

Received 1 December 2021, accepted 13 December 2021, date of publication 22 December 2021, date of current version 9 March 2023.

Digital Object Identifier 10.1109/ACCESS.2021.3137638

# T-YOLO: Tiny Vehicle Detection Based on YOLO and Multi-Scale Convolutional Neural Networks

DANIEL PADILLA CARRASCO<sup>1,2</sup>, HATEM A. RASHWAN<sup>1</sup>,  
MIGUEL ÁNGEL GARCÍA<sup>3</sup>, AND DOMÈNEC PUIG<sup>1</sup>

<sup>1</sup>Department of Computer Engineering and Mathematics, Universitat Rovira i Virgili, 43003 Tarragona, Spain

<sup>2</sup>Quercus Technologies, 43203 Reus, Spain

<sup>3</sup>Department of Electronic and Communications Technology, Universidad Autónoma de Madrid, 28049 Madrid, Spain

Corresponding author: Daniel Padilla Carrasco (daniel.padilla@estudiants.urv.cat)

This research was done thanks to AGAUR funding on Industrial Doctorate modality (018 DI 054) with Universitat Rovira i Virgili and Quercus Technologies collaboration.

**ABSTRACT** To solve real-life problems for different smart city applications, using deep Neural Network, such as parking occupancy detection, requires fine-tuning of these networks. For large parking, it is desirable to use a cenital-plane camera located at a high distance that allows the monitoring of the entire parking space or a large parking area with only one camera. Today's most popular object detection models, such as YOLO, achieve good precision scores at real-time speed. However, if we use our own data different from that of the general-purpose datasets, such as COCO and ImageNet, we have a large margin for improvisation. In this paper, we propose a modified, yet lightweight, deep object detection model based on the YOLO-v5 architecture. The proposed model can detect large, small, and tiny objects. Specifically, we propose the use of a multi-scale mechanism to learn deep discriminative feature representations at different scales and automatically determine the most suitable scales for detecting objects in a scene (i.e., in our case vehicles). The proposed multi-scale module reduces the number of trainable parameters compared to the original YOLO-v5 architecture. The experimental results also demonstrate that precision is improved by a large margin. In fact, as shown in the experiments, the results show a small reduction from 7.28 million parameters of the YOLO-v5-S profile to 7.26 million parameters in our model. In addition, we reduced the detection speed by inferring 30 fps compared to the YOLO-v5-L/X profiles. In addition, the tiny vehicle detection performance was significantly improved by 33% compared to the YOLO-v5-X profile.

**INDEX TERMS** Convolutional neural networks, tiny objects, smart parking.

## I. INTRODUCTION

The ever-increasing city population has reached a point where the management of city resources has become a critical and important problem for large cities. In fact, to address the management of resources, the concept of a smart city has been coined for city resource data exploitation [1]. One of the biggest challenges of large cities is the improvement and enhancement of driving experience [2]: traffic control, surveillance, or parking guidance, which can help improve the mobility experience in these cities. Following this objective, one of the most time-consuming processes for drivers is finding a parking spot. A driver will travel extra kilometres per year to find an available slot, which not only has a direct impact on the driver's time consumption but also on environmental pollution [3]. In addition, in large parking, where the

most desired spots [4] are usually concentrated, it will yield inefficient traffic, which will again contribute to worsening the problem.

This problem was managed using sensors in each parking spot to detect the spot occupancy. However, magnetometer-based parking sensors rapidly decrease the battery life with increasing accuracy requirements. In addition, modern vehicles often do not have ferromagnetic parts. Thus, the progress in computer vision and deep learning makes it possible to use smart cameras to control several parking spots and provide much cheaper solutions for the parking occupancy problem. In fact, the problem of finding available parking spots given an image has been addressed in several studies [5]–[8]. However, these techniques cannot be generalized, and even the adaptation of a specific solution to a different parking lot is not possible in many cases. Thus, vacant parking space detection based only on visual information remains a challenge for researchers. Most of these solutions use a simpler

The associate editor coordinating the review of this manuscript and approving it for publication was Chao Yang<sup>1</sup>.

approach to determine the occupancy of each parking spot by detecting the availability of a parking spot through spot classification [9]–[11] instead of localizing vehicles using object detection techniques and determining if their positions are over parking spots [5], [7], [8]. Although these approaches can achieve good precision scores, they lack the possibility of extracting potential information about cars (i.e., road congestion, human interactions with cars, single car occupying two spots, etc.). Only by using a vehicle detection approach can we obtain information that cannot be obtained using a parking spot classification approach.

To obtain more reliable vacant parking space detection, the proposed classification solutions can be adapted to different vehicles (e.g., cars, motorbikes, etc.). In addition, they should be capable of determining the locations of these vehicles in parking, as well as spot occupancy. In addition, these techniques should be adapted with different sizes of vehicles and with different fields of view, and they are robust to occlusions and lighting changes. All of these will facilitate the development of intelligent vehicle tracking systems.

Vehicle detection can be easily addressed using current state-of-the-art deep CNN models. Vehicle detection methods have been in development for several years in academia and industry. The main problems for vehicle detection are large variations in light, dense occlusion, and large variations in object scales. Most object detection models use general-purpose datasets, such as COCO [12], ImageNet [13], and VOC [14]), where vehicle (i.e., cars) images are usually taken from a lateral or frontal view instead of a cenital plane. This makes them very good for general-purpose images, but they cannot be easily generalized to parking solutions.

An ideal solution for a parking occupancy system should be easily installed in the same camera. Thus, lightweight models are preferable over large models because they can be easily integrated with embedded systems with limited memory and computations. Although lightweight models usually have lower precision, they have a good inference speed. This inference speed gives the model a real-time feature that is highly desirable to be able to extract more information from the car and parking. Object detection models are usually classified into two families: one-stage and two-stage. The two-stage family is called regions with convolutional neural networks (R-CNN family [15]). In turn, SSD [16], and YOLO [17]–[20] are related to one-stage family detectors. The R-CNN family is a region-based detector that includes two stages. First, the model suggests a set of regions of interest (ROIs) using a regional proposal network. Because the potential bounding box candidates can be infinite, the proposed regions are sparse. Next, candidate regions were processed using a classifier. In turn, the one-stage family skips the region proposal stage and directly runs the detection over a dense sampling of possible locations. This yields a faster and simpler detection process but may potentially reduce the performance slightly. Thus, we propose the use of a one-stage detector as a baseline for the proposed vehicle detector.

Consequently, this work aims to develop real-time and lightweight vehicle detection for different vehicle scales based on advanced convolutional neural networks (CNNs) for large parking with a particular camera with a cenital plane view and many parking spots. This camera configuration is highly desired because locating the camera at high heights with a proper view of the parking allows the monitoring of the entire parking with just one camera (as shown in Fig. 1). Moreover, using such a system will have the capability to extract extra information about the detected vehicles, such as color, brand, trajectory, in future upgrades, if desired. Such a system will feed more data into the smart city environment and allow more advanced data exploitation.



**FIGURE 1.** Overview of desired large parking configuration and a cenital view camera.

However, this scenario implies an image with multiple small objects, and to achieve such a system, detection must maintain the inference speed of one-stage detectors while improving their performance for small objects in such scenarios. Small and tiny object detection methods depend on four pillars: multiscale representation, contextual information, super-resolution, and changing the region proposal. Among them, a multiscale approach, such as feature pyramid networks and similar [8], [21], to detect tiny and small objects is the most common procedure in this subfield. However, this approach can quickly escalate and worsen the inference speed of the detector; therefore, we used it sparingly to ensure the same inference speed. We also adopted a contextual-information-based approach using attention modules, which should improve performance with a minimal increase in resources and inference speed.

Consequently, in this work, we present a deep learning model to improve object detection for small and tiny objects, specifically, cars. Our contributions are:

- Introducing a new first layer in the state-of-the-art model, that is, YOLOv5. This new first layer was used to extract discriminative features from different scales. The proposed work suggested replacing the focus layer of YOLO-v5 with a multi-scale layer based on an Efficient Neural Network (ENet) [22]. The introduction of this new layer in YOLO-v5 resulted in a significant outperformance of the baseline model.
- To avoid redundant low-level features in the backbone of YOLO, we assess the impact of spatial and channel information using attention modules on the performance

of tiny object detection. The proposed attention modules can capture meaningful information of the “where” and “what” for tiny object detection. Although its impact is not as large as that of the new multi-scale layer, the use of attention modules improves the performance of the proposed model by a minimal margin.

The rest of this paper is structured as follows: Section II presents the proposed methodology. In Section III, different experiments are performed and discussed. Finally, Section IV concludes the study.

## II. METHOD

The YOLO network has the advantage of being much faster than other networks of the one-stage family. Moreover, it achieved comparable results to the state-of-the-art and still maintained accuracy, and its predictions depended on the global context of the input image. Consequently, our proposed model is based on the Yolo architecture as a baseline.

The architecture of the YOLO network contains many layers that connect with each other. The operations performed in each step can summarise the YOLO-v5 network into three different sections. The first section, the backbone (i.e., called *csppdarknet* [23]), is composed, in the case of Yolo-v5, of the most common operations in CNNs (e.g., convolutions, concatenations, max-pooling) and a simple forwarding mechanism that is constructed to extract multiple features for the next section. The concept of the backbone is a common and old topic in multiple deep learning networks of object detection and is used as a simple base network. For instance, in SSD [16], the common VGG-16 network [24] is used as the backbone. The *csppdarknet* network helps the YOLO model to have sufficient ability to learn the complex features of the input images. In addition, the *csppdarknet* network copes with the problems of repeated gradient information in large-scale backbones. It also integrates the gradient changes into the feature map, yielding a significant reduction in the trainable parameters and floating-point operations per second, which increases the inference speed and accuracy.

In the next section, the neck is responsible for adding and mixing all the different features computed across all convolutions in the backbone and preparing them to feed into the head section. Yolo-v5 applied an improved PANet [25], named bi-directional feature pyramid network (Bi-FPN) as its neck, in order to allow easy and fast multi-scale feature fusion. Bi-FPN introduces learnable weights, enabling the network to learn the importance of different input features, and repeatedly applies top-down and bottom-up multi-scale feature fusion.

Finally, the head section on the other hand is composed of convolutional layers for bounding boxes and classes predictions. Yolo-v5 integrates a compound scaling method that uniformly scales the resolution, depth, and width for all: backbone, feature network, and box/class prediction networks at the same time, which ensures maximum accuracy and efficiency under limited computing resources.

In general, most of the detectors failed to detect tiny objects properly. The results in section III show that the precision of the YOLO network on tiny objects is very low compared to the benchmark based on COCO. Thus, in this paper, we adapt the YOLO-v5 model to be more efficient with small and tiny objects (i.e., vehicles).

In the last released version of Yolo [20], the Focus layer is one of the important modules of Yolo-v5. As shown in Fig. 6-(a), the Focus layer first copies the input image size (e.g.,  $3 \times 256 \times 256$ ) to four copies. The four copies were then sliced into four slices by sampling with a step size of 2 (i.e.,  $3 \times 128 \times 128$ ). The four slices are then concatenated in-depth with an output of  $12 \times 128 \times 128$ , and then passed to the next convolutional layer with 32 kernel filters to generate an output of  $32 \times 128 \times 128$ , and the result is fed into the next convolutional layer through batch normalization and RELU as an activation function.

Based on our experiments on the pre-trained models of Yolo-v5, we suspect that the Focus layer is not capable of properly extracting spatial information for tiny objects that degrades the model performance. Thus, we propose two different mechanisms to enhance spatial and channel information extraction for tiny object detection: 1) The first mechanism is to use channel and position attention modules [26], as shown in Fig. 6(b and d), respectively. 2) The second mechanism involves substituting the Focus layer with a multi-scale module (MSM), as shown in Fig. 6(c, d, and e).

To stack the RGB information, the Focus layer shown in Fig. 4 splits the image and translates spatial information into nonqualitative features. Although it is a quick operation and greatly reduces the inference time to fully utilize GPU operations, it was created to translate spatial information into depth information [27] by simply stacking the quarters of the image. However, Focus layer does not properly represent small and tiny spatial features. A good way to enhance tiny spatial features is to use multi-scale convolutional operations, such as that proposed in [21], [22]. By up-sampling the input image rather than down-sampling, the deep network can be better adapted to anchors, even with too tiny objects.

The MSM shown in Fig. 3 is composed of three branches, which up-samples the input image to multiple scales (i.e., in this work we used  $\times 1$ ,  $\times 2$ , and  $\times 4$  scales) using a bilinear interpolation approach and then each scaled image is fed to an initial block of ENet's [22] (Fig. 2). The ENet model can be run on embedded boards because it is a very light model and is more applicable to mobile robotics systems. ENet's initial block uses a concatenation of two parallel operations. The first 13 filters of a  $3 \times 3$  convolution of stride 2, and the second is a max-pooling operation on the input image. The concatenation of the two branches resulted in a tensor of 16 feature maps. Indeed, in most deep learning models, the pooling operation is achieved after a convolution to increase the feature map depth; however, it is computationally expensive. Therefore, as proposed in [22], we chose to perform a pooling operation in parallel with a convolution and concatenate the resulting feature maps. This technique allowed us to speed

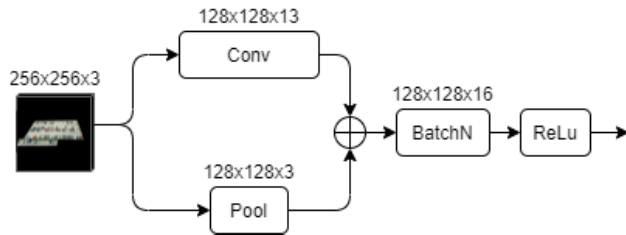


FIGURE 2. ENet initial block architecture.

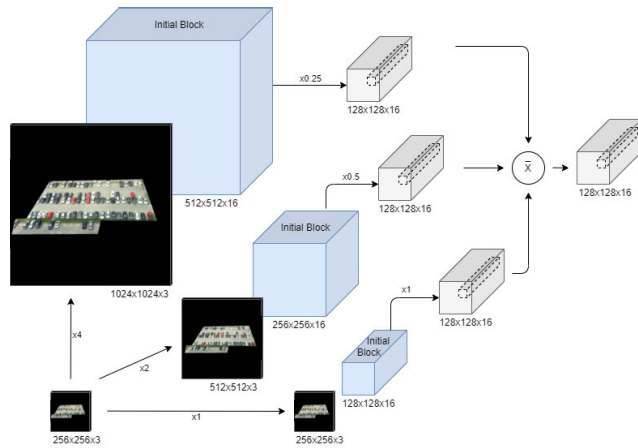


FIGURE 3. Multi-scale module architecture.

up the inference time of the initial block 10 times. To reduce the computational cost of 2D convolution filters, we used a 1-D kernel factorized convolution that is  $1 \times k$  followed by  $k \times 1$ . The neuron's receptive field as the patch of the total field of view (i.e.,  $k = 3, 5, 7$ ) is separately defined for each scale. Consequently, similar to the focus layer, this adaptation allows us to avoid the trainable parameters of YOLO-v5 to grow [20]. At the same time, this provides the first-layer potential to learn features on multiple scales. The resulting tensor is downsampled again to the  $x1$  scale (i.e., the original size of the input image), resulting in three branches with an original scale tensor that are merged together using a combination operation. Such an operation typically computes the average activation of the corresponding units in each branch. The motivation behind this choice is to be able to up-sample the input and have a trainable layer while keeping the layer parameters at a minimum. By scaling the image with several factors, the method can enhance the spatial information of tiny objects while maintaining a relatively low number of parameters. In this way, the network should be able to locate features on the enlarged objects.

However, the use of a multi-scale approach may tend to feed redundant information in likeable low-level backbone features. Furthermore, the contextual information of the MSM branches may be different, degrading the performance of pixel-wise recognition. To overcome such problems and refine the input feature, we propose spatial-channel attention modules (SCAM) [26], as shown in Fig. 5, that a framework that resolves the weakness of using the multi-scale approach

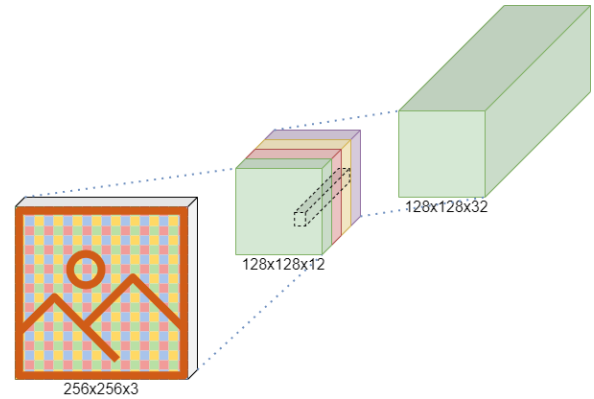


FIGURE 4. Standard focus layer of YOLO-v5 model.

in the detection task. On the one hand, the Spatial Attention Module (SAM) is a module that utilizes the inter-spatial relationship of features (i.e., in our case, vehicle or no vehicle). Unlike channel attention, spatial attention focuses on where is an informative part. To compute the spatial attention, the original feature map is reinterpreted with two different convolution matrices, and reshaping and transpose operations are applied to the results to multiply both matrices and extract an  $N \times N$  ( $H \times W \times H \times W$ ) spatial attention matrix that defines the position of the image where the information is high. This matrix is used to perform an element-wise sum operation with another convolutional feature map from the original one to obtain a spatial-weighted original feature map. On the other hand, the channel attention module (CAM) exploits the inter-channel relationship of features. Since each channel of a feature map is considered as a feature detector, channel attention focuses on 'what' is meaningful given an input image. To achieve this, CAM uses a similar approach to SAM, in which there are no new feature maps or the reshape-multiplication target. Instead of obtaining a  $H \times W \times H \times W$  matrix, we obtain a  $C \times C$  matrix, thus focusing on which elements are important rather than where is the information. The fused feature representation of the SCAM is obtained by adding the space-wise representation of the SAM and channel-wise representation of the CAM. Every spatial attention map is summed back to the channel-attention tuned feature maps for adaptive feature refinement. The attention mechanism can be directly adapted to any feature representation problem, and it encourages the network to capture rich contextual relationships for better feature representations. In this study, we take advantage of this finding to create an SCAM that collects context information from all pixels to adaptively recalibrate the spatial and channel responses of the objects (i.e., vehicles) in a resulting convolutional feature map.

For the YOLO-v5 adaptation, we used different variations:

- 1) just adding SCAM at the end of the backbone section before feeding into the Neck section, as shown in Fig. 6-b.
- 2) just replacing the focus layer of the YOLO-v5 by the proposed MSM, as shown in Fig. 6-c.



- 3) combining between the two proposed modules; MSM and SCAM that is within the Backbone and Neck sections, as shown in Fig. 6-d.
- 4) applying SCAM to each scale branch of MSM as shown in Fig. 6-e.

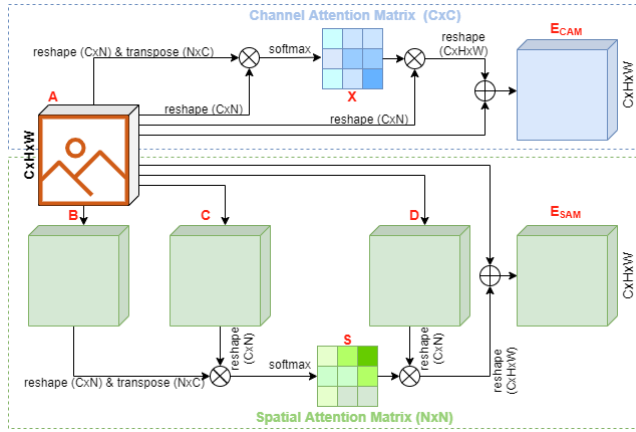


FIGURE 5. Spatial and channel attention modules.

### III. EXPERIMENTS

#### 1) DATASET

PKLot<sup>1</sup> is the dataset used to train and validate the proposed model. This dataset is a folder structure based on the parking location, meteorological states, and specific days. From all the images, we only used a subset consisting of the cenital plane (PUCPR). The 4474 images have almost 100 parking spots tagged with the location of bounding boxes and occupancy for a total of 424269 tagged spots. These spots were randomly distributed to 80% and 20% for the training and testing sets, respectively.

To prepare the dataset for training and validating the proposed model, we applied the following procedure:

- Since each occupied parking spot is used by a car and the cars are within the parking spot, we used these occupied spots annotation as the localization (i.e., bounding box) and the class (i.e., vehicle or no vehicle).
- Since there are more cars present in the image than those tagged, we applied a mask to only include the area where the tagged cars found and exclude non-tagged cars. In short, we apply a region of interest to our monitored/tagged area.
- We adapted and translated PKLot format annotations to coco format annotations to be able to use easily with the YOLO-v5 model.

#### 2) EVALUATION METRICS

Several metrics have been used to assess the performance of deep-learning detection models. Precision (P) is the proportion of True Positive among all Positive detected:

$$P = \frac{TP}{TP + FP}$$

<sup>1</sup><https://web.inf.ufpr.br/vri/databases/parking-lot-database/>

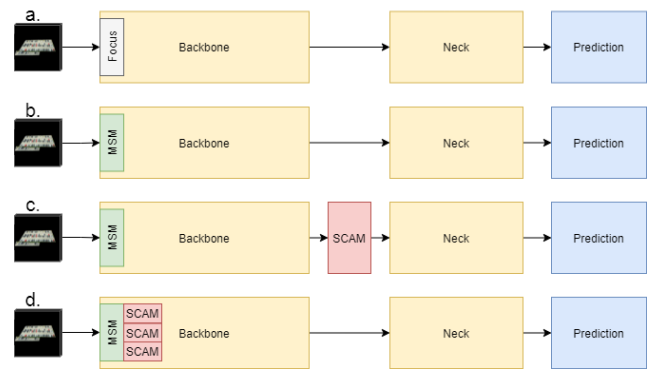


FIGURE 6. Proposed architectures based on Yolo-v5. a) Yolo-v5 baseline architecture. b) Yolo-v5 with replacing focus layer by multi-scale module (MSM). c) Yolo-v5 with MSM and spatial and channel attention module (SCAM) at the end of backbone. d) Merging SCAM into each branch from the MSM layer.

Recall (R) is the proportion of detected Positives among all ground true positives:

$$R = \frac{TP}{TP + FN}$$

Here,  $mAP.5$  and  $mAP.95$ , representing the mean Average Precision of all detections with an Intersect of Union (IoU) of 50% and 95%, respectively, where the IoU is the result of obtaining the intersection of both bounding boxes (detected and ground truth) normalized by the union of both bounding boxes. The Average Precision (AP) is then computed based on the detection of a given class with an IoU greater than 50% or 95%. Finally, the mean Average Precision (mAP) was computed using the average for all the classes.

In addition, some other validation errors for bounding boxes ( $Box$ ), ( $Obj$ ), and ( $Cls$ ) were calculated, as shown in Table 2, where the  $Box$  error is computed using IoU, as a result of the intersection of the predicted and ground-truth normalized by its union. The  $Obj$  error is the objectness score, which is used to compute the likelihood that a specific bounding box is an object. The  $Cls$  error corresponds to a multi-classification score. The  $Obj$  and  $Cls$  errors are computed using the Focal loss function, which is an extension of the cross-entropy loss function that would downweight easy examples and focus training on hard negatives.

Furthermore, there are other metrics based on the efficiency of the model, such as the inference speed, which normally refers to frames per second (FPS), and the number of parameters that is normally a good indicator of the model complexity.

#### 3) DATA AUGMENTATION

The dataset images were augmented using YOLOv5 standard augmentation<sup>2</sup> following different techniques: mosaic/mixup [28], letterbox, perspective, HSV color space, and flipping (up-down, reight-left). No changes were made to avoid any bias in the results and to check the implications

<sup>2</sup>[hyp.scratch.yaml](https://github.com/ultralytics/yolov5/blob/master/hyp.scratch.yaml)

of the MSM and attention modules outside the data augmentation methods.

#### 4) IMPLEMENTATION

The experiments were conducted in a Ubuntu 18.04 system with a GPU of Nvidia 2080-Ti and a Pytorch library.<sup>3</sup> We used the available Pytorch-based implemented code of YOLO-v5<sup>4</sup> as a baseline. This baseline also implies using hyperparameters and configurations as in YOLOv5 (as in *hyp.scratch.yaml*), most destacable:

- SGD optimizer with  $lr0 = 0.01$ ,  $momentum = 0.937$  and  $weight_decay = 0.0005$
- Anchors:  $P3/8[(10, 13) (16, 30) (33, 23)]$ ,  $P4/16[(30, 61) (62, 45) (59, 119)]$ ,  $P5/32[(116, 90) (156, 198) (373, 326)]$

#### 5) RESULTS AND ANALYSIS

First, to show the effects of different improvements in the baseline YOLO-v5 model, we performed an ablation study on our proposed model with different variations. In this ablation study, we analyzed the effect of the addition of the two mechanisms of multi-scaling and attention models on the performance of a baseline model, YOLO-v5. In Table 1, we present the variations in the proposed network. We changed the proposed architecture of the MSM by adding an attention network to the multi-scale branches or to the backbone, in addition to using different types of pooling.

First, we assessed the baseline YOLO-v5 model by validating the PKLot dataset into both a COCO-trained YOLO-v5 model and a PKLot fine-tuned YOLO-v5 model. In the first two experiments, no modifications were made to the model, as shown in Fig. 6-a. We used four profiles of the YOLO-v5 network: small (YOLO-v5s), medium (YOLO-v5m), large (YOLO-v5l), and extreme (YOLO-v5x). Note that among the four profiles, the s profile is the fastest model in terms of speed and the smallest models in terms of size, and the x profile is the highest in size and the lowest in speed.

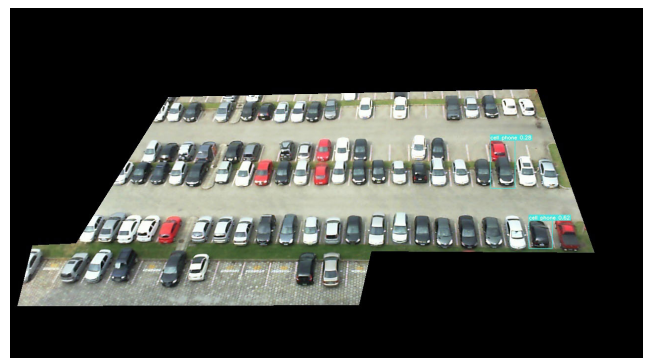
As shown in Table 2, as expected, the COCO-trained YOLO-v5s and YOLO-v5m models yielded the worst results in terms of R and mAP. However, the fine-tuned YOLO-v5m achieved a much better performance in terms of R, mAP0.5 and mAP95 compared to the COCO-trained baseline models. The R and mAP values improve drastically from  $R = 0.261$  and  $mAP.5 = 0.562$  with the COCO-trained YOLO-v5m model to  $R = 0.995$  and  $mAP.5 = 0.9938$  with the fine-tuned YOLO-v5s model. However, the P value was significantly reduced by 10%. We can explain the reduction in Precision, P, compared to the significant improvement in R and mAP values, because the COCO-trained model is overflowing with candidates for the targets. Using qualitative results, we can see the differences between the COCO-trained YOLO-v5 model in Fig. 7, and the fine-tuned YOLO-v5 model in Fig. 8. It can be seen that not only does the COCO-trained model

miss the detection of several cars in the input images, but it also misclassifies them by detecting cell phones (blue) instead of cars (orange). In the fine-tuned YOLO-v5 model, YOLO-v5 can correctly classify the objects in the input images, although we still observe some false positives (i.e., out of the region of interest of the monitored area).

Second, we add the SCAM blocks to the baseline YOLO-v5 model to assess the performance of the adapted variation. To minimize the parameters of the model, we used YOLO-v5s (s profile). In particular, we tested three different tension modules between the backbone and neck sections of YOLOv5: spatial attention module (*Yolov5\_SAM\_backbone*), channel attention module (*Yolov5\_CAM\_backbone*), and channel-spatial attention module (*Yolov5\_SCAM\_backbone*). As shown in Table 2, adding the SAM block only gained a marginal and a small improvement of (+0.1% to +0.4%) on Recall compared to the fine-tuned model YOLOv5s model, which yields a reduction of 1% in the precision values. Thus, we can say that adding SAM, CAM, or even SCAM does not yield a significant improvement in true positive detection compared to the fine-tuned YOLO-v5 models.

Third, we replaced the focus layer of YOLO-v5 with the MSM layer (i.e., named *Yolov5\_MSM*) (Fig. 6-b)). As we can see in Table 2, there is a significant improvement in the Precision values outperforming the Focus layer from 0.6273 of YOLOv5s to 0.9203 in *Yolov5s\_MSM*). Although it is slight, there is also a small improvement in the other performance values of R, mAP, Box, and Obj, except for the Cls loss. Qualitatively, Fig. 9 shows the same example of the PKLot dataset shown in Fig. 7 with a much better detection rate. Adding the MSM block to YOLOv5 was able to properly detect and classify cars in the images. Fig. 9 also showed the ability of the *Yolov5\_MSM* model in localizing and classify the cars even in a crowded scenario. Indeed, the substitution of the Focus layer for an MSM in the YOLO-v5 model provides an efficient feature representation for tiny cars (objects) present in PKLot.

After the replacement, we added the SAM, CAM, and SCAM blocks to the MSM blocks to observe if we could benefit from the marginal improvement of SCAM. The *Yolov5s\_MSM* model is modified in two



**FIGURE 7.** An example of the PKLot dataset with the pretrained YOLO-v5 model.

<sup>3</sup><https://pytorch.org/>

<sup>4</sup><https://github.com/ultralytics/yolov5>

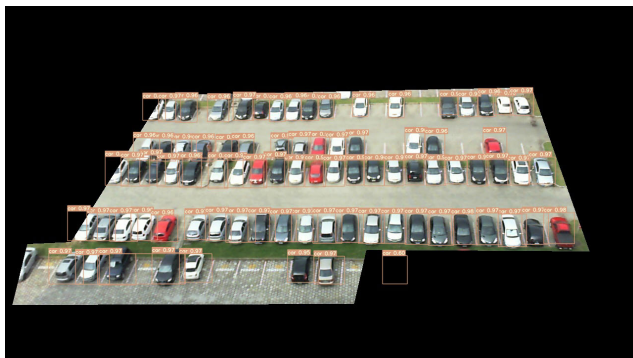
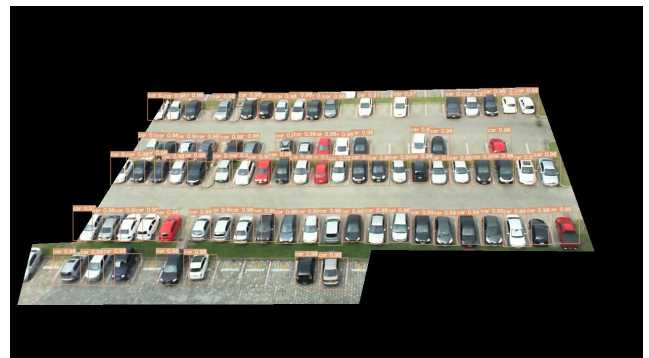
**TABLE 1.** Compilation of the different variations of the proposed model using multi-scaling, attention and pooling techniques.

Features	Multiscale Branches (x1, x2, x4)		
	Single branch	Combinations of 2 branches	All 3 branches
<b>Base</b>	Yolov5_SM_x1, Yolov5_SM_x2, Yolov5_SM_x4	Yolov5_MSM_x2&x1, Yolov5_MSM_x4&x1, Yolov5_MSM_x4&x2	Yolov5_MSM
<b>CAM</b>	Yolov5_SM_x2_CAM		Yolov5_MSM&CAM
<b>CAM Backbone</b>	Yolov5_SM_x2_CAM_backbone		Yolov5_MSM&CAM_backbone
<b>SAM</b>	Yolov5_SM_x2_SAM		Yolov5_MSM&SAM
<b>SAM Backbone</b>	Yolov5_SM_x2_SAM_backbone		Yolov5_MSM&SAM_backbone
<b>SCAM</b>	Yolov5_SM_x2_SCAM		Yolov5_MSM_SCAM
<b>SCAM Backbone</b>	Yolov5_SM_x2_SCAM_backbone		Yolov5_MSM_SCAM_backbone
<b>NoPool</b>			Yolov5_MSM_NoPool
<b>AvgPool</b>			Yolov5_MSM
<b>MaxPool</b>			Yolov5_MSM_MPool
<b>Max&amp;Avg Pool</b>	Yolov5_SM_x2_MAPool		Yolov5_MSM_MAPool, Yolov5_MSM_MAPool_CAM_backbone

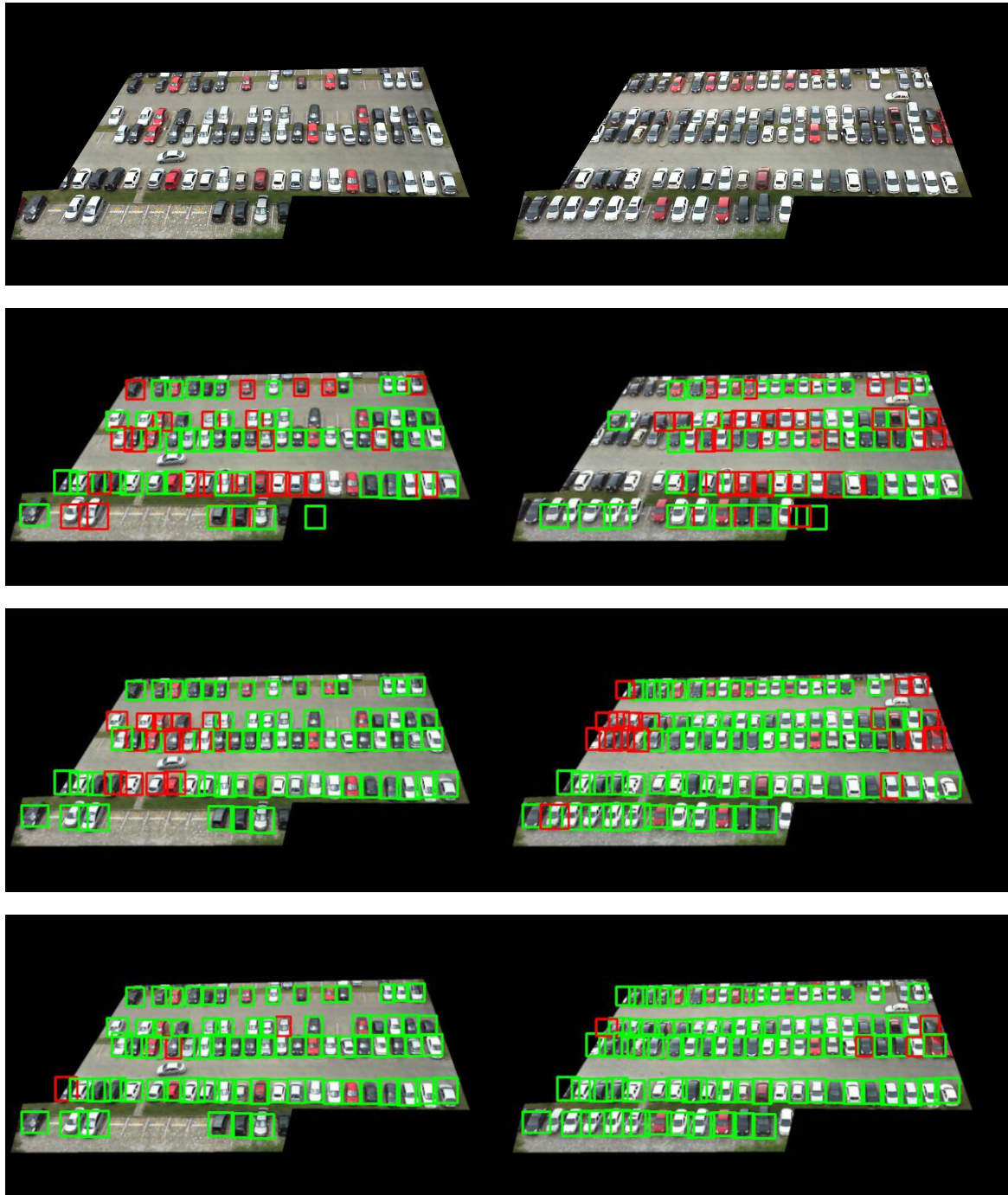
different ways. First, we added attention blocks, as in the second experiment (Fig. 6-c), between the Backbone and Neck using SAM (*Yolov5\_MSM&SAM\_backbone*), CAM (*Yolov5\_MSM&CAM\_backbone*) and channel-spatial (*Yolov5\_MSM&SCAM\_backbone*) modules. In addition, we integrated the attention blocks (i.e., SAM, CAM, and SCAM) into each branch of the MSM (Fig. 6-d) for spatial (*Yolov5\_MSM&SAM*), channel (*Yolov5\_MSM&CAM*) and channel-spatial (*Yolov5\_MSM&SCAM*). The combination of the MSM and attention modules yields an improvement from 3% to +4% in the Precision values. Among the six variations, the (*Yolov5\_MSM&CAM\_backbone*) model yielded the best precision value with an improvement of more than 4, 31% compared to *Yolov5s\_MSM*. However, *Yolov5\_MSM&CAM* yields results comparable to *Yolov5\_MSM&CAM\_backbone* with an improvement of +4.0 in the Precision values. Thus, we can note that MSM helps YOLO-v5 to find discriminative spatial features, and the CAM/SCAM attention module helps the network to enrich the extracted features with discriminative multi-channel features. Again, the variations in the other values (R, mAP, and the different errors) are marginal and likely not significant. A comparison between the baseline YOLO-v5 model and its variations is shown in Fig. 10 that clarifies the ability of the proposed adaption on the detection

rate. Besides, Fig. 10 shows that the integration of MSM and CAM with the YOLOv5 model helps to maximize the overlap of predicted bound-boxes versus actual bounding boxes with  $IoU > 0.95$  for tiny car detection.

However, in order to fully exploit the dataset knowledge and adapt our model to the dataset and obtain the best model for that application, we suspect that the use of a multiscale of three branches might not be the most efficient network that fits this dataset. The size of the cars on the datasets, although tiny, does not change, so it might be better for this application to use an MSM of just one branch or even two branches. For this, from the MSM three branches (x1, x2, x4 being the scale used on the branch), we experiment on both a combination of MSM for 2 branches (*Yolov5\_MSM\_x2\_x1*, *Yolov5\_MSM\_x4\_x1* and *Yolov5\_MSM\_x4\_x2*) and a single branch (*Yolov5\_SM\_x1*, *Yolov5\_SM\_x2* and *Yolov5\_SM\_x4*) without attention modules. Thus, in Table 2, we can see the different results of these experiments. Taking note of the Precision value (again, the other values are so similar that are unlikely to be significant), we observe that from all the experiments, *Yolov5\_SM\_x2* is the one with the highest Precision (95,05%). However, it is still comparable, even under the values of these values from the MSM experiment with the SCAM. Therefore, to determine if the

**FIGURE 8.** An example of the PKLot dataset with the fine-tuned YOLO-v5 model.**FIGURE 9.** An example of the PKLot dataset with the YOLO-v5 model with multi-scale module (MSM).





**FIGURE 10.** Two examples of the PKLot dataset with (Row 1) original images, (Row 2) car detection results with the fine-tuned YOLO-v5 model with the PKLot dataset, (Row 4) car detection results with the *Yolov5s\_MSM* model, and (Row 5) car detection results with the *Yolov5s\_MSM\_CAM\_backbone* model. Green bounding-boxes are related to a detection with  $IoU > 0.95$ , and the red ones with otherwise.

multiscale is better in these kinds of scenarios, we apply the same CAM modifications to the *Yolov5\_SM\_x2* model for comparison. As shown in Table 2, *Yolov5\_SM\_x2* obtains a higher score in P than the experiments using SCAM (*Yolov5\_SM\_x2\_CAM*, *Yolov5\_SM\_x2\_CAM\_backbone*,

*Yolov5\_SM\_x2\_SAM*, *Yolov5\_SM\_x2\_SAM\_backbone*, *Yolov5\_SM\_x2\_SCAM*, *Yolov5\_SM\_x2\_SCAM\_backbone*).

However, the use of a pooling layer on MSM is controversial, and we use only 13 convolution features to keep the layer light and three more features from pooling. With these



**TABLE 2.** Comparison between the baseline model and different variations of the proposed model in terms of different evolution metrics.

	P	R	mAP.5	mAP.95	box	obj	cls
Coco_yolov5s	0.748	0.0386	0.119	0.0411			
Coco_yolov5m	0.813	0.261	0.562	0.193			
Yolov5s	0.6273	0.995	0.9938	0.988	0.01481	0.06332	<b>0.0006731</b>
Yolov5m	0.6333	0.9886	0.9934	0.991	0.01417	0.06179	0.000800
Yolov5l	0.6365	0.9867	0.9922	0.9908	0.01408	0.06117	0.00082391
Yolov5x	0.6387	0.9865	0.9908	0.9897	0.01387	0.0626	0.0008749
Yolov5_SAM_backbone	0.6355	0.9985	0.9939	0.9834	0.01536	0.06534	0.0007774
Yolov5_CAM_backbone	0.6227	0.9996	0.9954	0.9877	0.01524	0.06529	0.0007801
Yolov5_SCAM_backbone	0.6245	0.9961	0.9936	0.984	0.0154	0.06667	0.0007828
Yolov5_MSM	0.9203	<b>0.9998</b>	0.998	0.9974	0.002557	0.01896	0.0008869
Yolov5_MSM&CAM_backbone	<b>0.9634</b>	0.9959	0.9979	0.9967	0.002904	0.01798	0.001231
Yolov5_MSM&CAM	0.9515	0.9988	0.9984	0.9978	0.002635	<b>0.01655</b>	0.0009467
Yolov5_MSM&SAM_backbone	0.9565	0.9994	0.9983	0.997	0.003024	0.01736	0.001218
Yolov5_MSM&SAM	0.9601	0.9974	0.9984	0.9976	0.002713	0.01668	0.0009961
Yolov5_MSM_SCAM_backbone	0.9565	0.9994	0.9983	0.997	0.003024	0.01736	0.001218
Yolov5_MSM_SCAM	0.9608	0.9996	<b>0.9985</b>	<b>0.9979</b>	0.002679	0.01663	0.0009461
Yolov5_MSM_x2&x1	0.9233	<b>0.9998</b>	0.9981	0.9973	0.00254	0.01881	0.0008857
Yolov5_MSM_x4&x1	0.9328	0.9997	0.9984	0.9976	0.002575	0.01795	0.0008972
Yolov5_MSM_x4&x2	0.9167	<b>0.9998</b>	0.9982	0.9975	0.002671	0.0189	0.0008928
Yolov5_SM_x1	0.9241	0.9997	0.9979	0.9969	0.002644	0.0191	0.0009034
Yolov5_SM_x2	0.9505	0.9994	0.998	0.9971	0.002587	0.01694	0.0009089
Yolov5_SM_x4	0.9247	<b>0.9998</b>	0.9983	0.9977	0.002583	0.01837	0.0009054
Yolov5_SM_x2_CAM	0.9158	0.9997	0.9982	0.9976	<b>0.00252</b>	0.01927	0.0008922
Yolov5_SM_x2_CAM_backbone	0.9152	<b>0.9998</b>	0.998	0.9974	0.002627	0.01981	0.001194
Yolov5_SM_x2_SAM	0.931	<b>0.9998</b>	0.9982	0.9975	0.002602	0.01752	0.000895
Yolov5_SM_x2_SAM_backbone	0.9206	<b>0.9998</b>	0.998	0.997	0.002607	0.01919	0.001195
Yolov5_SM_x2_SCAM	0.9189	<b>0.9998</b>	0.9981	0.9974	0.002614	0.02016	0.0008909
Yolov5_SM_x2_SCAM_backbone	0.9411	0.9997	0.9984	0.9977	0.002611	0.0178	0.001192
Yolov5_MSM_NoPool	0.9341	0.9997	0.9983	0.9974	0.002617	0.01778	0.0009123
Yolov5_MSM_MPool	0.9202	<b>0.9998</b>	0.998	0.9974	0.002558	0.01897	0.0008862
Yolov5_MSM_MAPool	0.943	<b>0.9998</b>	0.9983	0.9976	0.00263	0.01752	0.000888
Yolov5_MSM_MAPool_CAM_backbone	0.9254	<b>0.9998</b>	0.9982	0.9971	0.002803	0.01908	0.001184
Yolov5_SM_x2_MAPool	0.9112	<b>0.9998</b>	0.998	0.9973	0.002571	0.01969	0.0008979

low numbers of convolution features to extract spatial information, one may think that pooling has no effect or even a detrimental effect. To determine if this pooling actually affected the performance of the model, we checked the pooling layer in order to see if there was improvement by using a pooling layer. Considering that *Yolov5\_MSM* uses average(avg) pooling, we trained the model without attention modules again 1) without any pooling and 16 features of convolution (*Yolov5\_MSM\_NoPool*), 2) using max pooling instead of avg pooling (*Yolov5\_MSM\_MPool*), and 3) using 10 layers of convolution and six layers from pooling (max and avg pooling) (*Yolov5\_MSM\_MAPool*). The results show a marginal improvement of 1 or 2% if we use none of the pooling. Finally, we check that this marginal improvement can be used to further improve our best models by testing max and avg pooling (best pooling performance) with the best models. As seen in Table. 2, both models (*Yolov5\_MSM\_MAPool\_CAM\_backbone* and

*Yolov5\_SM\_x2\_MAPool*) worsen their performances about 4%.

Because YOLO-v5 is a deep detection model that focuses on low parameters and high frames-per-second (fps) inference to be fit in low-end terminals, we must ensure that the changes in the model do not significantly affect this aspect. Thus, Table 3 shows the different evaluations of the different variations of the YOLO-v5 model. The trainable parameters of the *Yolov5s\_MSM* and *Yolov5s\_MSM\_CAM* models have a comparable number of parameters, even fewer parameters compared to the *Yolov5s* model. However, the *Yolov5s\_MSM* model does deteriorate in fps values, but the achieved fps value is maintained within YOLO-v5 larger profiles, very close to the fps values resulting from the *Yolov5l* profile. The *Yolov5s\_MSM* model can achieve almost 30 fps, which yields a reduction of 5 fps compared to the *Yolov5s* model. In addition, the addition of attention modules did not significantly affect the fps values, maintaining approximately 30 fps.

TABLE 3. Performance table.

	fps(secs / 895 imgs)	Params
Yolov5s	35.29(25.358)	7276605
Yolov5m	33.20(26.955)	21375645
Yolov5l	30.54(29.299)	47056765
Yolov5x	28.49(31.417)	87775965
Yolov5_MSM	29.78(30.051)	7265021
Yolov5_MSM&CAM_backbone	29.97(29.862)	7430398
Yolov5_SM_x2	<b>42.82(20.903)</b>	<b>7254653</b>

In fact, by using the single-branch solution (*Yolov5\_SM\_x2*), we can outspeed *Yolov5m* from 35fps to 42 fps, achieving a much greater precision. Thus, we recommend the use of the adapted version of *Yolov5s\_MSM\_CAM* with a high Precision value of 96% and 30 fps for use as an industrial detection model for large parking, or *Yolov5\_SM\_x2* if speed is required.

#### IV. CONCLUSION

In this paper, we proposed a reliable modification of the YOLO-v5 model targeting tiny car objects from a cenital view. Using a multi-scale module and channel/spatial attention mechanisms, the modified version outperformed the original YOLO-v5 for this specific application with a precision of up to 96,34% compared to a precision of 63,87% with the baseline YOLO-v5 model. The proposed model also slightly improved the Recall and mAP values while maintaining the same number of trainable parameters. In addition, the adapted variation yields a decrease in speed compared to the small and medium profiles of the YOLO-v5 network, although it exceeds the large and extreme profiles. On the other hand, the single-branch solution outspeeds the Yolo-v5 small profile and was almost as precise as the multi-branch solution. Ongoing work aims at developing a reliable tracker based on the developed detector. Future work aims to use the developed detector and tracker in low-end terminals, such as a Field-Programmable Gate Array (FPGA) or an NVIDIA Jetson Nano Developer Kit for real-time parking monitoring.

#### ACKNOWLEDGMENT

This research was done thanks to AGAUR funding on Industrial Doctorate modality with Universitat Rovira i Virgili and Quercus Technologies collaboration.

#### REFERENCES

- [1] S. B. Atitallah, M. Driss, W. Boulila, and H. B. Ghézala, "Leveraging deep learning and IoT big data analytics to support the smart cities development: Review and future directions," *Comput. Sci. Rev.*, vol. 38, Nov. 2020, Art.no. 100303. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1574013720304032>
- [2] T. Lin, H. Rivano, and F. Le Mouél, "A survey of smart parking solutions," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 12, pp. 3229–3253, Dec. 2017.

- [3] S. J. Faraji and M. J. Naozar, "Smart parking: An efficient approach to city's smart management and air pollution reduction," *J. Air Pollut. Health*, vol. 4, no. 1, pp. 53–72, Feb. 2019. [Online]. Available: <https://japh.tums.ac.ir/index.php/japh/article/view/199>
- [4] O. Cats, C. Zhang, and A. Nissan, "Survey methodology for measuring parking occupancy: Impacts of an on-street parking pricing scheme in an urban center," *Transp. Policy*, vol. 47, pp. 55–63, Apr. 2016. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0967070X15300858>
- [5] G. Amato, F. Carrara, F. Falchi, C. Gennaro, C. Meghini, and C. Vairo, "Deep learning for decentralized parking lot occupancy detection," *Expert Syst. Appl.*, vol. 72, pp. 327–334, Apr. 2017. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S095741741630598X>
- [6] S. C. K. Tekouabou, E. A. A. Alaoui, W. Cherif, and H. Silkan, "Improving parking availability prediction in smart cities with IoT and ensemble-based model," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 34, no. 3, pp. 687–697, Mar. 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1319157819312613>
- [7] M. A. Merzoug, A. Mostefaoui, G. Gianini, and E. Damiani, "Smart connected parking lots based on secured multimedia IoT devices," *Computing*, vol. 103, no. 6, pp. 1143–1164, Jun. 2021.
- [8] M. dos Santos de Arruda, L. P. Osco, P. R. Acosta, D. N. Gonçalves, J. M. Junior, A. P. M. Ramos, E. T. Matsubara, Z. Luo, J. Li, J. de Andrade Silva, and W. N. Gonçalves, "Counting and locating high-density objects using convolutional neural network," 2021, *arXiv:2102.04366*.
- [9] M. Farag, M. Din, and H. Elshenbary, "Deep learning versus traditional methods for parking lots occupancy classification," *Indonesian J. Elect. Eng. Comput. Sci.*, vol. 19, pp. 964–973, Aug. 2020.
- [10] S. Nurullayev and S.-W. Lee, "Generalized parking occupancy analysis based on dilated convolutional neural network," *Sensors*, vol. 19, no. 2, p. 277, Jan. 2019. [Online]. Available: <https://www.mdpi.com/1424-8220/19/2/277>
- [11] A. Farley, H. Ham, and Hendra, "Real time IP camera parking occupancy detection using deep learning," *Proc. Comput. Sci.*, vol. 179, pp. 606–614, Jan. 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1877050921000533>
- [12] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft COCO: Common objects in context," 2014, *arXiv:1405.0312*.
- [13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
- [14] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Jun. 2010, doi: [10.1007/s11263-009-0275-4](https://doi.org/10.1007/s11263-009-0275-4).
- [15] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," 2015, *arXiv:1506.01497*.
- [16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector," in *Computer Vision—ECCV*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham, Switzerland: Springer, 2016, pp. 21–37.
- [17] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*.
- [18] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*.
- [19] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Scaled-YOLOv4: Scaling cross stage partial network," 2020, *arXiv:2011.08036*.
- [20] G. Jocher, A. Stoken, J. Borovec, L. Changyu, A. Hogan, A. Chaurasia, L. Diaconu, Doug, Durgesh, F. Ingham, Frederik, Guilhen, A. Colmagro, H. Ye, Jacobsolawetz, J. Poznanski, J. Fang, J. Kim, K. Doan, and L. Yu, "Ultralytics/yolov5: V4.0—Nn.SiLU() activations, weights & biases logging, PyTorch hub integration," Ultralytics, Los Angeles, CA, USA, Jan. 2021, doi: [10.5281/zenodo.4418161](https://doi.org/10.5281/zenodo.4418161).
- [21] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," 2016, *arXiv:1612.03144*.
- [22] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "ENet: A deep neural network architecture for real-time semantic segmentation," 2016, *arXiv:1606.02147*.
- [23] C. Kumar B., R. Punitha, and Mohana, "YOLOv3 and YOLOv4: Multiple object detection for surveillance applications," in *Proc. 3rd Int. Conf. Smart Syst. Inventive Technol. (ICSSIT)*, Aug. 2020, pp. 1316–1321.

- [24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
- [25] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8759–8768.
- [26] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, "Dual attention network for scene segmentation," 2019, *arXiv:1809.02983*.
- [27] T. Ridnik, H. Lawen, A. Noy, E. B. Baruch, G. Sharir, and I. Friedman, "TRResNet: High performance GPU-dedicated architecture," 2020, *arXiv:2003.13630*.
- [28] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," 2017, *arXiv:1710.09412*.



**DANIEL PADILLA CARRASCO** received the B.S. degree in telecommunications from the Technical School of Castelldefels, Castelldefels, Spain, and the M.S. degree in artificial intelligence from the Polytechnic University of Catalonia, Barcelona, Spain, Rovira i Virgili University, Tarragona, Spain, and the University of Barcelona, Barcelona, in 2015. He is currently pursuing the Ph.D. degree with the Department of Computer Engineering and Mathematics, Rovira i Virgili University.

He is currently a Software Engineer with Quercus Technologies, Reus, Spain. His research interests include image processing, computer vision, and machine learning.



**HATEM A. RASHWAN** received the B.S. and M.S. degrees in electrical engineering from South Valley University, Egypt, in 2002 and 2007, respectively, and the Ph.D. degree in computer vision from Rovira i Virgili University, Spain, in 2014. From 2004 to 2009, he was an Assistant Lecturer with the Electrical Engineering Department, South Valley University. From January 2010 to October 2014, he joined the IRCV Group, Department of Computer Engineering and Mathematics, Rovira i Virgili University, a Research Assistant.

From November 2014 to August 2017, he was a Postdoctoral Researcher with the VORTEX Group, IRIT, CNRS, INP-Toulouse, University of Toulouse, France. From 2018 to 2020, he was the Beatriu de Pinós Researcher at URV. He is currently the Director of research and an Assistant Professor at DEIM, URV. His research interests include image processing, computer vision, machine learning, and pattern recognition.



**MIGUEL ÁNGEL GARCÍA** received the B.S., M.S., and Ph.D. degrees in computer science from the Polytechnic University of Catalonia, Barcelona, Spain, in 1989, 1991, and 1996, respectively. In 1996, he joined the Department of Software, Polytechnic University of Catalonia, as an Assistant Professor. From 1997 to 2006, he was with the Department of Computer Science and Mathematics, Rovira i Virgili University, Tarragona, Spain, where he was the Head of the

Intelligent Robotics and Computer Vision Group. In 2006, he joined the Department of Informatics Engineering, Autonomous University of Madrid, Spain, where he is currently an Associate Professor. His research interests include mobile robotics, image processing, and 3-D modeling.



**DOMÈNEC PUIG** received the M.S. and Ph.D. degrees in computer science from the Polytechnic University of Catalonia, Barcelona, Spain, in 1992 and 2004, respectively. In 1992, he joined the Department of Computer Engineering and Mathematics, Rovira i Virgili University, Tarragona, Spain, where he is currently a Professor. Since July 2006, he has been the Head of the Intelligent Robotics and Computer Vision Group, Rovira i Virgili University. His research interests

include image processing, texture analysis, perceptual models for image analysis, scene analysis, and mobile robotics.

...