Predicting Regional Accents of Bengali Language using Deep Learning

Md. Abu Ibrahim
Computer Science & Engineering
Brac University
Dhaka, Bangladesh
md.abu.ibrahim@g.bracu.ac.bd

Md. Nawaz-S-Salekeen Nayeem
Computer Science & Engineering
Brac University
Dhaka, Bangladesh
md.nawaz.s.salekeen.nayeem@g.bracu.ac.bd

Sadaf Al Arabi
Computer Science & Engineering
Brac University
Dhaka, Bangladesh
sadaf.al.arabi@g.bracu.ac.bd

Moin Mostakim
Computer Science & Engineering
Brac University
Dhaka, Bangladesh
mostakim@bracu.ac.bd

Warida Rashid
Computer Science & Engineering
Brac University
Dhaka, Bangladesh
warida.rashid@bracu.ac.bd

***Abstract*—Accent is a huge challenge in communication for all languages. Different people who speak the same language might pronounce the same word differently. In a conversation, if two people are from different regions and they have different accents, we can use our intuition to make sense of what the other person is saying. Sometimes, even our intuition cannot help determining the meaning of the words because of the difference in accent. Therefore, it is extremely difficult for an ASR (Automatic Speech Recognition) system to properly understand the words when the speaker uses different accent instead of the standard or formal accent as most of the time the ASR systems are trained on the formal or standard language. Now a days, most of these issues caused by accents are somewhat worked upon in most used languages like English, Mandarin and few other languages. However, the ASR systems used for Bengali Language is still at its infancy and different accents are a major issue. Finding audio features that differentiate the accents from one another and creating models to predict the accent using Deep Learning techniques will help to create a much better ASR System for Bengali Language. This paper will emphasize on creating few models which can determine the regional accent of Bengali language given an audio sample. Furthermore, after getting the accuracy of the individual models we can choose the model which results in the most accuracy. Further work can be done based on the models to create an ASR System for Bengali language which will be able to handle few more accents than the standard one.**

***Index Terms*—Deep Learning; Bengali Accent; Accent Prediction; Prediction; Neural Networks; MLP; CNN; RNN; ASR**

I. Introduction

A. Thoughts behind the thesis

In the recent years, we have seen drastic improvements in various Speech Recognition technologies. Smart speech recognizing AIs like Siri, Alexa, Google Assistant, Cortana etc. are just one tap away from us. These systems work flawlessly in most of the cases. However, depending on the variation in population of a country, every language has many different dialects based on various regions. Even sophisticated and highly advanced systems like the aforementioned ones have to face trouble when the speaker does not speak in the standard accent of that language [5] . To tackle this issue, quite a lot of research is already done in English, Mandarin and few other prominent languages. However, while we were researching for related papers in this field, we barely found any work that was done in Bengali language. Worldwide almost 210 million [3] people speak in Bengali language. Among them, 100 million are from Bangladesh. Bangladesh is divided into 8 divisions and these divisions are divided into different number of districts which totals in 64 districts. People from these divisions speak in different dialect and accent. Furthermore, even in the same division people's accent varies greatly from district to district. These different accents and dialect affect the performance of any Speech Recognizing system significantly. In this paper, we will try to build a model by training it on various audio samples from few divisions and districts of Bangladesh. By using the approaches discussed in this paper, it will be possible to easily distinguish accent of the speaker. If separate ASR systems are created for each accents then our models will be able to redirect the speakers audio to the ASR model which was built for the accent of that speaker. Using the prediction of our models it might also be possible to build a single ASR system which will be able to detect the spoken words accurately regardless of the accent.

B. Research Problem

Our research paper focuses on detecting different Bengali accents. So, the primary research problem of our thesis if finding out exactly what makes each accent so different from each other. The local people of the district Noakhali speak in a noticeably different accent from standard Bengali accent which is known as Cholito Bhasa. Moreover, even though Old Dhaka is within the Dhaka district but people from this place speaks in a very different accent from people who lives in the main city of Dhaka. Let's look at a table with few examples of how the word ‘Khabo’ which translates to ‘Will Eat’ is different based on few different regions of Bangladesh [2]. As we can see, pronunciation of words can vary greatly depending on the region a speaker is from. Now, these difference in accent can cause various problems and significantly reduce the accuracy of a Speech Recognizing system. Our paper tries to tackle this issue for Bengali Language. To implement an ASR system for a language, gender dependent models are created. However, this does not solve the issue that is caused by different accent. In the year 2000, extensive experiments were done on Microsoft Mandarin Speech Engine. It was found that tone-related-information are the most important feature of a language in different accents. Later, in the year 2004, on a paper [8] by Chao Huang, Tao Chen, Eric I-Chao Chang, which was based on the information from the aforementioned experiments, they described cross-accent models for ASR system had 40 to 50 percent more error that than an accent-dependent model. There are few important factors in this rise of error rate. There are a lot of features when creating a model for a speaker as there is a lot of variability in the tone, pitch etc. This results in a very complex model as the number of dimensions increases. Few tools such as PCA (Principal Component Analysis) and ICA (Independent Component Analysis) can help to lower the number of dimensions and reduce the complexity. Quite a long time has passed since these papers were published. Moreover, in today’s age these problems are mostly solved for the prominent languages like English and Mandarin. However, these problems still exist in the Bengali Language.

English	Standard Bangla (চলিত ভাষা)	Khulna (খুলনা)	Barisal (বরিশাল)	Old Dhaka (পুরান ঢাকা)	Faridpur (ফরিদপুর)
Will Eat	‘Khabo’	‘Khabani’	‘Khamuoni’	‘Khamu’	‘Khaum’
	খাবো	খাবানি	খামুওনি	খামু	খাউম

Fig. 1. Phonetics variation of the same word across regions.

C. Research Objectives

The main goal of this paper is to create a model which will be able to detect few different accents of Bengali Language given an audio sample of a specific sentence. This paper focuses on accents used in various regions of Bangladesh. As discussed in the “Research Problem” section 1.1, different accents have major impact on the performance of an ASR system. Since there is almost no research on the impact of various Bengali Accent in an ASR system for Bengali language, we want to start it by working on audio samples from few different regions of Bangladesh where people have different accents and dialects. The objectives of our research are explained below in bullet points:

- Determining what main features such as Pitch, Tone, Prosodic feature, Formants etc. of certain accents plays major role in the change of that accent from the standard Bengali accent. We will take inspirations from the previous works that was done in other Languages and implement it in the Bengali Language.
- Creating models based on various techniques which will be able to differentiate between the accents from the taken samples. We will use MLP (Multilayer Perceptron), CNN (Convoluted Neural Network), RNN (Recurrent Neural Network) etc. to create these models.
- Improving the accuracy and reducing the error rate to optimize each model.
- Lastly, determining which model results in the highest accuracy and discussing the room for improvements on future implementation of this paper.

II. Related Works

A. Literature Review

Before diving into the specific literature let us have a brief discussion on the overall process each of these literature works upon.

The primary goal of any accent detection model is to make ASR systems much more general-purpose by solving challenges such as variability of volume, word-speeds, speaker, pitch, etc. As Automatic Speech Recognition Systems takes any continuous audio speech and output equivalent text, accent recognition serves as an essential step to all ASR systems.

To do this the basic approach followed by most of the researchers goes something like this. First they had to collect data from various region of their interest based on differences of how they speak. Note that, in many cases these regions can be considered as countries. After that, they pre-process the data and extracts feature to feed into the ML model. MFCC is a feauture that has been used for almost all the researches. After this phase, all the suitable Machine Learning models are used like SVM, random forest, DNN, RNN to classify accents based on the region.

B. Related Works

K. Mannepalli and V. Rajesh in their work “Accent detection of Telugu speech using Prosodic and Formant feature” [11], used predefined features like pitch, energy, power spectral density, short-time energy, intensity; extracted using COLEA and PRAAT to feed into a Nearest Neighbor Classifier (NNC), achieving 72% accuracy in classifying three different regional accents of Telegu Language spoken in Southern part of India.

In [16], rather than using NNC to classify, they proposed a system with Gaussian Mixture Model (GMM) and Support Vector Machine (SVM). They created a GMM supervector by mapping an utterance to a high-dimensional vector. As we know SVM is widely used for classifying data belonging in a high dimensional vector space, many researchers used this method. Similar to these two aforementioned approaches can also be in the works of [14, 6, 10, 4].

A bit different strategy based on Deep Learning is found in the works of F. Weninger and Yang Sun in their work

"Deep Learning-based Mandarin Accent Identification for Accent Robust ASR".[15] They found great success classifying 15 different geographical regions by accents in China even though some of them were not even mutually intelligible. Utilizing two different ASR models, standard and accented, they suggested using the bLSTM (bidirectional Long Short-Term Memory) accent classifier to quickly switch between these two models depending on the given scenario. They had gathered 466 speakers and collected 135k utterances (84.6 hours). The reason behind using bLSTM was to capture the longer-term acoustic context in each utterance thus supposedly improving the accent identification process.

A much more probabilistic approach is taken in the work "Accent Detection and Speech recognition for Shanghai-Accented Mandarin".[17] They used MFCC with GMM to classify accentedness (the level of deviation from the standard accent) level into three tiers. Also, they defined two different speaker groups, more standard or more accented. In the end, to select a suitable model given a speaker they calculated MAP (Maximum a posterior) of different models and thus choosing the best one. By using MAP with traditional approaches, in their experiment, results show a 1 to 1.4% absolute reduction of character error rate (CER).

One of the primary reasons why the ASR system has a higher error rate for an accented speech is simply due to the fact that the speaker might just be mispronouncing the given word just a bit. A much more sophisticated approach was designed by some Microsoft Researcher in their paper "Accent Issues in Large Vocabulary Continuous Speech Recognition".[8] They developed a new adaptation method called Pronunciation Dictionary Adaptation, which is fundamentally a dictionary, capturing the pronunciation variations due to the mistake of the speaker(mispronunciation) for an accent by feeding the system a small amount of adaptation data. The character error rate (CER) of the system was 13.2% - 13.6% given that the system has 3 to 5 utterances available for an individual speaker.

Lastly, A paper from Bangladeshi Researchers from Daffodil University, titled "Bengali Speech Recognition from Speech" [1] did an amazing work. They collected data from 9 regions and extracted features like Chroma Features, RollOff, MFCCs, ZCR, RMSE, Spectral Centroid, Spectral Bandwidth. They fed the data into a Random Forest based model to get 86% accuracy. They also used few other models but Random Forest got them the most accuracy. This paper will be mostly inspired by their paper.

III. Dataset and Features

A. Data Collection

Data collection part was assumed to be the most difficult part of this thesis as we have to collect pure accented audio samples from different districts of our country. However, we were able to find [1] this paper by some researchers from Daffodil University on similar topic. We contacted them and they were kind enough to provide us with their dataset. Although, further cleaning of the audio samples and preprocessing was necessary. They used three different methods to collect the audio samples. The methods are described below:

1) **Manually:** They physically approached various people from different districts to record their voices. They used a specific script to record the audio samples from them. However, this proved to be quite time consuming and tedious. So, small number audio samples were collected using this process.
2) **Google Form:** A google form was also created to gather audio samples from individuals. But the variety in audio quality and lack of willingness to record and upload an audio sample to the form resulted in very small amount of audio samples being collected via this process.
3) **Youtube:** Lastly, this method proved to be the most effective way for them to collect data. Many regular channels and News channels existed on Youtube where the host was from certain region of Bangladesh and that person only spoke in the accent of that specific region. Most of the audio samples of the dataset is collected using this method. They segmented each sample audio into 5 seconds and converted them into 'wav' formatted audio files. Sample rate of each audio file was 16MHz. They used the website YT Cutter to create the 5 seconds audio samples.

Total of 9263 audio samples of 5 seconds each were collected by using the 3 methods described above.

B. Data Cleaning And Denoising

After getting the dataset, we started going through the individual audio files. During this time, we found few flaws in the dataset. The three major problems are described below:

- The audio samples in the formal accent were in various formats. None of them were in '.wav' format. In order to process those audio samples, we had to convert all of them in the '.wav' format to process the samples using librosa. A script was written in order to process all the files into '.wav' files. Nonetheless, the Formal samples were the most clear and crisp samples among all the accents.
- The samples which were taken from YouTube, for example from various regional news channels or from various regional Bengali Natoks (Films) had a significant amount of background noise and extra music which were not part of the speech. In order to extract only the features of the accents we needed to create a noise or background noise free samples. We used an open source software called 'Audacity' to clean the audio samples. However, the samples of the Sylhet region were taken from a news channel which contained heavy music when transitioning from reading one news headline to another. There were few samples which did not even contain any speech at all. In these cases, we tried to remove the portion of the audio which contained certain musical noises. Unfortunately, we had to discard few samples because it was not possible to denoise the audio sample without causing significant

loss in the speech. The audio samples before and after removing the musical noise can be seen in Figure 2 and Figure 3 respectively. These screenshots were taken from the Audacity software.

The audio samples shown in figure 2 are from the Sylhet region. These were collected from a Sylhet news channel. There were musical sounds between each headline as the news anchor read them. we can see that these 3 audio files have a similar amplitude in certain time domains. These parts had to be removed using Audacity. The samples shown in figure 3 are the same samples as the previous ones, only this time the musical part is muted. At the beginning this method was mainly used to remove the musical sounds from the audio samples. However, very soon this proved to be quite cumbersome as it was an extremely labor intensive work to go through each and every audio samples of the dataset.

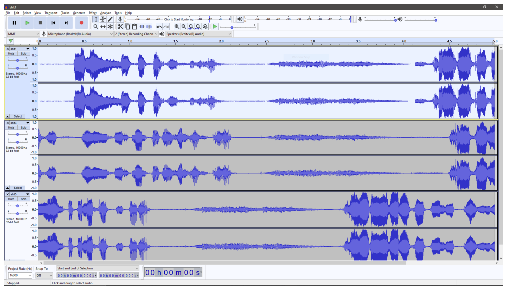


Fig. 2. Before removing noise

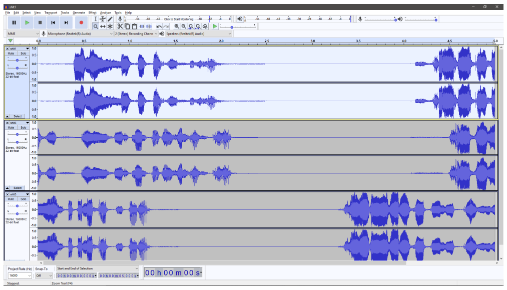


Fig. 3. After manually removing noise

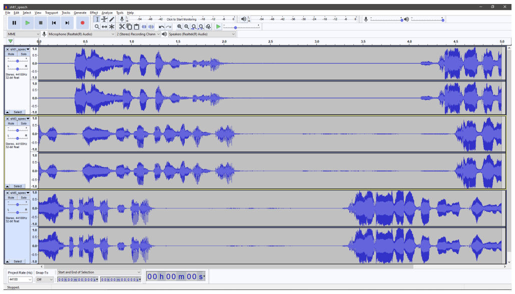


Fig. 4. After removing noise using the pre-trained model spleeter

- In order to remove the music part from our dataset We had to opt for a better and more efficient way which would enable us to write a python a script to automate this “noise” removal process. After looking for a while we found a pre-trained ML model named Spleeter [7] which was able to separate speech and music from a given audio sample. Fortunately, this pre-trained model can be accessed using a python module created by them. A script was written in python Using this pre-trained model in order to separate the speech from the music and keep only the speech audio files. Python’s OS module was used as well in this script to organize the cleaned audio samples. Albeit, the speech audio samples generated using this approach was not as clean as they could be, if we manually cleaned them. But it was almost impossible to go through all the 9263 samples one by one. The figure 4 shows the output speech samples of the same data in the figure 2, but this it was cleaned using the pre-trained model. The difference between the outputs of using these two different approaches can be easily compared using the figure 3 and figure 4 which shows manual and using the pre-trained model respectively. We can see that there is barely any difference between the two.
- Lastly, for proper MFCC features extraction we had to make sure that all the samples of the dataset are of same length otherwise it would not be possible to properly segment each sample and take same amount of MFCC features. A python script had to be written using the pydub [9] module in order to determine whether all the samples were of the same length. After running the script we realized that most of the samples in the formal folder were not exactly 5 seconds in length. The samples which were lower than 5 seconds were thrown away and those which were longer than 5 seconds had to be trimmed down to 5 seconds using another python script which used the pydub module as well.

C. Final Dataset

After completing this data cleaning process, we had total of 9075 audio samples. The sample distribution of our dataset is given in the figure 5.

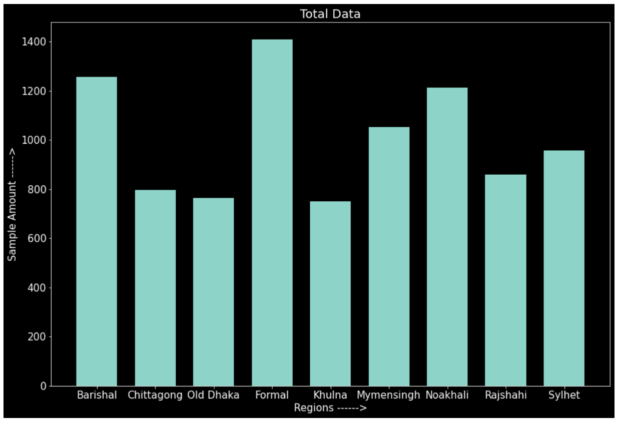


Fig. 5. Sample distribution

D. Features

1) Amplitude Envelope: This is a time domain feature. It refers to the max amplitude value of all samples in a frame. This is an important property of sound, because it is what allows us to effortlessly identify sounds, and uniquely distinguish them from other sounds. The way we calculate amplitude envelope is by this equation:

$$AE_t = \max_{k=t.K}^{(t+1).K-1} s(k) \tag{1}$$

Here, $s(k)$ refers to the amplitude calculated at the kth sample, K is the frame size, and t refers to the sample number of a given frame in the iteration.
Amplitude envelope gives us idea of loudness of the signal we are working with. Though it is sensitive to outliers, it is extremely useful in onset detection. In figure 6 the amplitude envelope of the first sample of Barisal region is shown.

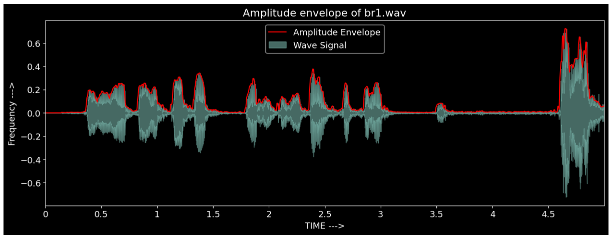


Fig. 6. Amplitude Envelope of a sample from Barisal Region

2) Zero Crossing Rate: It is also an time domain feature of a signal. It tells us the number of times a signal crosses the horizontal axis. The way we calculate Zero Crossing Rate is by this equation:

$$ZCR_t = \frac{1}{2} \sum_{k=t.K}^{(t+1).K-1} |sgn(s(k)) - sgn(s(k+1))| \tag{2}$$

Intuitively, it means that we calculate amplitude value of consecutive pairs of samples, and look for sign differences in those pairs of values. We define the $sgn()$ function as such:

- $s(k) > 0 \longrightarrow +1$
- $s(k) < 0 \longrightarrow -1$
- $s(k) = 0 \longrightarrow 0$

We take the sgn of amplitude at sample k and then we subtract that with the sgn of the amplitude at sample $k+1$. For same sgn values we get zero. And otherwise we get value of 2 indicating a crossing has happened in that pair of samples.
Zero Crossing Rate is a fairly popular audio feature in Audio Signal Processing. It is used for recognizing between percussive and pitched sound, monophonic pitch estimation. In figure 7 we have shown ZCR of a sample in Barisal Region.

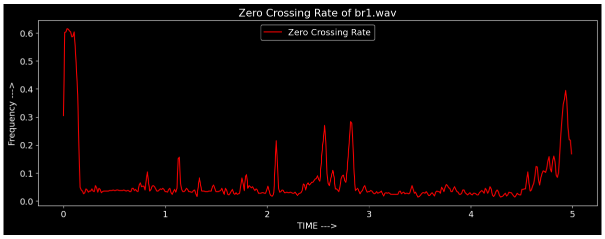


Fig. 7. Zero Crossing Rate of a sample from Barisal Region

3) Root Mean Square Energy: The concept of Root Mean Square Energy is quite simple. As the name suggests, it takes the root mean square value of the Amplitude or the energy of all samples in a single time frame. That is why it is a time domain feature. The equation used to calculate RMSE is given below:

$$RMS_t = \sqrt{\frac{1}{K} \sum_{k=t.K}^{(t+1).K-1} s(k)^2} \tag{3}$$

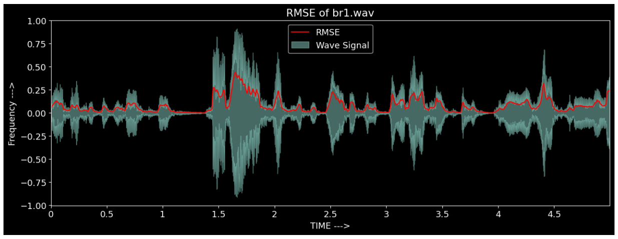


Fig. 8. RMSE of a sample from Barisal Region

In the formula, $s(k)$ is the energy of the k^{th} sample. This formula is summing up the energy of all the samples in frame t. Here, K is the frame size or the number of samples in a given frame. A visual representaiton of RMSE is given in figure 8, it is showing the ZCR value of the first sample from the Barisal Region.

4) Spectral Centroid: If we think intuitively, then we can think of spectral centroid as the 'Brightness' of the sound. This feature of audio easily maps to the 'Timbre' of a sound. It is the center of gravity of the magnitude spectrum in a given audio sample. In other words, it gives us the frequency bins where most of the energy in a given sample is stored.

Just like other frequency domain features, we need to apply STFT to get the spectrogram information, then we can move on to extract the spectral centroid. In the formula of Spectral centroid, we can see that the weighted mean of. Here, Mt(n) is the magnitude of the signal at time frame 't' and frequency bin 'n'. N is the total number of bins. The equation we use to calculate Spectral Centroid is given below:

$$SC_t = \frac{\sum_{n=1}^{N} m_t(n) \cdot n}{\sum_{n=1}^{N} m_t(n)} \tag{4}$$

In this spectrogram below in figure 9 we can see the white line as the spectral centroid of "br1.wav" file which is the first sample from the Barisal region. This concept is similar to RMSE whereas during RMSE the calculated mean is Amplitude and in this case the mean is Frequency. This

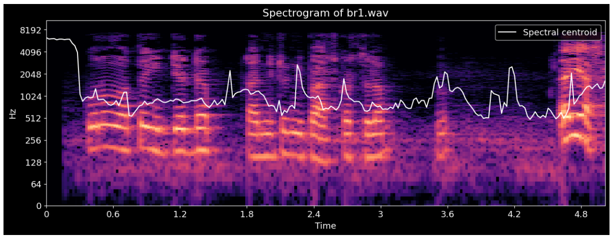


Fig. 9. Spectral Centroid of a sample from Barisal Region

feature can help us determine the difference between the accents using the variety of frequency bins that can found in each regional accent.

5) Spectral Bandwidth: This feature is derived from the previously mentioned Spectral Centroid. It gives us the spectral range around the centroid. If we think of the spectral centroid as the mean of the spectral magnitude distribution then spectral bandwidth can be thought of as the 'Variance' of that mean. This feature can be also mapped to the 'Timbre'. So, spectral bandwidth is also a weighted mean. But this time, it is the weighted mean of the distances of frequency bands from the Spectral Centroid.

$$BW_t = \frac{\sum_{n=1}^{N} |n - SC_t| \cdot m_t(n)}{\sum_{n=1}^{N} m_t(n)} \tag{5}$$

From the formula we can clearly see the similarities with the Variance formula. Here, $m_t(n)$ is the magnitude of the signal at time frame t and frequency bin n. This time, in the formula we are using the difference between the Spectral Centroid value and the current frequency bin value. N is the total number of bins. The spectral bandwidth gives the idea that how the energy of the given sample is spread throughout all the frequency bands. It basically means, if the energy is spread across the frequency bins, then the value of Spectral Bandwidth will be higher. On the other hand, if the energy is focused on specific frequency bins, then the value of Spectral Bandwidth will be lower.

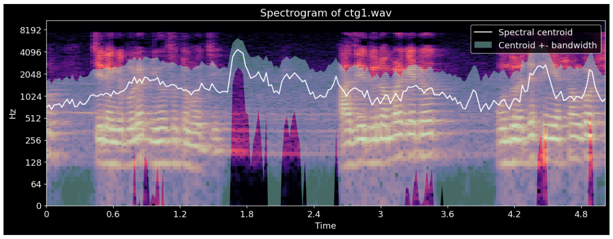


Fig. 10. Spectral Centroid & Bandwidth of a sample from Chittagong Region

6) MFCCs: Mel-Frequency Cepstral Coefficients is somewhat related to the idea of mel-spectrogram. Basically we are using a mel scale, which is a perceptually relevant scale for pitch. Depending on the implementation it can give up to 39 features, for our work we are using 13 features.

To understand MFCC we need to formalize the idea of **Cepstrum** mathematically. We compute cepstrum with this formula:

$$C(x(t)) = F^{-1}[log(F(x(t))] \tag{6}$$

where $x(t)$ is the time domain signal, then we take the discrete Fourier transform of the signal, giving us the Spectrum of the signal in the frequency domain. Taking Log of the spectrum we get Log Amplitude Spectrum. And finally we take inverse Fourier transform, to get Cepstrum.

Now we turn our attention to **Mel-Scale**. Its a mapping between our perceived frequency or pitch of a pure tone to its actual frequency or pitch. The reason we use Mel-Scale because we human are much better at discerning small changes in pitch at low frequencies than they are at high frequencies. Incorporating this scale makes our features match more closely what humans hear. Given a frequency we calculate its mel-frequency with this formula:

$$M(f) = 1125ln(1 + f/100) \tag{7}$$

And to go from mel-scale to frequency, we use this:

$$M^{-1}(m) = 700(exp(m/1125) - 1) \tag{8}$$

Now we take a quick look on the implementation of the MFCCs. We start with an audio signal which is sampled with 22kHz.The overview of the whole process is given in the figure 11 :

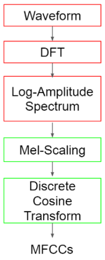


Fig. 11. MFCC Steps

- The frame size we used was 2048 samples per frame. We let Librosa decide the hop length so it can give us 13 features. Then for each frame 13 mfcc coefficients were extracted. Let $s(n)$ be our time domain signal. After framing the whole signal we have $s_i(k)$ where i ranges over 1 to 256 indicating each samples and k denotes the frame number. Also, let $P_i(k)$ denotes the power spectrum of frame i.
- Now we apply Discrete Fourier Transform to each frame. For that we use this equation:

$$S_i(k) = \sum_{n=1}^{N} s_i(n)h(n)e^{-j2\pi kn/N} \tag{9}$$

where $h(n)$ denotes the hamming window. Now to get power-spectral we use,

$$P_i(k) = \frac{1}{N}|S_i(k)|^2 \tag{10}$$

This is called the Periodogram estimate of the power spectrum.

- Now we take our attention to compute mel-spaced filterbanks. A set of triangular filters, mostly 13, were applied to periodogram power spectral estimated from the previous part.
- We take the log of 13 energies from the last step. This will create a log filterbank.
- Then we take the Discrete Cosine Transform (DCT) of our log filterbank to get 13 MFCC coefficients of each frame.

  Figure 12 shows a spectogram of 13 MFCC extracted from the first audio sample of the the Barisal region which was called “br1.wav”. By observing closely, 13 horizontal segments can be seen in the spectogram which are generated from values of each MFCC.

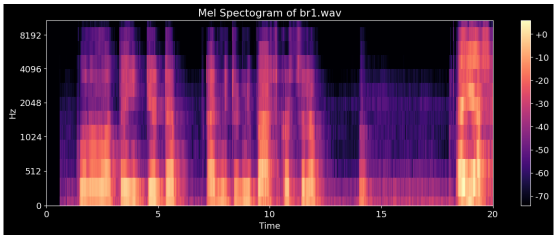


Fig. 12. Mel Spectrogram using the 13 Coefficients

IV. Proposed Methodology

A. Work Plan

In order to differentiate various accents from each other, the first step was to find and collect the proper dataset for the thesis. The next step was to decide which accent to collect and start the sample collection process. However, thanks to researchers of [17] paper, this process was skipped as their dataset was used for this work. The python module “librosa” was used to process the dataset. Moreover, before processing few scripts were written to clean the audio samples. After cleaning and de-noising the samples, 9075 usable samples were stored. Each sample was a 5 seconds long audio file which was sampled at 22050hz sample rate. Various audio features were extracted from the samples and using “librosa” and using the python module “matplotlib” the features were visualized. An initial CSV file was created to store the mean values of extracted features as the first dataset. For a second dataset each 5 seconds audio was broken into 1 second audio each containing 22050 quantized samples. After segmenting the 5 seconds audio files into 1 second parts, the total number of audio files were 45375. From each of these 45375 files 13 MFCC features were extracted. Three neural network algorithms were applied which are Multilayered Perceptron Neural Network, Convolutional Neural Network and Recurrent Neural Network on the second dataset. Tweaking the parameters of these networks was necessary to get the best result. Only accuracy alone is not enough to determine the best model. So, confusion matrices, accuracy, precision, recall and f1 score were used to determine the best model from the thesis. Figure 13 shows a diagram of the work plan containing the summary of the individual steps.

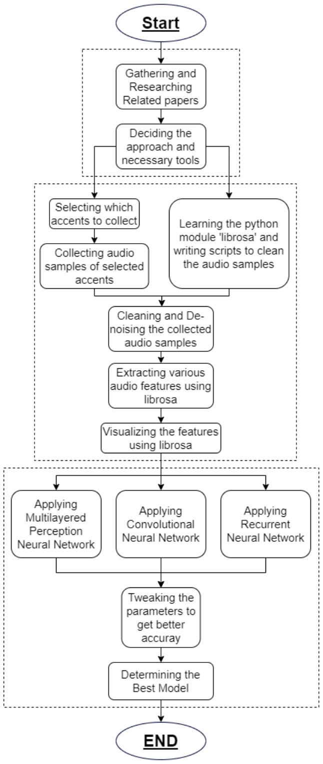

Start
Gathering and Researching Related papers
Deciding the approach and necessary tools
Selecting which accents to collect
Collecting audio samples of selected accents
Learning the python module 'librosa' and writing scripts to clean the audio samples
Cleaning and De-noising the collected audio samples
Extracting various audio features using librosa
Visualizing the features using librosa
Applying Multilayered Perception Neural Network
Applying Convolutional Neural Network
Applying Recurrent Neural Network
Tweaking the parameters to get better accuray
Determining the Best Model
END

Fig. 13. Work Plan

B. Feature Extraction

Using the python module Librosa, all the samples were processed and the features were extracted. The mean values of 18 extracted features from each 5 second sample was taken and stored in the CSV file.This was the first created Dataset.

A second dataset was created in order to store only the MFCC features. There were total 9075 processed samples. However, there was a high possibility of irrecoverable level of overfitting when training a neural network using only this small amount of samples. Furthermore, segmentation the samples was necessary in order to get even more detailed features of these audio samples. Thus, each of the audio samples were divided into 5 segments, which functioned as individual samples of 1 second each. During feature extraction we used sample rate of 22050Hz. So, each of these audio files contained 22050 samples. MFCC features were extracted by applying Fast Fourier Transform 2048 times and using hop length of 512 samples from the 22050 samples of each audio segment of 1 second. So, ceiling value of (22050/512) which is 44 segments of 13 MFCC features were extracted from 1 second of audio. Meaning, total of (44*13) = 572 MFFC were extracted from 1 second of audio sample. This whole segmentation process and MFCC feature extraction process is shown in figure 14.

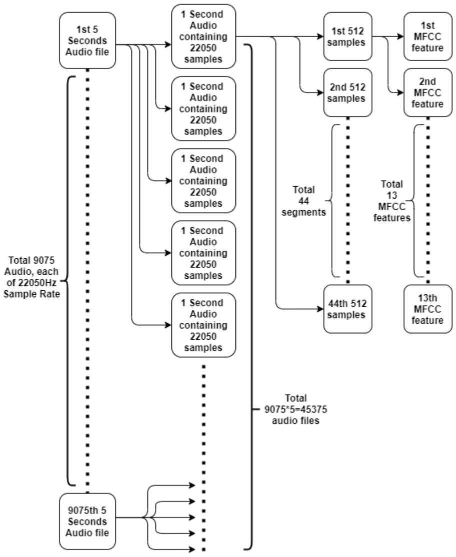

1st 5 Seconds Audio file
1 Second Audio containing 22050 samples
1 Second Audio containing 22050 samples
1 Second Audio containing 22050 samples
1 Second Audio containing 22050 samples
1 Second Audio containing 22050 samples
1st 512 samples
2nd 512 samples
44th 512 samples
Total 44 segments
1st MFCC feature
2nd MFCC feature
13th MFCC feature
Total 13 MFCC features
Total 9075 Audio, each of 22050Hz Sample Rate
Total 9075*5=45375 audio files
9075th 5 Seconds Audio file

Fig. 14. Audio segmentation and MFCC features extraction process

C. Proposed Architecture

1) MLP: Three different Neural Network architectures was used to process the MFCC features from the JSON dataset. The python module Tensorflow GPU [12] was used to build and train these neural networks. The architectures of these neural networks are described below:

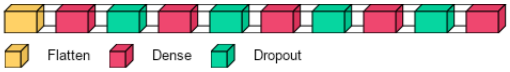
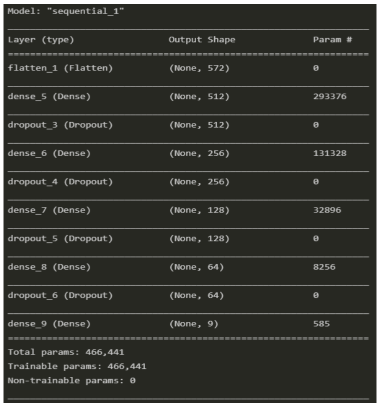


Fig. 15. High level view of the MLP model

This was the first implementation of the neural network. It is a very simple fully connected architecture containing, an input layer then 4 hidden dense layers and the output layer consisted of 9 neurons representing 9 different regions. The input shape of our dataset was (44x13), 44 different segment each containing 13 MFCC features. However, the input layer of a MLP neural network has be 1 dimensional. So, these 2 dimensional data was flattened into 1 dimensional data. Therefore, the input layer had (44x13)= 572 data points. A high level view of the architecture is shown in figure 15.

These values were passed onto the 1st hidden layer, which had 512 neurons. Then, the second layer contained 256 neurons, the third layer contained 128 neurons and lastly the fourth layer consisted of 64 neurons. For each of the neurons the activation function ’ReLu’ was used. Finally, the output layer contained 9 neurons and ’Softmax’ was used as the activation function in order the maximize the output neurons value so that the the highest value stood out.



```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 572)               0
_________________________________________________________________
dense_5 (Dense)              (None, 512)               293376
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_6 (Dense)              (None, 256)               131328
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_7 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_5 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_8 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_6 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_9 (Dense)              (None, 9)                 585
=================================================================
Total params: 466,441
Trainable params: 466,441
Non-trainable params: 0
_________________________________________________________________
```

Fig. 16. Summary of the MLP model

However, after the training process, the realization came that there was a lot of overfitting as the testing accuracy was not keeping up with the training accuracy. That is why, L2 regularization was implemented where the value of lambda was 0.001. Moreover, 10% of the neurons were dropped after each of the dense layers. In figure 16 the detailed architecture of this MLP architecture is shown.

2) CNN: Our CNN architecture consisted of two 2D convolution layer followed by Max-pooling layer after each of the convolution layer. As, the CNN architecture is designed to process 2D image data, the data of figure 12 from each of the samples were fed into the network. In the figure 12 image 13 horizontal lines can be seen which represents the 13 MFCC values. These can be thought of as the Y axis and for the X axis it was 44 different segments each containing 512 bins from 1 second audio. Therefore, the value passed into this architecture can be thought of as a 44x13 pixel image. Since, the batch size was 64, so 64 of these 2D data was fed into the network.

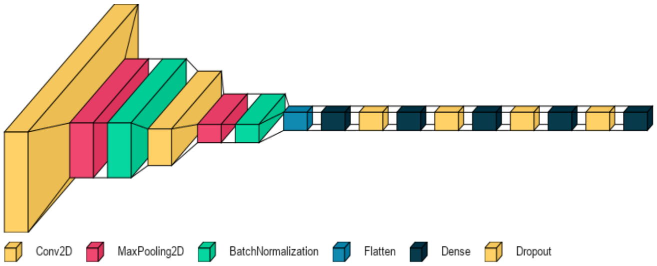
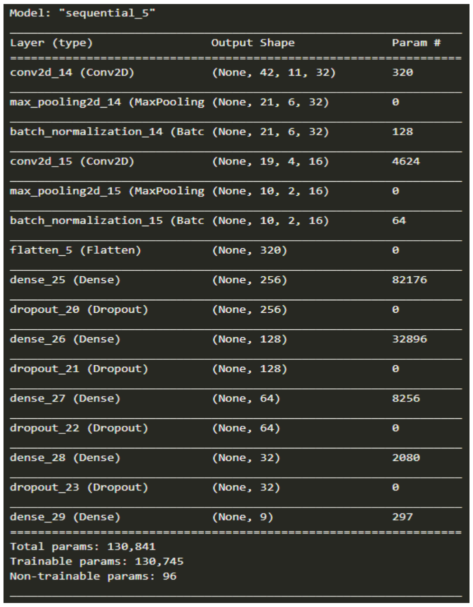


Fig. 17. High level view of the Hybrid CNN architecture

In the first convolution layer 32 kernels were used each consisting of (3,3) shape. The stride value was (1,1). Then, the scanned values of these kernels were passed onto the next Max-Pooling layer in order to downscale and take only the important features from the first layer. The pooling window size were (3,3) and stride was (2,2). Same padding was used as the last convolution layer which is (1,1). After the max-pooling, Batch Normalization layer was used in order avoid vanishing or exploding gradients while back propagating through the network. Now, the second 2D convolution layer consisted of 16 kernels each of same shape and stride of the first convolution layer. The same max-pooling layer was used after the second convolution layer. An high level overview of the hybrid CNN architecture can be seen in figure 17.

After the convolution layers, the shape of the data was (10,2,16). So, it was flattened to (10*2*16)=320 1 Dimensional data. Finally, it was passed into a network of 4 dense layers containing 256, 128, 64 and 32 neurons. As activation function, 'ReLu' was used in all these layers. In order to reduce overfitting after each dense layer 10% of the neurons were dropped and L2 regularization was used where the value of Lambda was 0.001.



```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_14 (Conv2D)           (None, 42, 11, 32)        320
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 21, 6, 32)         0
_________________________________________________________________
batch_normalization_14 (Batc (None, 21, 6, 32)         128
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 19, 4, 16)         4624
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 10, 2, 16)         0
_________________________________________________________________
batch_normalization_15 (Batc (None, 10, 2, 16)         64
_________________________________________________________________
flatten_5 (Flatten)          (None, 320)               0
_________________________________________________________________
dense_25 (Dense)             (None, 256)               82176
_________________________________________________________________
dropout_20 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_26 (Dense)             (None, 128)               32896
_________________________________________________________________
dropout_21 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_27 (Dense)             (None, 64)                8256
_________________________________________________________________
dropout_22 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_28 (Dense)             (None, 32)                2080
_________________________________________________________________
dropout_23 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_29 (Dense)             (None, 9)                 297
=================================================================
Total params: 130,841
Trainable params: 130,745
Non-trainable params: 96
_________________________________________________________________
```

Fig. 18. Summary of the Hybrid CNN architecture

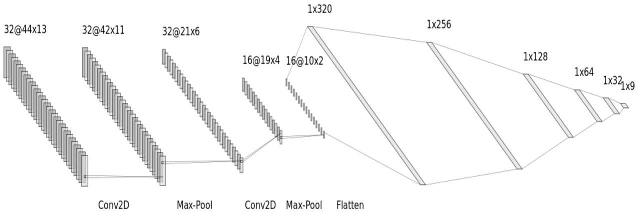
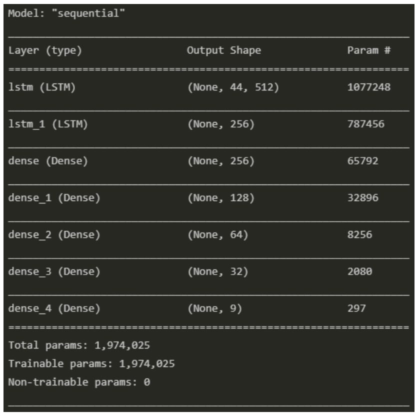


Fig. 19. Visual representation of the Hybrid CNN architecture

The output layer consisted of 9 neurons where 'Softmax' was used as activation function. Here, the 9 different neurons contained the probability of being in 9 different regions. 'Softmax' to increase the value of the highest probability and to reduce the other probabilities.

Summary of the architecture is shown in figure 18. Moreover, an even more detailed visual representation is also shown in figure 19.

3) RNN-LSTM: As explained in the algorithm portion Recurrent neural networks are perfect for time-series type of data. In our case, it was expected that RNN would give us the best accuracy because audio data are perfect example of time-series data. As we know, speech is a continuous form of data. In order to recognize accent it is vital to consider information from the sample bins. Memory cells of RNN architecture give the network the ability to keep information from the previous cycle. LSTM type of memory cell was used in this architecture which stands for Long Short Term Memory. This special type of cell allows the network to not only store the Short Term information but also the Long Term information. Therefore, the information from even more recurrents earlier has impacts on how the weights are adjusted for the current neuron.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 44, 512)           1077248
_________________________________________________________________
lstm_1 (LSTM)                (None, 256)               787456
_________________________________________________________________
dense (Dense)                (None, 256)               65792
_________________________________________________________________
dense_1 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_3 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_4 (Dense)              (None, 9)                 297
=================================================================
Total params: 1,974,025
Trainable params: 1,974,025
Non-trainable params: 0
_________________________________________________________________
```

Fig. 20. Summary of the Hybrid RNN-LSTM architecture

As usual the input shape of the data was (44,13), 44 segments each containing 13 MFFC values. This input was passed into the first LSTM layer. It consisted of 512 recurrent neurons. Each of these memory cells had 44 recurrent neurons inside it. Therefore, each of these 44 neurons were actually Dense layers on their own and by back propagation they changed their own weights and biases. In this first LSTM layer, the 'tanh' function was as acitvation function for the LSTM neurons and 'sigmoid' function was used as activation for the neurons of the recurrent layer. In order to reduce overfitting, L2 regularization was used in both the LSTM outer layer and also the recurrent layer. In both of these cases, the value of Lambda was 0.01. This LSTM layer passed a sequence to the next layer. Thus, this layer outputted a seuqence after taking in a sequence.

Another LSTM layer was added after the previous one. It consisted of 256 recurrent neurons. Rest of of the structure was same as before. However, this LSTM layer passed a Vector to the next layer. So, it took a sequence from the previous LSTM layer and passed a Vector in the following layer.

Finally, the exact same dense layers which was in CNN architecture was also created after the previous LSTM layer. To summarize, 4 dense layers each respectively containing 256, 128, 64 and 32 neurons. To reduce overfitting, L2 relarization as applied in all these layers where the value of Lambda was 0.001. However, no neurons were dropped.

Just like the previous two architectures, the output layer contained 9 neurons each representing different regions. In the figure 20 summary of the LSTM architecture is shown.

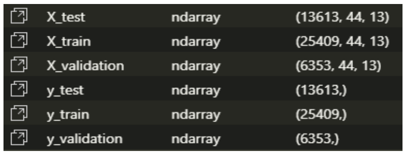
V. Implementation, Result & Analysis

A. Preparing the Data and Models

The networks were trained on the JSON dataset which contained only the MFCC features of the audio samples. This decision was taken because generally all the features described in III-D is more or less compressed into a Mel Spectrogram of an audio file and the MFCC features are extracted from this Mel Spectogram. Therefore, the MFCC itself can represents almost all the major features of an audio file.

B. Train, Validation & Test Split

Before fitting the models using the JSON dataset, it was split into three different parts. These are train, test and validation data. The JSON dataset contained MFCC features from total of (9075*5)=45375 audio samples since the 5 seconds audio files were segmented into 5 parts. At first the dataset was divided into training and testing data. From the whole dataset, 30% of the data, meaning ceiling value of (45375*30%)= 13613 samples were used for testing the models. On the other hand, 70% data from the dataset, meaning (45375-13613)= 31762 samples were used for training the models. For the training process, among the 31763 samples 20% was used for validation. So, ceiling value of (31763*20%)= 6353 samples were used for validation and (31762-6353)= 25409 samples were used for the actual training of the neural network models. Figure 21 shows this distribution of the dataset.



X_test	ndarray	(13613, 44, 13)
X_train	ndarray	(25409, 44, 13)
X_validation	ndarray	(6353, 44, 13)
y_test	ndarray	(13613,)
y_train	ndarray	(25409,)
y_validation	ndarray	(6353,)

Fig. 21. Splitting of the Dataset

C. Training Parameters

In order to train the models, Adam optimizer was used with a learning rate of 0.0001 in all the models. Adam was used because, in general it is very effective in most of the neural networks. Moreover, in most of the cases Adam is able to reach the local maxima quite fast within fewer number of epoch. Furthermore, as the loss function “Sparse Categorical Crossentropy” was used because the output layer contained 9 neurons representing 9 different regions. For, this type of multi class classification “Sparse Categorical Crossentropy” loss function gives the optimum result. Lastly, all of these models were fed the data in batch size of 64.

D. Accuracy and Loss Graph

At the end of training MLP had training accuracy of 77.9% and validation accuracy of 65.1%. It was a very simple architecture so this type of accuracy was expected. However, there still exists overfitting which can be seen in the 22. Using L2 regularization and dropout of certain amount of neurons overfitting was solved a bit but it was not possible to get rid of it fully.

In figure 26, the graph containing training and validation accuracy and loss of the Hybrid CNN model is shown. The training ended with training accuracy of 81.5% and validation accuracy of 69.4%. This is a slight improvement from the previous model and the graph is more stable.

Lastly, the graph for the LSTM model is shown in the figure 24. It can be seen clearly that the accuracy of this model is quite better compared to the the previous two model. It was to be expected because, RNN architecture is good for time-series data. Albeit, the progress was a bit unstable throughout the training period. At the end, the validation accuracy was 76.2%.

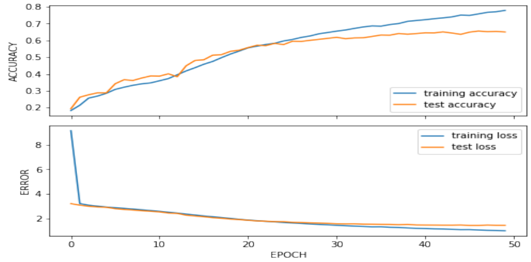


Fig. 22. Accuracy and loss graph of MLP

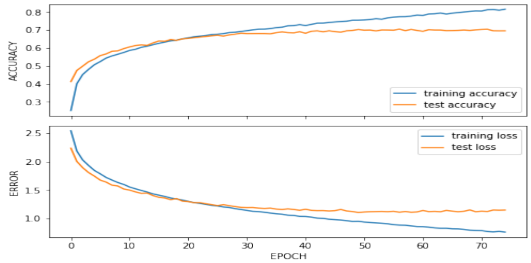


Fig. 23. Accuracy and loss graph of Hybrid CNN

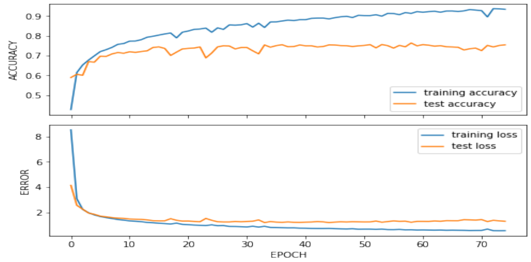
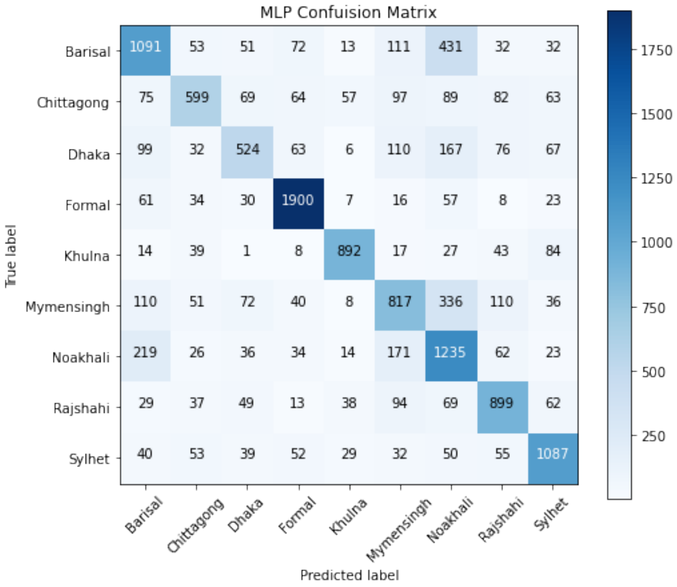


Fig. 24. Accuracy and loss graph of LSTM

E. Confusion Matrix

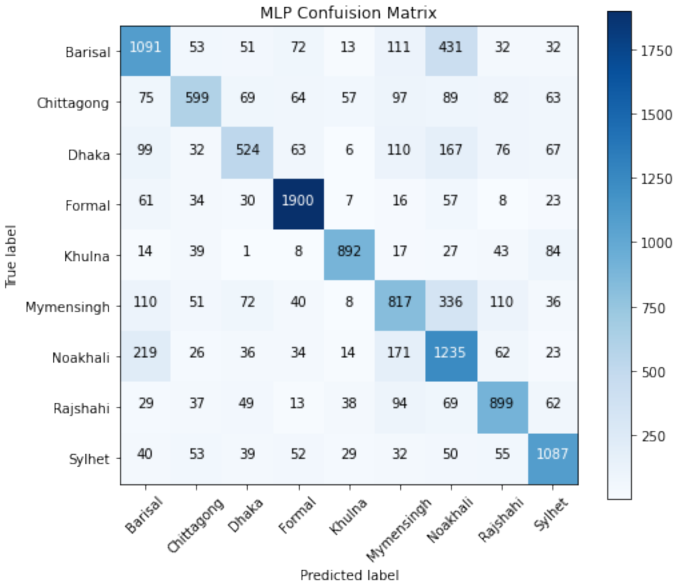
Confusion matrix is a great way to thoroughly compare different models. It enables one to judge the accuracy of individual classes of the output layer. In the confusion matrices shown here, the rows means true labels and the columns are predicted labels. There, in a perfect confusion matrix only the diagonal cells will be the darkest and the rest would be white. Confusion matrix also allows one to see which classes are mislabeled the most and which are the least. This can help to pinpoint some problems in the dataset.

The confusion matrices of the MLP, Hybrid CNN and LSTM models are shown in 25, 26 and 27 respectively. It seems that, the diagonal values are darkened the most. However, we can find very similar pattern between three of the matrices. All three of the models are mostly predicting Barisal as Noakhali and vice versa. Moreover, same pattern can be seen between Barisal and Mymensingh. From the percentage values, the improvement from MLP to CNN and then CNN to LSTM can be seen quite easily. Although the accurate predictions numbers were increased from model to model but the pattern among the three different confusion matrices are quite the same. From these matrices, it can be concluded that in order to improve the accuracy even more the samples from Mymensingh, Barisal and Noakhlai needs to be checked manually. Furthermore, not even a single mispredicted cell contained 0. Meaning, for every single actual region the models predicted all the other regions at least once. This means, in many cases the models are just randomly predicting a region. Most likely, the reason behind this is how the dataset was generated and it was not well built. Most importantly, the dataset does not contain isolated speeches. Using isolated speech the models would have a higher possibility of reaching their full potential.



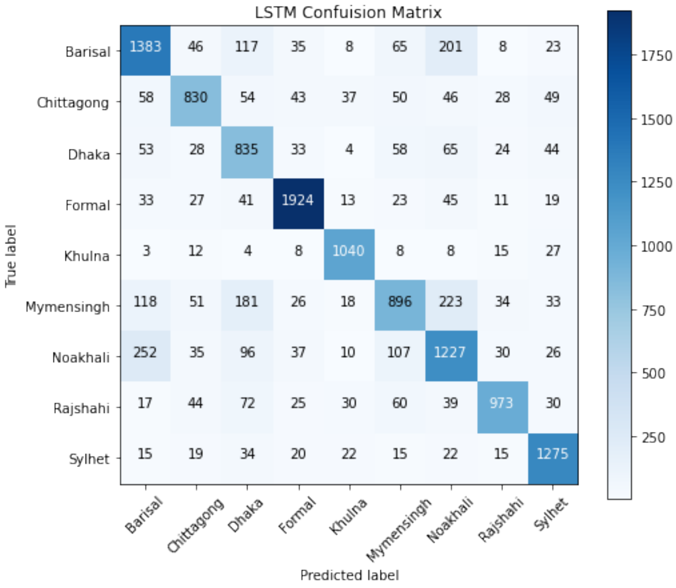
	Barisal	Chittagong	Dhaka	Formal	Khulna	Mymensingh	Noakhali	Rajshahi	Sylhet
Barisal	1091	53	51	72	13	111	431	32	32
Chittagong	75	599	69	64	57	97	89	82	63
Dhaka	99	32	524	63	6	110	167	76	67
Formal	61	34	30	1900	7	16	57	8	23
Khulna	14	39	1	8	892	17	27	43	84
Mymensingh	110	51	72	40	8	817	336	110	36
Noakhali	219	26	36	34	14	171	1235	62	23
Rajshahi	29	37	49	13	38	94	69	899	62
Sylhet	40	53	39	52	29	32	50	55	1087

Fig. 25. Confusion matrix of MLP



	Barisal	Chittagong	Dhaka	Formal	Khulna	Mymensingh	Noakhali	Rajshahi	Sylhet
Barisal	1091	53	51	72	13	111	431	32	32
Chittagong	75	599	69	64	57	97	89	82	63
Dhaka	99	32	524	63	6	110	167	76	67
Formal	61	34	30	1900	7	16	57	8	23
Khulna	14	39	1	8	892	17	27	43	84
Mymensingh	110	51	72	40	8	817	336	110	36
Noakhali	219	26	36	34	14	171	1235	62	23
Rajshahi	29	37	49	13	38	94	69	899	62
Sylhet	40	53	39	52	29	32	50	55	1087

Fig. 26. Confusion matrix of Hybrid CNN



	Barisal	Chittagong	Dhaka	Formal	Khulna	Mymensingh	Noakhali	Rajshahi	Sylhet
Barisal	1383	46	117	35	8	65	201	8	23
Chittagong	58	830	54	43	37	50	46	28	49
Dhaka	53	28	835	33	4	58	65	24	44
Formal	33	27	41	1924	13	23	45	11	19
Khulna	3	12	4	8	1040	8	8	15	27
Mymensingh	118	51	181	26	18	896	223	34	33
Noakhali	252	35	96	37	10	107	1227	30	26
Rajshahi	17	44	72	25	30	60	39	973	30
Sylhet	15	19	34	20	22	15	22	15	1275

Fig. 27. Confusion matrix of LSTM

F. Accuracy, Precision, Recall & F1 Score

There is no single easy parameter based on which AI models can be judged and be compared with other AI models. Some models might be better at predicting specific classes so naturally it gives high accuracy for those cases whereas other models might just give a overall high value of accuracy. Especially when the accuracies and the confusion matrices are quite similar. However, information extracted from the confusion matrix can be used to get such values which can help to compare the models of this paper quite thoroughly. These are individual_accuracy, precision, recall and F1 score.

For definitions below, TP, TN, FP and FN means True Positive, True Negative, False Positive and False Negative respectively.

1) Individual Accuracy: Individual accuracy is calculated by converting the matrix into a (2,2) matrix where only except for the current cell everything else is reduced and formula is applied. The numerator means all the correct results and the denominator mean the total amount of testing data. Like the True Positive and True Negative are the actually corrected predicted values. That is how, accuracy of individual classes are calculated.

$$\begin{aligned}&Individual\_Accuracy\\&= (TP + TN)/Total\_Testing\_Data\\&= Total\_Correct\_Predictions/Total\_Testing\_Data\end{aligned}$$

2) Precision: Precision of a certain output class basically means, among all the predicted positive results how many were actually correctly predicted. That is why in the numerator True Positive and the Denominator contains Total_Predicted_Value. It helps to point out the precise accuracy of that specific output class excluding rest of the classes.

$$\begin{aligned}Precision &= TP/(TP + FP)\\&= TP/Total\_Predicted\_Positive\end{aligned}$$

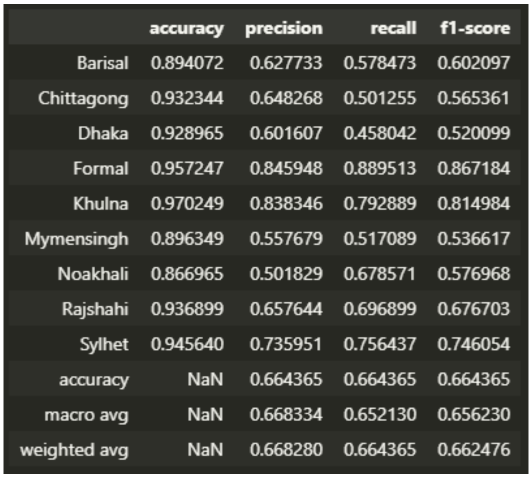
3) Recall: Recall value of a certain output class calculates the rate of prediction among the actually correct values. In the equation it can be seen that the numerator is True Positive value and the numerator is the Total Actual Positive for the current output class only.

$$\begin{aligned}Recall &= TP/(TP + FN)\\&= TP/Total\_Actual\_Positive\end{aligned}$$

4) F1 Score: F1 Score is calculated by using the Precision and Recall values of a specific output class. In the equation the value of β means how many times Recall is more important than precision. For the F1 Score shown here the value of β was 1, which means precision and recall both were equally prioritized while calculating the F1 Score. F1 Score can be thought of as an overall judgement of the model's performance for a specific output class.

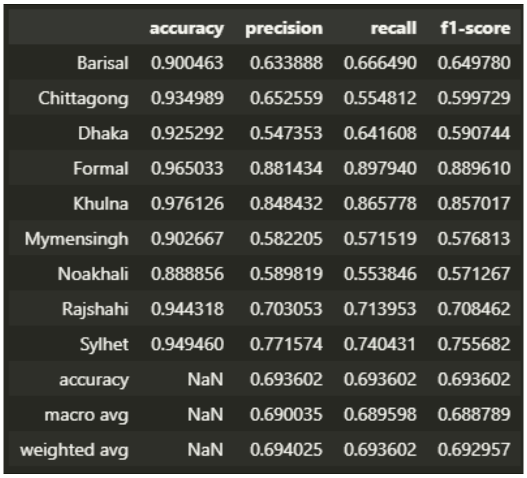
$$F1\_Score = (1 + \beta^2) * \frac{Precision * Recall}{\beta^2 * Precision + Recall}$$

Finally, with these definitions it is possible to get a good enough idea to compare the three different model which are created. The individual accuracy, precision value, recall value and f1 score of the different regions are shown in figures 28, 29, 30. The figure 28 shows the performance measurement of the MLP architecture. Moreover, figure 29 and figure 30 shows the performance measurements of the Hybrid CNN and LSTM respectively. First of all, individual accuracy is quite good in the each of the models which should be quite obvious. But precision, recall and f1 score is lower compared to the accuracy value. By analysing these values it can clearly be stated that the LSTM architecture is the best model among the three.



	accuracy	precision	recall	f1-score
Barisal	0.894072	0.627733	0.578473	0.602097
Chittagong	0.932344	0.648268	0.501255	0.565361
Dhaka	0.928965	0.601607	0.458042	0.520099
Formal	0.957247	0.845948	0.889513	0.867184
Khulna	0.970249	0.838346	0.792889	0.814984
Mymensingh	0.896349	0.557679	0.517089	0.536617
Noakhali	0.866965	0.501829	0.678571	0.576968
Rajshahi	0.936899	0.657644	0.696899	0.676703
Sylhet	0.945640	0.735951	0.756437	0.746054
accuracy	NaN	0.664365	0.664365	0.664365
macro avg	NaN	0.668334	0.652130	0.656230
weighted avg	NaN	0.668280	0.664365	0.662476

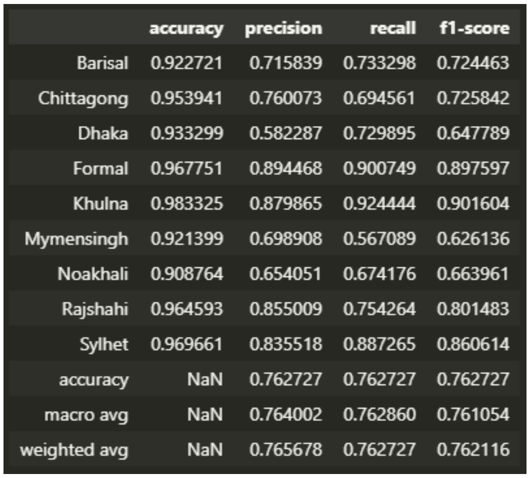
Fig. 28. Performance of each region in MLP



	accuracy	precision	recall	f1-score
Barisal	0.900463	0.633888	0.666490	0.649780
Chittagong	0.934989	0.652559	0.554812	0.599729
Dhaka	0.925292	0.547353	0.641608	0.590744
Formal	0.965033	0.881434	0.897940	0.889610
Khulna	0.976126	0.848432	0.865778	0.857017
Mymensingh	0.902667	0.582205	0.571519	0.576813
Noakhali	0.888856	0.589819	0.553846	0.571267
Rajshahi	0.944318	0.703053	0.713953	0.708462
Sylhet	0.949460	0.771574	0.740431	0.755682
accuracy	NaN	0.693602	0.693602	0.693602
macro avg	NaN	0.690035	0.689598	0.688789
weighted avg	NaN	0.694025	0.693602	0.692957

Fig. 29. Performance of each region in Hybrid CNN

In our opinion, the sporadic results in the confusion matrix and the low F1 score are caused by irregularities in the dataset. These models have a great possibility of performing much better if a properly created and cleaned dataset was given which would contain isolated speeches from different regions.



	accuracy	precision	recall	f1-score
Barisal	0.922721	0.715839	0.733298	0.724463
Chittagong	0.953941	0.760073	0.694561	0.725842
Dhaka	0.933299	0.582287	0.729895	0.647789
Formal	0.967751	0.894468	0.900749	0.897597
Khulna	0.983325	0.879865	0.924444	0.901604
Mymensingh	0.921399	0.698908	0.567089	0.626136
Noakhali	0.908764	0.654051	0.674176	0.663961
Rajshahi	0.964593	0.855009	0.754264	0.801483
Sylhet	0.969661	0.835518	0.887265	0.860614
accuracy	NaN	0.762727	0.762727	0.762727
macro avg	NaN	0.764002	0.762860	0.761054
weighted avg	NaN	0.765678	0.762727	0.762116

Fig. 30. Performance of each region in LSTM

VI. Future Works and Conclusion

A. Future Work

There are still quite a lot of future works that can be done using the models described in this paper.

First of all, the networks were only trained on the JSON dataset which contained only the MFCC features of the audio files. Although, theoretically MFCCs can represent all the major features of a audio sample but other features except for MFCCs which are described in III-D can also be used to train these networks alongside the MFCC features. The CSV dataset was generated which contained extra 5 features including the MFCC features. This CSV dataset can be applied in the neural network architectures described in this paper. By doing so, it is possible to get a greater accuracy, better performance and more insight on some of the irregularities in the dataset.

Secondly, only deeplearning techniques was used in this paper to generate the models. There are various Machine Learning algorithms as well which is generally used for classifying speech audio data. In the future, those algorithms can be implemented on the dataset to get a completely different result from the ones explained in this paper.

Lastly, as it can be seen in the result and analysis section V-D that slowly the accuracy and the overall performance improved when more and more complex neural network architectures were used. However, there is far more room for improvements. A similar pattern can be seen in all the confusion matrices no matter what architecture was used. This common pattern was described in V-E that none of the mispredicted cells contained 0 in it. It was further stated that this was a result of how the dataset was generated and it does not isolated speeches. Therefore, by creating a brand new dataset containing isolated speeches from different regions and using that dataset to train this model will definitely result in far more robust models which would generate a more cleaner and better confusion matrices, highly improved accuracy and F1 score.

B. Conclusion

Automatic speech recognition systems are getting embedded in our lives more and more nowadays. As it helps with fluent workflow reducing the manual labor required to interact with machines, it should have very minimal to no errors. A great barrier to achieve this is various accents of a language. Since, even modern ASR models struggle to predict when something other than the formal accent is used. This is a huge challenge for ASR systems as processing accented speech increases error rate. The best solution for this is to create different models for different accents. After classifying the accent, the audio data could be sent to the model which is best suited for the classified accent.

As for the Bengali language, few decent ASR systems have been built. However, almost next to none work has been done to solve the accent issue. The first step towards solving this would be classifying different Bengali accents. The main goal of this paper was to contribute a little in order to solve this problem. Three different deep learning techniques was used create models which resulted in moderate accuracy. Albeit, there is much more room for improvements but the goal was to pave a better way for the future works of Bengali ASR Systems. Training using a fresh dataset consisting of isolated bengali speech from different regions has a great chance of increasing the robustness of these models significantly. Therefore, if this paper can provide some valuable knowledge or insight to anyone seeking to solve the accent issue in Bengali ASR, it will certainly be a worthwhile journey.

Acknowledgment

Firstly, a special thanks to S. M. Saiful Islam [1] for providing us the dataset of their thesis which enabled us to move forward with our thesis.

Secondly, thanks to Valerio Velardo [13] for providing us with in-depth tutorial on how to analyze and extract features from audio data and implement deep learning techniques.

Finally, thanks to our supervisor Moin Mostakim sir and our co-supervisor Warida Rashid ma'am for their kind support and advice in our work. They helped us throughout the whole research period.

References

[1] S M Badhon et al. "Bengali Accent Classification from Speech Using Different Machine Learning and Deep Learning Techniques". In: Jan. 2021, pp. 503–513. ISBN: 978-981-15-7393-4. DOI: 10.1007/978-981-15-7394-1_46.

[2] *Bengali dialects*. May 2021. URL: https://en.wikipedia.org/wiki/Bengali_dialects.

[3] *Bengali language*. URL: https://www.britannica.com/topic/Bengali-language.

[4] Too Chen et al. “Automatic accent identification using Gaussian mixture models”. In: *IEEE Workshop on Automatic Speech Recognition and Understanding, 2001. ASRU ’01*. 2001, pp. 343–346. DOI: 10.1109/ASRU.2001.1034657.

[5] Jon Fingas. *Voice assistants still have problems understanding strong accents*. May 2021. URL: https://www.engadget.com/2018-07-19-voice-assistant-problems-understanding-accents.html.

[6] J.H.L. Hansen and L.M. Arslan. “Foreign accent classification using source generator based prosodic features”. In: *1995 International Conference on Acoustics, Speech, and Signal Processing*. Vol. 1. 1995, 836–839 vol.1. DOI: 10.1109/ICASSP.1995.479824.

[7] Romain Hennequin et al. “Spleeter: a fast and efficient music source separation tool with pre-trained models”. In: *Journal of Open Source Software* 5.50 (2020). Deezer Research, p. 2154. DOI: 10.21105/joss.02154. URL: https://doi.org/10.21105/joss.02154.

[8] Chao Huang, Tao Chen, and Eric Chang. “Accent Issues in Large Vocabulary Continuous Speech Recognition: Special Double Issue on Chinese Spoken Language Technology”. In: *International Journal of Speech Technology* 7 (Jan. 2004).

[9] Jiaaro. *jiaaro/pydub: Manipulate audio with a simple and easy high level interface*. URL: https://github.com/jiaaro/pydub.

[10] Liu Wai Kat and P. Fung. “Fast accent identification and accented speech recognition”. In: *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No.99CH36258)*. Vol. 1. 1999, 221–224 vol.1. DOI: 10.1109/ICASSP.1999.758102.

[11] Kasiprasad Mannepalli, P. Nrahari Sastry, and V. Rajesh. “Accent detection of Telugu speech using prosodic and formant features”. In: *2015 International Conference on Signal Processing and Communication Engineering Systems*. 2015, pp. 318–322. DOI: 10.1109/SPACES.2015.7058274.

[12] Martın Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[13] Musikalkemist. *musikalkemist /DeepLearningForAudioWithPython: Code and slides for the ”Deep Learning (For Audio) With Python” course on TheSoundOfAI Youtube channel.* URL: https://github.com/musikalkemist/DeepLearningForAudioWithPython.

[14] C. Teixeira, I. Trancoso, and A. Serralheiro. “Accent identification”. In: *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP ’96*. Vol. 3. 1996, 1784–1787 vol.3. DOI: 10.1109/ICSLP.1996.607975.

[15] Felix Weninger et al. “Deep Learning Based Mandarin Accent Identification for Accent Robust ASR”. In: *Proc. Interspeech 2019*. 2019, pp. 510–514. DOI: 10.21437/Interspeech.2019-2737. URL: http://dx.doi.org/10.21437/Interspeech.2019-2737.

[16] Shilei Zhang and Yong Qin. “Semi-supervised accent detection and modeling”. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2013, pp. 7175–7179. DOI: 10.1109/ICASSP.2013.6639055.

[17] Yanli Zheng et al. “Accent detection and speech recognition for Shanghai-accented Mandarin.” In: Jan. 2005, pp. 217–220.