International Journal of Intelligent Transportation Systems Research
https://doi.org/10.1007/s13177-024-00392-6

RESEARCH PAPER

Docker Container-Based Framework of Apache Kafka Node Ecosystem: Vehicle Tracking System by License Plate Recognition on Surveillance Camera Feeds

Seda Kul[1,2] · **Sarper Kumcu**[1] · **Ahmet Sayar**[1]

/ Accepted: 11 March 2024
© The Author(s) 2024

Abstract
Vehicle tracking and license plate recognition (LPR) over video surveillance cameras are essential intelligent traffic monitoring systems. Due to the enormous amount of data collected each day, it would be difficult to track vehicles by license plate in a real-world traffic setting. Large volumes of data processing, real-time request responses, and emergency scenario response may not be possible using conventional approaches. By combining license plate recognition with the docker container-based structure of the Apache Kafka node ecosystem, the suggested solution takes a novel approach to vehicle tracking. The primary components of our suggested framework for reading license plates are the identification of license plates and text data queries. License plate localization is performed with You Only Look Once version 3 (YOLOv3) and character recognition with Optical Character Recognition (OCR). The detected vehicle images with license plate results are published on related topics with Apache Kafka. Apache Kafka is a publish-subscribe (producers-consumers) messaging system and one of the most popular architectures used for streaming data. For each license plate search, a topic will be created in the framework where producers publish and consumers receive data. Thus, the workload of the operators will be reduced and they will be able to pay attention to more important events in traffic.

Keywords Intelligent transportation system · Distributed systems · Publish-subscribe messaging system · Apache Kafka · Docker · Vehicle tracking · License plate recognition · Optical character recognition

1 Introduction

Every day, the latest traffic data is required to maintain a safe and effective flow of traffic, as well as to identify and apprehend lawbreakers. In numerous circumstances, such as security control [1–3], automatic toll collection [4], unattended parking lots [5–7], and enforced traffic safety [8–12], the recognition of the license plate has been utilized to gather information about a vehicle. As technology advances, more and more locations now have traffic surveillance cameras. The requirement for real-time data processing and running data through a machine learning algorithm increases along with the amount of data that cameras are collecting. To address this challenge, we developed a Docker Container-Based Framework based on the Apache Kafka Node Ecosystem that not only allows customers to query the license plate number of the vehicle they want to track, but also offers a unique feature: the ”Plate Specific Video Reconstruction”. This feature enables users to generate specific video clips of vehicles from accumulated data, providing a customized insight into individual vehicle activities.

In our framework, microservices were leveraged for flexibility and scalability. Each microservice is made up of various components. The duties of each component are different. In practice, it prevents us from being too closely connected. When one of the services is unavailable, the application still functions. Real-time processing on a single node with a sin-

✉ Seda Kul
seda.kul@tubitak.gov.tr

Sarper Kumcu
s.sarperkumcu@gmail.com

Ahmet Sayar
ahmet.sayar@kocaeli.edu.tr

[1] Computer Engineering Department, Kocaeli University, Kocaeli, Turkey

[2] BİLGEM, The Scientific and Technological Research Institution of Turkey (TUBITAK), TÜBİTAK-BİLGEM, Gebze Yerleşkesi Marmara Araştırma Merkezi, 41470 Gebze Kocaeli, Turkey

Published online: 15 April 2024

gle GPU runs out of memory and is not feasible. It is crucial to distribute processes among various nodes to address this. Communication must be established between jobs that have been divided into smaller tasks.

Middleware message queue provides a practical way to address the issues described above. There are systems for middleware message queueing, including RabbitMQ [15], ActiveMQ [14], and ZeroMQ [13]. Research shows that in some situations Apache Kafka outperforms other solutions [16].

For massive data streaming, Apache Kafka [17] has recently been released. Real-time data in motion can be handled using streams of computation. Real-time processing is necessary in the industry for applications that use sensors, log files, and web streams. A distributed publish-subscribe messaging system called Apache Kafka enables real-time data processing by connecting several microservices.

This study suggests a real-time processing pipeline that is evaluated using Apache Kafka to analyze feeds from several sources of surveillance cameras. The pipeline is built on an open-source framework. The remaining portions of the essay are structured as follows: Related work on license plate detection and combination designs of license plate recognition and distribution systems is presented in Section 2 of this article. The proposed approach is described in Section 3. The results of the experiment are presented in Section 4, and a conclusion is provided in Section 5.

2 Related Work

2.1 Vehicle Detection and License Plate Recognition Systems

Donget et al. [18] proposed a classification of vehicle types with a Convolutional Neural Network (CNN) from the frontal view of the vehicles. The accuracy of their proposed approach reached 96. 1% during the day and 89.4% during the night. Kul et al. [19] used the Support Vector Machine (SVM), Adaboost, and Artificial Neural Network (ANN) to classify vehicles. They obtained an accuracy of 87.5%, 81.6% and 85.4%, respectively. Kul et al. [20, 21] proposed a two-stage vehicle classification system. Tashiev et al. [22] developed a vehicle classification system that uses the YOLO real-time object classification framework and tested their work on the BITvehicle dataset. Their work reached an accuracy of 90.35%. They used TinyYOLO [23] and two CNN; one for vehicle detection and the other for vehicle classification. Their method with Tiny-YOLO reached 89.19% accuracy in terms of the Intersection over Union (IOU) metric. Kul et al. [24] showed that vehicle classification is a challenging problem due to the different dimensions of vehicles. Goncalves et al. [25] have established an optical character recognition system with a convolutional neural network for plate recognition. Their proposed detection approach reached an accuracy of 79.3% , and their proposed recognition approach reached an accuracy of 85.6%. Laroca et al. [26] identified the license plate recognition system based on the YOLO [23] object detector. They trained the YoloV2 architecture, which is built on the Darknet [27] library, with their own tagged data set. This model does not perform well when the frames have inclined angles. Their system performed with an accuracy rate of 78.33. Yonetsu et al. [28] proposed a two-stage YOLOv2 for the detection of license plates. They extracted cars and license plates from images with objects of cars. They also built a database of Japanese license plates. Their proposed license plate detection method reached an accuracy of 87% in clear weather, 74% at nightfall, and 11% in the dark. In [29], Nam et al. used the background modeling and subtraction (BGS) model for vehicle detection. To make their classifier robust, they performed the Gaussian functions of the OpenCV library. Z. Selmi and colleagues achieved an accuracy of 92.7%. Selmiet et al. [30] proposed a deep learning license plate recognition system. Before recognizing the characters on a license plate, they built a CNN model for the classification of plates and nonplates. Their proposed method reached 94.8% accuracy with the Caltech dataset. Their system fails in some of the cases where there are multiple license plates. D. Pu et al. [31, 32] presented a real-time CNN-based license plate detection system. But their approach did not work well for small license plates. Abdullah et al. [33], presented a real-time license plate recognition system for Bangladesh. Their dataset includes photos that have different environmental conditions and is public. They used the YOLOv3 algorithm for their system and their method achieved 85% in terms of IoU for digit recognition. Silva et al. [34] have developed a system that includes vehicle detection, license plate detection, and plate reading steps. Their approach allows reading license plates from different angles, and they also have created a synthetSilva ic data set.

2.2 Distributed Messaging Platform

Distributed messaging systems are based on reliable message queueing. Asynchronously, messages are transmitted through the system between applications. Applications have two types of filters to select messages: topic-based and content-based. There are different distributed messaging systems: Apache Kafka [35], RabbitMQ [36], JMS (Java Message Service) [37], ActiveMQ [38], ZeroMQ [39], and Kestrel [40]. Apache Kafka is a messaging platform designed to write more than 10 million messages per day on average of 172,000 messages per second [41]. Although Kafka's original purpose is for log processing, it is also used for different scenarios. There are studies in the literature that use Apache Kafka for frame transmission. Yoon-Ki Kim et al.

[42] suggested that the color channels of the images to be transmitted over Kafka should be separated and collected under separate topics. There are other studies in plate recognition systems, distributed messages, and platforms that are used together. S. Jung et al. [43], in the vehicle tracking system they developed, read plates that they captured after graying-singularizing and filtering processes for license plate recognition with Tesseract OCR. They used Apache Kafka to transmit the visuals. H. Chen et al. [44] proposed a video processing framework based on Spark Stream and Kafka. They used Apache Kafka to acquire the camera data in real time. Kul et al. [45] proposed a real-time vehicle classification system based on the Java Message Service. They separated vehicle types into three classes: small, medium and big. Each class has a topic. After classification is done, the vehicles are published according to their labels. They used ANN, AdaBoost and SVM algorithms. The best accuracy rate is obtained by using the ANN algorithm.

3 Proposed Method

Our proposed method basically consists of three steps: plate detection with YOLOv3 [46], plate recognition with Tesseract OCR, and transmission of detected plates through Apache Kafka. To provide a comprehensive understanding of our methodology and its advantages over existing systems, we present a comparison table detailing the features and capabilities of our proposed system relative to others. This comparison is provided in Table 1. The detailed flow chart that illustrates the steps and processes of our method is shown in Fig. 1.

3.1 Vehicle Detection and License Plate Detection

Generally, in plate recognition systems, plate detection is usually performed first and then character recognition is performed on the detected area. In our study, we first performed plate detection with YOLOv3 based on Darknet-53. YOLO is an open source artificial neural network library written in C with the CUDA programming language. The parameters batch is 64 and the number of subdivisions is 8. As shown in Fig. 2, different vehicle types have different position on the license plate. Therefore, we trained the YOLOv3 network accordingly and used the BIT-Vehicle data set for this. In this data set, there are 6 classes of vehicles, namely buses, microbusses, minivans, sedans, SUVs, and trucks. These classes include 558, 883, 476, 5,922, 1,392, and 822 vehicles, respectively.

Table 1 comparison Table

	Proposed System	Jung et. al.
ALPR	Yes	Yes
Multiple Video Source Support	Yes	Yes
Scalability	Yes	Yes
Portability	Yes	Not Mentioned
Vehicle Trace Video Construction	Yes	No

3.2 Distributed Publish Subscribe Message Platform

With the developed system, users will enter the license plate number they want to query into the system. By creating a topic with the detected plate number, the user will only see images of that vehicle.

Apache Kafka is an open source stream processing platform developed by LinkedIn [35], where producers publish messages into topics and consumers receive messages from the subscribed topics. A topic is a category of messages and may be stored in one or more partitions in the Apache Kafka cluster. Here, there are brokers, also known as Kafka Servers. Brokers are used to distribute all partitions and replications.

As shown in Fig. 1, we used 3 brokers, 2 topics, 2 partitions, and 1 replication. We use the Apache Avro format for metadata creation to send data over the cluster. The metadata contains three pieces of information: the date the image was saved, the camera ID, and the vehicle image in byte array format. Figure 3 shows how Apache Avro was used to serialize these JSON files before deploying them on Kafka.

3.3 Vehicle Trace Video Construction

Our proposed method introduces a clear strategy for effectively managing large volumes of video data. By doing so, it improves the user experience, allowing users to directly access relevant video clips without the need to browse through long recordings. Importantly, the system is designed with scalability in mind. As more surveillance feeds or vehicles are added, the system remains manageable due to the easy procedure of adding new Kafka topics.

Upon initiating a query via the Web interface, users specify the start date, end date, and license plate number to be queried. Subsequently, they are assigned a UUID. The server subscribes to the Kafka topic, which is named as the specified license plate number, using the uniquely assigned UUID as the group number and begins asynchronously reading messages under this topic. The specific assignment of the group ID ensures that all messages are retrieved without any loss of data. Upon receipt, each message undergoes decoding, followed by date filtering. Messages whose date headers fall

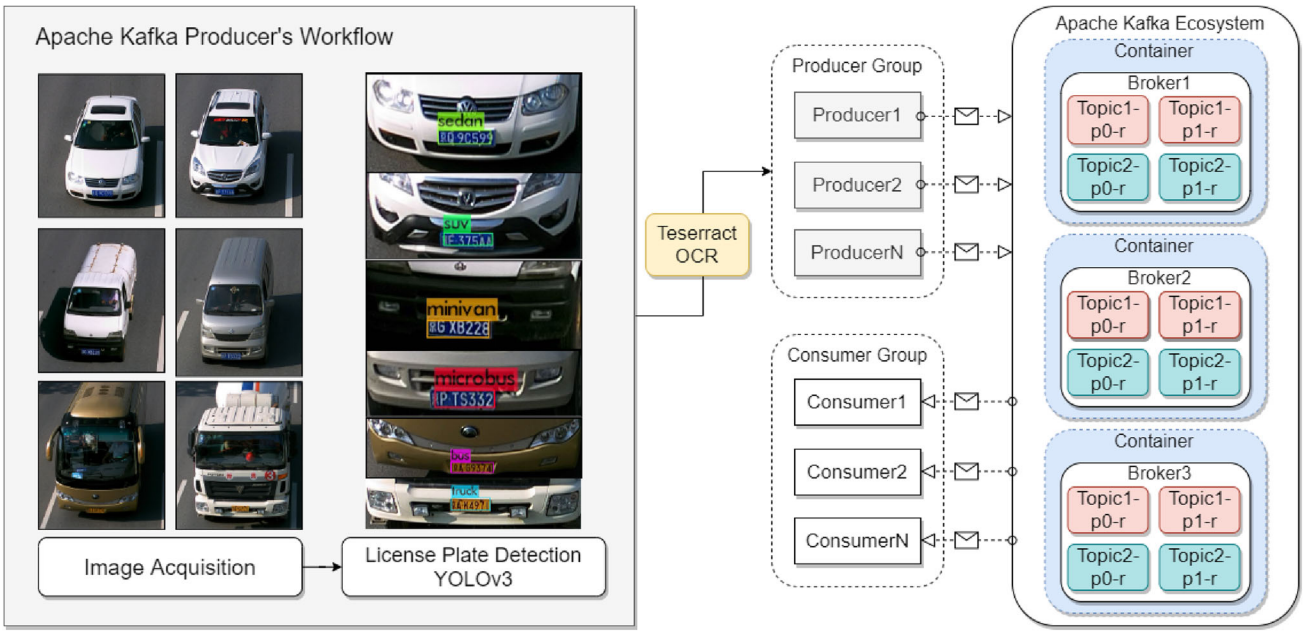

Fig. 1 System Architecture

outside the user-defined range are ignored and not retained in memory. The selected messages are first sorted by date and subsequently by the timestamp (in milliseconds) of the captured image.

Once the data are sorted by date, we use the OpenCV library to compile and produce a video, which is then saved on the server in mp4 format as seen in Fig. 4.

4 Experimental Results

All our experiments were performed on a machine with Intel ® Core™ i9-9900KF CPU @ 3.60GHz × 16, GeForce RTX 2080 Ti/PCIe/SSE2. CuDNN: 7.6.5, CUDNNHALF = 1, GPUcount: 2 and OpenCVversion: 4.1.2.



Fig. 2 Vehicle Detection and License Plate Recognition

Fig. 3 Avro Metadata



4.1 Plate Detection and Recognition

In YOLOv3, F1 scores and IoU metrics are used to track the accuracy of the model. Model training was carried out by labeling the plates of different types of vehicles separately. LabelImg [47] is a tool used to obtain ground-truth labels. Our trained model has achieved IoU = 79.01%, precision = 0.96, recall = 0.98, F1 score = 0.97, and average precision = 98.28%, as shown in Table 2.

4.2 Apache Kafka

Apache Kafka can run both locally and in a Docker container. The Docker container microservice architecture provides many advantages. However, performance may differ. In Fig. 5, the application performance (message / second) is shown. Messages were sent in batches of 10, 100, 1000, 10,000, and 100,000.

The primary objective of this experiment was to measure the throughput (messages per second) of an application using

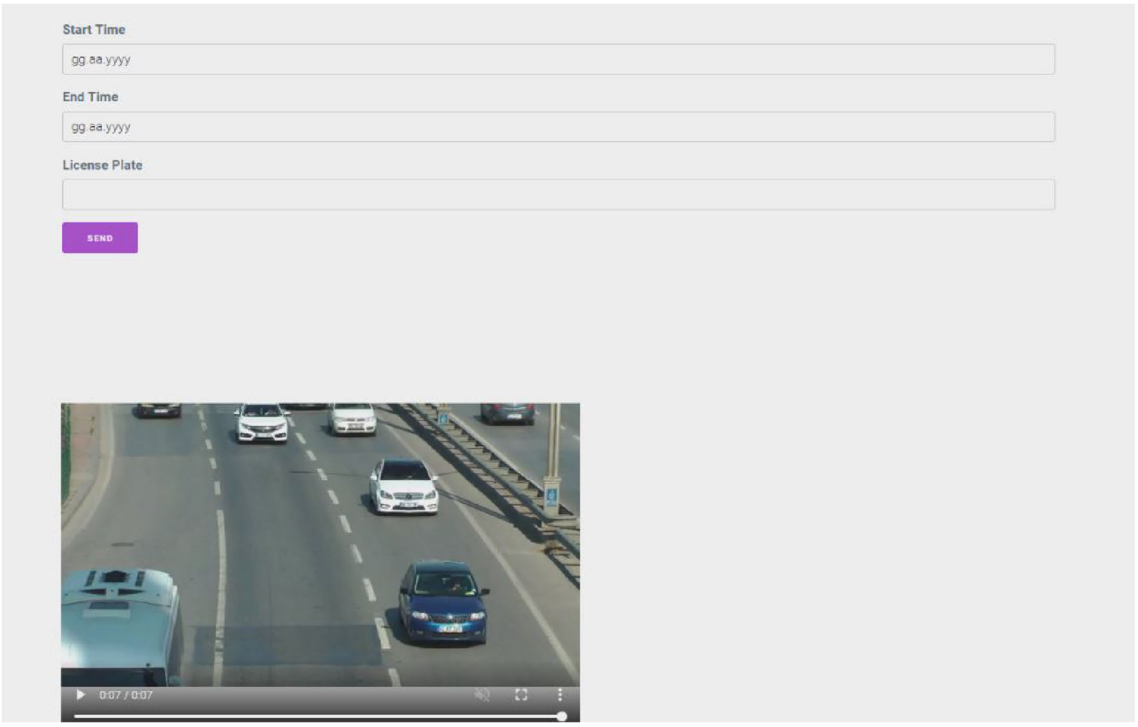

Fig. 4 Constructed video clip of given plate

Table 2 YOLOv3 Performance on BIT-Vehicle Dataset

Vehicle Class	True Positive & False Positive	Average Precision
Bus	TP=496, FP=0	99.08%
Microbus	TP=574, FP=21	98.77%
Minivan	TP=422, FP=7	99.41%
Sedan	TP=491, FP=29	95.02%
SUV	TP=450, FP=44	97.55%
Truck	TP=474, FP=18	99.15%
Average	TP=2907, FP=119	98.28%

Kafka, comparing its performance in a Docker environment against a local setup. Throughput was observed by transmitting messages in different batch sizes: 10, 100, 1,000, 10,000, and 100,000.

As depicted in Fig. 5, the performance varies with the size of the message batch and the environment. For smaller batch sizes, the performance on both Docker and locally remains relatively close, but as we scale up the number of records, a noticeable difference in throughput is observed.

For a batch size of 100,000 records, Kafka's performance on Docker reached a throughput of approximately 163,666.12 messages/second, while locally it achieved around 148,439.97 messages/second.

This shows that in our setup, Docker slightly outperformed the local environment when dealing with larger record numbers.

5 Conclusion

In this study, we developed a vehicle recognition system based on license plates using container-based Apache Kafka. Vehicle license plates were processed by each node and sent to their topics. Memory capacity has been the biggest limitation of monolithic applications. Although the GPU is widely used, it is still expensive and has limited capacity. With the developed system, we try to overcome the capacity problem and provide the user with flexibility of use.

In our future work, we will work on the vehicles in Turkey and create a data set that contains Turkish license plates. We will develop a smart search system with Apache Kafka.

List of Abbreviations

Abbreviation	Definition
License Plate Recognition	LPR
You Only Look Once	YOLO
You Only Look Once version 3	YOLOv3
Optical Character Recognition	OCR
Convolutional Neural Network	CNN
Support Vector Machine	SVM
Artificial Neural Network	ANN
Intersection over Union	IoU
Background Modeling and Subtraction	BGS
Java Message Service	JMS

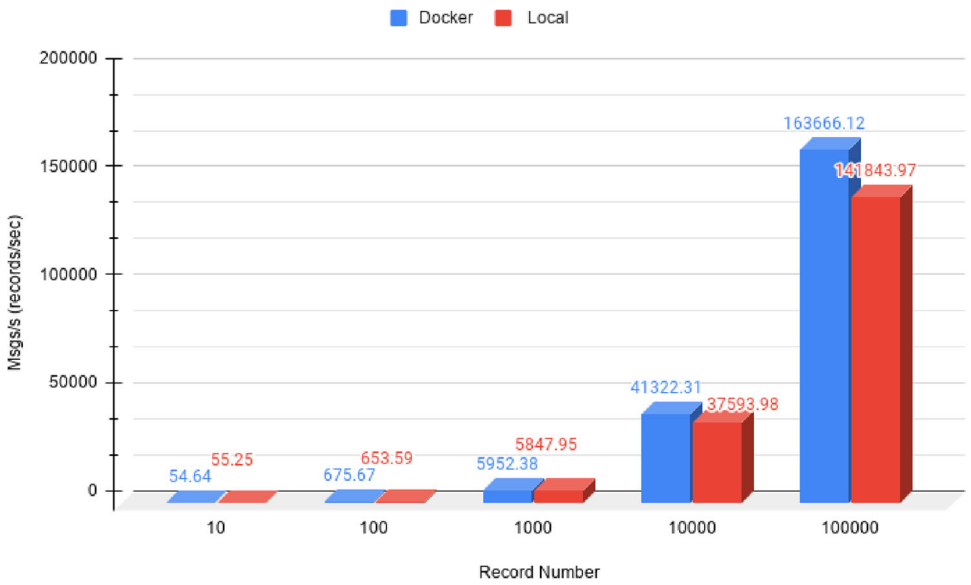

Fig. 5 Throughputs of platform-based distributed stream processing

Acknowledgements The work presented in this article took the first place in the Tübitak 2242 University Student Software Project Competition, Smart Cities and Transportation Categories (2019).

Author Contributions **AS** devised the project, the main conceptual ideas and the proof outline. **SK** and **SK** contributed to the design and implementation of the research, the analysis of the results, and the writing of the manuscript.

Funding Open access funding provided by the Scientific and Technological Research Council of Türkiye (TÜBİTAK). This research received no specific grant from any funding agency in the public, commercial or non-profit sectors.

Availability of data and materials Data will be available on a reasonable request.

Declarations

Competing interests The author declares that he has no known competing financial interests or personal relationships that could have appeared to influence the work reported in this document.

Ethical Approval It is not applicable.

Consent to participate Informed consent was obtained from all individual participants included in the study.

Consent for publication Participants have consented to the submission of the article to the journal.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Draghici, S.: A neural network based artificial vision system for licenseplate recognition. Int. J. Neural Syst. **8**, 113126 (1997)
2. Cherng, S., Fang, C.Y., Chen, C.P., Chen, S.W.: Critical motion detection of nearby moving vehicles in a vision-based driver-assistance system. IEEE Trans. Intell. Transp. Syst. **10**(1), 7082 (2009)
3. Omitaomu, O.A., Ganguly, A.R., Patton, B.W., Protopopescu, V.A.: Anomaly detection in radiation sensor data with application to trans-portation security. IEEE Trans. Intell. Transp. Syst. **10**(2), 324334 (2009)
4. Lotufo, R.A., Morgan, A.D., Johnson, A.S.: Automatic number-plate recognition. Inst. Elect. Eng. Colloquium on Image Analysis forTransport Applications, pp. 6/16/6, (1990)
5. Huang, Y.P., Chen, C.H., Chang, Y.T., Sandnes, F.E.: An intelligentstrategy for checking the annual inspection status of motorcycles basedon license plate recognition. Expert Syst. Appl. **36**(5), 92609267 (2009)
6. Chen, B., Cheng, H.H.: A review of the applications of agent technology in traffic and transportation systems. IEEE Trans. Intell. Transp. Syst. **11**(2), 485497 (2010)
7. Yung, N.H.C., Au, K.H., Lai, A.H.S.: Recognition of vehicleregistration mark on moving vehicles in an outdoor environment. Proc. IEEE Int. Conf. Intell. Transp. Syst., 418422 (1999)
8. Huang, Y.S., Weng, Y.S., Zhou, M.C.: Critical scenarios and theiridentification in parallel railroad level crossing traffic control systems. IEEE Trans. Intell. Transp. Syst. **11**(4), 968977 (2010)
9. Yamaguchi, K., Nagaya, Y., Ueda, K., Nemoto, H., Nakagawa, M.: Amethod for identifying specific vehicles using template matching. Proc. IEEE Int. Conf. Intell. Transp. Syst. 813 (1999)
10. Karsli, M., Satilmiş, Y., Şara, M., Tufan, F., Eken, S., Sayar, A.: End-to-end learning model design for steering autonomous vehicle: 26th Signal Processing and Communications Applications Conference (SIU). Izmir **2018**, 1–4 (2018)
11. Satılmış, Y., Tufan, F., Şara, M., Karslı, M., Eken, S., Sayar, A.: CNN based traffic sign recognition for mini autonomous vehicles. Information Systems Architecture and Technology: Proceedings of 39th International Conference on Information Systems Architecture and Technology - ISAT 2018. Advances in Intelligent Systems and Computing, vol 853 (2018)
12. Kul, S., Eken, S., Sayar, A.: evaluation of real-time performance for bgslibrary algorithms: a case study on traffic surveillance video. 2016 6th International Conference on IT Convergence and Security (ICITCS), Prague, pp. 1–4. (2016)
13. Hintjens, P., ZeroMQ.: Messaging for many applications. O'Reilly Media, Inc. (2013)
14. Snyder, B., Bosanac, D., Davies, R.: Introduction to apache activemq. Active MQ in Action, pp. 6-16 (2011)
15. Richardson, A., et al.: Introduction to rabbitmq-an open source message broker that just works. Google London, London, UK (2008)
16. Nannoni, N.: Message-oriented middleware for scalable data analytics architectures. Master's Thesis, KTH - Information and Communication Technology School, (2015)
17. Kreps, J., Narkhede, N., Rao, J., et al.: Kafka: A distributed messaging system for log processing. In: Proceedings of 6th International Workshop on Networking Meets Databases (NetDB), Athens, Greece, (2011)
18. Dong, Z., Wu, Y., Pei, M., Jia, Y.: Vehicle Type Classification Using a Semisupervised Convolutional Neural Network. IEEE Trans. Intell. Transp. Syst. **16**(4), 2247–2256 (2015)
19. Kul, S., Eken, S., Sayar, A.: Measuring the Efficiencies of Vehicle Classification Algorithms on Traffic Surveillance Video. In Proc. of International Conference on Artificial Intelligence and Data Processing, pp. 1-6, IEEE, Malatya (2016)
20. Sentaş, A., et al.: Performance Evaluation of Support Vector Machine and Convolutional Neural Network Algorithms in Real-Time Vehicle Type Classification. In: Barolli, L., Xhafa, F., Javaid, N., Spaho, E.,Kolici, V. (eds.) Advances in Internet, Data &Web Technologies. EIDWT 2018. Lecture Notes on Data Engineering andCommunications Technologies, vol. 17. Springer, Cham (2018)
21. Sentas, A., Tashiev, I., Kucukayvaz, F., et al.: Performance evaluation of support vector machine and convolutional neural network algorithms in real-time vehicle type and color classification. Evol. Intel. **2018**, 1–9 (2018)
22. Tashiev, İ., et al.: Konvolusyonel Sinir Agı Kullanarak Gerçek Zamanlı Araç Tipi Sınıflandırması Real-Time Vehicle Type Classification Using Convolutional Neural Network'., 1. Ulusal Bulut Bilisim ve Büyük Veri Sempozyumu B3S'17At: Antalya, Türkiye (2017)

23. Redmon, J., Farhadi, A.: YOLO9000: Better, faster, stronger. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition(CVPR), pp. 6517-6525 (2017)
24. Kul, S., Eken, S., Sayar, A.: A concise review on vehicle detection and classification. 2017 International Conference on Engineering and Technology (ICET), Antalya, pp. 1-4 (2017)
25. Gonçalves, G.R., Menotti, D., Schwartz, W.R.: License Plate Recognition Based On Temporal Redundancy, (2016)
26. Laroca, R., Severo, E., Zanlorensi, L.A., Oliveira, L.S., Gonçalves, G.R., Schwartz, W.R., Menotti, D.: A robust real-time automatic license plate recognition based on the YOLO detector. In: 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1-10, (2018)
27. Redmon.: Darknet: Open source neural networks in C. http://pjreddie.com/darknet/ (2018)
28. Yonetsu, S., Iwamoto, Y., Chen, Y.-W.: Two-Stage YOLOv2 for accurate license-plate detection in complex scenes. 2019 IEEE International Conference on Consumer Electronics (ICCE) pp. 1-4 (2019)
29. Nam, Y.C., Nam, Y.: Vehicle classification based on images from visible lightand thermal cameras. EURASIP J. Image Video Process. 2018(5) (2018)
30. Selmi, Z., Halima, M.B., Alimi, A.M.: Deep learning system for automaticlicense plate detection and recognition. Proc. Int. Conf. on Document Analysisand Recognition (ICDAR), Japan, pp. 1132-1137 (2017)
31. Pu, D., Gu, N., Zhang, X.: A robust and real-time approach for license plate detection. 2018 14th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD) pp. 1-7 (2018)
32. Liu, W., Anguelov, D., Erhan, D., et al.: Ssd: Single shot multibox detector. European conference on computer vision. Springer, Cham, pp. 21-37 (2016)
33. S. Abdullah, M., Hasan, M., Islam, S.M.S.: YOLO-Based three-stage network for bangla license plate recognition in Dhaka Metropolitan City. 2018 International Conference on Bangla Speech and Language Processing (ICBSLP), Sylhet, pp. 1-6 (2018)
34. Silva, S.M., Jung, C.R.: License Plate Detection and Recognition in Unconstrained Scenarios. In: Ferrari V., Hebert M., Sminchisescu C., Weiss Y. (eds) Computer Vision - ECCV 2018. ECCV 2018. Lecture Notes in Computer Science, vol 11216 (2018)
35. Kreps, J., Narkhede, N., Rao, J., et al.: Kafka: A distributed messaging system for log processing. In Proceedings of the NetDB, pp. 1-7, (2011)
36. RabbitMQ documentation [online]. http://www.rabbitmq.com/documentation.html. Accessed 21 Mar 2019
37. Curry, E.: Message-oriented middleware. InMiddleware for Communications, Mahmoud, Q.H., (ed). John Wiley &Sons, Ltd.: Chichester, UK, 22 Mar 2019 (2019)
38. The Apache Software Foundation. Apache ActiveMQ – Index. Apache ActiveMQ http://actiemq.apache.org/. Accessed 22 Mar 2019
39. ZeroMQ documentation [online], https://zeromq.org/. Accessed 22 Mar 2019
40. Kestrel [online], https://github.com/twitter-archive/kestrel. Accessed 22 Mar 2019
41. Goodhope, K., Koshy, J., Kreps, J., Narkhede, N., Park, R., Rao, J., Ye, V.Y.: Building linkedIn's real-time activity data pipeline. IEEE Data Eng. Bull. **35**, 33–45 (2012)
42. Kim, Y.K., Jeong, C.S.: Large Scale Image Processing in Real-Time Environments with Kafka. (2017)
43. Jung, S., Kim, Y., Hwang, E.: Real-time car tracking system based on surveillance videos. EURASIP J. Image Video Process. **28**(1), 133 (2018)
44. Chen, H., Luo, F., Zhao, L., Li, Y.: Design and implementation of real-time video big data platform based on spark streaming. (2018)
45. Kul, S., Eken, S., Sayar, A.: Distributed and collaborative real-time vehicle detection and classification over the video streams. Int. J. Adv. Robot. Syst. **2017**, 1–12 (2017)
46. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. CoRR, abs/1804.02767, (2018)
47. Tzutalin. LabelImg. (2015) https://github.com/tzutalin/labelImg

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Seda Kul received the B.S, M.S., and PhD degrees in computer engineering from Kocaeli University, Kocaeli, Turkey, in 2015, 2018, and 2021 respectively. She is also working as a Senior Researcher at TUBITAK, Kocaeli, Türkiye. Her research interests include distributed systems, image processing, and machine learning algorithms.



Sarper Kumcu obtained his Bachelor of Science degree in Computer Engineering from Kocaeli University, Kocaeli, Turkey, in 2019. Since obtaining his degree, he has pursued a professional career as a Software Architect, collaborating with various companies.



Ahmet Sayar is a faculty member at Kocaeli University, Department of Computer Engineering. He received his B.S. in Management Engineering from Istanbul Technical University (ITU), his M.S. in Computer Science from Syracuse University, and his Ph.D. in Computer Science from Indiana University. He has worked at Istanbul Gas Distribution Company (IGDAS) in Turkey, Los Alamos National Laboratories (LANL), and NASA Jet Propulsion Labs (JPL) in the USA. His current research interests are distributed systems, data science, big data, artificial intelligence, image processing, data-intensive computing, geographic information systems, and intelligent traffic systems.