OBJECT DETECTION AND SOUND RECOGNITION FOR BLIND PERSON

Muhammad Talha, Mohsin Iqbal, Behzad Ahmed
17-NTU-1099, 17-NTU-1088, 17-NTU-1079

Supervised by Dr.Asif Habib

Department of Computer Science
National Textile University, Faisalbad

Abstract

Blind persons, face a lot of challenges with respect to their autonomous movement. Most of the time they need someone for assistance, especially in the indoor environment, which is also not a feasible thing. There are already very good GPS based products available but these products fail when it comes to an indoor environment. So, in order to tackle these issues, a computer vision-based system was proposed to assist a blind person throughout an unknown indoor environment by detecting the hurdles coming on his way during the movement. The basic proposed objects were Doors, Stairs, Chairs, Tables, and Vehicles. This report contains different methodologies, their comparisons, summary, and future depictions.

Acknowledgements

I acknowledge the guidance which I have received from my supervisors Dr. Asif Habib and Dr. M Adeel throughout the project. Also Umar Hayat one of my seniors also helped me in the technical assistant and Arslan Sadiq one of my juniors helped me in the collection of the Dataset.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Strong and productive indoor object detection can help individuals with extreme vision hindrance to autonomously get to new indoor conditions and evade threats [1]. While GPS-guided electronic way-finding helps show a lot of guarantee in outdoor situations, there are not many indoor direction and route helps. Computer vision innovation on a fundamental level can possibly help blind people to autonomously get to, comprehend, and investigate such situations. However it stays a test for the accompanying four reasons: First, there are huge intra-class varieties of appearance and plan of items in various structural conditions. Second, there are generally little between class varieties of various item models. Third, comparative with lavishly finished and colored objects in the regular scene or open-air situations, most indoor texture are man-made and have little surface. Highlight descriptors that function admirably for outside situations may not adequately portray indoor items. At last, object with huge view varieties, and regularly only parts of objects (inside the field of view) are caught when a visually impaired client moves. A compelling indoor way-finding help should deal with object impediment and view varieties. This report consists of an introduction and the main idea of the whole project, the portion of the work to be completed, following by the literature review, comparison of different methodologies used with their results, and the future plans.

1.1 Motivation

The motivation behind the project is to make the blind people autonomous such that the dependency of them to the others could be minimized. As it is very difficult for both parties to remain together 24/7. So by keeping these things in mind, an idea of the

product came into the place.

1.2 Description of the work

The purposed system consists of two parts, one is the object classification model and the other one is the mobile interface. The image captures taken from the Mobile camera will be sent to the model which will then return the prediction results, and later on which will be conveyed to the blind person through a voice message. The communication has made possible with the help of Tensorflowlite API which is working as a tool of the integration between the model and the mobile interface.

1.3 Technologies Used

In this project, the following technologies were used:

1.3.1 Python (used for training the model)

Python is one of the most popular general-purpose programming languages. It is among the world's fastest-growing programming languages and is used by software engineers, mathematicians, data analysts, scientists, network engineers, students, and accountants. The features that make Python such a powerful language are:

- It is an Interpreted, object-oriented, and a high-level programming language. Python is called an interpreted language as its source code is compiled to bytecode which is then interpreted. CPython usually compiles Python code to bytecode before interpreting it.

- It supports dynamic typing and Dynamic binding. In languages like Java, C and C++ you cannot initialize a string value to an int variable and in such cases, the program will not compile.Python does not know the type of the variable until the code is executed.

- Python has an easy syntax which enhances readability and reduces the cost of code maintenance. The code looks elegant and simple.

- Automatic memory management. Memory management in Python involves a private heap(a data structure that represents a queue) containing all Python objects and data structures. On-demand, the Python memory manager allocates the heap space for Python objects and other internal buffers. The management of this private heap is ensured internally by the Python memory manage

1.3.2 Keras (for building the model))

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research.

- Simple – but not simplistic. Keras reduces developer cognitive load to free you to focus on the parts of the problem that really matter.

- Flexible – Keras adopts the principle of progressive disclosure of complexity: simple workflows should be quick and easy, while arbitrarily advanced workflows should be possible via a clear path that builds upon what you've already learned.

- Powerful – Keras provides industry-strength performance and scalability: it is used by organizations and companies including NASA, YouTube, or Waymo.

1.3.3 Colab Notebook (Cloud ML Platform)

Colab is a free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members - just the way you edit documents in Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook.

As a programmer, you can perform the following using Google Colab.

- Write and execute code in Python.
- Document your code that supports mathematical equations
- Create/Upload/Share notebooks.
- Import/Save notebooks from/to Google Drive
- Import external datasets e.g. from Kaggle
- Integrate PyTorch, TensorFlow, Keras, OpenCV
- Free Cloud service with free GPU

1.3.4 Tensorflowlite (Used to convert the model for the mobile use)

A TensorFlow Lite model is represented in a special efficient portable format known as FlatBuffers (identified by the .tflite file extension). This provides several advantages over TensorFlow's protocol buffer model format such as reduced size (small code footprint) and faster inference (data is directly accessed without an extra parsing/unpacking step) that enables TensorFlow Lite to execute efficiently on devices with limited compute and memory resources.

A TensorFlow Lite model can optionally include metadata that has human-readable model description and machine-readable data for automatic generation of pre- and post-processing pipelines during on-device inference.

We can generate a TensorFlow Lite model in the following ways:

- Use an existing TensorFlow Lite model: Refer to TensorFlow Lite Examples to pick an existing model. Models may or may not contain metadata.
- Create a TensorFlow Lite model: Use the TensorFlow Lite Model Maker to create a model with your own custom dataset. By default, all models contain metadata
- Convert a TensorFlow model into a TensorFlow Lite model: Use the TensorFlow Lite Converter to convert a TensorFlow model into a TensorFlow Lite model. During

conversion, you can apply optimizations such as quantization to reduce model size and latency with minimal or no loss in accuracy. By default, all models don't contain metadata.

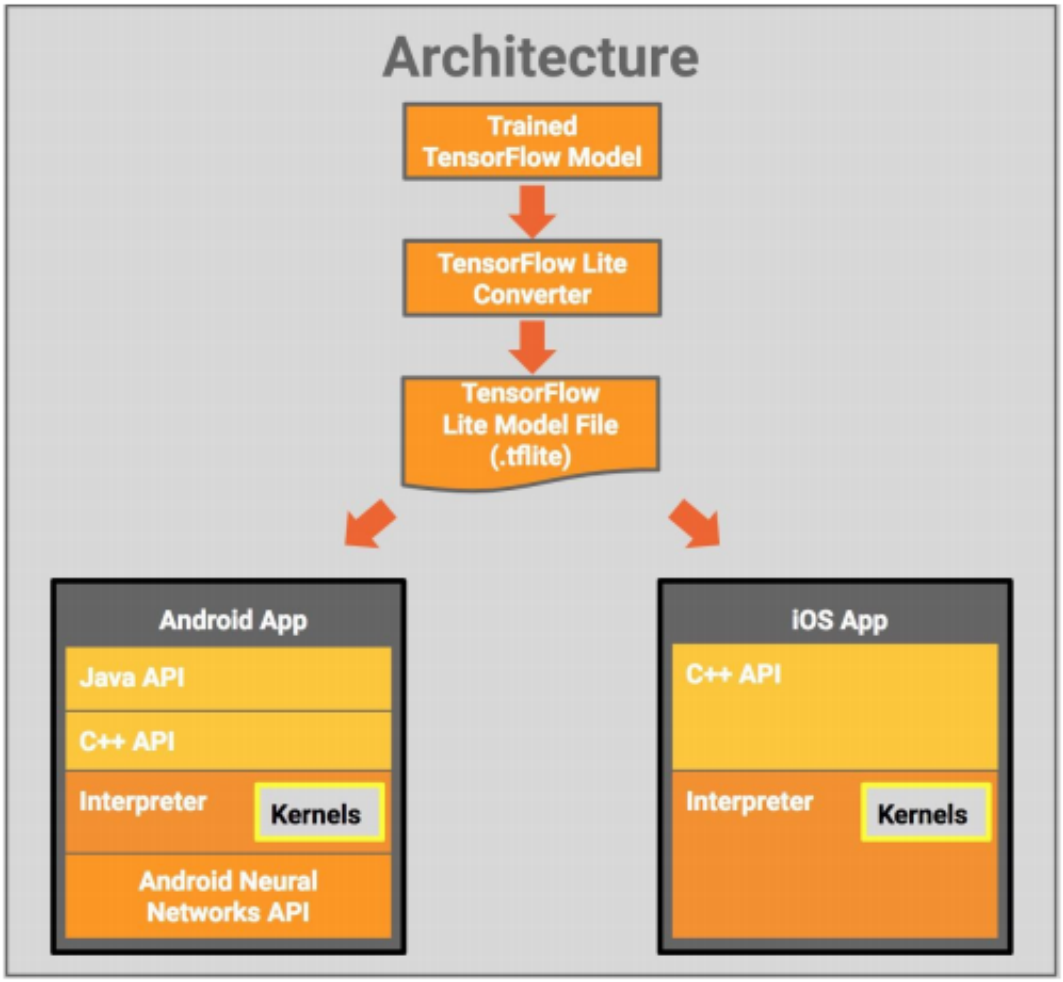


Figure 1.1: Architecture

1.3.5 Android Studio with Java (Used to implement the android app)

- Mobile phones have become a very common platform for communication and personal use.
- Android is an development platform for the application and it is becoming the most widely used platform among the mobile technologies.

- According to Google 2013 research , there are more than 1 million android application published and over 50 billion applications downloaded.
- A developer survey conducted in April-May 2013 found that 71
- Android is most widely used operating system in the world.

1.4 Project management

Following is the Project timeline, showing the work break down through out the year.

S.No.	Elapsed time since start of the project	Milestone	Deliverable
1.	7 weeks	Initial requirements Selected tools and technologies, Fetching Data	Prototype 1 (Initial Interface and Data Fetching)
2.	16 weeks	Collecting and Checking the Best test that can give accurate Results.	Tests
3.	27 weeks	Checking the app With people and how well it works.	Filtered Alpha Users
4.	38 weeks	Finalizing Tool and Optimizing it by training it on different data sets.	Final Application

Figure 1.2: Project Timeline

Following is the Gantt chart, showing the work break down through out the year.

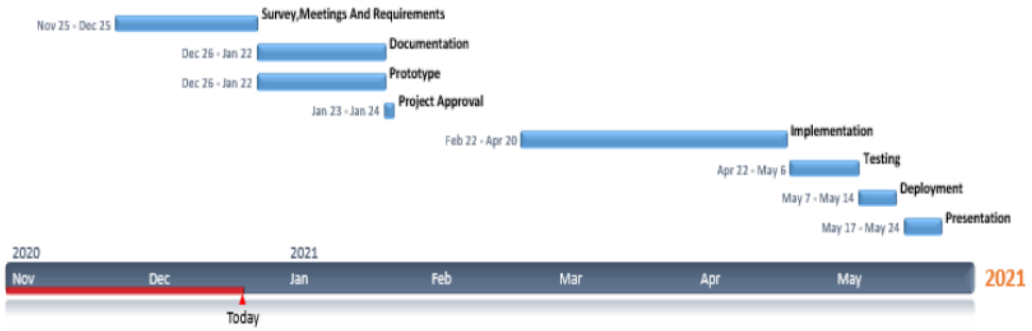


Figure 1.3: Gantt Chart

1.5 System Design

The overall design is described as follow which is an MVC design Pattern and some people also termed it as architecture as well.

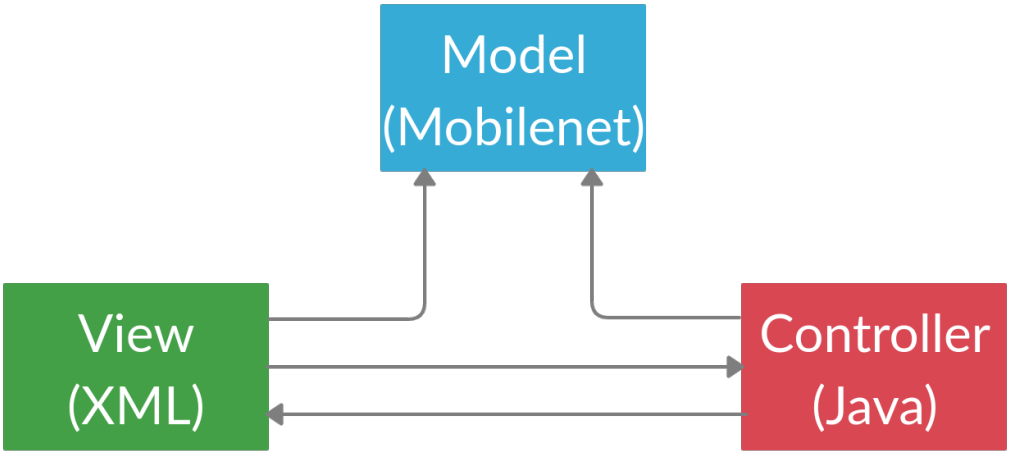


Figure 1.4: System Design

1.6 Risks and Risk mitigation

1.6.1 Environmental Factors

Environmental conditions affecting camera views and video images affect the performance of object detection and analysis. You can improve performance by considering aspects of the environment for your video camera.

1.6.2 Strong shadows

Distinct shadows can distort the bounding box of objects. This distortion interferes with object type and color classification. Shadows can also cause an object to trigger a location alert, for example a tripwire, when it should not be triggered. Shadows can also prevent an alert from being triggered when it should. For example, a shadow that is connected to an object can cause the object to appear to be larger than it is. Consider these effects when you deploy Intelligent Video Analytics on outdoor cameras, particularly when you configure certain parameters, such as a region of interest, in alerts or drawing shapes. For example, you might need to increase the maximum object size parameter because object sizes are often overestimated during periods when there are long shadows.

1.6.3 Bright lights

Bright lights can cause blooming and glare. These effects interfere with tracking, color classification, and other processing. Lights that reflect off objects at night can cause false events. This effect is especially a problem when rain makes objects shiny.

1.6.4 Obstructions

Large obstructions can break the track of a single object into multiple short segments. Such short tracks might be ignored by the system, or might be ignored in searches.

1.6.5 Night scenes

Analytics can function on night scenes to a greater or lesser extent, depending on the lighting and the camera setup. Objects are detected to some extent, depending on the underlying detection methodology, but headlight bloom can cause significant problems, and details like color might not be apparent to a human observer.

1.6.6 Dirt or dust on the camera lens

Dirt on the camera lens can obscure or blur objects and reduce contrast. As a result, the image quality is poor. If this problem is prevalent on important cameras, stress to the operators that performance is affected if the lens is not kept clean.

Chapter 2

Literature Review

Numerous route frameworks have been proposed, which depend on various methodologies and join various advances. Customarily, the visually impaired utilize guide pooches OR white sticks to assist them with moving around. However, they are constrained to move just around well-known spots [2] and can see little data. Ultrasonic-based route frameworks [3] utilize ultrasonic to gauge the separation to the objects and show the data to the visually impaired via sound or voice. In any case, these frameworks can't identify the specific area of obstructions because of the wide bar edge of ultrasonic. In addition, these frameworks can't perceive the sort of impediments (for example a vehicle or a bike). The vast majority of the business items for giving versatility help depend on GPS [4]. Be that as it may, the exactness of GPS isn't solid when working inside or in urban communities with tall structures because of constrained satellite gathering [5]. Additionally, these items likewise can't decide the snag in the close encompassing of the visually impaired [6]. A few restrictions can be defeated utilizing computer vision dependent frameworks. For instance, [7] actualizes ego-motion tracking indoors by using visual-inertial sensors. [8, 9] with the help of Kinect which we know contains the IR profundity sensors can recognize the objects all the more precisely. In [10] a stereo vision framework that gauges a 3D map is presented that during navigation is utilized to recognize objects. These frameworks can give the progressively the exact area of the visually impaired and the hindrances. In any case, enough keenness isn't in them to mention the difference between an object and a man to the visually impaired.. [11] utilizes outer GPS recipient to improve the finding precision, joins a current Intelligent Transportation System to help the visually impaired in going across the street and utilization Text-to-Speech engine introduced on

a cell phone to associate with the visually impaired. Be that as it may, this framework must be applied outside. A multi-sensor combination based route framework is displayed in [15]. It utilizes GPS and vision to actualize outside routes, use Wi-Fi and vision to execute indoor routes and use RFID to distinguish the tourist spots set in the ground and in this way perceive the spot. Despite the fact that this framework can give data about points-of-interest (POI), this framework despite everything has impediments because of the use of GPS and RFID as previously mentioned. This paper proposed a cloud and vision-based route framework. The proposed framework uses a cell phone to cooperate with the visually impaired through voice, to transmit the voice and the video caught by the stereo cameras to the distributed computing stage through Wi-Fi or 4G portable correspondence. The distributed computing stage incorporates discourse recognition, global way arranging, route, object identification, and recognition.Navigation: right off the bat, the visually impaired advises the cell phone where he needs to go and the cell phone sends this voice data to the cloud stage. Next, the stage calls the discourse acknowledgment module to change over the voice to text, the worldwide way arranging module to produce a worldwide way, which can be seen through the web application.

Chapter 3

System Requirements

In this project, two types of data are being used and collected. Both of them are used for different methodologies.

3.1 Funcational Requirements

3.1.1 Open Camera

- User shall open the camera.

3.1.2 Capture Image

- System shall capture the image.

3.1.3 Pre-processsing Image

- System shall pre-process the image.

3.1.4 Detect Object

- System shall detect the object.

3.1.5 Generate Audio

- System shall generate the audio

3.1.6 Output Audio

- User shall listen the output audio to act accordingly.

3.2 Non-Functional Requirements

3.2.1 Performance

- How fast does the system return results? How much will this performance change with higher workloads?

3.2.2 Availability and Reliability

- How often does the system experience critical failures? and how much time is it available to users against downtimes?

3.3 Use Case

Use case diagram of our system object detection and sound recognization for blind person is given below:

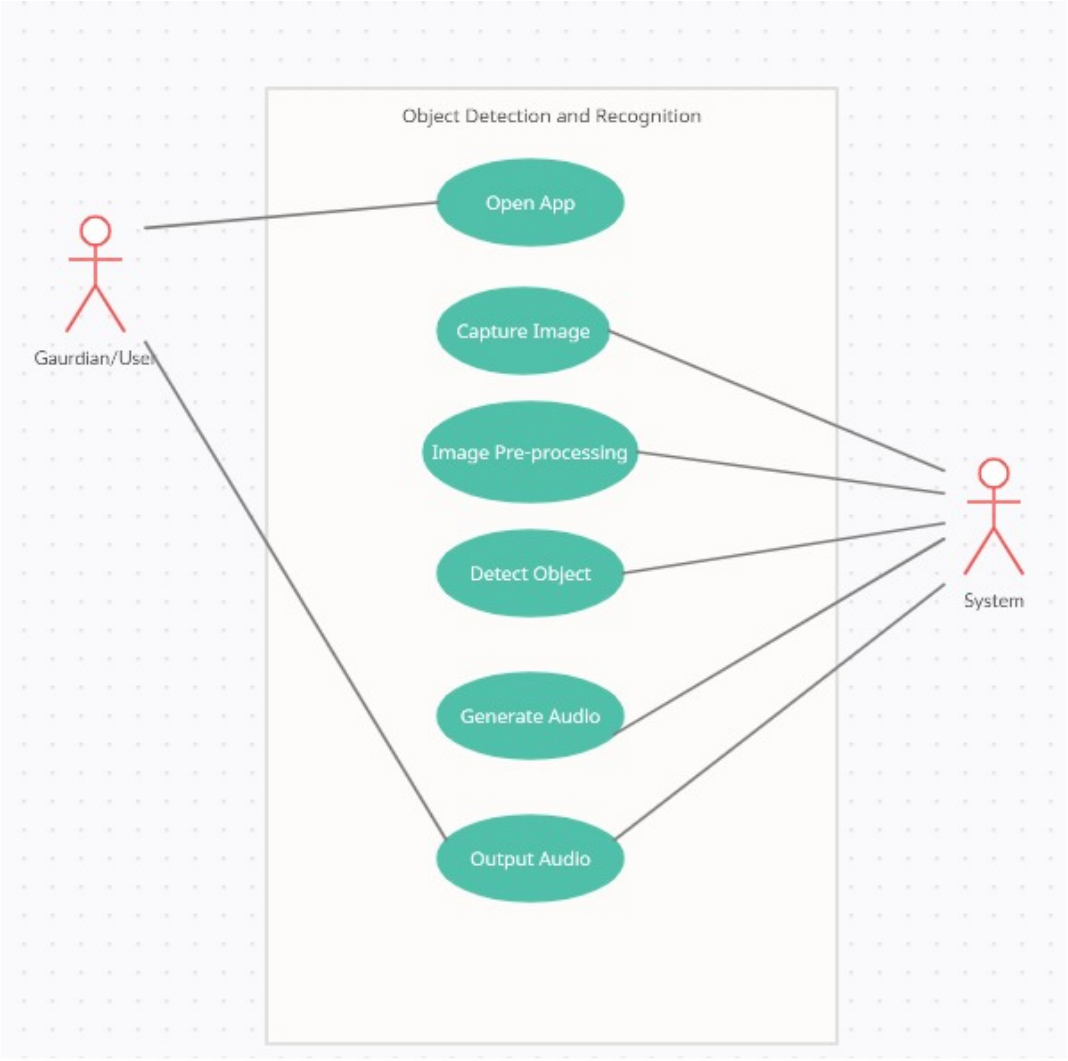


Figure 3.1: Use Case for Object Detection

3.4 Sequence Diagram

Sequence diagram of our system object detection and sound recognization for blind person is given below:

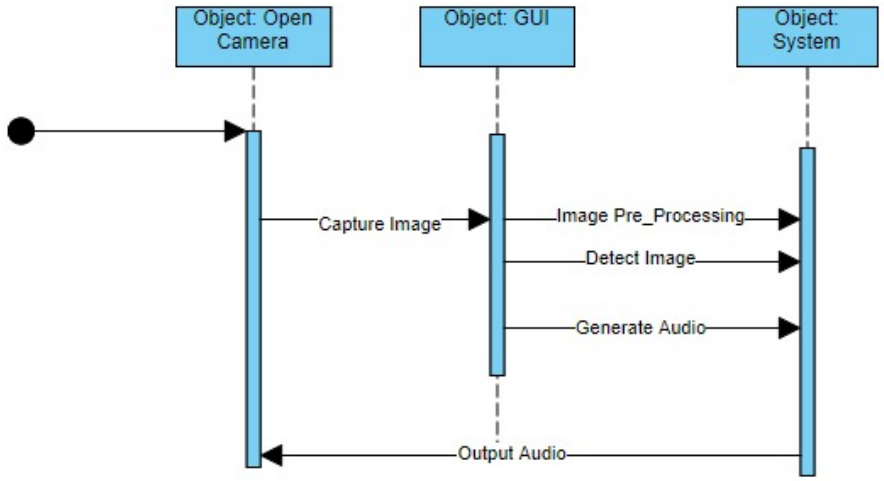


Figure 3.2: Sequence Diagram for Object Detection

3.5 Module Diagram of System

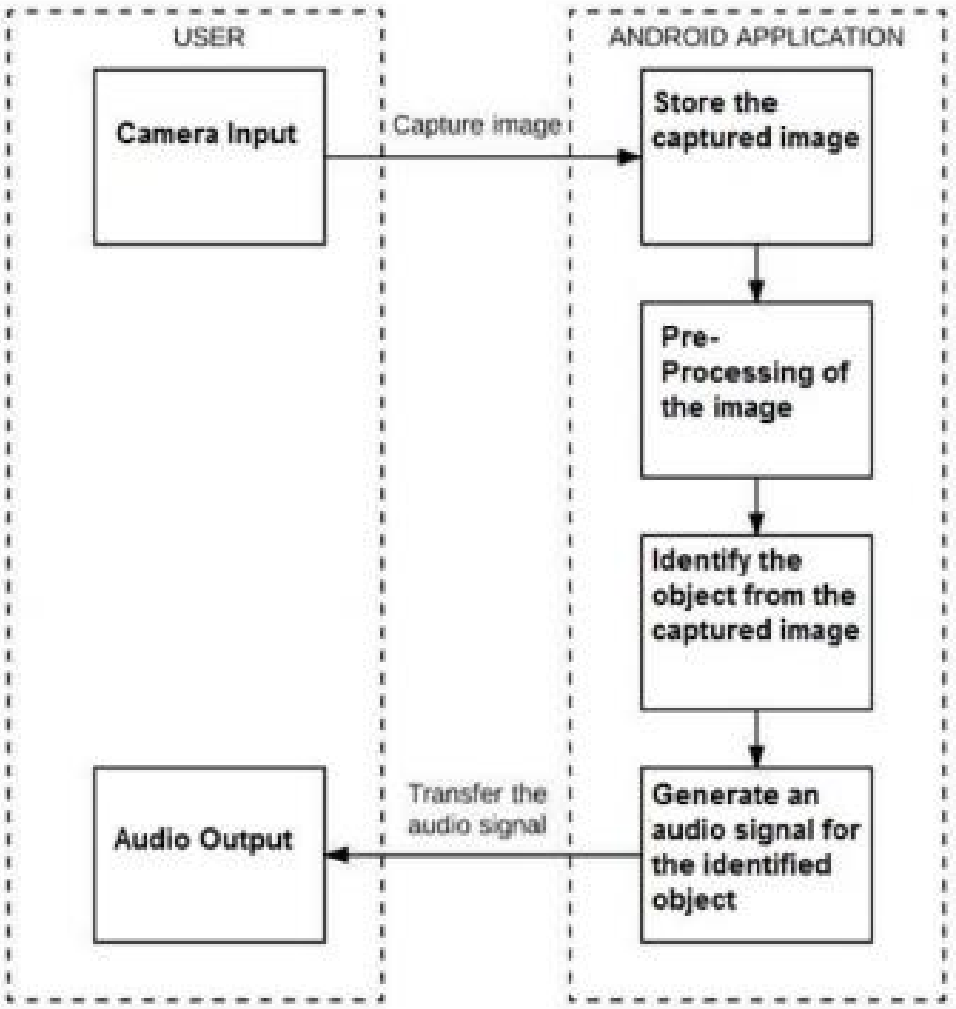


Figure 3.3: Module of System

3.6 Dataset 1

This data is collected from the Flicker, termed as Flicker 8k dataset [16]. This set was prepared and labeled by the author especially for the use of Image captioning. It is open-source data and available for use in projects. The more definitive description is available in the paper "Framing Image Description as a Ranking Task: Data, Models, and Evaluation Metrics" from 2013 [16]. According to the author:

"We introduce a new benchmark collection for sentence-based image description and search, consisting of 8,000 images that are each paired with five different captions that provide clear descriptions of the salient entities and events.

...

The images were chosen from six different Flickr groups, and tend not to contain any well-known people or locations, but were manually selected to depict a variety of scenes and situations." [16]

The dataset contains two zip files, one contains 8092 images in jpeg format and the second zip file contains a number of files containing different source information and descriptions for the images files. Train/Dev/Test set distribution is already been defined which is 6000/1000/1000 respectively.

3.7 Dataset 2

TThis data is being collected manually and also from different online resources. For the doors and stairs, some part of the data is collecting manually by using a mobile camera from the indoor environment of the different places, some part from GitHub MCIndoor and the remaining part is fetched using python scrapper randomly from the google. For the chairs, we collected data from Ikea [28], Pinterest pint, and randomly from google using scrappers, and the same scrapper phenomenon was used for the remaining objects. Total data has 6000 images with 6 classes (Door, Stairs, Table, Chair, Car, and Background), and each class contains 1000 images. Data were divided into three sets, Train, dev, and test with a ratio of 70 percent, 10 percent, and 20 percent respectively.

Chapter 4

Different Methodologies Used

In this chapter, a brief description is going to be provided about the different methodologies used so far.

4.1 Performance Evaluation Metrics

State of the art evaluation metrics for supervised binary classification problems are given below:

- `True Positive (TP):` where the model correctly predicts the positive class.
- `True Negative (TN):` where the model correctly predicts the negative class.
- `False Positive (FP):` where the model incorrectly predicts the positive class.
- `False Negative (FN):` where the model incorrectly predicts the negative class.

4.1.1 Accuracy

Accuracy represents the correctly predict both the positives and negatives out of all the predictions.

Measure to evaluate how accurate model's performance is:

$$\frac{TP + TN}{TP + FP + FN + FP}$$

4.1.2 Precision

Precision represents the correctly predict the positives out of all the positive prediction it made.

Measure to evaluate how accurate model's performance is:

$$\frac{TP}{TP + FP}$$

4.1.3 Recall

Recall represents the correctly predict the positives out of actual positives.

Measure to evaluate how accurate model's performance is:

$$\frac{TP}{TP + FN}$$

4.1.4 F1-score

F1 is a combination of both precision and recall.

Provides information of both sides TN and TP.

$$2 * \frac{Precision * Recall}{Precision + Recall}$$

4.1.5 AUC-ROC Curve

AUC stands for Area under Curve and ROC stands for Receiver Operating Characteristics.

We use for the classification problems at various threshold settings.

True Positive Rate (TPR)

$$\frac{TP}{TP + FN}$$

False Positive Rate (FPR)

$$\frac{FP}{FP + TN}$$

4.2 A Caption Generation Model (VGG16 + LSTM)

In this approach, the idea was to build a caption generation model which will be able to generate different captions to assist a blind person. The dataset which we used is flicker dataset as described in the chapter 3.

In this approach we develop a model similar to the one developed by Jason Jason Brownlee in one of his blog post [17]. The model consists of two parts the one part consists VGG16 [18] and the other is LSTM [19].

4.2.1 Preparing Image Data

So in this methodology we used the pre-trained VGG16 model also non as Oxenford Visual Geometry Group, which won the imagenet competition in 2014. We have used the pre-trained model directly provided by the Keras API. We could utilize this model as a feature of a more extensive image caption model. The issue is, it is a large model, and running every photograph through the network each time we need to test another language model setup (downstream) is repetitive.So, we extract the photo features before the actual training and save them into a file. So after which we use as an interpreter for the images later on during the training. This is an optimization that will make preparing our models quicker and devour less memory.We loaded the VGG model and remove its output layer as we are not interested in the classification in this methodology but are interested in the information we get before a prediction is made.We used the reshaping built function of the Keras to reshape an image into 224x224x3. Also, we used extractfeatures() method to stack every photograph into 1x4096 vector.

4.2.2 Preparing Text Data

The dataset contains different portrayals for each photo and the content of the depictions requires some cleaning. We will do the things:

1- Lower-Case Conversion

2- Punctuation Removal

3- Small length or single characters removal

4- Removal of the words containing numbers

In a perfect world, we need a vocabulary that is both expressive and as little as could be expected under the circumstances. A little vocabulary will bring about a little model that will train quicker. For reference, we can change the spotless descriptions into a set and print its size to get a thought of the size of our dataset jargon.

So at the end we have saved the mappings of the descriptions with the images in a single file.

4.2.3 Model Development

Model development has been done into the three steps:

1- Data Loading

2- Model Definition

3- Fitting the Model

Loading Data

First of all we have to load our dataset in order give to the model. In the flicker dataset we have train, test and dev dataset. We have train set for the training, dev for the tunning and test for the model evaluation. So we have loaded the clean descriptions and the images features. Next in order to input the text data we have to convert it into the numerical data, so we create tokens for each word and map it to a unique integer. Every depiction

will be part into words. The model will be given a single word and the photograph and produce the following word. At that point, the initial two expressions of the description will be furnished to the model as a contribution with the picture to create the following word. This is the means by which the model will be prepared.

Defining Model

In this methodology, we used the "merge-model", explained by Marc Tanti, et al in his papers [20] [21] They have provided the representation of the merge model as in figure 4.1.

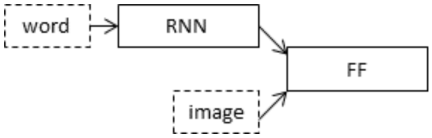


Figure 4.1: A merge-model Representation

The is being described into three parts:

- **Feature Extractor:** For this part, a 16 layered architecture model VGG-16 is used along with 'ImageNet' weights without the output layer. It outputs a feature matrix of the image.
- **Sequence Processor** This part which is a word embedding layer, responsible to handle the text data which is followed by the LSTM(Long Term Short Memory) which is an RNN layer.
- **Decoder** You can consider it a black box layer that takes outputs coming from the upper described parts which actually are the fixed-sized matrices. This layer acts as a merger and which are next being processed by a dense layer to make predictions.

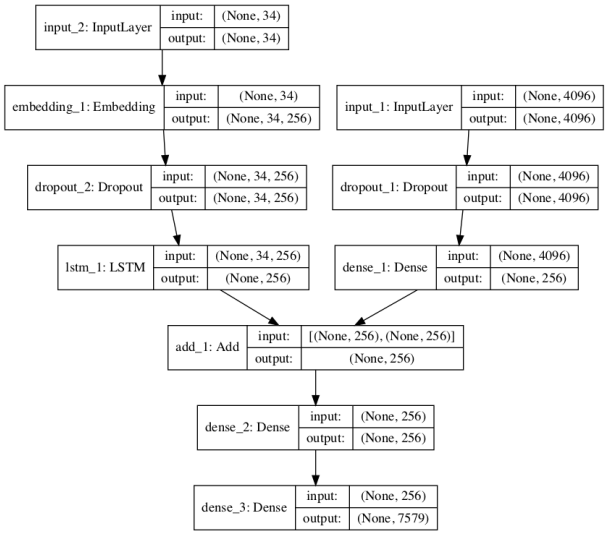


Figure 4.2: A merge-model Summary

Fitting Model

In this step, we just train our mode over 20 epochs by using checkpoint callback over Validation Loss.

4.2.4 Results

The model was trained on 306,404 samples and validated on 50,903 samples. The model was saved after 2nd epoch with a loss=3.245 and validation-loss=3.612. In order to evaluate the translated text, BLEU score [22] was calculated. NLTK library of the python was used. Following are the scores for different n-grams.

Grams	Score	weights
1	0.579114	(1, 0, 0, 0)
2	0.344856	(0.5, 0.5, 0, 0)
3	0.252154	(0.3, 0.3, 0.3, 0)
4	0.131446	(0.25, 0.25, 0.25, 0.25)

Table 4.1: BLEU Score for 4 grams

4.3 A classification model using VGG16

In this approach, we used the VGG16 model, which has already discussed in section 4.1 and used the 2nd dataset which consists of 6 classes and 6000 images (1000 classes per image). Dataset was divided into the train, dev, and test with a ratio of 7:1:2 respectively.

4.3.1 Model Development

Following steps are being used in the model development:

1. Loading Data
2. Defining Model
3. Fitting Data

Loading Data

Data were organized into the separate folders of train/dev/test and for each class. With the help of the Keras built-in data-generator object, we loaded resized (224,224,3) image data. Now our data is ready for the manipulations.

Defining Model

In order to define our model, we first load the VGG16 model with the ImageNet weights. Removed the output layer, and freeze all of the remaining ones. After this, we simply add a dense layer having 6 output with 'SoftMax' activation function which will work as a custom output layer for our model. After this, we compile our model by setting the loss as "categorical-crossentropy" and use SGD [23] as an optimizer with a learning rate of 0.0001. The model summary is shown in figure 4.3.

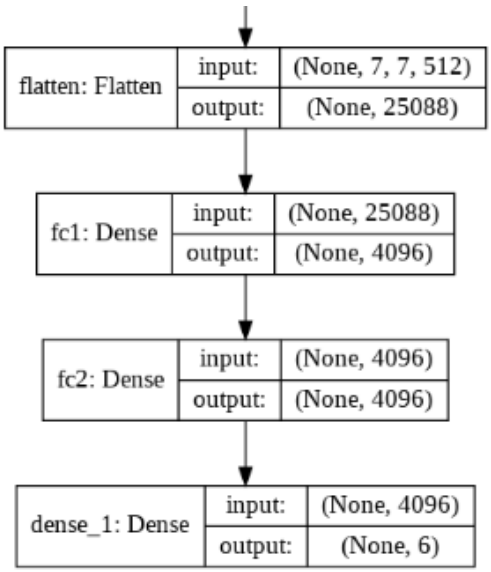


Figure 4.3: Summary of Dense layers added

Fitting Model

After defining the model, we just fit the training data into the model by using the following parameters.

- 100 epochs
- Dev-set as validation data
- 2 steps per epoch
- Callbacks
  - Checkpoint: by setting Validation Accuracy as monitor to save the best model
  - Early-Stopping: by setting patience 20 on the validation data set to stop the training if it does not improve anything

4.3.2 Results

During training, the best model on the basis of the validation accuracy was achieved on the 5th epoch. The table shows all the training and testing results followed by the graphs.

Data	Accuracy
Train	0.9531
Validation	1.0000
Testing	0.943

Table 4.2: Results of VGG16 Model

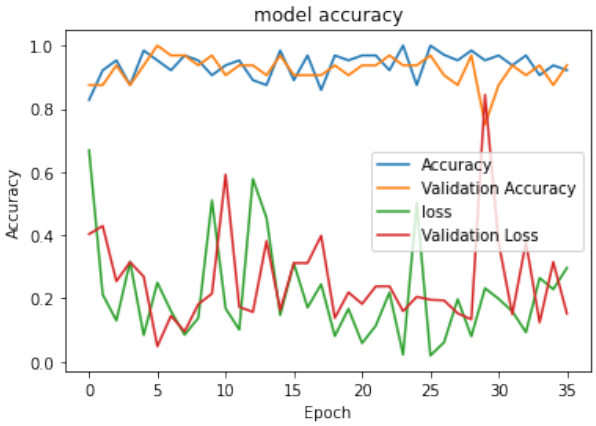


Figure 4.4: Loss and Accuracy Graph of VGG16 Model

4.4 A classification model using Mobile Net

MobileNet is a CNN model which requires very less computation power. This model is perfect for the use in mobiles. It is efficient for the use in mobiles and the devices having no GPUs, but yeah it compromises the accuracy. They reason behind the lightness of their weights is the use of depth wise separable convolution. The model is briefly described in Andrew G. Howard's paper[25]. This model was also trained using the same dataset as in VGG16. The split was also same as of 7:1:2 for the train,dev and test respectively. The same steps were also followed as in VGG16's case.

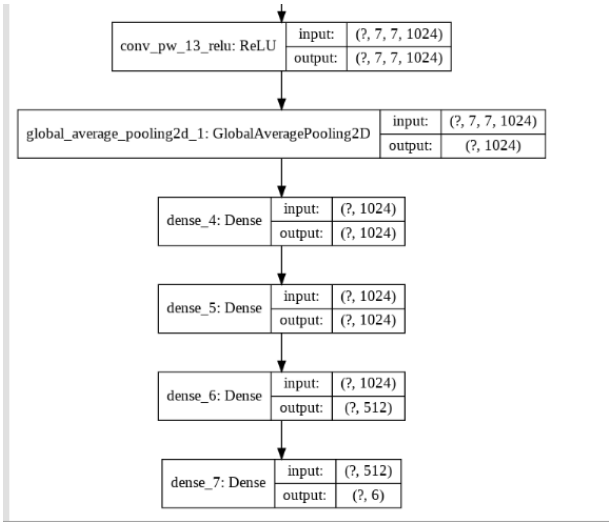
1. Loading Data
2. Building Model
3. Fitting Model

Loading Data

We will load the data in the same way as in the case of VGG16.

Building Model

First of all we load the built-in mobbilenet model from keras and removed it's output layer and freeze the remaining. Four new dense layers were added 3 with the 'relu' activation function and the one with the 'SoftMax' and made it ready for the transfer learning. We compile it by using 'SGD' optimizer and the set the learning rate of 0.001.
The part where the was modified is given in the figure 4.4. First of all we load



Modified part of MobileNet Model

Fitting Model

After defining the model, we just fit the training data into the model by using the following parameters.

- 50 epochs
- Dev-set as validation data
- 2 steps per epoch
- Callbacks

- Checkpoint: by setting Validation Accuracy as monitor to save the best model
- Early-Stopping: by setting patience 20 on the validation data set to stop the training if it does not improve anything

4.4.1 Results

During training, the best model on the basis of the validation accuracy was achieved on the 37th epoch. The table shows all the training and testing results followed by the graphs.

Data	Accuracy
Train	0.9219
Validation	0.9375
Testing	0.9208

Table 4.3: Results of MobileNet Model

4.4.2 Loss Graph

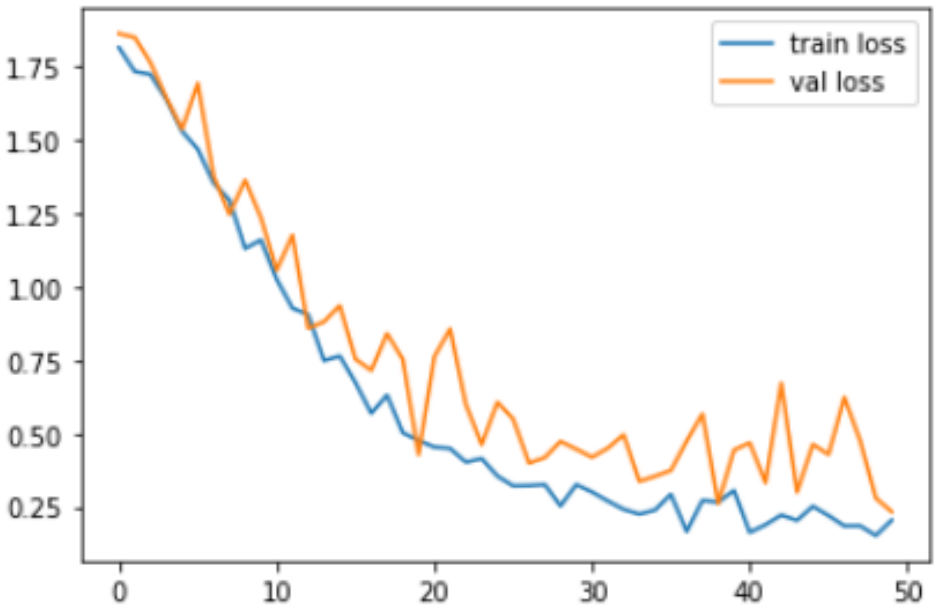


Figure 4.5: Loss Graph of MobileNet Model

4.4.3 Accuracy Graph

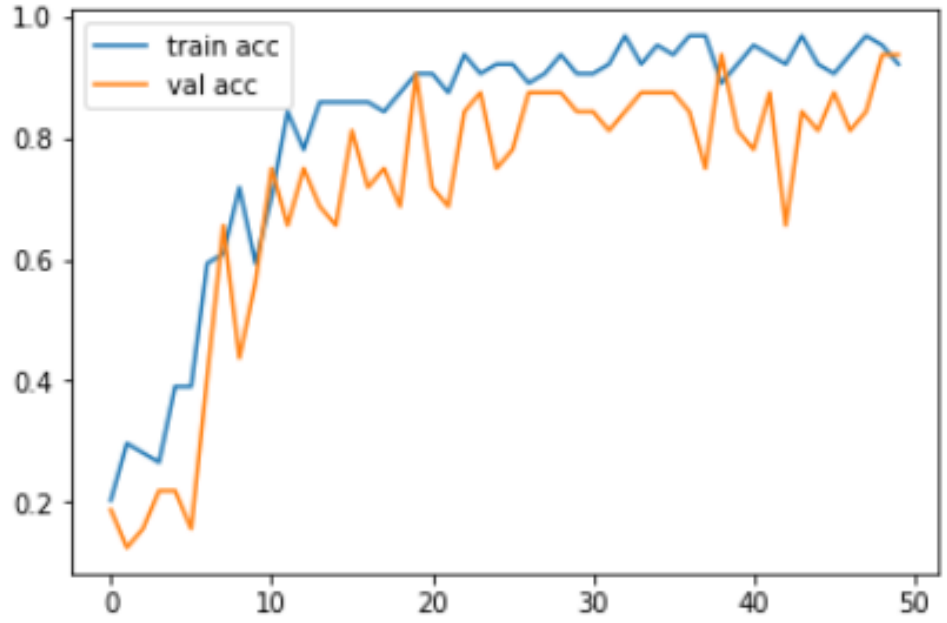


Figure 4.6: Accuracy Graph of MobileNet Model

4.4.4 Precision

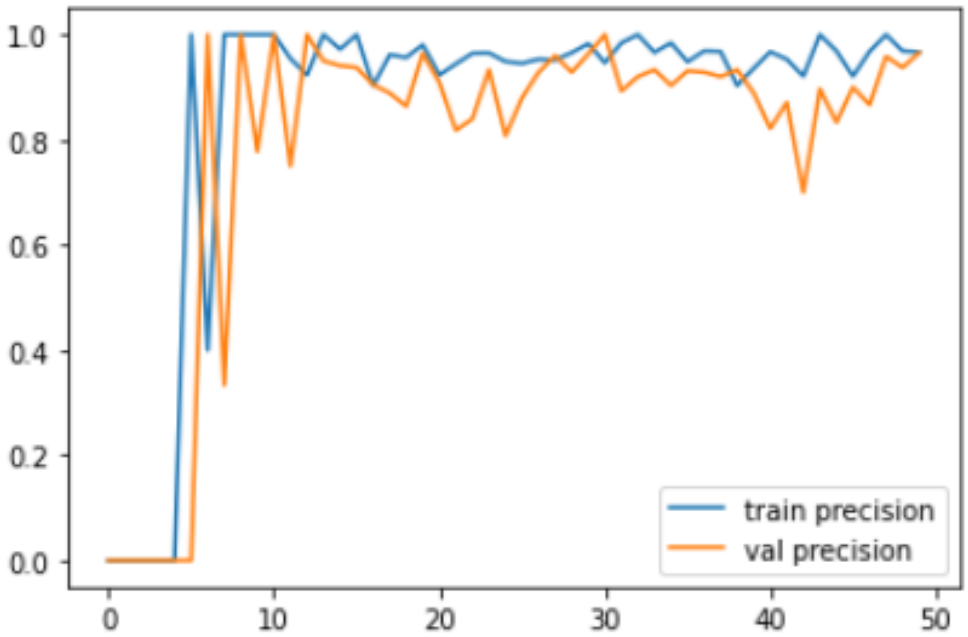


Figure 4.7: Precision Graph of MobileNet Model

4.4.5 Recall

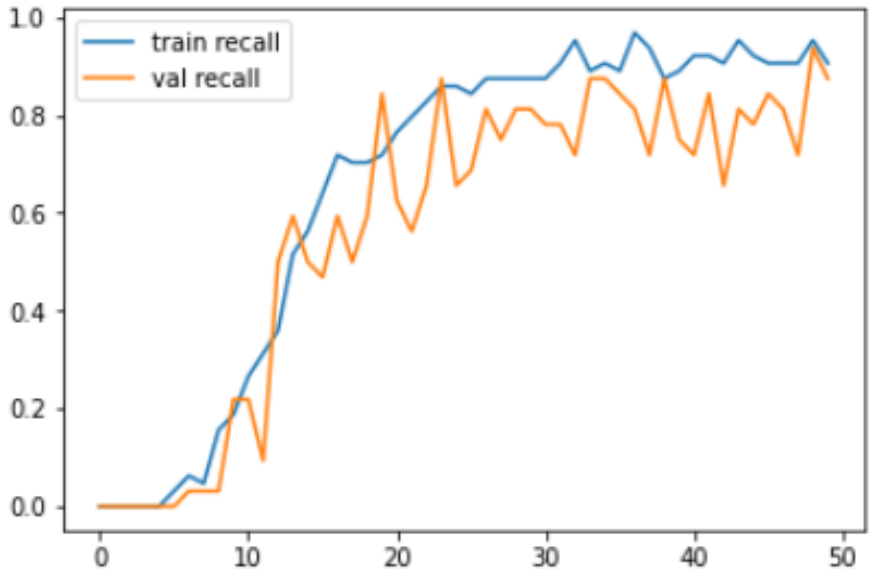


Figure 4.8: Recall Graph of MobileNet Model

4.4.6 F1-score

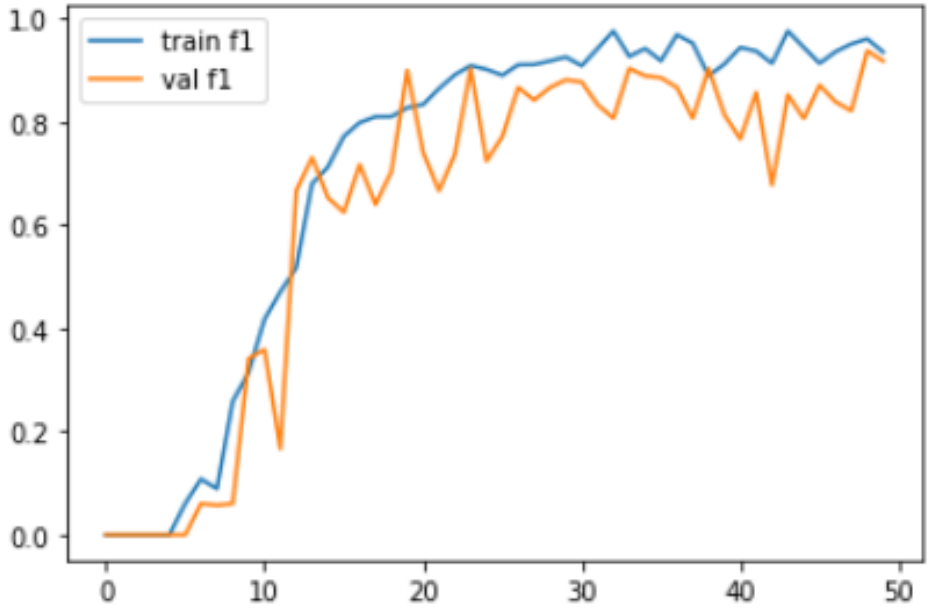


Figure 4.9: F1-score Graph of MobileNet Model

4.4.7 AUC Graph

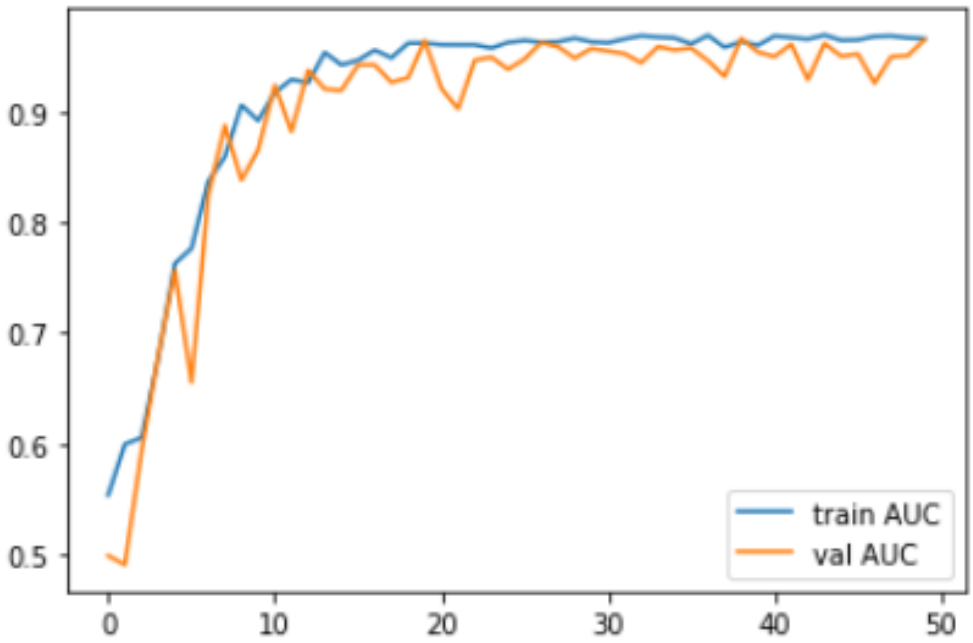


Figure 4.10: AUC Graph of MobileNet Model

4.4.8 ALL Graph

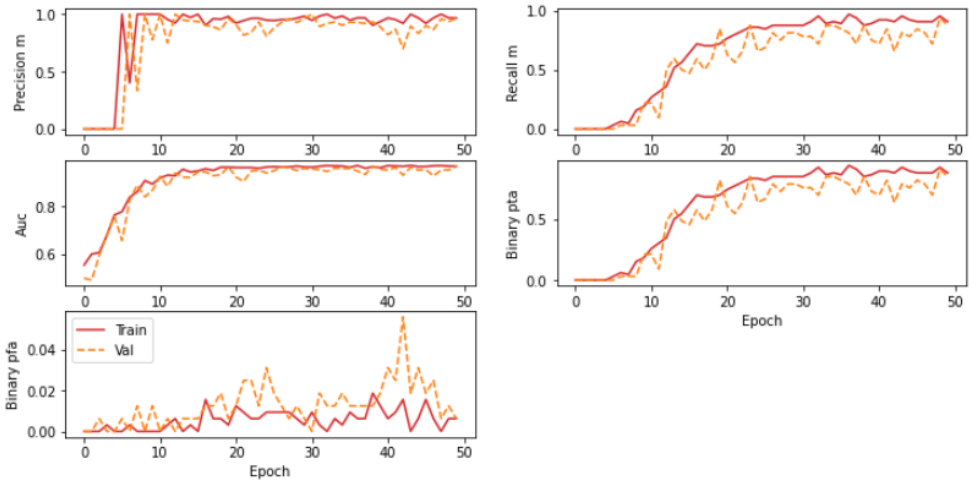


Figure 4.11: All Graph of MobileNet Model

4.5 Comparison

In this section, we will describe which model we had used in our project and why.
Let's talk about the first VGG16+LSTM merge-model. This model is good to describe a scenario in a picture but not good to alert a person. Maybe we can do in the future with different datasets but not by using the flicker's data. But yeah it can be a good integration as an entertaining part for a blind person if he wants to know what is happening in front of him.
Let's talk about VGG16. If you look at the results, it performs very well in the accuracy point of view, but it lacks speed as compared with the Mobilenet. Also, the VGG16 model weights are larger in size than the mobilenet, so it is difficult to embed it into the mobile due to its size. So, we prefer mobilenet due to it's smaller size and easy convertible by using tflite to operate with mobile applications.

Chapter 5

Mobile Appication

In order to make our model usable, we had developed an android application using the MVC design pattern as shown in figure 1.1. Our app gets the video frames and send them to the model after 2 sec of delay just to make sure the smooth completion of a cycle. Also, text to voice module of the android was used and tflite android version was used to communicate with the model.

5.1 Class Diagram

Figure 5.1 shows a class diagram of the app.

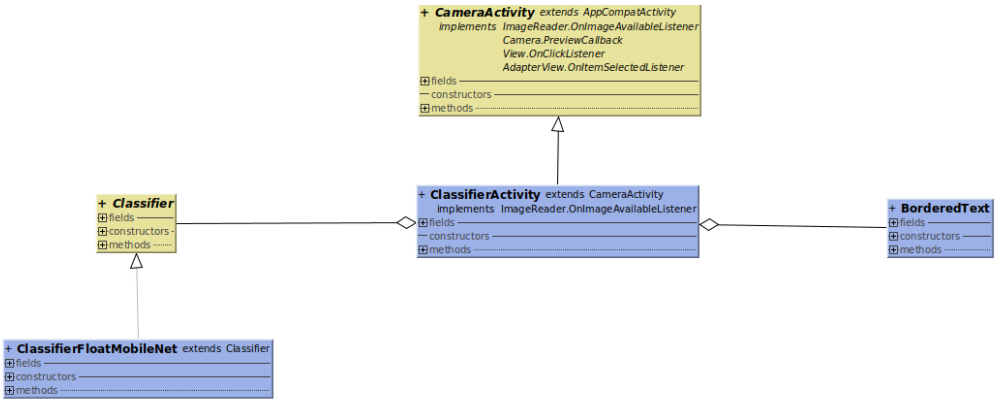


Figure 5.1: Class Diagram of the Application

5.2 Interface Diagram

The application only has single activity with a screen showing camera footage embedded in a fragment. At the bottom, label names, their confidence level, and other image Info's are also there with an option to select between models and device like CPU, GPU, etc.

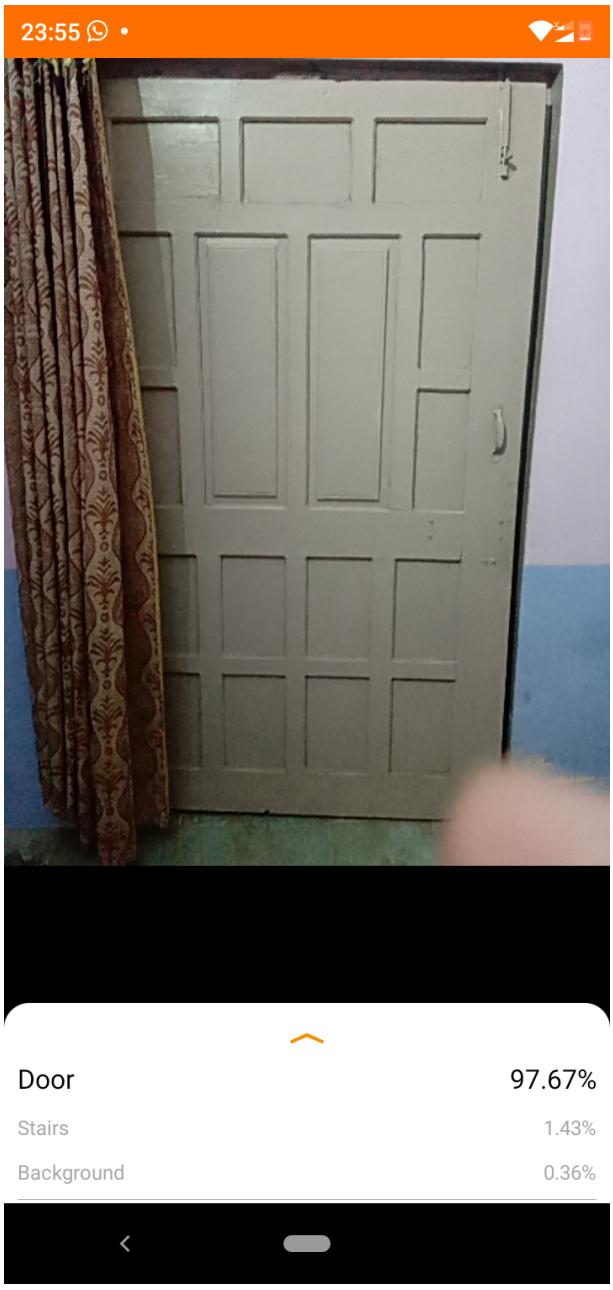


Figure 5.2: Interface Diagram (a)

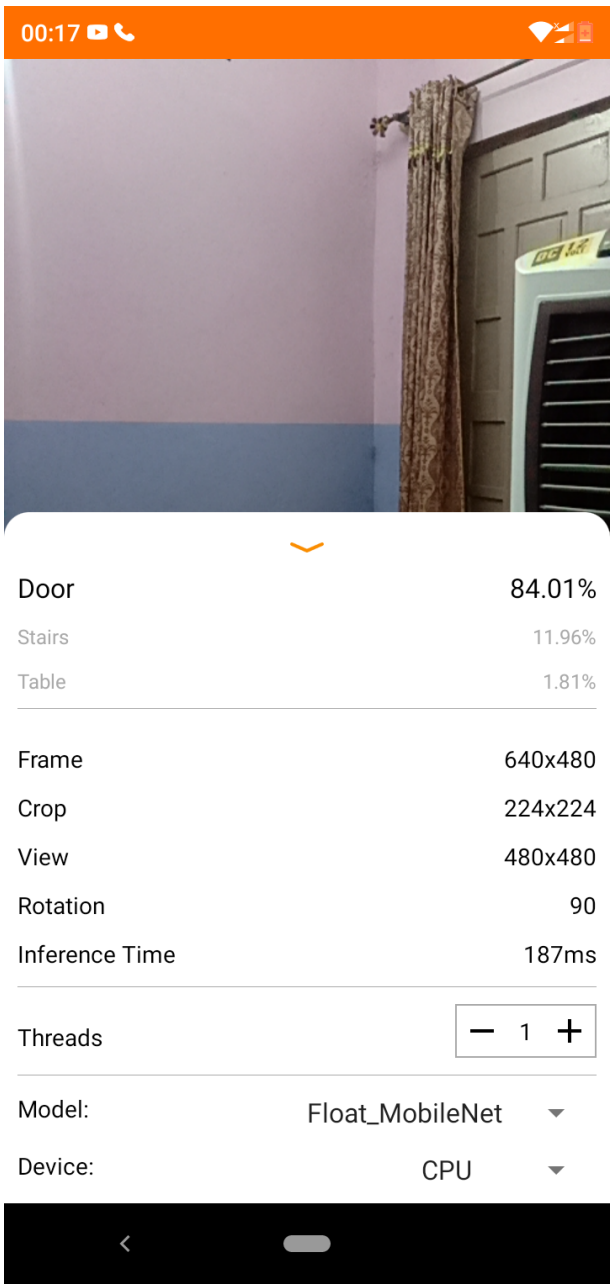


Figure 5.3: Interface Diagram (b)

Chapter 6

Summary and Reflections

So far, we have discussed different solutions to make a navigation system for a blind person. Some have scope limitation and others need high computation. In the end, we found that mobilenet, if we are using a mobile application, then it's a good one. Although, a lot of work has been done still a lot of work is remaining.

6.1 Future Work

What more innovations we can bring into this project.

1. The first thing which could be done is the collection of more quality data.
2. Increasing the states as an open door or closed door, stairs downwards, or stairs upwards and adding other objects classes.
3. What about detection instead of classification, which may be helpful in determining the direction of the object.
4. What about stereo vision or other depth sensors to measure the object distance.
5. What about the map navigation of an building using routers or access points to find directions.
6. What about dedicated hardware for this application, which should be capable of high computation if required.
7. These are the areas, where this project can be taken into. I hope so, in future we will achieve something great.

6.2 Contributions and Reflections

In this project, we have successfully classified the objects especially the case of the doors and stairs where very little has been done so far. We agree that there is still a lot of room to cover but the work which has been done so far is itself is an achievement as well. We are imagining, what people will gonna do next by making some great innovations.

Bibliography

[1] Baker A, *Blind Man is Found Dead in Elevator Shaft..* The New York Times; City Room: May 1, 2010.

[2] Syed Rizal Alfam Wan Alwi, Mohamad Noh Ahmad, *Survey on Outdoor Navigation System Needs for Blind People..* IEEE Student Conference on Research and Development, 2013, pp. 144-148. DOI=10.1109/SCOReD.2013.7002560.

[3] Yiting Yi, Lunfu Dong, *A Design of Blind-guide Crutch Based on Multi-sensors..* International Conference on Fuzzy Systems and Knowledge Discovery, 2015, pp. 2288-2292. DOI=10.1109/FSKD. 2015.7382309.

[4] R. Ivanov,*Real-time GPS Track Simplification Algorithm for Outdoor Navigation of Visually Impaired..* Journal of Network and Computer Applications, vol. 35, 2012, pp. 1559-1567. DOI=10.1016/j. jnca.2012.02.002.

[5] Michel Owayjan, Ali Hayek, Hassan Nassrallah, et al.,*Smart Assistive Navigation System for Blind and Visually Impaired Individuals..* International Conference on Advances in Biomedical Engineering, 2015, pp. 162-165. DOI=10.1109/ICABME.2015. 7323277.

[6] Ruxandra Tapu, Bogdan Mocanu, Titus Zaharia,*A Computer Vision System that Ensure the Autonomous Navigation of Blind People.* The 4th IEEE International Conference on E-Health and Bioengineering, 2013, pp. 1-4. DOI=10.1109/EHB.2013.6707267.

[7] Hongsheng He, Yan Li, Yong Guan, et al.,*Wearable Ego- Motion Tracking for Blind Navigation in Indoor Environments.*, 3rd ed. IEEE Transactions on Automation Science and Engineering, vol. 12, 2015, pp. 1181-1190.DOI=10.1109/TASE.2015.2471175.

[8] Samleo L. Joseph, Xiaochen Zhang, Ivan Dryanovski, et al.,*Semantic indoor navigation with a blind-user oriented augmented reality.*, IEEE International Conference on Systems, Man, and Cybernetics, 2013, pp.3585-3591. DOI=10.1109/SMC.2013.611.

[9] Javier Hernández Aceituno,Rafael Arnay, Jonay Toledo, et al., *Using Kinect on an Autonomous Vehicle for Outdoors Obstacle Detection.*, IEEE Sensors Journal, vol. 16, 2016, pp. 3603-3610. DOI=10.1109/JSEN.2016.2531122.

[10] Nikolaos Bourbakis, Sokratis K. Makrogiannis, Dimitrios Dakopoulos, *A System-Prototype Representing 3D Space via Alternative-Sensing for Visually Impaired Navigation.* IEEE Sensors Journal, vol. 13, 2013, pp. 2535-2547. DOI=10.1109/JSEN. 2013.2253092.

[11] Anna N. Lapyko, Li-Ping Tung, Bao-Shuh Paul Lin, *A Cloud-based Outdoor Assistive Navigation System for the Blind and Visually Impaired.* Wireless and Mobile Networking Conference, 2014, pp. 1-8. DOI=10.1109/WMNC.2014.6878884.

[12] AYOOSH KATHURIA, *How to implement a YOLO (v3) object detector from scratch in PyTorch: Part 2* https://blog.paperspace.com/how-to-implement-a-yolo-v3-object-detector-from-scratch-in-pytorch-part-2/.

[13] Bashiri, Fereshteh S. LaRose, Eric Peissig, Peggy P. Tafti, Ahmad. *(2018). MCIndoor20000: A fully-labeled image dataset to advance indoor objects detection.* Data in Brief. 17. 10.1016/j.dib.2017.12.047.

[14] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi *You Only Look Once: Unified, Real-Time Object Detection* University of Washington, Allen Institute for AI, Facebook AI Research

[15] Giuseppe Air'o Farulla, Ludovico O. Russo, Stefano Rosa, et al.,*ORIENTOMA: A Novel Platform for Autonomous and Safe Navigation for Blind and Visually Impaired.* The 10th International Conference on Design Technology of Integrated Systems in Nanoscale Era, 2015, pp. 1-6. DOI=10.1109/DTIS.2015. 7127390.

[16] Micah Hodosh, Peter Young, Julia Hockenmaier *Framing Image Description as a Ranking Task: Data, Models and Evaluation Metrics* Department of Computer Science University of Illinois at Urbana-Champaign

[17] Jason Brownlee *How to Develop a Deep Learning Photo Caption Generator from Scratch,* Machine Learning Mastery

[18] Karen Simonyan and Andrew Zisserman *Very Deep Convolutional Networks for Large-Scale Visual Recognition* International Conference on Learning Representations, 2015

[19] Sepp Hochreiter, Jurgen Schmidhuber, *LONG SHORT-TERM MEMORY* Neural Computation 9(8): 1735-1780, 1997

[20] Marc Tanti, Albert Gatt, Kenneth P. Camilleri *Where to put the Image in an Image Caption Generator* University of Malta, 2017

[21] Marc Tanti, Albert Gatt, Kenneth P. Camilleri, *What is the Role of Recurrent Neural Networks (RNNs) in an Image Caption Generator?* 10th International Conference on Natural Language Generation (INLG'17), 2017

[22] Kishore Papineni, Salim Roukos, Todd Ward, Wei-Jing Zhu, *BLEU: a Method for Automatic Evaluation of Machine Translation* 1IBM T. J. Watson Research Center Yorktown Heights, NY 10598, USA, 2002

[23] Sebastian Ruder, *An overview of gradient descent optimization algorithms* Insight Centre for Data Analytics, NUI Galway Aylien Ltd., Dublin, 2017

[24] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,* 2017

[25] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,* 2017

[26] MCIndoor20000, *a fully-labeled image dataset to advance indoor objects detection,* https://github.com/bircatmcri/MCIndoor20000

[27] Jason Brownlee *Chairs* https://www.pinterest.com/wendyshat/chairs/

[28] *Chairs,* https://www.ikea.com/us/en/cat/chairs-fu002/