

Object Detection with Voice Feedback

Submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering/Technology degree in Computer Science and Engineering

By

Sneha Gupta (38110543), and
Suchismita Chakraborti (38110565)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade “A” by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

MAY - 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Sneha Gupta (Reg no.: 38110543)**, **Suchismita Chakraborti (Reg no.: 38110565)** who carried out the project entitled **"Object Detection with Voice Feedback"** under my/our supervision from January 2022 to March 2022.

Internal Guide

Ms. R. Yogitha, M.E., Ph. D.

External Guide (if applicable)

Head of the Department

DR. S. VIGNESHWARI, M.E., PH.D.

DR. LAKSHMANAN L, M.E., PH.D.

Submitted for Viva Voce Examination held on _____

Internal Examiner
G. KALAIARASI

External Examiner
NIRMALA RANI

DECLARATION

I, **Suchismita Chakraborti** and **Sneha Gupta** hereby declare that the Project Report entitled “**Object Detection with Voice Feedback**” done by me under the guidance of **R. Yogitha, M.E., Ph.D.** at **Sathyabama Institute of Science and Technology, Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600119** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering.

DATE:

PLACE:

SIGNATURE OF CANDIDATE:

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the Board of Management of SATHYABAMA for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to Dr. T. Sasikala, M.E., Ph.D. Dean, School of Computing, Dr. S. Vigneshwari, M.E., Ph.D. and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my project guide Dr. R. Yogitha M.E., Ph.D., for her valuable guidance, suggestions, and constant encouragement that paved the way for the successful completion of my project work.

I wish to express my thanks to all teaching and Non-Teaching staff members of the Department of Computer Science and Engineering who were helpful in many ways to complete the project.

ABSTRACT

Vision loss is a serious factor that can affect one's course of life and quality of life, making them barred from independence, easy mobility, unnecessary tripping over objects, etc. As per World Health Organization's most recent studies, there are around 2.2 billion people on the planet who are visually impaired, and a major proportion of the population is over the age of 50 years. Population growth and aging are expected to increase the risk that more people acquire vision impairment. However, visual impairment in children can happen due to various factors.

With the advent of technology, a huge part of their day-to-day obstacles can be resolved if not eradicated. A handful amount of devices and facilities have been invented and designed for aiding the visually impaired. The main catch of which is to address the cost-effectiveness and accuracy ratio of the same.

The primary cornerstone of this project is about how a simple Object Detection model, done with the help of some python libraries and APIs, shall prove to be helpful for visually impaired people, and help with those subtle challenges. In our proposed methodology, to overcome the aforementioned drawbacks, we shall implement a Machine Learning Model whose key purpose is to detect the location of the object surrounding the person and provide voice feedback using Google Text-to-Speech API. To train the model around various aspects of the object, the MS-COCO dataset is being used.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NUMBER
	ABSTRACT	(i)
	LIST OF FIGURES	(ii)
	LIST OF ABBREVIATIONS	(iii)
1	INTRODUCTION	10
	1.1 BRIEF	10
	1.2 ABOUT THE PROJECT	11
	1.3 PROJECT OUTLINE	12
2	LITERATURE SURVEY	13
	2.1 LITERATURE REVIEW	13
3	AIM AND SCOPE OF THE PRESENT INVESTIGATION	15
	3.1 PROBLEM STATEMENT	15
	3.2 OBJECTIVE	15
	3.3 SCOPE AND LIMITATION	16
4	EXPERIMENTAL OR MATERIAL AND METHODS, ALGORITHM USED	18
	4.1 DATASET AND PACKAGES	18
	4.2 ALGORITHM	20
5	RESULT, DISCUSSION, PERFORMANCE ANALYSIS	30
	5.1 ANALYSIS	30
	5.2 HARDWARE AND SOFTWARE REQUIREMENTS	31
	5.3 TESTING	31
	5.4 FUTURE ENHANCEMENTS	32

6	SUMMARY AND CONCLUSION	33
	APPENDIX	34
	SOURCE CODE	
	SCREENSHOTS	
	REFERENCES	40

LIST OF FIGURES

FIGURE NUMBER	FIGURE NAME	PAGE NUMBER
1	ALGORITHM	
	4.1 YOLOV3 COMPARISON	20
	4.2 INTERSECTION OVER UNION VISUALIZATION	22
	4.3 DARKNET-52 ARCHITECTURE	26
	4.4 MULTI-SCALE FEATURE EXTRACTOR	27
	4.5 COMPLETE NETWORK ARCHITECTURE	28
2	SCREENSHOTS	
	B.1 OUTPUT IMAGE SCREENSHOT FROM OUTPUT VIDEO	39
	B.2 OUTPUT AUDIO SCREENSHOT	39

LIST OF ABBREVIATIONS

SERIAL NUMBER	ABBREVIATION	EXPLICATION
1	YOLO	You Only Look Once
2	MS COCO	Microsoft Common Objects in Context
3	API	Application Programming Interface
4	OPENCV	Open Computer Vision
5	GTTS	Google text to speech
6	NMS	Non Max Suppression
7	IEEE	Institute of Electrical and Electronics Engineering
8	OCR	Optical Character Recognition
9	IC-APR 9600	Integrated Circuit APR 9600
10	R-CNN	Region based Convulated Neural Network
11	NUMPY	Numerical Python
12	GPU	Graphics Processing Unit
13	SSD	Single Stage Detector

CHAPTER 1

INTRODUCTION

1.1 BRIEF

The fourth Industrial revolution that is also known as the technological revolution has almost gasped our industry. The foremost technology that is turning this revolution into reality is Artificial Intelligence, either it is in the form of Machine Learning or Deep Learning. There are many areas in the Machine Learning domain such as Speech Recognition, computer vision, and natural language processing. With the humongous availability of data around the world, it has become comparatively easier to train the algorithm, manipulate it and use it accordingly.

In this project, our prime concept is to simplify the lives of people affected visually by translating the perceptible surroundings around them into audio comments such that they can understand what is going on around them. Visual impairment is a serious cause that segregates people from the basic way of survival in times like these, making them go devoid of various opportunities and general chores that people don't even pay proper heed to. However, with the advent of technology and the technological revolution, the life of visually impaired people can be made comparatively smoother, helping them experience their surroundings in the most optimal process. Thus, for our proposed concept and to reach our required objective, we shall be using the aforementioned domain in ways that shall be explained hereunder.

For the same purpose, we use various Machine Learning libraries, of which OpenCV is a prime one. Open Source Computer Vision Library, generally termed as OpenCV is an open-source library that deals with image manipulation and various instantaneous operations. In this paper, we will be seeing the use of OpenCV for detecting objects. Other than that Google Speech API and a suitable training algorithm are used to get the proper outcome from the model.

1.2 ABOUT THE PROJECT

In this project, our goal is to build a prototype of a model that easily sees the objects around us and translates them in a way that visually impaired people can perceive it with zero to no difficulty. We use areas of machine learning such as speech recognition and computer vision to conduct this project. The dataset used to administer this project is the MS COCO dataset.

The MS COCO dataset is comprehensive object detection, segmentation, and labeling dataset published by Microsoft. COCO stands for Common Objects in Context, as the image data set was created with the aim of advancing image recognition, visual datasets for computer vision, mostly state-of-the-art neural networks. The COCO dataset offers a variety of features including object segmentation with detailed instance annotation, in-context detection, superpixel segmentation to name a few. Out of a total of 300,000 images, 200,000 are tagged. In addition, COCO also provides 80+ object categories called COCO Classes, 91 object categories called COCO Things, and provides 5 captions per image. Also, for the purpose of pose estimation, 250,000 people with 17 different pre-trained key points are generally used.

The project can be segmented into various steps for a better understanding of the work. The partial objective of the project is to detect objects around us and to identify them using the YOLOv3 algorithm that supports Darknet architecture, thus giving a wide spherical view of the object detection and recognition process as a whole. Furthermore, the latter half of the project aims on translating the recognized objects into speech through the Google-Text-to-Speech API of the python library. Combining both of these gives us the primary objective of the prototype.

Thereby, the initial designs of the project prototype are made, thus addressing the required and mentioned objectives.

1.3 PROJECT OUTLINE

This project is developed and structured using various python libraries for working with various domains of Machine Learning, such as speech recognition, computer vision, natural language processing. The whole project is made in Jupyter Notebook, which is a very usable platform for projects like such.

The project can be implemented after the successful implementation of OpenCV 3.4+. Since OpenCV 4 is still in beta right now and the official release has not been initiated yet, it is safer to use the version 3 of the OpenCV library. Apart from that, we need to install the YOLOv3 training dataset and MS COCO dataset to start our project implementation.

The whole project can be visualised in tree command of the terminal as 4 directories and 19 files initially. The 4 directories can be further enhanced as

- image** : The path to the input image. We shall be detecting objects in this image using YOLO.
- yolo** : This is the base path to the YOLO directory. The scripts will load the required YOLO files in order to perform object detection on the image.
- confidence** : Minimum probability to filter weak detections is tracked in this directory. The default value is given as 0.5 and the value is open to experimentation.
- threshold** : This is our non-maxima suppression (NMS) threshold with a default value of 0.3.

A variety of object detection techniques is used using the YOLOv3 algorithm which shall be explained further and the detected output is translated to audio for audio feedback. Implementing the same on a video based input meets the objective output that we have set to plan out.

CHAPTER 2

LITERATURE SURVEY

2.1 LITERATURE REVIEW

As a part of research for building our model, a literature review of a handful of IEEE published papers were made. We shall briefly be discussing the review hereunder with appropriate required details.

2.1.1 Real Time Object Detection with Audio Feedback using Yolo vs. Yolo_v3:

The first paper is titled “Real Time Object Detection with Audio Feedback using Yolo vs. Yolo_v3” and was published in the year 2021. This paper uses algorithms and techniques like the OpenCV library, Yolo, Yolo v3. The performance recorded in this paper indicates that it works better for smaller objects with future works mentioned as the expansion of the research on self explored dataset [1]

2.1.2 Reader and Object Detector for Blind:

The next paper is titled “Reader and Object Detector for Blind” which was published in the year 2020. This paper uses algorithms and techniques such as Raspberry pi, OCR, tesseract, and tensorflow for carrying out the project. Text reading and object detection was successful but not for smaller than 16 font size is what was recorded in the performance of the paper. As the objective for future works, making it available for multi languages is recorded as of now [2].

2.1.3 Obstacle Detection for Visually Impaired Patients:

The paper that was studied next is titled “Obstacle Detection for Visually Impaired Patients” which was published in the year 2014. The techniques and algorithms used in this paper are stereoscopic sonar system, sound buzzers, voice IC-APR 9600. Wearable optical detection system is provided that provides full body vibration effect on obstacle detection. However, the device has a very limited range when compared to its own size and is also found difficult for users to comprehend the guidance signals in time [3].

2.1.4 Voice Based Smart Assistive Device for Visually Challenged:

The paper that was studied next is titled “Voice Based Smart Assistive Device for Visually Challenged” and was published in the year 2020. The Raspberry Pi, Deep Learning, conversational AI, speech recognition, Assistive Technology, and algorithms and methodologies were described in the article. After being trained on only 50 photos of each object, the model has an accuracy of 83 percent and detects campus objects that are commonly available. However, because it was trained on 8000 photos from the Flickr 8k dataset, the accuracy drops as the image complexity grows.[4].

2.1.5 A Wearable Assistive Technology for the Visually Impaired with Door Knob Detection and Real-Time Feedback for Hand-to-Handle Manipulation:

The next paper is titled “A Wearable Assistive Technology for the Visually Impaired with Door Knob Detection and Real-Time Feedback for Hand-to-Handle Manipulation” and was published in the year 2017. Algorithms and techniques such as YOLOv2, Deep Learning, Neural Network were used. The performance of the device is increased to folds if the hand detection is stable. The biggest difficulty, however, is the consistency of the hand detection performance. More images will be added to the database in the future, and the door knob identification feature will be extended to more general door handles. [5].

2.1.6 VISION- Wearable Speech Based Feedback System for the Visually Impaired using Computer Vision:

The paper is titled as “VISION- Wearable Speech Based Feedback System for the Visually Impaired using Computer Vision”, published in 2020. It is a wearable device based on Raspberry pi, gTTs and YOLO. The text will be read out in English and at a slow speed that is saved as an mp3 file and future work is mentioned as location navigation that works in low-light conditions while remaining cost-effective.[6]

2.1.7 YOLO-compact:An Efficient YOLO Network for Single Category Real-time Object Detection:

This IEEE paper named, “YOLO-compact:An Efficient YOLO Network for Single Category Real-time Object Detection”, published in 2020, is an efficient way for a location navigation that works in low-light conditions while remaining cost-effective.. The model would be more precise if the depth, width and precision is improved. [7].

2.1.8 CPU based YOLO: A Real Time Object Detection Algorithm:

The next 2020 published paper having title “CPU based YOLO: A Real Time Object Detection Algorithm” is based on Faster R-CNN, YOLO, R-CNN, Fast R-CNN, SSD, Mask R-CNN, R-FCN, OpenCV and RetinaNet. The Model discovers objects from video at a pace of “10.12–16.29 frames per second” on many non-GPU platforms, with an accuracy of 80–99 percent. mAP of 31.05 percent is achieved by CPU Based YOLO with aforementioned future work as increment of FPS and mAP by optimizing the model.[8].

2.1.9 A Novel YOLO-based Real-time People Counting Approach:

A 2017 paper titled “A Novel YOLO-based Real-time People Counting Approach” is based on YOLO-PC and CNN. IT provides us an automatic way to count people in a huge crowd and the accuracy states that it works perfectly fine for a huge crowd and for a single person as well. Future Work stated in the paper is to add abnormal behavior detection during counting and children counting.[9]

2.1.10 Edge detection based boundary box construction algorithm for improving the precision of object detection in YOLOv3:

The last paper that we have reviewed is “Edge detection based boundary box construction algorithm for improving the precision of object detection in YOLOv3” published in 2017. YOLOv3, Edge Detection, YOLO, YOLO9000, Boundary Box Prediction, and Object Detection are all used in this research. The intersection over union for the proposed algorithm and YOLO v3 is determined, and the proposed approach outperforms YOLO v3 in terms of boundary box accuracy. When there are sharp objects in the image or there is too much noise, the model becomes constrained.[10]

CHAPTER 3

AIM AND SCOPE OF THE PRESENT INVESTIGATION

3.1 PROBLEM STATEMENT

The user is the visually impaired people who shall be addressed by the proponent to develop this model. Common users who will get to access the model for their own benefit upon the supervision of the medical facility within their own comfortable accommodation.

The following are the things to be addressed by the proponents:

- Accuracy
- Reliability
- Usability
- Efficiency
- User Friendliness

3.2 OBJECTIVE

The prime objective of this project is to understand the underlying factors of visual impairment and to help people suffering from visual impairment know what they are surrounded by in a very friendly cost.

Adding on to that, the project also aims at running seamlessly without any system barrier and high usability. The accuracy of the built model is one to look out for, since it shall be dealing with detecting objects and recognizing them in real time.

3.3 SCOPE AND LIMITATION

3.3.1 SCOPE

- The built model shall be carrying out the idea and concept of healthcare machine learning by accurately detecting and recognizing objects around the user.
- The model shall also be providing the best suited and most accurate ML classification algorithm for achieving the required objective as mentioned in this project.
- The model is most accurate when it comes to the given objective, thereby resulting in most desirable and precise predictions to be made.
- The model is easy to use and understandable by regular people, giving them the authority of high accessibility and reliability.

3.3.2 LIMITATION

- The foremost drawback of the model is that although it is highly accurate, it can never be 100% accurate.
- This is a prototype of the idea that we have proposed and thus we are still to use hardware that must be required for the proper execution of this project.
- Although it is highly reliable and gives the accurate algorithm to follow, the accuracy of the algorithms used might differ depending upon the processing unit that has been used.

CHAPTER 4

EXPERIMENTAL OR MATERIALS AND METHODS, ALGORITHMS USED

4.1. DATASET AND PACKAGES

4.1.1 DATASET

COCO Dataset stands for Common Objects In Context. It is a pre defined dataset by Microsoft. The COCO dataset is a collection of demanding, high-quality datasets for computer vision, mostly using state-of-the-art neural networks. This name is also used to refer to the format in which the datasets are stored. It is an object detection, segmentation, and captioning dataset. It contains around 330k images in which more than 200k images are labelled, which makes it even more easier to recognize the class (category) of detected object. It has around 1.5 million object instances and 80 object categories. COCO annotations employ the JSON file format, which has a top value of dictionary (key-value pairs inside braces). It can also have nested dictionaries or lists (ordered collections of objects inside brackets), as shown below:

```
{  
  "info": {...},  
  "licenses": [...],  
  "images": [...],  
  "categories": [...],  
  "annotations": [...]  
}
```

Info Section: It contains metadata about the dataset like description, url, version etc .

Licenses section: It contains the links to the licenses for the images present in the dataset. All the license contains the id field which is used to recognize the license.

Image: It is the second most important dictionary of the dataset. It has the fields like licence, file_name, coco_url, height, width and date_captured.

Categories Section: It contains classes of the objects that may be detected on images.

Annotations Section: This is the most important section of the dataset, which contains information vital for each task for specific COCO dataset.

4.1.2 PACKAGES

The following packages has been used for building the model:

(i) **NumPy**: NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

(ii) **OpenCV**: Open Source Computer Vision. It is one of the most widely used tools for computer vision and image processing tasks. It is used in various applications such as face detection, video capturing, tracking moving objects, object disclosure. now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human.

(iii) **gTTs**: gTTS is a very easy to use tool which converts the text entered, into audio which can be saved as a mp3 file. The gTTS API supports several languages including English, Hindi, Tamil, French, German and many more. The speech can be delivered in any one of the two available audio speeds, fast or slow. However, as of the latest update, it is not possible to change the voice of the generated audio.

4.2 ALGORITHM

In order to build the model, we have cross validated various ML classification algorithms. However, the high accuracy rated algorithms that we have further used to fit the model shall be discussed henceforth.

4.2.1 YOLO v3:

The term 'You Only Look Once' is abbreviated as YOLO. This is an algorithm for detecting and recognising different items in a photograph (in real-time). Object detection in YOLO is done as a regression problem, and the identified photos' class probabilities are provided. Convolutional neural networks (CNN) are used in the YOLO method to recognise objects in real time. To detect objects, the approach just takes a single forward propagation through a neural network, as the name suggests. This indicates that a single algorithm run is used to forecast the entire image. The CNN is used to forecast multiple bounding boxes and class probabilities at the same time. It is an instantaneous object identification algorithm that has a COCO test-dev mAP of 57.9% while analyzing images at 30 frames per second rate. The main features of YOLOv3 lie in it being very fast and accurate, which can easily be traded off by simply customizing the size of the model, thereby requiring no retraining whatsoever.

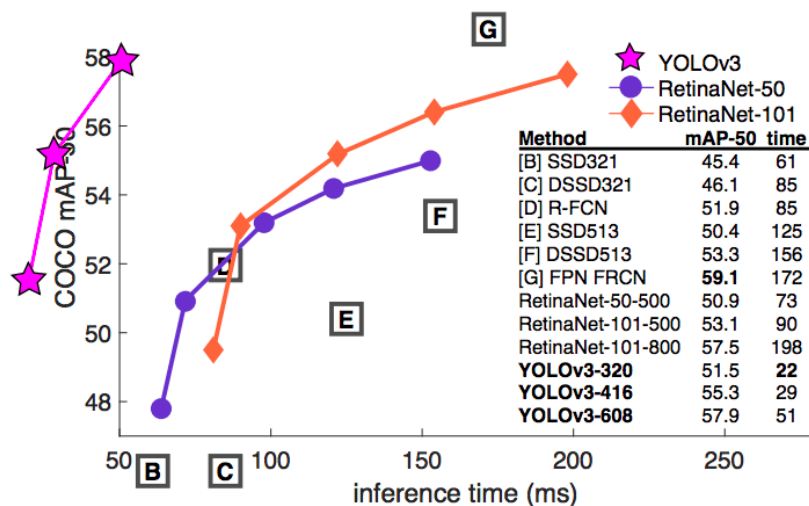


Fig 4.1: YOLOv3 comparison

4.2.1.1 Working of YOLO algorithm:

YOLO is regression-based. Initially it takes the video input and segments the video into 24 frames. Each frame is then divided into cells. Image classification and localization are applied on each grid. YOLO then predicts the bounding boxes and their corresponding class probabilities for objects.

To break down it into more simpler terms, the labelled data lets say is divided into 3x3 grids and there are total of 3 classes in which we want it to be classified. So, for each grid cell, label y will be defined as eight dimensional vector.

$$Y = pc, bx, by, bh, bw, c1, c2, c3$$

Here,

pc is the probability of whether the object is present in the grid or not.

bx, by, bh, bw specify the bounding box if there's an object, and $c1, c2, c3$ are the classes of the detected objects.

Bounding boxes i.e bx, by, bh and bw are calculated relative to the grid cell it is dealing with. bx and by are the x and y coordinates of the midpoint of the object with respect to this grid. bh is the ratio of the height of the bounding box to the height of the corresponding grid cell. bw is the ratio of the width of the bounding box to the width of the grid cell.

bx and by will always range between 0 and 1 as the midpoint will always lie within the grid. Whereas bh and bw can be more than 1 in case the dimensions of the bounding box are more than the dimension of the grid.

4.2.1.2. Intersection over Union and Non-Max Suppression

How can we tell if the anticipated bounding box is producing a good (or terrible) result? This is where the concept of Intersection over Union comes into play. It computes the intersection of the actual bounding box and the predicted bounding box over their union.

IoU, or Intersection over Union, will calculate the area of the intersection over union of these two boxes. That area will be:

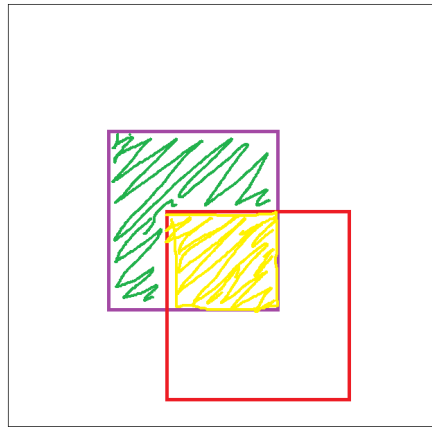


Fig 4.2: Intersection over Union visualization

$\text{IoU} = \text{Area of the intersection} / \text{Area of the union, i.e.,}$

$\text{IoU} = \text{Area of yellow box} / \text{Area of green box}$

If IoU is greater than 0.5, we can say that the prediction is good enough. 0.5 is an arbitrary threshold we have taken here, but it can be changed according to the specific problem. Intuitively, the more we increase the threshold, the better the predictions become.

There is one more technique that can improve the output of YOLO significantly – Non-Max Suppression. To avoid the multiple detection of object in the single frame, Non-Max Suppression technique is used. It first looks at the probabilities associated with each detection and takes the largest one. Then, it looks at all the other boxes in the image. The boxes which have high IoU with the current box are suppressed. After the boxes have been suppressed, it selects the next box from all the boxes with the highest probability, this process is repeated and all the boxes have either been selected or suppressed and the final bounding boxes are calculated.

4.2.1.3. Training and Testing :

The input for training our model will obviously be images and their corresponding y labels.

Consider the scenario where we are using a 3 X 3 grid with two anchors per grid, and there are 3 different object classes. So the corresponding y labels will have a shape of 3 X 3 X 16.

Now, suppose we use 5 anchor boxes (multiple bounding boxes in a single grid) per grid and the number of classes has been increased to 5. So the target will be $3 \times 3 \times 10 \times 5 = 3 \times 3 \times 50$. This is how the training process is done – taking an image of a particular shape and mapping it with a 3 X 3 X 16 target (this may change as per the grid size, number of anchor boxes and the number of classes).

The new image will be divided into the same number of grids which we have chosen during the training period. For each grid, the model will

predict an output of shape 3 X 3 X 16 (assuming this is the shape of the target during training time).

The 16 values in this prediction will be in the same format as that of the training label. The first 8 values will correspond to anchor box 1, where the first value will be the probability of an object in that grid. Values 2-5 will be the bounding box coordinates for that object, and the last three values will tell us which class the object belongs to. The next 8 values will be for anchor box 2 and in the same format, i.e., first the probability, then the bounding box coordinates, and finally the classes.

Finally, the Non-Max Suppression technique will be applied on the predicted boxes to obtain a single prediction per object.

4.2.2 DARKNET ARCHITECTURE

Yolo V3 is an improvement over the previous two YOLO versions where it is more robust but a little slower than its previous versions. This model features multi-scale detection, a stronger feature extraction network, and a few changes in the loss function. For understanding the network architecture on a high-level, let's divide the entire architecture into two major components: Feature Extractor and Feature Detector (Multi-scale Detector). The image is first given to the Feature extractor which extracts feature embeddings and then is passed on to the feature detector part of the network that spits out the processed image with bounding boxes around the detected classes.

4.2.2.1 Feature Extractor

Darknet-19 (a custom neural network architecture developed in C and CUDA) was utilised as a feature extractor in prior YOLO versions, with 19 layers as the name suggests. Darknet-19 now has a total of 30 layers thanks to YOLO v2, which adds 11 extra layers. However, because to the downsampling of the input image and the loss of fine-grained characteristics, the system had difficulty detecting small objects.

The feature extractor utilised in YOLO V3 was a combination of YOLO v2, Darknet-53 (an ImageNet-trained network), and Residual networks, which resulted in a better architecture (ResNet). The network is formed with consecutive 3x3 and 1x1 convolution layers followed by a skip connection, resulting in a total of 53 convolution layers (thus the name Darknet-53) (introduced by ResNet to help the activations propagate through deeper layers without gradient diminishing).

The darknet's 53 layers are piled on top of another 53 for the detection head, giving YOLO v3 a total of 106 layers of fully convolutional underlying architecture. As a result, it has a huge architecture, which makes it a little slower than YOLO v2, but improves accuracy at the same time.

	Type	Filters	Size	Output
	Convolutional	32	3×3	256×256
	Convolutional	64	$3 \times 3 / 2$	128×128
1x	Convolutional	32	1×1	128×128
	Convolutional	64	3×3	
	Residual			
	Residual			
	Convolutional	128	$3 \times 3 / 2$	64×64
2x	Convolutional	64	1×1	64×64
	Convolutional	128	3×3	
	Residual			
	Residual			
	Convolutional	256	$3 \times 3 / 2$	32×32
8x	Convolutional	128	1×1	32×32
	Convolutional	256	3×3	
	Residual			
	Residual			
	Convolutional	512	$3 \times 3 / 2$	16×16
8x	Convolutional	256	1×1	16×16
	Convolutional	512	3×3	
	Residual			
	Residual			
	Convolutional	1024	$3 \times 3 / 2$	8×8
4x	Convolutional	512	1×1	8×8
	Convolutional	1024	3×3	
	Residual			
	Residual			
	Avgpool		Global	
	Connected		1000	
	Softmax			

Fig 4.3: Darknet-53 architecture taken from YOLO:An Incremental Improvement

If the aim was to perform classification as in the ImageNet, then the Average pool layer, 1000 fully connected layers, and a SoftMax activation function would be added as shown in the image, but in our case, we would like to detect the classes along with the locations, so we would be appending a detection head to the extractor. The detection head is a multi-scale detection head hence, we would need to extract features at multiple scales as well.

4.2.2.2 Multi-Scale Detector

For visualizing how the multi-scale extractor would look like, I'm taking an example of a 416x416 image. A stride of a layer is defined as the ratio by which it downsamples the input, and hence the three scales in our case would be 52x52, 26x26, and 13x13 where 13x13 would be used for larger objects and 26x26 and 52x52 would be used for medium and smaller objects.

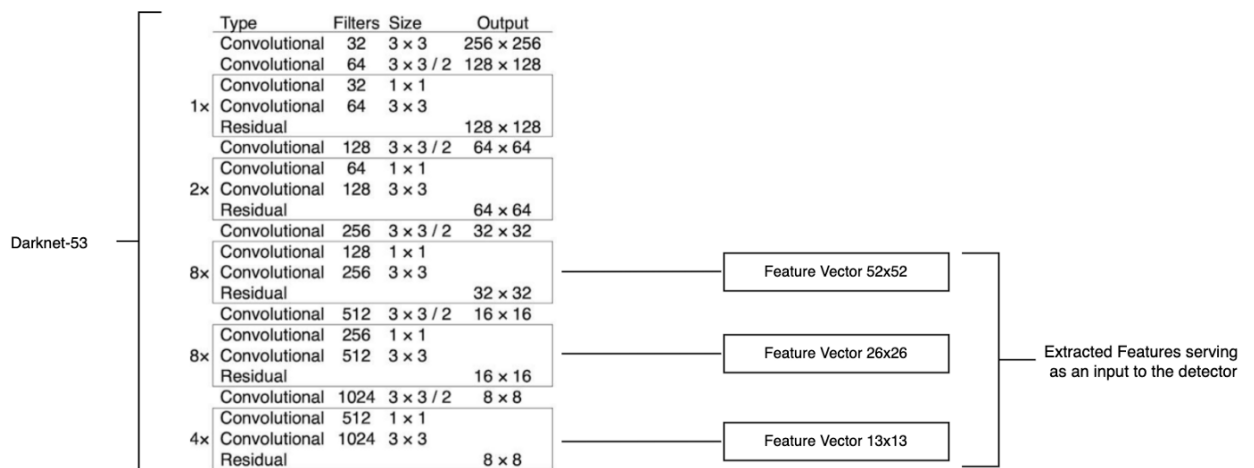


Fig 4.4: Multi scale Feature Extractor for 416x416 image

An important feature of the YOLO v3 model is its multi-scale detector, which means that the detection for an eventual output of a fully convolutional network is done by applying 1x1 detection kernels on feature maps of three different sizes at three different places. The shape of the kernel is $1 \times 1 \times (B^*(5+C))$.

The complete network architecture can be explained as in below.

4.2.2.3 Complete Network Architecture

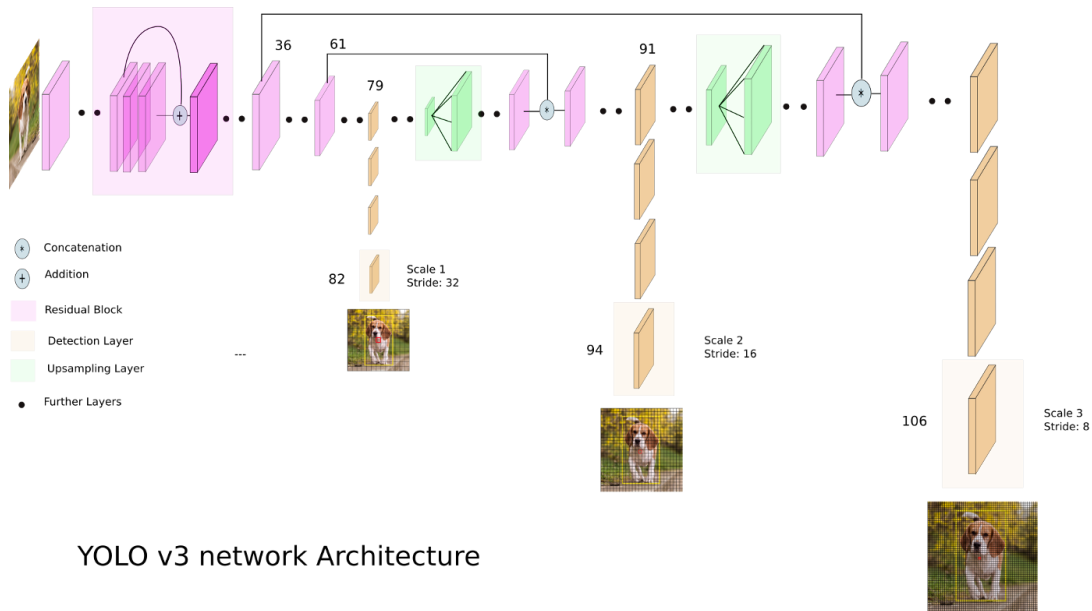


Fig 4.5: Complete architecture of YOLO v3 combining both the extractor and the detector

The three scales where the detections are produced are at the 82nd layer, 94th layer, and 106th layer, as seen in the above image of a 416x416 image.

For the first detection, the first 81 layers are downsampled until the 81st layer has a stride of 32 (a layer's stride is defined as the ratio by which it downsamples the input), resulting in a 13x13 feature map and a 1x1 kernel, resulting in a 13x13x255 detection 3D tensor.

For the second detection, convolutional layers are used from the 79th layer onwards before upsampling to 26x26 dimensions. This feature map is then depth concatenated with layer 61's feature map to create a new feature map, which is then fused with the 61st layer using 1x1 convolution layers. The second detection layer, with a 3D tensor of dimension 26x26x255, is located at the 94th layer.

The feature map of the 91st layer is subjected to convolution layers before being depth concatenated and fused with a feature map from the 36th layer for the final(third) detection layer, which follows the same process as the second detection layer. With a feature map of size 52x52x255, the final detection is made at the 106th layer.

The multi-scale detector is used to make certain that the small gadgets also are being detected not like in YOLO v2, in which there has been steady grievance concerning the same. Upsampled layers concatenated with the preceding layers grow to be retaining the fine-grained capabilities which assist in detecting small gadgets.

CHAPTER 5

RESULTS, DISCUSSION, PERFORMANCE ANALYSIS

5.1 ANALYSIS

This section encompasses the results obtained through the observations and analysis of the built model. Upon full execution of the model, it can be concluded that the prototype recognizes and translates objects around the user with much efficiency.

Hence, the YOLOv3 algorithm that has been used for the purpose of this prototype function is fast, compact, and easy to use. With a required time of 22ms in GPU based devices, the speed and accuracy of the algorithm can be justified with ease. To help increase the speed of deep learning-based object detectors, both Single Shot Detectors (SSDs) and YOLO use a one-stage detector strategy.

These algorithms treat object detection as a regression problem, taking a given input image and simultaneously learning bounding box coordinates and corresponding class label probabilities. In general, single-stage detectors tend to be less accurate than two-stage detectors but are significantly faster. YOLO is a great example of a single stage detector.

The model is conducted initially through OpenCV. Using OpenCV, the surrounding around the user can be evaluated. Upon receiving this input the algorithm is triggered and the input gets divided into frames and subsequent functions of the algorithm proceeds to detect objects around the user and thereby recognize those objects in real time alongwith audio feedback of the name of the objects.

Thus, the prototype is conducted and executed.

5.2 HARDWARE AND SOFTWARE REQUIREMENTS

5.2.1 HARDWARE

- 64-bit versions of Microsoft Windows 10, 8, 7 (SP1)
- 4 GB RAM minimum, 8 GB recommended
- 2 GB of available hard-disk space for installations
- 1024x768 display, recommended 1280x800; with 16-bit color and 512 MB of VRAM (1 GB recommended)
- Intel Pentium 4 or later; or AMD Ryzen 5000 series
- Internet with about 25 MB/s
- GPU enabled devices or CUDA API for GPU enabling

5.2.2 SOFTWARE

- Jupyter Notebook
- Numpy (pip install numpy)
- Speech Recognition
- Google Text-to-Speech API
- OpenCV 3.4+
- YOLO training dataset
- MS COCO dataset

5.3 TESTING

The final step, which is arguably one of the most important, is testing. It is of crucial importance to ascertain whether the built model does what it is supposed to do and fulfills the requirements set forth by the stakeholders of the system. Testing occurred through the development process and the findings helped shape development and the features of the model.

A few examples of inconsistencies that has now been rectified are:

- Not getting to read the dataset properly
- Difficulty in parsing arguments
- Extended time to run due to lack of GPU

5.4 FUTURE ENHANCEMENT

This model works very accurately for MS COCO dataset. However, for future work, this model shall be aimed at giving the same and more accurate result for self explored datasets and a many other customisable datasets.

A transforming development can be done by integrating face recognition techniques of Machine Learning in this project so as to help people suffering from visual impairment gain the authority of recognizing faces by the device itself with name.

Technology should never be a barrier. And hence, for future, a proper hardware that would integrate the same idea can be built either as a form of spectacles or lenses alongwith supportive earplugs for hearing audio translation. With Machine Learning and automation techniques, people can record what they see and playback whenever it is needed.

As the world becomes more and more advanced, lives of people have also become more advanced than ever before, and with ideas as has been discussed here, it is not only helpful, but also life saving actually. And that is basically the future vision upon the issue that has been addressed here, so as to fight visual impairment and bring a change for it without waiting for it to take over at any cost.

CHAPTER 6

SUMMARY AND CONCLUSION

6.1 CONCLUSION

In this project, we have hereby built a prototype of a model that can accurately and efficiently detect, recognize and give an audio feedback of objects around us with minimum ease.

Visual impairment is a serious condition that affects lives in a multitude of ways that we can only imagine. Day to day activities of life gets affected by the same. With this project, we thus have built an idea that can help people suffering from any of such vision related issues. The plight of these suffering people is beyond our measures of control but we can help them substitute vision with the advent of technology and empathy.

For building this project, all that has been learnt from various sources were made to utility and hence, it resulted in the successful implementation of the project. Although there are quite a few drastic future enhancements that have been suggested as aforementioned, the model that is right now shall be the strong foundation to that all, giving a right start that future enhancements would require.

Visual impairment is ugly and disruptive to normal leading of life in a variety of pace, and with the booming of technology, it can not only be cured, but also in the far future, eradicated from the face of earth.

APPENDIX

A. SOURCE CODE

```
# import the necessary packages
import numpy as np
import argparse
import time
import cv2
import os
import speech_recognition as sr
from gtts import gTTS

#construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,
                help="path to input image")
ap.add_argument("-y", "--yolo", required=True,
                help="base path to YOLO directory")
ap.add_argument("-c", "--confidence", type=float, default=0.5,
                help="minimum probability to filter weak detections")
ap.add_argument("-t", "--threshold", type=float, default=0.3,
                help="threshold when applying non-maxima suppression")
args = vars(ap.parse_args())

# load COCO class labels to YOLO model
labelsPath = os.path.sep.join([args["yolo"], "coco.names"])
LABELS = open(labelsPath).read().strip().split("\n")

#initialize a list of colors to represent each possible class label
np.random.seed(42)
COLORS = np.random.randint(0, 255, size=(len(LABELS), 3),
```

```
dtype="uint8")
```

```
# derive the paths to the YOLO weights and model configuration
```

```
weightsPath = os.path.sep.join([args["yolo"], "yolov3.weights"])
```

```
configPath = os.path.sep.join([args["yolo"], "yolov3.cfg"])
```

```
# load YOLO object detector trained on COCO dataset (80 classes)
```

```
print("[INFO] loading YOLO from disk...")
```

```
net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)
```

```
# load input image and grab the spatial dimensions
```

```
image = cv2.imread(args["image"])
```

```
(H, W) = image.shape[:2]
```

```
# determine only the output layer names that we need from YOLO
```

```
ln = net.getLayerNames()
```

```
ln = [ln[i[0] - 1] for i in net.getUnconnectedOutLayers()]
```

```
# construct a blob from the input image and then perform a forward pass  
of the YOLO object detector giving us the bounding box and associated  
probabilities
```

```
blob = cv2.dnn.blobFromImage(image, 1 / 255.0, (416, 416),
```

```
    swapRB=True, crop=False)
```

```
net.setInput(blob)
```

```
start = time.time()
```

```
layerOutputs = net.forward(ln)
```

```
end = time.time()
```

show timing information on YOLO

```
print("[INFO] YOLO took {:.6f} seconds".format(end - start))
```

initialize lists of detected bounding boxes, confidences, and class IDs, respectively

```
boxes = []
```

```
confidences = []
```

```
classIDs = []
```

```
ID = 0
```

loop over each of the layer outputs

for output in layerOutputs:

 for detection in output:

 # extract the class IDs and confidence of the current object detection

```
            scores = detection[5:]
```

```
            classID = np.argmax(scores)
```

```
            confidence = scores[classID]
```

 # filter out weak predictions by ensuring the detected probability is greater than the minimum probability

```
            if confidence > args["confidence"]:
```

 # scale the bounding box coordinates back relative to the size of the image, keeping in mind that YOLO actually returns the center (x, y) coordinates of the bounding box followed by the boxes width and height

```
                box = detection[0:4] * np.array([W, H, W, H])
```

```
                (centerX, centerY, width, height) = box.astype("int")
```

use the center (x, y) coordinate to derive the top and the left corner of the bounding box

```
x = int(centerX - (width / 2))
y = int(centerY - (height / 2))
```

update our list of bounding box coordinates, confidences, and class IDs

```
boxes.append([x, y, int(width), int(height)])
confidences.append(float(confidence))
classIDs.append(classID)
```

apply non-maxima suppression to suppress weak, overlapping bounding boxes

```
idxs = cv2.dnn.NMSBoxes(boxes, confidences, args["confidence"],
                        args["threshold"])
```

#ensure atleast one detection exists

```
if len(idxs) > 0:
```

```
    list1 = []
```

```
    for i in idxs.flatten():
```

extract bounding box coordinates

```
        (x, y) = (boxes[i][0], boxes[i][1])
```

```
        (w, h) = (boxes[i][2], boxes[i][3])
```

```
        centerx = round((2*x + w)/2)
```

```
        centery = round((2*y + h)/2)
```

```
        if centerX <= W/3:
```

```
            W_pos = "left "
```

```
        elif centerX <= (W/3 * 2):
```

```
            W_pos = "center "
```

```
        else:
```

```
            W_pos = "right "
```

```
if centerY <= H/3:
    H_pos = "top "
elif centerY <= (H/3 * 2):
    H_pos = "mid "
else:
    H_pos = "bottom "
list1.append(H_pos + W_pos + LABELS[classIDs[i]])
```

```
description = ', '.join(list1)
```

```
# convert text output into speech using the google text to speech api
```

```
myobj = gTTS(text=description, lang="en", slow=False)
```

```
myobj.save("object_detection.mp3")
```

B. SCREENSHOTS

Hereunder I attach the screenshots of the project's analysis and the model.

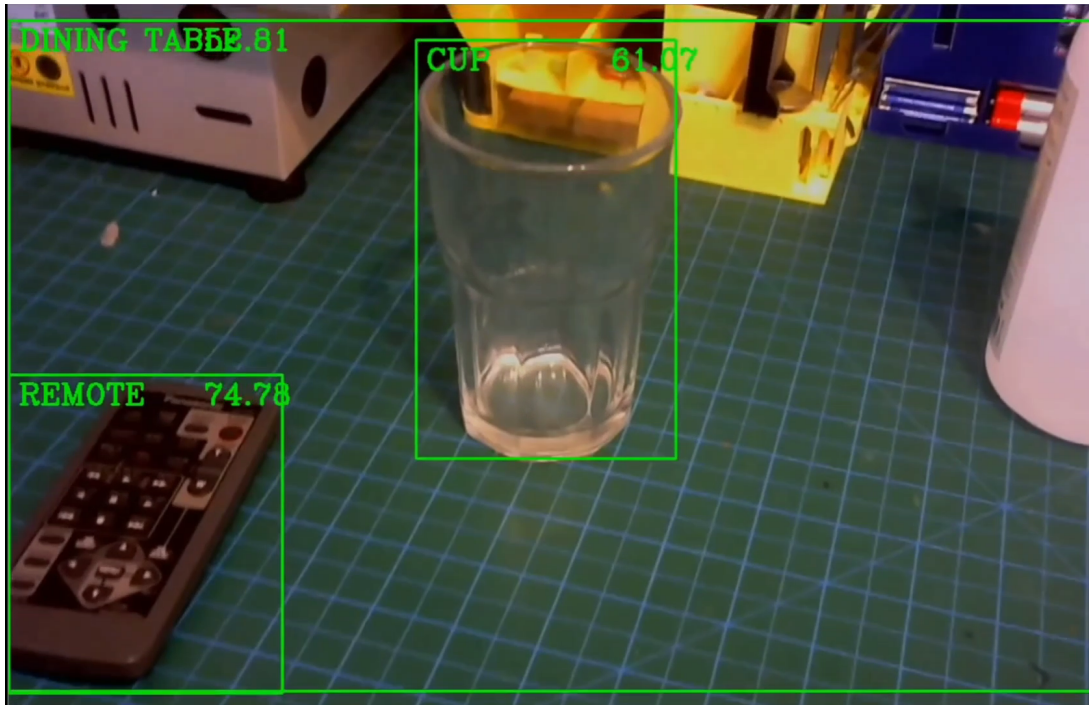


Fig B.1: Screenshot of output from output video



object_detection

Fig B.2: Output audio

REFERENCES

- [1] Patel, P., & Bhavsar, B. (2021). Object Detection and Identification. *International Journal*, 10(3).
- [2] Mahendru, M., & Dubey, S. K. (2021, January). Real Time Object Detection with Audio Feedback using Yolo vs. Yolo_v3. In *2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)* (pp. 734-740). IEEE.
- [3] Annapoorani, A., Kumar, N. S., & Vidhya, V. (2021). Blind-Sight: Object Detection with Voice Feedback.
- [4] Murali, M., Sharma, S., & Nagansure, N. (2020, July). Reader and Object Detector for Blind. In *2020 International Conference on Communication and Signal Processing (ICCSP)* (pp. 0795-0798). IEEE.
- [5] Srikanteswara, R., Reddy, M. C., Himateja, M., & Kumar, K. M. (2022). Object Detection and Voice Guidance for the Visually Impaired Using a Smart App. In *Recent Advances in Artificial Intelligence and Data Engineering* (pp. 133-144). Springer, Singapore.
- [6] Karmarkar, M. R. R., & Honmane, V. N. OBJECT DETECTION SYSTEM FOR THE BLIND WITH VOICE GUIDANCE.
- [7] Dewangan, R. K., & Chaubey, S. (2021). Object Detection System with Voice Output using Python.
- [8] Samhita, M. S., Ashrita, T., Raju, D. P., & Ramachandran, B. (2021). A critical investigation on blind guiding device using cnn algorithm based on motion stereo tomography images. *Materials Today: Proceedings*.
- [9] Potdar, K., Pai, C. D., & Akolkar, S. (2018). A convolutional neural network based live object recognition system as blind aid. *arXiv preprint arXiv:1811.10399*.
- [10] Lakde, C. K., & Prasad, P. S. (2015, April). Navigation system for visually impaired people. In *2015 International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC)* (pp. 0093-0098). IEEE.
- [11] Papageorgiou, C. P., Oren, M., & Poggio, T. (1998, January). A general framework for object detection. In *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)* (pp. 555-562). IEEE.
- [12] 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.
- [13] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).
- [14] Mao, Q. C., Sun, H. M., Liu, Y. B., & Jia, R. S. (2019). Mini-YOLOv3: real-time object detector for embedded applications. *IEEE Access*, 7, 133529-133538.
- [15] Zhao, L., & Li, S. (2020). Object detection algorithm based on improved YOLOv3. *Electronics*, 9(3), 537.

- [16] Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.
- [17] Kumar, M. N., & Usha, K. (2013). Voice based guidance and location indication system for the blind using GSM, GPS and optical device indicator. *International Journal of Engineering Trends and Technology (IJETT)*, 4(7), 3083-3085.
- [18] Dev, S., Jaiswal, S., Kokamkar, Y., Deshpande, K. B., & Upadhyaya, K. (2020, February). Voice Based Smart Assistive Device for the Visually Challenged. In *2020 International Conference on Convergence to Digital World-Quo Vadis (ICCDW)* (pp. 1-5). IEEE.
- [19] Manawadu, M., & Wijenayake, U. (2021). Voice-Assisted Real-Time Traffic Sign Recognition System Using Convolutional Neural Network.
- [20] Kurlekar, S., Deshpande, O., Kamble, A., Omana, A., & Patil, D. (2020). Reading Device for Blind People using Python OCR and GTTS. *International journal of Science and Engineering Applications*, 9(4), 049-052.
- [21] AlSaid, H., AlKhatib, L., AlOraidh, A., AlHaidar, S., & Bashar, A. (2019, October). Deep learning assisted smart glasses as educational aid for visually challenged students. In *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)* (pp. 1-6). IEEE.
- [22] Dave, H., Gobse, A., Goel, A., & Bairagi, S. (2020). OCR Text Detector and Audio Converter. *Int. J. Res. Appl. Sci. Eng. Technol*, 8, 991-999.
- [23] 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.
- [24] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).
- [25] Mao, Q. C., Sun, H. M., Liu, Y. B., & Jia, R. S. (2019). Mini-YOLOv3: real-time object detector for embedded applications. *IEEE Access*, 7, 133529-133538.
- [26] Zhao, L., & Li, S. (2020). Object detection algorithm based on improved YOLOv3. *Electronics*, 9(3), 537.
- [27] Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.
- [28] Kumar, M. N., & Usha, K. (2013). Voice based guidance and location indication system for the blind using GSM, GPS and optical device indicator. *International Journal of Engineering Trends and Technology (IJETT)*, 4(7), 3083-3085.
- [29] Dev, S., Jaiswal, S., Kokamkar, Y., Deshpande, K. B., & Upadhyaya, K. (2020, February). Voice Based Smart Assistive Device for the Visually Challenged. In *2020 International Conference on Convergence to Digital World-Quo Vadis (ICCDW)* (pp. 1-5). IEEE.
- [30] Manawadu, M., & Wijenayake, U. (2021). Voice-Assisted Real-Time Traffic Sign Recognition System Using Convolutional Neural Network.

- [31] Kurlekar, S., Deshpande, O., Kamble, A., Omana, A., & Patil, D. (2020). Reading Device for Blind People using Python OCR and GTTS. *International journal of Science and Engineering Applications*, 9(4), 049-052.
- [32] AlSaid, H., AlKhatib, L., AlOraidh, A., AlHaidar, S., & Bashar, A. (2019, October). Deep learning assisted smart glasses as educational aid for visually challenged students. In *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)* (pp. 1-6). IEEE.
- [33] Dave, H., Gobse, A., Goel, A., & Bairagi, S. (2020). OCR Text Detector and Audio Convertor. *Int. J. Res. Appl. Sci. Eng. Technol*, 8, 991-999.
-