

Towards Global Technological Excellence”

A Project Report on

**AI-POWERED OFFLINE OBJECT
DETECTION DEVICE FOR VISUALLY
IMPAIRED USERS**

Submitted to

Government College of Engineering, Amravati

for the partial fulfilment of requirements of degree of Bachelor of Technology in

Electronics and Telecommunication Engineering

By

Mr. Nilesh Vikas Mahajan (21004043)

Mr. Ahesan Zafir Zuberulislam (21004060)

Mr. Atharv Santosh Talokar (21004066)

Supervised By

A.M. Shah

Assistant Professor



Department of Electronics and Telecommunication Engineering

Government College of Engineering, Amravati

(An Autonomous Institute of Government of Maharashtra)

Maharashtra State, India.

2024-2025

Government College of Engineering, Amravati
Department of Electronics and Telecommunication Engineering
2024-2025

CERTIFICATE



This is to certify that the project report entitled as “**AI-Powered Offline Object Detection Device for Visually Impaired Users**”, is a work carried out Bonafide in the VIII semester **Mr. Nilesh Vikas Mahajan (21004043)**, **Mr. Atharv Santosh Talokar (21004066)**, **Mr. Ahesan Zafir Zuberulislam (21004060)** in partial fulfilment for the award of the Degree **Bachelor of Technology in Electronics and Telecommunication from Government College of Engineering, Amravati** affiliated to **Sant Gadge Baba Amravati University, Amravati** under my supervision and guidance, in the Academic Year 2024-25.

A. M. Shah

Project Guide
Assistant Professor
Department of Electronics and
Telecommunication Engineering
Government College of Engineering,
Amravati

Dr. P. R. Deshmukh

Head of Department
Department of Electronics and
Telecommunication Engineering
Government College of Engineering,
Amravati

Dr. A. M. Mahalle

Principal
Government College of Engineering,
Amravati

External Examiner

DECLARATION

We would like to express our deep and sincere gratitude and We hereby declare that the project report entitled, “AI-Powered Offline Object Detection Device for Visually Impaired Users” was carried out and written by us under the guidance of A. M. Shah, Department of Electronics and Telecommunication Engineering, Govt. College of Engineering, Amravati. This work has not been previously formed the basis for the award of any degree or diploma or certificate nor has been submitted elsewhere for the award of any degree or diploma.

Submitted By-

Mr. Nilesh Vikas Mahajan (21004043)

Mr. Ahesan Zafir Zuberulislam (21004060)

Mr. Atharv Santosh Talokar (21004066)

Date:

Place:

ACKNOWLEDGEMENT

We would like to express our deep and sincere gratitude to A. M. Shah for his critical comments, advice and guidance which has been very valuable for the completion of the Report. He showed different possible approaches for problem solving and guided to develop new desirable features in our work. We are also thankful to Dr. P.R. Deshmukh, Head of Department of Electronics and Telecommunication, for his cooperation and support to complete this Seminar work. We would also like to extend my thanks Dr. A. M. Mahalle, Principal, Government College of Engineering Amravati for providing all the facilities and continuous support. Finally, thanking to all those who directly or indirectly helped us during this work.

Mr. Nilesh Vikas Mahajan (21004043)

Mr. Ahesan Zafir Zuberulislam (21004060)

Mr. Atharv Santosh Talokar (21004066)

Date:

Place:

ABSTRACT

Object detection has emerged as a transformative technology with applications spanning diverse fields, from autonomous vehicles to healthcare. This project focuses on leveraging object detection to assist visually impaired individuals in navigating their surroundings more effectively. The proposed system integrates a Raspberry Pi with a Pi camera and headphones, utilizing TensorFlow's Object Detection API to identify objects in real-time. Once an object is detected, the system employs Text-to-Speech (TTS) technology to announce the object's name through the headphones, providing auditory feedback to the user. The model is trained using the Single Shot Multi box Detector (SSD) with Mobile Net V2 architecture, ensuring lightweight yet accurate detection suitable for edge devices like the Raspberry Pi. This project not only demonstrates the practical implementation of object detection but also highlights its potential to enhance accessibility for visually impaired individuals. By combining hardware, software, and machine learning, this device aims to empower blind users by providing them with real-time information about their environment.

List of Abbreviations

S.NO	ABBREVIATION	EXPANSION
1	CNN	Convolutional Neural Network
2	ANN	Artificial Neural Network
3	RELU	Rectified Linear Unit
4	SVM	Support Vector Machine
5	SSD	Single Shot MultiBox Detector
6	TF	TensorFlow
7	TFLite	TensorFlow Lite
8	RPI	Raspberry PI
9	VNC	Virtual Network Computing
10	CSV	Comma-Separated Values
11	XML	Extensible Markup Language
12	PIP	Pip Install Package
13	ML	Machine Learning
14	GPU	Graphic Processing Unit
15	RDP	Remote Desktop
16	SSH	Secure Shell
17	CV	Computer Vision
18	SIFT	Scale-Invariant Feature Transform
19	YOLO	You Look Only Once
20	CAP	Credit Assignment Path
21	DNN	Deep Neural Network
22	TL	Transfer Learning
23	ARM	Advanced RISC Machine
24	POE	Power Over Ethernet
25	API	Application Program Interface
26	UVC	USB Video Device Class
27	PIL	Python Imaging Library
28	FOSS	Free and Open Source Software
29	RCNN	Region Convolutional Neural Network
30	CKPT	Check Point Process
31	TOCO	TensorFlow Lite Optimizing Converter
32	AI	Artificial Intelligence

Contents

1	INTRODUCTION	1
1.1	OBJECT DETECTION:	1
1.1.1	METHODS OF OBJECT DETECTION:	1
1.1.2	ARTIFICIAL INTELLIGENCE VS MACHINE LEARNING VS DEEP LEARNING	2
1.2	PROBLEM STATEMENT	6
1.3	OBJECTIVES OF THE PROJECT	6
1.4	SIGNIFICANCE OF THE PROJECT	7
1.5	OVERVIEW OF THE SYSTEM	7
2	LITERATURE SURVEY	9
2.1	LITERATURE SURVEY	9
2.2	PROPOSED MODEL	10
2.3	OVERVIEW OF PROJECT	11
2.4	ADVANTAGES OF THE PROPOSED SYSTEM	11
2.5	LIMITATIONS AND FUTURE SCOPE	11
3	HARDWARE COMPONENTS	13
3.1	GENERAL DESCRIPTION:	13
3.1.1	PI CAMERA MODULE	13
3.1.2	HARDWARE	13
3.1.3	ADDITIONAL HARDWARE SPECIFICATIONS	14
3.2	PIN DIAGRAM OF RASPBERRY PI 3B+	15
3.3	WEBCAM	17
3.3.1	INTERFACE	17
3.3.2	USES	18
3.4	RASPBERRY PI CAMERA	18
3.4.1	SPECIFICATIONS	19
3.4.2	APPLICATIONS	19
4	SOFTWARE REQUIREMENT	20
4.1	GENERAL	20
4.1.1	VNC VIEWER	20
4.1.2	PuTTY	21
4.1.3	OPENCV	21
4.1.4	OPENCV PYTHON	22
4.1.5	TENSORFLOW LITE	22
4.1.6	RASPBIAN OS	22
4.2	INSTALLATION OF RASPBIAN OPERATING SYSTEM ON RASP- BERRY Pi	23
4.3	CAMERA MODULE CONFIGURATION IN RASPBERRY PI	27
4.3.1	TO CAPTURE AIMAGE WITH RASPBERRY PI CAM- ERA MODULE	29
4.3.2	TO RECORD A VIDEO WITH RASPBERRY PI CAM- ERA MODULE	29

5	TESTING AND TRAINING	30
5.1	TENSORFLOW	30
5.1.1	HISTORY OF TENSOR FLOW	30
5.1.2	TENSORFLOW ARCHITECTURE	30
5.1.3	RUNNING TENSORFLOW	30
5.1.4	COMPONENTS OF TENSORFLOW	31
5.2	TRAINING TENSORFLOW OBJECT DETECTION CLASSIFIER	31
5.2.1	ANACONDA	31
5.2.2	SETTING UP OBJECT DETECTION DIRECTORY STRUC- TURE AND ANACONDA VIRTUAL ENVIRONMENT: . .	32
5.2.3	GATHERING AND LABELLING IMAGES	34
5.2.4	GENERATING TRAINING DATA	36
5.2.5	CREATE LABEL MAP AND CONFIGURE TRAINING . .	38
5.2.6	RUN THE TRAINING	39
5.2.7	TENSORFLOW OBJECT DETECTION API ON THE RASP- BERRY PI	40
6	FUTURE SCOPE AND CONCLUSION	45

List of Figures

1.1	AI vs ML vs DL	2
1.2	Basic Architecture of a Neural Network	4
1.3	Architecture of Deep Neural Networks	5
1.4	Architecture of Convolutional Neural Network	6
3.1	Raspberry Pi 3 B+ and Pi Camera Module	14
3.2	Pin diagram of Raspberry Pi 3 B+ Model	15
3.3	Blocks of Raspberry Pi 3 model	17
3.4	Raspberry pi camera	18
4.1	Downloading Raspbian Stretch	24
4.2	Using Etcher for flashing	25
4.3	Opening Raspberry Pi Configuration menu	26
4.4	Enabling System in Raspberry Pi Configuration	26
4.5	Enabling SSH and VNC in Raspberry Pi Configuration	27
4.6	Placing Cable in the Raspberry Pi camera port	28
4.7	Selecting Interfacing Option in Configuration Tool	28
5.1	Download Python 3.6 version of Anaconda	32
5.2	Installing Anaconda 3.5.0.1 version	32
5.3	Gathering images for annotation	35
5.4	Labeling images using Labellmg	35
5.5	XML content generated for animage	36
5.6	Generated csv file	36
5.7	Creating label map	38
5.8	Total loss graph	40
5.9	Enabling Camera Module in Raspberry Pi configuration menu	43
5.10	Final Output of testing	44

List of Tables

3.1	Raspberry Pi hardware specifications	13
3.2	Pin Specification of Raspberry Pi 3B+ Model	16
4.1	Stretch Version and its compatibility of different Raspberry Pi's	23
5.1	Versions specified Libraries	34
5.2	Data content inside a CSV file	37

1 INTRODUCTION

Computer Vision (CV) is the science of understanding and manipulating digital videos and images. Computer Vision plays a vital role in many applications, which includes Face recognition, image retrieval, industrial inspection, and augmented reality etc. With the emergence of deep learning, computer vision has proven to be useful for various applications. Deep Learning is an Artificial Neural Network (ANN) collection of methods, which is a machine learning branch. On the human brain, ANNs are modelled where nodes are connected to each other that pass data to each other. The use of deep learning for computer vision can be categorized into various categories: generation, segmentation, detection and classification of both videos and images. Image classification labels the image as a whole Finding the position of the object in addition to labelling the object is called object localization. Typically, the position of the object is defined by rectangular coordinates. Finding multiple objects in the image with rectangular coordinates is called detection. Segmentation is detecting exact objects like creating a transparent mask above the object with exact edges. An image classification or model of image recognition merely detects an object's likelihood in an image. In comparison the location of objects relates to the place of an item in the picture. A localization algorithm for object will output the place coordinates of an item with regard to the picture or image. Object detection is a problem of importance in CV. Similar to image classification tasks, deeper networks have shown better performance in detection. At present, the accuracy of these techniques is excellent. Hence it used in many applications. The difference is the number of objects. In detection, there are a variable number of objects. This small difference makes a big difference when designing the architectures for the deep learning model concerning localization or detection.

Let's see in detail about the terminologies involved in our project.

1.1 OBJECT DETECTION:

In today's fast-paced world, technological advancements have significantly improved the quality of life for many individuals. However, people with visual impairments often face challenges in performing daily tasks due to their limited ability to perceive their surroundings. Traditional mobility aids, such as canes and guide dogs, are helpful but lack the ability to provide detailed information about the environment. For instance, while a cane can detect obstacles, it cannot identify whether an object is a chair, table, or staircase. Similarly, guide dogs are trained to navigate but cannot communicate the names of objects or hazards.

To address these limitations, there is a growing need for assistive technologies that can provide real-time, context-aware information to visually impaired individuals. Advances in computer vision, particularly object detection, offer promising solutions. Object detection algorithms can recognize and classify objects in images or video streams, enabling systems to "see" and interpret the environment. By integrating these algorithms with accessible hardware, we can create devices that bridge the gap between visual perception and auditory feedback.

1.1.1 METHODS OF OBJECT DETECTION:

Methods for object detection generally fall into either machine learning-based approaches or deep learning-based approaches. For Machine Learning approaches,

it becomes necessary to first define features using one of the methods below, then using a technique such as support vector machine (SVM) to do the classification. On the other hand, deep learning techniques are able to do end-to-end object detection without specifically defining features, and are typically based on convolutional neural networks (CNN).

Machine learning approaches:

- Viola–Jones object detection framework based on Haar features
- Scale-invariant feature transform (SIFT)
- Histogram of oriented gradients (HOG) features

Deep learning approaches:

- Region Proposals (R-CNN, Fast R-CNN, Faster R-CNN)
- Single-Shot Refinement Neural Network for Object Detection (RefineDet)
- Retina-Net
- Deformable convolutional networks
- Single Shot MultiBox Detector (SSD)
- You Only Look Once (YOLO)

For this project we have used a deep learning-based technique based on Convolutional Neural Networks (CNN). The approach used is Single Shot MultiBox Detector (SSD). The model used is: SSD MobileNet v2 coco model.

1.1.2 ARTIFICIAL INTELLIGENCE VS MACHINE LEARNING VS DEEP LEARNING

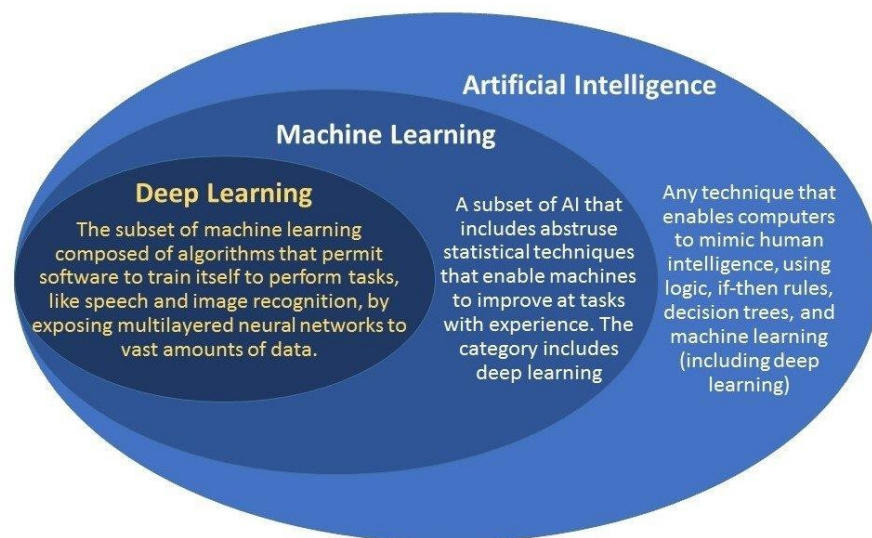


Figure 1.1: AI vs ML vs DL

DEEP LEARNING is a technical term. It refers to the number of layers in a neural network. A shallow network has one so-called hidden layer, and a deep

network has more than one. Multiple hidden layers allow deep neural networks to learn features of the data in a so-called feature hierarchy, because simple features (e.g. two pixels) recombine from one layer to the next, to form more complex features (e.g. a line). Nets with many layers pass input data (features) through more mathematical operations than nets with few layers, and are therefore more computationally intensive to train. Computational intensity is one of the hallmarks of deep learning, and it is one reason why GPUs are in demand to train deep-learning models

WORKING OF DEEP LEARNING MODEL:

In deep learning, each level learns to transform its input data into a slightly more abstract and composite representation. In an image recognition application, the raw input may be a matrix of pixels; the first representational layer may abstract the pixels and encode edges; the second layer may compose and encode arrangements of edges; the third layer may encode a nose and eyes; and the fourth layer may recognize that the image contains a face.

Importantly, a deep learning process can learn which features to optimally place in which level on its own. The "deep" in "deep learning" refers to the number of layers through which the data is transformed. More precisely, deep learning systems have a substantial credit assignment path (CAP) depth. The CAP is the chain of transformations from input to output. CAPs describe potentially causal connections between input and output. For a feedforward neural network, the depth of the CAPs is that of the network and is the number of hidden layers plus one (as the output layer is also parameterized).

For recurrent neural networks, in which a signal may propagate through a layer more than once, the CAP depth is potentially unlimited. No universally agreed upon threshold

of depth divides shallow learning from deep learning, but most researchers agree that deep learning involves CAP depth > 2 . CAP of depth 2 has been shown to be a universal approximator in the sense that it can emulate any function.] Beyond that more layers do not add to the function approximator ability of the network. Deep models (CAP > 2) are able to extract better features than shallow models and hence, extra layers help in learning features.

NEURAL NETWORK is a network or circuit of neurons, or in a modern sense, an artificial neural network, composed of artificial neurons or nodes.[1] Thus a neural network is either a biological neural network, made up of real biological neurons, or an artificial neural network, for solving AI problems.

The connections of the biological neuron are modeled as weights. A positive weight reflects an excitatory connection, while negative values mean inhibitory connections. All inputs are modified by a weight and summed.

This activity is referred to as a linear combination. Finally, an activation function controls the amplitude of the output. For example, an acceptable range of output is usually between 0 and 1, or it could be -1 and 1.

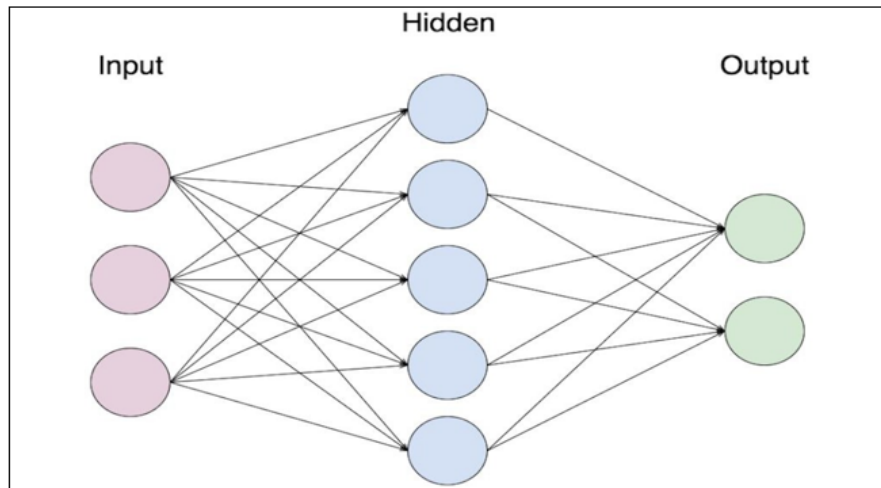


Figure 1.2: **Basic Architecture of a Neural Network**

TYPES OF NEURAL NETWORK:

- Artificial Neural networks
- Deep Neural networks
- Convolutional Neural networks

ARTIFICIAL NEURAL NETWORKS:

An Artificial Neural Network ANN is based on a collection of connected units called artificial neurons, (analogous to biological neurons in a biological brain). Each connection (synapse) between neurons can transmit a signal to another neuron. The receiving (postsynaptic) neuron can process the signal(s) and then signal downstream neurons connected to it. Neurons may have state, generally represented by real numbers, typically between 0 and 1. Neurons and synapses may also have a weight that varies as learning proceeds, which can increase or decrease the strength of the signal that it sends downstream. Typically, neurons are organized in layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first(input), to the last (output) layer, possibly after traversing the layers multiple times.

The original goal of the neural network approach was to solve problems in the same way that a human brain would. Over time, attention focused on matching specific mental abilities, leading to deviations from biology such as backpropagation, or passing information in the reverse direction and adjusting the network to reflect that information. Neural networks have been used on a variety of tasks, including computer vision, speech recognition, machine translation, social network filtering, playing board and video games and medical diagnosis.

As of now neural networks typically have a few thousand to a few million units and millions of connections. Despite this number being several orders of magnitude less than the number of neurons on a human brain, these networks can perform many tasks at a level beyond that of humans.

DEEP NEURAL NETWORK:

A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers. The DNN finds the correct mathematical manipulation to turn the input into the output, whether it be a linear relationship or a non-linear relationship. The network moves through the layers calculating the probability of each output. For example, a DNN that is trained to recognize dog breeds will go over the given image and calculates the probability that the dog in the image is a certain breed. The user can review the results and select which probabilities the network should display (above a certain threshold, etc.) and return the proposed label. Each mathematical manipulation as such is considered a layer, and complex DNN have many layers, hence the name "deep" networks. The goal is that eventually, the network will be trained to decompose an image into features, identify trends that exist across all samples and classify new images by their similarities without requiring human input. DNNs can model complex non-linear relationships. DNNs are typically feedforward networks in which data flows from the input layer to the output layer without looping back.

At first, the DNN creates a map of virtual neurons and assigns random numerical values, or "weights", to connections between them. The weights and inputs are multiplied and return an output between 0 and 1. If the network didn't accurately recognize a particular pattern, an algorithm would adjust the weights. That way the algorithm can make certain parameters more influential, until it determines the correct mathematical manipulation to fully process the data.

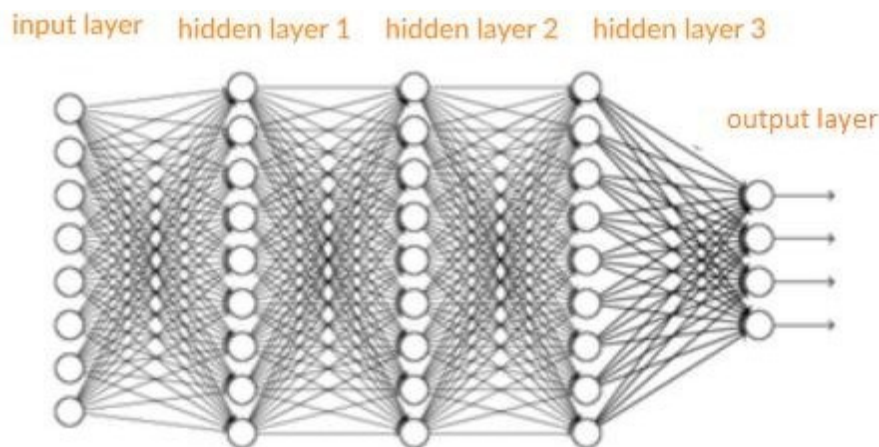


Figure 1.3: Architecture of Deep Neural Networks

CONVOLUTIONAL NEURAL NETWORKS:

A convolutional network receives a normal color image as a rectangular box whose width and height are measured by the number of pixels along those dimensions, and whose depth is three layers deep, one for each letter in RGB. Those depth layers are referred to as channels.

As images move through a convolutional network, we will describe them in terms of input and output volumes, expressing them mathematically as matrices of multiple dimensions in this form: $30 \times 30 \times 3$. From layer to layer, their dimensions change for reasons that will be explained below. Now, for each pixel of an image, the intensity of R, G and B will be expressed by a number, and that number will be an element in one of the three, stacked two-dimensional matrices, which together

form the image volume. Those numbers are the initial, raw, sensory features being fed into the convolutional network, and the ConvNets purpose is to find which of those numbers are significant signals that actually help it classify images more accurately. (Just like other feed forward networks we have discussed.)

Rather than focus on one pixel at a time, a convolutional net takes in square patches of pixels and passes them through a filter. That filter is also a square matrix smaller than the image itself, and equal in size to the patch. It is also called a kernel, which will ring a bell for those familiar with support-vector machines, and the job of the filter is to find patterns in the pixels.

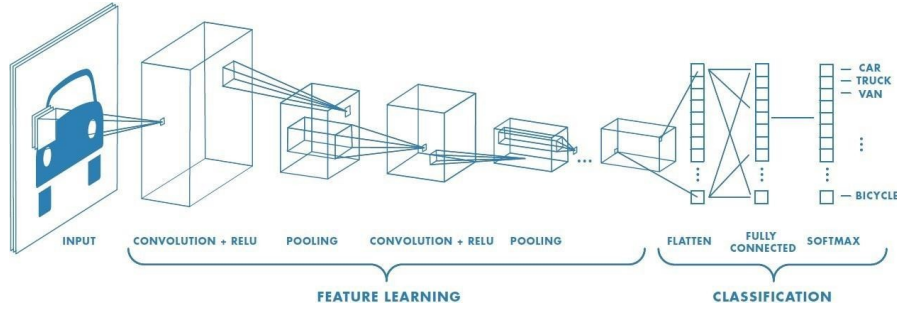


Figure 1.4: **Architecture of Convolutional Neural Network**

1.2 PROBLEM STATEMENT

Visually impaired individuals often rely on external assistance or rudimentary tools to navigate their surroundings. While existing technologies like GPS and voice assistants are helpful, they lack the granularity required to identify nearby objects in real-time. For example, a blind person walking down a street may encounter obstacles such as bicycles, trash bins, or low-hanging branches, which traditional mobility aids cannot identify. Additionally, indoor navigation poses challenges, as objects like chairs, tables, or kitchen items are difficult to locate without tactile feedback.

This project aims to address these challenges by developing a portable, affordable, and efficient assistive device. The system uses a Raspberry Pi coupled with a Pi camera to capture live video feeds, processes the images using TensorFlow's Object Detection API, and announces the names of detected objects through headphones using Text-to-Speech (TTS) technology. By providing real-time auditory feedback, the device enables visually impaired users to gain a better understanding of their environment, enhancing their independence and safety.

1.3 OBJECTIVES OF THE PROJECT

The primary objectives of this project are as follows:

1. **Real-Time Object Detection** : Develop a system capable of detecting and classifying objects in real-time using TensorFlow's Object Detection API.
2. **Auditory Feedback** : Integrate Text-to-Speech (TTS) functionality to announce the names of detected objects, providing auditory feedback to the user.
3. **Portability and Accessibility** : Design a compact and lightweight device using a Raspberry Pi and Pi camera, ensuring it is easy to carry and use.

4. Customization for Specific Use Cases : Train the object detection model to recognize everyday items commonly encountered by visually impaired individuals, such as furniture, household objects, and street obstacles.
5. Performance Optimization : Optimize the system for edge devices to ensure low latency and high accuracy, even in resource-constrained environments.

1.4 SIGNIFICANCE OF THE PROJECT

This project holds significant implications for improving the quality of life for visually impaired individuals. By providing real-time information about their surroundings, the device empowers users to navigate independently and confidently. Unlike traditional mobility aids, this system offers contextual awareness, enabling users to identify and interact with objects in their environment. Moreover, the use of open-source tools like TensorFlow and Raspberry Pi ensures that the solution is cost-effective and accessible to a wide audience.

The integration of machine learning with assistive technologies represents a paradigm shift in how we approach accessibility. By combining hardware, software, and artificial intelligence, this project demonstrates the potential of technology to create inclusive solutions that cater to the needs of marginalized communities. Furthermore, the modular design of the system allows for future enhancements, such as recognizing text, identifying faces, or detecting specific gestures, making it a versatile platform for further research and development.

1.5 OVERVIEW OF THE SYSTEM

The proposed system consists of the following components:

1. Hardware : A Raspberry Pi serves as the central processing unit, interfaced with a Pi camera for capturing video input and headphones for delivering auditory output.
2. Software : TensorFlow's Object Detection API is used to train and deploy the object detection model. The Single Shot Multibox Detector (SSD) with MobileNet V2 architecture is chosen for its balance of speed and accuracy, making it ideal for edge devices.
3. Text-to-Speech (TTS) : Detected objects are announced using TTS libraries, converting text labels into spoken words that are relayed to the user via headphones.
4. Training Data : The model is trained on a custom dataset comprising images of everyday objects such as cups, jugs, flasks, chairs, tables, and other items relevant to visually impaired users.

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. Well-researched domains of object detection include face detection and pedestrian detection. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance.

It is widely used in computer vision tasks such as image annotation, activity recognition, face detection, face recognition, video object co-segmentation. It is

also used in tracking objects, for example tracking a ball during a football match, tracking movement of a cricket bat, or tracking a person in a video.

Every object class has its own special features that helps in classifying the class – for example all circles are round. Object class detection uses these special features. For example, when looking for circles, objects that are at a particular distance from a point (i.e. the center) are sought. Similarly, when looking for squares, objects that are perpendicular at corners and have equal side lengths are needed. A similar approach is used for face identification where eyes, nose, and lips can be found and features like skin color and distance between eyes can be found.

2 LITERATURE SURVEY

2.1 LITERATURE SURVEY

1. Gayathri et al (2019) has proposed a faster model of object detection namely Single Shot Multi-Box Detection (SSD) along with Mobile Net, published in second International Conference on Computational Intelligence in Data Science (ICCIDS-2019) It is shown that for real time applications where inference needs to be quick, this model does the work elegantly. The paper compares other deep learning models like RCNN, Fast RCNN, Faster RCNN, but owing to their complex networked structure, the accuracy is lesser when detecting multiple objects in a single frame. As SSD uses convolution filters for detecting the objects depth it loses its accuracy if the frame is of low resolution. So we use the Mobile Net technique as it use depth wise separable convolution which significantly reduces the parameters size when compare with normal convolution filters of same depth. Thus, we can get the light weight deep neural network as a result. On whole SSD with MobileNet is nothing but a model in which meta architecture is SSD and the feature extraction type is MobileNet
2. Abdelhakim Zeroual (2019) et al published in the Proceedings of the International Conference on Networking and Advanced Systems (ICNAS) IEEE and proposed that there are many deep neural network frameworks for embedded platforms. Caffe2 is a framework, which cares multiple embedded platforms such as, Android, iOS, Tegra, and Raspbian. TensorFlow Lite is another is a lightweight and accessible framework and a new trend, which supports a variety of embedded plat
3. We can implement and run deep neural network using Python and C++. In the domain of security on Mobile Cloud Environment (MCC), for facial recognition based on Deep convolutional neural network to authenticate the device, the training and recognition phases were done on cloud because of huge computation. Based on this work, we propose to use the TensorFlow lite for mobile device to reduce the time of transfer between cloud and mobile. By focusing on the cited work, the authors have achieved an accuracy of 100% compared to the last one with 99.50%. To reduce the transfer time, they have suggested using a TensorFlow lite framework destined for mobile to do a recognition without a need of cloud, contrary to the previous work.
4. Tiagrajah V (2019) et al published in the Proceedings of the 9th International Conference on System Engineering and Technology (ICSET) IEEE have used Convolutional Neural Networks for object detection. They have presented a deep learning approach for robust detection of traffic light by comparing two object detection models and by evaluating the flexibility of the TensorFlow Object Detection Framework to solve the real-time problems. The models are Single Shot Multibox Detector (SSD) MobileNet V2 and Faster-RCNN. They showed that Faster-RCNN delivers 97.015%, which outperforms SSD by 38.806% for a model which had been trained using 441 images. They concluded that SSD shows more false negatives in object detection, but when used in mobile computing devices like Raspberry Pi, Android Devices, the inference time is faster in SSD compared to the Faster RCNN model.

5. Sumeet Sanjay Walam (2018) et al published in the Proceedings of the 2nd International Conference on Inventive Communication and Computational Technologies (ICICCT 2018) IEEE, proposes to detect and separate objects from a set based on their color. Proposed method of categorization is done on the basis of color of the object using a raspberry Pi 3. System categorizes the object into three different colors. The detection of the particular color is done by a light intensity camera to frequency convertor method. This was the motivation for our mini project; we tried implementing the same on an Arduino board and finally ended with an artificial intelligence project. We used SSD mobileNet model to detect objects.

2.2 PROPOSED MODEL

Based on the papers, we came to the conclusion of using Raspberry Pi 3B + for object detection by using the SSD-mobilenet V2 model along with TensorFlow lite as the framework. It was decided on the basis of the following reasons.

1. Object detection without deep learning algorithms, would mean thousands of lines of code which would be a cumbersome process. With lot of research going under Object Detection and lots of models becoming open-source, there will always be newer models in the future having better accuracy, lesser time for detection. But that is technological advancements, which needs to be dealt in a positive manner.
2. A micro-controller like device with onsite computing capabilities will reduce the cost and time of the process by reducing the server latency time and also the need for employing servers for the same..
3. An SSD-mobilenet based model gives better accuracy and lesser inference time compared to the other models. Based on the trade-off between time and accuracy, it is safer to select this model.

2.3 OVERVIEW OF PROJECT

Step1:

The first step involves capturing the images for training, we have collected 150 images of cup, flask, jug as our dataset, which were later split into test images and train images

Step2:

After collecting the dataset we have to label the images further in order to teach the machine, this is performed by using a software tool called Lableimg.

Step3:

After labeling, particular algorithm should be chosen for the training purpose, we have chosen SSD MOBILE NET MODEL v2, which is made to run with 2000 steps in i5 processor machine.

Step4:

After successful training the test images were given so as to understand the model's accuracy and speed, in our case the chosen model has provided us with an accuracy of 96% and has detected all the three labelled datasets.

Step5:

After successful test results, now the real time object detection should be carried out. Through the webcam in our machine we were able to detect the images with greater accuracy.

Step6:

The final step is to convert the model to TFLITE model so that it works efficiently on our low processing device Raspberry Pi, the end results were with same accuracy as obtained in i5 machine, but the frames per second is slightly less i.e 0.9 frames per second.

2.4 ADVANTAGES OF THE PROPOSED SYSTEM

- **Lightweight and Fast:** The use of SSD-MobileNet V2 with TensorFlow Lite enables faster detection even on low-power devices.
- **Cost-Efficient:** Raspberry Pi 3B+ significantly cuts down hardware costs without compromising performance.
- **Offline Capability:** The system does not require constant internet access, making it ideal for remote or critical environments.
- **Ease of Deployment:** The simplicity of the model architecture and open-source tools ensures ease of installation and updates.
- **Scalability:** The model can be retrained with additional objects without major infrastructural changes.

2.5 LIMITATIONS AND FUTURE SCOPE

- **Low Frame Rate:** On Raspberry Pi, the system processes fewer frames per second, which might be insufficient for high-speed applications.
- **Limited Dataset:** A larger, more diverse dataset could improve detection performance under varying conditions like different lighting or object orientations.

- Hardware Constraints: Upgrading to Raspberry Pi 4B or integrating Google Coral TPU could significantly boost performance.
- Model Optimization: Future enhancements could involve model quantization or pruning to reduce memory usage and increase inference speed.

3 HARDWARE COMPONENTS

3.1 GENERAL DESCRIPTION:

Raspberry Pi3 (Model B+): Raspberry Pi is a low-cost, basic computer that was originally intended to help spur interest in computing among school-aged children. All the hardware components of our project are mostly connected through the Raspberry Pi. The Raspberry Pi is contained on a single circuit board and features ports for:

- HDMI
- USB 2.0
- Composite video
- Analog audio
- Power
- Internet
- SD Card

The computer runs entirely on open-source software and gives students the ability to mix and match software according to the work they wish to do.

3.1.1 PI CAMERA MODULE

The Pi camera module v2 replaced the original camera module. The Raspberry pi camera will act as like the inbuilt camera as like in laptops. It helps in capturing pictures, recording videos as per our wish and save those images and videos in the specified path.

3.1.2 HARDWARE

Taking into account the relatively high-performance requirements of image processing in general and the equipment currently available to the faculty, as a relatively inexpensive and powerful embedded platform the Raspberry Pi was an obvious choice. The hardware specifications taken into consideration for this work can be seen in Table 1.

Specifications	Raspberry Pi 3 B+
CPU type/speed	ARM Cortex-A53 1.4GHz
RAM size	1GB SRAM
Integrated Wi-Fi	2.4GHz and 5GHz
Ethernet speed	300Mbps
PoE	Yes
Bluetooth	4.2

Table 3.1: Raspberry Pi hardware specifications

Another contributing factor to this choice was the availability of the Pi camera module, which can capture high-definition video as well as stills and requires the user to simply update Raspberry's firmware to the latest version. It can easily be used in OpenCV based applications. Although using the officially supported camera module, which can be accessed through the MMAL and V4L APIs and is supported by numerous third-party libraries, any USB web camera can be used.



Figure 3.1: **Raspberry Pi 3 B+ and Pi Camera Module**

Raspberry Pi 3 Model B is single board computer. Its CPU speed ranges between 700MHz and 1.2GHz. It also has on board memory between 256MB and 1GB Ram. This is used at transmitter or user end. It is the heart of the system. OS installed on it is Raspbian. The Raspberry Pi 3 Model B is the latest version of the Raspberry Pi computer. The Pi isn't like your typical machine, in its cheapest form it doesn't have a case, and is simply a credit-card sized electronic board of the type you might find inside a PC or laptop but much smaller. The Raspberry Pi 3 Model B+ is the latest product in the Raspberry Pi 3 range, boasting an updated 64-bit quad core processor running at 1.4GHz with built-in metal heatsink, dual-band 2.4GHz and 5GHz wireless LAN, faster (300mbps) Ethernet, and PoE capability via a separate PoE HAT. The Raspberry Pi 3 Model B+ maintains the same mechanical footprint as both the Raspberry Pi 2 Model B and the Raspberry Pi 3 Model B+. It can still use all the favorite Raspbian or PIXEL software with this update. We should make sure to upgrade our Raspbian operating system installs to the latest version so that the firmware can support the new chips. Old SD cards from previous releases will not work without an upgrade.

3.1.3 ADDITIONAL HARDWARE SPECIFICATIONS

- Chipset: Broadcom BCM2837
- CPU: 1.2GHz quad-core 64-bit ARM cortex A53
- Ethernet: 10/100 (Max throughput 100Mbps)
- USB: Four USB 2.0 with 480Mbps data transfer
- Storage: MicroSD card or via USB-attached storage
- Wireless: 802.11n Wireless LAN (Peak transmit/receive throughput of 150Mbps), Bluetooth4.1
- Graphics: 400MHz Video Core IV multimedia Memory: 1GB LPDDR2-900 SDRAM

- Expandability: 40 general purpose input-output pins
- Video: Full HDMI port
- Audio: Combined 3.5mm audio out jack and composite video
- Camera interface (CSI)
- Display interface (DSI)

If Raspbian OS is not installed using the Noobs installer, and they may result in “running out of space”, they can also go into the terminal and type 'sudorasp-config' and then select the option to 'Expand root partition to fill SD card', which will ensure that current available space on the card can also be used.

3.2 PIN DIAGRAM OF RASPBERRY PI 3B+

RASPBERRY PI 3B+ is a development board in PI series. It can be considered as a single board computer that works on LINUX operating system. The board not only has tons of features it also has terrific processing speed making it suitable for advanced applications. PI board is specifically designed for hobbyist and engineers who are interested in LINUX systems and IOT (Internet of Things).

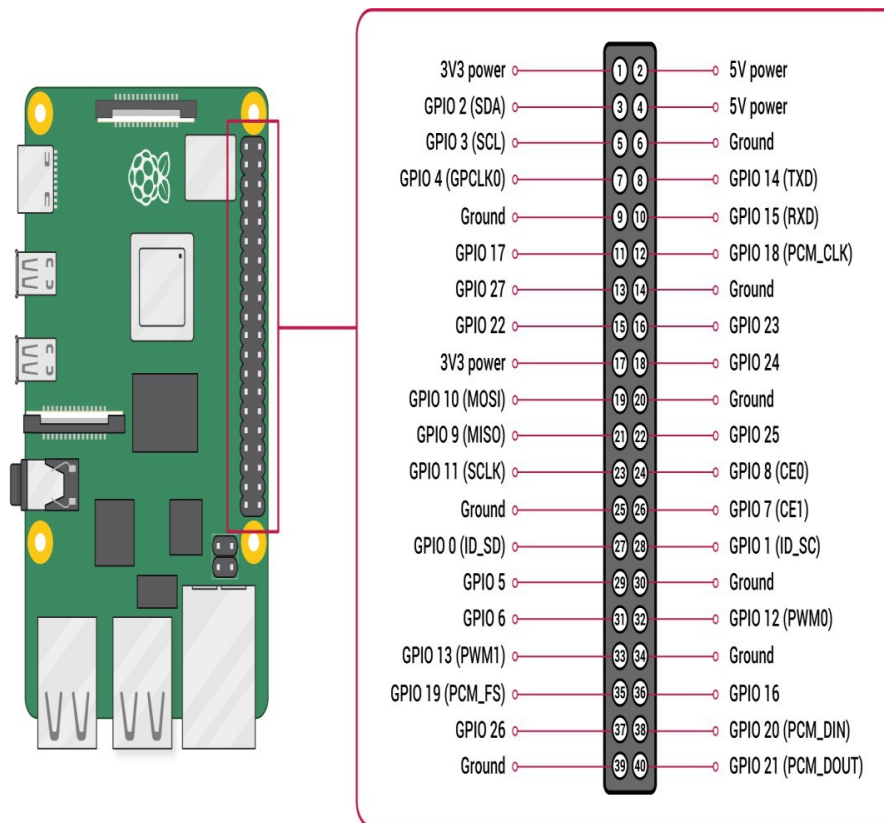


Figure 3.2: Pin diagram of Raspberry Pi 3 B+ Model

There are 40 Pins in Raspberry Pi 3 B+, where there are 7 Pin groups namely

- Power source
- Communication Interface

- SPI Interface
- TWI Interface
- Input output Pins
- PWM
- External Interrupts

PIN GROUP	PIN NAME	DESCRIPTION
POWER SOURCE	+5V, +3.3V, GND and Vin	+5V -power output +3.3V -power output GND – GROUND pin
COMMUNICATION INTERFACE	UART Interface (RXD, TXD) [(GPIO15, GPIO14)]	UART (Universal Asynchronous Receiver Transmitter) used for interfacing sensors and other devices.
SPI Interface (MOSI, MISO, CLK, CE) x2 [SPI0- (GPIO10,GPIO9, GPIO11,GPIO8)] [SPI1--(GPIO20, GPIO19, GPIO20, GPIO7)]	SPI (Serial Peripheral Interface) used for communicating with other boards or peripherals.	
TWI Interface (SDA, SCL) x2 [(GPIO2,GPIO3)] [(ID_SD, ID_SC)]	TWI (Two Wire Interface) Interface can be used to connect peripherals.	
INPUT OUTPUT PINS	26 I/O	Although these some pins have multiple functions they can be considered as I/O pins.
PWM	Hardware PWM available on GPIO12, GPIO13, GPIO18, GPIO19	These 4 channels can provide PWM (Pulse Width Modulation) outputs. Software PWM available on all pins
EXTERNAL INTERRUPTS	All I/O	In the board all I/O pins can be used as Interrupts.

Table 3.2: Pin Specification of Raspberry Pi 3B+ Model

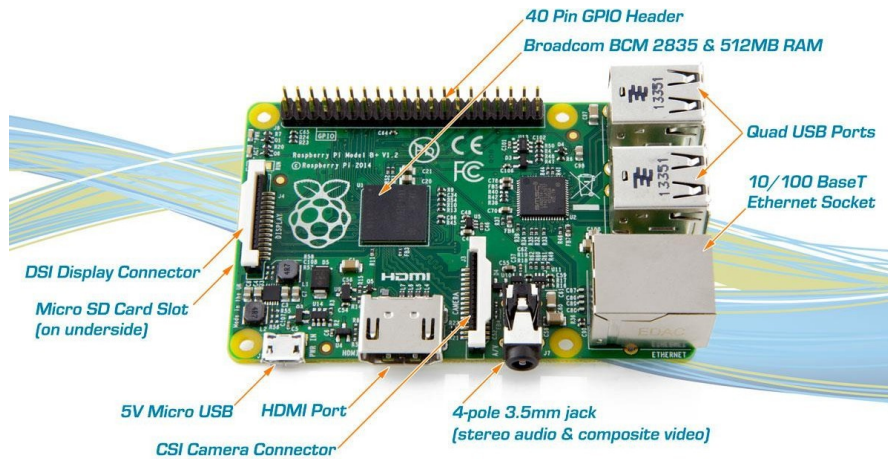


Figure 3.3: Blocks of Raspberry Pi 3 model

3.3 WEBCAM

A webcam is a video camera that feeds or streams an image or video in real time to or through a computer to a computer network, such as the Internet. Webcams are typically small cameras that sit on a desk, attach to a user's monitor, or are built into the hardware. Webcams can be used during a video chat session involving two or more people, with conversations that include live audio and video. For example, Apple's iSight camera, which is built into Apple laptops, iMacs and a number of iPhones, can be used for video chat sessions, using the iChat instant messaging program (now called Messages). Webcam software enables users to record a video or stream the video on the Internet.

As video streaming over the Internet requires a lot of bandwidth, such streams usually use compressed formats. The maximum resolution of a webcam is also lower than most handheld video cameras, as higher resolutions would be reduced during transmission. The lower resolution enables webcams to be relatively inexpensive compared to most video cameras, but the effect is adequate for video chat sessions. The term "webcam" (a clipped compound) may also be used in its original sense of a video camera connected to the Web continuously for an indefinite time, rather than for a particular session, generally supplying a view for anyone who visits its web page over the Internet. Some of them, for example, those used as online traffic cameras, are expensive, rugged professional video cameras.

3.3.1 INTERFACE

Typical interfaces used by articles marketed as a "webcam" are USB, Ethernet and IEEE 802.11 (denominated as IP camera). Further interfaces such as e.g. Composite video or S- Video are also available. The USB video device class (UVC) specification allows inter- connectivity of webcams to computers without the need for proprietary device drivers

CHARACTERISTICS

- Low manufacturing cost
- High flexibility, making them the lowest-cost form of videotelephony. As webcams evolved simultaneously with display technologies, USB interface

speeds and broadband internet speeds, the resolution went up from gradually 320×240 , to 640×480 , and some even offering 1280×720 (aka 720p) or 1920×1080 (aka 1080p) resolution.

- Low-end webcams offering resolutions of 720p,
- Mid-range webcams offering 1080p resolution, and
- High-end webcams offering 4K resolution at 60 fps.

3.3.2 USES

The most popular use of webcams is the establishment of video links, permitting computers to act as videophones or videoconference stations. Other popular uses include security surveillance, computer vision, video broadcasting, and for recording social videos. The video streams provided by webcams can be used for a number of purposes, each using appropriate software.

3.4 RASPBERRY PI CAMERA

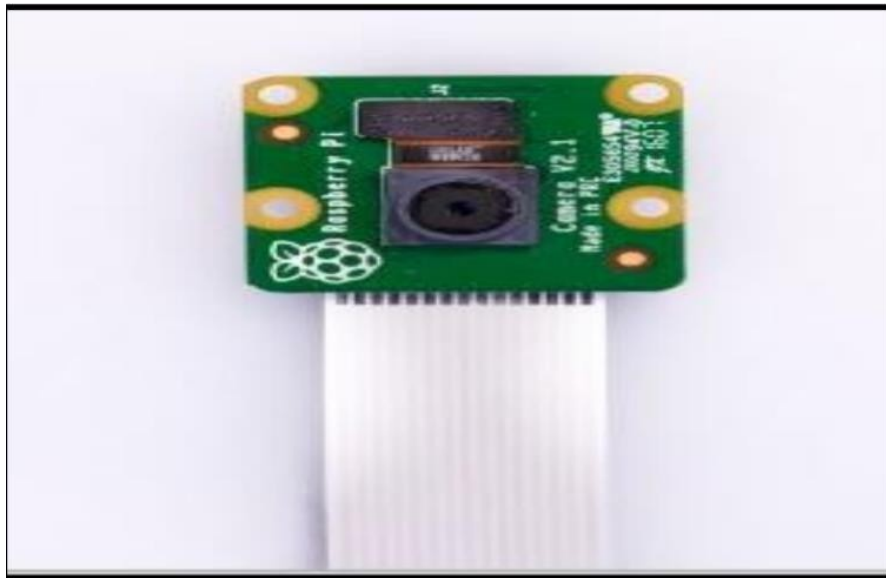


Figure 3.4: Raspberry pi camera

The Pi camera comes with a flex cable. The flex cable is inserted into the connector which is located between the Ethernet and HDMI port with the silver connectors facing the HDMI port. The flex cable connector is opened by pulling the tabs on the top of the connector upwards then towards the Ethernet port. The flex cable then is inserted firmly into the connector. The top part of the connector then is pushed towards the HDMI connector and down, while the flex cable is held in place.

Here the Pi camera is being utilized for the face detection and the face recognition process where firstly are captured and stored it in the database using python and then again using the camera while the automation and the surveillance part. The Raspberry Pi Camera Module v2 is a high quality 8 megapixel Sony IMX219 image sensor custom designed add-on board for Raspberry Pi, featuring a fixed focus lens. It's capable of 3280×2646 images static images, and also supports

1080p30, 720p60 and 640x480p60/90 video. It attaches to Pi by way of one of the small sockets on the board upper surface and uses the dedicated CSI interface, designed especially for interfacing to cameras.

3.4.1 SPECIFICATIONS

- mega pixel camera capable of taking photographs of 3280 x 2464 pixels
- Capture video at 1080p30, 720p60 and 640x480p90 resolutions
- All software is supported within the latest version of Raspbian Operating System

3.4.2 APPLICATIONS

- CTV security camera
- Motion detection
- Time lapse photography

4 SOFTWARE REQUIREMENT

4.1 GENERAL

The Raspberry Pi Foundation provides Raspbian, a Debian-based (32-bit) Linux distribution for download, as well as third-party Ubuntu, Windows 10 IoT Core, RISC OS. It promotes Python and Scratch as the main programming languages, with support for many other languages. The default firmware is closed source, while an unofficial open source is available. Many other operating systems can also run on the Raspberry Pi. Third-party operating systems available via the official website include Ubuntu MATE, Windows 10 IoT Core, RISC OS and specialized distributions for the Kodi media center and classroom management. The formally verified microkernel seL4 is also supported.

4.1.1 VNC VIEWER

A virtual network computing system is platform dependent. This means that the client working on one type of operating system can't connect to the VNC server that operates on a different type of operating system. There are many different types of clients and servers available for different GUI based interfaces. In addition, a virtual network computing system is available for Java. Some of the VNC programs only work for the Windows operating system. VNC was originally developed by an AT and T Research Team, but virtual network computing systems are extremely popular in handling business and personal uses.

VNC works by transmitting all of your keyboard and mouse movements from your thin client computer to the other, large client computer. VNC can allow you to use an older computer to run a recent program. This sometimes allows businesses to purchase fewer computers. Businesses no longer need the same type of disk memory or processing capability, for VNC makes things easier. In addition to saving data, the benefits include saving time and money.

4.1.1.1 USES OF VNC

People can use VNC to remotely access files on certain computers in a wide range of situations.

- VNC can be effectively used by people who need to access work files from their office computer. It is also great for people who happened to forget to bring their presentation to work. They can use VNC to connect to their home computer in order to retrieve any files.
- This type of program is often used by system administrators, IT support, and help desks. It is used by administrators in order to take control of an employee's computer. Administrators may need to do this in order to help certain employees with any type of program.
- It can also be used in the classroom by teachers. Teachers may take advantage of this program in order to allow students to view what is happening at the teacher's computer. This can help teachers teach their students easily and more effectively.

4.1.2 PuTTY

PuTTY is an open-source application making use of network protocols like including SCP, SSH Telnet and rlogin in Windows and UNIX platforms in conjunction with an X term terminal emulator.

Over a network, PuTTY makes use of all the above protocols to enable a remote session on a computer. It is a popular tool for text-based communication and is also a popular utility for connecting Linux servers from Microsoft operating system-based computers.

Putty (software) generally has two purposes:

- **Used as a File Transfer Protocol**

Most of the hosting services, both online and offline are built on LINUX OS, rightly so because it provides better safety for client data. Especially when thousands of client's data stored in a single place safety is the first priority. However, this poses a greater challenge for non-LINUX OS users to deal with. Here comes the third-party applications like PuTTY which enables non-Linux users install this particular software (PuTTY), and interact with Linux servers from a non-Linux OS. Interface of PuTTY is similar to windows terminal; however, user need to be aware of Linux commands to interact with its PuTTY provides various File transfer features like FTP and SFTP depending on user's security requirements.

- **Used to generate Hash key**

PuTTY also used to generate SSH key. Nowadays using passwords are prone to security threats especially when you are dealing with lot of confidential data online. PuTTY allows you to generate a series of keys, which is a combination of hundreds of Alpha Numeric and special characters, which is almost impossible to crack. SSH generates two types of key combination. Public key which is used to access the terminal by authorised people and Private key which should not be shared with anyone. Private key is encrypted into the particular server of the user which can only be opened with a public key.

When you are using a particular system to access a server you can just point to the public key and you can login to the server any time without entering any password or user name One place where particularly found the use of PuTTY is "Digital Ocean hosting services".

4.1.3 OPENCV

OpenCV was started at Intel in 1999 by Gary Bradsky, and the first release came out in 2000. Vadim Pisarevsky joined Gary Bradsky to manage Intel's Russian software OpenCV team. In 2005, OpenCV was used on Stanley, the vehicle that won the 2005 DARPA Grand Challenge. Later, its active development continued under the support of Willow Garage with Gary Bradsky and Vadim Pisarevsky leading the project. OpenCV now supports a multitude of algorithms related to Computer Vision and Machine Learning and is expanding day by day. OpenCV supports a wide variety of programming languages such as C++, Python, Java, etc., and is available on different platforms including Windows, Linux, OS X, Android, and iOS. Interfaces for high-speed GPU operations based on CUDA and OpenCL are also under active development. OpenCV- Python is the Python API

for OpenCV, combining the best qualities of the OpenCV C++ API and the Python language.

4.1.4 OPENCV PYTHON

OpenCV-Python is a library of Python bindings designed to solve computer vision problems. Python is a general-purpose programming language started by Guido van Rossum that became very popular very quickly, mainly because of its simplicity and code readability. It enables the programmer to express ideas in fewer lines of code without reducing readability. Compared to languages like C/C++, Python is slower. That said, Python can be easily extended with C/C++, which allows us to write computationally intensive code in C/C++ and create Python wrappers that can be used as Python modules.

This gives us two advantages: first, the code is as fast as the original C/C++ code (since it is the actual C++ code working in background) and second, it is easier to code in Python than C/C++. OpenCV-Python is a Python wrapper for the original OpenCV C++ implementation. OpenCV-Python makes use of NumPy, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from NumPy arrays. This also makes it easier to integrate with other libraries that use NumPy such as SciPy and Matplotlib.

4.1.5 TENSORFLOW LITE

When a ML model is implemented and trained using TensorFlow, you usually end up with a model file that requires ample storage space and a GPU to run inference. Luxuries such as large storage space and GPUs are not available on most mobile devices. TensorFlow lite is the solution to enabling ML models within mobile devices.

TensorFlow Lite models exist within the devices and have low latency, which is essential for real-time inference application, as network round trips from mobile devices to cloud servers where models are hosted might test the patience of your application users. TensorFlow Lite takes existing TensorFlow models and converts them into an optimized and efficient version in the form of a .tflite file. The streamlined model is small enough to be stored on devices and sufficiently accurate to conduct suitable inference.

TensorFlow Lite is used in:

- Mobile devices are prime devices to utilize the TensorFlow Lite model. TensorFlow Lite provides a variety of pre-trained models that can be easily converted to .tflite versions and integrated with mobile applications in their dedicated development environments.
- Internet of Things Devices
- Raspberry pi

4.1.6 RASPBIAN OS

Raspbian is a Debian operating system of Raspberry pi. There are several versions of Raspbian including Raspbian Buster and Raspbian Stretch. Since 2015 it

has been officially provided by Raspberry pi foundation, as the primary operating system for the family of Raspberry Pi as a single board computer. Raspbian was created by Mike Thompson and Peter Green as an independent project. The initial build was completed in June 2012. The operating system is still under active development. Raspbian is highly optimized for the Raspberry Pi line's low-performance ARM CPUs. Raspbian uses PIXEL, Pi Improved X-Window Environment, Lightweight as its main desktop environment as of the latest update. It is composed of a modified LXDE desktop environment and the Open Box stacking window manager with a new theme and few other changes. The distribution is shipped with a copy of the algebra program Wolfram and Mathematica and a version of Minecraft called Minecraft Pias well as a lightweight version of Chromium as of the latest version. It is composed of a modified LXDE desktop environment and the Open Box stacking window manager with a new theme and few other changes. The initial build was completed in June 2012. The operating system is still under active development.

2017-08-17	2017-08-16	9 (Stretch)	4.14	6.3	1.4.6		✓	✓	✓	✓	✗	✗
2017-09-08	2017-09-07						✓	✓	✓	✓	✗	✗
2017-11-29	2017-11-29						✓	✓	✓	✓	✗	✗
2018-03-13	2018-03-13						✓	✓	✓	✓	✗	✗
2018-04-18	2018-04-18				✓		✓	✓	✓	✓	✗	
2018-06-29	2018-06-27				✓		✓	✓	✓	✓	✗	
2018-10-09	2018-10-09				✓		✓	✓	✓	✓	✗	
2018-11-13	2018-11-13				✓		Yes	✓	✓	✓	✗	
2019-04-08	2019-04-08				10 (Buster)		4.19	8.3	1.4.9	✓	✓	✓
2019-06-24	2019-06-20	✓	✓	✓		✓			✓	✓		
2019-07-10	2019-07-10	✓	✓	✓		✓			✓	✓		
2019-09-30	2019-09-26	✓	✓	✓		✓			✓	✓		
2020-02-07	2020-02-05	✓	✓	✓		✓			✓	✓		
2020-02-14	2020-02-13	✓	✓	✓		✓			✓	✓		
Release Date	Release Name	Debian Version	Linux Kernel	GCC	apt	X Server	Pi 1/1+	Pi 2	Pi 3	Pi Zero W	Pi 3+	4

Table 4.1: **Stretch Version and its compatibility of different Raspberry Pi's**

The stretch version has been chosen according to the specifications of the raspberry pi 3 B+.

4.2 INSTALLATION OF RASPBIAN OPERATING SYSTEM ON RASPBERRY Pi

Installing Raspbian on the Raspberry Pi is really clear. Raspbian will be downloaded and writing the disc image to a micro SD card, at that point booting the Raspberry Pi to that microSD card. For this undertaking, one needs a microSD card (with no less than 8 GB), a PC with a space for it, and, obviously, a Raspberry Pi and fundamental peripherals (a mouse, console, screen, and power source). This isn't the main strategy for introducing Raspbian (more on that in a minute), yet it's a valuable method to learn on the grounds that it can likewise be utilized to introduce such a significant number of other working projects on the Raspberry Pi. When one realizes how to compose a circle picture to a microSD card, we open up a great deal of alternatives for Raspberry Piprojects.

Step1: Download the Raspbian

Turn on the PC and download the Raspbian disc image. One can locate the most recent variant of Raspbian on the Raspberry Pi Foundation's site here. It will take some time, particularly in the event when one intends to utilize the conventional download alternative as opposed to the other download sources. It can take a stretch of half hour or more to download. Choose Raspbian Stretch with Desktop if you want to have access to the Raspbian GUI; in other words, if you want to log in and be able to access a desktop, icons, etc. like you would with Windows or MacOS. Choose Raspbian Stretch Lite if you only need to boot to the command line. For simpler Raspberry Pi projects, this is often a good choice since the Lite version uses less power and fewer resources.

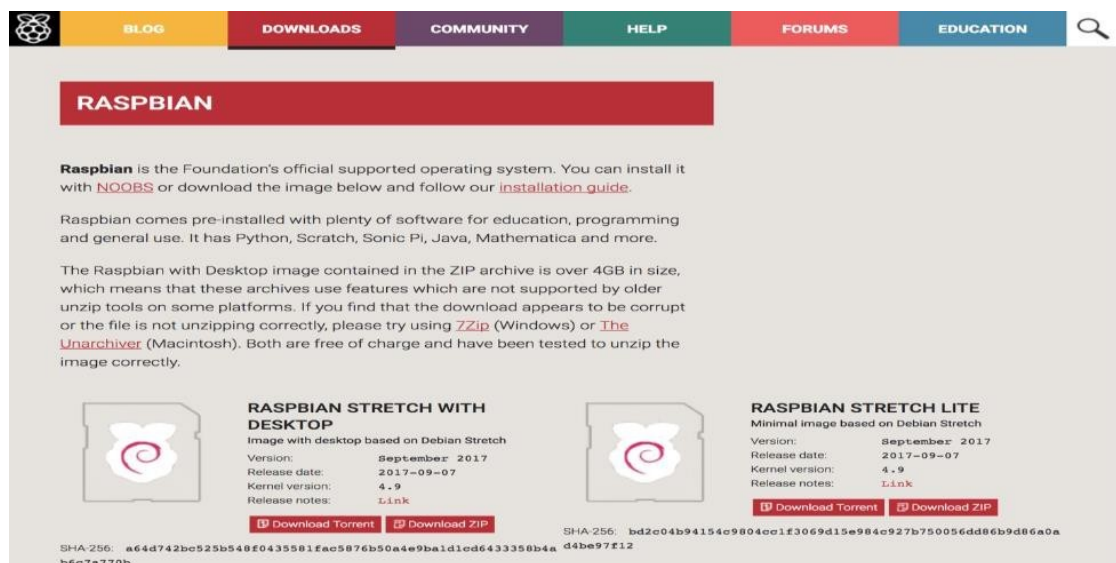


Figure 4.1: Downloading Raspbian Stretch

Step2: Unzip the file

The Raspbian disc image is compressed, so it should be unzipped. The file uses the ZIP64 format, so depending on how current built-in utilities are, one needs to use certain programs to unzip it. Linux users will use the appropriately named Unzip.

Step 3: Use Etcher

One has to pop the microSD card into our computer and write the disc image to it. The process of actually writing the image will be slightly different across these programs, but it's pretty self-explanatory no matter what is being used. Each of these programs will have us select the destination and the disc image (the unzipped Raspbian file). Choose, double-check, and then button to write.

The easiest way to flash Raspbian Stretch to your SD card is to download and install Etcher. After opening Etcher, select the Raspbian disk image, your SD card, and click Flash. After Etcher finishes running, the process ends.

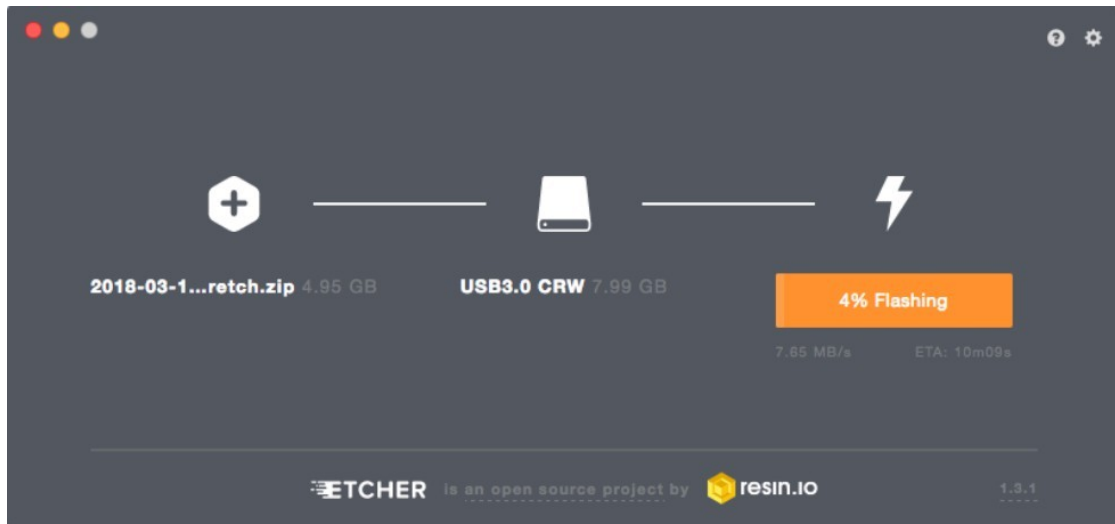


Figure 4.2: Using Etcher for flashing

Step 4: Put the microSD card in your Pi and boot up

When the disc image has been kept in touch with the microSD card, it is prepared to go. Put that microSD into your Raspberry Pi, plug in the peripherals and power source. The present release to Raspbian will boot straightforwardly to the desktop. Our default credentials are username pi and password raspberry.

Step 5:

Connect the Raspberry Pi to your network using a network cable, connect the keyboard and mouse and plug in the HDMI cable between the Pi and the Desktop.

Step 6:

After the Pi finishes booting up then the Raspbian desktop should appear. Open the Raspberry Menu, go to Preferences, then open Raspberry Pi Configuration.

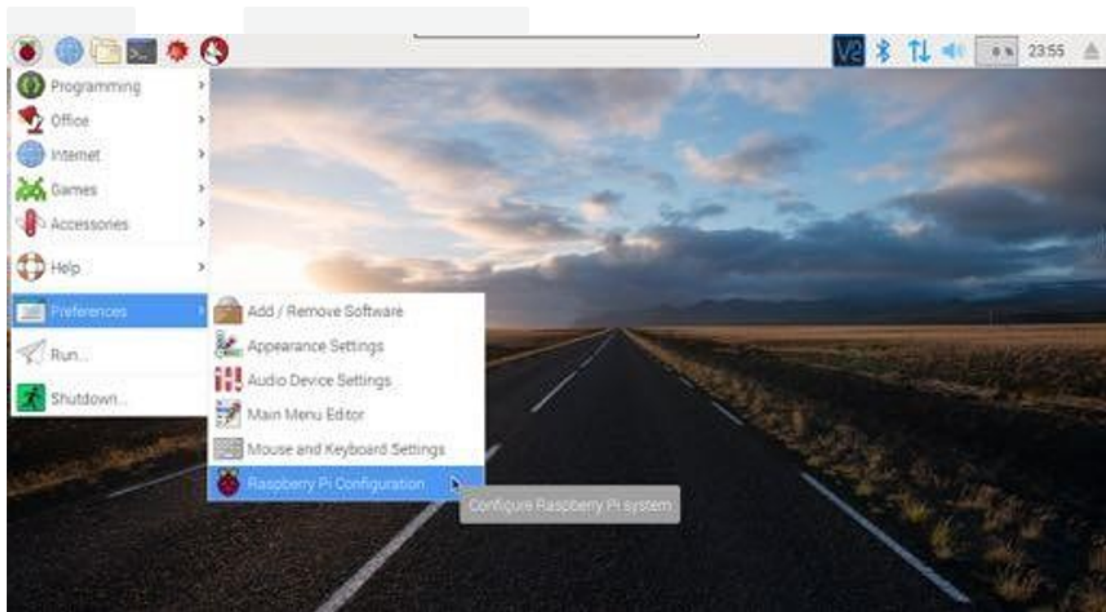


Figure 4.3: Opening Raspberry Pi Configuration menu

Step 7:
Check Wait for Network. This means the Pi will wait to start the desktop, and XLink, until we have a network connected

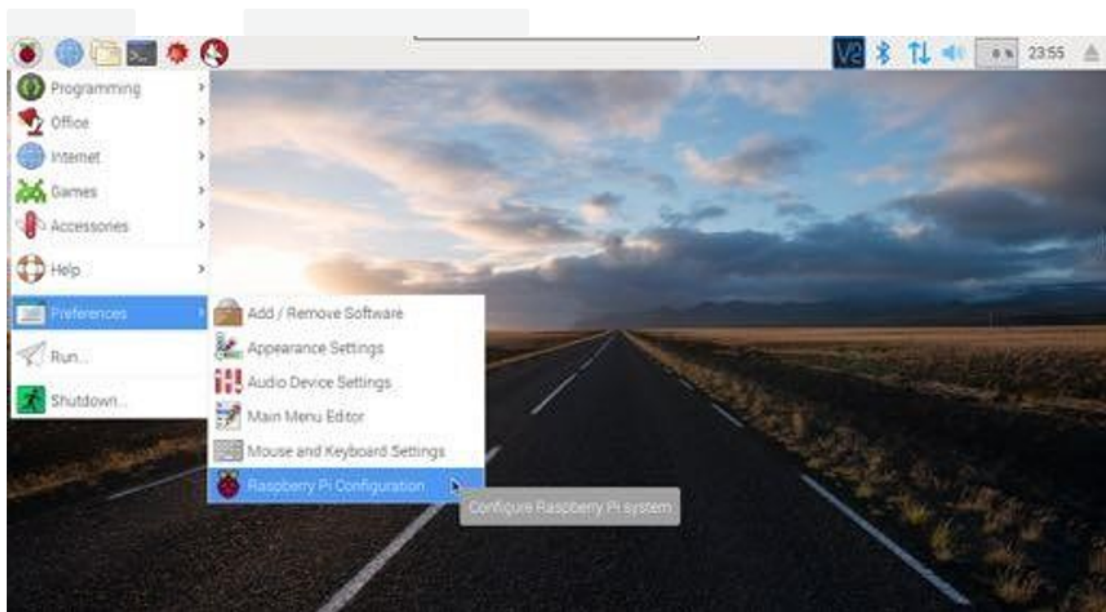


Figure 4.4: Enabling System in Raspberry Pi Configuration

Step 8:

Enable SSH and VNC, this will enable us to access the Raspberry Pi remotely without needing a TV/Keyboard later on.

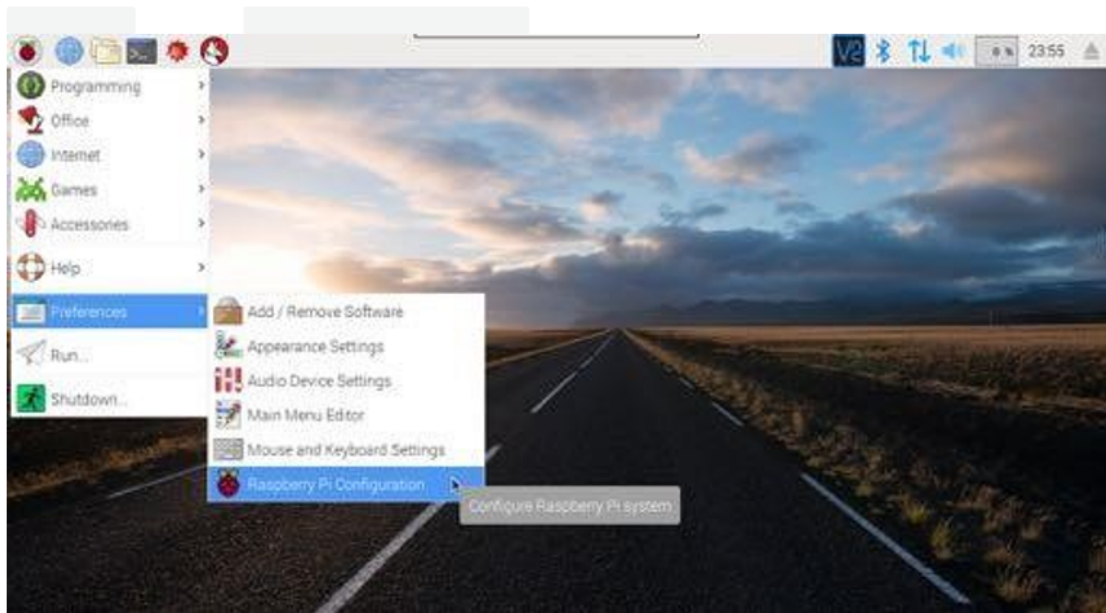


Figure 4.5: **Enabling SSH and VNC in Raspberry Pi Configuration**

Step 9:

When prompted reboot the raspberry pi.

4.3 CAMERA MODULE CONFIGURATION IN RASPBERRY PI

Install the Raspberry Pi Camera module by inserting the cable into the Raspberry Pi camera port. The cable slots into the connector situated between the USB and micro- HDMI ports, with the silver connectors facing the micro-HDMI ports. In case of confusion, just make sure the blue part of the cable is facing the USB ports on the Raspberry Pi:



Figure 4.6: **Placing Cable in the Raspberry Pi camera port**

Now boot the Raspberry Pi (plug the power in and turn it on).
Once booted, update the Raspberry Pi by running the following commands in a terminal window:

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

Now run the following command to go into the Raspberry Pi configuration tool:

```
sudo raspi-config
```

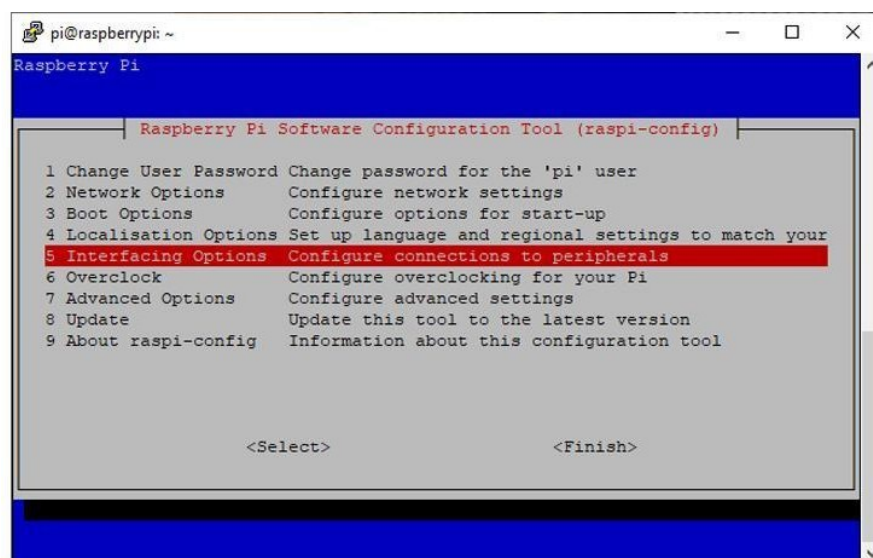


Figure 4.7: **Selecting Interfacing Option in Configuration Tool**

Navigate to Interfacing Options and hit Enter. Now select the 'Camera' option, and hit the Enter key to enable it. Select "Finish" and select to reboot the Raspberry Pi.

4.3.1 TO CAPTURE A IMAGE WITH RASPBERRY PI CAMERA MODULE

"Raspistill" is a command line application that allows to capture images with the camera module. To capture an image in jpeg format, type the following command into the terminal window:

raspistill -o image.jpg

Here, "image" is the name of the image that will be saved to the Raspberry Pi.

4.3.2 TO RECORD A VIDEO WITH RASPBERRY PI CAMERA MODULE

"Raspivid" is a command line application that allows you to capture video with the camera module. To capture a 10 second video with the Raspberry Pi camera module, run the following command:

raspivid -o video.h264 -t 10000

Here, "video" is the name of the video and "10000" is the number of milliseconds

5 TESTING AND TRAINING

5.1 TENSORFLOW

TensorFlow is an open source machine learning framework for all developers. It is used for machine learning and deep learning applications. To develop and research on fascinating ideas on artificial intelligence, Google team created TensorFlow. Google product uses machine learning in all of its products to improve the search engine, translation, image captioning or recommendations. It is designed to be executed on single or multiple CPUs and GPUs, making it a good option for complex deep learning tasks.

5.1.1 HISTORY OF TENSOR FLOW

As deep learning started to outperform all other machine learning algorithms when giving a massive amount of data. Google saw it could use these deep neural networks to improve its services:

- Gmail
- Photo
- Google search engine

They build a framework called TensorFlow to let researchers and developers work together on AI model. It was first made public in late 2015, while the first stable version appeared in 2017. It is open source under Apache Open Source license. We can use it, modify it and redistribute the modified version for free

5.1.2 TENSORFLOW ARCHITECTURE

TensorFlow architecture works in three parts

- Pre-processing the data
- Build the model
- Train and estimate the model

It is called TensorFlow because it takes input as a multi-dimensional array, also known as tensors. We can construct a sort of flowchart of operations called a Graph that we want to perform on that input. The input goes in at one end, and then it flow through this system of multiple operations and comes out the other end as output.

5.1.3 RUNNING TENSORFLOW

We can train it on multiple machines like desktop, laptop etc., then we can run it on a different machine, after we trained the model. The model can be trained and used on GPUs as well as CPUs. Stanford researchers found that GPU was very good at matrix operations and algebra so that it makes them very fast for doing these kinds of calculations. Deep learning relies on a lot of matrix. TensorFlow is very fast at computing the matrix multiplication. A significant feature of TensorFlow is the Tensor Board, which enables to monitor graphically and visually what TensorFlow is doing.

5.1.4 COMPONENTS OF TENSORFLOW

1. **Tensor:** TensorFlow's name is directly derived from its core framework: the tensor. In TensorFlow, all computations involve tensors. A tensor is a vector or matrix of n-dimensions that represents all types of data. All values in a tensor have the same data type and a known shape. The shape defines the dimensionality of the matrix or array. A tensor can originate from input data or be the result of a computation.

In TensorFlow, all operations are conducted inside a graph. A graph is a series of computations that take place successively. Each operation is called an "op" (operation) and is connected to others. The graph outlines the operations and the connections between the nodes but does not display the actual values. The edges between the nodes are tensors, and they serve as a way to supply data to the operations.

2. **Graphs:** TensorFlow makes use of a graph framework. The graph collects and describes all the sequence of computations performed during training. The graph offers several advantages:
 - It is designed to run on multiple CPUs, GPUs, or even mobile operating systems.
 - The graph can be saved, allowing computations to be preserved for immediate or future use.
 - All computations in the graph are performed by connecting tensors together. In this setup, nodes represent mathematical operations, while edges represent the tensors, showing the input/output relationships between nodes.

5.2 TRAINING TENSORFLOW OBJECT DETECTION CLASSIFIER

To detect the required object from each and every object given as input, can be done by training the object detection classifier with the set of input data. By this we will make the classifier to learn from the inputs and get trained for the detection in real time inputs. For this training of classifier, we will be using anaconda environment with python and TensorFlow

5.2.1 ANACONDA

Anaconda is a FOSS (Free and Open Source Software) distribution of the Python and R programming languages for scientific computing which includes machine learning. This software provides 7,500 + open source packages and we will be using some of these packages in this project.

1. **Download Anaconda:** Anaconda is easy to use as we can create virtual environment for any python in this software. We will be downloading anaconda python package.

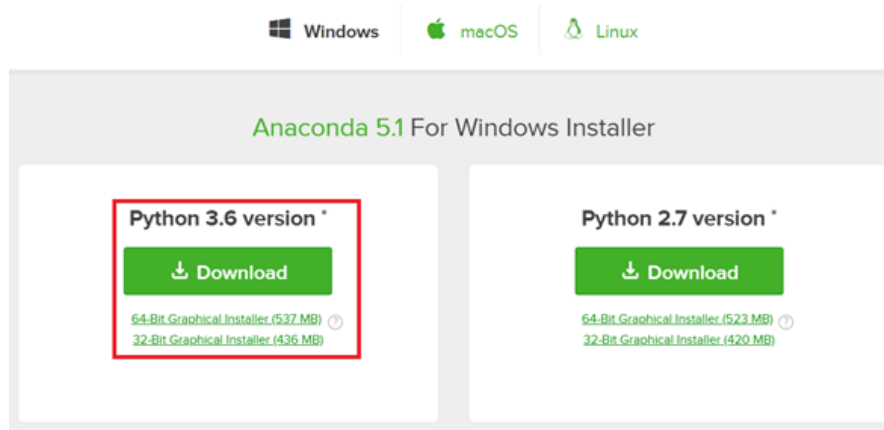


Figure 5.1: Download Python 3.6 version of Anaconda

2. **Install Anaconda:** Installing is very easy and quick once downloaded. Open the setup and follow the wizard instructions. Important thing is that it will automatically install python and some basic libraries with it.

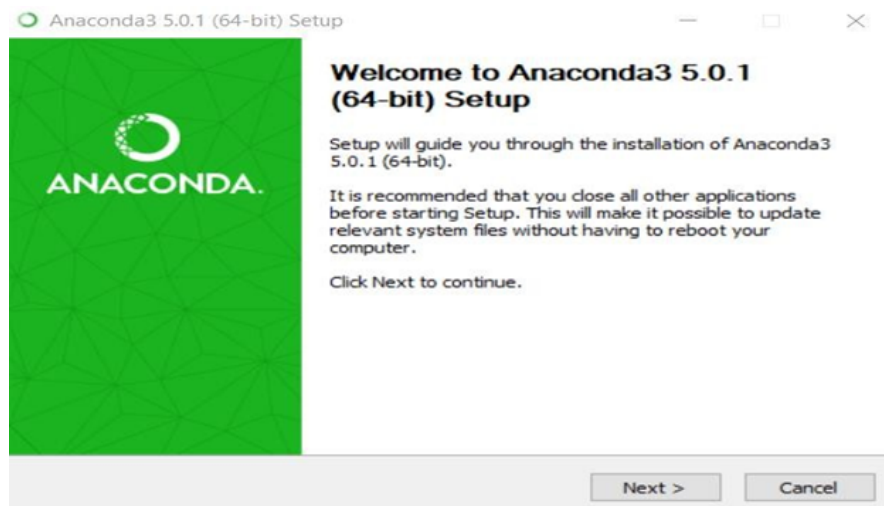


Figure 5.2: Installing Anaconda 3.5.0.1 version

5.2.2 SETTING UP OBJECT DETECTION DIRECTORY STRUCTURE AND ANACONDA VIRTUAL ENVIRONMENT:

1. **TensorFlow Object Detection API:** We created a folder named tensorflow1. This working directory will contain the full TensorFlow Object Detection framework as well as our training images, training classifier, training data, trained classifier, configuration files and everything needed for object detection classifier.

The TensorFlow models repository's code which contains the object detection API is continuously updated by the developers. Sometimes they break functionality with old versions of TensorFlow. So it is best to use correct version of TensorFlow and download the suitable models repository. We in our project used TF v1.13 [1] version. This model is downloaded and placed in the tensorflow1 folder.

2. **SSD MobileNet v2 coco model from TensorFlow's model zoo:** TensorFlow provides several object detections models (pre trained classifiers with specific neural network architectures) in its model zoo [1]. This model zoo provides a collection of detection models pre-trained on the COCO dataset, the Kitti dataset and the other things. SSD-Mobile Net model have an architecture that allows for faster detection but with less accuracy but some models such as Faster-RCNN model give slower detection but with more accuracy. We initially started with Faster-RCNN model and we are not able to convert the model in to tflite [1] for raspberry pi. So, we retrained our detector on the SSD-Mobile Net model and the detection worked considerably better. Later we are able to find that for using Object Detector on a device with low computational power (such as a smart phone or raspberry pi) use the SSD-Mobile Net model.
3. **Anaconda virtual Environment:** We will create a new anaconda environment for our specific usage so that will not affect the base of anaconda. We created a virtual environment named tensorflow1 for our project and also installed python version 3.6.6 as this is the version which currently satisfies all our libraries which are used. Being Open Source there is many difficulties in setting the working lab environment Using Pip Install Package command for TensorFlow we installed TensorFlow version 1.13. This is also we installed specifically as we use Object Detection API for our project. After clearing environment setting errors, we came to conclusion to use this version. After creating new virtual environment activate the environment update the pip.

```
(tensorflow1) c:\user\nilmaha
Python 3.6.6 :Anaconda, Inc.|(default, jun 28 2018,11:27:44)
[MSC v.1900 64
bit (AMD64)] on win32
```

4. **TensorFlow installation:** There exist two generic variants of TensorFlow, which utilize different hardware on our computer to run our computationally heavy Machine Learning algorithms. They are TensorFlow CPU and TensorFlow GPU.As we do not have the GPU processors, we decided to make use of the CPU version of TensorFlow. TensorFlow CPU which runs directly on the CPU of the machine and it is the slowest in terms of performance. Then install the other necessary packages in the table.The pandas and OpenCV-python packages are not needed by TensorFlow, but they are used in the python scripts to generate TFRecords and to work with images, videos and webcam feeds

Name	Version
Pillow	7.0.0
<u>Lxml</u>	4.5.0
<u>Cython</u>	0.29.15
Contextlib2	0.6.0.post1
<u>Jupyter</u>	1.0.0
Matplotlib	3.2.1
Pandas	1.0.3
OpenCV-python	4.2.0.32

Table 5.1: Versions specified Libraries

5. **Configuring PYTHONPATH environment variable:** A PYTHONPATH is an environment variable which can set to add additional directories where python will look for modules and packages. So, we created PYTHONPATH that points to the \models\research\slim directories. We can add from any directory using this command.

```
(tensorflow1) C:\> set PYTHONPATH=C:
\tensorflow1\models;
C:\tensorflow1\models\research;
C:\tensorflow1\models\research\slim
```

5.2.3 GATHERING AND LABELLING IMAGES

This step is very important in this project we have to give the images to learn. This is like the study material which we are providing for a student to learn and get good grades. Here also the same way we will collect all images and we will arrange and label them in a proper way and we will give that annotated images for the Neural Network to understand.

1. **Gather image:** TensorFlow needs minimum hundreds of images for an object to train a good detection classifier. To train a robust classifier, the training images should have random objects in the image along with the desired objects, and should have a variety of backgrounds and different lighting conditions. There should be some images where the desired object is partially obscured, overlapped with something else or only halfway in the picture. For our project we have 3 different objects (the jug, flask, cup). We used our android phone to take about 40 images of each object and then we took about another 60 pictures with multiple objects in the picture. We know that to detect when objects are overlapping. So we took many images with overlapping

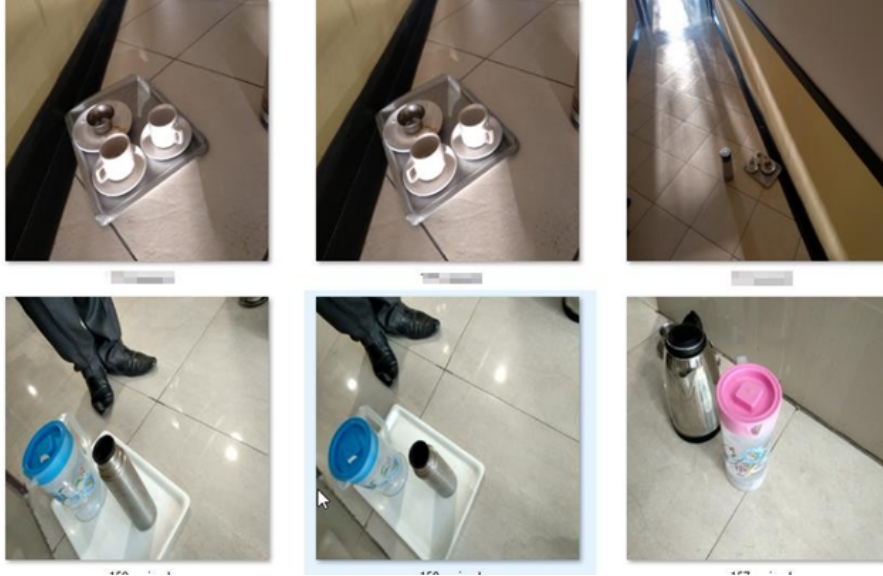


Figure 5.3: **Gathering images for annotation**

In this as all our images size was more than 3 megabyte per image it took around 800 seconds to complete 1 step in the training process ..so we decided to change all pictures size to 200 kilo bytes .To resize the images we used a script which is resizer.py .After we have all the pictures we need, move 20% of the images were moved to the \object_detection\images\test directory. 80% of the images were moved to the \object_detection\images\train directory.

2. **Label pictures:** With all the images gathered, it's time to label the desired objects in every picture. LabelImg [1] is a great tool for labeling images. This is a time-consuming process. LabelImg saves a .xml file containing the label data for each image. These .xml files will be used to generate TFRecords, which are one of the inputs to the TF trainer.

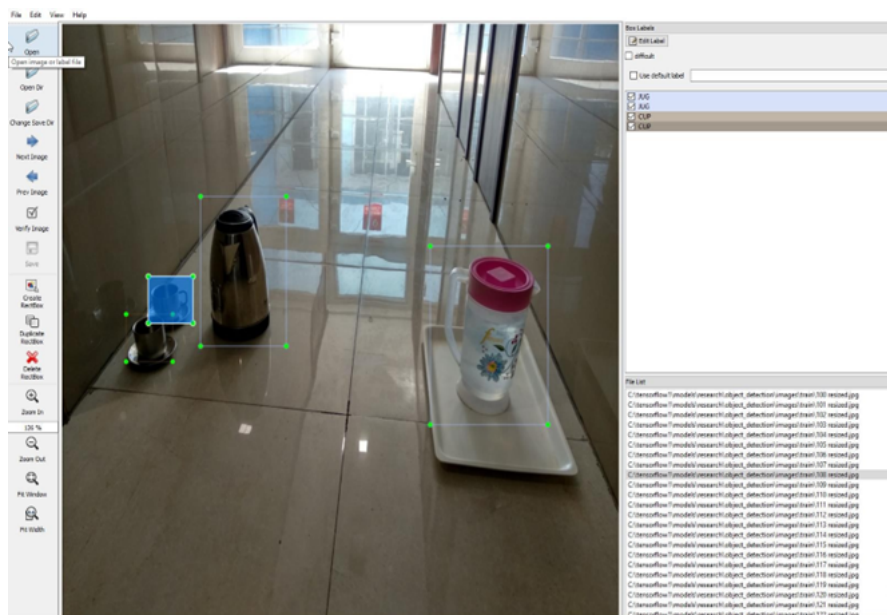


Figure 5.4: **Labeling images using LabelImg**

Once we have labeled and saved each image, there will be one .xml file for each image in the test and train directories.



Figure 5.5: XML content generated for animage

This is the LabelImg tool in this the open dir is used to open the train directory and the images in the directory are selected for the annotation. The suitable .xml file will be created for the .jpg file in the same directory. The xml file will contains the class name with the coordinates for that class name. Here class name is that the object name and the coordinates for that in the particular image is given in the xml file which would be used by the object detection classifier to learn from the group of .xml files which is the TFRecord.

5.2.4 GENERATING TRAINING DATA

With the images labeled we generated the TFRecords that serve as input data to the TensorFlow training model. We used xml_to_csv.py and generate_tfrecord.py scripts to generate the training data.

1. **CSV file:** First we will convert the .xml data to .csv files containing all the data for the train and test images.

```

(Tensorflow1)c:\tensorflow1\models\research\object_detection>
python xml_to_csv.py
Successfully converted xml to csv.
Successfully converted xml to csv.

```

This is the csv file from the above script.

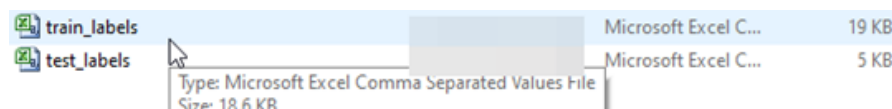


Figure 5.6: Generated csv file

1	filename	width	height	class	xmin	ymin	xmax	ymax	
2	100 resize	800	800	CUP	541	399	572	422	
3	100 resize	800	800	JUG	516	347	596	403	
4	101 resize	800	800	JUG	427	431	487	496	
5	101 resize	800	800	JUG	211	375	276	445	
6	101 resize	800	800	CUP	155	437	196	465	
7	101 resize	800	800	CUP	161	409	197	435	
8	102 resize	800	800	CUP	19	273	60	302	
9	102 resize	800	800	CUP	16	248	55	277	
10	102 resize	800	800	JUG	80	213	158	296	

Table 5.2: Data content inside a CSV file

The xml data seen above is all combined and obtained as a .csv file .This file is used to generate the TFRecord file .If you see in this the 100 image number is having 2 objects in the picture . So it has 2 rows filled up with the coordinates with the class name specified that is the CUP and JUG.

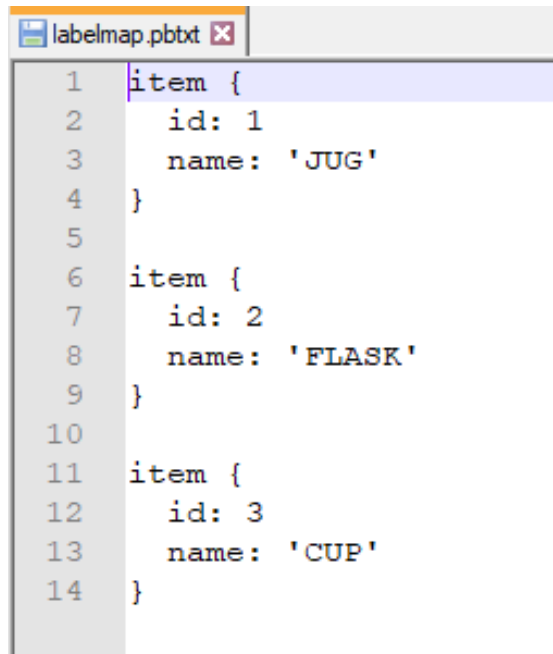
2. **TFRecord:** As we are working with large datasets, using a binary file format for storage of our data can have a significant impact on the performance of our import pipeline and for the training time of our model. Binary data takes up less space on the disk, takes less time to copy and can be read much more efficiently from disk. Along with its advantages in performance It also makes it easy to combine multiple datasets and integrates with the data import and preprocessing functionality provided by the library. Especially for datasets that are too large to be stored fully in memory this is an advantage as only the data that is required at the time is loaded from the disk and then processed and it is possible to store sequence data

A TFRecord [1] file stores data as a sequence of binary strings. This means you need to specify the structure of our data before we write it to the file. Using TensorFlow TFRecords is a convenient way to get our data into our machine learning pipeline. A protocol buffer is a method developed by Google to serialize structure data in an efficient way. To generate the TFRecord files by issuing these commands from the \object detection folder

```
(tensorflow1) C:\tensorflow1\models\research\object_detection>
python generate
_tfrecord.py --csv_input=images\train_labels.csv --image_dir=images\
train --
output_path=train.record
Successfully created the TFRecords: C:\tensorflow1\models
\research\object_det
ection\train.record
(tensorflow1) C:\tensorflow1\models\research\object_detection
>python generate
_tfrecord.py --csv_input=images\test_labels.csv
--image_dir=images\test --out
put_path=test.record
Successfully created the TFRecords:
C:\tensorflow1\models\research\object_
detection\test.record
```


5.2.5 CREATE LABEL MAP AND CONFIGURE TRAINING

1. **Label map:** The label map tells the trainer what each object is by defining a mapping of class names to class ID numbers. Use a text editor we created a new file and saved it as labelmap.pbtxt.



```
1 item {
2   id: 1
3   name: 'JUG'
4 }
5
6 item {
7   id: 2
8   name: 'FLASK'
9 }
10
11 item {
12   id: 3
13   name: 'CUP'
14 }
```

Figure 5.7: Creating label map

In this class names are JUG, CUP, FLASK these names are mapped to class ID numbers. This label map ID numbers is already specified in the generate_TFRecord.py script. So this ID numbers in the labelmap.pbtxt should match with that generate_TFRecord.py

2. **Configure Training:** The object detection training pipeline must be configured. It defines which model and what parameters will be used for training. The ssd_mobilenet_v2_coco.config file is copied and pasted in the \object detection\training directory from the \object detection\samples\configs directory. There are several changes to make to the .config file mainly changing the number of classes and examples and adding the file paths to the training data. Made following changes to the ssd_mobilenet_v2_coco.config file. The paths must be entered with single forward slashes, or TensorFlow will give a file path error when trying to train the model.

Line 9 – changed num_classes to the number of different objects wanted to classify,

```
model {
  ssd{
    num_classes:3
    box_coder{
```

Line 156 – changed file_tune_checkpoint to:

Line 175 and 177 – In the `train_input_reader` section we changed `input_path` and `label_map_path` to :

```
    train_input_reader: {
      tf_record_input_reader {
        input_path: "c:/tensorflow1/models/research/object_detection/tain.record"
      }
      label_map_path "c:/tensorflow1/models/research/object_detection/training
/labelmap.pbtxt"
    }
  }
```

Line 181 – changed `num_examples` to the number of images having in the `\images \test` directory

```
    eval_config:{
      num_examples:41
    }
```

Line 189 and 191 – In the `eval_input_reader` section, changed `input_path` and `label_map_path` to:

```
    eval_input_reader:{
      tf_record_input_reader{
        input_path: "c:/tensorflow1/models/research/object_detection
/test.record"
      }
      label_map_path: "c:/tensorflow1/models/research/object_detection
/training/labelmap.pbtxt"
```

5.2.6 RUN THE TRAINING

This is the training section where we will be giving training to the pretrained model using our data. TensorFlow will initialize the training, if everything has been set up correctly. We struggled in this training part for more than a week and finally we were able to run the training section. The command we used to begin the training:

```
(tensorflow1)c:\tensorflow1\models\research\object_detection
>python train.py
--loglostderr --tain_dir=training/ --pipeline_configuring
_path=training/ssd_mobilenet_v2_coco.config
```

Each steps of training reports the loss. It will start high and get lower as training progresses. For our training on the `ssd_mobilenet_v2_coco` model, it started at about 20 and reduced slowly. It is recommended to train until the loss consistently drops below 2.0 but it will take about 40,000 steps (depending on how powerful our CPU are) In this we have shown the loss at the step 265 and it goes on. As we are using CPU for processing so the time for each step is approximately 30

seconds. Suppose if we use GPU then the time will reduce to minimum 5 sec/step. We scripted the training routine periodically to save checkpoints about every five minutes. We can terminate the training by pressing ctrl+c. We typically wait until just after a checkpoint has been saved to terminate the training.

We can view the progress of the training job by using tensor board. To do this, open a new instance of Anaconda prompt, activate the tensorflow1 virtual environment, and change to the object detection directory and issue following command:

We can terminate training and start it later, and it will restart from the last saved checkpoint. The checkpoint at the highest number of steps will be used to generate the frozen inference graph.

```
(tensorflow1) c:\tensorflow\models\research\object_detection>tensorboard
--logdir+training --host 192.168.1.5
```

This will create a webpage on our local host which can be viewed through a web browser. The Tensor board page provides information and graphs that show how the training is progressing. One important graph is the loss graph, which shows the loss of the classifier over time

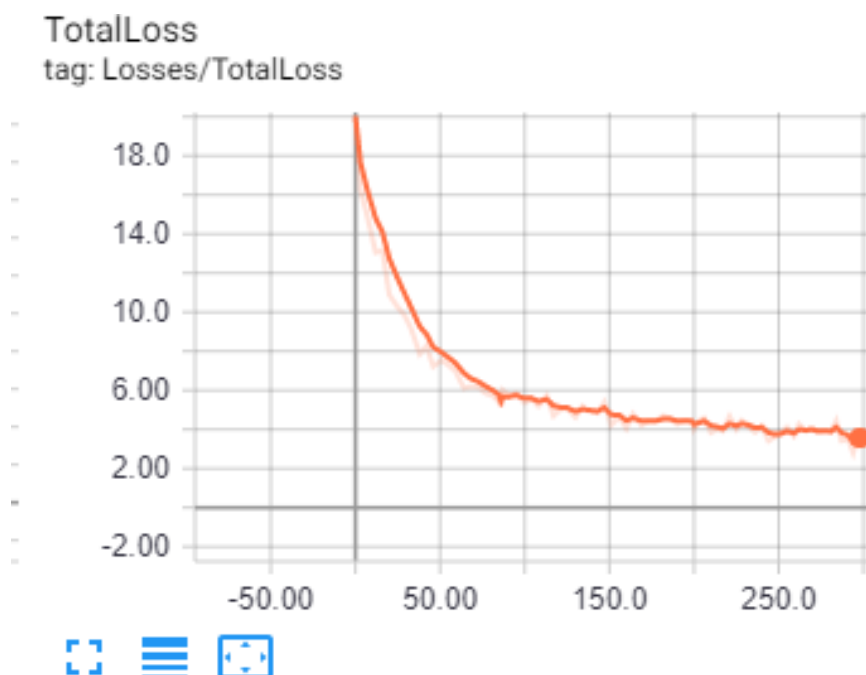


Figure 5.8: Total loss graph

From this we can understand that at first the loss starts from above 18 which is exactly 20 as already mentioned. The x-axis which represents the number of steps as we have 300 steps the it is shown in the graph. The y-axis represents the loss value. At first the loss was 20 for 1st step then after each step increasing the loss reduces rapidly and it comes to around 3.45 loss.

5.2.7 TENSORFLOW OBJECT DETECTION API ON THE RASPBERRY PI

The following steps are done in Raspberry Pi to perform object detection API on live video feeds from aPicamera or USB webcam. This module includes the

Objectdetectionpicamera.py script, which is a Python script that loads an object detection model in TensorFlow and uses it to detect objects in a Picamera video feed. We use TensorFlow v1.8.0 on a Raspberry Pi Model 3B+ running on Raspbian Stretch.

Step 1: Update the Raspberry Pi

First, the Raspberry Pi needs to be fully updated. Open the terminal window and type:

```
sudo apt-get update  
sudo apt-get upgrade
```

The upgrade will happen anywhere between a minute and an hour

Step 2. Install TensorFlow

The next step is to install TensorFlow which is more than 100MB in size. The TensorFlow is installed by using the following command in the terminal window.

pip3 install TensorFlow

TensorFlow also need some dependencies to work with so in order to install such dependencies in the terminal window we issue the following commands:

```
sudo pip3 install pillow lxmljupyter matplotlib cython
```

```
sudo apt-get install python-tk
```

Step 3. Install OpenCV

The next step is to install OpenCV by installing all its dependencies first and installing OpenCV by issuing the following commands in the terminal window:

```
sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
```

```
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
```

```
sudo apt-get install libxvidcore-dev libx264-dev
```

```
sudo apt-get install qt4-dev-tools libatlas-base-dev
```

Now that we've got all those installed, we can install OpenCV. Issue

```
sudo pip3 install opencv-python
```

Step 4. Compile and Install Protobuf

The TensorFlow object detection API uses Protobuf, a package that implements Google's Protocol Buffer data format. We need to compile this from source, by issuing the following command in terminal window:

```
sudo apt-get install protobuf-compiler
```

After installing the protoc version is made to run.

Step 5. Set up TensorFlow Directory Structure and PythonPath Variable

In the previous steps the packages are installed, the next step is to set up a TensorFlow path variable, for that we must move back to the tensor flow directory and make cd into it.

```
mkdir tensorflow1
```

```
cd tensorflow1
```

Next, we need to modify the PYTHONPATH environment variable to point at some directories inside the TensorFlow repository we just downloaded. We want PYTHONPATH to be set every time we open a terminal, so we have to modify the .bashrc file. Open it by issuing:

```
sudo nano ~/.bashrc
```

Move to the end of the file, and on the last line, add:

```
export
```

```
PYTHONPATH =
```

```
$PYTHONPATH: /home/pi/tensorflow1 / models/research: /home/pi/tensorflow1 / models/research/slim
```

Then, save and exit the file. This makes it so the “export PYTHONPATH” command is called every time you open a new terminal, so the PYTHONPATH variable will always be set appropriately. Close and then re-open the terminal. Now, we need to use Protoc to compile the Protocol Buffer (. proto) files used by the Object Detection API. The. proto files are located in /research/object_detection/protos, but we need to execute the command from the /research directory.

Issue:

```
cd /home/pi/tensorflow1/models/research
```

```
protoc object_detection/protos/*.proto --python_out=.
```

This command converts all the "name". proto files to "name_pb2".py files. Next, move into the objectdetection directory:

```
cd /home/pi/tensorflow1/models/research/object_detection
```

Now, we have to download the SSD Lite model from the TensorFlow detection model zoo. The model zoo is Google’s collection of pre-trained object detection models that have various levels of speed and accuracy. The Raspberry Pi has a weak processor, so we need to use a model that takes less processing power. Though the model will run faster, it comes at a tradeoff of having lower accuracy. We have to download the SSDLiteMobileNet model and unpack it by issuing:

```
wget  
http://download.tensorflow.org/models/object_detection/  
ssdlite_mobilenet_v2_coco_
```

2018_05_09.tar.gz
tar.gz

Now the model is in the object_detection directory and ready to be used.

Step 6. Detect Objects

The Object_detection_picamera.py, detects objects in live feeds from a Picamera or USB webcam. Basically, the script sets paths to the model and label map, loads the model into memory, initializes the Picamera, and then begins performing object detection on each video frame from the Picamera. Before using Pi Camera, we have to enable its configuration.

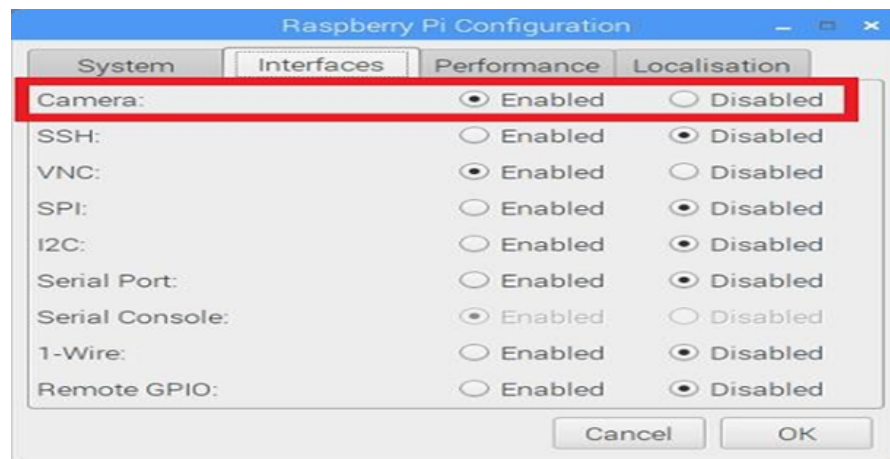


Figure 5.9: Enabling Camera Module in Raspberry Pi configuration menu

Then we have to Download the Object_detection_picamera.py file into the object_detection, the for running the pi camera we should issue:

python3 Object_detection_picamera.py

The script defaults to using an attached Picamera.

Once the script initializes (which can take up to 30 seconds), we will see a window showing a live view from our camera. Common objects inside the view will be identified and have a rectangle drawn around them.

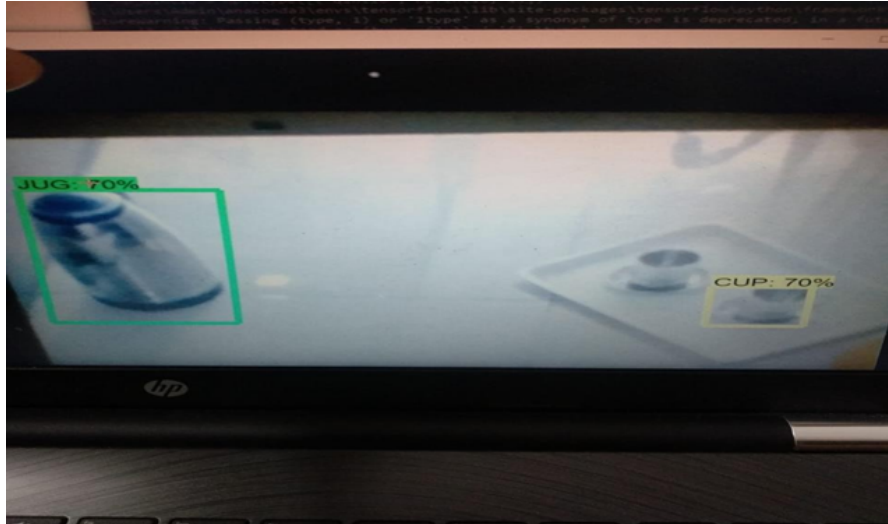


Figure 5.10: **Final Output of testing**

With the SSDLite model, the Raspberry Pi 3 performs fairly well, achieving a frame rate higher than 1FPS. This is fast enough for most real-time object detection applications. Now, open the `Object_detection_picamera.py` script in a text editor. Go to the line where `MODEL_NAME` is set and change the string to match the name of the new model folder. Then, on the line where `PATH_TO_LABELS` is set, change the name of the labelmap file to match the new label map. Change the `NUM_CLASSES` variable to the number of classes your model can identify. Now, when we run the script, it will use our model rather than the SSDLiteMobileNet model.

6 FUTURE SCOPE AND CONCLUSION

The proposed system was trained with 3 different kinds of datasets namely jug, cup and flask using SSDmobilenet model v2, and the test results were with 96% accuracy and 0.9 frames per second. Although the accuracy seems to be good, the frames per second is very low for a real time detection. In case of low traffic object detection where the objects are slow moving through the frame, then we can certainly consider using the Raspberry Pi for deep learning object detection. However, if we are developing an application that involves many objects that are fast moving, we should instead consider faster hardware.

In future this application could be improved with the increased processing power by using a tool named Coral which is a USB device that provides an edge Tensor Processing Unit (TPU) that highly accelerates machine learning model inference when attached to a Linux host computer (including Raspberry Pi), such that frames per second could be enhanced when compared to raspberry pi which operates on a lesser RAM of 1 GB. Thus, the project will be useful to detect and track the objects and makes the life easier.

References

- [1] Mohammed Noman, Vladimir Stankovic and Ayman Tawfik (2020) “Portable offline indoor object recognition system for the visually impaired,” *Cogent Engineering*, 7:1, 1823158, DOI: 10.1080/23311916.2020.1823158
- [2] G. E. Sakr, M. Mokbel, A. Hadi, and A. Darwich, “Comparing deep learning and support vector machines for autonomous waste sorting,” in *2016 IEEE International Multidisciplinary Conference on Electrical Technology (IMCET)*, vol. 26, no. 3, pp. 657–853, March 2016.
- [3] V. Tiagarajah and M. S. Subuhan, “Traffic light using TensorFlow object detection framework,” in *2017 IEEE 9th International Conference on System Engineering and Technology (ICSET)*, vol. 27, no. 4, pp. 567–598, March 2017.
- [4] A. Zeroual, P. F. Rocha, and M. P. Silva, “SSD MultiBox object detection,” in *Proceedings of the International Conference on Networking and Advanced Systems (ICNAS)*, IEEE, vol. 25, no. 2, pp. 543–538, March 2018.
- [5] W. Liu, D. Anguelov, D. Erhan, and C. Szegedy, “SSD: Single shot multibox detector,” *arXiv preprint*, Cornell University, 2019.
- [6] R. De Charette and F. Nashashibi, “Traffic light recognition using image processing compared to learning processes,” in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 15, no. 6, pp. 345–214, October 2009.
- [7] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “SSD: Single shot multibox detector,” *arXiv preprint*, 20 December 2014.
- [8] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “SSD: Single shot multibox detector,” in *European Conference on Computer Vision (ECCV)*, 2016.
- [9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “SSD: Single shot multibox detector,” *arXiv preprint*, 30 March 2018.
- [10] Raspberry Pi Foundation, “About Raspberry Pi,” [Online]. Available: <https://www.raspberrypi.org/about/>
- [11] Python Software Foundation, “Python documentation,” [Online]. Available: <https://docs.python.org/>
- [12] TensorFlow, “About TensorFlow,” [Online]. Available: <https://www.tensorflow.org/about/>