DOI: https://doi.org/10.33103/uot.ijccce.24.2.4

Vehicle Classification and Counting for Traffic Analysis based on Single-stage YOLOv8 Model

Saif B. Neamah[1], Abdulamir A. Karim[2]
[1,2]Department of Computer Sciences, University of Technology, Baghdad, Iraq
[1]saif.b.neamah@uotechnology.edu.iq, [2]abdulamir.a.karim@uotechnology.edu.iq

Abstract*— Intelligent traffic systems are emerging and becoming part of smart city infrastructure that requires computer vision applications to provide traffic information like vehicle classification and counting in real-time. Vehicle counting helps detect heavy traffic on roads, and vehicle classification helps enforce further processes like speed estimation to enforce speed limit laws based on the vehicle class. Deep learning computer vision-based systems provide automatic feature extractions that are robust to changes in lighting, shadows, and occlusions. This paper proposes a software solution for a real-time traffic monitoring system based on a cutting-edge single-stage deep learning model through the state-of-the-art YOLOv8 algorithm. YOLOv8 is the most recent model of the YOLO family, which provides object detection and classification through its CNN architecture. The proposed work detects vehicles and counts them based on their class. The four common vehicle classes are sedan cars, buses, trucks, and motorcycles, and a counter for each class is displayed on the system's output screen in real-time and recorded in a log file. The results of the proposed system running on the Nvidia GTX 1070 GPU show an average accuracy of 96.58% with an average error of 3.42% for vehicle detection and an average accuracy of 97.54% with a 2.46% average error for vehicle counting. For vehicle classification, the results for the four vehicle classes (car, bus, truck, and motorcycle) show an accuracy of (94.7%, 94.7%, 96.2%, 99.7%), precision (95%, 100%, 81.4%, 100%), recall (97.9%, 36.3%, 100%, 66.6%), and the f1-score (96.3%, 53.2%, 89.7%, 79.9%), respectively.*

Index Terms*— Computer Vision, Deep Learning, Vehicle Detection, Traffic Analysis, YOLO.*

I. INTRODUCTION

The emergence of smart cities and intelligent traffic systems nowadays aims at improving human live by enabling efficient use of infrastructure resources, reducing risks like traffic collisions, and improving drivers and pedestrians' safety. Road traffic accidents are one of the leading causes of mortality globally. In Iraq, traffic accidents have increased considerably, especially since 2003, as a result of the growth in the economy and population. The number of vehicles as of 2015 was 5.775 million, according to the 2018 World Health Organization (WHO) road safety report, sourced from the statistics department of Iraq's Ministry of Health [1].

From the traffic law enforcement side, there is no clear enforcement of the speed limit traffic law on Iraqi roads. WHO ranked the speed limit enforcement law in Iraq at 2 out of 10. *Fig. 1* shows the number of road deaths from traffic accidents in the range of 12 to 18 per 100,000 people between 2007 and 2014. WHO also estimated the rate at 20.7 per 100,000 people as of 2016 [1-3].

Received 28/September/2023; Accepted 28/October/2023

DOI: https://doi.org/10.33103/uot.ijccce.24.2.4

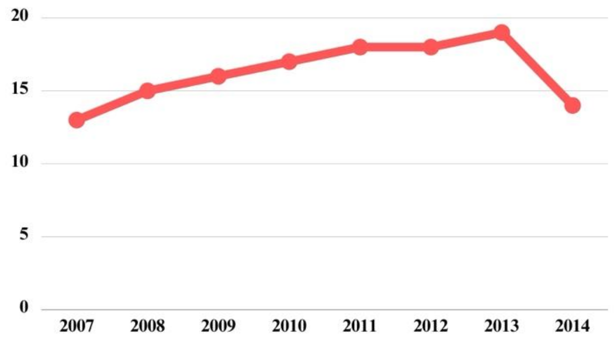

FIG. 1. ROAD DEATHS PER 100,000 PEOPLE IN IRAQ [1].

This paper implements a robust real-time traffic monitoring system that detects, classifies, and counts multiple vehicles using a cost-effective deep learning-based software solution. The proposed system mainly consists of four stages: the preprocessing stage, the vehicle detection stage, the vehicle classification stage, and the vehicle counting stage.

II. RELATED STUDIES

There are a number of studies that are conducted on vehicle detection and counting; some of these studies are listed below:

In 2019, Huangsheng Song et al. presented in their work entitled “Vision-based Vehicle Detection and Counting System Using Deep Learning in Highway Scenes” the implementation of a real-time vehicle detection and counting system. They based their work on the YOLOv3 model and ORB algorithm to determine the vehicle’s direction. The results they got show that the average accuracy of vehicle driving direction and vehicle counting is 92.3% and 93.2%, respectively [4].

In 2020, Muhammed Fachri presented in his work entitled “A Simple Vehicle Counting System Using Deep Learning with the YOLOv3 Model” the implementation of a real-time vehicle detection and counting system based on the YOLOv3 model for counting and classification. The results show an average accuracy of 97.72% [5].

In 2021, Cheng-Jian Lin et al. presented in their work entitled “A Real-time Vehicle Counting, Speed Estimation, and Classification System Based on Virtual Detection Zone and YOLO” the implementation of a real-time traffic monitoring system to count vehicles and estimate their speed with a classification facility. They based their work on the Gaussian mixture model (GMM) and YOLOv4. The GMM and a virtual detection zone are used for vehicle counting and detection. The GMM is used to perform background subtraction. The YOLOv4 model is used for vehicle classification, with a classification accuracy of 98.91% and an average absolute percentage error of vehicle speed estimation of 7.6% [6].

In 2022, Maryam Raad Shihab et al. presented in their work entitled “Machine Learning Techniques for Vehicle Detection” the implementation of a vehicle detection and classification system using two methods based on Haar cascade and YOLOv3. The detection results they got were 86.9% for the Haar cascade approach and 91.31% for the YOLOv3 approach, concluding that YOLO-based algorithms

Received 28/September/2023; Accepted 28/October/2023

DOI: https://doi.org/10.33103/uot.ijccce.24.2.4

have better detection results than Haar cascade-based methods and are robust to different lighting conditions [7].

The previous works are based on earlier versions of the YOLO algorithm; faster and more accurate models are needed for real-time traffic monitoring systems. The classification process of the previous works did not classify cars, while our work classifies cars into four sub-classes (sedan, bus, truck, and motorcycles).

III. TRADITIONAL TRAFFIC MONITORING SYSTEMS

A range of hardware technologies are available for the purpose of gathering traffic data to support traffic surveillance systems. These technologies include sensors, induction loops, and microwave radars. Every hardware technology has inherent restrictions. The geographical coverage of induction loops is limited to point measurements, restricting their ability to capture a comprehensive representation of the whole area. Moreover, in situations where there is a high volume of traffic, the accuracy of induction loops may deteriorate. The installation and maintenance expenses of most surface sensors are typically significant. Handheld radar guns, which operate based on the Doppler effect, have some limitations in addition to their expensive equipment. These limitations include the need for an operator to be present at the location and a direct line of sight for accurate speed calculation. Furthermore, these devices are only capable of assessing the detection of one vehicle at a time. Additionally, these radar systems experience the phenomenon of shadowing, when several waves are reflected from vehicles of varying heights [8].

IV. DEEP LEARNING-BASED TRAFFIC MONITORING SYSTEMS

Deep learning and convolutional neural network (CNN)-based methodologies have shown the ability to automatically extract features from video frames. These approaches exhibit enhanced resilience to variations in lighting, shadows, and partial occlusions. Two methodologies, the single-stage approach and the two-stage approach, dominated the field of object detection. Single-stage detectors, such as the You Only Look Once (YOLO) models, tackle the task of object identification by treating it as a regression issue. These models directly predict the bounding box coordinates and object classes, and they analyze the data in a single forward pass. This characteristic makes them well suited for real-time applications. Nevertheless, two-stage detectors, such as the region-based convolutional neural network (R-CNN), consist of two distinct stages. To begin the search process, it is important to generate a substantial quantity of area proposals by using a search methodology, such as a selective method or a region proposal network. The next stage involves the submission of these area ideas for the purposes of categorization and bounding box regression. Two-stage detectors often exhibit higher detection rates compared to one-stage detectors. However, this advantage is accompanied by increased processing time due to the additional processing involved [9][10].

The convolutional neural network (CNN) is a widely used deep learning architecture that can learn directly from raw data without the need for manual human feature extraction. CNNs include multiple convolution and pooling layers; they are specifically intended to deal with a variety of 2-dimensional shapes and are widely employed in the applications of computer vision, image segmentation, and object detection. The rapid development of GPU technology made CNNs so popular. In fact, one of the bottlenecks of deep neural networks

Received 28/September/2023; Accepted 28/October/2023

DOI: https://doi.org/10.33103/uot.ijccce.24.2.4

is that training takes a long time because of the many hidden units in the network. But as GPUs became faster, this bottleneck was overcome. In CNNs, the states of each layer are arranged according to a spatial grid structure; these spatial relationships are inherited from one layer to the next because each feature value is based on a small local spatial region in the previous layer. Each layer in the convolutional network is a 3-dimensional grid structure that has a height, width, and depth. The depth of a single layer refers to the number of channels in each layer. *Fig. 2* demonstrates the traditional CNN model [11-14].

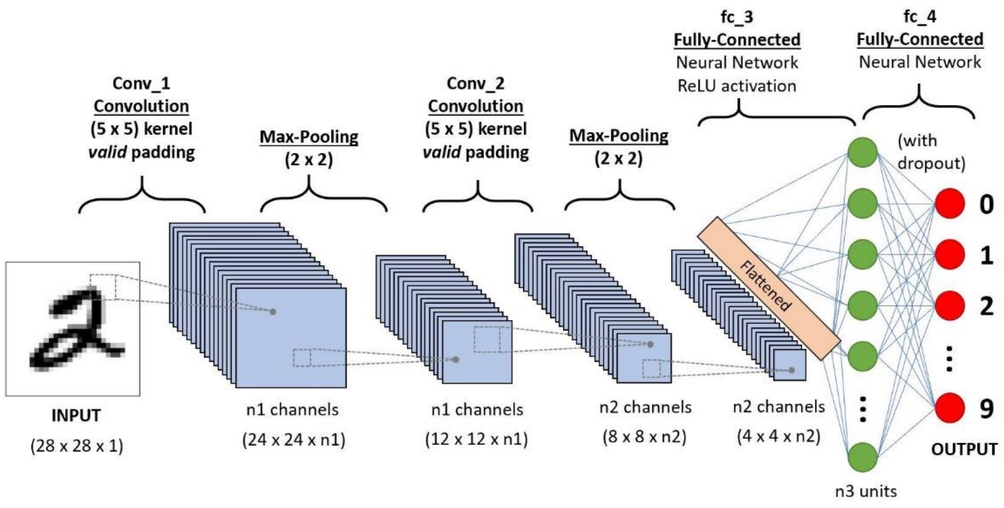

FIG. 2. TRADITIONAL CNN MODEL [12].

V. CNN-BASED OBJECT DETECTION

The problem of object detection can be efficiently solved with deep learning models. Object detection based on CNNs consists of two main tasks: recognizing and locating objects in the image. Recognition is a classification task that involves providing category information and the probability of the target. The other is a positioning task that involves finding the specific location of the target by utilizing bounding boxes with labels. There are various algorithms for object detection using CNNs, which are mainly divided into two main categories: ***Two-stage algorithms*** like the R-CNN series generate at the first stage regions of interest (ROIs) that represent a set of category-independent bounding boxes in the image; at the second stage, they make corrections based on the bounding box region to improve the final detection results. The two-stage algorithms give more accurate results, but they are more computationally expensive as they require the classification network to be applied to a large number of ROIs, so this approach is slower than the single-stage algorithms [15].

Single-stage algorithms: merge the classification and regression into a single pass, like the YOLO series, which divides the image into a fixed grid and applies a classification network to each segment. The single-stage algorithms are fast but give less accurate detection results than the R-CNN algorithms [16].

VI. YOLOv8 SINGLE-STAGE CNN MODEL

YOLO is a real-time object detection algorithm that uses a single CNN to predict the bounding boxes and class labels of objects in an image. YOLOv1 was introduced in 2015 and has gone through multiple revisions and upgrades. YOLOv2 was introduced in 2016

Received 28/September/2023; Accepted 28/October/2023

DOI: https://doi.org/10.33103/uot.ijccce.24.2.4

with various improvements, including detecting over 9,000 object categories. YOLOv3 was introduced in 2018, which evolved from Darknet-19 to Darknet-53 with residual connections. YOLOv4 was introduced in 2020 with various improvements. YOLOv5 was introduced in 2021 and implemented under the PyTorch framework. YOLOv6 was introduced in 2022 with a plain single-path backbone for small models and efficient multi-branch blocks for large models [17][18].

YOLOv7 was introduced in 2022, proposed a planned re-parameterized model by merging multiple computational modules into one at the inference stage, and made some architectural reforms. YOLOv8 was introduced in 2023 and outperformed all previous models, as *Fig. 3* illustrates [19][20].

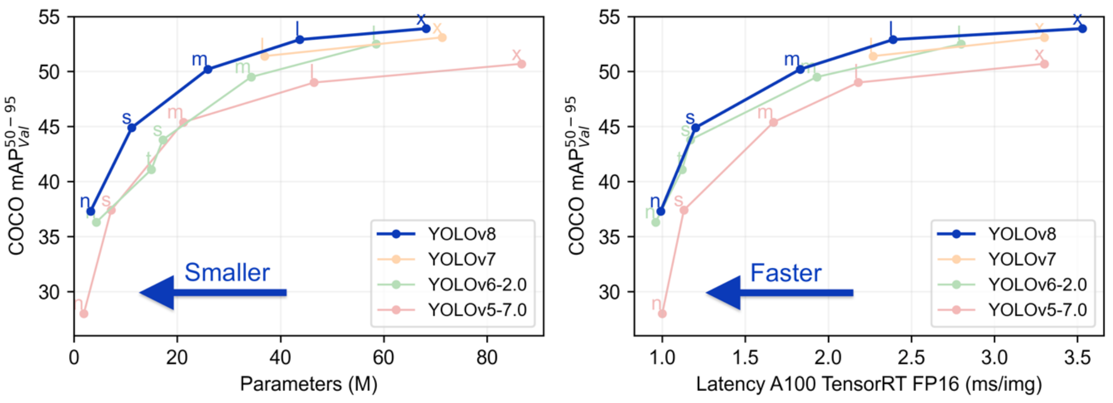

FIG. 3. YOLOV8 EVALUATION [21].

YOLOv8 proposed a new backbone network with a new anchor-free detection head, which means it predicts directly the center of an object instead of the offset from a known anchor box. It also proposes a new loss function. The basic architecture of YOLOv8 consists of two major parts: the backbone for extracting feature maps and the head for detection. The backbone contains a series of convolutional layers for different image resolutions and sizes, and then the features detected are passed through the advanced head for detection based on a loss function. New convolutional layers are used [19].

The backbone contains a series of convolutional layers for different image resolutions and sizes, and then the features detected are passed through the advanced head for detection based on a loss function. New convolutional layers are used. The stem's first (6*6) convolution is replaced by a (3*3); the main building block was changed, and C2f replaced the YOLOv5 C3. The C2f model can extract richer gradient flow information while maintaining lightweight [22][23].

YOLOv8 utilizes anchor-free detection, as the model directly predicts the center of an object instead of the offset from a known anchor box. The advantage of anchor-free detection is that it is more flexible and efficient, as it does not require the manual specification of anchor boxes, as was the case for older versions of YOLO [24][25].

Received 28/September/2023; Accepted 28/October/2023

DOI: https://doi.org/10.33103/uot.ijccce.24.2.4

VII. PORBLEM STATEMENT

With the emergence of intelligent traffic systems worldwide, software solutions for traffic monitoring became an essential part of those types of systems. Traffic monitoring systems feed the ITS with information for further traffic analysis. The more accurate the detection models, the more stable and accurate the system results will be. Deploying a deep learning-based solution is a challenging task for a real-time multi-object system because of the heavy processing it requires. Careful fine-tuning is essential for the system to produce stable results in real-time. This paper solves the above-mentioned problems by providing a deep-learning-based system for multi-object detection and classification in real-time. The system is provided with various fine-tuning parameters and flags to make it applicable to different configurations. The system performs vehicle counting based on the vehicle's class (sedan, bus, truck, or motocycle). The classification is useful for further upgrading the system, like adding a speed estimation process based on the vehicle class.

VIII. THE PROPOSED SYSTEM ARCHITECTURE

The general architecture of the proposed system consists of four stages that start with the preprocessing stage, then the detection stage for specified classes of vehicles that implement the YOLOv8 pretrained detection model, then the classification stage, and finally the vehicle counting stage. The following block diagram in *Fig. 4* illustrates a general overview of the proposed system.

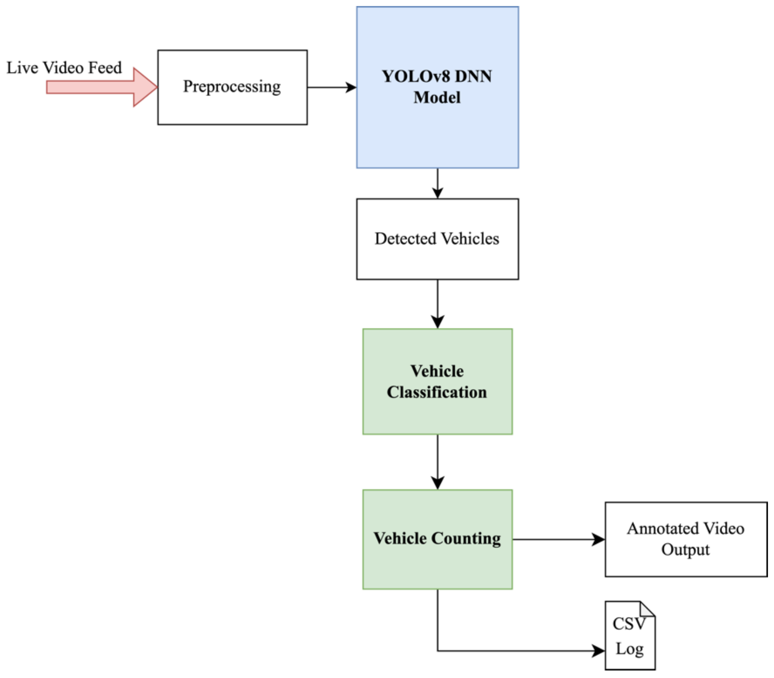

FIG. 4. THE PROPOSED SYSTEM'S BLOCK DIAGRAM.

A. The Preprocessing Stage

The first stage contains two steps, which are the ROI calculation step and the input preparation step. In the proposed system, two ROIs are specified, namely, ROI1 and ROI2, and three lines, namely, Line 1, Line 2, and Line 3, as *Fig. 5* shows.

Received 28/September/2023; Accepted 28/October/2023

DOI: https://doi.org/10.33103/uot.ijccce.24.2.4

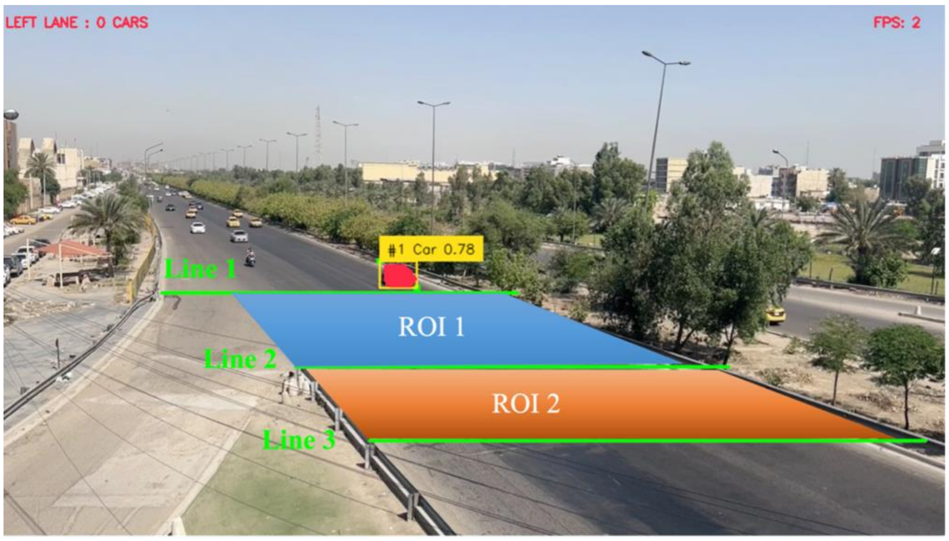

FIG. 5. THE PROPOSED SYSTEM'S ROIS.

The reason behind dividing the ROI into two regions is that we want to get the most accurate results possible, so the vehicle is counted in ROI1, and if a miss detection happens, it will be counted in ROI2.

B. The Vehicle Detection Stage

The detection stage deploys pretrained YOLOv8 CNN models. Which consists of multiple layers of convolution (Conv), coarse-to-fine (C2f), concatenation (Concat), upsampling (Upsample), and spatial pixel per feature (SPPF). The Conv layer involves the standard convolution operation of a sliding window with predefined kernels, a stride of 1, no padding, and the SiLU activation function, followed by batch normalization to improve the overall learning process. The C2f process involves a convolution operation with a kernel of size (1*1), no padding, and a stride of 1, then the output is entered into a split operation that is fed to the bottleneck; the bottleneck itself consists of two convolutional layers with residual connections. Then the concatenation operation is performed, and finally another convolution is performed. The SPPF is spatial pyramid pooling (fast), similar to the SPP used in YOLOv5, which involves two convolution layers and three max-pooling layers and is used to capture multi-scale information and improve the detection performance of the network, enabling the network to detect objects of different sizes more accurately. The upsample process involves increasing the size of the matrices by using a transposed convolution process with a stride of 2 and a padding of 1. This increases the spatial resolution of the feature maps, which helps improve object localization. The concatenation process involves the concatenation of a list of tensors into a tensor of one dimension. This will enable feature fusion, model flexibility, and the handling of multiple inputs. The general steps of the YOLOv8 model are illustrated in *Fig. 6.*

Received 28/September/2023; Accepted 28/October/2023

DOI: https://doi.org/10.33103/uot.ijccce.24.2.4

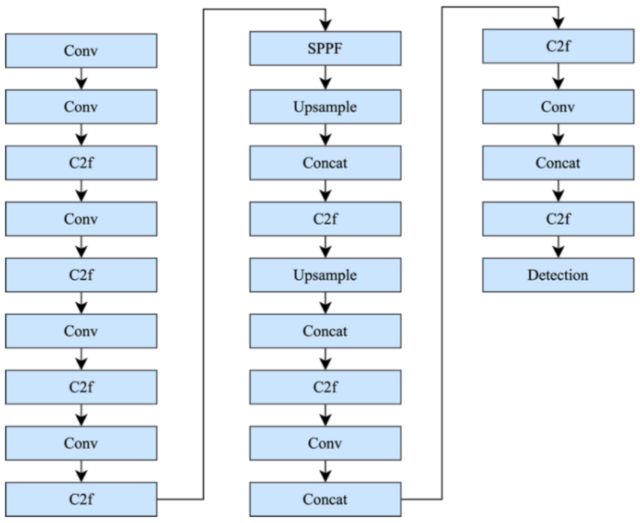

FIG. 6. YOLOV8 DETECTION MODEL.

C. The Vehicle Classification Stage

The YOLOv8 pretrained model detects objects in video frames and classifies them according to the COCO dataset into 80 object categories. Since our system works on vehicles, we need to detect only four classes that represent the most common vehicle types, namely (car, motorcycle, bus, and truck); these classes correspond to numbers 2, 3, 5, and 7, respectively. YOLOv8 will generate class scores for the four predefined detected vehicles. The classification process gives the class number of the detected object; this number is used as the index for the class name in the COCO dataset. Each class is annotated with a different color for more clarification of the classification process, as shown in *Fig. 7*.

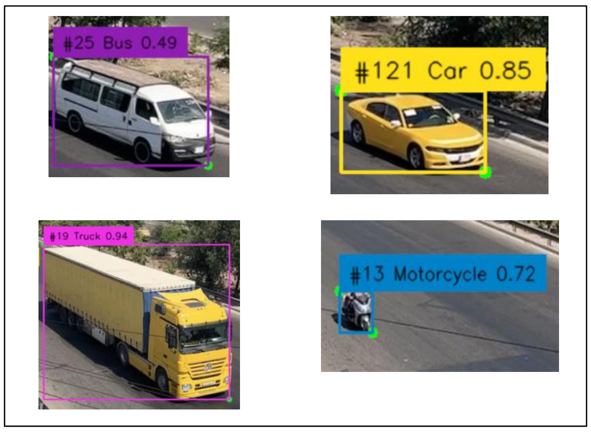

FIG. 7. VEHICLE CLASSIFICATION ANNOTATION.

D. The Vehicle Counting Stage

The vehicle counting stage counts the vehicles passing through the ROIs in real-time based on the center of the detected vehicle passing within the coordinates of the ROI lines; the center is calculated from the coordinates of the bounding box as in Equation 1.

$$(cx, cy) = \frac{(x + (x + w))}{2}, \frac{(y + (h + y))}{2} \tag{1}$$

Where (x, y) is the detected object's bounding box's top left corner coordinates. (w, h) is the detected object's bounding box width and height. (cx, cy) is the center coordinate of the detected object. The counter for each vehicle class is displayed on the video output, and the

Received 28/September/2023; Accepted 28/October/2023

DOI: https://doi.org/10.33103/uot.ijccce.24.2.4

total number of vehicles passed in the observation period for each class is recorded in the log file in CSV format. The output of the proposed system is presented as a video stream that is displayed to the viewer as long as the system is running. The stream can also be saved to the hard disk as a video file in MP4 format. The details of the vehicle counting stage are described in Algorithm 1.

IX. RESULTS AND DISCUSSION

The results of the proposed system are presented on the system's output screen with all the annotations applied to them by the multiple stages of the system. *Fig. 8* shows the proposed system's screen of results.

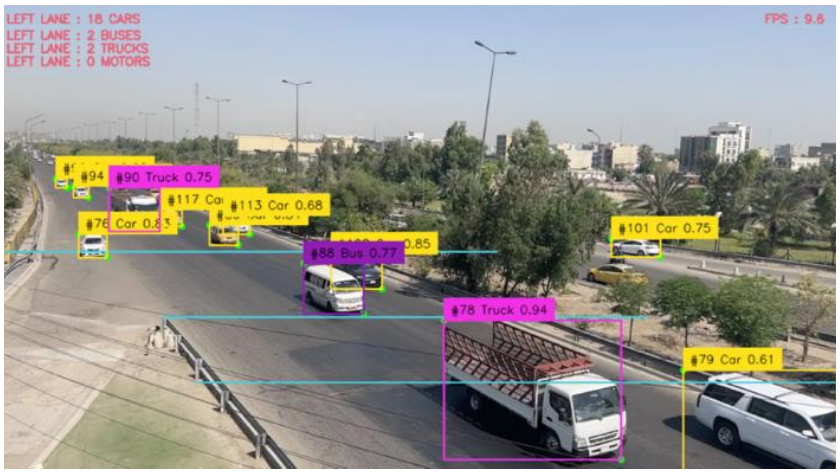

FIG. 8. THE PROPOSED SYSTEM'S OUTPUT SCREEN.

```
Algorithm 1: The Vehicle Counting Algorithm
--------------------------------------------------------------------------------
Input:  D: list of detections
Output: Car_Count, Bus_Count, Truck_Count, Motor_Count
--------------------------------------------------------------------------------
Start:
 CARS_Q = [ ];  offset = 20
 Car_Count = Bus_Count = Truck_Count = Motor_Count = 0
 COUNTERS = [Car_Count, Bus_Count, Truck_Count, Motor_Count]
Step 1:
    L1 = start(x1, y1) to end(x1, y1)                          // Line 1
    L2 = start(x2, y2) to end(x2, y2)                          // Line 2
    L3 = start(x3, y3) to end(x3, y3)                          // Line 3
Step 2: For each object in D:
      (x, y, w, h) = D.xywh
      car_id = D.obj_id
      class_id = D.class_id
Step 3: cx = (x + (x + w)) / 2; cy = (y + (y + h)) / 2
Step 4: check1 = cy <= (L1.start.y1 + offset)
     check2 = cy >= (L1.start.y1 - offset)
     check3 = cx <= L1.end.x1
     checkQ = car_id not in CARS_Q
Step 5: check4 = cy <= (L2.start.y1 + offset)
     check5 = cy >= (L2.start.y1 - offset)
     check6 = cx <= L2.end.x1
Step 6: if check1 and check2 and check3 and checkQ
        CARS_Q.append([car_id, [cx, cy]])
        Increase_counter(class_id, [cx, cy], COUNTERS)
Step 7:if check4 and check5 and check6 and checkQ
        CARS_Q.append([car_id, [cx, cy]])
        Increase_counter(class_id, [cx, cy], COUNTERS)
Step 8:
   Display(frame, Car_Count, Bus_Count, Truck_Count, Motor_Count)
    Log(Car_Count, Bus_Count, Truck_Count, Motor_Count)
 End For
End
```

Received 28/September/2023; Accepted 28/October/2023

DOI: https://doi.org/10.33103/uot.ijccce.24.2.4

The detection stage will draw a bounding box around the detected vehicles; the bounding box of each vehicle will be annotated with an ID, the class name extracted from the COCO dataset after performing classification by the CNN model, and the confidence score. The vehicle counters are displayed at the top left corner of the screen. The proposed system was evaluated on test videos taken at different locations and lengths with different camera positions to evaluate the detection, classification, and counting stages, while the system ran on an Nvidia GTX 1017 GPU with different YOLOv8 model sizes as listed in Tables I, II, III, and IV.

TABLE I. VEHICLE DETECTION EVALUATION

Test Video	Model Size	Ground-truth Vehicles	System's Detections	FPS	Accuracy	Error
Video1	Nano	70	67	48.8	95.7%	4.3%
Video1	Small	70	68	42	97.1%	2.9%
Video1	Medium	70	69	28.7	98.5%	1.5%
Video1	Large	70	69	21.5	98.5%	1.5%
Video1	Extra-Large	70	69	15.9	98.5%	1.5%
Video2	Nano	45	41	50.2	91.1%	8.9%
Video2	Small	45	43	43.8	95.5%	4.5%
Video2	Medium	45	43	29.6	95.5%	4.5%
Video2	Large	45	43	21.7	95.5%	4.5%
Video2	Extra-Large	45	44	16	97.7%	2.3%
Video3	Nano	28	25	50	89.2%	10.8%
Video3	Small	28	26	43.3	92.8%	7.2%
Video3	Medium	28	27	29.3	96.4%	3.6%
Video3	Large	28	28	21.7	100%	0%
Video3	Extra-Large	28	28	16	100%	0%
Video4	Nano	50	48	48	96%	4%
Video4	Small	50	49	41.5	98%	2%
Video4	Medium	50	49	28.5	98%	2%
Video4	Large	50	50	21	100%	0%
Video4	Extra-Large	50	50	15.6	100%	0%
Video5	Nano	37	34	46	91.8%	8.2%
Video5	Small	37	35	40	94.5%	5.5%
Video5	Medium	37	36	28	97.2%	2.8%
Video5	Large	37	36	21	97.2%	2.8%
Video5	Extra-Large	37	37	15.7	100%	0%
Averages					**96.58%**	**3.42%**

Received 28/September/2023; Accepted 28/October/2023

DOI: https://doi.org/10.33103/uot.ijccce.24.2.4

TABLE II. VEHICLE COUNTING EVALUATION

Test Video	Model Size	Ground-truth Count	System's Count	FPS	Accuracy	Error
Video1	Nano	60	61	49.4	98.3%	1.7%
Video1	Small	60	60	42.6	100%	0%
Video1	Medium	60	60	28.9	100%	0%
Video1	Large	60	61	21.5	98.3%	1.7%
Video1	Extra-Large	60	62	16	96.6%	3.4%
Video2	Nano	45	45	50.8	100%	0%
Video2	Small	45	44	43.6	97.7%	2.3%
Video2	Medium	45	43	29.3	95.5%	4.5%
Video2	Large	45	45	21.6	100%	0%
Video2	Extra-Large	45	45	16	100%	0%
Video4	Nano	40	40	47.6	100%	0%
Video4	Small	40	43	41	93%	7%
Video4	Medium	40	44	28.1	90%	10%
Video4	Large	40	43	21.1	93%	7%
Video4	Extra-Large	40	43	15.7	93%	7%
Video5	Nano	21	22	44.6	95.4%	4.6%
Video5	Small	21	21	39.8	100%	0%
Video5	Medium	21	21	27.5	100%	0%
Video5	Large	21	21	20.7	100%	0%
Video5	Extra-Large	21	21	15.5	100%	0%
Averages					**97.54%**	**2.46%**

TABLE III. MULTICLASS CONFUSION MATRIX

		Ground-truth Class			
		Car	Bus	Truck	Motorcycle
Predicted Clas	Car	**95**	5	0	0
	Bus	0	**4**	0	0
	Truck	2	2	**22**	1
	Motorcycle	0	0	0	**2**

TABLE IV. MULTICLASS CONFUSION MATRIX

Vehicle Class	Precision	Recall	Accuracy	F1-score
Car	95%	97.9%	94.7%	96.3%
Bus	100%	36.3%	94.7%	53.2%
Truck	81.4%	100%	96.2%	89.7%
Motorcycle	100%	66.6%	99.7%	79.9%

Received 28/September/2023; Accepted 28/October/2023

DOI: https://doi.org/10.33103/uot.ijccce.24.2.4

The proposed system was also tested with various illumination changes at day and night, as *Fig. 9* demonstrates the proposed system's detection capability at night.

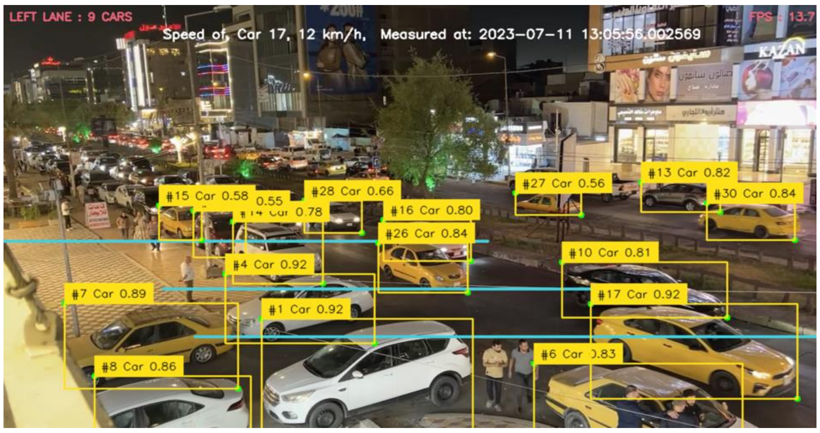

FIG. 9. THE PROPOSED SYSTEM'S NIGHT TIME DETECTION.

The results of the proposed system showed excellent estimations for the multiple system stages. For the detection stage, the results we tested on multiple video samples and different CNN model sizes showed an average accuracy of 96.58% and an average error of 3.42%. An average accuracy of 97.54% with a 2.46% average error for the counting stage. For the vehicle classification stage, the results from a single test video show an accuracy of 94.7% for the salon car class, 94.7% for the bus class, 96.2% for the truck class, and 99.7% for the motorcycle class. The precision values for the car, bus, truck, and motorcycle were 95%, 100%, 81.4%, and 100%, respectively. The recall values obtained were 97.9%, 36.3%, 100%, and 66.6%. The f1-score values were 96.3%, 53.2%, 89.7%, and 79.9% for the four vehicle classes. The low accuracy in the bus class is because there are no buses in the test videos, only minibuses, and the model is trained on large regular buses; minibuses are classified as salon cars.

X. CONCLUSIONS

The proposed system provided traffic monitoring information in real-time using a combination of cutting-edge deep learning technology and a state-of-the-art YOLOv8 CNN model. YOLOv8 was chosen for its performance speed, accuracy on a variety of object detection benchmarks, and robustness to a variety of challenges, including occlusion, noise, and different lighting conditions. The system gave excellent results in all of its stages, with high accuracy and low error rates. Deep learning models require heavy processing, so a GPU needs to be utilized for real-time processing, benefiting from its parallel processing capability. Although the proposed system can run on a CPU, this will affect the system's performance. The information provided includes vehicle classification and a count number for each class. This information will be useful in the upcoming system development for the speed estimation facility, which will enforce traffic laws for speed limits based on the vehicle class.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

ACKNOWLEDGMENT

We appreciate the University of Technology staff for providing the support necessary for us to finish this research.

DOI: https://doi.org/10.33103/uot.ijccce.24.2.4

REFERENCES

[1] WHO, “Global Status Report on Road Safety 2018,” World Health Organization, Geneva, pp. 165, 2018.

[2] A. F. Jabbar, R. F. Ghani and A. A. Salman, “Collision Prediction Based on Vehicular Communication System,” *Iraqi Journal of Computers, Communications, Control & Systems Engineering,* vol. 22, no. 3, pp. 72-80, 2022.

[3] F. H. A. Asad, “Road Traffic Accidents in Iraq: A Review of Evidence-based Literature,” *International Journal for Traffic and Transport Engineering,* vol. 7, pp. 256-275, 2017.

[4] H. Song, H. Liang, H. Li, Z. Dai and X. Yun, “Vision-based Vehicle Detection and Counting System Using Deep Learning in Highway Scenes,” *European Transport Research Review*, vol. 11, no. 51, 2019.

[5] M. Fachrie, “A Simple Vehicle Counting System Using Deep Learning with YOLOv3 Model,” *Jurnal RESTI,* vol. 4, no. 3, pp. 462-468, 2020.

[6] C. Lin, S. Jeng and H. Lioa, “A Real-Time Vehicle Counting, Speed Estimation, and Classification System Based on Virtual Detection Zone and YOLO,” *Mathematical Problems of Applied System Innovations for IoT Applications*, no. 1577614, 2021.

[7] M. R. Shihab, R. F. Ghani and A. J. Mohammed, “Machine Learning Techniques for Vehicle Detection,” *Iraqi Journal of Computers, Communications, Control and Systems Engineering,* vol. 22, pp. 1-12, 2022.

[8] H. Koyuncu and B. Koyunco, “Vehicle Speed Detection by Using Camera and Image Processing Software,” *THE IJES,* vol. 7, pp. 64-72, 2018.

[9] C. Liu, D. Q. Huynh, Y. Sun, M. Reynolds and S. Atkinson, “A Vision-Based Pipeline for Vehicle Counting, Speed Estimation, and Classification,” *IEEE Transactions on Intelligent Transportation Systems,* vol. 22, no. 12, pp. 7547-7560, 2020.

[10] A. B. Ahmad and T. Tsuji, “Traffic Monitoring System Based on Deep Learning and Seismometer Data,” *applied sciences,* vol. 11, no. 10, p. 4590, 2021.

[11] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaria, M. A. Fhadel, M. Al-Amidie and L. Farhan, “Review of Deep Learning: Concepts, CNN Architectures, Challenges, Applications, Future Directions,” *Journal of Big Data,* vol. 8, no. 53, 2021.

[12] J. B. Awotunde, R. G. Jimoh, A. L. Imoize, A. T. Abdulrazaq, C. Li and C. Lee, “An Enhanced Deep Learning-Based DeepFake Video Detection and Classification System,” *electronics,* vol. 12, no. 87, 2023.

[13] I. H. Sarker, “Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions,” *SN Computer Science,* vol. 2, no. 420, 2021.

[14] S. Dong, P. Wang and K. Abbas, “A survey on Deep Learning and its Applications,” *Computer Science Review,* vol. 40, no. 100379, 2021.

[15] K. AlNujaidi, G. AlHabib and A. AlOdhieb, “Spot-The-Camel Computer Vision for Safer Roads,” *International Journal of Artificial Intelligence and Applications,* vol. 14, no. 2, 2023.

[16] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu and B. Xiao, “Deep High-Resolution Representation Learning for Visual Recognition,” *arXiv,* no. 1908.07919, 2020.

[17] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie, Y. Li, B. Zhang, Y. Liang, L. Zhou, X. Xu, X. Chu, X. Wei and X. Wei, “YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications,” *arXiv,* 2022.

[18] J. Terven and D. Cordova-Esparza, “A Comprehensive Review of YOLO: From YOLOv1 and Beyond,” *arXiv,* no. 2304.00501, 2023.

[19] M. Hussain, “YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection,” *machines,* vol. 11, no. 7, p. 677, 2023.

[20] C.-Y. Wang, A. Bochkovskiy and H.-Y. M. Liao, “YOLOv7: Trainable bag-of-freebies Sets New State-of-the-art for Real-time Object Detectors,” *arXiv,* no. 2207.02696, 2022.

[21] G. Jocher, A. Chaurasia and J. Qiu, “YOLO by Ultralytics (Version 8.0.0),” 2023. [Online]. Available: https://github.com/ultralytics/ultralytics. [Acessed 25 September 2023].

[22] P. Li, J. Zheng, P. Li, H. Long, M. Li and L. Gao, “Tomato Maturity Detection and Counting Model Based on MHSA-YOLOv8,” *sensors,* vol. 23, no. 15, p. 6701, 2023.

[23] H. Lou, X. Duan, J. Guo, H. Liu, J. Gu, L. Bi and H. Chen, “DC-YOLOv8: Small Size Object Detection Algorithm Based on Camera Sensor,” *electronics,* vol. 12, 2023.

[24] A. Aboah, B. Wang, U. Bagci and Y. Adu-Gyamfi, “Real-time Multi-Class Helmet Violation Detection Using Few-Shot Data Sampling Technique and YOLOv8,” *arXiv,* no. 2304.08256, 2023.

[25] M. Ma and H. Pang, “SP-YOLOv8s: An Improved YOLOv8s Model for Remote Sensing Image Tiny Object Detection,” *applied sciences,* vol. 13, no. 8161, 2023.

Received 28/September/2023; Accepted 28/October/2023