IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 19, 2021, accepted December 12, 2021, date of publication January 14, 2022, date of current version January 24, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3143365

Improved Vision-Based Vehicle Detection and Classification by Optimized YOLOv4

JINGYI ZHAO[1], SHENGNAN HAO[1], CHENXU DAI[1], HAIYANG ZHANG[2], LI ZHAO[3], ZHANLIN JI[1,4], (Member, IEEE), AND IVAN GANCHEV[4,5,6], (Senior Member, IEEE)

[1]College of Artificial Intelligence, North China University of Science and Technology, Tangshan 063210, China
[2]Department of Computing, Xi'an Jiaotong-Liverpool University, Suzhou 215000, China
[3]Research Institute of Information Technology, Tsinghua University, Beijing 100080, China
[4]Telecommunications Research Centre (TRC), University of Limerick, V94 T9PX Limerick, Ireland
[5]Department of Computer Systems, University of Plovdiv "Paisii Hilendarski," 4000 Plovdiv, Bulgaria
[6]Institute of Mathematics and Informatics—Bulgarian Academy of Sciences, 1040 Sofia, Bulgaria

Corresponding authors: Zhanlin Ji (zhanlin.ji@gmail.com) and Ivan Ganchev (ivan.ganchev@ul.ie)

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFE0135700; in part by the MES under Grant D01-387/18.12.2020 for NCDSC part of the Bulgarian National Roadmap on RIs; and in part by the Telecommunications Research Centre (TRC), University of Limerick, Ireland.

ABSTRACT Rapid and precise detection and classification of vehicles are vital for the intelligent transportation systems (ITSs). However, due to small gaps between vehicles on the road and interference features of photos, or video frames containing vehicle images, it is difficult to detect and identify vehicle types quickly and precisely. For solving this problem, a new vehicle detection and classification model, named YOLOv4_AF, is proposed in this paper, based on an optimization of the YOLOv4 model. In the proposed model, an attention mechanism is utilized to suppress the interference features of images through both channel dimension and spatial dimension. In addition, a modification of the Feature Pyramid Network (FPN) part of the Path Aggregation Network (PAN), utilized by YOLOv4, is applied in order to enhance further the effective features through down-sampling. This way, the objects can be steadily positioned in the 3D space and the object detection and classification performance of the model can be improved. The results, obtained through experiments conducted on two public data sets, demonstrate that the proposed YOLOv4_AF model outperforms, in this regard, both the original YOLOv4 model and two other state-of-the-art models, Faster R-CNN and EfficientDet, in terms of the mean average precision (*mAP*) and *F1 score*, by achieving respective values of 83.45% and 0.816 on the BIT-Vehicle data set, and 77.08% and 0.808 on the UA-DETRAC data set.

INDEX TERMS Computer vision, object detection, object classification, vehicle model identification, attention mechanism, feature fusion, you only look once (YOLO), region-based convolutional neural network (R-CNN), EfficientDet.

I. INTRODUCTION

Presently, object detection and classification are widely used in intelligent transportation systems (ITSs), and some industrial and military systems. ITSs, for instance, can perform vehicle detection and classification for comprehensive analysis of passing vehicles for the purposes of accomplishing an efficient vehicle traffic management and control, and urban planning. The existing object detection methods can be divided into two groups [1]: (i) *hardware-based*; and (ii) *vision-based* methods. The latter try to localize objects in a photo / video frame by making bounding boxes (BBoxes) around the detected objects. If, in addition, object classification is performed, then the predicted class label is also shown on the image along with the confidence score associated with each bounding box (BBox) [2]. The vision-based object detection methods can be further categorized, as per [1], as: (i) *dimension-based*; (ii) *logo-based*; and (iii) *feature-based* methods. The traditional (before 2012) feature-based object detection methods, such as Haar [3], the histogram of oriented gradients (HOG) [4], the HOG–support vector

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

machine (HOG-SVM) [5], etc., consist of three phases [2]: (i) informative region selection; (ii) feature extraction; and (iii) classification. Later, these methods were replaced by deep learning (DL) feature-based methods, due to the continuous growth of large volumes of data (Big Data) and the fast development of (multicore) processors and Graphical Processing Units (GPUs) [2]. Currently, the DL feature-based methods are considered the state-of-the-art, thanks to their remarkable object detection accuracy and operational speed. Compared to traditional feature-based methods, which use manual extraction of features done by experts, DL methods can automatically learn feature characteristics from huge volumes of data over time [2].

Currently, many object detection models are based on convolutional neural networks (CNNs) due to their rich representation power [6]. The visual recognition performed by the CNN feature extraction is close to the visual mechanism of human beings [2]. A typical CNN contains different types of layers, e.g., convolutional, pooling, fully connected, etc., whereby each layer transforms the 3D input volume into a 3D output volume of neuron activations [1]. Different CNN architectures have been elaborated to date. Among these, the Region-based CNN (R-CNN) was the first one that successfully has applied DL for object detection and other computer vision tasks by realizing automatic extraction of image features. Recent advances in object detection are indeed driven by the success of R-CNNs, whose cost has been significantly reduced thanks to sharing convolutions across object proposals [7]. The original R-CNN model was followed by other incarnations, such as Fast R-CNN [8], Faster R-CNN [7], Mask R-CNN [9], and Mesh R-CNN[10]. All these are representatives of the *two-stage* object detection models, which first generate a series of sparse candidate frames (i.e., region proposals extracted from a scene), followed (in the second stage) by candidate frames verification, classification, and regression to improve the scores and locations [2]. Among the pros of these models are the high accuracy and localization achieved in object recognition, whereas the more complex training required and lower operational speed achieved are their main cons [2], especially if taking into account that *real-time* object detection is now playing an increasingly important role in practical applications. In this regard, members of the other group of *single-stage* object detection models, such as You Only Look Once (YOLO) [11] and Single Shot MultiBox Detector (SSD), perform better by directly adopting a regression method for object detection, resulting in higher operational speed. However, SSD does not consider the relationship between different scales, so it has limitations in detecting small objects, whereas for YOLO it is easier to learn general features, and its operational speed is higher [12]. Both SSD and YOLO, however, cannot perfectly handle the graphic area, resulting in high detection error- and missing rates.

In 2017, Google came up with a new idea to completely replace the traditional CNN and recurrent neural network (RNN) structures with an attention structure, which was subsequently widely adopted for use in different fields. With the attention mechanism utilized for object detection, new weights are added to images for marking, learning, and training so as to achieve the effect of paying attention to more important areas, and solve the problem of information resources allocation.

Regarding the object classification, a model which is optimal for this task may not be optimal for object detection. As pointed out in [13], in contrast to a classifier, a detector requires: (i) a larger size of the input network (i.e., a higher resolution) for detecting multiple objects of small sizes; (ii) more network layers for increasing the receptive field and covering the larger size of the input network; and (iii) more parameters for improving the model performance in detecting multiple objects of different sizes in a single image.

The objective of this paper is to come up with a novel model, called YOLOv4_AF, based on a YOLOv4 optimization, as to achieve better balance between the two tasks (object detection and object classification) and get better performance results for vision-based vehicle detection and classification, compared to the state-of-the-art models used for the same purpose. For this, YOLOv4_AF utilizes an attention mechanism to suppress interference features of photos / video frames through both channel dimension and spatial dimension. In addition, a modification of the Feature Pyramid Network (FPN) part of the Path Aggregation Network (PAN), used in YOLOv4, is proposed and implemented, followed by a maximum pooling for feature fusion. The novelty of the proposed model is in the combined use of these new ideas, incorporated into the original YOLOv4 model, allowing to achieve better performance results for vehicle detection and classification, as presented further in the paper.

The rest of the paper is organized as follows. The next section presents the background information, followed by the related work considered in Section III. Section IV describes the proposed YOLOv4_AF model, followed by the experimental Section V. Finally, Section VI concludes the paper.

II. BACKGROUND

A. CONVOLUTION

The physical meaning of the convolution process, used in CNNs, can be generally summarized as follows. The output of a system at a given point of time is generated through a joint effect (superimposition) of multiple inputs. In LeNet-5, an early CNN proposed by LeCun *et al.* in [14], the original image can be gradually converted into a series of feature maps through alternately connected convolutional layers and down-sampling layers, which allows objects, presented in the image, to be easily classified according to these maps. Later, through gradual improvement and development, AlexNet [15], GoogLeNet [16], [17], ResNet [18], VGG-16 [19], etc. emerged, whereby various convolutional models can be selected independently. The

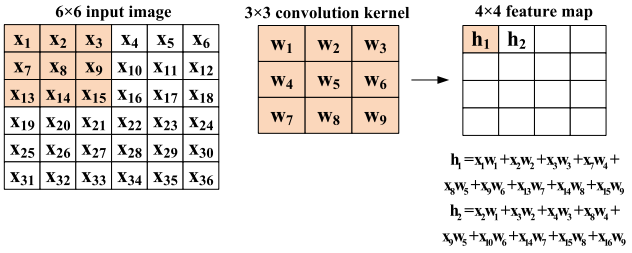

FIGURE 1. A 3 × 3 convolution computation example.

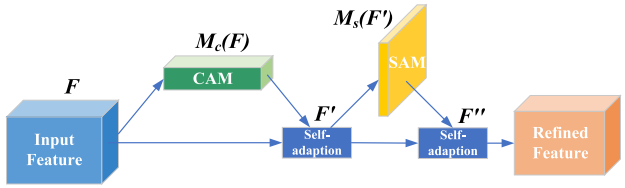

FIGURE 2. The structure of the convolutional block attention module (CBAM), utilized by the proposed YOLOv4_AF model.

purpose of the convolution process is to convert the input image into a matrix according to the corresponding pixel value. For instance, if a 6 × 6 pixels image is considered for convolution with a 3 × 3 convolution kernel and a step size of 1, then a 4 × 4 feature map is formed as shown in Figure 1.

In reality, the data volume is relatively large, and the computing workload of convolution is also quite heavy, so some pre-processing is required before convolution computation, e.g., compressing the output of the previous layer as to lessen the workload, instead of causing too many losses. For instance, with a Region Proposal Network (RPN), the Faster R-CNN model, described in Subsection III.A.2, inputs the convolutional features into the RPN to obtain candidate frame information, thus replacing the original R-CNN's selective search and, by this, improving the frame detection speed. Another approach, e.g., utilized by YOLO as described in Subsection III.B, uses just one convolution to achieve faster processing, which is suitable for real-time applications.

B. ATTENTION

The attention mechanism originates from the research conducted on human vision. In information processing, people selectively focus on part of the information and ignore the rest due to the effect of cognitive processing. Called ''attention'', this mechanism performs two main functions [20], i.e., deciding on: (i) which part(s) of the input the attention needs to be paid to; and (ii) which important parts of the input the (limited) information processing resources should be allocated to. According to the characteristics of the attention effect, the attention mechanisms could be divided into term-based and location-based ones. Based on the domain, the attention mechanisms could be classified into spatial, channel, and mixed domain attention mechanisms. Overall, attention mechanisms have shown significant potential in the area of object detection due to their intuitiveness, versatility, and interpretability [21].

The YOLOv4_AF model, proposed in this paper, utilizes the lightweight general-purpose convolutional block attention module (CBAM), proposed in [6], which enables the network to notice the informative parts of the images [22]. By combining a channel-wise and spatial-wise attention, CBAM can achieve better detection results than the one-type attention mechanisms, such as the channel-attention based Senet [23]. This is because the spatial-wise attention focuses on 'where' the informative part is, whereas the channel-wise attention focuses on 'what' is meaningful in the input image [6]. For this, CBAM utilizes two modules –a channel attention module (CAM) and a spatial attention module (SAM)–, which could be placed either in parallel or sequentially. However, the sequential arrangement outperforms the parallel one. For the former, the CAM-first order (shown on Figure 2) is slightly better than the SAM-first order, as proved in [6], and thus it was utilized in the proposed YOLOv4_AF model. CAM utilizes different feature information, whereby each channel of the features represents a special detector. After adding the features together, the activation function gets their weights and figures out what features are meaningful. SAM is spliced with the channel descriptions, obtained through average pooling and maximum pooling, and processed by the convolutional layer to obtain the weight coefficients.

In general, CBAM is easy to use, can be integrated into any CNN architecture in a seamless way, can be trained on an end-to-end basis, and its overall overhead is quite small both in terms of the computation time and the number of parameters. In addition, it shows great potential for use on low-end devices [6].

C. FEATURE FUSION

In object detection, an important means for improving the segmentation performance is to fuse features of different scales. The resolution and information of different layers are different. The lower-layer features undergo less convolution, resulting in lower semantics but higher resolution which can provide more location information. The result is opposite for the upper-layer features, where the lower resolution leads to poor perception of detailed information. The object detection performance of a model could be improved through fusion of multi-layer features [24]. Depending on which one (fusion or prediction) takes place before the other, fusion is divided into two types – early fusion and late fusion. Early fusion includes classic feature fusion methods such as *concatenation*, *addition*, etc., whereas late fusion combines the detection results of different layers. For instance, FPN [25], [26] uses both high-resolution and high-semantic information to achieve better results by fusing these different lower-layer and higher-level features. For this, FPN first performs pyramid fusion followed by prediction performed separately on each fused

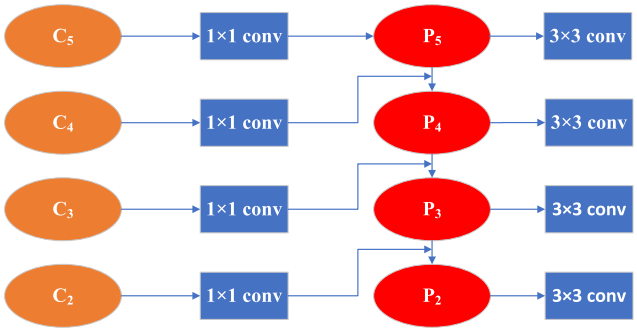

FIGURE 3. The FPN structure, part of the PAN utilized by YOLOv4.

feature layer. FPN conveys strong semantic features from top to bottom, and combines upper-layer feature information through up-sampling to obtain the prediction map.

As the conventional top-down FPN is limited by the one-way information flow [27], in YOLOv4, described in Subsection III.B.4, after FPN, an extra bottom-up pathway aggregation is added, resulting in the so-called Path Aggregation Network (PAN) [28]. The FPN structure, utilized by YOLOv4, is shown in Figure 3, where C_i represents the i^{th} ResNet convolution group (i.e., the i^{th} feature level with resolution of $1/2^i$ of the input image) and P_i represents the i^{th} feature map obtained after a 1×1 convolution of C_i. Fusion with the up-sampled feature maps is used to obtain the new feature maps P_4, P_3, and P_2 from the corresponding features of C_4, C_3, and C_2, respectively. After the addition, a 3×3 convolution is used by the fusion to generate the final feature map.

III. RELATED WORK

In general, each object detector consists of five parts [29]: an input, a backbone, a neck, a head, and an output. The (pre-trained) *backbone* could be [13]: (i) VGG-Nets [30], ResNet [18], ResNeXt [31], DenseNet [32], etc., for the GPU-based detectors; and (ii) SqueezeNet [33], MobileNet [34] [35], ShuffleNet [36], [37], etc., for the central processing unit (CPU)-based detectors. Composed of several bottom-up and top-down paths, the *neck* is usually used to collect feature maps from different stages [13]. Examples of networks with such mechanism include FPN [25], [26], PAN [28], BiFPN [27], and NAS-FPN [38]. The *head* part, used to predict classes and BBoxes of objects [13], may comprise either one stage or two stages. Correspondingly, the object detection models that adopt CNNs are divided into two main groups [12]:

1) *Two-stage models*, mostly represented by the R-CNN series, which extract and reposition the candidate frame and generate detection output through independent CNN channels. Having demonstrated an exceptional precision, these models, however, have much lower operational speed compared to that of the one-stage models, and thus cannot meet the requirements for real-time operation [12].
2) *One-stage models*, mostly represented by SSD [39] and different versions of YOLO, which skip the process of generating the selected area through the candidate framework and directly generate the category probability and location coordinate value of the object to be detected, identified, and classified, which increases their operational speed despite the slight flaw in precision. In addition, these models are easier to optimize [2].

The main (anchor-based) representatives of these two groups are briefly described in the subsections below.

A. R-CNN

R-CNN achieves excellent object detection accuracy by using deep CNNs to classify object locations, a.k.a. ''object proposals'' [7]. For this, it trains CNNs end-to-end to classify the proposal regions into object categories or background. Its accuracy depends on the performance of the region proposal method (adopted as an external module performing a selective search [7]), used to extract regions of interest (RoIs). Generally, a complex, precisely tuned, and very slow pipeline is used, whereby first the selective search generates potential BBoxes, then a CNN extracts features, an SVM scores the BBoxes, a linear model adjusts BBoxes, and finally a non-max suppression eliminates duplicate detections [11]. In addition, if images of different size are used, a scaling process must be applied.

In general, as pointed out in [7], R-CNN suffers from remarkable drawbacks, such as slow object detection (due to performing a CNN forward pass for each object location, without sharing computation) and expensive (both in space and time) multi-stage pipeline training. The emerged incarnations of R-CNN are described briefly in the following subsections.

1) FAST R-CNN

Fast R-CNN was proposed by Ross Girshick in 2015 for object detection [8]. Compared to the regular R-CNN which independently computes the output features on each RoI, Fast R-CNN runs the CNN only once on the entire image. A single-stage training is utilized, using a multi-task loss, which can update all network layers, and jointly learns to classify object proposals and refine their spatial locations. In addition, for feature caching, no disk storage is needed.

Firstly, Fast R-CNN inputs the image and multiple RoIs into a fully convolutional network. Then, each RoI is pooled into a fixed-size feature map to produce a feature vector by fully connected layers. There are two output vectors per RoI: *softmax* probabilities and per-class BBox regression offsets (Figure 4).

Compared to R-CNN, Fast R-CNN circumvents the redundant feature extraction operation and performs feature extraction only once for the whole image's full region, combining a RoI pooling layer and introducing suggestion frame information to extract the corresponding suggestion frame features. Similarly to R-CNN, Fast R-CNN utilizes

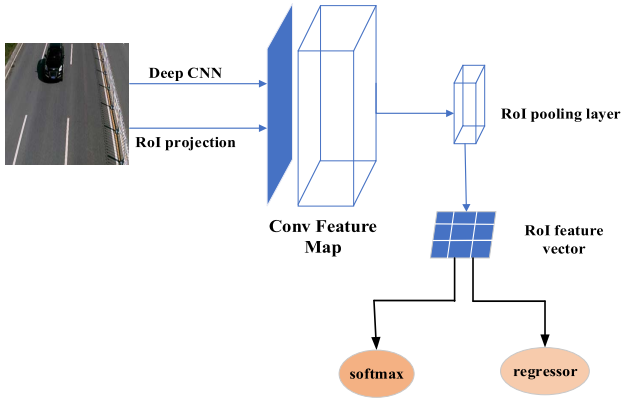

FIGURE 4. The Fast R-CNN model.

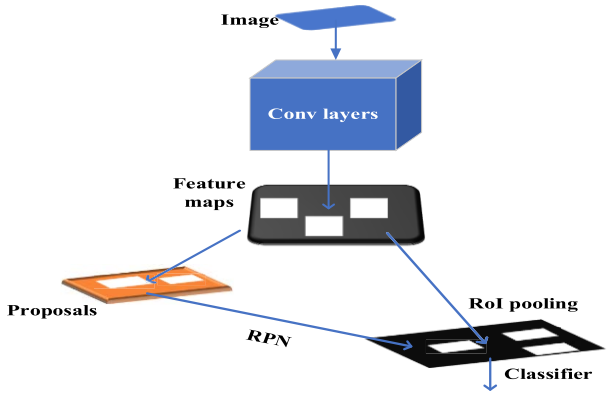

FIGURE 5. The Faster R-CNN model.

the very deep VGG-16 network, but trains it 9 times faster. Moreover, it is 213 times faster than R-CNN in testing and achieves higher mean average precision (*mAP*) on PASCAL VOC 2012 [8]. Having reduced the running time of the detection network, Fast R-CNN has exposed the region proposal computation as a main speed bottleneck [7].

2) FASTER R-CNN

Faster R-CNN was proposed by Ren et. al in 2017 [7]. It replaces the selective search, used in R-CNN and Fast R-CNN, with a neural network to propose BBoxes [11]. By introducing a RPN, which shares full-image convolutional features with the detection network, it enables nearly cost-free region proposals. Working on an input image, RPN outputs a set of rectangular object proposals, each with an objectiveness score. RPN is trained end-to-end (by back-propagation and stochastic gradient descent) to generate high-quality region proposals (with a wide range of scales and aspect ratios), which are then used by Fast R-CNN for object detection. The main training scheme alternates between fine-tuning for the region proposal task and fine-tuning for object detection, while keeping the proposals fixed, which allows the scheme to converge quickly. In addition, RPN and Fast R-CNN are merged into a single network by sharing their convolutional features with attention mechanisms, whereby RPN tells the unified network where to look. More specifically, CNN and the maximum pooling layer are firstly used to extract input image features and obtain the feature map. For each candidate region, the pooling layer in the RoI extracts feature vectors with fixed lengths for operations and sends them to the multi-classifier for classification. Finally, a BBox regression is performed, the offset value of the graph frame coordinates is obtained, and the sample frame is modified to classify the region, as shown in Figure 5. Recently, Wang *et al.* have optimized the Faster R-CNN structure and improved the model performance by adding a FPN [40].

The model performs well when trained and tested using single-scale images, which also benefits the operational speed. Overall, Faster R-CNN is not only a cost-efficient model, but also presents an effective way for improving the accuracy of object detection [7]. It is the current leading model used in several benchmarks [9]. Thus, it was selected as the main representative of the R-CNN group for performance comparison with the YOLOv4_AF model proposed in this paper.

3) MASK R-CNN

Mask R-CNN [9] extends Faster R-CNN by adding a branch for predicting segmentation masks on each RoI, in parallel with the existing branch for classification and BBox regression. The mask branch is a small fully CNN applied to each RoI, predicting a segmentation mask in a pixel-to-pixel manner.

Basically, Mask R-CNN combines elements of object detection (i.e., identifying individual objects and localizing each of them using a BBox) and semantic segmentation (i.e., classifying each pixel into a fixed set of categories without differentiating object instances) [9]. Firstly, Mask R-CNN uses a CNN to extract the features of the input image through a RPN and gets several feature maps of different size (Figure 6). From these maps, it obtains the candidate frames RoIs. Then, in order to solve the problem of misalignment between the mask and the object in the original image, it uses a quantization-free RoIAlign layer that faithfully preserves exact spatial locations, followed by a BBox regression, a mask branching on the RoIs, and RoIs generating. In general, Mask R-CNN is simple to train but adds a (small) overhead compared to Faster R-CNN. In addition it is not optimized for speed [9].

4) MESH R-CNN

To solve the problem of the 2D target detection ignoring the 3D spatial information, Facebook researchers proposed the Mesh R-CNN model [10], which can detect multiple objects in varying lighting conditions and different contexts. The Mesh R-CNN augments Mask R-CNN with a mesh prediction branch, which outputs meshes with a varying topological structure by first predicting coarse voxel representations that

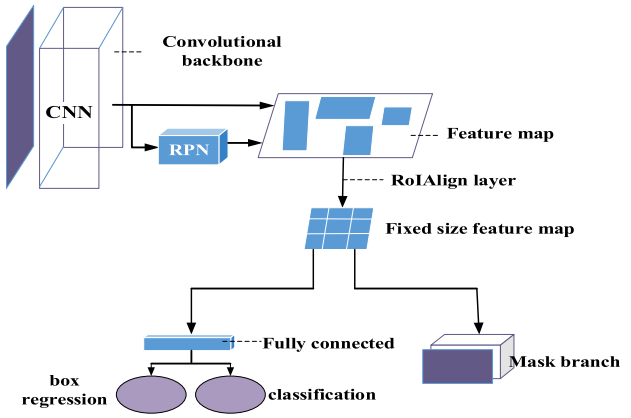

FIGURE 6. The Mask R-CNN model.

are converted into meshes and refined with a graph CNN operating over the mesh's vertices and edges [10]. This allows the model to detect multiple objects in the input image and to achieve fine prediction of arbitrary geometric structures and the corresponding 3D triangular mesh for each object, successfully extending the capability of the 2D perception to 3D target detection and shape prediction.

B. YOLO

YOLO is a family of models started out in 2016 by Joseph Redmon [11]. With its different versions, YOLO presents a new approach to object detection as it only needs to ''look'' once at an image to detect the objects and their locations on it. For this, instead of repurposing classifiers to perform detection, it frames object detection as a single regression problem to spatially separated BBoxes and associated class probabilities, which are predicted by a single CNN directly from the entire image in one step. YOLO trains on full images and directly optimizes its performance for object detection. Compared to traditional models, it provides the following advantages [11]: (i) a higher operational speed, which makes it suitable for use in real-time applications, such as video surveillance and live web cameras (all R-CNN representatives fall short of real-time performance); (ii) a global reasoning about the image when making predictions, resulting in less number of background errors (e.g., 50%+ less errors compared to Fast R-CNN); (iii) it is highly generalizable, which makes it less likely to break down when applied to new domains or unexpected inputs (for instance, when trained on natural images and tested on artwork, it outperforms R-CNN by a wide margin). However, in terms of detection precision, it is outperformed by R-CNN due to struggling to precisely localize small objects [11].

YOLO versions, briefly presented in the following subsections, are the most balanced object detectors in terms of detection accuracy and operational speed [2]. However, a new set of object detection models, called EfficientDet [27], has been recently proposed, utilizing a weighted bi-directional FPN in trying to achieve better accuracy and efficiency [2]. That is why EfficientDet was also included in the performance compassion of models, presented in Section V.

1) YOLOv1

The first version of YOLO [11] achieves good detection speed by dividing a single image into multiple grids, and classifying and retrieving each grid [12]. It uses a CNN architecture, similar to GoogLeNet, consisting of 24 convolutional layers followed by two fully connected layers. But instead of the GoogLeNet inception modules, it uses 1×1 reduction layers followed by 3×3 convolutional layers. There is also a fast version of YOLOv1, called Fast YOLO, which uses just 9 instead of 24 convolutional layers and fewer filters in those layers, with the same training and testing parameters. Compared to other models used before it for real-time object detection, Fast YOLO is more than twice as accurate [11]. However, YOLOv1 has limitations in the training data size, the input image needs to be adjusted to the size of $448 \times 448 \times 3$ before being processed by the CNN, and finally the results need to be thresholded by the confidence of the model. In addition, [11] points out to other limitations of YOLOv1, such as: (i) strong spatial constraints on BBox predictions, limiting the number of nearby objects that can be predicted, e.g., small objects appearing in groups, such as flocks of birds; (ii) struggling to generalize to objects in new or unusual aspect ratios and/or configurations; (iii) using relatively coarse features for predicting BBoxes; (iv) incorrect object localizations (e.g., compared to Fast R-CNN, YOLO makes a significant number of localization errors [41]) due to using a loss function that treats errors the same way in small and large BBoxes. Moreover, YOLO has relatively low recall, e.g., compared to R-CNN [41].

2) YOLOv2

Compared to YOLOv1, YOLOv2 [41] is: (i) better, in terms of accuracy (outperforming also Faster R-CNN); (ii) faster (outperforming also Faster R-CNN); and (iii) stronger (more than 9000 different object categories can be detected in real-time). In addition, it can run at varying sizes, offering an easy tradeoff between speed and accuracy. The YOLOv2 variation, YOLO9000, can jointly train on object detection and classification, which allows it to predict detections for object classes that do not have labelled detection data. The objects are classified using a hierarchical view, allowing to combine distinct data sets together, whereby a detection data set could be expanded with a huge amount of data from a classification data set. Compared to YOLOv1, the following improvements have been made in YOLOv2, as pointed out in [41]: (i) a batch normalization on all convolutional layers (resulting in significant improvement in convergence and 2%+ improvement in *mAP*); (ii) a high-resolution classifier (leading to an increase of almost 4% of *mAP*); (iii) removing the fully connected layers and using convolutional with anchor boxes to predict BBoxes (resulting in a *mAP* decrease, but increasing the recall); (iv) dimension clusters; (v) direct location prediction (making the parametrization easier to

learn and the network more stable); (vi) fine-grained features (resulting in 1% performance increase); (vii) multi-scale training (allowing the same network to predict detections at different resolutions), etc.

3) YOLOv3

YOLOv3 [42] introduced different small changes in the design of YOLOv2 to improve it. Compared to YOLOv2, it is a little bigger but more accurate. While still predicting BBoxes using dimension clusters as anchor boxes, it predicts an objectiveness score for each BBox using logistic regression. In addition, instead of *softmax*, YOLOv3 uses independent logistic classifiers. During training, it utilizes a binary cross-entropy loss for the class predictions. It predicts boxes at three different scales, whereby features from these scales are extracted using a similar concept to FPN. For feature extraction, it uses a new neural network, called DarkNet53 –a hybrid between the YOLOv2 network, DarkNet19, and an innovative residual network–, which better utilizes the GPU, making it more efficient and faster. Overall, YOLOv3 improves the object detection precision while maintaining the operational speed. At the same time, it has got an enhanced ability to identify small objects, thanks to the utilized multi-scale prediction, but comparatively worse performance on medium- and large-size objects. In addition, the speed and precision can be traded off by changing the size of the model. However, as the intersection over union (IoU) threshold increases, the model's performance decreases and YOLOv3 does not fit well with the ground truth [12]. For sonar data sets with small effective samples and low signal-to-noise ratios, an improved YOLOv3 algorithm, named YOLOv3-DPFIN, was proposed in [12] for accurate real-time detection (e.g., in underwater applications) of noise-intensive multi-category sonar objects/targets with minimum time consumption, achieving higher precision and speed than the original YOLOv3 model.

4) YOLOv4

Even though the overall structure of YOLOv4 [13] is similar to that of YOLOv3 (both versions use the same *head*), its detection precision is significantly higher, as demonstrated by Yang *et al.* in [43]. Recently, Jang *et al.* [44] have conducted a new study on the utilization of GPU resources, yielding best performance for YOLOv4, having dealt also with the correlation between the GPU and CPU.

YOLOv4 is not only better than YOLOv3 in terms of operational speed and accuracy, but also easier to implement. In addition, it exhibits a strong real-time detection ability, and can operate on a single conventional GPU, which is sufficient also for its training [29]. The *backbone* of YOLOv4 adopts the CSPDarkNet53 neural network [45]. During sampling, 3×3 convolutional layers (29 in total) and a 725×725 receptive field are used. In the *neck*, a Spatial Pyramid Pooling (SPP) module [46] is used to increase the receptive field and separate out the most significant context features, with almost no speed reduction. In addition, for parameter aggregation from different backbone levels, the *neck* utilizes a PAN instead of the FPN used in YOLOv3. In fact, the PAN adds an extra bottom-up pathway aggregation on top of the FPN [27]. The loss function in YOLOv4 is also changed, namely to a complete IoU (CIoU) loss [47], which considers simultaneously and in a more comprehensive manner the three important geometric factors, i.e., the overlap area, the distance between central points, and the aspect ratio, thus allowing to describe better the BBox regression, and achieve faster convergence speed and better regression accuracy. In addition, it also makes the detection model friendlier to small objects. The CIoU loss is formulated in [47] as:

$$CIoU = 1 - IoU + \frac{\rho^2(d, d^r)}{c^2} + f.v, \tag{1}$$

where f measures the consistency of the aspect ratio between the prediction box and groundtruth, and v is a trade-off parameter used to balance the scale, defined respectively as:

$$f = \frac{4}{\pi^2}\left(arctan\frac{w^r}{h^r} - arctan\frac{w}{h}\right)^2; \tag{2}$$

$$v = \frac{f}{(1 - IoU) + f}, \tag{3}$$

where w^r and h^r denote the width and height of the real frame, w and h denote the width and height of the predicted frame, $\rho^2(d, d^r)$ denotes the Euclidean distance between the predicted frame and real frame, and c denotes the diagonal distance of the minimum closure area between the predicted frame and real frame. If the width and height of the real frame are similar to those of the predicted frame, then $v = 0$, and the penalty term produces no effect.

The YOLOv4 structure is shown in Figure 7. Convolution (Conv), Batch normalization (Bn), and Mish activation function constitute the smallest component, denoted as CBM. Conv, Bn and Leaky_ReLU activation function form another component, denoted as CBL. The Cross-Stage Partial connections (CSP) component, consists of three convolutional layers and ResNet unit modules. All convolution kernels in front of the CSP are of size 3×3, equivalent to down-sampling [48]. SPP adopts the pooling operation of fixed blocks. The maximum pooling for the blocks with a kernel's size of 1×1, 5×5, 9×9, and 13×13 refers to a series of *concat* operations, tensor splicing, dimension expansion, and finally outputting.

5) YOLOv5

Shortly after the appearance of YOLOv4, Ultralytics released YOLOv5, but it did not get the approval of the authors of the original YOLO model. To date, its ability remains controversial even though some of the initially provided (incorrect) performance evaluations were later corrected. Even though YOLOv5 trains faster, YOLOv4 can be optimized to achieve higher processing speed. So, the Darknet-based YOLOv4 is still the most accurate YOLO version, especially if a computer-vision engineer is in pursuit of state-of-the-art results and is capable of performing additional customization

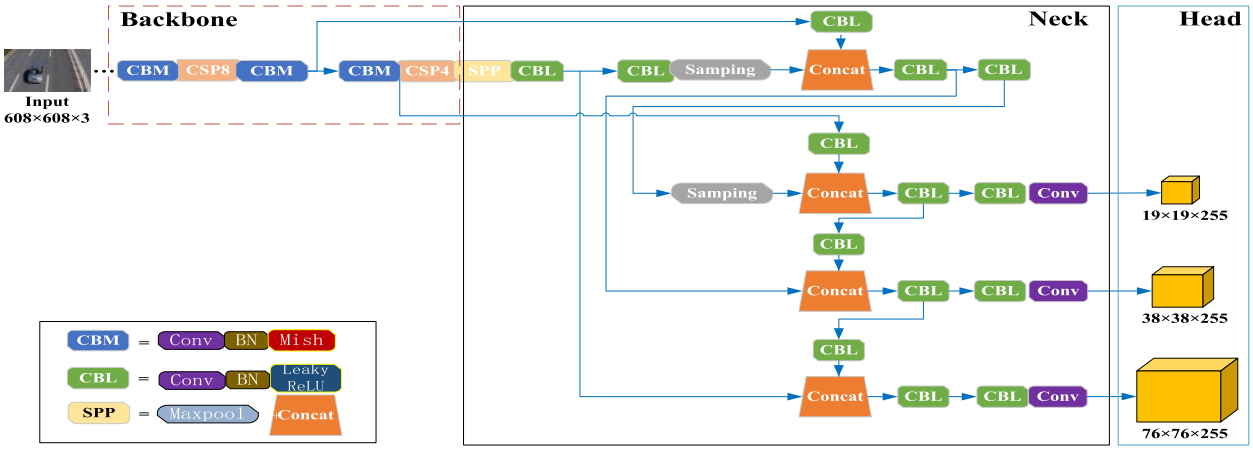

FIGURE 7. The YOLOv4 model.

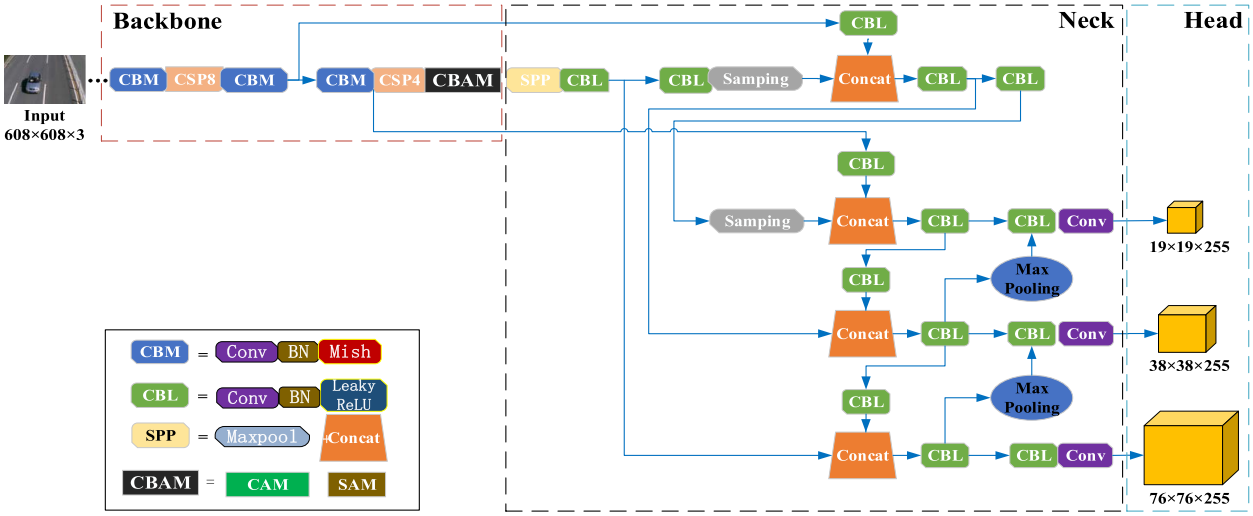

FIGURE 8. The proposed YOLOv4_AF model.

on the model [49]. That is why YOLOv4 was selected as the main YOLO representative for the performance comparison of models, presented in Section V.

IV. PROPOSED YOLOv4_AF MODEL

When the detection target's background is highly diversified and there are many kinds of objects in the image, the fast identification of the target object is the first problem to be solved. The spatial attention mechanism adopted by YOLOv4 for extracting the feature map information mainly focuses on the most informative part of the image, ignoring the extraction of the whole information during the image input. To enrich this operation, we propose to add a CBAM module to YOLOv4 for realizing a bidirectional attention on the global and local feature information. The introduction of a CBAM module in forward-propagation (Figure 8), allows to expand further the receptive field through channel training during the neural network transmission process and in combination with different weight coefficients to display the part that the detection network should pay more attention to.

In addition, we propose to select high-level features in the FPN part of the PAN, utilized by YOLOv4, and modify it accordingly, in order to improve the detection and classification performance of the proposed YOLOv4_AF model. To achieve this, we replace the residual connections in the network and insert additional *neck* layers as to better extract fusion features.

So, the novelty of the proposed model is in improving further the object detection and classification performance of YOLOv4 without deepening the neural network, by applying an appropriate optimization to it, as detailed further in the next subsections.

A. USING A CBAM

For automatic learning of the places needing attention in the images and improving the intensity of feature expression of each channel, the proposed YOLOv4_AF model utilizes a CBAM module, as shown in Figure 8, which allows to increase the influencing factors of channel features and balance the interaction of the three dimensions of length, width, and height.

CBAM combines the channel and spatial dimensions to suppress the interference characteristics and compensate the

information loss in the global average pooling, and then incorporates an attention mechanism. First, a channel-wise attention mechanism directly averagely pools the information in the channel, with a wide processing range but rough comparisons. Then, a spatial-wise attention mechanism processes the feature maps in the channels during the original image feature extraction stage.

The main CBAM computation is performed as per [6] (c.f. Figure 2):

$$F' = M_C(F) \otimes F \tag{4}$$

$$F'' = M_S(F') \otimes F', \tag{5}$$

where $\otimes$ denotes an element-wise multiplication, F denotes the input feature map, $M_C(F)$ denotes the channel attention map, F' denotes the channel-refined feature, $M_S(F')$ denotes the spatial attention map, and F'' denotes the final refined output.

In CBAM, the attention operation is performed in the channel dimension first. Given a convolution kernel $Y = [y_1, y_2, \cdots, y_i, \ldots]$, where y_i represents its i^{th} parameter, the global average pooling operation F_{avg} and the global maximum pooling operation F_{max} can be expressed as follows:

$$F_{avg}(y_i) = \frac{1}{H \times W} \sum_{k=1}^{H} \sum_{j=1}^{W} y_i(k, j); \tag{6}$$

$$F_{max}(y_i) = argmax\left(\frac{1}{H \times W} \sum_{k=1}^{H} \sum_{j=1}^{W} y_i(k, j)\right), \tag{7}$$

where H and W denote the length and width of the feature map, respectively, and (k, j) denotes the points on the feature graph of size $H \times W$, whose transverse and vertical coordinates are k and j, respectively.

$F_{avg}(y_i)$ and $F_{max}(y_i)$ are inputted into the first fully connected layer for dimension reduction. Then, the result, obtained by a *Leaky_ReLU* function, is inputted into the next fully connected layer, and the final result is outputted by a *sigmoid* function. The final output is formed by summing the two corresponding ouptputs together:

$$output_c = output_{avg} + output_{max}. \tag{8}$$

Next, the feature weighting operation is conducted through matrix multiplication and the channel feature $P = [p_1, p_2, \ldots, p_i, \ldots]$ is obtained as follows:

$$P = y_i \times output_c. \tag{9}$$

Further on, P is inputted into the SAM part of CBAM for spatial attention extraction by means of global maximum pooling and global average pooling performed on the feature maps of all channels. Then, dimensions are reduced through convolution. Finally, the output and input feature dimensions are made consistent for the whole CBAM module.

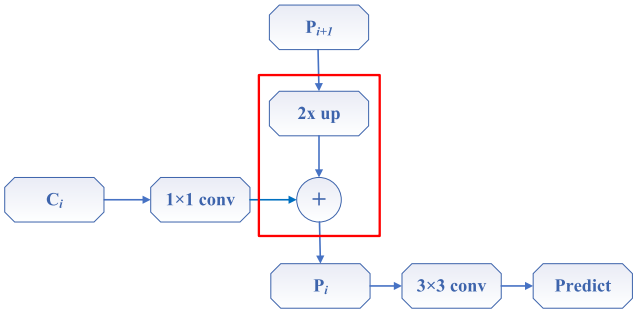

FIGURE 9. The side link schematic of the FPN part of the PAN, utilized by YOLOv4.

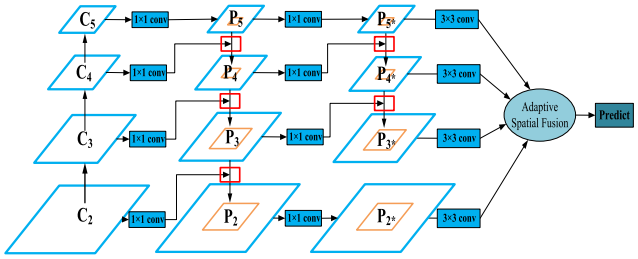

FIGURE 10. The proposed FPNi structure, part of the PAN utilized by YOLOv4_AF.

B. USING A FPNi

The stronger the semantic information, the more beneficial it is for the target object recognition. Here we propose a modification, named FPNi, to the FPN part of the PAN utilized by YOLOv4. In FPNi, the double route is expanded to a triple route with double side connections, so as to improve the performance of object detection and classification by combined operations.

In the FPN part of the PAN, utilized by the original YOLOv4 model, the bottom-top CNN and the top-bottom part are connected by the side, as shown in Figure 9. A 1×1 convolution operation is performed on the feature maps C_i obtained by different ResNet convolution groups and the result is fused with the up-sampled feature map P_{i+1} to obtain a new feature map P_i, which has the same size as the lower-layer feature map (the $\oplus$ sign in Figure 9 represents an element-wise addition operation of feature vectors).

In order to solve the problem of multi-scale variation in YOLOv4, we propose a FPNi structure, shown in Figure 10 (where the red rectangles have the same structure as the red rectangle in Figure 9), which is utilized by the YOLOv4_AF model presented in this paper. In FPNi, there are additional fusion operations performed on P_3 and P_4. The flow of feature map information is strengthened by the side linking of high-level semantics, and the functioning of the small target feature extraction is improved. At the final stage, after the regular 3×3 convolutional operations, an adaptive spatial fusion is performed before prediction.

Thanks to the FPNi modification, the overlap effect during the up-sampling process is eliminated and the outputted

feature maps combine the features of different layers and thus have richer information.

Maximum pooling is adopted during the down-sampling process. The input images are classified into several rectangular areas. The maximum value for each sub-area is outputted. The pooling layer imitates the human visual structure to reduce dimensions, abstract the data, and reduce the size of the image so as to match the size of the display area. This improves the robustness of the feature map, avoids overfitting, and retains more texture information. Through maximum pooling, the new image dimensions are calculated as follows:

$$W_2 = \frac{W_1 - F_w}{S} + 1; \tag{10}$$

$$H_2 = \frac{H_1 - F_h}{S} + 1; \tag{11}$$

$$D_2 = D_1, \tag{12}$$

where W_1, H_1, and D_1 denote the width, height, and number of channels of the input image, respectively, F_w and F_h denote the filter's width and height, and S denotes the *stride*.

With the proposed FPNi design, the FPN operation is no longer limited by one-way information flow, and the high-level information with strong semantic can flow down through side-to-side connections, thus achieving better results than the original FPN design utilized by YOLOv4. However, the FPNi design increases the computational effort of the proposed YOLOv4_AF model, compared to that of YOLOv4, which has negative impact on the model's operational speed.

V. EXPERIMENTS

A. DATA SETS

Two public data sets, BIT-Vehicle [50] and UA-DETRAC [51], were used in the experiments. BIT-Vehicle consists of 9850 high-quality photos, containing frontal views of vehicles, collected under different conditions, such as bad weather, lighting, background confusion, etc. Six different classes of vehicles are included in the data set, namely Bus, Microbus, Minivan, SUV, Sedan, and Truck. The spatial resolution of photos is either 1920 × 1080 or 1600 × 1200 pixels. Each photo contains the image of one or more vehicles. In total, the photos contain the images of 558 buses, 883 microbuses, 476 minivans, 1392 SUVs, 5919 sedans, and 823 trucks. Some sample photos are shown in Figure 11.

The UA-DETRAC data set includes 10 hours of road monitoring videos made at 24 different locations at Beijing and Tianjin in China, totaling in more than 140,000 video frames with resolution of 960 × 540 pixels, containing 8,250 (different) vehicles divided into four classes – Bus, Van, Car, and Others. As most of the video frames, following one another, contain the same vehicles passing through the cameras, only 20,000 frames containing as much as possible different vehicles were used in the experiments.

A five-fold cross-validation was used. Accordingly, each data set was randomly split into five folds of equal size, resulting in 1970 photos for BIT-Vehicle and 4,000 video frames for UA-DETRAC. In each iteration, one of the five folds was used as a test set, whereas the remaining four folds were used for training the models. Five iterations were conducted in total on each data set to ensure each fold was tested. The obtained (averaged) results are reported in Subsection V.C.

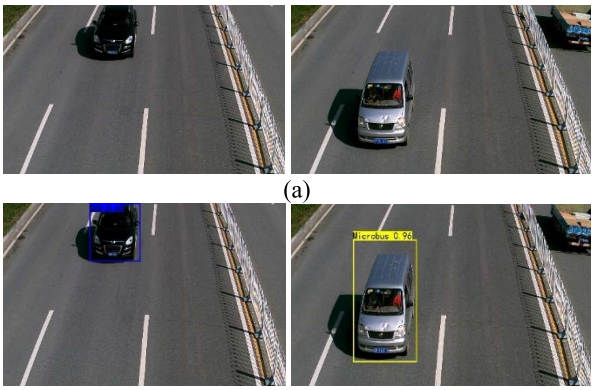

FIGURE 11. Sample photos from the BIT-Vehicle data set: (a) original photos; (b) photos, processed by the proposed YOLOv4_AF model, showing the detected vehicle's BBox with the predicted confidence score and class label.

B. METRICS

In the conducted experiments, the performance of the proposed YOLOv4_AF model was evaluated and compared to that of three state-of-the-art models, namely Faster R-CNN, YOLOv4, and EfficientDet, based on precision and recall, which are commonly used metrics for detection and classification problems. Precision is used to indicate the proportion of true positive (TP) samples in the prediction results, whereas recall is used to indicate the proportion of correct predictions in all positive samples, as follows:

$$precision = \frac{TP}{TP + FP}; \tag{13}$$

$$recall = \frac{TP}{TP + FN}, \tag{14}$$

where TP represents the number of samples that are actually positive and are classified as positive, FP (false positive) represents the number of samples that are incorrectly classified as positive, i.e., the number of samples that are actually negative but are classified as positive, and FN (false negative) represents the number of samples that are actually positive but are classified as negative.

For the performance evaluation of the compared models, the *F1 score* and mean average precision were used as the main metrics because they take into account both precision and recall.

The *F1 score* is defined as follows:

$$F1 = (2 * precision * recall)/(precision + recall). \tag{15}$$

TABLE 1. Average precision of faster R-CNN on BIT-Vehicle.

Classes	1st fold	2nd fold	3rd fold	4th fold	5th fold
Bus	97.60%	92.54%	90.83%	92.00%	92.29%
Microbus	65.91%	63.98%	63.62%	60.75%	67.55%
Minivan	45.76%	41.25%	48.71%	50.15%	51.50%
SUV	50.20%	45.83%	51.58%	49.64%	52.87%
Sedan	96.04%	94.82%	96.13%	95.12%	95.98%
Truck	89.22%	90.85%	88.42%	94.13%	89.81%

TABLE 2. Average precision of YOLOv4 on BIT-Vehicle.

Classes	1st fold	2nd fold	3rd fold	4th fold	5th fold
Bus	97.36%	96.80%	94.22%	96.32%	92.26%
Microbus	46.76%	50.57%	43.09%	47.03%	37.15%
Minivan	52.20%	43.37%	38.02%	38.02%	52.74%
SUV	82.67%	85.41%	82.86%	86.72%	88.53%
Sedan	97.45%	97.01%	96.91%	97.24%	97.45%
Truck	98.92%	96.08%	95.83%	98.01%	97.00%

TABLE 3. Average precision of EfficientDet on BIT-Vehicle.

Classes	1st fold	2nd fold	3rd fold	4th fold	5th fold
Bus	91.11%	100.00%	100.00%	100.00%	100.00%
Microbus	85.80%	76.35%	85.67%	73.36%	82.10%
Minivan	81.13%	63.35%	49.16%	62.42%	77.68%
SUV	82.36%	64.57%	82.92%	65.40%	83.57%
Sedan	99.01%	97.11%	97.97%	98.42%	98.67%
Truck	90.25%	70.75%	84.41%	71.92%	80.56%

TABLE 4. Average precision of YOLOv4_AF on BIT-Vehicle.

Classes	1st fold	2nd fold	3rd fold	4th fold	5th fold
Bus	90.33%	87.16%	86,16%	98.96%	95.39%
Microbus	64.05%	59.60%	69.40%	66.64%	65.90%
Minivan	60.02%	53.22%	70.89%	76.14%	66.61%
SUV	88.79%	87.51%	86.95%	76.24%	71.96%
Sedan	97.01%	97.16%	98.50%	97.04%	97.04%
Truck	94.36%	94.30%	97.44%	95.88%	97.77%

The mean average precision (*mAP*) is a quantitative metric for evaluating the effectiveness of multi-class object detection [12]. It can be calculated as follows:

$$mAP = \frac{\sum_{i=1}^{N} AP_i}{N}, \tag{16}$$

where AP_i denotes the average precision of class i and N denotes the total number of classes. The average precision (AP) corresponds to the area under the precision-recall curve, i.e.:

$$AP = \int_0^1 p(r)\, dr, \tag{17}$$

where $p(r)$ denotes the precision function of recall (r).

C. RESULTS

First, in the experiments, we used the BIT-Vehicle data set. The precision-recall curves were created for each of the compared models, for each of the six classes of vehicles, based on the values of recall and precision obtained from the conducted experiments for each fold. Then, these curves were used to calculate the average precision (AP) of each model for each class, separately for each fold, based on (17), as shown in Tables 1-4. Finally, in order to compare the overall performance of the models across all classes of vehicles, the mean average precision (*mAP*) values were calculated, based on (16), separately for each fold, and then averaged to obtain the final *mAP* result for the particular model, presented in Table 5. These results confirm that the proposed YOLOv4_AF model outperforms, in terms of *mAP*, all three state-of-the-art models used in the comparison. Faster R-CNN is the most outperformed (by 9.96 points), YOLOv4 is in the middle (outperformed by 5.65 points), and EfficientDet is the least outperformed (by only 0.25 points).

TABLE 5. Mean average precision (*mAP*) of compared models on BIT-Vehicle.

Model	1st fold	2nd fold	3rd fold	4th fold	5th fold	Final result
Faster R-CNN	74.12%	71.55%	73.15%	73.63%	75.00%	**73.49%**
YOLOv4	79.23%	78.21%	76.83%	77.23%	77.52%	**77.80%**
EfficientDet	88.28%	78.70%	83.35%	78.59%	87.10%	**83.20%**
YOLOv4_AF	82.42%	84.89%	82.35%	85.15%	82.44%	**83.45%**

Then the other metric, *F1 score*, was used for the performance comparison of the models. The obtained results, presented in Table 6, confirm that the proposed YOLOv4_AF model outperforms the three state-of-the-art models on this metric too. More precisely, Faster R-CNN is outperformed by 0.102 points, YOLOv4 is outperformed by 0.007 points, and EfficientDet is outperformed by 0.050 points.

Then, the UA-DETRAC data set was used for the comparison of the models. First, the average precision (AP) of each model for each class of vehicles was calculated, as presented in Tables 7-10. Then, the mean average precision (*mAP*) values were calculated, as shown in Table 11. The

TABLE 6. *F1-score* of compared models on BIT-Vehicle.

Model	1st fold	2nd fold	3rd fold	4th fold	5th fold	Final result
Faster R-CNN	0.719	0.697	0.714	0.720	0.722	**0.714**
YOLOv4	0.818	0.801	0.805	0.807	0.812	**0.809**
EfficientDet	0.824	0.758	0.816	0.702	0.728	**0.766**
YOLOv4_AF	0.810	0.805	0.812	0.835	0.820	**0.816**

TABLE 7. Average precision of faster R-CNN on UA-DETRAC.

Classes	1st fold	2nd fold	3rd fold	4th fold	5th fold
Bus	90.46%	86.72%	89.58%	89.18%	90.12%
Van	44.93%	43.22%	48.49%	49.65%	50.43%
Car	66.34%	66.43%	65.58%	67.31%	67.21%
Others	89.65%	88.62%	89.00%	86.22%	86.37%

TABLE 8. Average precision of YOLOv4 on UA-DETRAC.

Classes	1st fold	2nd fold	3rd fold	4th fold	5th fold
Bus	90.97%	85.68%	89.68%	90.18%	89.53%
Van	43.95%	42.92%	50.49%	51.68%	52.77%
Car	68.54%	69.53%	63.57%	67.57%	67.68%
Others	87.70%	89.00%	89.00%	86.42%	87.31%

TABLE 9. Average precision of EfficientDet on UA-DETRAC.

Classes	1st fold	2nd fold	3rd fold	4th fold	5th fold
Bus	92.78%	93.29%	92.90%	92.74%	92.90%
Van	87.17%	89.01%	86.53%	85.07%	83.40%
Car	77.11%	75.29%	74.71%	80.73%	78.88%
Others	48.87%	54.68%	52.90%	43.75%	44.11%

obtained results confirm that the proposed YOLOv4_AF model outperforms, in terms of *mAP*, all three state-of-the-art models on this data set too. More specifically, Faster R-CNN and YOLOv4 are outperformed by a similar degree of 4.30 and 3.87 points, respectively, whereas EfficientDet is outperformed by only 0.74 points.

Finally, the *F1 score* values were calculated, as presented in Table 12, and these results also confirm that the proposed YOLOv4_AF model outperforms here all three state-of-the-art models on this metric as well. More specifically, Faster R-CNN and YOLOv4 are outperformed by a similar degree

TABLE 10. Average precision of YOLOv4_AF on UA-DETRAC.

Classes	1st fold	2nd fold	3rd fold	4th fold	5th fold
Bus	90.10%	91.26%	90.33%	90.20%	90.28%
Van	57.42%	42.92%	60.71%	63.27%	67.73%
Car	63.65%	59.48%	64.60%	61.26%	65.38%
Others	91.21%	92.34%	92.65%	92.02%	92.36%

TABLE 11. Mean average precision (*mAP*) of compared models on UA-DETRAC.

Model	1st fold	2nd fold	3rd fold	4th fold	5th fold	Final result
Faster R-CNN	72.85%	71.25%	73.16%	73.09%	73.53%	**72.78%**
YOLOv4	72.79%	71.78%	73.19%	73.96%	74.32%	**73.21%**
Efficient Det	76.48%	78.07%	76.76%	75.57%	74.82%	**76.34%**
YOLOv4_AF	75.60%	77.05%	77.08%	76.74%	78.94%	**77.08%**

TABLE 12. *F1-score* of compared models on UA-DETRAC.

Model	1st fold	2nd fold	3rd fold	4th fold	5th fold	Final result
Faster R-CNN	0.763	0.754	0.721	0.773	0.732	**0.749**
YOLOv4	0.748	0.738	0.748	0.763	0.758	**0.751**
EfficientDet	0.808	0.805	0.800	0.765	0.773	**0.790**
YOLOv4_AF	0.803	0.803	0.813	0.804	0.817	**0.808**

of 0.059 and 0.057 points, respectively, whereas EfficientDet is outperformed by 0.018 points only.

Although the proposed YOLOv4_AF model outperforms all three state-of-the-art models on both metrics and both data sets, its classification of some vehicles was not completely accurate. For some photos (video frames) with high similarity, the classification result was slightly biased, resulting in less confidence score.

VI. CONCLUSION

In this paper, a more precise vehicle detection and classification model, based on YOLOv4 with applied additional optimization, has been presented. The main idea, incorporated into it, was to increase the receptive field in both channel and spatial dimensions by introducing an attention mechanism in the form of a CBAM module. In addition, in the FPN part, the feature fusion is modified, and an additional up-sampling operation is performed. Then, the output features are fused again, and the detection results of different layers are combined to improve the detection performance of the proposed model, called YOLOv4_AF. The performance of this model was experimentally evaluated and compared to that of the original YOLOv4 model and two

other state-of-the-art object detection models, Faster R-CNN and EfficientDet, based on two public data sets – BIT-Vehicle and UA-DETRAC. The obtained results clearly demonstrate that the proposed YOLOv4_AF model outperforms all three state-of-the-art models, used in the performance comparison, in terms of the mean average precision (*mAP*) and *F1 score*, on both data sets.

The elaborated model could be used also for detection of other types of objects, and can pave the way for improving the regression algorithms in general. However, due to the introduction of the CBAM module, compared to the original YOLOv4 model, the calculation complexity and time increase. In the future, we plan to carry out tracking of moving objects, conduct traffic statistics, and study the detection and classification ability of the proposed model on other objects.

REFERENCES

[1] K. F. Hussain, M. Afifi, and G. Moussa, "A comprehensive study of the effect of spatial resolution and color of digital images on vehicle classification," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 3, pp. 1181–1190, Mar. 2019.

[2] M. Haris and A. Glowacz, "Road object detection: A comparative study of deep learning-based algorithms," *Electronics*, vol. 10, no. 16, p. 1932, Aug. 2021.

[3] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Kauai, HI, USA, vol. 1, Dec. 2001, pp. I–I.

[4] W. T. Freeman and M. Roth, "Orientation histograms for hand gesture recognition," in *Proc. Int. Workshop Autom. Face Gesture Recognit.*, Zurich, Switzerland, 1995, pp. 296–301.

[5] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, San Diego, CA, USA, vol. 1, Jun. 2005, pp. 886–893.

[6] S. Woo, J. Park, J.-Y. Lee, and I.-S. Kweon, "CBAM: Convolutional block attention module," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 3–19.

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[8] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Santiago, Chile, Dec. 2015, pp. 1440–1448.

[9] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Venice, Italy, 2017, pp. 2980–2988.

[10] G. Gkioxari, J. Malik, and J. Johnson, "Mesh R-CNN," presented at the IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Seoul, South Korea, 2019.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 779–788.

[12] W. Kong, J. Hong, M. Jia, J. Yao, W. Cong, H. Hu, and H. Zhang, "YOLOv3-DPFIN: A dual-path feature fusion neural network for robust real-time sonar target detection," *IEEE Sensors J.*, vol. 20, no. 7, pp. 3745–3756, Apr. 2020.

[13] A. Bochkovskiy, C.-Y. Wang, and H.-Y. Mark Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*.

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[15] L. Xiao, Q. Yan, and S. Deng, "Scene classification with improved AlexNet model," in *Proc. 12th Int. Conf. Intell. Syst. Knowl. Eng. (ISKE)*, Nanjing, China, Nov. 2017, pp. 1–6.

[16] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Boston, MA, USA, Jun. 2015, pp. 1–9.

[17] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 2818–2826.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778.

[19] Y. T. Li and J. I. Guo, "A VGG-16 based faster RCNN model for PCB error inspection in industrial AOI applications," in *Proc. IEEE Int. Conf. Consum. Electron.-Taiwan (ICCE-TW)*, Taipei, Taiwan, May 2018, pp. 1–2.

[20] Z. Duan, H. Chen, and J. Deng, "AAFM: Adaptive attention fusion mechanism for crowd counting," *IEEE Access*, vol. 8, pp. 138297–138306, 2020.

[21] Y. Li, S. Li, H. Du, L. Chen, D. Zhang, and Y. Li, "YOLO-ACN: Focusing on small target and occluded object detection," *IEEE Access*, vol. 8, pp. 227288–227303, 2020.

[22] L. Xie and C. Huang, "A residual network of water scene recognition based on optimized inception module and convolutional block attention module," in *Proc. 6th Int. Conf. Syst. Informat. (ICSAI)*, Shanghai, China, Nov. 2019, pp. 1174–1178.

[23] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 7132–7141.

[24] A. Raza, H. Huo, and T. Fang, "PFAF-Net: Pyramid feature network for multimodal fusion," *IEEE Sensors Lett.*, vol. 4, no. 12, pp. 1–4, Dec. 2020.

[25] B.-Y. Sun, X.-M. Zhang, J. Li, and X.-M. Mao, "Feature fusion using locally linear embedding for classification," *IEEE Trans. Neural Netw.*, vol. 21, no. 1, pp. 163–168, Jan. 2010.

[26] Z. Baojun, Z. Boya, T. Linbo, W. Wenzheng, and W. Chen, "Multi-scale object detection by top-down and bottom-up feature pyramid network," *J. Syst. Eng. Electron.*, vol. 30, no. 1, pp. 1–12, Feb. 2019.

[27] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Seattle, WA, USA, Jun. 2020, pp. 10778–10787.

[28] 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.*, Salt Lake City, UT, USA, Jun. 2018, pp. 8759–8768.

[29] Y. Cai, T. Luan, H. Gao, H. Wang, and L. Chen, "YOLOv4-5D: An effective and efficient object detector for autonomous driving," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–13, 2021.

[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015, *arXiv:1409.1556*.

[31] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 5987–5995.

[32] G. Huang, Z. Liu, L. V. D. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 2261–2269.

[33] A. S. Gaikwad and M. El-Sharkawy, "Pruning convolution neural network (squeezenet) using Taylor expansion-based criterion," in *Proc. IEEE Int. Symp. Signal Process. Inf. Technol. (ISSPIT)*, Louisvile, KY, USA, Dec. 2018, pp. 1–5.

[34] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 4510–4520.

[35] A. Howard, M. Sandler, B. Chen, W. Wang, L.-C. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le, "Searching for MobileNetV3," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Seoul, South Korea, Oct. 2019, pp. 1314–1324.

[36] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6848–6856.

[37] N. N. Ma, X. Y. Zhang, H. T. Zheng, and J. Sun, "ShuffleNet V2: Practical guidelines for efficient CNN architecture design," in *Proc. Eur. Conf. Comput. Vis.* (Lecture Notes in Computer Science), vol. 11218, Munich, Germany, 2018, pp. 122–138.

[38] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "NAS-FPN: Learning scalable feature pyramid architecture for object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, Jun. 2019, pp. 7029–7038.

[39] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, and C.-Y. Fu, "SSD: Single shot multibox detector," in *Computer Vision–(ECCV)*. Cham, Switzerland: Springer, 2016, pp. 21–37.
[40] L. Wang, Y. Lin, W. Sun, and Y. Wu, "Improved faster-RCNN algorithm for mask wearing detection," in *Proc. IEEE 4th Adv. Inf. Manage., Communicates, Electron. Autom. Control Conf. (IMCEC)*, Chongqing, China, Jun. 2021, pp. 1119–1124.
[41] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 6517–6525.
[42] X. Wang, S. Wang, J. Cao, and Y. Wang, "Data-driven based tiny-YOLOv3 method for front vehicle detection inducing SPP-net," *IEEE Access*, vol. 8, pp. 110227–110236, 2020.
[43] W. Yang and W. Zhang, "Real-time traffic signs detection based on YOLO network model," in *Proc. Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discovery (CyberC)*, Chongqing, China, Oct. 2020, pp. 354–357.
[44] H.-J. Jang, Y.-G. Yim, and H.-W. Jin, "Vertical autoscaling of GPU resources for machine learning in the cloud," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Atlanta, GA, USA, Dec. 2020, pp. 5710–5712.
[45] C.-Y. Wang, H.-Y. Mark Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "CSPNet: A new backbone that can enhance learning capability of CNN," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 1571–1580.
[46] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 16–1904, Sep. 2015.
[47] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-IoU loss: Faster and better learning for bounding box regression," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 7, pp. 12993–13000.
[48] Y. Wang, L. Wang, H. Wang, and P. Li, "Information-compensated downsampling for image super-resolution," *IEEE Signal Process. Lett.*, vol. 25, no. 5, pp. 685–689, May 2018.
[49] J. Nelson and J. Solawetz. (2020). *Responding to the Controversy About YOLOv5*. Accessed: Nov. 1, 2021. [Online]. Available: https://blog.roboflow.com/yolov4-versus-yolov5/
[50] Z. Dong, Y. Wu, M. Pei, and Y. Jia, "Vehicle type classification using a semisupervised convolutional neural network," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 4, pp. 2247–2256, Aug. 2015.
[51] L. Y. Wen, D. Du, Z. Cai, M.-C. Chang, H. Qi, J. Lim, M.-H. Yang, and S. Lyu, "UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking," *Comput. Vis. Image Understand.*, vol. 193, Apr. 2020, Art. no. 102907.

JINGYI ZHAO was born in 1998. She received the B.S. degree from the North China University of Science and Technology, in 2019, where she is currently pursuing the master's degree. Her research interests include machine vision and graphic image processing.

SHENGNAN HAO received the B.S. degree from the North China University of Science and Technology, China, in 1996, and the M.S. degree from the Beijing University of Technology, China, in 2009. She joined the North China University of Science and Technology, in 1996, and became an Associate Professor, in 2009. Her current research interests include complex systems, impulsive systems, and stochastic control.

CHENXU DAI received the B.S. and M.S. degrees from North China University. She is currently a Research Associate with North China University. Her current research interests include neural networks, intelligent optimization algorithms, and deep learning.

HAIYANG ZHANG received the B.S. degree from the School of Software Engineering, Jilin University, China, in 2013, and the Ph.D. degree from the Department of Electronic & Computer Engineering, University of Limerick, Ireland, in 2018.

She is currently a Lecturer with Xi'an Jiaotong-Liverpool University, Suzhou, China. Her current research interests include recommender systems, data mining, collaborative filtering, and natural language processing.

LI ZHAO received the B.S. and Ph.D. degrees from Tsinghua University, Beijing, China, in 1997 and 2002, respectively.

He is currently working as an Associate Professor with the Research Institute of Information Technology, Tsinghua University. His current research interests include mobile computing, the Internet of Things (IoT), E-health systems, intelligent transportation systems (ITS), home networking, machine learning, and digital multimedia.

ZHANLIN JI (Member, IEEE) received the M.Eng. degree from Dublin City University, in 2006, and the Ph.D. degree from the University of Limerick, Ireland, in 2010.

He is currently a Professor with the North China University of Science and Technology, China, and a Researcher with the Telecommunications Research Centre (TRC), University of Limerick. He has authored/coauthored more than 100 research papers in refereed journals and conferences. His research interests include ubiquitous consumer wireless world (UCWW), the Internet of Things (IoT), cloud computing, big data management, and data mining.

IVAN GANCHEV (Senior Member, IEEE) received the Dip.Eng. and Ph.D. *(summa cum laude)* degrees from the Saint-Petersburg University of Telecommunications, in 1995 and 1989, respectively. He is an International Telecommunications Union (ITU-T) Invited Expert and an Institution of Engineering and Technology (IET) Invited Lecturer, currently associated with the University of Limerick, Ireland, and the University of Plovdiv "Paisii Hilendarski" and IMI-BAS, Bulgaria. Prof. Ganchev was involved in more than 40 international and national research projects. He has served on the TPC of more than 350 prestigious international conferences/symposia/workshops, and has authored/coauthored one monographic book, three textbooks, four edited books, and more than 300 research papers in refereed international journals, books, and conference proceedings. Prof. Ganchev seats on the editorial board of and has served as a guest editor for multiple international journals.