Received: 20 February 2023 | Revised: 15 August 2023 | Accepted: 28 December 2023

CAAI Transactions on Intelligence Technology

DOI: 10.1049/cit2.12309

ORIGINAL RESEARCH

Vision based intelligent traffic light management system using Faster R-CNN

Syed Konain Abbas[1] | Muhammad Usman Ghani Khan[1] | Jia Zhu[2] | Raheem Sarwar[3] | Naif R. Aljohani[4] | Ibrahim A. Hameed[5] | Muhammad Umair Hassan[5]

[1]Department of Computer Science and Engineering, University of Engineering and Technology, Lahore, Pakistan

[2]Zhejiang Key Laboratory of Intelligent Education Technology and Application, Zhejiang Normal University, Jinhua, China

[3]OTHEM, Manchester Metropolitan University, Manchester, UK

[4]Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia

[5]Department of ICT and Natural Sciences, Norwegian University of Science and Technology (NTNU), Ålesund, Norway

Correspondence
Jia Zhu and Muhammad Umair Hassan.
Email: jiazhu@zjnu.edu.cn and muhammad.u.hassan@ntnu.no

Funding information
National Key R&D Program of China, Grant/Award Number: 2022YFC3303600; National Natural Science Foundation of China, Grant/Award Number: 62077015; Natural Science Foundation of Zhejiang Province, Grant/Award Number: LY23F020010

Abstract
Transportation systems primarily depend on vehicular flow on roads. Developed countries have shifted towards automated signal control, which manages and updates signal synchronisation automatically. In contrast, traffic in underdeveloped countries is mainly governed by manual traffic light systems. These existing manual systems lead to numerous issues, wasting substantial resources such as time, energy, and fuel, as they cannot make real-time decisions. In this work, we propose an algorithm to determine traffic signal durations based on real-time vehicle density, obtained from live closed circuit television camera feeds adjacent to traffic signals. The algorithm automates the traffic light system, making decisions based on vehicle density and employing Faster R-CNN for vehicle detection. Additionally, we have created a local dataset from live streams of Punjab Safe City cameras in collaboration with the local police authority. The proposed algorithm achieves a class accuracy of 96.6% and a vehicle detection accuracy of 95.7%. Across both day and night modes, our proposed method maintains an average precision, recall, *F*1 score, and vehicle detection accuracy of 0.94, 0.98, 0.96 and 0.95, respectively. Our proposed work surpasses all evaluation metrics compared to state-of-the-art methodologies.

KEYWORDS
access control, artificial intelligence, computer vision, intelligent control

1 | INTRODUCTION

Intelligent transportation systems (ITS) use technology to make our roads, highways, and transit systems easier, safer, and more efficient [1]. The collection of traffic data has always been one of the major research areas in operational analysis and capacity evaluation of the transportation network [2]. Traffic jams are one of the major challenges in today's urban traffic systems. As stated by the transportation department, traffic jams are of two types [3]. Non-recurring is the first one that

Syed Konain Abbas and Muhammad Umair Hassan contributed equally to this work.

occurs because of temporary disturbances, such as road accidents and bad weather. Traffic jams that occur due to inadequate or lack of road infrastructure are recurring, and around the world, half of the traffic jams are recurring.

In 2023, the Traffic Index Report from the Dutch navigation technology company Tom provided statistics on traffic congestion in 387 cities across 55 countries [4]. According to the most recent index report, city traffic congestion increased by 75%. A study showed that every year, drivers in the UK waste more than a day in traffic jams [5]. According to a survey, people waste 8.15 million hours yearly in traffic jams [6]. Traffic congestion is a concern in Pakistan's crowded cities, including Karachi, Faisalabad, Lahore, Hyderabad, and Gujranwala. Vehicle counting is an essential component of ITS for city traffic management, and information on traffic flow is gathered by counting vehicles [7]. Traffic controlled by fixed time intervals is one of the leading causes of traffic jams.

People must wait long because this technology only operates when traffic is on the road, leading to inefficiency. The survey found that of the 200,000 traffic signals in the US, 7% of travellers' trip time was lost waiting at them [8]. The typical one-way traffic light automation system, which regulates traffic by using Radio-frequency identification tags on the number plates, is one solution for the inefficiencies of fixed time interval traffic lights. The other solution detects vehicles simultaneously by placing pressure plates on the road. It detects the car when it reaches a precisely specified limit. The vehicle will be given a maximum and minimum time until the traffic load returns to normal if the vehicle is present. Additionally, these existing solutions do not count the cars. As a result, the sensors will only adjust the signal light when vehicles are around; the result is still ineffective. To solve this problem, we need a dynamic traffic light system that counts the number of vehicles and then assigns the time accordingly. Figure 1 shows a traffic jam that is captured by Punjab Safe City cameras.

So, in most countries, traffic control based on traffic density is done in two ways [9–11]. The first method involves a traffic warden manually directing traffic. However, it is quite challenging for a human to stand in the middle of the road and direct traffic from all directions. The cost of managing human resources in this way is comparatively high. The second method involves utilising closed circuit television (CCTV) cameras to monitor traffic via the control room. Still, it is difficult for a human to monitor the screens continuously. A model for detecting traffic congestion by camera images was proposed by Chakraborty et al. [12]. They employed two methods for detection: the first is known as *you only look once* (YOLO), and the second is *deep convolutional neural network* (DCNN). They used 1400 images in the dataset. They used a support vector machine to compare and apply deep learning to determine the improvement. Their proposed research found that the accuracy of the support vector machine was 85.2%. The hardware-based methods for detecting vehicles, such as radars and light detection and ranging technologies, are relatively expensive. Most software techniques still employ traditional methods such as the Gaussian Mixed Model and the Histogram of Gradients [13]. Traditional image processing techniques have limitations, including weather conditions and low light [14]. Many of the methods described here have focused on limited visual data. We use local image data to generalise our traffic situation. This work presents a framework for automating the traffic light signals using an improved Faster R-CNN deep learning-based approach to local image data.

The aforementioned methods have only utilised a limited amount of visual image data. However, our research incorporates local image data, enabling us to address critical traffic scenarios effectively. Our primary focus lies in developing a Faster R-CNN-based traffic light automation system. By leveraging live video streams from CCTV cameras, we have devised an algorithm that utilises state-of-the-art computer vision technology to accurately count vehicles. This algorithm automatically adjusts the traffic signal timing based on the traffic volume, thereby optimising traffic flow.

We leverage the Faster R-CNN for vehicle detection tasks in this work. However, YOLO object detection algorithms are also widely used in vehicle detection applications. Faster R-CNN and YOLOv6 [15] are both state-of-the-art object detection algorithms, but they have some critical differences in their advantages. Faster R-CNN generally achieves higher detection accuracy compared to YOLOv6 [16]. It utilises a region proposal network (RPN) to generate potential object bounding boxes, which allows it to have a more precise localisation of objects. This two-stage approach improves accuracy, especially for detecting small objects or objects with complex shapes. Faster R-CNN performs well in detecting small objects due to its region proposal mechanism. The RPN generates region proposals at different scales, which helps capture small objects more effectively [17].

In contrast, YOLOv6 struggles to detect small objects accurately because it relies on a single-scale prediction. Faster R-CNN allows for end-to-end training, enabling joint optimisation of the RPN and the object detection network. This flexibility is advantageous when working with limited labelled data or adapting the model to specific domains. YOLOv6, on the other hand, adopts a fixed architecture, limiting the training process. Faster R-CNN employs selective search as the default region proposal method, an efficient algorithm for generating potential object locations. This selective search approach helps reduce the number of region proposals to be processed, making the algorithm faster and more efficient than the grid-based approach used in YOLOv6. Faster R-CNN handles overlapping objects more effectively than YOLOv6. The two-stage architecture of Faster R-CNN allows it to assign objects to different regions in the image accurately, even when they are closely packed or overlapping. YOLOv6, being a single-shot detector, may struggle in such scenarios and may have difficulty distinguishing individual objects [18].

To train our algorithm, we employed a local dataset, which proved to be more suitable for our specific situation. We labelled the images by utilising local CCTV cameras and converting the video streams into images for training purposes. We divided the dataset into a 70% training set and a 30% test set. The model underwent training for up to 200,000 epochs.

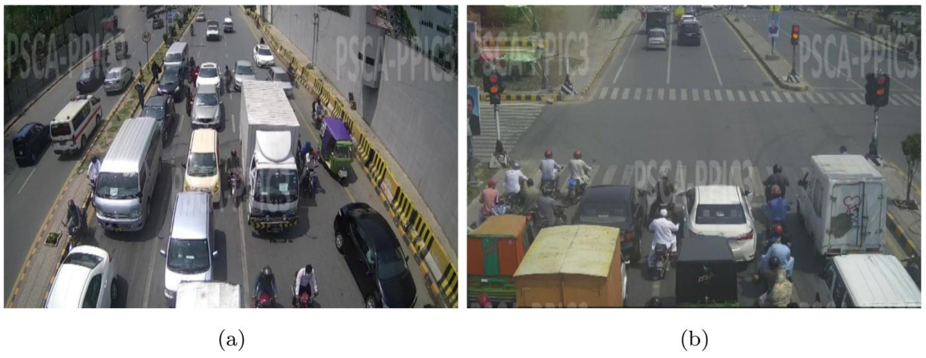


FIGURE 1 Traffic jams captured from closed circuit television camera's (a) front and (b) back views.

We rigorously evaluated the performance of our model using real-time video streams from CCTV cameras. Our proposed model detected and counted all vehicles within the camera frame, including front and back views. The accuracy of our vehicle detection algorithm, based on Faster R-CNN, reached an impressive 95.7%. For training, we utilised the COCO Vehicle dataset [19] and trained our model for 200,000 epochs, resulting in a class accuracy (CA) of 96.6% for our proposed model.

The contributions made in this work are as follows.

- We proposed an algorithm by fine-tuning the Faster R-CNN for vehicle detection and classification. Our proposed algorithm can detect smaller objects (vehicles) that are even obscured from the camera.
- We generated a local dataset with the collaboration of local Police authorities because every country has its traffic patterns, which generalises our traffic congestion problem, and an approach for scheduling traffic lights based on traffic density at the road junctions is proposed.
- We leveraged the Vehicle COCO dataset, which our proposed algorithm detects and classifies the vehicles perfectly day and night. Our proposed algorithm improved the accuracy of the detection and classification of vehicles.

The organisation of this work is as follows. Section 2 presents the related works available in the literature. We explain the methodology of our proposed architecture in Section 3. The implementation details of this work are presented in Section 4. Section 5 highlights our proposed architecture's experimentation details and results. A discussion of our work is available in Section 6. The threats to validity are presented in Section 7. Finally, we conclude this work in Section 8.

2 | RELATED WORK

Osman et al. [9] proposed a traffic management system based on computer vision and image processing. Two methods were used, one based on hardware and one designed without hardware support. This study was conducted in a replicated environment where optimisation time was measured with substantial evidence. With a fixed travel time of 180 s, approximately 720 s are needed to complete a cycle at the intersection of four roads. The dynamic system needs 720 s for cycle completion. However, based on the strength of vehicles on the road at a junction, the waiting time for each vehicle is reduced given the vehicle count, allocating a different waiting time and execution time. Gomaa et al. [20] proposed a system to detect the vehicle and then count the vehicle. They used Faster R-CNN for the detection of the vehicles. They used the 12 videos for detection and counting, applied the Faster R-CNN, and compared Faster R-CNN with YOLOv2; according to the paper, Faster R-CNN accuracy is greater than the YOLOv2. In this paper, they detect and count the vehicles but use only one class in their proposed algorithm. In contrast, we used different classes, such as cars, buses, motorcycles, and trucks. Also, we proposed an algorithm to time-schedule for signal automation to control the traffic light signals based on traffic density. We used the COCO Vehicle dataset and a self-generated local dataset because each country has different traffic patterns, so that will more generally solve the issue of traffic jams in our country.

Guo et al. [21] proposed an intelligent traffic light control system. The system is suffering from many problems because the system is manually managed. They used the reinforcement learning technique, which would be used to control the traffic light problem. For the dataset, they used 1704 surveillance cameras containing 405 million vehicle records. The paper discussed the synthetic and real-world experiments to show the advantages of the proposed system. Also, there would be some limitations to our proposed system, which would be eliminated shortly, so this would be more beneficial for the real world. One of the plans is to convert the two-phase traffic light to multiple-phase traffic, which is highly complicated to design, but it will be a more realistic state transition.

Alghyaline et al. [22] proposed a model for counting vehicles. They used YOLO and Kalman filters and the Hungarian algorithm for object detection and tracking, respectively. They divided the road into two zones, and vehicles that are income in these two zones will be counted. They used 90 min of Highway Roads YouTube videos in their dataset containing 200,000 frames and 939 samples for training their proposed convolutional neural network (CNN) model. According to this paper there, the proposed model accuracy is 88.72%.

Dai et al. [23] proposed a model for counting vehicles using CCTV video. They followed the deep learning technique. They used 6134 RGB images in their dataset. The proposed model is based on three steps: detecting the object, tracking the object, and using the trajectory process for traffic flow information. The vehicle counted according to the vehicle type. According to this paper, its accuracy can reach more than 90%.

Mirthubashini et al. [24] presents a comparative study of vehicle detection using deep learning technologies. According to the paper, if we count vehicles on different road lanes, we quickly set the traffic light signals dynamically, which is helpful for intelligent traffic light systems. The authors reviewed different deep learning techniques, for example, the YOLO technique, which is high-speed detection and is best used in real-time processing, but it is not good in the case of accuracy. A single shot detector (SSD) is used for the best results, but it is too slow and not best for smaller object detection. Faster R-CNN works best for dynamic things and is also good for overlapping objects, but it is used in many passes for single-object detection. Haar cascades work well if our concern is only with speed. Background subtraction is suitable for a quick recovery but not for sudden changes. According to the paper, YOLO gives 86.7% accuracy. Faster R-CNN gives 69%, and the future result is to improve the accuracy, and the system is automated for the select needy region to improve accuracy.

Zhu et al. [25] proposed a deep learning model for estimating urban traffic density. The authors used the deep learning technique single shot multi box detector to detect vehicles. They collected images from the unnamed aerial vehicle from five different intersections. They used 101,970 frames, and an almost 56-min video-enhanced single-shot multiplex used ResNet instead of VGGnet and Alex net. ResNet can detect objects deeply. The reported model accuracy is 93.7%. Qi et al. [26] proposed a system for traffic analysis that used computer vision approaches. They used SSDs to detect objects; their work's objective is to find and monitor vehicle pedestrians. The system takes the images and detects the objects using SSD; based on this, the next move to another section, traffic modelling, contains the statistics of vehicle humans and traffic signs. This next section is the traffic analysis based on traffic modelling and environmental information, which defines the two types of traffic analytics: traffic volume information and traffic congestion information. They used 24 h of video from each hour and 10 min for traffic volume estimation. According to the paper based on different types of vehicles, detection accuracy is more than 70%. Kamath et al. [27] introduced a groundbreaking method for connected and autonomous vehicles (CAVs) in urban road traffic. The paper discusses the challenges of unreliable navigation system connectivity in such environments. They proposed a traffic analysis model to generate accurate navigation paths for CAVs by analysing sensor-oriented traffic data. A knowledge exchange mechanism was developed to gather and create new traffic knowledge from on-road vehicles. CAVs utilise this information for navigation, resulting in improved precision.

Wang et al. [28] proposed a model for vehicle classification. It used Faster R-CNN for object detection. It consists of two elements regional proposal network and five other convolution layers that work as a detection network. Faster R-CNN is an updated version of Fast R-CNN; it eliminates the selected search with the RPN. The proposed model built the regional proposed network with the help of the top of the last convolution layer. Two-hundred and fifty-six dimension is the feature map of the sliding window. The ReLU feature converted it into two fully connected (FC) layers. Bounding box regression is known as the reg layer; another layer is the box classification layer and the *cls* layer. The prediction of 4K coordinates in the *k* proposal used a *reg* layer. For output of 2K scores, which are the chances of objects involved in *k*, proposed to use *cls* layer. The proposed model used a 3×3 convolution layer and two pairs of 1×1 convolution layers. Previously shared convolution layers of RPN and detection network in the proposed network used ZFNet as the network's backbone. They modified the ZFNet by adding two more convolution layers and one new max-pooling layer. Now, it contains 7 convolution layers. According to the paper, doing this improves the performance of the expression network. For the dataset, they used 37,578 pictures for training. According to the paper, the proposed model accuracy is 90%.

Suhao et al. [29] used Faster R-CNN on MIT and Caltech car datasets for vehicle classification. They used Faster R-CNN and improved the regional proposal network, which was an important part of CNN, and that is a deep learning approach for the detection of objects. In this paper, they improve the layer in the Faster R-CNN model by modifying the RPN network. It outputs 256 characteristic maps via the one time 3×3 convolution kernel. By doing this, the network structure becomes more superficial and improves computational complexity. The vehicle type detection system first gets the dataset, then makes the dataset enter vehicle images for training, then CNN extracts features and then enters picture features. Then, it moves to RPN networks, Fast R-CNN, and Faster R-CNN moves forward to the training vehicle classification model, and then the result is vehicle type detection. For the dataset, they used 5038 car images, 987 minibus images, and 1207 SUV images. According to the paper, the accuracy of different vehicles based on different sample sizes and networks is different. The highest accuracy for the car, bus, and SUV using the VGG16 network is 84%, 83%, and 78%, respectively.

Sridevi et al. [30] proposed an effective traffic management model based on the computer vision technique. Gaussian Mixture Model is used for traffic automation. They used the MIT traffic dataset that contains 20 videos, and each video contains 8294 frames. According to Ref. [30], their proposed model shows 95% accuracy in the MIT dataset. The highway data set contains 254 videos out of 20 that are used to input to the project to measure accuracy; each video contains 53 frames, and our dataset shows 15% accuracy. Yadav et al. [31] proposed a system to improve ITS by developing a self-adapting algorithm to control road traffic based on deep learning techniques. It used state-of-the-art real-time object detection based on the DCNN YOLO. YOLO offers breakneck interface speed with minor compromises in accuracy, specifically at lower resolutions and with small objects.

Wu et al. [32] discussed the challenges of object detection in low light and proposed a method for picture enhancement and object detection that makes use of edge computing and a multi-task-driven framework. Accurate results are obtained when picture enhancement and object detection technologies are combined. The experiment on a cloud computing system illustrates the significance of the suggested approach for low-light object detection efficiency in mobile photos and environments with poor lighting.

Tang et al. [33] examined the impact of many aspects on the evolution and effectiveness of such transformations, as well as the significance of borderless collective remodelling in the intelligent connected car industry. The findings indicate that an advanced collaborative effort, including technology policy providers and auto endeavours, is vital for the successful evolution and change of the intelligent connected car business. Chen et al. [34] proposed a procedure enhancing the validity of the AI model deployment in the 6G vehicular edge computing and took the object detection task as an example. The approach is based on model stabilisation and model adaptation; for model stabilisation, the latest model needs to be implemented to improve its validity. For the adaptation stage, there are two methods: knowledge distillation and model parameter pruning. These methods adjust between model performance and run-time resources to maintain the implementation of the AI model. When these model strategies are deployed onboard, the edge terminal proposed model mean average precision accuracy is 80.1%.

Previously mentioned approaches have solely utilised limited visual picture data. In contrast, our research combines local images and publicly available data, enabling us to address critical traffic challenges effectively. Our main objective is to develop a faster R-CNN-based traffic signal automation system. By leveraging live video streams from CCTV cameras, we have devised an algorithm that accurately counts vehicles using advanced computer vision technology. This method enhances traffic flow by automatically adjusting traffic signal timing according to the traffic load.

3 | PROPOSED METHODOLOGY

Many of the methods described here have focused on limited visual data. We use local image data to generalise our traffic situation. Our proposed architecture is presented in Figure 2. The input image is passed through the convolution neural network for object detection and class detection. A feature map of the image is generated. The generated feature map is assigned to the RPN block, where the feature vectors are extracted, and object proposals are generated. We assign those object proposals to the Faster R-CNN block, where an output of the detected class is returned. CNN is used for vehicle counting of images. Faster R-CNN [35] uses the image as input and provides the convolution feature map; a separate network is used to predict region proposals.

Before training the Faster R-CNN model, we need to adjust specific hyperparameters to achieve optimal performance. Some of these hyperparameters include learning rate, image resizing, number of classes, feature extractor, batch size, optimiser, and number of epochs, among others. Fine-tuning these hyperparameters is crucial to obtain better results, considering the unique characteristics of our dataset, problem requirements, and available hardware resources. Detailed information regarding the hyperparameter settings can be found in this section.

3.1 | Pre-processing

When images are received at the edge server, they are preprocessed through the vehicle object detection algorithm. In preprocessing, we set the image size, cropped the unwanted empty areas, and adjusted the contrast to the proper brightness and low light areas adjustment of the images. The contrast transforming function is proposed as follows.

$$\psi(c) = 4c - 6c^2 + 4c^3 - c^4 \tag{1}$$

where c is the neighbourhood's contrast value and gives the updated contrast value images received on servers from all the cameras at a traffic intersection. At the same time, after vehicle detection, the specific class is assigned to the vehicle based on the object probability score and the class probability scores, respectively, and then based on the vehicle density time scheduling for the signal automation and, after that, based on the above steps decision making. The classification framework of our proposed work is shown in Figure 3.

3.2 | Transmission of data

Due to the increasing number of IoT devices, most data are uploaded daily to the cloud. The best way to handle this is first to process the data on edge and then send it to the cloud. Different methods are used for fast data processing, such as cloud computing [36] and cloudlet, but cloud computing is not good when we need fast processing. It is quite complicated and time-consuming to first put all the data on the cloud and then search for this method. We need robust computation and processing power, so we used edge computing. We process the data on the edges of the node edge, which means we use the computing devices or resources of the network along with a dedicated path and cloud server.

We set a node with a tiny microprocessor in our proposed system containing the Nvidia Jetson GPU. This node acts as an edge device. The process streams data for all traffic light signals placed at the road's intersection and passes them to the node, which makes a decision based on the density of the vehicles. Figure 4 illustrates our proposed data transmission framework.

3.3 | Feature extraction

In this module, we focus on feature extraction, a process that aligns with the operations of CNNs. The extracted features are represented as an n-dimensional array. Specifically, our

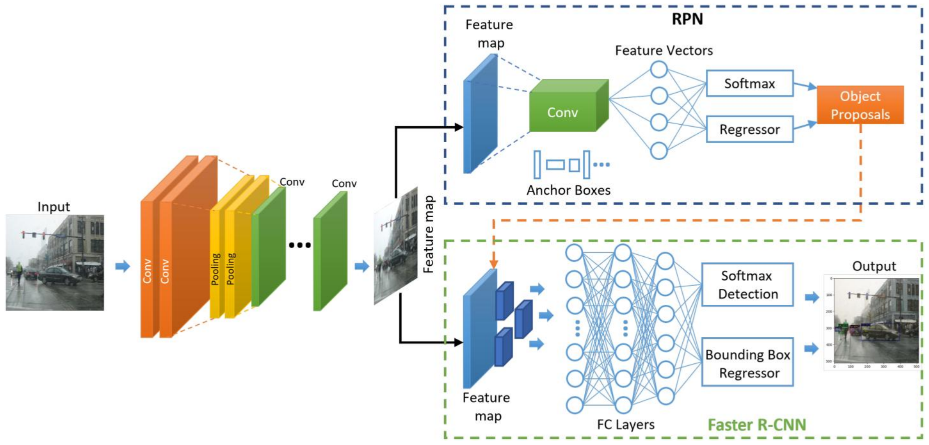


FIGURE 2 Our proposed architecture for vehicle detection and identification.

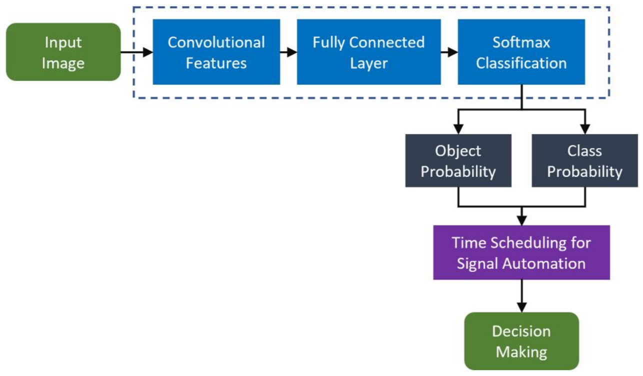


FIGURE 3 Classification framework of our proposed work.

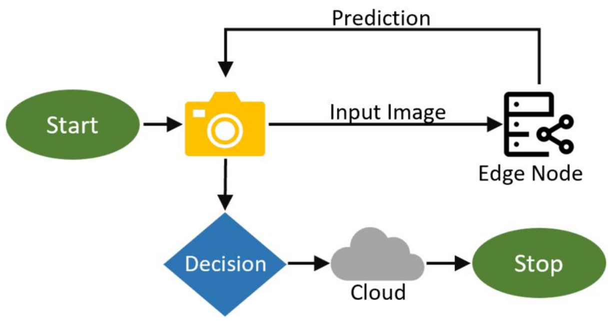


FIGURE 4 Data transmission flow in our proposed framework.

architecture, as illustrated in Figure 2, employs the VGG 16 model to generate feature maps. This transformation process elevates the information from low-level to high-level across successive layers. VGG 16 is structured into five convolutional blocks: the initial two blocks each comprise two 2D convolutional layers followed by a max-pooling layer, while the latter three blocks each contain three convolutional layers paired with max-pooling layers. In our configuration, we omit the dense and flatten layers of the network. Consequently, an input image with dimensions 1920 × 1080 × 3 is transformed into a feature map of dimensions 35 × 62 × 512 after processing. This feature map then serves as the input for the subsequent module and the softmax classifier. The initialisation of weights and biases is set to zero, with adjustments made through backpropagation as the model learns.

3.4 | Region proposal network

For implementing the work of Ren et al. [35], 9 anchors with rules are pinched on every receptive field of the feature map. Each anchor's coordinate is the regression layer's output in the RPN module. An anchor is a box that moves overall on the image. The foreground and background probability score is the output of the class layer. After applying non-max suppression with intersection over union (IoU), the threshold value is 0.3 for non-positive, which represents the background, and 0.7 for positive anchors representing the foreground. The region of interest (RoI) pooling layer is applied to selected candidate proposals and feature maps for uniformly sized feature maps.

3.5 | Backbone architecture

We utilised the VGG-16 architecture to extract the convolution features, which is this work's backbone. The RPN and classification network both use the features extracted from CNN's design. The frames coming from the videos are first normalised into the fixed range. In our case, we have resized each frame into 256 × 256 × 3.

After resizing, the image is passed to the CNN architecture. A stride of 1 is used in the third layer, whereas a stride of 2 is used in the fourth layer. After max pooling in the fourth layer, the size is reduced to 56 × 56 × 128. Pooling is implemented in the fifth and maximum convolution layers by applying 256 filters with a 3 × 3 kernel size. From 7 to 12 layers, we have two sets of 3 convolution layers. Five hundred and twelve filters are applied on these layers with a kernel size of 3 × 3 and a stride of 1. After the 12th convolution layer, we got a feature map of size 7 × 7 × 512. In the 13th FC layer, we flattened the features into the 25,088 features.

In our architecture, there are three FC layers; the first two FC layers map the 4096 features, whereas, in the last FC layer, the output is mapped to the number of classes on which we have trained our system. The RPN receives the characteristics from the convolution layer for object identification and classification.

3.6 | RoI classifier and bounding box regressor

Feature maps of size 7 × 7 go through R-CNN, classifying RoI into specific classes like a vehicle, human, and background. RoI, which has a background class, is not considered for the training process. The network contains two FC layers that output 4096D features and predict the class probability scores. The bounding box regressor updates the coordinate and RoI proposal size. The loss function used in this work is the sum of RPN loss and R-CNN loss. The detected object count gives the vehicle count, and the total number of vehicle counts in the frame shows the traffic density.

3.7 | Calculating lane with high traffic density and next signal time

We get videos from cameras located at four-lane intersections, count the number of vehicles in each lane, and calculate the lane with high priority using a threshold value. The threshold value is dynamic, and it is measured to be 2 × 3 of the maximum count of the object at a specific timestamp (see Equation 2).

$$Threshold = \frac{2}{3} \times \text{maximum number of vehicle} \tag{2}$$

The green signal time for high-priority lanes is calculated by multiplying the fixed signal time by 4. For example, if the fixed signal value is 30 s, then high-dense lane signal time is calculated by multiplying 30 × 4 s.

There are two streams in the proposed model for vehicle detection; one is used for the RPN, and the other is used to recognise the vehicles. In older versions, the selective search method is used in the R-CNN to detect the object boundaries within the image. However, this technique is costly in terms of time and computation. In an RPN, the time cost is lower than the selective search because most of the computation time is spent by this process for object recognition. Figure 5 explains the flow of the proposed system.

3.8 | Vehicle detector

For the vehicle detectors, the RPN used anchors to draw the anchors and find the output, which contains most likely the objects. Top anchors that give more output are selected; for example, we selected the top 2000 anchors based on their score output. RPN network is the lightweight used to scan the image by sliding windows over the anchors. For the similarity of the current boundary box with the other boundary boxes, overlap of IoU is used. Intersection over union is calculated by using the following equation.

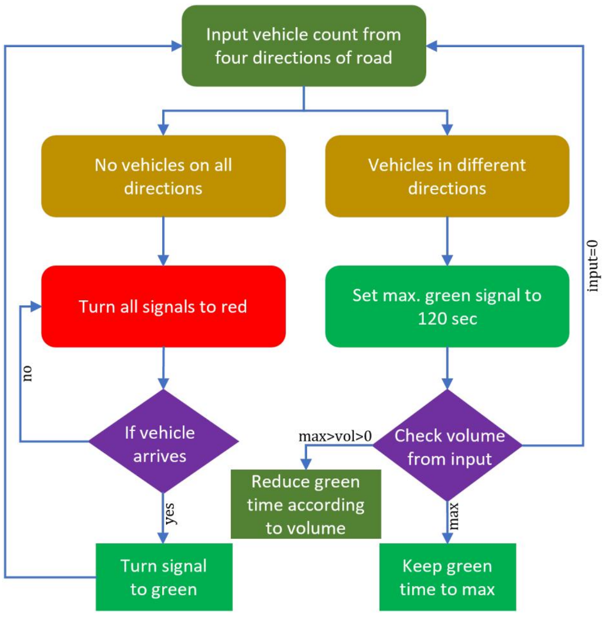

Input vehicle count from four directions of road
No vehicles on all directions
Vehicles in different directions
Turn all signals to red
Set max. green signal to 120 sec
input=0
no
If vehicle arrives
max>vol>0
Check volume from input
Reduce green time according to volume
yes
max
Turn signal to green
Keep green time to max

FIGURE 5 Time scheduling for signal automation in our proposed work.

$$IoU = \frac{\text{Area of overlap of compared boxes}}{\text{Area of union of compared boxes}} \tag{3}$$

For the training, the anchor value must be positive or negative. If the value is neither negative nor positive, it is not considered for training. So from all the above information, the loss function is described by this equation.

$$\begin{aligned} Loss(P_k, T_k) = {} & \frac{1}{n_{class}} \sum_k M_{class}\left(Pk, P_k^*\right) \\ & + \lambda \frac{1}{n_{reg}} P_k^* M_{reg}\left(T_k, T_k^*\right) \end{aligned} \tag{4}$$

In the above equation, K is the number of objects. Pk represents the probability of objects that define the network's anchors. Pk^* represents the ground truth probability value whose value is positive for 1, and the value is negative when it is 0.

Tk defines the four coordinates of the image based on network prediction decided where the object is found. $Pk^* M_{reg}(T_k, T_k{}^*$ defines that regression loss is activated if the anchor value is positive; otherwise, it will not activate. Anchors or sliding windows of the regional proposal network scan parallel because of the convolutional nature of RPN using GPU.

The RPN does not scan the whole image. It uses extracted features of other networks within the same architecture, defined as the backbone's feature map. A backbone feature map is useful for the RPN to extract features efficiently and ignore feature duplication. The proposed model predicts two outputs.

After getting the RoI, one is used for object classification, and the second defines the bounding box size and location. There is a challenge when it defines detected object classification. RPN proposes the variable size of the RoI; due to this reason, the classification is not correctly performed. Because for the classification, we need images of the same size. The solution to the problem is using RoI pooling, which crops, fixes the image size that contains the object, and then resizes it. The Backbone Architecture of the convolution feature for classifying the detected objects is defined in the next section of the methodology.

3.9 | Proposed CNN for vehicle detection

A deep CNN design with fixed structure sizes of 2262 entities is utilised to solve the N-ways classification problem. A CNN is applied for every training frame I_x, $x = 1 \ldots X$ and the score is computed by $y_x = w\pi(I_x) + B\epsilon R^n$ through the FC layer having N linear predictors $R^{n*D}, B\epsilon\mathbb{Z}$ each for one identity. For the computation of the loss value, given scores are compared with the ground truth label value of each class [37]. When training is completed, the classification layer is removed, and the given score is used for vehicle detection using Euclidean distance. Table 1 shows the layer information used in our proposed network.

The cropped face image size of 224 × 224 × 3 is used as input in our proposed architecture. A total of eight blocks are used of which five blocks are used as a convolution, and three are FC layers. Each convolution layer is based on non-linearities such as ReLU, the rectification layer, and max-pooling. The three Conv layers and the two FC layers have an output of 4096 dimensional, and the output of the last FC layer is 2622 dimensional. It depends on the number of classes used in the dataset. To normalise the unnormalised vector, we used the softmax layer on to the second FC layer and presented predictions into the probabilistic form. The given equation can calculate the probabilistic score.

TABLE 1 Information of layers used in our proposed architecture.

No.	Layers	Filter size	Kernel	Stride	Padding
1	Conv	11 × 11	96	4	0
2	Conv	7 × 7	128	1	2
3	Conv	5 × 5	256	1	1
4	Conv	3 × 3	256	1	1
5	Conv	3 × 3	384	1	1
6	FC	1	4096	1	0
7	FC	1	2622	1	0
8	FC	1	2622	1	0

Abbreviation: FC, fully connected.

$$P_s = \frac{e^{P_s}}{\sum_x e^{P_x}} \text{ for all } y \text{ in}\{1, 2..n\} \tag{5}$$

4 | IMPLEMENTATION DETAILS

4.1 | Training parameters of proposed architecture

A locally generated dataset is used in our proposed architecture. We collected over 5000 images from live-streaming cameras of the safe cities placed on traffic signals. All images are labelled using the 'Labelimg' software that converts the image label into XML code. We also collected the dataset on weather conditions, such as rainy and sunny days and different daylight conditions. Figure 6 shows the sample instances from our collected dataset. Each image label defines the name of the particular vehicle. We have the following objects on which the system is trained for detecting vehicles: Car (Pickup, Jeep, and Double Cabin), Motor Cycle, Rickshaw (QingQi), Bus, Wagon (Hiace, Hiroof, and Coaster), Minitruck (Mazda and Shehzore), Trolley, and Truck (All type, Single/Double).

To reduce average loss during network training, we identify the parameters for the softmax layer. The proposed architecture weight starts with the help of the random sampling of the Gaussian Mixture and with the value of zero means, and the standard deviation is set to 10^{-2}. From the start, the bias value is set to zero. All the training images' values are resized correctly with the minimum frame width, and the size is 256.

Nevertheless, the colour augmentation is not performed. Weight is also optimised through stochastic gradient descent (SGD) with the help of the batch size of 64, having a coefficient momentum of 0.9. For the model, regularisation weight, decay, and momentum are used.

After that, the coefficient value is set to 5×10^{-4}, and after the FC layer, the drop-out layer is used with a ratio of 0.5. From the start, the learning rate is 10^{-2}; after that, its value decreases with the multiple of 10^{-1} if the accuracy of the

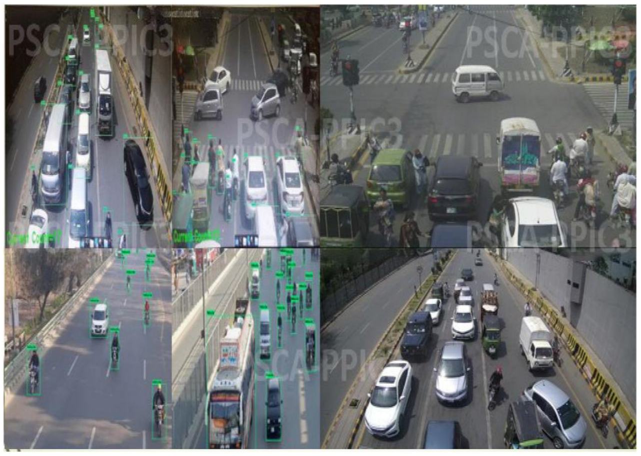


FIGURE 6 Collected samples of our dataset.

validation data is not increasing. We used 200,000 epochs for the training of our proposed network.

4.2 | Loss function

In deep learning approaches, the error is calculated by comparing the predicted label value to the actual label value. The loss function is used for the computation of error. This technique is used to verify the working of the trained model.

If the model prediction varies significantly from the actual results of the training data set, the loss function's value will be high. With the help of some optimisation parameters, the value can be reduced. For the performance of the model, the loss function is critical. The mean square error (MSE) function is used for such calculation. MSE is a commonly used loss function. It finds by comparing the value of the actual label with the prediction. The loss function is only concerned with an average magnitude error rather than the direction.

In more detail, the value is penalised compared to less deviated prediction because the square prediction value is far from the actual value. Due to the large dataset and the complexity of the deep neural network, computation power is significant.

$$MSE = \frac{1}{N}\sum_{n=1}^{N}\left(\hat{P}_n - P_n\right)^2 \tag{6}$$

In the above equation, N represents the number of data points, while the predicted value defines the $\hat{P}$ and P for the actual values of the particular sample.

5 | EXPERIMENTAL EVALUATION

The base of our proposed network is built on two elements: the computation power and the amount of data. The aforementioned variables have an inverse relationship with the accuracy of computer vision and deep learning systems. Deep learning application is, therefore, essential in our proposed technique, which combines computer vision and deep learning. The data set description is defined in the following section.

5.1 | Dataset preparation

5.1.1 | Punjab Safe City cameras dataset

In our proposed work, we used two types of datasets. In the first dataset, we used videos from the traffic signal-mounted Punjab Safe City CCTV cameras in Lahore, Pakistan. Our data collection was constructed utilising several cameras, various viewpoints, and various times. To test the functionality of our proposed algorithm and how it will work in various settings, we obtained the dataset at various times by changing the camera angles. For example, we used rainy-day data and sunny-day data. We also utilised night-view camera videos to test how it behaves at night.

For the dataset, the camera captures images of vehicles from the front and back. Using both the front and back perspectives of the traffic signal road, we evaluated the usability and precision of our proposed method to determine which viewpoint yields good results. Table 2 represents the dataset distribution for different views regarding daylight illumination and camera angles. Such views are illustrated in Figure 7.

Once the videos are obtained, the following step turns the collected video data into image frames. We leverage a software-based conversion, ‘Free Video to JPG Converter’, to convert videos into *.jpg* formats. After receiving frames, we move on to image preprocessing by annotating the images with ‘Labelimg’ software. The XML files for each image utilised during model training are saved in the backend.

Fine-tuning is performed for the inception v3 module of our CNN-based architecture. Following the inception module, the linear network is designed. The dataset is divided into 70% training and 30% testing data. We used the tensor flow deep learning framework to train the inception module. Python is used to implement the code of our proposed network.

5.1.2 | COCO Vehicle dataset

We also used the COCO Vehicle dataset, a publicly available dataset [19]. We also trained our model on that dataset.

TABLE 2 Dataset distribution over views.

Views (angles)	Data (frames)
Front view	6000
Back view	4000
Sunny view	5000
Rainy view	2000
Night view	3000

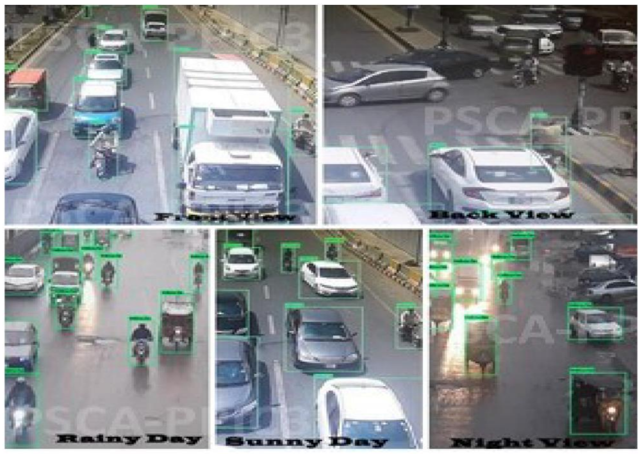


FIGURE 7 Sample instances of our collected dataset with respect to different views.

Individual images of size 640 × 640 and classes such as car, bus, motorcycle, and truck are assigned to the network. We used 13,000 images for training, 3800 images for validation, and 1900 for testing. Samples of the COCO Vehicle dataset are shown in Figure 8.

5.1.3 | Crowdhuman

We also trained our model on the Crowdhuman dataset, which is a publicly available dataset [38]. This dataset has two classes: head and person. We used 3300 images for the training set and 1100 for the validation set. We used a 75% dataset for training and a 25% dataset for validation. The size of images is 640 × 640. Samples of the Crowdhuman dataset are shown in Figure 9.

5.2 | Object detection results

Nvidia-1080 Ti GPU, 12 GB memory, was used for training. We used the TensorFlow deep learning framework for the architecture. We initially annotate the data before training our proposed Faster R-CNN-based network for object detection and recognition. On annotating data, we draw attention to a particular object in the dataset's images. Many epochs are needed to complete the training procedure because we are training the model from scratch. The front and back views for detected vehicle counts are represented in Figure 10a,b, respectively.

If there is a significant discrepancy between the loss value, training accuracy, and test accuracy, then the model has likely been overfitted. Overfitting would happen if the model learned the noise and fluctuation of the training dataset. The issue with overfitted models is that they need to perform better on new datasets, and their capacity to generalise is compromised. The answer to the overfitting issue is to drop out. The dropout amount was initially set to 0.7, but it can be modified based on the predefined threshold between training and testing accuracy and loss. After the completion of training on our local self-generated data set from Punjab Safe City cameras, our training accuracy is 97.7%, and validation accuracy is 95.7% as shown in Figure 11a,b, respectively.

For the COCO Vehicle dataset, our model detected different vehicle categories (see Figure 12a), a car far from the camera, for example, the smallest object (see Figure 12b), and in Figure 12c, our model detected the three cars at night time. Our model CA is more than 96.6% on the COCO Vehicle dataset, as shown in Figure 13.

The model is trained in over 200,000 iterations. SGD is utilised for weight optimisation. For the first 1000 epochs, we predetermine a learning rate of 10^{-1}, which declines by 10% after 20,000 epochs. The dropout ratio was also added to reduce overfitting. After 3500 iterations, as shown in Figure 14a, we were able to achieve 91.26% accuracy. The graph shows we have a 93.6% accuracy rate after 3500 iterations on the training data. When we reach the highest level of testing accuracy, the training is terminated. Figure 14b shows how accurate the test results are. It can be seen from Figure 14 that the training and testing accuracy have grown simultaneously, indicating that our proposed model does not overfit.

For the Crowdhuman dataset, we used 5000 epochs; our proposed model's CA is 92.3%, as shown in Figure 15. Our model detected humans accurately (see Figure 16a), the model also detected six persons accurately (see Figure 16b), and in Figure 16c, our model detected the people in line accurately.

5.3 | Comparative analysis

According to a study [39], the suggested model's vehicle detection accuracy is 94.20% since Arora et al. [39] used Faster R-CNN and used a self-generated dataset of 3975 pictures, as shown in Table 3. Mirthubashini et al. [24] used the COCO dataset and trained YOLOv3, and their model's detection accuracy is 76%. The proposed model's vehicle detection accuracy for the work of Dai et al. [23] is 87.6% when using YOLOv3 and a self-generated dataset containing

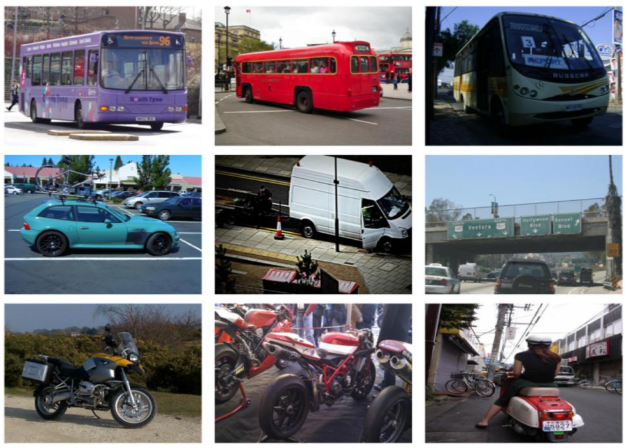


FIGURE 8 Sample instances from the COCO Vehicle dataset.

FIGURE 9 Sample instances from the Crowdhuman dataset.

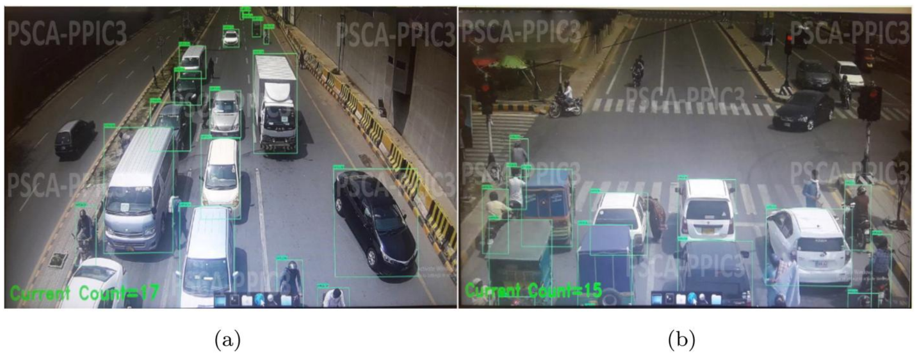


FIGURE 10 Detected vehicles count from (a) front and (b) back views of Punjab Safe City cameras.

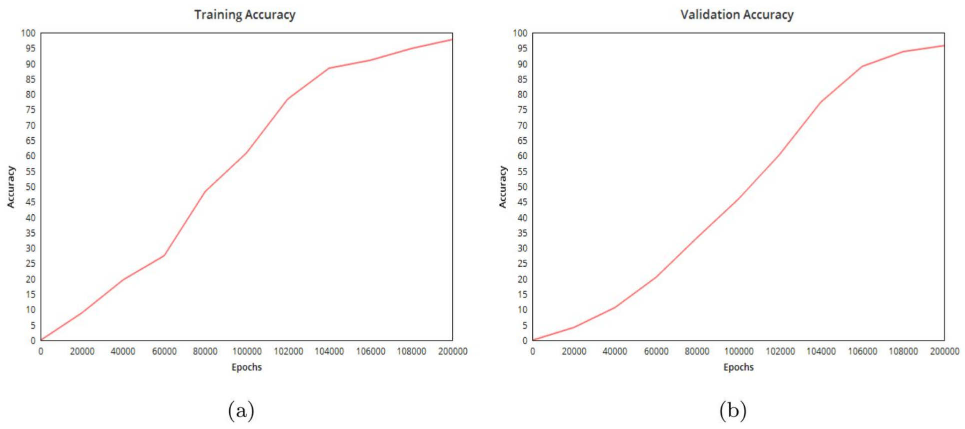


FIGURE 11 The (a) training and (b) validation accuracies of our local dataset after 200,000 epochs.

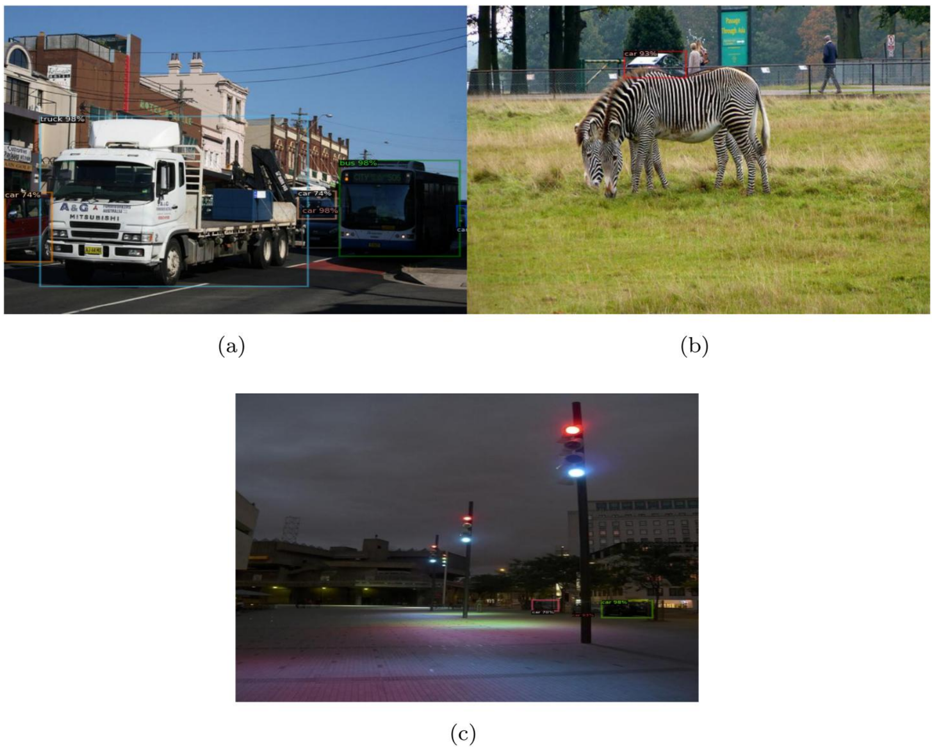


FIGURE 12 Detection of vehicles (a) car, bus and truck, (b) smallest vehicle objects, and (c) at night.

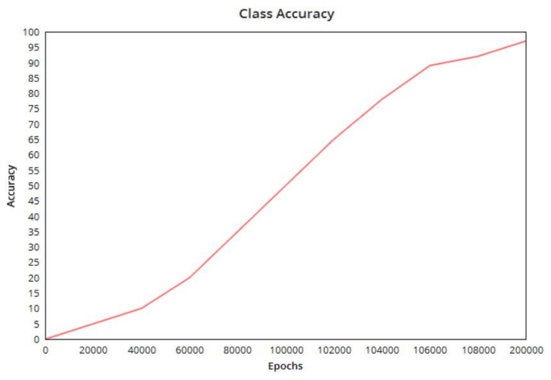


FIGURE 13 Class accuracy on test data by using Faster R-CNN on the COCO Vehicle dataset.

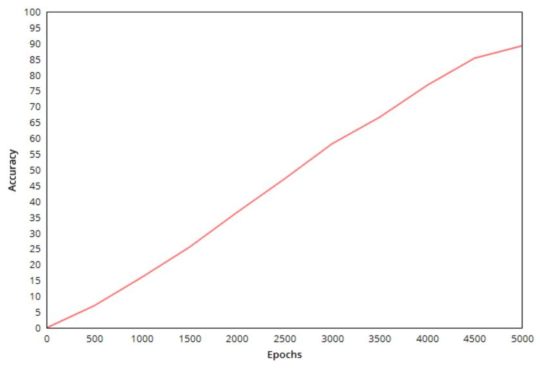
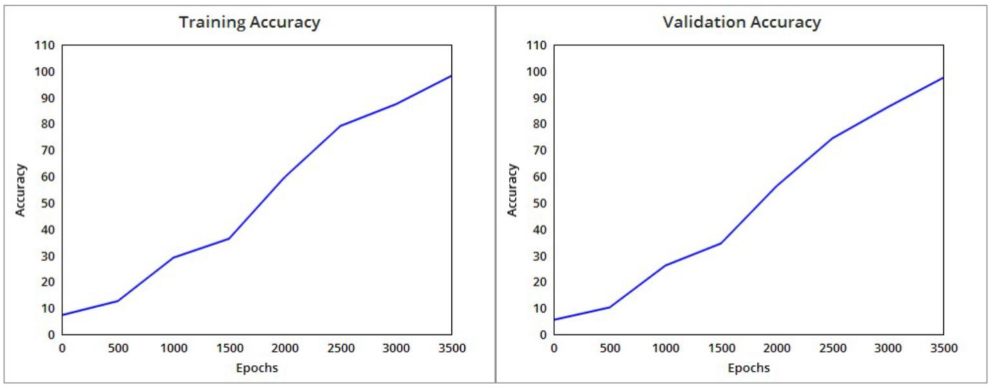


FIGURE 15 Class accuracy by using Faster R-CNN on the Crowdhuman dataset.



(a) (b)

FIGURE 14 Accuracy of (a) train and (b) test data after 3500 epochs.

6134 photos. According to Qi et al. [40], the proposed model's accuracy is 81%, and they employed SSD and 24-h videos as a dataset. Chakraborty et al. [12] suggested a model that uses YOLO and uses 2418 photos as a dataset, with an accuracy rate of 91.5%.

Song et al. [41] suggested a model that used YOLO v3 for object detection and used 11,129 images as a dataset. We compared our proposed work with their work, and it can be seen from Table 3 that our method outperformed theirs. We also considered another recent work for comparison in which Mittal et al. [42] proposed a hybrid approach of using Faster R-CNN and YOLO together using a majority voting classifier. However, due to using a majority voting classifier, their proposed method is prone to biased predictions, ignoring the minority class. This results in poor performance for the minority class and leads to reduced overall accuracy.

Our proposed model uses the COCO Vehicle dataset [19] and the self-generated dataset from Punjab Safe City cameras. Our model detection accuracy is 95.7%, and our CA is 96.6% after we fine-tuned the Faster R-CNN. Our proposed model's average precision, recall, $F1$ score, and vehicle detection accuracy in the day and night modes were 0.94, 0.98, 0.96, and 0.95, respectively.

6 | DISCUSSION

Our proposed model is applied to the Lahore Smart City cameras dataset and achieved state-of-the-art performance. The work using Faster R-CNN for traffic light automation also lacks literature. We used a local image dataset that will more generally address the issue of traffic jams because each country has different traffic patterns. When we have a moving dataset, a Faster R-CNN algorithm is best for object detection, and in the case of the practical, this is the best technique.

The proposed system delivers the current count based on detected vehicles and determines the time based on the current density achieved by the proposals. Faster R-CNN is utilised for vehicle detection, which detects the automobile after the detection vehicle is classified. The vehicles are found once the categorisation and object detection network has passed the frame. On employing our proposed technique from a rear perspective, the detected object remains useful only up to the extent of the pedestrian crossing. However, the front view identifies only vehicles in the current lane (road) and does not include the opposite side of the road. Therefore, in the case of the back view, we retrieved the objects in front of the red lane, and in the case of the front view, we took that lane into

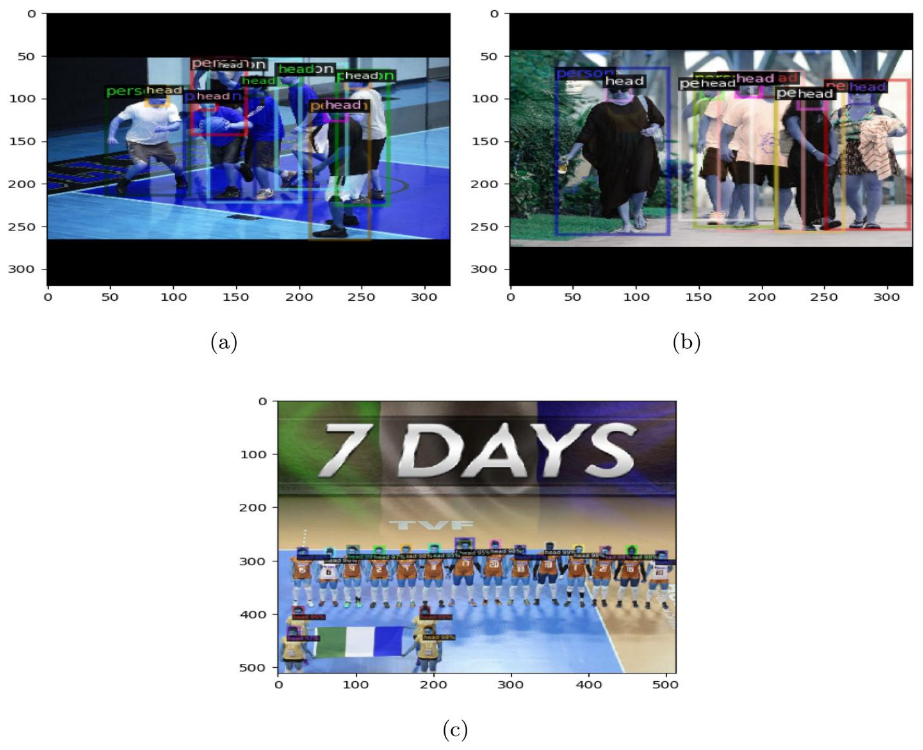


FIGURE 16 Detection results of the Crowdhuman dataset (a) Detection of head and body in the playing court, (b) detection of the people's head and body on the road and (c) detection of heads.

TABLE 3 Comparative analysis of our proposed method with state-of-the-art methods.

Paper	Year	Model	Dataset	Precision	Recall	*F*1 score	Accuracy
Chakraborty et al. [12]	2018	YOLO	Self-generated 2418 images	0.88	0.94	0.91	91.5%
Qi et al. [40]	2019	SSD	Self-generated of 24 h videos	0.76	0.70	0.73	81%
Song et al. [41]	2019	YOLOv3	Self-generated 11,129 images	0.88	0.89	0.88	94.9%
Dai et al. [23]	2019	YOLOv3	Self-generated 6134 images	0.87	0.79	0.83	87.6%
Mirthubashini et al. [24]	2020	YOLOv3	COCO	0.83	0.76	0.79	76%
Arora et al. [39]	2022	Faster R-CNN	Self-generated 3975 images	0.90	0.97	0.94	94.20%
Mittal et al. [42]	2023	Faster R-CNN + YOLOv5	KITTI and FLIR	0.95	0.96	0.93	95.4%
Our proposed method	2023	Faster R-CNN	Self-generated Punjab Safe City cameras, COCO vehicle	0.94	0.98	0.96	95.7%, 96.6% (CA)

Abbreviation: CA, class accuracy.

consideration. We assign the label associated with the detected object if the RoI of the detected object matches.

The passenger car unit (PCU) or passenger car equivalence metric is used in transportation engineering to measure traffic speed on the road. Different values are given to various vehicles depending on how much area they occupy. Table 4 displays typical PCU values used in our proposed system. Our proposed model, which uses a simple neural network design, outperforms current state-of-the-art techniques. Our suggested technique determines the current number of various cars on the road, assigns the PCU values of each vehicle to the traffic signal time, and then adjusts the signal times appropriately.

Our proposed model has some limitations. For example, we need large labelled image data to train the model and high computational resource requirements for training. The system may struggle when vehicles occlude partially or fully with other objects such as buildings, trees etc. Our method is sometimes

TABLE 4 PCU values of different types of vehicles.

Vehicle type	PCU value
Motorcycle	0.25
Qing-qi/auto rickshaw	0.6
Bicycle	0.15
Passenger car/pickup/jeep	1
Large bus/coaster/hiace	3
Cranes/tractor trolly	3
Animal driven cart	4

Abbreviation: PCU, passenger car unit.

also prone to overlapping objects coming in the Smart City cameras, where we struggle to detect the objects. An adversarial attack can also force our method to misclassify the objects, which is also one of the failure cases for our proposed method.

The computational complexity of the proposed methodology, similar to other deep learning approaches, is influenced by various factors such as the input image size, the number of RoI, the depth of the backbone network, and the number of classes. In our methodology details, we have comprehensively described each of these aspects and their implications on the overall computational workload.

Previous automation methodologies depend on sensors designed to identify vehicles on narrow pathways. However, they cannot count the vehicle because the system only detects it and is set to repeat the value until there is no traffic on the road, increasing the time accordingly. In contrast, our proposed system operates in real time, sets the time based on traffic density, and counts the vehicle until it stops. The proposed system takes input from four roadside locations, and there are then two cases. First, it determines whether vehicles are on all sides of the road. If vehicles are on all sides of the road, our system has set the maximum green time to 120 s. Next, it determines whether the traffic volume is at its maximum. If it is, the system increases the green time. If it is below the maximum threshold, the system decreases the time based on the current traffic volume based on the PCU value. The system will turn the light red until a vehicle arrives if there is no traffic on the road; however, once a vehicle arrives, the system will switch the signal green and function immediately.

This work presented a framework for automating the traffic light signals using an improved Faster R-CNN deep learning-based approach to local image data. Our proposed methodology for signal automation by estimating vehicle counting consisted of the following steps.

1. In the central server, taken images are communicated.
2. Cameras at the intersection are used for pre-processing the dataset.
3. Detection of vehicles and counting at each lane.
4. Line estimation with maximum count and timestamp of the current signal.
5. Predict the next duration of the signal.

7 | THREATS TO VALIDITY

The most significant threat to ITS is network attacks that specifically target the regular operational activities of devices and equipment, resulting in service delays and potential loss of information. Cybercriminals often employ malware as a common tool to disrupt various public and commercial businesses, including government offices and other infrastructures that rely heavily on connectivity and communication. External factors such as changes in regulations, policies, or infrastructure development may impact the performance and relevance of our proposed ITS solution over time. Inaccurate or imprecise data collection methods can lead to measurement bias, affecting the quality and reliability of the information used in our method.

To mitigate these threats, thorough research, testing, and evaluation are essential before implementing ITS on a large scale. It is crucial to validate the ITS using real-world data, conduct robust pilot studies, and engage stakeholders to gain valuable feedback and insights. Addressing privacy, security, and ethical concerns is crucial in ensuring public trust and acceptance of these technologies.

8 | CONCLUSION

In this work, we propose an intelligent traffic light system to solve traffic jams. Our proposed system counts the vehicles based on detected vehicles and dynamically assigns the traffic signal time according to each vehicle's PCU value. We fine-tune two different architectures for traffic classification of the image and object detection for traffic light automation. We label the local image dataset based on image classification, and based on object detection, we generate the predictive instances. The experiments show that our proposed method gives better accuracy than the traditional method. Our proposed method is a real-time traffic signal automation system that uses Faster R-CNN for vehicle detection. Based on the labelled dataset, our proposed method yields the vehicle's current count based on vehicle density. After getting a current count of vehicles on all sides of the road, the system takes live streaming of CCTV cameras placed on the traffic signal. There is communication between central servers, and then, based on density, it will dynamically set the time based on each vehicle's PCU value.

Our proposed work is helpful for ITS by automating traffic signals. The proposed traffic light management system based on real-time data solves many problems by saving resources such as time, energy, fuel etc. It also saves economic costs, which will be helpful for any country's economy. In the future, we aim to propose efficient methods using YOLOv8 for better detection performance and enhancing the traffic management system. We will also propose methods to track the vehicles properly and measure the vehicle's current speed.

ACKNOWLEDGEMENTS

This work was supported by the National Key R&D Program of China under Grant (2022YFC3303600), the National Natural Science Foundation of China under Grant (62077015), and the Natural Science Foundation of Zhejiang Province under Grant (LY23F020010).

CONFLICT OF INTEREST STATEMENT

The authors declare no conflicts of interest.

DATA AVAILABILITY STATEMENT

Data will be made available on request.

ORCID

Jia Zhu https://orcid.org/0000-0002-5959-390X
Muhammad Umair Hassan https://orcid.org/0000-0001-7607-5154

REFERENCES

1. Rani, P., Sharma, R.: Intelligent transportation system for internet of vehicles based vehicular networks for smart cities. Comput. Electr. Eng. 105, 108543 (2023). https://doi.org/10.1016/j.compeleceng.2022.108543
2. Avşar, E., Avşar, Y.Ö.: Moving vehicle detection and tracking at roundabouts using deep learning with trajectory union. Multimed. Tool. Appl. 81(5), 6653–6680 (2022). https://doi.org/10.1007/s11042-021-11804-0
3. Tseng, F.-H., et al.: Congestion prediction with big data for real-time highway traffic. IEEE Access 6, 57311–57323 (2018). https://doi.org/10.1109/access.2018.2873569
4. Traffic index ranking | TomTom traffic index. https://www.tomtom.com/traffic-index/ranking/. Accessed 02/01/2024
5. Imisiker, S., Tas, B.K.O., Yildirim, M.A.: Stuck on the road: traffic congestion and stock returns. Available at SSRN 2933561
6. Chowdhury, M., et al.: A Novel Approach to Forecast Traffic Congestion Using CMTF and Machine Learning. Ph.D. thesis, BRAC University (2018)
7. Xu, H., et al.: Efficient CityCam-to-edge cooperative learning for vehicle counting in ITS. IEEE Trans. Intell. Transport. Syst. 23(9), 16600–16611 (2022). https://doi.org/10.1109/tits.2022.3149657
8. Habtemichael, F., Egro, A., Krile, B., et al.: Crime Prevention for Truckers Study, Tech. Rep. United States. Department of Transportation. Federal Motor Carrier Safety … (2022)
9. Osman, T., et al.: Intelligent traffic management system for a cross-section of roads using computer vision. In: 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), pp. 1–7. IEEE (2017)
10. Han, K., Shah, S.H.H., Lee, J.W.: Holographic mixed reality system for air traffic control and management. Appl. Sci. 9(16), 3370 (2019). https://doi.org/10.3390/app9163370
11. Shah, S.H.H., Han, K., Lee, J.W.: Real-time application for generating multiple experiences from 360° panoramic video by tracking arbitrary objects and viewer's orientations. Appl. Sci. 10(7), 2248 (2020). https://doi.org/10.3390/app10072248
12. Chakraborty, P., et al.: Traffic congestion detection from camera images using deep convolution neural networks. Transport. Res. Rec. 2672(45), 222–231 (2018). https://doi.org/10.1177/0361198118777631
13. Khalifa, O.O., et al.: Vehicle detection for vision-based intelligent transportation systems using convolutional neural network algorithm. J. Adv. Transport. 2022, 1–11 (2022). https://doi.org/10.1155/2022/9189600
14. Jiang, J., et al.: Lane-level vehicle counting based on V2X and centimeter-level positioning at urban intersections. Int. J. Intell. Transport. Syst. Res. 20(1), 11–28 (2022). https://doi.org/10.1007/s13177-021-00271-4
15. Li, C., et al.: YOLOv6 v3. 0: a full-scale reloading. arXiv preprint arXiv:2301.05586
16. Taşyürek, M.: ODRP: a new approach for spatial street sign detection from EXIF using deep learning-based object detection, distance estimation, rotation and projection system. Vis. Comput. 40(2), 1–21 (2023). https://doi.org/10.1007/s00371-023-02827-9
17. Pazhani, A.A.J., Vasanthanayaki, C.: Object detection in satellite images by faster R-CNN incorporated with enhanced ROI pooling (FrRNet-ERoI) framework. Earth Sci. Inf. 15(1), 553–561 (2022). https://doi.org/10.1007/s12145-021-00746-8
18. Gupta, C., et al.: A novel finetuned YOLOv6 transfer learning model for real-time object detection. J. Real-Time Image Process. 20(3), 42 (2023). https://doi.org/10.1007/s11554-023-01299-3
19. Mscoco, V.: Vehicles-COCO dataset, visited on 2022-12-31. https://universe.roboflow.com/vehicle-mscoco/vehicles-coco (2022)
20. Gomaa, A., et al.: Faster CNN-based vehicle detection and counting strategy for fixed camera scenes. Multimed. Tool. Appl. 81(18), 1–29 (2022). https://doi.org/10.1007/s11042-022-12370-9
21. Guo, M., et al.: A reinforcement learning approach for intelligent traffic signal control at urban intersections. In: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pp. 4242–4247. IEEE (2019)
22. Alghyaline, S., Hsieh, J.-W., Chuang, C.-H.: Video action classification using symmelets and deep learning. In: 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 414–419. IEEE (2017)
23. Dai, Z., et al.: Video-based vehicle counting framework. IEEE Access 7, 64460–64470 (2019). https://doi.org/10.1109/access.2019.2914254
24. Mirthubashini, J., Santhi, V.: Video based vehicle counting using deep learning algorithms. In: 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), pp. 142–147. IEEE (2020)
25. Zhu, J., et al.: Urban traffic density estimation based on ultrahigh-resolution UAV video and deep neural network. IEEE J. Sel. Top. Appl. Earth Obs. Rem. Sens. 11(12), 4968–4981 (2018). https://doi.org/10.1109/jstars.2018.2879368
26. Qi, B., Kang, L., Banerjee, S.: A vehicle-based edge computing platform for transit and human mobility analytics. In: Proceedings of the Second ACM/IEEE Symposium on Edge Computing, pp. 1–14 (2017)
27. Kamath B, N., et al.: TAKEN: a traffic knowledge-based navigation system for connected and autonomous vehicles. Sensors 23(2), 653 (2023). https://doi.org/10.3390/s23020653
28. Wang, X., et al.: Real-time vehicle type classification with deep convolutional neural networks. J. Real-Time Image Process. 16(1), 5–14 (2019). https://doi.org/10.1007/s11554-017-0712-5
29. Suhao, L., et al.: Vehicle type detection based on deep learning in traffic scene. Procedia Comp. Sci. 131, 564–572 (2018). https://doi.org/10.1016/j.procs.2018.04.281
30. Sridevi, T., Harinath, K., Swapna, P.: Automatic generation of traffic signal based on traffic volume. In: 2017 IEEE 7th International Advance Computing Conference (IACC), pp. 423–428. IEEE (2017)
31. Yadav, A., et al.: Adaptive traffic management system using IoT and machine learning. Int. J. Sci. Res. Sci. Eng. Technol. 6, 216–229 (2019)
32. Wu, Y., et al.: Edge computing driven low-light image dynamic enhancement for object detection. IEEE Trans. Network Sci. Eng. 10(5), 3086–3098 (2023). https://doi.org/10.1109/tnse.2022.3151502
33. Tang, J., et al.: Evolutionary game and simulation analysis of intelligent connected vehicle industry with cross-border collaborative innovation. IEEE Access 11, 17721–17730 (2023). https://doi.org/10.1109/access.2023.3245078
34. Chen, C., et al.: Enhancing the robustness of object detection via 6G vehicular edge computing. Digital Commun. Netw. 8(6), 923–931 (2022). https://doi.org/10.1016/j.dcan.2022.10.013
35. Janahan, S.K., et al.: IoT based smart traffic signal monitoring system using vehicles counts. Int. J. Eng. Technol. 7(2.21), 309–312 (2018). https://doi.org/10.14419/ijet.v7i2.21.12388
36. Shirazi, M.S., Morris, B.T.: Vision-based turning movement monitoring: count, speed & waiting time estimation. IEEE Intell. Transport. Syst. Mag. 8(1), 23–34 (2016). https://doi.org/10.1109/mits.2015.2477474
37. Khan, M.Z., et al.: Deep unified model for face recognition based on convolution neural network and edge computing. IEEE Access 7, 72622–72633 (2019). https://doi.org/10.1109/access.2019.2918275

38. N. Workspace: Human Crowd dataset, visited on 2023-07-29. https://universe.roboflow.com/new-workspace-5uval/human-crowd-vbdc9 (2021)
39. Arora, N., et al.: Automatic vehicle detection system in different environment conditions using fast R-CNN. Multimed. Tool. Appl. 81(13), 18715–18735 (2022). https://doi.org/10.1007/s11042-022-12347-8
40. Qi, B., et al.: Automated traffic volume analytics at road intersections using computer vision techniques. In: 2019 5th International Conference on Transportation Information and Safety (ICTIS), pp. 161–169. IEEE (2019)
41. Song, H., et al.: Vision-based vehicle detection and counting system using deep learning in highway scenes. Eur. Transport Res. Rev. 11(1), 1–16 (2019). https://doi.org/10.1186/s12544-019-0390-4
42. Mittal, U., Chawla, P., Tiwari, R.: EnsembleNet: a hybrid approach for vehicle detection and estimation of traffic density based on faster R-CNN and YOLO models. Neural Comput. Appl. 35(6), 4755–4774 (2023). https://doi.org/10.1007/s00521-022-07940-9

How to cite this article: Abbas, S.K., et al.: Vision based intelligent traffic light management system using Faster R-CNN. CAAI Trans. Intell. Technol. 1–16 (2024). https://doi.org/10.1049/cit2.12309