

# **OBJECT DETECTION WITH MOTION TRACKING IN SECURITY AND SURVEILLANCE**

*Submitted by*

<b>NAGATEJA GUTTIKONDA</b>	<b>1104370</b>
<b>PRAVEEN KUMAR VENKATESAN</b>	<b>1111385</b>
<b>RIKHIL GADE ROZARIO</b>	<b>1098167</b>

**COMP - 5435 – FB - Topics in Computer Vision**

**2019 FALL**

## **ABSTRACT**

Security and Surveillance plays a vital role in various economical applications and in many military applications. Military applications include intruder alert systems as it is a matter of security breach leading to many consequences and causalities. A Surveillance application system has been implemented using YOLO (You Only Look Once) that detects the Unmanned Ariel Vehicles generally termed to be drones. The detection The Applications detects the Unmanned Ariel vehicles and alerts the professionals regarding the security breach. The breach is inspected by the officials for further action. The detection system implemented increases the speed and accuracy for faster response in military grade applications. The system identifies the objects through their feed and classifies the object with the machine as it is trained with the similar images based on the parameters specified and then the detection of the object is carried out by the system. The experimental results has been obtained with the test data proving that the speed of the detection and the accuracy of the prediction has been substantially increased eventually increasing the response rate of the action to be done.

## **TABLE OF CONTENTS**

<b>CHAPTER NO</b>	<b>TITLE</b>	<b>PAGE NO</b>
	<b>ABSTRACT</b>	<b>ii</b>
	<b>LIST OF FIGURES</b>	<b>v</b>
	<b>LIST OF ABBREVIATION</b>	<b>vi</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 OVERVIEW	1
	1.2 OBJECTIVE	2
<b>2</b>	<b>LITERATURE REVIEW</b>	<b>3</b>
	2.1 LITERATURE SURVEY	3
	2.2 CHALLENGES	7
<b>3</b>	<b>METHODOLOGY</b>	<b>8</b>
	3.1 YOLO (YOU ONLY LOOK ONCE)	8
<b>4</b>	<b>DESCRIPTION AND DESIGN</b>	<b>12</b>
	4.1 SYSTEM DESCRIPTION	12
	4.2 SYSTEM DESIGN	13
<b>5</b>	<b>SYSTEM REQUIREMENTS</b>	<b>17</b>
	5.1 REQUIREMENTS	17
	5.2 UBUNTU	17
<b>6</b>	<b>SYSTEM IMPLEMENTATION</b>	<b>18</b>
	6.1 DARKNET INSTALLATION	18
	6.2 ANNOTATION OF IMAGES	19

	6.3	TRAINING AND TESTING	23
	6.4	OUTPUT SCREENSHOTS	36
7		<b>COMPARISON</b>	<b>38</b>
8		<b>DISCUSSION</b>	<b>39</b>
9		<b>CONCLUSION AND FUTURE WORK</b>	<b>40</b>
		<b>REFERENCES</b>	<b>41</b>

## **LIST OF FIGURES**

<b>FIGURE NO</b>	<b>FIGURE NAME</b>	<b>PAGE NO</b>
<b>1</b>	<b>BOUNDING BOX PREDICTION</b>	<b>8</b>
<b>2</b>	<b>YOLO ALGORITHM DETECTING OBJECTS</b>	<b>9</b>
<b>3</b>	<b>YOLO V3 NETWORK ARCHITECTURE</b>	<b>10</b>
<b>4</b>	<b>FLOW DIAGRAM</b>	<b>13</b>
<b>5</b>	<b>ACTIVITY DIAGRAM</b>	<b>14</b>
<b>6</b>	<b>SEQUENCE DIAGRAM</b>	<b>15</b>
<b>7</b>	<b>USE-CASE DIAGRAM</b>	<b>16</b>

## **LIST OF ABBREVIATION**

<b>ABBREVIATION</b>	<b>EXPANSION</b>
<b>YOLO</b>	YOU ONLY LOOK ONCE
<b>UAV</b>	UNMANNED ARIEL VEHICLE
<b>CNN</b>	CONVOLUTIONAL NEURAL NETWORKS
<b>R-CNN</b>	REGION BASED CONVOLUTIONAL NEURAL NETWORKS

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 OVERVIEW**

Computer Vision and image processing has found its way in today's day to day application as it is found to be efficient and useful in most of the industries and organizations. Object detection being one of the live applications of Image processing is being used to detect objects in images/live feed.

Now a days as, technology evolves, need for security and surveillance also gradually increases. Intelligent Security and Surveillance systems are required by the military in order to maintain peace and order as well as the protection of confidential information in an efficient and less time consuming way has gained much importance.

Manual detection of Unmanned Aerial vehicles is not visually possible due to many reasons which includes its shift speed and its variation of small sizes and optical illumination. Additional to that, changes in appearance, and viewpoint and noise significantly reduces the performance. Detection of UAV vehicles using radars has high false alarm rates also limited detection rates. Radar returns of birds are more similar to drones. Also setup cost of these systems is expensive and more time consuming. Hence Optical detection is found to be an efficient way in detection of UAVs. In traditional object detection algorithms, the bounding boxes are first identified and the feature extraction is carried out. Finally the object is classified. These traditional methods lack accuracy and the detection rate is comparatively less.

This detection system was also trained for detection of weapons but the false negative rate is comparatively high. Hence to overcome these issues and to increase more speed and accuracy, we implement the system using YOLO (You only look once) for detection of UAVs and Weapons. The system implemented reduces the installation cost.

The only component required for the application is a High resolution Camera.YOLO version 3 is used in the system application. The System application captures the drones from live feed /images. Then the image is processed and the bounding boxes are calculated in a single step. Then the image is classified and then the detection is carried out with the trained system. The system is trained with wide variety of images which takes more amount of time. If the detection is true positive the personnel authorized will take further action. If it is a true negative, then the system neglects it and doesn't alert the people.

## **1.2 OBJECTIVE**

The Main Objective of the project is to predict the Objects in a very fast and accurate manner.

1. Detect the drones and weapons to avoid security breach.
2. Increase the efficiency of the detection by increasing the detection rate.
3. Increase the speed of the detection of objects.
4. High accuracy in the prediction of the objects to avoid false positives.



## CHAPTER 2

### LITERATURE REVIEW

#### 2.1 LITERATURE SURVEY

1. Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi,” You Only Look Once: Unified, Real-Time Object Detection” 9 May 2016

A new approach for object detection is presented in form of YOLO where we frame object detection as regression problem to spatially separated bounding boxes and associated class probabilities. Whole detection pipeline is a single network where it can be optimized directly on the detection performance. YOLO allows more localization errors compared to state-of - the-art detection systems but is less likely to predict false positives on the context. While generalising from natural images to other areas such as artwork, it outperforms other methods of detection, including DPM and R-CNN. YOLO predicts many bounding boxes per cell on the grid. We want only one bounding box predictor to be responsible for each object at the time of learning. We designate one predictor to be "responsible" to predict an event based on which prediction with ground truth has the highest current IOU. YOLO places strict spatial restrictions on predictions of boundary boxes as each grid cell predicts only two boxes and can only have one group. One of the highest performing detection methods is our combined Fast R-CNN + YOLO system. From the combination with YOLO, Quick R-CNN gets a 2.3 percent increase, improving 5 spots on the public leader board.

2. Ajeet Ram Pathak, “Application of Deep Learning for Object Detection”, Procedia Computer Science · January 2018

There are also enunciations of deep learning systems and services available for object detection. This paper explores in-depth training methods for state-of - the-art object detection systems. How the CNN learns about the feature is a major problem. Object

detection systems are required to robustly perform invariant object detection for illumination, occlusions, deformations, and variations in intra-class. Because occlusions and deformations follow long-distance statistical distribution, data sets are likely to miss the uncommon occlusions and deformations of objects. Object detection is applicable in many domains, ranging from defence (surveillance), interaction of human computers, robotics, transport, retrieval, etc. Within a few hours, sensors used for persistent monitoring generate petabytes of image data. These data are reduced to geospatial data and integrated with other data in order to get a clear understanding of the current scenario. The frameworks are studied from the point of view of features exhibited, interface, support for deep learning model. Benchmarked datasets for object localization and detection released in worldwide competitions are also covered. Object detection systems are currently small with 1-20 cluster nodes with GPUs. These systems should be expanded to enable full-time video frames to be created in real time at 30 to 60 per second.

### 3. Juan Wu, Bo Peng, Zhenxiang Huang, Jietao Xie,” Research on Computer Vision-Based Object Detection and Classification”

Due to their fast response, high accuracy and strong adaptability, computer vision techniques become particularly important in the various applications of farming. Under certain complex conditions affected by nature and humans, the role is difficult due to variability in product quality disparities. Color-based strategies strive to segment the appropriate colours of the region of interest for further processing. To segment image with strong contrast between object and context, colour thresholding is particularly useful and easy. Thresholding can be achieved by setting the colour value range and pixel identification, and the image can be segmented. For each pixel if its colour value is within the reference range, the pixel can be considered as correct object pixel, if not a background pixel. Basic idea of color indexing is to compare two images of coloured objects with their colour histograms, as the colour histogram is used to index the object

and store it in the template database. While Hough transform was initially used to detect lines and circles, its associated transforms were eventually generalised to detect arbitrary shapes. Support Vector Machine extended statistic learning theory to classification, employing Structural Risk Minimization principal to improve generalization capability.

4. Joseph Redmon Ali Farhadi,” YOLOv3: An Incremental Improvement”,8 April 2018

YOLOv3 uses logistic regression to estimate an objectness score for each bounding box. This should be 1 if the boundary box above overlaps an element of ground truth with more than any other boundary box preceding. If the boundary box above is not the best but overlaps an object of ground truth by more than a certain amount, we ignore the prediction. Each box predicts the classes the bounding box may contain using multi label classification technique. We do not use a SoftMax as we have found it is redundant for good performance, instead we simply use independent logistic classifiers. YOLOv3 predicts boxes at 3 different scales. We extract various features across scales using the similar technique of pyramid networks. The last of these predicts a 3-d tensor encoding bounding box, objectness, and class predictions. A new network is used for performing the feature extraction. Our new network is a hybrid approach between the network used in YOLOv2, Darknet-19, and that newfangled residual network stuff. Our network uses successive  $3 \times 3$  and  $1 \times 1$  layers of convolution but now has some shortcut for connections. We use 53 layers of convolution, so it is named as Darknet-53. Darknet neural network framework is used for both the training and testing. YOLOv3 is an excellent detector. It's quick, and it's precise. On the COCO average AP between .5 and .95 IOU metric, it's not that great. But on the old detection metric of .5 IOU, it's very good.

5. Xie Weige, Zhiguo Shi, Xiufang Shi,” Real-Time Drone Detection Using Deep Learning Approach”, Third International Conference, MLICOM 2018, Hangzhou, China, July 6-8, 2018

The use of drones arbitrarily poses a major threat to public safety and privacy, so it is necessary to detect the intruding drones in the real time. By changing its architecture and adjusting its parameters to better accommodate drone detection, we improve a well-performed deep learning system, i.e., You Only Look Once. The robust detector needs to be trained with many training images, and we also suggest a semi-automatic data labelling system based on the Kernelized Correlation Filters tracker to speed up the pre-processing of the training images. We use a public domain drone dataset, USC Drone Dataset, consisting of 30 YouTube video sequences and capturing various drone models with different appearances for our training and testing work. These video clips have a frame resolution of  $1280 \times 720$  and their duration time is about one minute. For the identification of drones, labelling the positions of drones particularly in the videos is an intensive and tedious activity. So, we use the KCF tracking algorithm to semi-automatically label these videos. With a  $416 \times 416$  input image, we get an output feature map of  $13 \times 13$ . Secondly, sliding window sampling is performed on the feature map, and each center predicts  $k$  anchor boxes with different sizes and aspect ratios. Thirdly, the anchor box simultaneously predicts the class probability and coordinates. In order to detect drones in real time and accurately, we resize images to a higher resolution to suit small objects. Our model's second improvement is to change the dimensions of the predefined anchor box to make the network more adaptable to drone detection. Extensive tests have shown that the detector can achieve high accuracy in real time detection.

6. Song Han, William Shen, Zuozen Liu,” Deep Drone: Object Detection and Tracking for Smart Drones on Embedded System”

Capturing high-quality images or videos with the most sophisticated drones requires precise manual control and is highly error prone. We propose Deep Drone, an embedded

system framework, to provide vision for drones: to allow the drone to detect and track automatically. We use vision component which is an integration of advanced detection and tracking algorithm. We have introduced our device on various hardware platforms, including both desktop GPU (NVIDIA GTX980) and embedded GPU (NVIDIA Tegra K1 and NVIDIA Tegra X1) and tested frame rate, power consumption and accuracy on multiple drone recorded images. Drones are a special hardware platform with limited space and weight. We can't afford to use desktop GPUs for detection, even though it's much faster. For tracking we use KCF Algorithm over others the only reason behind this decision is that we want real-time tracking for our drone so that we could give consistent control command to the drone. Using the DJI Onboard SDK, we track the camera. We need to send activation data to the CoreAPI driver first. Our system achieved real time performance at 71 frames per second (fps)for tracking and 1.6fps for detection on NVidia TX1.

## **2.2 CHALLENGES**

Some of the challenges faced by the current detection systems:

1. Detection of UAV using radars has high false alarm rates.
2. The installation and maintenance cost of the system is very high.
3. Faster prediction of the objects is not achieved in traditional methods.
4. Misprediction of the detection with unwanted objects.

## CHAPTER 3

### METHODOLOGY

#### 3.1 YOLO (YOU ONLY LOOK ONCE)

YOLO primarily has 24 convolutional layers that act as two dense levels for prediction and detection research. Its architectural feature is Darknet, which is a neural networking framework. YOLO divides the picture over a cell grid from 13 to 13. Depending on the input size, the size of these 169 cells varies. In our experiments, the cell size for a 416 bis 416 input size was 32 tant32. The prediction of a number of boxes in the image is made by each cell. The network also calculates the confidence that the bounding box actually holds an item and that the enclosed entity is likely to be a certain category for each bounding box.

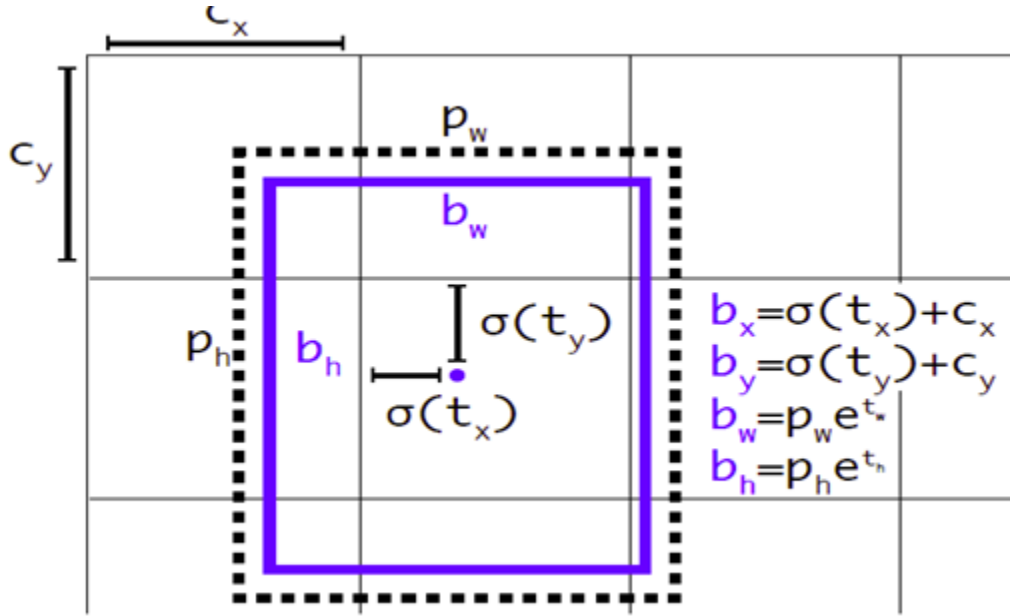
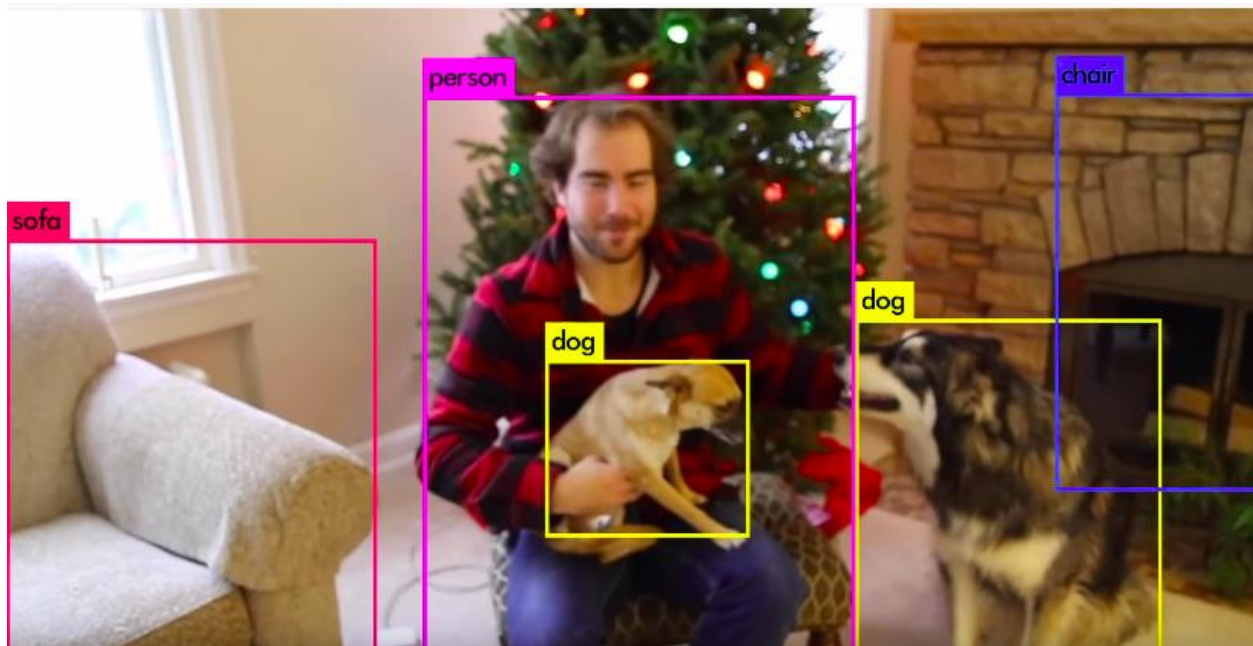


Fig 1: Bounding box prediction

The algorithm functions by splitting an image into a cell grid; both class probabilities are anticipated for each cell bounding box and its confidence values.

Confidence is measured in terms of IOU (intersection over union), a metric that practically analyses how much an identified object intersects with the truth of the ground as part of the overall area that is distributed over by both. Most of these bounding boxes were removed because they have low confidence or because they have a very high confidence level in the same item as another bounding box. This method is known as non-maximum exclusion. The failure of the algorithm takes account of the locality estimates, proportions, confidence values of those predictions and the classes expected.



*Fig 2: YOLO algorithm detecting objects*

YOLOv3 introduces further significant improvements by contrasting previous versions, including the calculation of bounding boxes at various scales. YOLOv3 uses a logistic regression to estimate the target rating for each bounding container. The bounding box should be 1 if an element of the ground truth is already overlaid by more than any other bounds. If the bounding box before is not the best and overlaps a real object with a threshold, we do not know the forecast.

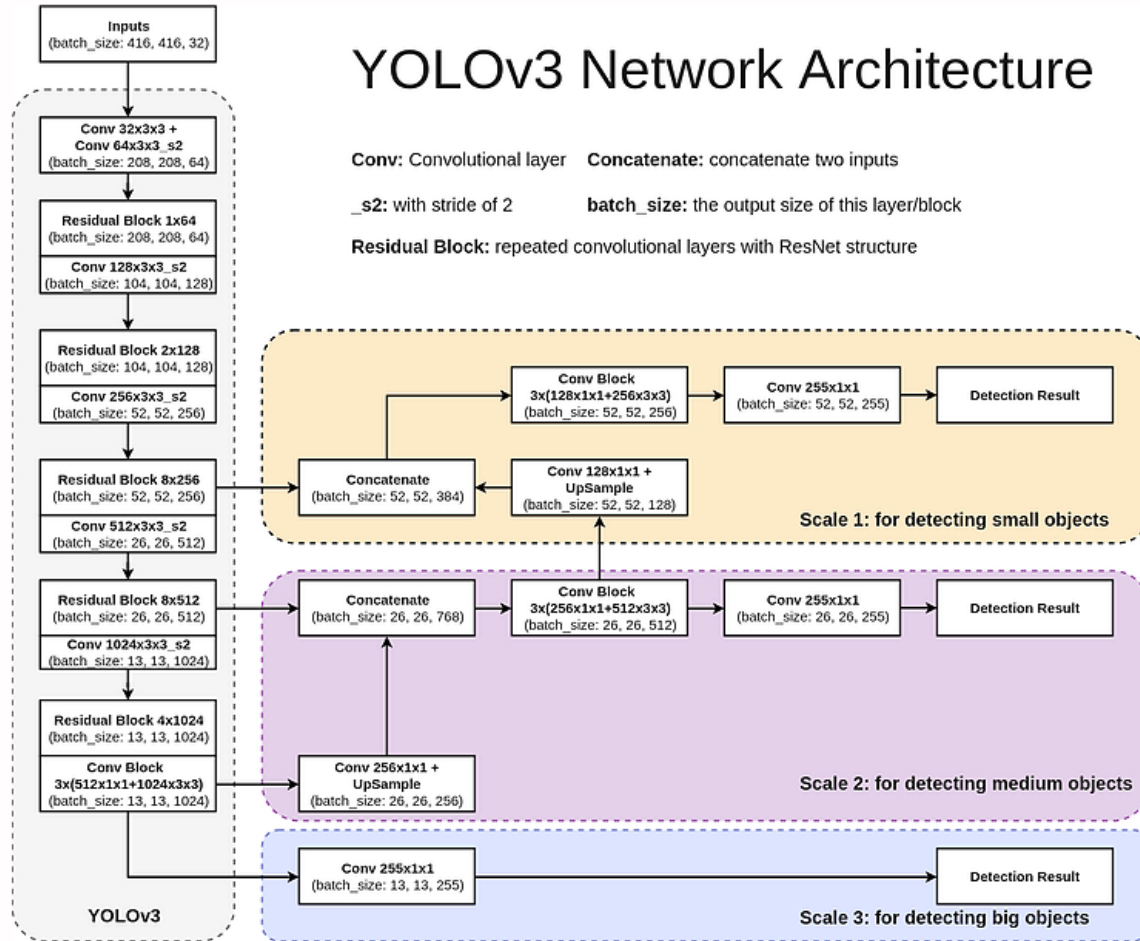


Fig 3: YOLOv3 Network Architecture

The most impressive features of YOLOv3 are the identification based on three different scales, and the detection on three different sizes is achieved through the use of 1 x 1 detector kernels in function maps in three different sizes. Down sample of the input picture dimensions by 32, 16 and 8 are correctly defined. YOLO v3 uses nine anchor boxes in total. Every scale uses three.

The fundamental component of the network is Darknet, which is extended to 53 convolution layers in this version. Darknet is an open-source and C / CUDA-written framework for neural networks and the foundation of YOLO. Darknet is used as a YOLO



learning platform, which implies that it establishes network architecture. The repo is provided with several configuration files for training on various architectures.

YOLO struggled with little things in the past for prediction. But now, this trend is a reversal. We can see YOLOv3's relatively high quality with the new multi-scale predictions. When we draw precision versus speed on  $AP_{50}$ , we can see that YOLOv3 has considerable advantages over all other detection systems. It's faster and better. On the other side, YOLO addresses the issue of object detection totally differently. It only transmits the entire image once via the network.

## **CHAPTER 4**

### **DESCRIPTION AND DESIGN**

#### **4.1 SYSTEM DESCRIPTION**

##### **1. IMAGE ACQUISITION:**

The Image of the object for detection is obtained from the live surveillance feed. The resolution of the Feed should be comparatively moderate for faster prediction.

##### **2. IMAGE PREPROCESSING:**

Annotate the objects in all the images of the dataset

##### **3. FEATURE EXTRACTION:**

The frame in the images is divided into number of cells in the grid. The bounding box has been calculated for each frame and the confidence score of the bounding Box has been calculated. The bounding Box is calculated if the center of the object is present in the box having 5 parameters namely Object Class ID,x\_center, y\_center, box width, box height. If multiple objects are present in the same image, then the labels are created for each object in the image.

##### **4. MODEL CONFIGURATION:**

Yolo-drone.cfg, drones.data, drones.names are created for training and testing of the model.Drone.data contains the path for all the files required for YOLOV3.Yolo.cfg is the configuration file for the model. Drone.names contains the name of the class objects(one class since one object is detected).

## 5. TRAINING:

The model uses the existing yolo v3 weights to train on the drone image dataset and creates a new weights file for our custom drone class detection.

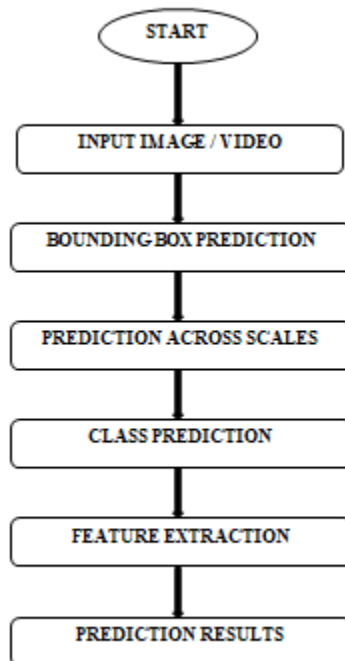
## 6. TESTING:

Using the trained weights on the drone data set, we detect the image and the object is displayed in the console.

## 7. PREDICTION OUTPUT:

The Image is displayed with its bounding box over the object along with its class label.

## 4.2 SYSTEM DESIGN



*Fig 4: Flow Diagram*

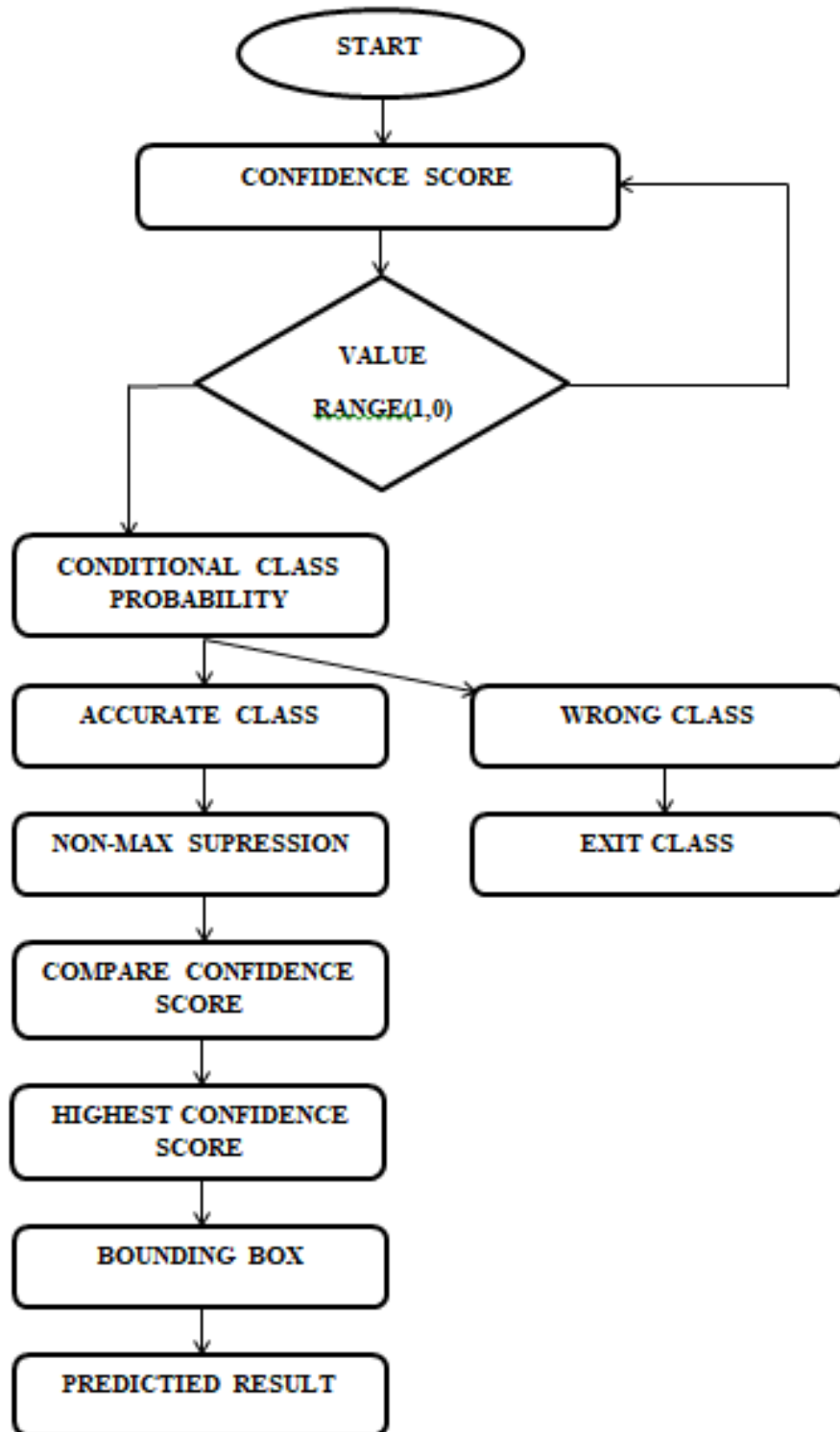


Fig 5: Activity diagram

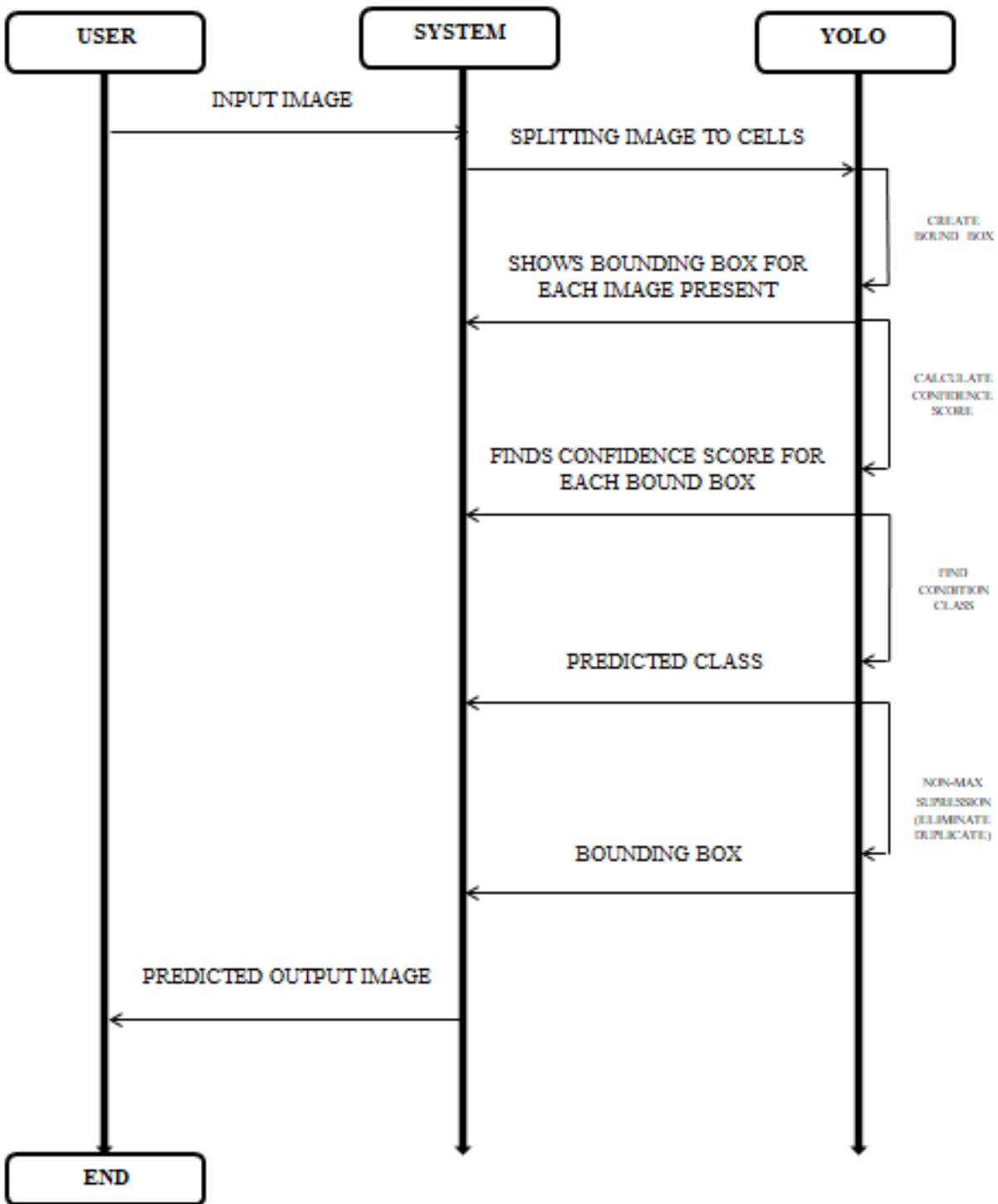


Fig 6: Sequence diagram

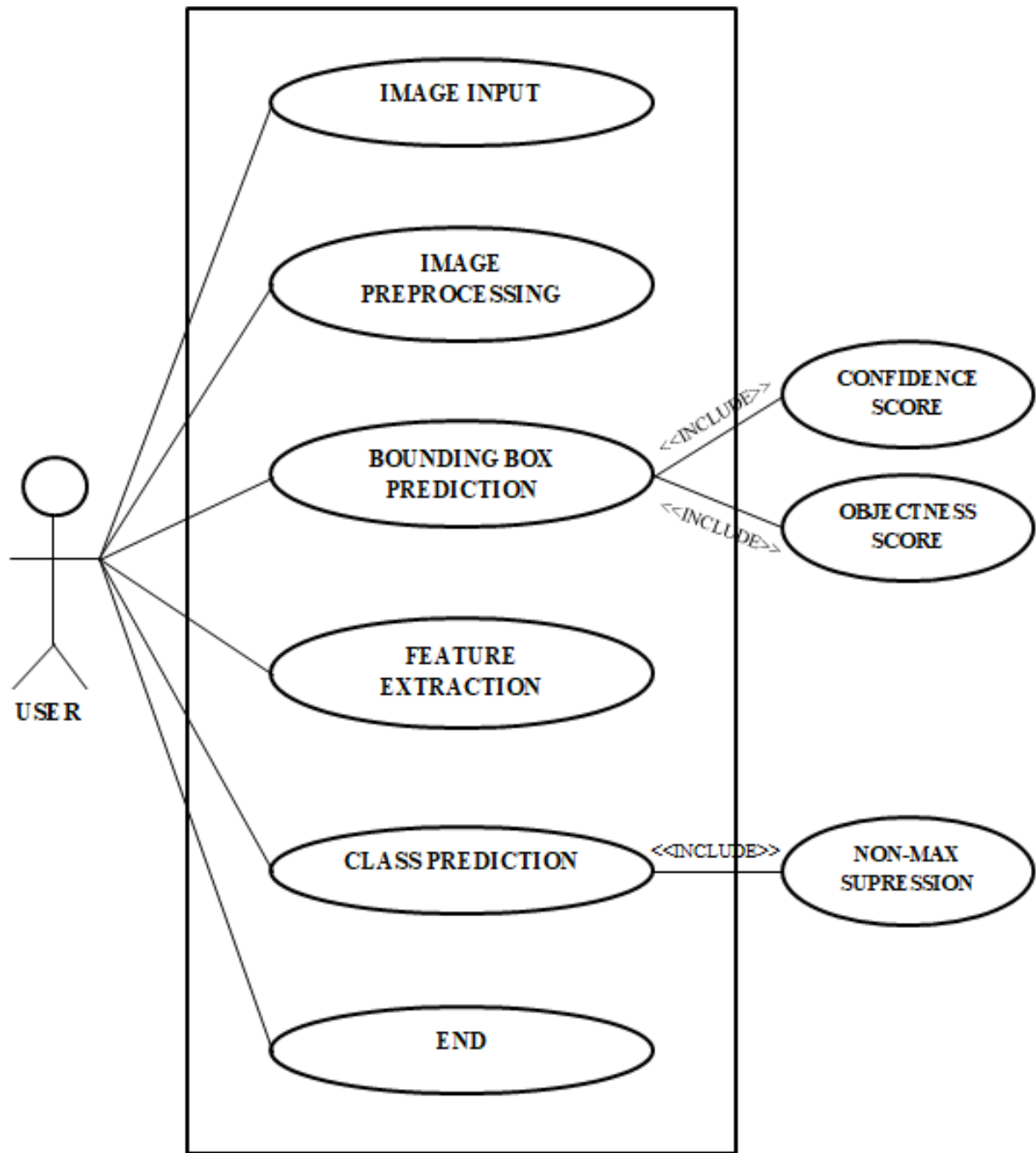


Fig 7: Use-Case diagram

## **CHAPTER 5**

### **SYSTEM REQUIREMENTS**

#### **5.1 REQUIREMENTS**

PROCESSOR & GPU: i7(8<sup>TH</sup> GEN) & NVIDIA

OPERATING SYSTEM: Ubuntu (version 18.04.3)

PROGRAMMING LANGUAGE: Python (version 3.6)

#### **5.2 UBUNTU**

Ubuntu software consists of many pre-installed software's and also operates under the General Public License (GPL). While coming to security it is highly secured and also has "sudo" tool which helps to administer performing tasks. We can also root the Ubuntu system according to our needs and also most of the network is protected by a built-in security firewall to access all the control over the network, which also supports in full encryption of the disk. Ubuntu has many variant versions which allows us to get the choice of the installation for the specific systems and devices. It is also used in the Ubuntu cloud where we can access all the data of the images. Most of the prime companies use Ubuntu software to store and secure the data's. One of the latest additions to Ubuntu family is Ubuntu open stack which can be accessed by the users alone. Ubuntu over took Linux distribution and was most loved by many developers. And also unlike windows which are paid software Ubuntu is open source and also it has a Ubuntu community for help in case of problem. Loco which is a local community created to develop and promote Ubuntu product to people and encourage students as well. Whereas Linux running based on Ubuntu makes it difficult for hackers to breach one's system and also it has no Spyware which terminates the background running applications.

## CHAPTER 6

### SYSTEM IMPLEMENTATION

#### 6.1 DARKNET INSTALLATION:

Darknet is built from the link “<https://github.com/pjreddie/darknet.git>” repository

```
nagendra@nagendra:~/Desktop/cv_final_new$ git Clone
```

```
https://github.com/pjreddie/darknet.git
```

```
nagendra@nagendra:~/Desktop/cv_final_new$ cd darknet
```

```
nagendra@nagendra:~/Desktop/cv_final_new$ vi Makefile
```

```
nagendra@nagendra:~/Desktop/cv_final_new$ make
```

**Run the Darknet with the command:**

```
./darknet
```

**To check the installation:**

usage: ./darknet <function>

```
nagendra@nagendra:~$ cd Desktop/cv_final_new
nagendra@nagendra:~/Desktop/cv_final_new$ cd ..
nagendra@nagendra:~/Desktop$ mkdir dump
nagendra@nagendra:~/Desktop$ cd dump
nagendra@nagendra:~/Desktop/dump$ clear
nagendra@nagendra:~/Desktop/dump$ git clone https://github.com/pjreddie/darknet.git
Cloning into 'darknet'...
remote: Enumerating objects: 5901, done.
Receiving objects: 83% (4898/5901), 2.55 MiB | 4.95 MiB/s
remote: Total 5901 (delta 0), reused 0 (delta 0), pack-reused 5901
Receiving objects: 100% (5901/5901), 6.16 MiB | 7.59 MiB/s, done.
Resolving deltas: 100% (3915/3915), done.
nagendra@nagendra:~/Desktop/dump$
nagendra@nagendra:~/Desktop/dump$ cd darknet
nagendra@nagendra:~/Desktop/dump/darknet$ make
mkdir -p obj
mkdir -p backup
mkdir -p results
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/gemm.c -o obj/gemm.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/utils.c -o obj/utils.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/cuda.c -o obj/cuda.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/deconvolutional_layer.c -o obj/deconvolutional_layer.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/convolutional_layer.c -o obj/convolutional_layer.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/list.c -o obj/list.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/image.c -o obj/image.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/activations.c -o obj/activations.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/in2col.c -o obj/in2col.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/col2im.c -o obj/col2im.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/blas.c -o obj/blas.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/crop_layer.c -o obj/crop_layer.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/dropout_layer.c -o obj/dropout_layer.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/maxpool_layer.c -o obj/maxpool_layer.o
```



## 6.2 ANNOTATION OF IMAGES:

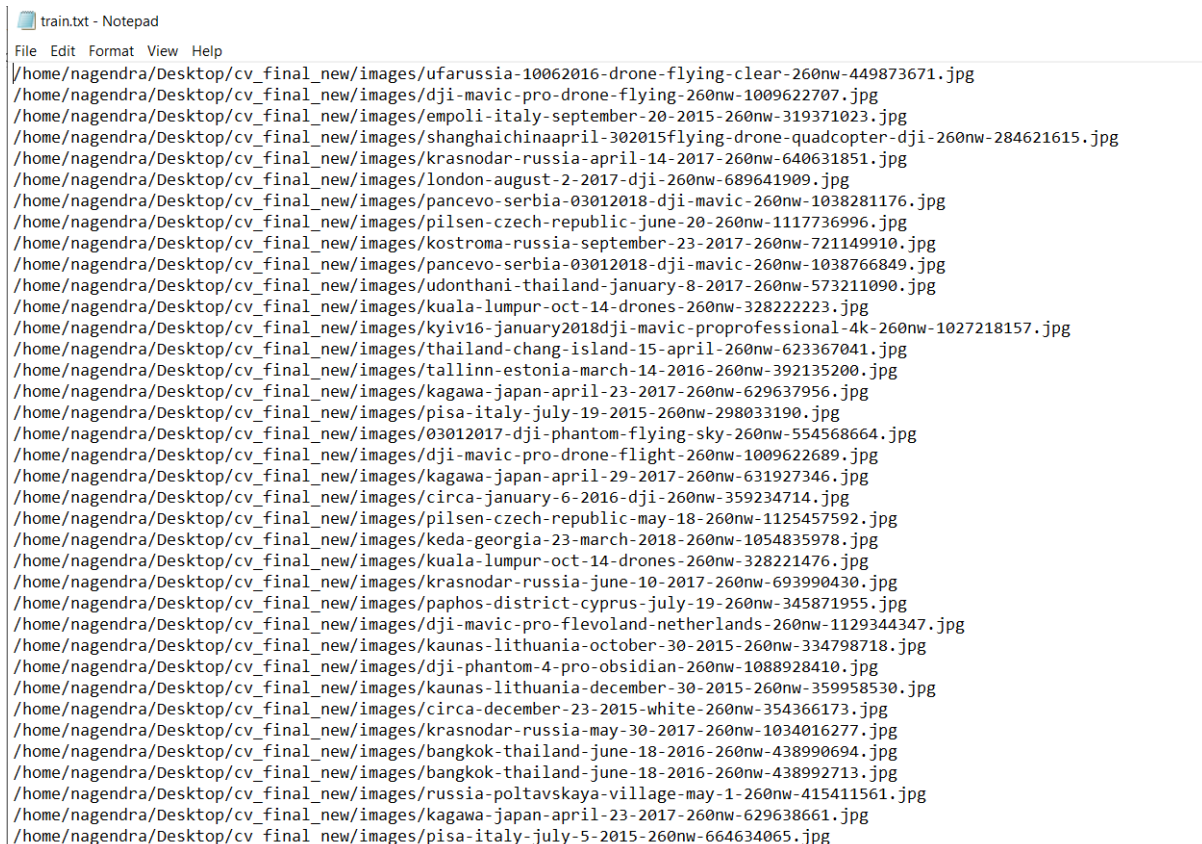
The images in dataset is converted to text files with extension .txt having the following parameters. Object class-it is a integer value and since one class is used the value is 0.

X\_center and Y\_center – These two parameters indicate centre of the bounding Box.

Width-This parameters gives the width of the image.

Height-This parameter gives the height of the image.

### Train.txt:



```
train.txt - Notepad
File Edit Format View Help
/home/nagendra/Desktop/cv_final_new/images/ufarussia-10062016-drone-flying-clear-260nw-449873671.jpg
/home/nagendra/Desktop/cv_final_new/images/dji-mavic-pro-drone-flying-260nw-1009622707.jpg
/home/nagendra/Desktop/cv_final_new/images/empoli-italy-september-20-2015-260nw-319371023.jpg
/home/nagendra/Desktop/cv_final_new/images/shanghaichinaapril-302015flying-drone-quadcopter-dji-260nw-284621615.jpg
/home/nagendra/Desktop/cv_final_new/images/krasnodar-russia-april-14-2017-260nw-640631851.jpg
/home/nagendra/Desktop/cv_final_new/images/london-august-2-2017-dji-260nw-689641909.jpg
/home/nagendra/Desktop/cv_final_new/images/pancevo-serbia-03012018-dji-mavic-260nw-1038281176.jpg
/home/nagendra/Desktop/cv_final_new/images/pilsen-czech-republic-june-20-260nw-1117736996.jpg
/home/nagendra/Desktop/cv_final_new/images/kostroma-russia-september-23-2017-260nw-721149910.jpg
/home/nagendra/Desktop/cv_final_new/images/pancevo-serbia-03012018-dji-mavic-260nw-1038766849.jpg
/home/nagendra/Desktop/cv_final_new/images/udonthani-thailand-january-8-2017-260nw-573211090.jpg
/home/nagendra/Desktop/cv_final_new/images/kuala-lumpur-oct-14-drones-260nw-328222223.jpg
/home/nagendra/Desktop/cv_final_new/images/kyiv16-january2018dji-mavic-proprofessional-4k-260nw-1027218157.jpg
/home/nagendra/Desktop/cv_final_new/images/thailand-chang-island-15-april-260nw-623367041.jpg
/home/nagendra/Desktop/cv_final_new/images/tallinn-estonia-march-14-2016-260nw-392135200.jpg
/home/nagendra/Desktop/cv_final_new/images/kagawa-japan-april-23-2017-260nw-629637956.jpg
/home/nagendra/Desktop/cv_final_new/images/pisa-italy-july-19-2015-260nw-298033190.jpg
/home/nagendra/Desktop/cv_final_new/images/03012017-dji-phantom-flying-sky-260nw-554568664.jpg
/home/nagendra/Desktop/cv_final_new/images/dji-mavic-pro-drone-flight-260nw-1009622689.jpg
/home/nagendra/Desktop/cv_final_new/images/kagawa-japan-april-29-2017-260nw-631927346.jpg
/home/nagendra/Desktop/cv_final_new/images/circa-january-6-2016-dji-260nw-359234714.jpg
/home/nagendra/Desktop/cv_final_new/images/pilsen-czech-republic-may-18-260nw-1125457592.jpg
/home/nagendra/Desktop/cv_final_new/images/keda-georgia-23-march-2018-260nw-1054835978.jpg
/home/nagendra/Desktop/cv_final_new/images/kuala-lumpur-oct-14-drones-260nw-328221476.jpg
/home/nagendra/Desktop/cv_final_new/images/krasnodar-russia-june-10-2017-260nw-693990430.jpg
/home/nagendra/Desktop/cv_final_new/images/paphos-district-cyprus-july-19-260nw-345871955.jpg
/home/nagendra/Desktop/cv_final_new/images/dji-mavic-pro-flevoland-netherlands-260nw-1129344347.jpg
/home/nagendra/Desktop/cv_final_new/images/kaunas-lithuania-october-30-2015-260nw-334798718.jpg
/home/nagendra/Desktop/cv_final_new/images/dji-phantom-4-pro-obsidian-260nw-1088928410.jpg
/home/nagendra/Desktop/cv_final_new/images/kaunas-lithuania-december-30-2015-260nw-359958530.jpg
/home/nagendra/Desktop/cv_final_new/images/circa-december-23-2015-white-260nw-354366173.jpg
/home/nagendra/Desktop/cv_final_new/images/krasnodar-russia-may-30-2017-260nw-1034016277.jpg
/home/nagendra/Desktop/cv_final_new/images/bangkok-thailand-june-18-2016-260nw-438990694.jpg
/home/nagendra/Desktop/cv_final_new/images/bangkok-thailand-june-18-2016-260nw-438992713.jpg
/home/nagendra/Desktop/cv_final_new/images/russia-poltavskaya-village-may-1-260nw-415411561.jpg
/home/nagendra/Desktop/cv_final_new/images/kagawa-japan-april-23-2017-260nw-629638661.jpg
/home/nagendra/Desktop/cv_final_new/images/pisa-italy-july-5-2015-260nw-664634065.jpg
```

## Test.txt:

test.txt - Notepad

File Edit Format View Help

```
/home/nagendra/Desktop/cv_final_new/images/tenerife-spain-13022017-man-remout-260nw-578581357.jpg
/home/nagendra/Desktop/cv_final_new/images/chonburi-thailand-10-march-2017-260nw-599801336.jpg
/home/nagendra/Desktop/cv_final_new/images/moscow-russia-april-1-2017-260nw-617465405.jpg
/home/nagendra/Desktop/cv_final_new/images/krasnodar-russia-june-10-2017-260nw-693990457.jpg
/home/nagendra/Desktop/cv_final_new/images/negeri-sembilan-malaysia-march-9-260nw-1042387021.jpg
/home/nagendra/Desktop/cv_final_new/images/zobnatica-serbia-may-29th-2017-260nw-653298130.jpg
/home/nagendra/Desktop/cv_final_new/images/papar-sabah-my-12-march-260nw-1044012472.jpg
/home/nagendra/Desktop/cv_final_new/images/moscow-russia-24-september-2017-260nw-721160377.jpg
/home/nagendra/Desktop/cv_final_new/images/drone-dji-mavic-air-on-260nw-1050687500.jpg
/home/nagendra/Desktop/cv_final_new/images/kuala-lumpur-malaysia-january-31-260nw-570151768.jpg
/home/nagendra/Desktop/cv_final_new/images/moscow-russia-april-1-2017-260nw-617465549.jpg
/home/nagendra/Desktop/cv_final_new/images/krasnodar-russia-may-30-2017-260nw-1034016271.jpg
/home/nagendra/Desktop/cv_final_new/images/pilsen-czech-republic-july-16-260nw-297579926.jpg
/home/nagendra/Desktop/cv_final_new/images/istanbul-turkey-may-19-2015-260nw-279704978.jpg
/home/nagendra/Desktop/cv_final_new/images/bangkok-thailand-august-14-2016-260nw-468332957.jpg
/home/nagendra/Desktop/cv_final_new/images/pecs-baranya-hungary-march-1-260nw-747462463.jpg
/home/nagendra/Desktop/cv_final_new/images/moers-germany-january-19-2017-260nw-559490962.jpg
/home/nagendra/Desktop/cv_final_new/images/brasil-bernardo-do-campo-abr-260nw-1061324156.jpg
/home/nagendra/Desktop/cv_final_new/images/astana-kazakhstan-september-2-2017-260nw-721055803.jpg
/home/nagendra/Desktop/cv_final_new/images/rostov-region-russia-05202018-professional-260nw-1095689264.jpg
/home/nagendra/Desktop/cv_final_new/images/milan-italy-september-27th-2017-260nw-723135685.jpg
/home/nagendra/Desktop/cv_final_new/images/bangkok-thailand-june-18-2016-260nw-438990670.jpg
/home/nagendra/Desktop/cv_final_new/images/udonthani-thailand-august-14-2016-260nw-470091734.jpg
/home/nagendra/Desktop/cv_final_new/images/pilsen-czech-republic-july-07-260nw-673357708.jpg
/home/nagendra/Desktop/cv_final_new/images/astana-kazakhstan-december-18-2017-260nw-778118074.jpg
/home/nagendra/Desktop/cv_final_new/images/kagawa-japan-april-03-2017-260nw-615757394.jpg
/home/nagendra/Desktop/cv_final_new/images/krasnodar-russia-may-30-2017-260nw-651799615.jpg
/home/nagendra/Desktop/cv_final_new/images/chonburi-thailand-december-7-2017-260nw-786563434.jpg
/home/nagendra/Desktop/cv_final_new/images/bangkok-thailand-may-15-2016-260nw-420618253.jpg
/home/nagendra/Desktop/cv_final_new/images/mavic-pro-dji-on-white-260nw-1135186448.jpg
/home/nagendra/Desktop/cv_final_new/images/pilsen-czech-republic-july-26-260nw-299942393.jpg
/home/nagendra/Desktop/cv_final_new/images/drone-quadcopter-dji-phantom-3-260nw-438110464.jpg
/home/nagendra/Desktop/cv_final_new/images/kagawa-japan-april-23-2017-260nw-629637950.jpg
/home/nagendra/Desktop/cv_final_new/images/nakornpratom-thailand-nov-17-editorial-260nw-518837116.jpg
/home/nagendra/Desktop/cv_final_new/images/kagawa-japan-april-29-2017-260nw-631926491.jpg
/home/nagendra/Desktop/cv_final_new/images/kiulu-sabah-malaysia-sep-2-260nw-709281637.jpg
/home/nagendra/Desktop/cv_final_new/images/silhouette-man-web-author-taking-260nw-1087557533.jpg
```


The format of the text file will be as follows:



09-february-2017-ufa-russia-260nw-1062076784.txt - Notepad

File Edit Format View Help

```
0 199.5 145.0 313.0 180.0
```

 18th-july-2017-kuantanpahangmalaysia-dji-260nw-681114880.txt - Notepad

File Edit Format View Help

0 137.0 173.0 206.0 98.0

For train the model, three files are created :

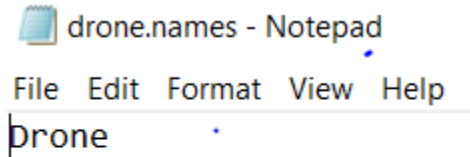
### 1. **drone.data;**

This file generally defines the classes for the object to be trained and also specifies the location of the training and validation files and the file that contains the categories for detection.

```
classes = 1
train = /home/nagendra/Desktop/cv_final_new/train.txt
valid = /home/nagendra/Desktop/cv_final_new/test.txt
names = /home/nagendra/Desktop/cv_final_new/drone.names
backup = backup/
```

### 2. **drone.names:**

This is a simple file the contains the names of the object that corresponds to the object class id that has been created.



### 3. yolo\_drone.cfg:

Parameters of the neural network are defined in this file.

```
[convolutional]
size=1
stride=1
pad=1
filters=18
activation=linear

[yolo]
mask = 1,2,3
anchors = 10,14, 23,27, 37,58, 81,82, 135,169, 344,319
classes=1
num=6
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1
```

In the model created, final layer is the Yolo layer that gives the final output from the 18 filter processed layers from the convolutional layers.

## 6.3 TRAINING AND TESTING:

### Sample code for splitting the training and testing data:

```
import glob

import os

current_dir = os.getcwd() + '/images'

# Percentage of images to be used for the test set

percentage_test = 10

# Create and/or truncate train.txt and test.txt

file_train = open('train.txt', 'w')

file_test = open('test.txt', 'w')

# Populate train.txt and test.txt

counter = 1

index_test = round(100 / percentage_test)

for pathAndFilename in glob.iglob(os.path.join(current_dir, "*.jpg")):

    title, ext = os.path.splitext(os.path.basename(pathAndFilename))

    if counter == index_test:

        counter = 1
```

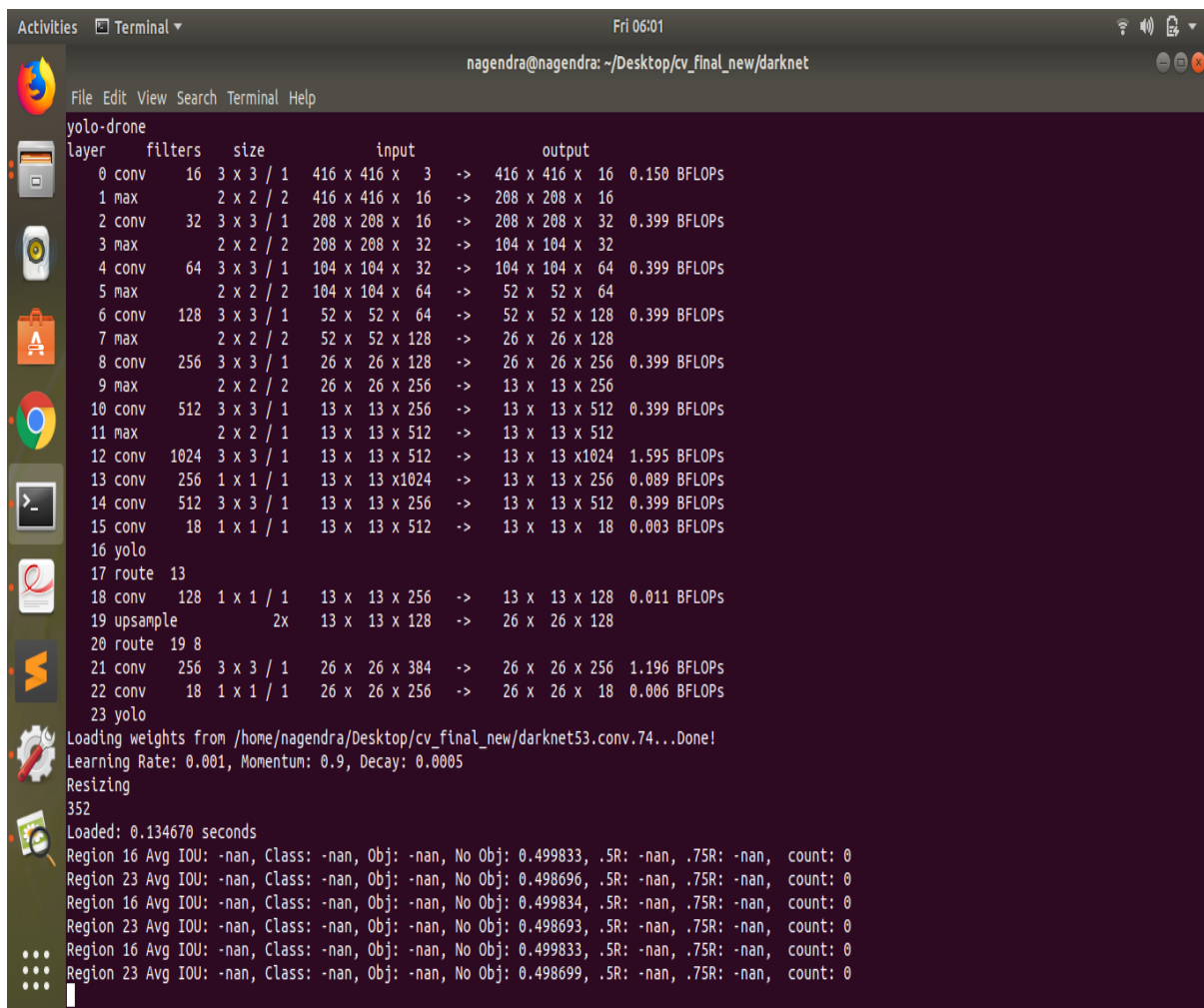
```
file_test.write(current_dir + "/" + title + '.jpg' + "\n")
```

else:

```
file_train.write(current_dir + "/" + title + '.jpg' + "\n")
```

```
counter = counter + 1
```

## TRAINING:



```
Activities Terminal Fri 06:01
nagendra@nagendra: ~/Desktop/cv_final_new/darknet

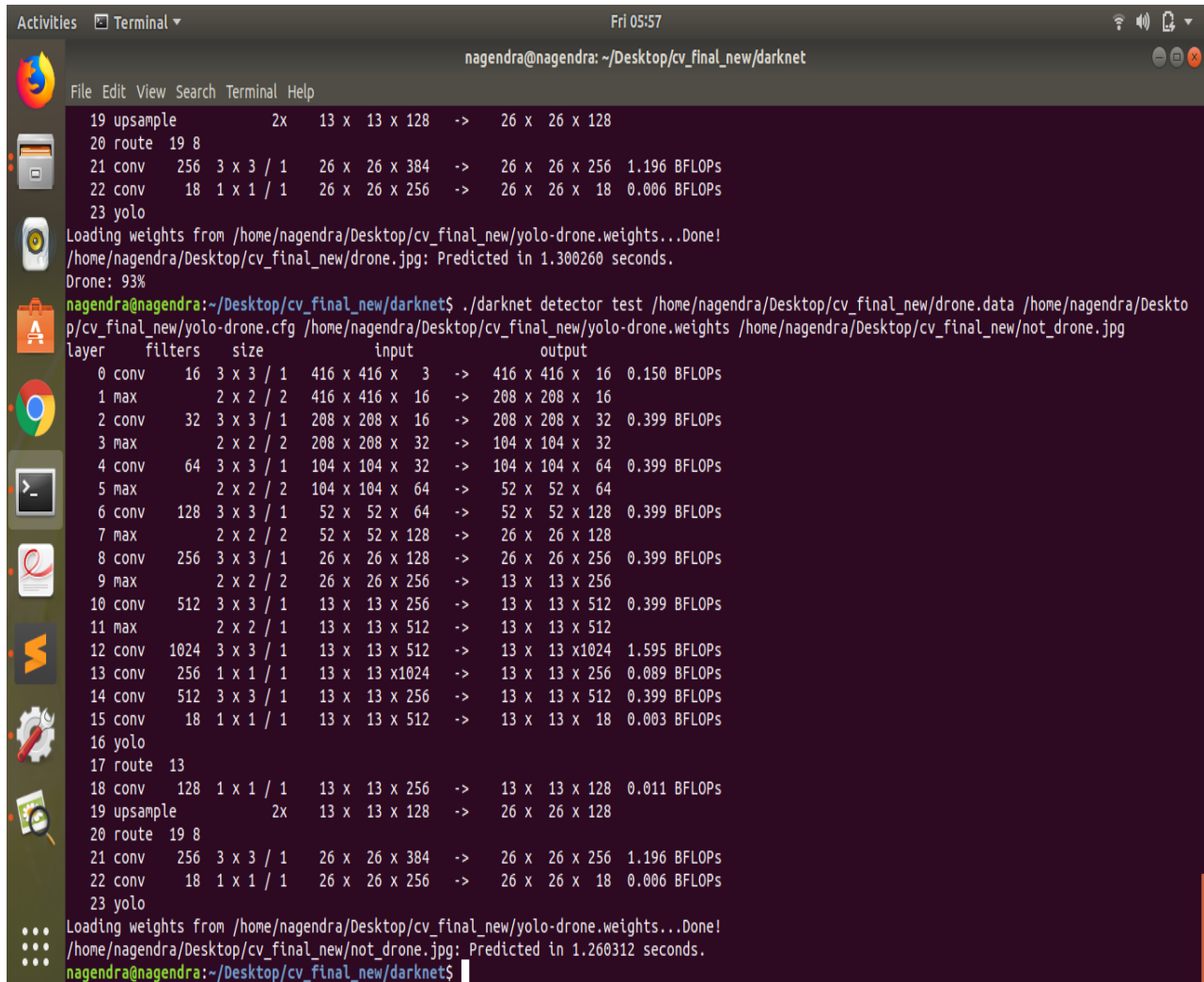
File Edit View Search Terminal Help

yolo-drone
layer  filters  size      input      output
0 conv   16  3 x 3 / 1  416 x 416 x 3  -> 416 x 416 x 16  0.150 BFLOPs
1 max    2  2 x 2 / 2  416 x 416 x 16  -> 208 x 208 x 16
2 conv   32  3 x 3 / 1  208 x 208 x 16  -> 208 x 208 x 32  0.399 BFLOPs
3 max    2  2 x 2 / 2  208 x 208 x 32  -> 104 x 104 x 32
4 conv   64  3 x 3 / 1  104 x 104 x 32  -> 104 x 104 x 64  0.399 BFLOPs
5 max    2  2 x 2 / 2  104 x 104 x 64  -> 52 x 52 x 64
6 conv  128  3 x 3 / 1   52 x 52 x 64  -> 52 x 52 x 128  0.399 BFLOPs
7 max    2  2 x 2 / 2   52 x 52 x 128  -> 26 x 26 x 128
8 conv  256  3 x 3 / 1   26 x 26 x 128  -> 26 x 26 x 256  0.399 BFLOPs
9 max    2  2 x 2 / 2   26 x 26 x 256  -> 13 x 13 x 256
10 conv  512  3 x 3 / 1   13 x 13 x 256  -> 13 x 13 x 512  0.399 BFLOPs
11 max    2  2 x 2 / 1   13 x 13 x 512  -> 13 x 13 x 512
12 conv 1024  3 x 3 / 1   13 x 13 x 512  -> 13 x 13 x1024  1.595 BFLOPs
13 conv  256  1 x 1 / 1   13 x 13 x1024  -> 13 x 13 x 256  0.089 BFLOPs
14 conv  512  3 x 3 / 1   13 x 13 x 256  -> 13 x 13 x 512  0.399 BFLOPs
15 conv   18  1 x 1 / 1   13 x 13 x 512  -> 13 x 13 x 18   0.003 BFLOPs
16 yolo
17 route 13
18 conv  128  1 x 1 / 1   13 x 13 x 256  -> 13 x 13 x 128  0.011 BFLOPs
19 upsample 2x 13 x 13 x 128 -> 26 x 26 x 128
20 route 19 8
21 conv  256  3 x 3 / 1   26 x 26 x 384  -> 26 x 26 x 256  1.196 BFLOPs
22 conv   18  1 x 1 / 1   26 x 26 x 256  -> 26 x 26 x 18   0.006 BFLOPs
23 yolo

Loading weights from /home/nagendra/Desktop/cv_final_new/darknet53.conv.74...Done!
Learning Rate: 0.001, Momentum: 0.9, Decay: 0.0005
Resizing
352
Loaded: 0.134670 seconds
Region 16 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.499833, .5R: -nan, .75R: -nan, count: 0
Region 23 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.498696, .5R: -nan, .75R: -nan, count: 0
Region 16 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.499834, .5R: -nan, .75R: -nan, count: 0
Region 23 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.498693, .5R: -nan, .75R: -nan, count: 0
Region 16 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.499833, .5R: -nan, .75R: -nan, count: 0
Region 23 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.498699, .5R: -nan, .75R: -nan, count: 0
```

The model is trained as mentioned above. The weights begin to store after every 1000 iterations until 10000 ten thousands and then the weights will be stored for every ten thousand iterations.

## TESTING:



```
nagendra@nagendra: ~/Desktop/cv_final_new/darknet
File Edit View Search Terminal Help
19 upsample      2x   13 x 13 x 128  ->  26 x 26 x 128
20 route 19 8
21 conv 256 3 x 3 / 1  26 x 26 x 384  ->  26 x 26 x 256  1.196 BFLOPs
22 conv 18 1 x 1 / 1  26 x 26 x 256  ->  26 x 26 x 18   0.006 BFLOPs
23 yolo
Loading weights from /home/nagendra/Desktop/cv_final_new/yolo-drone.weights...Done!
/home/nagendra/Desktop/cv_final_new/drone.jpg: Predicted in 1.300260 seconds.
Drone: 93%
nagendra@nagendra:~/Desktop/cv_final_new/darknet$ ./darknet detector test /home/nagendra/Desktop/cv_final_new/drone.data /home/nagendra/Desktop/cv_final_new/yolo-drone.cfg /home/nagendra/Desktop/cv_final_new/yolo-drone.weights /home/nagendra/Desktop/cv_final_new/not_drone.jpg
layer  filters  size  input              output              BFLOPs
0 conv 16 3 x 3 / 1  416 x 416 x 3  ->  416 x 416 x 16  0.150 BFLOPs
1 max 2 x 2 / 2  416 x 416 x 16  ->  208 x 208 x 16
2 conv 32 3 x 3 / 1  208 x 208 x 16  ->  208 x 208 x 32  0.399 BFLOPs
3 max 2 x 2 / 2  208 x 208 x 32  ->  104 x 104 x 32
4 conv 64 3 x 3 / 1  104 x 104 x 32  ->  104 x 104 x 64  0.399 BFLOPs
5 max 2 x 2 / 2  104 x 104 x 64  ->  52 x 52 x 64
6 conv 128 3 x 3 / 1  52 x 52 x 64  ->  52 x 52 x 128  0.399 BFLOPs
7 max 2 x 2 / 2  52 x 52 x 128  ->  26 x 26 x 128
8 conv 256 3 x 3 / 1  26 x 26 x 128  ->  26 x 26 x 256  0.399 BFLOPs
9 max 2 x 2 / 2  26 x 26 x 256  ->  13 x 13 x 256
10 conv 512 3 x 3 / 1  13 x 13 x 256  ->  13 x 13 x 512  0.399 BFLOPs
11 max 2 x 2 / 1  13 x 13 x 512  ->  13 x 13 x 512
12 conv 1024 3 x 3 / 1  13 x 13 x 512  ->  13 x 13 x 1024  1.595 BFLOPs
13 conv 256 1 x 1 / 1  13 x 13 x 1024  ->  13 x 13 x 256  0.089 BFLOPs
14 conv 512 3 x 3 / 1  13 x 13 x 256  ->  13 x 13 x 512  0.399 BFLOPs
15 conv 18 1 x 1 / 1  13 x 13 x 512  ->  13 x 13 x 18   0.003 BFLOPs
16 yolo
17 route 13
18 conv 128 1 x 1 / 1  13 x 13 x 256  ->  13 x 13 x 128  0.011 BFLOPs
19 upsample      2x   13 x 13 x 128  ->  26 x 26 x 128
20 route 19 8
21 conv 256 3 x 3 / 1  26 x 26 x 384  ->  26 x 26 x 256  1.196 BFLOPs
22 conv 18 1 x 1 / 1  26 x 26 x 256  ->  26 x 26 x 18   0.006 BFLOPs
23 yolo
Loading weights from /home/nagendra/Desktop/cv_final_new/yolo-drone.weights...Done!
/home/nagendra/Desktop/cv_final_new/not_drone.jpg: Predicted in 1.260312 seconds.
nagendra@nagendra:~/Desktop/cv_final_new/darknet$
```

When the model is tested, if the image has the UAV then the prediction rate is calculated. The accuracy of the prediction is also displayed. If there is no UAV in the image, then there will be no bounding box and the process is done. The output image of the prediction is stored in the “darknet\predictions.jpg”

## **CFG FILE:**

[net]

# Testing

batch=24

subdivisions=8

# Training

# batch=64

# subdivisions=2

width=416

height=416

channels=3

momentum=0.9

decay=0.0005

angle=0

saturation = 1.5

exposure = 1.5

hue=.1



learning\_rate=0.001

burn\_in=1000

max\_batches = 500200

policy=steps

steps=400000,450000

scales=.1,.1

[convolutional]

batch\_normalize=1

filters=16

size=3

stride=1

pad=1

activation=leaky

[maxpool]

size=2

stride=2

[convolutional]

batch\_normalize=1

filters=32

size=3

stride=1

pad=1

activation=leaky

[maxpool]

size=2

stride=2

[convolutional]

batch\_normalize=1

filters=64

size=3

stride=1

pad=1

activation=leaky

[maxpool]

size=2

stride=2

[convolutional]

batch\_normalize=1

filters=128

size=3

stride=1

pad=1

activation=leaky

[maxpool]

size=2

stride=2

[convolutional]

batch\_normalize=1

filters=256

size=3

stride=1

pad=1

activation=leaky

[maxpool]

size=2

stride=2

[convolutional]

batch\_normalize=1

filters=512

size=3

stride=1

pad=1

activation=leaky

[maxpool]

size=2

stride=1

[convolutional]

batch\_normalize=1

filters=1024

size=3

stride=1

pad=1

activation=leaky

#####

[convolutional]

batch\_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=512

size=3

stride=1

pad=1

activation=leaky

[convolutional]

size=1

stride=1

pad=1

filters=18

activation=linear

[yolo]

mask = 3,4,5

anchors = 10,14, 23,27, 37,58, 81,82, 135,169, 344,319

classes=1

num=6

jitter=.3

ignore\_thresh = .7

truth\_thresh = 1

random=1

[route]

layers = -4

[convolutional]

batch\_normalize=1

filters=128

size=1

stride=1

pad=1

activation=leaky

[upsample]

stride=2

[route]

layers = -1, 8

[convolutional]

batch\_normalize=1

filters=256

size=3

stride=1



pad=1

activation=leaky

[convolutional]

size=1

stride=1

pad=1

filters=18

activation=linear

[yolo]

mask = 1,2,3

anchors = 10,14, 23,27, 37,58, 81,82, 135,169, 344,319

classes=1

num=6

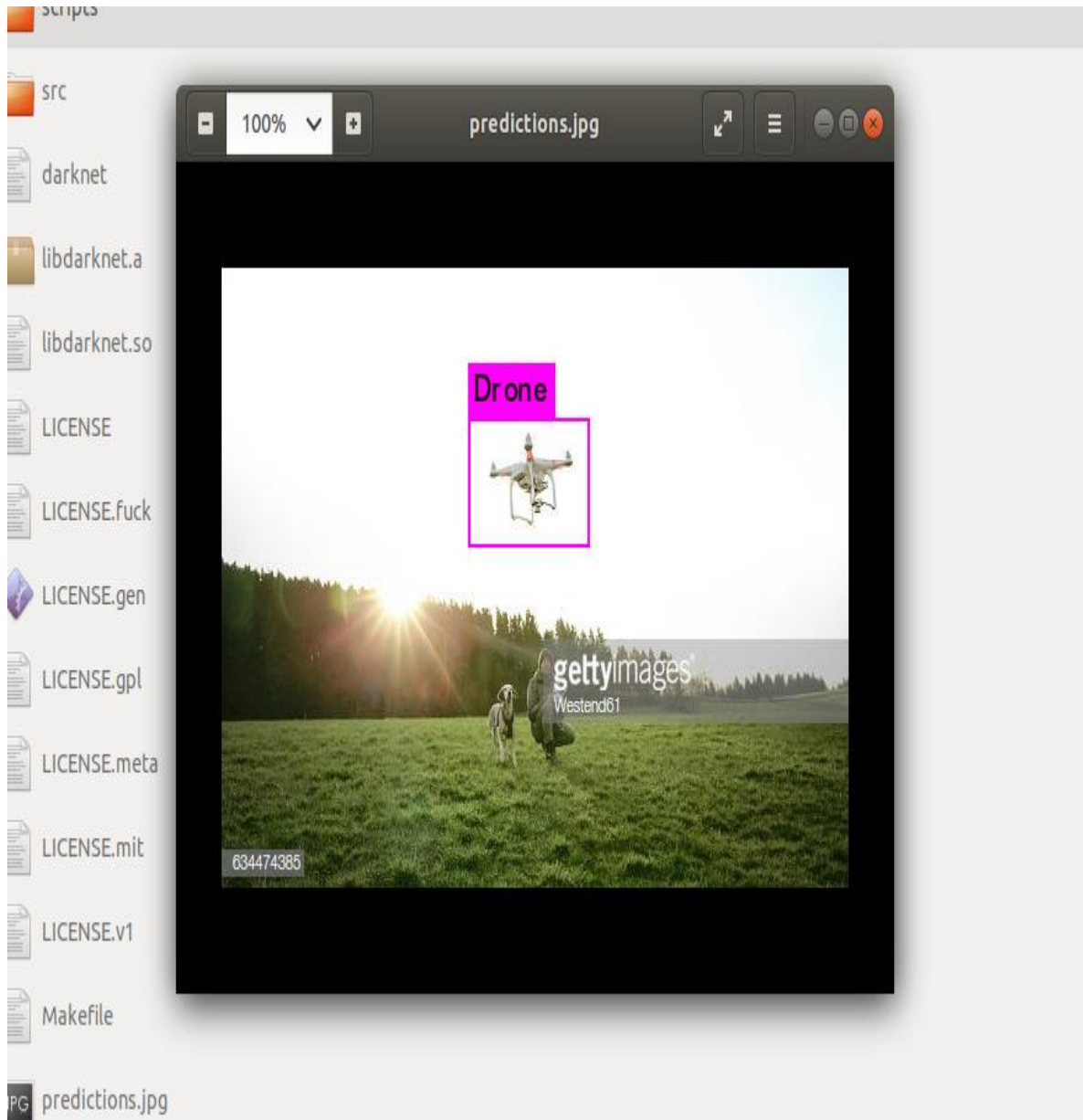
jitter=.3

ignore\_thresh = .7

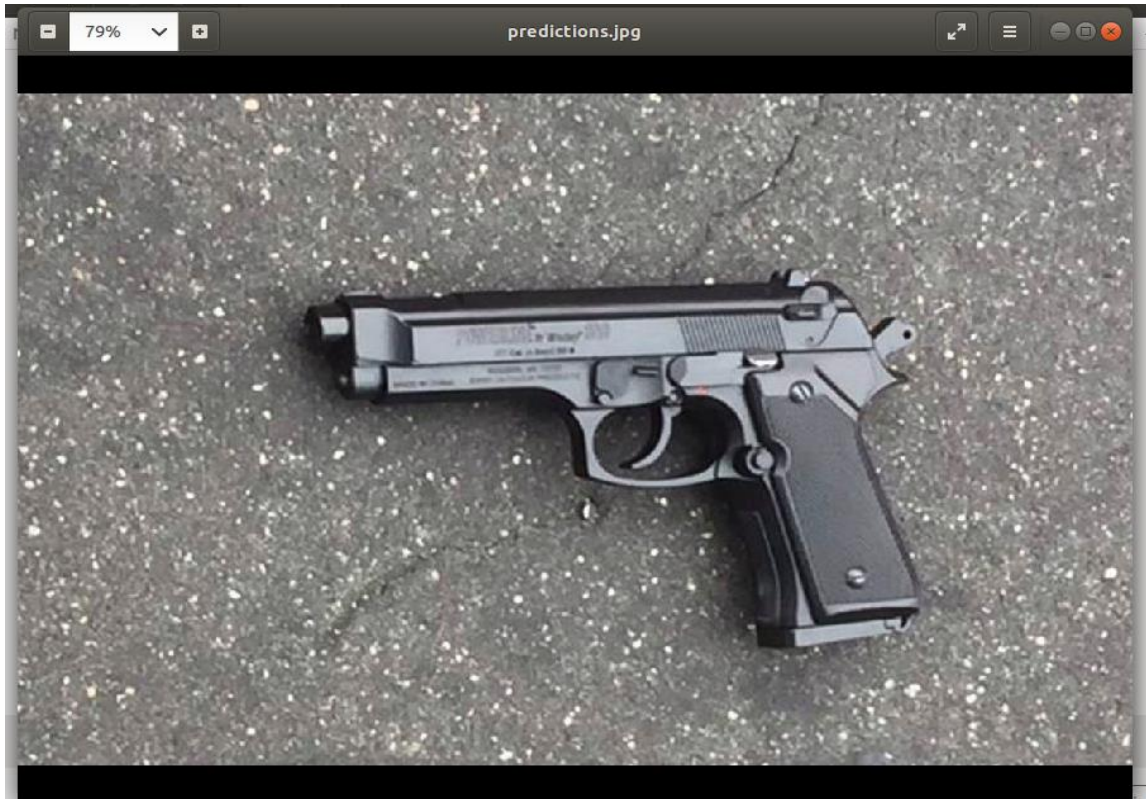
truth\_thresh = 1

random=1

## 6.4 OUTPUT SCREENSHOTS



The output is stored in the file with the bounding box along with its class name.



Here the system didn't detect any UAV hence there is no bounding box.

## **CHAPTER 7**

### **COMPARISON**

The various techniques used for the detection of objects is compared below

ACF generates many features by using obtained channels. By combining two or more first-order channel features, ACF extracts First-order channel features from a single channel by summing pixels and higher-order channel features. It uses decision tree at the end for classification. The efficiency of this algorithm is on the lower side due to plethora of operations involved. Region-Based Convolutional Neural Network (RCNN) is an object detector based on Convolutional Neural Network (CNN). Regional proposals with more than 0.5 IoU (Intersection over Union) overlap with ground truth (defined by a user) are categorised as positive and the rest of the proposals are categorised as negative. It has 3 convolutional layers and 2 full connected layers. The drawback of using RCNN is it will be slow because it performs a CNN for each proposal, without performing the computation sharing

Fast RCNN shares computation of convolutional layers among different proposals. The whole image can be computed through the convolutional layer once and the processing time is saved. It is almost 10 times faster than the RCNN technique, but the drawback of this method is it cannot be applied for real time problems. Faster RCNN uses a region proposal network (RPN) after the last convolutional layer. RPN takes an image feature map as input and outputs a set of rectangular object proposals. Faster RCNN is 10 times faster than Fast RCNN and it can be applied to real time problems.

We choose YOLO V3 because it is a single step regression process so the objects can be detected at fast pace. Therefore, the speed of the detection eventually gets increased. The average precision for small sized objects is improved. The localization errors got decreased

## **CHAPTER 8**

### **DISCUSSION**

Using YOLO V3 we achieved 93 percent of accuracy and the prediction time is comparatively less. The model was trained with various learning rates and the momentum of 0.9, decay of 0.005. The best results were achieved when the learning rate was kept to 0.001. The model can also be trained for prediction by customizing the convolutional layers based on the requirements. The model developed was compared with other applications and we observed the improvements in speed and accuracy of the system. From the noted accuracy and the speed of the detection we come to know that the proposed algorithm has achieved higher precision rate with increased efficiency.

## **CHAPTER 9**

### **CONCLUSION AND FUTURE WORK**

In the above developed application, the UAVs are detected from the system. The system predicts the UAVs with a prediction rate varying between one to two seconds. But as of now system predicts the UAVs in the given image but not for live feed. The accuracy of the developed system is about 92%.The model will increases the security in various places such as military bases etc. Accuracy rate of model is affected since model has been trained with limited number of images due to insufficient hardware. The model can predict in a more precise manner if the system is trained with a large dataset and if the more parameters are included. Another drawback to be noticed is that the inappropriate prediction of UAVs and weapons due to Bad weather. Despite all these errors, the system predicts the Objects in an increased speed and improved accuracy compared to other methods.

In the field of security and surveillance, the application developed can be integrated with detection in the live surveillance feed. Though the algorithm speed has been increased, precise results are needed to be predicted in confidential places to avoid false positives. Hence the accuracy should be increased. For medium and large sized objects, the average precision can be improvised. This system can also be developed for detecting weapons in UAVs to avoid collateral damage and casualties in terms of crisis.

## REFERENCES

- [1] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi,” You Only Look Once: Unified, Real-Time Object Detection” 9 May 2016
- [2] Ajeet Ram Pathak, “Application of Deep Learning for Object Detection”, Procedia Computer Science· January 2018.
- [3] Juan Wu, Bo Peng, Zhenxiang Huang, Jietao Xie,” Research on Computer Vision-Based Object Detection and Classification”, CCTA, October 2012.
- [4] Joseph Redmon Ali Farhadi,” YOLOv3: An Incremental Improvement”, 8 April 2018
- [5] Xie Weige, Zhiguo Shi, Xiufang Shi,” Real-Time Drone Detection Using Deep Learning Approach”, Third International Conference, MLICOM 2018, Hangzhou, China, July 6-8, 2018