Received April 14, 2021, accepted May 26, 2021, date of publication June 3, 2021, date of current version June 15, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3085736

Event-Based Microservices With Apache Kafka Streams: A Real-Time Vehicle Detection System Based on Type, Color, and Speed Attributes

SEDA KUL, ISABEK TASHIEV, ALI ŞENTAŞ, AND AHMET SAYAR
Department of Computer Engineering, Kocaeli University, 41001 Kocaeli, Turkey

Corresponding author: Seda Kul (seda.kul.06@gmail.com)

This work was supported by the TUBITAK under Grant 116E202.

ABSTRACT The work presented in this paper proposes a novel approach to tracking a specific vehicle over the video streams published by the collaborating traffic surveillance cameras. In recent years, smart, effective transportation systems and intelligent traffic management applications are among the topics that have been given importance by various institutions. Developing a scalable, fault-tolerant, and resilient traffic monitoring system that retrieves video chunks with the desired query is challenging. For these challenging problems, stream processing and data retrieval systems have been developed over the years. However, there are still existing shortcomings between users and retrieval systems. This paper investigates the problem of retrieving video chunks by key-value query based on publish/subscribe model. Thus, we propose a hybrid of an asynchronous and synchronous communication mechanism for the Event-Based Microservice framework. We aim to develop generic techniques for better utilization of existing platforms. In the proposed framework, (i) first of all, microservices detect vehicles and extract their type, color, and speed features, and stored them in the metadata repository. (ii) Microservices publish each feature as events (iii) Other microservices self-join subscribe to those events, which leads to more events being published by combing all the possibilities: type-color, type-speed, color-speed, and type-color-speed. Finally, (iv) the system visualizes the query result and system status in real-time. When the user has selected color or/and a type or/and a speed feature, the system will return the best-matched vehicles without re-processing the videos. Experimental results show that our proposed system filters messages in real-time and supports easy integration of new microservices with the existing system.

INDEX TERMS Intelligent transportation system (ITS), microservices, publish-subscribe system, stream computing, vehicle detection.

I. INTRODUCTION

Intelligent Transportation System (ITS) is becoming pervasive and actively used in recent years to increase traffic efficiency, decrease traffic congestion, and provide road safety. ITS uses communication technologies such as various sensors and cameras to produce useful information for operators. Implementing electronics, wireless and communication technologies on roads is costly. Therefore, surveillance camera systems such as Closed-circuit television (CCTV) and IP cameras have become widespread and actively used. The operators monitor surveillance camera feeds in real-time or recorded video. The need for access to up-to-date, accurate, and relevant data increases day by day. For this reason, how to retrieve and distribute live video surveillance streams to users as per their interest promptly is one of the biggest challenges.

Current video retrieval studies are mostly based on Text-based and Content-based indexing methods. Content-Based image retrieval (CBIR) is also known as query by image. In the CBIR method, the system extract features from each image and stores them in a database. When the user queries an image, the feature vector of the query image is extracted. The similarity between the feature vector of the query image and images in the database is measured. However, this makes it very slow to be used in real-time [1] due to working with the high dimensional vectors. Then, the system returns the images that most closely resemble the query image [2].

Unlike CBIR, the Text-Based Image Retrieval (TBIR) method annotates the images with file name, image size, dimension, and format. Then, it stores them in a database [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Razi Iqbal.

The first disadvantage of the TBIR is creating metadata manually for an extensive database is impractical. The second drawback is: How to find an accurate representation for both text captions and video data? Various methods for representing text for encoding captions [4]–[8] depend on the annotator. However, the annotator can give different explanations for similar images. Also, text captions will be language-dependent [9]. The common feature of video retrieval by query systems in the literature is that when the query incomes, the system initiates a search on the database's recorded data. We propose a topic-based query framework to prevent data from being re-processed repeatedly (for each query). Although video retrieval and automatic analysis systems exist for traffic monitoring applications and real-time vehicle detection/tracking systems to the best of our knowledge, no system allows us to collaboratively query vehicles in real-time according to their physical characteristics key-value matches with AND/OR combinations and publish result to user. Unlike other studies, we do not search on recorded data. We shorten the query time by pre-classification. When the user has selected vehicle type and/or color and/or speed, the system automatically subscribes to the relevant topic.

Thus, instead of watching the whole video, the operator only examines the relevant part based on his query. Topics act as filters, and each filter is based on the physical attributes of the vehicles (type, color, and speed).

Publish/subscribe architecture has been used to filter and distribute live video surveillance streams to users as per their interest promptly. A pub/sub method is a messaging paradigm where in charge of delivering events to interested subscribers [10], [11]. Messages are produced to "topics" by publishers and consume by consumers who subscribe to the topics. Event brokers and zookeepers that are part of the system are responsible for routing messages between producers and consumers. Cao *et al.* [12], a pub/sub system introduced recently, and companies worldwide use it to create event-driven microservices, real-time streaming applications, and streaming data pipelines. Besides, it can handle real-time data feeds and provides low-latency and a high-throughput platform.

The contributions of this paper are summarized as follows:

- Thus, unlike other studies, the work presented here is a distributed and collaborative computing environment capable of doing text-based queries over real-time video streams. Vehicle detection done over the real-time video streams is obtained by the collaboration of many distributed services, and those services are grouped according to their tasks. Some services get raw streams from cameras and apply color filtering, and creates sub-streams of colors. Some filters get color streams and apply speed filtering and get sub-streams of color-speed. Some other services get color-speed streams and apply size filtering, and creates color-peed-size streams. All the messaging between the collaborative filters is done by publish-subscribe messaging paradigm enabling persistent communication by using queue structures. This approach makes real-time video monitoring scalable and responsive.
- We design a hierarchical pub/sub microservice data structure for retrieval of events
- We evaluate the proposed approach for vehicle detection and color classification
- We developed an online user interface system for the user to search vehicles and monitoring system status dynamically.

The paper's remainder is organized as follows: Section 2 presents related works about image processing-based vehicle classification studies, color detection, and speed detection, message delivery services, and publish/subscribe applications on streaming data. In Section 3, the proposed method is explained. Experiment results are shown in Section 4, and the conclusion is given in Section 5.

II. RELATED WORK

The issue of managing and querying dynamic data is relevant to several research areas. Most database management systems (DBMS) have failed to process video data dynamically. There is no application or project-based work in the literature based on retrieve video clips by text query in real-time. However, the study's sub-techniques, vehicle type, color, speed detection, text-video retrieval systems, and publish/subscribe applications are listed and explained as follows.

A. VEHICLE CLASSIFICATION BASED ON TYPE, COLOR, AND SPEED ATTRIBUTES

Vehicle detection and classification hold an essential place in traffic surveillance systems and traffic safety. Therefore, it has been an important area for researchers. Many studies have been carried out with machine learning and deep learning algorithms in the field of vehicle classification [13], [14].

Wang *et al.* [15] aimed to classify cars and trucks in real-time using faster region-convolutional neural networks (Faster R-CNN), reaching 90.65 and 90.51% accuracy. Chakraborty *et al.* [16] presented traffic congestion detection over CCTV cameras in real-time using three different approaches: you only look once (YOLO), deep convolutional neural network (DCNN), and support vector machine (SVM) to compare results. YOLO, DCCN, and SVM achieved 91.5%, 90.2%, and 85.2% accuracy. Kul *et al.* [17] used binary image features for the classification of vehicles. These features were tested with ANN, SVM, and Adaboost classification algorithms and their accuracy values are 87.5, 81.6, and 85.4 respectively. Chen *et al.* [18] conducted studies to classify road vehicles using images obtained from CCTV cameras; they compared two different classification algorithms (SVM and random forests) using the vehicle's shapes and dimensions. With a total of four vehicle classes (cars, minibusses, buses, and bicycles/motorcycles), the results showed that SVM performed better than RF with an accuracy of 96.26%. The highest misclassification rate emerged among automobiles and minibusses, where both size and shape are similar.

Some current studies on the color detection of vehicles are as follows. Wu *et al.* [19] proposed a real-time vehicle color recognition system using You Only Look Once (YOLO) 9000, a state-of-the-art, real-time object detection system for intelligent transportation systems applications. Vitabile *et al.* [20] proposed a dynamic HSV subfield of processed images. Seng *et al.* [21] classified vehicle color based on DCNN. In this method, vehicle colors are divided into six categories (white, black, red, green, blue, and yellow), and their dataset contains 3520 samples (2530 for training, 1000 for testing). They achieved 92.68% overall accuracy. Rachmadi and Purnama [22] proposed a CNN-based vehicle color detection system. The dataset used in the experiment had five color classes, including black, blue, white, red, and yellow and reached 94.47% overall accuracy. Chen *et al.* [23] proposed a vehicle color recognition system in which their goal was to select the region of interest (ROI) for recognizing a vehicle's dominant color. Linear Support Vector Machine was used for the classification process, and they reached 91.89% overall accuracy. Şentaş *et al.* [24], [25] proposed a vehicle type and color classification method; they classified buses, minivans, microbuses, SUVs, sedans, and trucks using the YOLO classification algorithm and reached 97.90% precision, 99.60% recall, and 89.29% IOU.

Vehicle speed detection is an essential measure for traffic safety, and there are many studies in the literature to determine or estimate the speed of moving objects. These studies usually either use extra hardware or solve the problem with image processing techniques [26]. Wilder *et al.* [27], Pelegri *et al.* [28], and Koide *et al.* [29] used computer software with magnetic sensors to detect vehicle speed. Koide *et al.* [30] proposed a laser-based continuous detection system to detect speed in real-time. Osman and Chahine [31] used microwave signals. Çelik and Kusetoğullar? [32] proposed a method to detect vehicles that violate the speed limit, but the system showed poor performance when shadows increased in the input video. Aliane *et al.* [33] proposed a system that warns the driver in acoustic messages from the car speakers to the driver when the speeding limit is violated. If the driver continues to infringe the speed limit, the vehicle's GPS location and the captured image are saved in the database. However, it has been observed that this car may not be useful because the cost of the equipment is prohibitive.

B. VIDEO RETRIEVAL SYSTEMS FROM TEXT QUERY

CBIR calculates the similarity between the query image and sources in the database. TBIR annotates the images with a filename, tag, image size, or dimension and stores them in a database. Removal of stop words, stemming process are next step. After the NLP step, the query is processed in the database [34].

Patel [35] proposed a video retrieval system for automated object-based indexing queries. Yang *et al.* [36] proposed a retrieval based on object-specific characteristics. Kantorov and Laptev [37] used the motion information of object to characterize events to allow video clips retrieval. Kar and Kanungo [7] proposed a query method based on trajectory extraction and encoding relationships between objects. In [2], [38], authors proposed a pre-trained Convolutional Neural Network (CNN) to detect objects in video frames. For query representation, they proposed complex linguistic rules for extracting relevant parts from video data. All these mentioned studies process the query with several NLP methods and then perform a query on the database.

Liu *et al.* [39] proposed a system for retrieving vehicle surveillance videos. Authors retrieve vehicle images by using vehicle color similarity or/and type. They stored the vehicle's image and its color feature in a database. When a user selected vehicles color or/and type feature, the system returns the most similar images. However, this system is not working in real-time. Baran *et al.* [40] proposed a method for vehicle tracking systems. They specified model recognition, license plate recognition, and color recognition features for detecting vehicles. They extracted descriptor vectors and vocabulary tree created.

Unlike these studies, in our proposed framework, we use a key-value representative query rule for the video-retrieval system in real-time. Thus, without high dimensional vectors, NLP methods, or calculating similarity between query and target, this method allows us to match video chunks with the text in the concept space.

C. PUBLISH-SUBSCRIBE COMMUNICATION MODELS

Apache Kafka is a publish-subscribe distributed messaging system and an event streaming platform [41]. In this section, we observed how to leverage Apache Kafka and implement a stream processing system in practice. Kafka performs data ingestion tasks and makes it possible to perform machine learning algorithms in real-time. We aim to improve the efficiency of a real-time stream data process.

There are several hybrid pub/sub messaging pattern-based machine learning methods in the literature. Kul *et al.* [42] proposed a middleware system based on the publish-subscribe messaging system. They classified the vehicles as small, medium, and large, with the Neural Network, AdaBoost, and SVM classifiers based on their size. They got the best result with 95% accuracy from the NN algorithm. Then, these classified vehicles were published over the network for subscribers to obtain information via Java Message Servers. Banno *et al.* [43] have developed a distributed event-based system for locally consuming locally generated messages. Gascon-Samson *et al.* [44] have demonstrated that a standard, dynamic, scalable, channel-based publish-subscribe system can be implemented in the cloud. They tested the efficiency of the system according to the number of clients active at the same time. Gray & Nutt proposed publish-subscribe architecture that allows the distribution of the data stream and querying it [45]. Zou *et al.* [46] proposed a publish-subscribe model for real-time video surveillance system by using wireless sensor networks.

III. DATASET

Three publicly available data sets are used to determine vehicle type (BIT-Vehicle) [47], color [48], and speed [49].

The Beijing Institute of Technology University Media Computing and Intelligent Systems Laboratory staff and shared academic studies [50] were preferred in determining vehicle types. Includes 9,850 vehicle images in different lighting conditions (about 10% night), different resolutions (1600 × 1200 and 1920 × 1080), and different viewing angles.

All vehicle images are presented in six categories: buses (558), minibusses (883), trucks (822), vans (476), sedans (5,922), and SUVs (1,392). Figure 1 contains some sample images of the BIT Vehicle data set.

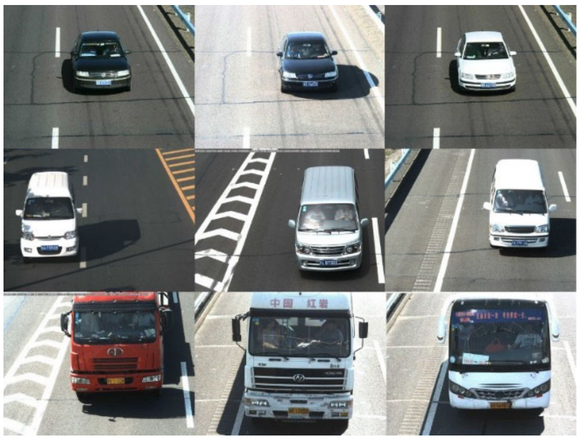


FIGURE 1. Sample images of the BIT vehicle data set.

The dataset published by Chen *et al.* [48] was used to detect vehicle colors. In this data set, there are only images of vehicles belonging to different colors. These are 3.442 black, 1.086 blue, 281 cyan, 3.046 gray, 482 green, 1.941 red, 4.742 white, and 581 yellow. Since the data set is unbalanced, we used the Image Data Generator [50]. It artificially creates the training images through different processing methods or multiple operations such as random rotation, shifting, cutting, and rotating. Thus, we have obtained training data as follows 3442 black, 3027 blue, 2939 gray, 3046 green, 3101 red, 3056 white, and 4742 yellow. Figure 2 contains some sample images of the data set.

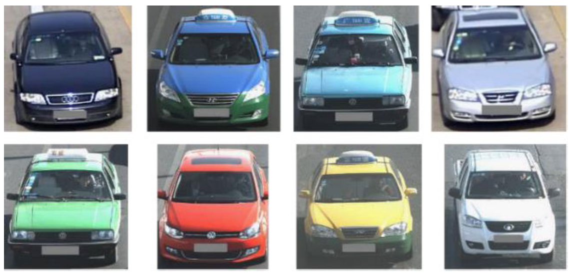
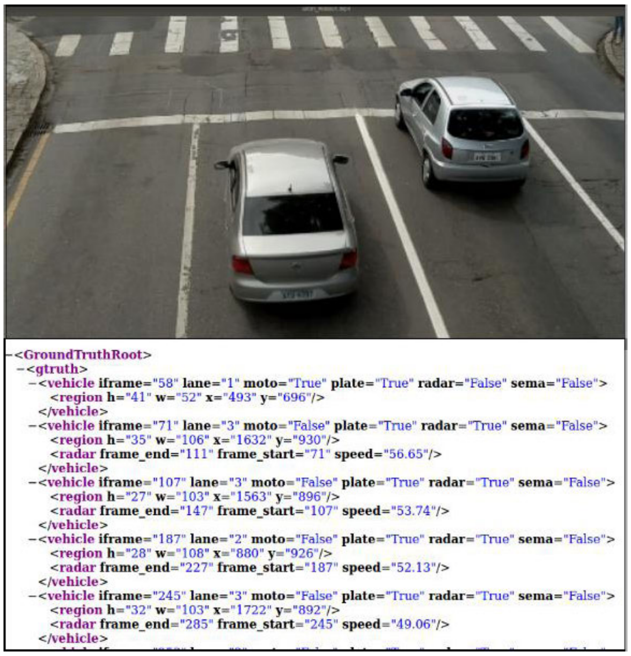


FIGURE 2. Sample images of vehicle color dataset.

Finally, the dataset published by Garibotto *et al.* [49] was used to determine vehicle speed. A 5-megapixel CMOS image sensor was used to capturing the videos. There are 20 videos with a resolution of 1920 × 1080 at 30.15 FPS. Each video image has an XML file. This file contains actual speed values (ground truth) obtained from a speedometer with high precision based on an inductive loop detector, certified by the Brazilian national metrology agency. In addition, there are location values of the plates of the vehicles. Figure 3 contains some sample images of the data set [49].



```
-<GroundTruthRoot>
 -<gtruth>
  -<vehicle iframe="58" lane="1" moto="True" plate="True" radar="False" sema="False">
     <region h="41" w="52" x="493" y="696"/>
   </vehicle>
  -<vehicle iframe="71" lane="3" moto="False" plate="True" radar="True" sema="False">
     <region h="35" w="106" x="1632" y="930"/>
     <radar frame_end="111" frame_start="71" speed="56.65"/>
   </vehicle>
  -<vehicle iframe="107" lane="3" moto="False" plate="True" radar="True" sema="False">
     <region h="27" w="103" x="1563" y="896"/>
     <radar frame_end="147" frame_start="107" speed="53.74"/>
   </vehicle>
  -<vehicle iframe="187" lane="2" moto="False" plate="True" radar="True" sema="False">
     <region h="28" w="108" x="880" y="926"/>
     <radar frame_end="227" frame_start="187" speed="52.13"/>
   </vehicle>
  -<vehicle iframe="245" lane="3" moto="False" plate="True" radar="True" sema="False">
     <region h="32" w="103" x="1722" y="892"/>
     <radar frame_end="285" frame_start="245" speed="49.06"/>
   </vehicle>
```

FIGURE 3. Sample images of vehicle speed dataset.

IV. PROPOSED METHOD

Figure 4 briefly shows the architecture of the proposed framework. Surveillance video cameras capture and transmit data to microservices. Microservices then filter vehicles based on their type, color, speed and publish data into the relevant topic. Merger-classifiers self-join these topics and combine all possibilities. Thus, type-color, type-speed, color-speed, and type-color-speed topics are formed. Therefore, when the subscribers make a query, the system subscribes the user to the relevant topic. All the messaging between the collaborative filters are done by the publish-subscribe messaging paradigm enabling persistent communication by using queue structures.

The messaging protocol Apache Kafka a publish/subscribe middleware system is realized by using Microservices and containerization architectures for flexibility and scalability features. Kafka publish-subscribe system acts as a central, mission-critical nervous system. The realization of the system function is divided into two steps. The first step is to obtain vehicle information based on video automatically, and the second step is to stream vehicle information to

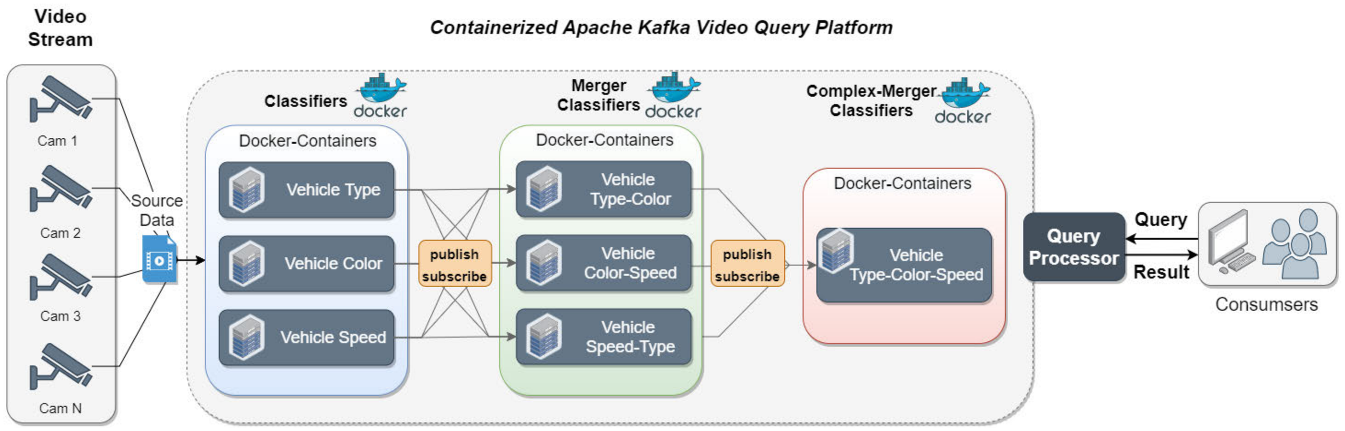


FIGURE 4. Diagram of proposed framework.

subscribers. Since our study includes two main modules as classification and messaging system, these stages will be explained under subheadings in this section.

A. VEHICLE TYPE CLASSIFICATION

In this paper, two different architecture used for type classification YOLOv4 [50] and tiny-YOLOv4. YOLO does the feature extraction step by itself, which saves classification systems from feature extraction steps. Also, it makes our system robust against camera movement or changes in the environment. The most significant advantage of using YOLOv4 is its excellent speed; it can process 45 frames per second.

The CNN family, one of the deep learning methods, primarily uses regions to position objects within the image.

The network does not observe the whole image; it only looks at parts of it that may have a higher chance of containing a searched object. On the other hand, The YOLOv4 framework takes the entire image in a single instance. YOLOV4 estimates the bounding box coordinates and their class probabilities.

The backbone is the first layer of YOLOv4, also known as the feature-extraction process. We used two different backbone architectures. First is Tiny-YOLO, which consists of 9 convolutional layers. The second is Darknet53 consists of 53 convolutional layers.

To realize vehicle detection and classification based on YOLO v4, the main steps are as follows.

- Data organization. All images manually labeled for the data set to be trained in YOLOv4, "LabelImg" graphical image annotation tool [51] used. Annotations are saved in YOLO format.
- Parameter tuning and training of the YOLOv4 and YOLOv4-tiny shown in table 1.

B. VEHICLE COLOR DETECTION

CNN, a popular Deep Learning method classifier used mostly to classify the objects based on the information of the shape. We chose CNN, for color determination based on Alexnet architecture. 8 different color class labels were created as black, blue, cyan, gray, green, red, yellow, and white. The model architecture is given in Figure 5 and Table 2. Our CNN model consists of 5 convolutional, 5 max-pooling, 1 flatten, 6 drop out, and 3 dense layers. Mean square error for loss function and Adam optimizer were used.

TABLE 1. Model parameters.

Parameters	Value
Input Size	288 x 288
Learning Rate	0.001
Bath Size	64
Subdivision	16
Max Batches	12000
Steps	9600,10800
Filter	33

The loss function as shown in Eq. (1), ŷ is the predicted value, y is the target and N is the sample number.

$$L(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{N}(y - \hat{y}_i)^2 \tag{1}$$

C. VEHICLE SPEED DETECTION

Vehicle speed detection from a single camera is challenging because the camera parameters such as angle, height, etc., may not be known, and determining them is difficult.

We made our calculations on a road whose distance is known to solve the problem. To detect the speed of vehicles, vehicle detection and tracking processes must be performed first. The vehicle tracking process obtains the distance taken by the vehicle, and also, we know that the rate of this value with time gives us the speed of the vehicle. We used the YOLOv4 algorithm to detect vehicles. Since no background subtraction method is used during vehicle detection, vehicle queues, slow traffic conditions, and long waiting times on the road do not cause any problem for our work.

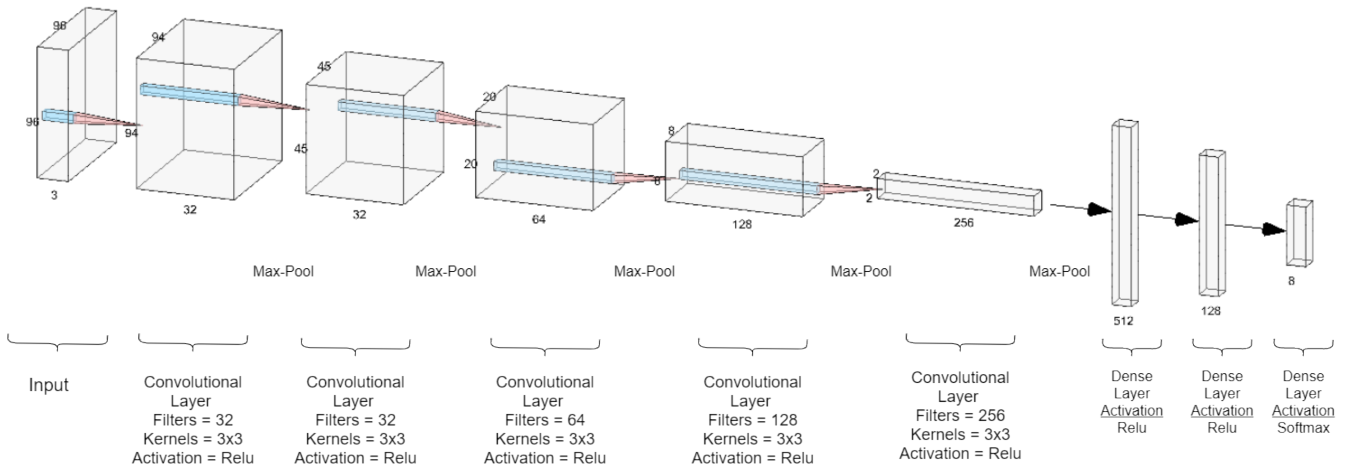

96
96
3
94
94
32
45
45
32
20
20
64
8
8
128
2
2
256
512
128
8
Max-Pool
Max-Pool
Max-Pool
Max-Pool
Max-Pool
Input
Convolutional Layer Filters = 32 Kernels = 3x3 Activation = Relu
Convolutional Layer Filters = 32 Kernels = 3x3 Activation = Relu
Convolutional Layer Filters = 64 Kernels = 3x3 Activation = Relu
Convolutional Layer Filters = 128 Kernels = 3x3 Activation = Relu
Convolutional Layer Filters = 256 Kernels = 3x3 Activation = Relu
Dense Layer Activation Relu
Dense Layer Activation Relu
Dense Layer Activation Softmax

FIGURE 5. CNN architecture for vehicle color detection.

TABLE 2. Model architecture.

Layer Type	Layer Parameters
Conv2D	Filters = 32, kernel_size = (3, 3), Activation = relu
MaxPooling2D	Pool Size = (2, 2)
Dropout	0.25
Conv2D	Filters = 32, kernel_size = (3, 3), Activation = relu
MaxPooling2D	Pool Size = (2, 2)
Dropout	0.25
Conv2D	Filters = 64, kernel_size = (3, 3), Activation = relu
MaxPooling2D	Pool Size = (2, 2)
Dropout	0.25
Conv2D	Filters = 128, kernel_size = (3, 3), Activation = relu
MaxPooling2D	Pool Size = (2, 2)
Dropout	0.25
Conv2D	Filters = 256, kernel_size = (3, 3), Activation = relu
MaxPooling2D	Pool Size = (2, 2)
Dropout	0.25
Dense	Units = 512, Activation = relu
Dropout	0.25
Dense	Units = 128, Activation = relu
Dropout	0.25
Dense	Units = 8, Activation = softmax

Many algorithms proposed in the literature to follow moving objects in the video, called tracking algorithms. OpenCV (Open Source Computer Vision), an open-source image processing library, has been used in our work [52]. In OpenCV, there are implementations for the tracking algorithms such as Boosting [53], Goturn [54], KCF [55], MedianFlow [56], and TLD [57]. MedianFlow algorithm is used in the project. The MediaFlow algorithm treats the tracked object as a bounding box and pops a new bounding box to track the object in a new frame. Finally, vehicle speeds are measured by comparing the bounding box trajectories surrounding vehicles with real-world measurements.

We have defined the speed categories as follows 30-40, 40-50, 50-60, 60-70, 70-80, 80-90, and above 90 km/h.

D. HIERARCHICAL TOPIC BASED PUBLISH/SUBSCRIBE MICROSERVICES

The publish/subscribe concept [58] is a communication unit in a distributed system. The pub/sub messaging pattern consists of publishers (producers) and subscribers (consumers/clients). Unlike other messaging architectures, in the pub/sub architecture, messages are not sent to specific consumers. Instead, the messages post to specific categories called topics without any information about the consumers. Consumers likewise do not know anything about producers and receive messages by subscribing to topics of interest.

Apache Kafka is a distributed, partitioned, replicated publish/subscribe system. Kafka maintains the real-time stream of data in topics. Producers are the processes also as known as publishers that publish messages to topics. Likewise, subscribers are processes that read and process the messages from topics by subscribing to them. Kafka runs as a cluster and consists of one or more servers called brokers. Kafka uses another server called Zookeeper to provide coordination, management, and configuration for the brokers. We used Docker technology to establish the microservice architecture. It is known that there are different types of setups that provide a platform for running applications: servers, virtual machines, and Docker containers. Traditional servers have many disadvantages such as high cost, slow deployment time, complex configuration, etc. Likewise, virtual machines also have negative aspects, problematic resource allocation, and complex configurations. Docker uses OS-level virtualization to make applications portable and isolates them by packaging them in containers [59]. Kafka and Zookeeper were used to build the Apache Kafka architecture using containers on the Docker.

Communication of these two containers was achieved over the port numbers specified with Docker Compose shown in Table 3.

TABLE 3. Docker compose.

Pseudo Docker-Compose Yml File for Apache Kafka and Zookeeper

```
services
    zookeeper
        image: zookeeper
        ports:
          - 2181:2181
    kafka
        image: kafka
        environment:
            Kafka_Zookeeper_Connect:
            zookeeper1:2181
        ports:
          - 9092:9092
```

Considering the situation that partition and replication units in Kafka architecture may affect performance, we chose how many brokers, partitions, and replications we will use by evaluating them according to the performance (Records/sec (MB/sec) and Msgs/s(records/sec)) of the producer and the consumer. Results are given in section 4.

Kafka serves as a microservice and acts as a central nervous system. Each topic represents a class.

- There are six classes of vehicles; auto, bus, truck, minivan, minibus, and sedan.
- Eight class of color, red, green, gray, black, white, yellow, blue, and cyan
- Eight class of speed categories, below 30 km/h, 30-40, 40-50, 50-60, 60-70, 70-80, 80-90, and higher than 90 km/h).

There are three layers in our architecture: classifiers, mergers, and complex-mergers classifiers. These classifiers to work together and publish data, ensured that they were first informed of each other. Before Merger classifiers start working, they begin to wait for a signal to understand that the previous layer's classifiers are standing. Classifiers also send the ack signal to the mergers. After receiving the ack signal, merger servers start listening to the related topic. Algorithm 1 gives the pseudocode for listen-before-talk scheme between microservices.

Figure 6 shows architecture for listen-before-talk scheme between microservices.

Cameras start broadcasting after all servers send the ACK signal. Each classifier creates the data structure called a chunk. as shown Table 4.

As soon as the type classifier starts to receive images, it starts to classify images according to the query's incoming search. In this chunk, data structure, camera id, frame number, FPS information is kept and published to the relevant topical.

Algorithm 1 Architecture of Publish-Subscribe Framework

```
1   Publisher
2     If (is All Classifiers Available != True)
4       If (Classifier is Not Available)
5         send (CLASSIFIER_REQUEST)
6       else
7         send (CLASSIFIER _REQUEST_ACK)
8       If (Merger is Not Available)
9         send (MERGER_REQUEST)
10      else
11        send (MERGER_REQUEST_ACK)
12      If (Complex-Merger is Not Available)
13        send (CLASSIFIER_REQUEST)
14      else
15        send (MERGER_REQUEST_ACK)
```

TABLE 4. Data structure of activity descriptors.

Components	Values
camera_id	ID of Camera
chunk_id	ID of Video Chunk/Part
chunk.frames[]	Sequence of Frames
FPS	Frame Per Second
speed_start	Start Point of Speed Measurement
speed_end	End Point of Speed Measurement
speed_length	Total Distance for Speed Measurement
video_start_frame	Frame Start Number of Video Chunk
topic	Topic Name

It serves as metadata. At this point, Merger works as both a consumer and a publisher.

The working order of the mergers is as follows:

- Type-Color Classifier: Color classification is performed on the images taken from the type topic.
- Speed-Type Classifier: Type classification is performed on vehicles whose speed is calculated.
- Speed-Color Classifier: Color classification is performed on vehicles whose speed is calculated.
- Speed-Type-Color: The type and color of the vehicle is performed on vehicles whose speed is calculated.

The classifier (node) in each layer works depending on whether the previous layer's classifier is alive. For a merger node to publish data, the nodes that come before it must be up and running. If a node is not alive, all nodes start to listen to the Apache Kafka Broker port for the ACK signal. Here, we present the microservice algorithm that works synchronously with each other. Algorithm 2 gives the pseudocode for collaborative working pub/sub system.

We designed the User Interface Monitor so that the proposed system can be used by the end-user and monitor the system status. With this application interface developed using Javascript and HTML, information about the status of the system is obtained, and the classified images can be viewed. The Apache Kafka monitoring screenshot is given in

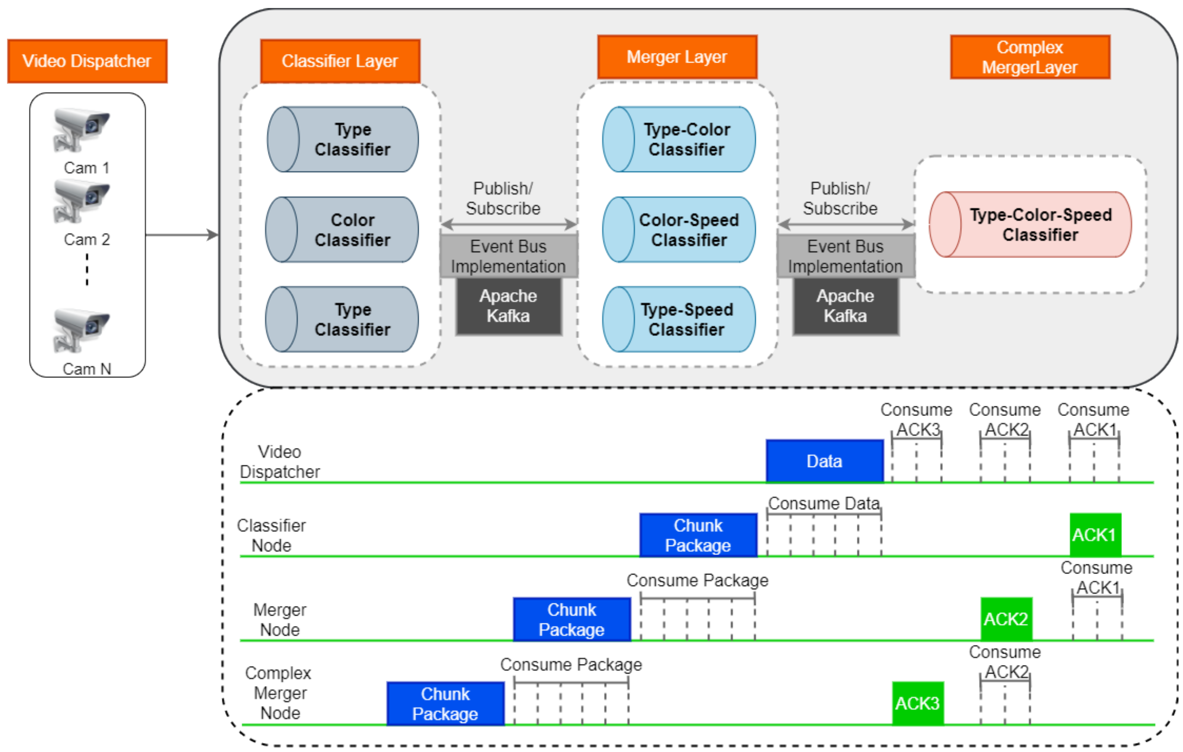


FIGURE 6. Architecture of publish-subscribe framework.

Algorithm 2 Architecture of Publish-Subscribe Framework

```
     Classifiers
1      If (is All Classifiers Available = True)
2          If (topic list = NULL)
3              then create topic for each classification node
4      end
5      Publish(Type-Classfication)
6      Publish(Color-Classfication)
7      Publish(Speed-Measurment)
     Merger TC
8      Subscribe(Type)
9      Subscribe(Color)
10     Publish(TC)
     Merger TS
11     Subscribe(Type)
12     Subscribe(Speed)
13     Publish(TS)
     Merger CS
14         Subscribe(Color)
15         Subscribe(Speed)
16         Publish(CS)
     Complex-Merger TCS
17         Subscribe(TC)
18         Subscribe(Speed)
19         Publish(TCS)
      end if
```

figure 7 [60]. With this monitoring page, we were able to keep information about the system's performance as well as its operation.

TABLE 5. YOLOV4, YOLOV4-Tiny performance and comparison different approaches on bit-vehicle dataset.

Vehicle Class	YOLOV 4	Tiny YOLOV 4	YOLOv 2 Vehicle [61]	Model Comp [61]	Faster R-CNN + ResNet [62]
Bus	97.80%	**97.80%**	97.54%	97.43%	90.62%
Microbus	96.43%	**98.33%**	93.76%	94.47%	94.42%
Minivan	95.06%	**98.18%**	92.18%	90.86%	90.67%
Sedan	99.12%	**99.19%**	98.48%	97.46%	90.63%
SUV	96.57%	**97.04%**	94.62%	93.05%	91.25%
Truck	97.23%	**99.81%**	92.09%	91.69%	90.07%
Average	97.04%	**98.39%**	94.78%	94.16%	91.28%

V. EXPERIMENTAL RESULTS

All our experiments were performed on a machine with Intel®Core^{TM} i9-9900KF CPU @ 3.60GHz × 16, GeForce RTX 2080 Ti/PCIe/SSE2, Ubuntu 18.04 OS. C++ was used as the programming language.

A. EXPERIMENTAL ANALYSIS OF VEHICLE DETECTION AND CLASSIFICATION

Intersection over Union (IOU), Average Precision, Recall, and F-Score metrics used to measure the accuracy of the model on the test dataset. Table 5 shows our test result for YOLOv4, YOLOv4-Tiny and compared them with the model results proposed by Sang *et al.* [61] and Sang *et al.* [62]. As shown in Table 5, we achieved higher AP of 98.39% with YOLOv4-Tiny. Our model shows better performance in all vehicle class of BIT-Vehicle Dataset. Table 6 shows

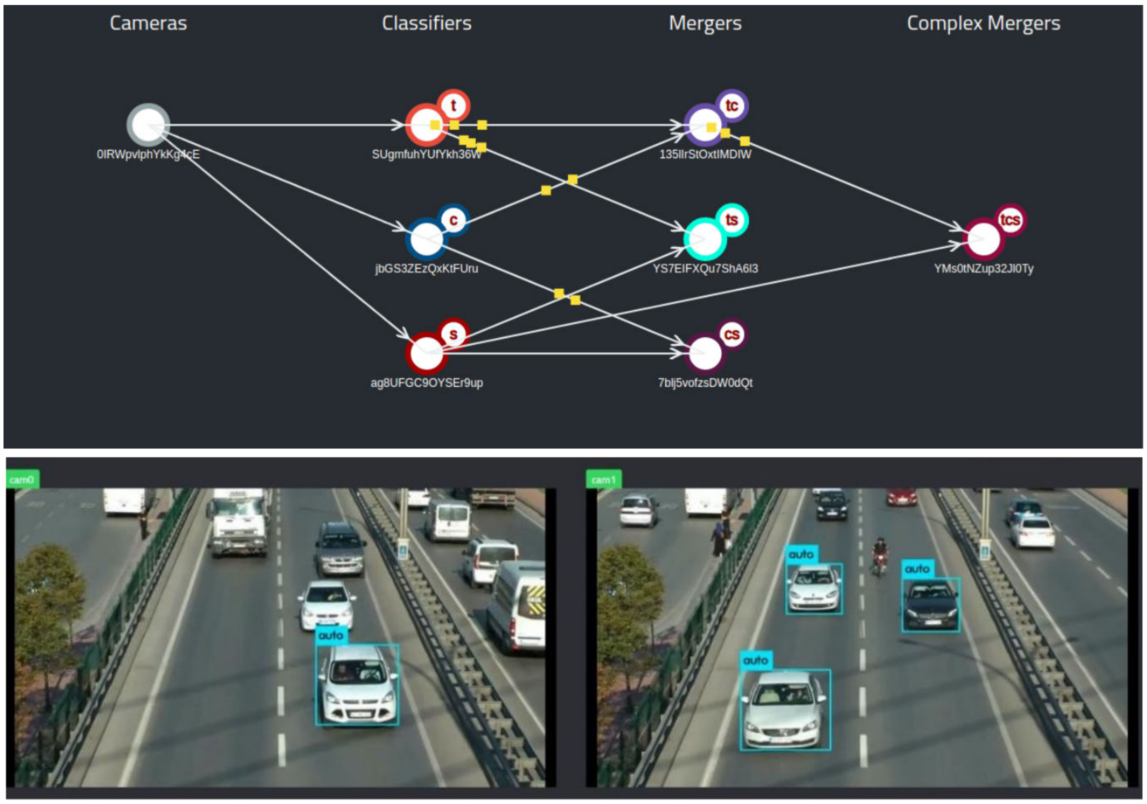


FIGURE 7. Examples of UI monitoring system.

TABLE 6. Execution time for Yolov4, Yolov4-Tiny model for each frame.

	YOLOV4	Tiny YOLOV4
Avg. time (ms)	263.29	87.58

Execution Time for Yolov4, Yolov4-Tiny Model for each frame.

The performance results of YOLOV4 and Tiny-YOLOV4 training completed up to 10000 epochs.

We achieved 98% Recall, 96% F1-score, and 86% Average IoU scores with YOLOV4. We achieved 98% Recall, 96% F1-score and 83.01% Average IoU scores with Tiny-YOLOV4.

B. EXPERIMENTAL ANALYSIS OF VEHICLE COLOR

We used two different color spaces RGB and HSV to measure the accuracy of the color model and to compare performance. We divided our data set according to 80%-20% rates as training and testing. Our input shape resized into 96 × 96 × 3 resolution.

Table 7 shows our test result with RGB and HSV color spaces and compared them with the model results proposed by Chen *et al.* [48] and Rachmadi and Purnama [22]. We achieved a higher average accuracy of 0.9563% with HSV color space. Our model shows better performance in Blue, Cyan, Green, Red, and Yellow colors.

TABLE 7. CNN performance on vehicle color [48].

Vehicle Color Class	Model Accuracy RGB	Model Accuracy HSV	Chen et al. [48]	Rachmadi et. al. [22]
Black	0.9723	0.9560	0.9730	**0.9738**
Blue	0.9513	**0.9579**	0.9535	0.9410
Cyan	0.9973	**0.9986**	0.9787	0.9645
Gray	0.8371	0.8555	0.8466	**0.8608**
Green	0.9468	**0.9544**	0.7884	0.8257
Red	0.9884	**0.9923**	0.9878	0.9897
White	0.9547	0.9453	0.9423	**0.9666**
Yellow	0.9752	**0.9905**	0.9553	0.9794
Average	0.9529	**0.9563**	0.9282	0.9447

We achieved 0.9534 f1-score, 0.9554 precision and 0.9529 recall scores with RGB, and 0.9563 f1, 0.9575 precision and 0.9568 recall scores with HSV color spaces.

Table 8 shows the execution time for RGB CNN Model and HSV CNN model for each frame.

C. EXPERIMENTAL ANALYSIS OF VEHICLE SPEED DETECTION

Our system calculates speeds based on the tracking method. With the data set, our system's assumption is the known distance between lines and fps values. Vehicle tracking is performed in a known environment and reference range.

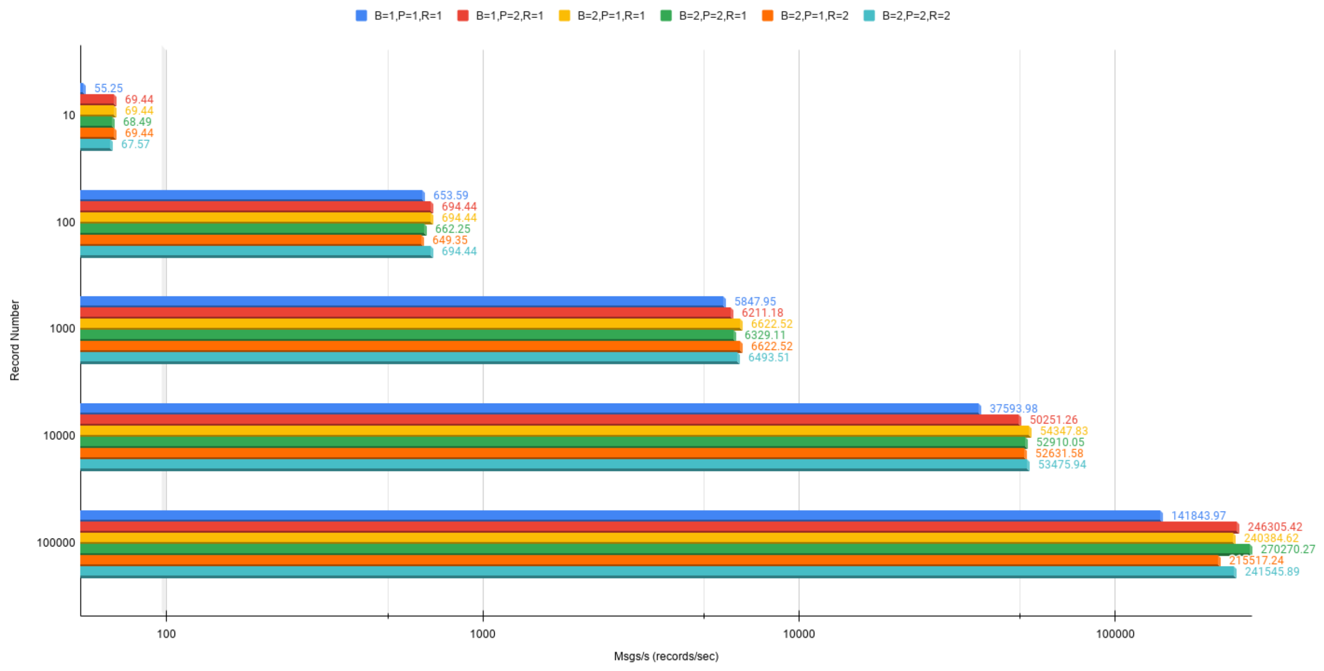


FIGURE 8. Experiment results of producer packet throughput according to record numbers.

TABLE 8. Execution time for CNN model for each frame.

	RGB CNN Model	HSV CNN Model
Avg. time (ms)	24.5	25.65

These reference points are 4.8 meters wide and 2.0 meters high. The location of the vehicle is monitored with the bounding box surrounding the vehicle. From the moment the bounding box enters the reference area, time is kept, and the average speed is calculated when the vehicle leaves the area.

The speed measurement performance has been evaluated by comparing calculated speed values with the ground truth. According to the USA's standards, a measurement should be within the error range of minus 3 km/h, +plus 2 km/h. Results are given in Table 9.

Average time for speed detection is 22.3 ms.

D. EXPERIMENTAL ANALYSIS OF APACHE KAFKA

In this section, it has been observed how the number of partitions, replications, and brokers has an effect on producer-send packet throughput (mb/sec). Graphs are given in figure 8 shows that producer throughput varies depending on the number of brokers, partitions of topics, and replications. It is observed that throughput results are better when Partition is 2, Broker is 2 and Replication is 1.

Time calculations based on the production and consumption of the messages are given in table 10 and figure 9. In measuring these values, the tests were repeated ten times, and the average results were obtained.

TABLE 9. Speed detection.

Time Step	Real Speed	Calculated Speed
0	56.65	59.2
2	53.74	55
4	52.13	56.23
6	49.06	51.47
8	50.51	53.6
10	48.89	50.03
12	59.57	62.37
14	48.83	50.81
16	56.01	59.65
18	51.12	55.2
20	41.97	42.67

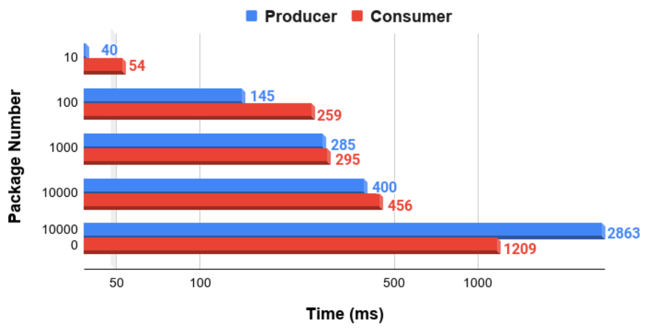


FIGURE 9. Apache kafka performance test.

VI. CONCLUSION

There are various studies about streaming data processing using deep learning methods in the literature. However, most of these studies focused on a single computer. We propose a framework that tracking a specific vehicle over the video

TABLE 10. Apahce kafka performance test.

Message/ Package Number	Produced (Data Transmission) / Consumed (Data Acquisition)	Time (ms)
10	produced	40
10	consumed	54
100	produced	145
100	consumed	259
1000	produced	285
1000	consumed	295
10000	produced	400
10000	consumed	456
100000	produced	2863
100000	consumed	1209

streams published by the collaborating traffic surveillance cameras in real-time. We transmitted data between microservices among the network. Also, we enable the user to use text queries among the video streams.

We discussed potential use cases for video retrieval in mission-critical real-time applications leveraging Apache Kafka with this framework. Kafka is a great complementary tool for machine learning infrastructure and may benefit areas including inference of the analytic model in real-time, monitoring, and alerting.

In future studies, it is planned to add license plate recognition among the attributes. Thus, developing more comprehensive smart traffic systems can contribute to smart and safe city projects.

REFERENCES

[1] S.-I. Yu *et al.*, "InformediaTRECVID 2014 MED and MER," in *Proc. TRECVID Video Retr. Eval. Workshop (NIST)*, 2014, pp. 1–14. [Online]. Available: https://www-nlpir.nist.gov/projects/tvpubs/tv.pubs.14.org.html

[2] S. Tang, L. Feng, Z. Kuang, Y. Chen, and W. Zhang, "Fast video shot transition localization with deep structured models," in *Computer Vision—ACCV* (Lecture Notes in Computer Science Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 11361. Cham, Switzerland: Springer, 2019, pp. 577–592.

[3] R. Gaikwad and J. R. Neve, "A comprehensive study in novel content based video retrieval using vector quantization over a diversity of color spaces," in *Proc. Int. Conf. Global Trends Signal Process., Inf. Comput. Commun. (ICGTSPICC)*, 2017, pp. 38–42.

[4] G. S. N. Kumar, V. S. K. Reddy, and S. S. Kumar, "High-performance video retrieval based on spatio-temporal features," in *Microelectronics, Electromagnetics and Telecommunications*. Singapore: Springer, 2018, pp. 433–441.

[5] Z. Z. Lan, Y. Yang, N. Ballas, S. I. Yu, and A. Haputmann, "Resource constrained multimedia event detection," in *Multimedia Modeling* (Lecture Notes in Computer Science Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 8325. Cham, Switzerland: Springer, 2014, pp. 388–399.

[6] K. M. Shah and R. M. Makwana, "Shot boundary detection using logarithmic intensity histogram: An application for video retrieval," *Int. J. Appl. Eng. Res.*, vol. 12, no. 22, pp. 12616–12624, 2017.

[7] T. Kar and P. Kanungo, "A motion and illumination resilient framework for automatic shot boundary detection," *Signal, Image Video Process.*, vol. 11, no. 7, pp. 1237–1244, Oct. 2017.

[8] P. Haloi, A. K. Bordoloi, and M. K. Bhuyan, "Unsupervised broadcast news video shot segmentation and classification," in *Proc. 2nd Int. Conf. Innov. Electron., Signal Process. Commun. (IESC)*, Mar. 2019, pp. 243–251.

[9] S. S. Gornale, A. K. Babaleshwar, and P. L. Yannawar, "Analysis and detection of content based video retrieval," *Int. J. Image, Graph. Signal Process.*, vol. 11, no. 3, p. 43, 2019.

[10] M. A. Abdelwahab, "Accurate vehicle counting approach based on deep neural networks," in *Proc. Int. Conf. Innov. Trends Comput. Eng. (ITCE)*, Aswan, Egypt, Feb. 2019, pp. 1–5.

[11] E. P. Wigner, "Theory of traveling-wave optical laser," *Phys. Rev.*, vol. 134, pp. A635–A646, Dec. 1965.

[12] X. Cao, C. Wu, P. Yan, and X. Li, "Linear SVM classification using boosting HOG features for vehicle detection in low-altitude airborne videos," in *Proc. 18th IEEE Int. Conf. Image Process.*, Brussels, Belgium, Sep. 2011, pp. 2421–2424.

[13] P. Spagnolo, M. Leo, and A. Distante, "Moving object segmentation by background subtraction and temporal analysis," *Image Vis. Comput.*, vol. 24, vol. 5, pp. 411–423, 2006.

[14] R. A. Hadi, G. Sulong, and L. E. George, "Vehicle detection and tracking techniques: A concise review," *Signal Image Process.*, vol. 5, no. 1, p. 1, 2014.

[15] X. Wang, W. Zhang, X. Wu, L. Xiao, Y. Qian, and Z. Fang, "Real-time vehicle type classification with deep convolutional neural networks," *J. Real-Time Image Process.*, vol. 16, no. 1, pp. 5–14, 2019, doi: 10.1007/s11554-017-0712-5.

[16] P. Chakraborty, Y. O. Adu-Gyamfi, S. Poddar, V. Ahsani, A. Sharma, and S. Sarkar, "Traffic congestion detection from camera images using deep convolution neural networks," *Transp. Res. Rec.*, vol. 2672, no. 45, pp. 222–231, 2018, doi: 10.1177/0361198118777631.

[17] S. Kul, S. Eken, and A. Sayar, "Measuring the efficiencies of vehicle classification algorithms on traffic surveillance video," in *Proc. Int. Conf. Artif. Intell. Data Process.*, Malatya, Turkey, Sep. 2016, pp. 1–6.

[18] Z. Chen, T. Ellis, and S. A. Velastin, "Vehicle type categorization: A comparison of classification schemes," in *Proc. 14th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Washington, DC, USA, Oct. 2011, pp. 74–79.

[19] X. Wu, S. Sun, N. Chen, M. Fu, and X. Hou, "Real-time vehicle color recognition based on YOLO9000," in *Communications, Signal Processing, and Systems* (Lecture Notes in Electrical Engineering), vol. 516, Q. Liang, X. Liu, Z. Na, W. Wang, J. Mu, and B. Zhang, Eds. Singapore: Springer, 2020.

[20] S. Vitabile, G. Pollaccia, C. Pilato, and E. Sorbello, "Road sign recognition using a dynamic pixel aggregation technique in the HSV color space," presented at the 11th Int. Conf. Image Anal. Process., Palermo, Italy, 2001.

[21] D. Seng, B. Lin, and J. Chen, "Convolutional neural network and the recognition of vehicle types," *NeuroQuantology*, vol. 16, no. 6, pp. 1–5, Jun. 2018, doi: 10.14704/nq.2018.16.6.1641.

[22] R. F. Rachmadi and I. K. E. Purnama, "Vehicle color recognition using convolutional neural network," 2015, *arXiv:1510.07391*. [Online]. Available: http://arxiv.org/abs/1510.07391

[23] P. Chen, X. Bai, and W. Liu, "Vehicle color recognition on an UrbanRoad by feature context," *IEEE Trans. Intell. Transp. Syst.*, vol. 99, pp. 1–7, 2014.

[24] A. Şentaş, İ. Tashiev, F. Küçükayvaz, S. Kul, S. Eken, A. Sayar, and Y. Becerikli, "Performance evaluation of support vector machine and convolutional neural network algorithms in real-time vehicle type and color classification," *Evol. Intell.*, vol. 13, no. 1, pp. 83–91, 2018, doi: 10.1007/s12065-018-0167-z.

[25] A. Şentaş, İ. Tashiev, F. Küçükayvaz, S. Kul, S. Eken, A. Sayar, and Y. Becerikli, "Performance evaluation of support vector machine and convolutional neural network algorithms in real-time vehicle type classification," in *Advances in Internet, Data & Web Technologies* (Lecture Notes on Data Engineering and Communications Technologies), vol. 17, L. Barolli, F. Xhafa, N. Javaid, E. Spaho, V. Kolici, Eds. Cham, Switzerland: Springer, 2018.

[26] C. Pornpanomchai and K. Kongkittisan, "Vehicle speed detection system," in *Proc. IEEE Int. Conf. Signal Image Process. Appl.*, Kuala Lumpur, Malaysia, Nov. 2009, pp. 18–19.

[27] J. L. Wilder, A. Milenkovic, and E. Jovanov, "Smart wireless vehicle detection system," in *Proc. 40th Southeastern Symp. Syst. Theory (SSST)*, New Orleans, LA, USA, Mar. 2008, pp. 16–18.

[28] J. Pelegri, J. Alberola, and V. Llario, "Vehicle detection and car speed monitoring system using GMR magnetic sensors," in *Proc. IEEE 28th Annu. Conf. Ind. Electron. Soc. (IECON)*, vol. 2, Nov. 2002, pp. 1693–1695.

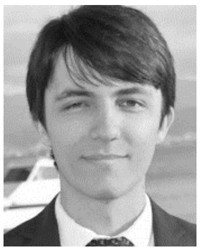
[29] R. Koide, S. Kitamura, T. Nagase, T. Araki, M. Araki, and H. Ono, "A punctilious detection method for measuring vehicles' speeds," in *Proc. Int. Symp. Intell. Signal Process. Commun.*, Tottori, Japan, Dec. 2006, pp. 12–15.

[30] H. H. Cheng, B. D. Shaw, J. Palen, B. Lin, B. Chen, and Z. Wang, "Development and field test of a laser-based nonintrusive detection system for identification of vehicles on the highway," *IEEE Trans. Intell. Transp. Syst.*, vol. 6, no. 2, pp. 147–155, Jun. 2005.

[31] Z. Osman and S. A. Chahine, "Novel speed detection scheme," in *Proc. 14th Int. Conf. Microelectron.*, Beirut, Lebanon, Dec. 2002, pp. 165–168.
[32] T. Celik and H. Kusetogullari, "Solar-powered automated road surveillance system for speed violation detection," *IEEE Trans. Ind. Electron.*, vol. 57, no. 9, pp. 3216–3227, Sep. 2010.
[33] N. Aliane, J. Fernandez, S. Bemposta, and M. Mata, "Traffic violation alert and management," in *Proc. 14th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2011, pp. 1716–1720.
[34] N. Kumar and N. Sukavanam, "Keyframes and shot boundaries: The attributes of scene segmentation and classification," in *Advances in Intelligent Systems and Computing*, vol. 741. Singapore: Springer, 2019, pp. 771–782.
[35] B. V. Patel, "Content based video retrieval systems," *Int. J. UbiComp*, vol. 3, no. 2, pp. 13–30, Apr. 2012.
[36] Y. Yang, Z. Ma, Z. Xu, S. Yan, and A. G. Hauptmann, "How related exemplars help complex event detection in Web videos?" in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 2104–2111.
[37] V. Kantorov and I. Laptev, "Efficient feature extraction, encoding, and classification for action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 2593–2600.
[38] A. A. Sewisy, K. F. Hussain, and A. D. Suleiman, "Speedup video segmentation via dual shot boundary detection (S-DSBD)," *Int. Res. J. Eng. Technol.*, vol. 3, no. 12, pp. 11–14, 2016.
[39] C. Liu, G. Chen, Y. Ma, X. Chen, and J. Wu, "A system for indexing and retrieving vehicle surveillance videos," in *Proc. 4th Int. Congr. Image Signal Process.*, vol. 1, Oct. 2011, pp. 451–455, doi: 10.1109/CISP.2011.6099951.
[40] R. Baran, T. Rusc, and P. Fornalski, "A smart camera for the surveillance of vehicles in intelligent transportation systems," *Multimedia Tools Appl.*, vol. 75, no. 17, pp. 10471–10493, Sep. 2016, doi: 10.1007/s11042-015-3151-y.
[41] *Apache Kafka*. Accessed: Jun. 6, 2019. [Online]. Available: https://kafka.apache.org/
[42] S. Kul, S. Eken, and A. Sayar, "Distributed and collaborative real-time vehicle detection and classification over the video streams," *Int. J. Adv. Robot. Syst.*, vol. 14, no. 4, 2017, Art. no. 1729881417720782.
[43] R. Banno, S. Takeuchi, M. Takemoto, T. Kawano, T. Kambayashi, and M. Matsuo, "A distributed topic-based Pub/Sub method for exhaust data streams towards scalable event-driven systems," in *Proc. IEEE 38th Annu. Comput. Softw. Appl. Conf.*, Vasteras, Sweden, Jul. 2014, pp. 21–25.
[44] J. Gascon-Samson, F.-P. Garcia, B. Kemme, and J. Kienzle, "Dynamoth: A scalable pub/sub middleware for latency-constrained applications in the cloud," in *Proc. IEEE 35th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Columbus, OH, USA, Jun. 2015, pp. 486–496.
[45] A. J. G. Gray and W. Nutt, "A data stream publish/subscribe architecture with self-adapting queries," in *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, Berlin, Germany: Springer, pp. 420–438, 2005.
[46] Y. Zou, J. Cao, and H. Wu, "TraffiCast: Real-time pub/sub based video surveillance system over interconnected WMNs and WSNs," in *Proc. 6th IEEE Int. Conf. Distrib. Comput. Sensor Syst. Workshops (DCOSSW)*, Santa Barbara, CA, USA, Jun. 2010, pp. 21–23.
[47] Z. Dong, Y. Wu, M. Pei, and Y. Jia, "Vehicle type classification using a semisupervised convolutional neural network," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 4, pp. 2247–2256, Aug. 2015.
[48] P. Chen, X. Bai, and W. Liu, "Vehicle color recognition on urban road by feature context," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 5, pp. 2340–2346, Oct. 2014.
[49] G. Garibotto, P. Castello, E. D. Ninno, P. Pedrazzi, and G. Zan, "Speed-vision: Speed measurement by license plate reading and tracking," in *Proc. IEEE Intell. Transp. Syst.*, Aug. 2002, pp. 585–590.
[50] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*. [Online]. Available: http://arxiv.org/abs/2004.10934
[51] F. Chollet. (2015). Building Powerful Image Classification Models Using Very Little Data. Keras Blog. Accessed: Jun. 5, 2021. [Online]. Available: https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html
[52] *Monitoring Kafka Performance Metrics*. Accessed: Feb. 20. 2018. [Online]. Available: https://www.datadoghq.com/blog/monitoring-kafka performance-metrics/
[53] *Open Source Computer Vision Library*. Accessed: Sep. 15, 2017. [Online]. Available: https://github.com/itseez/opencv
[54] H. Grabner and H. Bischof, "On-line boosting and vision," in *Proc. CVPR*, vol. 1, 2006, pp. 260–267.
[55] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 FPS with deep regression networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 749–765.
[56] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 702–715.
[57] Z. Kalal, K. Mikolajczyk, and J. Matas, "Forward-backward error: Automatic detection of tracking failures," in *Proc. 20th Int. Conf. Pattern Recognit.*, Aug. 2010, pp. 23–26.
[58] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1409–1422, Dec. 2012.
[59] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surveys*, vol. 35, no. 2, pp. 114–131, Jun. 2003.
[60] J. Barbier. (Jun. 9, 2014). *It's Here: Docker 1.0*. Docker, Inc. Accessed: Sep. 30, 2019. [Online]. Available: https://web.archive.org/web/20190121004233/https://blog.docker.com/2014/06/its-here-docker-1-0/
[61] J. Sang, Z. Wu, P. Guo, H. Hu, H. Xiang, Q. Zhang, and B. Cai, "An improved YOLOv2 for vehicle detection," *Sensors*, vol. 18, no. 12, p. 4272, Dec. 2018, doi: 10.3390/s18124272.
[62] J. Sang, P. Guo, Z. Xiang, H. Luo, and X. Chen, "Vehicle detection based on faster-RCNN," *J. Chongqing Univ.*, vol. 40, no. 7, pp. 32–36, 2017.

SEDA KUL received the B.S. and M.S. degrees in computer engineering from Kocaeli University, Kocaeli, Turkey, in 2015 and 2018, respectively, where she is currently pursuing the Ph.D. degree in computer engineering. She is also working as a Researcher with the Marmara Research Center (MRC), TUBITAK, Kocaeli, Turkey. Her research interests include distributed systems, image processing, and machine learning algorithms.

ISABEK TASHIEV received the B.S. degree in computer engineering from Kyrgyzstan Turkey Manas University, Bishkek, Kyrgyzstan, and the master's degree in computer engineering from Kocaeli University, Kocaeli, Turkey. He is currently working as a Software Engineer with betPawa, Tallinn, Estonia.



ALI ŞENTAŞ received the B.S. degree in computer engineering from Kocaeli University, Turkey. He is currently pursuing the M.S. degree with Istanbul Technical University, Turkey. He is also working as a Senior Software Engineer with AVL Turkey. His research interests include natural language processing and linguistics.

AHMET SAYAR received the M.Sc. degree in computer science from Syracuse University, in 2001, and the Ph.D. degree in computer science from Indiana University, USA, in 2009. He has worked with the Los Alamos National Laboratory, New Mexico, USA, and the Community Grids Laboratory, Indiana, USA, as a Researcher. Since 2010, he has been a Professor with the Department of Computer Engineering, Kocaeli University, Turkey. He has (co) authored three books and around 70 articles. His current research interests include distributed systems, big data, data-intensive computing, remote sensing, GIS, and spatial data-databases.