2019

Efficient object detection model for real-time UAV applications

Subrahmanyam Vaddi
Iowa State University

Efficient object detection model for real-time UAV applications

by

Subrahmanyam Vaddi

A thesis submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

Major: Computer Science

Iowa State University

Ames, Iowa

2019

DEDICATION

I would like to dedicate this thesis to my mother Bharathi and to my brother Gyani for their endless love, encouragement, support, and prayers from thousands of miles away.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

Abbreviation	Explanation
UAV	Unmanned Aerial Vehicle
GPU	Graphic Processing Unit
DFPN	Deep Feature Pyramid Network
mAP	mean Average Precision
CNN	Convolutional Neural Network
R-CNN	Region Convolutional Neural Network
R-FCN	Region Fully Convolutional Network
YOLO	You Only Look Once
SSD	Single Shot Multi-Box Detector
FPN	Feature Pyramid Network
SVM	Support Vector Machine
ROI	Region Of Interest
RPN	Region Proposal Network
ROS	Robot Operating System
SGD	Stochastic Gradient Descent
NMS	Non Maximum Suppression
IoU	Intersection over Union

ACKNOWLEDGMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis. First and foremost, Dr. Jannesari Ali for his guidance, patience and support throughout this research and the writing of this thesis. I would also like to thank my committee members for their time and support for this work: Dr. Matthew Darr and Dr. Carl Chang. I would additionally like to thank my department program administrators Carla Harris and Mindy Hanna for their help and support they provided throughout my graduate career at ISU. I would also like to thank my friends and faculty for their loving guidance, support and encouragement.

ABSTRACT

Unmanned Aerial Vehicles (UAVs) especially drones, equipped with vision techniques have become very popular in recent years, with their extensive use in wide range of applications. Many of these applications require use of computer vision techniques, particularly object detection from the information captured by on-board camera. In this paper, we propose an end to end object detection model running on a UAV platform which is suitable for real-time applications. We propose a deep feature pyramid architecture which makes use of inherent properties of features extracted from Convolutional Networks by capturing more generic features in the images (such as edge, color etc.) along with the minute detailed features specific to the classes contained in our problem. We use VisDrone18 dataset for our studies which contain different objects such as pedestrians, vehicles, bicycles etc. We provide software and hardware architecture of our platform used in this study. We implemented our model with both ResNet and MobileNet as convolutional bases. Our model combined with modified focal loss function, produced a desirable performance of 30.6 mAP for object detection with an inference time of 14 fps. We compared our results with RetinaNet-ResNet-50 and HAL-RetinaNet and shown that our model combined with MobileNet as backend feature extractor gave the best results in terms of accuracy, speed and memory efficiency and is best suitable for real time object detection with drones.

CHAPTER 1. INTRODUCTION

In recent years, autonomous Unmanned Aerial Vehicles (UAVs) especially drones equipped with cameras have become very popular with the wide variety of their applications such as surveillance[1], aerial mapping[2], search and rescue[3], infrastructural inspection[4], precision agriculture[5] etc. Understanding visual data collected from these drones autonomously, has been the area of increasing interest. Visual object detection is one of the prominent aspects of applications of drones and is critical to have in fully autonomous systems. However, object detection with drones is very challenging and complex as it is affected by various imaging conditions such as noise in the images, blur, low resolution, small target sizes etc. The task is even more difficult because of the limited computational resources available on the drones and the need for almost real time performance in many applications such as navigation, traffic management etc.

Many UAV studies have been made to detect certain application specific single objects such as vehicles[6], pedestrians[7], autonomous navigation and landing[8] and some studies which detect multiple objects such as[9], [10].The key challenges in deploying the vision and intelligence capabilities on a UAV platform are 1) to achieve minimum power consumption in order to minimize its effect on battery power draw and overall flight time of the drone 2) require less memory footprint and computational power as typical UAVs have resource limitations 3) process the input data from its camera with low latency and perform faster in order to perform critical tasks such as object detection, avoidance and path navigation in real time. Conventionally UAV's on-board computing framework consists mainly of general purpose devices, such as multicore CPUs and micro-controllers that consume low power. Later the high computational requirement of modern computer vision techniques such as deep learning, led to introduction of massively parallel computing architectures with rise of GPUs as accelerators. Even though there are many alternatives to on-board processing such as cloud-centric solutions, they pose additional overheads such as latency in video feed and information security risk. Hence, in this paper we are focused on developing deep learning algorithms for object detection running on UAV equipped with embedded hardware platforms suitable for real time applications.

Traditionally hand-tuned features were used for object detection and recognition. With the breakthrough of deep learning using Convolutional Neural Networks there was a striking performance increase in dealing with these computer vision tasks. The key idea is to learn object features and model from raw pixel data of image. Training these deep learning models usually requires large datasets but this problem has also been overcame by the release of many large scale labeled datasets such as ImageNet, COCO, Pascal VOC etc. However, these methods require huge amount of computational power and built-in memory. There has always been a trade-off between the detection performances with the speed. Using these techniques on general purpose low-priced drones has thus became challenging because of the memory, power and computational requirements of these devices. In this thesis, we propose an object detection model which is computationally less expensive, memory efficient and fast without compromising the detection performance, running on a drone. We report on experiments measuring mean Average Precision (mAP) and inference time using low cost quad-copter drone as a hardware platform, in the scenario of detecting target objects in an environment containing objects like pedestrians, buses, bicycles etc. We propose a novel Deep Feature Pyramid Network (DFPN) architecture combined with a modified loss function to avoid class imbalance and achieved real time object detection performance on real drone environment.

In this thesis, we implemented our DFPN Network architecture with both ResNet and MobileNets as backbone and shown that our model with MobileNet as backbone feature extractor gave best results well suited for the drone applications with a mAP of 29.2 and real time detection speed of 14fps. We have compared our results with two existing object detectors, 1) RetinaNet which is the current state of the art object detector and 2) HAL-RetinaNet, the best model performer of VisDrone 2018 competition. We show that our model outperforms both of them in terms of combined speed and mAP metrics.

In brief, the rest of the thesis is organized as follows: It begins with the review of literature survey of different existing object detection models and their evolution as current state of the art models. Chapter 2 also elaborates the details of each model, implementation and drawbacks. Chapter 3 gives details of our system hardware and software specifications along with the dataset we used. Chapter 4 provides the methodology of our proposed model and explains the DFPN architecture in detail. Chapter 5 gives the experimental analysis of our model performance in terms of speed, mAP, memory footprint and power consumed by each model. Finally Chapter 6 gives the conclusion remarks and provides scope for the future work.

CHAPTER 2. REVIEW OF LITERATURE

The first part of this chapter gives a brief summary of related work done on UAV based object detection whereas the second part explains the evolution of several deep learning models widely used for object detection.

2.1 UAV-based Object Detection

Vision assisted UAVs have wide range of applications such as intelligent navigation, path planning and autonomous driving. In early studies of UAVs, [11] explored the vision-based strategies during UAV landing period for autonomous control. In [12], authors enhanced the navigation performance by incorporating vision techniques into their UAV design. UAVs equipped with computer vision could also be used for various video surveillance platforms in different application scenarios. Earlier, in [13], authors gave a general overview towards the implementation of object detection based computer vision techniques for aerial image surveillance. Later many studies focused on data driven applications of UAVs. For example, UAVs been used for smart urban surveillance in [14]. Recently an end to end IoT system based on drones has been proposed in [5] for data driven precision agriculture application. Many recent studies make use of flight control systems utilizing deep learning frameworks for vision assisted tasks to perform scene analysis of aerial images. PIXHAWK[15] is one of the popular flight control systems designed with computer vision capabilities to execute the obstacle recognition and autopilot mode during UAV operation, which we have used on-board for our experiments.

2.2 Object Detection

Object detection is identification of one or several objects present within an image. It can be broadly divided into two steps namely classification and localization. It is an added challenge to object classification in such a way that it accurately locates all the objects present in the image along with detecting their presence. Object Detection have several important applications where numerous tasks which require human supervision can be automated by detecting objects in images. Generic object detection aims at classifying and locating objects, labeling them with rectangular bounding boxes indicating the confidence score of prediction.

The first deep learning object detector model was called Overleaf Network[13] 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[16], YOLO[17], R-FCN[18], SSD[19] and RetinaNet[20]. The main goal of these designs is to improve (1) the detection accuracy measure in terms of mAP and (2) the computational complexity of their models so that they can achieve real-time performance for embedded and mobile platforms[21]. These detection models can be divided into two categories based on their high-level architecture: 1) the single step approach and 2) the two step (region-based) approach. The single step approach achieved faster results and shown higher memory efficiency whereas the two-step approach achieved better accuracy than former. In the following sections we discuss both of these approaches with few models in each of them.

2.2.1 Region Based Detectors

Region based detectors divide object detection in two stages. The first stage 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. Some of the region based detectors are R-CNN, Fast R-CNN, Faster R-CNN, FPN and R-FCN.

2.2.1.1 Region Convolutional Neural Network (R-CNN)

The R-CNN Model[22] was a highly influential model that has shaped the structure of modern object detectors. It was the first detector which proposed the two step approach. It has a region proposal model which proposes blobs or regions in the image which have higher probability of being objects. R-CNN generates these region proposals using an algorithm called Selective Search. Selective Search processes the image through sliding windows of different sizes, and for each size tries to group together the adjacent pixels by color, intensity, or texture to identify objects. Selective Search outputs around 2,000 region proposals of various scales which are then cropped from input image and are warped to 7 x 7 pixels. Once these regions are created, R-CNN passes each of them through a modified AlexNet to generate feature maps as shown in Figure 2.1. Each feature map is then unrolled and fed into two fully connected convolutional layers to generate a 4078 dimensional vector. This vector is then input to two separate small Support Vector Machine (SVM) networks namely a classification network head and a regresssion network head. The classification head is responsible for predicting the class of object that the region belongs to and the regression head is responsible for predicting the offsets to the coordinates of the region to better fit the object.

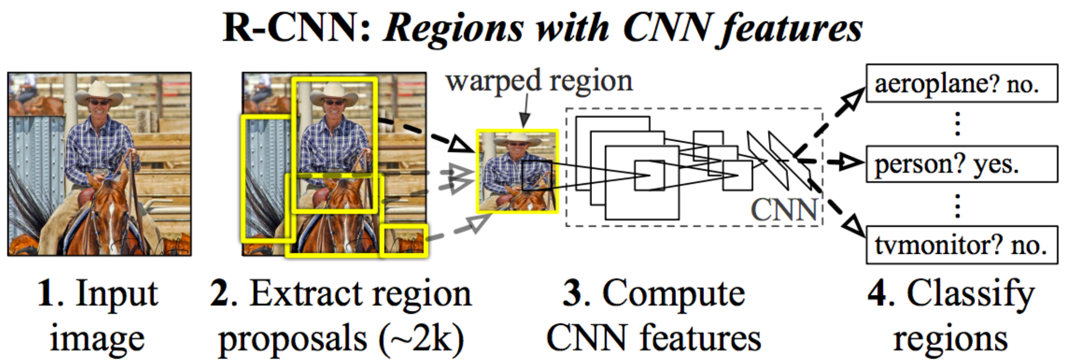

Figure 2.1 Region Convolutional Neural Network (RCNN)[22].

The R-CNN model takes 47 seconds to detect objects in a single image since it has a complex multi-step training pipeline which requires careful tweaking of parameters.

2.2.1.2 Fast R-CNN

Fast R-CNN[23] is an improvement of R-CNN which consists of a convolutional head and two SVM heads for classification and regression. RCNN is quite inefficient as it has overhead of running every region proposal through the convolutional base. The Fast RCNN aims to reduce this overhead by running the convolutional base just once over the entire image to generate a feature map. It crops the regions from this generated feature map instead of the entire input image thus reducing both the time and space. The cropping procedure is done by an algorithm called ROI pooling which takes the coordinates of the regions obtained via the Selective Search and directly crops it out from the feature map of the original image. Also, Fast RCNN introduced a single step training pipeline and a multitask loss, enabling the classification and regression heads to be trained simultaneously as shown in Figure 2.2.

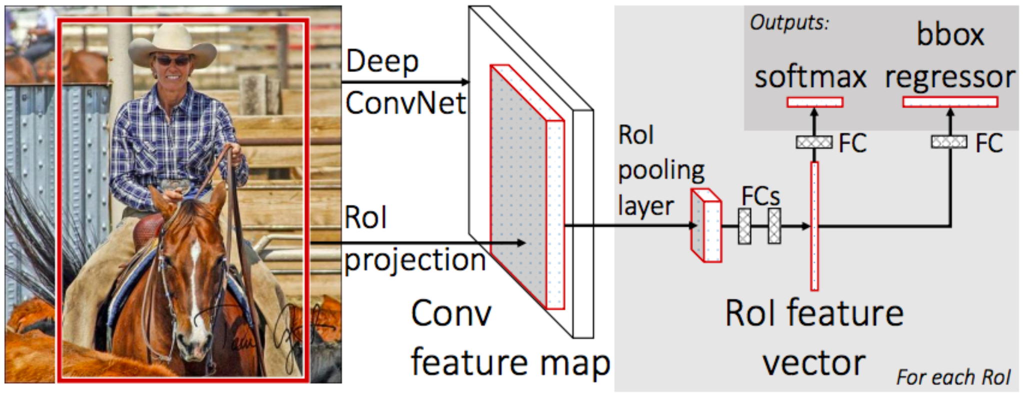

Figure 2.2 Fast RCNN Architecture[23].

Fast RCNN is more speed and memory efficient compared to RCNN. Specifically, it reduces training time exponentially from 84 hrs to 9 hrs and also reduces inference time from 47 seconds to 0.32 seconds. Secondly, it introduces a simpler single step training pipeline and a new loss function. This loss function is easier to train and does not suffer with the gradient explosion problem.

2.2.1.3 Faster R-CNN

Even though Fast RCNN[16] improved a lot in terms of speed and memory footprint, it still suffers with an overhead of region proposal step done by selective search algorithm, taking lot of time to generate proposals. The Faster RCNN introduced the Region Proposal Network (RPN) to replace Selective Search, which have the capability of predicting regions of multiple scales and aspect ratios across the image using concept called anchors. Anchors are rectangular crops of image covering all shapes and sizes of regions. Some of common aspect rations such as 1:1, 1:2 and 2:1 and scales like 32px, 64px and 128px were considered. Once a location in the image is decided upon, these 9 anchors are cropped from that location. Anchors are cropped out using sliding window approach, cropping after every x number of pixels in the image. The architecture remains similar to Fast RCNN, where the input image is fed into convolutional base, which outputs a feature map which is in turn fed into two sub-networks, a classification sub-network and a regression sub-network as shown in Figure 2.3. Faster RCNN has end to end deep learning pipeline and has achieved an inference time of 198ms per image.

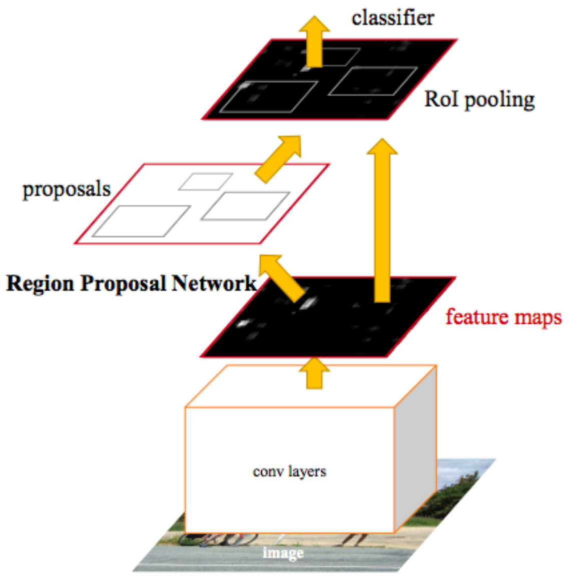

Figure 2.3 Faster RCNN Architecture[16].

2.2.1.4 Feature Pyramid Network (FPN)

Scale invariance was one of the problems not dealt by Faster RCNN. The model should be able to detect the object up close and also from far way. FPN[24] model solves this problem by creating multiple feature maps that are designed to represent the image at different scales. Hence the anchors no longer have to take care of the scales. The convolutional base takes the input through various scales, creating feature maps at each scale. These feature maps are taken from the last layer of each scale, the intuition being that the deepest features contain the most salient information for that scale. Each of the feature maps goes through a 1 by 1 convolution to bring the channel depth to C, where C = 256. These maps are then added element wise to the upsampled version of the feature map one scale above them as shown in Figure 2.4. This procedure is also called a lateral connection. Once lateral connections have been performed for each scale, the updated feature maps go through a 3 by 3 convolution to create the final set of feature maps. After generating these multiple feature maps, the Faster RCNN network runs on each scale.

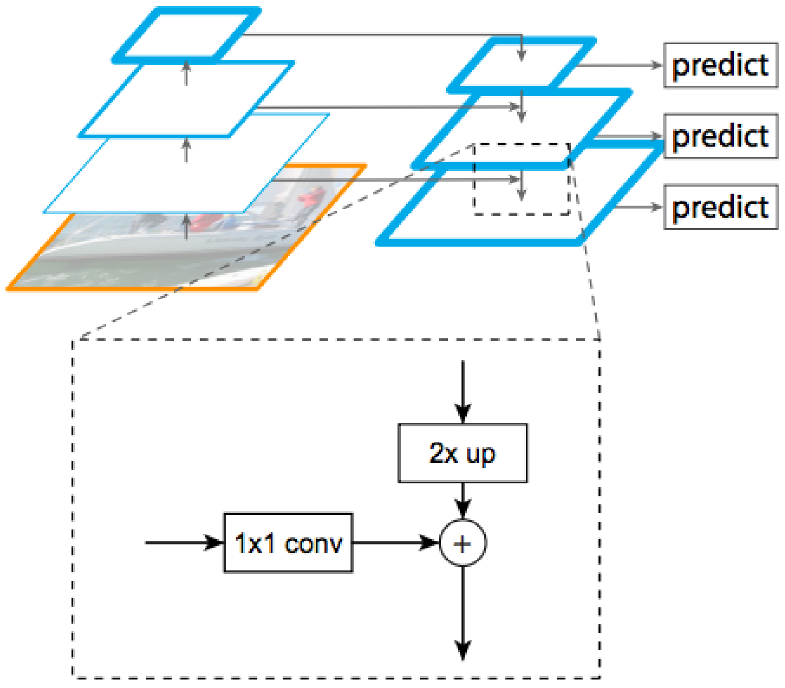

Figure 2.4 FPN Architecture[24].

2.2.2 Single Step Detectors

Single step detectors are simple and fast models with reasonable accuracy compared to region based networks. They are similar to RPN networks however, instead of predicting object/non-objects, they directly predict object classes and coordinate offsets. Some of the single step detectors are SSD, YOLO and RetinaNet.

2.2.2.1 You Only Look Once (YOLO)

YOLO[17] detection model is more targeted for real time performance. YOLO divides the image into SxS grid where the authors considered S to be 7. The concept of anchors remains the same except that each of the grid cell act as a pixel point on the original image. Each grid cell predicts only one object even though it represents a fixed number of bounding boxes (2 in case of YOLO v1). Unlike Faster RCNN, YOLO has only one neural network head. It outputs feature maps of size 7 by 7 by ((x+1+5*k)) where k indicates the number of anchors, (x+1) term is the total number of object categories including the background class.

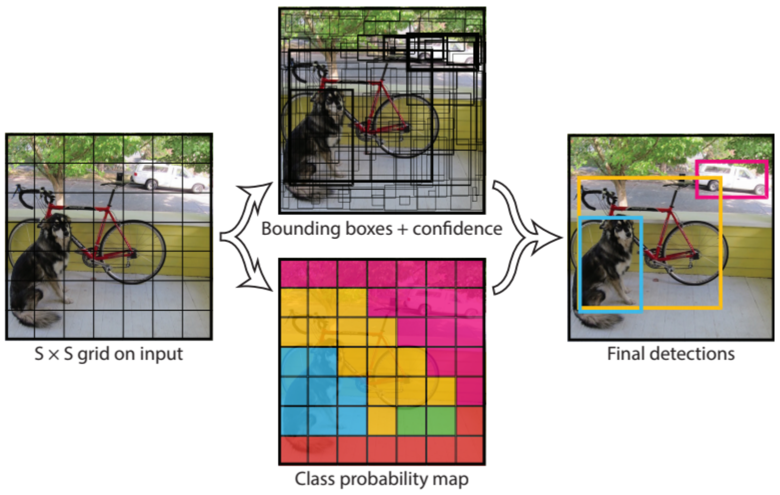

Figure 2.5 YOLO processing[17].

The number 5 comes from the four offsets of bounding box (x, y, height, width) and an extra parameter that indicates box confidence score, which reflects how likely the box contains an object (objectness) and how accurate the bounding box is. Training of YOLO is based on a loss function which corresponds directly to detection performance and it trains the entire model jointly. The following figure 2.5 shows how a YOLO network detects objects from an image. Although YOLO is close to real time, it has spatial limitations on bounding box predictions as each grid cell predicts only two bounding boxes and can have only one category. Thus it does not perform well on small objects and those appear in groups. Also it faces large class imbalance problem during the training.

2.2.2.2 RetinaNet

RetinaNet[20] is a one stage object detector with a backbone network and two subnetworks one for classification and one for regression. The backbone network generates the convolutional feature map over an entire input image similar to Faster R-CNN. RetinaNet employs a Feature Pyramid Network (FPN) built on top of ResNet as backbone. Classification subnet is responsible for predicting the probability of object presence at each spatial position for each of the A anchors and K object classes. It takes an input feature map with C channels from a pyramid level, and applies four 3x3 conv layers, each with C filters and each followed by ReLU activations. Finally sigmoid activations are attached to the outputs. Focal loss is applied as the loss function. Box Regression subnet outputs the object location with respect to anchor box if an object exists. Figure 2.6 shows the architecture of RetinaNet.

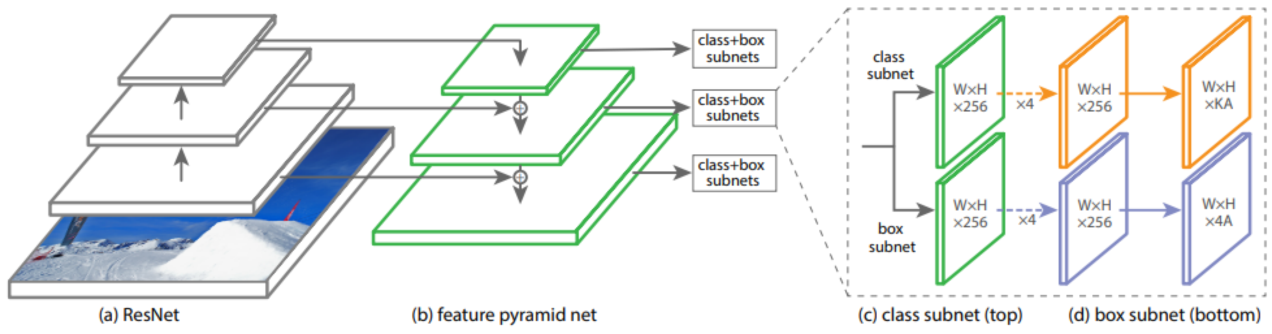

Figure 2.6 RetinaNet Architecture[20].

The reason why one step detectors before RetinaNet have lagged behind two step detectors in accuracy was an implicit class imbalance problem encountered while training. RetinaNet solved this problem with their novel loss function and boasts the state of the art results in accuracy and speed compared to all other detectors.

Even though RetinaNet performed well on ImageNet dataset, it had major drawbacks in performance on aerial image dataset because of presence of very small scaled objects in aerial images. In the following chapter, we explain our methodology which uses inherent Deep Feature Pyramid Network (DFPN) architecture to overcome this problem.

CHAPTER 3. METHODOLOGY

3.1 Introduction

Our object 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 [24] and [20]. 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.

Class imbalance occurs because a single step detector densely samples regions from all over the image. This leads to a high majority of regions belonging to the background class. The two step detectors avoid this problem by using an attention mechanism (RPN) which focuses the network to train on a small set of examples. SSD [19] tries to solve this problem by using techniques like a fixed foreground-background ratio of 1:3, online hard example mining [26] or bootstrapping [27]. These techniques are performed in single step implementations to maintain a manageable balance between foreground and background examples. However, they are inefficient as even after applying these techniques, the training data is still dominated by easily classified background examples. To avoid class imbalance, we used a dynamic loss function as in [20] which down weights the loss contributed by easily classified examples. The scaling factor decays to zero when the confidence in predicting a certain class increases. This loss function can automatically down weight the contribution of easy examples during training and rapidly focus the model on hard examples.

3.2 Performance Impacting Factors

There are many design and data factors that might majorly impact the performance of object detection. Feature extractors play a vital role in determining how well a model learn. Many state of the art feature extractors can be used for our task such as VGG16, ResNet, Inception, MobileNet etc. We have chosen both ResNet and MobileNet for our model. Also the combination of parameters for extractors such as number of filters, strides, padding may also effect the quality of features extracted. Input image resolution is another vital factor as poor resolutions might drastically reduce the performance. We use 800 x 1333 pixel resolution for input images. The other factors are matching strategy and IoU used while calculating loss, number of proposals made by the model, bounding box encodings, data augmentation techniques used, training dataset, localization loss function used etc. We discuss these factors we considered for our model in the following sections.

3.3 Object Detector Model

We use a simple object detector model by combining the best practices gained from previous research studies.

Input: An aerial image fed as an input to the model.

Targets: The network uses the concept of anchors to predict regions. As our DFPN is integrated with the model, 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 $1:1$, $1:2$, $2:1$ have been augmented by the factors of 1, 2/3 and 4/3 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 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 R-CNN. 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.

Architecture: 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.

In this study, we introduce a Deep Feature Pyramid Network (DFPN) architecture as shown in Figure 3.1. Similar to FPN [24], our goal is to leverage a ConvNets 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 topdown 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 as shown in Figure 3.1. 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.

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

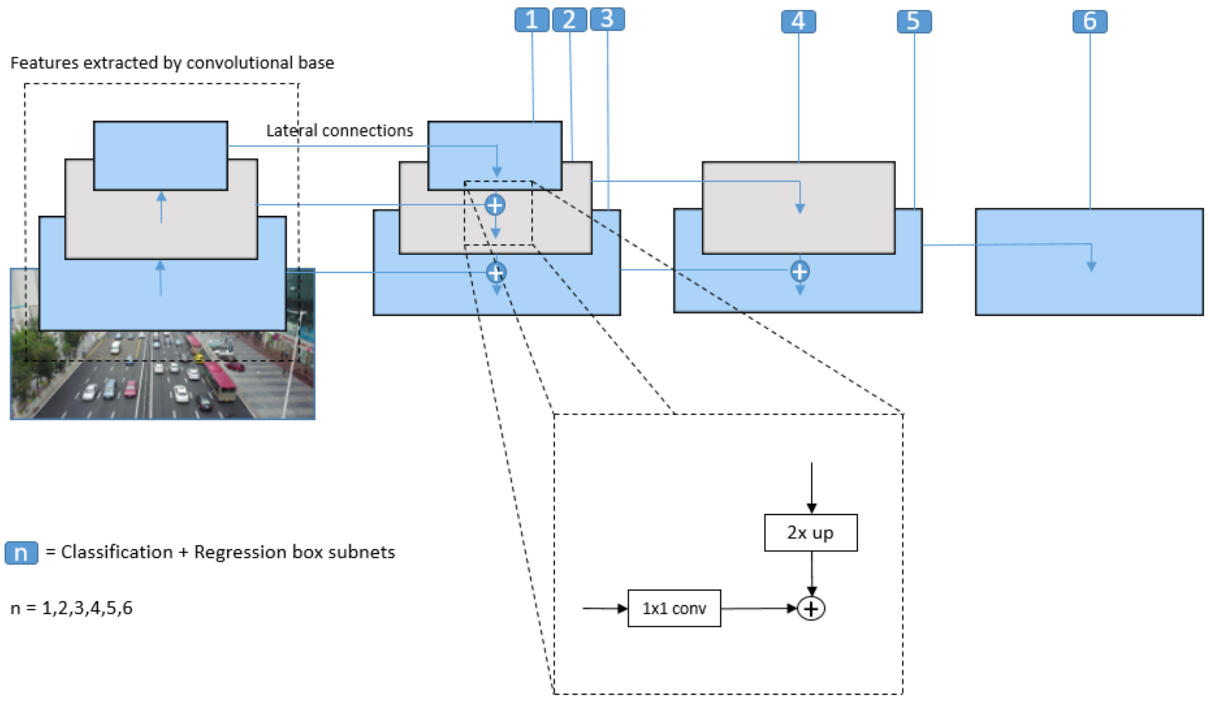
Features extracted by convolutional base
Lateral connections
1
2
3
4
5
6
2x up
1x1 conv
n = Classification + Regression box subnets
n = 1,2,3,4,5,6

Figure 3.1 Our Network Architecture.

feature map of depth AK 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 coordinates offsets of bounding boxes.

Loss Function: 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 [20] given by equation 3.1,

$$F = \alpha^2 * (1 - p)^\gamma * c \tag{3.1}$$

where c is cross entropy loss, α is the hyper parameter 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. 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.

Training and Inference: 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 1000 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.

Convolutional bases: Object detectors use 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 basis of three factors such as speed, accuracy and memory footprint. As per our requirement, we chose ResNet [28] and MobileNet[29] 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 depth wise separable convolutions which form their backbone and have led to a speedup in computing the feature map without sacrificing the overall quality.

CHAPTER 4. SYSTEM ARCHITECTURE

In this chapter, we give the details about our system architecture specifically hardware and software specifications and dataset we used for our experiments.

4.1 Hardware Setup

We deployed our object detector model on a real drone equipped with onboard GPU and autopilot capable for autonomous operation. Our drone comprises of DJI Flame Wheel F450 quadcopter with an open source Pixhawk PX4 autopilot mounted on a Orbitty 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 @2.20GHz(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 drone. Figure 4.1 shows the hardware setup of our drone.

4.2 Software Setup

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

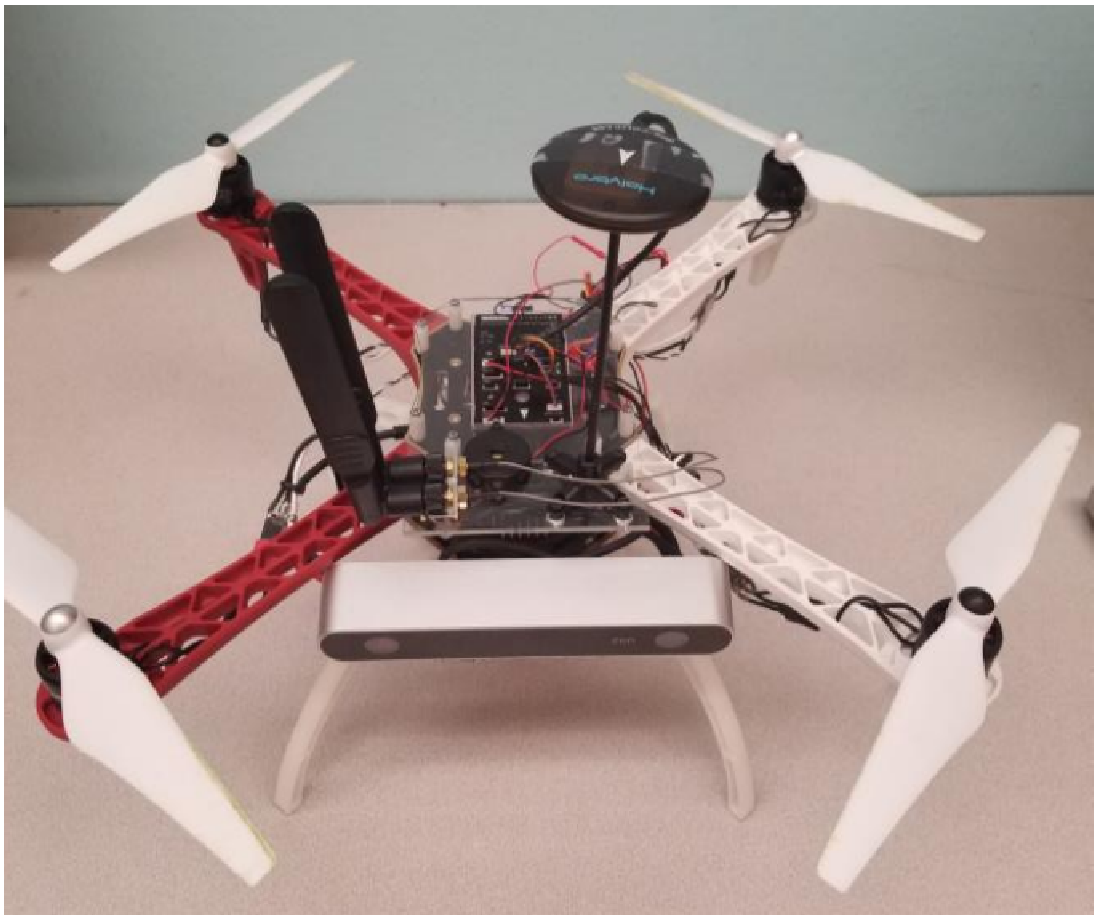

Figure 4.1 Jetson TX2 board attached on our DJI Flame Wheel drone.

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) mav (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 by Secure Socket Shell (SSH) network protocol. We further use standard UART (Universal Asynchronous Receiver/Transmitter) communications for controls.

4.3 Dataset

For our experiments we use VisDrone [25] dataset. 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. VisDrone is available at http://www.aiskyeye.com.

CHAPTER 5. EXPERIMENTAL RESULTS

In this section, we present a comprehensive quantitative evaluation of our object detection model. The performance of each model is evaluated on the basis of three metrics namely Intersection over Union (IoU), 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 size of 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 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 5.1 shows the comparison of these two designs in terms of their performance, power drawn and network size.

Table 5.1 Our models performance comparison with ResNet and MobileNet as feature extractors

Backbones for our model	**No. of parameters (in billions)**	**mAP**	**Inference time**	**Inference power draw (in W)**
ResNet	36.7	30.6	8.6 fps	120
MobileNet	13.5	29.2	14 fps	85

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. Also 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 5.2 Comparison of object detection performance between RetinaNet, HAL-RetinaNet and our model in terms of mAP and inference time.

Model	**mAP**	**Inference time**
RetinaNet	23	8 fps
HAL-RetinaNet	31.8	N/A
Our Model (With MobileNet)	29.2	14 fps

Table 5.2 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 (mAP) on VisDrone 2018 validation and test data aerial images. From the table 1, we can see that our model operates at 14 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. Figure 5.1 shows the object detection result on one of the aerial images from VisDrone dataset and Figure 5.2 shows that on one of the frames captured by our drone.

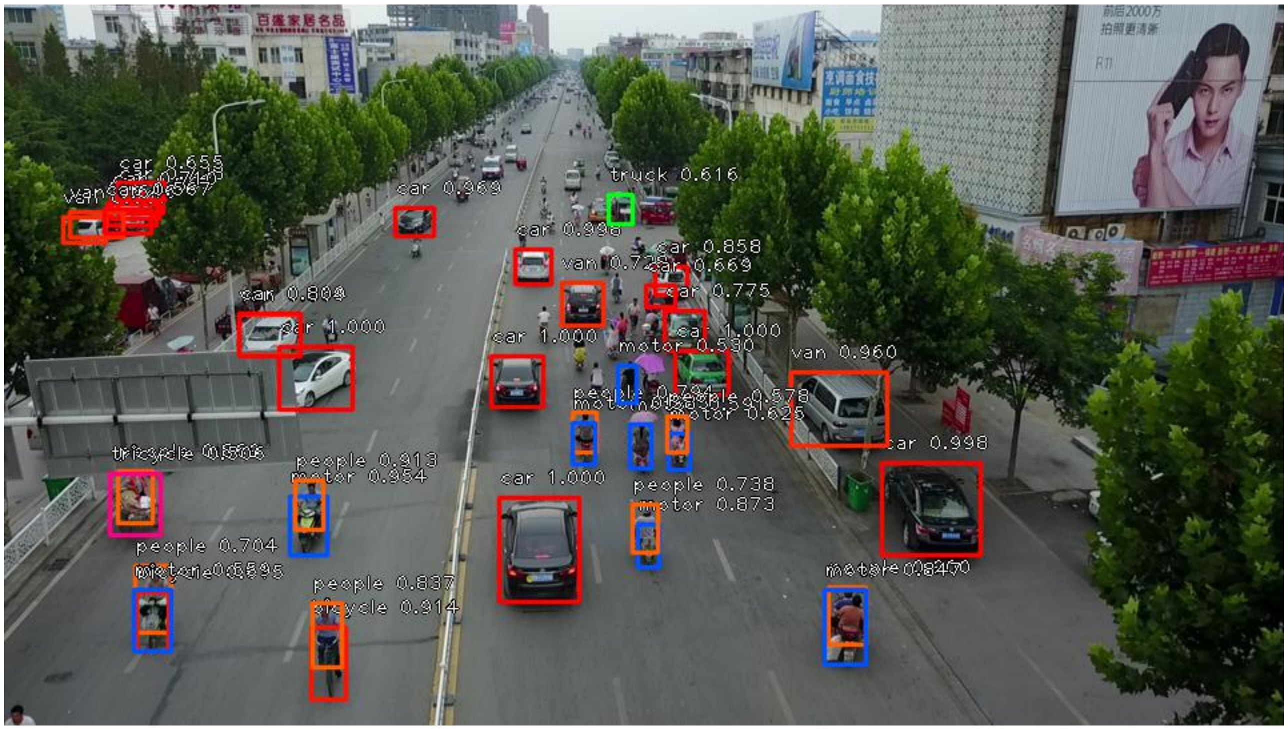

Figure 5.1 Our models prediction output on one of the aerial images in VisDrone18 dataset.

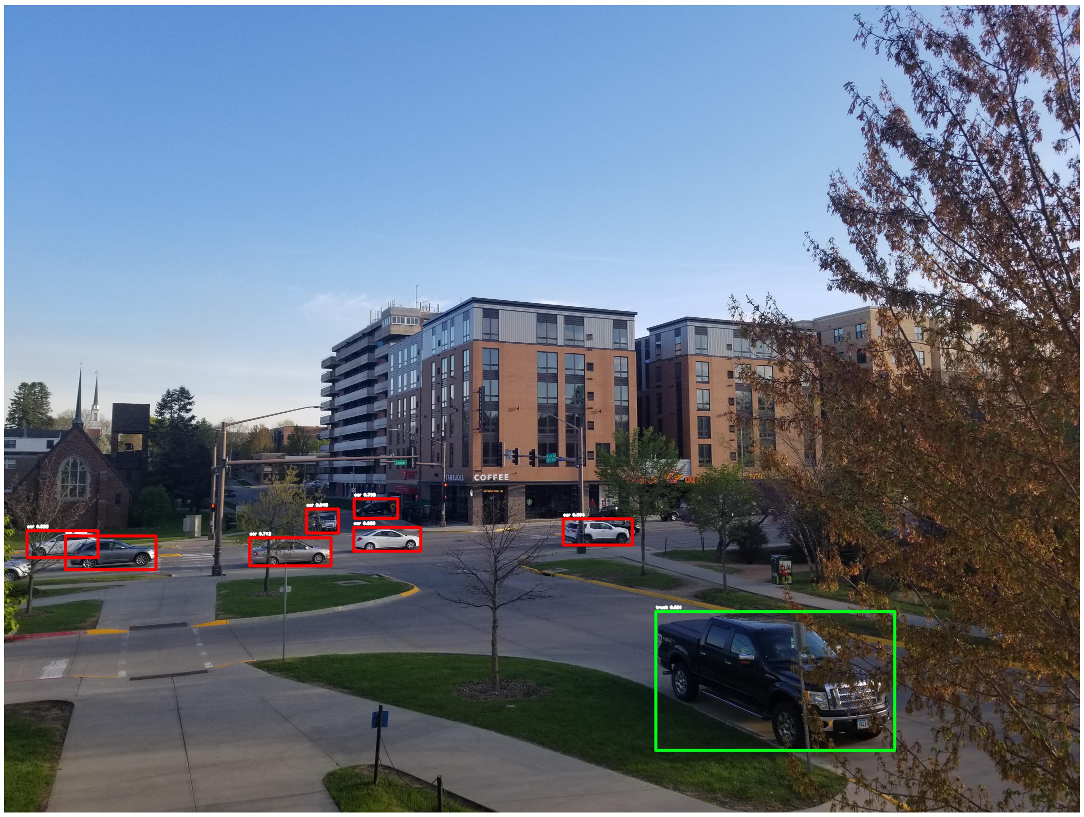

Figure 5.2 Our models prediction output on one of the frames captured by our drone.

CHAPTER 6. CONCLUSION AND FUTURE WORK

UAVs equipped with intelligence and vision techniques such as object detection have wide variety of real world applications. Drones require memory efficient, computationally less expensive and fast solutions for the task of object detection. In this paper we proposed a Deep Feature Pyramid Network (DFPN) architecture embedded in object detector model combined with modified focal loss function to avoid 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 results in terms of detection accuracy, combination of MobileNet resulted in real time speeds 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. 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 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.

REFERENCES

[1] Bhaskaranand,M, a. J. (n.d.). *Low-complexity video encoding for UAV reconnaissance and surveillance. Proc. IEEE Military Communications Conference (MILCOM), pp. 1633-1638, 2011.*

[2] Madawalagama, S. *Low Cost Aerial Mapping with Consumer Grade Drones. 37th Asian Conference on Remote Sensing,2016.*

[3] Rudol, P. D. *A UAV search and rescue scenario with human body detection and geolocalization. Advances in Artificial Intelligence, pp. 1-13,2007.*

[4] I. Sa, S. H. *Outdoor flight testing of a pole inspection UAV incorporating highspeed vision. Springer Tracts in Advanced Robotics, pp. 107-121,2015.*

[5] Chandra, R. *FarmBeats: Automating Data Aggregation. Farm Policy Journal Vol 15: pp. 7-16, 2018*

[6] Gleason,J, A. N. *Vehicle detection from aerial imagery. Proc. IEEE International Conference on Robotics and Automation (ICRA), pp. 2065-2070,2011.*

[7] Lim, a. S. *Monocular Localization of a moving person onboard a Quadrotor MAV. Proc. IEEE International Conference on Robotics and Automation (ICRA), pp. 2182-2189,2015.*

[8] Forster,C, M. F. *Continuous on-board monocular-vision-based elevation mapping applied to autonomous landing of micro aerial vehicles. Proc. IEEE International Conference on Robotics and Automation (ICRA),pp 111-118, 2015*

[9] al, T. T. *Fast Vehicle Detection in UAV Images. International Workshop on Remote Sensing with Intelligent Processing (RSIP),2017*

[10] Christos Kyrkou, G. P. *DroNet: Efficient convolutional neural network detector for real-time UAV applications. Design, Automation and Test in Europe Conference and Exhibition (DATE),2018.*

[11] Saripalli, S. M. *Vision-based autonomous landing of an unmanned aerial vehicle. Robotics and automation,Proceedings. ICRA'02. IEEE international conference on (Vol. 3, pp. 2799-2804). IEEE,2002.*

[12] Scaramuzza, D. A. *Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in GPS-denied environments. IEEE Robotics and Automation Magazine,2014.*

[13] Ryan, A. Z. *An overview of emerging results in cooperative UAV control. 43rd IEEE Conference on (Vol. 1, pp. 602-607),2004.*

[14] Chen, N. C. *Enabling smart urban surveillance at the edge. In Smart Cloud (SmartCloud). IEEE International Conference on (pp. 109- 119),2017.*

[15] Meier, L. T. *PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision. Autonomous Robots, 33(1-2), pp.21-39,2012.*

[16] Ren,S. K. H. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 11371149,2017.*

[17] Redmon,J., S. D. *You Only Look Once: Unified, Real-Time Object Detection. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779788,2016.*

[18] Dai,J, Y. L. *R-FCN: Object Detection via Regionbased Fully Convolutional Networks. NIPS, pp. 379387,2016.*

[19] Liu, D. A.-Y. *SSD: Single Shot MultiBox Detector. ECCV, pp. 2137,2016.*

[20] Tsung-Yi Lin, P. G. *Focal Loss for Dense Object Detection. ICCV,2017.*

[21] Subarna Tripathi, B. K. *Low-complexity object detection with deep convolutional neural network for embedded systems. Proc.SPIE, vol. 10396, pp. 10 396 10 396 15,2017.*

[22] Ross Girshick et al. *Rich feature hierarchies for accurate object detection and semantic segmentation, CVPR, 2014.*

[23] Ross Girshick. *Fast R-CNN, 2015 IEEE International Conference on Computer Vision (ICCV)*

[24] Lin,T, P. D. *Feature pyramid networks for object detection. CVPR,2017.*

[25] Pengfei Zhu, L. W. *Vision meets drones: A challenge. CoRR, abs/1804.07437. URL http://arxiv.org/abs/1804.07437, 2018*

[26] Shrivastava, A. G.. *Training region-based object detectors with online hard example mining. CVPR, 2016.*

[27] Rowley,H. S. B. *Human face detection in visual scenes. Technical Report CMUCS-95-158R, Carnegie Mellon University,1995.*

[28] He, X. J. *Deep residual learning for image recognition. CVPR,2016.*

[29] Andrew G Howard et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. CoRR, 2017.*