E-VISION
Eyes for Visually Impaired using Smartphone Implementing Object Detection

Shital Patil[1], Aishwarya Raut[2], Sagar Jaiswal[3]
[1]Professor
[2]Final Year Student
[3]Final Year Student
Suman Ramesh Tulsiani Technical Campus

Abstract

Observing the difficulties faced by blind people we have developed this system to help the visually impaired people using simple mobile Application with the help of smartphones. By using object detection and speech synthesis, we aim to provide them aid in their daily task. Detecting object using object detection technique can be used in multiple industries as well as social applications. The project purpose is to use object detection for blind people and give them audio or vocal information about it as well as we are doing feature extraction to search for objects in a camera view. To improve the day to day lives of visually impaired blind people and people with low vision. Keeping in mind the challenges faced by them we felt the need for an Android application to provide the necessary information about the environment and the objects. This system will identify the objects around the user and convert the data to speech and then finally conveying the results to the user. To achieve this, we are using text to speech and Convolutional neural network integrated with SSD Mobilenetv1 architecture, which will ultimately enable the user to take a live recording and detect the objects and hear a text about them.

Keywords: *Image Processing, Object Detection, Android, Convolutional Neural Network, Artificial Intelligence, Text-to-Speech, Machine Learning, Computer Vision, Visually Impaired.*

1. Introduction

The estimated number of people who are visually impaired in the world is 285 Million, out of which 13% are blind while the rest are Visually impaired. The blind people have a complete loss of vision while visually impaired have partial blindness like Cataract, Albinism, Cushing Syndrome, etc. Although visually impaired can perform a task with a certain degree but blind people have very much difficulty in completing this task. Hence there has to be something that can aid these people to some extent in their daily task. They need something that can act as their eye when they need them. Of course, they cannot depend on people all the time as sometime no one might be there to help.

According to W.H.O, studies have also shown that 65% of visually impaired and 82% of blind people are above the age of 50. Even though these people have less chances of using a smartphone, we can still target a lot of people. To solve the issue for visually impaired people we needed something that can act as their eye and is also portable. Since almost 61% of the world is already using a smartphone, we can help them with the aid of these smartphones. These smartphones already have cameras built in hence we can use this to act as their eyes.

With current advancements in Computer vision state-of-art, we can now perform image processing and determine the contents of the image of the image by classifying objects into known classes. Along with this, we can also do localization to know the location of the object by drawing bounding boxes around these objects. This can be done using object detection techniques. With the help of Object Detection, we can find the objects in the image and using localization we can find the location of the objects which will help the user in guiding them towards the object.

Object detection has a good performance on Pc's while the performance is slower on a smartphone due to constraints on memory and processing power. The model trained are also very big in size (ranging in ~100-200 Mbs). Loading such files into a smartphone will make the app size huge which might not be good for real-time use on low-end devices. To overcome this, we have used Quantization which will be discussed later. The whole model is ported into TensorFlow-lite as it allows on-device machine learning inference with low latency and small binary size. We have used SSD MobileNetv1 which has been trained on COCO dataset for running inferences. Android text-to-speech API is used for converting the output to human voice form understandable to the user.

2. Process Flow

The working of the system involves interaction between the user and the app interface through voice. A specific sound is played at each event to let the user know that he has successfully launched the event. This helps them to know when to perform an action even if there is a lag in the application. This will not disrupt his process. A single big button is provided to help him click easily and ask for a specific task to be executed. The user input is processed by the machine and the object to be searched is extracted.

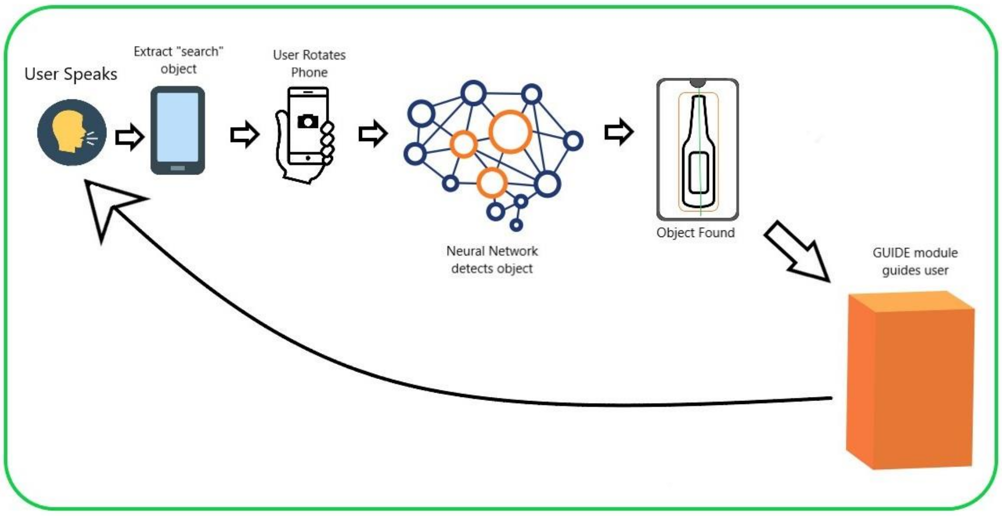

Figure 1. Process Flow

The camera starts capturing frames in the background and feeds it to the trained model. This model has been trained on COCO dataset using transfer learning. The model predicts objects in each frame and checks whether it is the object the user is searching for. If it is the object, it notifies the user through speech synthesis using text-to-speech. The user is guided using the guiding system. Once the object is reached, the user is asked whether he has the object. If not, the user is further guided until he reaches the destination.

The user is prompted with notification voice after a certain period of time if the object is not found. The application returns to the start state and repeats the same process all over again.

3. The System

Here are the different modules of the system that we have used to design the application. The models and some things that we have tried to increase the performance of the app. The model SSD Mobilenetv1 is a two-part module where Mobilenet acts as a base layer that provides feature maps for prediction to SSD.

3.1. The Model

The convolutional layers in the convolutional neural network perform complex and heavy matrix multiplications where the matrix sizes can be quite big. Even though these operations can be performed easily on computers, it takes quite a load on mobile devices due to limited power. Hence to solve this issue [2] MobileNet uses depth-wise separable convolutions. Hence There are two tasks performed in each convolutional layer where the
first depth-wise convolutional layer filters the input, followed by a 1x1 convolutional layer that combines these features to create new features. It does the same thing as a normal convolutional layer but instead, it performs much faster. The full MobileNet V1 architecture consists of regular 3x3 convolution for the first layer, followed by 13 depth-wise separable convolutional layers.

After getting the feature maps from MobileNet, [3] SSD detects objects in these feature maps by dividing it into cells. Each cell makes 4 object predictions. Each of these predictions composes of a boundary box and *number of the classes+1* score and then we pick the top class with the highest score. It also does multi-scale feature detection to detect objects independently. For each cell 4 boundary boxes are used where the size of each of these boundary boxes are arbitrary. But in real life the boxes are arbitrary. Like a person or a car has a certain height-width ratio. Hence, we use k-means clustering during training to determine the best anchors required for prediction.

Even after making predictions there are lot boundary boxes predicted which needs to be reduced. We can set the prediction threshold to determine the minimum probability of an object to be displayed. Further, these boxes can be reduced by using NMS (Non-Max Suppression) where the box with the highest probability for containing an object is selected. For the remaining boxes, IOU is calculated with respect to the previous selected box and if IOU greater than or equal to 0.5, they will be discarded. This gives the final bounding Boxes for the image.

The Model has been trained on COCO dataset using transfer learning and fine-tuning the layers. The model has been reduced in size drastically by using quantization. TensorFlow provides model optimization where we can quantize the model to reduce its size. Inference efficiency is a critical issue when deploying machine learning models to mobile devices because of the model size, latency, and power consumption. Quantizing the model uses the technique where we reduce the precision of weights from FLOAT to UINT8. This means 4 bytes of the variable is replaced with 1 byte. Since there are millions of parameters in a convolutional neural network. This leads to a reduction in size at the cost of small precision. We have applied post training quantization to our model.

3.2. The Guiding System

The object detection model was used to find the objects and their respective position in the image, but these boxes can also be used to guide the user towards the object he might be searching for. To deal with this first we had to align the object to the center with respect to the screen. Hence for each bounding box, we find the center X and center Y coordinates are calculated and they are compared with the center of the screen coordinates. If the center coordinates are less than or equal to 1/10th of either part of the screen, it will produce a message to shift the mobile either left, right, up or down based on position calculated. 1/10th of one part of the screen is used as a boundary because we don't want the user to align the phone exactly at the center of the object. Hence, we are using a soft boundary for the object to be perfectly aligned.

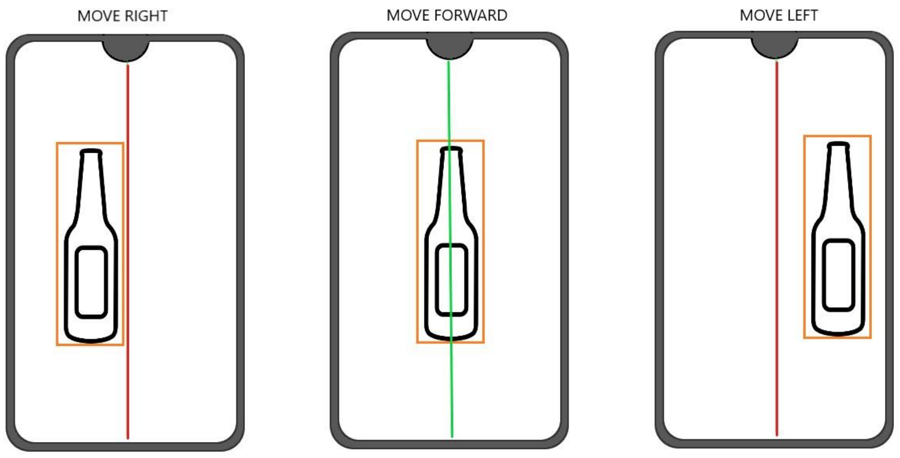

Figure 2. Phone orientation

After the object has been aligned with the center of the screen we have to determine when the user should move forward and when to stop. To solve this issue there are 2 methods. The first method revolves around determining the distance of the object with the user. For this, we would need to know the focal length of the device camera and original height and width of the object. Even though not all objects have the same width and height, we can provide a mean height and width for each class of object. Using this we can calculate the distance of the object from the camera based on triangle similarity.

$$\text{Distance} = (\text{Width x Focal length}) / \text{object width in pixels} \qquad (1)$$

Using this can give an approximate distance which we can use to build a soft boundary to stop navigating when a certain value is reached. But this work is not completely feasible with our system as the focal length of the camera is not generic rather than specific. Hence, we need to calibrate first to know the focal length of the camera, which can be a difficult task for a visually impaired person.

Therefore, we resolved to another idea where we can use dimensions of the object on the screen to determine the closeness of the object. Whenever an object is detected we first determine the longer side of the object. If the longer side is the height of the object, we will check whether the object has covered at least 90% of the screen height. Since most average size object ranging in size close to a bottle are in the range of hands when they occupy at least 90% of the screen. We don't need to find the exact or approximate distance of object as it will be of no use to the visually impaired user hence, we resolved to this method to know when the user should stop moving forward.

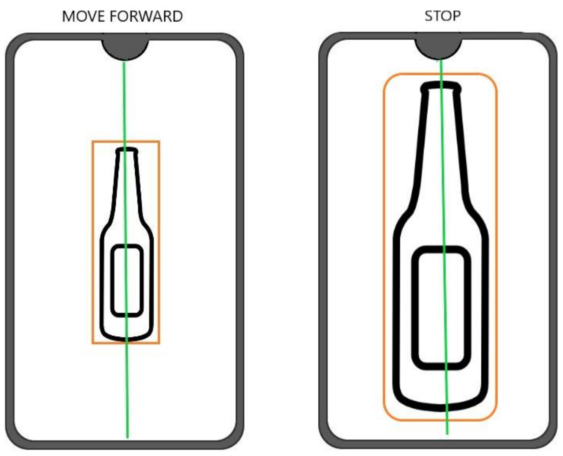

Figure 3. Guide using area covered

3.3. Text-to-speech and Speech recognition

The visually impaired user has only 2 means to interact with the application. One is through voice and another is through touch. But touch sensation alone won't be enough for the user to interact with the user hence we use both together to help the user through the application. The directions to turn left or right will be given to the user through text—speech in his understandable language. The speech recognition is used to take inputs from the user through voice. Notification sounds are provided at certain action to help the user know that certain action has been performed. This makes the app easy-to-use for the user.

4. Results

We were able to develop the application with a size of just ~12 MB. The model was tested on 2 devices. One was an old device which runs on Octa-core 1.5 GHz Cortex-A53 and a new device which runs on Octa-core 1.9 GHz Snapdragon 660.

Table 1. Inference Times

Sr. No.	Device	Inference time
1.	Octa-core 1.5 GHz Cortex-A53	200ms
2.	Octa-core 1.9 GHz Snapdragon 660	110ms

It was obvious the inference time on high-end device will be better in comparison to an older device due to an older CPU architecture. It was observed that the inference time was around ~130ms on an older device at the start but later it increased to 200-300ms. While in a newer device the inference was almost constant. The model had an accuracy of 60% but it is a trade-off we make for lower size and faster inferences. The app performed weakly in low light conditions as it wasn't able to predict the object even after being in a hand distance but the app runs well in well-lighted condition. Due to the use of quantized model the size of the model reduced from 28mb to 4mb.

We have also found out that the range of object detection supported by the model is approximately 1.2 meters. This has been verified on both devices. Hence, the detection process is independent of the hardware of the mobile architecture.

5. Conclusion

Thus, this paper represents the development of E-VISION considering CNN and TTS stages, to create an application that was gradually improved and refined over the project. The project consisted of the construction of an application composed by several parts, integrating the system of live video recording by the mobile device, which is used by a CNN technology for recognition of its objects, which is then synthesized through a process of TTS. Optimizations carried out for improving outcomes resulted in a more efficient application, capable of responding to the challenge set by the theme of the project: Talking camera that speaks out what it is in surrounding for the blind.

6. Future Work

There are several things we want to try in the future to make the application more useful for the visually impaired user. Firstly, we want to add support for more languages in the future. Currently, it only supports English. Secondly, we want to add support for OCR to read texts on medicines, newspaper, etc. This will help them search for specific medicines or read texts that are not in Braille script. Lastly, we want to create a more compact device where we can integrate a camera inside the spectacles of the user which will provide a live feed to a dedicated device which will do the computation. This can eliminate the load of carrying a smartphone in hand all the time.

We would also like to use GPU computation power of TensorFlow-lite which is currently supported for only float models. There are also certain restrictions on Ops

7. References

7.1. Journal Article

[1] *Joseph Redmon, Ali Farhadi, YOLOv3: An Incremental Improvement.*

[2] *Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, MobileNet: Efficient Convolutional Neural Network for Mobile Vision Applications.*

[3] *Prof. S.P. Jadhav, Stelly Tomy, Sonali S. Jayswal, Harshvardhan D. Dhaware, Akshay R. Vijapure," Object Detection in Android Smartphone for Visually Impaired Users" November 2016.*

[4] *Parijat Dube, Bishwaranjan Bhattacharjee, Elisabeth Petit-Bois, and Matthew Hill," Improving Transferability of Deep Neural Networks", IBM Research AI, Yorktown Heights NY 10598, USA July 2018.*

[5] *Zhongjie Li, Rao Zhang," Object Detection and Its Implementation on Android Devices" May 2017.*

[6] *Raghuraman Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper", June 2018*