Real-Time Vehicle Queue Length Prediction Using YOLOv11 and Machine - Learning Techniques

Original Scientific Paper

Syukri Akbar
National Institute of Technology (ITN) Malang, Department of Electrical Engineering
Jalan Raya Karanglo, KM 2, Malang, Indonesia
akbarsyukri@gmail.com

Aryuanto Soetedjo*
National Institute of Technology (ITN) Malang, Department of Electrical Engineering
Jalan Raya Karanglo, KM 2, Malang, Indonesia
aryuanto@lecturer.itn.ac.id

I Komang Somawirata
National Institute of Technology (ITN) Malang, Department of Electrical Engineering
Jalan Raya Karanglo, KM 2, Malang, Indonesia
kmgsomawirata@lecturer.itn.ac.id

*Corresponding author

Abstract *– Vehicle queue length is an important parameter in traffic congestion, which provides valuable information for effective traffic management and control. The paper proposes real-time vehicle queue length prediction using a combination of computer vision techniques, specifically YOLOv11, and machine learning-based prediction. YOLOv11, a state-of-the-art object detection model, is adopted to detect vehicles captured from a closed-circuit television camera. Based on the detected vehicles, the queue length is computed and the machine learning techniques perform short-term prediction. Several machine learning techniques, including the recurrent neural network, long short-term memory (LSTM) networks, Random Forest, and eXtreme Gradient Boosting, are evaluated. The effects of the time step and introduction of weather data are also considered. The algorithms are implemented on an embedded device to evaluate their real-time performance. The experimental results show that the best prediction is achieved by the LSTM with a time step of 128 using multiple features (queue length and weather data). An evaluation across three signalized intersections (nine cameras) yields a mean absolute error of 7.10, 6.96, and 4.32 m for Intersections A, B, and C, respectively, and a root mean square error of 8.79, 8.60, and 5.27 m, respectively. These results are lower than those reported in related existing work. Furthermore, the algorithm runs with training and prediction times of 3.95 min and 7.55 ms, respectively which is sufficiently fast compared with the time interval of 10 min used in this real-time application.*

Keywords*: vehicle queue length, prediction, weather data, YOLOv11, machine learning*

Received: October 22, 2025; Received in revised form: January 16, 2026; Accepted: January 29, 2026

1. INTRODUCTION

Traffic congestion is a significant issue in urban areas because of various factors, such as road capacity, vehicle density, and queues at signalized intersections [1]. Prediction techniques can be used to estimate traffic parameters in advance, thereby enabling effective traffic management and control [2]. Recent trends in artificial intelligence and Internet of Things (IoT) technologies have led to increased research on traffic prediction tasks. A traffic flow prediction method using radio frequency identification travel data was proposed in [3]. The long short-term memory (LSTM) network was employed for traffic flow prediction using an inductive loop sensor for traffic data acquisition in [4].

Traffic flow prediction using IoT devices installed on roadways or in automobiles and historical data was proposed in [5]. The integration of traffic flow data with speed and occupancy from a multi-sensor input was used to in-

crease the traffic prediction accuracy [6]. Multi-step traffic flow prediction was proposed in [7], in which the spatio-temporal data from real-world traffic flow datasets were used for the next step of prediction. Traffic flow prediction using a large language model that integrated multimodal data with language-based representations was proposed in [8]. Traffic congestion prediction has also been investigated using LSTM [9] and transformer models [10] based on vehicular ad hoc network data. Queue length prediction using a Kalman filter (KF) from a drone-collected dataset was proposed in [11].

Vehicle detection using video images incurs lower costs than that using traditional vehicle sensors [12]. Most common vehicle detection techniques using computer vision algorithms are based on the You Only Look Once (YOLO) [13] model, such as YOLOv4 [14], YOLOv5 [15], YOLOv7 [16], [17], YOLOv8 [18], YOLOv10 [19], and YOLOv11 [20]. According to the benchmarking in [21], YOLOv11 outperforms previous versions in object detection in terms of accuracy and model efficiency, making it suitable for real-time implementation.

Although many studies have addressed traffic prediction and vehicle detection, several research gaps remain. Most existing methods estimate the vehicle queue length using either computer vision techniques (YOLO) or time-series prediction (LSTM), but these methods are applied separately. However, few works have integrated both approaches into a single framework that combines visual detection and temporal prediction. This study addresses this gap by proposing a real-time vehicle queue length prediction framework that combines YOLOv11-based queue detection and machine learning-based prediction. The proposed approach offers a traffic prediction framework that can be incorporated into an embedded intelligent transportation system.

The main contributions of this study are summarized as follows:

1. A hybrid approach is proposed that integrates computer vision (YOLOv11)-based vehicle queue detection with machine learning-based short-term traffic prediction.
2. Several machine learning models, namely deep-learning models (the recurrent neural network (RNN) and LSTM) and ensemble models (Random Forest (RF) and eXtreme Gradient Boosting (XGB)) are evaluated for suitable queue length prediction.
3. The effect of weather information is considered in the evaluation of queue length prediction errors.
4. A lightweight implementation on an embedded device is demonstrated for real-time applicability.
5. A detailed analysis of the temporal parameter, i.e., time step, affecting the prediction accuracy is presented.

The remainder of this paper is organized as follows: Section 2 presents related works. Section 3 describes the research methods. Section 4 presents the results and a discussion. Section 5 provides concluding remarks.

2. RELATED WORKS

Queue length prediction using a KF was proposed in [11]. As opposed to estimating the traffic state on the highway, it is estimated at signalized intersections. This method addresses the problem of the low penetration of vehicle data for traffic estimation in turning-movement traffic by introducing a two-stage KF. In the first stage, the KF is used to estimate and predict the turning-movement rates. The KF then uses the predicted turning-movement rates to estimate the queue length in the second stage. This approach improved queue-length estimation by 32.8% compared with the baseline without KF.

Queue length estimation using the YOLOv4 model was proposed in [14]. Instead of using roadside sensors, which require high installation and maintenance costs, this approach uses low-resolution surveillance cameras. The queue length is calculated by counting the number of stopped vehicles detected during a red signal, with a YOLO model employed to detect vehicles and Deep Simple Online Real-time Tracking (SORT) used to identify stopped vehicles. As the queue length is estimated by multiplying the number of vehicles by the average vehicle length (5 m), the method does not require camera calibration. A queue length estimation accuracy of 93% was achieved.

Real-time vehicle queue state classification based on a deep-learning approach was proposed in [22]. This method integrates vehicle detection, tracking, and speed estimation to classify the queue into three states: heavy traffic, stable flow, and free flow. The system employs the YOLOv5 object detection and SORT for vehicle tracking. The queue state is then determined by averaging the vehicle speeds. This approach offers effective, low-cost vehicle queue state classification under varying traffic conditions.

Queue length estimation using data-driven deep learning was proposed in [23]. Unlike the traditional approach that uses model-based estimation, this method uses a deep-learning algorithm that learns the traffic patterns and queue length variations to improve the estimation results. The system incorporates an ensemble deep learning approach, namely a feed-forward neural network for mapping the relationship between the data sensor and cycle-based maximum queue length, and an LSTM for exploiting the temporal variation. The experimental results showed that the approach achieved high accuracy in queue length estimation and prediction using a single loop detector installed at an intersection.

A single-channel advanced video (SCAV)-based queue length estimation was proposed in [24] to avoid multiple data sources and overcome unstable estimation. This method estimates the maximum queue

length based on the saturation flow rate and trajectory of the last vehicle in the queue. This method is suitable for typical SCAV sensors, which are commonly installed at signalized intersections. The results showed a significant improvement compared with similar works, with the mean absolute percentage error (MAPE) for the two intersections under study being 14.77% and 15.1%, respectively.

YOLOv8 was used to detect vehicles and estimate their queue length in [25]. This approach uses real-time video recorded by surveillance cameras installed at intersections to reflect the actual conditions. In this system, the queue length is calculated as the number of vehicles detected during the red signal (until 3 s before the red light ends), followed by multiplication by the passenger car units to provide the standard units of the different vehicle types (car, truck/bus, and motorcycle). The evaluation results showed accuracies of 80% to 100% and 40% to 75% for low and busy traffic conditions, respectively.

3. RESEARCH METHODS

3.1. OVERVIEW OF PROPOSED SYSTEM

The vehicle queue length is essential information that can be used by the traffic light controller to control the green light duration. The queue length may change dynamically owing to the traffic conditions. Therefore, predicting the queue length in advance is beneficial for effectively controlling traffic lights. This study addresses this issue by proposing a real-time vehicle queue length prediction method, as described below.

An overview of the proposed system is presented in Fig. 1. The YOLOv11 [26] computer-vision object-detection algorithm was used to detect vehicles from video captured using a closed-circuit television (CCTV) camera installed on an urban road (signalized intersection). The vehicle queue length was calculated based on the number of vehicles detected on a road segment. The current and historical queue length data were then processed using a prediction algorithm to estimate the queue length in the future.

As traffic flow is affected by weather conditions such as temperature, humidity, and wind speed, combining the weather information and queue length improves the prediction accuracy. Thus, as shown in Fig. 1, the vehicle queue length data were merged with weather data (temperature, humidity, and wind speed). The weather data were collected using the OpenWeather API [27]. Following feature merging, the machine-learning algorithm was used to predict the vehicle queue length. Notably, the predicted future queue length—for instance, 10 min in advance—can be used to control traffic lights effectively.

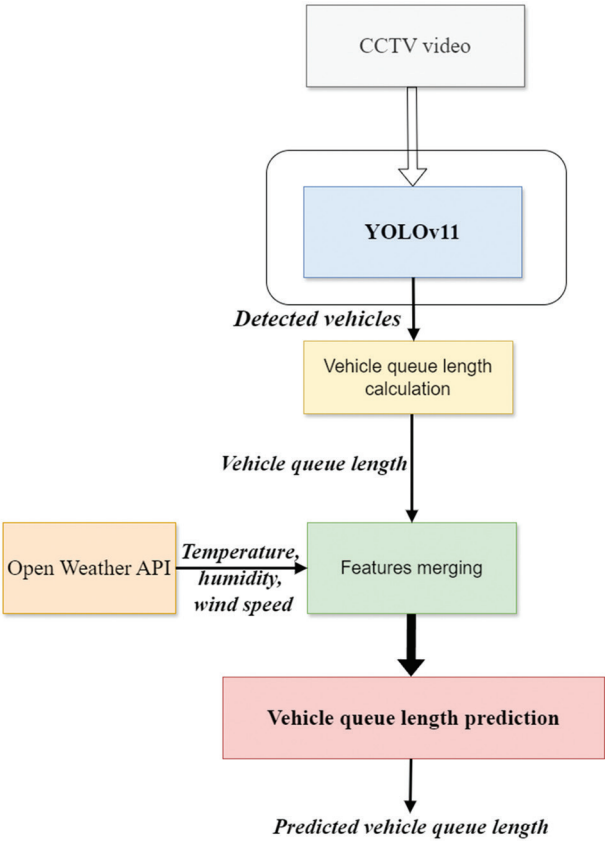

Fig. 1. System overview

3.2. HARDWARE AND SOFTWARE ARCHITECTURE

The hardware architecture of the proposed system is shown in Fig. 2. It consists of three main components: a CCTV camera, a weather sensor, and embedded systems. The Transportation Agency of Malang City offers CCTV cameras that can be accessed publicly via the Internet [28]. Weather data were collected using the freely accessible OpenWeather API. Two embedded devices were used: Raspberry Pi #1 for image capture and data collection, and Raspberry Pi #2 for vehicle queue length detection and prediction. Raspberry Pi #1 and #2 both utilize Raspberry Pi 5, which is powered by a 2.4 GHz quad-core 64-bit Arm Cortex-A76 CPU with 8 GB of RAM.

The data from the CCTV cameras and OpenWeather API were accessed using Python programming and the OpenCV library, which was implemented on Raspberry Pi #1. The images were captured from CCTV videos from 06:00 to 18:00 daily, with a time interval of 1 min. The weather data were collected every 15 min. Following data collection and merging, the data were accessed by Raspberry Pi #2 as input for the vehicle queue length detection and prediction. The vehicle queue length detection was carried out based on the YOLOv11 object-detection model. The queue length prediction was performed based on the machine learning-based prediction algorithm. Both the YOLOv11 and prediction algorithms were implemented on Raspber-

ry Pi 5 using Python programming, with the Ultralytic YOLOv11 model [26], TensorFlow [29] and sci-kit learn [30] machine-learning platforms.

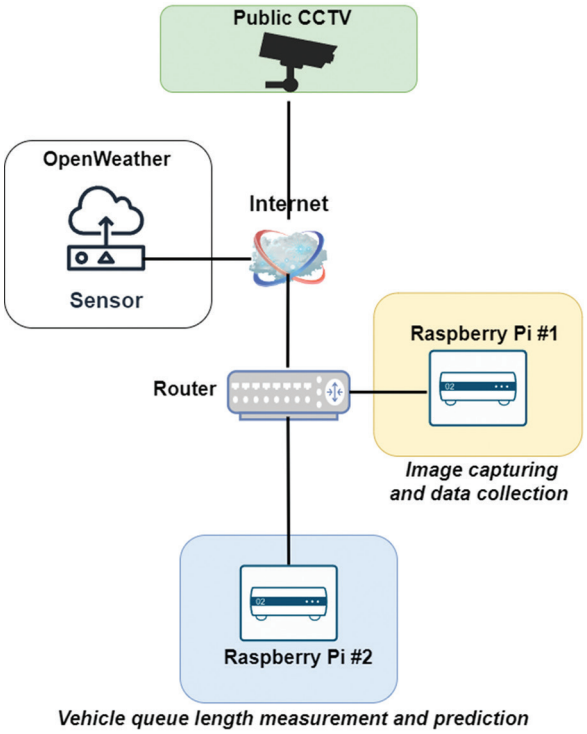

Fig. 2. Hardware architecture

3.3. VEHICLE QUEUE LENGTH MEASUREMENT

A flowchart of the vehicle queue length measurement is shown in Fig. 3. Fig. 4, which better explains the algorithm, shows the green-light check area, queue area, and projection line of a road segment at a signalized intersection. The algorithm starts with the initialization of the green light check area, queue area, and projection line. The green light area (green polygon) is the area in front of the stop line, which is used to check the green light status of the signalized intersection. When a vehicle is detected in this area, the green light for the road segment is assumed to be on. Therefore, the queue length is not calculated. The queue area (light yellow polygon) is the area in which detected vehicles are considered in the queue length calculation. The projection line (red line) is a normalized line parallel to the road segment that is used to project the detected vehicles for queue length calculation.

To handle overlapping vehicles, the detected vehicle in the queue area is represented by its bounding box center. When vehicles overlap in the image, their projected center coordinates may coincide or lie very close to each other along the projection line. Then the queue length is calculated as the distance between the foremost and rearmost projected vehicle positions.

Following initialization, the system reads the CCTV image, which is used as the input for the YOLOv11 object detection. YOLOv11 is a state-of-the-art object-detection model. It propagates an entire image in a single forward pass through a convolutional neural network-based object detection algorithm, which processes the entire image in a single stage.

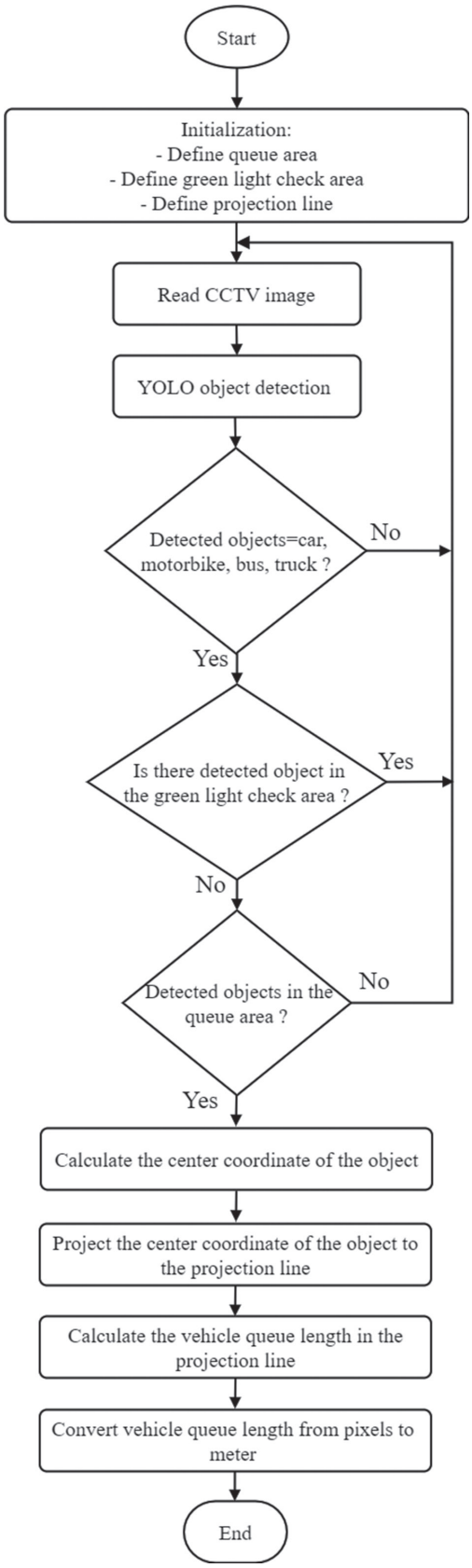

Fig. 3. Flowchart of vehicle queue length measurement

Thus, it offers fast and accurate object detection that is suitable for real-time implementation. In this study, YOLOv11 with a pretrained model was adopted. Using this model, the detected objects of interest—that is, cars, motorbikes, buses, and trucks—could be easily filtered.

Because the road segment shown in Fig. 4 is located at a signalized intersection, the vehicle queue length is calculated when the red traffic light is on (the green light is off). A simple approach was adopted to verify the traffic light status by checking the vehicles in the green light check area. As described previously, when the green light check area is empty, the red light is on and the vehicle queue length is calculated. Otherwise, the algorithm skips the queue length calculation and proceeds to read the next image.

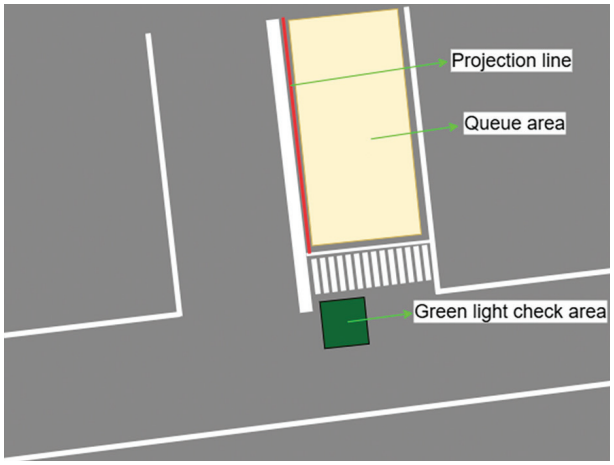

Fig. 4. Illustration of the queue area, green light check area, and projection line

The queue length is calculated by detecting vehicles in the queue area. When a vehicle is detected in this area, its center coordinates are calculated based on the detected bounding box. As the road segment orientation is not always horizontal or vertical, the projection line, as shown in Fig. 4, is required to align the center coordinates of the vehicle parallel to the road segment. Once the coordinates are projected, the queue length is calculated using the projected coordinates in the projection line. Finally, the queue length in pixel units is converted to meter units using a calibrated pixel-to-meter conversion factor. In the experiment, this parameter is obtained by selecting a reference distance along the projection line on the road segment. Then the corresponding distance in pixel units is measured along the same projection line. The pixel-to-meter parameter is then computed as the ratio of the known real-world distance (in meters) to the measured pixel distance.

3.4. VEHICLE QUEUE LENGTH PREDICTION

A block diagram of the queue length prediction is shown in Fig. 5. The machine learning model was adopted as the prediction algorithm. As shown in Fig. 5, four parameters, namely queue length, temperature, humidity, and wind speed, were used as the input features for the machine-learning model. These input features are historical data that are collected from T–$n\Delta T$, T-(n-1)ΔT, …, T, where T is the current time, ΔT is the time interval, and n is the time step. The prediction algorithm predicts the queue length in $T+\Delta T$ based on the historical data.

As described previously, queue length prediction can be used used to address changes in traffic conditions. A more detailed explanation is provided below. The adaptive traffic light controller adjusts the green light duration based on the traffic conditions. The duration of a green light on a road segment will be longer when the vehicle queue length is longer, and vice versa. A common approach is to measure the queue length during the red light and adjust the green light duration for the next cycle. However, the traffic flow in the next cycle may differ from the current flow owing to traffic fluctuations. Therefore, adjusting the green light duration is ineffective. This suggests that using the predicted queue length in the next cycle to adjust the green-light duration accordingly is preferable, as proposed in this study

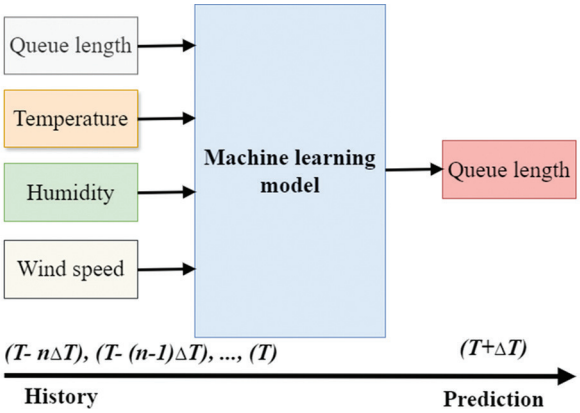

Fig. 5. Block diagram of queue length prediction

The mean absolute error (MAE) and root mean square error (RMSE), which are expressed in Eqs. (1) and (2), respectively, were calculated to evaluate the performance of the queue length prediction algorithm.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right| \quad (1)$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2} \quad (2)$$

where y, $\hat{y}$, n represent the actual value, predicted value, and number of samples, respectively.

3.5. MACHINE LEARNING-BASED PREDICTION

In this study, two types of machine-learning models, namely deep learning and ensemble models, were evaluated. The deep learning models employed time-series prediction algorithms, including RNNs and long

short-term memory (LSTM) networks. In contrast, the ensemble models utilized RF and XGB, as described briefly in this section.

RNNs are a class of deep-learning models designed to handle sequential data [31]. As opposed to the feed-forward neural network, the RNN has a feedback mechanism that allows the hidden layer to memorize the information from previous time steps to infer the output. Thus, it is effective for time-series prediction, speech recognition, and natural language processing.

One drawback of the RNN is the vanishing gradient problem, in which the gradients become smaller, which means that the network cannot be trained over long sequences. To overcome this problem, the LSTM network was proposed [32]. The LSTM architecture consists of a cell state and a gating mechanism with three gates, namely the input, forget, and output gates. This architecture enables the network to learn long-term temporal relationships.

RF is a tree-based ensemble machine learning method that was proposed by [33]. It is a class of decision tree algorithms that uses ensembles of decision trees and takes the average of the results to improve the prediction accuracy. RF is effective in modeling the nonlinearity of the input features.

XGB is an ensemble machine learning algorithm proposed by [34], which is based on the gradient boosting machine. The algorithm builds the trees sequentially, minimizing the residual error through regularization and gradient optimization. It is a high-speed, efficient, and scalable prediction algorithm.

4. RESULTS

4.1. DATA COLLECTION

The CCTV cameras used in the experiments were provided by the Transportation Agency of Malang City [28]. Two datasets were used: the primary and secondary datasets. The primary dataset was collected over one month at an intersection and was intended to evaluate the full proposed techniques and features, such as different machine learning algorithms and time steps. The secondary dataset was collected over one week at three intersections. This dataset was used to evaluate the best prediction algorithm determined previously at different intersections.

For the primary dataset, the cameras were installed at a signalized intersection located at a latitude of -7.941737 and longitude of 112.642123 (named Intersection A), as shown in Fig. 6. Three CCTV cameras were used, as illustrated in the figure. CCTV-1, CCTV-2, and CCTV-3 captured the road segments indicated by the red, blue, and yellow arrows, respectively.

As described previously (see Fig. 2), the CCTV images were streamed and recorded on Raspberry Pi #1. The data were recorded daily from 06:00 to 17:59 at 1-min intervals for one month (06/08/2025 to 06/09/2025). The image resolution was 1280 x 720 pixels. The primary dataset is summarized in Table 1, which shows of the attributes (time stamp, queue length, temperature, humidity, and wind speed) and numbers of data entries for each camera. Notably, owing to network traffic or other factors, some images were not recorded. Therefore, the number of data entries in CCTV-1, CCTV-2, and CCTV-3 differed. Weather data from the OpenWeather API [27] were simultaneously recorded at 15-min intervals and merged with the image dataset.

Fig. 6. Location map of CCTV cameras

Table 1. Primary dataset description

Camera	Attributes	Numbers of data entries
CCTV-1	Time stamp (hour : minute) Queue length (m) Temperature (°C) Humidity (%) Wind speed (m/s)	17641
CCTV-2		17760
CCTV-3		17746

For the secondary dataset, the cameras were installed at three signalized intersections: Intersection A (the same as in the primary dataset), Intersection B, located at a latitude of -7.975767 and longitude of 112.620560, and Intersection C, located at a latitude of -7.950777 and longitude of 112.631633. Intersections A, B, and C represent provincial (heavy traffic), arterial (heavy/medium traffic), and city (low traffic) roads, respectively.

Table 2. Secondary dataset description

Camera	Attributes	Numbers of data entries
CCTV-1	Time stamp (hour : minute) Queue length (m) Temperature (°C) Humidity (%) Wind speed (m/s)	3872
CCTV-2		3784
CCTV-3		3434
CCTV-4		4188
CCTV-5		4261
CCTV-6		3942
CCTV-7		2910
CCTV-8		3459
CCTV-9		3758

The secondary dataset is described in Table 2, where CCTV-1–CCTV-3 belong to Intersection A, CCTV-4 and CCTV-5 belong to Intersection B, and CCTV-6–CCTV-9 belong to Intersection C. Similar to the primary dataset,

the data were recorded from 06:00 to 17:59 at 1-min intervals for one week (04/01/2026 to 10/09/2026). It should be noted that, although Intersection B had four legs, only two cameras were reliable for use in this study.

4.2. RESULTS OF QUEUE LENGTH MEASUREMENT

Sample images of the vehicle queue length measurements during the red and green light periods are shown in Figs. 7 and 8, respectively. In the figures, the blue and orange rectangles represent the area of the detected vehicles (queue area) and green light check area, respectively. The green boxes represent the bounding boxes of the detected vehicles. The red dots represent the center coordinates of the detected vehicles. Fig. 7 shows that the green light check area is empty, indicating a red light period. Therefore, the queue length should be calculated. As shown in the top left of the figure, the queue length is 62 m. In contrast, in Fig. 8, the green light check area is occupied by vehicles, which indicates a green light period. Under this condition, queue length measurement was not performed. Using this approach, the queue length during the red light period could be calculated effectively without requiring traffic light information from the respective traffic-light control system.

Fig. 7. Sample image of vehicle queue length measurement during the red-light period

Fig. 8. Sample image of vehicle queue length measurement during the green-light period

4.3. EVALUATION OF QUEUE LENGTH PREDICTION

The machine learning parameters used for the evaluation are listed in Tables 3 and 4, which present the RNN and LSTM, and RF and XGB parameters, respectively. The algorithms were tested under different numbers of features and time steps. The evaluation of the number of features consisted of single feature (queue length) and multiple features (queue length, temperature, humidity, and wind speed). The variations in the time steps were 8, 32, 64, and 128 min. The resample interval was 10 min for all scenarios.

Table 3. RNN and LSTM parameters

Parameter	Value	
	RNN	LSTM
No. of units	64	32
Optimizer	Adam	Adam
Loss function	Mean square error (MSE)	Huber
Batch size	32	32
No. of epochs	50	50

Table 4. RF and XGB parameters

Parameter	Value	
	RF	XGB
No. of estimators	200	200
Max. depth	12	5
Min. samples to split	4	-
Min. samples in leaf	2	-
Max. features to split	Square root of features	-
Learning rate	-	0.05
Sub-sample	-	0.8
Fraction of features	-	0.8
Min. loss reduction	-	0.1
Loss function	-	MSE

Tables 5 to 8 present the results of the queue length prediction error of the single feature for the RNN, LSTM, RF, and XGB, respectively. Meanwhile, Tables 9 to 12 present the prediction errors of multiple features for the RNN, LSTM, RF, and XGB, respectively. In the tables, the MAE and RMSE for CCTV-1, CCTV-2, CCTV-3, and the average of the cameras are presented for the time-step variation (8, 32, 64, 128).

Table 5. RNN prediction errors for single feature

Time step	CCTV-1		CCTV-2		CCTV-3		Average	
	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)
8	6.31	8.12	7.25	9.51	9.01	11.54	7.52	9.72
32	6.51	8.37	7.37	9.61	9.10	11.60	7.66	9.86
64	6.47	8.36	7.35	9.62	9.04	11.54	7.62	9.84
128	6.26	8.04	7.33	9.69	9.14	11.73	7.58	9.82

Table 6. LSTM prediction errors for single feature

Time step	CCTV-1		CCTV-2		CCTV-3		Average	
	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)
8	6.26	8.08	7.19	9.38	8.97	11.52	7.47	9.76
32	6.23	8.04	7.26	9.46	9.17	11.82	7.55	9.91
64	6.21	7.97	7.12	9.36	8.94	11.52	7.42	9.73
128	6.11	7.83	7.09	9.32	9.02	11.64	7.41	9.71

Table 7. RF prediction errors for single feature

Time step	CCTV-1		CCTV-2		CCTV-3		Average	
	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)
8	6.24	8.12	7.15	9.45	9.08	11.87	7.49	9.81
32	6.43	8.18	7.28	9.46	8.98	11.73	7.56	9.79
64	6.61	8.33	7.37	9.53	9.07	11.74	7.68	9.87
128	6.59	8.21	7.49	9.67	9.33	12.10	7.80	9.99

Table 8. XGB prediction errors for single feature

Time step	CCTV-1		CCTV-2		CCTV-3		Average	
	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)
8	6.21	8.03	7.13	9.35	8.99	11.73	7.44	9.70
32	6.20	8.02	7.12	9.35	9.04	11.76	7.45	9.71
64	6.21	8.02	7.13	9.31	8.97	11.60	7.44	9.64
128	5.86	7.62	7.07	9.31	9.05	11.80	7.33	9.58

Table 9. RNN prediction errors for multiple features

Time step	CCTV-1		CCTV-2		CCTV-3		Average	
	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)
8	6.48	8.24	7.37	9.60	9.06	11.62	7.64	9.82
32	6.68	8.57	7.52	9.82	9.26	11.97	7.82	10.12
64	7.18	9.09	7.58	9.94	9.03	11.65	7.93	10.23
128	6.72	8.50	7.51	9.86	9.27	11.79	7.83	10.05

Table 10. LSTM prediction errors for multiple features

Time step	CCTV-1		CCTV-2		CCTV-3		Average	
	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)
8	6.17	7.96	7.09	9.37	9.52	12.37	7.59	9.90
32	5.99	7.82	7.09	9.33	8.70	11.43	7.26	9.53
64	5.96	7.79	7.13	9.28	8.79	11.54	7.29	9.54
128	5.79	7.51	7.03	9.22	8.73	11.39	7.18	9.37

Table 11. RF prediction errors for multiple features

Time step	CCTV-1		CCTV-2		CCTV-3		Average	
	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)
8	6.17	7.96	7.09	9.37	9.52	12.37	7.59	9.90
32	5.99	7.82	7.09	9.33	8.70	11.43	7.26	9.53
64	5.96	7.79	7.13	9.28	8.79	11.54	7.29	9.54
128	5.79	7.51	7.03	9.22	8.73	11.39	7.18	9.37

Table 12. XGB prediction errors for multiple features

Time step	CCTV-1		CCTV-2		CCTV-3		Average	
	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)
8	6.26	8.07	7.17	9.43	8.95	11.75	7.46	9.75
32	6.19	8.01	7.10	9.35	9.08	11.92	7.46	9.76
64	6.18	8.05	7.12	9.34	9.03	11.91	7.44	9.77
128	6.08	7.82	7.04	9.27	9.23	12.14	7.45	9.74

The results listed in Tables 5 to 12 more effectively are summarized and displayed in the bar charts in Figs. 9 to 15 for more effective evaluation. The effects of the time step variation are explained by Figs. 9 to 12. Meanwhile, a comparison of the single and multiple features is provided by Figs. 13 and Fig. 14. Four prediction algorithms are compared in each figure.

Figs. 9 and 10 show the MAEs of the single and multiple features under time step variations, respectively. The figures show that for the time-series-based prediction algorithms (RNN and LSTM), increasing the time step reduced the MAE. However, the ensemble-based algorithms, especially RF, showed the opposite behavior. This result demonstrates that, when increasing the time step, the LSTM network utilizes a longer history of data when making predictions. Thus, the network learns more contexts and dependencies, making more effective prediction. It is worth noting that, although XGB is not a time-series-based prediction algorithm, it could improve the prediction performance with an increasing time step.

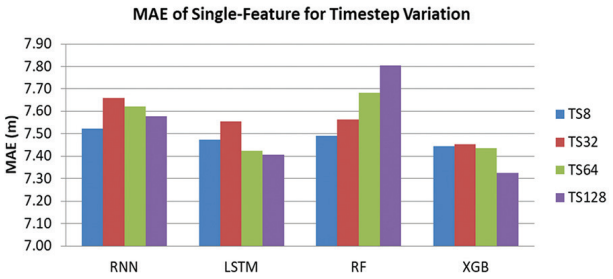

Fig. 9. MAE of single feature under time step variation

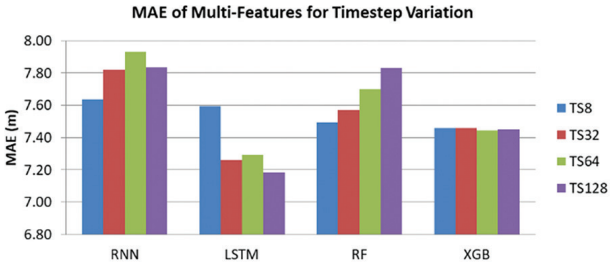

Fig. 10. MAE of multiple features under time step variation

Figs. 11 and 12 show the RMSEs of the single and multiple features under time step variations, respectively. The figures show similar behavior to that in Figs. 9 and 10. Thus, both the MAE and RMSE were reduced when the time step increased, especially when employing LSTM and XGB.

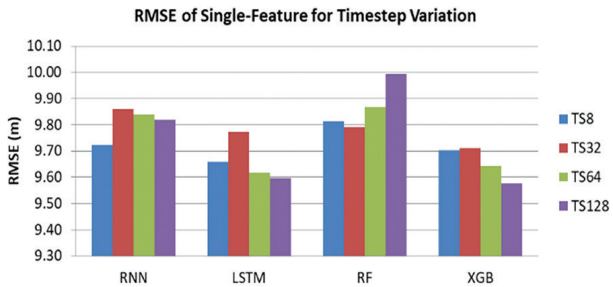

Fig. 11. RMSE of single feature under time step variation

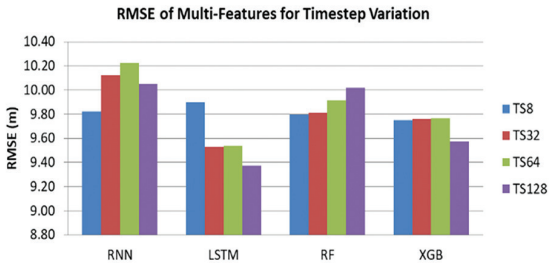

Fig. 12. RMSE of multiple features under time step variation

Figs. 13 and 14 show the MAE and RMSE for different features, respectively. In the figures, each prediction algorithm is represented by the implementation that achieved the lowest error among the four time step values as outlined in Tables 5 to 12. For instance, as the RNN with a time step of 8 had the lowest error among the time step values, it was selected for the comparison (shown as RNN-TS8), and similarly for LSTM-TS128, RF-TS8, and XGB-TS128.

Figs. 13 and 14 show that the effects of multiple features on reducing the prediction error appeared clearly in the LSTM network. However, it the prediction of the other methods was not improved. In the LSTM, the multiple features reduced the MAE from 7.41 to 7.18 m and the RMSE from 9.71 to 9.37 m. As these error reductions were small, it can be understood that only strong time-series predictions, such as LSTM, could capture the impact of the additional features, i.e., weather data. The other algorithms (RNN, RF, and RGB) failed to capture the relationships. However, the addition of weather data still contributed to enhanced prediction accuracy. In particular, the queue length measured in the signalized intersection as examined in this study was more dominantly influenced by the traffic light than by weather conditions.

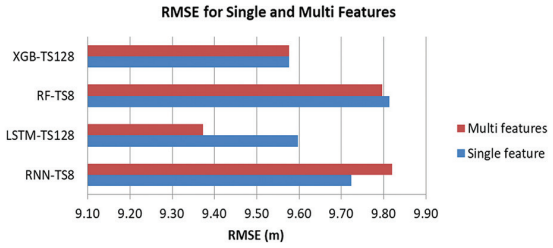

Fig. 13. MAE for different features

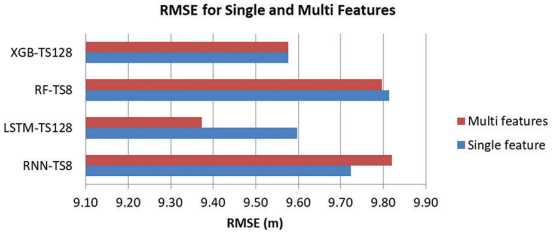

Fig. 14. RMSE for different features

Fig. 15 shows as comparison of the lowest prediction errors for the single feature and multiple features. The figure shows that the lowest MAE and RMSE were achieved by the LSTM using multiple features.

This demonstrates the superiority of LSTM in predicting time-series data and capturing relevant features, despite having only a slight impact.

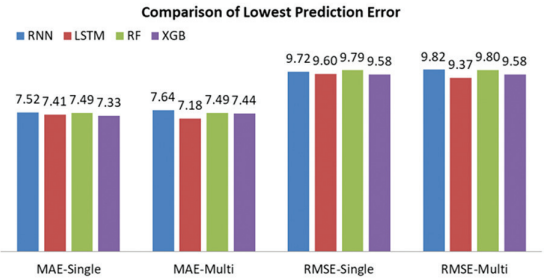

Fig. 15. Comparison of lowest prediction errors

Figs. 16 to 18 show a comparison between the actual and predicted graphs of the vehicle queue length measurements from CCTV-1, CCTV-2, and CCTV-3, respectively. The figures show the best prediction performance scheme, namely that using the LSTM with a time step of 128 and multiple features. It can be observed that the prediction profiles followed the actual profile patterns. However, they did not match when high fluctuations occurred. This is because the prediction algorithm attempted to minimize the average errors across the dataset, rather than accurately matching sudden changes.

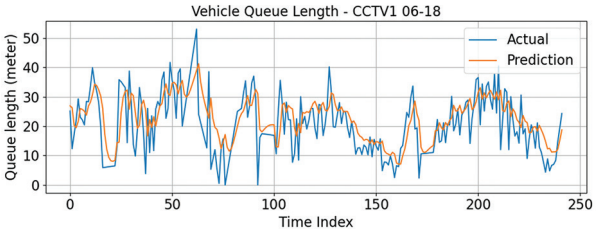

Fig. 16. Profiles of actual and predicted queue lengths for CCTV-1

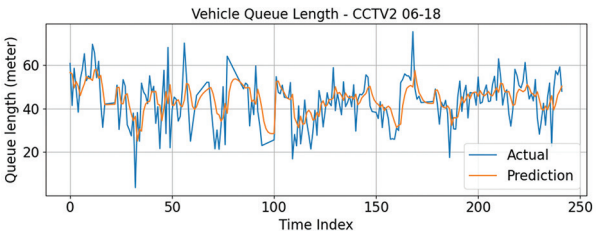

Fig. 17. Profiles of actual and predicted queue lengths for CCTV-2

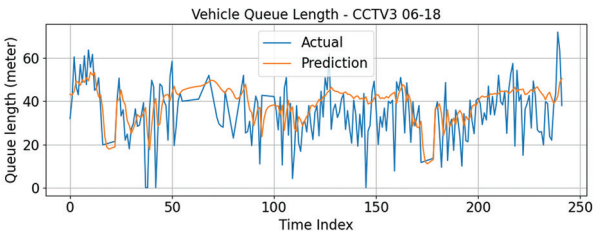

Fig. 18. Profiles of actual and predicted queue lengths for CCTV-3

To evaluate the execution time of the algorithms, the training and prediction times were calculated and are presented in Table 13. As the execution time increased according to the time step, the time step of 128 was selected for the evaluation. The LSTM exhibited the longest training and prediction times, with a training time of 236920 ms (3.95 min) and a prediction time of 7.55 ms. Given a 10-min time interval, the total training and prediction time is sufficiently fast to be implemented in real time. Therefore, it is suitable for real-time implementation at actual signalized intersections.

Table 13. Training and prediction times

Training and prediction time (ms) – (Time step = 128)							
RNN		LSTM		RF		XGB	
Train	Predict	Train	Predict	Train	Predict	Train	Predict
137163	0.6	236920	7.55	1936	0.04	1350	0.002

4.4. EVALUATION OF DIFFERENT INTERSECTIONS

To evaluate the effectiveness of the queue length prediction algorithm across different intersections, the best approach determined previously was adopted: LSTM with a time step of 128. The secondary dataset was then used for this evaluation. In the evaluation, the MAE and RMSE for the single-feature and multiple-features models were calculated across three intersections (nine cameras). The results are presented in Table 14.

Table 14 shows that, for Intersection A, the MAE and RMSE for the single- and multiple-feature models were slightly lower than those obtained using the primary dataset (Tables 6 and 10 for the single- and multiple-feature models, respectively). This result suggests the generality of the algorithm across different data sources. The lower error may have resulted from the data homogeneity and reduced variability in the smaller dataset, which captured a one-week sample, compared with the one-month samples in the primary dataset.

Table 14. LSTM prediction errors for different intersections

Camera	Single feature		Multiple features	
	MAE (m)	RMSE (m)	MAE (m)	RMSE (m)
Intersection - A				
CCTV-1	5.72	7.09	5.93	7.20
CCTV-2	6.14	7.84	6.42	8.04
CCTV-3	9.23	11.61	8.96	11.13
Average	**7.03**	**8.85**	**7.10**	**8.79**
Intersection - B				
CCTV-4	9.23	11.61	8.96	11.13
CCTV-5	4.99	6.21	4.97	6.08
Average	**7.11**	**8.91**	**6.96**	**8.60**
Intersection - C				
CCTV-6	5.99	7.05	6.05	7.10
CCTV-7	1.64	2.05	1.57	1.93
CCTV-8	7.19	8.96	7.12	9.04
CCTV-9	2.54	2.98	2.57	3.03
Average	**4.34**	**5.26**	**4.32**	**5.27**

The average prediction errors for Intersection B were close to those for Intersection A. Intersection B had a heavy/medium traffic road; thus, these results suggest that the prediction algorithm was affected by road conditions. This conclusion is supported by the errors for Intersection C, which were lower than those for Intersections A and B owing to lower traffic conditions.

Table 14 also shows that the multiple-feature models exhibited reduced errors (MAE and RMSE), although the reductions were small. These results are similar to those obtained using the primary dataset discussed previously.

4.5. COMPARISON WITH EXISTING WORKS

Table 15 presents a comparative analysis of the proposed method and several related studies with respect to the queue length measurement capability, prediction performance, and vehicle sensor modality. As opposed to several studies that estimated the queue length only [14], [24], [25], the proposed approach integrates estimation and prediction, similar to the methods of [11], [23]. However, instead of using a traditional loop detector and probe vehicle sensor, the proposed method uses a standard CCTV camera.

In terms of prediction performance, the proposed method exhibited a lower MAE (7.18 m) and RMSE (9.37 m) than the approach in [23], which reported an MAE of 11.15 m and RMSE of 16.94 m, despite the fact that the proposed method utilizes a public CCTV camera rather than the dedicated loop sensor. Compared with the method of [11], which achieved a low MAPE of 1.4%, the proposed method demonstrates competitiveness while using a more cost-effective sensor.

Table 15. Comparison with existing works

Ref.	Queue length measurement	Prediction performance	Vehicle sensor
[11]	Estimation and prediction	MAPE = 1.4%	Probe vehicle sensor
[14]	Estimation, no prediction	-	CCTV
[23]	Estimation and prediction	MAE = 11.15 m RMSE = 16.94 m	Single loop detector
[24]	Estimation, no prediction	-	SCAV
[25]	Estimation, no prediction	-	CCTV
Proposed	Estimation and prediction	MAE = 7.18 m RMSE = 9.37 m	CCTV

5. CONCLUSION

This study has proposed a prediction method for vehicle queue length that integrates computer vision and machine-learning techniques. The proposed system uses the YOLOv11 computer vision technique to detect vehicles from CCTV cameras. A simple approach was used to calculate the queue length, accounting for the red and green light durations at the signalized intersection, without requiring traffic light control system information. Several machine learning algorithms, namely

the RNN, LSTM, RF, and XGB, were evaluated for their short-term queue length prediction.

The experimental results demonstrated a promising queue length prediction system, with the LSTM outperforming the other algorithms with an MAE of 7.18 m and an RMSE of 9.37 m, both lower than those of the related existing system. Further, the proposed method, which combines vision based estimation and data-driven prediction within a single embedded system using only low-cost CCTV sensors, achieved a prediction latency of 7.55 ms. Therefore, it can be implemented in real time to support actual traffic management systems.

In the future, the proposed system will be extended to accommodate other road traffic parameters, such as traffic flow and congestion, by utilizing large datasets of CCTV images. Furthermore, integration with traffic management systems and actual implementation will be investigated.

6. ACKNOWLEDGEMENT

This work was supported by a Research Grant from the Ministry of Higher Education, Science, and Technology, Republic of Indonesia, 2025 (No.: SP DIPA-139.04.1.693320/2025, 4th Revision, 30 April 2025).

REFERENCES

[1] M. Gupta, H. Miglani, P. Deo, A. Barhatte, "Real-time traffic control and monitoring", e-Prime - Advances in Electrical Engineering, Electronics and Energy, Vol. 5, 2023, pp. 1-6.

[2] B. Gomes, J. Coelho, H. Aidos, "A survey on traffic flow prediction and classification", Intelligent Systems with Applications, Vol. 20, 2023, pp. 1-13.

[3] X. Cai, Z. Li, J. Dai, L. Lv, B. Peng, "Flow prediction of mountain cities arterial road network for real-time regulation", Journal of King Saud University - Computer and Information Sciences, Vol. 36, No. 8, 2024, pp. 1-13.

[4] F. Harrou, A. Zeroual, F. Kadri, Y. Sun, "Enhancing road traffic flow prediction with improved deep learning using wavelet transforms", Results in Engineering, Vol. 23, 2024, pp. 1-14.

[5] S. T. Mrudula *et* al. "Internet of things and optimized knn based intelligent transportation system for traffic flow prediction in smart cities", Measurement: Sensors, Vol. 35, 2024, pp. 1-9.

[6] R. Abduljabbar, H. Dia, S. Liyanage, "Machine Learning Traffic Flow Prediction Models for Smart and Sustainable Traffic Management", Infrastructures, Vol. 10, No. 7, 2025, pp. 1-23.

[7] L. He, S. Shi, D. Zhang, W. Luo, "ST-RLNet: Spatio-temporal representation learning for multi-step traffic flow prediction", Neurocomputing, Vol. 652, 2025, pp. 1-13.

[8] X. Guo *et* al. "Towards Responsible and Reliable Traffic Flow Prediction with Large Language Models", Vol. 4, 2024, pp. 1-17.

[9] C. I. Huang, J. S. Chang, J. W. Hsieh, J. H. Wu, W. Y. Chang, "xLSTM-Based Urban Traffic Flow Prediction for Intelligent Transportation Governance", Applied Sciences, Vol. 15, No 14, 2025, pp. 1-15.

[10] W. Chango *et* al. "Predicting Urban Traffic Congestion with VANET Data", Computation, Vol. 13, No. 4, 2025, pp. 1-25.

[11] A. K. Shafik, H. A. Rakha, "Real-Time Turning Movement, Queue Length, and Traffic Density Estimation and Prediction Using Vehicle Trajectory and Stationary Sensor Data", Sensors, Vol. 25, No. 3, 2025, pp. 1-27.

[12] E. Dilek, M. Dener, "Computer Vision Applications in Intelligent Transportation Systems: A Survey", Sensors, Vol. 23, No. 6, 2023, pp. 1-65.

[13] D. Doria Usta, R. Hundelshaussen, C. López Martínez, J. F. Coimbra Leite Costa, D. Machado Marques, "Intelligent Traffic Management: Comparative Evaluation of YOLOv3, YOLOv5, and YOLOv8 for Vehicle Detection in Urban Environments in Montería, Colombia", Future Transportation, Vol. 5, No. 4, 2025, pp. 1-19.

[14] M. Umair, M. U. Farooq, R. H. Raza, Q. Chen, B. Abdulhai, "Efficient video-based vehicle queue length estimation using computer vision and deep learning for an urban traffic scenario", Processes, Vol. 9, No. 10, 2021, pp. 1-19.

[15] M. I. B. Ahmed *et* al. "A Real-Time Computer Vision Based Approach to Detection and Classification of Traffic Incidents", Big Data and Cognitive Computing, Vol. 7, No. 1, 2023, pp. 1-20.

[16] C. Niu, Y. Song, X. Zhao, "SE-Lightweight YOLO: Higher Accuracy in YOLO Detection for Vehicle Inspection", Applied Sciences, Vol. 13, No. 24, 2023, pp. 1-20.

[17] H. A. Saputri, M. Avrillio, L. Christofer, V. Simanjaya, I. N. Alam, "Implementation of YOLO v7 algorithm in estimating traffic flow in Malang", Procedia Computer Science, Vol. 245, 2024, pp. 117-126.

[18] Y. Liu, S. Shen, "Vehicle Detection and Tracking Based on Improved YOLOv8", IEEE Access, Vol. 13, 2025, pp. 24793-24803.

[19] A. Ji, X. Ma, "Vehicle detection and classification for traffic management and autonomous systems using YOLOv10", Alexandria Engineering Journal, Vol. 127, 2025, pp. 804-816.

[20] Z. Zhang, Z. Zhang, G. Li, C. Xia, "ZZ-YOLOv11: A Lightweight Vehicle Detection Model Based on Improved YOLOv11", Sensors, Vol. 25, No. 11, 2025.

[21] M. L. Ali, Z. Zhang, "The YOLO Framework: A Comprehensive Review of Evolution, Applications, and Benchmarks in Object Detection", Computers, Vol. 13, No. 12, 2024, pp. 1-37.

[22] A. Battal, Y. E. Avci, A. Tuncer, "Real-Time Detection of Vehicle Queue States in Urban Traffic Using Deep Learning", El-Cezeri Journal of Science and Engineering, Vol. 12, No. 3, 2025, pp. 356-364.

[23] W. Abewickrema, M. Yildirimoglu, J. Kim, "An Ensemble Deep Learning Framework for Real-Time Queue Length Estimation at Signalized Intersections", Data Science for Transportation, Vol. 7, No. 2, 2025, pp. 1-18.

[24] P. Pudasaini, A. Karimpour, Y. J. Wu, "Real-Time Queue Length Estimation for Signalized Intersections Using Single-Channel Advance Detector Data", Transportation Research Record, Vol. 2677, No. 7, 2023, pp. 144-156.

[25] Z. Fahrunnisa, R. Rahmadwati, R. A. Setyawan, "Adaptive Traffic Light Signal Control Using Fuzzy Logic Based on Real-Time Vehicle Detection from Video Surveillance", Jurnal Ilmiah Teknik Elektro Komputer dan Informatika, Vol. 10, No. 2, 2024, pp. 235-251.

[26] G. Jocher, J. Qiu, "Ultralytics YOLO11," https://github.com/ultralytics/ultralytics (accessed: 2025)

[27] "OpenWeather API", https://openweathermap.org/api (accessed: 2025)

[28] "System Monitoring CCTV Kota Malang", http://cctv.malangkota.go.id/cameras_dua (accessed: 2025)

[29] "TensorFlow: An end-to-end platform for machine learning", https://www.tensorflow.org/ (accessed: 2025)

[30] "scikit-learn: Machine Learning in Python", https://scikit-learn.org/stable/ (accessed: 2025)

[31] Y. Lecun, Y. Bengio, G. Hinton, "Deep learning", Nature, Vol. 521, No. 7553, 2015, pp. 436-444.

[32] Felix A. Gers, J. Schmidhuber, F. Cummins, "Learning to Forget: Continual Prediction with LSTM", Neural Computation, Vol. 12, No. 10, 2000, pp. 2451-2471.

[33] L. Breiman, "Random forests", Machine Learning, Vol. 45, No. 1, 2001, pp. 5-32.

[34] T. Chen, C. Guestrin, "XGBoost: A scalable tree boosting system", Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13-17 August 2016, pp. 785-794.