Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2021) 29: 2101 – 2115

doi:10.3906/elk-2008-62

Visual object detection for autonomous transport vehicles in smart factories

Nazlıcan GENGEÇ[1], Onur EKER[2], Hakan ÇEVİKALP[2], Ahmet YAZICI[3], Hasan Serhan YAVUZ[2]*

[1]Department of Electrical-Electronics Engineering, Faculty of Engineering and Architecture, Kastamonu University, Kastamonu, Turkey

[2]Department of Electrical and Electronics Engineering, Faculty of Engineering and Architecture, Eskişehir Osmangazi University, Eskişehir, Turkey

[3]Department of Computer Engineering, Faculty of Engineering and Architecture, Eskişehir Osmangazi University, Eskişehir, Turkey

Received: 13.08.2020 • **Accepted/Published Online:** 10.12.2020 • **Final Version:** 26.07.2021

Abstract: Autonomous transport vehicles (ATVs) are one of the most substantial components of smart factories of Industry 4.0. They are primarily considered to transfer the goods or perform some certain navigation tasks in the factory with self driving. The recent developments on computer vision studies allow the vehicles to visually perceive the environment and the objects in the environment. There are numerous applications especially for smart traffic networks in outdoor environments but there is lack of application and databases for autonomous transport vehicles in indoor industrial environments. There exist some essential safety and direction signs in smart factories and these signs have an important place in safety issues. Therefore, the detection of these signs by ATVs is crucial. In this study, a visual dataset which includes important indoor safety signs to simulate a factory environment is created. The dataset has been used to train different fast-responding popular deep learning object detection methods: faster R-CNN, YOLOv3, YOLOv4, SSD, and RetinaNet. These methods can be executed in real time to enhance the visual understanding of the ATV, which, in turn, helps the agent to navigate in a safe and reliable state in smart factories. The trained network models were compared in terms of accuracy on our created dataset, and YOLOv4 achieved the best performance among all the tested methods.

Key words: Object detection, deep learning, autonomous transport vehicles, smart factories, safety sign detection

1. Introduction

Industry 4.0 aims to increase the degree of autonomy in smart factories [1]. Autonomous robots are one of the enabling technologies in this era and advanced perception systems are a precondition for the success of the robots. Autonomous navigation combines many technologies such as low-level control, high-level control, vehicle dynamics, sensing, mapping, and motion planning [2]. Autonomous robots use computer vision for better interactions with the environment during missions.

Transportation is one of the main tasks that are expected to be realized by autonomous transport vehicles (ATVs) in the factories. ATVs perceive the environment and can navigate in a large, dynamic, and unknown environment according to the mission. While carrying out the duties, ATVs should be able to perceive their surroundings and comply with the safety rules in order to avoid any possible accident in their travels. For a safe navigation, these vehicles need to perceive some important directional or safety signs around them and

*Correspondence: hsyavuz@ogu.edu.tr

they have to take the proper action according to the meaning of the sign, such as stopping at the pedestrian crossing, parking at the forklift parking area, or recognize other robots for possible robot–robot interactions. Therefore, visual object detection is critical for the success of ATVs. Advances in computer vision and deep learning methodologies make it possible for agents to easily detect the objects or signs in the environment. Visual-based object recognition applications can be used in this context to detect and sense important signs correctly in order to move safely.

In recent years, visual object detection and traffic sign recognition applications have made a great progress with the help of deep learning algorithms. While models that are trained with a small amount of images taken in a controlled environment may not give good results on the task, models that are trained using millions of images acquired in the wild give very satisfactory results in visual object detection problems. Millions of images mentioned here are usually obtained with large databases that are collected from images shared on the Internet. There are many image databases that are recorded in outdoor environments (see [3] and the references therein) and can be used for smart traffic networks but there are not many datasets including special factory signs for indoors.

In the literature, there are also some computer vision approaches that are used to detect some of the objects in the factories for various applications. For example, in [4], a system is developed to detect pallets in factories and warehouses. In [5], a visual perception system is developed for industrial robots. In [6], object detection is used to increase the safety in cyber-physical production systems. In [7], an image dataset is developed for benchmark of vision-based control of industrial robots. A detailed survey regarding machine vision techniques in industrial environments is given in [8]. As mentioned before, these are not mature as in the studies done for outdoor applications.

This work has been conducted to fulfill the need for studies regarding the visual object detection tasks in smart factories. To this end, we constructed a laboratory to simulate a smart factory environment including some important occupational safety and orientation signs and objects. Unlike conventional traffic signs in the outdoor applications, some of the signs used in the factory are not on a post on the side of the road, but sometimes on the ground floor and sometimes on the wall or glued with painted adhesive tapes. Additionally, the objects can be another robot, or some other platforms that are in the navigation or mission environment. Therefore, a proper computer vision methodology should be used in order to detect these signs and objects successfully. In this study, we experimented to visually detect the safety/orientation signs and some common objects seen in factories by training the state-of-the-art real-time working CNN methods: faster R-CNN [9], YOLOv3 [10], YOLOv4 [11], SSD [12], and RetinaNet [13], and they are compared in terms of accuracy. The rest of the paper is given as follows: the next section summarizes the related methods. We give details about the detection methods used in the study in Section 3. The experimental results are given in Section 4. Finally, conclusions are presented in the last section.

2. Related methods

2.1. Traditional methods used in sign recognition and object detection

Conventional methods used for sign recognition generally focus on extracting features of signs. These methods usually use the information related to color and shape of the signs. Changing lighting conditions affect the RGB color space; therefore, images are often converted to different color spaces like hue saturation value [14]. The structures of the traffic signs in certain shapes enable the use of morphology. The edge information derived by various algorithms is used to find the contour lines of the traffic signs. In [15], an ellipse detection method is

proposed to identify circular traffic signs. In [16], the Hough-like algorithm is proposed to detect traffic signs with circular and triangular shapes. The authors in [17] have designed templates for each traffic sign class to match the signs during testing time. Another shape-based study used gradient density and orientation in images to recognize smooth polygonal and circular traffic signs [18].

In addition to the characteristics such as color and shape, some feature descriptors have also been used for traffic sign recognition. In [19], the histogram of oriented gradients (HOGs) [20] and the support vector machine (SVM) classifiers are used in the identification of traffic signs. They used the benchmark dataset of German traffic sign detection. In [21], an algorithm based on color and shape, which detects traffic signs from videos, was developed. The features of the signs were extracted using color and morphology, and the classification stage was carried out with neural network classifier.

Feature extraction and classification are two important components for traditional visual object detection methods. Features are usually extracted using descriptors such as HOG, scale invariant feature transform, and local binary patterns [22–24]. Final decision of whether a window of interest contains a background or an object instance is given by the classifiers. For this purpose, various types of classifiers such as SVMs, boosting cascades, polyhedral conic classifiers, and binary decision trees are used [25–28]. Classifier training is done using tagged images belonging to the positive class (object) and negative class (background). After the training phase, the learned classifiers are applied to the windows of different sizes that are systematically selected through the image by using sliding window method as in [29] or using region proposal methods [26].

2.2. Recent methods used in sign recognition and object detection

In recent years, the convolutional neural networks (CNNs), also known as the deep learning methods, have achieved considerable success in object recognition. One of the most important reasons for this success is that CNNs serve an integrated structure that enables end-to-end process including simultaneous learning of feature extraction and classification. In [30], the authors proposed a deep CNN model that achieved a great success on a large dataset, ILSVRC2013. Since then, there has been an increasing interest in deep learning studies due to results with higher success. In [31], the authors used a deep CNN architecture in the recognition of Chinese traffic signs dataset. The authors in [32] proposed region-proposal CNN (R-CNN) method for the detection of objects. In the R-CNN method, approximately 2000 candidate fields are generated using a selective search algorithm, and then CNN features of the selected regions were extracted with a CNN architecture. Finally, the features that are extracted by CNN are used as inputs to SVM classifier to decide whether the current region corresponds to an object or background. A faster version of this method proposed in [9], where the separate stages of feature extraction and classification for R-CNN are combined with a single architecture.

Faster R-CNN has also been used in the traffic sign recognition area [33]. Autonomous cars or robots require object detection systems that run faster. You look only once (YOLO) [34] which uses the Darknet library[1] and single-shot MultiBox detector (SSD) methods stand out with their accuracy and speed for real-time applications. The revised versions of YOLO detector are given as YOLOv2 [35], YOLOv3 [10], and YOLOv4 [11], respectively. Initially, YOLO had difficulty finding small objects in images. In addition, YOLO produced relatively rough properties for generalizing objects. SSD is proposed to avoid these problems [12]. When a specific property map is given instead of fixed grids as used in the YOLO, the SSD uses a set of anchor boxes with various scales and aspect ratios. To handle objects of different sizes, the network combines estimates

[1]Darknet: Open source neural networks in C (2013-2016) [online]. Website http://pjreddie.com/darknet [accessed July 2020].

from multiple featured maps of different resolutions. Another successful object detection method is RetinaNet [13] that uses focal loss for improving average precision (AP) in single-stage object detectors. As a result, these successful CNN-based detectors are widely used in different applications. For example, [36] used YOLOv2 for Chinese traffic sign detection.

3. Method

We used the following CNN-based object detection methods in our study: faster R-CNN, YOLOv3, YOLOv4, SSD, and RetinaNet. Among these tested methods, faster R-CNN is regarded as a two-stage method, whereas the remaining detectors are one-stage detectors. In the following subsections, we briefly describe each mentioned method.

3.1. Faster R-CNN detector

Faster R-CNN [9] is a two-stage object detection network comprised of a region proposal network (RPN) and a fast R-CNN detector. RPN is a fully convolutional neural network that takes an image as input to propose rectangular regions and objectness score for each of these regions. Since RPN shares the image convolution properties with the detector network, it increases the speed of region proposing. RPN architecture is given in Figure 1.

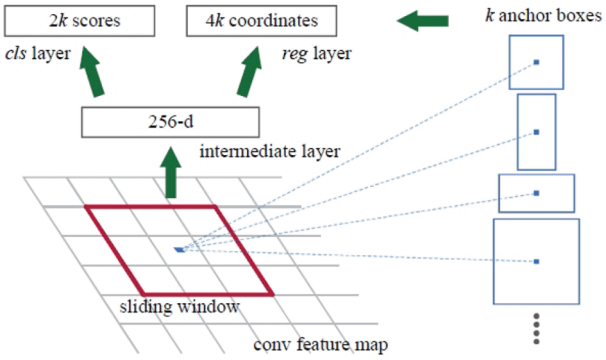


Figure 1. Region proposal network in faster R-CNN [9]. The network produces fixed-length feature vectors, and they are fed to classification and regression layers.

Also different from its predecessor R-CNN methods, faster R-CNN uses multiscaled anchor boxes. This both simplifies and speeds up the process of region proposal generations with different aspect ratios. The network slides over the feature map with a fixed window size of 3×3, and generates k anchor boxes. At each location, a total number of 9 anchor boxes are obtained with 3 aspect ratios and 3 different scales. Every sliding window yields a feature vector which is fed into fully connected bounding box regression and classification layers. The loss function used in faster R-CNN is similar to fast R-CNN, and it is given as

$$L\left(p_i, t_i\right) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*), \tag{1}$$

where L_{cls} is a binary log loss, L_{reg} is smoothed L1 loss, p_i is the probability of the anchor i being an object, and p_i^* is the ground truth label that becomes 1 if anchor is positive, and 0 otherwise. The terms t_i, t_i^* represent the predicted bounding box and ground truth bounding box, respectively. Regression and classification losses

are normalized with N_{cls} (size of mini-batch) and N_{reg} (number of anchors). The method randomly samples 128 positive and 128 negative examples without using a hard-example mining strategy. Faster R-CNN achieves high accuracy and speed, but the two-stage detector faces class imbalance problem and cannot detect small objects accurately.

3.2. YOLOv3 detector

YOLOv3 [10] is an extremely fast one-stage object detection network and made several improvements over previous YOLO [34, 35] versions. YOLO detector divides the image into S×S grids. A fixed number of bounding boxes is predicted for each grid cell, and each bounding box consists of five parameters $(tx, ty, tw, th,$ $box\_confidence\_score)$. Instead of directly predicting locations, YOLOv3 predicts the offsets to the anchor boxes (tx, ty, tw, th) and box confidence score which represents the class probabilities to estimate the object class. This strategy results in a more robust training procedure. Output tensor dimension of the network is $N \times N \times (3 \cdot (4 + 1 + C))$. Here, the number of the grid cells is $N \times N$, 3 is the number of detections at three scales, $4 + 1$ is the bounding box offsets and objectness score, and C is the number of classes. In the loss function of YOLOv3, binary cross-entropy loss is chosen to model the data better in complex datasets using a multilabel approach for class predictions. Loss function used in YOLOv3 is given as,

$$\begin{aligned}\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}\left[(x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2\right]+\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}\left[\left(\sqrt{w_i}-\sqrt{\hat{w}_i}\right)^2+\left(\sqrt{h_i}-\sqrt{\hat{h}_i}\right)^2\right]+\\ \sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}\left(C_i-\hat{C}_i\right)^2+\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{noobj}\left(C_i-\hat{C}_i\right)^2+\sum_{i=0}^{S^2}1_{i}^{obj}\sum_{c\in classes}\left(p_i(c)-\hat{p}_i(c)\right)^2,\end{aligned} \tag{2}$$

where the first term penalizes incorrect cell center estimations, the second term penalizes width/height of the bounding box where error in smaller bounding boxes are penalized compared to bigger bounding boxes due to the square root operation. The third term tries to maximize the confidence score if an object exists, the fourth term tries to minimize the confidence score if there is not an object, and the fifth term is the classification loss. Instead of using Darknet-19, YOLOv3 network uses the Darknet-53 structure which is a more robust and a deeper architecture. It consists of 53 convolutional layers that use skip connections similar to ResNet [37] structure. The method adopts k-means clustering on the training bounding boxes to predict the scales and aspect ratios to get good priors for anchor boxes for a better learning. YOLOv3 makes prediction in three different feature maps to make use of better semantic information of layers at the end of the network and fine-grained features of earlier layers. Predictions at three different feature map increase the performance of detecting smaller objects in the image.

3.3. YOLOv4 detector

YOLOv4 [11] is a recently proposed one-stage object detection method which makes some important improvements over previous YOLO architecture, YOLOv3. In YOLOv4, the authors choose to use a novel architecture called CPSDarknet53 as backbone network. Cross stage partial (CSP) architecture [38], which is a similar architecture to DenseNet [39], enhances the learning capacity of the network by dividing the input feature maps into two parts. One part passes through the dense layer, while the other part is concatenated to the feature map that is produced from the dense layer. CSPDarknet53 consists of 29 convolutional layers. Spatial pyramid

pooling (SPP) [40] block is adapted to CSPDarknet53. Adding SPP block increases the receptive field and pools the most relevant context information from feature maps, and this block does not reduce the operation speed of the detector. Instead of using FPN in the neck module as in YOLOv3, PANet [41] is used to aggregate parameters from different backbone layers. PANet augments the top-down path in FPN by adding a bottom-up path. Moreover, it implements adaptive feature pooling by fusing features from all stages of the pyramid. The authors used YOLOv3 head as the head component in the architecture. Aside from these improvements, the authors introduce new data augmentation methods such as mosaic and self-adversarial training. YOLOv4 significantly increases the performance of the previous YOLOv3 architecture in terms of accuracy and speed.

3.4. Single-shot MultiBox detector (SSD)

SSD [12] is a single-stage detector which directly predicts object class scores and bounding box offsets similar to YOLO. It uses a VGG backend, and the network concatenates predictions from multiple feature maps to handle objects with different scales and aspect ratios. The SSD adds extra convolutional layers at the end of the VGG backbone network to make predictions in multiple resolutions. These layers predict boxes and their corresponding objectness score with different aspect ratios and scales. SSD limits the output bounding boxes to increase the network performances. In the network, one scale per feature layer is used, and this scale parameter is decided by using the equation given below,

$$s_k = s_{min} + \frac{s_{max} - s_{min}}{m - 1}(k - 1), \quad k \in [1,\ m] \tag{3}$$

In Equation (3), s_{min} is 0.2, s_{max} is 0.9, and m is the number of feature layers. For each scale, a total of six anchor boxes are used and they have aspect ratios of 1, 2, 3, 1/2, 1/3. The default box is given as $s'_k = \sqrt{s_k s_{k+1}}$. Architecture of SSD is shown in Figure 2.

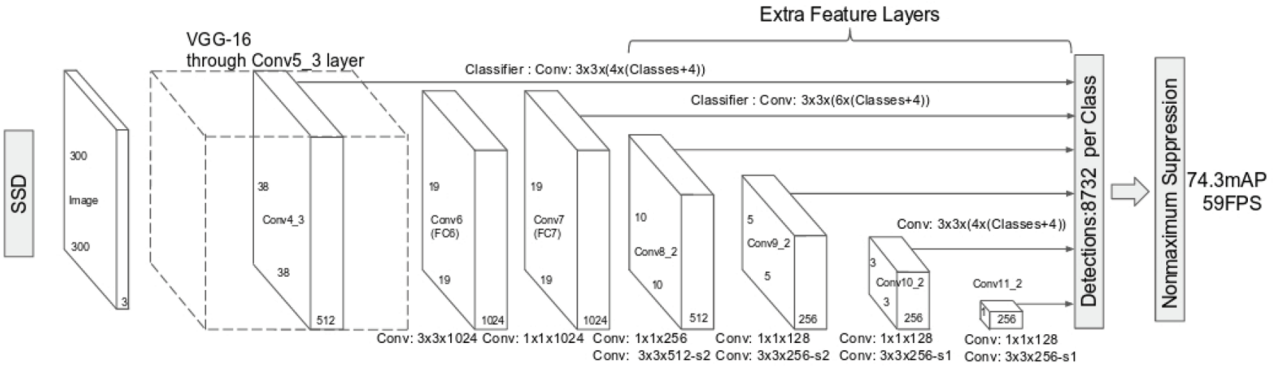


Figure 2. Architecture of SSD [12]. Predictions from several feature maps fused at the end of the network to make final detections.

SSD uses smooth L1-Norm to calculate the location loss and categorical cross-entropy to calculate confidence loss. Final loss function can be written as in equation below:

$$L\,(x,\ c,\ l,\ g) = \frac{1}{N}(L_{conf}\,(x, c) + \alpha L_{loc}\,(x, l,\ g)), \tag{4}$$

where l is the predicted bounding box, g is the ground truth box, c is the confidence score, N is the number of

positive matches, and α is the weight for the localization loss. Moreover, the method integrates hard negative mining to address class imbalance problem. SSD chooses the best samples that have the highest confidence loss and makes the ratio between the negative and positive samples at most 3-to-1 for each default box. This strategy leads to a more robust training and faster convergence.

3.5. RetinaNet

RetinaNet [13] is another one-stage object detection network that uses feature pyramid networks (FPNs) on top of ResNet feature extractor. FPN is a top-down architecture that builds semantically strong multiscale feature maps and it is fast to compute. Architecture of FPN is given in Figure 3.

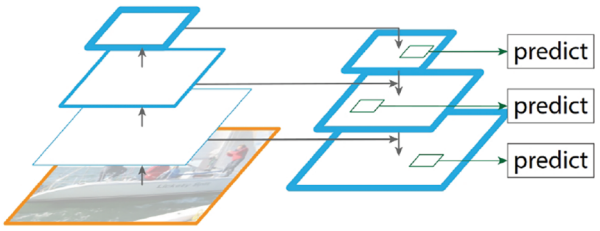


Figure 3. The structre of FPN [13]. FPN combines both single-scale features and pyramidal feature hierarchy.

A total of nine anchor boxes with the aspect ratios 1:2, 1:1, 2:1 and three scales 2^0, $2^{1/3}$, $2^{2/3}$ are extracted from each pyramid level of FPN (on pyramid levels $P3$ to $P7$). Predicted bounding boxes and classification targets are fed into box regression and classification subnet which are fully convolutional networks, respectively. Classification subnet estimates the probability of becoming an object and box regression subnet determines the offsets to ground truth object (if exists) at each location for each anchor box and object class. In one-stage object detectors, the issue of foreground-background class imbalance reduces the network performance. RetinaNet aims to solve the class imbalance problem using the focal loss. The focal loss reduces the loss obtained by well-classified or easy examples. Thus, focal loss concentrates on more difficult samples during training rather than overwhelming the detector with well-classified or easy samples. The focal loss function is given as

$$FL\left(p_t\right) = -\alpha_t(1-p_t)^{\gamma}\log\left(p_t\right), \tag{5}$$

where γ is the focusing, and α is the balancing parameter. Focusing parameter down-weights the well-classified examples and gives more weights to hard examples. Balancing parameter, on the other hand, is the inverse of the number of examples in a class. This results in a smaller weight if the number of examples is large in a specific class. RetinaNet takes advantage of the fast speed of one-stage detectors and overcomes the problem of positive-negative class imbalance.

4. Experiments

4.1. ESOGU intelligent factory image dataset

Eskişehir Osmangazi University Intelligent Factory and Robotics Laboratory (ESOGU- IFARLAB, https://ifarlab.ogu.edu.tr/) (see Figure 4) is designed to test ATVs. The laboratory environment includes many of the standard typical objects in the factory. One of the test areas for the ATVs is correct identification of the visual objects in the environment. In this way, ATVs could realize high-level autonomy during missions.

Figure 4. Eskişehir Osmangazi University Intelligent Factory and Robotics Laboratory.

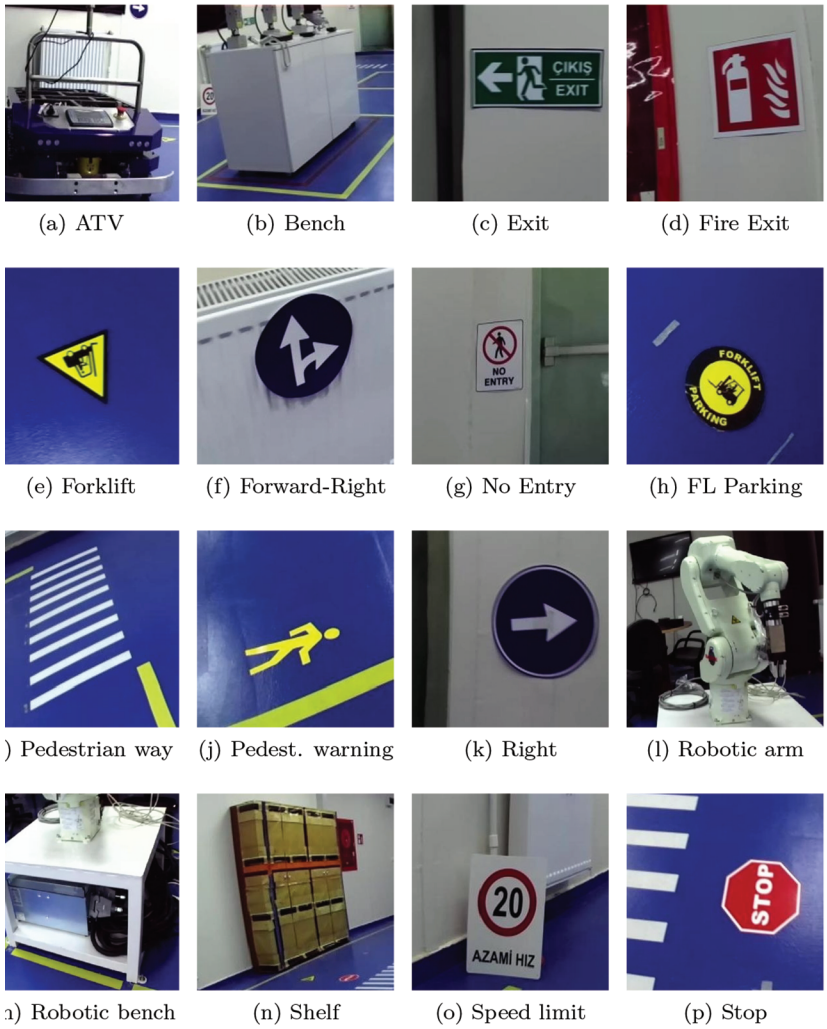
In this study, we prepared ESOGU Intelligent Factory Image dataset for visual object detection applications of ATVs. The dataset includes the images of objects and signs seen in Figure 5. The most widely used sixteen objects and signs that are informative for both the drivers of vehicles and the human employees for various behaviors are selected. ATVs or other autonomous robots should also use these for various tasks or regulations.

The tested images are collected by using a ZED stereo camera mounted on the ATV shown in Figure 6. It is a differential drive autonomous robot to carry goods in factory environments. It has an RGB-D camera, sonars, laser, bumper, infrared, IMU, ultrawide band, and encoder sensors. The electronic layer consists of Jetson TX2 for visual computations and an embedded computer for intelligent control. Robot Operating System (ROS) is used for intercommunications of inner system and overall control of the vehicle.

The images were obtained under four different illumination conditions. In the first case, the laboratory environment was illuminated only by day light. In the second scenario, the environment was illuminated only by fluorescent light. In the third scenario, the environment was illuminated only by led light. In the last scenario, the environment was illuminated by using fluorescent light and led light together. Video recordings were also taken from the test environments for various orientations of the camera with different sign/object layout scenarios. The video frames have 2560×720 resolution. Then, images were obtained from the recorded videos. This is followed by the manual annotation of objects and signs in the collected images. To this end, 2 people manually annotated the selected objects and signs approximately in 3 months. After all the preprocessing steps were completed, a total of 7570 training and 800 test images data were created.

4.2. Experimental evaluation

We compared faster R-CNN, YOLOv3, YOLOv4, SSD, and RetinaNet object detection methods on our sign dataset we created. We used the classical PASCAL-VOC metrics to evaluate the accuracy. Average precision (AP) is the area under the precision-recall curve. If the bounding box R and the ground-truth annotation's box Q overlaps by more than a certain percentage where the overlap is computed as $\frac{area|Q \cap R|}{area|Q \cup R|}$; this overlap is considered to correspond to a positive sample. The overlap threshold was set to 50% as usual. The precision



(a) ATV (b) Bench (c) Exit (d) Fire Exit

(e) Forklift (f) Forward-Right (g) No Entry (h) FL Parking

) Pedestrian way (j) Pedest. warning (k) Right (l) Robotic arm

ı) Robotic bench (n) Shelf (o) Speed limit (p) Stop

Figure 5. Objects and signs used in ESOGU Intelligent Factory Image dataset.

and recall measures are computed as follows:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{6}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{7}$$

Before testing the detectors on test dataset, we trained all the methods on the same training data as described below.

4.2.1. Training stage

Descriptions of the objects in the dataset were first created to train the detectors by making manual annotations. This was realized by constructing a ground-truth label file for each image in the dataset. Label files were

Figure 6. The ATV used for collecting images.

converted to the appropriate format for each model. For example, R-CNN needs XML format whereas YOLOv3-4 uses txt annotation files. To train the models, we used 7570 training data coming from 16 classes. Each model was trained for approximately 2 days using a GPU server that includes Nvidia Quadro P5000 GPUs. The training of all the methods was completed in about 9 days in total.

Training and test folders and label files in XML format were prepared to train the faster R-CNN detector model. We used Resnet50 network architecture by fine-tuning the network from a model trained on MS-COCO dataset. The training was carried out using the available code in [9]. The training of faster R-CNN was optimized by using stochastic gradient descent with momentum value that was set to 0.9. The learning rate was set to 0.0005 and we trained the network for 100K steps.

To train the YOLOv3 model, we edited the configuration file of Darknet library based on training images and annotation files. We fine-tuned the network from the YOLOv3 model pretrained on MS-COCO. We made an optimization using stochastic gradient descent with a momentum value of 0.9. The learning rate and the weight decay term were set to 0.001 and 0.005, respectively. We applied k-means clustering on the training bounding boxes of the dataset to determine the optimal scales and aspect ratios of the anchor boxes.

Similar to YOLOv3 training steps, we fine-tuned the network from the YOLOv4 model pretrained on MS-COCO dataset. The momentum and weight decay values were set to 0.9 and 0.005, respectively. We set the learning rate to 0.1 initially and used a polynomial decay learning rate scheduling. We applied k-means clustering on the training bounding boxes of the dataset to determine the optimal scales and aspect ratios of the anchor boxes as in YOLOv3 training stage.

Training and test folders and label files in XML format were prepared to train the SSD model with the dataset we prepared. The SSD MobileNet v1 network model trained on the MS-COCO dataset was used to fine-tune the network. During learning, we set the batch size to 64 and we employed Adam optimizer with a learning rate of 0.0001.

For training with the RetinaNet model, annotation files were prepared in CSV format. The model was trained with Keras implementation of RetinaNet [13]. We utilized the ResNet-50 as backbone architecture fine-tuned from a model trained on MS-COCO dataset. The learning rate was set to 0.001 and the network was trained for 50 epochs.

4.2.2. Testing stage

After training all the detectors on the training data, we tested the trained detectors on the test set which includes 800 images. The resulting precision-recall curves for the best and the worst detection models are given in Figure 7. The mAP scores of the tested detectors obtained from the curves are given in Table. As seen in Table, the best accuracy is obtained by YOLOv4 followed by YOLOv3 and faster R-CNN methods. SSD is the worst performing method. YOLOv4, YOLOv3, and faster R-CNN detectors significantly outperform SSD and RetinaNet in terms of accuracy. Among all tested 16 classes, YOLOv4 achieves the best accuracies for 9 classes, YOLOv3 obtains the best scores for 3 classes, and faster-RCNN achieves the best mAP scores for the remaining 4 object classes.

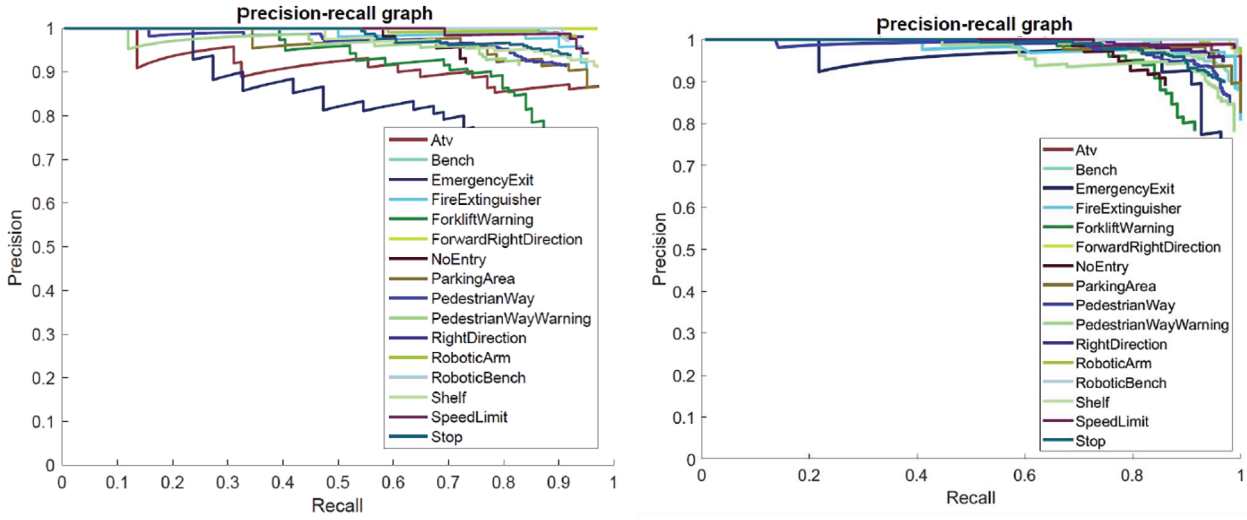


Figure 7. Precision recall curves for the worst and best detectors: SSD (on the left) and YOLOv4 (on the right).

The results show that the robotic arm, ATV, fire extinguisher, speed limit sign, and forward-right direction categories are the easiest ones to detect, whereas the emergency exit, forklift warning, and pedestrian way categories are the most difficult ones. Overall, the accuracy levels are encouraging, and they indicate that the ATVs can autonomously operate in smart factory environments without human intervening.

We also show some examples of outputs from the tested detectors in Figure 8. The examples in the first row show the successful detections whereas some failure cases are given in the second row. As seen in the figure, the detectors can successfully detect the signs and objects even though some of them are on the floor or on the side wall. When we examine the failure cases, we see that the detectors mostly fail with occluded or cropped objects. As seen at the bottom left of the figure, most detectors fail to detect pedestrian way warning and shelf classes since only some part of the objects are visible. The right direction is slightly occluded with a larger bench object in the example given at the bottom right of the figure. As a result, some of the detectors also missed this sign. The detectors also fail to detect small far away signs as illustrated in the bottom center example. As seen in this image, all detectors missed the far away right direction sign.

5. Conclusion

The Industry 4.0 factories have machines powered by wireless connectivity and sensors that can make intelligent productions by making decisions on their own. Autonomously navigating carrier vehicles are used in the

Table . Average precision scores (mAP) (%) of the tested detectors. The best scores are given with bold characters.

Object classes	Faster R-CNN	YOLOv3	YOLOv4	SSD	RetinaNet
ATV	95.95	95.95	**98.31**	87.79	94.58
Bench	91.91	91.37	**94.76**	90.92	93.17
Emergency exit	75.03	86.71	**90.68**	68.55	85.44
Fire extinguisher	96.01	**97.59**	97.41	93.10	94.40
Forklift warning	81.53	85.58	**88.65**	82.01	78.77
Forward right direction	97.89	94.95	**98.95**	95.96	95.82
No entry	**88.92**	86.99	84.09	71.49	88.78
Parking area	**97.42**	95.06	97.39	92.02	81.73
Pedestrian way	87.94	91.99	**94.76**	88.49	74.59
Pedestrian way warning	91.53	91.52	**95.30**	79.69	83.63
Right direction	**97.46**	96.33	94.48	93.91	90.71
Robotic arm	95.71	98.40	**98.85**	92.26	98.36
Robotic bench	96.11	**97.97**	92.62	91.24	94.12
Shelf	96.93	**98.04**	92.74	94.31	95.81
Speed limit sign	**97.65**	96.45	97.38	93.85	96.59
Stop	94.64	94.56	**95.31**	90.61	82.46
Average	92.66	93.72	**94.48**	87.91	89.31

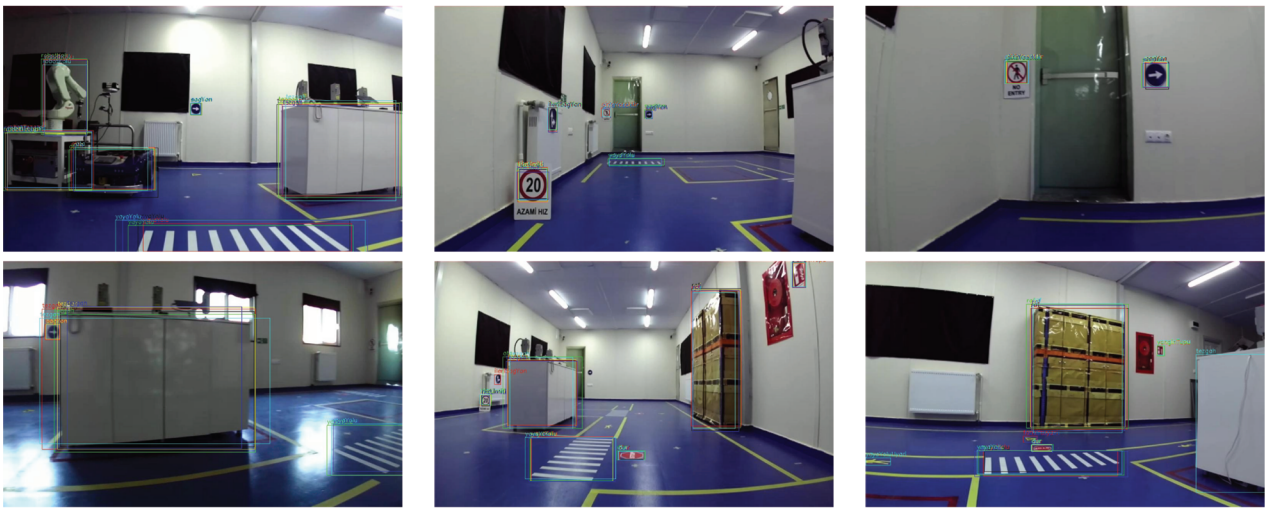


Figure 8. Some examples of detections of the tested methods. The first row shows the successful detections whereas the second row shows some failure cases.

transportation of products by these smart machines. These vehicles, which can travel on their own, need to sense the visual objects in the environment and also be able to move without risking work safety. In this study, we performed visual object detection in indoor factory environment. Considering the lack of important indoor safety signs and object datasets for smart factories, we collected a new dataset of 16 common safety signs and objects encountered in factories by using a stereo camera mounted on an ATV. Then, the dataset was used

to train different popular deep learning object detection methods: Faster R-CNN, YOLOv3, YOLOv4, SSD, and RetinaNet. The trained network models were compared in terms of accuracy on our created test dataset. YOLOv4 achieved the best performance closely followed by YOLOv3 and faster R-CNN whereas both SSD and RetinaNet yielded significantly lower accuracies compared to these good performing methods. Experimental results showed that the detector failures occur only when occluded and cropped objects or far away small signs exist in the scene. We would also like to emphasize that the correct recognition rates obtained in this study can be increased by using more diverse training data in real settings.

Acknowledgment

This work was funded in part by the Scientific and Technological Research Council of Turkey (TUBİTAK) under Grant number EEEAG-116E731.

References

[1] Muhuri PK, Shukla AK, Abraham A. Industry 4.0: A bibliometric analysis and detailed overview. Engineering Applications of Artificial Intelligence 2019; 78: 218-235. doi: 10.1016/j.engappai.2018.11.007

[2] Sales DO, Correa DO, Fernandes LC, Wolf DF, Osorio FS. Adaptive finite state machine based visual autonomous navigation system. Engineering Applications of Artificial Intelligence 2014; 29: 152-162. doi: 10.1016/j.engappai.2013.12.006

[3] Wali SB, Abdullah MA, Hannan MA, Hussain A, Samad S et al. Vision-based traffic sign detection and recognition systems: current trends and challenges. Sensors 2019; 19 (9): 2093. doi: 10.3390/s19092093

[4] Syu JL, Li HT, Chiang JS, Hsia CH, Wu PH et al. A computer vision assisted system for autonomous forklift vehicles in real factory environment. Multimedia Tools and Applications 2016; 76 (18): 18387-18407. doi: 10.1007/s11042-016-4123-6

[5] Castelli F, Michieletto S, Ghidoni S, Pagello E. A machine learning-based visual servoing approach for fast robot control in industrial setting. International Journal of Advanced Robotic Systems 2017; 14 (6): 172988141773888. doi: 10.1177/1729881417738884

[6] Islam SOB, Lughmani WA, Qureshi WS, Khalid A, Mariscal MA et al. Exploiting visual cues for safe and flexible cyber-physical production systems. Advances in Mechanical Engineering 2019; 11 (12): 168781401989722. doi: 10.1177/1687814019897228

[7] Luo C, Yu L, Yang E, Zhou H, Ren P. A benchmark image dataset for industrial tools. Pattern Recognition Letters 2019; 125: 341-348. doi: 10.1016/j.patrec.2019.05.011

[8] Perez L, Rodríguez I, Rodríguez N, Usamentiaga R, García D. Robot guidance using machine vision techniques in industrial environments: a comparative review. Sensors 2016; 16 (3): 335. doi: 10.3390/s16030335

[9] Ren S, He K, Girshick R, Sun J. Faster R-CNN: Towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence 2017; 39 (6): 1137-1149. doi: 10.1109/tpami.2016.2577031

[10] Redmon J, Farhadi A. Yolov3: An incremental improvement, arXiv preprint arXiv:1804.02767, 2018.

[11] Bochkovskiy A, Wang CY, Liao HYM. YOLOv4: Optimal speed and accuracy of object detection, arXiv preprint arXiv:2004.10934, 2020.

[12] Liu W, Anguelov D, Erhan D, Szegedy C, Reed S et al. SSD: Single shot multiBox detector. In: European Conference on Computer Vision (ECCV 2016), Amsterdam, the Netherlands; 2016. pp. 21-37. doi: 10.1007/978-3-319-46448-0_2

[13] Lin TY, Goyal P, Girshick R, He K, Dollar P. Focal loss for dense object detection. In: 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy; 2017. doi: 10.1109/iccv.2017.324

[14] Wang G, Ren G, Quan T. A traffic sign detection method with high accuracy and efficiency. In: Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering; 2013.

[15] Wang G, Ren G, Wu Z, Zhao Y, Jiang L. A fast and robust ellipse-detection method based on sorted merging. The Scientific World Journal 2014; 1-15. doi: 10.1155/2014/481312

[16] Houben S. A single target voting scheme for traffic sign detection. In: 2011 IEEE Intelligent Vehicles Symposium (IV), Baden, Germany, 2011. pp. 124-129.

[17] Liang M, Yuan M, Hu X, Li J, Liu H. Traffic sign detection by ROI extraction and histogram features-based recognition. In: the 2013 international joint conference on neural networks (IJCNN), Dallas, TX, USA; 2013. pp. 1-8.

[18] Loy G, Barnes N. Fast shape-based road sign detection for a driver assistance system. In: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566), Sendai, Japan; 2004. pp. 70-75.

[19] Wang G, Ren G, Wu Z, Zhao Y, Jiang L. A robust, coarse-to-fine traffic sign detection method. In: the 2013 international joint conference on neural networks (IJCNN), Dallas, Texas, USA; 2013. pp. 1-5.

[20] Dalal N, Triggs B. Histograms of oriented gradients for human detection. In: 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05), San Diego, CA, USA; 2005. pp. 886-893.

[21] Supreeth HSG, Patil CM. An approach towards efficient detection and recognition of traffic signs in videos using neural networks. In: 2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), Chennai, India; 2016. pp. 456-459.

[22] Cevikalp H, Triggs B. Efficient object detection using cascades of nearest convex model classifiers. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA; 2012. pp. 3138-3145.

[23] Hussain SU, Triggs W. Feature sets and dimensionality reduction for visual object detection. In: Proceedings of the British Machine Vision Conference 2010, Aberystwyth, Wales, UK; 2010. doi:10.5244/c.24.112

[24] Lowe DG. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 2004; 60 (2): 91-110. doi: 10.1023/b:visi.0000029664.99615.94

[25] Felzenszwalb P, McAllester D, Ramanan D. A discriminatively trained, multiscale, deformable part model. In: 2008 IEEE conference on computer vision and pattern recognition, Anchorage, Alaska, USA; 2008. pp. 1-8.

[26] Uijlings JRR, Van de Sande KEA, Gevers T, Smeulders AWM. Selective search for object recognition. International Journal of Computer Vision 2013; 104 (2): 154-171. doi: 10.1007/s11263-013-0620-5

[27] Cevikalp H, Triggs B. Polyhedral conic classifiers for visual object detection and classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Venice, Italy; 2017. pp. 261-269. doi: 10.1109/cvpr.2017.438

[28] Cevikalp H, Saglamlar H. Polyhedral Conic Classifiers for Computer Vision Applications and Open Set Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 2020; 1-1. doi: 10.1109/tpami.2019.2934455

[29] Cevikalp H, Triggs B. Visual object detection using cascades of binary and one-class classifiers. International Journal of Computer Vision 2017; 123 (3): 334-349. doi: 10.1007/s11263-016-0986-2

[30] Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Communications of the ACM 2017; 60 (6): 84-90. doi: 10.1145/3065386

[31] Qian R, Zhang B, Yue Y, Wang Z, Coenen F. Robust chinese traffic sign detection and recognition with deep convolutional neural network. In: 2015 11th International Conference on Natural Computation (ICNC), Zhangjiajie, China; 2015. pp. 791-796.

[32] Girshick R, Donahue J, Darrell T, Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA; 2014. pp. 580-587.

[33] Zuo Z, Yu K, Zhou Q, Wang X, Li T. Traffic signs detection based on faster R-CNN. In: 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW), Atlanta, GA, USA; 2017. pp. 286-288.

[34] Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, Las Vegas, NV, USA; 2016. pp. 779-788.

[35] Redmon J, Farhadi A. YOLO 9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition, Venice, Italy; 2017. pp. 7263-7271.

[36] Zhang J, Huang M, Jin X, Li X. A real-time Chinese traffic sign detection algorithm based on modified YOLOv2. Algorithms 2017; 10 (4): 127. doi: 10.3390/a10040127

[37] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA; 2016. pp. 770-778.

[38] Wang CY, Mark LHY, Wu YH, Chen PY, Hsieh JW et al. CSPNet: A new backbone that can enhance learning capability of CNN. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Virtual; 2020. pp. 390-391.

[39] Huang G, Liu Z, Van Der Maaten L, Weinberger KQ. Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, Venice, Italy; 2017. pp. 4700-4708. doi: 10.1109/CVPR.2017.243

[40] He K, Zhang X, Ren S, Sun J. Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 2015; 37 (9): 1904-1916. doi: 10.1109/tpami.2015.2389824

[41] Liu S, Qi L, Qin H, Shi J, Jia J. Path aggregation network for instance segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, Salt Lake City, UT, USA; 2018. pp. 8759-8768. doi: 10.1109/CVPR.2018.00913