ShuffleDet: Real-Time Vehicle Detection Network in On-Board Embedded UAV Imagery

Seyed Majid Azimi[1,2](✉)

[1] German Aerospace Center (DLR), Remote Sensing Technology Institute, Weßling, Germany
seyedmajid.azimi@dlr.de
[2] Chair of Remote Sensing, Technical University of Munich, Munich, Germany
seyedmajid.azimi@tum.de

Abstract. On-board real-time vehicle detection is of great significance for UAVs and other embedded mobile platforms. We propose a computationally inexpensive detection network for vehicle detection in UAV imagery which we call ShuffleDet. In order to enhance the speed-wise performance, we construct our method primarily using channel shuffling and grouped convolutions. We apply inception modules and deformable modules to consider the size and geometric shape of the vehicles. ShuffleDet is evaluated on CARPK and PUCPR+ datasets and compared against the state-of-the-art real-time object detection networks. ShuffleDet achieves 3.8 GFLOPs while it provides competitive performance on test sets of both datasets. We show that our algorithm achieves real-time performance by running at the speed of 14 frames per second on NVIDIA Jetson TX2 showing high potential for this method for real-time processing in UAVs.

Keywords: UAV imagery · Real-time vehicle detection · On-board embedded processing · Convolutional neural networks · Traffic monitoring

1 Introduction

On-board real-time processing of data through embedded systems plays a crucial role in applying the images acquired from the portable platforms (e.g., unmanned aerial vehicless (UAVs)) to the applications requiring instant responses such as search and rescue missions, urban management, traffic monitoring, and parking lot utilization.

Methods based on convolutional neural networks (CNNs), for example, FPN [1], FasterRCNN [19], R-FCN [3], multi-box single shot detectors (SSDs) [14], and Yolov2 [18], have shown promising results in many object detection tasks. Despite their high detection precision, these methods are computationally demanding and their models are usually bulky due to the deep backbone networks being used. Employing CNNs for the on-board real-time applications requires developing time and computation efficient methods due to the limited processing resources available on-board. A number of networks have been developed recently such as GoogleNet [20], Xception [2], ResNeXt [25], MobileNet [7], PeleeNet [23], SqueezeNet [10], and ShuffleNet [26] which have less complex structures as compared to the other CNNs while providing comparable or even superior results. For the real-time object detection applications (*e.g.*, vehicle detection), there are a few recent works proposing the methods such as MobileNet [7] with SSD [9], PVANET [11], and Tiny-Yolo [18]. They have shown computational efficiency to be deployed in mobile platforms.

Zhang et al. [26] employed ShuffleNet as the backbone network, which uses point-wise grouped convolutions and channel shuffle to greatly reduce the computations while maintaining the accuracy. The authors reported a better performance compared with MobileNet using Faster-RCNN detection approach. Kim et al. [11] developed PVANET by concatenating 3×3 conv layer with its negation as a building block for the initial feature extraction stage. Recently, Wang et al. [23] proposed PeleeNet that uses a combination of parallel multi-size kernel convolutions as a 2-way dense layer and a similar module to the Squeeze module. They additionally applied a residual block after feature extraction stage to improve the accuracy using the SSD [14] approach. The authors reported more accurate results compared to MobileNet and ShuffleNet on the Pascal VOC dataset despite the smaller model size and computation cost of PeleeNet. Redmon and Farhadi [18] proposed Yolov2, a fast object detection method, but yet with high accuracy. However, their method is still computationally heavy for real-time processing on an embedded platform. Tiny Yolov2 as the smaller version of Yolov2, although it is faster, but it lacks high-level extraction capability which results in poor performance. In the work of Huang et al. [9], they showed the SSD detection approach together with SqueezeNet and MobileNet as the backbone networks. Although SSD with SqueezeNet backbone results in a smaller model than MobileNet, its results are less accurate and its computation is slightly more expensive. In general, replacing the backbone network with SqueezeNet, MobileNet, or any other efficient network - though enhancing computational efficiency - can degrade the accuracy if no further modification is performed.

In this paper, we propose ShuffleDet, a real-time vehicle detection approach to be used on-board by mobile platforms such as UAVs. ShuffleDet network is composed of ShuffleNet and a modified variant of SSD based on channel shuffling and grouped convolution. We design a unit to appropriately transfer the pretrained parameters of the pretrained model on terrestrial imagery to aerial imagery domain. We call this unit domain adapter block (DAB) which includes deformable convolutions [4] and Inception-ResNetv2 units [21]. To the best of our knowledge, group convolution and channel shuffling have not been used before in real-time vehicle detection based on UAV imagery. ShuffleDet runs at 14 frames per second (FPS) on NVIDIA Jetson TX2 while having the computational complexity of 3.8 giga oating point operations (GFLOPs). Experimental results on the CARPK and PUCPR+ datasets [8] demonstrates that ShuffleDet achieves a good trade-off between accuracy and speed for mobile platforms while it is comparably computation and time efficient.

2 Method

In this section, a detailed description of the network architecture is presented. We use ShuffleNet [26] which is designed for object recognition to extract high-level features as our backend network.

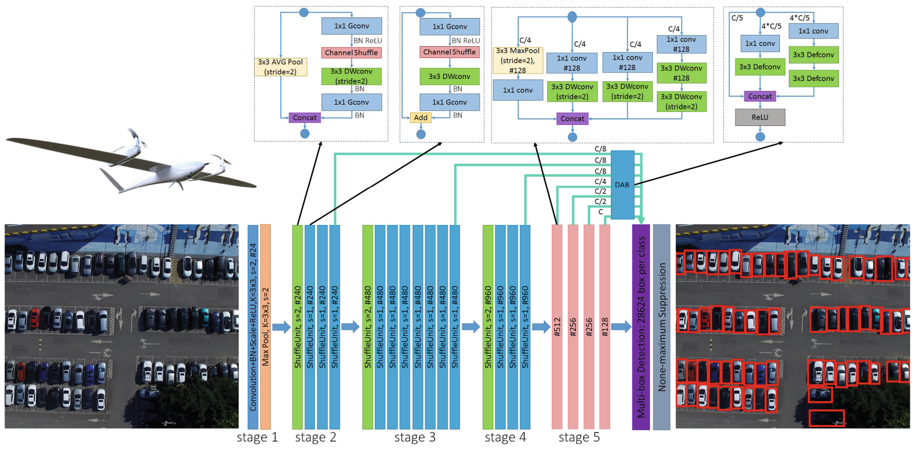

Fig. 1. Illustration of ShuffleDet architecture. The backbone network is ShuffleNet. Modified inception layers are applied as extra layers. *C* stands for channel. DAB unit is deployed to adapt to the new domain of UAV imagery using a residual block containing deformable convolution layers (UAV photo is from https://www.quantum-systems.com/tron.).

ShuffleNet [26] shows that by utilizing grouped or depth-wise separable convolutions, one can reduce the computational demand, while still boosting the performance through a decent representation ability. A major bottleneck can arise by replacing 1×1 convolution layers with stacked grouped convolutions which can degrade the accuracy of the network. This is due to the fact that a limited portion of input channels are utilized by the output channels. In order to solve this issue channel shuffling was proposed in [26] which we also use inside ShuffleDet architecture. Figure 1 illustrates the network architecture of ShuffleDet. In stage 1, a 3×3 convolutional layer is applied to the input image with a stride of 2 which downsamples the input by a factor of 2. This layer is followed by a maxpooling layer with a stride of 2 and kernel of 3×3. This maxpooling operation destroys half of the input information. This is critical as vehicles in our case are small objects [5,12,17,22]. Having said that without this operation, computation cost will be multiplied. Therefore, we keep the maxpooling layer and we try to enhance the performance especially via DABs units which will be discussed later. After the maxpooling three stages containing multiple units from ShuffleNet are performed. Stage 2 and 4 contain 3 ShuffleNet units while stage 3 in the middle is composed of 7 units. The whole stage 1 to 4 leads to 32x down-sampling factor. ShuffleUnit illustrated in Fig. 1 acts as residual bottleneck unit. Using stride 2 in ShuffleUnit, an average pooling is applied to the primary branch parallel with depthwise convolution with a stride 2 in the residual branch. To ensure that all of the input channels are connected to the output channels, channel shuffling is performed before the depthwise convolution. 1×1 grouped convolutions are applied before the channel shuffling as a bottleneck in order to reduce the number of feature maps in the output for the efficient computation. It has been shown [24,26] that the group convolutions also improve the accuracy. The second grouped convolution brings back the number of feature maps or channel to the number of input channels for a more accurate representation capability. Using a stride of 2, the features of average pooling and second grouped convolution is concatenated while having a stride of 1, maxpooling is omitted and depth-wise convolution is performed. Moreover, the outputs are summed up instead of using concatenation. Figure 1 shows the detailed structure of ShuffleNet units with and without stride of 2.

Stage 1, 2, 3 and stage 4 are employed to enhance the heat map resolution as input intermediate layers. In the detection module, we primarily inspire from SSD approach. In order to enrich the extracted features from the intermediate layers, we perform extra feature layers in stage 5. In our case, the output from stage 4 is passed through stage 5 as illustrated in Fig. 1 This is compatible with using multi-box strategy explained in the SSD method. In total, we extract 7 feature maps of different sizes from the backbone network.

To enhance the performance, instead of employing a conventional convolution layer similar to SSD method for each extra layer, we use a modified module of Reduction-B from Inception-ResNet-v2 [21] in stage 5. Unlike ResNet and VGG, inception modules have not been explored enough in object detection task due to their higher computation cost. We stack 4 modified inception modules as stage 5 for feature map extraction at different levels. Unlike original Inception-ResNet-v2 work, we add 1×1 conv layers after maxpooling and concatenation layer. The maxpooling layer reduces spatial-resolution and dimension as a bottleneck. 1×1 convolution in return expands the dimension to insert further non-linearity to the network resulting in a better performance. The same philosophy was used in the latter 1×1 conv layer. Applying the inception module adds more computational cost to the network. To compensate its load, we replace 3×3 convolution layers with 3×3 depthwise convolutions. Depth-wise convolution improves the performance slightly, yet it has $\frac{1}{N} + \frac{1}{k^2}$ times less computation cost compared with regular conv layers. N is the number of output channels and k is the kernel size. Furthermore, we divide the input channels equally among the branches. The output number of channels for each layer is an equally-divided concatenation of output channels from each branch. These modifications keep the model size as well as computational complexity small. We observe using this modified inception modules enhances the performance. We conjecture unlike the original SSD which uses 1×1 and 3×3 conv layers in series as extra layers, multi-size kernels parallel in inception modules capture features in different sizes simultaneously e.g. 1×1 kernels to detect small vehicles and 3×3 kernels for bigger ones which could be the reason for this enhancement. This shows by widening the network and augmenting the cardinality, we can achieve better results. This comes only with a marginal increase in computational complexity. Moreover, by using multi-size kernels, one does not need to worry which kernel size is more appropriate.

In order to regress bounding boxes and predict object classes from extra layers as illustrated in Fig. 1, the base-line SSD processes each feature map by only a single 3×3 convolution layer followed by `permute` and `flatten` layers in multi-box detection layer. This includes feature maps only from one of the high-resolution layers. This leads to a weak performance in detecting small-scale vehicles. The feature maps from higher-resolution layers e.g. in our case stage 2 and 3 are responsible to detect small-scale vehicles. Stage 1 is ignored due to its high computational complexity. Those corresponding feature maps are semantically weak and not deep enough to be capable of detecting small-scale vehicles. ResNet and VGG19 works denote that employing deeper features enhances the object recognition accuracy. However, those backbone networks are computationally heavy to be deployed on on-board processors in UAVs which work under strict power constraints. As an alternative, we propose using a residual module which we call DAB as shown in Fig. 1. Combination of 1×1 convention and 3×3 deformable convolution operations enrich the features further, but still introducing low computation burden. We choose a portion of input channels to keep the computation cost low. $1/8, 1/8, 1/8, 1/4, 1/2, 1/2, 1$ are used as the portion of input channels of output layers from stage 2 to the last extra layer and inside DAB unit we assign $1/5, 4/5, 4/5$ portion of input channels to each branch as illustrated in Fig. 1. The output channels remain similar to the original SSD. The only difference is the introduced extra multi-box feature map from stage 2. SSD calculates the number of default boxes per class by $W \times H \times B$ in which W and H are input width and height and B is from the set of $4, 6, 6, 4, 4$ for each feature map. We choose $B = 4$ for the stage 2 leading to 28642 boxes per class.

In aerial imagery, vehicles appear to be very small and almost always in rectangle geometric shape. On the other hand, the pre-trained ShuffleNet has been trained on ground imagery while our images are in another domain of aerial imagery. Therefore pre-trained weights should be adapted to the new domain. We use deformable convolution as introduced in [4] to take into account the new domain and the geometric properties of the vehicles. Deformable convolution adds an offset to the conventional conv layer in order to learn from the geometric shape of the objects. They are not limited to a fix kernel size and offset is learned during training by adding only an inexpensive conv layer to compute the offset field. Deformable conv layer shows considerable improvement in case of using images acquired from low-flying UAVs. However, the impact is less by using images from high-altitude platforms such as helicopter or airplanes. According to [4] the computation cost of deformable convolutions is negligible. Finally, we apply `ReLU` layer to element-wise added features in the DAB to add more non-linearity. In general, naive implementation of ShuffleNet with SSD has 2.94 GFLOPs while ShuffleDet has 3.8 GFLOPs. Despite an increase in the computation cost, ShuffleDet has considerable higher accuracy. As vehicles appear to be small objects in UAV images, we choose default prior boxes with smaller scales similar to [5]. Eventually, non-maximum suppression (NMS) is employed to suppress irrelevant detection boxes. It is worth mentioning that during training hard negative mining is employed with the ratio of 3:1 between negative and positive samples. This leads to more stable and faster training. We also apply batch normalization after each module in DAB as well as extra feature layers.

3 Experiments and Discussion

In this section, we provide ablation evaluation of our proposed approach and compare it to the state-of-the-art CNN-based vehicle detection methods. The experiments were conducted on the CARPK and PUCPR+ datasets [8], which contain 1573 and 125 images of 1280×720 pixels, respectively. The vehicles in the images are annotated by horizontal bounding boxes. To have a fair comparison with different baseline methods, we follow the same strategy as theirs for splitting the datasets into training and testing sets. Moreover, we train ShuffleNet as the backbone network on the ImageNet-2012 [6] dataset achieving similar performance compared to the original ShuffleNet work. The results are compared to the benchmark using MAE and RMSE, similar to the baseline [8]. In addition, we use data augmentation in a similar way to the original work on SSD.

3.1 Experimental Setup

We use Caffe to implement our proposed algorithm. It is trained using Nvidia Titan XP GPU and evaluated on NVIDIA Jetson TX2 as an embedded edge device. For the optimization, we use stochastic gradient descent with the base learning rate of 0.001, gamma 0.1, momentum 0.9 to train the network for 120k iterations. The learning rate is reduced after 80k and 100k by a factor of 10. Moreover, the images are resized to 512×512 pixels along with their annotations. Additionally, we initialize the first four layers with our pre-trained ShuffleNet weights and the rest with Gaussian noise. For the grouped convolutions, we set the number of groups to 3 throughout the experiments. Furthermore, NMS of 0.3 and confidence score threshold of 0.5 are considered.

3.2 Ablation Evaluation

In this section, we present an ablation study on the effect of the submodules in our approach. Table 1 shows the impact of the modified inception module compared to the original baseline. According to the results, introducing the first modified inception module (small scales) decreases RMSE by about 4 points indicating the importance of wider networks in first layers as the critical layers of the network for small object detection. Replacing the baseline's extra layers with more modified inception models further improves the performance. This highlights the role of higher-resolution layers in the vehicle detection tasks.

Table 1. Evaluation of modified inception module (mincep) in the stage 5 on the CARPK dataset. The DAB units are in place. Smaller the RMSE, better the performance.

Method	RMSE	Small scales	mincep-1	mincep-2	mincep-3	mincep-4
ShuffleNet-SSD-512	63.57	-	-	-	-	
ShuffleDet	52.75	-	-	-	-	
ShuffleDet	45.26	✓	-	-	-	
ShuffleDet	41.89	✓	✓	-	-	-
ShuffleDet	40.47	✓	✓	✓	-	-
ShuffleDet	39.67	✓	✓	✓	✓	-
ShuffleDet	38.46	✓	✓	✓	✓	✓

Table 2 represents the evaluation of DAB unit in which we observe a significant reduction in RMSE (almost 5 points) even by the first DAB unit on stage 2. This further indicates the significance of including higher-resolution layer. Furthermore, the results show that adding DAB modules to the extra layer can additionally enhance the performance to a lesser degree. This performance indicates that applying the DAB unit in the high-resolution layers can lead to a significant improvement in detecting small vehicles allowing a better utilization of the deformable convolution to adapt to the vehicle geometries.

Table 2. Evaluation of using DAB unit on the CARPK dataset. We refer to modified inception layers as `mincep`. The modified inception modules and small scales are in place.

Method	RMSE	DAB-stage2	DAB-stage3	DAB-stage4	DAB-mincep-1	DAB-mincep-2	DAB-mincep-3
ShuffleNet-SSD-512	63.57	-	-	-	-	-	-
ShuffleDet	49.26	-	-	-	-	-	-
ShuffleDet	44.17	✓	-	-	-	-	-
ShuffleDet	42.02	✓	✓	-	-	-	-
ShuffleDet	40.75	✓	✓	✓	-	-	-
ShuffleDet	39.81	✓	✓	✓	✓	-	-
ShuffleDet	39.14	✓	✓	✓	✓	✓	-
ShuffleDet	38.46	✓	✓	✓	✓	✓	✓

We choose $s_{min} = 0.05$ and $s_{max} = 0.4$ as minimum and maximum vehicle scales with ratio of $2, 3, 1/2, 1/3$ as hyper-parameters in the original SSD. This improves the performance significantly according to Table 1 by almost 7 RMSE points. It is worth noting that ShuffleNet-SSD-512 has 2.94 GFLOPs as complexity cost while ShuffleDet has 3.8 GFLOPs. This shows ShuffleDet adds only a marginal computation cost while achieving a significant boost in the accuracy. Figure 2 shows sample results of ShuffleDet on the CARPK and PUCPR+ datasets.

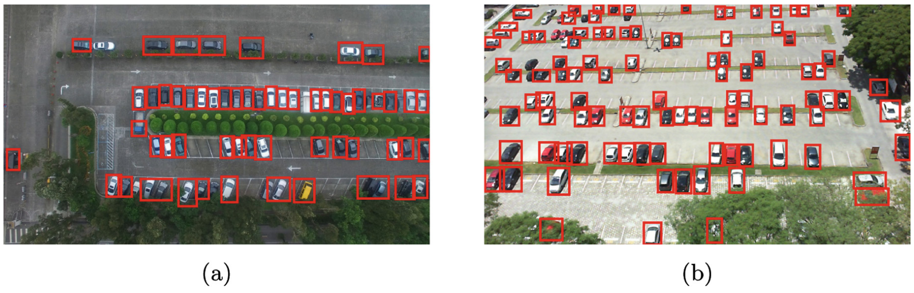

Fig. 2. Sample vehicle detection results using ShuffleDet on the CARPK (a) dataset and the PUCPR+ dataset (b).

3.3 Comparison with the Benchmark

In this part, compare our method with the benchmark. Tables 3 and 4 show that our method can achieve competitive performance while having significantly less computation cost compared with the state of the art. In comparison with the original implementation of Faster-RCNN [19] and Yolo [16], our method achieves significantly better results. ShuffleDet achieves comparative result with the state of the art with only about less 2 RMSE points in the CARPK dataset. The reason for the big gap between SSD-512, MobileNet-SSD-512 and shuffleDet is mostly due

Table 3. Evaluation of ShuffleDet with the benchmark on the PUCPR+ dataset. The less is better.

Method	Backbone	GFLOPs	MAE	RMSE
YOLO [16]	Custom	26.49	156.00	200.42
Faster-RCNN [19]	VGG16	118.61	111.40	149.35
Faster R-CNN (RPN-small) [19]	VGG16	118.61	39.88	47.67
One-look regression [15]	-	-	21.88	36.73
Hsieh et al. [8]	VGG16	-	22.76	34.46
SSD-512 [14]	VGG16	88.16	123.75	168.24
MobileNet-SSD-512 [9]	MobileNet	3.2	175.26	225.12
Our ShuffleDet	ShuffleNet	3.8	41.58	49.68

Table 4. Evaluation of ShuffleDet with the benchmark on the CARPK dataset. The less is better.

Method	Backbone	GFLOPs	MAE	RMSE
YOLO [16]	Custom	26.49	48.89	57.55
Faster-RCNN [19]	VGG16	118.61	47.45	57.39
Faster R-CNN (RPN-small) [19]	VGG16	118.61	24.32	37.62
One-look regression [15]	-	-	59.46	66.84
Hsieh et al. [8]	VGG16	-	23.80	36.79
SSD-512 [14]	VGG16	88.16	48.02	57.42
MobileNet-SSD-512 [9]	MobileNet	3.2	57.34	65.24
Our ShuffleDet	ShuffleNet	3.8	26.75	38.46

to our tuned scales and aspect ratios. This effect can also be observed between the original implementation of Faster-RCNN with and without small RPNs.

Moreover, ShufflDet achieves its superiority to Faster-RCNN and Yolo while it is significantly more computation efficient, 3.8 GFLOPs compared to 118 and 26.49 GFLOPs. While Faster-RCNN runs at Jetson TX2 with 1 FPS, tiny Yolov2 at 8 and Yolov2 at 4 FPS, and original SSD with 88.16 GFLOPs at 5 FPS, our ShuffleDet network runs at 14 FPS showing a great potential to be deployed in the real-time on-board processing in UAV imagery. In addition, our approach achieves almost 70% and 50% better performance than MobileNet-SSD-512 and the naive implementation of ShuffleNet-SSD on the CARPK dataset, relatively.

4 Generalization Ability

To evaluate the generalization ability of our method, we train it on the 3K-DLR-Munich dataset [13]. This dataset contains aerial images of 5616 $\times$ 3744 pixels over the Munich city. Due to the large size of each image similar to [5], we chop the images into the patches of 512 $\times$ 512 pixels which have 100 pixels overlap. To prepare the final results, for each image, we merge the detections results of the patches and then apply none-maximum suppression. Figure 3 illustrates a detection result of our algorithm for the 3K-DLR-Munich dataset.

Table 5 compares the performance of ShuffleDet and two implementations of Faster-RCNN on the 3K-DLR-Munich dataset. According to the table, ShuffleDet not only outperforms the Faster-RCNN methods but also its inference is much more time efficient. The consistent behavior of our proposed approach on the 3K-DLR-Munich dataset indicates that it could be generally applied to different datasets. ShuffleDet is capable of 2 FPS processing of high-resolution aerial images in Jetson TX2 platform while Faster-RCNN with VGG16 and ResNet-50 takes a couple of seconds.

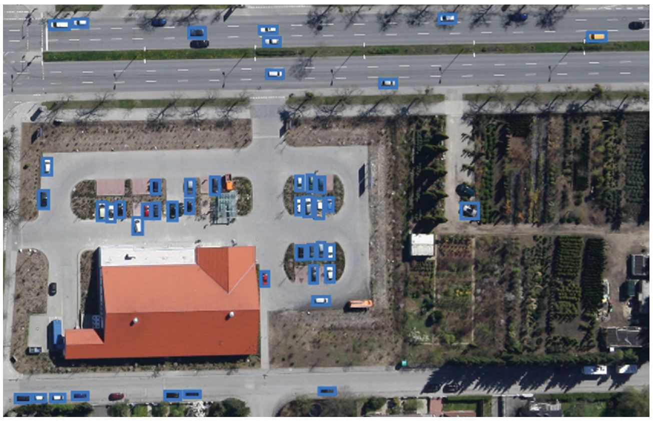

Fig. 3. Vehicle detection result using ShuffleDet on the 3K-DLR-Munich dataset.

Table 5. Evaluation of ShuffleDet on 3K-DLR-Munich dataset. Inference time is computed in Jetson TX2 as an edge device.

Method	Backend	GFLOPs	mAP	Inference time
Faster-RCNN [19]	VGG-16	118.61	67.45%	7.78 s
Faster-RCNN [19]	ResNet-50	22.06	69.23%	7.34 s
Our ShuffleDet	ShuffleNet	**3.8**	62.89	**524 ms**

5 Conclusions

In this paper, we presented ShuffleDet, a real-time vehicle detection algorithm appropriate for on-board embedded UAV imagery. ShuffleDet is based on channel shuffling and grouped convolution in its feature extraction stage. To evaluate the effect of different modules of ShuffleDet, an ablation study is performed to discuss its accuracy and time-efficiency. Joint channel shuffling and grouped convolution significantly boost the inference time. Inception modules with depthwise convolutions enhance the accuracy while introducing a marginal computation burden. Moreover, we show residual modules with deformable convolutions are effective modules for semantic representation enhancement in the small number of layers as well as domain adaptation. Experimental results on the CARPK and PUCPR+ datasets indicate that ShuffleDet outperforms the state-of-the-arts methods while it is much more time and computation efficient. Additionally, the consistent behavior of ShuffleDet on the 3K-DLR-Munich dataset demonstrate its generalization ability. Furthermore, the implementation of ShuffleDet on Jetson TX2, which runs at 14 FPS, showing a great potential of our approach to be used in UAVs for on-board real-time vehicle detection.

References

1. Lin, T., Dollár, P., Girshick, R.B., He, K., Hariharan, B., Belongie, S.J.: Feature pyramid networks for object detection. In: CVPR (2017)
2. Chollet, F.: Xception: deep learning with depthwise separable convolutions. arXiv preprint arXiv:1610.02357 (2017)
3. Dai, J., Li, Y., He, K., Sun, J.: R-FCN: object detection via region-based fully convolutional networks. In: NIPS (2016)
4. Dai, J., et al.: Deformable convolutional networks. In: ICCV (2017)
5. Azimi, S.M., Vig, E., Bahmanyar, R., Körner, M., Reinartz, P.: Towards multi-class object detection in unconstrained remote sensing imagery. In: ACCV (2018)
6. Deng, J., Dong, W., Socher, R., Li, L., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: CVPR (2009)
7. Howard, A.G., et al.: MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
8. Hsieh, M., Lin, Y., Hsu, W.H.: Drone-based object counting by spatially regularized regional proposal network. In: ICCV (2017)
9. Huang, J., et al.: Speed/accuracy trade-offs for modern convolutional object detectors. In: CVPR (2017)
10. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K.: SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5 mb model size. arXiv preprint arXiv:1602.07360 (2016)
11. Kim, K.H., Hong, S., Roh, B., Cheon, Y., Park, M.: PVANET: deep but lightweight neural networks for real-time object detection. arXiv preprint arXiv:1608.08021 (2016)
12. Azimi, S.M., Vig, E., Kurz, F., Reinartz, P.: Segment-and-count: vehicle counting in aerial imagery using atrous convolutional neural networks. In: ISPRS (2018)
13. Liu, K., Mattyus, G.: Fast multiclass vehicle detection on aerial images. IEEE GRSL Lett. **12**, 1938–1942 (2015)
14. Liu, W., et al.: SSD: single shot multibox detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 21–37. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_2
15. Mundhenk, T.N., Konjevod, G., Sakla, W.A., Boakye, K.: A large contextual dataset for classification, detection and counting of cars with deep learning. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9907, pp. 785–800. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46487-9_48
16. Redmon, J., Divvala, S.K., Girshick, R.B., Farhadi, A.: You only look once: unified, real-time object detection. In: CVPR (2016)
17. Azimi, S.M., Fischer, P., Körner, M., Reinartz, P.: Aerial LaneNet: lane marking semantic segmentation in aerial imagery using wavelet-enhanced cost-sensitive symmetric fully convolutional neural networks. arXiv preprint arXiv:1803.06904 (2018)
18. Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. In: CVPR (2017)
19. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: NIPS (2015)
20. Szegedy, C., et al.: Going deeper with convolutions. In: CVPR (2015)
21. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, Inception-ResNet and the impact of residual connections on learning. In: ICLR (2016)
22. Azimi, S.M., Britz, D., Engstler, M., Fritz, M., Mücklich, F.: Advanced steel microstructural classification by deep learning methods. Sci. Rep. - Nat. **8**, 2128 (2018)

23. Wang, R.J., Li, X., Ao, S., Ling, C.X.: Pelee: a real-time object detection system on mobile devices. arXiv preprint arXiv:1804.06882 (2018)
24. Wu, J., Leng, C., Wang, Y., Hu, Q., Cheng, J.: Quantized convolutional neural networks for mobile devices. In: CVPR (2016)
25. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: CVPR (2017)
26. Zhang, X., Zhou, X., Lin, M., Sun, J.: ShuffleNet: an extremely efficient convolutional neural network for mobile devices. arXiv preprint arXiv:1707.01083 (2017)