Department of Electrical and Computer Engineering

North South University

Senior Design Project

CSE499B Section-15

Object Detection and Recognition System using TensorFlow

Group 04:

Khadiza Sarker Bobby	1711049042
Md Abdul Aziz	1711757042
Tanvir Hossain Jishad	1610179042

Faculty Advisor:

Dr. Tanzilur Rahman

Assistant Professor

ECE Department

Letter of transmittal

February, 2021

To

Dr. Tanzilur Rahman

Assistant Professor

Department of Electrical and Computer Engineering

North South University

Subject: Submission of Final-report Thesis on Object detection and recognition system using TensorFlow

Dear Sir,

With due respect, we would like to submit our final thesis report on Object detection and recognition system using TensorFlow as a part of our CSE499A & CSE499B project. The report will discuss with an object detection system which ensure image detection and classification. We learned many things working on this project about Deep learning neural networks, which was a new concept to us.

We pray and hope you will be kind enough to accept this report and provide your valuable judgment.

Sincerely Yours,

………………………………

Khadiza Sarker Bobby

1711049042

ECE Department

North South University

……………………………....

Md Abdul Aziz

1711757042

ECE Department

North South University

………………………………

Tanvir Hossain Jishad

1610179042

ECE Department

North South University

Approval

The project entitled “Object detection and recognition system using TensorFlow” by Khadiza Sarker Bobby (ID: 1711049042), Md Abdul Aziz (ID: 1711757042), and Tanvir Hossain Jishad (ID: 1610179042), is approved in partial fulfillment of the requirement of the Degree of Bachelor of Science in Computer Science and Engineering on February, 2021 and has been accepted as satisfactory.

Supervisor:

……………………………......

Dr. Tanzilur Rahman

Assistant Professor

Department of Electrical and Computer Engineering

North South University

Dhaka, Bangladesh.

Chairman:

………………………………

Dr. Rezaul Bari

Associate Professor and Chairman

Department of Electrical and Computer Engineering

North South University

Dhaka, Bangladesh.

Declaration

We are here to declare that Object detection system using TensorFlow report is not copied from anywhere. We also admit that this particular project had not been used in any other course or related purpose or nor it will be used in future.

………………………………

Khadiza Sarker Bobby

1711049042

ECE Department

North South University

……………………………....

Md Abdul Aziz

1711757042

ECE Department

North South University

………………………………

Tanvir Hossain Jishad

1610179042

ECE Department

North South University

Acknowledgement

First of all, we would like to show our appreciation to Dr. Tanzilur Rahman, who accepted our project proposal and gave us the opportunity to work on this project. He had been motivating, showed his kind direction, gave us valuable recommendations, insightful counsel, and participated all through our work.

Then, we would like to thank all of the faculties who had provide all the theoretical and practical understanding of computer science throughout out whole BSc program.

Lastly, we thank our friends and family to support us from the past few months.

Abstract

We are presenting the design and implementation of a real-time approach to detect and track features in a structured context using TensorFlow. Object detection and recognition are essential and challenging tasks in many computer vision applications such as vehicle navigation, surveillance and monitoring, self-driving cars, home automation, and autonomous robot navigation. Object detection requires placing objects in the framework of a video or camera. Every tracking method entails an object detection mechanism either in every frame or when something first surfaces in the camera view. The advanced computers, high-quality camera, and economical video cameras, and the growing need for automated video analysis have produced a great deal of investment in object tracking algorithms. There are three key measures in object analysis: detecting objects, recognizing objects from each frame to frame, and interpreting object tracks to identify their behavior. Therefore, object tracking is relevant in the tasks of automatic detection, tracking, motion-based recognition, and many others. This paper discuss a deep learning method for the robust detection of distinctive objects by the object detection model and evaluating the flexibility of the TensorFlow Object Detection Framework to solve real-time problems.

Table of Contents

Letter of transmittal……………………………………………………………………….1

Approval......………………………………………………………………………………3

Declaration………………………………………………………………………………...4

Acknowledgement………………………………………………………………………...5

Abstract……………………………………………………………………………………...6

Table of contents…………………………………………………………………………..7

List of figures……………………………………………………………………………10

List of tables……………………………………………………………………………...12

Chapter 1: Introduction……………………………………..…………………………13

1.1 Overview…………………………………………………………………………..14

1.2 What is object detection and recognition?.........…………………………………..14

1.3 TensorFlow…………………………………………………………………….….15

1.4 History of different object detection approaches……………………………………..16

1.5 Motivation…………………………………………………………………………...18

1.6 Aim…………………………………………………………………………………..18

1.7 Objective ………………………………………………………………………19

1.8 Thesis Outline……………………………………………………………………...19

Chapter 2: Background Study…………………………………………………………20

2.1 Objects Talk-Object detection and Pattern Tracking using TensorFlow……………..21

2.2 Forest Monitoring System through RFID and TensorFlow Object Detection……......22

Chapter 3: Methodology……………………….…………………………………………24

3.1 CNN: Convolutional Neural Network……………………………………………...25

3.2 Workflow…………………………………………………………………………......26

3.3 Import all module…………………….………………………………………………23

3.3.1 TensorFlow Object Detection API…………………………………………….......28

3.4 Dataset………………………………………………………………………………29

3.5 Preprocessing………………………………………………………………………31

3.6 Deep learning neural network.…………………………….……………………….32

3.6.1 Convolution layer………………………………….………………………32

3.6.2 Pooling Layer………………………………………………………………34

3.6.3 Max Pooling…………………………………………………………….....34

3.6.4 Average Pooling……………………………………………………………35

3.7 Modules Download………………...………………………………………………36

3.7.1 SSD Architecture…………………..……………………………….………36

3.8 Create level map and config training……………………………………………….37

3.9 Convert load video into NumPy array………………………………………………37

3.10 Object Detecting with Bounding Boxes……………………………………………...38

3.11 Voice Generation…………………………………………………………………….39

Chapter 4: Experimental Results…………………………………………………………………40

4.1 Single Object Detection from Webcam……………………………………………….41

4.2 Multiple object Detect from webcam using SSD mobilenet v1……………………….47

4.3 Multiple object Detect from webcam using SSD inception v2………………………...48

4.4 Accuracy Comparison between SSD mobilenet vs SSD inception v2………………...49

4.5 Multiple object Detect from webcam using faster rcnn inception v2………………….50

4.6 Multiple object Detect from webcam using faster rcnn inception resnet………………51

4.7 Multiple object Detect from webcam using faster rcnn resnet 101…………………….51

4.8 Accuracy comparison between three model of faster rcnn…………………………….53

Chapter 5: Conclusion…………………………………………………………..…………..…….55

5.1 Future Work……………………………………………………………………….…….56

5.2 Limitations……………………………………………………………………………56

5.3 Conclusion……………………………………………………………………………56

References……………………………………………………………………………..…59

Declaration of member’s contribution………………………………………………………60

List of figures

1. Figure 1: Second convolutional layer of a CNN depicting how they extract circles and stripes from images.

2. Figure 2: Features of each convolutional layer in a CNN

3. Figure 3: Object detection from a webcam in our proposed model

4. Figure 4: How TensorFlow Object Detection API works

5. Figure 5: Some filter operation and their mask value

6. Figure 6: Complete diagram of CNN architecture

7. Figure 7: Output image dimension calculation

8. Figure 8: After applying filter output image

9. Figure 9: Max pooling workflow

10. Figure 10: Max pooling down-sampling

11. Figure 11: Average pooling workflow

12. Figure 12: SSD Architecture

13. Figure 13: Object detect with bounding boxes

14. Figure 14: Remote Detection from webcam

15. Figure 15: Vase Detection from webcam

16. Figure 16: Clock Detection from webcam

17. Figure 17: Cup Detection from webcam

18. Figure 18: Book Detection from webcam

19. Figure 19: Mouse Detection from webcam

20. Figure 20: Cell phone Detection from webcam

21. Figure 21: Chair Detection from webcam

22. Figure 22: Vase Detection from webcam

23. Figure 23: Bottle Detection from webcam

24. Figure 24: Car Detection from webcam

25. Figure 25: Multiple objects detect from webcam

26. Figure 26: Multiple object detect from webcam using inception v2

27. Figure 27: Multiple object detect from webcam using faster rcnn inception v2

28. Figure 28: Multiple object detect on webcam using faster rcnn inception resnet v2

29. Figure 29: Multiple object detect on webcam using faster rcnn resnet 101

30. Figure 30: Clock and Car Detection using SSD inception v2

List of tables

1. Table 1: Different deep learning frameworks and their characteristics

2. Table 2: Comparison between five different COCO-trained models

3. Table 3: Single object accuracy

4. Table 4: Accuracy comparison between SSD mobilenet and SSD inception

5. Table 5: Comparison accuracy between faster rcnn inception, faster rcnn inception resnet v2 and faster rcnn inception resnet 101

CHAPTER 1: INTRODUCTION

1.1 Overview

In recent years, significant and successful development has been witnessed in computer vision. This success has come partly from adopting and adapting machine learning methods, while others from the development of new representations and models for specific computer vision problems or from the development of efficient solutions. One area that has attained significant progress is object detection. Object detection has been used in many applications, with the most popular ones being: human-computer interaction (HCI), robotics (e.g., service robots), consumer electronics (e.g., smart-phones), security (e.g., recognition, tracking), retrieval (e.g., search engines, photo management), and transportation (e.g., autonomous and assissted driving). We will use this development to build a Raspberry pi object detection model to be implemented with the ability to immediately detect and recognize the object within its field of view.

Real-time object detection and estimation frameworks are very crucial tasks within industrial preparation. Object detection is often utilized in product quality control within the industry. The proposed framework can be connected to an industrial quality control system. Moreover, it can be utilized for different industrial systems or security purposes. By and large, it recognizes objects in the public region and measures the dimensions of each of them. The proposed model's proficiency has been confirmed by utilizing images taken from a Raspberry Pi camera. The execution of this method has a high computation rate, and it is subordinate to the resolution of frames. The accomplishment of distinguishing objects and separate these objects from the background is ideal.

1.2 What is object detection and recognition?

Object detection is the first stage in many computer vision systems, which has been carried out as it allows us to obtain further knowledge about the detected object and the background. Once an object instance has been detected, such as a person, it is possible to get related information, including identifying the specific instance such as to recognize the subject's face, to follow the object over an image sequence, for example, to trace the face in a video, and to gather more information about the object. Simultaneously, it is also feasible to extract other objects' distance or location in the background and estimate further data, among additional contextual information.

There are three major problems related to object detection. The first one is object localization, which consists of specifying the location and scale of a single object specimen known to be present in the vision; the second one is object presence classification, which conforms to determining whether at least one object of a given class is currently in the view, while the last problem is object recognition, which consists in determining if a specific object instance is present. Obtaining a precise and fast deduction from an object detection algorithm is challenging to overcome through this model. Typically only a few instances of the object are present in the image, but there is a very significant number of possible locations and scales at which they can occur and that need to be somehow explored. Each detection is reported with some form of pose information. It plays a vital role in selecting a proper feature in tracking. We use the deep learning approach, and the feature representation is learned instead of being designed by the user, and also we used a large number of training samples as required for training the classifier.

1.3 TensorFlow

The project is based on a Deep learning model running on TensorFlow's Object Detection API. A vital tool gives it straight to construct, train, and use object detection models. Generally, training a whole convolutional network from scratch consumes a lot of time and required a vast dataset. So, this problem can be fixed using the power of transfer learning with a pre-trained model using the TensorFlow API.

1.4 History of different object detection approaches

Machine learning-based approaches or deep learning-based approaches are the most widely used approach for an object detection model. For the Machine Learning approach, it is necessary to determine features using one of the methods mentioned below, later using a support vector machine (SVM) to do the model. On the contrary, deep learning techniques handle end-to-end object detection without precisely distinguishing features and are generally based on convolutional neural networks (CNN).

Machine learning approach [1]:

- Viola–Jones object detection framework based on Haar features
  The Viola-Jones Object Detection Framework was developed by Paul Viola and Michael Jones back in 2001. It can swiftly and precisely detect objects in images and works especially great with the human face. Though it came into work a long time ago, the framework is still a leading performer in face detection, contrary to many of its CNNs counterparts. The Viola-Jones Object Detection Framework includes the concepts of Haar-like Features, Integral Images, the AdaBoost Algorithm, and the Cascade Classifier to create a system for object detection that is quick and reliable.

- Scale-invariant feature transform (SIFT)
  In computer vision, the scale-invariant feature transform (SIFT) is a feature detection algorithm to detect and define local features in pictures, published by David Lowe in 1999. SIFT is implemented in various applications, including object recognition, gesture recognition, image stitching, robotic mapping and navigation, 3D modelling, video tracking, match moving, and individual identification of wildlife.

- Histogram of oriented gradients (HOG) features
  First used by Mitsubishi Electric Research Laboratories in 1994, the histogram of oriented gradients (HOG) is a feature defining image processing used for the purpose of object detection. The method calculates phenomena of gradient orientation in localized portions of a particular background. This process is comparable to that of scale-invariant feature transform but contrasts in that it is estimated on a dense grid of uniformly spaced cells and utilizes overlapping local contrast normalization for improved efficiency.

Deep learning approach [1]:

- Region Proposals (R-CNN, Fast R-CNN, Faster R-CNN, cascade R-CNN) The RPN network consists of several layers. One of them is a region proposal algorithm to bind a detected object with "bounding boxes" or see possible objects' locations in the image. Another is the feature generation stage to receive features of these objects, usually using a CNN. Then a classification layer to predict which category this object belongs to. And lastly, a regression layer to make the coordinates of the object bounding box more accurate.

- Single Shot MultiBox Detector (SSD)

  Designed for object detection in real-time, SSD expedites up the process by eliminating the need for the RPN. SSD applies a few improvements, including multi-scale features and default boxes, to improve accuracy. These improvements allow SSD to meet the Faster R-CNN's accuracy using lower resolution images to make detection precise.

- You Only Look Once (YOLO)
  YOLO is an innovative convolutional neural network (CNN) for doing object detection in real-time. It is an algorithm based on regression; instead of selecting the interesting part of a picture, it detects classes and bounding boxes for the whole image in one run of the algorithm.

- Single-Shot Refinement Neural Network for Object Detection (RefineDet) RefineDet achieves better accuracy than two-stage methods and maintains comparable efficiency of one-stage processes. The model consists of two inter-connected modules. The anchor refinement module and the alignment detection module.

- Retina-Net
  RetinaNet is one of the most reliable one-stage object detection models shown to work well with dense and small scale objects. It is modelled by making two improvements over existing single-stage object detection models - Feature Pyramid Networks (FPN) and Focal Loss.

- Deformable convolutional networks
  Deformable convolution comprises of two parts: regular convolutional layer and another convolutional layer to learn 2D offset for each input.

1.5 Motivation

Computer vision is extensively used by applications based on automation, consumer markets, medical fields, robot vision and animation, security, and traffic monitoring. The significance of object detection devices both in economical and sociological standpoints and in domestic environments as help to the elderly and disabled has been well recognized. We are primarily interested in a model that should be able to easily navigate in busy and crowded settings, detect obstacles, and maneuver objects. Although a large body of work exists related to detection systems, there are still no fully operational systems that can operate long-term in crowded areas. We expect our system to navigate through a home or an office environment and detect and classify objects. This type of model can be modified into various other models in near future, for example, a self-cleaning house robot, helping hand for a visually impaired person, a vehicle monitoring system, etc.

1.6 Aim

- Step by step processing for this system: image classification, localization, detection and programmed decision making
- To assess deep-learning procedures to develop an object detection system implemented with the help of Raspberry Pi.
- Developing a CNN architecture that trains with limited data and performs real-time inference on videos.
- Ensuring accurate detection of objects within a small and moving frame
- Improve segmentation and identifying every kind of object
- Increase model’s ability to interact with everyday objects in its surroundings.

1.7 Objective

- Overcome challenges arose due to complex object motion, the irregular shape of the object, occlusion of the object to object
- Ensuring minimum time interval passed between the capturing of a particular frame from the video stream and generating the processed outcomes.
- Minimize problems regarding memory and storage limitations, installing a cooling mechanism to make the model perform efficiently

1.8 Thesis outline

- Chapter 1: Introduction- This section gives an overview of the proposed model and several aspects of object detection systems. Also, the aim and objective to work on this model is mentioned.
- Chapter 2: Background Study- discuss a comprehensive literature review of previous works on object detection systems.
- Chapter 3: Methodology- Detailed description on hardware and software implementation of the proposed model.
- Chapter 4: Experimental Results- In this section, we discuss the results obtained by our developed model and we also show the accuracy level of the outcome.
- Chapter 5: Conclusion- As the name suggests, here we conclude the model by discussing ours works so far and also, the future works we intend to do. Moreover, we summarize the whole detection model in this section.

CHAPTER 2: BACKGROUND STUDY

In the first two sections of this chapter, we go through a couple of papers that we have studied and tried to follow how they carried out the work.

2.1 Objects Talk-Object detection and Pattern Tracking using TensorFlow [2]

We have a number of object detection models. Among all the systems, this particular one a unique characteristic of recognizing a pattern. Household objects are frequently in use and often follow specific patterns concerning time and geographical movement. This model aims to analyze these patterns that help the user keep better track of our objects and maximize productivity by reducing time and energy wasted in forgetting or looking for them. For example, a person always keeps her car keys beside the door, the object detection model will recognize and memorize this pattern, and if she forgets to take the key, it will alert her.

Objects Talk follows a similar methodology to us-they used Tensorflow to model their neural network and SSD to improve the precision and the range of objects detected. Once they design the system to detect objects, it is trained to track the object in the scope of the camera.

REMO framework is used as a base for generating a spacio-temporal pattern detecting algorithm. They have trained their dataset with around 150 images of each object. They used concepts of leadership, encounter, and convergence to predict the location of patterns of objects. Additionally, they have examined several deep learning frameworks for critical characteristics to determine which one works best.

Table 1: Different deep learning frameworks and their characteristics

	Caffe	TensorFlow	Torch	Theano
Language	C++, Python	Python	Lua	Python

Pretrained	Yes ++	Yes (Inception)	Yes ++	Yes (Lasagne)
GPU	CUDA, Opecl	CUDA	CUDA	CUDA, Opecl
Good at RNN	No	Yes (Best)	Mediocre	Yes

How Objects Talk works:

We will discuss how they implement the whole system in this section. They have specified in their system that more than one object can be in the defined radius, and one of them was heading for the direction before the others. It is assumed that moving objects do not change direction and speed; more than one entity will pass through a radius 'r,' and their paths will intersect at some point. The designers intend to extend the system further to recognize a user's symptoms and advises him to rest, suggest exercises, or visit a doctor. They will probably incorporate voice recognition and voice commands to the system to shut it down if a user do not want to be traced because, to some extent, it can violate a person's privacy.

2.2 Forest Monitoring System through RFID and TensorFlow Object Detection [3]

The majority of the journals we studied are designed to ease human lives, but this system is implemented to prevent deforestation and trafficking. This model detects intruders and alerts authorized legal officials when they enter the restricted forest area. With the increasing threat to forests and wildlife, this Forest Monitoring system will be focusing on minimizing the illegal activities by alerting the security offices using walkie-talkies or smartphones or any digital systems.

They have designed the model using cameras with image processing technology which detects any intruder's face, RFID (Radio-frequency identification), and an alert system using GSM technology. RFID applies electromagnetic fields to identify and track tags attached to objects automatically. GSM technology is employed for interaction between the model and officials in the forests. GSM is an open and digital cellular technology utilized for transmitting mobile voice and data services. This system was generated as a digital system using the time division multiple access (TDMA) technique for communication objectives.

Similar to our work, they used an object detection Algorithm provided by TensorFlow, a module of Python Programming language, embedded in the surveillance cameras installed in the forests. TensorFlow is used primarily for: classification, understanding, perception, discovering, and prediction. The algorithm to detect only humans is also embedded in the cameras. As soon as any human is detected, it will capture frames of the persons and then initiate another part of the project, i.e. RFID technology, which will work as identification for authorized officials. Moreover, it will process the pictures and merge them with the location's details of the area and send alerts to the Officials using the GSM module.

As it involves image processing technology, it uses an HD quality webcam with night-vision powered by solar power. It also utilizes solar energy that can be used to power systems such as RFID technology and GSM technology. They used COCO, a large image dataset designed for object detection, person key points detection, segmentation, stuff segmentation, and caption generation. Python 3.6 has offered unconditional support for the TensorFlow module and other modules required for object detection. The Raspberry Pi has been used to feed programs into the module and process the frames discovered from the camera after the non-identification of RFID tags in the area.

CHAPTER 3: METHODOLOGY

This chapter gives a diagram of the diverse parts of the work chronologically. It primarily examines the speculations, strategies, and step by step workflow of the work. Before starting with the methodology, we need to fully understand the CNN (Convolutional Neural Network), on which we built our project.

3.1 CNN: Convolutional Neural Network

CNN is a special kind of architecture that is widely used when working with image data. CNNs are one of the key innovations that led to the deep neural network revolution in computer vision. To understand CNN, we first need to know about how a computer interprets an image. In computer images are just collection or array of grid cells called pixels. Each pixel has a numerical value from 0-255. This value indicates the color channel of the image. Zero indicates the darkest and 255 indicates the lightest color. When we work with image data, we generally have huge number of inputs. If we have a 28x28 px grey-scaled image then the computer interprets the image as 2D array of 28X28 pixel values. We have (28 X 28) = 784 inputs for processing such a small image. If we want to feed the input to a multi layered perceptron (MLP) we need to feed a vector of input size 784 to it. Now if we are working with color images than we will have to deal with 3 extra arrays of 28X28 arrays for each red, green and blue color channel. In such case the input size will increase many folds and working with such large number of inputs in MLP is very problematic. That is why CNN is specifically preferred when working with large image inputs.

Convolutional Networks are neural networks that divide their parameters across space so it can remember spatial information. The preservation of these spatial information is known as convolutional layers. A convolutional layer provides sets of convolutional filters, kernels to input image. The kernels can extract various features from the image like outline of objects in the given picture or varying colors that distinguishes different types of objects from one another. For better understanding, let us look at the figure below where it shows how the second convolutional layer of a CNN extracts patterns like circles and stripes are done from specific images.

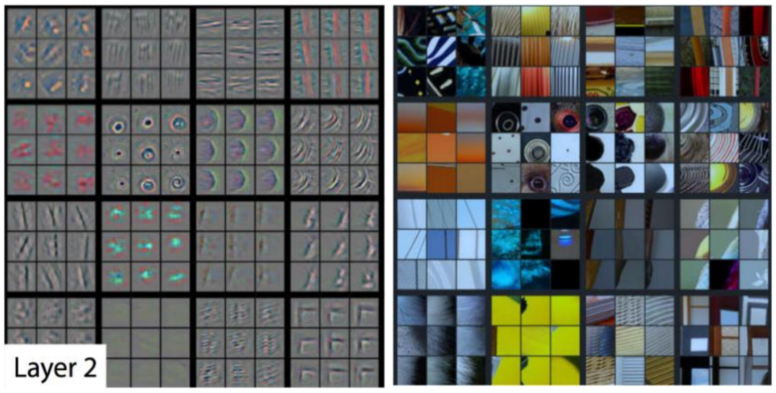


Figure 1: Second convolutional layer of a CNN depicting how they extract circles and stripes from images.

CNN are stack of convolutional layers that are utilized to extract patterns from images and pooling layers in between them to reduce dimensionality. These layers are then combined with a fully connected layer to build the final architecture. As we get deeper in the layer, each individual feature maps extract more and more complicated patterns of the image.

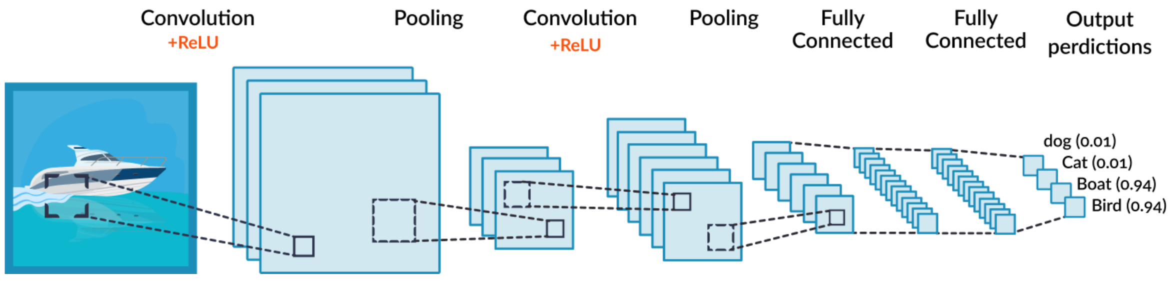


Figure 2: Features of each convolutional layer in a CNN

3.2 Workflow

A complete workflow diagram of the proposed method for object detection from a webacm is shown in the figure. Here, you can see after starting, we import all the necessary modules and prepare the model. After that, we download the SSD modules to detect images. On capturing the picture, the image in loaded into a numpy array and load the label map. A TensorFlow model is them loaded into the memory and start running the TF session and lastly, an object is detected with bounding boxes.

The diagram shows the complete picture for better understanding of the system of object detection from the webcam camera. When we give any picture to our model and connect to dataset, it will detect all the given object from the camera view in real-time.

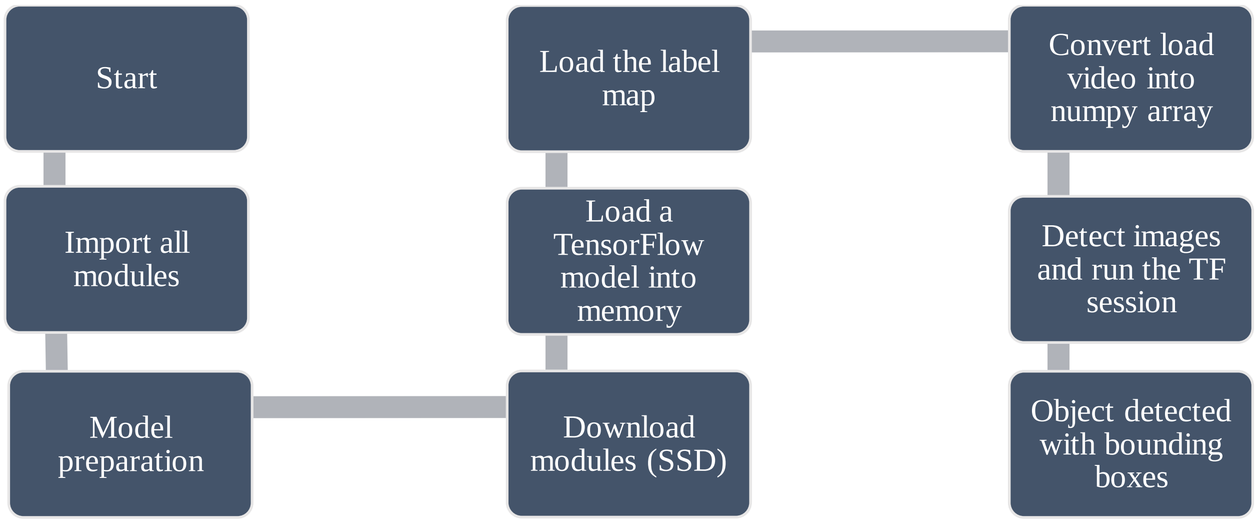

Start
Import all modules
Model preparation
Download modules (SSD)
Load a TensorFlow model into memory
Load the label map
Convert load video into numpy array
Detect images and run the TF session
Object detected with bounding boxes

Figure 3: Object detection from a webcam in our proposed model

3.3 Import All Modules

For complete the system we need to import many modules and set up virtual environment and some package which are given in step by step

- Firstly, we need to install the Anaconda virtual environment and set up some packages like Spyder, OpenCV - python, pandas, matplotlib, protobuf, pillow, lxml, NumPy.
- We need to configure PYTHONPATH to our environment.
- Secondly, we will install TensorFlow- GPU v2.0 with CUDA v11.0 and cuDNN v8.0

- Thirdly, we will download the official TF object detection API from google and set up the directory.
- Finally, compile the protobuf file and run the setup.py command in anaconda command prompt.

3.3.1 TensorFlow Object Detection API

TensorFlow Object Detection API is a package to determine object detection applications. It is a technique of detecting real-time objects in an image. The TensorFlow Object Detection API library includes multiple out-of-the-boxes object detection constructions like SSD, Faster R-CNN, and R-FCN (Region-based Fully Convolutional Networks).

In 2016, A Single Shot Multibox Detector (SSD) had been launched by investigators from Google. It is a quick single-shot object detector for several classes. Its structure is based on one feed-forward convolutional network. That leads to adopting straightforward and anchor stabilizers with no need for an additional stage for each proposal classification procedure. SSD's essential feature is the employ of multi-scale convolutional bounding box outputs connected to various feature maps at the highest of the network.

The main structure of SSD is extracted from the VGG-16 structure. VGG-16 is applied as the core network since it performs high-quality image classification functions and transfer-learning training to boost the results. Szegedy's project on MultiBox determines the bounding box method of SSD. The loss function of the MultiBox includes two essential parts that achieved their path to SSD. First is the confidence loss that estimates how confident the system in determining the bounding box. Simultaneously, the second part is location loss, which measures the distance of the network's predicted bounding boxes from the ground truth ones during the training mode.

For the location loss calculation, SSD uses smooth L1-Norm. Concerning the classification process, the SSD completes object classification. Thus, for each predicted bounding box, groups of N class predictions are determined for each possible class in the dataset. Besides, feature maps are a description of the dominant features of the image at several scales. Later, working MultiBox on multiple feature maps increases the probability of any object, whether big or small, is finally localized, detected, and correctly classified.

The dataset splits into the training and testing dataset. First, all the dataset is labeled by drawing a bounding box around the detected object. Then, save them as an XML file. The XML file converts to a CSV file, and TensorFlow converts all these files into “TFRecords” to make it readable for the TensorFlow application. The system uses SSD-Mobilenet as a per-trained model of the TensorFlow object detection API.

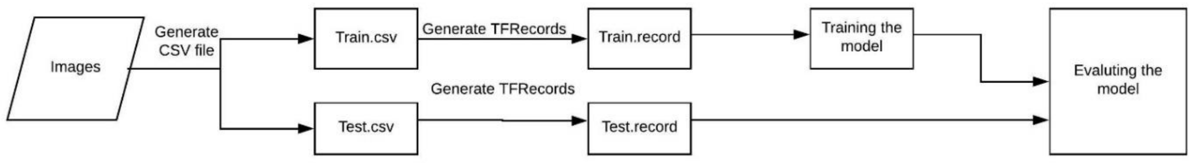


Figure 4: How TensorFlow Object Detection API works

3.4 Dataset

The server will be using a pre-trained SSD detection model trained on COCO datasets where we have 30k objects image with 90 objects. It will then test and the output class will get detected with an accuracy metrics.

We have implemented five different CCO-trained models to determine which one is the faster and accurate. From the below table (more on Chapter 4), we can see that faster_rcnn_inception_resnet_v2_atrous_coco is the slowest of all in regard of detecting objects, while ssd_mobilenet_v1_coco is the fastest. However, faster_rcnn_inception_resnet_v2_atrous_coco is the more precise than any other and ssd is the least precise model. Hence, this all depends on the application. If we want to use this model on a system that requires faster output we will use ssd, on the other hand, we will use the former one for accurate results. Now, for our model we will SSD for our model as we need fast results and we can work making the outputs more accurate by some other techniques.

Table 2: Comparison between five different COCO-trained models

Model Name	**Speed (ms)**	**COCO map [^]**	**Output**
ssd_mobilenet_v1_coco	30 (fast)	21	Boxes
SSD_inception_v2_coco	42 (fast)	24	Boxes
rfcn_resnet101_coco	92 (medium)	30	Boxes
faster_rcnn_resnet101_coco	106 (medium)	32	Boxes
faster_rcnn_inception_resnet_v2_atrous_coco	620 (slow)	37	Boxes

3.5 Preprocessing

Filtering: Convolution of an image with numerous filters can work out several operations like edge detection, blur and sharpen by using filters.

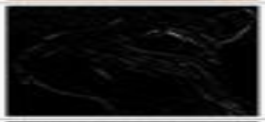
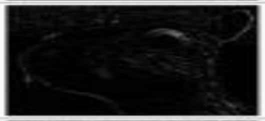
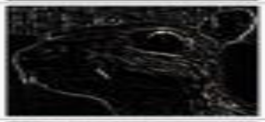
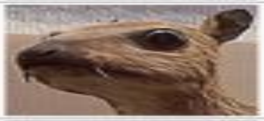
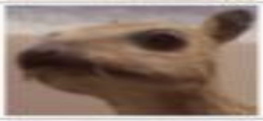
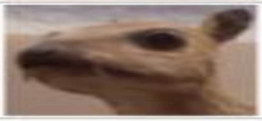
Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Figure 5: Some filter operation and their mask value

It can happen that filter does not quite fit the input image. If the image does not fit with the input image we can two things:

- Padding the image to fit with the input picture
- Crop some part of the image so the filter can fit.

3.6 Deep learning neural network algorithm

Since our work is object recognition from image or webcam input so we have used Convolutional neural network (CNN) for our work. Now, we will briefly discuss it.

Convolutional neural network (CNN):

Convolutional neural network is one of the primary kinds to recognize pictures and classify them. CNN is widely used in object detection systems. It is very easy to implement it that is the reason we choose this neural network. The below diagram depicts a complete process of how CNN works.

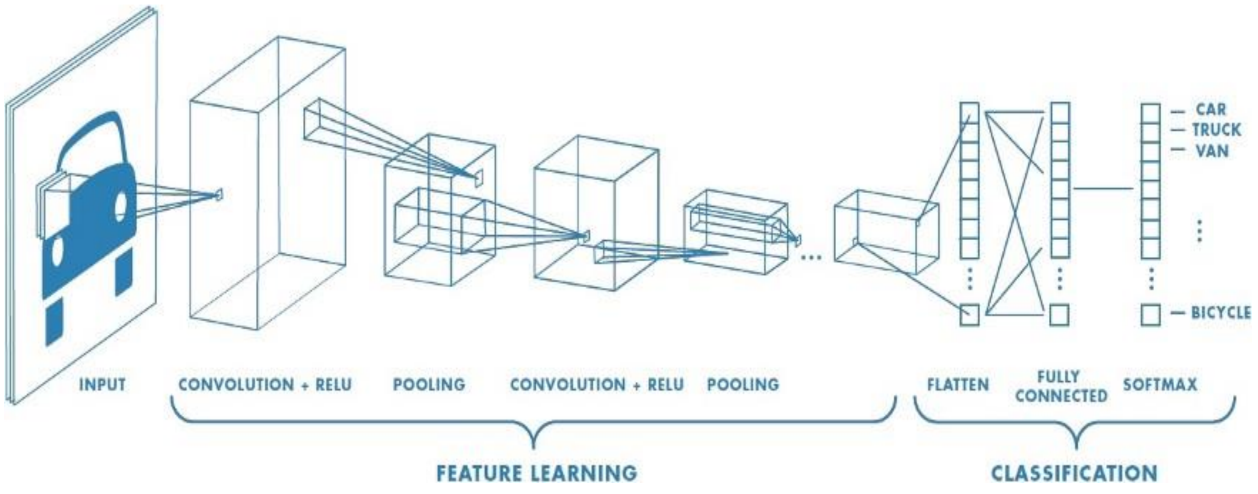


Figure 6: Complete diagram of CNN architecture

3.6.1 Convolution Layer

Convolutional layers are the significant structure blocks utilized in convolutional neural organizations. A convolution is the straightforward use of a channel to an information that outcomes in an enactment. Rehashed use of similar channel to an information brings about a guide of actuations called an element map, showing the areas and strength of a recognized component in an information, for example, a picture.

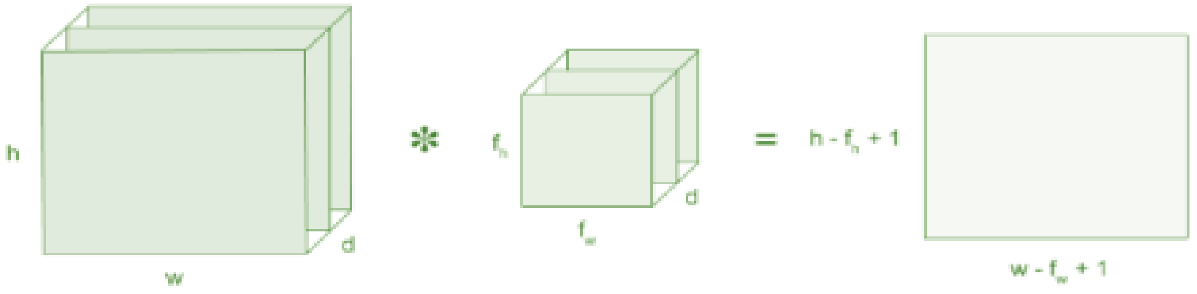


Figure 7: Output image dimension calculation

Let us consider a 5 x 5 with input image pixel values 0 and 1 shown in below, and a filter size with 3*3 then we apply the filtering. Then the convolution matrix multiplies with the filter matrix and it is called **"Feature Map"** as output shown in below. It will calculate pixel by pixel and multiple with filter value and sum them and produce an output pixel.

After applying filter, the output image will be 3*3 size as we apply a filter of size 3*3. Output image pixel will reduce size. The given figure will give better understanding.

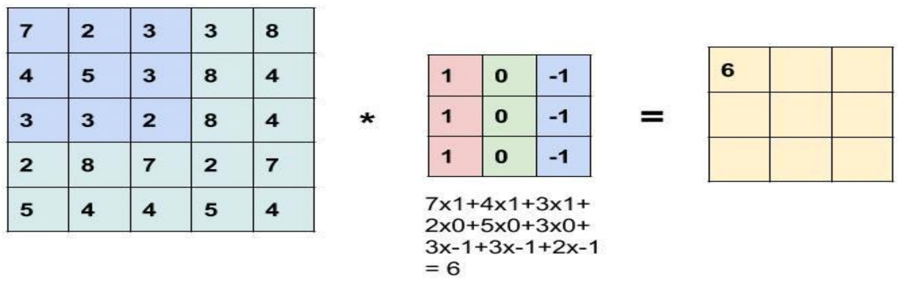


Figure 8: After applying filter in input image

3.6.2 Pooling Layer

Pooling layers section will reduce the number of parameters when the images are too large for the frame. Spatial pooling also called subsampling or down sampling which reduces the dimensionality of each map but retains important information. Spatial pooling can be of different types:

- Max Pooling
- Average Pooling
- Sum Pooling

3.6.3 Max Pooling

Max pooling is a system where the max bit value will come as an output and reaming bit value will not come. The given figure will give you a better understanding.

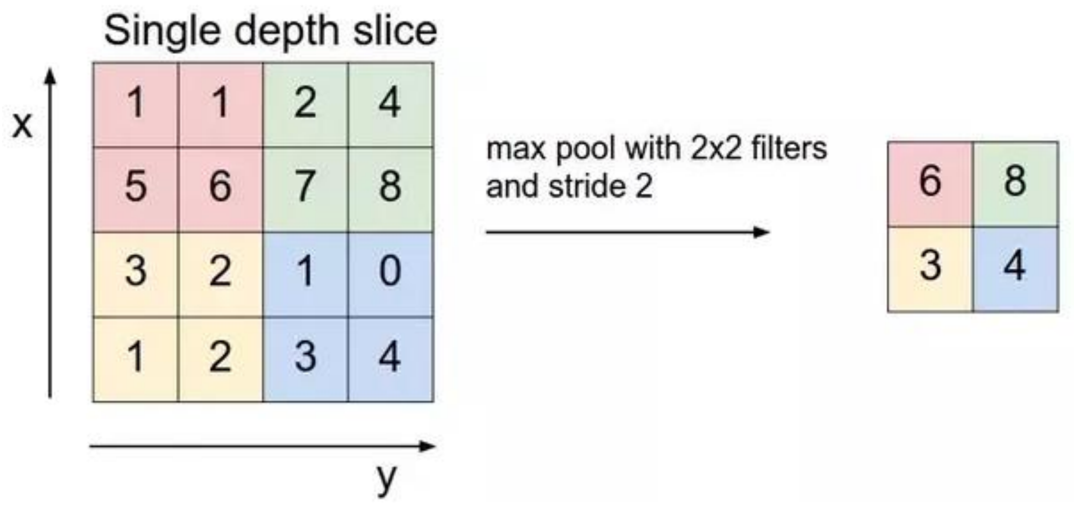


Figure 9: Max pooling workflow

The max pooling will give you the highest value information so it will be noisy as we lose the low value. It will call downsampling.

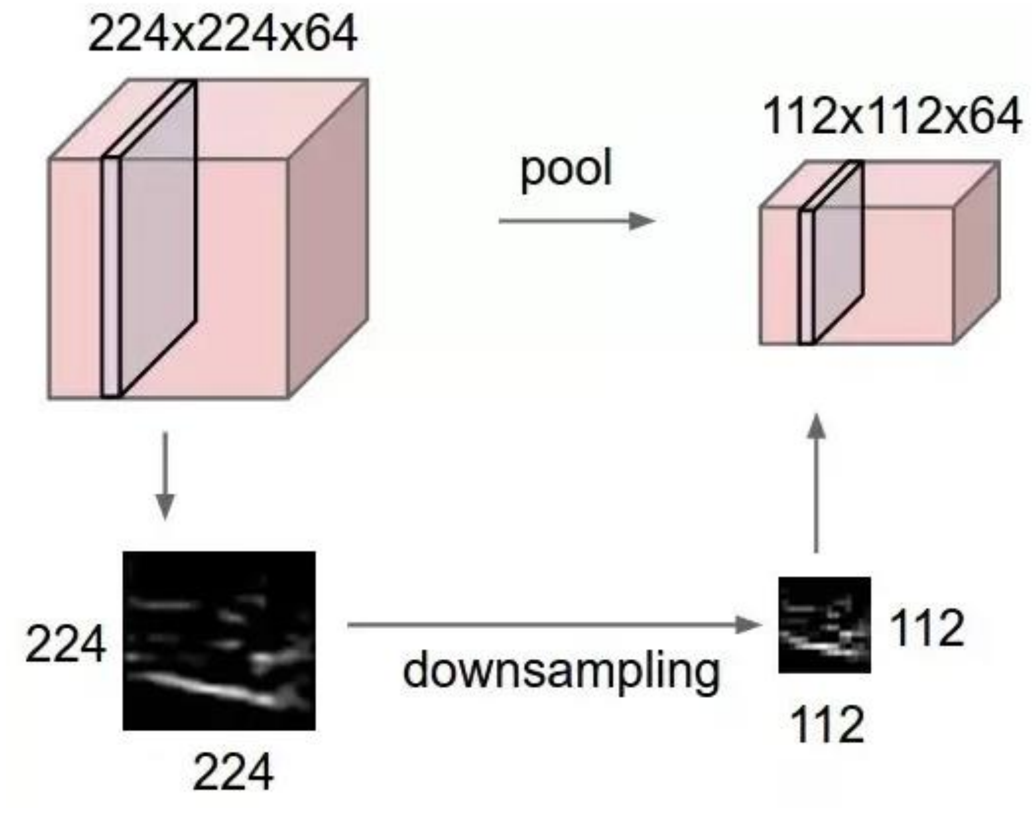


Figure 10: Max pooling downsampling

3.6.4 Average Pooling

The average pooling will average the neighbor four pixel value and give it as an output value. It will be better as we will not lose much information from it. The given figure will give you detailed information.

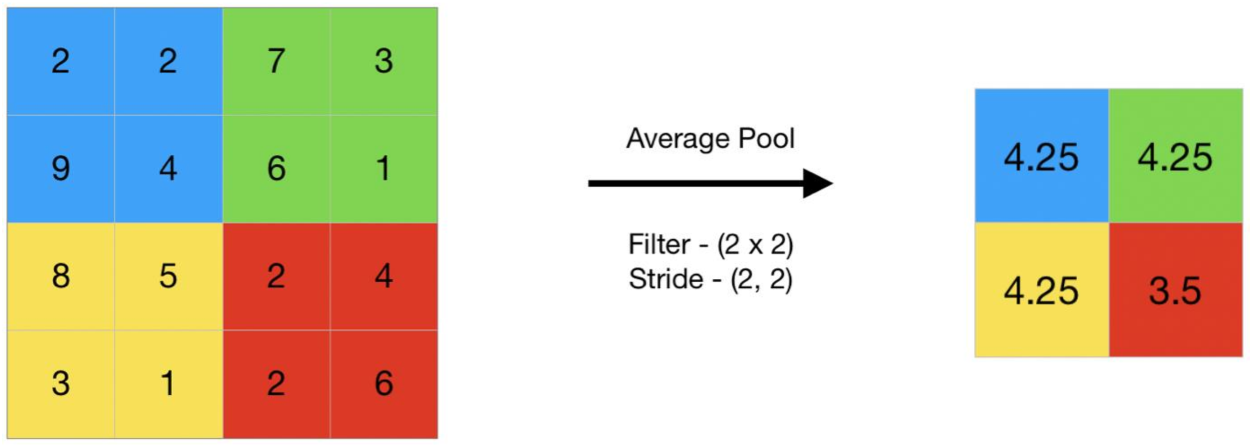


Figure 11: Average pooling workflow

3.7 Modules Download

We need to select a module to run our model. TensorFlow gives a few object detection models. A few models, (for example, the SSD-MobileNet model) have an architecture that takes into consideration quicker detection yet with less percent accuracy, while a few models, (for example, the Faster-RCNN model) give more slow identification however with more percentage accuracy.

For our model we choose SSD architecture firstly then we implement Faster RCNN models also.

We can download all the modules from the folling link : https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md

3.7.1 SSD Architecture

SSD consists of two major components, one is the head while the other is the backbone. The backbone model is the trained image classification network that is trained by ImageNet. With additional convolution layers to the backbone the head is formed. The output of this model is the bounding boxes and object classes in spatial location. Hence, we can work with a deep neural network that can extract semantic meaning from an input image all the while storing the spatial structure of the image at a lower resolution.

If we give an image, the backbone gives 256 7x7 feature maps in outcome with the help of ResNet34. Then SSD sections the picture with help of a grid and provide one grid for detecting objects in that region. It mainly predicts the class and location of the particular object in a specific region. In [7] have a brief discussion on SSD architecture.

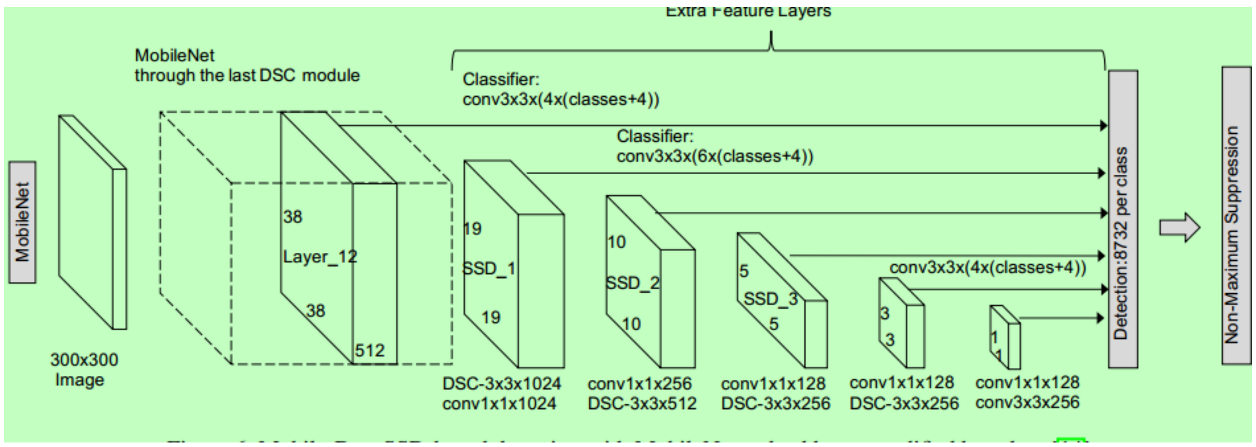


Figure 12: SSD Architecture

3.8 Create Label Map and Configure Training

The label map mentions to the mentor what each item is by characterizing a planning of class names to class ID numbers. Every class was predefined. There are 90 objects which has different id number. If [id: 1] then object [name: person], [id:2] then object [name: dog] and [id:3] [name: cat] etc. Every object has a unique id number. At last, the object detection preparing pipeline should be designed. It characterizes which model and what boundaries will be utilized for preparing.

3.9 Convert Load video into NumPy array

A video is noting but a collection of images with time. A computer cannot an image as we can see it as human beings. So, we need to convert an image into array of numbers. What usually a machine does image is represented as an array of numbers that corresponding to the intensity of each pixel.

Each pixel like a matrix. Each pixel represented a single bit pixel value in range of 0 or 255. Where 0 bit means black color and 255 means white. In-between all values are represented in different gray color. In our project we did it with python imaging library. For this we take the help of pillow.

3.10 Object Detect with Bounding Boxes

After saving the image into NumPy array then our system will match it with our pre-trained dataset and try to recognized it. It will display the detected object with bounded boxes and the accuracy of the detection with percentage. It shows in the given Fig. 14

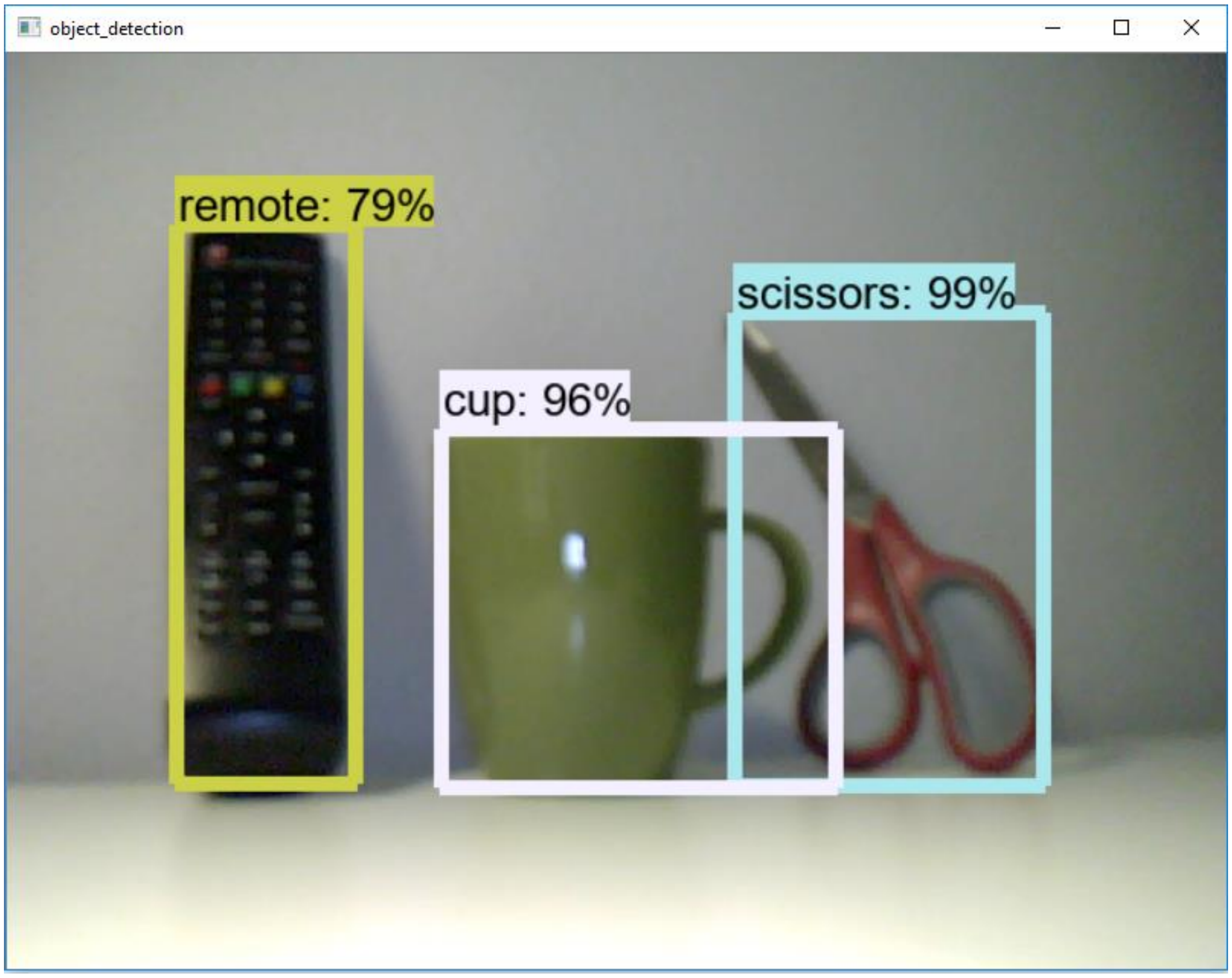


Figure 13: Object detect with bounding boxes

3.11 Voice Generation

We have implemented a feature that acknowledge the object detected after recognizing an object. We have used PYTTSX3 for the voice generation module. Pyttsx3 is a python library that converts text into speech. When the model detected, a function pyttsx.init() in provoked and the tool convert the name of the objects into speech.

There are certain libraries in Python that help us implementing the voice module so did not have to do the code from scratch. Moreover, some libraries even allow to give out specific voice outputs with the help of pytorch which is a machine learning library.

<u>CHAPTER 4: EXERIMENTAL RESULTS</u>

4.1 Single Object Detected from Webcam

Here we try to detect single object from webcam. We have tested it on remote, vase, clock, cup, book, mouse, cell phone, chair, bottle, car. You can see there is a bounding box around the image that shows the accuracy of the detected object. Sometimes, if the object is too blurry the percentage of accuracy drops. But for that, we have used some algorithms to filter out blurriness and sharping the image to the highest possible resolution. Tge SSD mobilenet v1 and TensorFlow we implemented is used to detect the object that we see below and get an accuracy of 94% for single image. In figure 14-24 we have the output of single object detect from webcam.

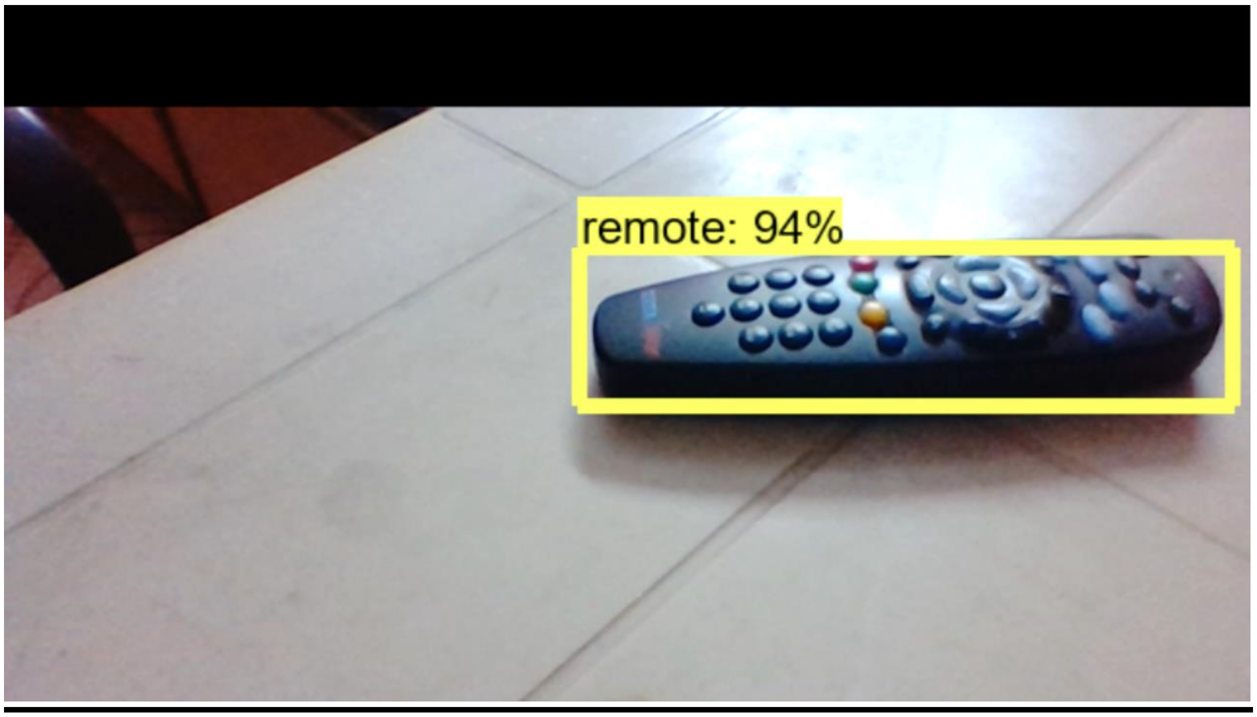


Figure 14: Remote Detection from webcam

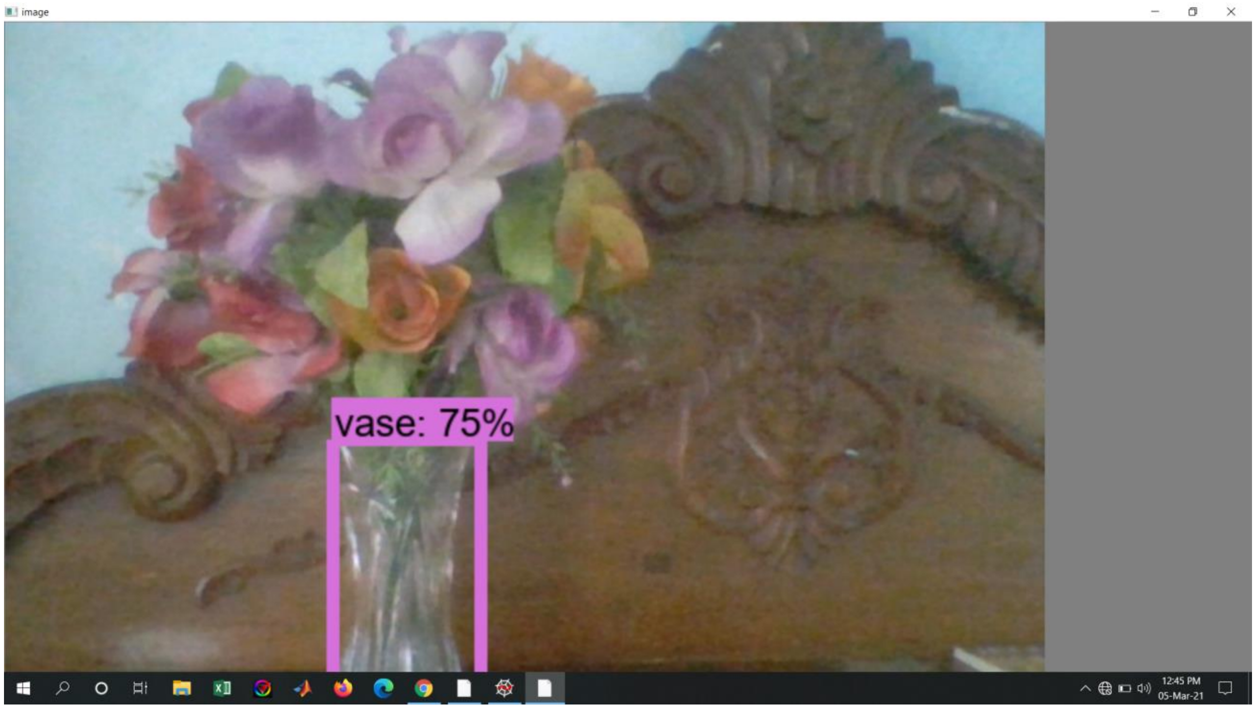
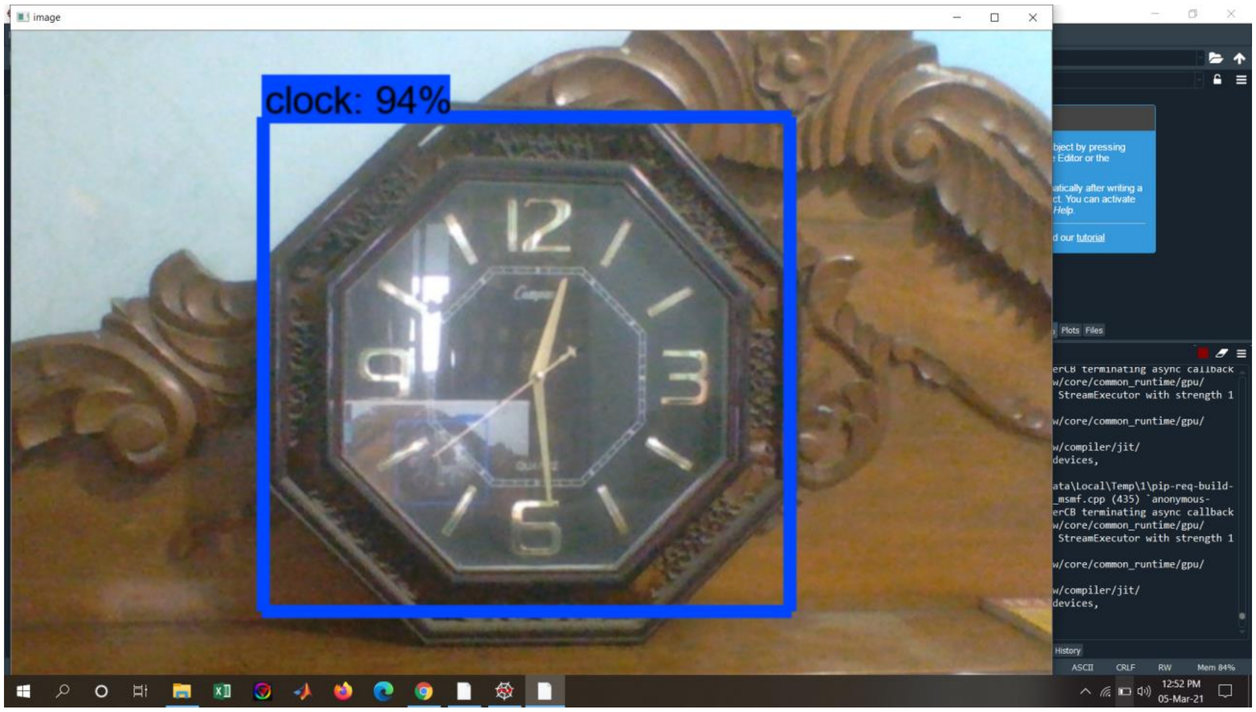


Figure 15: Vase Detection from webcam



bject by pressing
: Editor or the

atically after writing a
ct. You can activate
Help.

d our tutorial

```
erCB terminating async callback
w/core/common_runtime/gpu/
 StreamExecutor with strength 1

w/core/common_runtime/gpu/

w/compiler/jit/
devices,

ata\Local\Temp\1\pip-req-build-
_msmf.cpp (435) `anonymous-
erCB terminating async callback
w/core/common_runtime/gpu/
 StreamExecutor with strength 1

w/core/common_runtime/gpu/

w/compiler/jit/
devices,
```

Figure 16: Clock Detection from webcam

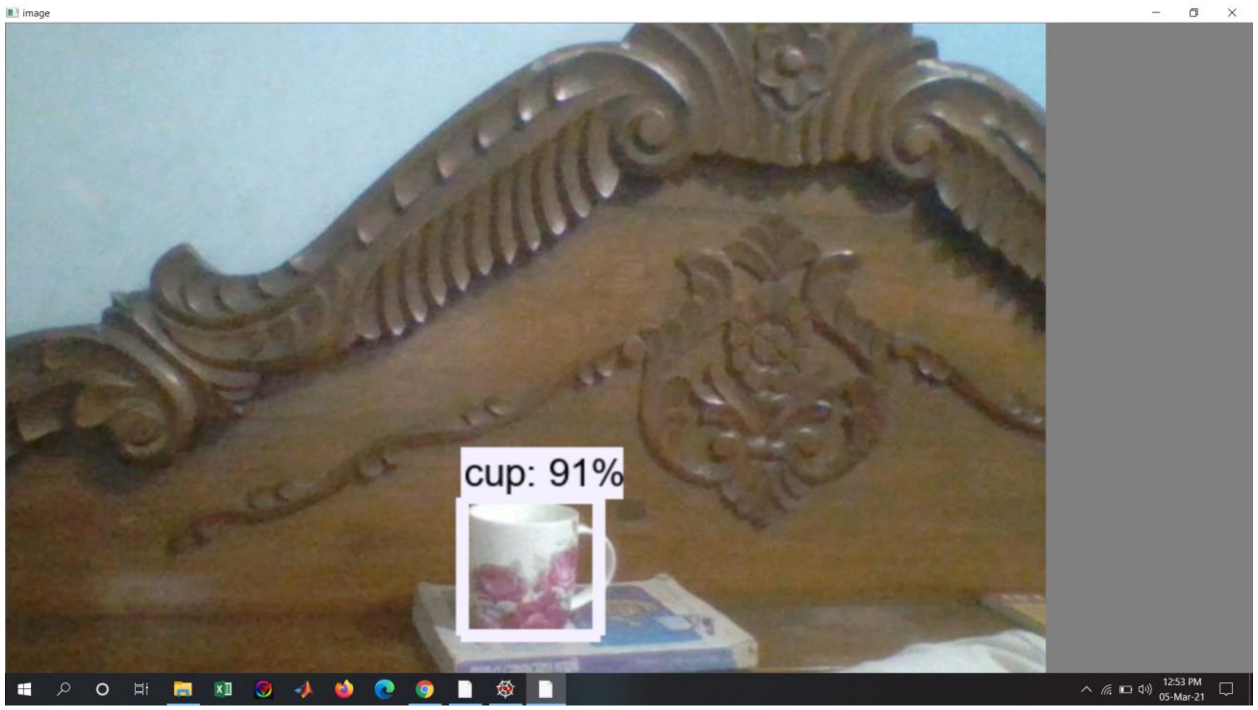


Figure 17: Cup Detection from webcam

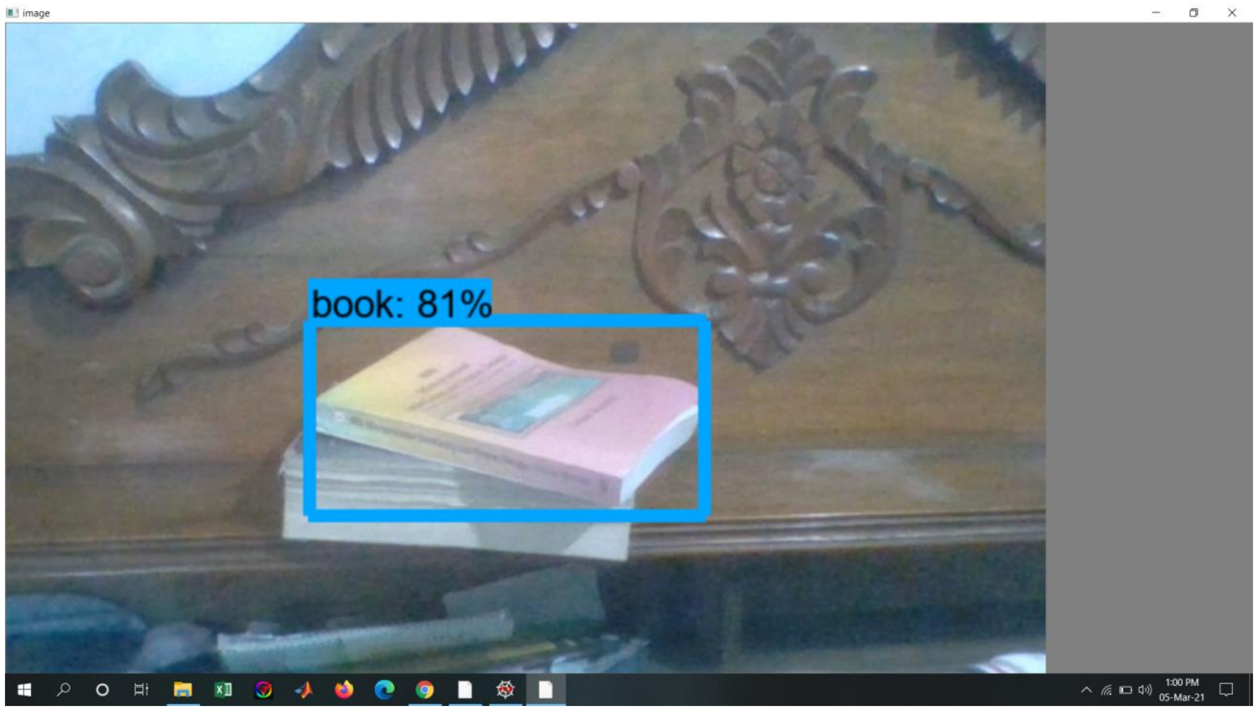


Figure 18: Book Detection from webcam

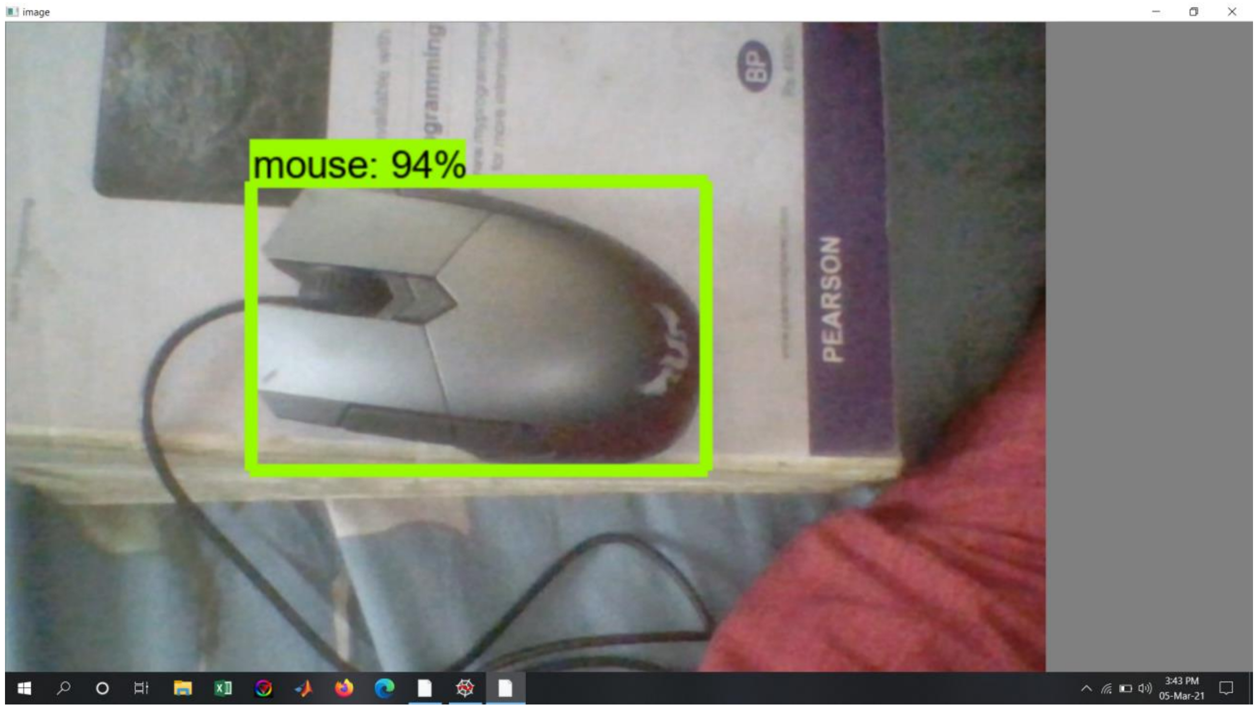


Figure 19: Mouse Detection from webcam

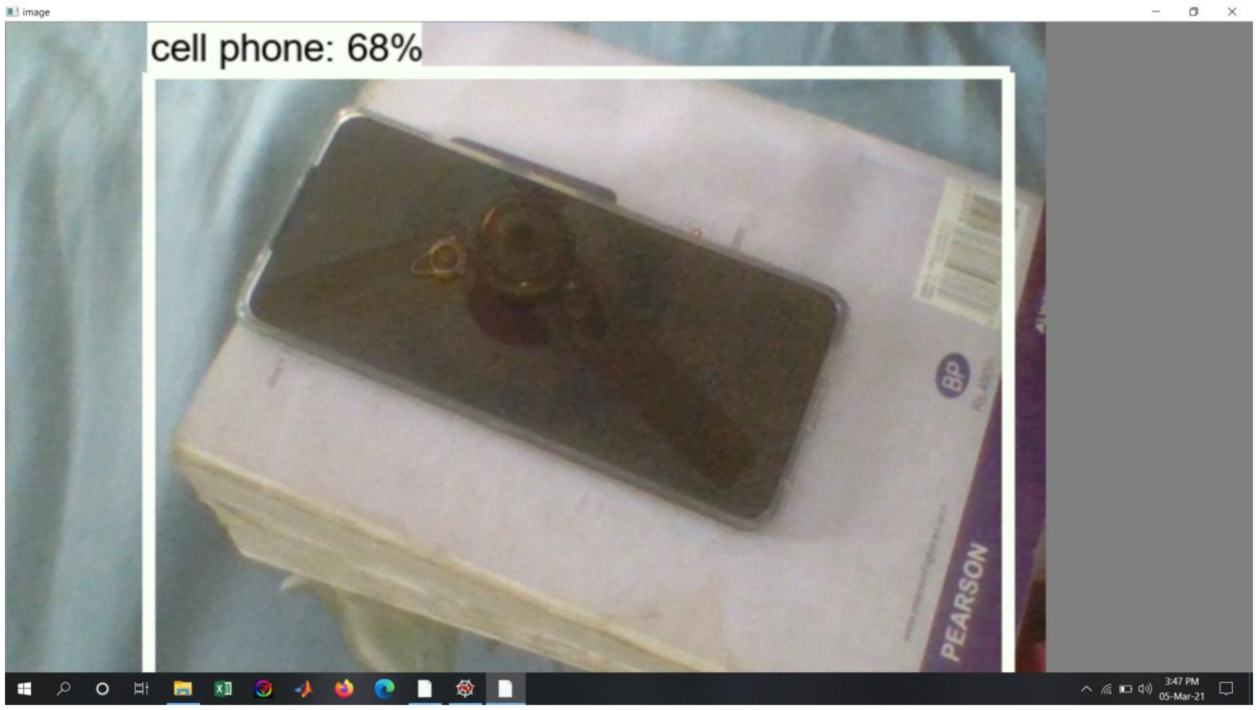


Figure 20: Cell phone Detection from webcam

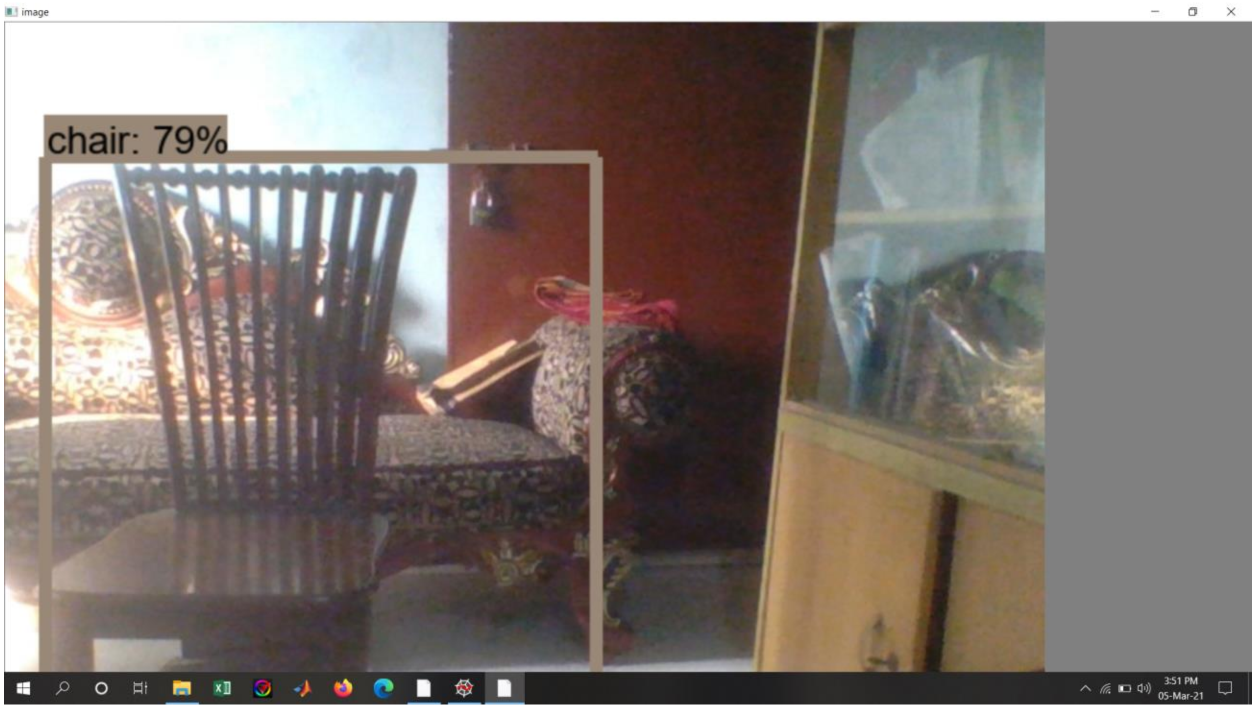


Figure 21: Chair Detection from webcam

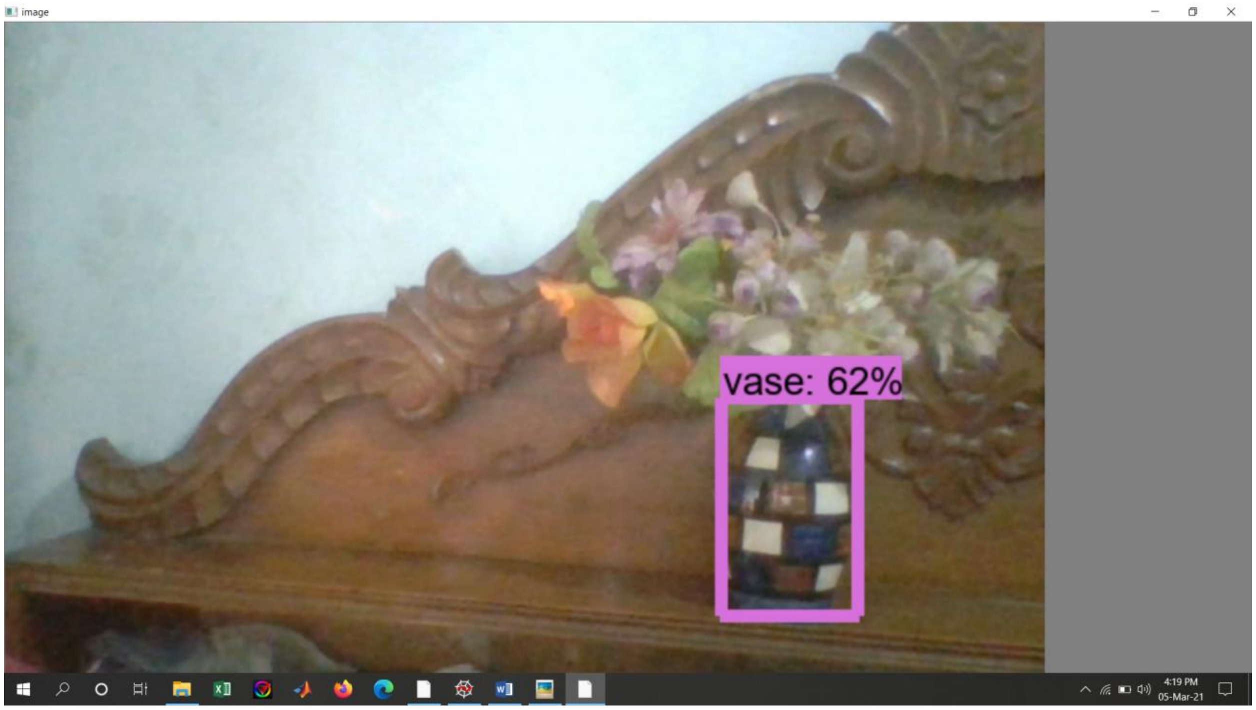
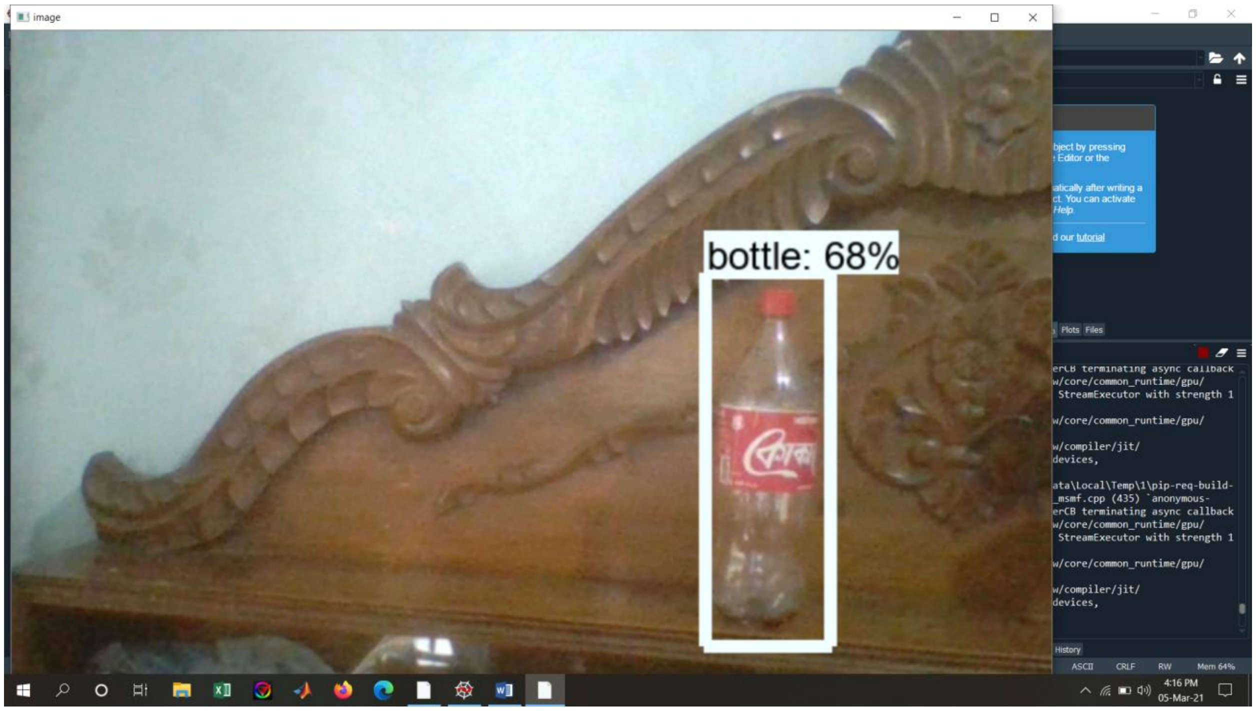
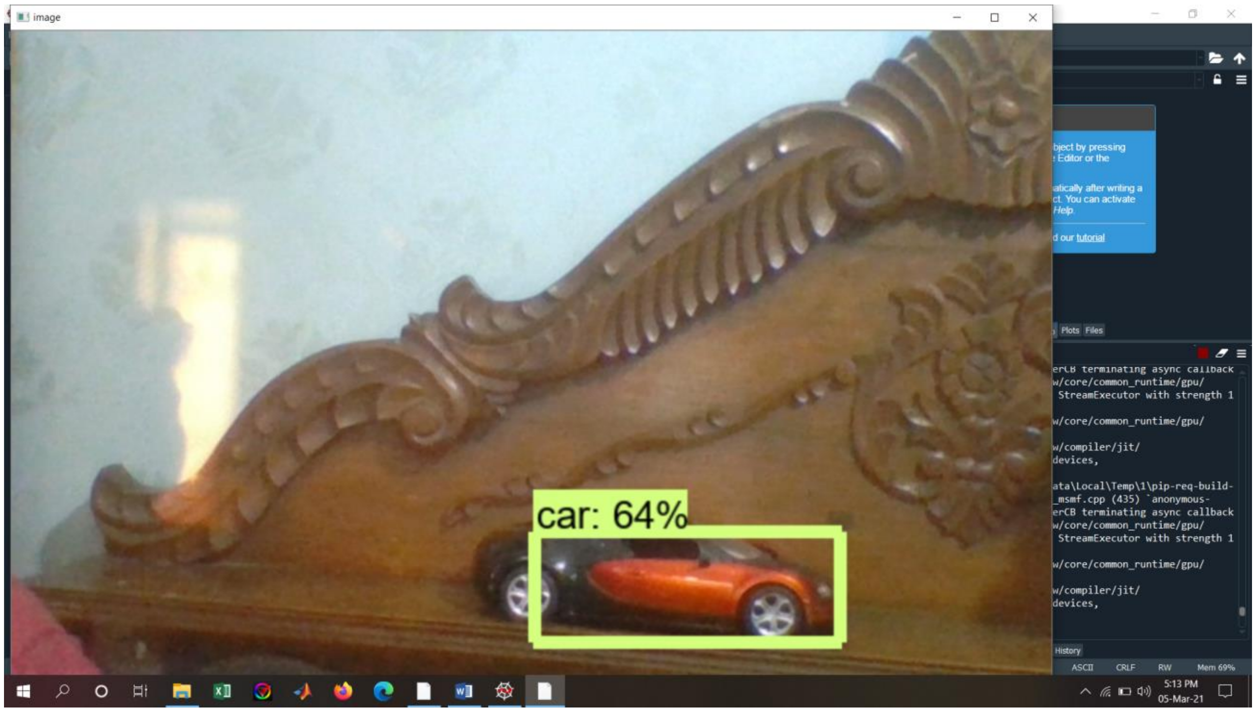


Figure 22: Vase Detection from webcam



```
erCB terminating async callback
w/core/common_runtime/gpu/
 StreamExecutor with strength 1

w/core/common_runtime/gpu/

w/compiler/jit/
devices,

ata\Local\Temp\1\pip-req-build-
_msmf.cpp (435) `anonymous-
erCB terminating async callback
w/core/common_runtime/gpu/
 StreamExecutor with strength 1

w/core/common_runtime/gpu/

w/compiler/jit/
devices,
```

Figure 23: Bottle Detection from webcam



```
erCB terminating async callback
w/core/common_runtime/gpu/
 StreamExecutor with strength 1

w/core/common_runtime/gpu/

w/compiler/jit/
devices,

ata\Local\Temp\1\pip-req-build-
_msmf.cpp (435) `anonymous-
erCB terminating async callback
w/core/common_runtime/gpu/
 StreamExecutor with strength 1

w/core/common_runtime/gpu/

w/compiler/jit/
devices,
```

Figure 24: Car Detection from webcam

The below table will give an idea about the accuracy of single object detection from webcam

Table 3: Single object accuracy

Object	Accuracy (%)
Remote	94
Base	75
Clock	94
Cup	91
Book	81
Mouse	94
Cell phone	68
Chair	79
Bottle	68
Car	64

From the table we can assume that the accuracy of detecting different single object from object is different. The main reason it is limited dataset. We used COCO dataset that contains 3,00,000 segment images with 80 different objects with very precious location labels. Each picture contains around 7 objects overall, and things show up at exceptionally wide scales. As supportive as this dataset is, object types outside of these 80 select classes won't be perceived if training exclusively on COCO.

4.2 Multiple Object detected from webcam

Our model detects objects with SSD mobilenet v1 using the webcam having multiple objects in its view and it detect the objects in real-time along with highest accuracy. The below image is a fine example that will demonstrate how accurate the detection is. The v1 speed is 30ms and COCO MAP is 21 that means the objects are detected in less than a second. However, the accuracy is quite low for the clock in the background with only 58% that shows even if SSD mobilenet v1 is fast, it is not that accurate.

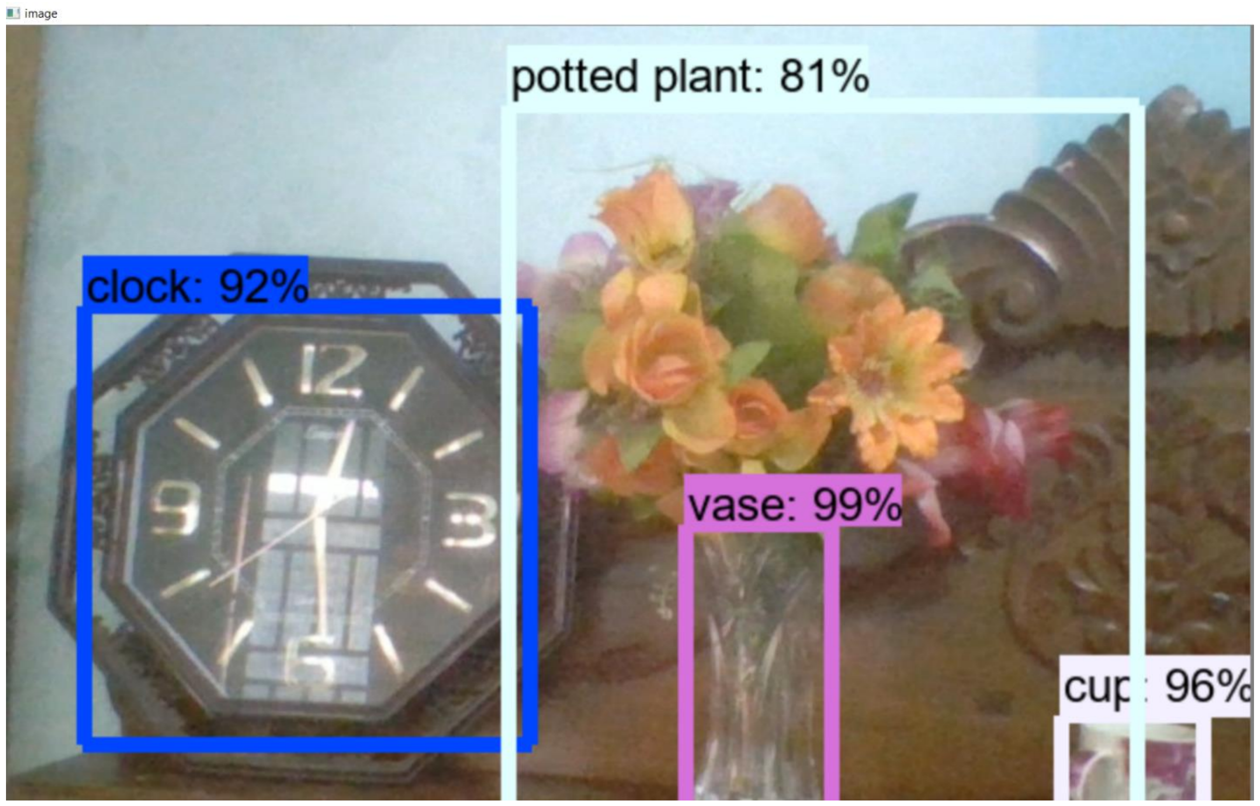


Figure 25: Multiple objects detect from webcam

4.3 Multiple object detection with SSD inception v2

In TensorFlow SSD inception v2 uses inception v2 as feature extraction named vgg16. The feature layers extracted in SSD mobilenet named feature extraction but inception v2 produced extra layer with different resolution. We have used SSD inception v2 in our model because v2 speed is 42ms and COCO MAP is 24 and it is more precise than SSD mobilenet v1. This model also gives the better accuracy from SSD mobilenet. The below image will give you a better demonstration.

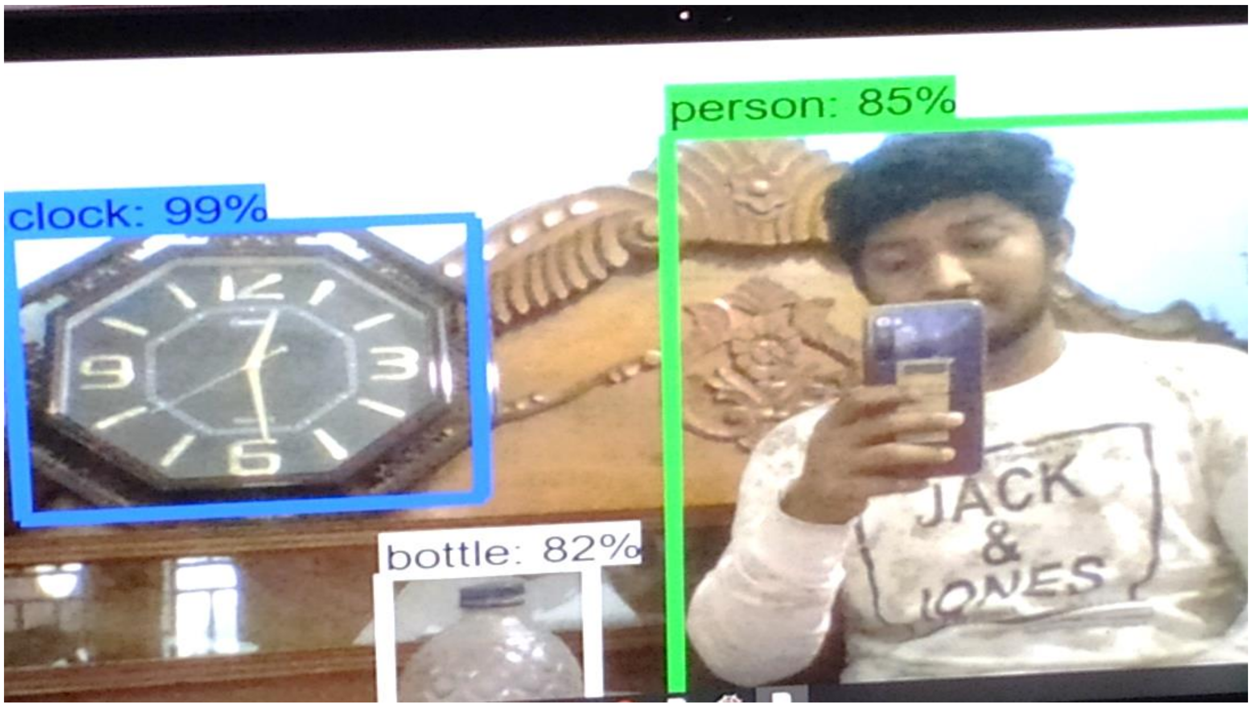


Figure 26: Multiple object detect from webcam using inception v2

4.4 Accuracy comparison between SSD mobilenet VS SSD inception v2

As SSD inception produced more layers and extracted more feature it will give more accuracy. The differences of accuracy given in Tab.

Table 4: Accuracy comparison between SSD mobilenet and SSD inception

Object	SSD mobilenet (%)	SSD inception v2(%)
Person	92	85
Clock	58	99
Bottle	78	82

From the table we can see that for clock and bottle the accuracy is better in SSD inception V2 person identification the better accuracy get from SSD mobilenet. Finally, we can say that SSD inception is better model than SSD mobilenet.

4.5 Multiple objects detect using faster rcnn inception v2

For Faster rcnn inception v2 speed is 58ms and COCO MAP is 28. It is slower than first two model. But as you can see in the below picture, that faster rcnn gives higher accuracy than the previous one. We can get better accuracy with less training time.

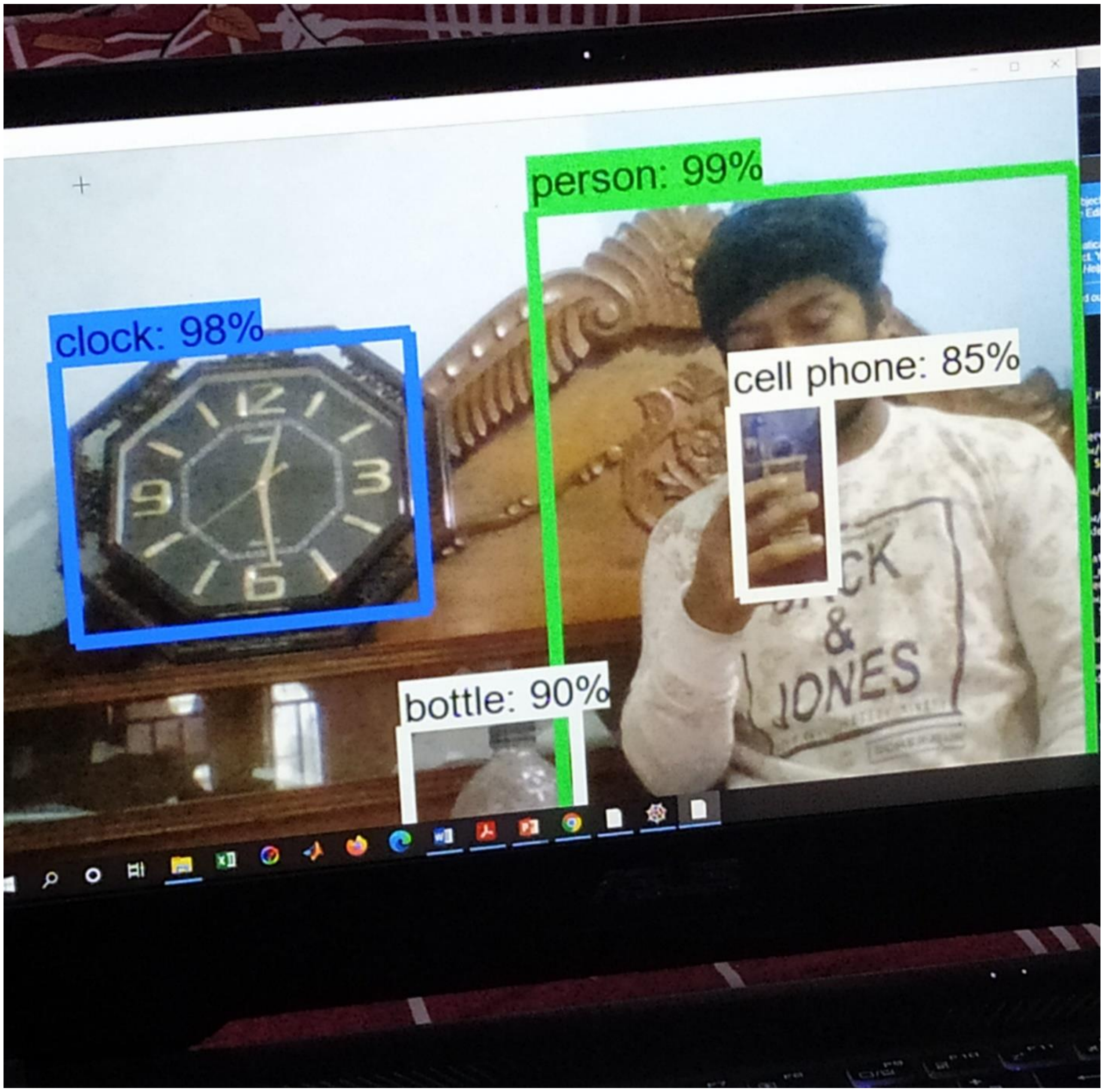


Figure 27: Multiple object detect from webcam using faster rcnn inception v2

4.6 Multiple objects detect using faster rcnn inception resnet v2 atrous

For this implementation, the speed is 620ms and COCO MAP is 37. It detects the object very fast but it is slower than other model we used. It takes more time to start the process. The figure below demonstrates the outcome that shows this is by far have the highest accurate results.

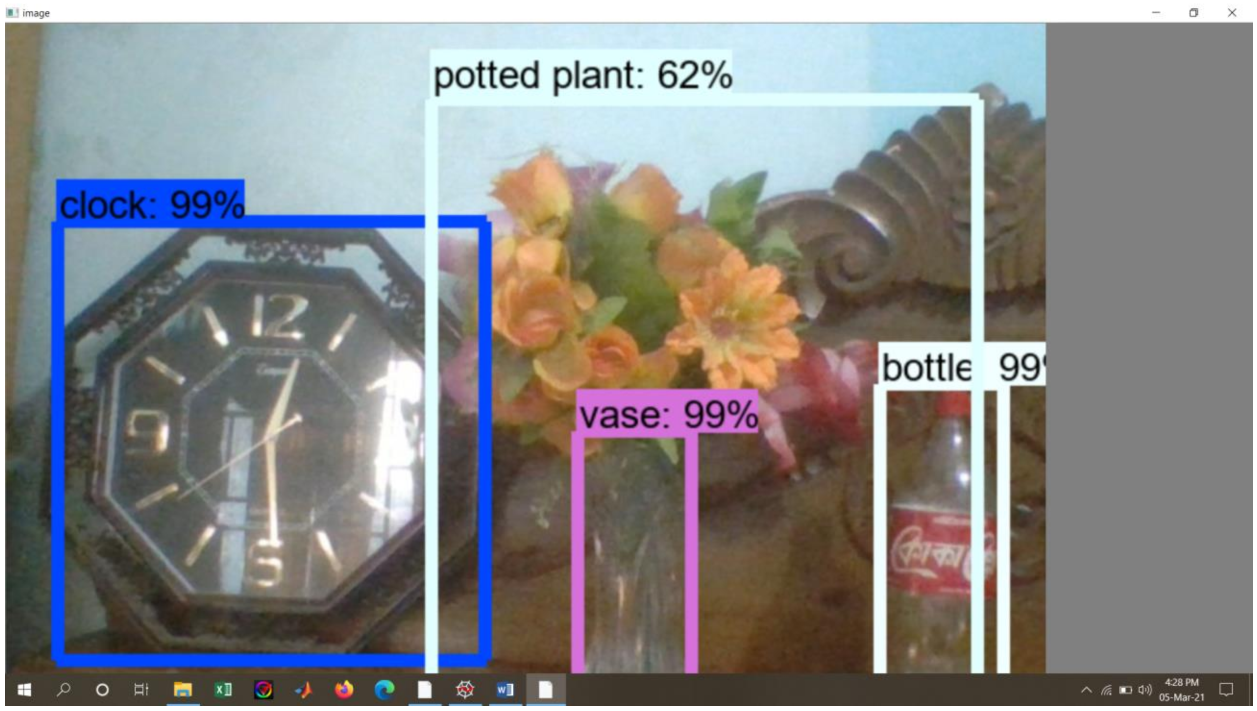


Figure 28: Multiple object detect on webcam using faster rcnn inception resnet v2

4.7 Multiple objects detect using faster rcnn inception resnet 101

This model is the best for the detection of objects in a single frame. With speed of 106ms and COCO MAP 32, it is a highly precise model with almost over 98% accuracy for all the objects in the frame regardless of their position.

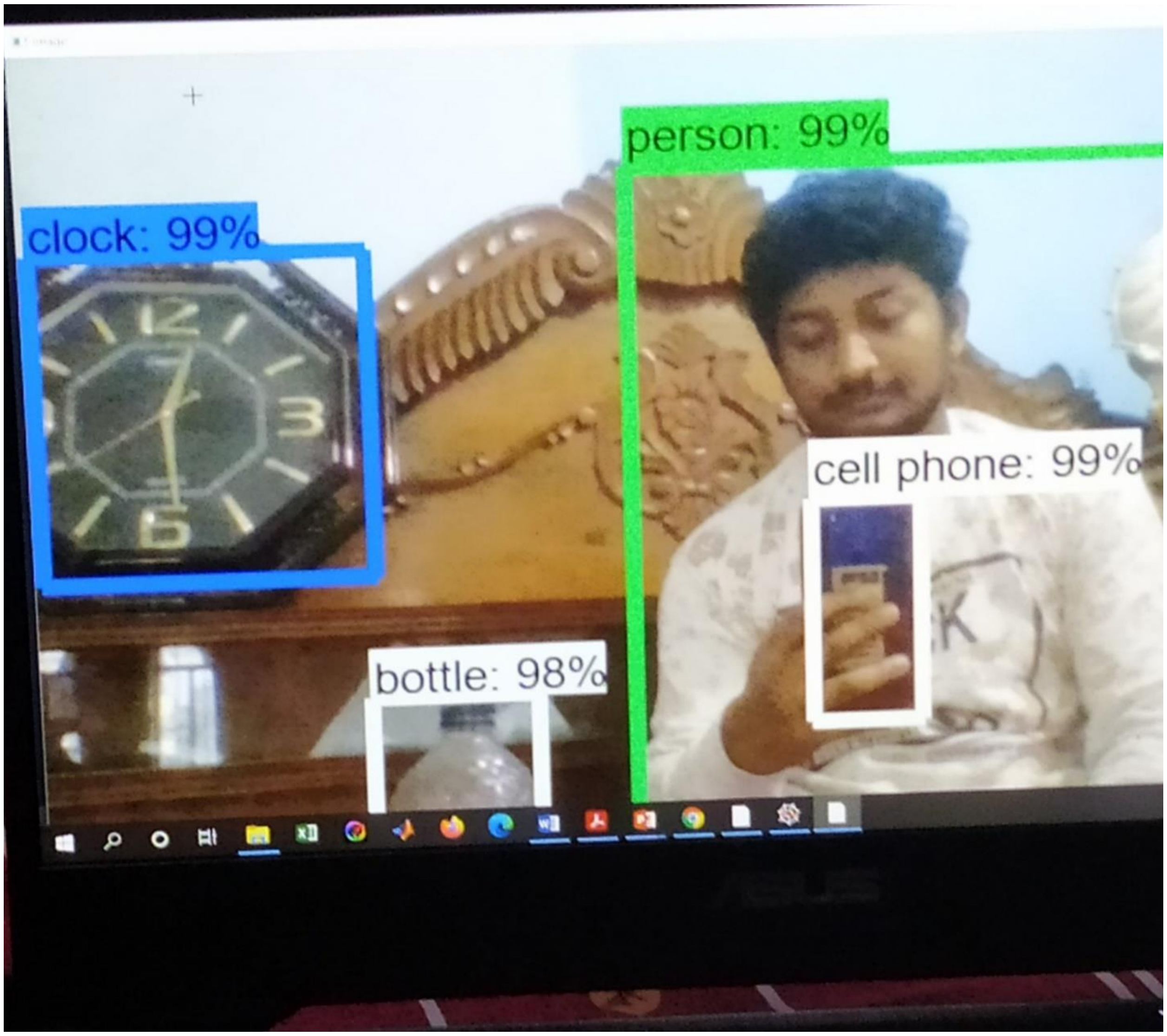


Figure 29: Multiple object detect on webcam using faster rcnn resnet 101

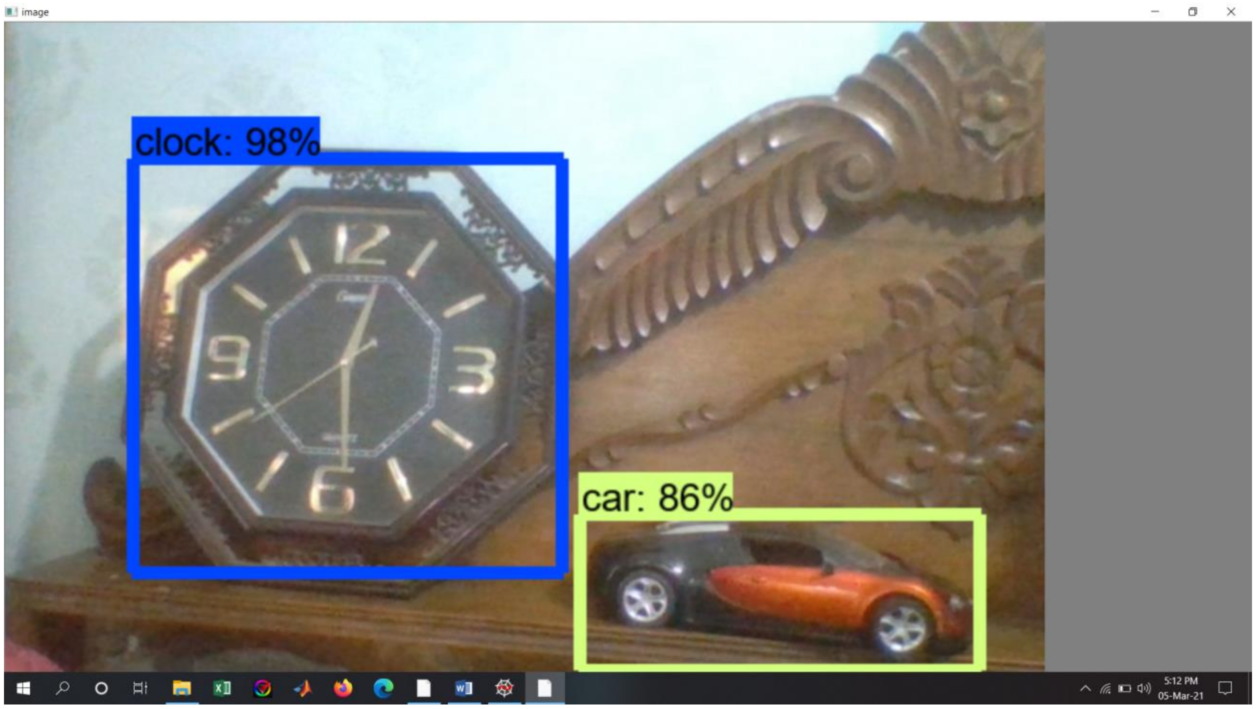


Figure 30: Clock and Car Detection using SSD inception v2

Now, we can deduce that the faster rcnn inception resnet 101 is the precise of all the model we have implemented. However, this is the one that takes most time to detect objects too. Suppose, when an object is to be detected in real-time in a time constraint projects like as assisting blinds or detecting accidents, this model cannot be used. But if there is no hurry for detecting an object this model will be the best.

We have also implemented a voice module that calls out the name of the object detected. This feature is a very useful for detection systems like blind assistant and alert systems.

4.8 Comparison between faster rcnn inception, faster rcnn inception resnet v2 and faster rcnn inception resnet 101

In this section we are going to compare accuracy the three model and see the differences. The comparison table is given below.

Table 5: Comparison accuracy between faster rcnn inception, faster rcnn inception resnet v2 and faster rcnn inception resnet 101.

Object	Faster rcnn inception (%)	Faster rcnn inception resnet v2 (%)	Faster rcnn inception resnet 101 (%)
Person	99	99	99
Clock	98	97	99
Mobile	85	99	99
Bottle	90	97	98
Car	88	94	97
Vase	79	93	96

From the table we know that Faster rcnn inception resnet 101 produce more accuracy from the other two models. It almost gives 99% accuracy for every object. From the comparison we can say that Faster rcnn inception resnet 101 model is better than other models. We already know from Tab.2 that this model speed is 106 ms and COCO map is 32. We can easily use it for high security reason as it detects object well and it can detect real time object faster.

CHAPTER 5: CONCLUSION

In this chapter, we are will discuss the work we have done so far and the issues we faced on this project and how we can make our model work faster and further develop the project.

5.1 Future Work

In future, we will try to implement the hardware version of it using raspberry pi and android app. The hardware version can be further extended to build a robot for various purposes, but it can be bit expensive as all the products are not available in our country. Moreover, with android app we can add features like filtering out types of objects to be detected. For example, in the user interface if we choose human, the model will detect only humans. In this way, the project will be more useful and can be implemented for various purposes.

5.2 Limitations

Though we can detect multiple object at a time. This module has some limitations. The limitations are:

- If we rotate the image or change the angle, the CNN have difficulty to classify the image.
- As the size of computer is small, it cannot detect more than 6/7 objects at a time.
- While running faster rcnn models, the system becomes slower as it need high performance GPU.
- If a computer does not have high performance GPU then it will take more time to train the image as CNN has several layers.

5.3 Conclusion

In conclusion, our project is an aid to the monitoring to various problems at single system. Using this project, we can prevent accidents, illegal activities, help others, ease household activities, and many others. In some other works [9-10] we have studies and not mentioned, they have used SSD mobilenet v1 with pre-trained coco dataset where there was 30k dataset with 90 objects. Although the model is fast but the accuracy was not satisfactory with around 87% and this cannot promptly detect multiple objects in a single frame. So, we want to build a model which will reduce the problem and detect object more precisely. Thus we have tried applying different model like SSD inception v2, faster_rcnn_inception_resnet_v2, faster_rcnn_inception_v2, faster_rcnn_resnet101 to analyse diffèrent outcome CocoAPI, that we used, is a dataset which is having as large number of objects in it which are proved to be very useful before in many programs developed before. It can help us identify as large number of objects in the frame. Also, our project if combined with some few programs can be useful to find the several problems such as fires as well as theft and unrecognized face entry for houses and offices.

It is a system which can give you prevention from several unwanted activities like theft, unprotected entrance, etc. Lastly, it is involved with many new technologies which is very recent to be developed and are proving to be boon for several other works which are related to other technical backgrounds. It is combined with image processing as well as Object Detection techniques which are very helpful in the today's present scenario and are very accurate which itself can perform several difficult tasks replacing many human eyes.

For example, a police officer may have to look at a CCTV footage to dig out accidents or particular actions. With a specific object detection model like human detection model or license plate recognition model, the officer can easily find what he/she is looking for without wasting time and energy.

References

1. En.wikipedia.org. 2020. *Object Detection*. [online] Available at: <https://en.wikipedia.org/wiki/Object_detection> [Accessed 11 December 2020].
2. R. Phadnis, J. Mishra, and S. Bendale, "Objects Talk - Object Detection and Pattern Tracking Using TensorFlow," 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), 2018.
3. R. Raj, S. Michael, and D. Femi, "Forest Monitoring System through RFID and Tensorflow Object Detection," 2019 International Carnahan Conference on Security Technology (ICCST), 2019.
4. Kumar, R., Lal, S., Kumar, S. and Chand, P., 2014, November. Object detection and recognition for a pick and place robot. In *Asia-Pacific world congress on computer science and engineering* (pp. 1-7). IEEE.
5. Ramík, D.M., Sabourin, C., Moreno, R. and Madani, K., 2014. A machine learning based intelligent vision system for autonomous object detection and recognition. *Applied intelligence*, *40*(2), pp.358-375.
6. Wang, R.J., Li, X. and Ling, C.X., 2018. Pelee: A real-time object detection system on mobile devices. *arXiv preprint arXiv:1804.06882*.
7. Cao, G., Xie, X., Yang, W., Liao, Q., Shi, G. and Wu, J., 2018, April. Feature-fused SSD: Fast detection for small objects. In *Ninth International Conference on Graphic and Image Processing (ICGIP 2017)* (Vol. 10615, p. 106151E). International Society for Optics and Photonics.
8. Kong, T., Sun, F., Tan, C., Liu, H. and Huang, W., 2018. Deep feature pyramid reconfiguration for object detection. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 169-185).
9. Chandan, G., Jain, A. and Jain, H., 2018, July. Real time object detection and tracking using Deep Learning and OpenCV. In *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)* (pp. 1305-1308). IEEE.
10. Cong, T., Yongshun, L., Kedong, Z., Xing, Y., Chao, Z., Hua, Y. and Wei, J., 2018. Object detection method of multi-view SSD based on deep learning. *红外与激光工程*, *47*(1), pp.126003-0126003.

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

Declaration of member’s contribution:

Khadiza Sarker Bobby **ID: 1711049042**	**Md Abdul Aziz** **ID: 1711757042**	**Tanvir Hossain Jishad** **ID: 1610179042**
Cover	Methodology	Methodology
Abstract	Experiment and Result (Object detection from webcam)	Background Study
Table of Contents	Table of Contents	Experiment and result (Object detect from image)
Introduction	Implement faster rcnn (3 part) code	Voice Generate
Background Study	Conclusion	Implement SSD mobile-net code
Methodology (Software specifications)	Appendix	
Distance Calculation		
Implement SSD inception v2 code		