



# VisDrone-DET2018: The Vision Meets Drone Object Detection in Image Challenge Results

Pengfei Zhu<sup>1</sup>(✉), Longyin Wen<sup>2</sup>, Dawei Du<sup>3</sup>, Xiao Bian<sup>4</sup>, Haibin Ling<sup>5</sup>,  
Qinghua Hu<sup>1</sup>, Qinqin Nie<sup>1</sup>, Hao Cheng<sup>1</sup>, Chenfeng Liu<sup>1</sup>, Xiaoyu Liu<sup>1</sup>,  
Wenya Ma<sup>1</sup>, Haotian Wu<sup>1</sup>, Lianjie Wang<sup>1</sup>, Arne Schumann<sup>31</sup>, Chase Brown<sup>6</sup>,  
Chen Qian<sup>28</sup>, Chengzheng Li<sup>29</sup>, Dongdong Li<sup>27</sup>, Emmanouil Michail<sup>20</sup>,  
Fan Zhang<sup>14</sup>, Feng Ni<sup>22</sup>, Feng Zhu<sup>21</sup>, Guanghui Wang<sup>10</sup>, Haipeng Zhang<sup>13</sup>,  
Han Deng<sup>25</sup>, Hao Liu<sup>27</sup>, Haoran Wang<sup>14</sup>, Heqian Qiu<sup>36</sup>, Honggang Qi<sup>18</sup>,  
Honghui Shi<sup>9</sup>, Hongliang Li<sup>36</sup>, Hongyu Xu<sup>7</sup>, Hu Lin<sup>11</sup>, Ioannis Kompatsiaris<sup>20</sup>,  
Jian Cheng<sup>34</sup>, Jianqiang Wang<sup>33</sup>, Jianxiu Yang<sup>14</sup>, Jingkai Zhou<sup>11</sup>,  
Juanping Zhao<sup>28</sup>, K. J. Joseph<sup>23</sup>, Kaiwen Duan<sup>18</sup>, Karthik Suresh<sup>6</sup>, Bo Ke<sup>12</sup>,  
Ke Wang<sup>14</sup>, Konstantinos Avgerinakis<sup>20</sup>, Lars Sommer<sup>31,32</sup>, Lei Zhang<sup>19</sup>,  
Li Yang<sup>14</sup>, Lin Cheng<sup>14</sup>, Lin Ma<sup>26</sup>, Liyu Lu<sup>1</sup>, Lu Ding<sup>28</sup>, Minyu Huang<sup>16</sup>,  
Naveen Kumar Vedurupaka<sup>24</sup>, Nehal Mamgain<sup>23</sup>, Nitin Bansal<sup>6</sup>,  
Oliver Acatay<sup>31</sup>, Panagiotis Giannakeris<sup>20</sup>, Qian Wang<sup>14</sup>, Qijie Zhao<sup>22</sup>,  
Qingming Huang<sup>18</sup>, Qiong Liu<sup>11</sup>, Qishang Cheng<sup>36</sup>, Qiuchen Sun<sup>14</sup>,  
Robert Laganière<sup>30</sup>, Sheng Jiang<sup>14</sup>, Shengjin Wang<sup>33</sup>, Shubo Wei<sup>14</sup>,  
Siwei Wang<sup>14</sup>, Stefanos Vrochidis<sup>20</sup>, Sujuan Wang<sup>34</sup>, Tiaoqiao Lee<sup>25</sup>,  
Usman Sajid<sup>10</sup>, Vineeth N. Balasubramanian<sup>23</sup>, Wei Li<sup>36</sup>, Wei Zhang<sup>25</sup>,  
Weikun Wu<sup>16</sup>, Wenchi Ma<sup>10</sup>, Wenrui He<sup>21</sup>, Wenzhe Yang<sup>14</sup>, Xiaoyu Chen<sup>36</sup>,  
Xin Sun<sup>17</sup>, Xinbin Luo<sup>28</sup>, Xintao Lian<sup>14</sup>, Xiufang Li<sup>14</sup>, Yangliu Kuai<sup>27</sup>,  
Yali Li<sup>33</sup>, Yi Luo<sup>11</sup>, Yifan Zhang<sup>34,35</sup>, Yiling Liu<sup>15</sup>, Ying Li<sup>15</sup>, Yong Wang<sup>30</sup>,  
Yongtao Wang<sup>22</sup>, Yuanwei Wu<sup>10</sup>, Yue Fan<sup>25</sup>, Yunchao Wei<sup>8</sup>, Yuqin Zhang<sup>16</sup>,  
Zexin Wang<sup>14</sup>, Zhangyang Wang<sup>6</sup>, Zhaoyue Xia<sup>33</sup>, Zhen Cui<sup>29</sup>, Zhenwei He<sup>19</sup>,  
Zhipeng Deng<sup>27</sup>, Zhiyao Guo<sup>16</sup>, and Zichen Song<sup>36</sup>

<sup>1</sup> Tianjin University, Tianjin, China  
zhupengfei@tju.edu.cn

<sup>2</sup> JD Finance, Mountain View, CA, USA

<sup>3</sup> University at Albany, SUNY, Albany, NY, USA

<sup>4</sup> GE Global Research, Niskayuna, NY, USA

<sup>5</sup> Temple University, Philadelphia, PA, USA

<sup>6</sup> Texas A&M University, College Station, USA

<sup>7</sup> University of Maryland, College Park, USA

<sup>8</sup> University of Illinois at Urbana-Champaign, Urbana-Champaign, USA

<sup>9</sup> Thomas J. Watson Research Center, Yorktown Heights, USA

<sup>10</sup> University of Kansas, Lawrence, USA

<sup>11</sup> South China University of Technology, Guangzhou, China

<sup>12</sup> Sun Yat-sen University, Guangzhou, China

<sup>13</sup> Jiangnan University, Wuxi, China

<sup>14</sup> Xidian University, Xi'an, China

<sup>15</sup> Northwestern Polytechnical University, Xi'an, China

- <sup>16</sup> Xiamen University, Xiamen, China  
<sup>17</sup> Ocean University of China, Qingdao, China  
<sup>18</sup> University of Chinese Academy of Sciences, Beijing, China  
<sup>19</sup> Chongqing University, Chongqing, China  
<sup>20</sup> Centre for Research and Technology Hellas, Thessaloniki, Greece  
<sup>21</sup> Beijing University of Telecommunication and Post, Beijing, China  
<sup>22</sup> Peking University, Beijing, China  
<sup>23</sup> Indian Institute of Technology, Hyderabad, India  
<sup>24</sup> NIT Trichy, Tiruchirappalli, India  
<sup>25</sup> Shandong University, Jinan, China  
<sup>26</sup> Tencent AI Lab, Bellevue, China  
<sup>27</sup> National University of Defense Technology, Changsha, China  
<sup>28</sup> Shanghai Jiao Tong University, Shanghai, China  
<sup>29</sup> Nanjing University of Science and Technology, Nanjing, China  
<sup>30</sup> University of Ottawa, Ottawa, Canada  
<sup>31</sup> Fraunhofer IOSB, Karlsruhe, Germany  
<sup>32</sup> Karlsruhe Institute of Technology, Karlsruhe, Germany  
<sup>33</sup> Tsinghua University, Beijing, China  
<sup>34</sup> Nanjing Artificial Intelligence Chip Research, Institute of Automation, Chinese Academy of Sciences, Beijing, China  
<sup>35</sup> Institute of Automation, Chinese Academy of Sciences, Beijing, China  
<sup>36</sup> University of Electronic Science and Technology of China, Chengdu, China

**Abstract.** Object detection is a hot topic with various applications in computer vision, *e.g.*, image understanding, autonomous driving, and video surveillance. Much of the progresses have been driven by the availability of object detection benchmark datasets, including PASCAL VOC, ImageNet, and MS COCO. However, object detection on the drone platform is still a challenging task, due to various factors such as view point change, occlusion, and scales. To narrow the gap between current object detection performance and the real-world requirements, we organized the Vision Meets Drone (VisDrone2018) Object Detection in Image challenge in conjunction with the 15th European Conference on Computer Vision (ECCV 2018). Specifically, we release a large-scale drone-based dataset, including 8,599 images (6,471 for training, 548 for validation, and 1,580 for testing) with rich annotations, including object bounding boxes, object categories, occlusion, truncation ratios, etc. Featuring a diverse real-world scenarios, the dataset was collected using various drone models, in different scenarios (across 14 different cities spanned over thousands of kilometres), and under various weather and lighting conditions. We mainly focus on ten object categories in object detection, *i.e.*, pedestrian, person, car, van, bus, truck, motor, bicycle, awning-tricycle, and tricycle. Some rarely occurring special vehicles (*e.g.*, machineshop truck, forklift truck, and tanker) are ignored in evaluation. The dataset is extremely challenging due to various factors, including large scale and pose variations, occlusion, and clutter background. We present the evaluation protocol of the VisDrone-DET2018 challenge and the comparison

results of 38 detectors on the released dataset, which are publicly available on the challenge website: <http://www.aiskyeye.com/>. We expect the challenge to largely boost the research and development in object detection in images on drone platforms.

**Keywords:** Performance evaluation · Drone  
Object detection in images

## 1 Introduction

Detecting objects in images, which aims to detect objects of the predefined set of object categories (*e.g.*, cars and pedestrians), is a problem with a long history [9, 17, 32, 40, 50]. Accurate object detection would have immediate and far reaching impact on many applications, such as image understanding, video surveillance, and anomaly detection. Although object detection attracts much research and has achieved significant advances with the deep learning techniques in recent years, these algorithms are not usually optimal for dealing with sequences or images captured by drone-based platforms, due to various challenges such as view point change, scales and occlusion.

To narrow the gap between current object detection performance and the real-world requirements, we organized the “Vision Meets Drone - Object Detection in Images (VisDrone-DET2018) challenge, which is one track of the “Vision Meets Drone: A Challenge” (or VisDrone2018, for short) on September 8, 2018, in conjunction with the 15th European Conference on Computer Vision (ECCV 2018) in Munich, Germany. We collected a large-scale object detection dataset in real scenarios with detailed annotations. The VisDrone2018 challenge mainly focus on human and vehicles in our daily life. The comparisons of the proposed dataset and previous datasets are presented in Table 1.

We invite researchers to submit algorithms to detect objects of ten predefined categories (*e.g.*, pedestrian and car) from individual images in the VisDrone-DET2018 dataset, and share their research results at the workshop. We believe this comprehensive challenge benchmark is useful to further boost research on object detection on drone platforms. The authors of the detection algorithms in this challenge have an opportunity to share their ideas and publish the source code at our website: <http://www.aiskyeye.com/>, which are helpful to promote the development of object detection algorithms.

## 2 Related Work

### 2.1 Existing Datasets

Several object detection benchmarks have been collected for evaluating object detection algorithms. Enzweiler and Gavrila [12] present the Daimler dataset, captured by a vehicle driving through urban environment. The dataset includes 3,915 manually annotated pedestrians in video images in the training set, and 21,790 video images with 56,492 annotated pedestrians in the testing set.

The Caltech dataset [11] consists of approximately 10 h of  $640 \times 480$  30 Hz videos taken from a vehicle driving through regular traffic in an urban environment. It contains  $\sim 250,000$  frames with a total of 350,000 annotated bounding boxes of 2,300 unique pedestrians. The KITTI-D benchmark [19] is designed to evaluate the car, pedestrian, and cyclist detection algorithms in autonomous driving scenarios, with 7,481 training and 7,518 testing images. Mundhenk *et al.* [34] create a large dataset for classification, detection and counting of cars, which contains 32,716 unique cars from six different image sets, different geographical locations and different imagers. The recent UA-DETRAC benchmark [33, 47] provides 1,210k objects in 140k frames for vehicle detection.

The PASCAL VOC dataset [15, 16] is one of the pioneering works in generic object detection, which is designed to provide a standardized test bed for object detection, image classification, object segmentation, person layout, and action classification. ImageNet [10, 41] follows the footsteps of the PASCAL VOC dataset by scaling up more than an order of magnitude in the number of object classes and images, *i.e.*, PASCAL VOC 2012 with 20 object classes and 21,738 images *vs.* ILSVRC2012 with 1,000 object classes and 1,431,167 annotated images. Recently, Lin *et al.* [31] release the MS COCO dataset, containing more than 328,000 images with 2.5 million manually segmented object instances. It has 91 object categories with 27.5k instances on average per category. Notably, it contains object segmentation annotations that are not available in ImageNet.

**Table 1.** Comparisons of current state-of-the-art benchmarks and datasets for object detection. Note that, the resolution indicates the maximum resolution of the videos/images included in the dataset.

Datasets	Scen.	#img.	Cat.	Avg. #labels/cat.	Res.	Occ.	Year
UIUC [1]	Life	1,378	1	739	$200 \times 150$		2004
INRIA [9]	Life	2,273	1	1,774	$96 \times 160$		2005
ETHZ [13]	Life	2,293	1	10.9k	$640 \times 480$		2007
TUD [2]	Life	1,818	1	3,274	$640 \times 480$		2008
EPFL Car [35]	Exhibition	2,000	1	2,000	$376 \times 250$		2009
Caltech [11]	Driving	249k	1	347k	$640 \times 480$	✓	2012
KITTI [19]	Driving	15.4k	2	80k	$1241 \times 376$	✓	2012
VOC2012 [14]	Life	22.5k	20	1,373	$469 \times 387$	✓	2012
ImageNet [41]	Life	456.2k	200	2,007	$482 \times 415$	✓	2013
MS COCO [31]	Life	328.0k	91	27.5k	$640 \times 640$		2014
VEDAI [36]	Satellite	1.2k	9	733	$1024 \times 1024$		2015
COWC [34]	Aerial	32.7k	1	32.7k	$2048 \times 2048$		2016
CARPK [26]	Drone	1,448	1	89.8k	$1280 \times 720$		2017
VisDrone- DET2018	drone	8,599	10	46.6k	$2000 \times 1500$	✓	2018

## 2.2 Review of Object Detection Methods

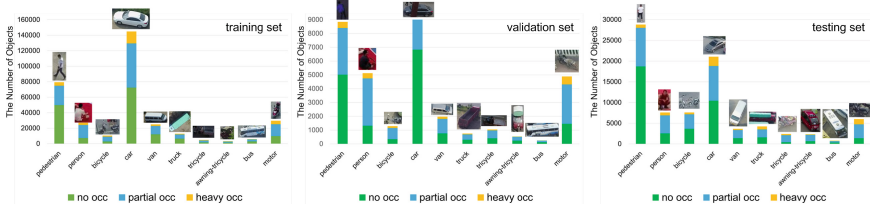
**Classical Object Detectors.** In early days, the object detection methods are constructed based on the sliding-window paradigm, which use the hand-crafted features and classifiers on dense image grids to locate objects. As one of previous most popular frameworks, Viola and Jones [45] use Haar feature and Adaboost algorithm to learn a series of cascaded classifiers for face detection, which achieves accurate results with high efficiency. Felzenszwalb *et al.* [17] develop an effective object detection method based on mixtures of multiscale deformable part models. Specifically, they calculate the Histograms of Oriented Gradients (HOG) features on each part of object and train the latent SVM (a reformulation of MI-SVM in terms of latent variables) for robust performance. However, the classical object detectors do not perform well in challenging scenarios. In recent years, with the advance of deep Convolutional Neural Network (CNN), the object detection field is dominated by the CNN-based detectors, which can be roughly divided into two categories, *i.e.*, the two-stage approach and the one-stage approach.

**Two-Stage CNN-Based Methods.** The two-stage approach first generates a pool of object proposals by a separated proposal generator and then predicts the accurate object regions and the corresponding class labels, such as R-CNN [21], SPP-Net [24], Fast R-CNN [20], Faster R-CNN [40], R-FCN [7], Mask R-CNN [23], and FPN [29].

R-CNN [21] is one of the pioneering works using the CNN model pre-trained on ImageNet, which extracts a fixed-length feature vector from each proposal using a CNN, and then classifies each region with category-specific linear SVM. SPP-Net [24] proposes the SPP layer that pools the features and generates fixed length outputs to remove the fixed input size constraint of the CNN model. In contrast to SPP [24], Fast R-CNN [20] designs a single-stage training algorithm that jointly learns to classify object proposals and refine their spatial locations in an end-to-end way. Faster R-CNN [40] further improves Fast R-CNN using a region proposal network instead of the selective search algorithm [44] to extract the region proposals. The R-FCN method [7] develops a fully convolutional network (FCN) to solve object detection, which constructs a set of position-sensitive maps using a bank of specialized convolutional layers to incorporate translation variance into FCN. Recently, Lin *et al.* [29] exploit the inherent multi-scale, pyramidal hierarchy of deep convolutional networks to construct feature pyramids with marginal extra cost to improve the detection performance. In [28], the head of network is designed as light as possible to decrease the computation cost, by using a thin feature map and a cheap R-CNN subnet (pooling and single fully-connected layer). Zhang *et al.* [49] propose a new occlusion-aware R-CNN to improve the pedestrian detection in the crowded scenes, which designs an aggregation loss to enforce proposals to be close and locate compactly to the corresponding objects. In general, the aforementioned methods share almost the same pipeline for object detection (*i.e.*, object proposal generation, feature extraction, object classification and bounding box regression). The region proposal generating stage is the bottleneck to improve running efficiency.

**One-Stage CNN-Based Methods.** Different from the two-stage approach, the one-stage approach directly predicts the object locations, shapes and the class labels without the proposal extraction stage, which can run in high efficiency. The community witnesses the noticeable improvements in this direction, including YOLO [37], SSD [32], DSSD [18], RefineDet [50], and RetinaNet [30].

Specifically, YOLO [37] formulates object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. After that, Redmon *et al.* [38] improve YOLO in various aspects, such as adding batch normalization on all of the convolutional layers, using anchor boxes to predict bounding boxes, and using multi-scale training. SSD [32] takes advantage of a set of default anchor boxes with different aspect ratios and scales to discretize the output space of bounding boxes and fuses predictions from multiple feature maps with different resolutions. DSSD [18] augments SSD with deconvolution layers to introduce additional large scale context in object detection to improve accuracy, especially for small objects. Zhang *et al.* [51] enrich the semantics of object detection features within SSD, by a semantic segmentation branch and a global activation module. Lin *et al.* [30] use Focal Loss (RetinaNet) to address the class imbalance issue in object detection by reshaping the standard cross entropy loss such that it down-weights the loss assigned to well-classified examples. In addition, Zhang *et al.* [50] propose a single-shot detector RefineDet. It is formed by two inter-connected modules, *i.e.*, the anchor refinement module and the object detection module, which achieves high accuracy and efficiency. Moreover, Chen *et al.* [6] propose a dual refinement network to boost the performance of the one-stage detectors, which considers anchor refinement and feature offset refinement in the anchor-offset detection.

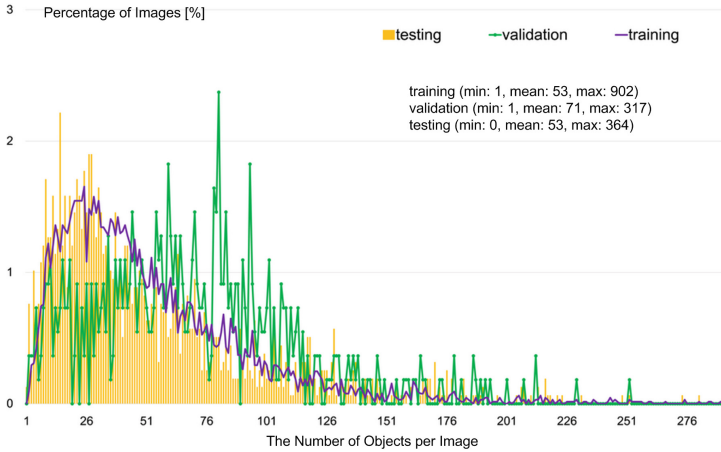


**Fig. 1.** The number of objects with different occlusion degrees of different object categories in the training, validation and testing sets for the object detection in images task.

### 3 The VisDrone-DET2018 Challenge

As mentioned above, to track and advance the developments in object detection, we designed the VisDrone-DET2018 challenge, which focuses on detecting ten predefined categories of objects (*i.e.*, *pedestrian*, *person*<sup>1</sup>, *car*, *van*, *bus*,

<sup>1</sup> If a human maintains standing pose or walking, we classify it as a *pedestrian*; otherwise, it is classified as a *person*.



**Fig. 2.** The number of objects per image *vs.* percentage of images in the **training**, **validation** and **testing** sets for object detection in images. The maximal, mean and minimal number of objects per image in the three subsets are presented in the legend.

*truck, motor, bicycle, awning-tricycle, and tricycle*) in images from drones. We require each participating algorithm to predict the bounding boxes of objects in predefined classes with a real-valued confidence. Some rarely occurring special vehicles (*e.g.*, machinshop truck, forklift truck, and tanker) are ignored in the evaluation. The VisDrone-DET2018 dataset consists of 8,599 images (6,471 for training, 548 for validation, 1,580 for testing) with rich annotations, including object bounding boxes, object categories, occlusion, and truncation ratios. Featuring a diverse real-world scenarios, the dataset was collected using various drone platforms (*i.e.*, drones with different models), in different scenarios (across 14 different cities spanned over thousands of kilometres), and under various weather and lighting conditions. The manually annotated ground truths in the **training** and **validation** sets are made available to users, but the ground truths of the **testing** set are reserved in order to avoid (over)fitting of algorithms. We encourage the participants to use the provided training data, while also allow them to use additional training data. The use of external data must be indicated during submission.

### 3.1 Dataset

The dataset and annotations presented in this workshop are expected to be a significant contribution to the community. As mentioned above, we have collected and annotated the benchmark dataset consisting of 8,599 images captured by drone platforms in different places at different heights, which is much larger than any previously published drone-based dataset. Specifically, we manually annotated more than 540k bounding boxes of targets of ten predefined categories. Some example images are shown in Fig. 3. We present the number of objects



**Fig. 3.** Some annotated example images of the object detection in images task. The dashed bounding box indicates the object is occluded. Different bounding box colors indicate different classes of objects. For better visualization, we only display some attributes. (Color figure online)

with different occlusion degrees of different object categories in the **training**, **validation**, and **testing** sets in Fig. 1, and plot the number of objects per image *vs.* percentage of images in each subset to show the distributions of the number of objects in each image in Fig. 2. The images of the three subsets are taken at different locations, but share similar environments and attributes.

In addition, we provide two kinds of useful annotations, occlusion ratio and truncation ratio. Specifically, we use the fraction of pixels being occluded to define the occlusion ratio, and define three degrees of occlusions: no occlusion (occlusion ratio 0%), partial occlusion (occlusion ratio 1%–50%), and heavy occlusion (occlusion ratio >50%). Regarding truncation ratio, it is used to indicate the degree of object parts that appear outside a frame. If an object is not fully captured within a frame, we annotate the bounding box inside the frame boundary and estimate the truncation ratio based on the region outside the image. It is worth mentioning that a target is skipped during evaluation if its truncation ratio is larger than 50%.

### 3.2 Evaluation Protocol

We require each participating algorithm to output a list of detected bounding boxes with confidence scores for each test image. Following the evaluation protocol in MSCOCO [31], we use the  $AP^{IoU=0.50:0.05:0.95}$ ,  $AP^{IoU=0.50}$ ,  $AP^{IoU=0.75}$ ,  $AR^{\max=1}$ ,  $AR^{\max=10}$ ,  $AR^{\max=100}$  and  $AR^{\max=500}$  metrics to evaluate the results of detection algorithms. These criteria penalize missing detection of objects as well as duplicate detections (two detection results for the same object instance). Specifically,  $AP^{IoU=0.50:0.05:0.95}$  is computed by averaging over all 10 Intersection over Union



(IoU) thresholds (*i.e.*, in the range [0.50 : 0.95] with the uniform step size 0.05) of all categories, which is used as the primary metric for ranking.  $AP^{IoU=0.50}$  and  $AP^{IoU=0.75}$  are computed at the single IoU thresholds 0.5 and 0.75 over all categories, respectively. The  $AR^{\max=1}$ ,  $AR^{\max=10}$ ,  $AR^{\max=100}$  and  $AR^{\max=500}$  scores are the maximum recalls given 1, 10, 100 and 500 detections per image, averaged over all categories and IoU thresholds. Please refer to [31] for more details.

## 4 Results and Analysis

### 4.1 Submitted Detectors

There are 34 different object detection methods from 31 different institutes submitted to the VisDrone-DET2018 challenge. The VisDrone committee also reports the results of the 4 baseline methods, *i.e.*, FPN (A.35) [29], R-FCN (A.36) [7], Faster R-CNN (A.37) [40], and SSD (A.38) [32]. For these baselines, the default parameters are used or set to reasonable values. Thus, there are 38 algorithms in total included in the VisDrone-DET2018 challenge. We present a brief overview of the entries and provide the algorithm descriptions in Appendix A.

Nine submitted detectors improve the Faster R-CNN method [40], namely JNU\_Faster RCNN (A.5), Faster R-CNN3 (A.7), MMN (A.9), CERTH-ODI (A.13), MFaster-RCNN (A.14), Faster R-CNN2 (A.16), IITH DODO (A.18), Faster R-CNN+ (A.19), and DPNet (A.34). Seven detectors are based on the FPN method [29], including FPN+ (A.1), DE-FPN (A.3), DFS (A.4), FPN2 (A.11), DDFPN (A.17), FPN3 (A.21), and DenseFPN (A.22). Three detectors are inspired from RetinaNet [30], including Keras-RetinaNet (A.27), RetinaNet2 (A.28), and HAL-Retina-Net (A.32). Three detectors, *i.e.*, RefineDet+ (A.10), RD<sup>4</sup>MS (A.24), and R-SSRN (A.30), are based on the RefineDet method [50]. Five detectors, *i.e.*, YOLOv3+ (A.6), YOLOv3++ (A.12), YOLOv3.DP (A.26), MSYOLO (A.29) and SODLSY (A.33), are based on the YOLOv3 method [39]. CFE-SSDv2 (A.15) is based on the SSD method [32]. SOD (A.23) is based on the R-FCN method [7]. L-H RCNN+ (A.25) is modified from the light-head RCNN method [28]. AHOD (A.31) is a feature fusion backbone network with the capability of modeling geometric transformations. MSCNN (A.20) is formed by two sub-networks: a multi-scale object proposal network (MS-OPN) [4] and an accurate object detection network (AODN) [5]. YOLO-R-CNN (A.2) and MMF (A.8) are the combinations of YOLOv3 and Faster R-CNN. We summarize the submitted algorithms in Table 3.

### 4.2 Overall Results

The overall results of the submissions are presented in Table 2. As shown in Table 2, we find that HAL-Retina-Net (A.32) and DPNet (A.34) are the only two algorithms achieving more than 30% AP score. HAL-Retina-Net (A.32) uses the SE module [27] and downsampling-upsampling [46] to learn channel attention

**Table 2.** Object detection results on the VisDrone-DET2018 **testing** set. The submitted algorithms are ranked based on the AP score. \* indicates that the detection algorithm is submitted by the committee.

Method	AP[%]	AP <sub>50</sub> [%]	AP <sub>75</sub> [%]	AR <sub>1</sub> [%]	AR <sub>10</sub> [%]	AR <sub>100</sub> [%]	AR <sub>500</sub> [%]
HAL-Retina-Net	<b>31.88</b>	46.18	<b>32.12</b>	0.97	7.50	34.43	<b>90.63</b>
DPNet	30.92	<b>54.62</b>	31.17	1.05	8.00	<b>36.80</b>	50.48
DE-FPN	27.10	48.72	26.58	0.90	6.97	33.58	40.57
CFE-SSDv2	26.48	47.30	26.08	1.16	<b>8.76</b>	33.85	38.94
RD <sup>4</sup> MS	22.68	44.85	20.24	1.55	7.45	29.63	38.59
L-H RCNN+	21.34	40.28	20.42	1.08	7.81	28.56	35.41
Faster R-CNN2	21.34	40.18	20.31	1.36	7.47	28.86	37.97
RefineDet+	21.07	40.98	19.65	0.78	6.87	28.25	35.58
DDFPN	21.05	42.39	18.70	0.60	5.67	28.73	36.41
YOLOv3_DP	20.03	44.09	15.77	0.72	6.18	26.53	33.27
MFaster-RCNN	18.08	36.26	16.03	1.39	7.78	26.41	26.41
MSYOLO	16.89	34.75	14.30	0.93	5.98	23.01	26.35
DFS	16.73	31.80	15.83	0.27	2.97	26.48	36.26
FPN2	16.15	33.73	13.88	0.84	6.73	23.32	30.37
YOLOv3+	15.26	33.06	12.50	0.68	5.77	21.15	23.83
IITH DODO	14.04	27.94	12.67	0.82	5.86	21.02	29.00
FPN3	13.94	29.14	11.72	0.81	6.08	22.98	22.98
SODLSY	13.61	28.41	11.66	0.60	5.20	19.26	23.68
FPN*	13.36	27.05	11.81	0.77	5.65	20.54	25.77
FPN+	13.32	26.54	11.90	0.84	5.87	22.20	22.20
AHOD	12.77	26.37	10.93	0.56	4.36	17.49	18.87
DFP	12.58	25.13	11.43	0.88	6.20	19.63	21.27
YOLO-R-CNN	12.06	27.98	8.95	0.50	4.39	19.78	23.05
MMN	10.40	20.66	9.43	0.41	5.22	18.28	19.97
YOLOv3++	10.25	21.56	8.70	0.48	4.31	15.61	15.76
Faster R-CNN+	9.67	18.21	9.54	1.19	6.74	16.40	16.40
R-SSRN	9.49	21.74	7.29	0.36	3.27	17.07	21.63
JNU_Faster RCNN	8.72	15.56	8.98	1.02	6.20	12.18	12.18
SOD	8.27	20.02	5.80	0.39	3.78	14.12	17.19
Keras-Retina-Net	7.72	12.37	8.68	0.62	5.65	10.76	10.80
MMF	7.54	16.53	6.03	1.28	5.91	14.28	14.36
R-FCN*	7.20	15.17	6.38	0.88	5.35	12.04	13.95
RetinaNet2	5.21	10.02	4.94	0.38	3.54	11.55	14.25
CERTH-ODI	5.04	10.94	4.12	<b>1.65</b>	5.93	9.05	9.05
Faster R-CNN3	3.65	7.20	3.39	0.64	2.41	10.08	21.85
Faster R-CNN*	3.55	8.75	2.43	0.66	3.49	6.51	6.53
MSCNN	2.89	5.30	2.89	0.59	2.18	9.33	15.38
SSD*	2.52	4.78	2.47	0.58	2.81	4.51	6.41

**Table 3.** The descriptions of the submitted algorithms in the VisDrone-DET2018 challenge. The tracking speed (in FPS), GPUs for training, backbone network, training datasets (I is imageNet, L is ILSVRC, P is COCO, V is VisDrone-DET2018 train set) and implementation details are reported. The \* mark is used to indicate the methods are submitted by the VisDrone committee.

Submission	Speed	GPUs	Backbone	Train data	Impl.
FPN+(A.1)	1	GTX 1080TI×6	ResNet-101	I,V	Python
YOLO-R-CNN(A.2)		GTX Titan XP×1		V	PyTorch
DE-FPN(A.3)		GTX 1080TI×4	ResNeXt-101	C,V	
DFS(A.4)		GTX Titan PASCAL×2	ResNet-101	I,V	Python
JNU_Faster RCNN(A.5)		GTX K80×4	ResNet-101	V	Python
YOLOv3+(A.6)	19.61	GTX TIT×1		C,V	Python
Faster R-CNN3(A.7)	0.14	GTX Titan Xp×2	ResNet-101	V	PyTorch
MMF(A.8)			ResNet-152 DarkNet-53	V	Python
MMN(A.9)	8.33	GTX 1080TI×2	ResNet-101	V	Python
RefineDet+(A.10)	10	GTX Titan X×4	VGG16	L,V	Caffe
FPN2(A.11)	1.8	GTX 1080TI×1	ResNet-50	I,V	Caffe
YOLOv3++(A.12)		GTX Titan XP×1	DarkNet-53	V	PyTorch
CERTH-ODI(A.13)	1	GTX 1070×1	Inception Resnet v2	V	Python
MFaster- RCNN(A.14)	1.7	GTX 1080TI×1	ResNet-101	V	PyTorch
CFE-SSDv2(A.15)	1	GTX Titan XP×4	VGG16	V	PyTorch
Faster R-CNN2(A.16)		GTX 1080×1	VGG16	I,V	Python
DDFPN(A.17)		GTX 1080TI×1	ResNet-101	I,V	Python C++
IITH DODO(A.18)	0.61	Tesla P-100×1	Inception ResNet-v2	C,V	Python
Faster R-CNN+(A.19)			VGG16	I,V	Python
MSCNN(A.20)		GTX 1080TI×1		V	Caffe Matlab
FPN3(A.21)			ResNet-50 ResNet-101	I,V	Python
DenseFPN(A.22)	8.33	GTX 1080TI×2	ResNet-101	I,V	Python
SOD(A.23)	7.25	GTX Titan X×1	VGG16	V	Python
RD <sup>4</sup> MS(A.24)		GTX Titan X×2	ResNet-50 SEResNeXt- 50	V	Caffe

**Table 3.** (continued)

L-H RCNN+(A.25)		GTX Titan X×1	ResNet-101	V	Python
YOLOv3-DP(A.26)	8.33	GTX Titan X×1		V	Python
Keras- RetinaNet(A.27)	1.28	GTX Titan X×1		V	Python
RetinaNet2(A.28)		GTX Titan X×1		V	Python
MSYOLO(A.29)		GTX 1080×1		V	Python
R-SSRN(A.30)	11.8	GTX 1080TI×1	VGG16	L,V	Python
AHOD(A.31)		GTX Titan X×1		V	Python
HAL-Retina- Net(A.32)	4	GTX Titan XP×6	SE- ResNeXt- 50	C,V	Caffe
SODLSY(A.33)	9	GTX 1080TI×1		V	
DPNet(A.34)		GTX Titan XP×8	ResNet-50 ResNet-101 ResNeXt	V	Caffe2
FPN* (A.35)	8	Tesla P100×1	ResNet-101	V	Python
R-FCN* (A.36)	7.3	GTX Titan X×1	ResNet-101	V	Python
Faster R-CNN* (A.37)	7	GTX Titan X×1	VGG16	V	Python
SSD* (A.38)	19	GTX Titan X×1	VGG16	V	Python

and spatial attention. DPNet (A.34) employs the framework of FPN [29] to capture context information in different scales of feature maps. DE-FPN (A.3) and CFE-SSDv2 (A.15) rank in the third and fourth places with more than 25% AP score, respectively. We also report the detection results of each object category in Table 4. As shown in Table 4, we observe that all the top three results of different kinds of objects are produced by the detectors with top four AP scores (see Table 2), *i.e.*, HAL-Retina-Net (A.32), DPNet (A.34), DE-FPN (A.3), and CFE-SSDv2 (A.15).

Among the 4 baseline methods provided by the VisDrone committee, FPN (A.35) achieves the best performance, SSD (A.38) performs the worst, and R-FCN (A.36) performs better than Faster R-CNN (A.37). These results of the algorithms are consistent with that in the MS COCO dataset [31].

- SSD (A.38) performs worst, only producing 2.52% AP score. CFE-SSDv2 (A.15) is an improvement of SSD (A.38), which uses a new comprehensive feature enhancement mechanism to highlight the weak features of small objects and adopts the multi-scale testing to further improve the performance. Specifically, it brings a significant improvement on AP score (*i.e.*, 26.48%), ranking the fourth place.
- Faster R-CNN (A.37) performs slightly better than 2.89% AP. DpNet (A.34) uses three Faster R-CNN models to detect different scales of objects. Specifically, the authors train FPN [29] architecture based Faster R-CNN models with multiple scales (*i.e.*,  $1000 \times 1000$ ,  $800 \times 800$ ,  $600 \times 600$ ), achieving the second best AP score (30.92%). Faster R-CNN2 (A.16) and Faster R-CNN+ (A.19) design the size of anchors to adapt to the distribution of objects, producing 21.34% and 9.67% AP score, respectively. MFaster-RCNN (A.14) replaces the ROI pooling layer with ROI align layer proposed in Mask R-CNN [23] to get better results for small object detection, *i.e.*, obtaining 18.08% AP score.
- R-FCN (A.36) achieves much better performance than SSD and Faster R-CNN, *i.e.*, producing 7.20% AP. However, its accuracy is still not satisfactory. SOD (A.23) use the pyramid-like prediction network for RPN and R-FCN [7] to improve object detection performance. In this way, the predictions made by higher level feature maps contain stronger contextual semantics while the lower level ones integrate more localized information at finer spatial resolution. It generates 0.93% high AP score than R-FCN (A.36), *i.e.*, 8.27% *vs.* 7.20%.
- FPN (A.35) performs the best among the 4 baseline methods by achieving 13.36 AP score, ranking in the middle of all submissions. We speculate that the extracted semantic feature maps at all scales are effective to deal with the objects with various scales. To further improve the accuracy, DE-FPN (A.3) enhances the data augmentation part by image cropping and color jitter, achieving 27.10% AP, ranking the third place. DDFPN (A.17) uses the DBPN [22] super resolution network to up-sample the image, producing 21.05% AP. FPN2 (A.11) implements an additional keypoint classification module to help locate the object, improving 2.79% AP score comparing to FPN (A.35).

### 4.3 Discussion

As shown in Table 3, we find that 18 detectors perform better than all the baseline methods. The best detector HAL-Retina-Net (A.32) achieves 31.88% AP score, which is still far from satisfactory in real applications. In the following, we discuss some critical issues in object detection on drone platforms.

**Table 4.** The  $AP_{IoU=0.50:0.05:0.95}$  scores on the VisDrone2018 testing set of each object category. \* indicates the detection algorithms submitted by the VisDrone committee. The top three results are highlighted in bold, italic and underline fonts.

Detectors	Ped.	People	Bicycle	Car	Van	Truck	Tricycle	Awn.	Bus	Motor
FPN+	26.54	24.58	22.29	19.40	15.82	11.90	8.00	3.78	0.84	0.03
YOLO-R-CNN	27.98	24.88	21.41	17.47	13.22	8.95	4.75	1.67	0.26	0.01
DE-FPN	<i>48.72</i>	<i>46.54</i>	<i>43.42</i>	<u>39.26</u>	<u>33.60</u>	<u>26.58</u>	<u>18.64</u>	<u>10.71</u>	3.34	0.15
DFS	31.80	29.96	27.64	24.46	20.57	15.83	10.43	5.20	1.38	0.06
JNU_Faster RCNN	15.56	14.68	13.66	12.22	10.70	8.98	6.59	3.76	0.99	0.03
YOLOv3+	33.06	29.90	26.48	22.19	17.71	12.50	7.37	2.89	0.46	0.01
Faster R-CNN3	40.18	37.85	34.92	30.93	26.14	20.31	13.72	7.16	2.10	0.12
MMF	16.53	15.07	13.25	10.85	8.47	6.03	3.56	1.42	0.23	0.01
MMN	20.66	18.74	16.80	14.50	12.10	9.43	6.61	3.77	1.29	0.06
RefineDet+	40.98	38.11	34.57	30.01	25.11	19.65	13.35	6.91	1.94	0.07
FPN2	33.73	30.73	27.16	23.08	18.55	13.88	8.98	4.32	1.05	0.04
YOLOv3++	21.56	19.85	17.87	15.11	11.98	8.70	5.08	1.96	0.32	0.01
CERTH-ODI	10.94	9.84	8.70	7.29	5.77	4.12	2.49	1.10	0.19	0.01
MFaster-RCNN	36.26	33.52	30.07	25.92	21.3	16.03	10.56	5.60	1.50	0.04
CFE-SSDv2	<u>47.30</u>	<u>45.23</u>	<u>42.40</u>	38.37	32.89	26.08	18.33	10.17	<u>3.69</u>	<i>0.29</i>
Faster R-CNN2	7.20	6.63	5.99	5.28	4.44	3.39	2.20	1.07	0.23	0.01
DDFPN	42.39	39.42	35.84	31.08	25.42	18.7	11.54	5.13	0.99	0.02
IITH DODO	27.94	25.72	23.04	19.94	16.57	12.67	8.68	4.57	1.28	0.04
Faster R-CNN+	18.21	16.88	15.51	13.94	11.91	9.54	6.67	3.21	0.81	0.03
MSCNN	5.30	4.96	4.55	4.11	3.60	2.89	2.09	1.10	0.30	0.02
FPN3	29.14	27.04	24.08	20.36	16.33	11.72	7.11	3.05	0.54	0.01
DenseFPN	25.13	23.16	20.77	17.97	14.8	11.43	7.85	3.70	0.93	0.02
SOD	20.02	17.46	14.85	12.10	9.00	5.80	2.67	0.71	0.08	0.00
RD <sup>4</sup> MS	44.85	41.74	37.87	32.97	27.11	20.24	13.38	6.67	1.85	0.08
L-H RCNN+	40.28	37.69	34.41	30.61	25.75	20.42	14.16	7.62	2.40	0.13
YOLOv3-DP	44.09	40.16	35.49	29.69	23.08	15.77	8.53	2.99	0.47	0.01
Keras-RetinaNet	12.37	12.10	11.58	10.97	10.07	8.68	6.61	3.79	1.02	0.03
RetinaNet	10.02	9.29	8.43	7.39	6.25	4.94	3.42	1.85	0.47	0.01
MSYOLO	34.75	32.37	29.32	25.31	20.18	14.3	8.54	3.53	0.57	0.01
R-SSRN	21.74	19.33	16.75	13.74	10.55	7.29	3.93	1.37	0.19	0.01
AHOD	26.37	24.45	21.88	18.68	14.82	10.93	6.81	3.13	0.58	0.01
HAL-Retina-Net	46.18	44.34	42.24	<i>39.63</i>	<i>36.27</i>	<b>32.12</b>	<b>26.87</b>	<b>20.88</b>	<b>16.01</b>	<b>14.24</b>
SODLSY	28.41	25.96	23.06	19.65	15.69	11.66	7.46	3.47	0.71	0.02
DPNet	<b>54.62</b>	<b>52.46</b>	<b>49.31</b>	<b>45.06</b>	<b>38.97</b>	<i>31.17</i>	<i>21.79</i>	<i>11.85</i>	<i>3.78</i>	<u>0.17</u>
FPN*	27.05	25.03	22.38	19.32	15.73	11.81	7.75	3.71	0.84	0.03
R-FCN*	15.17	13.59	12.09	10.58	8.8	6.38	3.76	1.39	0.19	0.01
Faster R-CNN*	8.75	7.62	6.53	5.03	3.72	2.43	1.08	0.32	0.04	0.00
SSD*	4.78	4.47	4.13	3.69	3.10	2.47	1.64	0.73	0.14	0.00

**Large Scale Variations.** As shown in Fig. 3, the objects have a substantial difference in scales, even for the objects in the same category. For example, as shown in the top-left of Fig. 3, cars on the bottom of the image appear larger than cars on the top-right side of the image. This factor greatly challenges the performance of the detectors. For better performance, it is necessary to redesign the anchor scales to adapt to scales of objects in the dataset, and it is also interesting to design an automatic mechanism to handle the objects with large scale variations in object detection. Meanwhile, fusing multi-level convolutional features to integrate contextual semantic information is also effective to handle scale variations, just like the architecture in FPN (A.35). In addition, multi-scale testing and model ensemble are effective to deal with the scale variations.

**Occlusion.** Occlusion is one of the critical issues challenging the detection performance, especially in our VisDrone2018 dataset (see Fig. 3). For example, as shown in Fig. 1, most of the instances in *bus* and *motor* categories, are occluded by other objects or background obstacles, which greatly hurt the detection performance. Specifically, the best detector HAL-Retina-Net (A.32) only produces less than 20% AP scores in these two categories. All the other detectors even produces less than 1% AP score on the *motor* class. In summary, it is important and urgent to design an effective strategy to solve the occlusion challenge to improve the detection performance.

**Class Imbalance.** Class imbalance is another issue of object detection. As shown in Fig. 1, there are much less *awning-tricycle*, *tricycle*, and *bus* instances in the **training** set than the instances in the *car* and *pedestrian* classes. Most of the detectors perform much better on the *car* and *pedestrian* classes than on the *awning-tricycle*, *tricycle*, and *bus* classes. For example, DPNet (A.34) produces 45.06% and 54.62% APs on the *car* and *pedestrian* classes, while only produces 11.85%, 21.79%, and 3.78% APs on the *awning-tricycle*, *tricycle*, and *bus* classes, see Table 4 for more details. The most straightforward and common approach is using the sampling strategy to balance the samples in different classes. Meanwhile, some methods (*i.e.*, Keras-RetinaNet (A.27), RetinaNet2 (A.28)) integrate the weights of different object classes in the loss function to handle this issue, such as Focal Loss [30]. How to solve the class imbalance issue is still an open problem.

## 5 Conclusions

This paper reviews the VisDrone-DET2018 challenge and its results. The challenge contains a large-scale drone-based object detection dataset, including 8,599 images (6,471 for training, 548 for validation, and 1,580 for testing) with rich annotations, including object bounding boxes, object categories, occlusion status, truncation ratios, etc. A set of 38 detectors have been evaluated on the released dataset. A large percentage of them have been published in recent top conferences and journals, such as ICCV, CVPR, and TPAMI, and some of them have not yet been published (available at arXiv). The top three detectors are HAL-Retina-Net (A.32), DPNet (A.34), and DE-FPN (A.3), achieving 31.8%, 30.92%, and 27.10% APs, respectively.

The VisDrone-DET2018 primary objective is to establish a community-based common platform for discussion and evaluation of detection performance on drones. This challenge will not only serve as a meeting place for researchers in this area but also present major issues and potential opportunities. We hope the released dataset allows for the development and comparison of the algorithms in the object detection fields, and workshop challenge provides a way to track the process. Our future work will be focused on revising the evaluation kit, dataset, as well as including more challenging vision tasks on the drone platform, through the feedbacks from the community.

**Acknowledgements.** This work was supported in part by the National Natural Science Foundation of China under Grant 61502332 and Grant 61732011, in part by Natural Science Foundation of Tianjin under Grant 17JCZDJC30800, in part by US National Science Foundation under Grant IIS-1407156 and Grant IIS-1350521, and in part by Beijing Seetatech Technology Co., Ltd and GE Global Research.

## A Submitted Detectors

In this appendix, we provide a short summary of all algorithms participated in the VisDrone2018 competition. These are ordered according to the submissions of their final results.

### A.1 Improved Feature Pyramid Network (FPN+)

*Karthik Suresh, Hongyu Xu, Nitin Bansal, Chase Brown, Yunchao Wei, Zhangyang Wang, Honghui Shi*  
*k21993@tamu.edu, xuhongyu2006@gmail.com, bansa01@tamu.edu*  
*chasebrown42@tamu.edu, wychao1987@gmail.com, atlaswang@tamu.edu*  
*honghui.shi@ibm.com*

FPN+ is improved from the Feature Pyramid Network (FPN) model [29]. The main changes we made are concluded as follows: (1) We resize the input images with different scales; (2) We use more scales of smaller anchors; (3) We ensemble FPN models with different anchors and parameters; (4) We employ NMS as another post processing step to avoid box overlap and multi-scale testing. Specifically, we use a FPN with ResNet-101 pre-trained weights on ImageNet as the backbone. We also attempt to make some changes to the training data (resizing it to different shapes, cutting it into pieces, etc.).

### A.2 Fusion of Faster R-CNN and YOLOv3 (YOLO-R-CNN)

*Wenchi Ma, Yuanwei Wu, Usman Sajid, Guanghui Wang*  
*{wenchima, y262w558, usajid, ghwang}@ku.edu*



YOLO-R-CNN is basically a voting algorithm specifically designed for object detection. Instead of the widely used feature-level fusion for deep neural networks, our approach works at the detection-level. We train two different DCNN models, *i.e.*, Faster R-CNN [40] and YOLOv3 [39]. Then the final detection results are produced by voting, weighted averages of the two above models.

### A.3 Data Enhanced Feature Pyramid Network (DE-FPN)

*Jingkai Zhou, Yi Luo, Hu Lin, Qiong Liu*  
 {201510105876, 201721045510, 201721045497}@mail.scut.edu.cn  
 liuqiong@scut.edu.cn

DE-FPN is based on the Feature Pyramid Network (FPN) model [29] with data enhancement. Specifically, we enhance the training data by image cropping and color jitter. We use ResNeXt-101 64-4d as the backbone of FPN with COCO pre-trained model. We remove level 6 of FPN to improve small object detection.

### A.4 Focal Loss for Object Detection (DFS)

*Ke Bo*  
 kebo3@mail2.sysu.edu.cn

DFS is based on ResNet-101 and Feature Pyramid Networks [29]. The features from *Conv2.x* are also used to detect objects, which gains about 1% improvements in mAP. Our model use other techniques including multiple scale training and testing, deformable convolutions and Soft-NMS.

### A.5 Faster R-CNN by Jiangnan University (JNU\_Faster RCNN)

*Haipeng Zhang*  
 6161910043@vip.jiangnan.edu.cn

JNU\_Faster RCNN is based on the Faster R-CNN algorithm [40] to complete the detection task. The source code is from Github repository named faster-rcnn.pytorch<sup>2</sup>. We use trainset and valset of the VisDrone2018-DET dataset without additional training data to train this model. The pre-trained model is Faster R-CNN with ResNet-101 backbone.

### A.6 Improved YOLOv3 (YOLOv3+)

*Siwei Wang, Xintao Lian*  
 285111284@qq.com

YOLOv3+ is improved from YOLO [37]. Specifically, we use the VisDrone2018-DET train set and pre-trained models on the COCO dataset to fine-tune our model.

<sup>2</sup> <https://github.com/jwyang/faster-rcnn.pytorch>.

## A.7 Improved Faster R-CNN: (Faster R-CNN3)

*Yiling Liu, Ying Li*

*liulingyi601@mail.nwpu.edu.cn, lybyp@nwpu.edu.cn*

Faster R-CNN3 is based on Faster R-CNN [40]. We only use VisDrone2018 train set as the training set. Our algorithm is implemented in TITAN XP×2, Ubuntu, pytorch. The testing speed is about 7s per image. The based network of Faster R-CNN is ResNet-101.

## A.8 The Object Detection Algorithm Based on Multi-Model Fusion (MMF)

*Yugin Zhang, Weikun Wu, Zhiyao Guo, Minyu Huang*

*{23020161153381,23020171153097}@stu.xmu.edu.cn*

*{23020171153021,23020171153029}@stu.xmu.edu.cn*

MMF is a multi-model fusion based on Faster-RCNN [40] and YOLOv3 [39]. The Faster-RCNN algorithm is a modification of a published one<sup>3</sup>. We re-write the codes and re-set the parameters including learning rate, gamma, step size, scales, anchors and ratios. We use the ResNet-152 as the backbone. The YOLOv3 algorithm is also a modification of a published one<sup>4</sup>. We modify the anchor setting by the K-means++ algorithm.

Since the number of objects in different categories is very unbalanced in the train set, we adopt the multi-model fusion method to improve the accuracy. Specifically, the car category is trained using the Faster R-CNN algorithm and the rest categories are trained using the YOLOv3 algorithm. Moreover, the rest categories are divided into two types: one for *pedestrian* and *people*, and the other one for *bicycle*, *van*, *truck*, *tricycle*, *awning-tricycle*, *bus* and *motor*. Finally, the detection result is determined by the three models.

## A.9 Multi-Model Net Based on Faster R-CNN (MMN)

*Xin Sun*

*sunxin@ouc.edu.cn*

MMN is based on the Faster R-CNN network [40]. We first crop the train images into small size to avoid the resize operation. Then there cropped images are used to train different Faster R-CNN networks. Finally we merge the results to obtain the best classification result.

<sup>3</sup> <https://github.com/endernewton/tf-faster-rcnn>.

<sup>4</sup> <https://github.com/AlexeyAB/darknet>.

### A.10 An improved Object Detector Based on Single-Shot Refinement Neural Network (RefineDet+)

*Kaiwen Duan, Honggang Qi, Qingming Huang*  
*duankaiwen17@mails.ucas.ac.cn, hgqi@jdl.ac.cn, qmhuang@ucas.ac.cn*

RefineDet+ improves the single-shot refinement Neural Network (RefineDet) [50] by proposing a new anchor matching strategy. Our anchor matching strategy is based on center point translation of anchors (CPTMatching). During the training phase, the detector needs to determine which anchors correspond to an object bounding box. RefineDet firstly matches each object to the anchor with the highest jaccard overlap and then matches each anchor to an object with jaccard overlap higher than a threshold (usually 0.5). However, some nearby anchors whose jaccard overlap lower than the threshold may also help the bounding box regression. In our CPTMatching, we first select bounding boxes predicted by the anchor refinement module (ARM) [50] to have a jaccard overlap with any object ground-truth higher than 0.5. For each selected bounding box, we compute a measurement  $\beta$ , which is a ratio of the center point distance between its corresponding anchor and its matched ground-truth box to the scale of its anchor. Discard those anchors whose  $\beta$  are larger than a threshold. The remaining anchors are called potential valid anchors. Finally, we align each center point of those potential valid anchors to the center of their nearest ground-truth boxes. Anchors are preserved if their jaccard overlap higher than 0.6 with the aligned ground-truth.

### A.11 An improved Object Detector Based on Feature Pyramid Networks (FPN2)

*Zhenwei He, Lei Zhang*  
*{hzw, leizhang}@cqu.edu.cn*

FPN2 is based on the Feature Pyramid Networks (FPN) object detection framework [29]. To obtain better detection results, we improve the original FPN in three folds:

- **Data expansion.** We extend the training set by clipping the images. The clipped images contain at least one object. New pictures have different proportions of the original pictures. In our implementation, the proportions to the width or height are set as 0.5 and 0.7, which results in totally 4 kinds of ratios ( $[0.5, 0.5]$ ,  $[0.5, 0.7]$ ,  $[0.7, 0.7]$ ,  $[0.7, 0.5]$  to the width and height, respectively). As a result, the extended datasets has 5 times number of training pictures compared to the original dataset.
- **Keypoint classification.** We implement an auxiliary keypoint classification task to further improve the detection accuracy. The bounding box is the border line of the foreground and background, therefore, we suppose the 4 corners and the center of the bounding box are the keypoints of the corresponding

object. 4 corners of the bounding box are annotated as background while the center is annotated as the category of the corresponding object in our implement.

- **Fusion of different models.** We train our deep model with different expanded datasets to obtain different models. First, we implement the NMS to generate the detection results of the each deep models. Then, we count the number of bounding boxes with the score greater than the threshold from different deep models. If the number is more than half of the deep models, we will keep the bounding box; otherwise we will discard it. Finally, we perform NMS again to generate the final detection results.

### A.12 Modified YOLOv3 (YOLOv3++)

*Yuanwei Wu, Wenchi Ma, Usman Sajid, Guanghui Wang*  
 {y262w558,wenchima,usajid,ghwang}@ku.edu

YOLOv3++ is based on YOLOv3 [39], which is a one stage detection method without using object proposals. We follow the default setting in YOLOv3 during training. To improve the object detection performance, we conduct experiments by increasing network resolution in inference and training time, and recalculating the anchor box priors on VisDrone dataset. We only use the provided training dataset to train YOLOv3 without adding additional training data, and evaluate the algorithm performance on the validation dataset. Then, the training and validation datasets are combined together to train a new YOLOv3 model, and the predicted classes probabilities and bounding boxes position on the testing dataset are submitted as our final submission.

### A.13 CERTH’s Object Detector in Images (CERTH-ODI)

*Emmanouil Michail, Konstantinos Avgerinakis, Panagiotis Giannakeris, Stefanos Vrochidis, Ioannis Kompatsiaris*  
 {michem, koafgeri, giannakeris, stefanos, ikom}@iti.gr

CERTH-ODI is trained on the whole training set of the VisDrone2018-DET dataset. However, since pedestrian and cars were dominant, compared to other classes, in order to balance the training set, we remove several thousand cars and pedestrians annotations. For the training we use the Inception ResNet-v2 Faster R-CNN model pre-trained on the MSCOCO dataset. In order to provide more accurate results, we use a combination of different training set-ups: One with all the available object classes trained until 800,000 training steps, one with four-wheel vehicles only (*i.e.*, *car*, *van*, *truck*, *bus*, because they share similar characteristics) and one with the remaining classes. We apply each model separately on each image, and NMS on the results, and afterwards we merge all the resulting bounding boxes from the different training models. Subsequently, we reject overlapping bounding boxes with an IoU of 0.6, which is chosen empirically, excluding several combinations, like people-bicycle, people-motor that tends to high overlap.

### A.14 Modified Faster-RCNN for Small Objects Detection (MFaster-RCNN)

Wenrui He, Feng Zhu  
 {hewenrui,zhufeng}@bupt.edu.cn

MFaster-RCNN is improved from the Faster R-CNN model [40]. Our method only uses the VisDrone2018-DET train set with data augmentation, including cropping, zooming and flipping. We use pre-trained ResNet-101 as backbone due to GPU limit. The tuned hyper-parameters are mainly presented as follows: (1) The anchor ratio is adjusted from  $[0.5, 1, 2]$  to  $[0.5, 1.5, 2.5]$  which is calculated by K-means with training data. (2) The base size of the anchors remains 16 but the multiplicative scale is adjusted from  $[4, 8, 16]$  to  $[1, 2, 4, 8, 16]$  to detect very small objects. (3) The RPN positive overlap threshold which decides whether the proposal is regarded as a positive sample to train the RPN is adjusted from 0.7 to 0.5, while the RPN negative overlap threshold is adjusted from 0.3 to 0.2. (4) the foreground and background thresholds for the Fast R-CNN part is 0.5 and 0.1, respectively. The foreground fraction is adjusted from 0.25 to 0.4 as we find these values perform the best in practice. (5) The maximal number of the groundtruth boxes allowed to use for training in one input image is adjusted from 20 to 60 as we have enormous training samples per image in average.

### A.15 SSD with Comprehensive Feature Enhancement (CFE-SSDv2)

Qijie Zhao, Feng Ni, Yongtao Wang  
 {zhaoqijie,nifeng,wyt}@pku.edu.cn

CFE-SSDv2 is an end-to-end one-stage object detector with specially designed novel module, namely Comprehensive Feature Enhancement (CFE) module. We first improve the original SSD model [32] by enhancing the weak features for detecting small objects. Our CFE-SSDv2<sup>5</sup> is designed to enhance detection ability for small objects. In addition, we apply multi-scale inference strategy. Although training on input size of  $800 \times 800$ , we have broadened the input size to  $2200 \times 2200$  when inferencing, leading to further improvement in detecting small objects.

### A.16 Faster R-CNN based object detection (Faster R-CNN2)

Fan Zhang  
 zhangfan\_1@stu.xidian.edu.cn

Faster R-CNN2 depends on the VisDrone2018-DET dataset, Faster R-CNN [40], and adjusts some parameters. For example, we add a small anchor scale  $64^2$  to detect small objects and reducing the mini-batch size from 256 to 128.

<sup>5</sup> <https://github.com/qijiezhao/CFENet>.

### A.17 DBPN+Deformable FPN+Soft NMS (DDFPN)

*Liyu Lu*

*coordinate@tju.edu.cn*

DDFPN is designed for small object detection. Since the dataset contains a large amount of small objects, so we scale up the original image first and then detect the objects. We use the DBPN [22] super resolution network to upsample the image. The model used for the detection task is Deformable FPN [8, 29]. Besides, we use Soft-NMS [3] as our non-maximum suppression algorithm.

For network training, we first divide the input image into patches with size of  $1024 \times 1024$ , and obtain 23,602 training images and their corresponding labels as training set to train Deformable FPN. Our training process uses OHEM training methods [42]. The learning rate we use in training is 0.001, and the image input size we use for training is  $1024 \times 1024$ . ResNet-101 is used as the backbone and the weights are initialized using model pre-trained on Image-Net.

For network testing, we use the same method as the training set to divide the test image into patches with size of  $512 \times 512$ . Next, we up-sample the previously obtained test patches to  $1024 \times 1024$  via the DBPN network. Then we send these testing patches to our trained Deformable FPN to obtain  $1024 \times 1024$  results. In fact, the size of image corresponds to the size of the original image is  $512 \times 512$ . Since the results in different scales are consistent with the characteristics of visual blind spots, we use multi-scale images for testing purpose, *i.e.*, [688, 688], [800, 800], [1200, 1200], [1400, 1400], [1600, 1600], [2000, 2000]. Finally, we merge the results in each scale derived from the same image back into one single image, hence we obtain the final test results.

### A.18 IIT-H Drone Object DetectiOn (IITH DODO)

*Nehal Mamgain, Naveen Kumar Vedurupaka, K. J. Joseph, Vineeth N. Balasubramanian*

*cs17mtech11023@iith.ac.in, naveenkumarvedurupaka@gmail.com*

*{cs17m18p100001, vineethnb}@iith.ac.in*

IITH DODO is based on the Faster R-CNN architecture [40]. Faster R-CNN has a Region Proposal Network which is trained end-to-end and shares convolutional features with the detection network thus ameliorating the computational cost of high-quality region proposals. Our model uses the Inception ResNet-v2 [43] backbone for Faster R-CNN, pre-trained on the COCO dataset. The anchor sizes are adapted to improve the performance of the detector on small objects. To reduce the complexity of the model, only anchors of single aspect ratio are used. Non-maximum suppression is applied both on the region proposals and final bounding box predictions. Atrous convolutions are also used. No external data has been used for training and no test-time augmentation is performed. The performance is the result of the detection pipeline with no ensemble used.

### A.19 Adjusted Faster Region-Based Convolutional Neural Networks (Faster R-CNN+)

*Tiaojiao Lee, Yue Fan, Han Deng, Lin Ma, Wei Zhang*  
 {tianjiao.lee, fanyue}@mail.sdu.edu.cn, 67443542@qq.com  
 forest.linma@gmail.com, davidzhang@sdu.edu.cn

Faster R-CNN+ basically follows the original algorithm of Faster R-CNN [40]. However, we make a few adjustments on Faster R-CNN algorithm to adapt to the VisDroneDet dataset. The dataset given consists of many variant-sized proposals which leads to a multi-scale object detection problem. In order to mitigate the impact of relatively rapid changes in scales of bounding boxes, we add more anchors with large sizes to fit those larger objects and keep small anchors unchanged for detecting tiny objects such as people and cars in long distance. Moreover, the VisDroneDet dataset has an unbalanced object distribution. When testing on validation dataset, we find that classification performance for car is much better than others for the reason that the appearance of cars is more frequent. To alleviate this problem, we mask out some car bounding boxes by hand for pursuing better classification performance.

### A.20 Multi-Scale Convolutional Neural Networks (MSCNN)

*Dongdong Li, Yangliu Kuai, Hao Liu, Zhipeng Deng, Juanping Zhao*  
 moqimubai@sina.cn

MSCNN is a unified and effective deep CNN based approach for simultaneously detecting multi-class objects in UAV images with large scales variability. Similar to Faster R-CNN, our method consists of two sub-networks: a multi-scale object proposal network (MS-OPN) [4] and an accurate object detection network (AODN) [5]. Firstly, we redesign the architecture of feature extractor by adopting some recent building blocks, such as inception module, which can increase the variety of receptive field sizes. In order to ease the inconsistency between the sizes variability of objects and fixed filter receptive fields, MS-OPN is performed with several intermediate feature maps, according to the certain scale ranges of different objects. That is, the larger objects are proposed in deeper feature maps with highly-abstracted information, whereas the smaller objects are proposed in shallower feature maps with fine-grained details. The object proposals from various intermediate feature maps are combined together to form the outputs of MSOPN. Then those object proposals are sent to the AODN for accurate object detection. For detecting small objects appear in groups, AODN combines several outputs of intermediate layers to increase the resolution of feature maps, enabling small and densely packed objects to produce larger regions of strong response.

## A.21 Feature Pyramid Networks for Object Detection (FPN3)

*Chengzheng Li, Zhen Cui*  
*czhengli@njust.edu.cn, zhen.cui@njust.edu.cn*

FPN3 follows the Faster R-CNN [40] which uses the feature pyramid [29]. We make some modifications of the algorithm. First of all, since most of the objects in the VisDrone-DET2018 dataset are quite small, we add another stage feature based on the original  $P2$ - $P6$  layer, we take the output of *conv1* which not pass the pool layer in ResNet [25] as  $C1$ , then transform it into  $P1$  whose stride is  $1/2$  like what has done in FPN, the anchor size of this stage is 16, the additional stage is used to detect smaller objects in images. Secondly, we change the up-sample by nearest pixel which has no parameters into deconvolution layer which has parameters just like convolution layer, since the layers with parameters have better performance compared with those without parameters. In the training phase, we trained two model based on ResNet-50 and ResNet-101 respectively, all training images are artificially occluded and flipped to make the model more robust. In the testing phase, we combine the two results from ResNet-50 and ResNet-101 as the final results.

## A.22 Dense Feature Pyramid Net (DenseFPN)

*Xin Sun*  
*sunxin@ouc.edu.cn*

DenseFPN is inspired by Feature Pyramid Networks [29] to detect small objects on the VisDrone2018 dataset. In the original FPN, they use the low-level feature to predict small objects. We use the same strategy and fuse high-level and low-level features in a dense feature pyramid network. Meanwhile, we crop the training images into small size to avoid the resize operation. Then we merge the results to obtain the best detection result.

## A.23 SJTU-Ottawa-Detector (SOD)

*Lu Ding, Yong Wang, Chen Qian, Robert Laganière, Xinbin Luo*  
*dinglu@sjtu.edu.cn, ywang6@uottawa.ca, qian\_chen@sjtu.edu.cn*  
*laganier@eecs.uottawa.ca, losinbin@sjtu.edu.cn*

SOD employs a pyramid like predict network to detect objects with large range of scales because pyramid like representations are widely used in recognition systems for detecting objects at different scales [29]. The prediction made by higher level feature maps contains stronger contextual semantics while the lower level ones integrate more localized information at finer spatial resolution. These predictions are hierarchically fused together to make pyramid-like decisions. We use this pyramid-like prediction network for RPN and region fully convolutional networks (R-FCN) [7] to perform object detection.



### A.24 Ensemble of Four RefineDet Models with Multi-scale Deployment (RD<sup>4</sup>MS)

*Oliver Acatay, Lars Sommer, Arne Schumann*  
 {oliver.acatay, lars.sommer, arne.schumann}@iosb.fraunhofer.de

RD<sup>4</sup>MS is a variant of the RefineDet detector [50], using the novel Squeeze-and-Excitation Network (SENet) [27] as the base network. We train four variants of the detector: three with SEResNeXt-50 and one with ResNet-50 as base network, each with its own set of anchor sizes. Multi-scale testing is employed and the detection results of the four detectors are combined via weighted averaging.

### A.25 Improved Light-Head RCNN (L-H RCNN+)

*Li Yang, Qian Wang, Lin Cheng, Shubo Wei*  
 liyang16361@163.com, {844021514,2643105823,914417478}@qq.com

L-H RCNN+ modifies the published algorithm light-head RCNN [28]. Firstly, we modify the parameter “anchor\_scales”, replacing  $32 \times 32$ ,  $64 \times 64$ ,  $128 \times 128$ , and  $256 \times 256$ ,  $512 \times 512$  with  $16 \times 16$ ,  $32 \times 32$ ,  $64 \times 64$ ,  $128 \times 128$ , and  $256 \times 256$ . Secondly, we modify the parameter “max\_boxes\_of\_image”, replacing 50 with 600. Thirdly, we perform NMS for all detection objects that belong to the same category.

### A.26 Improved YOLOv3 with Data Processing (YOLOv3-DP)

*Qiuchen Sun, Sheng Jiang*  
 345412791@qq.com

YOLOv3-DP is based on the YOLOv3 model [39]. We process the images of the training set. Firstly, we remove some images including pedestrians and cars. Secondly, we increase the brightness of some lower brightness pictures to enhance the data. Thirdly, we black out the ignored regions in the image and cut the image to a size of  $512 \times 512$  with a step size of 400. The images without objects will be removed. Thus the final training set contains 31,406 images with the size of  $512 \times 512$ .

### A.27 RetinaNet implemented by Keras (Keras-RetinaNet)

*Qiuchen Sun, Sheng Jiang*  
 345412791@qq.com

Keras-RetinaNet is based on the RetinaNet [30], which is implemented by the Keras toolkit. The source codes can be found in the website: <https://github.com/facebookresearch/Detectron>.

## A.28 Focal Loss for Dense Object Detection (RetinaNet2)

*Li Yang, Qian Wang, Lin Cheng, Shubo Wei*  
*liyang16361@163.com, 844021514@qq.com*  
*2643105823@qq.com, 914417478@qq.com*

RetinaNet2 is based on the RetinaNet [30] algorithm. The short size of images is set as 800, and the maximum size of the image is set as 1,333. Each mini-batch has 1 image per GPU for training/testing.

## A.29 Multiple-scale yolo network (MSYOLO)

*Haoran Wang, Zexin Wang, Ke Wang, Xiufang Li*  
*18629585405@163.com, 1304180668@qq.com*

MSYOLO is the multiple scale YOLO network [39]. We divide these categories into three cases according to the scale of object categories. First of all, ignored regions and the *others* category is the first case for areas that are not trained. Second, since many categories are not in the same scale, we divide them into big objects and small objects on the basis of their scale of boxes. The big objects include *car, truck, van* and *bus*, and small objects contain *pedestrian, people, bicycle, motor, tricycle* and *awning-tricycle*. The big objects as the center of cut images have the scale of  $480 \times 480$ , and small objects have the scale of  $320 \times 320$ .

## A.30 Region-Based Single-Shot Refinement Network (R-SSRN)

*Wenzhe Yang, Jianxiu Yang*  
*wzyang@stu.xidian.edu.cn, jxyang\_xidian@outlook.com*

R-SSRN is based on the deep learning method called RefineDet [50]. We do modifications as follows: (1) We remove the deep convolutional layers after *fc7* because they are useless for the VisDrone small objects detection; (2) We added additional small scales default boxes at *conv3\_3* and set new aspect ratios by using k-means cluster algorithm on the VisDrone dataset. The change of scales and aspect ratios can help default boxes more suitable for the objects; (3) Due to the small and dense objects, we split each image to 5 sub images (*i.e.*, bottom left, bot-tom right, middle, top left, top right), where the size of each sub image is 1/4 of that of original image. After testing the sub images, we merge them by using NMS.

## A.31 A Highly Accurate Object Detector in Drone Scenarios (AHOD)

*Jianqiang Wang, Yali Li, Shengjin Wang*  
*wangjian16@mails.tsinghua.edu.cn, liyali@ocrserv.ee.tsinghua.edu.cn*  
*wgsqj@tsinghua.edu.cn*

AHOD is a novel detection method with high accuracy in drone scenarios. First, a feature fusion backbone network with the capability of modelling geometric transformations is proposed to extract object features. Second, a special object proposal sub-network is applied to generate candidate proposals using multi-level semantic feature maps. Finally, a head network refines the categories and locations of these proposals.

### A.32 Hybrid Attention based Low-Resolution Retina-Net (HAL-Retina-Net)

*Yali Li, Zhaoyue Xia, Shengjin Wang*  
 {liyali13, wsggj}@tsinghua.edu.cn

HAL-Retina-Net is improved from Retina-Net [30]. To detect low-resolution objects, we remove  $P6$  and  $P7$  from the pyramid. Therefore the pyramid of the network includes three pathways, named as  $P1$ ,  $P3$ , and  $P5$ . We inherit the head design of Retina-Net. Furthermore, the post-processing steps include Soft-NMS [3] and bounding box voting. We find that bounding box voting improve the detection accuracy significantly. Furthermore, we note that by increasing the normalized size of images the improvement is also significant. To encourage the full usage of training samples, we split the images into patches with size  $640 \times 640$ . To avoid out-of-memory in detection, we use SE-ResNeXt-50 [27] as the backbone network and train the Retina-Net with the cropped sub-images. To further improve the detection accuracy, we add the hybrid attention mechanism. That is, we use additional SE module [27] and downsample-upsample [46] to learn channel attention and spatial attention. Our final detection results on test challenge are based on the ensemble of modified Retina-net with the above two kinds of attention.

### A.33 Small Object Detection in Large Scene based on YOLOv3 (SODLSY)

*Sujuan Wang, Yifan Zhang, Jian Cheng*  
 Wangsujuan@airia.cn, {yfzhang,jcheng}@nlpr.ia.ac.cn

SODLSY is used to detect objects in various weather and lighting conditions, representing diverse scenarios in our daily life. The maximum resolution of VOC images is  $469 \times 387$ , and  $640 \times 640$  for COCO images. However, the static images in VisDrone2018 are even  $2000 \times 1500$ . Our algorithm first increases the size of training images to 1184, ensuring the information of small objects is not lost during image resizing. Thus, we adopt multi-scale (800, 832, 864,  $\dots$ , 1376) training method to improve the detection results. We also re-generate the anchors for VisDrone-DET2018.

### A.34 Drone Pyramid Networks (DPNet)

*HongLiang Li, Qishang Cheng, Wei Li, Xiaoyu Chen, Heqian Qiu, Zichen Song  
hlli@uestc.edu.cn, cqs@std.uestc.edu.cn, weili.cv@gmail.com  
rychen9459@gmail.com, hqqiu@std.uestc.edu.cn, szc.uestc@gmail.com*

DPNet consists of three object detectors based on the Faster R-CNN [40] method, by Caffe2 deep learning framework, in parallel, on 8 GPUs. The design of DPNet follows the idea of FPN [29], whose feature extractors are ResNet-50 [25], ResNet101, and ResNeXt [48], respectively which are pre-trained on ImageNet only. To make full use of the data, the methods are designed as follows:

- No additional data other than the train + val dataset are used for network training.
- We train Faster-RCNN with FPN using multiple scales ( $1000\times 1000$ ,  $800\times 800$ ,  $600\times 600$ ) to naturally handle objects of various sizes, generating improvement of 4%.
- When selecting the prior boxes, we set multiple specific aspect ratios based on the scale distribution of the training data.
- We change the IOU threshold from 0.5 to 0.6 and removed the last FPN layer, yielding an improvement of 1.5%.

We use Soft-NMS [3] instead of the conventional NMS to select predicted boxes. We replace RoIPooling with RoIAlign [23] to perform feature quantification. We use multi-scale training and testing. On the validation set, our best single detector obtains mAP 49.6%, and the ensemble of three detectors achieves mAP 50.0%.

### A.35 Feature pyramid networks for object detection (FPN)

*Submitted by the VisDrone Committee*

FPN takes advantage of featurized image pyramids to construct deep convolutional networks with inherent multi-scale and pyramidal hierarchy. It combines low-resolution but semantically strong features and high-resolution but semantically weak features. Thus it exploits rich semantic information from all scales and is trained in an end-to-end way. The experimental results show this architecture can significantly improve the generic deep models in several fields. Please refer to [29] for more details.

### **A.36 Object Detection via Region-based Fully Convolutional Networks (R-FCN)**

*Submitted by the VisDrone Committee*

R-FCN is the region-based fully convolutional networks for object detection without ROI-wise sub-network. Different from previous methods such as Fast R-CNN and Faster R-CNN using a costly pre-region subnetwork, R-FCN addresses the dilemma between translation-invariance in image classification and translation-variance in object detection using the position-sensitive score maps. That is, almost all the computation is shared on the whole image. It also can adopt recent state-of-the-art classification network backbones (*e.g.*, ResNet and Inception) for better performance. Please refer to [7] for more details.

### **A.37 Towards Real-Time Object Detection with Region Proposal Networks (Faster R-CNN)**

*Submitted by the VisDrone Committee*

Faster R-CNN improves Fast R-CNN [20] by adding Region Proposal Network (RPN). RPN shares full-image convolutional features with the detection network in a nearly cost-free way. Specifically, it is implemented as a fully convolutional network that predict object bounding boxes and their scores at the same time. Given object proposals by the RPN, the Fast R-CNN model shares the convolutional features and then detect object efficiently. Please refer to [40] for more details.

### **A.38 Single Shot MultiBox Detector (SSD)**

*Submitted by the VisDrone Committee*

SSD is the one-stage object detection method based a single deep neural network without proposal generation. It uses a set of pre-set anchor boxes with different aspect ratios and scales, and then discretize the output space of bounding boxes. To deal with multi-scale object detection, the network combines predictions from several feature maps in different layers. Notably, it predicts the score of each object category and adjusts the corresponding bounding box simultaneously. The network is optimized via a multi-task loss including confidence loss and localization loss. Finally, the multi-scale bounding boxes are converted to the detection results using the NMS strategy. Please refer to [32] for more details.

## References

1. Agarwal, S., Awan, A., Roth, D.: Learning to detect objects in images via a sparse, part-based representation. *TPAMI* **26**(11), 1475–1490 (2004)
2. Andriluka, M., Roth, S., Schiele, B.: People-tracking-by-detection and people-detection-by-tracking. In: *CVPR*. IEEE Computer Society (2008)
3. Bodla, N., Singh, B., Chellappa, R., Davis, L.S.: Soft-NMS - improving object detection with one line of code. In: *ICCV*, pp. 5562–5570 (2017)
4. Cai, Z., Fan, Q., Feris, R.S., Vasconcelos, N.: A unified multi-scale deep convolutional neural network for fast object detection. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) *ECCV 2016*. LNCS, vol. 9908, pp. 354–370. Springer, Cham (2016). [https://doi.org/10.1007/978-3-319-46493-0\\_22](https://doi.org/10.1007/978-3-319-46493-0_22)
5. Cai, Z., Vasconcelos, N.: Cascade R-CNN: delving into high quality object detection. *CoRR abs/1712.00726* (2017)
6. Chen, X., Wu, Z., Yu, J.: Dual refinement network for single-shot object detection. *CoRR abs/1807.08638* (2018). <http://arxiv.org/abs/1807.08638>
7. Dai, J., Li, Y., He, K., Sun, J.: R-FCN: object detection via region-based fully convolutional networks. In: *NIPS*, pp. 379–387 (2016)
8. Dai, J., et al.: Deformable convolutional networks. In: *ICCV*, pp. 764–773 (2017)
9. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: *CVPR*, pp. 886–893 (2005)
10. Deng, J., Dong, W., Socher, R., Li, L., Li, K., Li, F.: ImageNet: a large-scale hierarchical image database. In: *CVPR*, pp. 248–255 (2009)
11. Dollár, P., Wojek, C., Schiele, B., Perona, P.: Pedestrian detection: an evaluation of the state of the art. *TPAMI* **34**(4), 743–761 (2012)
12. Enzweiler, M., Gavrila, D.M.: Monocular pedestrian detection: survey and experiments. *TPAMI* **31**(12), 2179–2195 (2009)
13. Ess, A., Leibe, B., Gool, L.J.V.: Depth and appearance for mobile scene analysis. In: *ICCV*, pp. 1–8 (2007)
14. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2012 (VOC 2012) Results. <http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html>
15. Everingham, M., Eslami, S.M.A., Gool, L.J.V., Williams, C.K.I., Winn, J.M., Zisserman, A.: The pascal visual object classes challenge: a retrospective. *IJCV* **111**(1), 98–136 (2015)
16. Everingham, M., Gool, L.J.V., Williams, C.K.I., Winn, J.M., Zisserman, A.: The pascal visual object classes (VOC) challenge. *IJCV* **88**(2), 303–338 (2010)
17. Felzenszwalb, P.F., Girshick, R.B., McAllester, D.A., Ramanan, D.: Object detection with discriminatively trained part-based models. *TPAMI* **32**(9), 1627–1645 (2010)
18. Fu, C., Liu, W., Ranga, A., Tyagi, A., Berg, A.C.: DSSD : Deconvolutional single shot detector. *CoRR abs/1701.06659* (2017). <http://arxiv.org/abs/1701.06659>
19. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the KITTI vision benchmark suite. In: *CVPR*, pp. 3354–3361 (2012)
20. Girshick, R.B.: Fast R-CNN. In: *ICCV*, pp. 1440–1448 (2015)
21. Girshick, R.B., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: *CVPR*, pp. 580–587 (2014)
22. Haris, M., Shakhnarovich, G., Ukita, N.: Deep back-projection networks for super-resolution. *CoRR abs/1803.02735* (2018)

23. He, K., Gkioxari, G., Dollár, P., Girshick, R.B.: Mask R-CNN. In: ICCV, pp. 2980–2988 (2017)
24. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. TPAMI **37**(9), 1904–1916 (2015)
25. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR, pp. 770–778 (2016)
26. Hsieh, M., Lin, Y., Hsu, W.H.: Drone-based object counting by spatially regularized regional proposal network. In: ICCV (2017)
27. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. CoRR abs/1709.01507 (2017)
28. Li, Z., Peng, C., Yu, G., Zhang, X., Deng, Y., Sun, J.: Light-head R-CNN: in defense of two-stage object detector. CoRR abs/1711.07264 (2017)
29. Lin, T., Dollár, P., Girshick, R.B., He, K., Hariharan, B., Belongie, S.J.: Feature pyramid networks for object detection. In: CVPR, pp. 936–944 (2017)
30. Lin, T., Goyal, P., Girshick, R.B., He, K., Dollár, P.: Focal loss for dense object detection. In: ICCV, pp. 2999–3007 (2017)
31. Lin, T.-Y., et al.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). [https://doi.org/10.1007/978-3-319-10602-1\\_48](https://doi.org/10.1007/978-3-319-10602-1_48)
32. 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](https://doi.org/10.1007/978-3-319-46448-0_2)
33. Lyu, S., et al.: UA-DETRAC 2017: report of AVSS2017 & IWT4S challenge on advanced traffic monitoring. In: AVSS, pp. 1–7 (2017)
34. 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](https://doi.org/10.1007/978-3-319-46487-9_48)
35. Özuysal, M., Lepetit, V., Fua, P.: Pose estimation for category specific multiview object localization. In: CVPR, pp. 778–785 (2009)
36. Razakarivony, S., Jurie, F.: Vehicle detection in aerial imagery : a small target detection benchmark. J. Vis. Commun. Image Represent. **34**, 187–203 (2016)
37. Redmon, J., Divvala, S.K., Girshick, R.B., Farhadi, A.: You only look once: unified, real-time object detection. In: CVPR, pp. 779–788 (2016)
38. Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. In: CVPR, pp. 6517–6525 (2017)
39. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. CoRR abs/1804.02767 (2018)
40. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. TPAMI **39**(6), 1137–1149 (2017)
41. Russakovsky, O., et al.: Imagenet large scale visual recognition challenge. IJCV **115**(3), 211–252 (2015)
42. Shrivastava, A., Gupta, A., Girshick, R.B.: Training region-based object detectors with online hard example mining. In: CVPR, pp. 761–769 (2016)
43. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, Inception-ResNet and the impact of residual connections on learning. In: AAAI, pp. 4278–4284 (2017)
44. Uijlings, J.R.R., van de Sande, K.E.A., Gevers, T., Smeulders, A.W.M.: Selective search for object recognition. IJCV **104**(2), 154–171 (2013)
45. Viola, P.A., Jones, M.J.: Rapid object detection using a boosted cascade of simple features. In: CVPR, pp. 511–518 (2001)

46. Wang, F., et al.: Residual attention network for image classification. In: CVPR, pp. 6450–6458 (2017)
47. Wen, L., et al.: UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking. CoRR abs/1511.04136 (2015)
48. Xie, S., Girshick, R.B., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: CVPR, pp. 5987–5995 (2017)
49. Zhang, S., Wen, L., Bian, X., Lei, Z., Li, S.Z.: Occlusion-aware R-CNN: detecting pedestrians in a crowd. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11207, pp. 657–674. Springer, Cham (2018). [https://doi.org/10.1007/978-3-030-01219-9\\_39](https://doi.org/10.1007/978-3-030-01219-9_39)
50. Zhang, S., Wen, L., Bian, X., Lei, Z., Li, S.Z.: Single-shot refinement neural network for object detection. In: CVPR (2018)
51. Zhang, Z., Qiao, S., Xie, C., Shen, W., Wang, B., Yuille, A.L.: Single-shot object detection with enriched semantics. In: CVPR (2018)