YOLO v3-Tiny: An improved One Stage Model for Detection and Recognition of Objects

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTER OF TECHNOLOGY
IN
SOFTWARE ENGINEERING

Submitted by:
Pranav Adarsh
(Roll No. 2K18/SWE/11)

Under the supervision of
Dr. Manoj Kumar
(Associate Professor)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi – 110042

JULY, 2020

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi – 110042

CANDIDATE'S DECLARATION

I, Pranav Adarsh, Roll No. 2K18/SWE/11 student of M.Tech (Software Engineering), hereby declare that the project Dissertation titled **"YOLO v3-Tiny: An improved One Stage Model for Detection and Recognition of Objects"** which is submitted by me to the Department of Computer Science and Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Pranav Adarsh

Place: Delhi **Pranav Adarsh**

Date: 26/06/2020 (2K18/SWE/11)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi – 110042

<u>CERTIFICATE</u>

I hereby certify that the Project Dissertation titled **"YOLO v3-Tiny: An improved One Stage Model for Detection and Recognition of Objects"** which is submitted by Pranav Adarsh, 2K18/SWE/11 Department of Computer Science & Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Technology, is a record of the project work carried out by the students under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

[illegible]

Place: Delhi **Dr. Manoj Kumar**

Date: 27/06/2020 **SUPERVISOR**

(Associate Professor)

Department of Computer Science & Engineering

Delhi Technological University

ACKNOWLEDGEMENT

The success of the Major II project requires help and contribution from numerous individuals and the organization. Writing the report of this project work gives me an opportunity to express my gratitude to everyone who has helped in shaping up the outcome of the project.

I express my heartfelt gratitude to my project guide **Dr. Manoj Kumar** for giving me an opportunity to do my project work under his guidance. I am deeply indebted to him for the support, advice and encouragement he provided without which the project could not have been a success. His constant support and encouragement has made me realize that it is the process of learning which weighs more than the end result.

I would like to express my gratitude to the university and staff for providing us with the library, laboratories, infrastructure, testing facilities, right academic resources and environment which allowed us to work without any obstructions.

I also reveal my thanks to all my classmates and friends for constant support. They helped me throughout by giving new ideas, providing necessary information and pushing me forward to complete the work. I would also like to express sincere gratitude to my parents for constantly encouraging me during the completion of work.

Pranav Adarsh

Pranav Adarsh

Roll No – 2K18/SWE/11

M. Tech (Software Engineering)

Delhi Technological University

ABSTRACT

Object detection has seen many changes in algorithms to improve performance both on speed and accuracy. By the continuous effort of so many researchers, deep learning algorithms are growing rapidly with an improved object detection performance. Various popular applications like pedestrian detection, medical imaging, robotics, self-driving cars, face detection, etc. reduces the efforts of humans in many areas. Due to the vast field and various state-of-the-art algorithms, it is a tedious task to cover all at once. This paper presents the fundamental overview of object detection methods by including two classes of object detectors. In two stage detector covered algorithms are RCNN, Fast RCNN, and Faster RCNN, whereas in one stage detector YOLO v1, v2, v3, and SSD are covered. Two stage detectors focus more on accuracy, whereas the primary concern of one stage detectors is speed. We will explain an improved YOLO version called YOLO v3-Tiny, and then its comparison with previous methods for detection and recognition of object is described graphically.

Keywords: Computer vision; YOLO v3; Faster RCNN; Deep learning; YOLO v3-Tiny; Object detection; image processing; Convolutional Neural Networks.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

List of Abbreviations and Nomenclature

1. CNN- Convolutional Neural Network
2. RCNN- Region based Convolutional Neural Network
3. FAST RCNN- Fast Region based Convolutional Neural Network
4. FASTER RCNN- Faster Region based Convolutional Neural Network
5. RPN- Region Proposal Networks
6. RRPN- Radar Region Proposal Networks
7. HOG- Histogram of Oriented Gradients
8. YOLO v1- You Only Look Once version one
9. YOLO v2- You Only Look Once version two
10. YOLO v3- You Only Look Once version three
11. SSD- Single Shot Detectors
12. GPU- Graphics Processing Unit
13. CPU- Central Processing Unit
14. RAM- Random Access Memory
15. IDE- Integrated Development Environment

CHAPTER 1
INTRODUCTION

1.1 Overview

In the recent few years, diverse research work happened to develop a practical approach to accelerate the development of deep learning methods. Numerous developments accomplished excellent results and followed by continuous reformations in deep learning procedures. Object localization is the identification of all the visuals in a photograph, incorporating the precise location of those visuals. By using deep learning techniques [1] [2] for object identification and localization, computer vision has reached a new zenith. Due to significant inconstancies in viewpoints, postures, dimensions, and lighting positions, it is challenging to succeed in the identification of objects perfectly. Accordingly, considerable concern has been given by researchers to this area in the past few years.

There are two types of object detection algorithms. Object detection algorithms using region proposal includes RCNN [3], Fast RCNN [4], and Faster RCNN [5], etc. These techniques create region proposal networks (RPN) [6], and then the region proposals are divided into categories afterward. On the other side, object detection algorithms using regression includes SSD [7] and YOLO [8], etc. These methods also generate region proposal networks (RPN) but divide these region proposals into categories at the moment of generation.

All of the procedures mentioned above have significant accomplishments in object localization and recognition. YOLO consolidates labels in diverse datasets to form a tree-like arrangement, but the merged labels are not reciprocally exclusive. YOLO9000 [9] enhances YOLO to recognize targets above 9000 categories employing hierarchical arrangement. Whereas YOLOv3 [10] uses multilabel classification, it replaces the approach of estimating the cost function and further exhibits meaningful improvement in distinguishing small targets.

1.1.1 Classification of Objects

If we need to recognize the object in an image, then it is termed as Objects-classification. The core principle of object classification is to categorize the objects into different categories and to predict the class of object present inside the image. Let's take an example; a picture has two objects which are bicycle and bike. When we classify them, then it is called a single-label classification. But if at once we need to classify between many objects like dogs, cats, bikes, bicycles etc. then we use multi-label classifier for their categorization over different classes by giving labels to them.

By looking at size, shape, structure and other similar properties of objects, class labels are decided as x, y, z etc. where each character represents a different object class. Image size matters a lot in classifying the objects as if image size is larger, then objects can easily be recognized. But if the size is smaller, then it becomes difficult to identify them during classification. Resolution of the image also has a vital role in determining the class of objects.

1.1.2 Localization of Objects

In any image to find the actual position of the objects by drawing a square box around them is called Objects-Localization. For finding the location of the objects in a picture, we take four dimensions. These dimensions are image_top_left_corner, image_top_right_corner, the height of the object in the image and width of the object in the image. The main problem is to know the size of the object because based on the size of the object, the size of the bounding box is decided. Because the posture of any human is different, i.e., running, sitting, sleeping etc. whereas the object is same so it should be appropriately localized. So the solution is to find the aspect ratio of the objects within the image by decreasing the size of the image until the minimum size of the object is not covered. Another problem is of object overlap when any two or more targets get over one another then by using anchor box we try to find the actual location of objects within the image.

1.1.3 Object Detection

It is a process of combining classification and localization to find location and class of objects. These algorithms are used in many day to day activity to make human life comfortable. Whether it is to differentiate between two objects or observe the objects in the video, all comes under object detection. Firstly model is trained based on the different categories of objects, and after that, it is being tested on real-time objects. There are many algorithms some are good in accuracy whereas others are good in speed with each having their own advantages so to pick any one of them depends upon the need of work. For getting an overall understanding of algorithms, these are divided into two types of Detectors based on the number of stages in the evaluation of results.

1.1.4 Two Stage Detectors

When accuracy is much needed than speed, then two stage detectors are very useful. They have good localization and detection accuracy in comparison to one stage detectors. The first stage of two stage detectors is using region-of-interest pooling layers which helps to identify the regions and apply bounding boxes to those regions. i.e. Region-proposal-network is used in Faster-RCNN. The second stage of two stage detectors is used to classify the objects which were identified in the first stage.

Let’s take an input image which has four different objects then two stage detectors will work through the following steps.

Step 1. Using any external algorithms location of the objects is to be identified by making the bounding box around them.

Step 2. In the bounding box, it should be observed that all four dimensions fully cover the object or it may be needed to apply aspect ratio to cover all dimensions.

Step 3. Object class is determined and based on which it is classified for detection and provide accuracy.

Many algorithms come under two stage detectors, but the most famous is region based detectors, and due to many advancements, it has improved a lot. It mainly follows the following four procedures.

1). Categorize regions based on different categories.

2). Vector is used, which is of fixed length and performs based on categorized regions.

3). Linear SVM based on specific classification.

4). For detecting the location of the object, bounding box regressor is applied.

1.1.5 One Stage Detectors

In this type of detectors, there is no region proposal step, unlike two stage detectors. One stage detectors are very fast in comparison to two stage detectors because they don't use a sliding window approach or region based approach. In a single pass, it predicts all objects classification score and by dividing the image into small grids inserts bounding boxes into that. YOLO family is the example of one stage detectors which divides the image into many regions before processing by which network is capable of calculating the probability class score with bounding-box dimensions of the object. Yolo-v2 uses four less anchors than Yolo-v3 anchors whereas no anchor concept was there in Yolo-v1. YOLO versions have seen so many changes in its design to improve the performance.

1). Normalization of batch improves the performance by two percent than its previous version.

2). Two times larger classifier improves the performance by four times than earlier.

3). Direct prediction of object location makes it faster and more efficient than two stage detectors.

4). Anchor boxes make it easier to find two or more objects overlap in the image, but due to it, a bit decrease in accuracy is faced.

If the image contains many small objects whose accuracy is not good, then one stage can improve the accuracy by using different scale iteration in one network training. Let an image with three object sizes a, b, c is calculated in the beginning with the initial size of the network.

Still, after a few iterations, it is observed that the network is not covering the entire object so the network will reduce its initial size and will do the same until it is not covering all objects.

1.2 Scope of the work

Human eyes can easily differentiate between any two different objects. If many objects, some with similarities and others with dissimilarities, are placed together, and we need to distinguish between those objects, then it becomes a tedious task for us to do. We can say that if there are so many objects, then the job of categorizing them in different categories and providing label of that category is difficult. So, for that, the role of machine learning for object detection and recognition makes this task easy for us. There are many significant applications based on Object detection and identification, and a few of them are described as following.

- Face detection- When we purchase any latest mobile, it mostly comes with the feature of a face security lock, and it is used by police to find criminals by matching crime scene photos to the history of criminal records. It is helpful to catch intruders who want to threaten the security of our personal information.

- Pedestrian detection- The self-driving car needs pedestrian detection to detect the safe boundary and danger zones, but it faces many challenges in crowd zones.

- Text detection- Whenever we visit any new place, they have different languages and understanding them without knowing that language is impossible or quite complicated. Therefore, many apps are available to convert the text in other languages to the text in our desired language.

Scientists have developed so many algorithms to improve accuracy and speed, but it was always difficult for them to achieve both simultaneously. So by the time, they have changed a lot in their algorithms to make them better than previous methods. But still, scientists are facing many challenges. For example, to detect humans, we have to train the model as it should consider different shapes, sizes, and colors of clothes and humans.

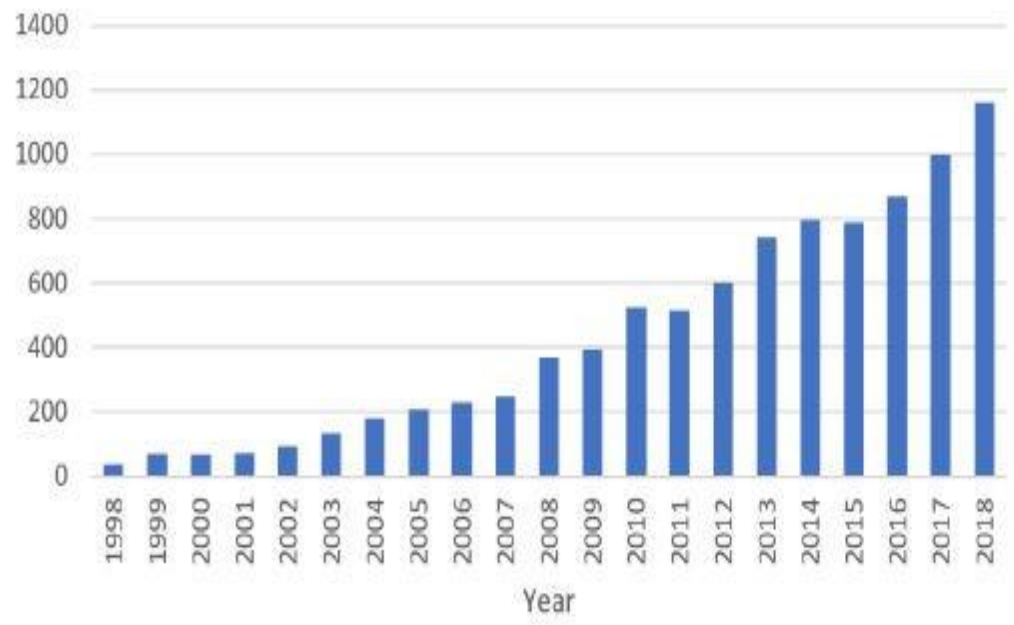

Fig.1.1: The Increasing Number of publications in Object detection

1.3 Problem Statement

There are many developments in machine learning tools that are used to develop and implement projects. Similarly, much progress is there in the approach to optimize results in less time. For example, after applying face detection on the crime scene, if our algorithm will take too much time to catch criminal or too late alert cars about danger zone on the road, so it will lose its usefulness.

In this project, we need to thoroughly examine the characteristics of three procedures for object detection, which are faster RCNN, YOLO v3, and SSD. If the requirement is accuracy, then faster RCNN is the best as it is much accurate compared to SSD and YOLO v3. If accuracy is not the primary concern and we want super-fast speed, then YOLO v3 takes less time than SSD and faster RCNN. But if at the same time, the requirement is of excellent accuracy and less running time, then SSD is a more favorable recommendation, as its speed is better than faster RCNN, and its accuracy is better compared to YOLO v3.

It can be observed that there is a possibility of further improvement in the YOLO model, as we have tested the model for different dimensions. It has been noted that results are getting changed and give different accuracy and execution time for different sizes of images or different resolutions. It means that if we apply the trained model on a good resolution photograph, then it will provide us the best result with much accuracy. We can improve the model to produce better results in less time.

1.4 Purpose of the project

The purpose of given research project work is to study all the existing significant works that have already been done in this field of object detection and recognition. We will analyze the accuracy of results and running time of various object detection and identification approaches. We will demonstrate this analysis graphically. We will examine the categorization of object detectors based on their working and categorize them into two categories. We are implementing YOLO v3 and YOLO v3-tiny to conclude the better and more favorable change in results. We are training our model by changing the dimension of photos and observing the results of the existing algorithms. Here, we are trying to improve exiting algorithm results to decrease the trade-off between speed and accuracy of these methods. After that, we will analyze the proposed YOLO v3-Tiny mechanism and compare its results and running time with other previous approaches. The final intention of this project is to obtain much accuracy and better speed of execution simultaneously.

1.5 Organization of Dissertation

The structure of the dissertation is as follows. In chapter 1, the scope and motive of this project work are explained in detail. Below in chapter 2, we present background information of object detectors. It elaborates the most representative and pioneering two-stage object detection methods with their significant contributions in object detection. First, we examine their methodologies and then explain their drawbacks. Chapter 3 describes the detailed literature review that explains significant work done in this field of research. Section 3.1, 3.2, and 3.3 provide information about the different versions of the YOLO algorithm. Section 3.4 explains the SSD method, and section 3.5 gives the difference between SSD and YOLO approaches. Chapter 4 describes the proposed approach in which section 4.1 tells the methodology of the YOLO v3-Tiny model, and section 4.2 shows the architecture of YOLO v3-Tiny. Section 4.3 describes the hardware requirements, software requirements, and data set used in the training and testing of the model. In section 4.3.5, all the steps of implementation are specified. Then in Chapter 5, the speed and accuracy of some object detectors are analyzed and compared for different sizes of objects. At first, the experimental results of YOLO v3 and YOLO v3-Tiny models are displayed clearly. After that, we will examine the performance of YOLO v3, and YOLO v3-Tiny models with other detectors described earlier. Finally, section 5.4 gives the conclusion and future scope. At last Chapter 6 contains references for resources used to gather the information related to this project.

CHAPTER 2

BACKGROUND

In this section, we present background information. It elaborates the most representative and pioneering two-stage object detection methods with their significant contributions in object detection. First, we examine their methodologies and then explain their drawbacks.

2.1 HOG

HOG is a feature descriptor that is extensively applied in various domains to distinguish objects by identifying their shapes and structures. Local object structure, pattern, aspect, and representation can usually be characterized by the arrangement of gradients of local intensity or the ways of edges. In the HOG [12] detection method, the first step is to break the source image into blocks and then distribute each block in small regions. These regions are called cells. Commonly, the blocks of image overlap each other, due to this corresponding cell may be a part of many blocks. For each pixel inside the cell, it calculates the gradients vertically and horizontally.

Drawbacks of HOG method-

Due to the emergence of deep learning and its significant applications, the reasonable opinion was to displace classifiers deployed on HOG [12] methodology with a classifier based on convolutional neural network [2] [13] because of its comparatively higher accuracy. But there were some problems. The computation cost of convolutional neural networks was high, and the speed is too slow. Therefore, it was challenging to run CNN based classifier on numerous patches produced by the detection approach of sliding window. This difficulty was resolved by RCNN [3]. It employs an algorithm based on object proposals termed selective search method [14]. This approach decreases the number of bounding boxes to 2000 region proposals that were supplied to the R-CNN classifier.

2.2 RCNN

Region based convolutional neural networks (RCNN) algorithm [3] uses a group of boxes for the picture and then analyses in each box if either of the boxes holds a target. It employs the method of selective search to pick those sections from the picture. In an object, the four regions are used. These are varying scales, colors, textures, and enclosure.

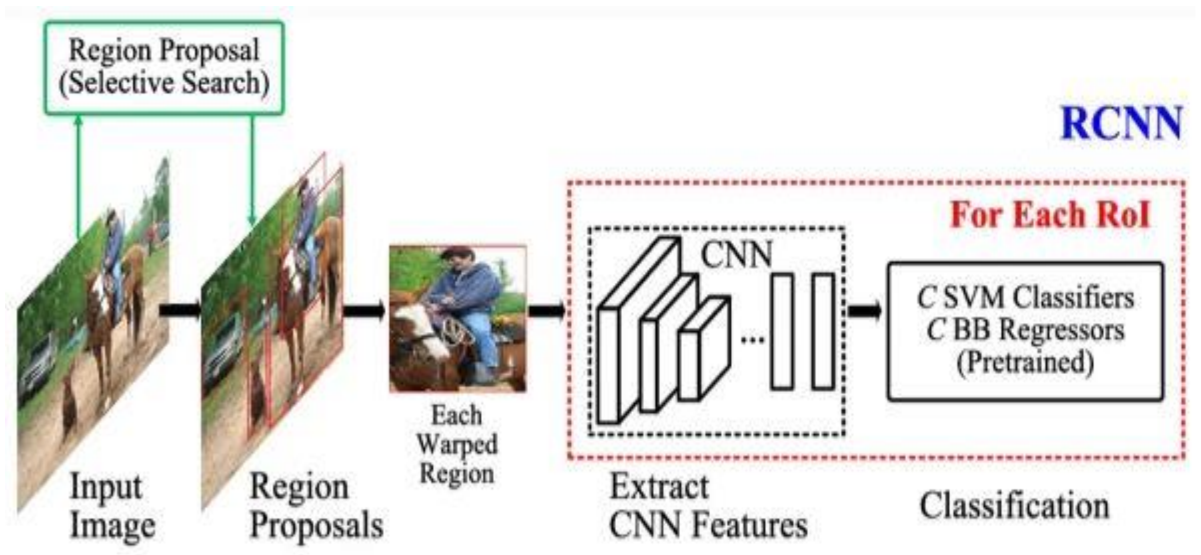

Fig.2.1: Architecture of RCNN

Drawbacks of RCNN method-

Based on a selective search [14], 2,000 sections are excerpted per image. For every region or part of the image, we have to select features using CNN. For this, if we have 'i' number of images, then selected regions will become i×2,000. The whole method of target identification through RCNN utilizes the following three models: Linear SVM classifier for the identification of objects, CNN is employed for characteristic extraction, and a regression model is required to tighten the bounding boxes. All these three processes combine to take a considerable amount of time. It increases the running time of RCNN method. Therefore, RCNN needs almost 40 to 50 seconds to predict the result for several new images.

2.3 FAST RCNN

In place of using three different models of RCNN, Fast RCNN [4] employs one model to excerpt characteristics from the different regions. Then it distributes the regions into several categories based on excerpted features, and the boundary boxes of recognized divisions return together. Fast RCNN uses the method of spatial pyramid pooling [16] to calculate only one CNN representation for the whole image. It passes one region for each picture to a particular convolutional network model by replacing three distinct models for excerption of characteristics, distributing into divisions, and producing bounding boxes.

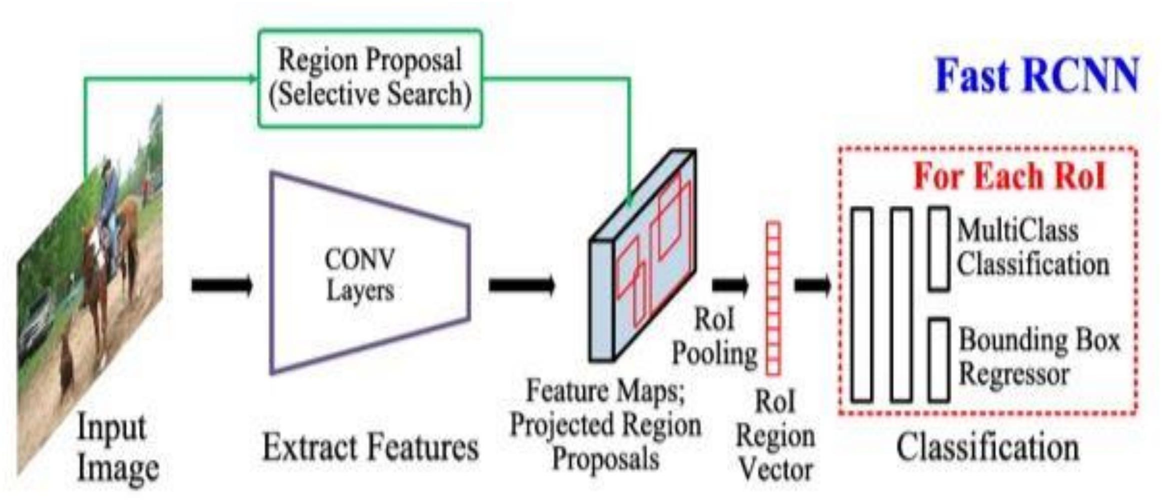

Fig.2.2: Architecture of FAST RCNN

Drawbacks of FAST RCNN method-

FAST RCNN also employ a selective search method [14] to detect concerned regions. This method is prolonged and demands a lot of time. Usually, for the detection of objects, this complete procedure needs almost two seconds for each picture. Therefore its speed is quite good in contrast to RCNN [15]. However, if we contemplate extensive real-life datasets, then the execution of fast RCNN approach is still lacked in speed.

2.4 FASTER RCNN

Faster RCNN [5] is a transformed variant of fast RCNN. The significant difference between both is that faster RCNN implements region proposal network (RPN) [6] [17], but fast RCNN utilizes a selective search technique for producing concerned regions. In input, RPN accepts feature maps of picture and produces a collection of object recommendations and an objectness score per recommendation in output. Usually, this approach takes ten times less time in contrast to fast RCNN approach because of RPN.

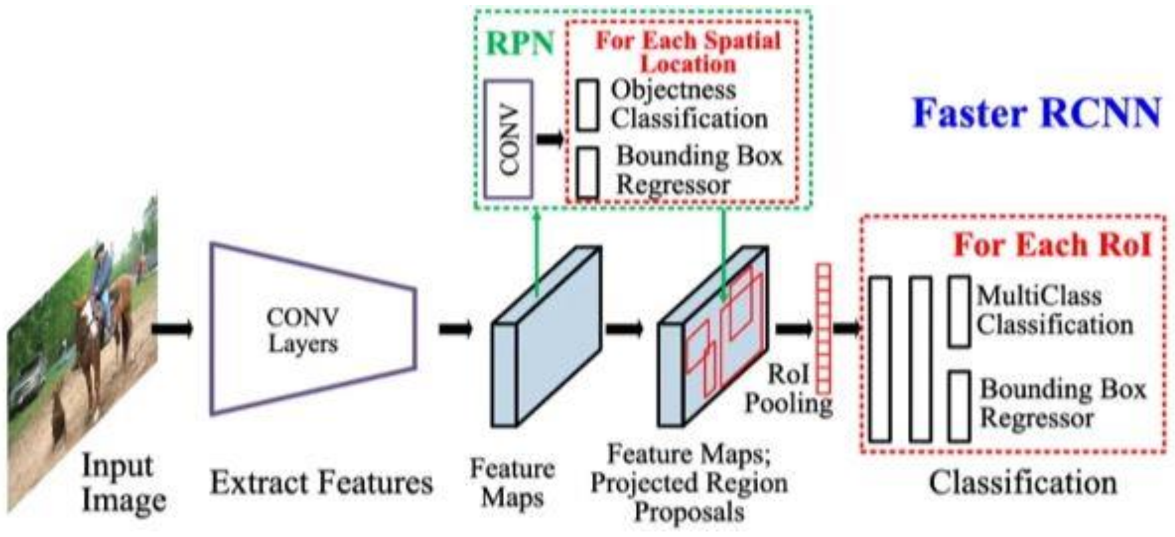

Fig.2.3: Architecture of FASTER RCNN

Drawbacks of faster RCNN method-

To excerpt all the targets in a given picture, this procedure needs multiple passes for that particular picture. Different systems are working in a sequence therefore, the performance of the upcoming operation is based on the performance of preceding operations. This approach uses region proposal networks to localize and identify the objects in a picture. But RPNs do not contemplate the complete picture because it uses only those portions of the picture, which have high probabilities of the presence of targets.

2.5 Comparison of Two stage Detectors

The table 2.1 given below will explain the comparison of these two stage object detection approaches based on their characteristics, computation time and drawbacks.

Table 2.1: Comparison of two stage object detectors

Type	**Characteristics**	**Computation time**	**Drawbacks**
RCNN	To generate regions, it uses selective search. From each picture, it extracts around 2000 regions.	Forty to Fifty seconds	Time taken for prediction is large because several regions pass through CNN definitely, and it employs three distinct models for the detection of targets.
Fast RCNN	To excerpt the features, each picture passes one time through CNN. All distinct models applied in RCNN are combined collectively to form a single model. It employs selective search method on the feature maps to produce result for target recognition.	Two seconds	The method used is prolonged and time-consuming Therefore, computation time is still high.
Faster RCNN	The previous approach is replaced with the region proposal networks. Therefore, this procedure works much faster compared to previous methods.	0.2 seconds	Object region proposal is time-consuming. Different types of systems are operating in sequence. Thus, the performance of entire procedure is based on the working of the preceding operations.

CHAPTER 3

LITERATURE REVIEW

Two stage detectors provide adequate accuracy, but the time taken for computation is high. Therefore, to process in less time by managing sufficient accuracy, one stage detectors are proposed. Some algorithms in one stage model are SSD and the variants of YOLO. By improving the architecture of two stage models and introducing some changes such as eliminating pipeline, one stage model has achieved excellent speed. But, it has not attained sufficient accuracy at the same time. Hence, continuous changes are under process by researchers.

3.1 YOLO v1

By using the approach of one stage detector YOLO, an input picture is distributed to a system of $S \times S$ grids [8]. The detection of the object depends on every grid of the input image. Grid cells are employed to predict targets inside boundary boxes. Five parameters are predicted for every boundary box. These five elements are i, j, k, l, and c. In the input picture, the centre of target inside the box is denoted by 'i' and 'j' coordinates. Here, 'k', 'l', and 'c' represent height, width, and score for confidence respectively. 'c' is measured as the probability of containing the target inside boundary box.

YOLO v1 uses the Darknet framework and ImageNet1000 dataset to train the model. It distributes the given picture to a grid of $S \times S$ cells. For every cell in the network, it computes confidence for 'n' bounding boxes. The predicted result is encoded into a tensor as $S \times S \times (n \times 5 + p)$ [8]. Here, input image is divided into total $S \times S$ sub-images. In $n \times 5$, five represents the detection of five attributes for each bounding box, which are height, weight, confidence score, and centre coordinates(x, y) of detected objects. Here, 'p' represents the probability of class. YOLO v1 also has many limitations. Therefore, the use of YOLO v1 is restricted to some extent. Limitations of YOLO version1 are based on the closeness of the objects in the picture.

If the objects appear as a cluster, they could not find the small objects. If the dimensions of the object are different from the image used in training data, then this architecture found difficulty in the localization and detection of objects [15][18]. The primary concern is to locate objects in a given picture due to the error of localization.

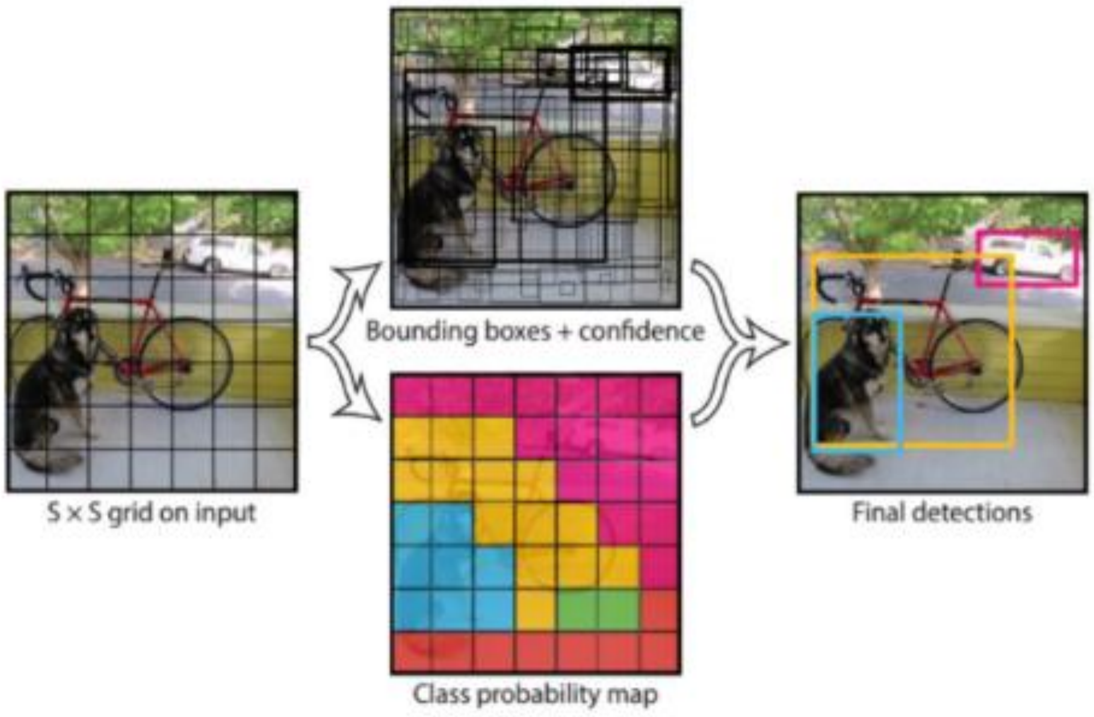

Fig.3.1: S×S grid representation of YOLO [8]

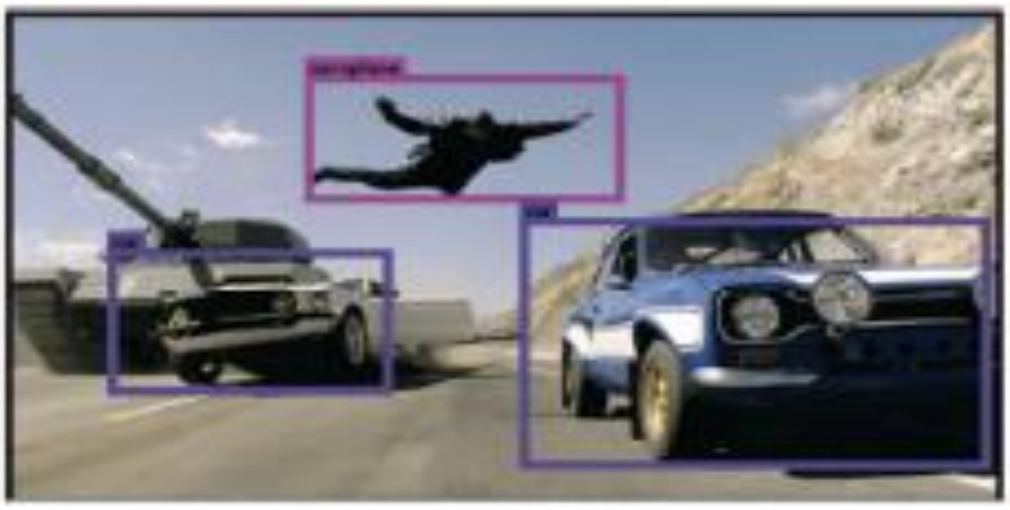

Fig.3.2: Drawback of YOLO v1 [8]

YOLO v1 fails sometimes, as in the above image it recognizes the man as an airplane, which is a drawback of this method.

3.2 YOLO v2

Yolo v2 supersedes Yolo by offering a great balance between running time and accuracy. For better accuracy, Yolo v2 introduces batch normalization, which helps to enhance 2 percent in map by attaching it into each layer of convolution. High resolution classifier is used to operate thoroughly by modifying its filters for giving a more extensive understanding of network time to work excellently. When it comes to the prophecy of bounding boxes, then by eliminating entirely connected layers to anchor boxes makes decrement of map value by 0.3, but recall value is incremented by 7% [9]. Hence, it gives more potential to detect the day to day objects.

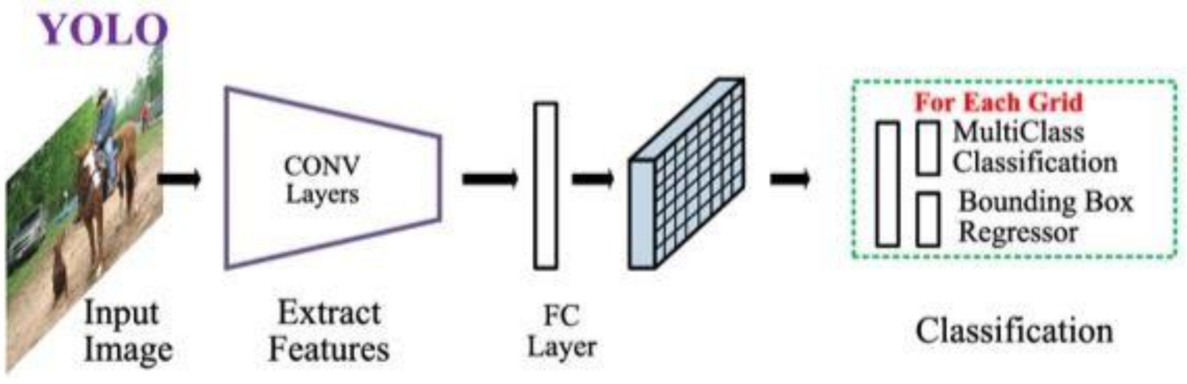

Fig.3.3: Architecture of YOLO v2

Fine grained features help to identify tiny objects by reshaping the layers employing a modified approach called pass through. As we have achieved accuracy now, our goal is to maintain a balance of running time with it. For this Yolo v2 uses darknet 19, although it could use google net, but that reduces accuracy by 2%. For learning to locate targets from any visual, Yolo 9000 is employed across nine thousand categories with a 9418 node WordTree. It is certainly progress for concluding the localization and distribution.

3.3 YOLO v3

Object detection is used in many fields of human life, for example, health and education, etc. As all these fields are growing rapidly, so to match their requirements, one stage models also need improvement. The next advanced variant of YOLO is version 3 that uses logistic regression to compute the targetness score. It gives the score for all targets in each boundry box. YOLO v3 can give the multilabel classification because it uses a logistic classifier for each class in place of the softmax layer used in YOLO v2.

YOLO v3 uses darknet 53. It has fifty-three layers of convolution. These layers are more in-depth compared to darknet 19 used in YOLO v2. Darknet-53 contains mainly 3x3 and 1x1 filters along with bypass links [8] [9] [10]. The formulas given below explain the transformation of network output for obtaining bounding box predictions. Here, d_x and d_y are center coordinates, d_w is width and d_h is the height of predicted result. Top left coordinates of the grid are m_x and m_y. Network outputs are t_x, t_y, t_w and t_h. Anchor dimensions for the box are p_h and p_w.

$$d_x = \sigma(t_x) + m_x \tag{3.1}$$

$$d_y = \sigma(t_y) + m_y \tag{3.2}$$

$$d_w = p_w e^{t_w} \tag{3.3}$$

$$d_h = p_h e^{t_h} \tag{3.4}$$

By using threshold value, a filter is applied to remove the box having class score less than the threshold value chosen. Because a score with less value represents that the box is insufficient for identifying the classes. Even after filtering by using a threshold value for the class scores, a lot of overlapping boxes still remain. When many boxes are overlapping with each other, then select only one box out of those overlapping boxes and identify the object. So, the second filter is used for choosing the desired boxes, which termed as nonmaximum suppression (NMS)[19]. It uses intersection over union (IOU) function.

$$IOU = \frac{B_1 \cap B_2}{B_1 \cup B_2} \tag{3.5}$$

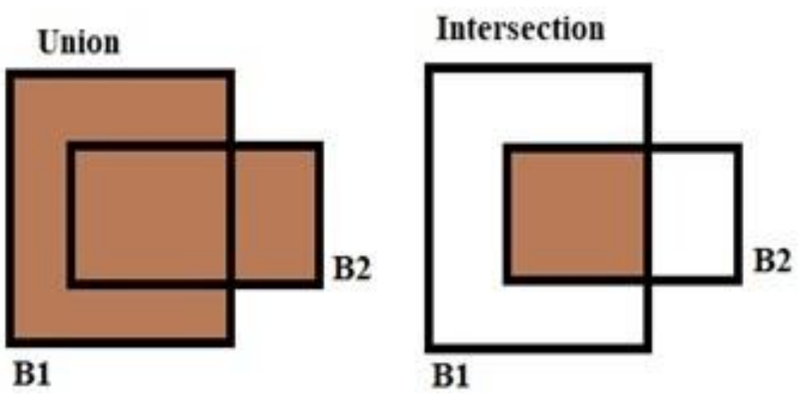

Fig. 3.4: Illustration depicting the definition of union and intersection

Above-left corner (a1, b1) and below right corner (a2, b2) are used to determine a box. For calculating the area of the rectangle, multiply its height (b2-b1) and its width (a2-a1). Then the coordinates (ai1, bi1, ai2, bi2) for the intersection of 2 boxes are obtained. Here, ai1 and bi1 are the highest value of a1 and b1 coordinate-position of the two boxes. Similarly, ai2 and bi2 is the lowest value of the a2 and b2 coordinate-position of the 2 boxes.

$$union(A,B) = A + B - inter(A,B) \tag{3.6}$$

$$inter\_area = (ai2 - ai1) \times (bi2 - bi1) \tag{3.7}$$

$$box1\_area = (box1[3] - box1[1]) \times (box1[2] - box1[0]) \tag{3.8}$$

$$box2\_area = (box2[3] - box2[1]) \times (box2[2] - box2[0]) \tag{3.9}$$

$$union\_area = (box1\_area + box2\_area) - inter\_area \tag{3.10}$$

$$IOU = inter\_area/union\_area \tag{3.11}$$

The advantage of YOLO v3 over YOLO v2 is that some changes are included in error function and for objects of small to a considerable size detection occurs on three scales. The multiclass problem turned in a multilabel problem, and the performance improved over small size objects.

3.4 SSD

SSD is a single shot detector [7] [20]. It manages an excellent balance of speed with the accuracy of result. In this, we apply for single time a CNN based model to the input picture for computing the feature map. It also employs anchor boxes similar to faster RCNN at various aspect ratios and learns the offset instead of determining the box. The processing occurs on numerous layers of CNN, where every layer functions on a varying range of scale and uses multiple feature maps. Therefore, the detection of targets of several sizes is possible. Experimentally, SSD has much better accuracy on different datasets even on inputs pictures of small size as compared to the other single stage methods.

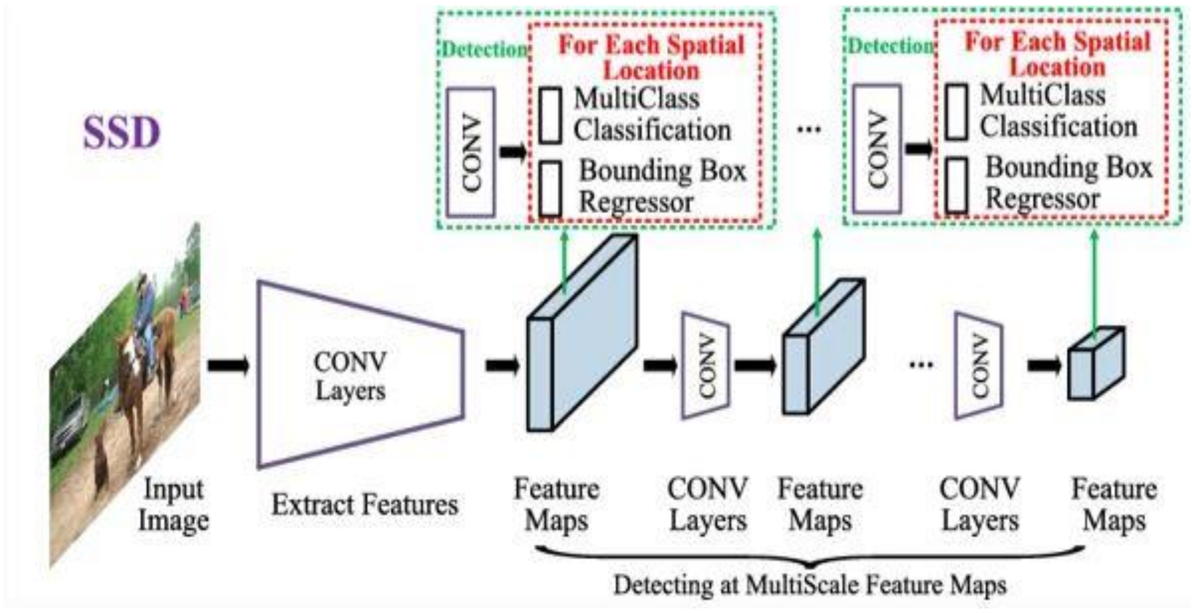

Fig.3.5: Architecture of SSD

3.5 SSD vs. YOLO

Unlike YOLO, SSD does not divide the image into grids of random size. For every location of the feature map, it predicts the offset of predefined anchor boxes (default boxes). Relative to the corresponding cell, each box has a fixed size, proportion, and position. In a convolutional manner, all the anchor boxes cover the entire feature map. Anchors of SSD are slightly different from the anchors of YOLO. Because YOLO makes all the predictions from a single grid, and the size of anchors used by YOLO ranges from dimensions of one grid cell to the dimensions of the entire picture.

The anchors of SSD specialize its detector for distinct feasible viewpoints and dimensional ratios of its target shapes, but not enough on the size of targets. The calculation for the anchors of SSD uses a simple formula, while the anchors of YOLO are calculated by applying k-means clustering on the training data [21] [15]. SSD doesn't use confidence score, but YOLO calculates it to show the faith in predicted results. A unique background class is employed by SSD for this work. A low value of confidence score in YOLO is equivalent to the predicted output of background class in SSD. Both indicate that for the detector, the possibility of getting a target is null [18].

CHAPTER 4

PROPOSED APPROACH

In this chapter, we will explain the proposed approach used in object detection and recognition using YOLO v3-Tiny model. The hardware requirements, software requirements and data set used in training and testing of model is listed along with the description of detailed procedure of implementation.

4.1 YOLO v3-Tiny

YOLO v3, with the decreased depth of the convolutional layer, is another version called YOLO v3-Tiny. It was proposed by Joseph Redmon [10]. Therefore, the running speed is significantly increased (approximately 442% faster than the former variants of YOLO), but detection accuracy is reduced. Darknet-53 architecture of YOLO v3 employs several 1x1 convolution layers along with 3x3 convolution layers for extracting features.

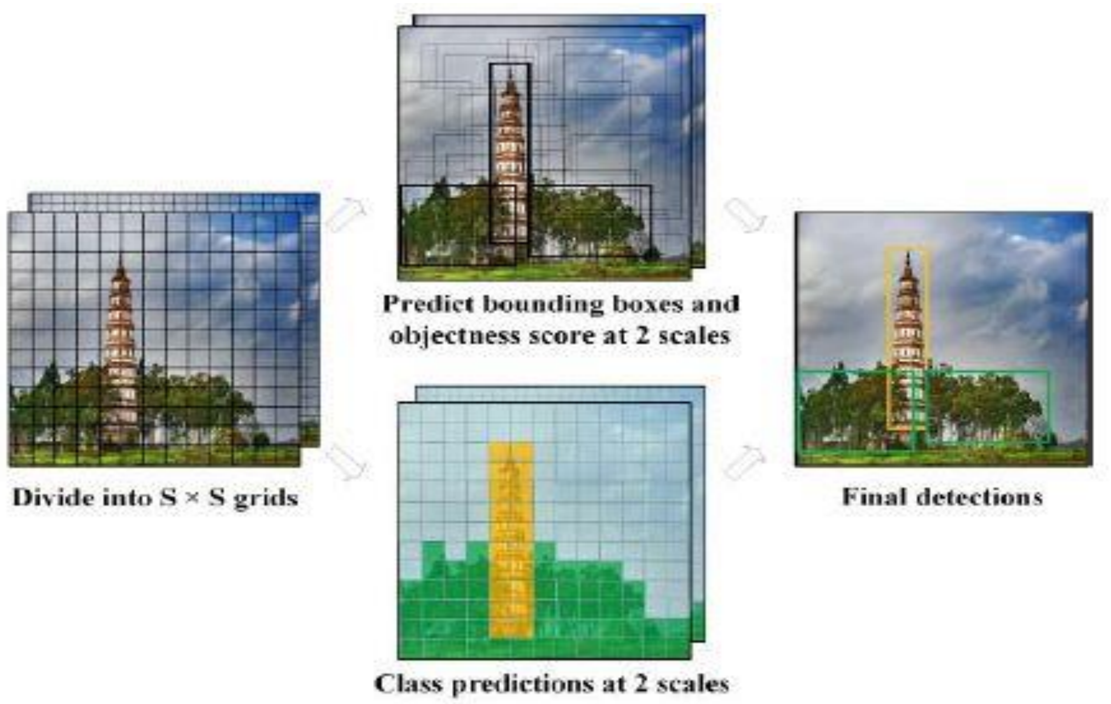
Predict bounding boxes and objectness score at 2 scales
Divide into S × S grids
Final detections
Class predictions at 2 scales

Fig.4.1: Main process of YOLO v3-Tiny [11]

4.2 Architecture of YOLO v3-Tiny

YOLO v3-Tiny uses pooling layer and reduces the figure for convolution layer. It predicts a three-dimensional tensor that contains objectness score, bounding box, and class predictions at two different scales. It divides a picture into S×S grid cells. For final detections, we will ignore the bounding boxes for which the objectness score is not best. For extracting features, convolution layers and max-pooling layers are utilized in the feed forward arrangement of YOLO v3-Tiny. Prediction of bounding boxes occurs at two different feature map scales, which are 13×13, and 26×26 merged with an upsampled 13×13 feature map.

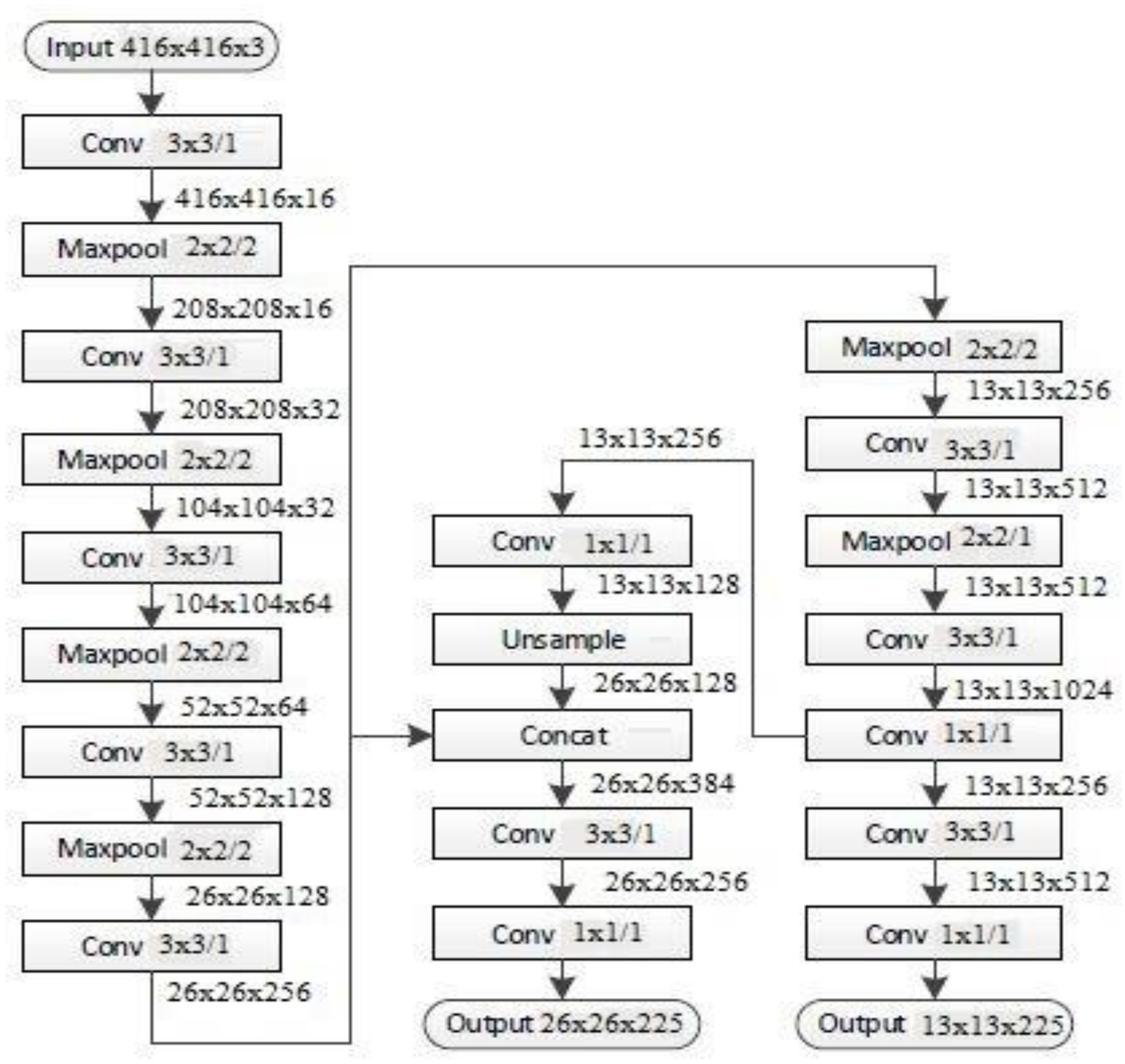

Fig.4.2: Architecture of YOLO v3-Tiny [11]

Table 4.1: Performance comparison of YOLO with their Tiny versions[25]

Detector	**Number of n-layers**	**Floating point operations**	**FPS**	**map value**	**Used set of data**
YOLO v1	26.00	Not-given	45.00	63.50	VOC-data
YOLO v1-Tiny	9.00	Not-given	155.00	52.80	VOC-data
YOLO v2	32.00	62.95	40.00	48.20	COCO-data
YOLO v2-Tiny	16.00	05.42	244.00	23.60	COCO-data
YOLO v3	106.00	140.70	20.00	57.80	COCO-data
YOLO v3-Tiny	24.00	05.57	220.00	33.20	COCO-data

In Table 4.1, we can see that on different datasets, the Tiny versions of YOLO have a lower value for map as compared to their original versions.

4.3 Implementation Details

Below in this section, the hardware requirements, software requirements and data set used in training and testing of model is listed along with the description of detailed procedure of implementation.

4.3.1 Hardware Requirements

The YOLO v3 and YOLO v3-Tiny models are implemented with the following hardware requirements.

- Operating system used is windows 10 because it is fast and very easy to use.
- To execute program instruction very fast we are using Intel core i5 eighth-generation quad-core CPU.
- When CPU doesn’t meet the requirement then we are using NVIDIA Geforce GTX 1050 GPU which is approx. ten times faster than CPUs.
- To run smoothly Minimum 8 gigabytes Size of RAM is required.
- Some files needs to be downloaded online, extremely high-speed internet is compulsory.

4.3.2 Software Requirements

The YOLO v3 and YOLO v3-Tiny models are implemented with the following software requirements.

- Anaconda Jupyter notebook is a very simple and excellent IDE used to run python codes.
- Open CV python is used to solve computer vision problems with complex numerical operations.
- TensorFlow 2.0 was developed by Google to provide numerical computing, and it creates models by data flow graphs.
- Python imaging library is used for opening, saving, and modifying any image related files.
- Numpy is used to provide support to high-level mathematical functions. It also supports to solve multidimensional arrays and matrices.
- We have used the latest python language version to run the code.

4.3.3 Data Set Used

The Darknet 53 convolution neural-network is employed as a previously trained network. It is trained earlier on the Imagenet dataset for classification. As training a model from scratch is much more complicated, so we employed transfer learning on our model using VOC 2012 dataset for localization of objects. VOC dataset is very popular for object detection, but it needs good internet speed and better disk performance.

Table 4.2: Darknet-53 data set

Model	Train	test	MAP	Flops	FPS	CFG	WEIGHTS
SSD300	COCO train_val	Test_dev	41.2	-	46	-	link
SSD500	COCO train_val	Test_dev	46.5	-	19	-	link
Yolov2 608*608	COCO train_val	Test_dev	48.1	62.94 bn	40	cfg	weights

Tiny Yolo	COCO train_val	Test_dev	23.7	5.41bn	244	cfg	weights
SSD321	COCO train_val	Test_dev	45.4	-	16	-	link
SSD513	COCO train_val	Test_dev	50.4	-	8	-	link
YOLOv3-320	COCO train_val	Test_dev	51.5	38.97 bn	45	cfg	weights
YOLOv3-416	COCO train_val	Test_dev	55.3	65.86 bn	35	cfg	weights
YOLOv3-608	COCO train_val	Test_dev	57.9	140.69 bn	20	cfg	weights
YOLOv3-tiny	COCO train_val	Test_dev	33.1	5.56 bn	220	cfg	weights

4.3.4 Implementation Method

Below in this section, the steps of implementation of YOLO v3 and YOLO v3-Tiny models are explained.

- We have taken input as preprocessed image.
- The output is in the form of bounding boxes, accuracy score percentage, and class label.
- During the training of model, ninety percent of data is applied for training, and the rest ten percent is used for the validation of our model.
- We trained our model for localizing twenty classes that include person, dog, and car, etc.
- During batch normalization, the size of each batch is sixteen and requires approximately twenty iterations to get a valid result.
- The duration of the period needed for each iteration is around ten minutes.
- Then the trained models of YOLO v3 and YOLO v3-Tiny are applied to the pictures taken by us in our college campus to examine the execution time and accuracy (map value) of the result obtained.
- Speed is calculated in terms of frame per second, which is the number of images processed by the model in each second.
- Mean average precision is computed to estimate accuracy. After each iteration, precision and recall value is calculated, and a curve is plotted between them.

- The area under this curve is an average precision of that iteration.
- At last, we will find the mean value of all obtained average precisions to get map value.
- We have observed that YOLO v3-Tiny is faster compared to YOLO v3, and therefore, it is more effective in real time target localization and identification of class. However, its accuracy decreases a bit in comparison to YOLO v3.

4.3.5 Steps of Implementation

1. Padding helps to add extra pixel to the corner of actual photo because due to passing through convolutional layers, we can lose pixels and for computational efficiency we need to use stride which traversed row and column to cover all locations.

```
def Conv2D(inputs, filters, kernel_size, data_format, strides=1):
                    if strides > 1:
                    inputs = fixed_padding(inputs, kernel_size, data_format)
                    return tf.layers.Conv2D(filters=filters,
                    kernel_size=kernel_size, strides=strides,
                    padding=('same' if strides == 1 else 'valid'),
                    use_bias=False, data_format=data_format)(inputs)
```

2. Performs a batch normalization to improve speed and performance of network training. It also normalize the input layer by rescaling and recentering.

```
def BatchNormalization(inputs, data_format):
    return tf.layers.BatchNormalization(axis=1
            if data_format == 'channels_first'
            else 3,momentum=_BATCH_NORM_DECAY,
            epsilon=_BATCH_NORM_EPSILON,
            scale=True)(inputs)
```

3. Performs a max pooling which helps to reduce the spatial size of initial box representation.

```
def MaxPooling2D(inputs, pool_size, strides, data_format):
    return tf.layers.MaxPooling2D(pool_size, strides, padding='same',
            data_format=data_format)(inputs)
```

4. Activation function leakyRELU() is used to provide values to the functions which tends to zero, it also helps to improve the dying relu problem by increasing the range from infinity to negative infinity.

```
def LeakyReLU(inputs):
    return tf.nn.leaky_relu(inputs, alpha=_LEAKY_RELU)
```

5. Upsamples to out_shape using nearest neighbor interpolation which use inputs as new_height and new_width where ‘channels_first’ defined as default data_format.

```
def upsample(inputs, out_shape, data_format):
if data_format == 'channels_first':
inputs = tf.transpose(inputs, [0, 2, 3, 1])
new_height = out_shape[3]
new_width = out_shape[2]
else:
new_height = out_shape[2]
new_width = out_shape[1]
inputs = tf.image.resize_nearest_neighbor(inputs,
        (new_height, new_width))
if data_format == 'channels_first':
inputs = tf.transpose(inputs, [0, 3, 1, 2])
return inputs
```

6. Computes top left and bottom right points of the boxes by providing the dimension of box and axis value as -1.

```
def build_boxes(inputs):
center_x, center_y, width, height, confidence, classes = tf.split(inputs,
[1, 1, 1, 1, 1, -1], axis=-1)
top_left_x = center_x - width / 2
top_left_y = center_y - height / 2
bottom_right_x = center_x + width / 2
bottom_right_y = center_y + height / 2
return tf.concat([top_left_x, top_left_y, bottom_right_x, bottom_right_y,
confidence, classes], axis=-1)
```

7. We are importing required YOLO v3 files to set input_size and anchors values. The required layers and weight files are loaded into memory to perform the module operations.

```
import tensorflow as tf
from core.layers import Conv2D, BatchNormalization, LeakyReLU,
yolo_layer, upsample, build_boxes, non_max_suppression
_INPUT_SIZE = [416, 416]
_MAX_OUTPUT_SIZE = 20
_ANCHORS = [(10, 13), (16, 30), (33, 23),
(30, 61), (62, 45), (59, 119),
(116, 90), (156, 198), (373, 326)
```

8. Creates a residual block for Darknet to provide conversion and batchNormalization by increasing filters size to some fixed values.

```
def darknet53_residual_block(inputs, filters, data_format, strides=1):
shortcut = inputs
inputs=Conv2D(inputs,filters=filters,kernel_size=1,strides=strides,
data_format=data_format)
inputs = BatchNormalization(inputs, data_format=data_format)
inputs = LeakyReLU(inputs)
filters *= 2
```

```
inputs=Conv2D(inputs,filters=filters,kernel_size=3,strides=strides,
data_format=data_format)
inputs = BatchNormalization(inputs, data_format=data_format)
inputs = LeakyReLU(inputs)
inputs += shortcut
return inputs
```

9. Creates Darknet53 model for YOLO v3 whereas for each range strides has default value 2 but filters are getting change by current value multiplied by 2.

```
def darknet53(inputs, data_format):
inputs = Conv2D(inputs, filters=32, kernel_size=3,
data_format=data_format)
inputs = BatchNormalization(inputs, data_format=data_format)
inputs = LeakyReLU(inputs)
inputs=Conv2D(inputs, filters=64, kernel_size=3, strides=2,
data_format=data_format)
inputs = BatchNormalization(inputs, data_format=data_format)
inputs = LeakyReLU(inputs)
inputs = darknet53_residual_block(inputs, filters=32,
data_format=data_format)
inputs=Conv2D(inputs,filters=128,kernel_size=3,strides=2,
data_format=data_format)
inputs = BatchNormalization(inputs, data_format=data_format)
inputs = LeakyReLU(inputs)
```

10. Creates convolution operations layer used after Darknet53 model for YOLO v3.

```
def feature_pyramid_network(inputs, filters, data_format):
inputs = Conv2D(inputs, filters=filters, kernel_size=1,
data_format=data_format)
inputs = Conv2D(inputs, filters=filters, kernel_size=1,
data_format=data_format)
inputs = LeakyReLU(inputs)
inputs=Conv2D(inputs, filters=(filters * 2), kernel_size=3,
data_format=data_format)
inputs = BatchNormalization(inputs, data_format=data_format)
inputs = LeakyReLU(inputs)
```

11. YOLO v3 Model with parameters as number of class labels, input size of the model, maximum number of boxes to be selected for each class, Threshold for the IOU, Threshold for the confidence score.

```
class YOLOv3(object):
def __init__(self, n_classes, iou_threshold, confidence_threshold):
self.n_classes = n_classes
self.input_size = _INPUT_SIZE
self.max_output_size = _MAX_OUTPUT_SIZE
self.iou_threshold = iou_threshold
self.confidence_threshold = confidence_threshold
self.data_format='channels_first'if tf.test.is_built_with_cuda()
else'channels_last'
self.scope = 'yolov3'
```

12. Creates Darknet model for YOLO v3-tiny v3 whereas for each range kernel_size has default value 3 but filters are started by number and getting change by current value multiplied by the same value.

```
def darknet(inputs, data_format):
filters = 16
for _ in range(4):
inputs = Conv2D(inputs, filters, kernel_size=3,
data_format=data_format)
inputs = BatchNormalization(inputs, data_format=data_format)
inputs = LeakyReLU(inputs)
inputs=MaxPooling2D(inputs,pool_size=[2,2], strides=[2, 2],
data_format=data_format)
filters *= 2
inputs = Conv2D(inputs, filters=256, kernel_size=3,
data_format=data_format)
inputs = BatchNormalization(inputs, data_format=data_format)
inputs = LeakyReLU(inputs)
route = inputs
inputs = MaxPooling2D(inputs, pool_size=[2, 2], strides=[2, 2],
data_format = data_format)
```

13. Creates convolution operations layer by taking inputs, data_format as parameter to perform operation for BatchNormalization and LeakyReLU used after Darknet YOLO v3-tiny.

```
def feature_pyramid_network(inputs, data_format):
inputs = Conv2D(inputs, filters=256, kernel_size=1,
data_format=data_format)
inputs = BatchNormalization(inputs, data_format=data_format)
inputs = LeakyReLU(inputs)
route = inputs
inputs = Conv2D(inputs, filters=512, kernel_size=3,
data_format=data_format)
inputs = BatchNormalization(inputs, data_format=data_format)
inputs = LeakyReLU(inputs)
return inputs, route
```

14. YOLO v3-tiny Model with parameters as number of class labels, input size of the model, maximum number of boxes to be selected for each class, Threshold for the IOU, Threshold for the confidence score.

```
class YOLOv3_tiny(object):
def __init__(self, n_classes, iou_threshold, confidence_threshold):
        self.n_classes = n_classes
        self.input_size = _INPUT_SIZE
        self.max_output_size = _MAX_OUTPUT_SIZE
        self.iou_threshold = iou_threshold
        self.confidence_threshold = confidence_threshold
        self.data_format='channels_first'if
        tf.test.is_built_with_cuda()
        else'channels_last'
        self.scope = 'yolov3_tiny'
```

15. Loads kernel, gamma, beta, mean, variance for Batch Normalization by creating a load_batch_norm method which takes five parameters by returning assign_ops and offset.

```
def load_batch_norm(idx, variables, weights, assign_ops, offset):
kernel = variables[idx]
gamma, beta, mean, variance = variables[idx + 1:idx + 5]
batch_norm_vars = [beta, gamma, mean, variance]
for var in batch_norm_vars:
shape = var.shape.as_list()
num_params = np.prod(shape)
var_weights = weights[offset:offset + num_params].reshape(shape)
                offset += num_params
assign_ops.append(tf.assign(var, var_weights))
shape = kernel.shape.as_list()
num_params = np.prod(shape)
var_weights=weights[offset:offset+num_params].reshape((shape[3],
shape[2], shape[0], shape[1]))
var_weights = np.transpose(var_weights, (2, 3, 1, 0))
offset += num_params
assign_ops.append(tf.assign(kernel, var_weights))
return assign_ops, offset
```

16. In this phase at first, for range value 52 we are loading official pretrained YOLO v3 weights & weight for darknet part and in second iteration for each convolution layer with batch normalization & Loading weights.

```
for i in range(52):
idx = 5 * i
assign_ops, offset = load_batch_norm(idx, variables, weights,
assign_ops, offset)
ranges = [range(0, 6), range(6, 13), range(13, 20)]
unnormalized = [6, 13, 20]
for j in range(3):
for i in ranges[j]:
idx = 52 * 5 + 5 * i + j * 2 assign_ops,
offset = load_batch_norm(idx, variables, weights, assign_ops, offset)
```

17. In this phase at first, for range value 52 we are loading official pretrained YOLO v3-tiny weights & weight for darknet part and in second iteration for each convolution layer with batch normalization & Loading weights. During batch normalization, the size of each batch is sixteen and requires approximately twenty iterations to get a valid result. The duration of the period needed for each iteration is around ten minutes.

```
for i in range(7):
idx = 5 * i
assign_ops, offset = load_batch_norm(idx, variables, weights,
assign_ops, offset)
ranges = [range(0, 2), range(2, 4)]
unnormalized = [2, 4]
for j in range(2):
for i in ranges[j]:
idx = 7 * 5 + 5 * i + j * 2 assign_ops,
offset = load_batch_norm(idx, variables, weights, assign_ops, offset)
bias = variables[7 * 5 + unnormalized[j] * 5 + j * 2 + 1]
shape = bias.shape.as_list()
num_params = np.prod(shape)
var_weights = weights[offset:offset + num_params].reshape(shape)
```

18. Parameter tiny is passed to main method which will further decide to call yolov3_tiny or yolov3 method to perform operations by loading model variables, weight to train the model. During the training of model, ninety percent of data is applied for training, and the rest ten percent is used for the validation of our model. We trained our model for localizing twenty classes that include person, dog, and car, etc.

```
def main(tiny):
if tiny:
model = YOLOv3_tiny(n_classes=80, iou_threshold=0.5,
confidence_threshold=0.5)
else:
model = YOLOv3(n_classes=80, iou_threshold=0.5,
confidence_threshold=0.5)
inputs = tf.placeholder(tf.float32, [1, 416, 416, 3])
model(inputs)
model_vars = tf.global_variables(scope=model.scope)
if tiny:
assign_ops = load_weights_tiny(model_vars, './weights/yolov3-
tiny.weights')
else:
assign_ops = load_weights(model_vars, './weights/yolov3.weights')
```

CHAPTER 5

RESULTS AND DISCUSSION

In this chapter, the speed and accuracy of some object detectors is analyzed and compared for different size of objects. At first, the experimental results of YOLO v3 and YOLO v3-Tiny models are demonstrated clearly. After that we will compare the performance of YOLO v3 and YOLO v3-Tiny modes with other detectors described earlier.

5.1 Analysis of various Object Detectors

It is tough to make a clear comparison between different object detection methods. So, we can't give a straight decision on the best model. For many real-life applications, we make choices to create an equilibrium of accuracy with speed. Therefore, we need to be aware of other characteristics that have a significant impact on performance. For example, matching strategy and IOU threshold, ratio of positive anchor and negative anchor, training dataset, utilization of varying scale and cropped pictures for training, location loss function, pace of learning, and learning rate decay etc.

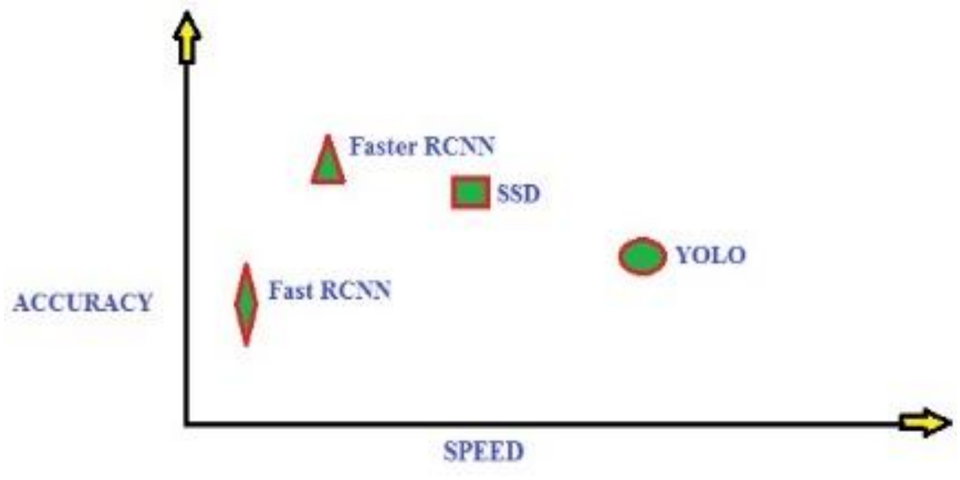

Fig.5.1: Speed vs. Accuracy for object detection methods

We have thoroughly examined the characteristics of three procedures for object detection, which are faster RCNN, YOLO v3, and SSD. If the requirement is accuracy (excellent quality of correctness), then faster RCNN is the best as it is much accurate compared to SSD and YOLO v3. But if accuracy is not the primary concern, we want super-fast speed, YOLO v3 takes less time compared to SSD and faster RCNN. But if at the same time requirement is of excellent accuracy and less running time, then SSD is a more favorable recommendation, as its speed is better than faster RCNN, and its accuracy is better compared to YOLO v3.

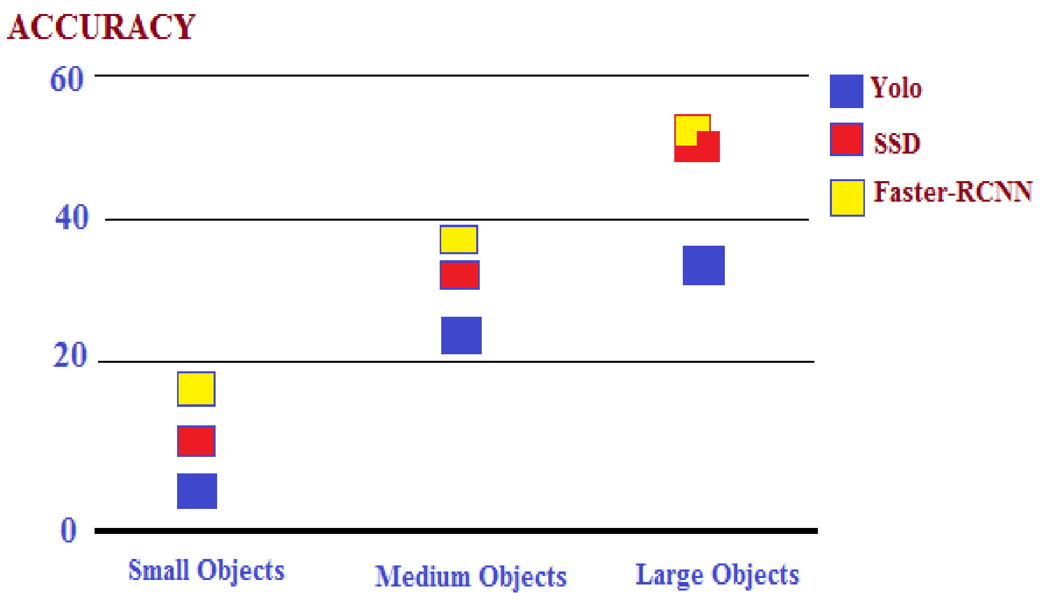

Fig.5.2: Accuracy comparison for different sizes of target objects

The chart given above in Fig.5.2 represents the faster RCNN, YOLO, and SSD performance on targets of distinct sizes. In accuracy comparison for the big targets, SSD performs alike faster RCNN. But as the object size decreases, the gap increases.

5.2 Demonstration of Results

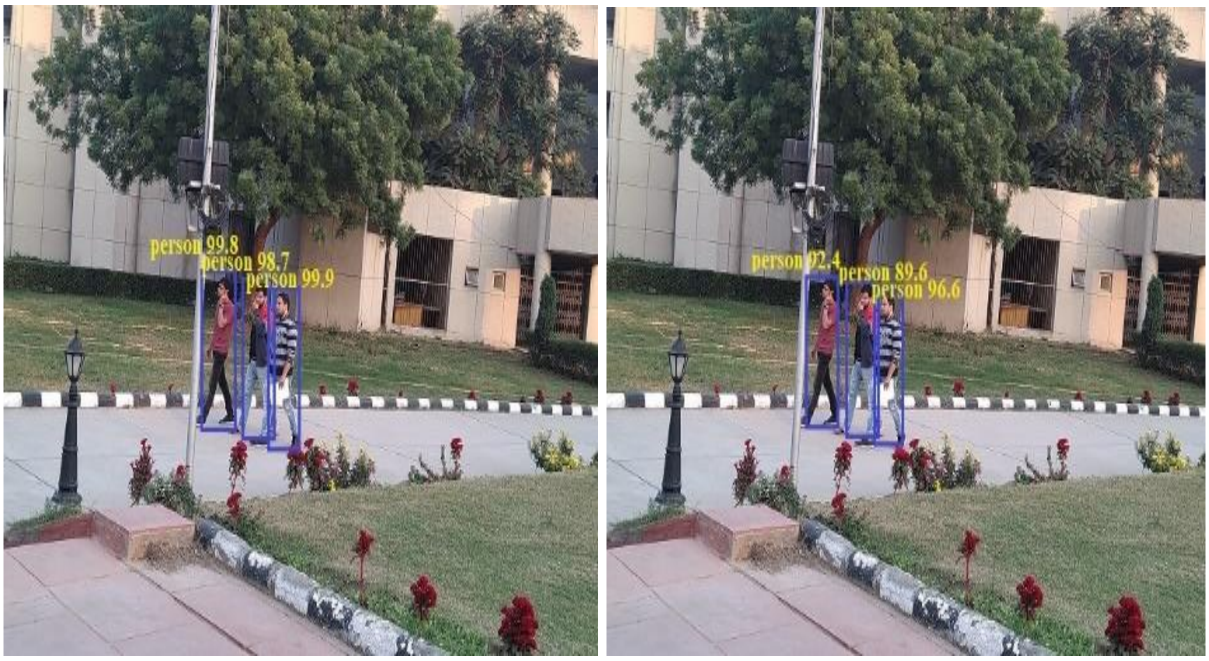

Fig.5.3: Experimental result-1 of YOLO v3 and YOLO v3-Tiny

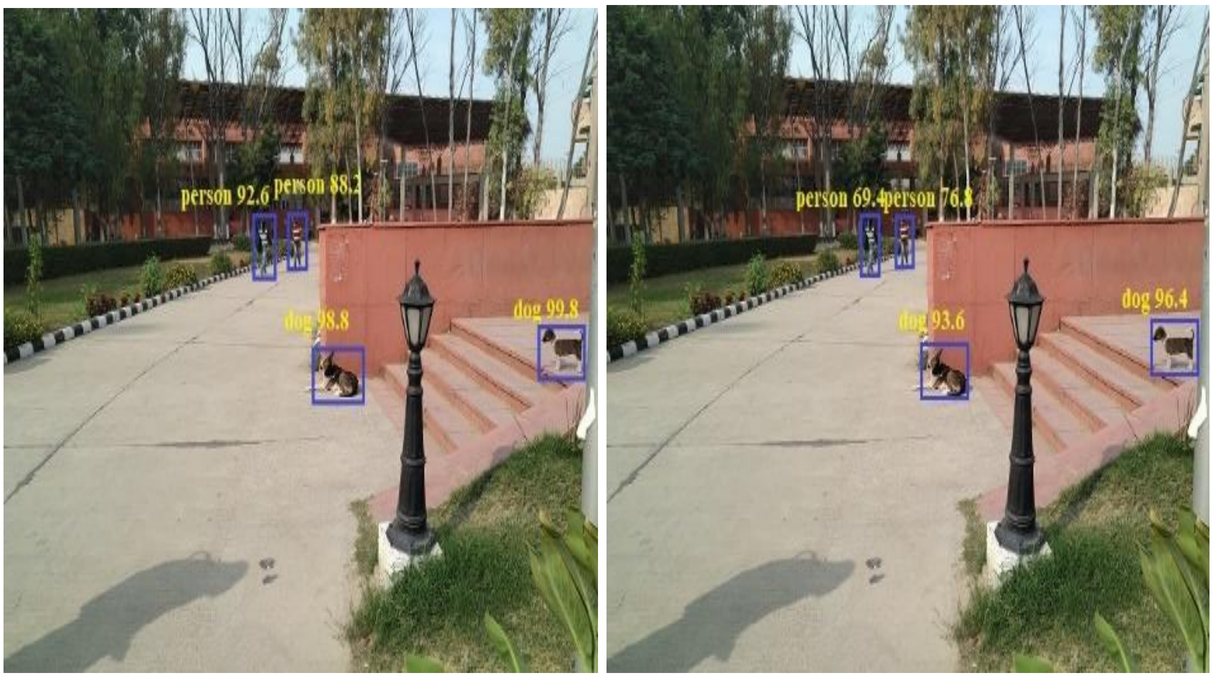

Fig.5.4: Experimental result-2 of YOLO v3 and YOLO v3-Tiny

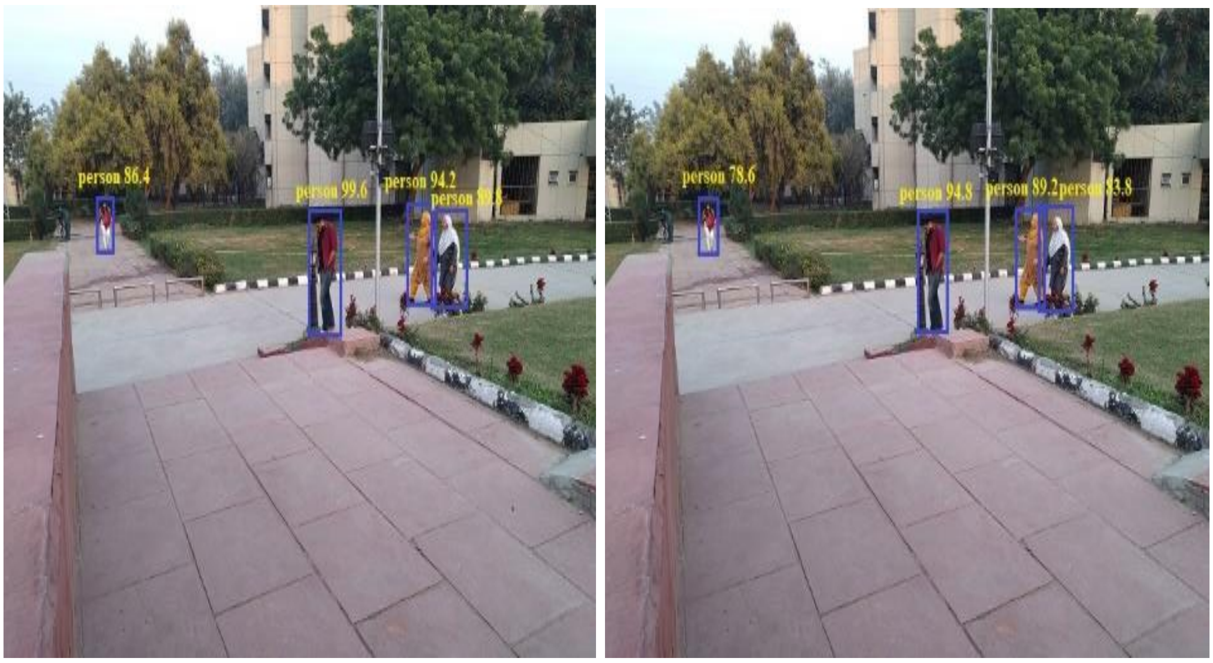

Fig.5.5: Experimental result-3 of YOLO v3 and YOLO v3-Tiny

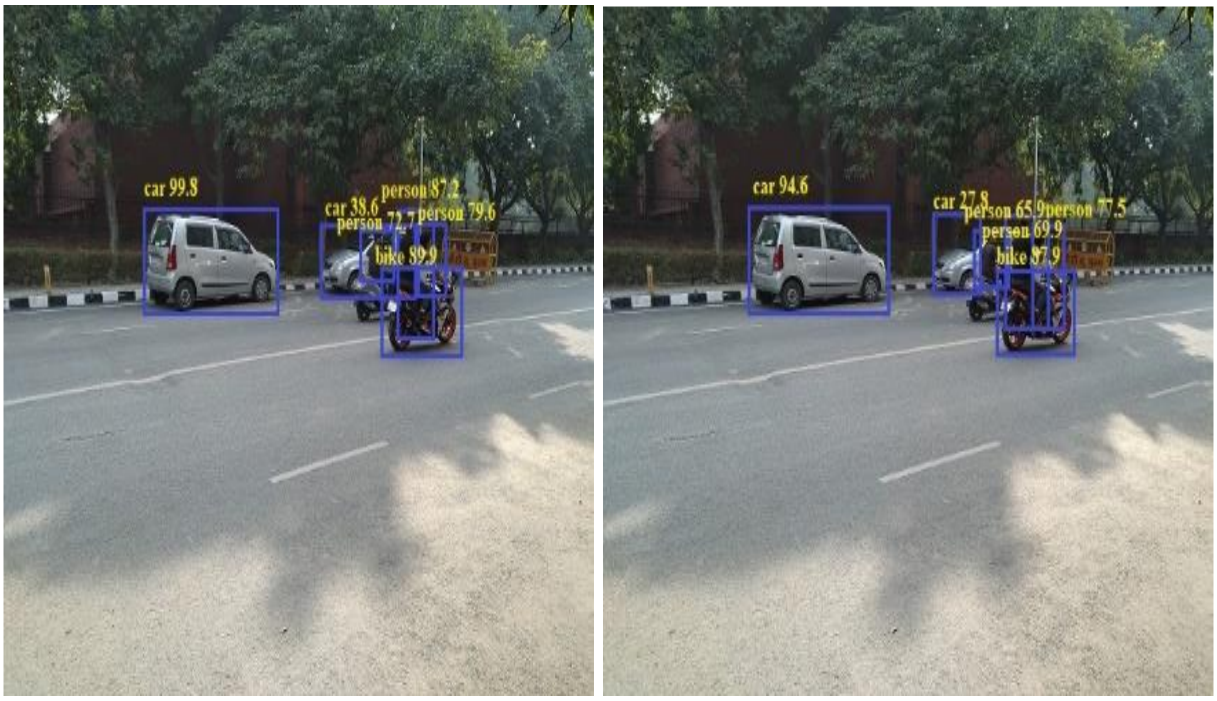

Fig.5.6: Experimental result-4 of YOLO v3 and YOLO v3-Tiny

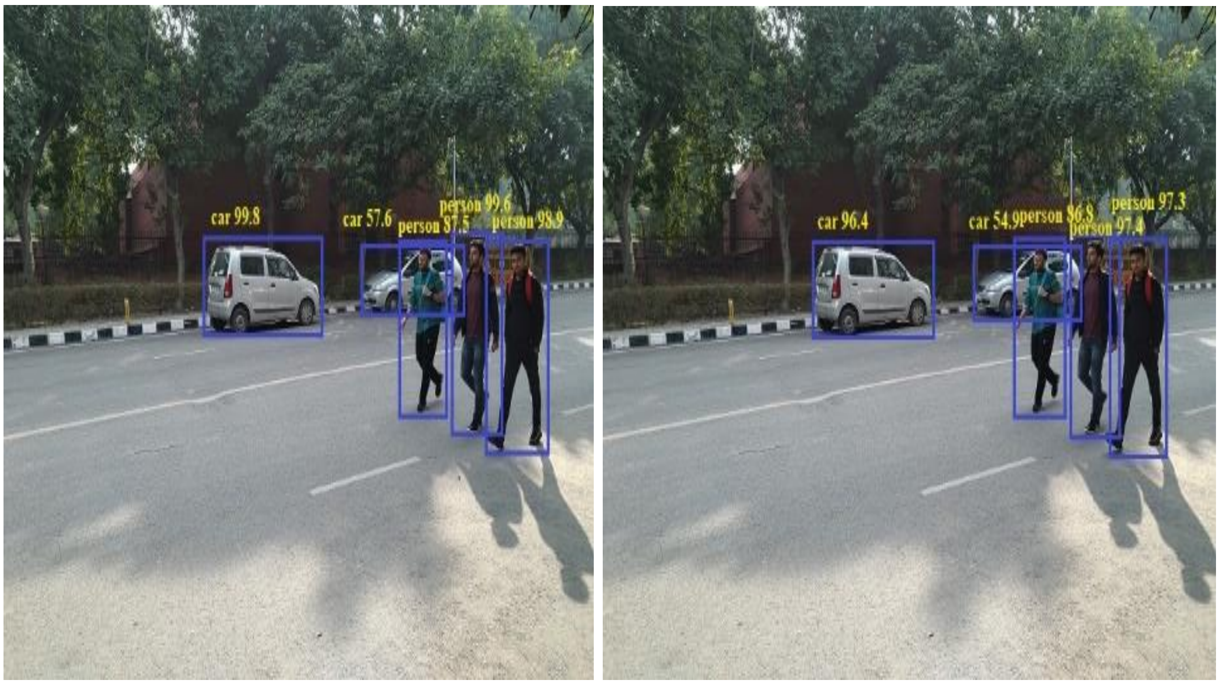

Fig.5.7: Experimental result-5 of YOLO v3 and YOLO v3-Tiny

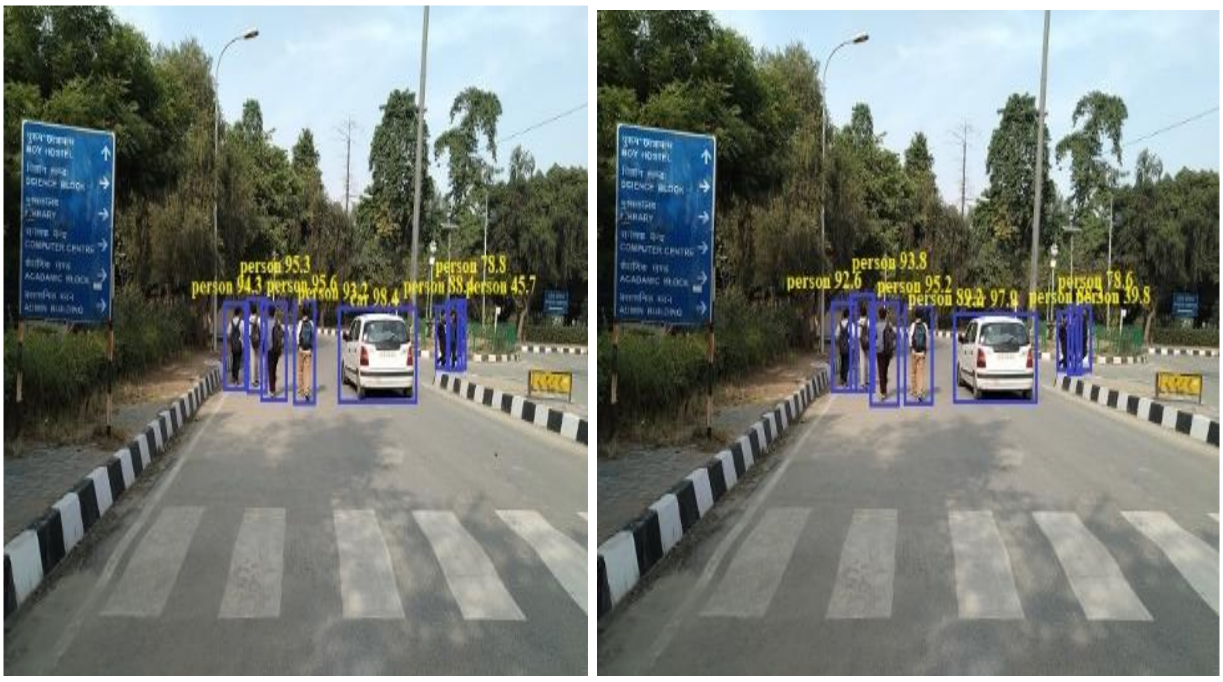

Fig.5.8: Experimental result-6 of YOLO v3 and YOLO v3-Tiny

5.3 Comparison of YOLO Algorithms

YOLO v3-Tiny is a lightweight variant of YOLO v3, which takes less running time and less accuracy when examined with YOLO v3. In Fig.5.9, we compared the accuracy of different versions of YOLO algorithms for given image pixels.

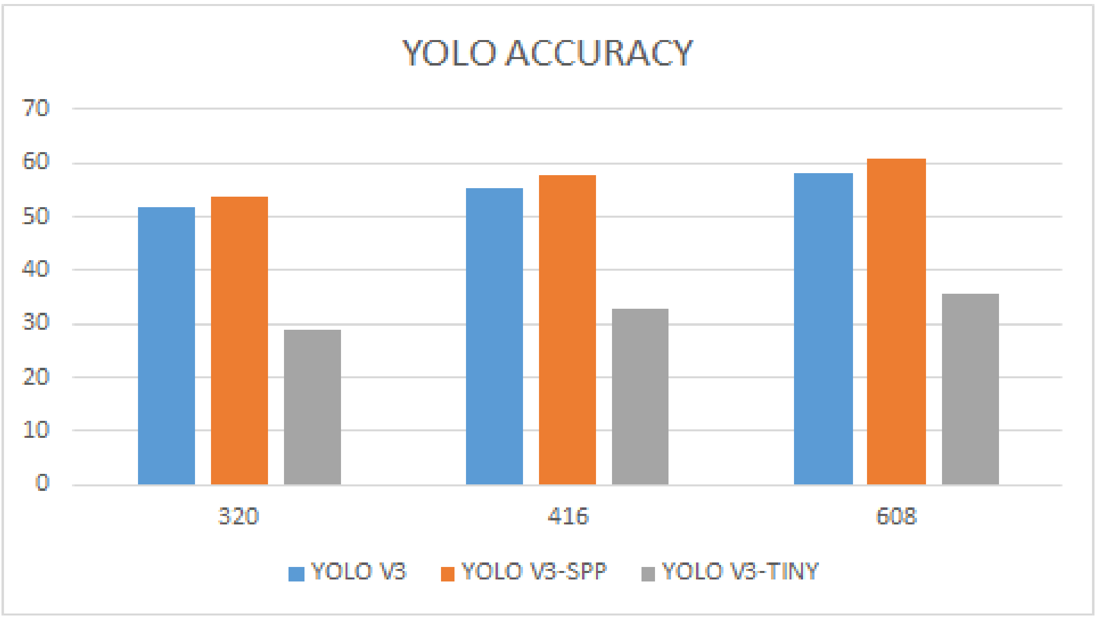

Fig.5.9: Accuracy comparison of YOLO algorithms

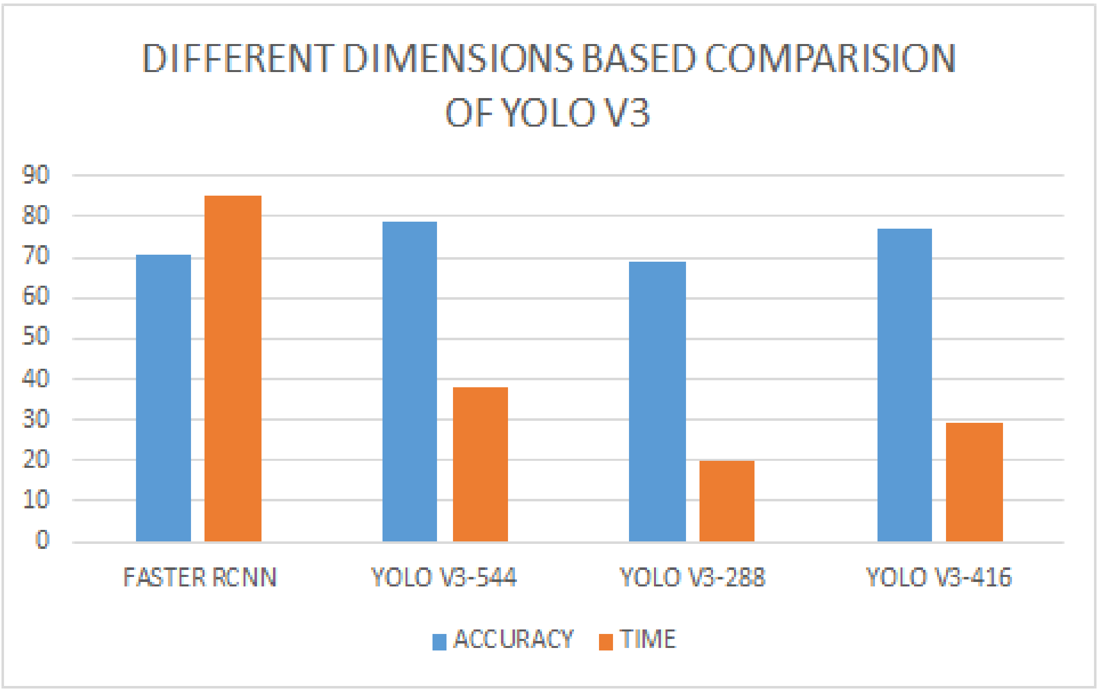

Fig.5.10: Accuracy and speed comparison of YOLO with faster RCNN

In Fig.5.10, the accuracy and speed performance of YOLO algorithms is compared with faster RCNN [27]. As per the result, we can say that YOLO v3 takes less time compared to Faster R-CNN, and YOLO v3-tiny is faster than YOLO v3.

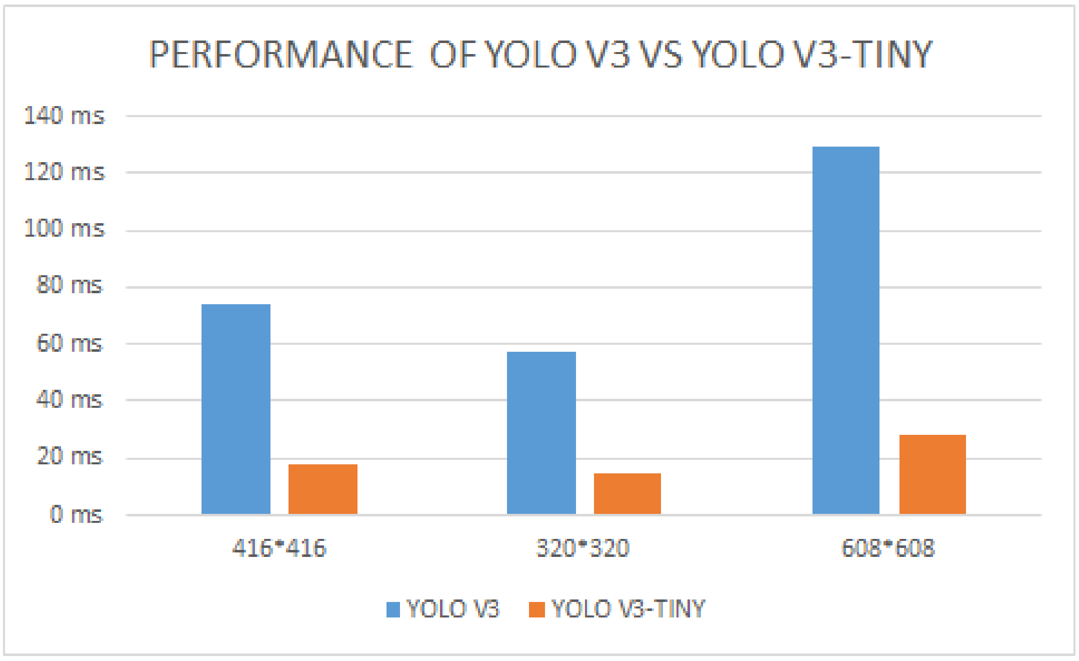

Fig.5.11: Speed Comparison of YOLO v3 vs. YOLO v3-Tiny

The above graph in Fig.5.11 shows a comparison of the running time of YOLO v3 with YOLO v3-Tiny for different dimensions of images. We have observed that YOLO v3-Tiny is faster compared to YOLO v3, and therefore, it is more effective in real time target localization and identification of class. However, its accuracy decreases a bit in comparison to YOLO v3.

5.4 Conclusion and Future Scope

To identify and localize objects, there exist many methods with a trade-off in speed performance and accuracy of result. But yet we can't say any single algorithm is best over others. One can always select the method that suits the requirement at best. In a short period, object detection applications got much popularity and still a lot to cover in this area because of its vast scope of research. This project presents the comparison of various algorithms to identify and localize objects based on accuracy, time, and parameter values with varying sizes of the input image. We have identified a new methodology of single stage model for improving speed without sacrificing much accuracy. The comparison results show that YOLO v3-Tiny increases the speed of object detection while ensures the accuracy of the result. We can also extend object localization and recognition from static pictures to a video containing the dynamic sequence of images. We are planning to increase the accuracy of the localization of small targets in the future.

CHAPTER 6
REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E.Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” NIPS, 2012, doi: 10.1201/9781420010749.

[2] R. L. Galvez, A. A. Bandala, E. P. Dadios, R. R. P. Vicerra, and J. M. Z. Maningo, “Object Detection Using Convolutional Neural Networks,” IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON, vol. 2018October, no. October, pp. 2023–2027, 2019, doi: 10.1109/TENCON.2018.8650517.

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp. 580–587, 2014, doi: 10.1109/CVPR.2014.81.

[4] R. Girshick, “Fast R-CNN,” Proc. IEEE Int. Conf. Comput. Vis., vol. 2015 International Conference on Computer Vision, ICCV 2015, pp. 1440–1448, 2015, doi: 10.1109/ICCV.2015.169.

[5] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards RealTime Object Detection with Region Proposal Networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, pp. 1137–1149, 2017, doi: 10.1109/TPAMI.2016.2577031.

[6] P. Dong and W. Wang, “Better region proposals for pedestrian detection with R-CNN,” 30th Anniv. Vis. Commun. Image Process., pp. 3–6, 2016, doi: 10.1109/VCIP.2016.7805452.

[7] W. Liu, D. Anguelov, D. Erhan, and C. Szegedy, “SSD: Single Shot MultiBox Detector,” ECCV, vol. 1, pp. 21–37, 2016, doi: 10.1007/9783-319-46448-0.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real- time object detection,” IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2016-Decem, pp. 779–788, 2016, doi: 10.1109/CVPR.2016.91.

[9] J. Redmon and A. Farhadi, “YOLO9000: Better, faster, stronger,” 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR, vol. 2017-Janua, pp. 6517–6525, 2017, doi: 10.1109/CVPR.2017.690.

[10] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” arXiv Prepr., 2018.

[11] S. Ding, F. Long, H. Fan, L. Liu, and Y. Wang, “A novel YOLOv3-tiny network for unmanned airship obstacle detection,” IEEE 8th Data Driven Control Learn. Syst. Conf. DDCLS, pp. 277–281, 2019, doi: 10.1109/DDCLS.2019.8908875.

[12] N. Dalal and B. Triggs, “Histograms of Oriented Gradients for Human Detection,” IEEE CVPR, vol. 1, pp. 886–893, 2005, doi: 10.1109/CVPR.2005.177.

[13] C. Szegedy, W. Liu, Y. Jia, and P. Sermanet, “Going Deeper with Convolutions,” CVPR, 2015, doi: 10.1108/978-1-78973-723320191012.

[14] J. R. R. Uijlings, K. E. A. Van De Sande, T. Gevers, and A. W. M. Smeulders, “Selective search for object recognition,” Int. J. Comput. Vis., vol. 104, no. 2, pp. 154–171, 2013, doi: 10.1007/s11263-013-06205.

[15] Z. Q. Zhao, P. Zheng, S. T. Xu, and X. Wu, “Object Detection with Deep Learning: A Review,” IEEE Trans. Neural Networks Learn. Syst., vol. 30, no. 11, pp. 3212–3232, 2019, doi: 10.1109/TNNLS.2018.2876865.

[16] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition,” ECCV, pp. 346–361, 2014, doi: 10.1023/B:KICA.0000038074.96200.69.

[17] R. Nabati and H. Qi, “RRPN : RADAR REGION PROPOSAL NETWORK FOR OBJECT DETECTION IN AUTONOMOUS VEHICLES,” IEEE Int. Conf. Image Process., pp. 3093–3097, 2019.

[18] L. Jiao et al., “A Survey of Deep Learning-Based Object Detection,” IEEE Access, vol. 7, pp. 128837–128868, 2019, doi: 10.1109/access.2019.2939201.

[19] D. Wang, C. Li, S. Wen, X. Chang, S. Nepal, and Y. Xiang, “Daedalus: Breaking Non-Maximum Suppression in Object Detection via Adversarial Examples,” arXiv Prepr., 2019.

[20] C. Ning, H. Zhou, Y. Song, and J. Tang, “Inception Single Shot MultiBox Detector for object detection,” IEEE Int. Conf. Multimed. Expo Work. ICMEW, no. July, pp. 549–554, 2017, doi: 10.1109/ICMEW.2017.8026312.

[21] Z. Chen, R. Khemmar, B. Decoux, A. Atahouet, and J. Y. Ertaud, “Real time object detection, tracking, and distance and motion estimation based on deep learning: Application to smart mobility,” 8th Int. Conf. Emerg. Secur. Technol. EST, pp. 1–6, 2019, doi: 10.1109/EST.2019.8806222.

[22] D. Xiao, F. Shan, Z. Li, B. T. Le, X. Liu, and X. Li, “A Target Detection Model Based on Improved Tiny-Yolov3 Under the Environment of Mining Truck,” IEEE Access, vol. 7, pp. 123757–123764, 2019, doi: 10.1109/access.2019.2928603.

[23] Q. C. Mao, H. M. Sun, Y. B. Liu, and R. S. Jia, “Mini-YOLOv3: RealTime Object Detector for Embedded Applications,” IEEE Access, vol. 7, pp. 133529–133538, 2019, doi: 10.1109/ACCESS.2019.2941547.

[24] W. Fang, L. Wang, and P. Ren, “Tinier-YOLO: A Real-time Object Detection Method for Constrained Environments,” IEEE Access, vol. 8, pp. 1935–1944, 2019, doi10.1109/ACCESS.2019.2961959.

[25] R. Huang, J. Pedoeem, and C. Chen, “YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers,” IEEE Int. Conf. Big Data, Big Data, pp. 2503–2510, 2019, doi: 10.1109/BigData.2018.8621865.

[26] J. Huang et al., “Speed/accuracy trade-offs for modern convolutional object detectors,” 30th IEEE

Conf. Comput. Vis. Pattern Recognition, CVPR, vol. 2017-Janua, pp. 3296–3305, 2017, doi: 10.1109/CVPR.2017.351.

[27] B. Benjdira, T. Khursheed, A. Koubaa, A. Ammar, and K. Ouni, "Car Detection using Unmanned Aerial Vehicles: Comparison between Faster R-CNN and YOLOv3," 1st Int. Conf. Unmanned Veh. Syst., pp. 1–6, 2019, doi: 10.1109/UVS.2019.8658300.

CHAPTER 7

PUBLICATION

[1] Pranav Adarsh, Pratibha Rathi, Dr. Manoj Kumar. "YOLO v3-Tiny: Object Detection and Recognition using one stage improved model", in Proceedings of the 6th International Conference on Advanced Computing and Communication Systems (ICACCS), IEEE, 2020.

Page No. 687-694.

978-1-7281-5197-7/20/$31.00 ©2020 IEEE

Object detection has seen many changes in algorithms to improve performance both on speed and accuracy. By the continuous effort of so many researchers, deep learning algorithms are growing rapidly with an improved object detection performance. Various popular applications like pedestrian detection, medical imaging, robotics, self-driving cars, face detection, etc. reduces the efforts of humans in many areas. Due to the vast field and various state-of-the-art algorithms, it is a tedious task to cover all at once. This paper presents the fundamental overview of object detection methods by including two classes of object detectors. In two stage detector covered algorithms are RCNN, Fast RCNN, and Faster RCNN, whereas in one stage detector YOLO v1, v2, v3, and SSD are covered. Two stage detectors focus more on accuracy, whereas the primary concern of one stage detectors is speed. We will explain an improved YOLO version called YOLO v3-Tiny, and then its comparison with previous methods for detection and recognition of object is described graphically.