Intelligent Mobility Aid for the Visually Impaired

Shantanu Deshmukh
San Jose State University
San Jose, California 95126
Email: shantanu.deshmukh@sjsu.edu

Akhilesh Jichkar
San Jose State University
San Jose, California 95126
Email: akhilesh.jichkar@sjsu.edu

Saketh Saxena
San Jose State University
San Jose, California 95126
Email: saketh.saxena@sjsu.edu

***Abstract*—Our sense of sight helps us traverse the world without much thought but when this sense is impaired even a trivial activity such as crossing the street becomes a challenge. Unfortunately, about 253 million people across the world live with visual impairment out of which 36 million are blind and require mobility aids like guide dogs and "Miniguides" to navigate [1]. While these aids are useful they are ridden with maintenance, cost issues, sub-optimal effectiveness and intuitive reliability. We propose to answer some of these challenges through an intelligent mobility aid application which serves as an inexpensive and hassle-free replacement for the guide dog. The application runs on an android smart phone with a simple User Interface(UI) and uses a Convoluted Neural Network(CNN) to classify images with obstacles. It detects the presence of obstacles such as benches, newspaper vending machines, bike riders, trash cans, and approaching people through image recognition using the phone's camera and outputs an auditory warning to the user. The application was trained using photographs of streets containing the obstacles mentioned above which we procured from the Internet and captured ourselves. Our applications assessment is based primarily on the confidence of classifying an image as containing an obstacle or not, the accuracy in classifying an object as an obstacle and the percept to actuator output latency. We aim to make an accessible mobility aid which people can use on their mobile phones which provides high usability and virtually no cost.**

1. Introduction

Visual impairment is a discomfiting disability affecting nearly 3% of the people in the world out of which 13% are blind [1]. For a visually impaired person, accessibility and mobility are pertinent challenges yet to be sufficiently resolved. The mobility aids which are widely used nowadays include the white cane and the guide dog, but these aids have to be maintained and are not fully reliable. For example, guide dogs are amazing navigators but they possess dichromatic vision and hence cant perceive red, orange, green and yellow objects solely on the basis of color [2]. This would make it extremely difficult for guide dogs to differentiate traffic signals. They typically require a lot of training and are expensive costing upto $16000 to train and $30 per month to maintain [3]. Also, dog is fatigued from being on duty all the time. There already exist a few technological mobility aids such as Miniguide, IOCane, PAM-AID, and Seeing-AI. These mobility aids are similar in principle to what our system proposes, but differ in the implementation. Most of these mobility aids require additional hardware and special training to use. We describe how each of these aids work in the succeeding paragraphs along with their advantages and shortcomings.

Miniguide
The Miniguide is a secondary navigation aid which uses ultrasonic echolocation to detect obstacles. It is a tiny hand-held device with sensors which vibrates to indicate presence of objects nearby and the distance to obstacles. While it may seem very useful as a standalone device it is described as a secondary aid to augment information provided by the primary aid and help in creating a better sketch of the environment around the visually impaired [4]. Miniguide has very specific use cases and it requires training and as well as time to adapt. The other drawback is that Miniguides are quite expensive and currently cost about $499.

IOCane: A Smart-Phone and Sensor-Augmented Mobility Aid for the Blind
The IOCane is a mobility aid for blind cane users. It is a lightweight plug and play device, which can be mounted on a white cane. IOCane uses ultrasonic sensors to collect environmental data about the height and proximity of obstacles and sends feedback to the mobile phone which then vibrates and provides auditory feedback to alert the user [5]. The IOCane is also designed to require minimal training but it is not commercially available and comes with the additional responsibility of carrying and maintaining the sensor.

PAM-AID
PAM-Aid (Personal Adaptive Mobility Aid) is a robotic agent that is designed for aiding the elderly and inform visually impaired persons in navigating indoors. It is essentially a robotic walker for the elderly which helps them navigate by avoiding obstacles and provides support

[3]. Based on our understanding, this product is developed for a specific niche of elderly persons who require assistance throughout the day and isnt suitable for navigating through streets. The PAM-Aid robot is more suitable for indoor needs whereas this application is designed to be used to avoid obstacles while walking in the streets.

Seeing-AI
A closely related major development in the area of mobile accessibility aids for the visually impaired, has been made by Microsoft's Seeing-AI, an iOS application which narrates the environment around a person. The 'Seeing AI' application has a broad focus and touches upon many areas besides mobility, it uses computer vision and the application interface is straight-forward where the user points the mobile camera at some text and the application reads it out using text to speech conversion. Seeing -AI has 6 channels to detect short text, currency, documents, products, persons and scene. The application can identify a person if the camera is pointed at someone who is known to the user and can describe what is happening in the scene [6]. It is not a specific mobility aid, it is more of an accessibility aid and it offloads all its heavy image processing work to the cloud. Hence, it requires an active internet connection to work. Our application is focused on creating a highly usable solution to assist the visually impaired with navigation and is a standalone application which does not require an internet connection.

While all the described applications aim at meeting the same goal of making the world easier to traverse for the visually impaired, they all have their advantages and disadvantages. Our application differs fundamentally in the aspects of the technology and hardware used. Our goal is to build an intelligent mobility aid for the visually impaired, and employs extensive use of various technologies, which are combined to form a seamless integration of the following: sensing image data from real world surroundings, processing it to detect objects, classifying images as obstacles, and sending an auditory and haptic output to the user all in real time using only a mobile phone. Our application works on any android device. We have selected the following entities as obstacles on a path for the purpose of our project-people, benches, trash cans, newspaper vending machines and bike riders.

The proposed architecture of the system is to use a cane but with wheels attached to the bottom to mount the mobile device. Unlike the IOCane model which uses a traditional white cane, our trolley cane has a provision to mount a phone at the top and wheels to stabilize the camera. The user would be able to roll the cane ahead while walking. The application then clicks images of the path and sends them to a Convoluted Neural Network model created using TensorFlow. The model acts as a black box which takes the images as input and returns the probability values for the obstacles which might be present in the image. In case the probability of a particular obstacle is higher than a perceived threshold the application triggers the phone to vibrate and produce beeps to alert the user of a nearby obstacle. The next few sections describe in detail the materials and methods used to build the application, the experiments that will be performed to test the application and the various datasets used.

2. Materials and Methods

Following technologies are used for this application:

1. Tensorflow [7]: Tensorflow is an open-source symbolic math library used for machine learning applications such as neural networks. It was designed to operate at a large scale and in heterogeneous environments. Tensorflow lite has been optimized to run in environments with limited computing power such as smart phones. It provides the full functionality of machine learning concepts on devices with limited resources. Hence, Tensorflow lite is used in this application. Docker Toolbox is used to install Tensorflow on Windows. Docker Toolbox creates an environment that meets all the requirements for installing Tensorflow. Once this environment is created, the Convolutional Neural Network (CNN) libraries are pulled from GitHub [7] into the Tensorflow container. CNNs are a category of Neural Networks, inspired by the connectivity patterns between neurons in the visual cortex of the brains of organisms. CNNs are used in this application since they have proven very effective image recognition and classification [8]. CNNs have been successful in identifying faces, objects and traffic signs apart from powering vision in robots and self driving cars [9].

CNNs can have a million of parameters, and training them can take weeks of time. Transferring training is a technique used to shorten this time. Tensorflow out of the box, comes with a few fully trained models like the ImageNet, Inception, Mobilenet, etc. Mobilenet is specifically optimized for smartphones. Mobilenet models have a much lesser size, but has lesser accuracy than the Inception models. All these models are implemented by Google in GoogLeNet, a 22 layers deep neural network used by Google in its products [10]. In our application, Mobilenet CNN model is re-trained on a set of images of obstacles which include benches, trash cans, bike riders, approaching people, and newspaper vending machines. The visualization of this Mobilenet Architecture having our trained layer can be seen in Figure 1. The trained CNN model is stored as Protocol Buffer(.pb) files, which consists of the neural network and its constituent weights.

2. Android: Android is the most commonly used mobile device worldwide. In addition to this, Android has a vast development toolkit and has features like Voice Search, TextToSpeech which can be leveraged to develop powerful applications. Also, both Tensorflow and Android are developed by Google, hence there is support for Tensorflow on Android. Android natively uses C++, upon which a Java Software Development Kit (SDK) runs. Tensorflow is also

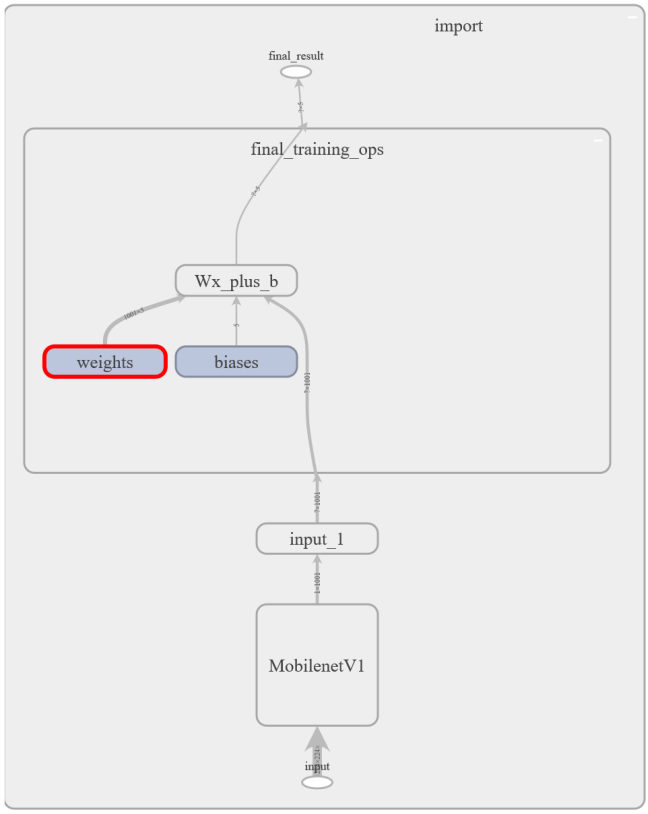

Figure 1. Overview of the MobileNet Architecture.

natively written in C++. Hence, Native Development Kit (NDK) is installed on Android, which communicates with the files imported from Tensorflow. Android Studio is used to create an Android Application on Windows environment. The Protocol Buffer file generated by Tensorflow is used as a blackbox classifier in the application, by importing it into the Android Application. A basic Android Application is created which captures images and uses the trained model to classify objects as obstacles.

3. DownThemAll Batch Download: DownThemAll is an add-on for Mozilla Firefox browser, which is used to download multiple images at once. DownThemAll was used for batch downloading images from Google Image Search for training the CNN. The downloaded images were curated to ensure that each image content matches the obstacle label assigned to the image and that each obstacle class has approximately equal number of training instances, that is, approximately 200 images per obstacle. Table 1 shows the information about the downloaded images.

The work flow of the application is as follows:

1) Once the Tensorflow environment is setup and the Mobilenet model is pulled from GitHub, a custom Python script is used to re-train the classifier at the last layer.

TABLE 1. COUNT OF DOWNLOADED TRAINING IMAGES

Image Type(Obstacles)	**Count**
Bench	200
Bike Rider	200
Trash Cans	144
Newspaper vending machine	200
People	200
Total	944

2) The training data set is transfered to the Tensorflow environment. The Python script re-trains the classifier. The parameters required for training include number of steps, bottleneck directories, model directory, output labels file path, output graph file path, and image directory. The Python script saves the output graph(classifier) and output labels to the specified directory.
3) Once the classifier is trained, it is transferred to the Android environment and the images captured by the camera are tested with the classifier. TensorFlowInferenceInterface is used in the android application to communicate with the classifier. If an object is detected with a probability of more then 90% it is considered as an obstacle
4) The user is then alerted about the obstacle in the form of auditory feedback.

2.1. Environment

The activity diagram representing the architecture diagram for the proposed application is shown in Figure 2.

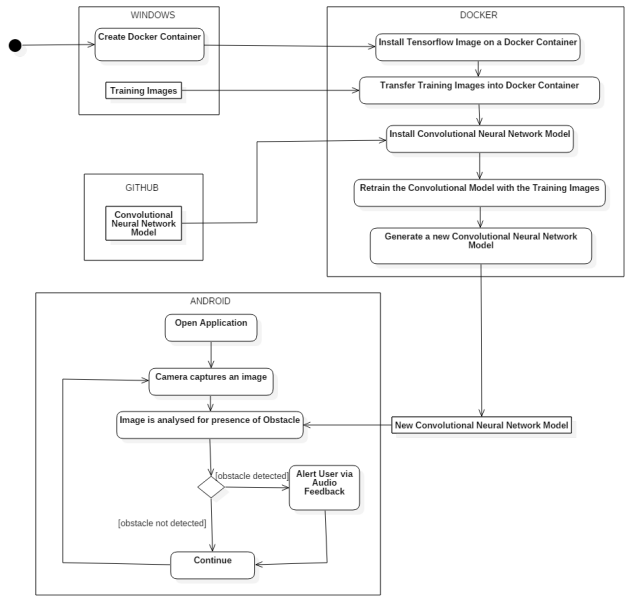

Figure 2. Activity Diagram representing the architecture of the Application.

The agent will operate in the real world environment, typically on pavements, streets and intersections. For the

purpose of this project, we have narrowed down the obstacles to the trash cans, benches, newspaper vending machines, bike riders and approaching people.

2.2. Sensors

The android devices camera is the sensor, it regularly captures images and sends to the tensorflow model for classification.

2.3. Percepts

The captured Jpeg image is converted to an android.graphics.Bitmap object and is passed on as a percept for classification.

2.4. Agent Function

The mobilenet based classifier model trained on images of obstacles acts as the agent, and is used to classify each of the Bitmap image percepts. The classifier returns a 'Recognition' object, which has the list of all the labels along with the confidence value of class prediction. The list is in a descending order, with the most confident label at the top.

2.5. Actuators

Once the obstacle is recognized by the agent and if its accuracy is more then 0.90 or 90%, it's label will be converted to speech and the user will be altered of the obstacle.

2.6. Performance Measures

The model is assessed on two primary performance measures to evaluate the agents performance.
Firstly, the accuracy of the agent in detecting the obstacle, which is calculated as follows:

$$accuracy = \frac{TP + FN}{TP + TN + FP + FN} \times 100$$

where TP is number of true positives(correctly predicted images), TN is the number of true negatives, FP is number of false positives and FN is number of false negatives.

Secondly, the confidence interval in detecting an obstacle. It is calculated as the summation of values obtained at the last layer of the CNN model for each label/class in the Fully Connected Dense Layer. It is a softmax function defined by the formula [11]:

$$p_j = \frac{exp(x_j)}{\sum_k exp(x_k)}$$

where p_j represents the class probability, x_j and x_k represents the total inputs to the units j and k of the same level respectively. This value is internally calculated by the mobilenet model and only when the confidence of a recognition is more then 90% we are considering it as an obstacle and giving an auditory feedback.

3. Experiments

3.1. Training Data Sets

Training data was obtained by batch downloading images from Google Image Search using DownThemAll Batch Download. A total of 1281 images were downloaded out of which 944 images(Table 1) were shortlisted for training. The images were carefully curated such that no image is misclassified in the training data. Also, these images were selected in such a way that the background of the obstacle differed greatly, such that the classifier is not biased on any other feature other than the features of the obstacle. Also, it was made sure that the images contained views of the obstacle from different angles.

3.2. Validation Data Sets

When Tensorfow is provided with Training data, Tensorflow randomly divides it into 3 parts. The first part, constituting of 80% of the images, are used for training the model. The second part consisting of 10% of images are used as a validation set. The remaining 10% are used for testing the accuracy of the trained model. All these steps are implemented by Tensorflow and we get our final output as the accuracy. [12]

3.3. Test Data Sets

Testing was done in two stages. Firstly, a test data set (Test Data Set 1) was obtained by downloading 125 images from Google Image Search. The test images consisted of 25 images for each obstacle. These images would be opened on a computer and our application would classify the images by capturing the image on the computer screen. This would help expand our testing set, but we may not get accurate results since the image is distorted by capturing angle, pixels, zoom levels, etc. To overcome this, another test dataset (Test Data Set 2) was created consisting of a set of images captured in real life scenarios. This dataset contains 15 images, where each class has 3 images each.

4. Results

The images in both the Test Data sets are tested on the Trained Classifier. Machine used for training had specifications as shown in Table 2 Time required for training the classifier: 285.13 seconds.

On testing the classifier with Testing Data 1, we obtain an accuracy of 90.9%. Also, it is observed that the classifier has difficulty in classifying people and benches. As depicted in Table 3 and Table 4

TABLE 2. SPECIFICATIONS OF MACHINE USED FOR TRAINING

Operating System	Ubuntu 16.04 LTS
CPU	AMD A10-4600M 64-bit
RAM	6.00 GB

TABLE 3. TEST PREDICTION ACCURACY ON DATA SET 1

Obstacles	**Number of Correct Predictions**	**Prediction Accuracy**
Bench	18	72.0%
Bike Rider	21	84.0%
Trash Cans	21	84.0%
Newspaper vending machine	25	100.0%
People	15	60.0%

TABLE 4. CONFUSION MATRIX FOR DATA SET 1

	Obstacle Detected	**Obstacle not Detected**
Obstacle Present	100	25
Obstacle Not Present	25	400

TABLE 5. TEST PREDICTION ACCURACY ON DATA SET 2(REAL LIFE SCENARIO)

Obstacles	**Number of Correct Predictions**	**Prediction Accuracy**
Bench	3	75.0%
Bike Rider	3	75.0%
Trash Cans	4	100.0%
Newspaper vending machine	4	100.0%
People	3	75.0%

TABLE 6. CONFUSION MATRIX FOR DATA SET 2

	Obstacle Detected	**Obstacle not Detected**
Obstacle Present	17	3
Obstacle Not Present	3	68

On testing the classifier with Testing Data 2, we obtain an accuracy of 93.4%. This might be because the application is receiving a much clearer image and is able to classify much better, as shown in Table 5 and Table 6. It might also be the case because the testing data is not quite large.

5. Conclusion

In summary, we have developed an intelligent android application which detects the five obstacles as mentioned above in the path and alerts the user. Our application is lightweight and can be improved upon to extend its functionality. As per our results our model has a relatively high accuracy which can be further improved for particular obstacles by retraining the model with a more comprehensive dataset. Thus, we have successfully created a lightweight, portable, inexpensive and intelligent solution to aid the visually impaired with navigation.

6. Future Scope

We intend to improve the application by implementing multiple obstacle detection if more than one obstacle is present in one frame. The effectiveness of the application can also be enhanced by calculating an approximate distance of the obstacle from the user and alerting the user accordingly.

References

[1] WHO, "Vision impairment and blindness fact sheet."

[2] J. Neitz, T. Geist, and G. H. Jacobs, "Color vision in the dog," *Visual neuroscience*, vol. 3, no. 2, pp. 119–125, 1989.

[3] G. Lacey and K. M. Dawson-Howe, "Personal adaptive mobility aid (pam-aid) for the infirm and elderly blind," in *AAAI Fall Symposium, November, Boston, USA*, 1996.

[4] G. Phillips, "The miniguide ultrasonic mobility aid," *GDP Research, South Australia*, pp. 606–614, 1998.

[5] B. Leduc-Mills, H. Profita, S. Bharadwaj, and P. Cromer, "iocane: A smart-phone and sensor-augmented mobility aid for the blind," 2013.

[6] "Medgadget: Microsoft seeing ai app for blind people describes the world around," July 2017.

[7] Tensorflow.org, "Tensorflow."

[8] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[9] U. Karn, "An intuitive explanation of convolutional neural networks."

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[11] Wikipedia, "Artificial neural network."

[12] Tensorflow.org, "Tensorflow validation."