Real-time object detection and face recognition system to assist the visually impaired

Anish Aralikatti*, Jayanth Appalla, Kushal S, Naveen G S, Lokesh S and Jayasri B S
Department of Computer Science and Engineering, The National Institute of Engineering, Mysuru, India

*E-mail: anisharalikatti24@gmail.com

Abstract: The application system presented here is for use in an Android based mobile phone by the visually impaired for detecting objects in their vicinity, which will help them move around safely, without crashing into objects. The detection of objects is done from a real time video taken from the mobile phone camera. People and objects are detected from this video using OpenCV, YOLO and FaceNet. If a human is detected, the system identifies the person. His identity will then be converted into audio and presented to the user. Similarly, objects detected in the vicinity will be presented in audio format to the user.

1. INTRODUCTION

Lakhs of people inhabit this planet with visual disabilities and hence incapable of comprehending their surroundings. As per the published report of World Health Organization (WHO), an estimated 28.5 crore people have visual disability, of which 3.9 crores are blind and 24.6 crores have partial vision. Though they can have other options to take care of day to day activities, they face difficulties in movement and hence social discomfiture.

Poor vision affects: routine activities requiring medium vision, communication, reading and writing requiring near vision, assessing the nearby space and movements requiring a far vision. Also, doing any activity requires continuous visual focus.

The recent development in computer vision technologies, has encouraged to take up research in developing tools for the people with vision impairment. These aids are intended to help the person to move around safely.

In this work, the possibility of using technologies to detect objects and humans in the near vicinity and their deployment in mobile devices have been explored. The outcome of the detection is given to the person using the cell phone in audio format. Vision and listening senses resemble in many ways. A real-time object detection and face recognition system, with the aim of creating awareness to the person about the objects and persons present in his vicinity is described.

2. PROBLEM STATEMENT

- To develop a real-time object detection system to help visually impaired persons. The objects present in the vicinity of the user are detected in real-time and presented to them.
- To develop a real-time face recognition system to help the visually disabled to recognise the
persons present in his vicinity in real-time.
- To collect the results and present it to the person in audio format. The results are presented to the user in order of their priorities, to ensure that the visually impaired have an accurate perception of their surroundings.

3. LITERATURE SURVEY

Accurate object detection from a real-time streaming video, relies heavily on machine learning. For a computer system to classify the objects present in a real-time video stream, it has to train with labelled data. The data consists of the labelled images for training, which requires us to have a large data set for training. Larger the data set, the greater is the accuracy of the trained model, hence we get better results during object detection.

Deep learning techniques making use of Convolutional Neural Networks (CNN) for object detection as shown in figure 1, are used extensively nowadays. [1] Neural networks consist of multi-layered architecture including an input layer, many hidden layers and an output layer. The input layer receives the extracted image of the object from the video stream. The hidden layers act as filters that receive an input from an upper layer, transform it with a specific pattern or feature and then send it to the next layer. The output layer classifies the image based on the input it receives from the hidden convolutional layers.

![Convolutional Neural Network](image1)

**Figure 1.** Convolutional Neural Network

There are many implementations of CNN for object detection from images and videos briefly described below:

**R-CNN (Region-Convolutional Neural Network)** shown in figure 2 is a region based approach using CNNs. Ross Girshick, Jeff Donahue, Trevor Darrell and Jitendra Malik [2] have achieved a mean average precision (mAP) of 53.7% with more than 30% improvement [2].

![R-CNN](image2)

**Figure 2.** R-CNN: Regions with CNN features [2]

Ross Girshick [3] developed **Fast R-CNN** which is an improvement over R-CNN.
In Fast R-CNN shown in figure 3, the entire complete image is processed to detect feature maps which are fed to the RoI layer to find region of interest in the given frame. The output given are probabilities of existence of the object in the frame which are then filtered based on some threshold value to give the final output. The training of all network layers is processed in a single-stage. It saves on storage space, and improves accuracy and efficiency with more appropriate training schemes.

A disadvantage of Fast R-CNN is that even though it increased the speed of object detection, the algorithm is still very slow to be used for real-time object detection applications.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun [4] developed an improvement on the Fast R-CNN algorithm called Faster R-CNN shown in figure 4.

It comprises two modules, the first being a deep convolutional network based on regions and the second a Fast R-CNN detector that uses these regions. This system forms one unified network for object detection.

The time spent on every image is reduced to 10ms making this algorithm very fast. However, this approach involves a lot of computation making it infeasible for applications that need to run on CPUs with lesser computing power and limited computing resources such as embedded systems, smartphones, etc.

Real time object detection using YOLO
To overcome the drawbacks of the above algorithms, it is better to use a regression/classification
based framework rather than the region proposed based frameworks mentioned above.

Joseph Redmon, Santosh Divvala, Ross Girshick and Ali Farhadi [5] in their paper titled ‘You Only Look Once: Unified, Real-Time Object Detection’ defined a framework called YOLO (You Only Look Once). Humans just need one glance at an object to know its properties such as which object it is, its dimensions, its location, depth, etc. The same analogy is used in this algorithmic framework to detect objects with a single look.

The process of detection is explained with reference to YOLO network architecture in figure 5.

![YOLO Network Architecture](image)

**Figure 5. YOLO Network Architecture [5]**

YOLO divides the input image into an $S \times S$ grid and each grid cell is responsible for predicting only one object. Each grid cell predicts a maximum of $B$ bounding boxes and their corresponding confidence scores [5] as seen in figure 6.

![YOLO Model](image)

**Figure 6. YOLO Model [5]**

The YOLO network can process images in real-time at 45 FPS and a simplified version Fast YOLO can reach 155 FPS with better results than other real-time detectors [5].

Advantages of YOLO:
- YOLO accesses the whole image in predicting boundaries while region proposal methods restrict the classifier to a specific region.
- This method is fast and suitable for processing in real-time.
- From a single network, object locations and classes are predicted. To enhance accuracy, end to end training is carried out.
- YOLO is generalised and better compared to other methods when used in domains such as artwork.
- YOLO demonstrates fewer false positives on background.
- YOLO detects one object per grid cell while enforcing spatial diversity during prediction.

Joseph Redmon and Ali Farhadi [6] in their paper titled ‘YOLO9000: Better, Faster, Stronger’ proposed an upgrade to the original YOLO framework known as YOLOv2. All of the improvements ensure that YOLOv2 is faster than YOLOv1 and has a higher degree of accuracy in object detection.

YOLOv3 [7] is another incremental improvement to the YOLO framework. On a Pascal Titan X, it processes images at 30 FPS and has a mAP of 57.9% using the COCO dataset [7]. Using this version
of YOLO, real-time object detection for various applications can be accurately performed.

**FaceNet for Face Recognition**

Florian Schroff, Dmitry Kalenichenko and James Philbin in their paper [8] titled “FaceNet: ‘A Unified Embedding for Face Recognition and Clustering’” present a unified system for face verification, recognition and clustering. It is based on learning a Euclidean embedding per image using a deep convolutional network. The network is trained to ensure that the square of the distances in the embedding space correspond to face similarity. This means that the faces of the same person have small distances and faces of dissimilar people have large distances. After embedding, face verification involves thresholding the distance between the two embeddings; recognition becomes a classification problem; clustering can be achieved using techniques such as K-means clustering.

The FaceNet system is used to extract high quality features from faces called face embeddings. These are used to predict a 128 element vector representation of these features. This is then used to train a face identification system [8]. The FaceNet model structure is shown in figure 7.

![FaceNet Model Structure](image)

**Figure 7.** FaceNet Model Structure [8]

On the established ‘Labelled faces in the wild’ dataset, it achieves an accuracy of 99.63% [8].

### 4. EXISTING SYSTEM

Many computer vision systems exist today to help the visually impaired in various aspects of their life. These include Augmented Reality based wearable goggles, video calling apps for the visually impaired to ask for assistance, AI and GPS based navigation systems, etc. These systems are built to work in specific cases or conditions, and cannot be broadly used. There are cases wherein the people with visual impairment have to realise about their surroundings, which is not easily possible with the existing systems.

**Drawbacks of existing system:**

- They are expensive. Most of the visually impaired people in India cannot afford such expensive technological products.
- The systems have complex functionality, making it difficult to be used by the visually impaired.
- Simple requirements such as providing the names of the objects in the surrounding or identifying the people in the surrounding are not provided.
- No personalised user experience.
- The systems are not real-time.

### 5. PROPOSED SYSTEM

We propose to build an Android smartphone based application which can detect and name the objects existing in the surrounding. It also provides names of known people that are present near the user after performing face recognition.

**Advantages of the proposed system:**

- It is free and easily available. Any person having an Android based smartphone can download the app and use it.
- Easy to use with a simple UI.
- Provides real-time results.
- Reliability: The system can be relied upon by the visually impaired user to give reliable results. Depending on the video quality, difference between various objects like chair and table etc. can be easily differentiated.

6. SYSTEM DESIGN

*System Architecture* is described in figure 8.

![Figure 8. System Architecture](image)
Data Flow Diagram is described in figure 9.

![Data Flow Diagram](image)

**Figure 9.** Data Flow Diagram

7. SYSTEM IMPLEMENTATION

**Tools and Technologies Used**

**Python:** It has a large number of libraries for applications such as Scikit-learn for machine learning, OpenCV for computer vision, TensorFlow for neural networks, etc.

**Java:** Android uses Java as one of its programming languages. Using Java, the Android Apps for object detection and face recognition have been built.

**OpenCV:** Used for performing real time computer vision tasks.

**YOLO:** Provides a framework that allows detection of objects in near real time speeds. For deployment in a mobile device we are using Tiny YOLO, which is a lightweight YOLO framework for mobile and edge devices.

**TensorFlow:** An open source software library and framework used for neural network and machine learning applications. It has a flexible architecture and can be deployed on servers, laptops, mobile phones, edge devices, etc.

**TensorFlow Lite:** TensorFlow for mobile devices has a separate smaller and lighter framework that utilises the less powerful computing power of smartphones and edge devices called TensorFlow Lite. This smaller framework allows for easy deployment of machine learning applications on the smaller and less powerful devices.

**Keras:** Keras is a deep neural network library that runs on top of other frameworks such as TensorFlow. It is user friendly and allows for the training of machine learning models with relative ease using neural networks. Keras contains a host of add-on features for neural networks such as layers, optimizers, activation functions, etc. making it a useful library for many applications.

**FaceNet:** Used to extract high quality features from faces and hence train a face identification system.

**Scikit-learn:** Consists of various clustering, regression and classification algorithms for various machine learning applications and can be interoprated along with other Python libraries such as SciPy and NumPy. Scikit-learn also uses SVM (Support Vector machines) along with FaceNet during the training process to create accurate models for face recognition.

**Development Environments Used**

i) Android Studio

ii) PyCharm

iii) Google Colabs

iv) Kaggle

**Phases in Project Implementation**

**Phase I:** In this phase the entire system was built using Python. The proposed system design was implemented using Python scripts for each module. The main reason for this approach was to ensure
that the complete system was functioning properly. Once the complete system was built and tested for functionality, the code was migrated to Java for implementation in Android.

Python was used on a Windows and Linux based machine to build and test the proposed functionality. In this phase, the system was being built on a laptop, which gave access to better computing resources such as CPU and GPU of the laptop. All the models for face recognition and object detection were trained during this phase.

Once the system was completely built and tested for proper functionality, the code was then migrated to Java.

**Phase II**: This was the second phase of implementation of the project where the entire code written in Python was migrated to Java for the purpose of implementation in an Android App. The trained models from phase I are directly implemented in the Android App for object detection and face recognition.

Since functionality of the system was already built using Python, the main job in this phase included changing the libraries to support Android and Java, migrating Python code and rewriting it in Java and importing the trained models into the application.

Once the main tasks were completed, the Android App was built and installed on a mobile phone for testing. The voice API for Android was also added to ensure that the results are converted into audio format.

Finally, a simple user interface was created, keeping in mind that the end user for the application should not face any difficulty in operating the application.

The initial models used in the application for face recognition were trained on the cloud using GPU on Google Colabs and Kaggle along with some training on the local GPU on the laptop. Later on, for subsequent face registration of new faces, the feature of training a model for a new face was added on the mobile app. This is to ensure that new faces can be added directly into the mobile device without the need to train a new model externally and then load the model into the mobile device.

**Pseudocode for Object Detection**

```
start object detection process
import OpenCV
import yolo-weights, yolo-config, coco.names
initialise queue
activate webcam using OpenCV
capture real-time video from device camera using OpenCV
split the video into its constituent frames
while there are input frames to be processed
   take one frame at a time
   convert frame to grayscale image
   convert image into NumPy array (or matrix for Java)
   perform object detection using YOLO
   display bounding box around object with label of detected object
   insert the names of detected objects into queue
parallel audio conversion process
   if queue is not empty
      remove item at head of queue
      convert name of item into audio for output
```
Implementation process of Face Recognition is described in figure 10.

Figure 10. Implementation process for Face Recognition

Pseudocode for Face Recognition

1. Start face recognition process
2. Import OpenCV
3. Import FaceNet model of trained faces
4. Import TensorFlow-Lite
5. Activate webcam using OpenCV
6. Capture real-time video from device camera using OpenCV
7. Split the video into its constituent frames
8. While there are input frames to be processed
   - Take one frame at a time
   - Perform face detection on the image
   - If a face is present in the image frame
     - Perform face recognition for the face
     - If face is recognized
       - Convert the name of person recognized into audio
       - Display bounding box around person with name of person
   - If there are no frames to be processed or camera is shut
   - End the face recognition process

if no frames to be processed or camera is shut
end object detection process
On Device Training for Face Recognition

The Android App also has the additional feature of registering and training a face on the mobile phone. This allows the user to register new faces on his local device without the need to perform training on another device and import the model. The registration process takes the images of the new person as input and trains a model for the new person. However, the accuracy of the model trained on the mobile device will be less than that of the model trained on an external device with a powerful GPU.

Pseudocode for implementation

if face to be added is already registered
    improve the face recognition model for the person
else
    add the new person to the face recognition model

8. TESTING

In this project, system testing was performed to ensure that the Android App was performing object detection and face recognition as per the pre-set requirements. Various test cases were developed for various classes of objects to ensure that the system was performing object detection in a satisfactory way and producing acceptable output. Similarly test cases were developed for face recognition to ensure that the Android App was recognising the faces correctly and giving proper acceptable output. The results are shown in table 1 and table 2.

| Sample Input                      | Expected Output                        | Obtained Output     | Confidence Score of prediction | Status  |
|----------------------------------|----------------------------------------|---------------------|-------------------------------|---------|
| Face of Anish with clear background | Face recognized with name of person as Anish | Anish recognized | 0.68                          | Success |
| Face of Anish in a normal room    | Face recognized with name of person as Anish | Anish recognized | 0.60                          | Success |

| Sample Input       | Expected Output         | Obtained Output | YOLO Confidence Score | Status |
|--------------------|-------------------------|-----------------|-----------------------|--------|
| Video of a person  | Person detected         | Person          | 0.34                  | Success|
| Video of a chair   | Chair detected          | Chair           | 0.42                  | Success|
| Video of a cell phone | Cell phone detected  | Cell Phone      | 0.46                  | Success|
| Video of a laptop  | Laptop detected         | Laptop          | 0.69                  | Success|
| Video of a TV      | TV detected             | TV              | 0.63                  | Success|
| Video of a bottle  | Bottle detected         | Bottle          | 0.30                  | Success|
| Video of a cup     | Cup detected            | Cup             | 0.75                  | Success|
| Video of a clock   | Clock detected          | Clock           | 0.95                  | Success|
| Video of a book    | Book detected           | Book            | 0.58                  | Success|
| Video of an umbrella | Umbrella detected   | Umbrella        | 0.31                  | Success|
9. SAMPLE OUTPUT

Figure 11. Sample output for object detection taken from Android App on mobile phone.

The sample output showing detected objects with bounding boxes, respective labels and confidence scores are shown in figure 11. These images have been taken from the object detection Android App on a mobile phone. The sample output showing face recognition on the Android based mobile app along with bounding box, name of person recognized and the confidence score is shown in figure 12.
10. CONCLUSION

One of the main issues involving object detection is object classification and object localisation within a scene. Use of deep neural networks has helped in addressing the subject of object detection. However, deploying such techniques on mobile devices requires high computational and memory resources. Hence using small deep neural network architectures for object detection such as Tiny YOLO are giving good results and show that they can be used for real time object detection using mobile devices which can help the visually challenged.

As high accuracy is required for face recognition, we have successfully used FaceNet architecture for face recognition on an Android device.

REFERENCES

1. Zhao Z, Zheng P, Xu S and Wu X Object Detection with Deep Learning: A Review 2019 IEEE Transactions on Neural Networks and Learning Systems 30 3212
2. Girshick R, Donahue J, Darrell T and Malik J Rich feature hierarchies for accurate object detection and semantic segmentation 2014 IEEE Conference on Computer Vision and Pattern Recognition (Columbus, OH) pp 580-587
3. Girshik R Fast R-CNN 2015 IEEE International Conference on Computer Vision (Santiago) pp 1440-1448
4. Ren S, He K, Girshick R and Sun J Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks 2017 IEEE Transactions on Pattern Analysis and Machine Intelligence 39 1137
5. Redmon J, Divvala S, Girshick R and Farhadi A You Only Look Once: Unified, Real-Time Object Detection 2016 IEEE Conference on Computer Vision and Pattern Recognition (Las Vegas) pp 779-788
6. Redmon J and Farhadi A YOLO9000: Better, Faster, Stronger 2017 IEEE Conference on Computer Vision and Pattern Recognition (Honolulu) pp 6517-6525
7. Redmon J and Farhadi A YOLOv3: An Incremental Improvement 2018
8. Schroff F, Kalenichenko D and Philbin J FaceNet: A Unified Embedding for face recognition and clustering 2015 IEEE Conference on Computer Vision and Pattern Recognition (Boston) pp 815-823