



Advanced Remote Sensing

<http://publish.mersin.edu.tr/index.php/arcej>

e-ISSN 2979-9104



Performance analysis of YOLO versions for automatic vehicle detection from UAV images

Melis Uzar*¹, Şennur Öztürk¹, Onur Can Bayrak¹, Tümay Arda¹, Nursu Tunalioglu Öcalan¹

¹Yıldız Technical University, Geomatics Engineering Department, Türkiye, auzar@yildiz.edu.tr, sennurozturk95@gmail.com, onurcb@yildiz.edu.tr, tarda@yildiz.edu.tr, ntunali@yildiz.edu.tr

Cite this study: Uzar, M., Öztürk, Ş., Bayrak, O. C., Arda, T., & Öcalan, N. T. (2021). Performance analysis of YOLO versions for automatic vehicle detection from UAV images. *Advanced Remote Sensing*, 1(1), 16-30

Keywords

UAV
Vehicle detection
YOLO
Deep learning
Remote sensing

Research Article

Received: 14.11.2021
Revised: 11.12.2021
Accepted: 18.12.2021
Published: 30.12.2021

Abstract

Automatic vehicle detection, one of the study areas in Remote Sensing facilities, has become widely used on several issues such as transportation, disaster management, highway management, parking lot management and real-time vehicle detection in smart cities. In recent years, deep learning methods have been widely preferred in vehicle detection. Although this method has advantages such as high accuracy and speed of detection, some problems such as not detecting vehicles, double detection and class confusion in detection from digital images caused by vapor and shadow in adverse weather conditions (i.e., rain, fog, sunlight) have been raised. Thus, vehicle detection is still a significant issue that should be studied. In this study, versions of You Only Look Once (YOLO), one of the deep learning (DL) architectures, have been investigated in terms of performance assessments of vehicle detection in parking lots. To perform the analysis, Unmanned Aerial Vehicle (UAV)-based images collected from Yıldız Technical University, Campus of Davutpasa (dated 2018) were used. The labeling process was performed for three classes (car, bus, and minibus) using the Visual Object Tagging Tool (VoTT). The labeled dataset has been trained via transfer learning in YOLOv4-CSP, YOLOv4-tiny, YOLOv4-P5, YOLOv4-P6, YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x architectures. The weights of YOLO versions have been implemented to the parking lots and results have been compared. To assess the performance of YOLO-based vehicle detection, mAP and F1-Score values were computed.

1. Introduction

Recently, the integration of remote sensing, photogrammetry and deep learning algorithms have obtained fast, real-time, and highly-accurate results. It is obvious that it has opened a new era in various disciplines with technological developments in the fields of deep learning and computer vision. Several object detection studies have been carried out in urban areas. Automatic vehicle detection has become one of the most important topics in the management of highways and parking lots in urban areas, considering the rapidly increasing population and the transportation vehicles that participate in traffic jams.

Cheng et al. [1] carried out a pixel-based classification study for vehicle detection by applying a color filter that separates the colors of vehicles and other objects. The vehicles were identified using the Canny edge detector and classified by Dynamic Bayesian Network (DBN). Chen et al. [2] tackled whether conventional methods are insufficient or not when there is a complex background. They decided to use Deep Neural Networks (DNN) because

of its structure that can learn rich features from the training data. Hence, they presented the Hybrid DNN (HDNN) model by making changes in the pooling layer for the model to detect data at different scales on the DNN. De Almeida et al. [3] introduced a dataset on parking lots and aimed to detect the empty and occupied areas in the parking lot by Support Vector Machines (SVM). In experiments, where the same data was used for both training and testing, accuracy rates of over 99% were achieved. However, when working with different data, the best result was obtained about 89%. In Tan [4], the vehicle dataset consisting of 2000 images of different brands and models was trained with Faster-R-CNN-ResNet 50, Faster-RCNN-ResNet 101, R-FCN-ResNet 101 and SSD-MobileNet using transfer learning. When the results were examined, the Faster RCNN-ResNet 50 model was the most successful with 94%.

Since object detection is a critical part in Automatic Driving Systems (ADS) and Driver Assistance Systems (DAS), current real-time detection models for small vehicle objects suffer from low sensitivity and poor performance. The detection speed of the YOLO algorithm has been of great benefit in the development of automatic driving systems [5-6]. Bui et al. [7] used object detection and tracking models such as YOLO and DeepSort to analyze traffic in complex areas (e.g., intersections). To improve the vehicle counting problem, they propose a zone monitoring approach that can work well with a variety of scenarios, especially in areas with complex movements. The proposed model has been evaluated in the CVPR AI City Challenge 2020 dataset. Accordingly, the method achieved 85% accuracy. Han et al. [8] proposed a new real-time object detection model named YOLOv2, which is optimized based on the YOLOv2 deep learning framework, to be applied to small vehicle objects. In the proposed model, a new structure is introduced to strengthen the feature extraction capability of the network by adding convolution layers to YOLOv2. 94% accuracy was achieved in the model, whose effectiveness was investigated using the open-source dataset KITTI. Wang et al. [9] tested Faster-RCNN, R-FCN, SSD, RetinaNet, and YOLOv3 models using the KITTI dataset. According to the results, R-CNN-type models that work locally in two stages give more accurate results, while the speed of models such as YOLO and SSD.

In this study, the latest versions of YOLO namely YOLOv4 and YOLOv5 were utilized for performance analysis of vehicle detection. Target object classes were aimed as follows: car, minibus and bus. The properties of the study area and the dataset obtained by the UAV system are presented in the next section, followed by an explanation of the methodology adopted. The results of the experiment are reported along with an accuracy assessment of the automatic vehicle detection results with the versions of YOLO deep learning analyses, in the Results and Discussion section, and the conclusion is given in the last section.

2. Material and Method

You Only Look Once (YOLO), which is a deep learning architecture, utilized for the automatic vehicle detection analysis with the data obtained by UAV. The workflow of the proposed methodology is given in Figure 1. In the first stage of automatic on-ground vehicle detection with YOLO architecture, data preparation, i.e., labeling, data augmentation, training, validation and test data split, was made. Model training with transfer learning was used for different versions of YOLO so-called YOLOv4-CSP, YOLOv4-tiny, YOLOv4-P5, YOLOv4-P6, YOLOv5s, YOLOv5l, YOLOv5m, YOLOv5n and YOLOv5x. The performance analysis of target classes (car, minibus and bus) was evaluated with nine YOLO versions and then accuracy and performance analyses were performed. In this study, Roboflow software was used for data augmentation processes using UAV images. The labeling processes of the target classes were created using Visual Object Tagging Tool (VoTT) software. Mean Average Precision (mAP), F1-Score, precision and recall parameters were computed for performance analysis of YOLO versions for vehicle detection.

2.1. Study area and dataset

In this study, parking lots located at Davutpasa Campus of Yildiz Technical University in Istanbul, Turkey were selected as regions of interest (RoI) (Figure 2). The aerial images were collected with the UAV system in 2018. Figure 3 shows samples from RoI including parking lots. Totally 94 images were obtained with a size of 5472 * 3648 pixels and a resolution of 72 dpi.

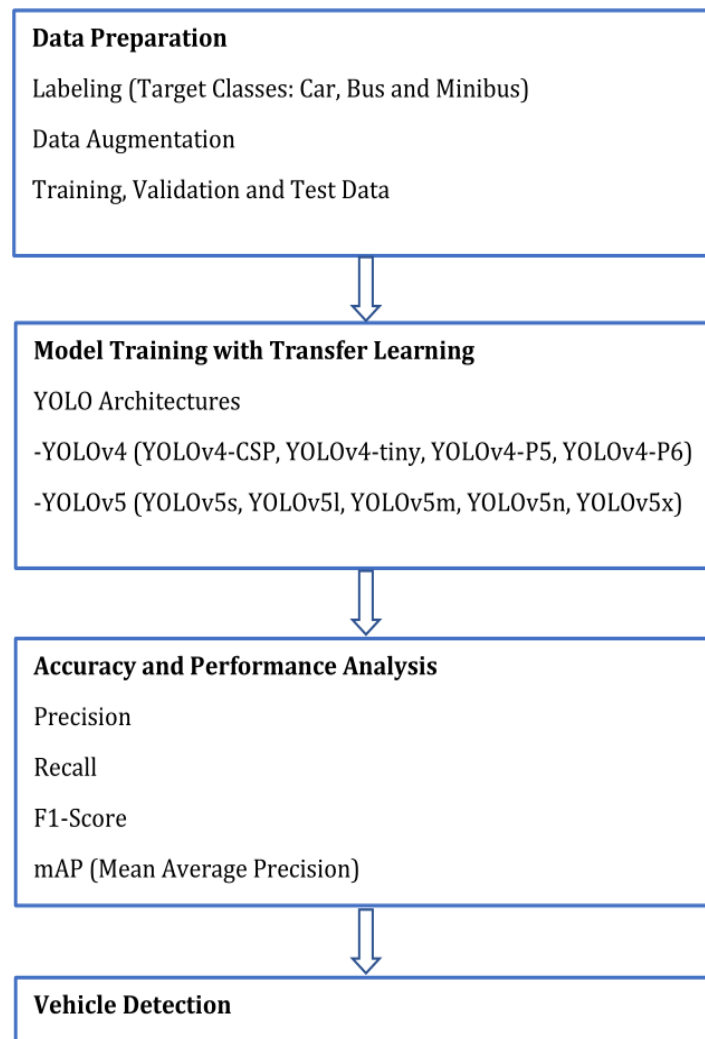


Figure 1. Workflow of vehicle detection

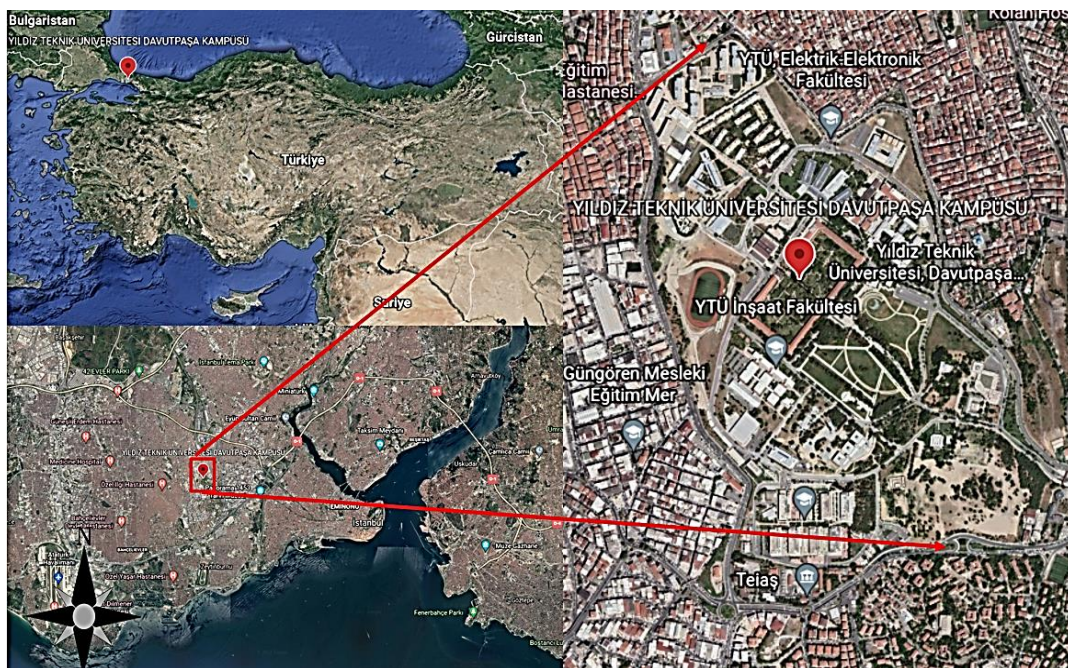


Figure 2. Study area

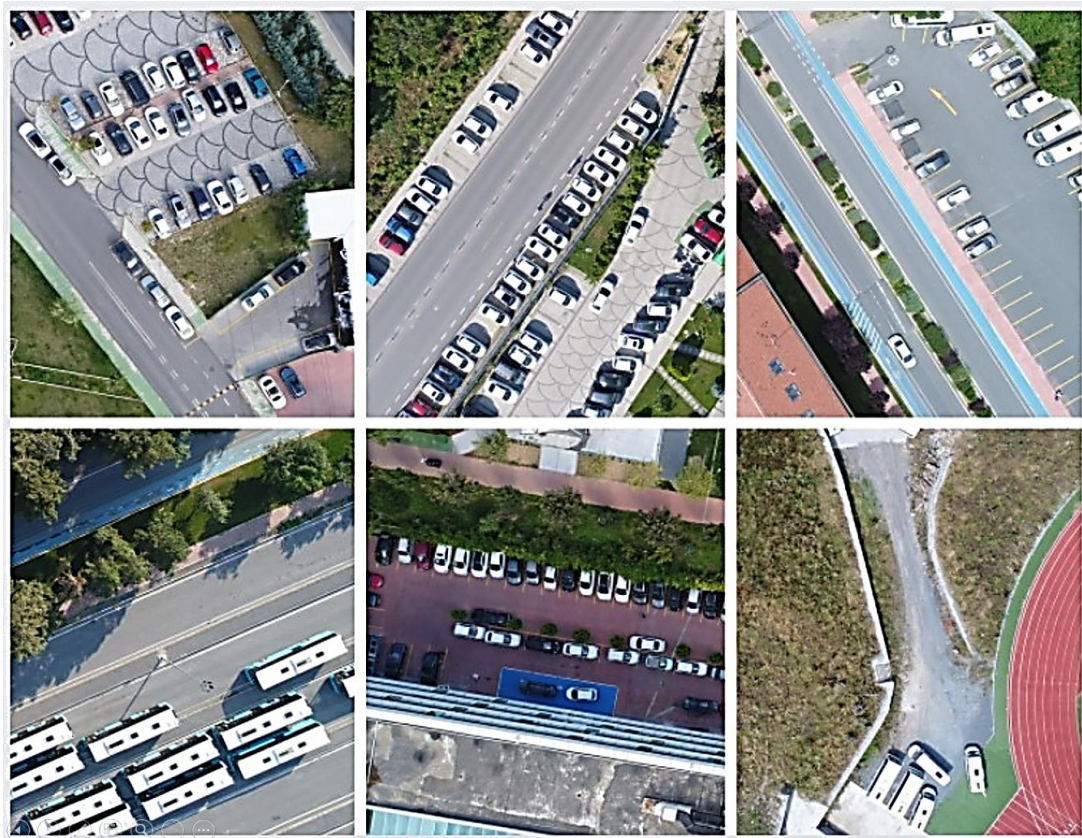


Figure 3. Some samples of parking lots from RoI

2.2. Data preparation

Data preparation phase was carried out in 3 main stages as (1) labeling, (2) data augmentation, and (3) training, validation and test data split.

2.2.1. Labelling

Data labeling is a process that transforms data into a descriptive format for artificial intelligence (AI) applications. Generating the desired output in AI-based models is highly dependent on accurate and well-labeled data. Labeled data should be reliable, accurate and consistent. Features that do not belong to any target class should not be labeled. The required number of labels should be made in the data and the data for each label should be sufficiently various.

2.2.2. Data augmentation

Data augmentation is a technique, which is commonly used to train large neural networks by increasing the diversity of the data without collecting new samples. This transformation technic is closely related to digital image processing in data analysis, which used to augment images in deep learning. In most of the studies, the data augmentation techniques are applied to the dataset to contribute to the representative ability of the dataset. In this study we utilized the fundamental three methods of data augmentation as follows; brightness analysis, shear and rotation. Brightness analysis is used to change the brightness of the image. After this technique the result image becomes darker or lighter than the original one. The process of shear transformation, which is to fix the one axis and stretch the image at a certain angle known as the shear angle. The image is rotated randomly by an angle in the range of (+) value to (-) value in the rotation process. The samples of brightness, shear and rotation images from study were given in [Figure 4](#).

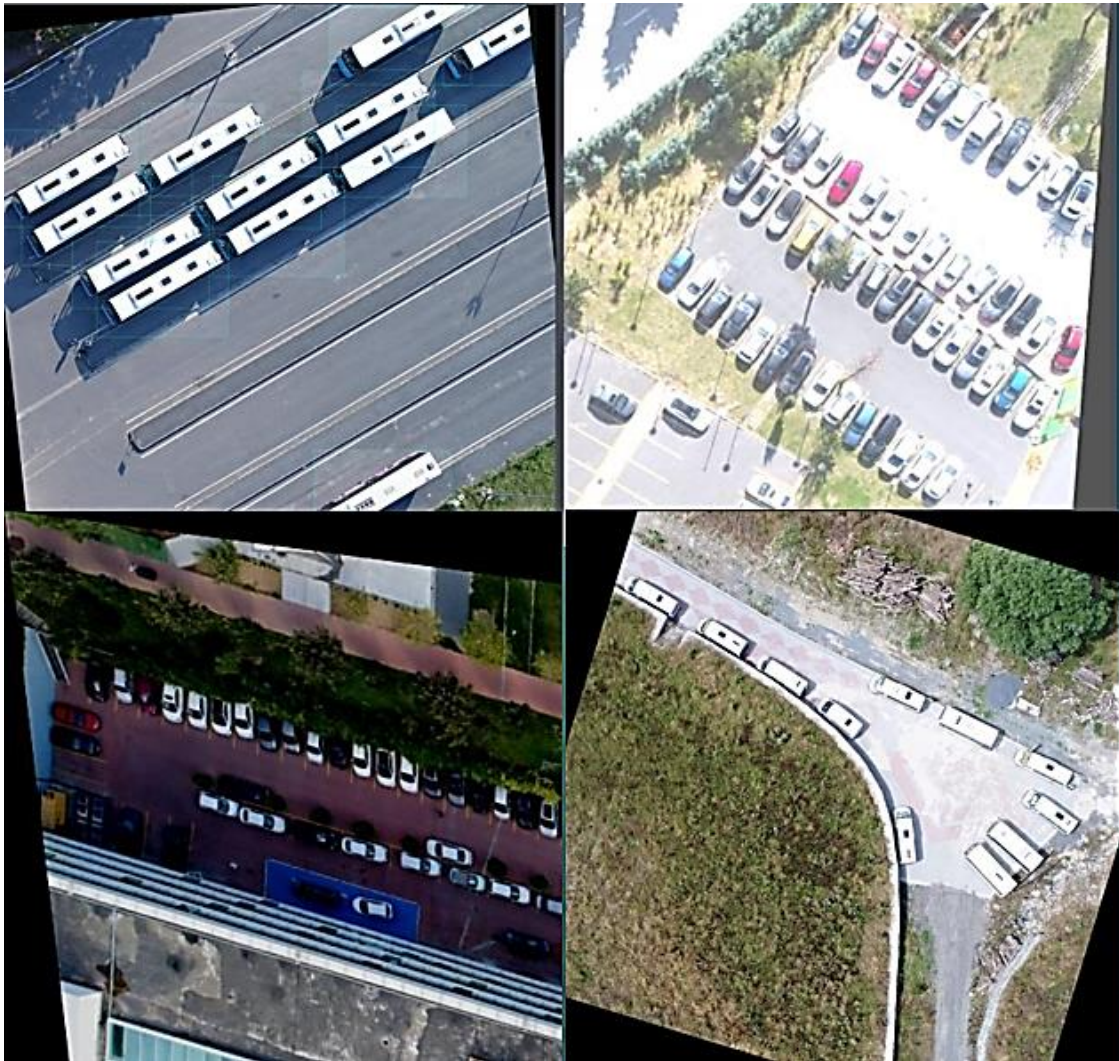


Figure 4. The samples of brightness, shear and rotation images

2.2.3. Training, validation and test data

One of the important issues of machine learning is the generalization of the developed model. Generalization is an indicator representing the performance and fitness of the model for new data. Data are separated into training, validation and test dataset to measure generalization. Training, validation and test data have a strong relation on fitness of model (Figure 5). Training dataset is a sample of data used to fit the model during the learning process. Training data determines the parameters of the model. The validation dataset is used to provide an unbiased evaluation of a model fit on the training dataset while tuning the hyperparameter. The evaluation process of the validation data is to determine the hyperparameter values for the model. Test data is the part of the dataset used to assess the final fit of the model performance.



Figure 5. Splitting the train, validation and test dataset

2.3. Model Training with Transfer Learning for YOLO Architectures

2.3.1 Transfer learning

Re-training a model on a new task that was pre-trained for another task is known as transfer learning. In order to train complex models, the dataset must be large enough and representative of the real situation. Generally, most remote sensing-based transfer learning works are focused on updating the weights of a deep learning solution from another context to the current task based on available training data [10]. However, previously trained dataset

is often used to take advantage of the time and expert knowledge. Transfer learning is valuable for data science because it reduces the need for large amounts of data. Using transfer learning instead of end-to-end learning provides better performance with less data and less training time. Hence, the use of MS COCO dataset, which includes 2.5 million labeled samples of 91 objects in 328,000 images was preferred. In addition, selection of this dataset's pre-trained models is appropriate since it covers car, bus, and minibus classes.

2.3.2. YOLO Architectures

In this study, the latest versions of YOLO namely YOLOv4 and YOLOv5 were used for performance analysis of vehicle detection. The YOLOv4 model was developed with Cross Stage Partial (CSP) structure [11]. This model is configured as backbone, neck and head. Backbone is the layer of the feature extraction process, which is added as a middle layer to find more features while estimating objects. YOLOv4 uses Spatial Attention Module (SAM), Path Aggregation Network (PAN) and spatial pyramid pooling (SPP) instead of Feature Pyramid Network (FPN). At head, there are bounding boxes and the estimated boxes of each class. In literature, there are versions of YOLOv4 namely YOLOv4-CSP, YOLOv4-tiny, YOLOv4-P5, YOLOv4-P6 and YOLOv4-P7. The Scaled YOLOv4 model was developed by [12], which has the YOLOv4 object detection neural network based on the CSP approach scaled up and down (Figure 6). This model is adaptable to all network structures while maintaining optimum speed and accuracy. The CSP layer, which was added to the YOLO architecture, has a simple structure and offers very efficient results. In this structure, half of the data produces semantic information as it moves along the line, while the other half is added to the model later, preserving both spatial information and some properties. Scaled YOLOv4 changes not only the depth, width, resolution of the mesh, but also its structure. After the scaled YOLOv4 model, the YOLOv4-tiny model was developed for low-capacity GPUs. The YOLOv4-tiny model has different considerations from the scaled YOLOv4 model because different constraints such as memory bandwidth and memory access are considered in it. The backbone structure of the YOLOv4-tiny model uses OSANet, which offers favorable computational complexity at small depth. After the YOLOv4-tiny model, the YOLOv4-Large model was developed for high-capacity GPUs and scaled from P5 to P7. In this model, as the scale increases along with the depth and width of the model, the complexity of the structure also increases. The depth scale is given as 1, 3, 15, 15, 7, 7 and 7 from small model to large model at each stage, respectively.

The YOLOv5 model was introduced by Glenn Jocher after the release of YOLOv4. The generalized structure of the YOLOv5 model is given in Figure 7. In this model, as in the YOLO v4 model, CSPNet backbone and PANet neck are used. Likewise, the same structure of the model in YOLOv4 is used in the head part of YOLOv5. The difference between YOLOv5 versions (YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l and YOLOv5x) is the scaling factors of the width and depth of the mesh (Figure 8). In Table 1, the number of layers, parameter numbers and performance information about the versions are given.

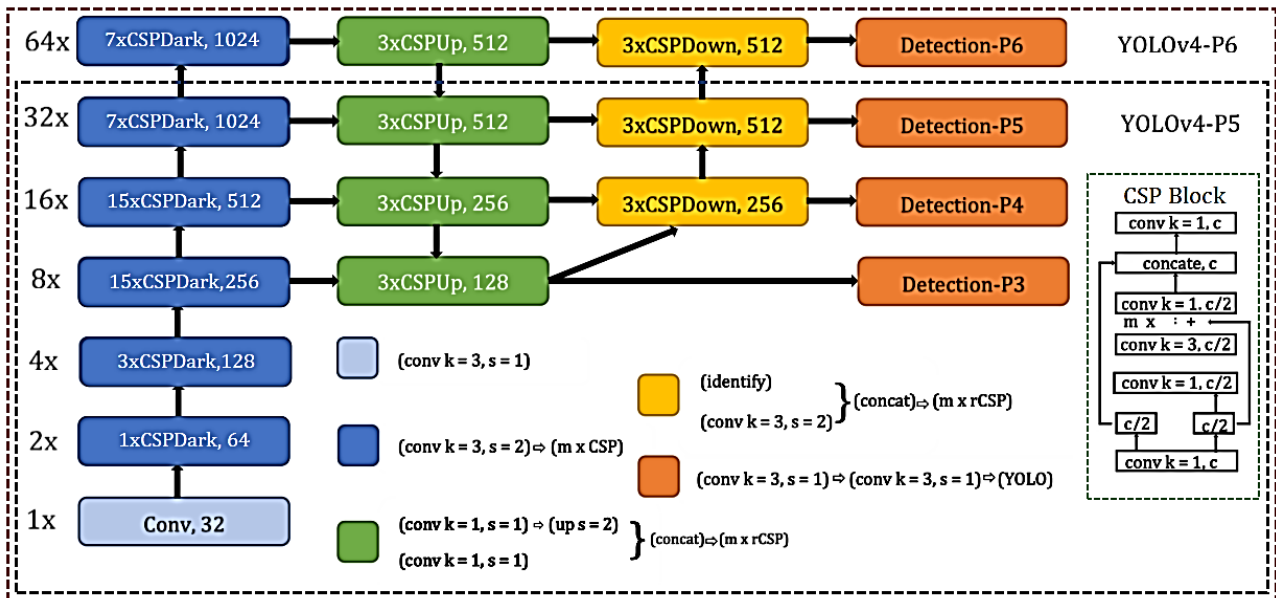


Figure 6. Scaled YOLOv4 layer architecture [12]

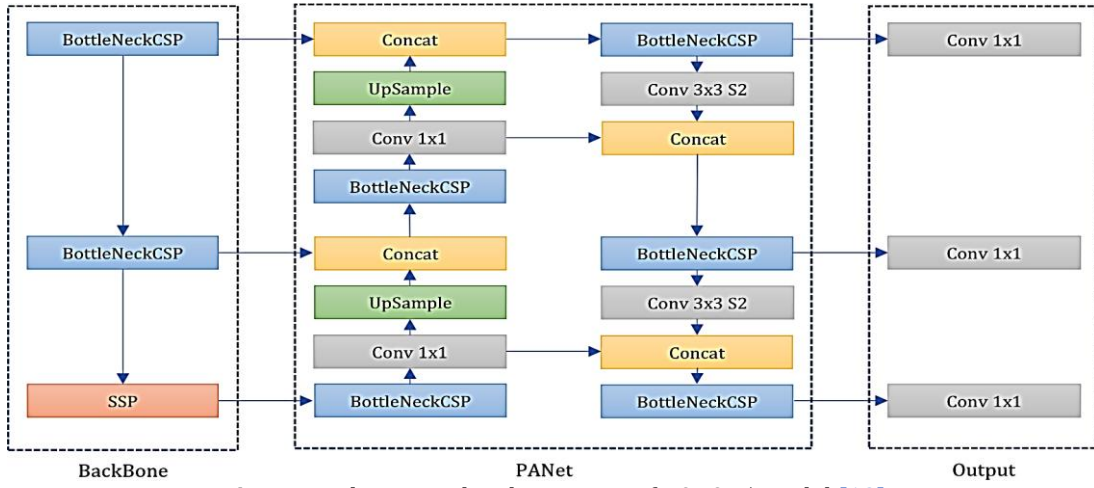


Figure 7. The generalized structure of YOLOv5 model [13]

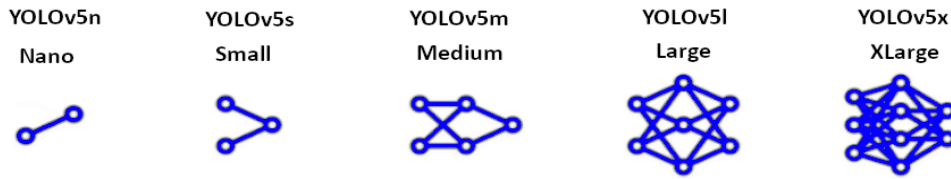


Figure 8. Scales of YOLOv5

Table 1. Specifications of YOLOv5 models

Model	Number of Layers	Number of Parameters	FLOPs
YOLOv5n	270	1.767.976	4.2
YOLOv5s	270	7.027.720	15.9
YOLOv5m	369	20.879.400	48.1
YOLOv5l	468	46.149.064	108.0
YOLOv5x	567	86.231.272	204.2

2.4. Accuracy and performance analysis

To assess the performance of YOLO-based vehicle detection, mAP and F1-Score values were computed. Actual annotations were compared with predictions according to confusion matrix (Table 2).

Table 2. Confusion Matrix

Confusion Matrix		Actual	
		Positive	Negative
Predicted	Positive	TP (True Positive)	FP (False Positive)
	Negative	FN (False Negative)	TN (True Negative)

Moreover, some performance metrics are given below in (Eq.1-5);

$$\text{Precision} = \frac{TP}{TP + FP} \quad (1)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2)$$

$$\text{F1 - Score} = \frac{2}{\left(\frac{1}{\text{Precision}}\right) + \left(\frac{1}{\text{Recall}}\right)} \quad (3)$$

$$\text{AP} = \sum_{k=0}^{k=n-1} [\text{Recall}(k) - \text{Recall}(k + 1)] * \text{Precision}(k) \quad (4)$$

$$\text{mAP} = \frac{1}{n} \sum_{k=1}^{k=n} \text{AP}_k \quad (5)$$

where, AP and mAP are the average and mean average precisions; n is the number of classes and AP_k is the AP value of the related class (k).

3. Results

3.1. Data preparation

Before the data preparation process, pre-processes were applied to ensure the availability and suitability of the aerial images of the study area with the YOLO architecture and to increase the accuracy. At this stage, the images were cropped as the dimensions of 2048 x 2048 pixels. In the data preparation process, the images were labeled in three classes: car, bus and minibus, which were vehicles on campus parking lots. There are 1223 labels of car, 122 labels of minibus and 54 labels of bus in the dataset. After this step, data augmentation was performed, followed by the dataset split process utilized as 75%, 15%, and 10% of training, validation and test, respectively. The values of ± 10 brightness, 15-degree vertical shear, and $\pm 15, 25, \text{ and } 90$ -degree rotation were applied in data augmentation.

3.2 YOLO model training

In the training process, the pre-trained weights of the MS COCO dataset were accepted as the initial weights and imported into YOLO architectures. The hyperparameters of the YOLOv4-CSP, YOLOv4-tiny, YOLOv4-P5, YOLOv4-P6, YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l and YOLOv5x models for training were given in [Table 3](#).

Table 3. The hyperparameter values used in training

Hyperparameters	Values
Batch Size	16
Input Size	416 x 416
Momentum	Momentum: 0.937 Initial Learning Size: 0.01
Epoch Size	200
Activation Function	YOLOv4: Mish; YOLOv5: Leaky ReLU
Optimization Algorithm	ADAM

3.3 Accuracy Analysis of Model Training

In this study, nine different YOLO versions were trained and [Table 4](#) includes the results of the accuracy analyses of trained models. Analyses were performed by comparing metrics computed between actuals and predictions.

Table 4. Training metrics of trained models (fps refers to frame per second)

MODEL	Precision	Recall	F1-Score	mAP	fps
YOLOv4-CSP	0.61	0.78	0.68	0.76	11
YOLOv4-tiny	0.86	0.91	0.89	0.75	63
YOLOv4-P5	0.47	0.86	0.61	0.73	9
YOLOv4-P6	0.38	0.89	0.53	0.75	8
YOLOv5n	0.77	0.80	0.79	0.82	63
YOLOv5s	0.85	0.78	0.81	0.80	40
YOLOv5m	0.8	0.78	0.79	0.84	25
YOLOv5l	0.84	0.75	0.79	0.79	12
YOLOv5x	0.87	0.74	0.80	0.79	10

According to [Table 4](#), the YOLOv4-tiny model provides the F1-Score of 0.89 as the highest value. Although YOLOv5 models perform similar results, it is seen that YOLOv4-CSP, YOLOv4-P5, YOLOv4-P6 models offer relatively lower F1-Scores compared to the other architectures. Additionally, all of the YOLOv5 models perform higher mAP values than YOLOv4 models. The YOLOv5m provides the mAP of 84% as the highest value. In order to perform real-time vehicle detection, the fps values of the models were considered and compared. In [Table 4](#), it is seen that YOLOv4-tiny and YOLOv5n models with a processing speed of 63 fps are the fastest ones, which are

developed for low-performance systems. The slowest models are YOLOv4-P5, YOLOv4-P6 and YOLOv5x that are developed for high performance systems.

As seen from Table 4, all of the YOLOv5 models provided the highest mAP values. Thus, mAP results regarding the classes i.e., car, minibus, and bus were analyzed for the YOLOv5 models. Table 5 represents the mAP results regarding the classes i.e., car, minibus, and bus for YOLOv5 models. The results show that cars are the most accurately detected class in all models. According to the table, the classes of minibus and bus reduce the overall accuracy. In Figure 9, detection results of target classes; minibus, bus and car for YOLOv5, in Figure 10 the sample display of results for cars and in Figure 11 the sample detection results of target classes; minibus, bus and car are given.

Table 5. The mAP analysis for classes

Model	Car	Minibus	Bus
YOLOv5n	0,94	0,80	0,60
YOLOv5s	0,82	0,73	0,77
YOLOv5m	0,90	0,73	0,74
YOLOv5l	0,94	0,82	0,66
YOLOv5x	0,93	0,82	0,61

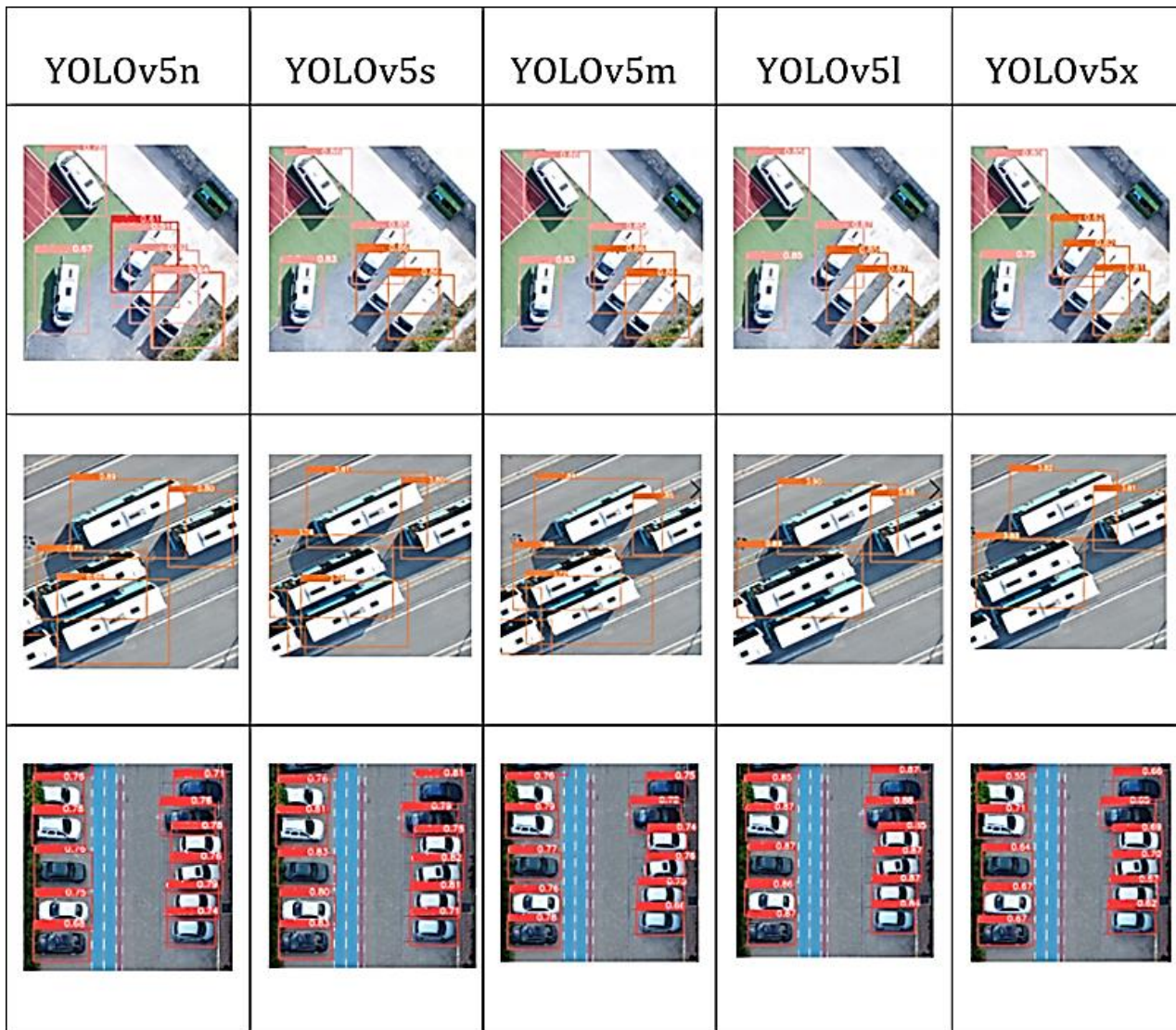


Figure 9. Detection results of target classes; minibus, bus and car for YOLOv5



Figure 10. The sample display of results for cars



Figure 11. The sample display of results from target classes

4. Discussion

YOLO versions were preferred to perform the performance analysis of vehicle detection in terms of speed, accuracy, and learning capabilities. The YOLO model improves the speed of detection due to its working structure as in real-time. This model requires only a single forward propagation through a neural network to detect vehicles and also use a single bounding box regression to estimate the height, width, center, and target class of vehicles. YOLO is a prediction technique that provides accurate results with minimum errors. The architecture of YOLO has a learning capability for vehicle detection that enables it to learn the representations of vehicles.

According to the results, the highest accuracy value for F1-Score was provided by the YOLOv4-tiny model with 89% among all of the YOLO models in detecting vehicles in parking lots. Moreover, all of the YOLOv5 models present accuracies in the range of 79%-81%. Nevertheless, the accuracies of YOLOv4-CSP, YOLOv4-P5 and YOLOv4-P6 models were found below 70%. According to the mAP criteria, preferred widely in the literature to perform detection performance, the best accuracy value was obtained in the YOLOv5m model with 84%. For the other YOLOv5 models, the lowest mAP value was 79% and for all of the YOLOv4 models, mAP values were in the range of 73%-76%. According to the mAP results, an accuracy value of over 82% was obtained for car class in all of the models. The overall accuracy reduced depending on bus and minibus classes. There are a low number of buses and minibus in the training dataset, because the study area was restricted by the campus boundaries.

In this study, another significant factor for selecting YOLO models in vehicle detection is the speed of detection. Labeling time of 94 images with a size of 2048 x 2048 took 5 hours. While model training in a cloud environment with the support of Google-Colaboratory GPU took approximately 2 hours for models with low depth, for the models with higher depth and complexity, that took about 7 hours. The total training duration for 9 models was about 40 hours. When the YOLO models were evaluated in terms of speed, it is seen that the speed results were provided directly proportional to the increase in scale. The fastest models are YOLOv5n and YOLOv4-tiny models with 63 fps. The YOLOv5s model with a speed of 40 fps and YOLOv5m model with a speed of 25 fps showed a performance close to the small-scale models. The detection speeds of the other models were in the range of 8-12 fps.

As a result, the advantages and disadvantages of different YOLO model versions were compared within the scope of vehicle detection and their performances were analyzed in this study. YOLOv4 and YOLOv5, which are the two advanced models of the YOLO architecture so far, have been preferred because of the detection success in accuracy. For this purpose, the most common problems; black vehicle detection, vehicle shadow detection, non-vehicle object detection, detection of vehicles covered by trees and not being able to detect vehicles were analyzed for YOLOv4 and YOLOv5 models.

In the model performance analysis, the correctly captured vehicle is shown with a frame in all of the figures. First, it was observed that there were incorrect or missing vehicle detection in the images obtained when trees covered the vehicles (Figure 12). Secondly, an important factor affecting the vehicle detection accuracy of the models is the errors due to shadows (Figure 13). Thirdly, in some YOLO models as shown in Figure 14 and Figure 15, vehicles were not detected correctly when dark colored vehicles such as black and gray glow under the

influence of sunlight and have the same pixel gray value as the road object. Another important problem is that inaccurate vehicle detections caused by non-vehicle objects with the same geometric and color characteristics as the vehicle have been observed. Sidewalk (Figure 16) and container (Figure 17) could be given as an example of non-vehicle object detection. One of the important reasons for this situation is that the model was not trained well enough. In other words, as a result of insufficient labeling of different objects, the model cannot learn the properties of the objects and detects the target classes incomplete or incorrectly.

It is seen that the color texture of the vehicles in the image has the same reflection value as the road, due to the fact that dark colored vehicles such as black and gray shine with the effect of sunlight. For this reason, the detection of road and vehicle objects according to the working principle and weight parameters in the structures of some YOLO models has not been carried out correctly. On the other hand, non-vehicle object detection problems have occurred in YOLOv4-CSP, YOLOv4-tiny, YOLOv4-P5, YOLOv4-P6, YOLOv5l, YOLOv5x models. Considering this problem, it has been observed that objects with similar geometry and close-image gray values to the training data labeled as vehicles are detected as vehicles incorrectly. On the other hand, in the YOLOv5n, YOLOv5s, and YOLOv5m models among all of the models, accurate vehicle detection has been achieved in containers, sidewalk and road objects, which have the same gray values as the bus.

5. Conclusion

In this study, the YOLO model, one of the commonly used object detection architectures in deep learning, was implemented for performance analysis of automatic vehicle detection using UAV-based aerial images within the scope of YTU Davutpasa Campus parking lots. For the performance analysis of vehicle detection, 9 different versions of the YOLO models namely YOLOv4-CSP, YOLOv4-tiny, YOLOv4-P5, YOLOv4-P6, YOLOv5s, YOLOv5l, YOLOv5m, YOLOv5n, YOLOv5x were utilized. Firstly, the data preparation stage, including labeling, data augmentation, training, validation and test data split, were performed. Secondly, model training with transfer learning was carried out for YOLO versions. At the training stage, the weights trained with the MS COCO dataset were accepted as initial weights and included in the deep learning network using transfer learning. The car, minibus and bus were labeled as target classes. A confusion matrix was created for the target classes and the results were compared in terms of mAP, recall, precision, and F1-Score values. In addition, the accuracy analysis and speed comparisons of the models were considered. At the last stage, the weights trained were applied to the test data containing the parking lots.

According to the results, when analyzing the dataset with a limited GPU support, it is seen that large-scale models could not be trained properly. Thus, to determine the real performances of YOLOv4-P5, YOLOv4-P6, YOLOv5l and YOLOv5x models, it is recommended to train models with an unlimited GPU support and more training epochs. Moreover, the number and diversity of the dataset should be increased with the use of high-capacity processors.

In recent years, integration of remote sensing and photogrammetry with deep learning methods will provide significant solutions especially in automatic object detection and extraction in terms of high accuracy, speed, and real-time data processing. In particular, this integration will be a very important data source for local and private administrations such as for the management of smart cities with accurate and real-time vehicle detection, planning at roads and intersections, and management of transportation. In deep learning algorithms, transferring the features of objects to the model as parameter values with different weights using training networks and image processing techniques will become the most preferred method of obtaining data information in the future. Today, where accurate data is the most valuable information, it can be concluded that object classes belonging to roads and road networks will be supportive for the management and planning of highways and parking lots thanks to large data processing and integration with geographic information systems.

Funding

This research received no external funding.

Author contributions

Melis Uzar: Conceptualization, Methodology, Writing-Original draft preparation, **Şennur Öztürk:** Methodology, Software, Data curation, **Onur Can Bayrak:** Investigation, Software, Data curation, **Tümay ARDA:** Visualization, Validation, **Nursu Tunalioglu Öcalan:** Methodology, Writing-Reviewing and Editing.

Conflicts of interest

The authors declare no conflicts of interest.

YOLOv4-CSP	YOLOv4-tiny	YOLOv4-P5	YOLOv4-P6	
				
YOLOv5n	YOLOv5s	YOLOv5m	YOLOv5l	YOLOv5x
				

Figure 12. The effect of trees on vehicle detection










YOLOv4-CSP	YOLOv4-tiny	YOLOv4-P5	YOLOv4-P6	
				
YOLOv5n	YOLOv5s	YOLOv5m	YOLOv5l	YOLOv5x
				

Figure 13. The effect of shadows in the image on vehicle detection

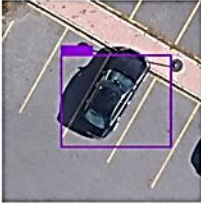

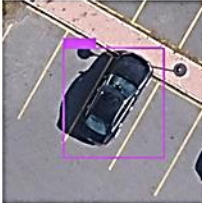






YOLOv4-CSP	YOLOv4-tiny	YOLOv4-P5	YOLOv4-P6	
				
YOLOv5n	YOLOv5s	YOLOv5m	YOLOv5l	YOLOv5x
				

Figure 14. The incorrect detection sample for the black colored vehicle in YOLOv4-P6










YOLOv4-CSP	YOLOv4-tiny	YOLOv4-P5	YOLOv4-P6	
				
YOLOv5n	YOLOv5s	YOLOv5m	YOLOv5l	YOLOv5x
				

Figure 15. The performance of YOLOv4 and YOLOv5 for the black colored vehicle detection










YOLOv4-CSP	YOLOv4-tiny	YOLOv4-P5	YOLOv4-P6	
				
YOLOv5n	YOLOv5s	YOLOv5m	YOLOv5l	YOLOv5x
				

Figure 16. Non-vehicle object detection: sidewalk example

YOLOv4-CSP	YOLOv4-tiny	YOLOv4-P5	YOLOv4-P6	
				
YOLOv5n	YOLOv5s	YOLOv5m	YOLOv5l	YOLOv5x
				

Figure 17. Non-vehicle object detection: container example

References

1. Cheng, H.Y., Weng, C. C., & Chen, Y. Y. (2011). Vehicle detection in aerial surveillance using Dynamic Bayesian Networks. *IEEE Transactions on Image Processing*, 21(4), 2152-2159.
2. Chen, X., Xiang, S., Liu, C. L., & Pan, C.H. (2014). Vehicle detection in satellite images by Hybrid Deep Convolutional Neural Networks. *IEEE Geoscience and remote sensing letters*, 11(10), 1797-1801.
3. De Almeida, P.R., Oliveira, L.S., Britto Jr, A.S., Silva Jr, E.J. & Koerich, A.L. (2015). PKLot–A Robust dataset for parking lot classification. *Expert Systems with Applications*, 42(11), 4937-4949.
4. Tan, Z. (2019). Vehicle classification with deep learning. Master's thesis, Firat University, Institute of Science, Elazığ, 63p.
5. Ataei, B. (2019). Development of video processing algorithm (YOLO) in autonomous vessels operations. Master's Thesis, University of South-Eastern Norway, 64p
6. Peng, B., Keskin, M. F., Kulcsár, B. & Wymeersch, H. (2021). Connected autonomous vehicles for improving mixed traffic efficiency in unsignalized intersections with deep reinforcement learning. *Communications in Transportation Research*, 1, 100017.
7. Bui, N., Yi, H. & Cho, J. (2020). A vehicle counts by class framework using distinguished regions tracking at multiple intersections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 578-579, Seattle, WA, USA.
8. Han, X., Chang, J. & Wang, K. (2021). Real-Time object detection based on YOLO-V2 for tiny vehicle object. *Procedia Computer Science*, 183, 61-72.
9. Wang, H., Yu, Y., Cai, Y., Chen, X., Chen, L. & Liu, Q. (2019). A comparative study of state-of-the-art deep learning algorithms for vehicle detection. *IEEE Intelligent Transportation Systems Magazine*, 11(2), 82-95.
10. Benbahria, Z., Sebari, I., Hajji, H. & Smiej, M. F. (2021). Intelligent mapping of irrigated areas from Landsat 8 images using transfer learning. *International Journal of Engineering and Geosciences*, 6(1), 40-50.
11. Bochkovskiy, A., Wang, C. Y. & Liao, H. Y. M. (2020). YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.
12. Wang, C. Y., Bochkovskiy, A. & Liao, H. Y. M. (2021). Scaled-YOLOv4: Scaling cross stage partial network. In *Proceedings of the IEEE/cvf Conference on Computer Vision and Pattern Recognition*, 13029-13038.
13. URL-1, (2020). <https://github.com/ultralytics/yolov5/issues/280>, [18.02.2022]



© Author(s) 2021. This work is distributed under <https://creativecommons.org/licenses/by-sa/4.0/>