, yields a value of 28.9, indicating a substantial enhancement in image quality. Subsequently, these de-noised frames are subjected to various image enhancement techniques to further refine the visual quality. These techniques include adjustments to brightness, contrast, and sharpness, ensuring that the frames are optimally prepared for the subsequent object detection tasks. Upon completion of the image enhancement process, the frames are then fed into the YOLOv4 CSPDarkNet53 model. This model has been specifically trained for the task of traffic signboard detection and is adept at performing object detection tasks under varied environmental conditions. The darknet helper function is utilized to load the pre-trained model, which then processes the input frames to identify and classify objects within them. The output from the model comprises a list of detected object classes, along with their associated confidence scores and bounding box coordinates, providing a comprehensive overview of the detected objects within each frame. Concurrently, the proposed VFF algorithm receives a list of objects detected by CM, along with the corresponding simulation times. This information is then compared with the output from the YOLOv4 CSPDarkNet53 model to assess the accuracy and reliability of the object detection process. By conducting a comparative analysis between the detected object classes from the YOLOv4 CSPDarkNet53 model and the CM object identification list, extracted from the embedded simulation data, we are able to evaluate the performance of the model across different simulation times. This comparative analysis not only provides valuable insights into the model's proficiency in traffic signboard detection but also highlights potential areas for improvement, ensuring that the model can be further refined and optimized for future applications.

Pseudocode of proposed VFF algorithm for validating traffic signboard detection is given below which designed to process video streams and produce enhanced video frames. Initially, a connection is established with a predefined socket using specified port and host IP parameters. Once connected, the algorithm continuously receives video data packets and processes them in chunks defined by a payload size. For each chunk, the video data is extracted, and if its length matches the expected image size, then it generates the frame data which is further changed into RGB format. Later, this RGB format is converted into a video frame. Subsequent steps involve pre-processing the video frame by de-noising it. Then, the frame undergoes two enhancement processes to improve its contrast and brightness.

Algorithm 2: VFF Algorithm for Validating Traffic Signboard Detection

```
Input: Video streams
Output: Enhanced video frames
Initialization: width, height, port, host_ip, image length (WxHx3)
(1)   Create socket;
(2)   Connect to the socket using port number and host_ip;
(3)   If socket connected then
      // Generation of Video Frames
(4)   Data → "b";
(5)   Payloadsize → "6Q" "8h"; // Define payload size;
(6)   While length of the data < payload size do
(7)   Packet → receive video streams for image length; // Receive the
         video streams as packet
(8)   Data → packet; // Append the data with packet;
(9)   Packet_msg_size → data[:payload_size];
(10)  Cm_vds → split embedded data from payload_size;
(11)  Data → data[payload_size:]; //Filter the video streams based on
         the payload_size;
(12)  Msg_size → calculate size from packed_msg_size;
(13)  While length of the data < img_len do
(14)  Data → append received data from img_len;
(15)  Frame_data → compute frame from msg_size;
(16)  If length of img_data == img_len then:
(17)  Frame → generate the frame data;
(18)  Frame → convert frame data to RGB;
      // Pre-processing of Video Frames
(21)  Frame → frame de-noising;
      // Enhancement of Contrast and Brightness
(22)  Frame → frame contrast enhancement;
(23)  Frame → frame brightness enhancement;
(24)  Detection → object detection model;
(25)  cm_time → get cm time from cm_vds;
(26)  cm_detection→ get cm_detection from cm_vds;
(27)  For time in cm_time do:
(28)  If cm_detection! = detection then
(29)  Object not identified simulation time;
(30)  Else
(31)  Object identification is correct;
(32)  End if
(33)  End for
(34)  End while
(35)  End while
(36)  End if
```

Following these enhancements, an object detection model is applied to the frame to identify potential traffic signs. The detected time from the CM simulation (CM time) is fetched and compared against the object detection output. If there's a mismatch between the CM's detection and the algorithm's detection, it's noted that the object was not identified correctly at that simulation time. Otherwise, the identification is marked as correct. This process is repeated for each video data packet until the end of the stream.

4. Results and discussion

In this section, we describe the evaluation of traffic signboard detection using the YOLOv4 (CSPDarknet53) model, which has been pre-trained with the GTSRB dataset. Observations show that with the modified YOLOv4, traffic signboard detection achieves an accuracy of approximately 96%. This accuracy was determined without considering various weather conditions. To enhance this accuracy without increasing hardware costs or the execution time for training and testing, we utilized the CM simulation tool. This tool was used to create a highway simulation environment in which multiple weather conditions were introduced simultaneously. This setup allows for an evaluation of the YOLOv4 CSPDarkNet53's performance under these conditions. Our analysis focuses on two main aspects: (i) identifying traffic signboards located within 150 meters from the driving vehicle, and (ii) detecting multiple traffic signs situated in the same region but separated by varying distances. All evaluations were conducted under six different weather conditions generated using the CM tool. A comparative analysis was then performed between the ground truth data from CM and the output from the YOLOv4 CSPDarkNet53 model, taking into account the influence of the proposed VFF algorithm for traffic signboard detection.

4.1. GTSRB Dataset

The GTSRB dataset is selected for the proposed research application. It comprises 900 images from 43 distinct traffic signboard classes, divided into four categories: prohibitory, danger, mandatory, and priority. As illustrated in Fig. 2, the GTSRB dataset has been classified for this study. The YOLOv4 darknet53 variant serves as the object detection model for this research. The dataset has been prepared in the YOLOv4 format, which means that the 900 images from the GTSRB dataset needed to be annotated accordingly. Prior to this, the GTSRB dataset underwent preprocessing techniques such as scaling, contrast enhancement, sharpening, and Principal Component Analysis (PCA) [29]. The PCA technique assists in reducing the dataset's dimensionality while ensuring minimal information loss, thereby enhancing its interpretability. An automated annotation algorithm facilitated the conversion of raw data to the YOLO format. Of the original 900 images, 80% were designated for training, and the remaining 20% for testing. The resultant dataset consists of images and their corresponding annotation files. These files detail the object's class, its coordinates, and its dimensions in terms of height and width. The training and testing dataset folders are renamed as OBJ and Test, respectively.

4.2. Experimental setup

For our experimental analysis, we randomly selected four traffic signboards from a total of 43. Based on the chosen environmental parameters, we developed a CM scenario. We then simulated CM at a constant vehicle speed of 72 kmph and evaluated the detection capabilities of the GTSRB model using the proposed VFF algorithm under six different weather conditions. Observations indicate that the model was able to classify the traffic signboard into the selected classes with an accuracy exceeding 95%.

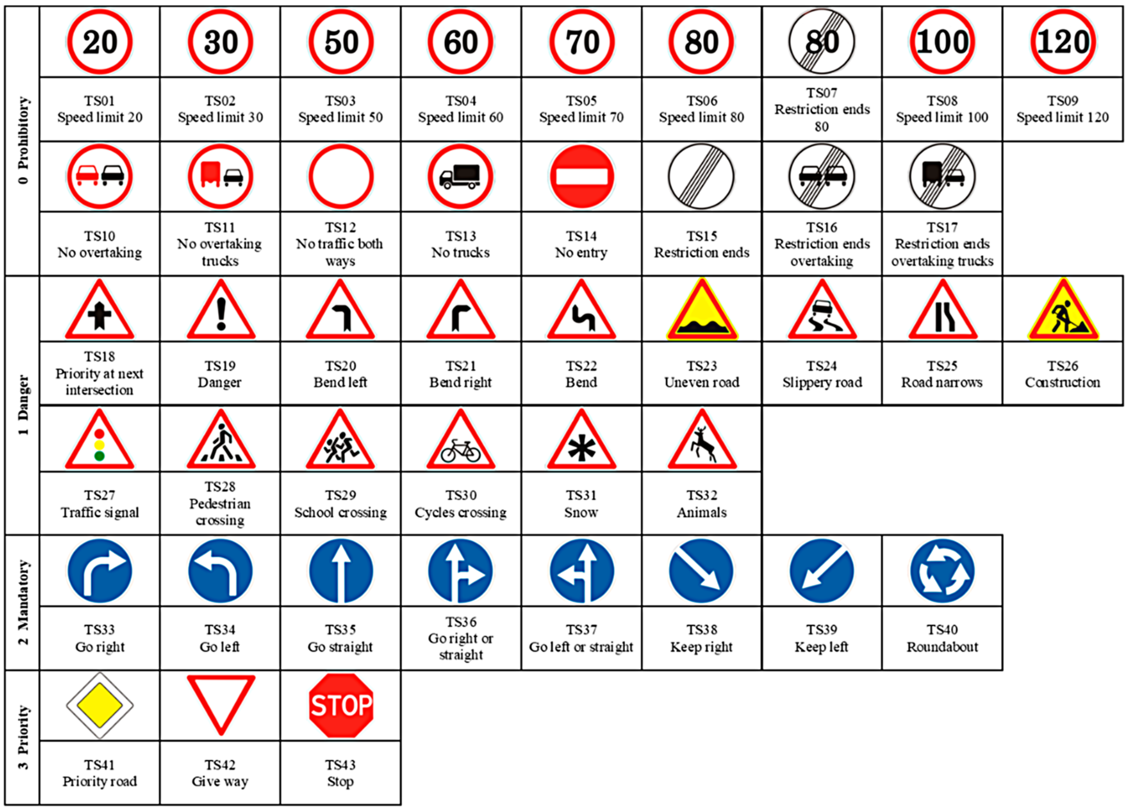
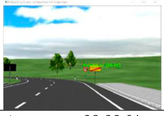
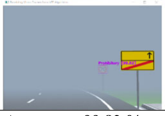
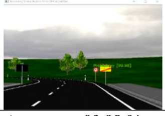
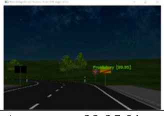
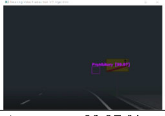
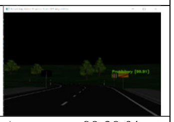
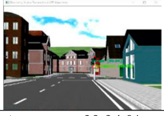
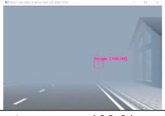
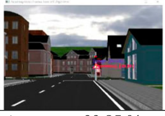
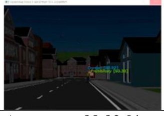
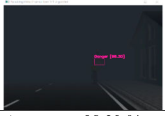
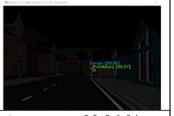
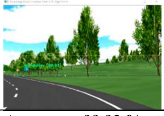
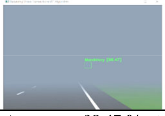
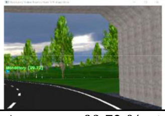
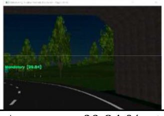
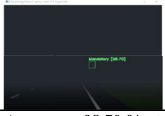
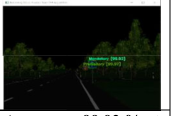
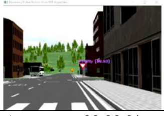
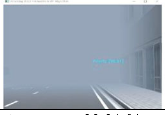
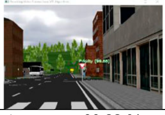
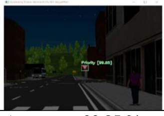
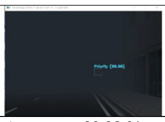
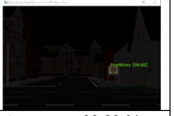


Figure 2. GTSRB dataset and its class description.

Table 3. Results of detection accuracy (%) with random GTSRB detection under six weather conditions.

Class \ Conditions	Day	Foggy Day	Cloudy	Dusk	Foggy Night	Night
Prohibitory						
Observation	Accuracy: 99.99 % at distance of 138 meter	Accuracy: 99.82 % at distance of 12 meter	Accuracy: 99.98 % at distance of 124 meter	Accuracy: 99.95 % at distance of 130 meter	Accuracy: 99.97 % at distance of 35 meter	Accuracy: 99.99 % at distance of 77 meter
Danger						
Observation	Accuracy: 99.94 % at distance of 148 meter	Accuracy: 100 % at distance of 52 meter	Accuracy: 99.85 % at distance of 145 meter	Accuracy: 99.92 % at distance of 150 meter	Accuracy: 98.30 % at distance of 38 meter	Accuracy: 99.96 % at distance of 60 meter
Mandatory						
Observation	Accuracy: 99.93 % at distance of 150 meter	Accuracy: 98.47 % at distance of 55 meter	Accuracy: 99.72 % at distance of >150 meter	Accuracy: 99.84 % at distance of 150 meter	Accuracy: 98.70 % at distance of 30 meter	Accuracy: 99.92 % at distance of 45 meter
Priority						
Observation	Accuracy: 99.93 % at distance of 130 meter	Accuracy: 99.91 % at distance of 90 meter	Accuracy: 99.88 % at distance of 125 meter	Accuracy: 99.85 % at distance of 95 meter	Accuracy: 99.98 % at distance of 60 meter	Accuracy: 99.82 % at distance of 30 meter

4.2.1. Analysis of detection accuracy (%) with random GTSRB detection under six weather conditions

This analysis primarily centres on the detection capabilities of the YOLOv4 DarkNet53 model, especially after being influenced by the proposed VFF algorithm. This model involves furnishing the model with clearer frames, which becomes notably crucial under certain weather conditions. For instance, in foggy day, foggy night, and standard night scenarios, the model required closer proximities to detect with precision. The detailed analyses is summarized per class as follows:

Prohibitory: Achieved near-perfect detection during the day at 138 meters, but on foggy days, the model required a much closer range of just 12 meters. Performance remained high during cloudy conditions and dusk at 124 and 130 meters respectively. Foggy nights and standard nights required distances of 35 and 77 meters for optimal detection.

Danger: Exhibited optimal performance during the day at 148 meters and on foggy days at a reduced 52 meters. Cloudy conditions and dusk demanded 145 and 150 meters respectively. However, on foggy nights, accuracy slightly dipped at 38 meters, but rebounded on standard nights at 60 meters.

Mandatory: Consistently detected during the day and at dusk at 150 meters. On foggy and cloudy days, the model required 55 meters and distances greater than 150 meters respectively. Foggy nights and standard nights needed 30 and 45 meters for accurate detection.

Priority: Demonstrated consistent performance across all conditions. During the day, it detected optimally at 130 and 90 meters on foggy days. Cloudy conditions, dusk, foggy nights, and standard nights required distances of 125, 95, 60, and 30 meters respectively.

In essence, while the YOLOv4 DarkNet53 model showed high accuracy across all classes, its detection range varied significantly based on weather conditions, with foggy scenarios necessitating closer proximities for accurate detection. It is clearly depicted in Table 3.

4.2.2. Analysis of accuracy (%) for traffic signs in close proximity but at different distances

In our study, we selected a traffic junction point at random, which has multiple traffic signboards separated by specific distances. For instance, there's a 15-meter gap between the prohibitory sign (TS10-No Overtaking) and the priority sign (TS43-Stop). Likewise, there's a 25-meter distance between the mandatory sign (TS33-Go right) and the priority sign (TS43-Stop). It's important to note that the danger and priority classes are positioned at the same location but are separated by a minimal distance of 2 meters. Fig. 3 illustrates the model verification in different environmental conditions. We observed variations in accuracy percentages, particularly for the Prohibitory and priority classes during foggy days, foggy nights, and nights. This variation can be attributed to cross-validation conflicts within the classes. For instance, within the prohibitory class, there's a conflict between TS10-no overtaking and TS11-no overtaking trucks. Similarly, within the priority class, we observed variations due to conflicts between TS28-pedestrian crossing and TS29-school crossing, which is clearly depicted in Table 4. Such cross-validations extend the execution time, reducing the detection accuracy. Consequently, more accurate

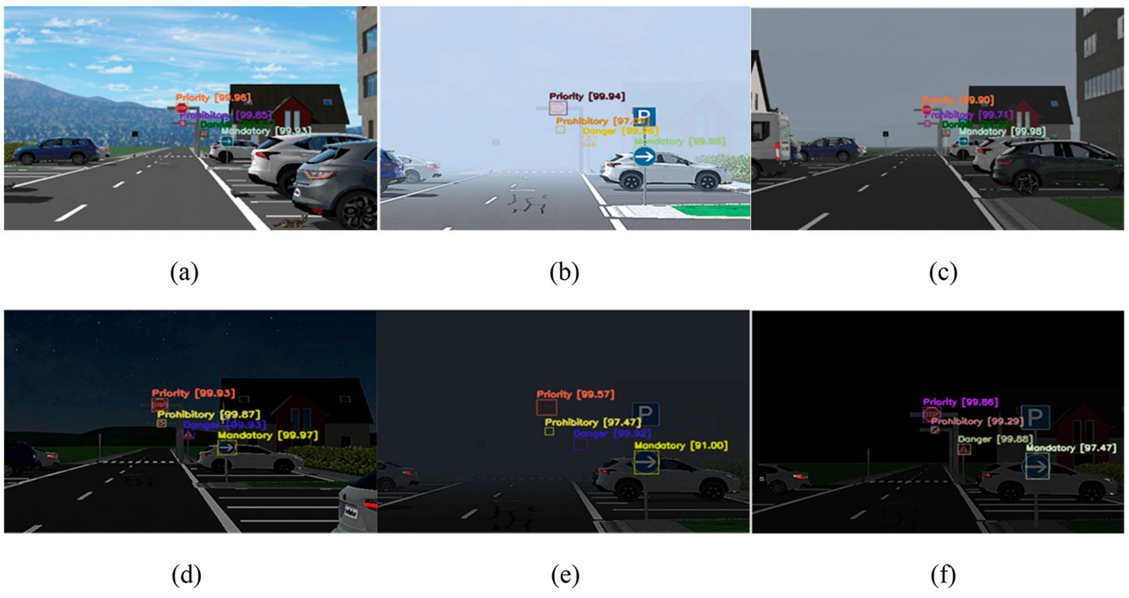


Figure 3. Model verification in different environmental conditions. (**a**) Day, (**b**) Foggy Day, (**c**) Cloudy, (**d**) Dusk, (**e**) Foggy Night, and (**f**) Night.
Note: traffic signboards in close proximity but at different distances.

Table 4. Results of YOLOv4 DarkNet53 model verification in various weather conditions.

Class	Color and shape	Day	Foggy Day	Cloudy	Dusk	Foggy Night	Night
Prohibitory	○	99.85	**97.09**	**97.71**	99.87	**97.47**	99.29
Danger	△	99.87	99.95	99.93	99.93	99.92	99.88
Mandatory	●	99.96	99.94	99.9	99.93	99.57	99.86
Priority	⬣	99.93	99.98	99.98	99.97	**91**	**97.47**

detection is achieved when the vehicle is closer to the target point. For the prohibitory class, the detection accuracy is as follows: during the day it's 99.85%, on foggy days it drops slightly to 97.09%, on cloudy days it's 97.71%, during dusk, it's at a high of 99.87%, on foggy nights it's 97.47%, and during regular nights it's 99.29%. For the danger class, the accuracy remains consistently high across all conditions: 99.87% during the day, 99.95% on foggy days, 99.93% on cloudy days and dusk, 99.92% on foggy nights, and 99.88% during the night. The mandatory class has the following accuracies: 99.96% during the day, 99.94% on foggy days, 99.9% on cloudy days, 99.93% during dusk, 99.57% on foggy nights, and 99.86% during the night. Lastly, for the priority class: the accuracy is 99.93% during the day, 99.98% on foggy days, 99.98% on cloudy days, 99.97% during dusk, it drops to 91% on foggy nights, and recovers to 97.47% during regular nights.

Table 4 reveals that under various weather conditions such as foggy days, cloudy weather, foggy nights, and nighttime, the detection accuracy for prohibitory and priority signs is notably lower compared to other categories. This observation stems from simulation video frames in CM, which were used to test traffic signboard detection under various weather scenarios. The diminished detection accuracy in these conditions is attributed to reduced visibility from greater distances; the system only recognizes the signs when the vehicle is relatively close to them. Consequently, the detection performance falls short of its standard accuracy levels. However, the implementation of the proposed VFF algorithm offers a solution by ensuring

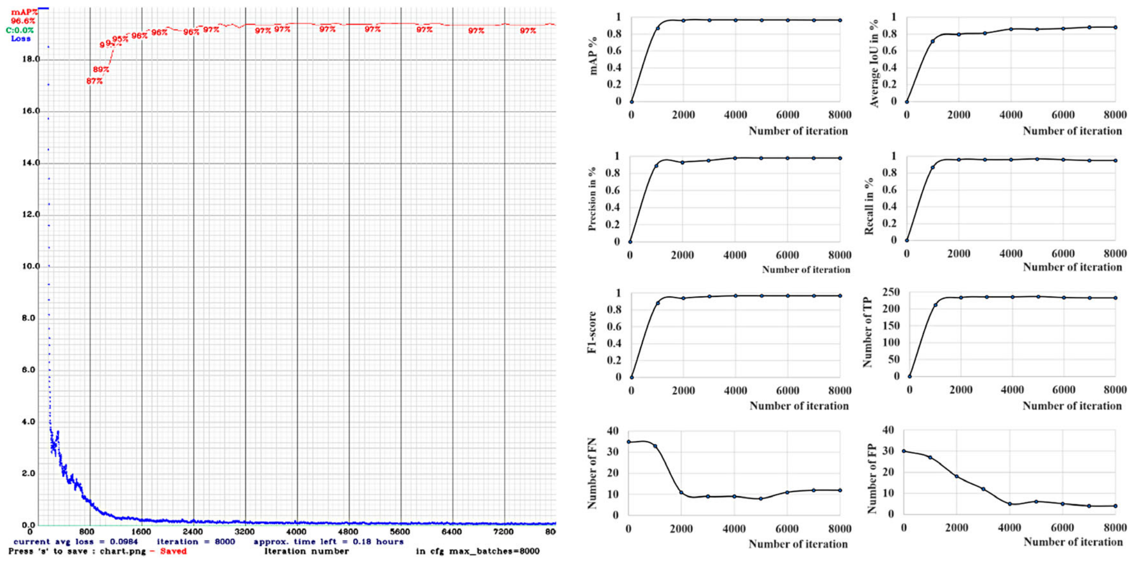


Figure 4. Evaluation chart. (**a**) Progression of average loss and mAP against the number of iterations during model training, and (**b**) Evaluation metrics of the entire 8000 iterations.

high detection accuracy from longer distances relative to the target point, thereby correctly identifying all four types of traffic signs regardless of the weather conditions. An improvement ranging from 2% to 5% in detection accuracy was noted with the VFF algorithm during foggy days (up to 99.24%) and cloudy conditions (up to 99.86%) for the prohibitory class. In the case of the priority class, the accuracy increased to 96.15% on foggy nights and 99.62% at night, after applying the VFF algorithm.

4.3. Evaluation metrics

Figure 4 (a) presents the output chart of average loss and mean Average Precision (mAP) against the number of iterations during model training. In the graph's early stages, the average loss begins at a notably high value, a common trait for models in their initial training phase. However, as the number of iterations increases, the loss shows a marked decline. This trend is emblematic of the model progressively refining its predictions, improving its alignment with the true labels of the training data. Generally, a model that exhibits a lower average loss is indicative of increased precision in its predictions. While the average loss gives insight into the model's evolving predictive capability, the mAP provides a measure of its accuracy over time. The mAP values, as displayed atop the graph, undergo fluctuations initially but seem to stabilize around a commendable 97% in the latter stages. This highlights the model's proficiency in accurately predicting the correct labels as training progresses. Specifically, an average loss can typically range from as low as 0.05 (in scenarios with a small model and a less complex dataset) to as high as 3.0 (when dealing with a larger model trained on intricate datasets). In the context of the traffic signboard detection that this graph represents, the model achieves an average loss of 0.098. This value, coupled with an mAP of 96.6%, signifies a highly precise model. Overall, the graph illustrates a successful training progression. The model's loss decreases significantly, indicating it's making better predictions as it learns, and the mAP values stabilize at a high percentage, showing that the model is achieving high precision in its predictions.

The evaluation matrix illustrates precision, recall, F1 score, True Positive (TP), False Positive (FP), False Negative (FN), average IoU, and mAP for the 8000 iterations depicted in Fig. 4(b). As the iterations progress, FP and FN identifications decrease, while TP increases. This leads to an enhancement in both precision and recall with each iteration. The model is becoming more adept at correctly identifying objects, resulting in fewer FP and FN identifications. The F1 score also shows an increasing trend with each iteration. An average Intersection over Union (IoU) greater than 50% is utilized to calculate the mAP of the model. As observed, the average IoU increases with each iteration, and concurrently, the mAP also rises.

Tables 5 and 6 provides a detailed evaluation of four distinct traffic signboard classes across different environmental conditions. This assessment uses several key metrics, including mAP, Recall, Precision, F1-score, TP, FP, and Average Intersection over Union (Avg. IoU). The data specifically highlights the performance of a newly created pre-processed video, which underwent contrast and brightness enhancement by CM, when fed into the YOLOv4 framework with the proposed VFF algorithm. The table aids in comparing the efficacy of the detection model under various conditions, such as day, foggy day, cloudy, dusk, foggy night, and night.

The comparative analysis distinctly showcases the superiority of the proposed YOLOv4 CSPDarkNet53 model in traffic signboard detection across diverse

Table 5. Evaluation metrics includes prohibitory and danger.

Class	Prohibitory							Danger						
Envi.	mAP	Recall	Precision	F1-score	TP	FP	Avg. IoU	mAP	Recall	Precision	F1-score	TP	FP	Avg. IoU
Day	0.8	0.85	0.76	0.8	220	30	0.7	0.78	0.83	0.75	0.79	215	35	0.69
Foggy day	0.75	0.8	0.71	0.75	205	45	0.67	0.73	0.78	0.7	0.74	200	50	0.66
Cloudy	0.77	0.82	0.73	0.77	210	40	0.68	0.75	0.8	0.72	0.76	205	45	0.67
Dusk	0.78	0.83	0.74	0.78	213	37	0.69	0.76	0.81	0.73	0.77	208	42	0.68
Foggy night	0.72	0.77	0.69	0.73	198	52	0.65	0.7	0.75	0.68	0.71	193	57	0.64
Night	0.7	0.75	0.67	0.71	190	60	0.63	0.68	0.73	0.66	0.69	185	65	0.62

Table 6. Evaluation metrics includes mandatory and priority.

Class	Mandatory							Priority						
Envi.	mAP	Recall	Precision	F1-score	TP	FP	Avg. IoU	mAP	Recall	Precision	F1-score	TP	FP	Avg. IoU
Day	0.79	0.84	0.76	0.8	217	33	0.7	0.81	0.86	0.78	0.82	223	27	0.71
Foggy day	0.74	0.79	0.72	0.76	207	43	0.68	0.76	0.81	0.73	0.77	210	40	0.69
Cloudy	0.76	0.81	0.74	0.77	213	37	0.69	0.78	0.83	0.75	0.79	218	32	0.7
Dusk	0.77	0.82	0.75	0.78	215	35	0.7	0.79	0.84	0.76	0.8	220	30	0.71
Foggy night	0.71	0.76	0.7	0.73	200	50	0.66	0.73	0.78	0.71	0.74	205	45	0.67
Night	0.69	0.74	0.68	0.71	193	57	0.64	0.71	0.76	0.69	0.72	198	52	0.65

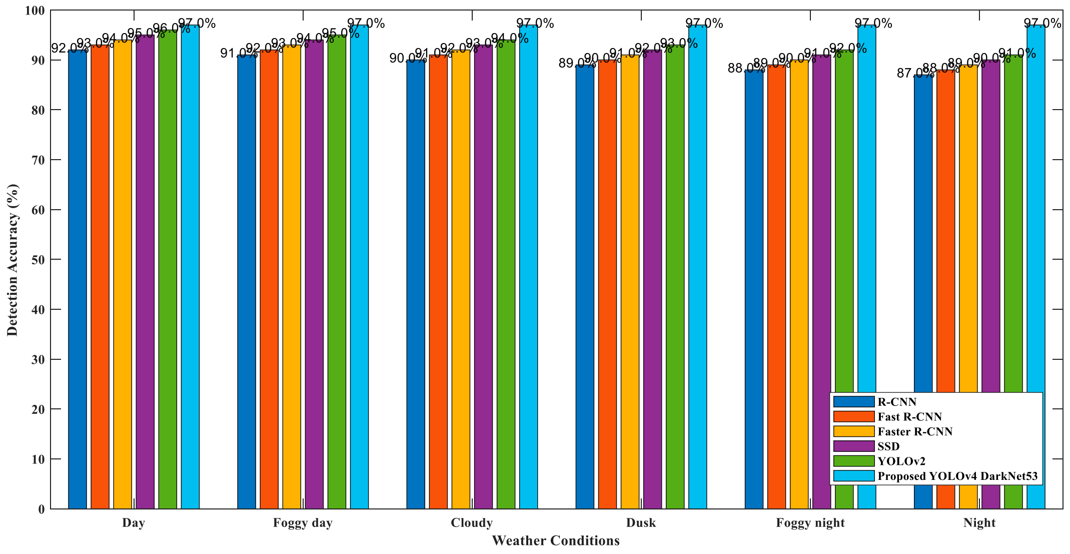


Figure 5. Comparative analysis of traffic signboard detection methods.

weather conditions as shown in Fig. 5. Even when juxtaposed with established detection methodologies such as R-CNN, Fast R-CNN [30], Faster R-CNN [31], SSD [32], and YOLOv2 [33], the proposed model consistently registers a remarkable detection accuracy of 97%. This unvarying high performance across all conditions underscores the model's robustness and adaptability. The enhancements incorporated into the YOLOv4 CSPDarkNet53 evidently render it a more efficient and reliable choice for real-world applications, particularly in dynamic environments where weather variations can pose significant challenges to accurate traffic sign detection.

5. Conclusion

The real-time testing and validation of object detection models have been simplified using the proposed method. In this study, the YOLOv4 object detection and classification model was trained using the GTSRB dataset. The model was validated in various environmental conditions such as day, foggy day, cloudy, dusk, foggy night, and night using the VFF approach, and was tested at a vehicle speed of 72 km/hr. Impressively, the model recognized traffic signs with an accuracy exceeding 95% at a detection speed of 30 fps under these varying conditions. A comparative analysis was performed by juxtaposing the model's traffic sign detection results with the ground truth detection list from CM. This comparison provided insights into the model's accuracy, detection range, and the number of TP, FP, and FN detections. Such a method makes it convenient to pinpoint inconsistencies in object identification for specific classes, thereby guiding dataset improvements and subsequent model retraining. The VFF plays a pivotal role in fine-tuning the model to enhance its performance. A salient feature of the proposed method is its seamless compatibility with various object detection models, including R-CNN, Fast R-CNN, Faster R-CNN, SSD, and YOLO. Moreover, VFF expedites the validation process demonstrates a noteworthy improvement in accuracy, achieving a 2% to 5% increase across diverse environmental conditions such as foggy days, overcast skies, fog-enshrouded nights, and clear nights. Given that CM is a simulation tool tailored for automotive environments, this approach is ideally suited for all automotive object detection models, facilitating testing and validation in real-world scenarios. It's worth noting that this study didn't encompass rain and snow conditions due to limitations within the CM tool. Looking ahead, there's potential to augment this study by integrating VFF with other simulation tools, enabling the evaluation of object detection models under a broader spectrum of weather conditions.

Informed consent

Informed consent was obtained from all the authors.

Disclosure statement

No potential conflict of interest was reported by the author(s).

ORCID

Keerthi Jayan http://orcid.org/0000-0002-3668-8500

References

[1] Prakash YS, Muskan J, Preeti R, et al. An improved deep learning-based optimal object detection system from images. Multimed Tools Appl. 2023: 1–28. doi:10.1007/s11042-023-16736-5.

[2] Kendrick B, Eng Kevin H, David PC. Area under the precision-recall curve: point estimates and confidence intervals. In: Joint European conference on machine learning and knowledge discovery in databases. Springer; 2013. p. 451–466. doi:10.1007/978-3-642-40994-3_29

[3] Owusu AE, Solomon M. Object detection in adverse weather condition for autonomous vehicles. Multimed Tools Appl. 2023. doi:10.1007/s11042-023-16453-z.

[4] Keerthi J, Muruganantham B. Advanced driver assistance system technologies and its challenges toward the development of autonomous vehicle. Proceedings of Intelligent Computing and Applications, Ghaziabad, India, 2021. pp. 55–72. doi:10.1007/978-981-15-5566-4_6

[5] Sarita G, Anuj K. Image-based automatic traffic lights detection system for autonomous cars: a review. Multimed Tools Appl. 2023;82:26135–26182. doi:10.1007/s11042-023-14340-1.

[6] Álvaro A-G, Álvarez-García Juan A, Soria-Morillo Luis M. Evaluation of deep neural networks for traffic sign detection systems. Neurocomputing. 2018;316:332–344. doi:10.1016/j.neucom.2018.08.009.

[7] Sebastian H, Johannes S, Jan S, et al. Detection of traffic signs in real-world images: the german traffic sign detection benchmark. Int Joint Conf Neural Netw. 2013: 1–8. doi:10.1109/IJCNN.2013.6706807.

[8] Amir A, Rico M. Switching network for mixing experts with application to traffic sign recognition. Multimed Tools Appl. 2023;82:43841–43864. doi:10.1007/s11042-023-14959-0.

[9] Alexey B, Chien-Yao W, Mark LH-Y. YOLOv4: optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934. 2020. doi:10.48550/arXiv.2004.10934.

[10] Yi L, Jinguo L, Ping M. Attention-YOLOV4: a real-time and high-accurate traffic sign detection algorithm. Multimed Tools Appl. 2023;82:7567–7582. doi:10.1007/s11042-022-13251-x.

[11] Asem K, Rahman AHSA, Azri RR, et al. Natural image noise removal using non local means and hidden Markov models in stationary wavelet transform domain. Multimed Tools Appl. 2018;77:20065–20086. doi:10.1007/s11042-017-5425-z.

[12] Teena S, Benoit D, Nicolas D, et al. Deep learning-based object detection and scene perception under bad weather conditions. Electronics (Switzerland). 2022;11(4):563), doi:10.3390/electronics11040563.

[13] Abu A-HQ, Manaf G, Ammar O. Detection in adverse weather conditions for autonomous vehicles via deep learning. Ai. 2022;3(2):303–317. doi:10.3390/ai3020019.

[14] Mamoona H, Farzeen A, Zaman JN, et al. Traffic management: multi-scale vehicle detection in varying weather conditions using YOLOv4 and spatial pyramid pooling network. Electronics (Basel). 2022;11(17):2748. doi:10.3390/electronics11172748.

[15] Yuhang L, Jianxiao M, Yuchen W, et al. A novel algorithm for detecting pedestrians on rainy image. Sensors. 2020;21(1):112. doi:10.3390/s21010112.

[16] Thomas R, Werner H. Performance evaluation of object detection algorithms under adverse weather conditions. International Conference on Intelligent Transport Systems, pp. 211–222. 2020. Cham: Springer International Publishing. doi:10.1007/978-3-030-71454-3_13.

[17] Mazin H, Hayder R. Object detection under rainy conditions for autonomous vehicles: a review of state-of-the-art and emerging techniques. IEEE Signal Process Mag. 2020;38(1):53–67. doi:10.1109/MSP.2020.2984801.

[18] Sinan H, Andreas R. Challenges in object detection under rainy weather conditions). Proceedings of First International Conference on Intelligent Transport Systems, Guimaraes, Portugal. 2018, pp. 53–65. doi:10.1007/978-3-030-14757-0_5.

[19] Talha A, Jinlong L, Hongkai Y, et al. Deep learning based computer vision methods for complex traffic environments perception: a review. arXiv preprint arXiv:2211.05120. 2020. doi:10.48550/arXiv.2211.05120.

[20] Hyung-Joon J, Dinh NV, Trung DT, et al. A deep learning framework for robust and real-time taillight detection under various road conditions. IEEE Trans Intell Transp Syst. 2022;23(11):20061–20072. doi:10.1109/TITS.2022.3178697.

[21] Xiu-Zhi C, Chieh-Min C, Chao-Wei Y, et al. A real-time vehicle detection system under various bad weather conditions based on a deep learning model without retraining. Sensors. 2020;20(20):5731), doi:10.3390/S20205731.

[22] Meixin Z, Jingyun H, Ziyuan P, et al. Traffic sign detection and recognition for autonomous driving in virtual simulation environment. arXiv preprint arXiv:1911.05626. 2019. doi:10.48550/arXiv.1911.05626.

[23] Lei R, Huilin Y, Wancheng G, et al. Environment influences on uncertainty of object detection for automated driving systems. Proceedings of 12th International Congress on Image and Signal Processing, BioMedical

Engineering and Informatics (CISP-BMEI), Suzhou, China, 2019. pp. 1–5. doi:10.1109/CISP-BMEI48845.2019.8965948.

[24] Yong-Lin K, Shih-Hsun L. Applications of deep learning to road sign detection in dvr images. Proceedings of IEEE International Symposium on Measurement and Control in Robotics (ISMCR), Houston, TX, USA, 2019. pp. A2-1-1-A2-1-6. doi:10.1109/ISMCR47492.2019.8955719.

[25] Yayun L, Takanori E, Ankit A R, et al. Semantic image segmentation on snow driving scenarios. Proceedings of IEEE International Conference on Mechatronics and Automation (ICMA), Beijing, China, 2020. pp. 1094–1100. doi:10.1109/icma49215.2020.9233538.

[26] Jia L, Zengfu W. Real-time traffic sign recognition based on efficient CNNs in the wild. IEEE Trans Intell Transp Syst. 2018;20(3):975–984. doi:10.1109/TITS.2018.2843815.

[27] Gámez SC, Yassine R. Traffic signs detection and classification for European urban environments. IEEE Trans Intell Transp Syst. 2019;21(10):4388–4399. doi:10.1109/TITS.2019.2941081.

[28] Chanho L, Ji-Hyun M. Robust lane detection and tracking for real-time applications. IEEE Trans Intell Transp Syst. 2018;19(12):4043–4048. doi:10.1109/TITS.2018.2791572.

[29] Chunzhi W, Pan W, Lingyu Y, et al. Image classification based on principal component analysis optimized generative adversarial networks. Multimed Tools Appl. 2021;80:9687–9701. doi:10.1007/s11042-020-10137-8.

[30] Ross G. Fast R-CNN. 2015 IEEE international conference on computer vision (ICCV), 2015. pp 1440–1448. doi:10.1109/ICCV.2015.169.

[31] Shaoqing R, Kaiming H, Ross G, et al. Faster r-cnn: towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell. 2017;39(6):1137–1149. doi:10.1109/TPAMI.2016.2577031.

[32] Wei L, Dragomir A, Dumitru E, et al. Ssd: single shot multibox detector. European Conf Comput Vis. 2016;2016:21–37. doi:10.48550/arXiv.1512.02325.

[33] Joseph R, Ali F. YOLO9000: better, faster, stronger. 2017 IEEE conference on computer vision and pattern recognition (CVPR), 2017. pp. 6517–6525. doi:10.1109/CVPR.2017.690.