Computing
https://doi.org/10.1007/s00607-020-00869-8

SPECIAL ISSUE ARTICLE

An improved YOLO-based road traffic monitoring system

Mohammed A. A. Al-qaness[1] · **Aaqif Afzaal Abbasi**[2] · **Hong Fan**[1] · **Rehab Ali Ibrahim**[3] · **Saeed H. Alsamhi**[4] · **Ammar Hawbani**[5]

Received: 24 June 2020 / Accepted: 12 November 2020

Abstract
The growing population in large cities is creating traffic management issues. The metropolis road network management also requires constant monitoring, timely expansion, and modernization. In order to handle road traffic issues, an intelligent traffic management solution is required. Intelligent monitoring of traffic involves the detection and tracking of vehicles on roads and highways. There are various sensors for collecting motion information, such as transport video detectors, microwave radars, infrared sensors, ultrasonic sensors, passive acoustic sensors, and others. In this paper, we present an intelligent video surveillance-based vehicle tracking system. The proposed system uses a combination of the neural network, image-based tracking, and You Only Look Once (YOLOv3) to track vehicles. We train the proposed system with different datasets. Moreover, we use real video sequences of road traffic to test the performance of the proposed system. The evaluation outcomes showed that the proposed system can detect, track, and count the vehicles with acceptable results in changing scenarios.

Keywords Intelligent traffic · Neural network · Computer vision · YOLOv3 · Traffic analysis

1 Introduction

Today, mankind has stepped far forward—there are more and more areas where human work is replaced by artificial intelligence (AI). There are many reasons for this: repetitive work, the human factor, the inability to compete in reaction speed, in some cases, the inability of our body and our senses to work in unusual situations. At present, traffic is regulated by signs, traffic lights, and so on. However, with the increasing development of artificial intelligence, a completely different environment will be required to control traffic. Statistics on car sales indicate a steady increase in the number of vehicles worldwide (2016 is considered a record year, as sales increased by almost 50%), which creates some serious problems, such as traffic congestions, safety, pol-

Extended author information available on the last page of the article

Published online: 06 January 2021

lution, and increased need for mobility. Most of these problems are being solved by constructing new, wider, and safer highways and additional lanes on existing ones, but it is expensive and often not feasible. Cities are limited with space, and at some point, growing demand may exceed constructional abilities.

The primary purpose of the Traffic Analysis System (TAS) is to collect data about the current traffic situation on the roads. The basis of TAS is the processing of the video stream and its subsequent analysis. The objective is to develop a system that will make it possible not only to estimate the number of passing vehicles in the lanes but also to take into account the direction of their movements in the lanes. Also, such a system should be able to classify the traffic flow into the main categories of vehicles for a more accurate prediction of traffic while optimizing coordination plans. Based on the density of vehicles on a particular route, it becomes possible to develop a mechanism that effectively reduces and predicts traffic congestion [1], taking some actions in advance. For example, the precise adjustment of traffic lights and signs. The detection algorithm requires high accuracy and image processing speed in classifying an object and determining its location in the image. The range of approaches to solving this problem is large enough, which allows people to choose the most suitable methods for specific needs.

A review of recent work has shown that the problem of video detection of vehicles is being studied by many researchers. In most of the studies, vehicle search is performed by detecting registration plates [2–4]. This method is one of the simplest in terms of recognition algorithms, because of the contrast of the background and the characters, and the limited number of characters. However, that approach does not allow detecting vehicles in situations where there are no license plates (bicycles) or when they are located in non-standard parts (like motorcycles or cars with temporary numbers). Moreover, the classification of vehicles requires access to the database of traffic police, which is not always necessary and possible. Along with this, there are studies in which a search is made for simple geometric primitives: straight lines in the area of the bumper or suspension, and circles enclosing the wheels. The restoration of vehicle parameters is carried out based on the relative position of the set of primitives and the history of their displacement in the coordinate system associated with the image. The main limitations of both approaches are instability to changing the angle and the presence of blurry frames (for example, due to short-term wind exposure), as well as the lack of the possibility of large-block classification for one frame. In this regard, existing methods do not allow solving the problem of assessing the intensity of traffic flows, taking into account their qualitative composition in conditions when vehicles change their angle. Thus, the problem of developing methods and algorithms for solving the video detection problem of vehicles of different classes, providing search and tracking regardless of camera angle and location, is relevant.

There are a lot of different techniques for object tracking. Vehicle tracking is often implemented by subtracting the dynamic component (moving objects) from the static background of the image. Background subtraction is often combined with blob analysis [5–7], Kalman filter [8, 9], Gaussian Mixture Model (GMM) [10, 11]. Particle filter dependent trackers [12, 13] are also used often to track moving vehicles. Nonetheless, the background subtraction method has drawbacks when processing data in dense traffic conditions. This leads to vehicle fusion due to partial occlusion in the processed image data and the prediction of an incorrect bounding box occurs. Shadows from vehicles also cause the wrong detection results. To solve this problem, in [14], a GMM is combined with the maximization of expectation for constructing a background model. Only then, vehicles are detected by background subtraction. Occlusions are processed using color histograms and morphological features. In [15], it was proposed to count vehicles in nighttime videos by extracting bright regions (headlights), followed by tracking of paired headlights. In [16], an off-line training method for vehicle detection has achieved good results. It is using the support vector machine to calculate the Haar wavelet function. In [17], an approach based on multilayer neural networks is used. Here, the network performs both feature extraction and classification and that simplicity enables real-time applications. The network is trained by a new algorithm that creates a feature space where the patterns have a desirable statistical distribution. In [18], gradient analysis and Adaboost classification were used to detect the vehicle from the back. In [19], an algorithm for detecting a vehicle from the side using a cascade of boosted classifiers presented. In [20], an algorithm for teaching real-time tracking of vehicles in a video sequence using an advanced feature selection method is presented. In [21], a framework for spatiotemporal vehicle tracking using unsupervised learning-based segmentation and object tracking is presented. In [22], an intelligent traffic system is able to automatically detect and apprehend traffic violators. As for China, a few years ago, Alibaba e-commerce company launched its traffic management service, City Brain [23]. Using cloud computing and machine learning, the service can reduce traffic congestion on the city's roads. City Brain analyzes huge amounts of data coming from hundreds of traffic lights, video cameras, and public transport systems to optimize traffic flow. It will use big data analysis, image classification, and other technology to make predictions about current traffic situations, and give recommendations, for example, determining the fast and free route for an ambulance through the city using the real-time analysis of road network density.

A similar system called Vivacity was established in Milton Keynes, United Kingdom [24]. It consists of more than 400 smart traffic cameras at the major junctions of the city. Vivacity is capable of counting and classifying road users. It has sensors that can measure how long it takes for vehicles to travel between certain intersections. In perspective, sharing such information with neighboring intersections to help them understand what is coming their way could help in autonomous decision making. It stores live photos and video footages to help with the development of future planning. Vivacity has a machine learning sub-module that, by analyzing the data, learns typical daily trends, and combines this with how the traffic reacts to some changes in the road network. It constantly evolves and adapts, increasing its predictive ability and reducing the amount of human intervention needed. It forecasts traffic flows for the day, providing historical and live data.

Effective traffic management is not only important to urban planning but also just about everyone. The Texas Transportation Institute conducted a research which revealed that because of traffic congestion the average urban motorist waste about 42 h a year stuck in traffic jams, resulting in damage of 160 billion dollars a year, including from lost productivity, gas burned while stuck in traffic and additional tear on vehicles [25]. This number can go even higher in other countries like London, where traffic congestion is more serious than in other cities around the globe.

Besides the negative impact on the economy, traffic jams cause enormous environmental damage. A pulmonary function test conducted in [26] showed that 1 out of 6 traffic policemen have a lung problem and suffer from breathing problems due to burgeoning air pollution. Applying AI to the traffic management system could minimize human intervention, which can protect people from further health problems.

In this paper, we present to an intelligent vehicle tracking system using an improved You Only Look Once (YOLOv3) algorithm. The proposed system is applied for detecting and classification of vehicles on video from surveillance cameras, that will carry out a full cycle of solving the traffic analysis problem, including the preparation of test data, marking the visibility percentage of detected vehicles and areas of interest, collection, and analysis of the results of the search, evaluation of the traffic flows intensity of different directions, visualization of search and maintenance results, ensure resistance to changing angles and camera position, an experimental study of the effectiveness of the system.

The paper is organized as follows. Section 3 explains the methods employed in the study. Section 3 describes the experimentation. Finally, Sect. 4 concludes the paper.

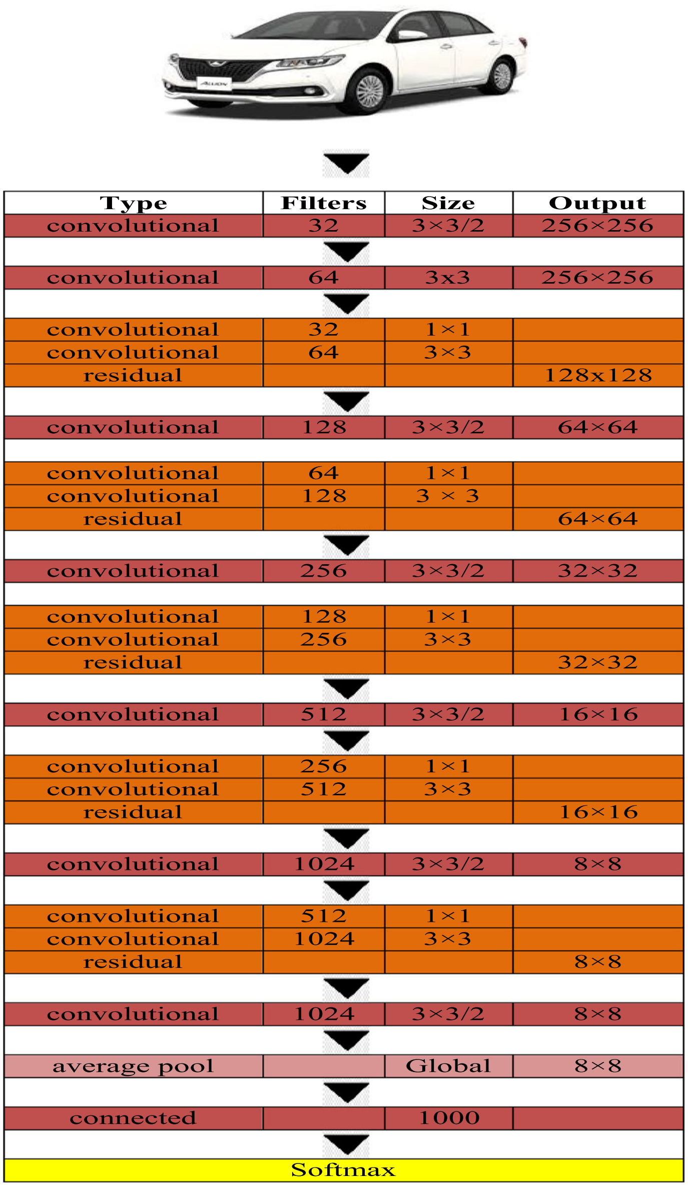
2 Methodology

In the proposed design we used Python programming language, OpenCV library for image processing, Google Colab cloud service, and Anaconda development environment. Python was chosen as the programming language for development. We used NumPy, Pandas, and OpenCV libraries. The system under development consists of two main subsystems—internal and external. The internal subsystem is a video stream processing algorithm for detecting objects and a tracking algorithm. Processing is performed using the You Only Look Once (YOLO) neural network model. The external system is a desktop application for the user to work with the tracking system.

At the initial stage of the system's operation, the video stream is processed frame by frame with a convolutional neural network in order to detect all objects on the frame. As a model of a convolutional neural network, the YOLOv3 model [27] was used. This model was chosen because it has the ability to recognize many objects on the frame with an indication of their approximate location, size, and class. Also, this model has an open-source code and license, which allow for free use and modification. YOLOv3 has a framework for conducting complete retraining of the neural network using its own data, as well as for configuring a ready-made network (fine-tuning). The general layered structure of the Darknet-53 network is shown in Fig. 1.

The general structure of YOLOv3 consists of another 53 layers, also based on Darknet-53, they are responsible for predicting the coordinates of objects and their sizes. Thus, the number of layers reaches 106. The network structure is shown in Fig. 2.

Prediction of the coordinates and sizes of objects is carried out at three different image scales. Thus, the network is equally capable of detecting both large and small objects in the frame. The main advantage of the YOLOv3 architecture is that objects are localized and classified in a single pass through the network. This allows for



Type	Filters	Size	Output
convolutional	32	3×3/2	256×256
convolutional	64	3x3	256×256
convolutional	32	1×1	
convolutional	64	3×3	
residual			128x128
convolutional	128	3×3/2	64×64
convolutional	64	1×1	
convolutional	128	3 × 3	
residual			64×64
convolutional	256	3×3/2	32×32
convolutional	128	1×1	
convolutional	256	3×3	
residual			32×32
convolutional	512	3×3/2	16×16
convolutional	256	1×1	
convolutional	512	3×3	
residual			16×16
convolutional	1024	3×3/2	8×8
convolutional	512	1×1	
convolutional	1024	3×3	
residual			8×8
convolutional	1024	3×3/2	8×8
average pool		Global	8×8
connected		1000	
Softmax			

Fig. 1 The general network structure of Darknet-53

very quick frame-by-frame processing, which makes it possible to process video in real-time.

For the task of detecting objects, a measure such as Average Precision (AP) is usually used. AP is obtained from two metrics: precision and recall. There are two possible events when an object of one particular class is detected. Precision denotes

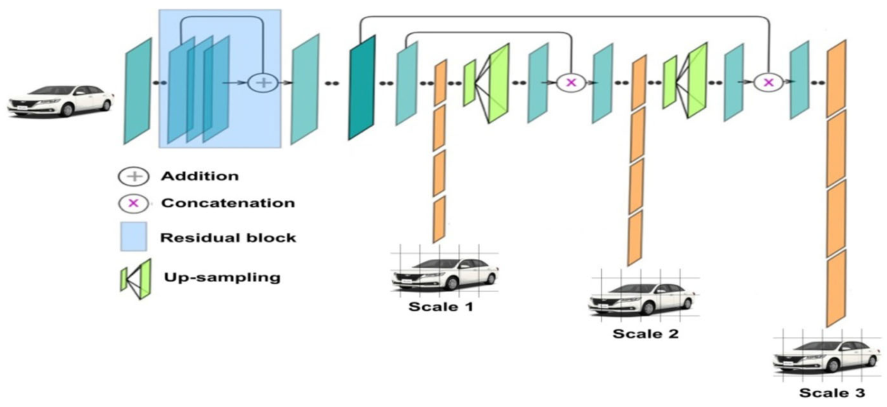


Fig. 2 The general network structure of YOLOv3

the percentage of true predictions for the framing window (true positives) among all the results for this class. Precision coefficient of a class is given by

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1}$$

where TP and FP represent true positive and false positive, respectively.

Recall indicates the percentage of framing windows found for the class, among all those presented in the ground truth of this image. The recall coefficient of a class is given by the following equation.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

where FN represents false negative.

In order to be able to assess the accuracy of the predicted bounding box of the object, the Intersections over Union (IoU) metric is used. A typical threshold value indicating correct detection is IoU > 50%. IoU is a metric used to evaluate the accuracy of algorithms that detect objects. As shown in Fig. 3, IoU value is equal to the partial intersection area of the predicted bounding box and the ground truth response box to the area of the union of these boxes.

Accuracy assessment for the detection and classification algorithm cannot be limited to AP, therefore, the concept of mean Average Precision (mAP) should be used in the future. The formula of mAP is given in Eq. 3:

$$\text{mAP} = \frac{\sum_{q=1}^{Q} \text{AP}(q)}{Q}, \tag{3}$$

where Q is the number of queries.

In this case, the number of queries corresponds to the number of classes for which the algorithm is trained. A search for objects based on a certain template assumes that there is an image of an object with highlighted features—a template and a test image

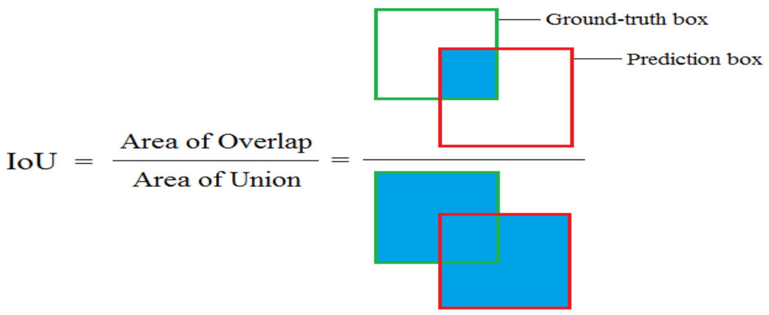


Fig. 3 The general formula of the Intersections over Union (IoU) metric

that matches this template. In the simplest case, the template may be a matrix of color intensities that are most characteristic of the object. More complex methods of the considered group use sets of feature vectors (descriptors), a geometric representation of an object [28] or probabilistic models of objects that contain information about the distribution of pixel intensities as a template. Matching with a template involves comparing the descriptions of the test and template images with some selected metrics. It should be noted that the search methods for a given pattern work effectively when searching for single objects because when overlapping occurs, some features in the description can disappear. Another common approach to solving the problem of detecting motion areas is to calculate the optical flow [29]. The optical flow allows for the determination of the offset of each pixel. The application of this approach requires the fulfilment of two basic conditions: the brightness of each point of the object does not change over time; the nearest points belonging to the same object move with similar speed in the image plane.

Probabilistic methods use an approach based on the concept of state space. It is believed that a moving object has a certain internal state, which is measured on each frame. In the simplest case, the state refers to the position of the object in the image. Typical examples of such methods are those based on the Kalman filter [30] and the particle filter [31]. When using the Kalman filter, it is assumed that the state is a random variable with a normal distribution, and in the case of a particle filter, the distribution is specified by a set of possible state values with an indication of the frequencies of their occurrence. In practice, it is implemented using the mean shift and its continuous modification (Continuous Adaptive Mean Shift, CAM Shift) [32] by using kernel-based tracking techniques.

2.1 Detection algorithm

At the detection stage, the neural network brings the original image to a given square size and divides it into square blocks based on hyper parameter H. Their number is calculated as HxH. Each block at the detection stage makes a prediction about objects whose centers are located inside the block. The prediction includes the coordinates of the center of the object inside the block and its dimensions. The number of predictions that each block makes is a hyper parameter P. Thus, during processing the image, at

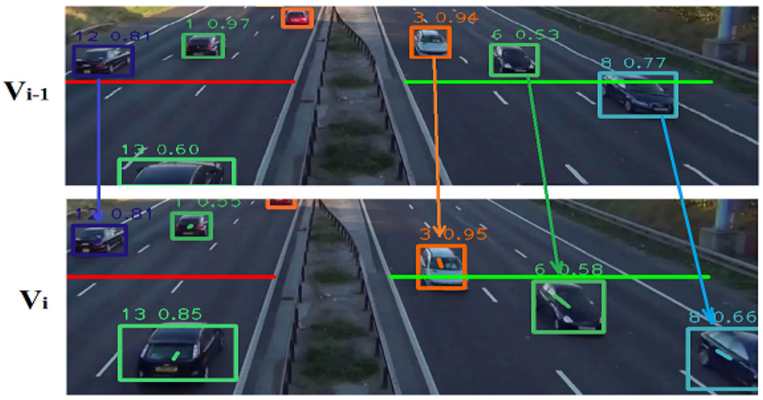


Fig. 4 The same objects on two consecutive frames V_i and V_{i-1}.

the output of each block, a matrix $M = (x_1\dots\ x_q,\ y_1\dots\ y_q,\ h_1\dots\ h_q,\ \dots w_1\dots\ w_q,\ c_1\dots\ c_q)$ is obtained, where x_i, y_i are the coordinates of the i-th object; h_i, w_i are the height and width of this object in pixels; c_i is the confidence coefficient.

Each block also makes a class prediction based on the characteristics identified in the convolution process. Depending on the number of classes that were initialized in the network during the learning process, the number of output indicators will also differ. Thus, at the output, for each block of the image, there is a vector: $\vec{p} = (p_1, p_2, \dots p_k)$, where p_i is the probability of the *i*th class in the block, and k is the total number of classes.

Next, a decision about the presence or absence of an object inside the block is made. The decision is made according to the rule: $(\exists\ i \in B)c_i > t$, where t is the specified confidence threshold. That is if there is at least one prediction, the confidence coefficient of which exceeds a specified threshold t, a decision about the presence of an object inside the block is made. The class of this object is defined as: $a_j = h;\ p_h = \max(\vec{p}_j) = \max\ (p_{j1}, p_{j2}, \dots p_{jk});\ h = \overline{1, k}$, where a_j is the class of jth object. Thus, each image block generates either one object or not a single one. The final set of objects forms an output matrix: $R = (x_1\ \dots\ x_m,\ y_1\ \dots\ y_m,\ h_1\ \dots\ h_m,\ w_1\ \dots\ w_m,\ a_1\ \dots\ a_m)$, where a_i is the class label of the ith object.

When processing the video frame-by-frame, a constantly updated sequence of detected objects R is obtained. Further processing of these objects to obtain the motion paths of each of the objects thus turns into a problem of association of objects of two consecutive frames V_i and V_{i-1}. An example is shown in Fig. 4.

The list of objects is inconstant. In the vehicle detection problem under consideration, new objects constantly appear on the frame, and old ones disappear at the same speed. Moreover, detection using a neural network still has a certain degree of error; the object may not be recognized on one of the frames but will appear again on the next. To ensure a constantly up-to-date list of objects, a comparison of V_i is conducted not from V_{i-1}, but with a constantly updated list of objects V'. Objects are removed from V' only after the number of consecutive frames on which the object is absent overcomes threshold. The general detection algorithm is presented below.

1. A list of object*Vi* has been received.
2. The association between the objects of *Vi* and *V′*.
3. Adding new objects from *Vi* to *V′*.
4. Removing old objects from *V′*.

The association between V_i and V' objects is made with the help of the Kalman filter. When associating detections with targets, the target state is updated using the detected bounding box, and the velocity components are solved using a Kalman filter framework [33]. In assigning detections to exist targets, the bounding box geometry of each target is determined by estimating its new position in the current frame. Next, the cost matrix of the assignment is calculated as the IOU distance between all obtained bounding boxes from the existing targets and detection. Also, to get rid of assignments where the detection to target overlap is lower than $IOU_{\min}$, a minimum IOU is applied. When vehicles appear and disappear in the image, the corresponding unique identifiers must be created or destroyed. To create trackers, any detection with an overlap lower than $IOU_{\min}$ is considered in order to show the presence of an untracked vehicle. The tracker is created using the bounding box geometry, with the velocity set to zero. The velocity is unknown at this moment. Thus, the velocity component's covariance is initialized with large values to express this uncertainty. Then, the new tracker goes through the trial period where the target must be associated with detections to collect (enough information in order to avoid tracking of false positives. Each object in V', to which no object from V_i was assigned, is marked as missing on the frame. If the object is absent for 3 frames in a row, it is declared to have left the frame and is removed from V'. All objects from V_i, for which no match was found, are added to V' as new objects not fixated previously.

2.2 Data processing

Websites such as Open Images [34], and COCO [35] provide large datasets of images that can be used to train CNN. The annotation file format of Open Images is just a text file. Microsoft's Common Objects in Context (COCO) dataset contain more than 200,000 images. It is relatively small, but all of those images have fairly accurate object segmentation. Also, COCO allows for five textual captions for each image, such as "a lot of cars are causing traffic jam" and "a highway of traffic with several transit buses riding down it." The annotation file of COCO is in JSON format. There are other datasets, such as Pascal VOC [36], Stanford Car Dataset [37], and others, which have thousands of vehicle images covering various lighting and weather conditions. Still, in most of them, the vehicles and their locations are not marked.

We use the OIDv4 toolkit to create a custom dataset. It downloads images from Open Images Dataset [34] and uses predefined commands to create a custom dataset.

2.3 Neural network training

The process of training the YOLOv3 network was carried out using the built-in functionality of the Darknet framework, which underlies the implementation of this network model. This framework allows users to set the network structure using configuration files and specify the hyper parameters for the network and its training. Also, with its help, users can launch a neural network to process images or videos, conduct training of a neural network, and check the quality of training on a test sample. Neural network training takes place over several epochs. During this process, at each stage, weights are optimized using gradient descent and back propagation methods. When training the network "from scratch," the weights are initialized by randomly nonzero values distributed according to the distribution law in a certain neighbourhood in order to avoid error propagation.

Also, in neural networks aimed at image recognition, the "fine-tuning" approach is quite common. With this approach, an already-trained network that performs a similar task is taken as the basis. The structure of the core network is adopted, and its weights are used to initialize the learning network. Next, the neural network is trained on the selected dataset. In this case, the training speed is set lower than during normal training to avoid over-fitting. Thus, this approach allows for performing the classification task using a network that solves a similar problem but is finely tuned to solve a specific problem. The advantage of this approach is that a relatively small data set is sufficient for training. For the YOLOv3 network, when learning from scratch, the recommended sample size is at least 2000 images for each class of the object. With fine-tuning, the amount of data can be reduced to 300 images per class [38].

The common name for this approach is transfer 1earning. The general principle of transfer 1earning is that the general representation of the external features of an object extracted by a convolutional neural network is usually similar. This is especially true for visually similar objects that can be grouped together—vehicles, feline animals, trees, etc. Moreover, a neural network trained to recognize an object from one group can be easily and efficiently retrained to recognize another object from the same group. In some cases, using this approach, it is possible to achieve a more accurate classification than when learning from scratch.

3 System design

In this section, the proposed system is tested and evaluated. We begin by data preparation followed by analysis.

3.1 Data preparation

We develop a neural network. About 10 thousand images from surveillance cameras are provided as an array of data. Most of them belong from Pascal VOC, Open Image, and COCO datasets and libraries. We converted these datasets for use in YOLO by using a special script. Some of the images had to be marked in order to indicate the location of all objects in the image and class labels ("car," "truck," "public transport", "motorbike") for each of these objects. This is necessary for training a neural network model.

Next, an analysis of all objects from the existing marked-up files was made. In the course of it, it was found that objects of the "car" class account for 80% of the total number of all available objects, which is a big imbalance that can interfere with the training of the neural network. To avoid this, all images that did not contain at least two classes were excluded from the sample. The resulting sample was still highly asymmetric, with a significant superiority in favor of the car class.

To ensure the quality of the final sample, we implemented a manual check of the entire sample to eliminate errors received in the primary processing. For this, the *LabelImg* software was used to process an array of images. This software allows users to view each image from the dataset. All objects whose data is contained in the corresponding text file are highlighted in a rectangle in the image. All rectangles are associated with some certain class: car, truck, public transport. Using this software, it is possible to remove an object's label from an image if this label was mistakenly set. It is also possible to change the classes of the object if an error occurred at the stage of classification and the borders of the object themselves are marked correctly.

The next step in preparing the dataset was separating from the total mass of images. Those images do not contain a single object (that is, the image has an empty road or does not have required objects). They were included separately in the sample to check their influence on the quality of neural network training. The resulting array of images was divided into two sets: training and test. The approximate distribution of images in the samples was 80% and 20% of the initial amount of images, respectively.

Further, the process of training an artificial neural network was performed. While working with the data, a modification of the neural network model YOLOv3 was carried out. This decision was made in order to facilitate the network structure and, accordingly, speed up the image processing. Based on the original model, by reducing the number of convolutional layers, and filters on them a modified model was obtained. Thus, the network structure was facilitated both in-depth and in-width.

The next step was training the resulting model and monitoring the quality of training. The training was conducted in Google Colab to reduce the development time. Colab provides the NVIDIA Tesla K80 graphics card with 12 GB of memory for free. Thus, only a stable internet connection is required for training. In case of a disconnect, downloaded files are deleted from the session, but the weights of the trained network can be saved on Google Drive every few epochs to resume learning if needed. Since the network structure was changed, the training was carried out from scratch with random initialization of weights in each network layer. The results of the modified network using the obtained weights can be seen in Fig. 5.

The YOLOv3 convolutional neural network model pre-trained on an open COCO sample containing 80 classes, including the classes cars, trucks, and public transport necessary for development was taken as the core network. Subsequently, an examination of the localization and classification quality of the resulting network objects on a test sample was made. An examination was also performed on the original network. The verification process is automated in the Darknet framework.

3.2 System development

System testing can be divided into three submodules. First, the system is initialized so that it is responsible for interaction with the user, and graphic interface of the

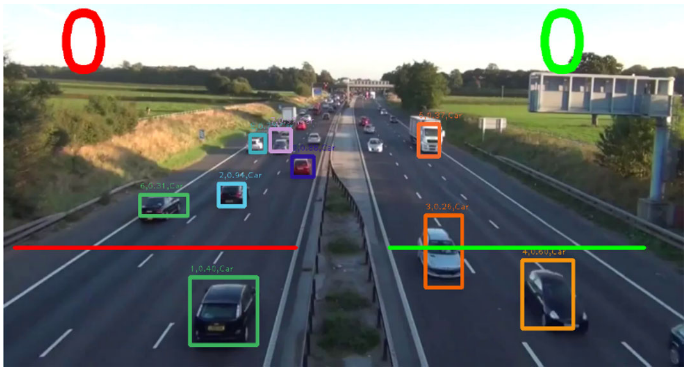


Fig. 5 The result of the modified neural network modified network using the obtained weights

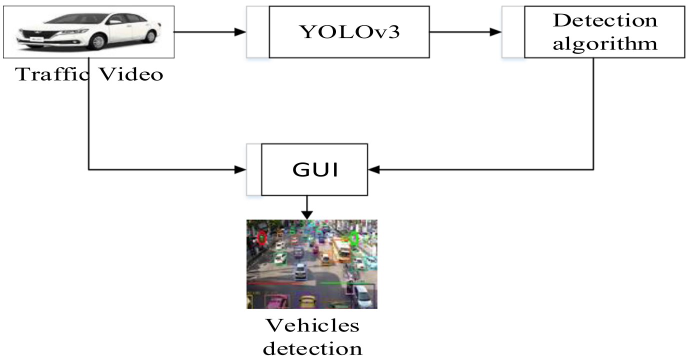


Fig. 6 The proposed traffic monitoring system workflow

system. The second submodule processes video with an artificial neural network. At the output, information about the list of objects on the frame is transmitted to the tracking submodule. The third submodule processes the obtained information that carries out the association of objects. A schematic illustration of the system is shown in Fig. 6.

The system is launched by the user using the Python command line. Using the OpenCV library, one frame is read and transmitted to the video processing submodule. Processing is performed every frame, or with some fixed regularity, for example, the processing is performed every 5 frames. The system's performance was tested at different frame processing frequencies, during which it was found that at a frequency of 1 frame per second, the tracking quality remains at the same level. With a further increase in frequency, tracking quality may be reduced.

In video processing, the video stream is processed by an artificial neural network. The neural network model is implemented using the built-in functionality of the Darknet. The network structure is attached to the network separately as a configuration file. This configuration file contains a layered network structure indicating the type of layer and its order. For convolutional layers, the filter sizes, their number, and their additional parameters are indicated. For fully connected layers, the number of output classes and the number of neurons in the layer are indicated. Network weights are loaded into the system as a binary file. The format is determined by the internal structure of the Darknet framework, with which the network is trained.

To process the image, it is converted into a blob with a depth of 3 by the number of input color layers (red, green, and blue). The width and height of the blob are determined by the given network parameters. If the input image does not match with these parameters, the image is scaled to these parameters. After conversion, it is processed by a neural network that was previously initialized. As a result of processing, an output array V is obtained. It contains all the objects that were recognized by the neural network. The output array V goes to the tracking submodule for processing.

At this stage, the processing of frame-by-frame lists of objects transferred from the video processing submodule takes place. Tracking is performed frame by frame when the list update method is called. During processing, the list V' is updated in accordance with the list Vi. Elements of Vi associated with elements from the main list transmit their position data on the frame to the corresponding objects of list V'. At each stage of processing, the coordinates of unsealed objects are compared with the coordinates of the detection zones. When the center of the object falls into one of the detection zones, the object is declared to be detected and added to the corresponding column counter of the zones.

To count the number of vehicles that have traveled a section of the road for a certain period of time, a special line is used. The coordinates of this line are set by the user depending on the position of the road in the frame. It should be perpendicular to the road for a more accurate calculation. The number of counting lines is unlimited, and thus, it is possible to count vehicles in several lanes simultaneously, regardless of the direction of movement. After all of the vehicles are detected and the boundary boxes around them are drawn, the intersection between the centroid of the box and the counting line is checked. In the case of an intersection, the counter of this particular line is incremented by one. All these counters are displayed on the top of the screen for the user.

The detection algorithm works on the whole image, but in practice, off-road areas are not necessary. To increase the accuracy and processing speed of the algorithm, it was decided to supply not the entire frame to the neural network, but only the road area. As shown in Fig. 7, before the detection and tracking stage, a mask is applied to each frame. After detection is done, the masked region is merged with the original frame, and the area of the mask is highlighted with a blue line, as shown in Fig. 8.

The graphical interface of the system displays the source video, on which labels from the main list of objects are stacked frame by frame. Figure 9 shows the number of cars passed on each lane and the period of time during which observation was carried out.

4 Results

The developed system was tested on three real traffic videos. The first video was shot in the afternoon, and the camera is located above the middle of the road. In the second video, the camera is located in the same way as in the first one, but that video was shot

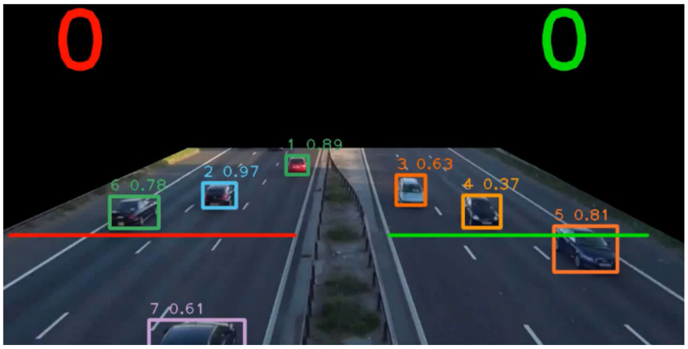


Fig. 7 Applying a mask to a frame before the detection and tracking stage

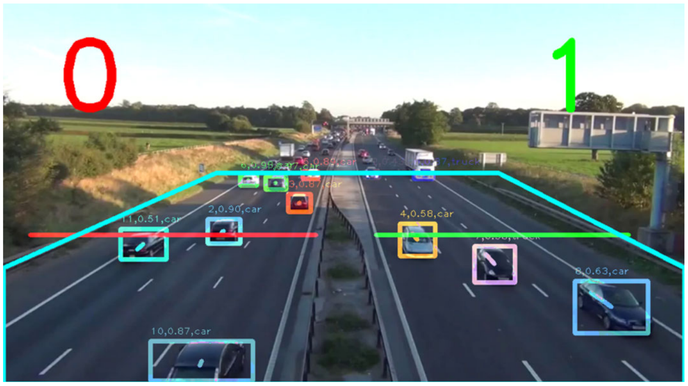


Fig. 8 Processed frame with a highlighted area of interest

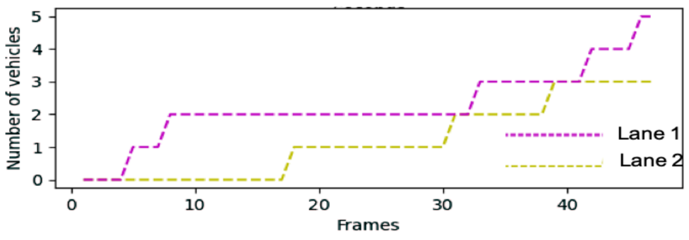


Fig. 9 The number of cars passed on each lane

in the winter with the presence of slight fog. The third video was shot in the night, and the camera is located on the side of the road. The test results are shown in Table 1.

Detection accuracy was calculated from each frame. That is, the detector may lose the vehicle for several frames, which will reduce the detection accuracy measurement, but if the detector finds the vehicle again before it crosses the counting line, this does not affect the accuracy of the calculation. YOLO determines the class of the vehicle according to visual features, and when the object is at a certain distance or a certain angle, objects of the class car, truck and bus may look quite similar.

Table 1 Experimental results of the proposed system

Estimations	Daytime video	Winter video	Night video
Detections in each frame	88%	82%	67%
Classification of vehicles	73%	65%	49%
Vehicles counted	98%	94%	85%
IoU	89%	86%	81%

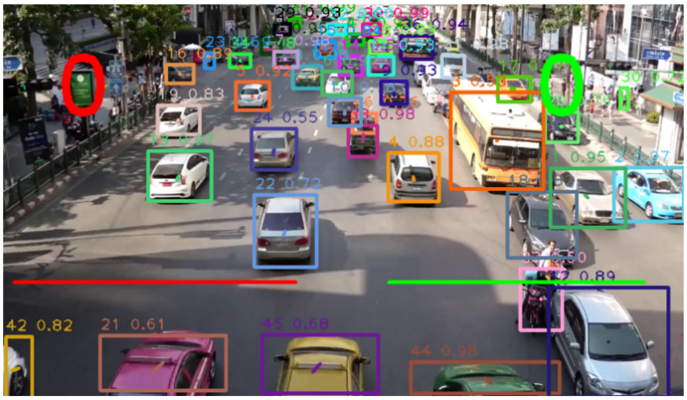


Fig. 10 Example of truncated car detection

Most vehicle tracking methods do not work well with very truncated vehicles. The reason for the problem is that to be able to detect the vehicle, they require the center of the bounding box of the vehicle to be inside the image. However, in practice, the detection of such vehicles from the image is not much necessary. In Fig. 10, there are several truncated cars at the bottom of the image. It is clearly seen that the designed method is very good at detecting even very truncated vehicles.

However, there are problems that appear during night video processing. In Fig. 11, several vehicles on the upper lane are not detected, but if it were not for the headlights, even the human eye would have difficulty recognizing those cars. The YOLO method needs more or less visible features of the objects to detect it. If good recognition at night is needed, it is better to apply the method described in [15]. In this case, it is not cars that are recognized, but pairs of headlights. It is important to mention that this method will not be functional if vehicle headlights are turned off.

The aforementioned methods can be combined using sensors such as radar or lidar for better accuracy. Finally, since the scene is changing and the vehicle is moving through light and dark sections of the road, it must have appeared at least a couple of times when driving under the lantern. Therefore, the system would have already had the information about the presence of those vehicles from the previous video frames.

The detection of objects with a big overlapping area is a problem in general, and this detector has this problem as well. In Fig. 12, there are three vehicles in the center: motorbike, mini-van, and passenger car, but only the mini-van in the center is detected. The proposed system predicts several bounding boxes on every object and

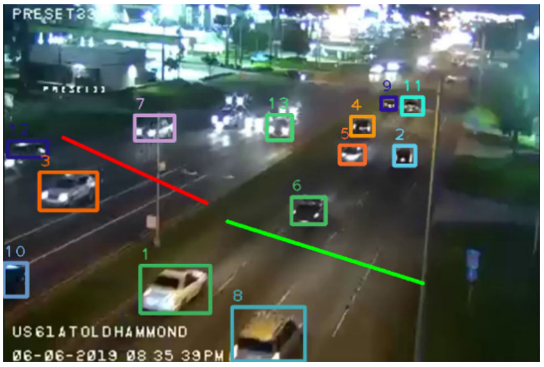


Fig. 11 Example of vehicle detection at night

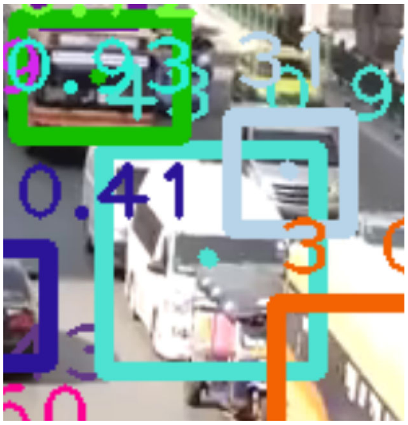


Fig. 12 Detection of overlapped vehicles

chooses the one with the highest confidence. The problem in our case is that a big area of overlapping and similarity of these vehicles makes it look like one big object. Thus, the proposed system mistakenly leaves only one bounding box in that area of the merged vehicles. Bounding box coordinate response maps create ambiguity in deciding regressive pixels. This problem can be solved if the TAS is equipped with at least two cameras at different angles, and thus these vehicles can be easily detected on different cameras. Associated with the previous problem is that the detector sometimes tends to predict multiple bounding boxes for the same vehicle. In Fig. 13, a detector predicted two bounding boxes on the same car and two bounding boxes on the same motorbike. To deal with that problem, the non-maxima suppression threshold can be decreased a little, but that can cause less accurate detection in situations when different objects are located close to each other. Thus it is important to find a golden mean for the NMS threshold.

Neural networks often produce detections with false rates due to incorrect input ranges (false positives) [39]. Examples of such detections using the developed model are in Fig. 14. It is seen that an object with an id of 234 is a billboard, not a vehicle. These detections may be caused, for example, by over-fitting to outliers in the training set, which can be reduced using dropout layers [40]. This technique was not used because

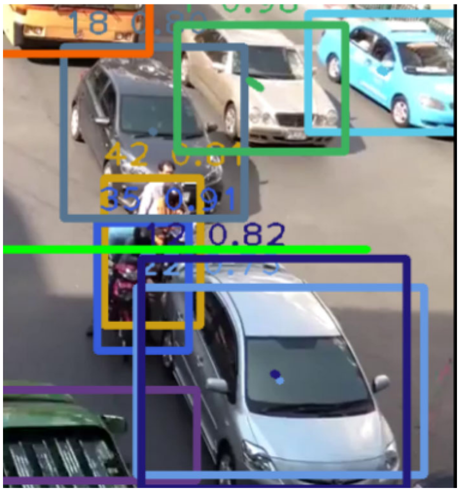


Fig. 13 A couple of bounding boxes predicted on the same vehicles

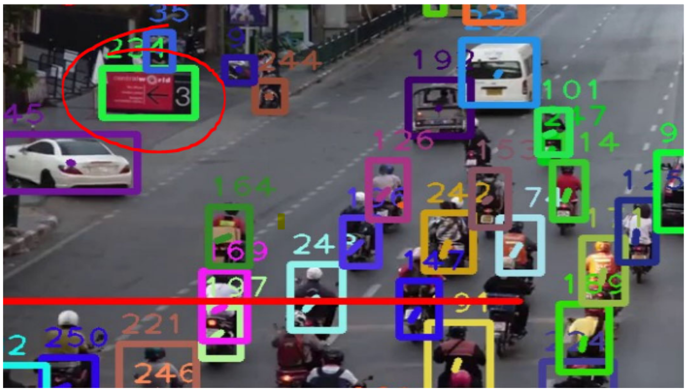


Fig. 14 Examples of false positives with high confidence

it prolongs training and training the network already took a significant amount of time (22 h), however, it might be a possible upgrade, which would reduce the false positive rate. Therefore, from previous results, we found that the improved YOLOv3 achieves significant performance due to its ability in detecting objects, as the original YOLOV3 showed in different applications, such as [41, 42].

5 Conclusions

The rapid growth witnessed in urban infrastructure gave tremendous rise to the need to improve road traffic management. Several techniques have been presented and discussed in the literature. In this paper, we present a real-time road traffic management approach using an improved YOLOv3. By using online available datasets, we trained our neural-network and applied the proposed solution to improve vehicle detection.

We used a convolution neural network for the traffic analysis system. The evaluation results showed that the proposed system achieved satisfactory performance. The developed system is low in cost and modest in hardware requirements compared with the traditional method of monitoring vehicle traffic. In addition, it does not need large-scale construction or installation work. In future work, we plan to test the system on cameras with wide and panoramic coverage and higher altitude to cover a significantly larger section of the road.

Acknowledgements This research was supported by the National Key Research and Development Program of China (Grant No. 2019YFB1405600).

References

1. Zhu Y, Wang J, Lu H (2008) A study on urban traffic congestion dynamic predict method based on advanced fuzzy clustering model. In: Proceedings of the 2008 international conference on computational intelligence and security, vol 2, pp 96–100. IEEE.
2. De Oliveira MB, Neto ADA (2013) Optimization of traffic lights timing based on multiple neural networks. In: Proceedings of the 2013 IEEE 25th international conference on tools with artificial intelligence, pp 825–832. IEEE.
3. Lee HJ, Chen SY, Wang SZ (2004) Extraction and recognition of license plates of motorcycles and vehicles on highways. In: Proceedings of the 17th international conference on pattern recognition, 2004. ICPR 2004, vol 4, pp 356–359. IEEE.
4. Comelli P, Ferragina P, Granieri MN, Stabile F (1995) Optical recognition of motor vehicle license plates. IEEE Trans Veh Technol 44(4):790–799
5. Dharamadhat T, Thanasoontornlerk K, Kanongchaiyos P (2009) Tracking object in video pictures based on background subtraction and image matching. In: Proceedings of the 2008 IEEE international conference on robotics and biomimetics, pp 1255–1260. IEEE.
6. Cancela B, Ortega M, Penedo MG, Fernández A (2011) Solving multiple-target tracking using adaptive filters. International conference image analysis and recognition. Springer, Berlin, Heidelberg, pp 416–425
7. McIvor A, Zang Q, Klette R (2001) The background subtraction problem for video surveillance systems. International workshop on robot vision. Springer, Berlin, Heidelberg, pp 176–183
8. Sekar G, Deepika M (2015) Complex background subtraction using kalman filter. Int J Eng Res Appl 5(3):15–20
9. Rabiu H (2013) Vehicle detection and classification for cluttered urban intersection. Int J Comput Sci Eng Appl 3(1):37
10. Wang K, Liang Y, Xing X, Zhang R (2015) Target detection algorithm based on gaussian mixture background subtraction model. In: Proceedings of the 2015 Chinese intelligent automation conference, pp 439–447. Springer, Berlin, Heidelberg.
11. Yazdi M, Bagherzadeh MA, Jokar M, Abasi MA (2014) Block-wise background subtraction based on gaussian mixture models. Applied mechanics and materials, vol 490. Trans Tech Publications Ltd, Switzerland, pp 1221–1227
12. Chan YM, Huang SS, Fu LC, Hsiao PY, Lo MF (2012) Vehicle detection and tracking under various lighting conditions using a particle filter. IET Intell Transp Syst 6(1):1–8
13. Niknejad HT, Takeuchi A, Mita S, McAllester D (2012) On-road multivehicle tracking using deformable object model and particle filter with improved likelihood estimation. IEEE Trans Intell Transp Syst 13(2):748–758
14. Long T, Jiao W, He G, Wang W (2013) Automatic line segment registration using Gaussian mixture model and expectation-maximization algorithm. IEEE J Sel Top Appl Earth Obs Remote Sens 7(5):1688–1699
15. Chen YL, Wu BF, Lin CT, Fan CJ, Hsieh CM (2009) Real-time vision-based vehicle detection and tracking on a moving vehicle for nighttime driver assistance. Int J Robot Autom 24(2):89–102
16. Sun Z, Bebis G, Miller R (2006) Monocular precrash vehicle detection: features and classifiers. IEEE Trans Image Process 15(7):2019–2034

17. Junior OL, Nunes U (2008) Improving the generalization properties of neural networks: an application to vehicle detection. In Proceedings of the 2008 11th international IEEE conference on intelligent transportation systems, pp 310–315. IEEE.
18. Yan G, Yu M, Yu Y, Fan L (2016) Real-time vehicle detection using histograms of oriented gradients and AdaBoost classification. Optik 127(19):7941–7951
19. Negri P, Clady X, Hanif SM, Prevost L (2008) A cascade of boosted generative and discriminative classifiers for vehicle detection. EURASIP J Adv Signal Process 2008:1–12
20. Withopf D, Jahne B (2006) Learning algorithm for real-time vehicle tracking. In: Proceedings of the 2006 IEEE intelligent transportation systems conference, pp 516–521. IEEE.
21. Chen SC, Shyu ML, Peeta S, Zhang C (2005) Spatiotemporal vehicle tracking: the use of unsupervised learning-based segmentation and object tracking. IEEE Robot Autom Mag 12(1):50–58
22. Uy ACP, Quiros ARF, Bedruz RA, Abad A, Bandala A, Sybingco E, Dadios EP (2016) Automated traffic violation apprehension system using genetic algorithm and artificial neural network. In: Proceedings of the 2016 IEEE region 10 conference (TENCON), pp 2094–2099. IEEE.
23. City Brain project (2016). https://www.alibabacloud.com/solutions/intelligence-brain/city. Accessed 26 Jan 2020
24. Vivacity traffic management system (2016). https://vivacitylabs.com/technology/. Accessed 26 Jan 2020
25. Traffic Congestion Survey (2015). https://www.reuters.com/article/us-usa-traffic-study/u-s-commuters-spend-about-42-hours-a-year-stuck-in-traffic-jams-idUSKCN0QV0A820150826, 2015. Accessed 26 Jan 2020.
26. Dutta T, Pal G (2010) Pulmonary function test in traffic police personnel in Pondicherry. Indian J Physiol Pharmacol 54(4):329–336
27. Redmon J, Farhadi A (2018) Yolov3: an incremental improvement. arXiv preprint arXiv:1804.02767.
28. Hilario CH, Collado JM, Armingol JM, De La Escalera A (2005) Pyramidal image analysis for vehicle detection. In: Proceedings of the IEEE intelligent vehicles symposium, 2005, pp 88–93. IEEE.
29. Sun D, Roth S, Black MJ (2010) Secrets of optical flow estimation and their principles. In: Proceedings of the 2010 IEEE computer society conference on computer vision and pattern recognition, pp 2432–2439. IEEE.
30. Kim G, Kim H, Park J, Yu Y (2011) Vehicle tracking based on kalman filter in tunnel. In: Proceedings of the international conference on information security and assurance, pp 250–256. Springer, Berlin, Heidelberg.
31. Gustafsson F, Gunnarsson F, Bergman N, Forssell U, Jansson J, Karlsson R, Nordlund PJ (2002) Particle filters for positioning, navigation, and tracking. IEEE Trans Signal Process 50(2):425–437
32. Exner D, Bruns E, Kurz D, Grundhöfer A, Bimber O (2010) Fast and robust CAMShift tracking. In: Proceedings of the 2010 IEEE computer society conference on computer vision and pattern recognition-workshops, pp 9–16. IEEE.
33. Bewley A, Ge Z, Ott L, Ramos F, Upcroft B (2016) Simple online and realtime tracking. In: Proceedings of the 2016 IEEE international conference on image processing (ICIP), pp 3464–3468. IEEE.
34. Open Images Dataset. https://storage.googleapis.com/openimages/web/index.html. Accessed 11 March 2020
35. COCO Dataset. http://cocodataset.org. Accessed 20 Feb 2020.
36. Pascal VOC Dataset. http://host.robots.ox.ac.uk/pascal/VOC/. Accessed 14 March 2020.
37. Stanford Cars Dataset. https://ai.stanford.edu/~jkrause/cars/car_dataset.html. Accessed 26 Feb 2020.
38. Sivaraman S, Trivedi MM (2010) A general active-learning framework for on-road vehicle recognition and tracking. IEEE Trans Intell Transp Syst 11(2):267–276
39. Nguyen A, Yosinski J, Clune J (2015) Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 427–436.
40. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
41. Kulikajevas A, Maskeliunas R, Damasevicius R, Ho ES (2020) 3D object reconstruction from imperfect depth data using extended YOLOv3 network. Sensors 20(7):2025
42. Li Y, Han Z, Xu H, Liu L, Li X, Zhang K (2019) YOLOv3-lite: a lightweight crack detection network for aircraft structure based on depthwise separable convolutions. Appl Sci 9(18):3781

Affiliations

Mohammed A. A. Al-qaness[1] · **Aaqif Afzaal Abbasi**[2] · **Hong Fan**[1] · **Rehab Ali Ibrahim**[3] · **Saeed H. Alsamhi**[4] · **Ammar Hawbani**[5]

✉ Mohammed A. A. Al-qaness
alqaness@whu.edu.cn

Aaqif Afzaal Abbasi
aaqif.afzaal@fui.edu.pk

Hong Fan
hfan3@whu.edu.cn

Rehab Ali Ibrahim
rehab100r@yahoo.com

Saeed H. Alsamhi
salsamhi@ait.ie

Ammar Hawbani
anmande@ustc.edu.cn

1 State Key Laboratory for Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan 430079, China

2 Department of Software Engineering, Foundation University, Islamabad 44000, Pakistan

3 Department of Mathematics, Faculty of Science, Zagazig University, Zagazig 44519, Egypt

4 Software Research Institute, Athlone Institute of Technology, Athlone, Ireland

5 School of Computer Science and Technology, University of Science and Technology of China, Hefei 230031, Anhui, China