Computer and Information Science; Vol. 14, No. 1; 2021
ISSN 1913-8989 E-ISSN 1913-8997
Published by Canadian Center of Science and Education

Efficient Object Detection Model for Real-time UAV Application

Subrahmanyam Vaddi[1], Dongyoun Kim[1], Chandan Kumar[1], Shafqat Shad[1] & Ali Jannesari[1]

[1] Department of Computer Science, Iowa State University, Ames, IA, USA

Correspondence: Ali Jannesari, Department of Computer Science, Iowa State University, 110 Atanasoff Hall, 2434 Osborn Dr, Ames, 50011 IA, USA.

Received: December 10, 2020 Accepted: January 21, 2021 Online Published: January 22, 2021

doi:10.5539/cis.v14n1p45 URL: https://doi.org/10.5539/cis.v14n1p45

Abstract

Unmanned Aerial Vehicles (UAVs) equipped with vision capabilities have become popular in recent years. Many applications have especially been employed object detection techniques extracted from the information captured by an onboard camera. However, object detection on UAVs requires high performance, which has a negative effect on the result. In this article, we propose a deep feature pyramid architecture with a modified focal loss function, which enables it to reduce the class imbalance. Moreover, the proposed method employed an end to end object detection model running on the UAV platform for real-time application. To evaluate the proposed architecture, we combined our model with Resnet and MobileNet as a backend network, and we compared it with RetinaNet and HAL-RetinaNet. Our model produced a performance of 30.6 mAP with an inference time of 14 fps. This result shows that our proposed model outperformed RetinaNet by 6.2 mAP.

Keywords: object detection, robotics, unmanned aerial vehicle, embedded HPC, visual computing

1. Introduction

In recent years, autonomous Unmanned Aerial vehicles (UAVs), especially drones equipped with cameras, have become significant with the wide variety of their applications such as surveillance (Bhaskaranand & Gibson, 2011), aerial mapping (Parrott, Panter, Morrissey, & Bezombes, 2019), search and rescue (Doherty & Rudol, 2007), infrastructural inspection (Sa, Hrabar, & Corke, 2015). An automatic understanding of visual data collected from these drones has been the area of increasing interest. Especially, many vision-based UAV applications have been researched to detect specific objects for achieving their missions – single object detects such as a vehicle (Gleason & Nefian, 2011), pedestrians (Lim & Shinha, 2015), autonomous navigation and landing (Forster, Faessler, Fontana, Werlberger, & Scaramuzza, 2015), Path planning (Asli, Roghair, & Jannesari, 2019), and some studies which detect multiple objects such as (Tang, Deng, Zhou, Lei, & Zou, 2017) and (Venieris, 2018). Therefore, Visual object detection is one of the important aspects of applications of drones and is critical to have in fully autonomous systems. The key challenges in deploying the vision and intelligence capabilities on a UAV platform are 1) to consume minimal power in order to minimize its effect on battery consumption and flight time of the drone 2) require less memory footprint and computational power as typical UAV has resource limitation 3) process the input data from its camera with low latency and perform faster in order to make critical decisions. Conventionally onboard computing infrastructure consists mainly of general-purpose machines such as multicore CPUs and low power microcontrollers. Such a case considers the hardware perspective, which might be a foreseeable solution. However, replacing the machine power is cost-inefficient since the cost of a single machine increases exponentially by increasing its computing power. Hence, we are concerned with developing algorithms based on deep learning for object detection running on UAV equipped with embedded hardware platforms suitable for real-time applications.

Traditionally hand-tuned features were used for object recognition and detection, such as Scale-Invariant Feature Transform (SIFT) (Lowe, 1999), Speeded-Up Robust Features (SURF) (Bay & Van, 2006), and Histogram of Oriented Gradient (HOG) (Mizuno et al., 2012). These methods used hand-crafted local features with image descriptions. With the breakthrough of deep learning using Convolutional Neural Networks, there was a striking performance increase in dealing with these computer vision tasks (Mammadli, Wolf, & Jannesari, 2019). The key idea is to learn the object model and features from raw pixel data of images. It requires large datasets to train the models. Even though this problem has also been overcoming by many large scales labeled datasets like ImageNet, these techniques basically require a huge amount of computational power and built-in memory. Recently, object detection methods are categorized as single-step and two-step, which can be called a

region-based model. The mask R-CNN (He, Gkioxari, Dollar, & Girshick, 2017) is one of the representative models in the region-based model. Even though the region-based models achieved higher accuracy than single-step methods, these methods require higher computational. This is unsuitable to employ them in UAV application. YOLO (Redmon, Divvala, Girshick, & Farhadi, 2016) and RetinaNet (Lin, Goyal, He, & Dollar, 2017) are representative methods in single-step methods. These methods are suitable because of their high performance and applicability to an embedded system. However, these methods are usually exposed to class imbalance problem because a single step detector densely samples regions from all over the image. Class imbalance occurs when the types of training examples are not equal in number.

In this article, we propose a feasible Deep Feature Pyramid Network (DFPN) architecture and a modified loss function to avoid class imbalance and achieved real-time object detection performance in a real drone environment. DFPN architecture is composed of the Feature Pyramid Network (FPN) by (Lin et al., 2017) model and modified local function. This enables the object detection model, which is computationally less expensive, memory efficient, and fast without compromising the detection performance, running on a drone. To evaluate the proposed model, we showed the evaluations of each model based on three metrics, namely Mean Average Precision (mAP) and inference time as frames per second (fps). DFPN outperforms RetinaNet (Lin et al., 2017) by 6.2 mAP when using MobileNet as a backbone. Furthermore, DFPN improves the overall performance by 0.6 mAP when the backbone is changed from ResNet to MobileNet, which is a lightweight and efficient network.

2. Related Work

Vision application of UAV could achieve navigation, path planning, and autonomous control for itself. In early studies of UAV, Saripalli et al. (2002) explored the vision-based strategies for autonomous control during the UAV landing period. Scaramuzza et al. (2014) incorporated vision techniques into their UAV design to enhance navigation performance. UAV equipped with computer vision technique also offer video or image surveillance services in different application scenarios. Earlier, Ryan et al. (2004), in their work, gave a general overview of the implementation of object detection involved computer vision techniques for aerial surveillance. Later many studies focused on data-driven applications of UAV. For example, vision-based UAV applications have been used for smart urban surveillance (Chen et al., 2017). Many recent studies make use of the flight control system utilizing deep learning frameworks for vision assisted tasks to perform scene analysis of an aerial image. pixhawk (Meier et al., 2012) is one of the popular flight control system designed with computer vision algorithms to execute the obstacle detection and autopilot mode during UAV operation, which we have used onboard for our experiment.

2.1 Object Detection Models

The first deep learning object detector model was called Overleaf Network (Ryan, Zennaro, Howell, Sengupta, & Hedrick, 2004) which used Convolutional Neural Networks (CNNs) along with the sliding window approach classifying each part of image as object or non-object and combined the results for predictions. This method of using CNNs to solve detection led to new networks being introduced which pushed the state of the art even further. In recent years, numerous object detectors have been proposed by the deep learning community, including Faster R-CNN (Ren, He, Girshick, & Sun, 2016), R-FCN (Dai, Li, He, & Sun, 2016), YOLO (Redmon et al., 2016) and SSD (Tripathi, Kang, Dane, & Nguyen, 2017). The main emphasis of these designs is placed on improving (1) the detection accuracy and (2) the computational complexity of their methods in order to achieve real-time performance for mobile and embedded platforms (Tripathi, Kang, Dane, & Nguyen, 2017). These detection models can be divided into two categories based on their high-level architecture - the single-step approach and the two-step (region-based) approach. The single step approach achieved faster results and shown higher memory eciency, whereas the two-step approach achieved better accuracy than the former. Region-based detectors divide object detection in two steps. The first step generates a set of regions in the image that have a high probability of being an object. The second step then performs the final detection and classification of objects by taking these regions as input. A prominent example of such a detector is Faster R-CNN (Ren et al., 2017), in which the first stage, called the Region Proposal Network (RPN), employs the feature extractor of a CNN (e.g. VGG-16, ResNet, etc.) to process images and utilizes the output feature maps of a selected intermediate layer in order to predict bounding boxes of class-agnostic objects on an image. In the second stage, the box proposals are used to crop features of the same intermediate feature maps and pass them through a classifier in order to both predict a class and refine a class-specific box for each proposal. With typically hundreds of proposals per image passed separately through the classifier, Faster R-CNN remains computationally heavy and poses a challenge in achieving high performance in embedded platforms and mobile devices. Other examples of region-based detectors are FPN (Lin et al., 2017) and R-FCN (Dai et al., 2016). This class of detectors aims to avoid the performance bottlenecks of the 2-step region-based systems. The YOLO

(Redmon et al., 2016) framework casts object detection to a regression problem and in contrast to the RPN + classifier design of Faster R-CNN, employs a single CNN for the whole task. YOLO divides the input image into a grid of cells and for each cell outputs predictions for the coordinates of a number of bounding boxes, the confidence level for each box, and a probability for each class. Compared to Faster R-CNN, YOLO is designed for real-time execution and by design provides a trade-o↵ that favors high performance over detection accuracy. In addition, the open-source released version of YOLO has been developed using the C and CUDA based Darknet (Redmon, 2013) framework, which enables the use of both CPUs and GPUs and is portable across a variety of platforms. SSD (Liu et al., 2016) is a single-shot detector aims to combine the performance of YOLO with the accuracy of region-based detectors. SSD extends the CNN architecture of YOLO by adding more convolutional layers and allowing the grid of cells for predicting bounding boxes to have a wider range of aspect ratios in order to increase the detection accuracy for objects of multiple scales. Despite the high detection accuracy and performance, the open source released version of SSD has been developed using the Caffe framework. RetinaNet is the latest single step object detector which boasts the state-of-the-art results at this point in time by introducing a novel loss function (Lin et al., 2017). This model represents the first instance where one step detectors have surpassed two-step detectors inaccuracy while retaining superior speed.

In this paper, DFPN is based on single-shot detectors due to their high performance and applicability to mobile and embedded systems. The experiment shows that compared RetinaNet with our baseline detector. The proposed model is based on DFPN architecture embedded within an object detector model with reducing class imbalance problem.

3. Method

The proposed detection model design features an efficient in-network Deep Feature Pyramid Network (DFPN) architecture and use of anchor boxes to predict outputs. It draws a variety of ideas from the Feature Pyramid network (Lin et al., 2017) and Focal loss for Dense object detection (Lin et al., 2017). The reason why one step detectors have lagged behind two-step detectors in terms of accuracy was an implicit class imbalance problem that was encountered while training. Class imbalance occurs when the types of training examples are not equal in number. Single-step detectors suffer from an extreme foreground/background imbalance, with the data heavily biased towards background examples.

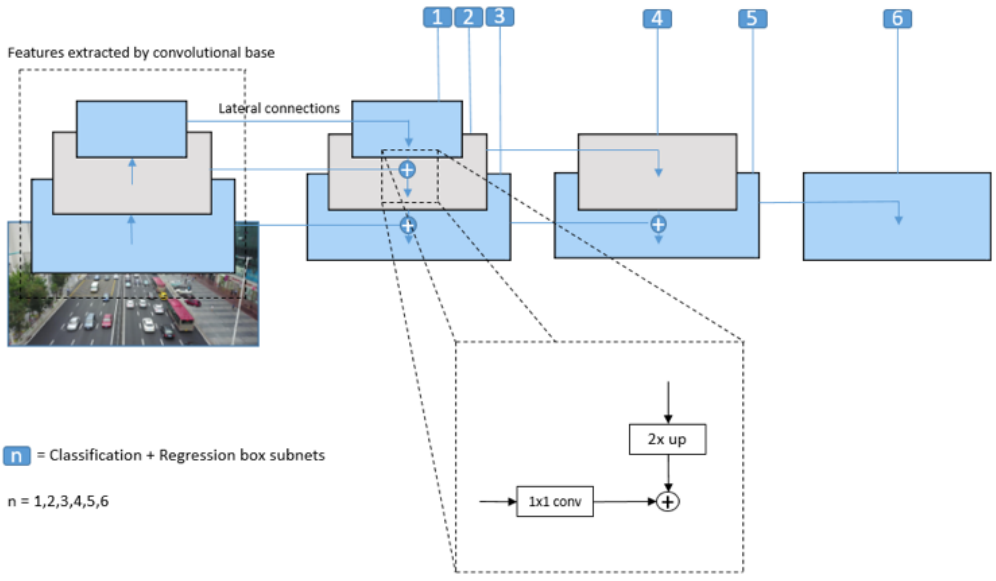

Figure 1. Proposed Deep Feature Pyramid Network (DFPN) depicting the lateral connections and how the deep pyramids are constructed

The architecture of the proposed model is shown in Figure1. Similar to FPN, the goal is to leverage a ConvNet's pyramidal feature hierarchy, which has semantics from low to high levels, and build deep feature pyramids with high-level semantics throughout. The construction of our pyramid involves a bottom-up pathway, a top-down pathway, and lateral connections. The bottom-up pathway is the feed-forward computation of the backbone

ConvNet, which computes a feature hierarchy consisting of feature maps at several scales with a scaling step of 2. The top-down pathway hallucinates higher resolution features by up-sampling spatially coarser, but semantically stronger, feature maps from higher pyramid levels. These features are then enhanced with features from the bottom-up pathway via lateral connections. Each lateral connection merges feature maps of the same spatial size from the bottom-up pathway and the top-down pathway. The bottom-up feature map is of lower-level semantics, but its activations are more accurately localized as it was sub-sampled fewer times. DFPN is built by further creating the FPN structures until the output feature maps converges to 1, creating a deep and rich feature pyramids of image. We have developed high-level semantic feature maps of all possible scales, thus making our solution to be scale-invariant. Also, this approach works pretty well with the aerial images, as most of the objects are usually very small in scale. Additionally, our network captures the abrupt changes in scales of objects, which typically occurs in videos captured by drones. To be more specific, we use a simple object detector model by combining the best practices gained from previous research studies. As our DFPN is integrated with the models, the anchor sizes do not need to account for different scales as that is handled by the multiple feature maps. Each level on the feature pyramid uses 9 anchor shapes at each location. The set of three aspect ratios such as 1:1, 1:2, 2:1 have been augmented by the factors of 1, $2^1/3$, $2^2/3$ for a more diverse selection of bounding box shapes. Similar to the RPN, each anchor predicts a class probability out of a set of K object classes (including the background class) and 4 bounding box offsets. Targets for the network are calculated for each anchor as follows. The anchor is assigned the ground truth class if the IOU (Intersection over Union) of the ground truth box and the anchor ≥ 0.5. A background class is assigned if the IOU ≤ 0.4 and if the $0.4 \leq$ IOU ≤ 0.5, the anchor is ignored during training. Box regression targets are computed by calculating the offsets between each anchor and its assigned ground truth box using the same method used by the Faster RCNN. No targets are calculated for the anchors belonging to the background class, as the model is not trained to predict offsets for a background region. The detector uses a single unified network composed of a backbone network and two task-specific subnetworks. The first subnetwork predicts the class of the region, and the second subnetwork predicts the coordinate offsets. The architecture is similar to an RPN augmented by an FPN except that we build a deep FPN model as explained below. The classification and the regression subnets are quite similar in structure. Each pyramid level is attached with these subnetworks, weights of the heads are shared across all levels. The architecture of the classification subnet consists of a small FCN consisting of 4 convolutional layers of filter size 3 by 3. Each convolutional layer has a relu activation function attached to it and maintains the same channel size as the input feature map. Finally, sigmoid activations are attached to output a feature map of depth $A*K$ where the value for A = 9 and it represents the number of aspect ratios per anchor, K represents the number of object classes. The box regression subnet is identical to the classification subnet except for the last layer. The last layer has the depth of $4 * A$. The 4 indicates the width, height, and x and y coordinate offsets of bounding boxes. The loss of a model is used to train the entire network, and it controls the amount by which the gradients are tweaked during back-propagation. A high loss for an object makes the network more sensitive for that object and vice versa. The loss function we used in similar to the focal loss of RetinaNet (Lin et al., 2017) given by equation (1),

$$F = \alpha^2 \times c(1-p)^{\gamma} \quad (1)$$

where c is cross-entropy loss, α is the hyperparameter which down weights the loss generated by background class. The value for α could be the inverse class frequency or can be treated as a hyper-parameter to be set during cross-validation. We considered square of α as there can be an exponential number of background anchors compared to target anchors because of the aerial view of images. γ is the exponential scaling factor which down weights the loss contributed by easy examples, thereby making the network sensitive only for difficult examples. We considered values of 0.25 for α and 2 for γ in our experiments.

The training is performed using Stochastic Gradient Descent (SGD), using an initial learning rate of 0.0001. Horizontal image flipping is the only data augmentation technique used. Inference is performed by running the image through the network. Only the top 1k predictions are taken at each feature level after thresholding the detector confidence at 0.05. Non-Maximum Suppression (NMS) is performed using a threshold of 0.5, and the boxes are overlaid on the image to form the final output. This technique is seen to improve training stability for both cross-entropy and focal loss in the case of heavy class imbalance. Object detectors use the convolutional base to create feature maps of images that is embedded with salient information about the image. The accuracy and performance of the detector is highly dependent on how well the convolutional base can capture the meaningful information about the image. The base takes the image through a series of convolutions that make the image smaller and deeper. This process allows the network to make sense of the various shapes in the image. Convolutional bases can be selected on the basis of three factors such as speed, accuracy, and memory footprint. As per our requirement, we chose ResNet (He, Zhang, Ren, & Sun, 2016) and MobileNet (Howard et al., 2017)

for our studies and compared the results obtained using both. ResNet is one of the popular convolutional bases with the concept of skip connections in convolutional networks. Skip connections add or concatenate features of the previous layer to the current layer. This leads to the network propagating gradients much more effectively during backpropagation. On the other hand, MobileNets are series of convolutional networks best suitable for applications where speed and memory efficiency are of high importance. They are proved to be well suited for embedded system applications. MobileNets introduce depthwise separable convolutions which form their backbone and have led to a speedup in computing the feature map without sacrificing the overall quality.

4. Experiment

4.1 Dataset

We deployed our object detector model on a real drone equipped with onboard GPU and autopilot capable of autonomous operation. In this section, we explain the details of hardware and software we used to build our drone. For our experiments, we use VisDrone dataset (Zhu, Wen, Bian, Ling, & Hu, 2018). VisDrone is the most recent dataset which includes aerial images. Images were captured by different drones flying over 14 different cities separated by thousands of kilometers in China, in different scenarios under various weather and lighting conditions. The dataset consists of 263 video sequences formed by 179,264 frames and 10,209 static images and contains different objects such pedestrian, vehicles, bicycles, etc., and density (sparse and crowded scenes). Frames are manually annotated with more than 2.5 million bounding boxes and some attributes, e.g. scene visibility, object class, and occlusion, are provided.

4.2 Hardware

In the Results Our drone comprises of DJI Flame Wheel F450 quadcopter with an open-source Pixhawk PX4 autopilot mounted on an Or- bitty carrier board. We use NVIDIA Jetson TX2 which is a CUDA capable device and runs on Ubuntu Linux4Tegra(L4T) and JetPack-L4T 3.2. It features an integrated 256 CUDA core NVIDIA Pascal GPU, a hex-core ARMv8 65-bit CPU complex, and 8GB of LPDDR4 memory with a 128-bit interface. We have also used a Windows 10, Intel(R) Xeon(R) Silver 4114 CPU @2.20 GHz (20 CPUs), 2.2GHz, 32GB RAM with Nvidia Titan Xp GPU for training purposes. Our drone is equipped with ZED stereo camera, which is a standard 3D USB camera for object detection and image capturing. Additionally, it adds depth perception, positional tracking, and 3D mapping using advanced sensing technology based on human stereo vision. We use Venom 35C, 14.8V, 5000 mAh lithium polymer battery and Holybro Ublox Neo-M8N GPS module, which provides high sensitivity and minimal acquisition time while maintaining low system power. We use a Linux host PC machine to communicate and access the location of the drone.

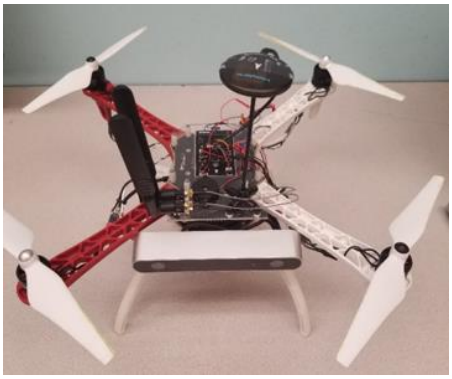

Figure 2. etson TX2 board attached on our DJI Flame Wheel drone.

4.3 Software

Jetson TX2 on our drone was first flashed with the latest OS image - Ubuntu 16.04. We further used JetPack installer to install developer tools on the drone. All other libraries useful for computer vision tasks such as TensorRT, cuDNN, CUDA, OpenCV were installed to jumpstart our development environment. We have used ROS Kinetic Kame distribution for hardware abstraction of Joystick and the Controller present on the drone. ROS also plays an important part in controlling devices at a very low-level as well as for transferring data. A ROS system typically consists of a number of independent nodes. For our environment, these nodes are MAVROS node, Controller Node, DNN, Camera and Joystick - each of which is able to communicate with each other using subscribe or publish messaging model. All nodes are registered with the master node (MAVROS in

our case), which in turn helps them to find and communicate with each other. The MAVROS enables MAVLink (Micro Air Vehicle Link) 2018 protocol to communicate with the PX4 (Flight Control Unit) on-board. The on-board Jetson TX2 had Wi-Fi access point enabled by us before installing it on the drone. As a result, the host PC could connect to the Jetson TX2 wirelessly and was able to access it remotely. By sending commands through the terminal, the host was able to control the drone as and when required by Secure Socket Shell (SSH) network protocol. We further use standard UART (Universal Asynchronous Receiver/Transmitter) communications for controls.

5. Results

In this section, we present a comprehensive quantitative evaluation of our object detection model. The performance of each model is based on these metrics, namely mean Average Precision (mAP) and inference time as frames per second (fps). In the context of object detection IoU metric captures the similarity between the predicted region and the ground truth region for an object present in the image and is calculated as the size of the intersection of predicted and ground-truth regions divided by their union. Precision is the widely used metric by the object detection community and is defined as the proportion of True Positives among all the detections of the system. The mean of average precisions for each category of objects calculated across all the possible IoUs is termed as mAP. Inference time is the rate at which an object detector is capable of processing the incoming camera frames. The rate is calculated as the number of frames per second.We trained our model on Nvidia Titan Xp GPU machine and deployed it on our onboard Nvidia Jetson TX2 for testing and inference(time taken by the model to predict output of test dataset).Our model was built in two designs. One with ResNet as backbone feature extractor and the other with MobileNet.

Table 1. Comparison performances depends on the backbone model in the proposed model

Backbone models	**No. of parameters (in billion)**	**mAP**	**Inference time (fps)**	**Inference power draw (in W)**
ResNet	36.7	30.6	8.6	120
MobileNet	13.5	29.2	14	85

Table 2. Comparison the proposed model performances with RetinaNet, HaL-RetinaNet

	Performances	
Model	**mAP**	**Inference time**
RetinaNet	23	8
HAL-RetinaNet	31.8	N/A
The proposed model	29.2	14

From our observations, our model with MobileNet base is very fast in terms of detection inference time compared to ResNet backbone. We achieved a real time speed of 14fps, maintaining the quality of detection at 29.2 mAP with MobileNet the overall size of our model decreased by a factor of 3, thereby reducing the memory footprint requirement on drone. We have calculated our power draw values on Nvidia Titan Xp GPU with a maximum possible power draw of 300W. Average power drawn with idle GPU is 15W. Our results show that our model combined with MobileNet as feature extractor draws very less power compared to the other models, which results in long battery life of the drone. Table shows the performance comparison of two models, RetinaNet and HAL-RetinaNet (the top performer of VisDrone 2018 object detection challenge) with our model. The results are evaluated on the basis of mean Average Precision on VisDrone 2018 validation and test data aerial images. The proposed model operates at 8.6 frames per second, which is well suited for real time object detection applications. Our deep feature pyramid network along with the modified loss function gave a better mean average precision of 30.6. Based on the accuracy, the class imbalance could reduce than the existing model.

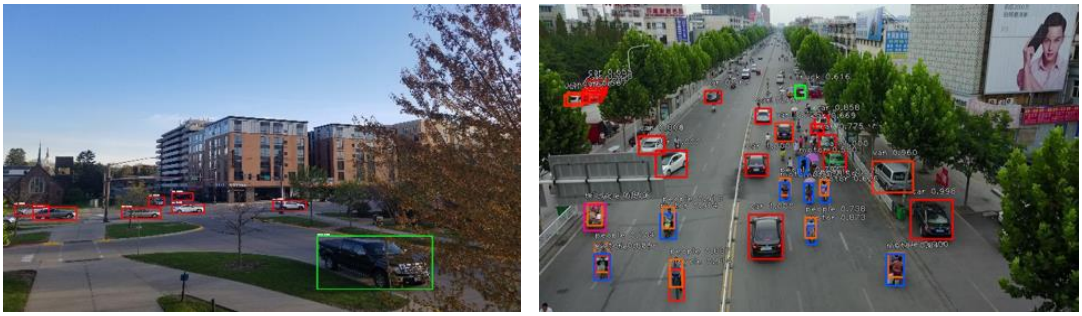

Figure 3. One of examples with the proposed model: left image from Ames, right image from Visdrone dataset

6. Discussion

UAVs equipped with intelligence and vision techniques such as object detection have a wide variety of real-world applications. Drones require memory efficient, computationally less expensive and fast solutions for the task of object detection. In this article, we proposed a Deep Feature Pyramid Network (DFPN) architecture embedded in object detector model combined with modified focal loss function to avoid the problem of inherent class imbalance and improve detection capability even with varying scales of images. We considered two networks, namely ResNet and MobileNet as backbone convolutional bases for our detection model. Even though our model combined with ResNet provided better accuracy about 1 mAP in terms of detection accuracy, a combination of MobileNet resulted in real time speeds with more about 5 fps without compromising the detection accuracy. We also compared our model with RetinaNet-ResNet50 and HAL-RetinaNet and shown that our model produced overall better results in terms of accuracy, speed and memory efficiency. Our model produced a performance of 30.6 mAP with an inference time of 14 fps. This result shows that our proposed model outperformed RetinaNet by 6.2 mAP. Therefore, we deployed our model on a real drone and were successfully able to detect objects in real-time.

Potential future works include performance improvements in terms of object detection by applying finer-level optimizations such as reducing the creation of redundant anchors, focusing more on missing occlusions etc. Safety is another major concern in terms of drones. There can be many situations such as collision with the objects, external disturbances like winds, chances of drone moving out of manual controller zone, battery issues, chances of getting stolen, and other safety hazards. We will be implementing these external drone-related safety features in our future work. Further, we focus on making our drone more intelligent by implementing features such as object tracking and path planning without class imbalance problem.

References

Bhaskaranand, M., & Gibson, J. D. (2011, November). Low-complexity video encoding for UAV reconnaissance and surveillance. In *2011-MILCOM 2011 Military Communications Conference* (pp. 1633-1638). IEEE. https://doi.org/10.1109/ism.2013.34

Parrott, E., Panter, H., Morrissey, J., & Bezombes, F. (2019). A Low Cost Approach to Disturbed Soil Detection Using Low Altitude Digital Imagery from an Unmanned Aerial Vehicle. *Drones*, *3*(2), 50. https://doi.org/10.3390/drones3020050

Doherty, P., & Rudol, P. (2007, December). A UAV search and rescue scenario with human body detection and geolocalization. In *Australasian Joint Conference on Artificial Intelligence* (pp. 1-13). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-76928-6_1

Sa, I., Hrabar, S., & Corke, P. (2015). Outdoor flight testing of a pole inspection UAV incorporating high-speed vision. In *Field and Service Robotics* (pp. 107-121). Springer, Cham. https://doi.org/10.1007/978-3-319-07488-7_8

Gleason, J., Nefian, A. V., Bouyssounousse, X., Fong, T., & Bebis, G. (2011, May). Vehicle detection from aerial imagery. In *2011 IEEE International Conference on Robotics and Automation* (pp. 2065-2070). IEEE. https://doi.org/10.1109/icra.2011.5979853

Lim, H., & Sinha, S. N. (2015, May). Monocular localization of a moving person onboard a quadrotor MAV. In *2015 IEEE international conference on robotics and automation (ICRA)* (pp. 2182-2189). IEEE.

https://doi.org/10.1109/icra.2015.7139487

Forster, C., Faessler, M., Fontana, F., Werlberger, M., & Scaramuzza, D. (2015, May). Continuous on-board monocular-vision-based elevation mapping applied to autonomous landing of micro aerial vehicles. In *2015 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 111-118). IEEE. https://doi.org/10.1109/icra.2015.7138988

Tang, T., Deng, Z., Zhou, S., Lei, L., & Zou, H. (2017, May). Fast vehicle detection in UAV images. In *2017 International Workshop on Remote Sensing with Intelligent Processing (RSIP)* (pp. 1-5). IEEE. https://doi.org/10.1109/rsip.2017.7958795

Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision* (pp. 2980-2988). https://doi.org/10.1109/iccv.2017.324

Venieris, S. (2018). Automated methodologies for mapping convolutional neural networks on reconfigurable hardware. https://doi.org/10.23919/date.2018.8342149

Saripalli, S., Montgomery, J. F., & Sukhatme, G. S. (2002, May). Vision-based autonomous landing of an unmanned aerial vehicle. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)* (Vol. 3, pp. 2799-2804). IEEE. https://doi.org/10.1109/robot.2002.1013656

Scaramuzza, D., Achtelik, M. C., Doitsidis, L., Friedrich, F., Kosmatopoulos, E., Martinelli, A., ... & Meier, L. (2014). Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in GPS-denied environments. *IEEE Robotics & Automation Magazine*, *21*(3), 26-40. https://doi.org/10.1109/mra.2014.2322295

Ryan, A., Zennaro, M., Howell, A., Sengupta, R., & Hedrick, J. K. (2004, December). An overview of emerging results in cooperative UAV control. In *2004 43rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601)* (Vol. 1, pp. 602-607). IEEE. https://doi.org/10.1109/cdc.2004.1428700

Chen, N., Chen, Y., Blasch, E., Ling, H., You, Y., & Ye, X. (2017, November). Enabling smart urban surveillance at the edge. In *2017 IEEE International Conference on Smart Cloud (SmartCloud)* (pp. 109-119). IEEE. https://doi.org/10.1109/smartcloud.2017.24

Meier, L., Tanskanen, P., Heng, L., Lee, G. H., Fraundorfer, F., & Pollefeys, M. (2012). PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision. *Autonomous Robots*, *33*(1-2), 21-39. https://doi.org/10.1109/icra.2011.5980229

Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster r-CNN: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, *39*(6), 1137-1149. https://doi.org/10.1109/tpami.2016.2577031

Dai, J., Li, Y., He, K., & Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. https://arxiv.org/abs/1605.06409

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788). https://doi.org/10.1109/cvpr.2016.91

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). Ssd: Single shot multibox detector. In *European conference on computer vision* (pp. 21-37). Springer, Cham. https://doi.org/10.1007/978-3-319-46448-0_2

Tripathi, S., Kang, B., Dane, G., & Nguyen, T. (2017, September). Low-complexity object detection with deep convolutional neural network for embedded systems. In *Applications of Digital Image Processing XL* (Vol. 10396, p. 103961M). International Society for Optics and Photonics. https://doi.org/10.1117/12.2275512

Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2117-2125). https://doi.org/10.1109/cvpr.2017.106

Redmon, J. (2013). Darknet: Open-source neural networks inc. From https://pjreddie.com/darknet/

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. https://arxiv.org/abs/1704.04861

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the*

IEEE conference on computer vision and pattern recognition (pp. 770-778). https://doi.org/10.1109/cvpr.2016.90

Zhu, P., Wen, L., Bian, X., Ling, H., & Hu, Q. (2018). Vision meets drones: A challenge. https://arxiv.org/abs/1804.07437

Lowe, D. G. (1999, September). Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision* (Vol. 2, pp. 1150-1157). Ieee. https://doi.org/10.1109/iccv.1999.790410

Bay, H., Tuytelaars, T., & Van Gool, L. (2006, May). Surf: Speeded up robust features. In *European conference on computer vision* (pp. 404-417). Springer, Berlin, Heidelberg. https://doi.org/10.1007/11744023_32

Mammadli, R., Wolf, F., & Jannesari, A. (2019). The art of getting deep neural networks in shape. *ACM Transactions on Architecture and Code Optimization (TACO)*, *15*(4), 1-21. https://dl.acm.org/doi/10.1145/3291053

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 2961-2969). https://doi.org/10.1109/iccv.2017.322

Asli, A. E., Roghair, J., & Jannesari, A. (2019). Energy-aware Goal Selection and Path Planning of UAV Systems via Reinforcement Learning. *arXiv preprint arXiv:1909.12217*. https://arxiv.org/abs/1909.12217

Mizuno, K., Terachi, Y., Takagi, K., Izumi, S., Kawaguchi, H., & Yoshimoto, M. (2012, October). Architectural study of HOG feature extraction processor for real-time object detection. In *2012 IEEE Workshop on Signal Processing Systems* (pp. 197-202). IEEE. https://doi.org/10.1109/sips.2012.57