YOLODrone: Improved YOLO Architecture for Object Detection in Drone Images

Oyku Sahin
Ozer Lab, Department of Computer Science
Bilkent University

Sedat Ozer
Ozer Lab, Department of Computer Science
Bilkent University

2021 44th International Conference on Telecommunications and Signal Processing (TSP) | 978-1-6654-2933-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/TSP52935.2021.9522653

***Abstract*—Recent advances in robotics and computer vision fields yield emerging new applications for camera equipped drones. One such application is aerial-based object detection. However, despite the recent advances in the relevant literature, object detection remains as a challenging task in computer vision. Existing object detection algorithms demonstrate even lower performance on drone (or aerial) images since the object detection problem is a more challenging problem in aerial images, when compared to the detection task in ground-taken images. There are many reasons for that including: (i) the lack of large drone datasets with large object variance, (ii) the larger variance in both scale and orientation in drone images, and (iii) the difference in shape and texture features between the ground and the aerial images. In this paper, we introduce an improved YOLO algorithm: YOLODrone for detecting objects in drone images. We evaluate our algorithm on VisDrone2019 dataset and report improved results when compared to YOLOv3 algorithm.**

***Keywords*—Object Detection, UAV image analysis, YOLO**

I. Introduction

Recent developments in aerial robotics introduced a variety of UAV applications for camera equipped drones as in [1], [2]. Consequently, the widespread use of the cameras on UAVs brought the attention to the vision based object detection algorithms. However, object detection in drone images remains as a challenging task which is shown to be even a harder challenge than it is in ground-taken images [2]. Therefore, there have been several public datasets released and workshops have been organized recently [3], [4], [5], [6], [7]. Detecting objects in aerial (drone) images is inherently a harder challenge than detecting objects in ground images and there are various reasons for that including: (i) objects can appear almost in any direction, while they mostly appear in a narrow range of direction in natural images (e.g., pedestrians or trees are mostly oriented perpendicular to the ground), (ii) objects appear in a large variety of scales in aerial images in the view of camera perspective, (iii) objects (depending on the height and the optical focus of the camera) typically appear significantly smaller on average in drone images and (iv) objects have different texture and shape information in aerial images than they appear in ground images. Figure 1 demonstrates two of such challenges in drone images. While there have been recent algorithms proposed to yield better detection accuracy (to see the performance of various detection algorithms, please refer to [6]), in many drone applications, the YOLO-based algorithms [8], [9] remain among the most commonly used detection algorithms and some reasons for that are: (a) various versions of YOLO implementations with their trained models are publicly available; (b) there are many online resources explaining the details of the algorithm are available; and (c) its frame per second (fps) rate is relatively higher than that of two-stage algorithms. After the original YOLO [8], there were two other versions also proposed by its original authors, namely YOLO version2 (YOLOv2) [10] and the most recent version: YOLO version3 (YOLOv3) [9].

Fig. 1. Two different images demonstrating sample challenges in aerial datasets: different viewing angle and objects appearing in much smaller size. Images are taken from the VisDrone2019 [6] dataset.

The most common techniques applying YOLO variants to aerial images mainly include modifications on various hyper-parameters such as fine-tuning, data augmentation and computing better hyper-parameters (see [6] for sample YOLO works). In this paper, we focus on that problem and study the affect of changing the network architecture of YOLO for aerial images. We call our new improved YOLO architecture as YOLODrone. We evaluate YOLODrone's performance on a large and public drone dataset and compare that to the performance of the most recent YOLO algorithm published by the original YOLO authors (YOLOv3 [9]). In this paper, we use VisDrone dataset because it is the largest and most recent aerial dataset including varying sizes of the objects relative to the image size. It contains many small sized objects along with their larger sized versions. We report significant improvement in mAP metric when compared to YOLOv3 algorithm. In particular, to detect the smaller objects and to deal with other issues that exist in aerial domain, we introduce increasing the total number of used prediction layers in the network. Our proposed YOLODrone architecture contains 95 convolutional layers and 5 prediction layers, while the YOLOv3 algorithm contains 75 convolutional layers with 3 prediction layers. We report an improvement of 0.185 in mAP_{50} value over YOLOv3 on VisDrone dataset in our experiments.

Our main contributions include: (1) introduction of a new YOLO architecture: YOLODrone for detecting objects in drone images, (2) studying the effect of using different data

Authorized licensed use limited to: Ozyegin Universitesi. Downloaded on December 09,2022 at 04:44:27 UTC from IEEE Xplore. Restrictions apply.

augmentation techniques, (3) reporting improved results in mAP_{50} value for the VisDrone2019 dataset, when compared to the YOLO algorithm (YOLOv3), (4) studying the performance of changing the total number of anchor boxes and the prediction layers in the YOLO architecture.

II. Related Work

Object detection is an essential task and has been widely studied in computer vision. Yet, it remains as a challenging task despite the recent advances using deep learning. While the earlier solutions used classical techniques with hand crafted features (such as [11]), the recent focus has been on utilizing deep learning based techniques due to the significant performance gains introduced by the deep learning as in [2], [12], [13]. The state-of-art object detectors are typically categorized under two names: *two stage methods* and *one stage methods*. Two stage methods such as R-CNN-based detectors (as in [14], [15], [16]) provide higher accuracy when compared to single stage ones. In two stage algorithms, the detection process contains two main steps: In the first step (stage), the candidate regions of interests are proposed and then, the following stage (the second step) is used to prone those detected regions to obtain the final detections and their bounding boxes. On the other hand, one stage methods such as CenterNet [17], SSD [18] and YOLO [8], typically provide higher frame per second (fps) rates. They are computationally more efficient since such detectors eliminate the need for using additional stage(s) (such as the region proposal network which increases the computational cost). Both single and double stage detectors are used for object detection in drone images. Examples for two-stage detectors used on aerial datasets can be found in [19], [20], [21], [2].

The relevant literature on object detection in drone images remains limited as it is a relatively a new domain and as the public datasets were not available until a couple of years ago. One of the largest drone datasets for object detection is VisDrone dataset and it has been released first in 2018. [7] In visDrone challenges, multiple solutions using YOLOv3 [9] variants can be observed. For example, YOLOv3 and Faster R-CNN has fused by [7].(A.8 and A.2). As stated in [7].(A.2), the reason behind that fusion is the dispersion of the dataset. Since the number of cars in the training data set is quite large, the detectors perform relatively well on the car class in the VisDrone dataset. However, some samples for many other classes are not that many in numbers as in the tri-cycle class. Thus, mostly, the detectors suffer from the imbalance problem and they could not perform well on that tri-cycle class. To solve that issue, the fusion of two detectors were proposed. Another work improved YOLOv3 by fine tuning in [7].(A.12) and applied initial data prepossessing to the training set in [7].(A.26). On the other hand, in visDrone 2019 [6] challenge only one group proposed a solution based on YOLOv3 in which the authors used a combination of YOLOv3-spp and faster-RCNN [6].(A.17). In [22], a car detection algorithm was proposed utilizing both YOLOv3 and R-CNN in aerial images. According to the reported results in that paper, YOLOv3 performed better than R-CNN to detect cars in aerial images. Another work using YOLOv3 on aerial images is proposed in [23] which improved YOLOv3 to make multi-scale prediction at four different scales instead of three. The main reason for that change is to use more information like contour and texture to detect smaller objects. Another important part when it comes to choosing algorithms for object detection on drones is having algorithms that can detect objects in real time. For example, in [24] YOLO-LITE is introduced to run in real-time to eliminate the need for a GPU by increasing the fps rate.

Our work differs from the existing work in the literature in many ways: (i) we introduce a YOLO architecture that is still considered as YOLO architecture but does perform better on aerial images, (ii) we show that including more prediction layers increases the prediction accuracy on VisDrone dataset, (iii) we show that re-assigning the anchor box sizes and their numbers also affect the accuracy.

III. Overview of YOLOv3

The original YOLO paper [8] was published in 2016 where the object detection problem was approached mainly as a regression problem. A deep network was used to predict the class score and bounding boxes at the same time on a given input image. The original YOLO contains 24 convolutional layers and two fully connected (FC) layers. YOLO version 2 (YOLOv2) [10] was introduced as an improved version of the original YOLO algorithm. In that version, a fully convolutional architecture was proposed. Batch normalization and anchor boxes were also used. YOLOv2 architecture was trained on a larger number of classes with higher resolution images. YOLOv3 [9] using DarkNet53 as feature extractor was first introduced in 2018 as an improved version over YOLOv2. Three main improvements introduced: First, YOLOv3 detects objects at three different scales with the help of predefined anchor boxes at different scales in three different layers. Each detection layer is responsible with a different number of grid cells and each one proposes 3 candidate bounding boxes, as there are 3 anchor boxes used at each detection layer. Second improvement in YOLOv3 is a novel loss function. Instead of penalizing the objectness scores, class and bounding boxes surrounds non-object part of the image, with respect to squared errors in YOLOv3 they are penalized by using cross-entropy. The third improvement is adding the ability of multi-label classification to YOLO algorithm. The YOLOv3 includes 53 convolutional layers and the detection algorithm yields higher performance on the MS COCO data set. Figure 2 (a) summarizes YOLOv3 architecture where each convolutional block is color coded (see Table I for further details).

IV. YOLODrone Architecture

The existing state-of-the-art object detectors are designed by considering and observing the existing problems in the datasets that are designed for object detection such as ImageNet, Pascal and COCO datasets. However, each of those three datasets predominantly contains ground-taken images. Therefore, object detectors trained on such datasets containing ground-taken

Authorized licensed use limited to: Ozyegin Universitesi. Downloaded on December 09,2022 at 04:44:27 UTC from IEEE Xplore. Restrictions apply.

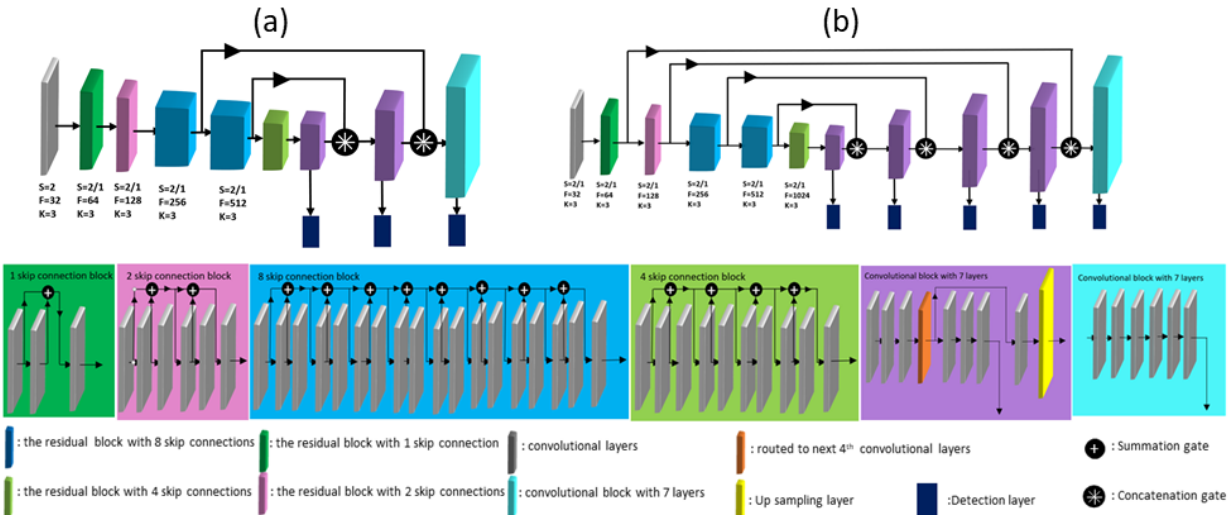

Fig. 2. (a) shows the original YOLOv3 and (b) shows our YOLODrone architecture. Different colored blocks in the architecture represent different blocks. The meaning of each color is given below the architectures. Our architecture has total of five output (prediction) layers.

images typically perform poorly on aerial images. That is also valid for the YOLO algorithm (as our experimental results also support that. See our experiments section).

A. Number of Detection Layers

One of the main issues where the YOLO algorithm suffers is dealing with the scale of the objects in aerial images. A potential reason for that is making predictions only at three different layers. As the number of prediction layers increases, the algorithm can adjust its parameters to focus on the range of scale better with the back-propagation algorithm. With that idea, we developed YOLODrone architecture which contains more layers and more detection layers when compared to original YOLOv3 architecture (compare YOLOv3's 3 detection layers to our YOLODrone's 5 detection layers). Our YOLODrone's architecture is given in Figure 2 and further details are provided in Table II. Similar to the original YOLOv3, YOLODrone resembles an encoder-decoder architecture where in the first part of its network, the height and width of the feature maps shrink, and then in the second half, they increase again. The shrinking process (reducing the height and width of the feature maps at each consecutive block) is done through the stride operation. In the first convolutional layer of each block, stride is set to 2, and after that first convolutional layer, stride was set to 1 for each aditional convolutional layer in the same block. Similarly, to increase the height and width of feature maps, an upsampling layer is used at the beginning of each block in the second half of the network. In our architecture, we use 6 different block types where each block is color-coded (as shown in Figure 2). Each color-coded block contains different number of convolutional layers and skip connections.

Anchor boxes are used in both YOLOv3 and YOLODrone and they act as hyperparameters affecting the performance of the final detection. In our proposed approach, we utilize three different anchor boxes for each grid cell in each detection layer. For both algorithms, we find the optimal anchor box values by applying K-Means algorithm on the given (drone) dataset. Table III compares our YOLODrone algorithm to YOLOv3 in various details. We utilize total of 6, 9, 12 anchor boxes in our experiments.

Block Connections -	Skip Layers 608x608	Convolutional -	Size (Input) -	Channel (Output) -	Channel Filters -	Number of Function heightInput	Activation Name -
Convolutional Layer	-	1	3x3	1	32	32	Leaky Relu
Residual Block	1	3	3x3	32	64	32 (x1),64 (x2)	Leaky Relu
Residual Block	2	5	3x3	64	128	64 (x2) ,128 (x3)	Leaky Relu
Residual Block	8	17	3x3	128	256	128 (x8) ,256 (x9)	Leaky Relu
Residual Block	8	17	3x3	256	512	256 (x8),512 (x9)	Leaky Relu
Residual Block	4	9	3x3	512	1024	512(x4),1024(x5)	Leaky Relu
Detection Layer	-	7	3x3	1024	512	1024(x3), 512(x3), 51(x1)	Leaky Relu
Detection Layer	-	7	3x3	512	256	512(x3), 256(x3), 51(x1)	Leaky Relu
Detection Layer	-	7	3x3	256	128	256(x3),128 (x3), 51(x1)	Leaky Relu
Output	-	-	608x608	-	-	-	-

TABLE I. SHOWS THE DETAILS OF YOLOV3 ARCHITECTURE ACCORDING TO DIFFERENT BLOCK TYPES AND THEIR ELEMENTS.

V. EXPERIMENTAL RESULTS

All the models are trained from scratch with random initialization over 300 epochs with $batchsize = 8$ on GeForce GTX 1080 Ti. The input image size is fixed at 416x416. As the optimizer, the Adam optimizer was used with 0.001 learning rate. We used mean average precision (mAP) [25] as our performance metric in our tables: IV,V,VI and in VII.

Block Name	Skip Connections	Convolutional Layers	Size	Channel (Input)	Channel (Output)	Number of Filters	Activation Function
Input	-	-	416x416	-	-	-	-
Convolutional Layer	-	1	3x3	1	8	8 (x1)	Leaky Relu
Residual Block	1	3	3x3	32	64	32 (x1),64 (x2)	Leaky Relu
Residual Block	2	5	3x3	64	128	64 (x2) ,128 (x3)	Leaky Relu
Residual Block	8	17	3x3	128	256	128 (x8) ,256 (x9)	Leaky Relu
Residual Block	8	17	3x3	256	512	256 (x8),512 (x9)	Leaky Relu
Residual Block	4	9	3x3	512	1024	512(x4),1024(x5)	Leaky Relu
Detection Layer	-	7	3x3	-	-	-	Leaky Relu
Detection Layer	-	7	3x3	-	-	-	Leaky Relu
Detection Layer	-	7	3x3	-	-	-	Leaky Relu
Detection Layer	-	7	3x3	-	-	-	Leaky Relu
Detection Layer	-	7	3x3	-	-	-	Leaky Relu
Output	-	-	416x416	-	-	-	-

TABLE II. SHOWS THE DETAILS OF YOLODRONE ARCHITECTURE.

	Original YOLOv3	YOLODrone
Number of Convolutions	75	95
Number of Skip Connections	23	23
Number of Detection Layers	3	5
Number of down sampling	6	8
Number of up sampling	2	4

TABLE III. COMPARISON OF YOLOV3 TO YOLODRONE.

Authorized licensed use limited to: Ozyegin Universitesi. Downloaded on December 09,2022 at 04:44:27 UTC from IEEE Xplore. Restrictions apply.

Fig. 3. This figure provides sample results for both YOLOv3 and YOLODrone algorithms on selected two images from the VisDrone dataset. The first column shows the ground truth, the second column shows the results of YOLOv3 and the third column shows the results of YOLODrone. The objects labelled with red circle represents the missed or mis-classified objects, the circles with green shows successfully detected objects.

A. *The Effect of Data Augmentation*

Data augmentation is one of the most common techniques to improve the performance where the goal is improving the variance in the training data. In addition to the common data augmentation techniques such as flip, crop and rotate, we also utilize another augmentation technique that blackens certain and vague regions in the datasets. The VisDrone dataset provides a class called ignored regions and since we do not want our model to learn from those parts, we blackened those ignored parts in the images as a part of data augmentation process. In addition to that operation, The following table demonstrates the effect of the augmentation technique we used. As it can be seen in Table IV, although in some classes the precision decreases, the overall mAP value increases with this augmentation technique. The total number of images in the training dataset is 6471 and after the data augmentation process that number is increased to 19413 images. Test data includes 1610 images (no augmentation is used on test data).

B. *The Effect of Varying the Number of Used Detection Layers*

As mentioned, one of the main issues where the YOLOv3 model suffers is dealing with the scale of the objects in aerial images. The scales that the objects appear in drone datasets are smaller than they appear in ground taken images. Therefore, in YOLODrone architecture, we focused on that scaling issue of aerial datasets. In our preliminary results, we observed that YOLOv3 deals with the scales better when compared to the previous YOLO versions. In a YOLO architecture, the total number of used grid cells is important in detecting objects because after the image divided into grid cells, each cell proposes a candidate anchor box. Therefore, by increasing the number of detection layers we also increase the number of candidate anchor boxes scaled in 5 different height width scale combinations. While experimenting on this idea, we wanted to compare our results with the actual YOLOv3 model. Therefore, we trained the original YOLOv3 on visDrone dataset with random initialization. Then, we modified the algorithm to have first 4 detection layers and then 5 detection layers (YOLODrone). We used K-means algorithm to find data-specific height-width anchor box pairs. Table V compares those results on each class of VisDrone2019 dataset for the network with 3, 4 and 5 detection layers, respectively.

Table V shows our experiments on using different numbers of detection layers. According to the table, mAP value increased the most on average with 5 detection layers.

Augmentation	Pedestrian	Person	Bicycle	Car	Van	Truck	Tricycle	Awn	Bus	Motor	mAP
No	0.151	0.199	0.104	0.215	0.209	0.131	0.0817	0.110	0.406	0.116	0.181
Yes	**0.198**	**0.216**	**0.212**	**0.359**	**0.263**	**0.17**	**0.102**	**0.116**	**0.468**	**0.199**	**0.299**

TABLE IV. SHOWS THE RESULTS OF YOLODRONE ON VISDRONE2019 DATASET WITH AND WITHOUT THE DATA AUGMENTATION.

C. *Number of Anchor Boxes*

The anchor boxes are used to better fit a candidate bounding box to the ground truth in YOLOv3 and similarly in YOLO-

Detection Layers	Pedestrian	Person	Bicycle	Car	Van	Truck	Tricycle	Awn	Bus	Motor	mAP
3	0.198	**0.216**	0.102	0.359	**0.263**	**0.17**	**0.102**	0.116	**0.468**	0.199	0.299
4	0.237	0.202	0.071	**0.433**	**0.229**	0.126	0.093	**0.117**	0.447	0.229	0.376
5	**0.251**	0.199	**0.114**	0.415	0.229	0.131	0.082	0.11	0.406	0.216	**0.391**

TABLE V. COMPARES THE RESULTS FOR USING 3 DETECTION LAYERS, 4 DETECTION LAYERS AND 5 DETECTION LAYERS ON VISDRONE2019 DATASET.

Anchor Boxes	Pedestrian	Person	Bicycle	Car	Van	Truck	Tricycle	Awn	Bus	Motor	mAP
6	0.216	0.195	0.046	**0.444**	0.185	0.095	0.040	0.079	0.321	0.196	0.216
9	0.198	0.216	**0.102**	0.359	**0.263**	**0.170**	**0.102**	**0.116**	0.468	**0.199**	0.299
12	**0.219**	**0.468**	0.0771	0.437	0.219	0.157	0.081	0.0886	**0.479**	0.186	**0.363**

TABLE VI. SHOWS RESULTS FOR DIFFERENT NUMBERS OF ANCHOR BOXES FOR YOLOV3 ALGORITHM.

Architecture	Pedestrian	Person	Bicycle	Car	Van	Truck	Tricycle	Awn	Bus	Motor	AP	mAP_{50}
YOLOv3	0.198	**0.216**	0.102	0.359	0.263	**0.170**	0.102	0.116	0.406	**0.199**	0.283	0.299
YOLODrone (ours)	**0.224**	0.163	**0.109**	**0.397**	**0.356**	0.167	**0.103**	**0.160**	**0.506**	0.191	**0.290**	**0.484**

TABLE VII. COMPARISON OF YOLOV3 TO YOLODRONE ON THE VISDRONE2019 DATASET (MAP VALUES ARE SHOWN FOR EACH CLASS).

Authorized licensed use limited to: Ozyegin Universitesi. Downloaded on December 09,2022 at 04:44:27 UTC from IEEE Xplore. Restrictions apply.

Drone, we use k-means clustering to find the initial the anchor box sizes. By changing the total number of used anchor boxes, we change the total number of candidate bounding boxes in the architecture. The anchor boxes are scaled at each layer individually. Since the detection layers are responsible for proposing the candidate bounding boxes, when the grid size is changed, the anchor boxes should also be adjusted accordingly. Table VI shows the experimental results demonstrating the effect of utilizing different numbers of anchor boxes in YOLOv3.

Table VII shows the best results (in mAP) of both algorithms (YOLOv3 and YOLODrone) for each class individually. In the table, we used 9 anchor boxes in the original YOLOv3 (as it was the originally proposed number) and we used 12 anchor boxes in our YOLODrone. On top of YOLOv3, YOLODrone is trained over 300 epochs. In both mAP and mAP_{50} values, YOLODrone yielded higher performance than YOLOv3 and out of the listed 10 classes in the table, YOLODrone yielded the best results for 7 classes. Figure 3 shows qualitative results obtained by both YOLOv3 and YOLODrone algorithms.

VI. Conclusion

Detecting objects in aerial images remains as a harder challenge when compared to the object detection problem in ground-taken images. In this paper, to deal with that problem, we studied various aspects of the common YOLO architecture (YOLOv3) and introduced a new YOLO architecture, which we call YOLODrone, that is crafted to detect objects particularly in drone images. Our algorithm YOLODrone have increased number of detection layers (including 5 detection layers) when compared to the original YOLOv3 algorithm. While different versions of YOLO (such as YOLOv4 [26] and YOLOv5 [27])) have also been proposed recently, at the time when this work was completed, YOLOv3 was the most recent version and consequently, YOLOv4 and YOLOv5 were not included in this paper. Using more detection layers and a deeper network helps our algorithm to assign different layers to different scales for object detection. Also, since we are dividing the image to predict the bounding boxes to more grids with 5 detection layers, we also let the model learn smaller objects more accurately as that can also be seen in Figure 3. Our experimental results show that, our proposed YOLODrone performed better than YOLOv3 algorithm on VisDrone dataset on average in terms of the mAP_{50} value.

Acknowledgment

This paper has been produced benefiting from the 2232 International Fellowship for Outstanding Researchers Program of TÜBİTAK (Project No:118C356). However, the entire responsibility of the paper belongs to the owner of the paper. The financial support received from TÜBİTAK does not mean that the content of the publication is approved in a scientific sense by TÜBİTAK.

References

[1] D. Gözen and S. Ozer, "Visual Object Tracking in Drone Images with Deep Reinforcement Learning," in *International Conference on Pattern Recognition (ICPR2020)*, 2020.

[2] B. M. Albaba and S. Ozer, "Synet: An ensemble network for object detection in uav images," in *IEEE, 25th International Conference on Pattern Recognition (ICPR2020), ITALY*, 10 - 15 January 2021.

[3] G.-S. Xia, X. Bai, J. Ding, Z. Zhu, S. Belongie, J. Luo, M. Datcu, M. Pelillo, and L. Zhang, "Dota: A large-scale dataset for object detection in aerial images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[4] J. Leskovec and A. Krevl, "Snap datasets: Stanford large network dataset collection," 2014.

[5] D. Lam, R. Kuzma, K. McGee, S. Dooley, M. Laielli, M. Klaric, Y. Bulatov, and B. McCord, "xview: Objects in context in overhead imagery," *arXiv preprint arXiv:1802.07856*, 2018.

[6] D. Du, P. Zhu, L. Wen, X. Bian, H. Ling, Q. Hu, T. Peng, J. Zheng, X. Wang, Y. Zhang, L. Bo, H. Shi, R. Zhu, A. Kumar, A. Li, A. Zinollayev, A. Askergaliyev, A. Schumann, B. Mao, and Z. Liu, "The vision meets drone object detection in image challenge results," 2019.

[7] P. Zhu, L. Wen, D. Du, X. Bian, H. Ling, Q. Hu, Q. Nie, H. Cheng, C. Liu, X. Liu *et al.*, "Visdrone-det2018: The vision meets drone object detection in image challenge results," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 0–0.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[9] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[10] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Computer vision and pattern recognition*, 2017, pp. 7263–7271.

[11] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *2008 IEEE conference on computer vision and pattern recognition*. IEEE, 2008, pp. 1–8.

[12] S. Xu, A. Savvaris, S. He, H. Shin, and A. Tsourdos, "Real-time implementation of yolo+jpda for small scale uav multiple object tracking," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2018, pp. 1336–1341.

[13] R. Valiente, M. Zaman, S. Ozer, and Y. P. Fallah, "Controlling steering angle for cooperative self-driving vehicles utilizing CNN and LSTM-based deep networks," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 2423–2428.

[14] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[15] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[16] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[17] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint triplets for object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 6569–6578.

[18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[19] J. C. van Gemert, C. R. Verschoor, P. Mettes, K. Epema, L. P. Koh, and S. Wich, "Nature conservation drones for automatic localization and counting of animals," in *European Conference on Computer Vision*. Springer, 2014, pp. 255–270.

[20] P. C. Gray, A. B. Fleishman, D. J. Klein, M. W. McKown, V. S. Bézy, K. J. Lohmann, and D. W. Johnston, "A convolutional neural network for detecting sea turtles in drone imagery," *Methods in Ecology and Evolution*, vol. 10, no. 3, pp. 345–355, 2019.

[21] H. Zhu, X. Chen, W. Dai, K. Fu, Q. Ye, and J. Jiao, "Orientation robust object detection in aerial images using deep convolutional neural network," in *2015 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2015, pp. 3735–3739.

[22] B. Benjdira, T. Khursheed, A. Koubaa, A. Ammar, and K. Ouni, "Car detection using unmanned aerial vehicles: Comparison between faster r-cnn and yolov3," in *2019 1st International Conference on Unmanned Vehicle Systems-Oman (UVS)*. IEEE, 2019, pp. 1–6.

[23] Y. Hu, X. Wu, G. Zheng, and X. Liu, "Object detection of uav for anti-uav based on improved yolo v3," in *2019 Chinese Control Conference (CCC)*. IEEE, 2019, pp. 8386–8390.

[24] R. Huang, J. Pedoeem, and C. Chen, "Yolo-lite: A real-time object detection algorithm optimized for non-gpu computers," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 2503–2510.

[25] R. Padilla, W. L. Passos, T. L. Dias, S. L. Netto, and E. A. da Silva, "A comparative analysis of object detection metrics with a companion open-source toolkit," *Electronics*, vol. 10, no. 3, p. 279, 2021.

[26] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," 2020.

[27] G. Jocher, "Yolov5," in *https://github.com/ultralytics/yolov5*, 2020.

Authorized licensed use limited to: Ozyegin Universitesi. Downloaded on December 09,2022 at 04:44:27 UTC from IEEE Xplore. Restrictions apply.