

# **SHIVA**

## **Smart Helmet for Impaired Vision Assistance**

**A Project Report**  
*Submitted by*

**Aakash Shetty**

**Rutvik Sanghvi**

**Siddhesh Save**

**Yashkumar Patel**

*Under the Guidance of*

**Prof. Ishani Saha**

*in partial fulfillment for the award of the  
degree of*

**BACHELORS OF TECHNOLOGY**  
**COMPUTER ENGINEERING**  
**At**



**MUKESH PATEL SCHOOL OF TECHNOLOGY**  
**MANAGEMENT AND ENGINEERING**

**October 2020**

# DECLARATION

I, Aakash Shetty, Rutvik Sanghvi, Siddhesh Save, Yashkumar Patel, Roll No. E034, E020, E023, E014 of B.Tech (Computer Engineering), VIII semester understand that plagiarism is defined as anyone or combination of the following:

1. Un-credited verbatim copying of individual sentences, paragraphs or illustration (such as graphs, diagrams, etc.) from any source, published or unpublished, including the internet.
2. Un-credited improper paraphrasing of pages paragraphs (changing a few words phrases, or rearranging the original sentence order)
3. Credited verbatim copying of a major portion of a paper (or thesis chapter) without clear delineation of who did wrote what. ( Source:IEEE, The institute, Dec. 2004)
4. I have made sure that all the ideas, expressions, graphs, diagrams, etc., that are not a result of my work, are properly credited. Long phrases or sentences that had to be used verbatim from published literature have been clearly identified using quotation marks.
5. I affirm that no portion of my work can be considered as plagiarism and I take full responsibility if such a complaint occurs. I understand fully well that the guide of the seminar/ project report may not be in a position to check for the possibility of such incidences of plagiarism in this body of work.

Signature of the Student: Aakash Shetty, Rutvik Sanghvi, Siddhesh Save, Yashkumar Patel

Name: Aakash Shetty, Rutvik Sanghvi, Siddhesh Save, Yashkumar Patel

Roll No. E034, E020, E023, E014

Place: Mumbai

Date:

# CERTIFICATE

This is to certify that the project entitled “SHIVA - Smart Helmet for Impaired Vision Assistance” is the bonafide work carried out by Aakash Shetty, Rutvik Sanghvi, Siddhesh Save, Yashkumar Patel of B.Tech (Computer Engineering), MPSTME (NMIMS), Mumbai, during the VIII semester of the academic year 2020-21, in partial fulfillment of the requirements for the award of the Degree of Bachelors of Engineering as per the norms prescribed by NMIMS. The project work has been assessed and found to be satisfactory.

---

Prof. Ishani Saha

Internal Mentor

---

Examiner 1

---

Examiner 2

---

Dean

# TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	<b>LIST OF FIGURES</b>	i
	<b>LIST OF TABLES</b>	ii
	<b>ABBREVIATIONS</b>	iii
	<b>ABSTRACT</b>	iv
1	<b>INTRODUCTION</b>	1
	1.1 Project Overview	
	1.2 Hardware Specifications	
	1.3 Software Specifications	
2	<b>REVIEW OF LITERATURE</b>	3
	2.1 Overview	
	2.2 Literature Survey of various papers	
	2.3 Comparison of Object detection models by literature survey	
3	<b>ANALYSIS &amp; DESIGN</b>	8
	3.1 Design of Wearable Device	
	3.2 Object Detection and OCR	
	3.3 Voice Assistant Module	
	3.4 UML diagrams	
	3.4.1 Use Case diagram	
	3.4.2 Activity Diagram	
	3.4.3 Class diagram	
4	<b>METHODS IMPLEMENTED</b>	15
	4.1 Object Detection Module	
	4.2 Voice Assistant Module	
	4.3 Optical Character Recognition Module	
5	<b>RESULTS &amp; DISCUSSION</b>	19
	5.1 Output of Object Detection Module	
	5.2 Output of Voice Assistant Module	
	5.3 Output of OCR Module	
6	<b>CONCLUSION &amp; FUTURE SCOPE</b>	25
	<b>ACKNOWLEDGEMENT</b>	26
	<b>REFERENCES</b>	26
	<b>REVIEW PAPER</b>	
	<b>RESEARCH PAPER</b>	

# LIST OF FIGURES

CHAPTER NO.	TITLE	PAGE NO.
3	<b>ANALYSIS AND DESIGN</b>	
	Fig 3.1 Front View of Helmet	8
	Fig 3.2 Bottom View of Helmet	8
	Fig 3.3 Top View of Helmet	9
	Fig 3.4 Raspberry Pi and Battery	9
	Fig 3.5 Flowchart of Object Detection and OCR with Voice Feedback	10
	Fig 3.6 Flowchart of Voice Assistant	11
	Fig 3.7 Use-Case diagram for SHIVA	12
	Fig 3.8 Activity diagram for SHIVA	13
	Fig 3.9 Class diagram for SHIVA	14
5	<b>RESULT AND DISCUSSION</b>	
	Fig 5.1 Output showing properties of Neural Network Layers – i	19
	Fig 5.2 Output showing properties of Neural Network Layers – ii	20
	Fig 5.3 Object Detection for Traffic Light	22
	Fig 5.4 Object Detection for Person	22
	Fig 5.5 Output of current date and time using Date Time Module	23
	Fig 5.6 Output of Wikipedia search using Wikipedia Module	23
	Fig 5.7 Output of sending email using SMTPLIB-i	24
	Fig 5.8 Output of sending email using SMTPLIB-ii	24
	Fig 5.9 Output for OCR	24

## **LIST OF TABLES**

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
2	<b>REVIEW OF LITERATURE</b>	
	Table 2.1 Comparison of object detection	5

\*\*\*\*\*

## **ABBREVIATIONS**

<b>ABBREVIATION</b>	<b>DESCRIPTION</b>
YOLO	You Look Only Once
SMTP	Simple Mail Transfer Protocol
CNN	Convolutional Neural Network
OCR	Optical Character Recognition
API	Application Programming Interface

## **ABSTRACT**

According to world health statistics 285 million out of 7.6 billion population suffers visual impairment; hence 4 out of 100 people are blind. Absence of vision restricts the mobility of a person to pronounced extent and hence there is a need to build an explicit device to conquer guiding aid to the prospect. This project attempts to make A visual and voice assistant for visually impaired people which can help them reach their destination and also help them to read sign boards using Computer Vision. This project proposes to build a prototype that performs real time object detection using deep neural network model, YOLOV3. Further the object, and the class of the object is prompted through speech stimulus to the blind person. Along with this we are augmenting a voice assistant for frequent requirements and utilities such as sending emails, getting information over internet, etc. This work uses a combination of YOLOv3 on pretrained dataset and darknet detection framework to build rapid real time multi object detection for a compact, portable and minimal response time device construction. Several prototypes and models have been made keeping in mind the blind people having different usages such as object recognizer for blinds, visual aid for blinds, google lookup, etc. Among these, we want to create a computer vision assisted solution keeping in mind their needs and their movements. Computer Vision based solutions are emerging as one of the most promising options due to their affordability and accessibility. This project proposes a system for visually impaired people. The proposed project aims to create a wearable visual aid for visually impaired people.



# Chapter 1

## INTRODUCTION

### 1.1 Project Overview

Visually impaired people face a lot of difficulties in their daily life. Many times, they rely on others for help. Several technologies for assistance of visually impaired people have been developed. Among the various technologies being utilized to assist the blind, Computer Vision based solutions are emerging as one of the most promising options due to their affordability and accessibility. This project proposes a system for visually impaired people. The proposed project aims to create a wearable visual aid for visually impaired people.

The project will include the following functionalities:

- A. Object Detection to detect obstacles in their path which will be helpful to walk on a road safely.
- B. Voice Assistant for basic features like performing calculations, searching on browser, etc.
- C. Optical Character Recognition to read sign boards in their path using audio output.

This will help the visually impaired person to manage day-to-day activities and to navigate through his/her surroundings. Moreover, one of our main goal is to make blind less different from the rest of us in society and the best way to do it is to help them walk without the walking stick. This device makes the walking stick obsolete and helps the blind blend into the crowd and make them look like any other individual.

### 1.2 Hardware Specifications

- Raspberry Pi: A small and efficient micro controller used for performing various tasks
- Raspberry Pi Camera or any USB Camera
- Micro SD card 16+ GB
- Power Supply to Raspberry Pi using battery.
- Internet Connection for voice assistant.
- Helmet

## 1.3 Software Specifications

- Visual Studio Code
- PyTorch
- Darknet
- YOLOv3
- MSCOCO Dataset
- Pyttsx3: Python Text to Speech Conversion Library
- SMTP Library
- Date Time Module
- Blender
- Tesseract

## Chapter 2

### REVIEW OF LITERATURE

#### 2.1 Overview

The following conclusions have been drawn from the results obtained through the literature review.

- From the results of review of various object detection models on the MS COCO dataset, it can be deduced that single staged object detection modules are efficient and gives high output speed as compared with single staged models.
- If accuracy is given preference over speed, then two staged object detectors are better, but in recent times single staged detectors also challenges to provide higher accuracy as compared to two stage detectors.
- According to Redmond et al. YOLOv3: An Incremental Approach, YOLOv3 is able to detect 10 times faster than the state-of-the-art methods. Hence YOLOv3 been selected for the experimentation.
- From the results of review of remote systems having object detection modules it can be inferred that the raspberry pi is capable of building a remote object detection module with decent amount of FPS and also various other utilities such as voice assistant can work together with it.

It has to be kept in mind that blind people don't walk as fast a normal person could walk, speed need not be a concern as long as the algorithm is able to perform object detection and recognition in real-time.

#### 2.2 Literature Survey of various papers

- **Real Time Multi Object Detection for Blind Using Single Shot Multibox Detector [29]:**

This paper proposes to build a prototype that performs real time object detection using an SSD model. Further the object, its position with respect to the person and accuracy of detection is prompted through speech stimulus to the blind person.

The SSD model is used here. New model available is YOLOv3 and it has greater FPS. The SSD model is used here for object detection and nowadays few efficient models have been developed which can serve the same purpose with high amount of accuracy. New model available such as YOLOv3 which has greater FPS and accuracy.

- **YOLO v3-Tiny: Object Detection and Recognition using one stage improved model [1]**

This paper presents the fundamental overview of object detection methods by including two classes of object detectors. In two stage detector covered algorithms are

RCNN, Fast RCNN, and Faster RCNN, whereas in one stage detector YOLO v1, v2, v3, and SSD are covered. Two stage detectors focus more on accuracy, whereas the primary concern of one stage detectors is speed.

Classification of existing models along with newly existing YOLO v3. YOLO v3 gives more comparisons wrt its predecessor. Its slower as but has higher accuracy wrt its predecessor YOLO v2. And also gives higher output speed wrt SSD model. More emphasis is given on yolov3 tiny which has less accuracy as compared to normal yolov3 and if implemented it has difficulty in recognizing items which are easily detected by yolov3 model.

- **Object Detection with Deep Learning: A Review [30]**

The paper talks about the various object detection techniques that are available at present. It discusses techniques till YOLOv3 and compares it based on several parameters such as time, precision and CNN used.

This paper provides a detailed review on deep learning-based object detection frameworks that handle different subproblems, such as occlusion, clutter and low resolution, with different degrees of modifications on R-CNN. The review starts on generic object detection pipelines which provide base architectures for other related tasks. Then, three other common tasks, namely, salient object detection, face detection, and pedestrian detection, are also briefly reviewed.

- **YOLO9000: Better, Faster, Stronger'' [20]**

The improved model, YOLOv2, is state-of-the-art on standard detection tasks like PASCAL VOC and COCO and can detect over 9000 objects. At 40 FPS, YOLOv2 gets 78.6 mAP, outperforming state-of-the-art methods like Faster R- CNN with ResNet and SSD while still running significantly faster.

YOLOv2 is state-of-the-art and faster than other detection systems across a variety of detection datasets. Furthermore, it can be run at a variety of image sizes to provide a smooth tradeoff between speed and accuracy. As this method was developed few years ago, now we came across few areas which if improved can lead to high amount of accuracy and even can get good amount of output speed. But overall, this method delivers good amount of output speed.

- **YOLOv3: An Incremental Improvement [21]**

This paper presents a bunch of little design changes to make it better. Model is trained in this new network that's pretty swell. It's a little bigger YOLO v2 but more accurate. It's still fast though. At 320 x 320 YOLOv3 runs in 22 ms at 28.2 mAP, as accurate as SSD but three times faster.

YOLOv3 is a good detector. It's fast, it's accurate. It's not as great on the COCO average AP between .5 and .95 IOU metric. But it's very good on the old detection metric of .5 IOU. The hardware requirement of this algorithm especially in terms of GPU is high as compared to its earlier versions.

- **Performance Evaluation of Real-Time Object Detection Algorithms [31]**

This paper presents comparison of several techniques for object detection regarding to speed and accuracy. There are two categories of feature extraction in object detection algorithms: knowledge based and learning based approaches. They have shown the comparison on speed and accuracy of several algorithms.

In terms of accuracy, YOLO and R-CNN are the best method for applications in fast multiple object detection. R-CNN used CNN to extract features and SVM to classify objects. Hence, R-CNN is slow on common implementation. YOLO predicts bounding box and calculates class probabilities at the same time. Hence, time computation is less than R-CNN.

- **Study on Different Region-Based Object Detection Models Applied to Live Video Stream and Images Using Deep Learning [24]**

Object detection is the computer capability to accurately identify the multiple objects present in the image or video with the bounding boxes around them and the appropriate labels indicating their class along with the confidence score indicating the degree of closeness with the class. This paper has different types of region-based object detection models which can be applied on both live video stream and images. Two stage detection models like R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN and one stage object detection models like SSD, YOLO are discussed and compared.

SSD was able to output the result in less time with low accuracy having only one bounding box around the objects, whereas Faster R-CNN model generated results with more accuracy by putting multiple bounding boxes on objects but it took more time than SSD. There exists a trade-off between speed and accuracy among different types of deep learning-based object detection models. So, it is very important to choose the model which best suits our application. The paper could include comparison among single stage detection models and among two stage detection models considering accuracy and performance.

## 2.3 Comparison of object detection models by literature survey.

Object Detection Model	Advantage	Disadvantage
R-CNN	i. The network transforms the object detection problem into the classification problem and greatly improve the accuracy.	i. Selective Search algorithm is not flexible. ii. It cannot be used for real-time processing due to its slow speed. iii. Training requires lot of time as multiple stages are involved.

Fast R-CNN	<ul style="list-style-type: none"> <li>i. It's raining and testing are significantly faster than SPP-net.</li> <li>ii. The input image can be any size.</li> </ul>	<ul style="list-style-type: none"> <li>i. The approach is complicated and lengthy.</li> <li>ii. Calculation time is very high.</li> <li>iii. Computation time for the model is around 2 seconds.</li> </ul>
Faster R-CNN	<ul style="list-style-type: none"> <li>i. Faster R-CNN has a Region Proposal Network (RPN) to generate a fixed set of regions. The RPN is implemented as a fully convolutional network that predicts object bounds and objectness scores at each position.</li> <li>ii. The different components are combined in a single setup and are trained either end-to-end or in multiple phases (to improve stability).</li> </ul>	<ul style="list-style-type: none"> <li>i. It requires significantly more computational power.</li> <li>ii. The training process is complex, and there is still much room for optimization in the calculation process.</li> </ul>
SSD	<ul style="list-style-type: none"> <li>i. Different feature maps in the convolutional network correspond with different receptive fields and are used to naturally handle objects at different scales.</li> <li>ii. All the computation is encapsulated in a single network and fairly high computational speeds are achieved.</li> </ul>	<ul style="list-style-type: none"> <li>i. A larger image size will perform better as small object are often hard to detect, but it will have a significant computational cost.</li> <li>ii. The robustness of this network to small object detection is not high.</li> </ul>
YOLOv1	<ul style="list-style-type: none"> <li>i. The model is suitable for real time object detection having predictions made in one pass from a single neural network.</li> <li>ii. Region Proposal Network accesses the whole image in one pass by sliding window and the bounding boxes are limited to particular regions from the image. And in turn generates fewer false</li> </ul>	<ul style="list-style-type: none"> <li>i. Because of the low-resolution classifier of YOLOv1 the model could not determine the low-resolution objects present in the image.</li> <li>ii. Due to absence of anchor boxes the model cannot predict more than one box for a particular region.</li> <li>iii. Also due to congestion of some objects it can miss</li> </ul>

	<p>positives for the less specific regions of the image.</p>	<p>out some objects and can also miss some details.</p> <p>iv. The model is less flexible and cannot incorporate testing with various resolution, and finds difficulty in localizing.</p>
YOLOv2	<p>i. Batch normalization has been introduced.</p> <p>ii. It regularizes the pattern and deletion of dropout can take place from the sample without overfitting it.</p> <p>iii. High resolution classifier has been added which increases resolution to 448x448.</p> <p>iv. Anchor Boxes have been added for multi-class identification.</p>	<p>i. With the increase in speed, accuracy is compromised.</p> <p>ii. No. of comparisons boxes are less as compared to other newer models.</p>
YOLOv3	<p>i. YOLOv3 modifies the way the cost function is measured.</p> <p>ii. YOLOv3 has a better class predictor.</p> <p>iii. A multilabel approach in YOLOv3 gives 3 predictions for every location which gives a better prediction accuracy.</p> <p>iv. YOLOV3 has a better feature extractor.</p>	<p>i. High computational is required for better performance.</p> <p>ii. Less output frames w.r.t. YOLOv2.</p>
YOLOv4	<p>i. Each GPU is assigned with mini batches for training but this leads to slow and inappropriate training. To overcome this shortcoming, a smaller mini batch is created by YOLOv4 object detector.</p> <p>ii. There is use of data augmentation to increase robustness.</p>	<p>i. High computational is required for remote usage.</p>

Table 2.1 Comparison of object detection models

## Chapter 3

### ANALYSIS & DESIGN

#### 3.1 Design of Wearable Device

We have designed a wearable device in the form of a helmet. This device will hold a battery which will power our device. The wearable gear which we will design will hold the hardware in a way which will be most comfortable for the user and the camera will be placed in such a way that the user's complete path can be visible. The hardware will perform the tasks of object detection with voice feedback along with voice assistant. Hardware consists of Raspberry PI which provides processing power and battery along with it. The designing of helmet is done on Blender Application which is a powerful tool to design 3-D models.

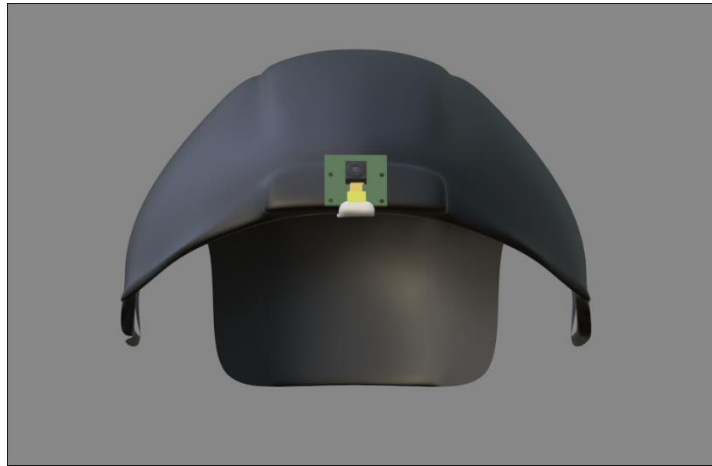


Fig 3.1 Front View of Helmet

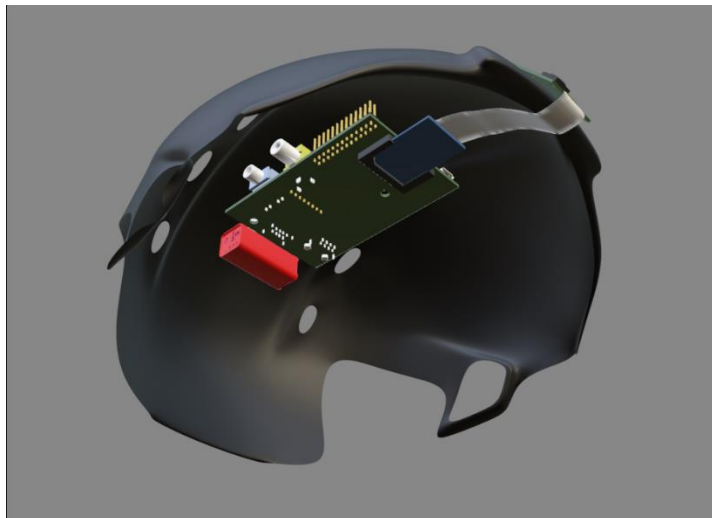




Fig 3.2 Bottom View of Helmet

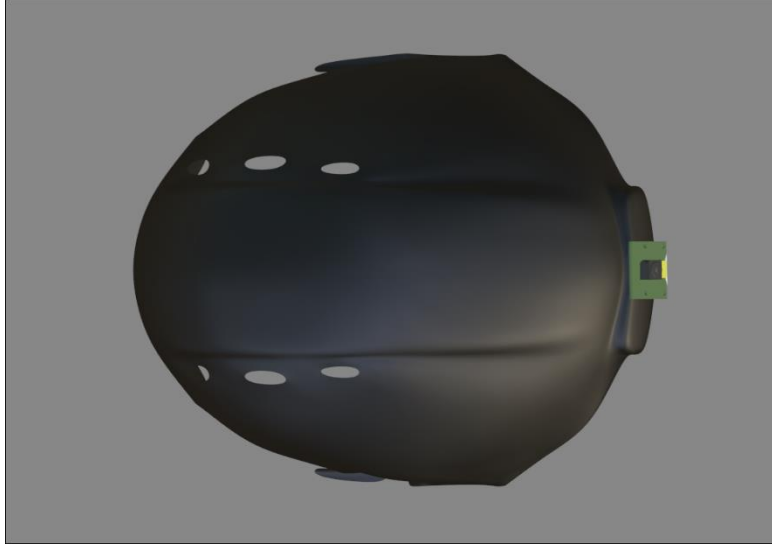


Fig 3.3 Top View of Helmet

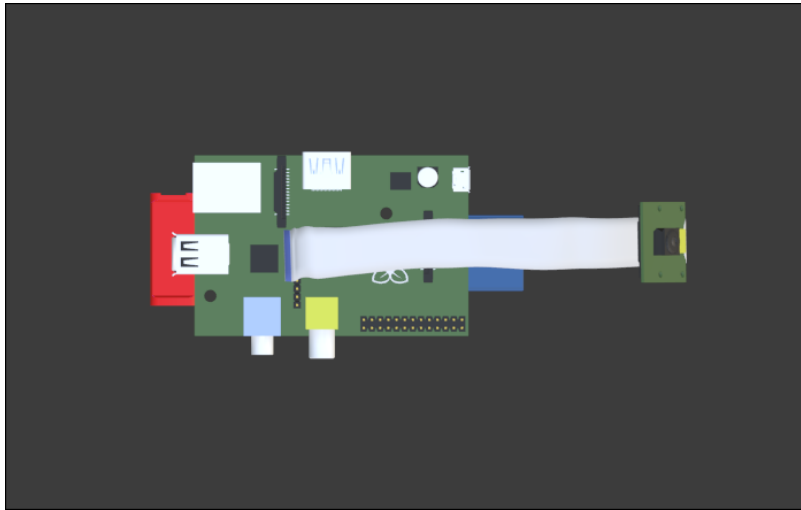


Fig 3.4 Raspberry Pi and battery

## 3.2 Object detection and OCR

Object detection is a task in computer vision that involves identifying the presence, location, and type of one or more objects in the given photograph. It is a challenging problem that involves building upon methods for object recognition, object localization, and object classification. Along with Object detection we are implementing OCR. All of these will take place by using models and algorithms of deep learning.

- First, we will get the Real time Video input from the raspberry Pi camera module. The

available real time video input can be used for OCR or for object recognition.

- For OCR, the incoming video must be pre-processed then the Features are extracted and are classified and post-processed for recognizing the output. This Text is passed through the speech recognition API and this module converts the given text into voice. The Python Library pytesseract is used for OCR and its augmentation with voice assistant module will result in voice feedback.
- For Object Detection, the incoming video is to be scaled into proper sizes and this incoming video is to be passed through a network of convolutional layers to get the output. TensorFlow library of python is used for object detection or object recognition. It segregates the objects based on their classes and the classes are passed to text to speech API for getting voice feedback.

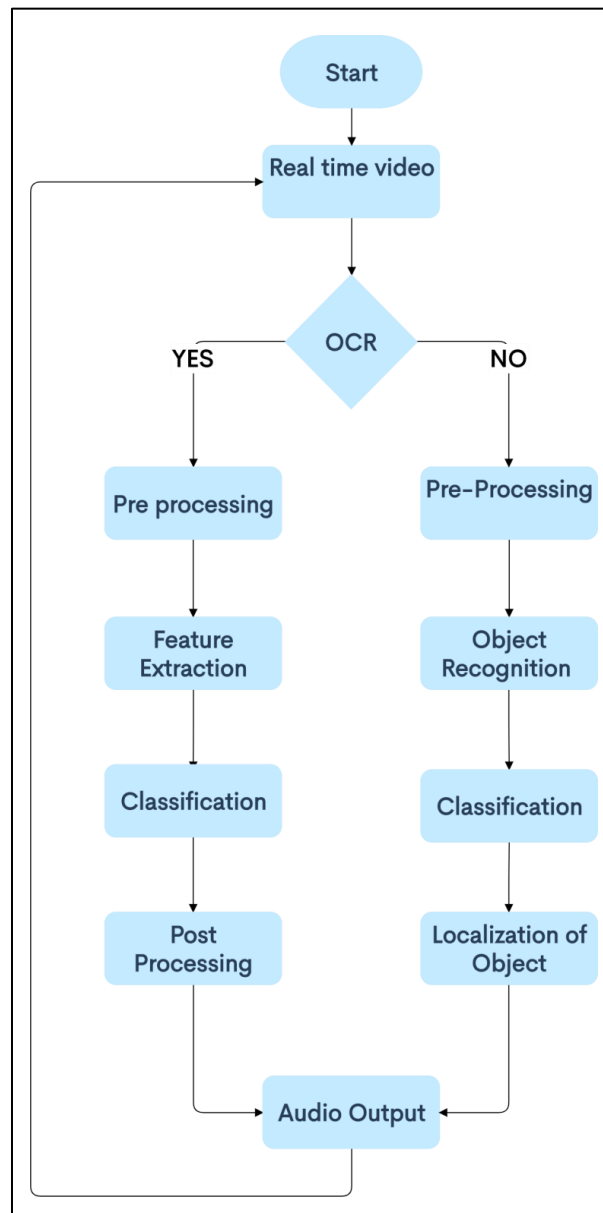


Fig 3.1 Flowchart of Object Detection and OCR with Voice Feedback

### 3.3 Voice Assistant Module

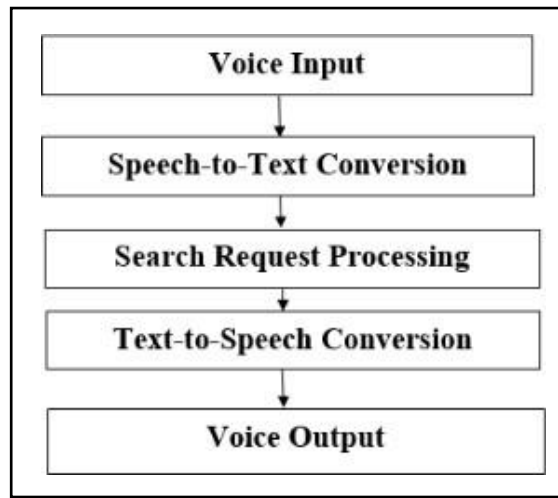


Fig 3.2 Flowchart of Voice Assistant

The system works on the principle of speech recognition. Firstly, the user starts the device (Raspberry Pi), on start up the assistant greets the user. The python script which consists of the program code is executed automatically (auto-start) on system startup. The input to the system is fed through the microphone in speech form, the system records this speech form until a pause is observed to know the user has finished the request and then it is converted in text form for the machine to understand. The algorithm is built in such a way that the system processes the request and searches for the keyword and delivers appropriate output in textual form, which is later converted to speech using the speech-recognition module. This speech is delivered via output speakers connected to 3.5 mm audio jack.

## 3.4 UML Diagrams

### 3.3.1 Use-Case Diagram

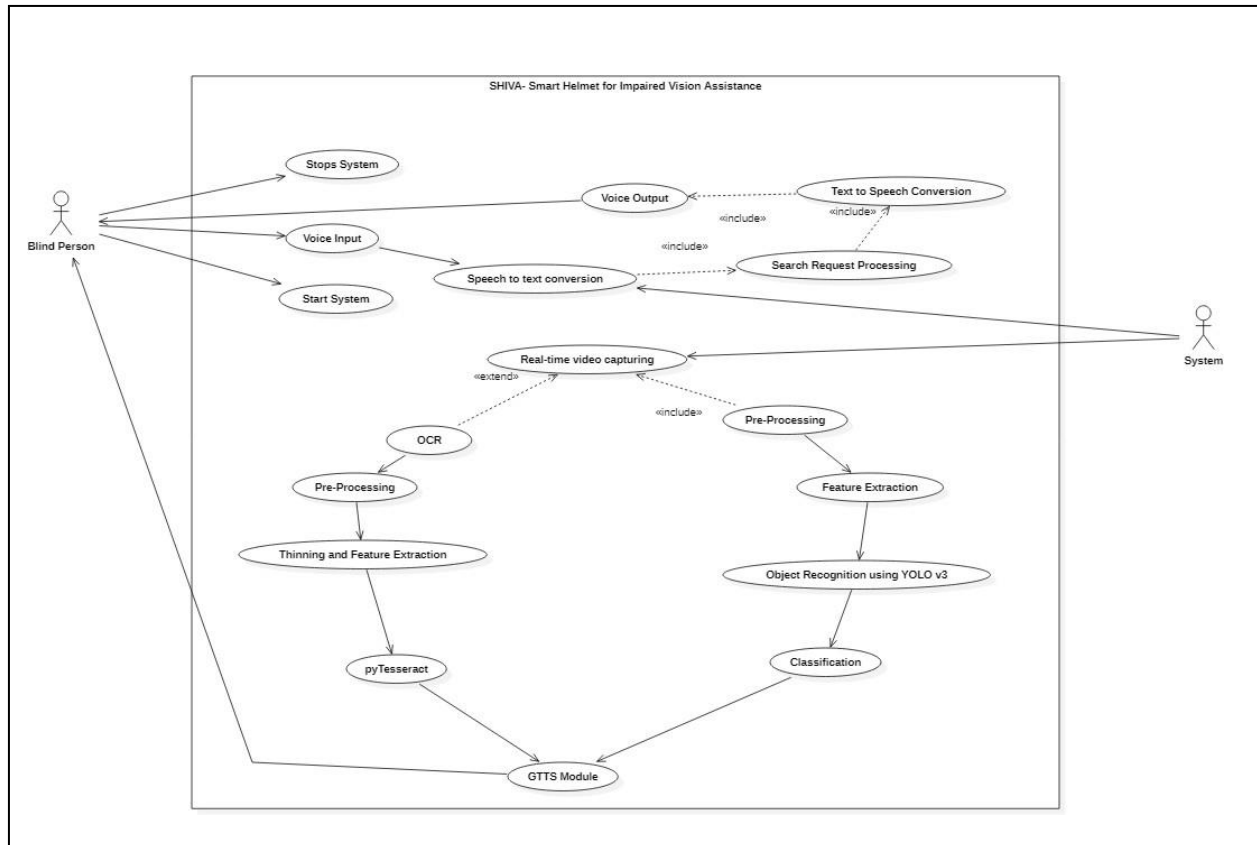


Fig 3.3 Use-case diagram for SHIVA

### 3.3.2 Activity Diagram

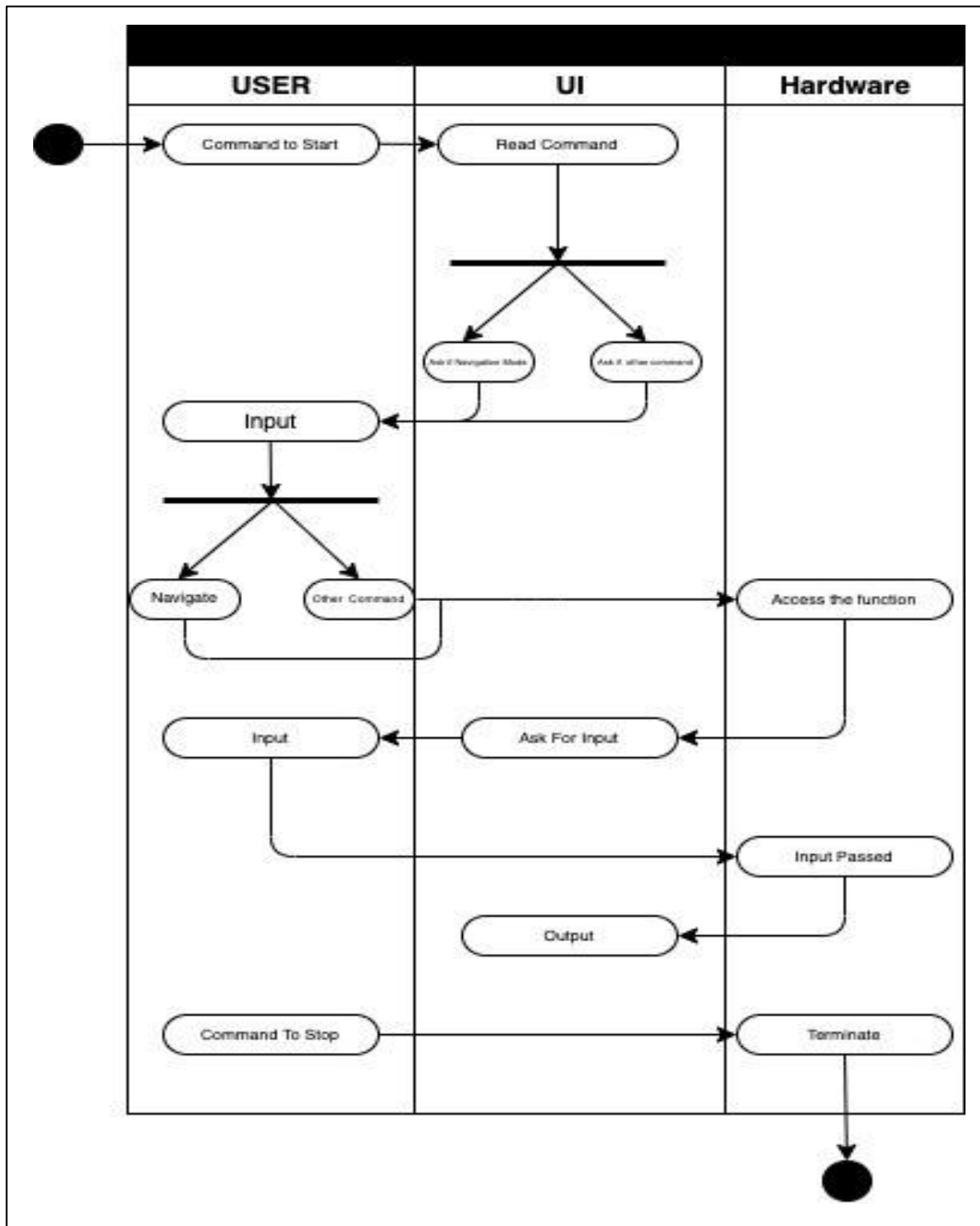


Fig 3.4 Activity diagram for SHIVA

### 3.3.3 Class Diagram

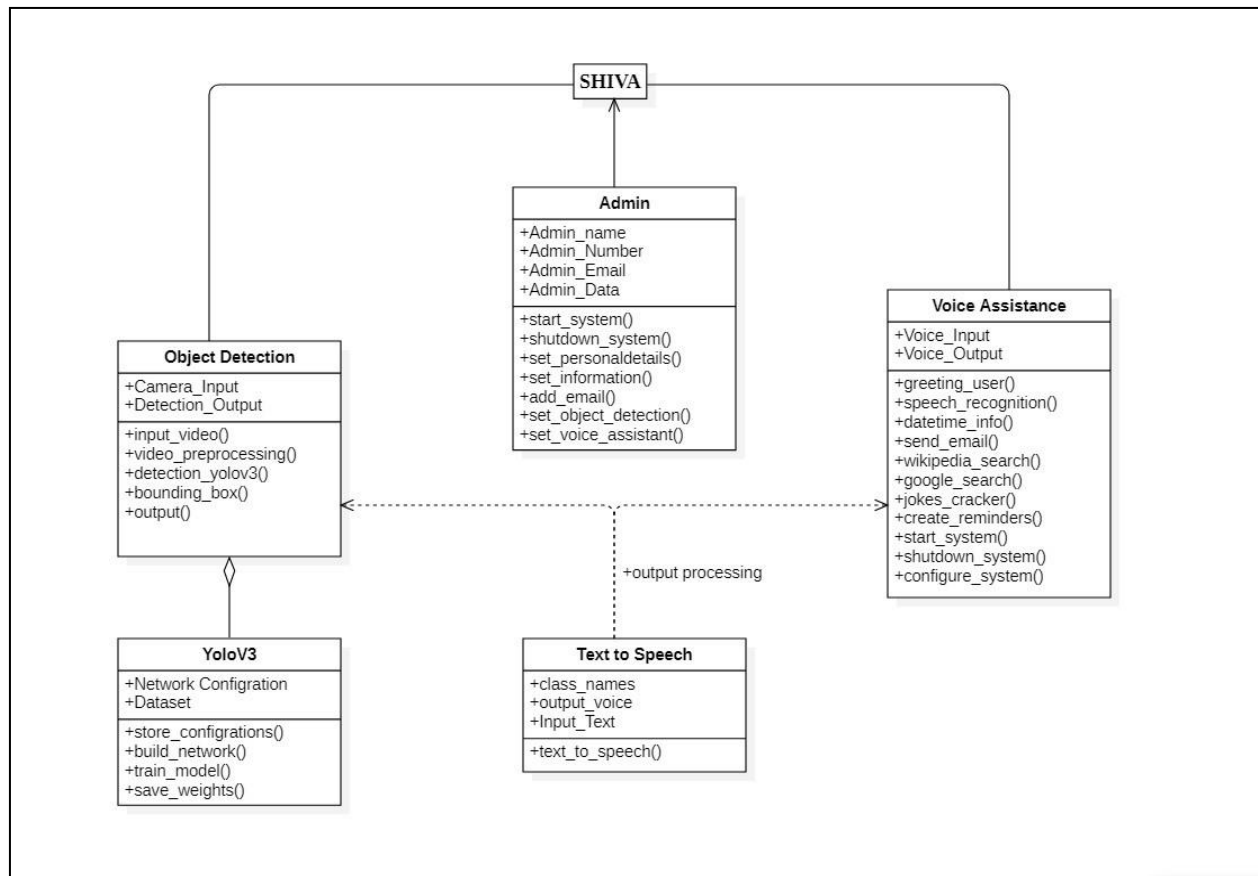


Fig 3.5 Class diagram for SHIVA

## Chapter 4

### METHODS IMPLEMENTED

#### 4.1 Object Detection Module:

Python supports creation of various neural network modules due to large support of inbuilt libraries and the community behind the libraries. TensorFlow and PyTorch are two frameworks for building neural network models. We have used PyTorch because it follows OOP paradigm and also PyTorch is newer as compared to TensorFlow with all the latest inbuilt packages and libraries along with easy implementation.

Steps to make an object detection module are:

- **Framework:**

Torch: Torchvision library, which is a part of Pytorch, contains all the important datasets as well as models and transformation operations generally used in the field of computer vision. It allows you to import datasets without any hassle.

Cuda: Cuda is an environment for GPU integration

```
pip install torch==1.6.0 torchvision==0.7.0 -f
https://download.pytorch.org/whl/torch_stable.html
```

- **Modules and Utilities:**

Torch.nn: class torch.nn. Module [source] Base class for all neural network modules. Your models should also subclass this class. Modules can also contain other Modules, allowing to nest them in a tree structure.

- **Build the YOLOV3 Network:**

Configuration file which contains all the information about all the layers present in YOLOV3 for building the network is given by the creators of the YOLOV3 in form of cfg file and by using the functions inherited from the torch.nn module we can create the Convolutional network for YOLOV3.

- **Defining the Forward Pass of Network:**

The forward pass of the network is implemented by overriding the forward method of the nn.Module class. Forward serves two purposes. First, to calculate the output, and second, to transform the output detection feature maps in a way that it can be processed easier (such as transforming them such that detection maps across multiple scales can be concatenated, which otherwise isn't possible as they are of different dimensions).

- **Transforming the output**

The function `predict_transform` is in the file `util.py` and `predict_transform` function takes an detection feature map and turns it into a 2-D tensor, where each row of the tensor corresponds to attributes of a bounding box, in the following order.

The dimensions of the anchors are in accordance to the height and width attributes of the net block. These attributes describe the dimensions of the input image, which is larger (by a factor of stride) than the detection map. The last thing we want to do here, is to resize the detections map to the size of the input image. The bounding box attributes here are sized according to the feature map (say, 13 x 13). If the input image was 416 x 416, we multiply the attributes by 32, or the stride variable.

- **Pre-trained Weights**

The weights are just stored as floats, with nothing to guide us as to which layer do they belong to. First, the weights belong to only two types of layers, either a batch norm layer or a convolutional layer. The weights for these layers are stored exactly in the same order as they appear in the configuration file. So, if a convolutional is followed by a shortcut block, and then the shortcut block by another convolutional block, you will expect file to contain the weights of the previous convolutional block, followed by those of the latter.

With our model built, and weights loaded, we can finally start detecting objects.

- **Applying Bounding Boxes:**

For each of the bounding box having a objectness score below a threshold, we set the values of it's every attribute (entire row representing the bounding box) to zero. The bounding box attributes we have now are described by the center coordinates, as well as the height and width of the bounding box.

- **Pre-Processing:**

OpenCV loads the input from live webcam by `cv2.VideoCapture()` function. PyTorch's image input format is (Batches x Channels x Height x Width), with the channel order being RGB. Therefore, we created the function `prep_image` in `util.py` to transform the numpy array into PyTorch's input format. Then we have scaled the webcam input into the dimensions which will give good enough accuracy and also will give good number of Frames Per Second. Then we transposed the webcam input and then this things are given to the object detector for feature extraction and classification.

- **The Detection Loop:**

We iterate over the batches, generate the prediction, and concatenate the prediction tensors of all the inputs we have to perform detections upon.

Every iteration, we keep a track of the number of frames captured in a variable called `frames`. We then divide this number by the time elapsed since the first frame to print the FPS of the video. After that, we print time taken for each detection as well as the object



detected in each image. If the output of the write\_results function for batch is an int(0), meaning there is no detection, we use continue to skip the rest loop.

- **Writing the predictions:**

The function write\_results outputs a tensor of shape D x 8. Here D is the true detections in all of images, each represented by a row. Each detections has 8 attributes, namely, index of the image in the batch to which the detection belongs to, 4 corner coordinates, objectness score, the score of class with maximum confidence, and the index of that class.

## 4.2 Voice Assistance Module:

- Python is a suitable language for script writers and developers. Let's write a script for Voice Assistant using Python. The query for the assistant can be manipulated as per the user's need.
- Many functionalities are added so as to make Voice Assistant very interactive and friendly to use.
- Speech recognition is the process of converting audio into text. This is commonly used in voice assistants like Alexa, Siri, etc. Python provides an API called SpeechRecognition to allow us to convert audio into text for further processing. In this article, we will look at converting large or long audio files into text using the SpeechRecognition API in python.
- Pyttsx3: - This module is used for conversion of text to speech in a program it works offline. To install this module type the below command in the terminal.
- pip install pyttsx3
- Wikipedia: - As we all know Wikipedia is a great source of knowledge and we have used Wikipedia module to get information from Wikipedia or to perform Wikipedia search. To install this module type the below command in the terminal.
- pip install wikipedia
- Speech Recognition: - Since we're building an Application of voice assistant, one of the most important things in this is that your assistant recognizes your voice (means what you want to say/ ask). To install this module type the below command in the terminal.
- pip install SpeechRecognition
- Web browser search: - To perform Web Search through Chrome Browser. This module comes built-in with Python.
- Datetime: - Date and Time is used to showing Date and Time. This module comes built-int with Python.
- Sending Email: - smtplib uses SMTP (Simple Mail Transfer Protocol) Library for sending emails.
- Adding Reminder: - User can ask the voice assistant to remember particular details and it will remind the user on being asked.
- Play Songs: - Songs on the device can be played on commanding voice assistant.
- Telling Jokes: - Pyjoke is an inbuilt function through which Voice Assistant can crack jokes.
- pip install Pyjoke
- Web browser: - To perform Web Search. This module comes buit-in with Python.

- Datetime: - Date and Time is used to showing Date and Time. This module comes built-in with Python.
- Sending Email: - smtplib uses SMTP (Simple Mail Transfer Protocol) Library for sending emails.

### 4.3 Optical Character Recognition Module:

Humans can understand the contents of an image simply by looking. We perceive the text on the image as text and can read it. Computers don't work the same way. They need something more concrete, organized in a way they can understand. This is where Optical Character Recognition (OCR) kicks in. Whether it's recognition of car plates from a camera, or handwritten documents that should be converted into a digital copy, this technique is very useful.

Optical Character Recognition involves the detection of text content on images/videos and translation of the images to encoded text that the computer can easily understand. An image containing text is scanned and analyzed in order to identify the characters in it. Upon identification, the character is converted to machine-encoded text.

We can now create a simple function that takes an image and returns the text detected in the image. The function is quite straightforward, in the first 5 lines we import image from the Pillow library and our PyTesseract library.

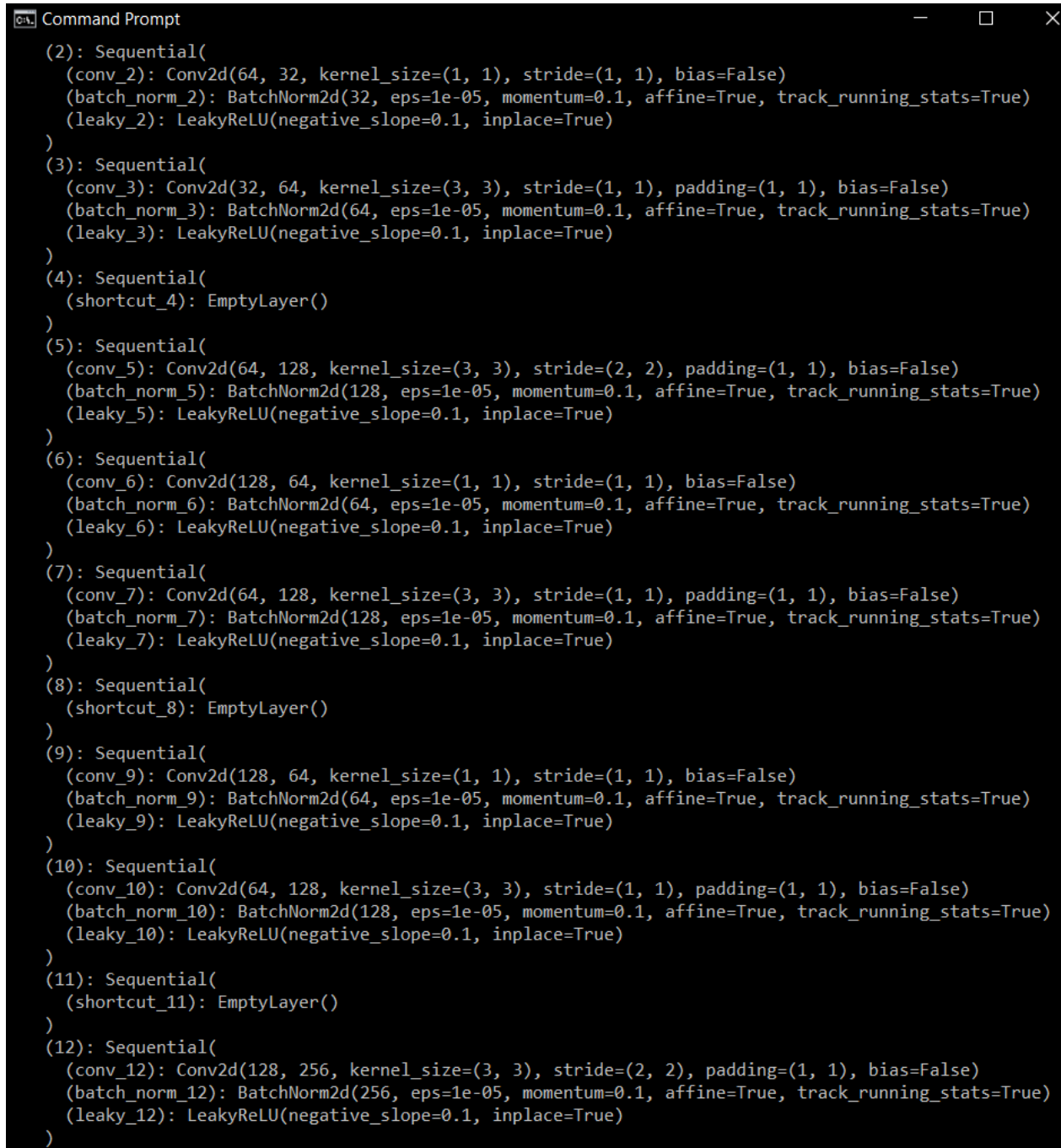
To render the image or the Video for the input. We check whether the user has really uploaded a file and use the function `allowed.file()` to check if the file is of an acceptable type. Upon verifying that the image is of the required type, we then pass it to the character recognition script we created earlier. The function detects the text in the image and returns it. Finally, as a response to the image upload, we render the detected text alongside the image for the user to see the results.

OCR helps the user in reading of signboards and other text. We have used libraries such as Tesseract, Imutil, Videostream, Pytesseract for the implementation.

## Chapter 5

### RESULTS & DISCUSSION

#### 5.1 Output of Object Detection Module:



```
Command Prompt
(2): Sequential(
  (conv_2): Conv2d(64, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (batch_norm_2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (leaky_2): LeakyReLU(negative_slope=0.1, inplace=True)
)
(3): Sequential(
  (conv_3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (batch_norm_3): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (leaky_3): LeakyReLU(negative_slope=0.1, inplace=True)
)
(4): Sequential(
  (shortcut_4): EmptyLayer()
)
(5): Sequential(
  (conv_5): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
  (batch_norm_5): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (leaky_5): LeakyReLU(negative_slope=0.1, inplace=True)
)
(6): Sequential(
  (conv_6): Conv2d(128, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (batch_norm_6): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (leaky_6): LeakyReLU(negative_slope=0.1, inplace=True)
)
(7): Sequential(
  (conv_7): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (batch_norm_7): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (leaky_7): LeakyReLU(negative_slope=0.1, inplace=True)
)
(8): Sequential(
  (shortcut_8): EmptyLayer()
)
(9): Sequential(
  (conv_9): Conv2d(128, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (batch_norm_9): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (leaky_9): LeakyReLU(negative_slope=0.1, inplace=True)
)
(10): Sequential(
  (conv_10): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (batch_norm_10): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (leaky_10): LeakyReLU(negative_slope=0.1, inplace=True)
)
(11): Sequential(
  (shortcut_11): EmptyLayer()
)
(12): Sequential(
  (conv_12): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
  (batch_norm_12): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (leaky_12): LeakyReLU(negative_slope=0.1, inplace=True)
)
```

Fig.5.1 Output showing properties of Neural Network Layers - i

```

Command Prompt
)
(98): Sequential(
  (route_98): EmptyLayer()
)
(99): Sequential(
  (conv_99): Conv2d(384, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (batch_norm_99): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (leaky_99): LeakyReLU(negative_slope=0.1, inplace=True)
)
(100): Sequential(
  (conv_100): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (batch_norm_100): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

  (leaky_100): LeakyReLU(negative_slope=0.1, inplace=True)
)
(101): Sequential(
  (conv_101): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (batch_norm_101): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

  (leaky_101): LeakyReLU(negative_slope=0.1, inplace=True)
)
(102): Sequential(
  (conv_102): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (batch_norm_102): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

  (leaky_102): LeakyReLU(negative_slope=0.1, inplace=True)
)
(103): Sequential(
  (conv_103): Conv2d(256, 128, kernel_size=(1, 1), stride=(1, 1), bias=False)
  (batch_norm_103): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

  (leaky_103): LeakyReLU(negative_slope=0.1, inplace=True)
)
(104): Sequential(
  (conv_104): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
  (batch_norm_104): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

  (leaky_104): LeakyReLU(negative_slope=0.1, inplace=True)
)
(105): Sequential(
  (conv_105): Conv2d(256, 255, kernel_size=(1, 1), stride=(1, 1))
)
(106): Sequential(
  (Detection_106): DetectionLayer()
)
)
)
C:\Users\Yash-PC\Desktop\YoloV3>

```

Fig.5.2 Output showing properties of Neural Network Layers - ii

The Convolutional Network of YOLOV3 consists of 106 layers. Each layer has its properties, type, inputs, activations and properties different that other layers. These network is used for predicting the objects present in the frame, and for making error reduction in the network in order to increase the precision.

Various layers present in the network are:

- 1) **Convolutional Layer:** The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), stride, bias, activation function, batch normalization. By using these parameters convolutional layers convolve the input and pass its result to the next layer. The input is a tensor with shape (number of images) x (image height) x (image width) x (image depth). Then after passing through a convolutional layer, the image becomes abstracted to a feature map, with shape (number of images) x (feature map height) x (feature map width) x (feature map channels).

**Parameters:**

- **Activation Function:** An activation function is a very important feature of an artificial neural network; they basically decide whether the neuron should be activated or not. In artificial neural networks, the activation function defines the output of that node given an input or set of inputs.
  - **Batch Normalization:** Batch normalization is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch. This has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks.
  - **Kernel Size:** It determines the size of the filter which convolves over the input tensor.
- 2) **Shortcut Layer:** A shortcut layer is a skip connection, akin to the one used in ResNet. The from parameter of the layer represents the output of the shortcut layer is obtained by adding feature maps from the previous and the nth layer backwards from the shortcut layer.
  - 3) **Route Layer:** It has an attribute layers which can have either one, or two values. When layers attribute has only one value, it outputs the feature maps of the layer indexed by the value. For example, if it is -4, so the layer will output feature map from the 4th layer backwards from the Route layer.
  - 4) **Upsample Layer:** Upsamples the feature map in the previous layer by a factor of stride using bilinear upsampling.
  - 5) **YOLO Layer:** YOLO layer corresponds to the Detection layer. The anchors describe 9 anchors, but only the anchors which are indexed by attributes of the mask tag are used. Here, the value of mask is 0,1,2, which means the first, second and third anchors are used. This make sense since each cell of the detection layer predicts 3 boxes. In total, we have detection layers at 3 scales, making up for a total of 9 anchors.

**Anchors:** Anchor boxes are a set of predefined bounding boxes of a certain height and width. These boxes are defined to capture the scale and aspect ratio of specific object classes you want to detect and are typically chosen based on object sizes in your training datasets.

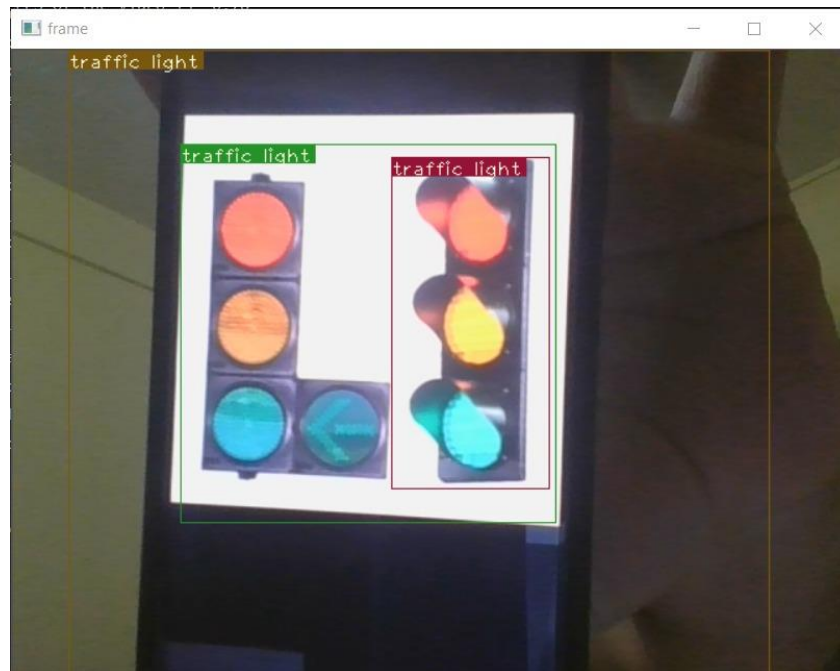


Fig 5.3 Object Detection for Traffic Light

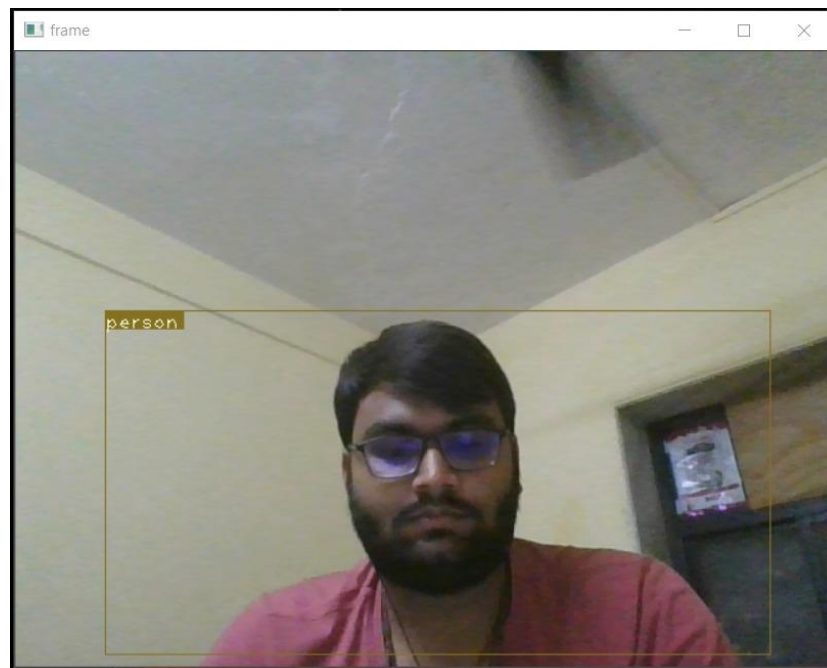
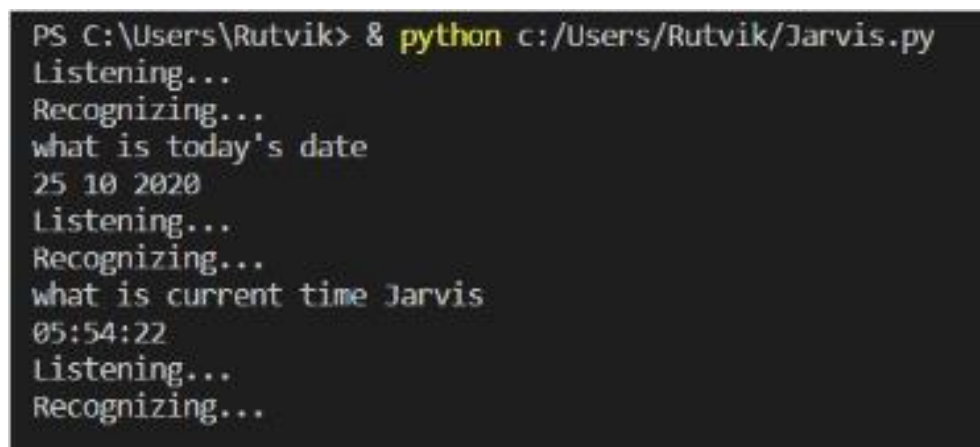


Fig 5.4 Object Detection for Person

## 5.2 Output of Voice Assistant Module

Voice Assistant performs different functions such as greeting, speech recognition, Wikipedia search, determining current time and date and sending email. We have imported several libraries in python for implementing voice assistant module, which are:

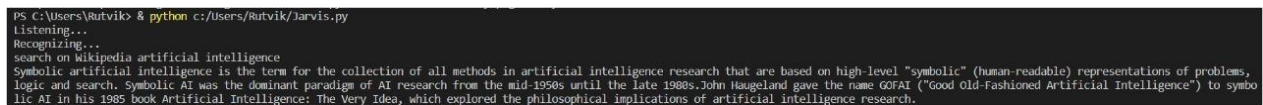
- 1) **Speech recognition module:** It also recognizes user's speech and converts it into text. This will allow us give commands to voice assistant. By using PYTTX3 module we are able to convert text to speech.
- 2) **Date Time Module:** It helps in determining current date and time.



```
PS C:\Users\Rutvik> & python c:/Users/Rutvik/Jarvis.py
Listening...
Recognizing...
what is today's date
25 10 2020
Listening...
Recognizing...
what is current time Jarvis
05:54:22
Listening...
Recognizing...
```

Fig.5.3 Output of current date and time using DateTime Module

- 3) **Wikipedia Search Module:** We can look up for information on Wikipedia.



```
PS C:\Users\Rutvik> & python c:/Users/Rutvik/Jarvis.py
Listening...
Recognizing...
search on wikipedia artificial intelligence
Symbolic artificial intelligence is the term for the collection of all methods in artificial intelligence research that are based on high-level "symbolic" (human-readable) representations of problems, logic and search. Symbolic AI was the dominant paradigm of AI research from the mid-1950s until the late 1980s. John Haugeland gave the name GOFAI ("Good Old-Fashioned Artificial Intelligence") to symbolic AI in his 1985 book Artificial Intelligence: The Very Idea, which explored the philosophical implications of artificial intelligence research.
```

Fig.5.4 Output of Wikipedia search using Wikipedia Module

- 4) **Send Email:** The user is able to store names of particular user along with their emails just like contact feature on mobile phones and by using that email we are able to send mail by using SMTP mail module.

```

PS C:\Users\Rutvik> & python c:/Users/Rutvik/Jarvis.py
Listening...
Recognizing...
send email
Listening...
Recognizing...
we are team Shiva
Listening...
Recognizing...
Aakash
Email sent

```

Fig.5.5 Output of sending email using SMTPLIB-i

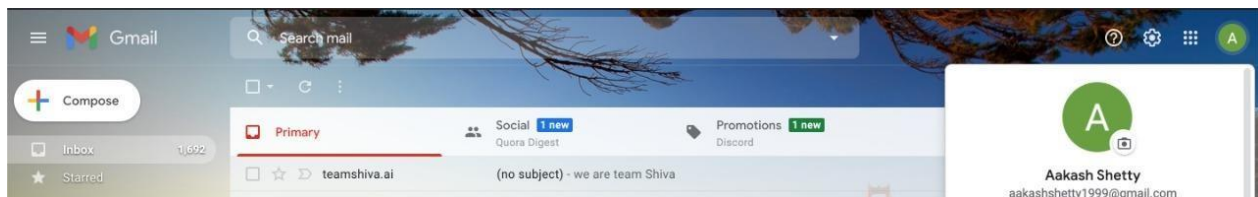


Fig.5.6 Output of sending email using SMTPLIB-ii

### 5.3 Output of OCR Module:



Fig 5.9 Output of OCR



## Chapter 6

### CONCLUSION & FUTURE SCOPE

According to the WHO report on Visual Impairments of 2010, there are approximately 285 million people, 39 million blind and 246 million having low vision in the year 2010. These numbers are only going to grow in the coming decades as the population increases. Our device sorts to help these visually impaired individuals to look at the world in a different way: through ears. The WHO report also states that 65% of the visually impaired and 82% of all blinds are above the age of 50. So, we wanted to come up with a device that reduces the user handling complexities by building it in such a way that could help all ages with no screen and tedious technical interactions. The only input our device will need is voice. With each year the number of people being affected from vision related difficulties keep growing. A growing ageing population is behind the rising numbers. According to the surveys the figure is expected to be more than 550 million by the year 2050 [9]. Now, many of the causes that lead to visual impairment (major or minor) can be avoided given good medical facilities and proper medical care. But those who fail to receive the necessary treatment, which is very common in third world countries, have to live the rest of their life depending on others for help which may or may not be available at times. The device we expect to build will make the impaired more independent and this aid will also be made at a low cost. With our device we will be one step closer to making the blinds and visually impaired navigate and know our world better.

We can conclude that YOLOv3 is a good option to implement Object Detection and Classification because of its high FPS and accuracy. MSCOCO dataset will be used as it is pre-trained and has large number of images for each class. Voice Assistant is coded in Python as it has many libraries which can be imported for easier implementation.

## Acknowledgement

We would like to thank our mentor Prof. Ishani Saha for great guidance and immense support.

## References

1. Pranav Adarsh, Pratibha Rathi, and Manoj Kumar. Yolo v3-tiny: Object detection and recognition using one stage improved model. In 2020 6<sup>th</sup> International Conference on Advanced Computing and Communication Systems (ICACCS), pages 687–694. IEEE, 2020.
2. Olga Barinova, Victor Lempitsky, and Pushmeet Kohli. On detection of multiple object instances using hough transforms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1773–1784, 2012.
3. Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.
4. C Dong, CL Chen, K He, et al. Learning a deep convolution network for image super-resolution [m]. *computer vision-eccv 2014*, 2014.
5. Juergen Gall, Nima Razavi, and Luc Van Gool. An introduction to random forests for multi-class object detection. In *Outdoor and largescale real-world scene analysis*, pages 243–263. Springer, 2012.
6. Dweepna Garg, Parth Goel, Sharnil Pandya, Amit Ganatra, and Ketan Kotecha. A deep learning approach for face detection using yolo. In 2018 IEEE Punecon, pages 1–4. IEEE, 2002.
7. R Girshick, J Donahue, T Darrell, and UC Berkeley. Malik., j.(2014). rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014.
8. Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.
9. Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.
10. J Han, D Zhang, G Cheng, N Liu, and D Xu. Advanced deep-learning techniques for salient and category-specific object detection: a survey. *ieee signal process mag* 35 (1): 84–100, 2017.
11. Zhanchao Huang, Jianlin Wang, Xuesong Fu, Tao Yu, Yongqi Guo, and Rutong Wang. Dc-spp-yolo: Dense connection and spatial pyramid pooling based yolo for object detection. *Information Sciences*, 2020.
12. Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, and Rong Qu. A survey of deep learning-based object detection. *IEEE Access*, 7:128837–128868, 2019.
13. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
14. Ashwani Kumar, SS Sai Satyanarayana Reddy, and Vivek Kulkarni. An object detection technique for blind people in real-time using deep neural network. In 2019 Fifth International Conference on Image Information Processing (ICIIP), pages 292–297. IEEE, 2019.
15. Yu Liu, Hongyang Li, Junjie Yan, Fangyin Wei, Xiaogang Wang, and Xiaoou Tang. Recurrent scale approximation for object detection in cnn. In *Proceedings of the IEEE*

International Conference on Computer Vision, pages 571–579, 2017.

16. Junyan Lu, Chi Ma, Li Li, Xiaoyan Xing, Yong Zhang, Zhigang Wang, and Jiuwei Xu. A vehicle detection method for aerial image based on yolo. *Journal of Computer and Communications*, 6(11):98–107, 2018.
17. C. Ning, Huajun Zhou, Yan Song, and J. Tang. Inception single shot multibox detector for object detection. *2017 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, pages 549–554, 2017.
18. Ajeet Ram Pathak, Manjusha Pandey, and Siddharth Rautaray. Application of deep learning for object detection. *Procedia computer science*, 132:1706–1717, 2018.
19. Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.
20. Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.
21. Joseph Redmon and Ali Farhadi. Yolo v3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.
22. Yun Ren, Changren Zhu, and Shunping Xiao. Object detection based on fast/faster rcnn employing fully convolutional architectures. *Mathematical Problems in Engineering*, 2018, 2018.
23. Seonkyeong Seong, Jeongheon Song, Donghyeon Yoon, Jiyoung Kim, and Jaewan Choi. Determination of vehicle trajectory through optimization of vehicle bounding boxes using a convolutional neural network. *Sensors*, 19(19):4263, 2019.
24. Jyothi Shetty and Pawan S Jogi. Study on different region-based object detection models applied to live video stream and images using deep learning. In *International Conference on ISMAC in Computational Vision and Bio-Engineering*, pages 51–60. Springer, 2018.
25. Qiwei Wang, Shusheng Bi, Minglei Sun, Yuliang Wang, Di Wang, and Shaobao Yang. Deep learning approach to peripheral leukocyte recognition. *Plos one*, 14(6):e0218808, 2019.
26. Xiaohan Yi, Liangcai Gao, Yuan Liao, Xiaode Zhang, Runtao Liu, and Zhuoren Jiang. Cnn based page object detection in document images. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 230–235. IEEE, 2017.
27. Yiqing Zhang, Jun Chu, Lu Leng, and Jun Miao. Mask-refined r-cnn: A network for refining object details in instance segmentation. *Sensors*, 20(4):1010, 2020.
28. Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232, 2019.
29. Arora, A., Grover, A., Chugh, R., & Reka, S. S. (2019). Real time multi object detection for blind using single shot multibox detector. *Wireless Personal Communications*, 107(1), 651-661.
30. Zhao, Z. Q., Zheng, P., Xu, S. T., & Wu, X. (2019). Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11), 3212-3232.
31. Srisuk, S., Suwannapong, C., Kitisriworapan, S., Kaewsong, A., & Ongkittikul, S. (2019, March). Performance Evaluation of Real-Time Object Detection Algorithms. In *2019 7th International Electrical Engineering Congress (iEECON)* (pp. 1-4). IEEE.

