TELKOMNIKA Telecommunication Computing Electronics and Control
Vol. 22, No. 1, February 2024, pp. 104~112
ISSN: 1693-6930, DOI: 10.12928/TELKOMNIKA.v22i1.25096

Real-time vehicle counting using custom YOLOv8n and DeepSORT for resource-limited edge devices

Abuelgasim Saadeldin, Muhammad Mahbubur Rashid, Amir Akramin Shafie, Tahsin Fuad Hasan
Department of Mechatronics, Faculty of Engineering, International Islamic University Malaysia (IIUM), Kuala Lumpur, Malaysia

Article Info

Article history:

Received Feb 27, 2023
Revised Dec 5, 2023
Accepted Dec 15, 2023

Keywords:

Edge computing
Vehicle counting
Vehicle detection
Vehicle tracking
You only look once version 8 nano

ABSTRACT

Recently, there has been a significant increase in the use of deep learning and low-computing edge devices for analysis of video-based systems, particularly in the field of intelligent transportation systems (ITS). One promising application of computer vision techniques in ITS is in the development of low-computing and accurate vehicle counting systems that can be used to eliminate dependence on external cloud computing resources. This paper proposes a compact, reliable and real-time vehicle counting solution which can be deployed on low-computational requirement edge computing devices. The system makes use of a custom-built vehicle detection algorithm based on the you only look once version 8 nano (YOLOv8n), combined with a deep association metric (DeepSORT) object tracking algorithm and an efficient vehicle counting method for accurate counting of vehicles in highway scenes. The system is trained to detect, track and count four distinct vehicle classeses, namely: car, motorcycle, bus, and truck. The proposed system was able to achieve an average vehicle detection mean average precision (mAP) score of 97.5%, a vehicle counting accuracy score of 96.8% and an average speed of 19.4 frames per second (FPS), all while being deployed on a compact Nvidia Jetson Nano edge-computing device. The proposed system outperforms other previously proposed tools in terms of both accuracy and speed.

Corresponding Author:

Muhammad Mahbubur Rashid
Department of Mechatronics, Faculty of Engineering, International Islamic University Malaysia (IIUM)
Gombak, Kuala Lumpur, Malaysia
Email: mahbub@iium.edu.my

1. INTRODUCTION

Over the past decade, the number of vehicles on the road have started to increase significantly, leading to traffic congestion and safety concerns [1]. As a result, there have been a growing interest in traffic flow analysis amongst researchers for the development of better traffic management systems. Traditionally, physical hardware devices were used to collect real-time data about moving vehicles, often placed underneath roadways [2]. However, with recent advancements in technology and computer vision techniques, researchers have started to explore vision-based solutions for gathering information such as the vehicle speed, type, direction of movement and traffic density [3] which can be used for creation of more effective traffic flow management systems and improved road safety.

Vision-based solutions are able to provide more detailed information and are significantly easier to install and maintain as compared to traditional hardware sensors [4]. These solutions utilize cameras to capture images or videos of traffic and then apply computer vision algorithms to extract useful information from the captured data. The integration of vision-based solutions in intelligent transportation systems (ITS) is relatively new. Despite recent substantial research efforts, there remains immense potential for advancements as continuous breakthroughs in artificial intelligence (AI) and edge-computing systems unfold [5]. Most

contemporary solutions used for addressing the issue of vehicle counting in highway scenes typically rely on cloud computing resources, which often demand high computational power, internet connectivity and are vulnerable to security breaches [6], thereby exposing private and confidential data, including vehicle license plate numbers through the web. On the other hand, edge computing systems have been proved to have lower latency, enhanced stability and superior security in contrast to cloud-based methods [7]. This is particularly useful in areas where privacy is essential such as in traffic management systems.

This research aims to develop a low-cost, efficient and secure real-time highway-based vehicle counting system using state-of-the-art object detection and tracking algorithms. The system will be deployed on low computationally expensive edge-computing devices where all of the computation will take place locally and only the live vehicle counts will be passed through the web. The proposed system will utilize you only look once version 8 nano (YOLOv8n), the latest and leading object detection algorithm [8], as the base model for the custom vehicle detection algorithm. This will ensure accurate detection of small-scale vehicles in highway scenes. To track the vehicles across different frames in the video sequence, the system will make use of the simple online and realtime tracking with a deep association metric (DeepSORT) [9]. Finally, a unique and efficient counting method will be implemented and used for counting of the tracked vehicles across the highway scenes.

2. METHOD

The proposed solution is composed of three main components: vehicle detection, tracking and counting. A custom vehicle detection algorithm based on the YOLOv8n will be developed and used to detect as well as classify the four different classes of vehicles. Additionally, DeepSORT object tracking algorithm will be used to assign vehicle IDs and track the vehicles as they move across in different frames in the video sequence. Finally, we will also be implementing a unique vehicle counting method that counts the tracked vehicles as they cross through a virtual polygon area on the highway in real-time. The summary of our proposed vehicle counting system is displayed in Figure 1.

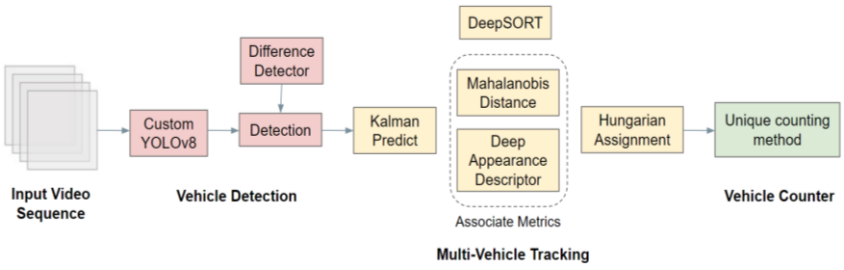

Figure 1. Proposed vehicle counting system

The dataset utilized in this study contains various sources of highway traffic images, including open-source datasets, footages from CCTV cameras and manually captured images. The main open-source datasets used in this research were the highway vehicle dataset [10] and miovision traffic camera dataset (MIO-TCD) [11]. The generated dataset consists of four distinct vehicle classes, namely: car, motorcycle, bus, and truck. Images of the vehicles were captured under various conditions, including different locations, times of the day, and weather conditions, while the vehicles themselves were of varying sizes. These factors contributed to a robust dataset that can be applied to a variety of environments and weather conditions. The images were annotated and divided into training, validation and test sets with a ratio of 0.7, 0.2, and 0.1, respectively. In total the dataset consisted of 11,982 images. The distribution of vehicle classes in the dataset are as follows: cars account for 58.23%, motorcycles account for 7.35%, buses account for 11.27%, and trucks account for 23.15%. The Table 1 shows more detailed information about the generated dataset, including the number of instances of each vehicle class in the training, validation, and test sets respectively.

Table 1. Generated vehicle dataset information

Subset	Number of images	Number of cars	Number of motorcycles	Number of buses	Number of trucks
All	11,982	23,360	1,950	2,975	9,286
Train	8,237	16,252	890	1,382	6,500
Validation	2,350	5,487	60	395	2,653
Test	1,184	2,336	45	198	929

2.1. Vehicle detection

The study employed a convolutional neural network-based (CNN-based) vehicle detection algorithm based on the YOLOv8, which is a popular object detection algorithm written in Python and makes use of the PyTorch deep learning framework [8]. YOLOv8 offers five different model scales, including nano, small, medium, large and extra-large, as illustrated in Figure 2. The model's scales vary in depth and width, maintaining the overall structure while increasing in both size and complexity [8]. The individual model structures can be modified by increasing the number of neurons, hidden layers or by performing batch normalization or weight initialization. In this study we utilized the smallest and fastest model, YOLOv8n as the base and modified the structure by adding a few extra layers to improve the detection accuracy of small vehicle objects observed in highway-scenes, while still maintaining a light-weight model suitable for deployment on embedded devices.

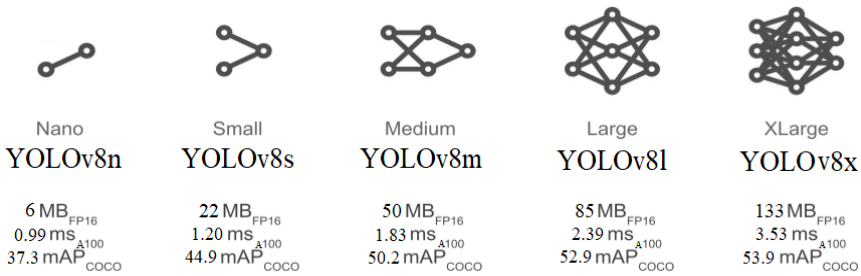

Figure 2. The different YOLOv8 model scales

Detection of small-scale vehicles in highway-scenes is a challenging task for the default smallest YOLOv8n model scale due to its simple model architecture. Hence, in order to further improve the detection accuracy of smaller vehicle objects without compromising much on the model's speed, we added an attention mechanism layer as well as some additional up-sampling layers to the feature mean average precision (mAP). The convolutional block attention module (CBAM) [12] replaces the original CONV module and reduces the attention of the model on roads and other complex backgrounds, providing more detailed information about the passing vehicles. The up-sampling layers help to detect as well as recognize different sizes and scales of vehicles [13]. The improved model architecture, shown in Figure 3, incorporates the added attention mechanism layer and small target detection layers (rows 12-13, 29-32, and 34-47, respectively).

```
1   nc: 4  # number of vehicle classes
2   depth_multiple: 0.33
3   width_multiple: 0.25
4
5   anchors:
6     - [10,13, 16,30, 33,23]
7     - [30,61, 62,45, 59,119]
8     - [116,90, 156,198, 373,326]
9
10  backbone:
11    [[-1, 1, Focus, [64, 3]],
12      [-1, 1, Conv_CBAM, [128, 3, 2]],
13      [-1, 3, C3, [128]],
14      [-1, 1, Conv, [256, 3, 2]],
15      [-1, 6, C3, [256]],
16      [-1, 1, Conv, [512, 3, 2]],
17      [-1, 9, C3, [512]],
18      [-1, 1, Conv, [1024, 3, 2]],
19      [-1, 3, C3, [1024]],
20      [-1, 1, SPP, [1024, 5]],
21    ]
22
23  neck:
24    [[-1, 1, Conv, [768, 1, 1]],
25      [-1, 1, nn.Upsample, [None, 2, 'nearest']],
26      [[-1, 8], 1, Concat, [1]],
27      [-1, 3, C3, [768, False]],
28
```

```
28
29      [-1, 1, Conv, [512, 1, 1]],
30      [-1, 1, nn.Upsample, [None, 2, 'nearest']],
31      [[-1, 6], 1, Concat, [1]],
32      [-1, 3, C3, [512, False]],
33
34      [-1, 1, Conv, [256, 1, 1]],
35      [-1, 1, nn.Upsample, [None, 2, 'nearest']],
36      [[-1, 4], 1, Concat, [1]],
37      [-1, 3, C3, [256, False]],
38
39      [-1, 1, Conv, [256, 3, 2]],
40      [[-1, 14], 1, Concat, [1]],
41      [-1, 3, C3, [512, False]],
42
43      [-1, 1, Conv, [512, 3, 2]],
44      [[-1, 10], 1, Concat, [1]],
45      [-1, 3, C3, [1024, False]],
46
47      [ -1, 1, Conv, [ 768, 3, 2 ] ],
48      [ [ -1, 12 ], 1, Concat, [ 1 ] ],
49      [ -1, 3, C3, [ 1024, False ] ],
50    ]
51
52  head:
53    [[17, 20, 23], 1, Detect, [nc, anchors]]
54
```

Figure 3. Proposed vehicle detection deep neural network architecture

The attention mechanism layer, Conv_CBAM, is incorporated after the focus module in the backbone network to improve the detection accuracy of small-scale vehicle objects. This replaces the original CONV module and enables the algorithm to obtain more detailed information about passing vehicles by reducing the inference on roads and other complex backgrounds. The CBAM works by making use of two main sub-modules known as the channel attention module (CAM) and spatial attention module (SAM) [12]. Channel attention is used to identify what important elements or features are present in an image, meanwhile, spatial attention is used to identify where the important features are located in the image [12]. CAM makes use of the average-pooling and max-pooling to generate the channel attention $Mc(F)$ (1) which is computed as (1):

$$Mc(F) = \sigma\left(MLP\big(AvgPool(F)\big) + MLP\big(MaxP\ ool(F)\big)\right) = \sigma(W_1(W_0(F^c_{avg})) + W_1(W_0(F^c_{max}))) \quad (1)$$

Where σ denotes the sigmoid function, F^c_{avg} and F^c_{max} denote the average-pooled feature and max-pooled feature respectively. Both features are passed to a shared network with multi-layer perceptron (MLP) and one hidden layer, outputting the channel attention mAP, Mc. On the other hand, SAM makes use of the average-pooling and max-pooling to generate spatial attention mAP $Ms(F)$ (2) which is computed as (2):

$$Ms(F) = \sigma(f^{7\times7}([AvgPool(F);\ MaxPool(F)]) = \sigma(f^{7\times7}([Fs\ avg;\ Fs\ max])) \quad (2)$$

Where σ denotes the sigmoid function and $f^{7\times7}$ denotes a convolutional operation with filter size of 7×7. The key innovation in this network lies in the integration of CBAM, enabling the model to learn spatial attention features of vehicles by correlating channel and space. This, coupled with the additional up-sampling layers, significantly increases the detection accuracy across various scales of vehicle objects [13].

2.2. Vehicle tracking

Once the detection of vehicles has been made by using the custom YOLOv8 algorithm, vehicle features are then extracted and used as input in the DeepSORT multi-object tracking algorithm to match the features with other video frames and correlate the same vehicle with other similar ones. DeepSORT uses a combination of the Kalman filer and Hungarian algorithm for the tracking [14]. The Kalman filer predicts the current state of a vehicle based on some previous value and provides uncertainties of that prediction [15]. Once predictions have been made and the measurements have been updated, the optimal state estimate of the vehicle is then obtained as can be seen in the cycle of the Kalman filter presented in Figure 4.

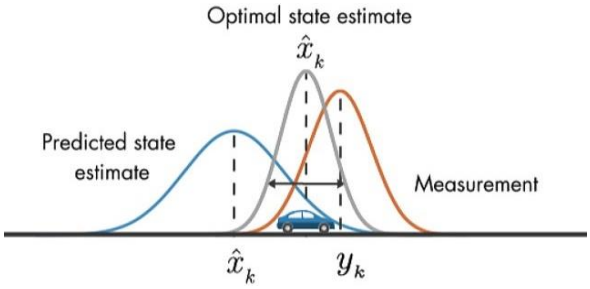

Figure 4. The cycle of Kalman filter [16]

In order to compute the current state estimate of the vehicle, the Kalman gain, measured value and previous or predicted estimation values are used as shown in (3):

$$X_k = K_n\,.Y_k + (1 - K_n)\,.X_{k-1} \quad (3)$$

Where X_k representes the current estimate of the vehicle in the k_{th} state. Y_k is the measured observation of the trajectory of the vehicle at the current time. K_n is the Kalman gain which is the weight given to the measurements and X_{k-1} is the predicted estimate of the vehicle in the previous state. Finally, once the optimal state of the vehicle has been obtained. Mahalanobis distance is then utilized to account for uncertainties introduced by the Kalman filter, determining whether each sample is an outlier or a member of the group [17]. The Hungarian algorithm is then used for vehicle association and ID attribution, assigning a unique identification to the vehicles based on the vehicle features [17]. Figure 5 illustrates the block diagram of the complete flow of data in the DeepSORT algorithm.

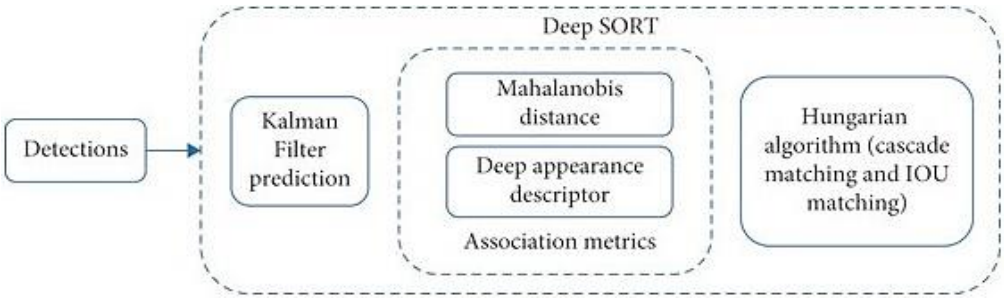

Figure 5. Flow of data in the DeepSORT algorithm [18]

2.3. Vehicle counting

In order to avoid ID switches which often occurs when similar features observed in some vehicles, we introduced a "virtual polygon area" to expand the robustness of the vehicle counting system. The virtual polygon area divides the scene into two regions, zone 1 and zone 2, as illustrated in Figure 6. Zone 1 refers to the region situated outside of the specified polygon area, whereas, zone 2 refers to the region situated within the polygon area. The vehicle counting takes place once the vehicles have crossed the highway and their center coordinates enters from zone 1 into zone 2, while maintaining a unique vehicle ID assignment. This approach reduces reliance solely on vehicle tracking and ID assignment for counting, ensuring more accurate vehicle counting results.

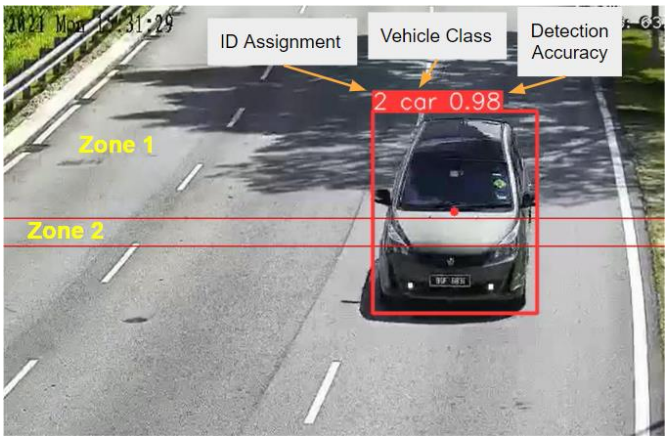

Figure 6. Virtual polygon area used for vehicle counting

2.4. Model deployment on low-computing edge device

An Nvidia Jetson Nano edge-computing device was utilized for deployment of our vehicle counting system. The device is small in size and makes use of a single board computer (SBC) which enables it to bring efficient computer performance to the edge. Furthermore, the decentralized character of edge platforms also makes them highly reliable infrastructures [19]. Data is processed locally, providing offline capabilities, a feature not readily available when continuously streaming video data to the cloud for processing, which is also undesirable from a privacy perspective [20].

The edge-computing system platform was deployed on a busy highway located in Kuala Lumpur, Malaysia and the flow of data was as follows; the process starts with acquiring video stream of vehicles using a Logitech C920 pro HD webcam, followed by transmitting the information to the edge-computing device's RAM. The Jetson Nano then utilizes this information as input and performs vehicle detection, tracking and counting while the Nvidia CPU controls modules such as the compute unified device architecture (CUDA) and tensor cores using heterogenous parallel computing to accelerate the model by hardware. The system then provides real-time outputs of vehicle detection and counting results for each of the trained vehicle classes, which are displayed on a 7-inch LCD screen.

3. RESULTS AND DISCUSSION

The vehicle detection model was trained on a dataset of 8,237 images, with an additional 2,350 images used for model validation. The primary metric used for evaluating the performance of the trained models were the mean average precision, mAP@.5, and loss function plots. The model training results are presented in Table 2, which include four distinct models. The first model utilizes the default YOLOv8n model architecture with the original generated dataset. The second model utilizes the same YOLOv8n but with data augmentation applied to the training images, this includes random image rotations, addition of noise, rain and varying brightness. The third model utilizes the modified YOLOv8n architecture which adds a small object detection layer and a CBAM to the network, in addition to data augmentation. Finally, the last model utilizes the same modified YOLOv8n architecture but with the original dataset and no data augmentation applied. All of the models were trained for 300 epochs and using an Nvidia RTX 3060 GPU.

Table 2. Trained vehicle detection models

Data augmentation	Small object detection layer	CBAM	mAP@.5	Recall	Precision	Epochs
			95.6	91.1	91.6	300
√			95.1	91.4	91.0	300
√	√	√	97.2	93.2	92.1	300
	√	√	97.5	92.5	93.3	300

In object detection, precision and recall are essential metrics used to evaluate the performance of a trained model in correctly identifying detected objects [21]. Precision (4) measures the proportion of positive vehicle identifications that were accurately identified using true positive (TP) and false positive (FP) detections. On the other hand, recall (5) measures the proportion of actual positive identifications that were correctly identified using true positive (TP) and false negative (FN) detections [21]. A high value of precision and recall indicates that the model can accurately detect all positive vehicles correctly and classify them accurately. Figure 7 depicts the precision-recall curve of our highest accuracy model.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad (4)$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (5)$$

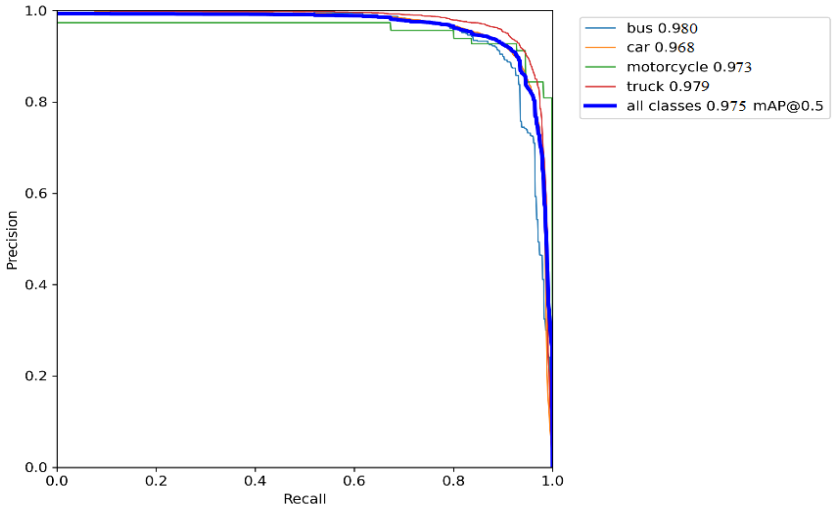

Figure 7. Precision-recall curve

Upon the successful completion of our training, which took approximately 9 hours and 13 minutes to complete, the loss function plots were also saved and are displayed in Figure 8. The plots depict three different types of losses, namely the classification, objectness and box losses. The box regression loss indicates the algorithm's proficiency in locating the vehicle's centre and how well the algorithm's prediction of the bounding boxes covered them. The classification loss evaluates the algorithm's capacity to predict the correct vehicle class upon detection of a vehicle object, and the objectness loss measures the likelihood that a vehicle exists in a predicted region of interest.

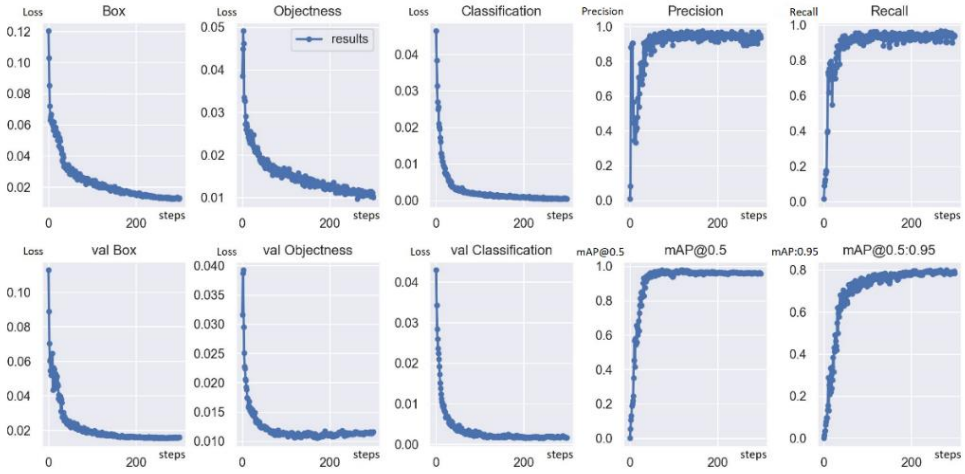

Figure 8. Model training plots over epochs on validation set

The modified YOLOv8n model architecture resulted in a faster convergence of the loss function plots as compared to the original YOLOv8n model architecture. At approximately 150 epochs, the model using the modified architecture had already converged with 11.58% lower loss as compared to using the default model architecture. The mean average precision, mAP@.5 of the model gradually stabilizes after 200 epochs of training and reached its highest score at 97.5% after 300 epochs of training. Applying data augmentation to the training images helped to improve the model's recall, allowing it to detect most of the positive vehicles correctly. However, this also reduced the model's precision, resulting in an increase in the number of false positive detections.

We evaluated the accuracy of our vehicle detection model on the validation set and compared its performance to the original YOLOv8n model architecture, which resulted in an average mAP@.5 score of 95.6%. We also compared our model's performance with other vehicle detection models, such as the YOLOv7 [22] which had obtained an average mAP@.5 score of 95.2%, YOLOv5n [23] which obtained an average mAP@.5 score of 93.2% and finally, the faster R-CNN [24] which had obtained an average mAP@.5 score of 89.05% on the same validation set. All of the models were deployed on a 4 GB Nvidia Jetson Nano and the inference speed was calculated. Our proposed algorithm demonstrated superior performance in terms of both average vehicle detection mAP score as well as inference speed, as shown in Table 3.

Table 3. Vehicle detection accuracy comparison

Algorithm	mAP@.5	Speed (FPS)	Model size (mb)
Faster R-CNN [24]	89.1	0.5	182
YOLOv5n [23]	93.2	16.3	14
YOLOv7 [22]	95.5	17.1	12
YOLOv8n [8]	95.6	19.5	6
Proposed algorithm	97.5	19.4	6

With regards to vehicle counting, our experimental setup was deployed on three different highway locations, each capturing real-time videos of vehicles including cars, motorcycles, buses and trucks crossing through the roadway which can be viewed in the following link https://cutt.ly/U8pE2w4. The system was deployed on a 4GB Nvidia Jetson Nano edge-computing device and our proposed method was able to achieve an average vehicle counting accuracy score of 96.8% on our captured video data. To benchmark its performance, we compared the counting accuracy with two other vehicle counting methods. The first method [25] utilized a distance measurement line counter, achieving an average vehicle counting accuracy score of 92.2%. The second method [10] utilized a virtual line counter, achieving an average vehicle counting accuracy score of 93.2%.

Our proposed vehicle counting method which makes use of a virtual polygon area counting approach was able to achieve the highest vehicle counting accuracy score, avoiding duplicate counts and missed objects. Additionally, the proposed counting method was also capable of counting vehicles on both on-going as well as out-going traffic simultaneously. Overall, the vehicle counting system consisted of a lightweight vehicle detection and tracking model and an efficient vehicle counting method, achieving an average inference speed of 19.4 FPS on a low-computing Nvidia Jetson Nano edge device and 84 FPS on an RTX 3060 laptop GPU.

4. CONCLUSION

To summarize, our proposed vehicle counting solution has demonstrated through qualitative analysis that it is well-suited for the task of vehicle detection and tracking in highway scenes. Our system combines a custom vehicle detection algorithm based on the YOLOv8n model architecture, DeepSORT object tracking algorithm and a unique "virtual polygon area" counting approach that yielded accurate and efficient vehicle counting results. The system is also light-weight, can be deployed on edge devices with low computing power and provides better security by keeping computation and data storage closer to the data source rather than relying on cloud computing resources. Consequently, our solution offers a low-cost alternative that delivers real-time performance on embedded devices such as on the Nvidia Jetson Nano.

ACKNOWLEDGEMENTS

The authors would like to express our sincere appreciation to the Faculty of Engineering at the International Islamic University Malaysia for generously providing us with the necessary research facilities and equipment required for conducting the research experiments presented in this paper.

REFERENCES

[1] A. E. Retallack and B. Ostendorf, "Current understanding of the effects of congestion on traffic accidents," *International Journal of Environmental Research and Public Health*, vol. 16, no. 18, 2019, doi: 10.3390/ijerph16183400.
[2] V. Mandal and Y. Adu-Gyamfi, "Object Detection and Tracking Algorithms for Vehicle Counting: A Comparative Analysis," *Journal of Big Data Analytics in Transportation*, vol. 2, no. 3, pp. 251–261, Dec. 2020, doi: 10.1007/s42421-020-00025-w.
[3] E. Verani and M. Pitsiava-Latinopoulou, "Traffic management integrated in the urban development of an area towards sustainability," in *Proceedings of the International Conference on Changing Cities I: Spatial, morphological, formal and socio-economic dimensions*, 2013, pp. 2408–2417.
[4] Q. Zhang, H. Sun, X. Wu, and H. Zhong, "Edge video analytics for public safety: A review," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1675–1696, 2019, doi: 10.1109/JPROC.2019.2925910.
[5] L. S. Iyer, "AI enabled applications towards intelligent transportation," *Transportation Engineering*, vol. 5, p. 100083, 2021, doi: 10.1016/j.treng.2021.100083.
[6] G. S. Sriram, "Edge Computing vs. Cloud Computing: An overview of Big Data challenges and opportunities for large enterprises," *International Research Journal of Modernization in Engineering Technology and Science*, vol. 4, no. 1, pp. 1331–1337, 2022.
[7] K. Tiba, R. Parizi, Q. Zhang, A. Dehghantanha, H. Karimipour, and K.-K. R. Choo, "Secure Blockchain-Based Traffic Load Balancing Using Edge Computing and Reinforcement Learning," 2020, pp. 99–128, doi: 10.1007/978-3-030-38181-3_6.
[8] G. Jocher *et al.*, "ultralytics/yolov3: v8 - Final Darknet Compatible Release." Zenodo, Nov. 2020. doi: 10.5281/zenodo.4279923.
[9] S. Guo *et al.*, "A Review of Deep Learning-Based Visual Multi-Object Tracking Algorithms for Autonomous Driving," *Applied Sciences*, vol. 12, no. 21, 2022, doi: 10.3390/app122110741.
[10] 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. 1, p. 51, Dec. 2019, doi: 10.1186/s12544-019-0390-4.
[11] Z. Luo *et al.*, "MIO-TCD: A New Benchmark Dataset for Vehicle Classification and Localization," *IEEE Transactions on Image Processing*, vol. 27, no. 10, pp. 5129–5141, 2018, doi: 10.1109/TIP.2018.2848705.
[12] S. Woo, J. Park, J. Lee, and I. S. Kweon, "CBAM: Convolutional Block Attention Module," *Eccv*, p. 17, 2018.
[13] S. Kundu, H. Mostafa, S. N. Sridhar, and S. Sundaresan, "Attention-based Image Upsampling," *arXiv preprint arXiv:201209904*, 2020.
[14] Z. Wang, L. Zheng, Y. Liu, Y. Li, and S. Wang, "Towards Real-Time Multi-Object Tracking," in *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI*, Berlin, Heidelberg: Springer-Verlag, 2020, pp. 107–122, doi: 10.1007/978-3-030-58621-8_7.
[15] Q. Li, R. Li, K. Ji, and W. Dai, "Kalman Filter and Its Application," in *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, IEEE, Nov. 2015, pp. 74–77, doi: 10.1109/ICINIS.2015.35.
[16] M. Ulusoy, "Insight into Kalman filtering-probability distribution fnc," 2023.
[17] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," *Proceedings - International Conference on Image Processing, ICIP*, vol. 2017-Septe, pp. 3645–3649, 2018, doi: 10.1109/ICIP.2017.8296962.
[18] D.-L. Dinh, H.-N. Nguyen, T. Thai, and K.-H. Le, "Towards AI-Based Traffic Counting System with Edge Computing," *Journal of Advanced Transportation*, vol. 2021, pp. 1–15, 2021, doi: 10.1155/2021/5551976.
[19] S. Leroux, B. Li, and P. Simoens, "Automated training of location-specific edge models for traffic counting," *Computers and Electrical Engineering*, vol. 99, no. April, pp. 1–7, 2022, doi: 10.1016/j.compeleceng.2022.107763.
[20] C. Ponnusamy, R. D., V. Praveena, A. Jeba, and B. B., "Data Security and Privacy Requirements in Edge Computing: A Systemic Review," 2021, pp. 171–187, doi: 10.4018/978-1-7998-4873-8.ch009.
[21] A. Badithela, T. Wongpiromsarn, and R. Murray, *Evaluation Metrics for Object Detection for Autonomous Systems*. 2022. doi: 10.48550/arXiv.2210.10298.
[22] 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," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.
[23] G. Jocher *et al.*, "ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation." Zenodo, Nov. 2022, doi: 10.5281/zenodo.7347926.
[24] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017, doi: 10.1109/TPAMI.2016.2577031.
[25] M. Fachrie, "A Simple Vehicle Counting System Using Deep Learning with YOLOv3 Model," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 4, no. 3, pp. 462–468, 2020, doi: 10.29207/resti.v4i3.1871.

BIOGRAPHIES OF AUTHORS

Abuelgasim Saadeldin received the B.Eng. degree in Mechatronics Engineering from the International Islamic University Malaysia (IIUM), in 2019, achieved his Master of Science in Engineering in July of 2023, and is currently pursuing his Doctor of Philosophy in Engineering. He has a great interest in AI, computer vision and backend software development. With over 2 years of industry working experience, he has worked on several research projects, including development of drones for automated crowd surveillance, AI vehicle parking system, autonomous AI vehicle with road following and collision avoidance and many more. In addition, he has also accomplished several achievements, including being awarded certified Jetson AI specialist by Nvidia in 2021 and awarded certified engineer in computer vision by Certifai in 2020. He can be contacted at email: abuelgasim.smm@live.iium.edu.my.

Muhammad Mahbubur Rashid received the B.Eng. degree from Bangladesh University of Engineering and Technology in Dhaka in Electrical and Electronic Engineering in 1992. He then went on to earn his Master of Science from University Malaya in 2003 and Doctor of Philosophy in Electrical Engineering from the same university in 2007. He is currently employed as an associate professor in the International Islamic University Malaysia's Department of Mechatronics Engineering. In 1992, he began his employment at Apex Spinning and Knitting Mills as an assistant engineer. More than 80 of his papers have been published in journals and conference proceedings. Advanced control systems, simulation, nonlinear modeling, instrumentation, deep learning, computer vision, renewable energy, and power electronics are some of his areas of interest in study. He can be contacted at email: mahbub@iium.edu.my.

Amir Akramin Shafie received the B.Eng. degree in Mechanical Engineering from the University of Dundee and an M.S. in Mechatronics from the University of Abertay Dundee in Scotland. In 2000, the University of Dundee in Scotland awarded him a doctorate in the discipline of engineering. International Islamic University Malaysia (IIUM) has employed he as a professor in the department of Mechatronics since 2005. He also serves as the Dean of Kulliyyah of Engineering (2010–2012), Deputy Dean Academic Affairs Kulliyyah of Engineering (2006–2010), and Director of Research Management Centre (RMC) from 2021. Current research interest span both autonomous mechatronic system and intelligent system including machine learning in manufacturing and health. He can be contacted at email: aashafie@iium.edu.my.

Tahsin Fuad Hasan received the B.Eng. degree in Mechatronics Engineering from the International Islamic University Malaysia (IIUM), in 2022 and is currently pursuing his Master of Science in Engineering. His interests lie in AI, machine learning, and human-machine interaction. He can be contacted at email: tahsin10797@gmail.com.