OBJECT TRACKING FROM VIDEO INPUT USING DEEP LEARNING

A project report submitted

in partial fulfilment for the award of the Degree of

Bachelor of Technology
in
Computer Science and Engineering
by

KOLLURI ANUDEEP	**(U18CN307)**
SETTYBOYANA SIVA	**(U18CN266)**
TALARI SIVA KOTESWARA RAO	**(U18CN268)**
KOTHA SAI GOPAL	**(U18CN278)**

Under the guidance of

Dr. B. Persis Urbana Ivy. M.E., P.hd.,

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

BHARATH INSTITUTE OF HIGHER EDUCATION AND RESEARCH

(Deemed to be University Estd u/s 3 of UGC Act, 1956)

CHENNAI 600 073, TAMILNADU, INDIA

April , 2022

CERTIFICATE

This is to certify that the project report entitled to **Object Tracking From Video Input using Deep Learning** submitted by Kolluri Anudeep (U18CN307), Settyboyana Siva (U18CN266), Talari Siva Koteswara Rao (U18CN268) & Kotha Sai Gopal (U18CN278) to the Department of Computer Science and Engineering, Bharath Institute of Higher Education and Research, in partial fulfillment for the award of the degree of B. Tech in (Computer Science and Engineering) is a bonafide record of project work carried out by them under my supervision. The contents of this report, in full or in parts, have not been submitted to any other Institution or University for the award of any other degree.

Dr. B,Persis Urbana Ivy
Department of Computer Science & Engineering,
School of Computing,
Bharath Institute of Higher Education and Research
April, 2022

Dr. B. Persis Urbana Ivy

Professor & Head Department of Computer Science & Engineering,

School of computing,

Bharath Institute of Higher Education & Research

April, 2022

INTERNAL EXAMINER EXTERNAL EXAMINER

DECLARATION

We declare that this project report titled **Object Tracking from video Input using Deep Learning** submitted in partial fulfilment of the degree of B. Tech in (Computer Science and Engineering) is a record of original work carried out by us under the supervision of **Dr. B. Persis Urbana Ivy** , and has not formed the basis for the award of any other degree or diploma, in this or any other Institution or University. In keeping with the ethical practice in reporting scientific information, due acknowledgments have been made wherever the findings of others have been cited.

KOLLURI ANUDEEP
(U18CN307)

SETTYBOYANA SIVA
(U18CN266)

TALARI SIVA KOTESWARA RAO
(U18CN278)

KOTHA SAI GOPAL
(U18CN278)

Chennai
23-04-2022

ACKNOWLEDGMENTS

First, we wish to thank the almighty who gave us good health and success throughout our project work.

We express our deepest gratitude to our beloved President **Dr. J. Sundeep Aanand,** and Managing Director **Dr.E. Swetha Sundeep Aanand** for providing us the necessary facilities for the completion of our project.

We take great pleasure in expressing sincere thanks to Vice Chancellor (I/C) **Dr. K. Vijaya Baskar Raju,** Pro Vice Chancellor (Academic) **Dr. M. Sundararajan**, Registrar **Dr. S. Bhuminathan** and Additional Registrar **Dr. R. Hari Prakash** for backing us in this project. We thank our Dean Engineering **Dr. J. Hameed Hussain** for providing sufficient facilities for the completion of this project.

We express our immense gratitude to our Academic Coordinator **Mr. G. Krishna Chaitanya** for his eternal support in completing this project.

We thank our Dean, School of Computing **Dr. S. Neduncheliyan** for his encouragement and the valuable guidance. We record indebtedness to our Head, Department of Computer Science and Engineering **Dr. B. Persis Urbana Ivy** for immense care and encouragement towards us throughout the course of this project.

We also take this opportunity to express a deep sense of gratitude to our Supervisor **Dr. B. Persis Urbana Ivy,** for her cordial support, valuable information and guidance, she helped us in completing this project through various stages.

We thank our department faculty, supporting staff and friends for their help and guidance to complete this project.

KOLLRUI ANUDEEP	**(U18CN307)**
SETTYBOAYANA SIVA	**(U18CN266)**
TALARI SIVA KOTESWARA RAO	**(U18CN268)**
KOTHA SAI GOPAL	**(U18CN278)**

ABSTRACT

Real time object detection is an immense, vibrant and complex area of computer vision. Assuming there is a single object to be distinguished in an image, it is known as Image Localization and in the event that there are various objects in an image, then, at that point, it is Object Detection. Mobile networks and binary neural networks are the most generally involved techniques for current deep learning models to perform different tasks on embedded systems. In this paper, we develop a method to distinguish an item thinking about the deep learning pre-prepared model MobileNet for Single Shot Multi-Box Detector (SSD). This algorithm is used for real-time detection and for webcam streaming to detect object in a video stream. Subsequently, we utilize an object detection module that can identify what is in the video stream. To carry out the module, we join the MobileNet and the SSD framework for a quick and efficient deep learning-based strategy for object identification.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

RCNN	-	Region based Convolution Neural Network
CNN	-	Convolution Neural Network
SSD	-	Single Shot Multi Box Detector
YOLO	-	You Look Only Once
CV	-	Computer Vision
ML	-	Machine Learning
DL	-	Deep Learning
GPU	-	Graphics Processing Unit
GUI	-	Graphical User Interface
IDLE	-	Integrated Development and Learning Environment
OS	-	Operating System

CHAPTER 1

INTRODUCTION

Object detection is one of the most important fields of exploration in computer vision today. It is an augmentation of image classification the objective is to identify one or more classes of objects in a picture and with the help of bounding boxes locate their presence. Consequently, object detection carries an important role in many real-world applications like image recovery and video surveillance. The main purpose of our analysis is to elaborate the accuracy of an object detection technique SSD and the pre trained deep learning model MobileNet and additionally feature a portion of the notable elements that make this method stand out. The trial results show that the Average Precision (AP) of the algorithm to recognize various classes as vehicle, person and chair is 99.76%, 97.76% and 71.07%, separately. This improves the accuracy of behavior detection at a handling speed which is needed for the real-time location and the necessities of day by day observing indoor and outside. The mix of MobileNet into the SSD framework forms one of the center parts of our work. However, MobileNet with the effective SSD framework has been a hot exploration point in recent times, to a great extent because of managing the functional limits of running strong neural nets on low-end devices like cell phones/laptops to additionally expand the horde of conceivable outcomes with respect to real-time applications

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of *building models of data*.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models *tunable parameters* that can be adapted to observed data; in this way the program can be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain.Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into *classification* tasks and *regression* tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as *clustering* and *dimensionality reduction.* Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven't surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, "to make decisions, based on data, with efficiency and scale".

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programing logic, in the problems that cannot be programmed inherently. The fact is that we can’t do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

Image processing is the process of transforming an image into a digital form and performing certain operations to get some useful information from it. The image processing system usually treats all images as 2D signals when applying certain predetermined signal processing methods

Image Processing:

There are five main types of image processing:

- Visualization - Find objects that are not visible in the image
- Recognition - Distinguish or detect objects in the image
- Sharpening and restoration - Create an enhanced image from the original image
- Pattern recognition - Measure the various patterns around the objects in the image
- Retrieval - Browse and search images from a large database of digital images that are similar to the original image

Image processing is often regarded as improperly exploiting the image in order to achieve a level of beauty or to support a popular reality. However, image processing is most accurately described as a means of translation between a human viewing system and digital imaging devices. The human viewing system does not see the world in the same way as digital cameras, which have additional sound effects and bandwidth. Significant differences between human and digital detectors will be demonstrated, as well as specific processing steps to achieve translation. Image editing should be approached in a scientific way so that others can reproduce, and validate human results. This includes recording and reporting processing actions and applying the same treatment to adequate control images.

OpenCV is a cross-platform library using which we can develop real-time **computer vision applications**. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection. In this tutorial, we explain how you can use OpenCV in your applications.

OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. All of the new developments and algorithms appear in the C++ interface. There are bindings in Python, Java and MATLAB/OCTAVE. The API for these interfaces can be found in the online documentation. Wrappers in several programming languages have been developed to encourage adoption by a wider audience. In version 3.4, JavaScript bindings for a selected subset of OpenCV functions was released as OpenCV.js, to be used for web platforms.

Tkinter is a python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's defacto standard GUI. Tkinter is included with standard GNU/LINUX, Microsoft Windows and macOS installs of Python. Tkinter is implemented as a Python wrapper around a complete TCL interpreter embedded in the Python interpreter. Tkinter calls are translated into Tcl commands, which are fed to this embedded interpreter, thus making it possible to mix Python and Tcl in a single application.

Applications of Object Detection in Real life:

Object detection is breaking into a wide scope of enterprises, with use cases extending from individual security to efficiency in the working environment. Object detection is applied in numerous territories of image processing, including picture retrieval, security, observation, computerized vehicle systems and machine investigation. Critical difficulties remain in the field of object detection. The potential outcomes are inestimable with regards to future use cases for object detection.

Tracking objects

An item/object detection framework is additionally utilized in tracking the objects, for instance tracking a ball during a match in the football world cup, tracking the swing of a cricket bat, tracking an individual in a video.

Object tracking has an assortment of uses, some of which are surveillance and security, traffic checking, video correspondence, robot vision and activity.

People Counting

Object detection can be additionally utilized for People counting.It is utilized for dissecting store execution or group measurements during festivals. These will, in general, be progressively troublesome as individuals move out of the frame rapidly (likewise in light of the fact that individuals are non-inflexible objects).

Automated CCTV surveillance

Surveillance is a necessary piece of security and watch. Ongoing advances in computer vision innovation need to prompt the improvement of different programmed surveillance systems. Be that as it may, their viability is influenced by numerous factors and they are not totally dependable. This examination researched the capability of an automated surveillance system to diminish the CCTV administrator outstanding task at hand in both discovery and following exercises.

Typically CCTV is running inevitably, so we need a huge size of the memory framework to store the recorded video. By utilizing an object discovery framework we can mechanize CCTV so that in the event that a few items are detected, at that point the record is going to begin. Utilizing this we can diminish the over and over account a similar picture outlines, which expands memory effectiveness. We can diminish the memory prerequisite by utilizing this object detection system.

Person Detection

Person detection is necessary and critical work in any intelligent video surveillance framework, as it gives the essential data to semantic comprehension of the video recordings. It has a conspicuous augmentation to automotive applications because of the potential for improving security frameworks. Person detection is undertakings of Computer vision frameworks for finding and following individuals. Person detection is the task of finding all examples of individuals present in a picture, and it has been most broadly achieved via looking through all areas in the picture, at all potential scales, and contrasting a little region at every area with known layouts or examples of individuals. Person detection is commonly viewed as the initial procedure in a video surveillance pipeline and can take care of into more significant level thinking modules, for example, action recognition and dynamic scene analysis.

Vehicle Detection

Vehicle Detection is one of the most important part in our daily life. As the world is moving faster and the numbers of cars are keep on increasing day by day, Vehicle detection is very important. By using Vehicle Detection technique we can detect the number plate of a speeding car or accident affected car. This also enables for security of the society and decreasing the number of crimes done by car. By using Vehicle Detection Technology Pixel Solutionz have successfully detected the speed of the vehicle and we have also detected the number plate of the car using Optical Character Recognition (OCR). By detecting the Number plate, Pixel Solutionz managed to measure the speed of the vehicle and for and oil company we have successfully developed a Safety Alert System with collision detection warning alert.

CHAPTER 2

LITERATURE SURVERY

2.1 SSD: Single Shot MultiBox Detector

Authors: Wei Liu and Alexander C. Berg

We present a method for detecting objects in images using a single deep neural network. Our approach, named SSD, discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. Additionally, the network combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes. Our SSD model is simple relative to methods that require object proposals because it completely eliminates proposal generation and subsequent pixel or feature resampling stage and encapsulates all computation in a single network. This makes SSD easy to train and straightforward to integrate into systems that require a detection component. Experimental results on the PASCAL VOC, MS COCO, and ILSVRC datasets confirm that SSD has comparable accuracy to methods that utilize an additional object proposal step and is much faster, while providing a unified framework for both training and inference. Compared to other single stage methods, SSD has much better accuracy, even with a smaller input image size. For 300×300 input, SSD achieves 72.1% mAP on VOC2007 test at 58 FPS on a Nvidia Titan X and for 500×500 input, SSD achieves 75.1% mAP, outperforming a comparable state of the art Faster R-CNN model. Code is available.
MobileNets: Efficient

2.2 Convolutional Neural Networks for Mobile Vision Applications

Authors: Andrew G. Howard, and Hartwig Adam

We present a class of efficient models called MobileNets for mobile and embedded vision applications. MobileNets are based on a streamlined architecture that uses depth-wise separable convolutions to build light weight deep neural networks. We introduce two simple global hyper-parameters that efficiently trade off between latency and accuracy. These hyper-parameters allow the model builder to choose the right sized model for their application based on the constraints of the problem. We present extensive experiments on resource and accuracy tradeoffs and show strong performance compared to other popular models on ImageNet classification. We then demonstrate the effectiveness of MobileNets across a wide range of applications and use cases including object detection, finegrain classification, face attributes and large scale geo-localization.

2.3 Ammunition Detection: Developing a Real-Time Gun Detection Classifier

Authors: Justin Lai, Sydney Maples

Using real-time object detection to improve surveillance methods is a promising application of Convolutional Neural Networks (CNNs). One particular application is the detection of hand-held weapons (such as pistols and rifles). Thus far, previous work has mostly focused on weapon-based detection within infrared data for concealed weapons. By contrast, we are particularly interested in the rapid detection and identification of weapons from images and surveillance data. For our project, we used a Tensorflow-based implementation of the Overfeat network as an integrated network for detecting and classifying weapons in images. Our best performance was achieved on Overfeat-3 with 93% training accuracy and 89% test accuracy with adjusted hyperparameters.

2.4 UAV: Application of Object Detection and Tracking Techniques for Unmanned Aerial Vehicles

Authors: Shreyamsh Kamate

In this research, the information captured by Unmanned Aerial Vehicles (UAVs) are eminently utilized in detecting and tracking moving objects which pose a primary security threat against the United States southern border. Illegal trespassing and border encroachment by immigrants is a huge predicament against the United States border security force and the Department of Homeland Security. It becomes insurmountable to warranty suspicious behaviour, monitoring by human operators for long periods of time, due to the massive amount of data involved. The main objective of this research is to assist the human operators, by implementing intelligent visual surveillance systems which help in detecting and tracking suspicious or unusual events in the video sequence. The visual surveillance system requires fast and robust methods of detecting and tracking moving objects. In this research, we have investigated methods for detecting and tracking objects from UAVs. Moving objects were detected using adaptive background subtraction technique successfully and these detected objects were tracked by using Lucas-Kanade optical flow tracking, Continuously Adaptive Mean-Shift tracking based techniques. The simulation results show the efficacy of these techniques in detecting and tracking moving objects in the video sequences acquired by the UAV.

2.5 Object detection with deep learning and OpenCV

Authors: Adrian Rosebrock

The object detection works on the Viola-Jones algorithm, which was proposed by Paul Viola and Michael Jones. The aforementioned algorithm is based on machine learning. The first step involves training a cascade function with a large amount of negative and positive labeled images. Once the classifier is trained, identifying features, namely “HAAR Features,” are extracted from these training images. HAAR features are essentially rectangular features with regions of bright and dark pixels.

2.6 Elegant and efficient algorithms for real time object detection, counting and classification for video surveillance applications from single fixed camera

Authors: Mohana and H. V. R. Aradhya

Video Surveillance is very important and essential task for security applications. Earlier surveillance was like capturing a video from camera, storing the information in a database, and then required contents were accessed manually from the database. It may lead to loss of sensitive information in real time. In such cases automated video surveillance is very essential. In automated video surveillance, object detection and tracking can be done in real time, and it finds the required information, also informs to the administrator in real time. This paper describes the detection of objects in real time and counts the number of objects. It also describes the objects classification; it is classified in to five predefined classes namely human beings, cars, motor bikes, busses and horses by the method of features extraction and Comparision.

2.7 Object Tracking Algorithms for video surveillance applications

Authors: Akshay Mangawati, Mohana, Mohammed Leesan, H. V. Ravish Aradhya

Object tracking is one of the most critical areas of research due to change in motion of object. Specifically, feature selection places a vital role in object tracking. It has wide range of applications such as object detection, traffic control, human-computer interaction, gesture recognition, video surveillance. Many approaches focus on the tracking algorithm to smoothen the video sequence in order to overcome the issues of tracking, related to object movement and appearance. These methods uses object shape, colour, texture, object of interest and motion in multi direction in tracking for video surveillance applications. This paper elaborate the exhaustive survey of various object tracking algorithms under different environmental conditions and identified efficient algorithm in various types of tracking. In this paper objects are tracked based on colour, motion of single and multiple objects (vehicles) are detected and counted in multiple frames. Further single algorithm may be designed for object tracking by considering shape, colour, texture, object of interest, motion of object in multi direction.

2.8 Feature extraction using Convolution Neural Networks (CNN) and Deep Learning

Authors: Manjunath Jogin, Mohana

The Image classification is one of the preliminary processes, which humans learn as infants. The fundamentals of image classification lie in identifying basic shapes and geometry of objects around us. It is a process which involves the following tasks of pre-processing the image (normalization), image segmentation, extraction of key features and identification of the class. The current image classification techniques are much faster in run time and more accurate than ever before, they can be used for extensive applications including, security features, face recognition for authentication and authorization, traffic identification, medical diagnosis and other fields. The idea of image classification can be solved by different approaches. But the machine learning algorithms are the best among them. These algorithms are based on the idea proposed years ago, but couldn't be implemented due to lack of computational power. With the idea of deep learning, the models are trained better and are able to identify different levels of image representation. The convolutional neural networks revolutionized this field by learning the basic shapes in the first layers and evolving to learn features of the image in the deeper layers, resulting in more accurate image classification. The idea of Convolutional neural network was inspired by the hierarchical representation of neurons by Hubel and Wiesel in 1962, their work was based on the study of stimuli of the visual cortex in cat. It was a fundamental breakthrough in the field of computer vision in understanding the working of visual cortex in humans and animals. In this paper feature of an images is extracted using convolution neural network using the concept of deep learning.

Further classification algorithms are implemented for various applications.

2.9 YOLO based Detection and Classification of Objects in video records

Authors: Arka Prava Jana, Abhiraj Biswas, Mohana

The primitive machine learning algorithms that are present break down each problem into small modules and solve them individually. Nowadays requirement of detection algorithm is to work end to end and take less time to compute. Real-time detection and classification of objects from video records provide the foundation for generating many kinds of analytical aspects such as the amount of traffic in a particular area over the years or the total population in an area. In practice, the task usually encounters slow processing of classification and detection or the occurrence of erroneous detection due to the incorporation of small and lightweight datasets. To overcome these issues, YOLO (You Only Look Once) based detection and classification approach (YOLOv2) for improving the computation and processing speed and at the same time efficiently identify the objects in the video records. The classification algorithm creates a bounding box for every class of objects for which it is trained, and generates an annotation describing the particular class of object. The YOLO based detection and classification (YOLOv2) use of GPU (Graphics Processing Unit) to increase the computation speed and processes at 40 frames per second.

CHAPTER 3

EXISTING SYSTEM AND PROPOSED SYSTEM

3.1 Existing System

In the existing system they use mat lab for image processing and object detection. MATLAB is a programming language that combines a desktop environment tuned for iterative analysis and design processes with a programming language that expresses matrix and array mathematics directly. It includes the Live Editor for creating scripts that combine code, output, and formatted text in an executable notebook. MATLAB is interpreted language and hence it takes more time to execute than other compiled languages such as C, C++. It is expensive than regular C or Fortran compiler. Individuals find it expensive to purchase. It requires fast computer with sufficient amount of memory. This adds to the cost for individuals willing to use it for programming. It is difficult to develop real time applications using MATLAB as it sits "on top" of windows. It is not free and hence users need to obtain licensed version from MathWorks, Inc. Hence we use OpenCV in our proposed system.

3.1.1 Disadvantages of Existing System

- Very Time taking Process
- Accuracy is very low
- Cost for project design is very high
- Computational complexity is high

3.2 Proposed System

In the Proposed System, we are going to detect objects in real time with the help of Mobilenet-SSD model in fast and efficient way. We will create the Python script for object detection using deep neural network with OpenCV.

Working of the system is as follow: Input will be given through Realtime video by camera or webcam, based on streamlined MobileNet Architecture which uses depth-wise separable convolutions to build light weight deep neural Networks. The input video divided into frames and pass it to MobileNet layers. Each feature value is determined as a difference between the amount of pixel intensity under the bright region and the pixel intensity under the dark area. Every one of the possible sizes and area of the image is utilized to compute these elements. An image may contain irrelevant features and few relevant characteristics that can be used to detect the object. The job of the MobileNet layers is to change over the pixels from the input image into highlights that describe the contents of the image. Then it passes to MobileNet-SSD model to determine the bounding boxes and corresponding class(label) of objects. After that the only last step is to show or display the Output.

3.2.1 Advantages of Proposed System

- Fast (gives around 60 fps)
- High Accuracy
- Easy to implement
- Light weight
- Can be implemented in Embedded systems

3.3 System Architecture

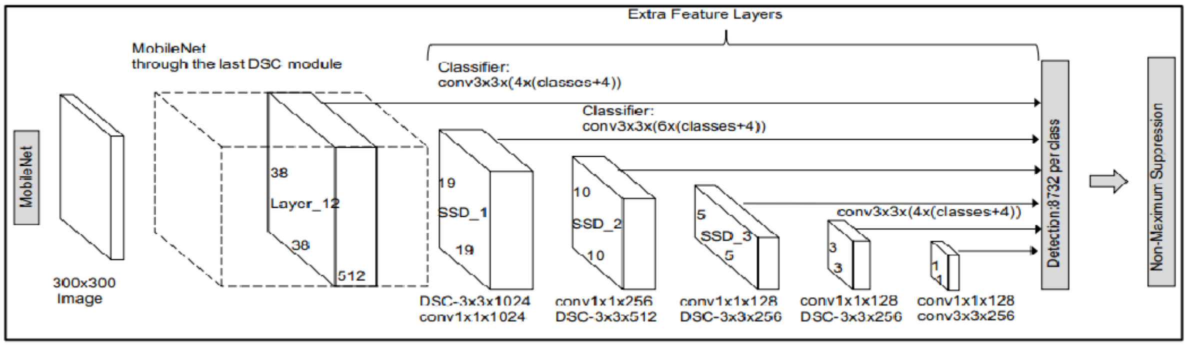


Fig 1.1 Model Architecture

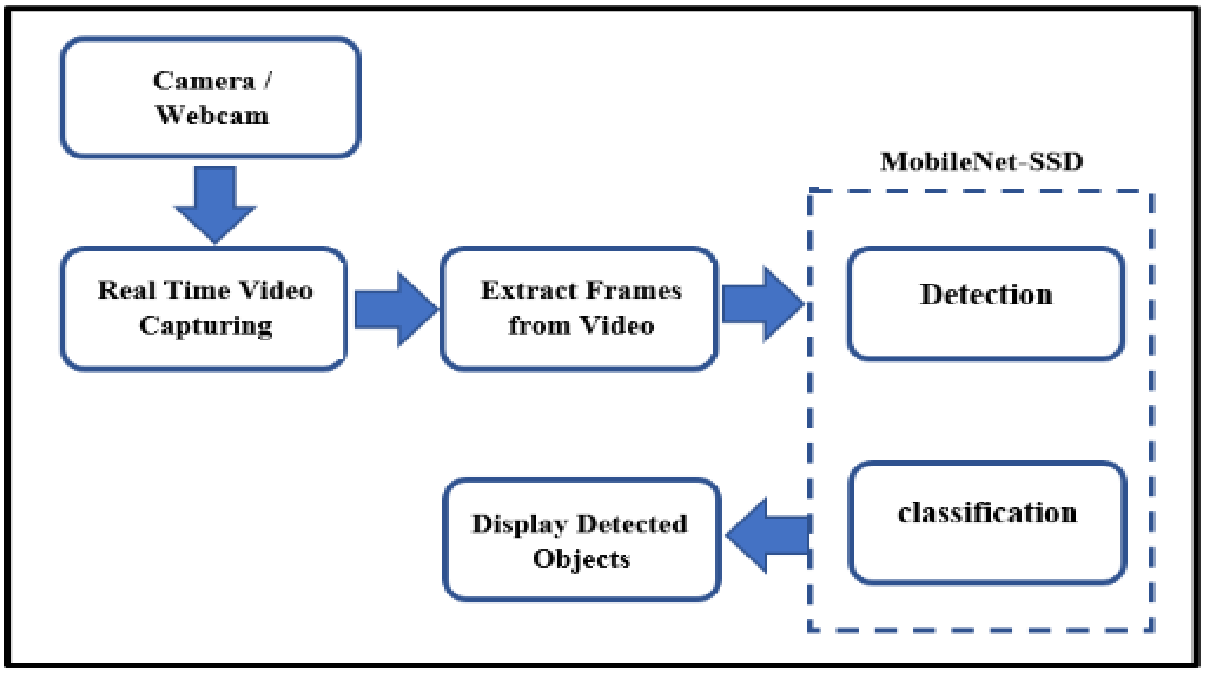


Fig 1.2 Block Diagram

CHAPTER 4

HARDWARE AND SOFTWARE REQUIREMENTS

4.1 Hardware Requirements

- **Processor** - Pentium –III
- **Speed** – 2.4 GHz
- **RAM** - 512 MB (min)
- **Hard Disk** - 20 GB
- **Floppy Drive -** 1.44 MB
- **Key Board** - Standard Keyboard
- **Monitor** – 15 VGA Colour

4.2 Software Requirements

- Python IDLE 3.7 version (or)
- Anaconda 3.7 (or)
- Jupyter Notebook (or)
- Google Colab

CHAPTER 5

IMPLEMENTATION

5.1 Module description

In the project first GUI is designed so that the project is interactive. This GUI is developed using Tkinter module. It is a module that comes with python. This module will take care of buttons, windows and popups.

After creating GUI, we create 2 functions, one is to track objects from an uploaded video and other from web cam. In either case, the video is split in multiple frames. Each frame is an input to a pre trained Mobilnet SSD model. Each frame is passed through the model and it detects the object and draws boundary boxes around it. It does this for each frame of the video. Since Mobilenet SSD is light weight, it can produce upto 60 fps which is equivalent to real time application.

MobileNet is a streamlined architecture that uses depthwise separable convolutions to construct lightweight deep convolutional neural networks and provides an efficient model for mobile and embedded vision applications. The structure of MobileNet is based on depthwise separable filters, as shown in Figure.

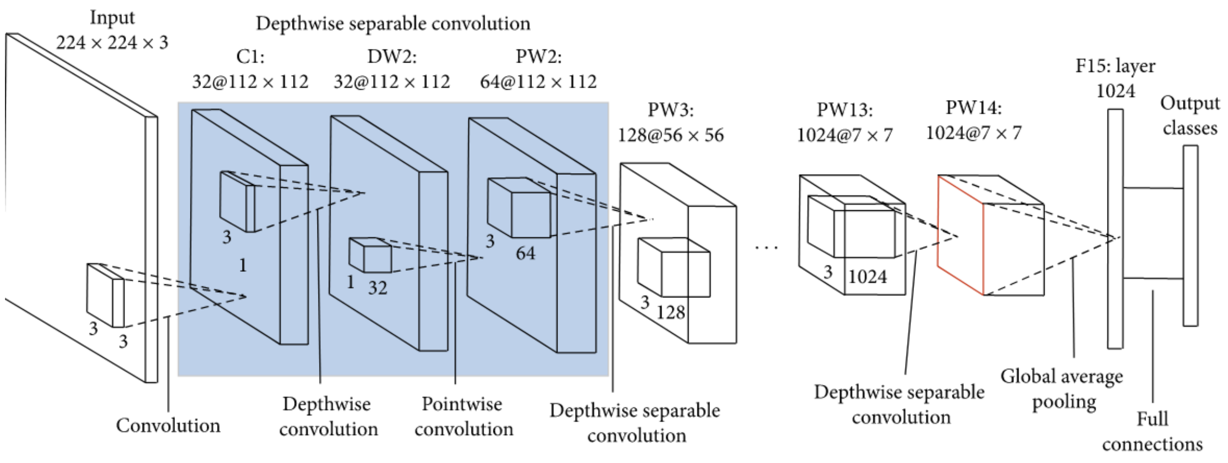


Fig 2.1 Mobilenets Architecture

Depthwise separable convolution filters are composed of depthwise convolution filters and point convolution filters. The depthwise convolution filter performs a single convolution on each input channel, and the point convolution filter combines the output of depthwise convolution linearly with $1 * 1$ convolutions.

DenseNet proposed a new connection mode, connecting each current layer of the network with the previous network layers, so that the current layer can take the output feature maps of all the previous layers as input features. To some extent, this kind of connection can alleviate the problem of gradient disappearance. Since each layer is connected with all the previous layers, the previous features can be repeatedly used to generate more feature maps with less convolution kernel.

DenseNet takes dense blocks as basic unit modules, a dense block structure consists of 4 densely connected layers with a growth rate of 4. Each layer in this structure takes the output feature maps of the previous layers as the input feature maps. Different from the residual unit in ResNet, which combines the sum of the feature maps of the previous layers in one layer, the dense block transfers the feature maps to all the subsequent layers, adding the dimension of the feature maps rather than adding the pixel values in the feature maps.

Single Shot detector like YOLO takes only one shot to detect multiple objects present in an image using multibox. It is significantly faster in speed and high-accuracy object detection algorithm. A quick comparison between speed and accuracy of different object detection models on VOC2007

- **SDD300** : 59 FPS with mAP 74.3%
- **SSD500 :** 22FPS with mAP 76.9%
- **Faster R-CNN** : 7 FPS with mAP 73.2%
- **YOLO** : 45 FPS with mAP 63.4%

High detection accuracy in SSD is achieved by using multiple boxes or filters with different sizes, and aspect ratio for object detection. It also applies these filters to multiple feature maps from the later stages of a network. This helps perform detection at multiple scales. To the base VGG network, we add additional convolutional layers for detection. Convolutional layers at the end of the base network decrease in size progressively. This helps with detection of objects at multiple scales. The convolutional model for detection is different for each feature layer.

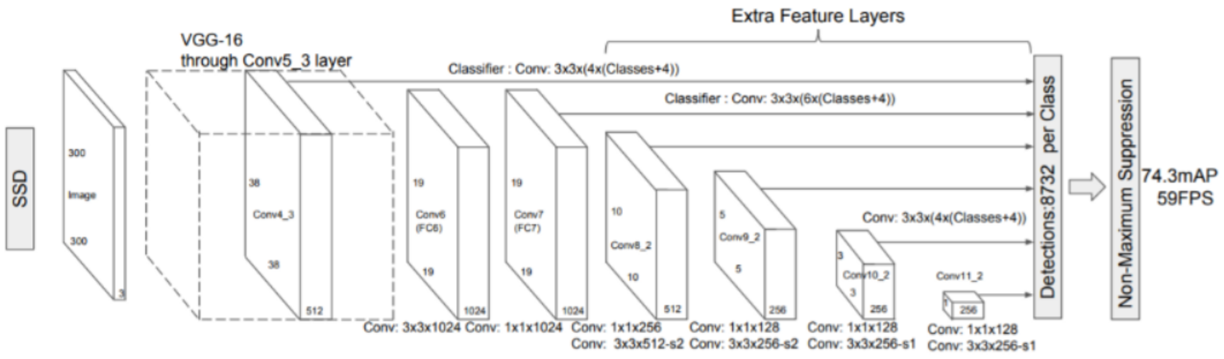


Fig 2.2 SSD Model Architecture

Progressively decreasing convolutional layers decreases the feature map size and increase the depth. The deep layers cover larger receptive fields and construct more abstract representations. This is helpful is detecting larger objects. Initial conv layers cover smaller receptive fields and are helpful in detecting smaller objects present in the image.

VGG-16 is the base network that performs the feature extraction. Conv layers evaluate boxes of different aspect ratios at each location in several feature maps with different scales. Multiboxes are like anchors of Fast R-CNN. We have multiple default boxes of different sizes, aspect ratio across the entire image as shown below. SSD uses 8732 boxes. This helps with finding the default box that most overlaps with the ground truth bounding box containing objects.

SSD is designed for object detection in real-time. Faster R-CNN uses a region proposal network to create boundary boxes and utilizes those boxes to classify objects. While it is considered the start-of-the-art in accuracy, the whole process runs at 7 frames per second. Far below what real-time processing needs. SSD speeds up the process by eliminating the need for the region proposal network. To recover the drop in accuracy, SSD applies a few improvements including multi-scale features and default boxes. These improvements allow SSD to match the Faster R-CNN's accuracy using **lower resolution images**, which further pushes the speed higher. According to the following comparison, it achieves the real-time processing speed and even beats the accuracy of the Faster R-CNN. (Accuracy is measured as the mean average precision mAP: the precision of the predictions.)

SSD uses **VGG16** to extract feature maps. Then it detects objects using the **Conv4_3** layer. For illustration, we draw the Conv4_3 to be 8 × 8 spatially (it should be 38 × 38). For each cell (also called location), it makes 4 object predictions. Each prediction composes of a boundary box and 21 scores for each class (one extra class for no object), and we pick the highest score as the class for the bounded object. Conv4_3 makes a total of 38 × 38 × 4 predictions: four predictions per cell regardless of the depth of the feature maps. As expected, many predictions contain no object. SSD reserves a class "0" to indicate it has no objects.

SSD does not use a delegated region proposal network. Instead, it resolves to a very simple method. It computes both the location and class scores using **small convolution filters**. After extracting the feature maps, SSD applies 3 × 3 convolution filters for each cell to make predictions. (These filters compute the results just like the regular CNN filters.) Each filter outputs 25 channels: 21 scores for each class plus one boundary box (detail on the boundary box later).

5.2 Browse System Videos: Using this module application allow user to upload any video from his system and application will connect to that video and start playing it, while playing if application detect any object then it will mark that object with bounding boxes, while playing video if user wants to stop tracking then he need to press ‘q’ key from keyboard to stop video playing. The detecte`d object is printed as log in the main window with confidence level. The confidence level is also shown on top of boundary boxes drawn by the SSD model.

The algorithm we used here is Mobilenet SSD which contains two parts, Mobilenet and SSD. MobileNet model is designed to be used in mobile applications, and it is TensorFlow’s first mobile computer vision model. MobileNet uses **depthwise separable convolutions.** It significantly **reduces the number of parameters** when compared to the network with regular convolutions with the same depth in the nets. This results in lightweight deep neural networks.

A depthwise separable convolution is made from two operations.

1. Depthwise convolution.
2. Pointwise convolution.

MobileNet is a class of CNN that was open-sourced by Google, and therefore, this gives us an excellent starting point for training our classifiers that are insanely small and insanely fast.

5.2.1 Depth-wise Separable Convolution:

This convolution originated from the idea that a filter’s depth and spatial dimension can be separated- thus, the name separable.

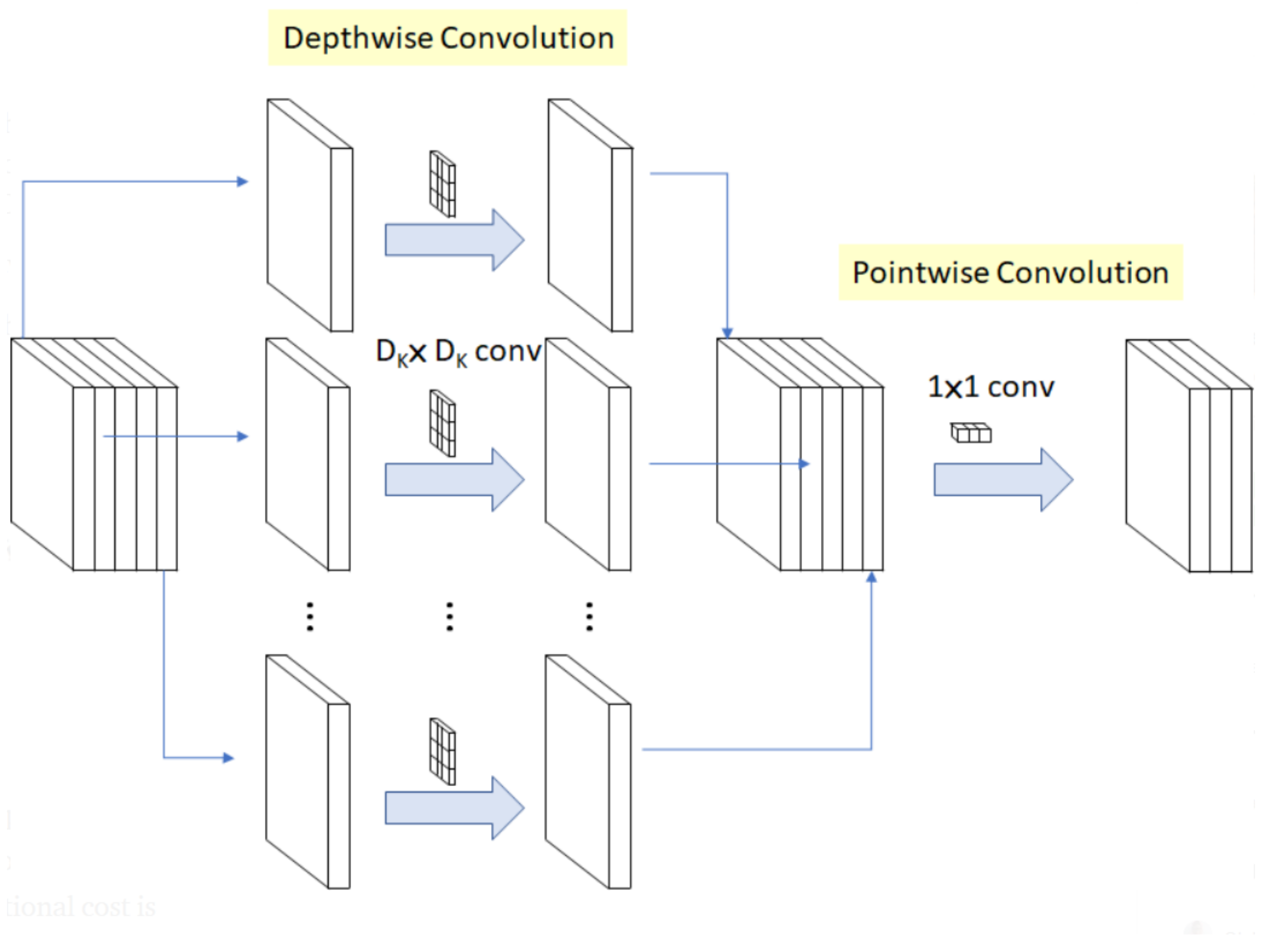


Fig 2.3 Depthwise Separable Convolution

1. **Depthwise convolution** is the **channel-wise DK×DK spatial convolution**. Suppose in the figure above, and we have five channels; then, we will have 5 DK×DK spatial convolutions.
2. **Pointwise convolution** is the **1×1 convolution** to change the dimension.
3. **Depthwise convolution.**

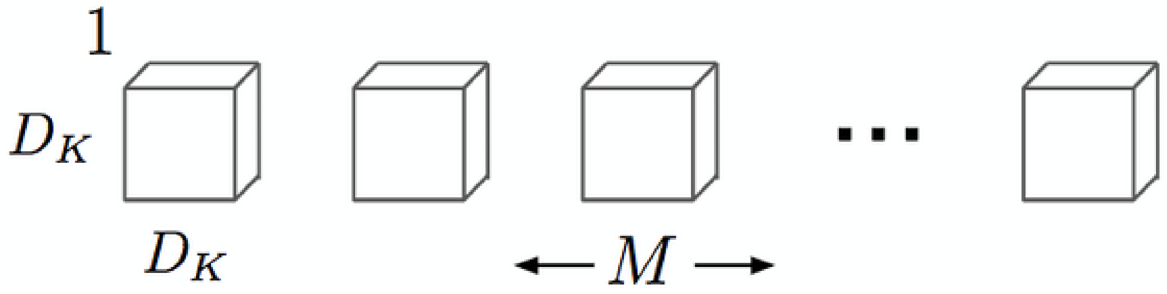


Fig 2.4 Depth Wise Separable Convolution Example

It is a map of a single convolution on each input channel separately. Therefore its number of output channels is the same as the number of the input channels. Its computational cost is

$$\mathbf{Df^2 * M * Dk^2}.$$

5.2.2 Pointwise convolution:

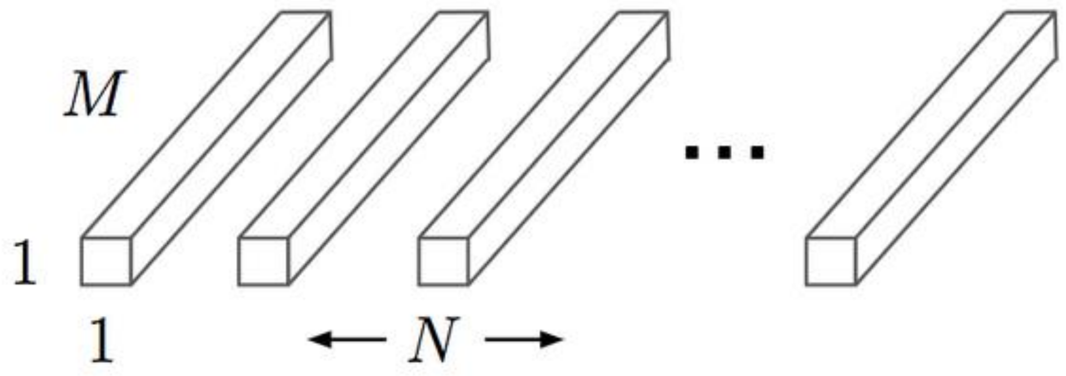


Fig 2.5 Point Wise Convolution

Convolution with a kernel size of 1x1 that simply combines the features created by the depthwise convolution. Its computational cost is

$$\mathbf{M * N * Df^2}$$

5.2.3 Mobilenet Layers:

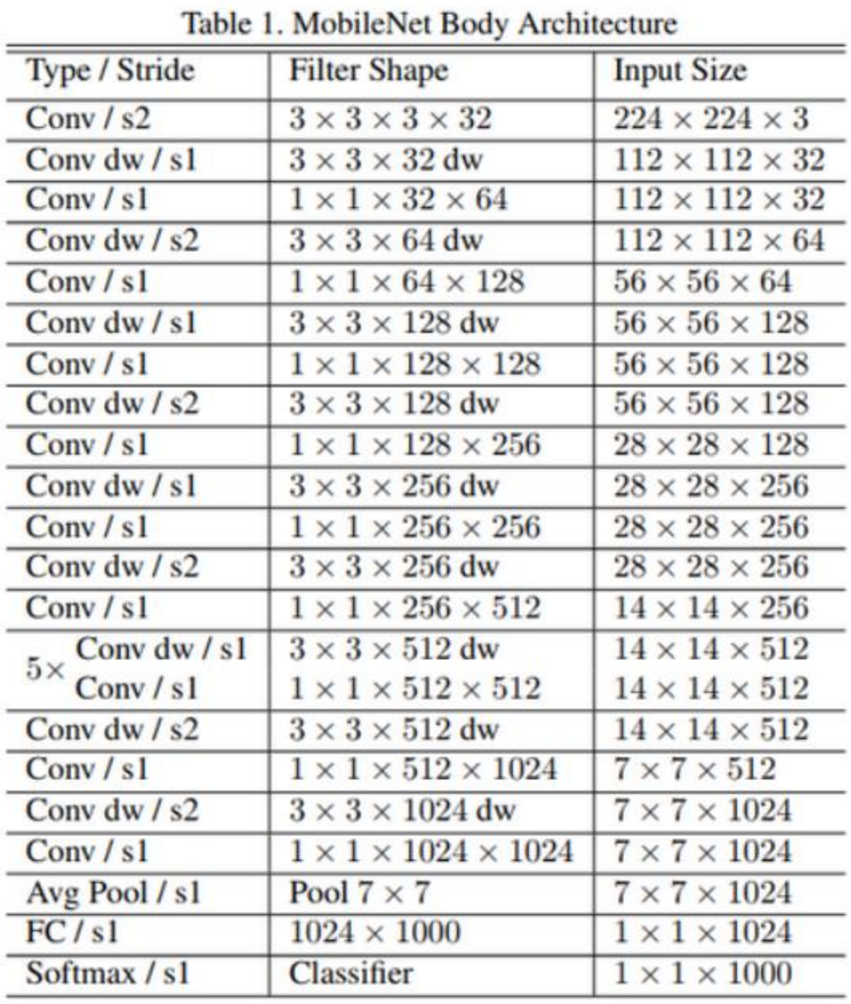


Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Fig 2.5 Mobilenet Layers

The dataset we used to train the model is Imagenet dataset:

- The ImageNet dataset is a very large collection of human annotated photographs designed by academics for developing computer vision algorithms.
- The ImageNet Large Scale Visual Recognition Challenge, or ILSVRC, is an annual competition that uses subsets from the ImageNet dataset and is designed to foster the development and benchmarking of state-of-the-art algorithms.
- The ILSVRC tasks have led to milestone model architectures and techniques in the intersection of computer vision and deep learning.

The **ImageNet** project is a large visual database designed for use in visual object recognition software research. More than 14 million images have been hand-annotated by the project to indicate what objects are pictured and in at least one million of the images, bounding boxes are also provided. ImageNet contains more than 20,000 categories with a typical category, such as "balloon" or "strawberry", consisting of several hundred images. The database of annotations of third-party image URLs is freely available directly from ImageNet, though the actual images are not owned by ImageNet. Since 2010, the ImageNet project runs an annual software contest, the ImageNet Large Scale Visual Recognition Challenge, where software programs compete to correctly classify and detect objects and scenes. The challenge uses a "trimmed" list of one thousand non-overlapping classes.

On 30 September 2012, a convolutional Neural Network (CNN) called AlexNet achieved a top-5 error of 15.3% in the ImageNet 2012 Challenge, more than 10.8 percentage points lower than that of the runner up. This was made feasible due to the use of Graphics Processing Unit (GPUs) during training, an essential ingredient of the deep learning revolution.

In 2015, AlexNet was outperformed by Microsoft's very deep CNN with over 100 layers, which won the ImageNet 2015 contest.

5.3 Start Webcam Video Tracking: Using this module application connect itself with inbuilt system webcam and start video streaming, while streaming if application detect any object then it will surround that object with bounding boxes, while playing press 'q' to stop web cam streaming. During this process, the OpenCV module takes the input from the webcam and converts the video into multiple frames. Each frame is fed into the Mobile Net SSD model. In the model, the image size is decreased to 300 x 300 image and then it is converted from rgb image to grey scale image using OpenCV library. This greyscale image is then fed into Mobilenet SSD where Object detection happens.

The bounding box regression technique of SSD is inspired by Szegedy's work on Multibox, a method for fast *class-agnostic* bounding box coordinate proposals. Interestingly, in the work done on MultiBox Inception-style convolutional network is used. The 1x1 convolutions that you see below help in dimensionality reduction since the number of dimensions will go down (but "width" and "height" will remain the same).

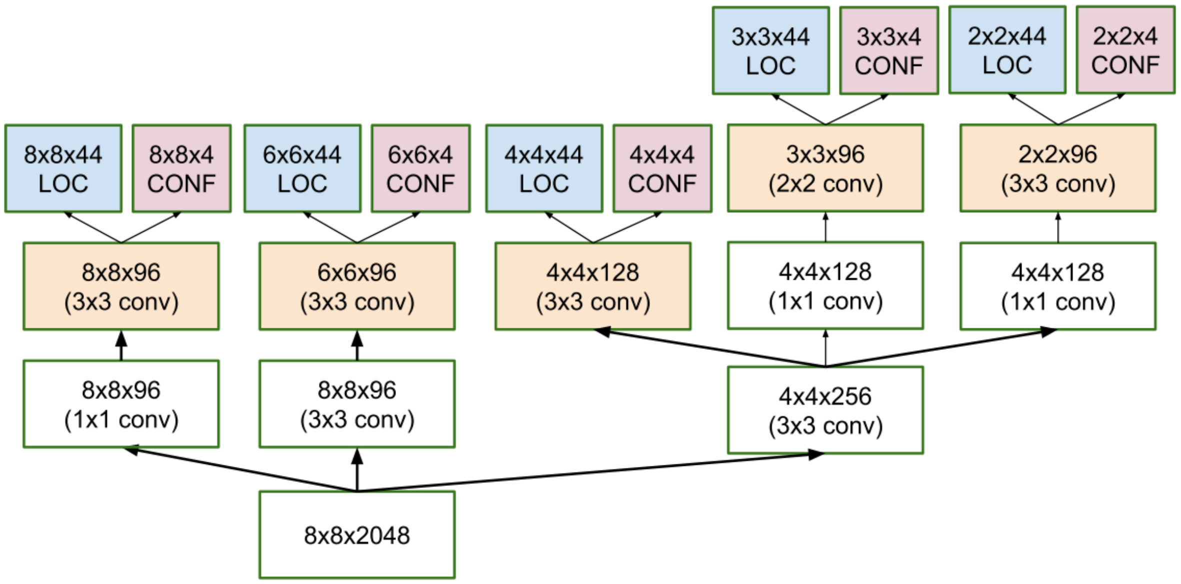


Fig 2.6 Architecture of multi-scale convolutional prediction of the location and confidences of multibox

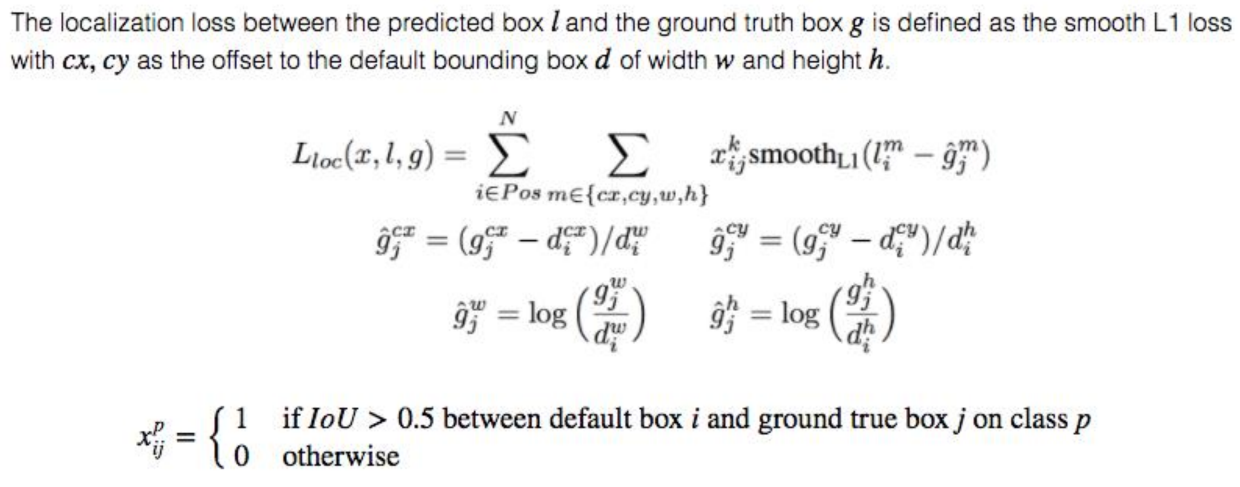
MultiBox's loss function also combined two critical components that made their way into SSD:

- **Confidence Loss**: this measures how confident the network is of the *objectness* of the computed bounding box. Categorical cross-entropy is used to compute this loss.
- **Location Loss:** this measures how *far away* the network's predicted bounding boxes are from the ground truth ones from the training set. L-2 Norm is used here.

Without delving too deep into the math (read the paper if you are curious and want a more rigorous notation), the expression for the loss, which measures how far off our prediction "landed", is thus:

multibox_loss = confidence_loss + alpha * location_loss

The *alpha* term helps us in balancing the contribution of the location loss. As usual in deep learning, the goal is to find the parameter values that most optimally reduce the loss function, thereby bringing our predictions closer to the ground truth.



The localization loss between the predicted box l and the ground truth box g is defined as the smooth L1 loss with cx, cy as the offset to the default bounding box d of width w and height h.

$$L_{loc}(x,l,g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx,cy,w,h\}} x_{ij}^{k} \text{smooth}_{\text{L1}}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^{w} = \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^{h} = \log\left(\frac{g_j^h}{d_i^h}\right)$$

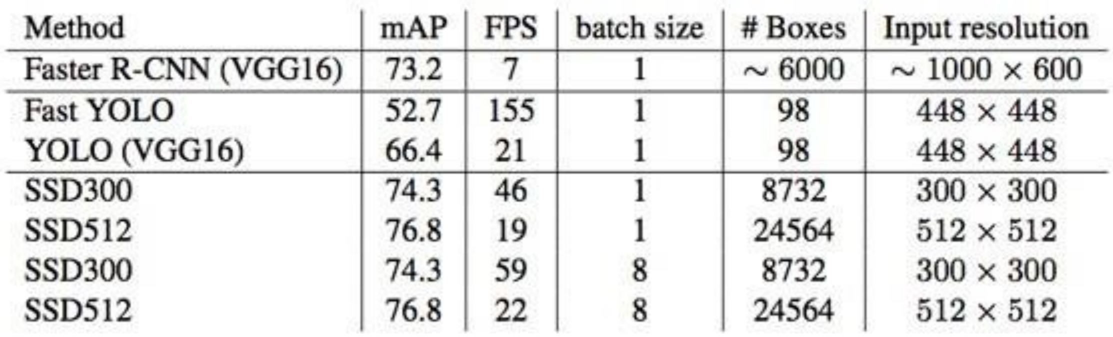
$$x_{ij}^{p} = \begin{cases} 1 & \text{if } IoU > 0.5 \text{ between default box } i \text{ and ground true box } j \text{ on class } p \\ 0 & \text{otherwise} \end{cases}$$

It is calculated as the softmax loss over multiple classes confidences c (class score).

$$L_{conf}(x,c) = -\sum_{i\in Pos}^{N} x_{ij}^{p} log(\hat{c}_i^p) - \sum_{i\in Neg} log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

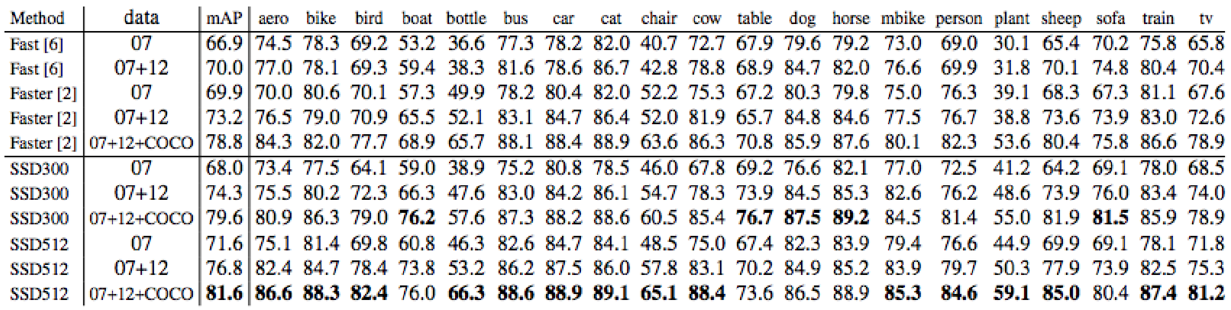
where N is the number of matched default boxes.

$$L(x,c,l,g) = \frac{1}{N}(L_{conf}(x,c) + \alpha L_{loc}(x,l,g))$$



Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	$\sim 1000 \times 600$
Fast YOLO	52.7	155	1	98	448×448
YOLO (VGG16)	66.4	21	1	98	448×448
SSD300	74.3	46	1	8732	300×300
SSD512	76.8	19	1	24564	512×512
SSD300	74.3	59	8	8732	300×300
SSD512	76.8	22	8	24564	512×512

Fig Speed performance in frames per second



Method	data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast [6]	07	66.9	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8
Fast [6]	07+12	70.0	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8	68.9	84.7	82.0	76.6	69.9	31.8	70.1	74.8	80.4	70.4
Faster [2]	07	69.9	70.0	80.6	70.1	57.3	49.9	78.2	80.4	82.0	52.2	75.3	67.2	80.3	79.8	75.0	76.3	39.1	68.3	67.3	81.1	67.6
Faster [2]	07+12	73.2	76.5	79.0	70.9	65.5	52.1	83.1	84.7	86.4	52.0	81.9	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6
Faster [2]	07+12+COCO	78.8	84.3	82.0	77.7	68.9	65.7	88.1	88.4	88.9	63.6	86.3	70.8	85.9	87.6	80.1	82.3	53.6	80.4	75.8	86.6	78.9
SSD300	07	68.0	73.4	77.5	64.1	59.0	38.9	75.2	80.8	78.5	46.0	67.8	69.2	76.6	82.1	77.0	72.5	41.2	64.2	69.1	78.0	68.5
SSD300	07+12	74.3	75.5	80.2	72.3	66.3	47.6	83.0	84.2	86.1	54.7	78.3	73.9	84.5	85.3	82.6	76.2	48.6	73.9	76.0	83.4	74.0
SSD300	07+12+COCO	79.6	80.9	86.3	79.0	**76.2**	57.6	87.3	88.2	88.6	60.5	85.4	**76.7**	**87.5**	**89.2**	84.5	81.4	55.0	81.9	**81.5**	85.9	78.9
SSD512	07	71.6	75.1	81.4	69.8	60.8	46.3	82.6	84.7	84.1	48.5	75.0	67.4	82.3	83.9	79.4	76.6	44.9	69.9	69.1	78.1	71.8
SSD512	07+12	76.8	82.4	84.7	78.4	73.8	53.2	86.2	87.5	86.0	57.8	83.1	70.2	84.9	85.2	83.9	79.7	50.3	77.9	73.9	82.5	75.3
SSD512	07+12+COCO	**81.6**	**86.6**	**88.3**	**82.4**	76.0	**66.3**	**88.6**	**88.9**	**89.1**	**65.1**	**88.4**	73.6	86.5	88.9	**85.3**	**84.6**	**59.1**	**85.0**	80.4	**87.4**	**81.2**

Fig Accuracy comparison for different methods

Multibox Priors and IOU :

The logic revolving around the bounding box generation is actually more complex than what I earlier stated. But fear not: it is still within reach.

In MultiBox, the researchers created what we call *priors* (or *anchors* in Faster-R-CNN terminology), which are pre-computed, fixed size bounding boxes that closely match the distribution of the original ground truth boxes. In fact those *priors* are selected in such a way that their Intersection over Union ratio (aka IoU, and sometimes referred to as Jaccard Index) is greater than 0.5. As you can infer from the image below, an IoU of 0.5 is still not good enough but it does however provide a strong starting point for the bounding box regression algorithm

The resulting architecture (check MultiBox architecture diagram above again for reference) contains 11 priors per feature map cell (8x8, 6x6, 4x4, 3x3, 2x2) and only one on the 1x1 feature map, resulting in a total of 1420 priors per image, thus enabling robust coverage of input images at multiple scales, to detect objects of various sizes.

SSD Improvements :

Back onto SSD, a number of tweaks were added to make this network even more capable of localizing and classifying objects.

Fixed Priors: unlike MultiBox, every feature map cell is associated with a set of default bounding boxes of different dimensions and aspect ratios. These priors are manually (but carefully) chosen, whereas in MultiBox, they were chosen because their IoU with respect to the ground truth was over 0.5. This in theory should allow SSD to generalise for any type of input, without requiring a pre-training phase for prior generation. For instance, assuming we have configured 2 diagonally opposed points *(x1, y1)* and *(x2, y2)* for each *b* default bounding boxes per feature map cell , and *c* classes to classify, on a given feature map of size $f = m * n$, SSD would compute $f * b * (4 + c)$ values for this feature map.

At the end, MultiBox only retains the top K predictions that have minimised both location (*LOC*) and confidence (*CONF*) losses.

Location Loss: SSD uses smooth L-1 Norm to calculate the location loss. While not as precise as L2-Norm, it is still highly effective and gives SSD more room for manoeuvre as it does not try to be “pixel perfect” in its bounding box prediction (i.e. a difference of a few pixels would hardly be noticeable for many of us).

Classification: MultiBox does not perform object classification, whereas SSD does. Therefore, for each predicted bounding box, a set of *c* class predictions are computed, for every possible class in the dataset.

Default Bounding Boxes:

It is recommended to configure a varied set of default bounding boxes, of different scales and aspect ratios to ensure most objects could be captured. The SSD paper has around 6 bounding boxes per feature map cell.

Feature Maps:

Features maps (i.e. the results of the convolutional blocks) are a representation of the dominant features of the image at different scales, therefore running MultiBox on multiple feature maps increases the likelihood of any object (large and small) to be eventually detected, localized and appropriately classified

Hard Negative Mining :

During training, as most of the bounding boxes will have low IoU and therefore be interpreted as *negative* training examples, we may end up with a disproportionate amount of negative examples in our training set. Therefore, instead of using all negative predictions, it is advised to keep a ratio of negative to positive examples of around 3:1. The reason why you need to keep negative samples is because the network also needs to learn and be explicitly told what constitutes an incorrect detection.

Data Augmentation :

The authors of SSD stated that data augmentation, like in many other deep learning applications, has been crucial to teach the network to become more robust to various object sizes in the input. To this end, they generated additional training examples with patches of the original image at different IoU ratios (e.g. 0.1, 0.3, 0.5, etc.) and random patches as well. Moreover, each image is also randomly horizontally flipped with a probability of 0.5, thereby making sure potential objects appear on left and right with similar likelihood.

Non-Maximum Supreesion (NMS):

Given the large number of boxes generated during a forward pass of SSD at inference time , it is essential to prune most of the bounding box by applying a technique known as *non-maximum suppression:* boxes with a confidence loss threshold less than *ct* (e.g. 0.01) and IoU less than *lt* (e.g. 0.45) are discarded, and only the top *N* predictions are kept. This ensures only the most likely predictions are retained by the network, while the more noisier ones are removed.

The following Observations are made on SSD

- more default boxes results in more accurate detection, although there is an impact on speed
- having MultiBox on multiple layers results in better detection as well, due to the detector running on features at multiple resolutions
- 80% of the time is spent on the base VGG-16 network: this means that with a faster and equally accurate network SSD's performance could be even better
- SSD confuses objects with similar categories (e.g. animals). This is probably because locations are shared for multiple classes
- SSD-500 (the highest resolution variant using 512x512 input images) achieves best mAP on Pascal VOC2007 at 76.8%, but at the expense of speed, where its frame rate drops to 22 fps. SSD-300 is thus a much better trade-off with 74.3 mAP at 59 fps.
- SSD produces worse performance on smaller objects, as they may not appear across all feature maps. Increasing the input image resolution alleviates this problem but does not completely address it

5.2 Sample Code

```
from tkinter import messagebox
from tkinter import *
from tkinter import simpledialog
import tkinter
from tkinter import filedialog
from tkinter.filedialog import askopenfilename
import imutils
import time
import cv2
import numpy as np


main = tkinter.Tk()
main.title("Object Tracking Using Python")
main.geometry("1300x1200")

net =
cv2.dnn.readNetFromCaffe("MobileNetSSD_deploy.prototxt.txt","MobileNetSSD_depl
oy.caffemodel")


global filename
global train
global ga_acc, bat_acc, bee_acc
global classifier


CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
      "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
      "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
      "sofa", "train", "tvmonitor",]
COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))


def uploadVideo():
    global filename
    filename = filedialog.askopenfilename(initialdir="videos")
    pathlabel.config(text=filename)
    text.delete('1.0', END)
    text.insert(END,filename+" loaded\n");
    vc = cv2.VideoCapture(filename)
    while True:
        frame = vc.read()
        frame = frame if filename is None else frame[1]
        if frame is None:
            break
        frame = imutils.resize(frame, width=500)
        (h, w) = frame.shape[:2]
        blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)),0.007843,
(300, 300), 127.5)
```

```
        net.setInput(blob)
        detections = net.forward()
        for i in np.arange(0, detections.shape[2]):
            confidence = detections[0, 0, i, 2]
            if confidence > 0.2:
                idx = int(detections[0, 0, i, 1])
                box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
                (startX, startY, endX, endY) = box.astype("int")
                if (confidence * 100) > 50:
                    label = "{}: {:.2f}%".format(CLASSES[idx],confidence *
100)
                    cv2.rectangle(frame, (startX, startY), (endX,
endY),COLORS[idx], 2)
                    y = startY - 15 if startY - 15 > 15 else startY + 15
                    cv2.putText(frame, label, (startX,
y),cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)
                    text.insert(END,label+"\n")

        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF
        if key == ord("q"):
             break

    vc.stop() if filename is None else vc.release()
    cv2.destroyAllWindows()

def exit():
    main.destroy()


font = ('times', 16, 'bold')
title = Label(main, text='Object Tracking Using Python')
title.config(bg='light cyan', fg='pale violet red')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)

font1 = ('times', 14, 'bold')
uploadButton = Button(main, text="Browse System Videos", command=uploadVideo)
uploadButton.place(x=50,y=100)
uploadButton.config(font=font1)

pathlabel = Label(main)
pathlabel.config(bg='light cyan', fg='pale violet red')
pathlabel.config(font=font1)
pathlabel.place(x=460,y=100)

webcamButton = Button(main, text="Start Webcam Video Tracking",
command=webcamVideo)
webcamButton.place(x=50,y=150)
webcamButton.config(font=font1)

exitButton = Button(main, text="Exit", command=exit)
```

```
exitButton.place(x=330,y=150)
exitButton.config(font=font1)


font1 = ('times', 12, 'bold')
text=Text(main,height=20,width=150)
scroll=Scrollbar(text)
text.configure(yscrollcommand=scroll.set)
text.place(x=10,y=250)
text.config(font=font1)


main.config(bg='snow3')
main.mainloop()
```

CHAPTER 6

RESULTS AND DISCUSSION

An accurate and efficient object detection system has been developed which achieves comparable metrics with the existing state-of-the-art system. This project uses recent techniques in the field of computer vision and deep learning.

In this research, we built a real time deep learning model to identify progressively the place of the object in pictures. The framework could distinguish the item with a normal accuracy like other best in class frameworks. In this way, we utilize an object detection module that can recognize what is in the real time video stream. To carry out the module, we join the MobileNet and the SSD framework for a quick and productive deep learning-based strategy for object detection. In future work, we will keep on enhancing our detection network model, including lessening memory utilization and speeding up and additionally we will add more classes.

The trial results show that the Average Precision (AP) of the algorithm to recognize various classes as vehicle, person and chair is 99.76%, 97.76% and 71.07%, separately.

6.1 Output Screenshots

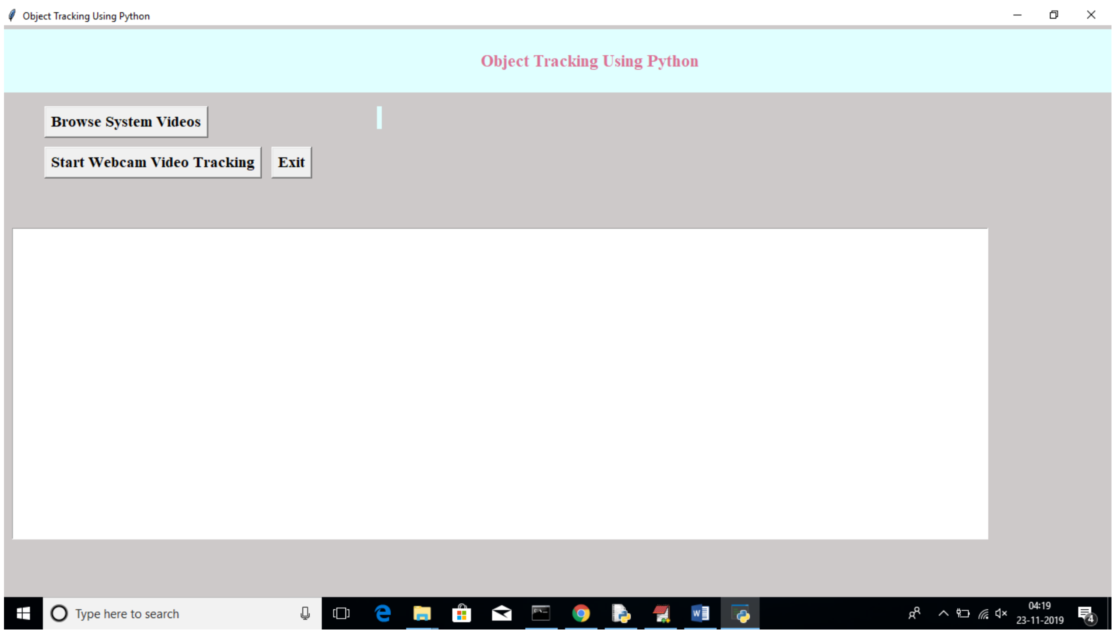

Object Tracking Using Python
Object Tracking Using Python
Browse System Videos
Start Webcam Video Tracking
Exit
Type here to search
04:19
23-11-2019

Fig 3.1 Main Interface

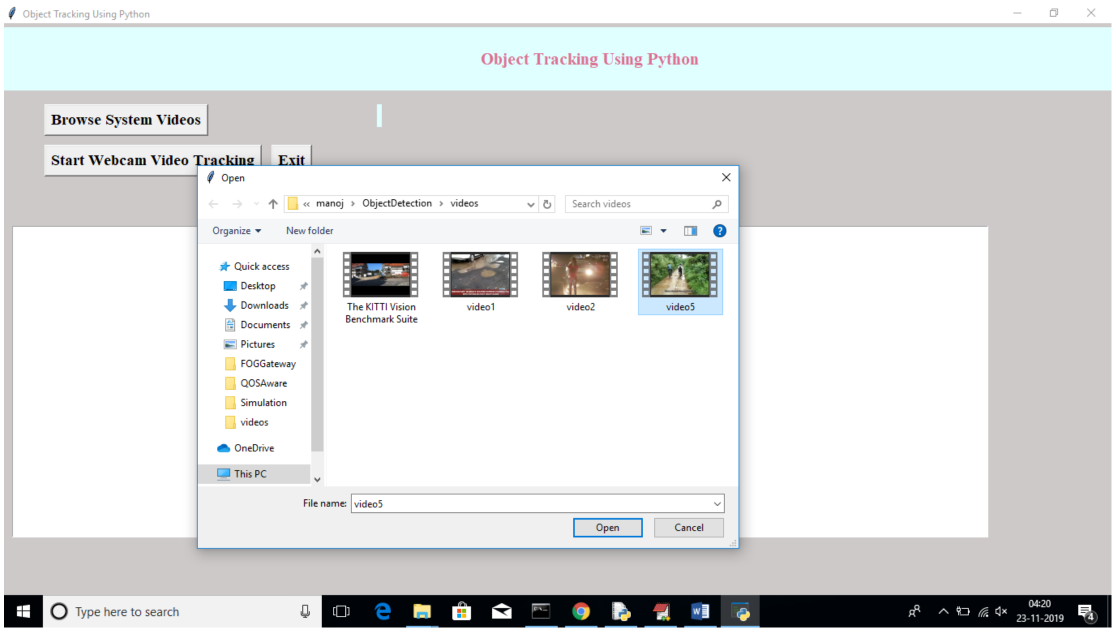


Fig 3.2 Browse Video

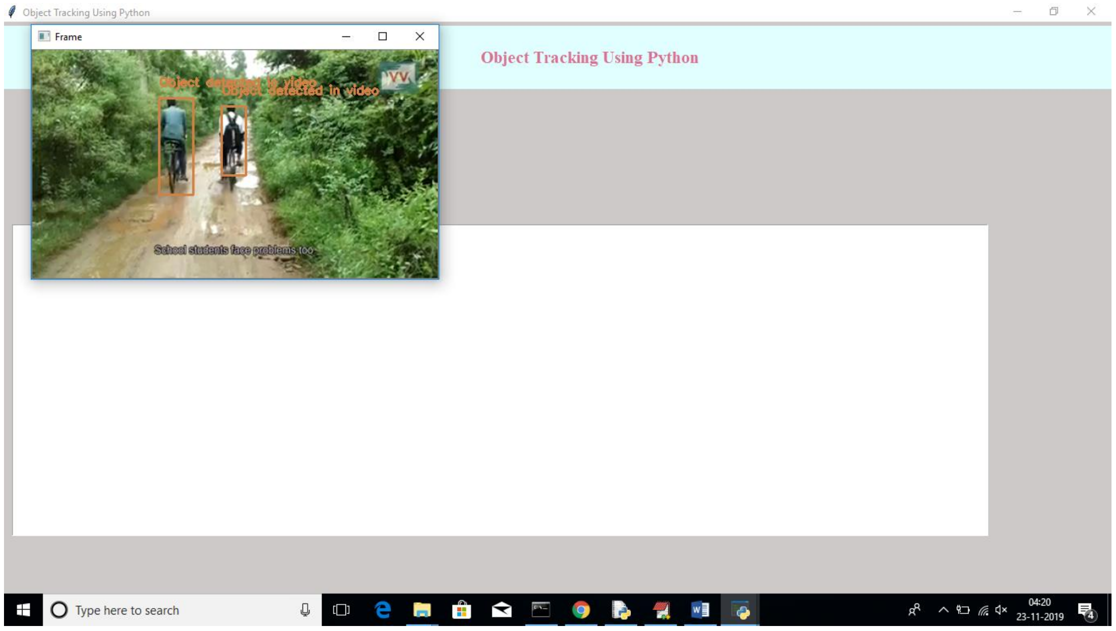

Object Tracking Using Python
Frame
Object Tracking Using Python
Object detected in video
Object detected in video
School students face problems too
Type here to search
04:20
23-11-2019

Fig 3.3 recognition of two people riding bicycle

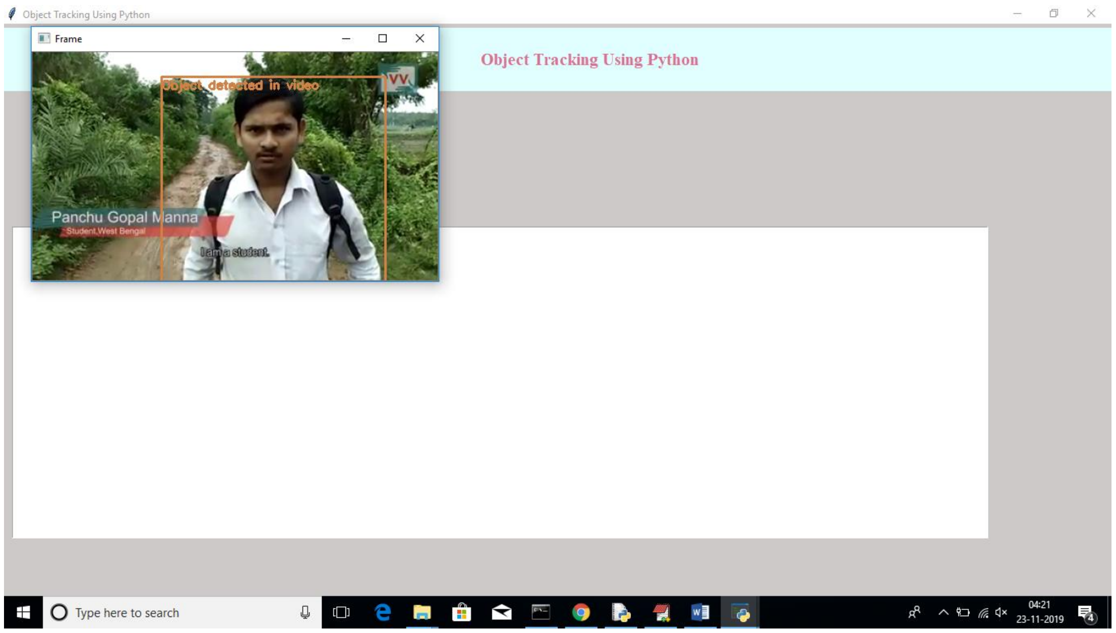
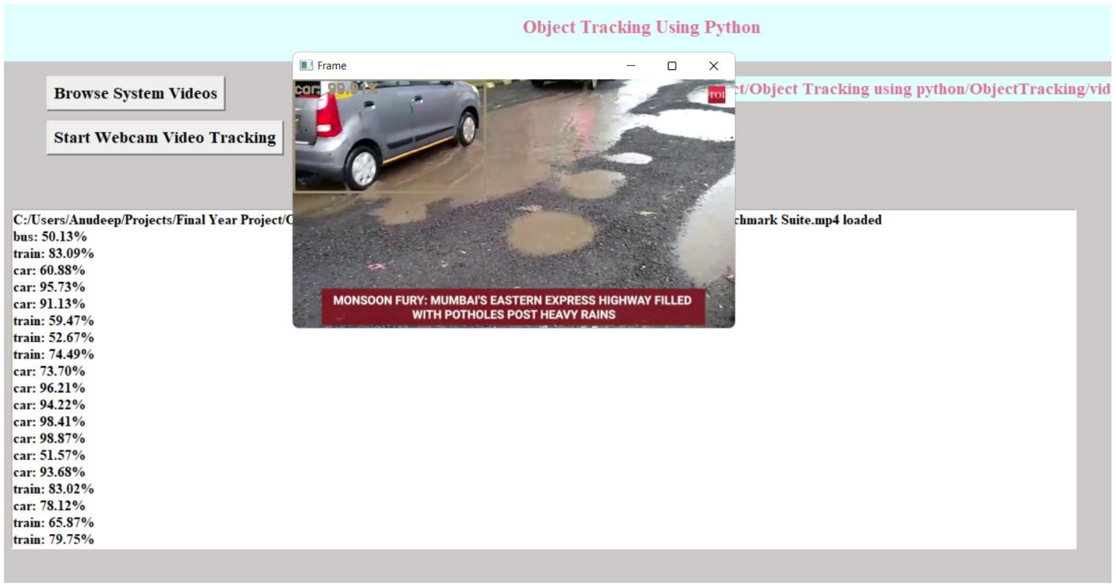


Fig 3.4 recognition of person from an image


Object Tracking Using Python
Browse System Videos
Start Webcam Video Tracking
Frame
ct/Object Tracking using python/ObjectTracking/vid
MONSOON FURY: MUMBAI'S EASTERN EXPRESS HIGHWAY FILLED WITH POTHOLES POST HEAVY RAINS

```
C:/Users/Anudeep/Projects/Final Year Project/C                              chmark Suite.mp4 loaded
bus: 50.13%
train: 83.09%
car: 60.88%
car: 95.73%
car: 91.13%
train: 59.47%
train: 52.67%
train: 74.49%
car: 73.70%
car: 96.21%
car: 94.22%
car: 98.41%
car: 98.87%
car: 51.57%
car: 93.68%
train: 83.02%
car: 78.12%
train: 65.87%
train: 79.75%
```

Fig 3.5 Vehicle Detection

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

In this research, we built a real time deep learning model to identify progressively the place of the object in pictures. The framework could distinguish the item with a normal accuracy like other best in class frameworks. In this way, we utilize an object detection module that can recognize what is in the real time video stream. To carry out the module, we join the MobileNet and the SSD framework for a quick and productive deep learning-based strategy for object detection. In future work, we will keep on enhancing our detection network model, including lessening memory utilization and speeding up and additionally we will add more classes.

The main purpose of our analysis is to elaborate the accuracy of an object detection technique SSD and the pre trained deep learning model MobileNet and additionally feature a portion of the notable elements that make this method stand out. The trial results show that the Average Precision (AP) of the algorithm to recognize various classes as vehicle, person and chair is 99.76%, 97.76% and 71.07%, separately.

Although the proposed model performs well, it fails to identify objects during nighttime and other harsh environments. The main reason for this failure is that the model is not well trained on images that are taken during nighttime. So, training the model on post day data will be beneficial for improving the model accuracy overall.

REFERENCES

[1] Wei Liu and Alexander C. Berg, “SSD: Single Shot MultiBox Detector”, Google Inc., Dec 2016.

[2] Andrew G. Howard, and Hartwig Adam, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”, Google Inc., 17 Apr 2017.

[3] Justin Lai, Sydney Maples, “Ammunition Detection: Developing a Real-Time Gun Detection Classifier”, Stanford University, Feb 2017

[4] Shreyamsh Kamate, “UAV: Application of Object Detection and Tracking Techniques for Unmanned Aerial Vehicles”, Texas A&M University, 2015.

[5] Adrian Rosebrock, “Object detection with deep learning and OpenCV”, pyimagesearch.

[6] Mohana and H. V. R. Aradhya, "Elegant and efficient algorithms for real time object detection, counting and classification for video surveillance applications from single fixed camera," 2016 International Conference on Circuits, Controls, Communications and Computing (I4C), Bangalore, 2016, pp. 1-7.

[7] Akshay Mangawati, Mohana, Mohammed Leesan, H. V. Ravish Aradhya, “Object Tracking Algorithms for video surveillance applications” International conference on communication and signal processing (ICCSP), India, 2018, pp. 0676-0680.

[8] Apoorva Raghunandan, Mohana, Pakala Raghav and H. V. Ravish Aradhya, “Object Detection Algorithms for video surveillance applications” International conference on communication and signal processing (ICCSP), India, 2018, pp. 0570-0575.

[9] Manjunath Jogin, Mohana, “Feature extraction using Convolution Neural Networks (CNN) and Deep Learning” 2018 IEEE International Conference On Recent Trends In Electronics Information Communication Technology,(RTEICT) 2018, India.

[10] Arka Prava Jana, Abhiraj Biswas, Mohana, “YOLO based Detection and Classification of Objects in video records” 2018 IEEE International Conference On Recent Trends In Electronics Information Communication Technology

[11] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015.

[12] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large scale machine learning on heterogeneous systems,

2015. Software available from tensorflow. org, 1, 2015.

[13] Timothy Dozat (2016). INCORPORATING NESTEROV MOMENTUM INTO ADAM. Workshop track - ICLR 2016. https://openreview.net/pdf?id=OM0jvwB8jIp57ZJjtNEZ.

[14] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[15] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 4(2),2012.

[16] Yundong Zhang, Haomin Peng haomin and Pan Hu, “Towards Real time Detection and Camera Triggering,” CS341.

[17] Ibai Gorordo Fernandez and Chikamune Wada, “Shoe Detection Using SSD-MobileNet Architecture,”2020 IEEE 2nd Global Conference on Life Sciences and Technologies (LifeTech 2020).

[18] Yu-Chen Chiu, Chi-Yi Tsai, Mind-Da Ruan, Guan-Yu Shen and Tsu Tian Lee, “Mobilenet-SSDv2: An Improved Object Detection Model for Embedded Systems,” ©2020 IEEE.

[19] Andres Heredia and Gabriel Barros-Gavilanes,” Video processing inside embedded devices using SSD-Mobilenet to count mobility actors,” 978- 1-7281-1614-3/19 ©2019 IEEE.

[20] G. Bradski and, A. Kaehler, “Learning OpenCV”, OReilly Publications, 2008.

[21] Animesh Srivastava1, Anuj Dalvi2, Cyrus Britto3, Harshit Rai4, Kavita Shelke5,” Explicit Content Detection using Faster R-CNN and SSD MobileNet v2,” e-ISSN: 2395-0056 © 2020, IRJET.

[22] R. Huang, J. Pedoeem, and C. Chen, “YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers,” in Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018.