Bangla Voice Recognition Based Interaction Tool at Public Places for Visually Impaired People

Md. Asif Iqbal
ID: 1606001
Sec: A1
Dept: EEE
Level: 3 Term:1

Mir Sayeed Mohammad
ID: 1606003
Sec: A1
Dept: EEE
Level: 3 Term:1

Dipon Paul
ID: 1606005
Sec: A1
Dept: EEE
Level: 3 Term:1

Sohan Mahmud
ID: 1606007
Sec: A1
Dept: EEE
Level: 3 Term:1

Azizul Zahid
ID: 1606033
Sec: A1
Dept: EEE
Level: 3 Term:1

***Abstract*—The main objective of this project is to build a voice based user interface for visually impaired individuals which can be used in any public places like restaurants, libraries etc. It should be noticed that this kind of user interface software is available for English speaking people but not much work has been done about using it for Bangla speaking people. So, our main interest was to enable the visually impaired people in Bangla speaking areas to communicate in any public place without knowing English.**

Keywords—Convolutional Neural Network (CNN), Spectrograms, Discrete Fourier Transform (DFT)

I. Introduction

In order to build a voice user interface using Bangla language, our main task was to develop a Bangla speech recognition system. Though, many works have been done in speech recognition but most of them are for English language. However, we chose to use Convolutional Neural Network (CNN) as a basis for our speech recognition algorithm. To use CNN, we needed a lot of sample data of Bangla speech. But, unfortunately, there was no available dataset in the internet which we can use. So, we planned to collect the dataset ourselves. We collected the speech data from most of the students of our batch, then processed the data and used them for training our neural network. After that, we built an user interface which can interact with the user through audio commands.

II. Data Collection

For data collection, we used simple microphone. Each student is asked to say their roll number digit by digit and then say the digits from zero to one repeatedly. Each student's sample was 40 seconds long.

Additionally, to make our neural network more robust, we needed to train it with noise signal too, so that it can distinguish between actual speech signal and noise signal. So, we recorded some random noise at random places and used them as samples to train the network.

III. Data Processing

After we have collected the data, we had to make the collected data suitable for training the neural network. Firstly, we had to crop the different digit from the audio files that we have collected. But, these were of different lengths. So, we padded zeros to make them of same length. It was possible to do because adding zeros to a signal does not change its frequency spectrum.

IV. Calculating Spectrograms

Spectrograms are a very efficient way of representing speech signals. Specially, in neural network applications, speech signal are nearly always represented using spectrograms.

Spectrograms are a two dimensional array more like an image. To compute spectrograms, the signal is first segmented into chunks of very small time lengths like 20ms. Then, for each of this segments, DFT is done. Then, an image is formed in which the horizontal axis represents time, the vertical axis represents frequency and the color of a point represents the power of the corresponding frequency component of the signal segment at the corresponding time.

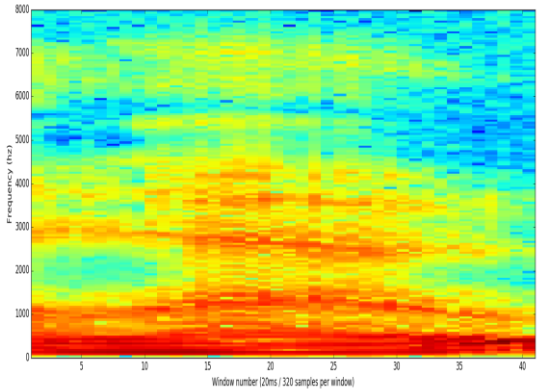

Fig 1. Example of a Spectrogram

We have performed this operation in our project using built-in functions of MATLAB.

V. Training The Neural Network

To train the neural network, we used Neural Network Toolbox of MATLAB. We created a simple network architecture as an array of layers. We used convolutional and batch normalization layers, and downsampled the feature maps "spatially" (that is, in time and frequency) using max pooling layers. Then, we added a final max pooling layer that pools the input feature map globally over time. This enforces (approximate) time-translation variance in the input spectrograms, which seems reasonable if we

expect the network to perform the same classification independent of the exact position of the speech in time. This global pooling also significantly reduces the number of parameters of the final fully connected layer. To reduce the chance of the network memorizing specific features of the training data, add a small amount of dropout to the inputs to the layers with the largest number of parameters. These layers are the convolutional layers with the largest number of filters. The final convolutional layers have 36864 weights each (plus biases). The final fully connected layer has 3840 weights.

Then, we used a weighted cross entropy classification loss. To give each class equal weight in the loss, we used class weights that are inversely proportional to the number of training examples of each class. When using the Adam optimizer to train the network, training should be independent of the overall normalization of the class weights. We used the Adam optimizer with a mini-batch size of 128 and a learning rate of 5e-4. Then, we trained for 25 epochs and reduce the learning rate by a factor of 10 after 20 epochs.

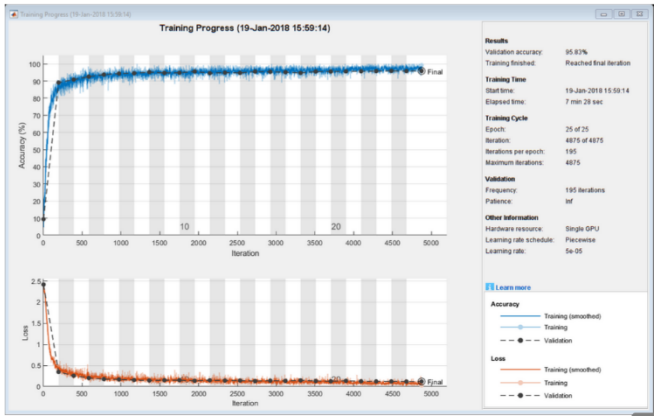

Fig 2. Training in progress

After the training process has been done, then we saved the trained network in a .mat file so that we can use it anytime we want without the need to train the network each time.

VI. Testing The Network

We have tested our newly trained command detection network on streaming audio from microphone. Then, we spoke one of the speech commands, for example, 'zero', 'one' etc. Most of the cases, it recognized the speech command. But, sometimes it got it wrong.

In Fig 3, we can see the interface through which we can test the speech recognition network. It continuously read data from microphone and sends a data of length 1 second to the network. The network classifies the signal and then the most probable command is set through a thresholding operation. It is shown as a title of the figure. In the figure, we can also see the signal and its spectrogram in the figure which is constantly changing as new speech signal is being input to it.

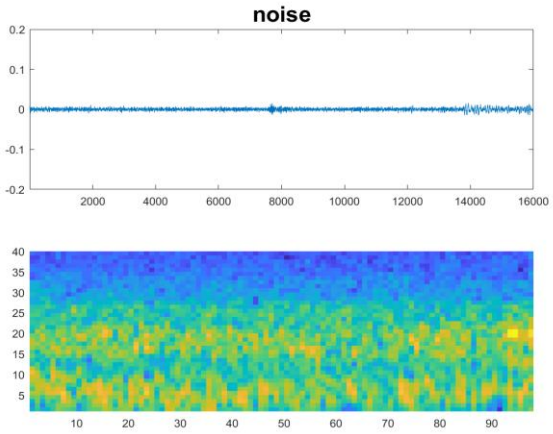

Fig 3. Interface of Testing the Speech Recognition System

VII. Creating The User Interface

For user interface, we have used subplots. In the top left subplot (see Fig 4), we have included the face detection segment. It uses the webcam of the laptop to stream video and detect faces. So, when a user is present in front of the laptop, the interface automatically detect their presence and starts operating. For face detection, we have used a built-in library. The icons below indicates when the interface talks and when the user in meant to talk. When the interface is saying, the speaker icon becomes green, and when the user is meant to respond, the microphone icon turns green.

In this case, we have implemented a library reception system. The interface first welcomes the user. Then, it describes different types of books assigned to different digits. Then, the user responds by saying a digit. Then, the speech recognition system recognizes the digit and then, further procedure is carried out.

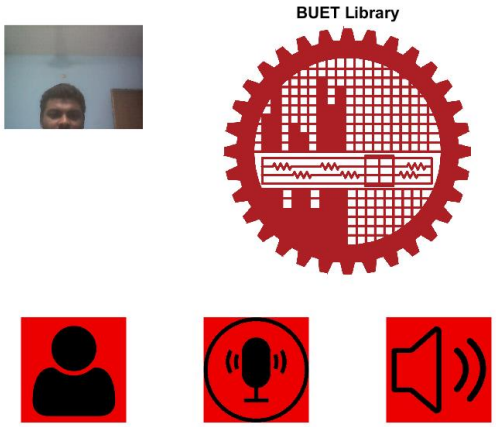

Fig 4. User Interface for Interaction Tool

VIII. Results

From the above discussion, it is evident that our Interaction Tool works as per our plan. But, due to time constraints, the user interface could not built completely. We had further plan to improve the interface by adding

purchase options and cart system like online shopping websites. Then, we could have added a billing system to it. However, being an initial development our tool is quite successful.

IX. Conclusion

It is quite evident that visually impaired people face some difficulties in public places where visual interaction is necessary. For this reason, a speech based interaction tool was very much necessary for them. But, not much work has been done about it, largely due to lack of research in Bangla speech recognition. So, it can be said that, our project may become the start of a trend about researching in Bangla speech recognition algorithms.

References

[1]. G.R. Bradski "Real Time Face and Object Tracking as a Component of a Perceptual User Interface", Proceedings of the 4th IEEE Workshop on Applications of Computer Vision, 1998.

[2]. Viola, Paul A. and Jones, Michael J. "Rapid Object Detection using a Boosted Cascade of Simple Features", IEEE CVPR, 2001.

[3]. Dalal, N. and B. Triggs. “Histograms of Oriented Gradients for Human Detection” Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, June 2005, pp. 886-893.

[4]. Warden P. "Speech Commands: A public dataset for single-word speech recognition", 2017. Available from http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz. Copyright Google 2017. The Speech Commands Dataset is licensed under the Creative Commons Attribution 4.0 license, available here: https://creativecommons.org/licenses/by/4.0/legalcode.