Classification of Bangla Regional Languages and Recognition of Artificial Bangla Speech using Deep Learning

by

Prommy Sultana Hossain
20166014

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
M.Sc. in Computer Science and Engineering

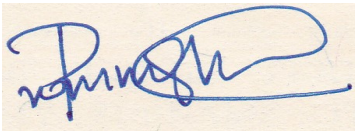
Department of Computer Science and Engineering
School of Data and Sciences
Brac University
March 2022

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:



Prommy Sultana Hossain
20166014
prommy.sultana.ferdawoos.hossain@g.bracu.ac.bd

Approval

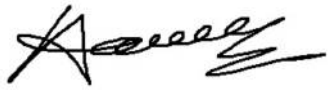
The thesis/project titled "Classification of Bangla Regional Language and Artificial Bangla Speech Recognition using Deep Learning recognition" submitted by

Prommy Sultana Hossain (20166014)

Of Spring, 2022 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of M.Sc. in Computer Science and Engineering on April 11, 2022.

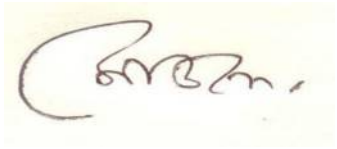
Examining Committee:

Supervisor:
(Member)



Dr. Amitabha Chakrabarty
Associate Professor
Department of Computer Science and Engineering
School of Data and Sciences
Brac University
amitabha@bracu.ac.bd

Examiner:
(External)



Dr. Md. Ekramul Hamid
Professor
Department of Computer Science and Engineering
University of Rajshahi

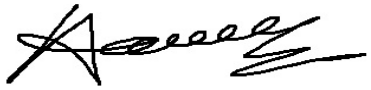
Examiner:
(Internal)

Dr. Md. Golam Rabiul Alam
Associate Professor
Department of Computer Science and Engineering
School of Data and Sciences
Brac University
rabiul.alam@bracu.ac.bd

Examiner:
(Internal)

Dr. Muhammad Iqbal Hossain
Assistant Professor
Department of Computer Science and Engineering
School of Data and Sciences
Brac University
iqbal.hossain@bracu.ac.bd

Program Coordinator:
(Member)



Dr. Amitabha Chakrabarty
Associate Professor
Department of Computer Science and Engineering
School of Data and Sciences
Brac University
amitabha@bracu.ac.bd

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi
Associate Professor
Department of Computer Science and Engineering
School of Data and Sciences
Brac University
skazi@bracu.ac.bd

Abstract

Since 1970, researchers have been attempting to recognize and comprehend spontaneous speech. For an automatic voice recognition system, many techniques were employed. People always choose English for voice recognition since it has been the subject of the majority of study and implementation. However, Bangla is fifth most widely spoken languages in the world. Bangla regional language voice recognition has the potential to have a significant influence on human-computer interaction and internet of things applications. Majority of the research performed in the past decade in Bangla speech recognition involves classification of age, gender, speaker identification and detection of specific words. However, classification of regional Bangla language from Bangla speech and the identification of artificial Bangla speech has not been researched heavily before. Due to the limitation of grammatical and phonetic database with various Bangla regional language. Hence the author of this paper has created 30 hours of Bangla regional language speech dataset, that covers the dialect spoken by the locals in seven districts/division of Bangladesh. Bangla speech was generated, by first converting Bangla words to English word aberration (used often as text language) that would ultimately translate to a English phrase. Additionally, to classify the regional language spoken by the speaker in the audio signal and determine its authenticity, the suggested technique was used. Stacked convolutional autoencoder (SCAE) and sequence of multi-label extreme learning machines (MLELMs). SCAE section of the model creates a detailed feature map from Mel Frequency Energy Coefficients (MFECs) input data by identifying the spatial and temporal salient qualities. Feature vector is then fed to the first MLELM network to produce soft classification score for each data. Based on which the second MLELM network would generate hard labels. The suggested method was excessively trained and tested on unsupervised data which is the formation of new sentence from the unique Bangla/English abbreviation words. The model is also able to categorization speaker's characteristic such as; age and gender. Through experimentation it was found that the model generates better accuracy score label for regional language with taking age class into consideration. As aging generates physiological changes in the brain that alter the processing of aural information, increasing classification accuracy from 75% to 92% without and with age class consideration, respectively. This was able to be achieved due to the usage of MLELMs networks, input data is a multi labeled dataset, that classify labels based on linked patterns between classes. The classification accuracy for synthesised Bangla speech labels 93%, age 95%, and gender class label 92%. The proposed methodology works well with English speech audio-set as well.

Keywords: Convolutional Autoencoder; Extreme Learning Machine; Bangla regional language; Speech Recognition

Acknowledgement

Firstly, all praise to the Great Almighty for whom my thesis have been completed regardless of the major interruption.

Secondly, my heartfelt appreciation to my supervisor **Dr. Amitabha Chakrabarty**, Associate Professor, Department of Computer Science and Engineering, Brac University, for his patience and support throughout this journey.

Thirdly, a wholehearted gratitude towards family members without their tremendous support and motivation I would not have been able to achieve any of this.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

AE Autoencoder

ASR Automatic Speech Recognition

$bLSTM$ bidirectional Long Short-Term Memory Recurrent Neural Network

CAE Convolutional Autoencoder

CER Character Error Rate

CNN Convolutional Neural Network

CSR Continuous speech recognition

CV Cross-Validation

DAE Dense Autoencoder

DCT Discrete Cosine Transform

DL Deep Learning

DNN Deep Neural Network

FN False Negative

FP False Positive

FS F1-score

$G2P$ Grapheme-to-Phoneme

GMM Gaussian Mixture Model

HCI Human-Computer Interaction

HMM Hidden Markon Model

ISR Isolated speech recognition

$LSTM - RNN$ Long Short-Term Memory Recurrent Neural Network

MAP Maximum a posterior

$MConv - LSTM$ Multi-channel Convolutional-Long Short-Term Memory Recurrent Neural Network

$MFCC$ Mel-Frequency Cepstral Coefficients

$MFEC$ Mel Frequency Energy Coefficients

ML Machine Learning

$MLELM$ Multi-Label Extreme Learning Machine

MLP Multi-layers preceptron

MOS Mean Opinion Score

MSE Mean Squared Error

NaN Not-a-Number

NLP Natural Language Processing

NN Neural Network

NNC Nearest Neighbor Classifier

P Precision

$PESQ$ Preceptual Evaluation of Speech Quality

R Recall

$ReLu$ Rectified Linear Unit

$RNNs$ Recurrent Neural Networks

SAE Stacked Autoencoder

$SCAE$ Stacked Convolutional Autoencoder

SR Speech Recognition

SVM Support Vector Machine

TN True Negative

TP True Positive

TTS Text-To-Speech

ANN Artificial Neural Network

ELM Extreme Learning Machine

Chapter 1

Introduction

Human voice is the most extensively used type of communication between humans and the machines they run on a global scale. Individual dialects are what bind individuals together. Folks can convey their ideas more effectively using such languages. The capacity of a machine or computer software to recognize phrases and words in spoken language and translate them into machine-readable form is known as speech recognition as stated by Nagrani et al. [2]. A voice signal not only carries information about the content of speech, but it also provides information about the speaker's identity, emotions, age, gender, and geographical area of origin. Voice signals are also important in the field of human–computer interaction (HCI).

Autonomously extracting characteristics from an audio stream has recently been a hot focus of research for the Bangla language [1][11][22][23][26-35][37]. The development of applications such as marketing based on consumer regional language, age/gender, and a caller-agent coupling in contact centers that correctly allocates agents based on the caller identity is made possible by the accurate and efficient extraction of a speaker identification from a voice signal. Bangla languages have been the subject of several research. However, in majority of these study articles, there is no categorization of the speaker's regional language or of synthetic or genuine voices in Bangla speech audio signals. Rahut [33], Sharma [13], and Gutkin [32] show that the outcome is a defective Automatic Speech Recognition (ASR) and Text-To-Speech (TTS) system for the Bangla language. Bangla sentence grammar and phonetic construction differ from English sentence grammar and phonetic construction. In Bangla phrases, the auxiliary verb is not utilized, and the subject comes before the object, as Ohi [26] points out. Speech recognition in Bangla requires a database with a big vocabulary and phoneme patterns, which is not accessible in the public or private datasets. Therefore, this is one of the key reasons why authors have been unable to classify the speaker's regional language.

1.1 Motivation behind Regional Language from Bangla Speech

The goal of human conversation is to help the other person grasp the depth of the uttered words, not only to communicate words from one person to another but to

understand the depth of the content of the speech. Humans not only analyze the information delivered to the ears while interpreting speech, but they also judge the information based on the context of the information. As a result, even in a loud setting, humans can readily interpret spoken language. Due to the dynamic nature of spoken languages, computer recognition of speech is extremely challenging. The Bangla language is well-known around the world, and it is the fifth most spoken language on the planet [2]. The population of Bangladesh speak two different varieties of Bangla. Only a few people speak the local language of the region in which they live. The mainstream Bangla language, which is spoken by about 290 million people, is the other variety. The population of Bangladesh speak 55 regional languages among the 64 districts. A regional language, often known as a dialect, is a language that a child learns organically without the use of written grammar and that varies by location [7]. It is a characteristic of a language that is widely spoken in a certain location that creates morphological differences in the sounds of the ideal language or literary language. The Bangla language may be split into six classes: Bangla, Manbhumi, Varendri, Rachi, Rangpuri, and Sundarbani, although having regional language variances. Seven regional languages were primarily studied for the purposes of this study; Khulna, Bogra, Rangpur, Sylhet, Chittagong, Noakhali, and Mymensingh divisions. A person's regional language is identified by the wave frequency (pronunciation) of a word pronounced in Bangla.

1.2 Motivation behind Artificial Bangla Speech

Any utterance manufactured by a computer is referred to as synthetic speech. Synthetic speech is getting closer to sounding natural because to advances in deep learning and other approaches. Even humans have difficulty recognizing actual speech from computer produced speech thanks to several cutting-edge technologies that attain such a high level of naturalness. Furthermore, these technologies enable a person to train a speech synthesizer with a target voice, resulting in a model capable of accurately reproducing that person's voice. Such technology might have negative repercussions, as it is possible to imitate someone's voice maliciously. An example would be training a model using the voice of a well-known person and then utilizing that model to construct an utterance with malevolent content in order to publicly slander the person. Several movies on the internet show this type of imitation, in which both picture and audio were synthesized to create a bogus video. We examine how synthetic speech is created in depth in this study and provide methods for detecting such synthesized utterances.

1.3 Aims and Objectives

To address these issues, the authors of this work created a database including extensive jargon and phoneme patterns from the seven dialects of the division stated above, as well as more than 100,000 Bangla spoken utterances recorded within the institution. The statement in Bangla language spoken by the speakers is free of ambiguities. Labels were applied to the input signals to indicate which class they belonged to.

1.4 Thesis Contribution

To summarize the observations from this research, we compiled a list of key findings in this thesis:

- The proposed model is able to classify an audio signal with MFECs and spectrogram format. It yields increased accuracy predication rate for regional language and synthesized Bangla speech, compared to previous researches.
- Build a dataset with 30+ hours of original and synthesized Bangla and English speech. As currently publicly or privately available dataset does not have high Bangla speech dataset.
- Since we created models for detecting regional language and synthetic voice, an application could later be made (such as a browser plugin or as an extension) that can identify if synthetic audio is being played in a web page or recorded call. This would assist the community by informing listeners whether the audio they are hearing is synthetic or genuine, minimizing the risk of successful impersonation assaults and aid customer services provider to better match caller and agent connections.
- Deep learning approaches for synthetic speech detection have the following advantages: The top-performing deep learning approaches consistently demonstrated better accuracy in all of our studies.

1.5 Organization of the Report

This report is organized in the following way; description about what and how is speech recognition performed, What is deep learning and what are the challenges that comes when running a Neural Network model. Previous researches conducted with Bangla Speech. Dataset preparation and its prepossessing. Detailed description of all parts in the proposed model and experimentation procedures and steps performed. Finally the results obtained and what does it mean for both Bangla and English speech. Lastly, conclusion and limitation and future work that be done.

Chapter 2

Speech Recognition

Around the world, speech recognition is a hot field of research. Speech recognition is a hot topic among scientists and academics. Speech recognition is primarily done in numerous languages across the world, with English being the most common. The majority of the world's languages have their own speech recognizers. However, speech recognizers aren't available in Bangla, our mother tongue. A little amount of research has been done on Bengali speech recognizers, however the results have been disappointing. Our major aim throughout the thesis study has been to implement categorization of Bangla Regional language and artificially synthesized Bangla voice. However, due to the scarcity of Bangla speech resources, we constructed a database containing seven regional languages. We attempted to learn about several technologies during the course of the project, and we decided to employ Stacked Convolutional Autoencoder (CAE) with Multi-label Extreme Learning Machine. CAE is a widely utilized technology that is rising in popularity and performance across many domains. We proceeded to study and prepare the tools, as well as the data and files/scripts that we would need to train, decode, and test the system. The whole report details all of the measures that were taken. But before we get started, let's go over the fundamentals.

2.1 Speech Recognition basics:

The act of turning an acoustic signal acquired by a microphone or a telephone into a collection of words is known as speech recognition (SR). It's a wide word that suggests it can identify practically anyone's speech, but it takes a lot of training data to make the computer voice-independent. There are two fundamental types of SR:

- Isolated speech recognition - ISR
- Continuous speech recognition - CSR

A continuous voice recognition system does not need the speaker to pause between words, whereas an isolated-word speech recognition system must. Continuous speech is made up of a series of utterances that are representative of genuine speech. A phrase made up of linked words, on the other hand, does not resemble genuine speech because it is made up of isolated words. The assumption in Isolated Word is that the speech to be identified consists of a single word or phrase, and that it should

Figure 2.1: Overview of Bangla speech Recognition system

be recognized as a complete entity with no explicit knowledge or consideration for the phonetic content of the word or phrase. As shown in the figure 2.1.

Some terminology that are mentioned throughout the study and that you should be familiar with in order to comprehend SR technology are:

- Utterance
  A vocalization (saying) of a word or words that reflect a single meaning to the computer is referred to as an utterance. A single word, a few words, a sentence, or even several sentences can be used as utterances.

- Speaker Dependence
  Systems that are speaker dependent are built around a single speaker. They are more accurate for the correct speaker, but not for other speakers. They anticipate that the speaker will talk in a constant tone and speed. Speaker independent systems are intended to accommodate a wide range of speakers. Adaptive systems often begin as speaker-independent systems and then use training approaches to adapt to the speaker in order to improve recognition accuracy.

- Vocabularies
  The SR system recognizes words or utterances from vocabularies (or dictionaries). Smaller vocabulary are simpler to identify by computers, whereas bigger vocabularies are more challenging. In contrast to traditional dictionaries, each entry does not have to be a single word. They might be a single phrase or two long. Smaller vocabulary may include only one or two recognized utterances (e.g., "Wake Up"), whereas larger vocabularies may have hundreds of thousands or more.

- Training
  Training is the process of learning the qualities of sound units. Using a set of example speech signals known as the training database, the trainer learns the parameters of sound unit models (TD).

- A Language Dictionary
  Accepted Words in the Language are mapped to sound unit sequences that describe pronunciation, which might include syllabification and stress in some cases.

- A Filter Dictionary
  Non-Speech sounds are mapped to corresponding non-speech or speech like sound units.

- Phone
  In terms of sound units, this is a way of describing the pronunciation of words. The International Phonetic Alphabet, or IPA, is the standard technique for expressing phones. The English language employs an ASCII-based transcription scheme, whereas Bangla uses Unicode characters.

- HMM
  The Hidden Markov Model is a finite set of states, each of which has a (usually multidimensional) probability distribution associated with it. A collection of probabilities known as transition probabilities governs transitions between states. According to the corresponding probability distribution, a result or observation can be produced in a specific condition. It is only the outcome that is visible to an external observer, not the state itself, and therefore states are "hidden" to the outside world; hence the term Hidden Markov Model.

- Language Model
  A probability distribution is used to provide a probability to a series of m words in a language model. We can use a regular grammar to model it.

2.2 Overview of the Full System:

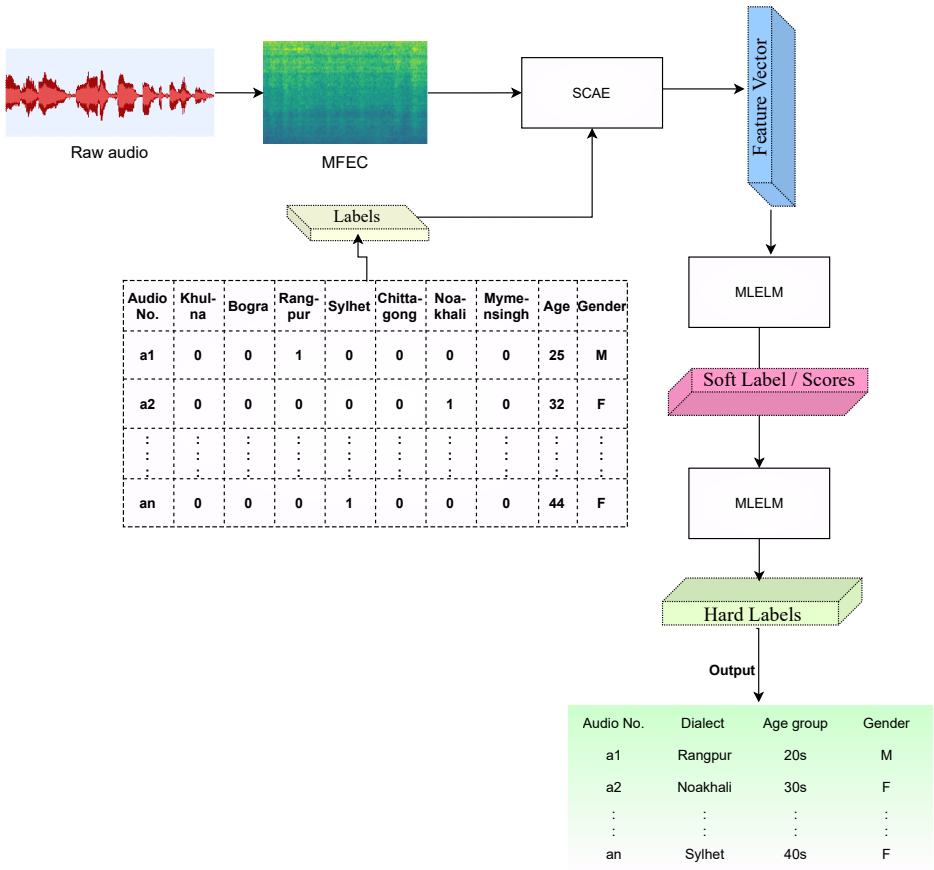


Figure 2.2: Proposed system workflow

The proposed method uses stacked convolutional autoencoder (SCAE) and Multi-label Extreme Learning Machine (MLELM) framework to detect dialect, origi-

nal/synthesized voice, and gender/age from MFEC speech input data. Brief overview of the work can be seen in figure 2.2. Through experimentation with various type of DL models, the best yielded model is a fully connected SCAE with MLELM for soft classification and score approximation for classes. First the raw audio signals are converted to MFECs file format. At the same time the labels are attached with each audio file. The labels consist of Speaker id, Speech id, Sentence id, word ids, the Bangla/English abbreviate sentence that was created with the help of text recognizer build by [69]. And lastly the regional language used by the speaker. This labels tables along with the MFEC data is passed onto the proposed model. Where after extensive training the SCAE model. Detailed feature maps are generated from this input and passed to MLELMs network to predict the labels for each data.

Chapter 3

Deep Learning

Deep learning is a machine learning approach that allows computers to learn by example in the same way that people do. Deep learning is a critical component of self-driving automobiles, allowing them to detect a stop sign or discriminate between a pedestrian and a lamppost. It enables voice control in consumer electronics such as phones, tablets, televisions, and hands-free speakers. Deep learning has gotten a lot of press recently, and for good cause. It's accomplishing accomplishments that were previously unattainable. A computer model learns to execute categorization tasks directly from pictures, text, or sound in deep learning. Deep learning models can attain state-of-the-art accuracy, even surpassing human performance in some cases. A large set of labeled data and neural network architectures are used to train models.

All content used in this section to are taken from Bishop[62]; Bengio[63]; Jones[64]; Karn [65]; Karpathyetal et al. [66]; Talwalkar [67]; Chollet [68]; Ng [60]; Ngand aet al. [61], and Ng [59] unless stated otherwise. I section will explain the origin of machine learning, requirements of a Neural Network and the working of a Convolutional Neural Network.

3.1 Machine Learning Origins

Machine learning is an area of Computer Science that causes a computer to learns autonomously from a given set of data. Through this automatic learning process, the computer will be able to make prediction from unseen data based on the data used during the training of the system. Machine Learning aid to solve complex problems.

To learn, the computer often requires tagged training data, as well as a mechanism to assess the distance between its present output and the predicted output. This measurement offers feedback to the algorithm, allowing it to change its inner workings to get closer to the predicted predictions and, as a result, allowing the system to learn. The method is known as supervised learning when the training data is labeled, however there are other circumstances when the algorithm might learn from unlabeled data. Unsupervised learning is referred to as k-means clustering and autoencoders are examples of such approaches. Machine learning techniques of various kinds are now widely employed globally in sectors such as voice recognition, search engines, and bioinformatics as stated by Sun et al. [56].

3.2 Data fitting and splitting

A machine learning model can be tweaked to nearly perfectly fit the training data, but this does not guarantee that it will provide accurate predictions on previously unknown data. Overfitting is a term used to describe when a model fails to generalize to new data. Underfitting is also a possibility, in which the model fails to grasp the structure in the training data and so fails to make effective predictions on observed data. Overfitting and underfitting will both result in poor prediction outcomes on unseen data and should be avoided. This can be accomplished using a variety of ways, as detailed in section 3.3.5.

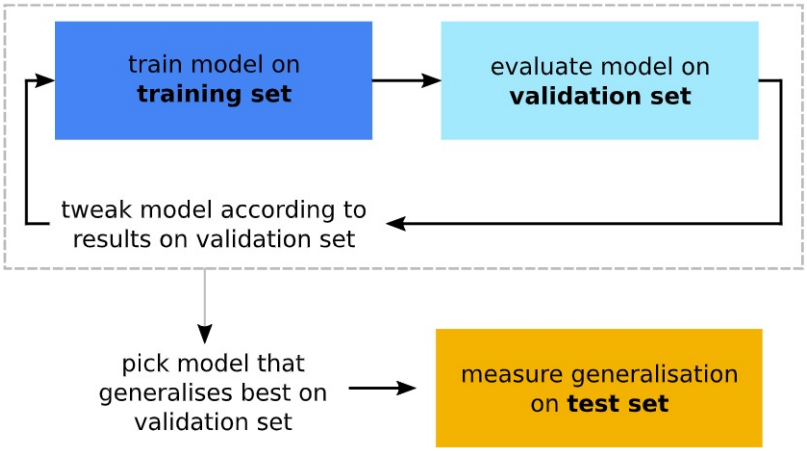


Figure 3.1: Machine learning model fitting work flow using holdout validation.

To test the algorithm's performance, the dataset is often partitioned into three distinct subgroups to get a decent model fit. The training data, validation data, and test data are the three subsets described above. Each of the three datasets should be completely separate from the others and represent the same attributes and structure. The training data are the samples from which the algorithm learns on its own and are used to fit the model. The model recognizes the label each sample in the training set bears and iteratively adjusts its parameters to get the projected outcome closer to the expected.

To assess the present model's fit, a validation set is used. This validation set is seen by the algorithm, but it is not used to train it. This set estimates how the model will perform on unknown data, and by watching its outcomes, tiny adjustments to the model may be made to achieve greater generalisation. Both the validation and training sets go through the algorithm several times, and the constant training, assessing, and tuning is referred to as the training stage. The model with the best predictions on the validation set is picked after numerous trips through this step with the data.

However, because little amounts of knowledge about the validation set leak indirectly into the model, this model should be examined again for its ability to generalize to unknown data. This is due to the model's overfitting caused by frequent modest modifications depending on validation set performance. To avoid this, a test set is

used to assess how effectively the final model generalises to data that the model has not seen directly or indirectly. As a result, the algorithm should only use the test set once. Figure 3.1 depicts the use of several subgroups during holdout validation. Where the dashed box contains the training stage through which both the training and validation data make multiple passes adapted from Google Developers [49].

The data split percentages are chosen based on the amount of samples in the dataset, as well as the model and the number of parameters. Brownlee [43] recommends using k-fold cross-validation (CV) to limit the danger of overfitting on the validation set if the dataset is not too huge. The original dataset is initially divided into two randomly generated sets, known as the train and test sets, using this procedure. For this first division, an 80/20 ratio is typically employed. The test set is set aside to be used as a test set, but the train set is randomly divided into k equally sized and disjunct groups or folds, and it goes through k training phases. A distinct fold serves as the single validation set in each iteration, while the other k1 folds serve as training data. As a validation set, each fold may only be used once.

After fitting the model, a performance measure M_i is assigned to and kept for each iteration i and the model is discarded so that a new model may be trained on the next split of training and validation sets. After k rounds, the average of the overall performance scores is computed to provide an overall performance score $M = \frac{1}{k} \sum_{i=1}^{k} M_i$. This score M indicates how effectively the model can generalize to unknown input, and the global features of the model are optimized depending on its value. Once the model characteristics have been determined, the model is fit to the training dataset as a whole, ignoring the subgroup distribution, and assessed on the reserved test set.

Figure 3.2 depicts a 5-fold CV example. The train and test set makeup the original dataset that has been split using a ratio of 80/20, and the training set is subsequently divided into five groups or folds for use during 5-fold CV. The CV consists of five iterations, where in each iteration i a new model is fitted using four folds as training sets and the fifth fold as a validation set, and a performances core M_i is calculated for the best fitted model. Every iteration,the validation fold switches so every fold is used as a validation set only once. When CV has ended, an overall performances core M is calculated that reflects how well the model is able to generalise to unseen data, and the model is fitted onto the whole training dataset (striped block) and evaluated on the test set that was put aside (yellow block).

3.3 Neural Networks

Artificial neural networks are a set of machine learning techniques that includes deep learning. To learn from the data and solve the problem at hand, it employs various levels of abstraction. The data is broken down into smaller ideas that get more intricate over time, and it is passed through multiple layers that can perform data transformations in order to arrive at a solution to the problem. Figure 3.3 depicts a simplified version of this broken-down concept. In this case, the deep learning algorithm's goal is to detect faces in a photograph, and the system will learn on its own which pixels, edges, and shapes are significant for human face identification and which are not. This is in contrast to other machine learning algorithms, in which

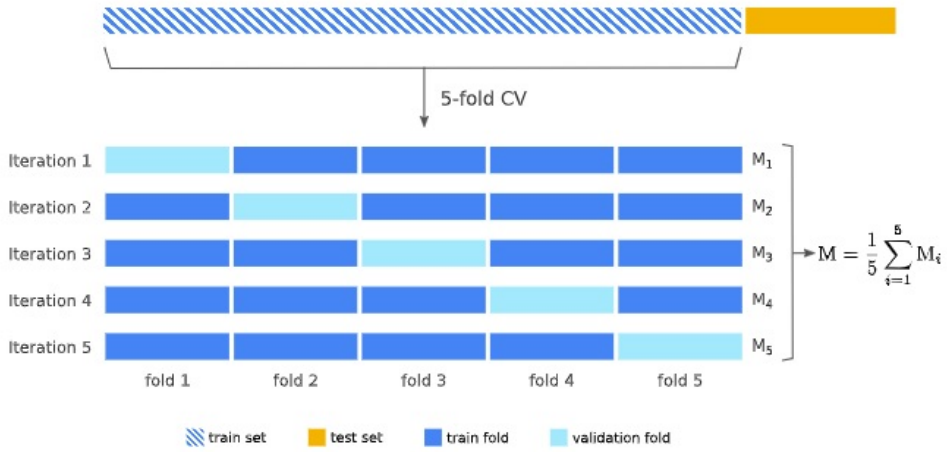


Figure 3.2: Example of k-fold CV with k= 5.

the programmer often does feature extraction manually.

3.3.1 Input data

In supervised learning, the neural network (NN) must learn from training data that consists of a numerical input vector **x** and a numerical true label vector **y**. To fit inside the input vector x, each sample must be the same length. If this is not the case, the longest sample's length is used as the input vector's width, while shorter samples are padded with zeros. If the raw data does not contain numerical values, they must be coded before being sent across the network.

There are other encodings available, including one-hot encoding, which maps distinct values to different bits, and ordinal encoding, which maps values to decimal values between 0 and 1. Varied encodings can result in different study outcomes and have an impact on how effectively the NN can predict. The network is supplied with the input data in batches. The model's internal parameters are changed as a batch comprises a specified amount of samples that transit through the network. Iteration is the term for such a pass. The algorithm repeats this process numerous times until the total number of samples processed through the network equals the number of samples in the training set. The so-called epoch then expires, and the sequence of batches and iterations begins anew.

3.3.2 General layout

A deep learning model may be thought of as a deep NN, a word derived from nature since it mirrors the way organic nerve systems function. The neurons, which may send messages to other neurons in their network, form the computational foundation of both theories. Individual neurons are linked across layers via edges in an artificial NN. Neurons receive signals from either an external source or from a neuron from another layer, and the i^{th} input of a neuron is referred to as x_i, with $x = [x_0, ..., x_n]$ the column vector of all inputs of that neuron. This input vector also has a column vector of weights $w = [w_0, ..., w_n]$ associated with it, with the i^{th} weight of the x_i^{th} input referred to as w_i. The first elements of the input vector x and the weight vec-

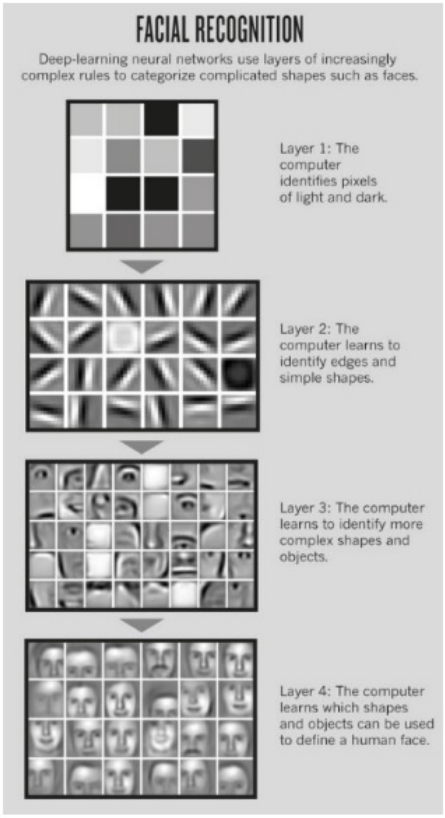


Figure 3.3: Levels of abstraction in a face recognition deep learning algorithm Jones [15].

tor w are special cases as they have no other connections with the network except for the one going into the calculating neuron. These values are respectively called the bias x_0, which has a non-adjustable value of 1, and the bias weight $w - 0$, which has a variable value.

All weights tied to a neuron are associated with the edges across the layers, and indicate how important each input is relative to the others, with higher absolute values indicating a higher importance. They are parameters of the NN that can be adjusted by the algorithm during learning. To produce an output, the neuron calculates the weighted sum $\sum_{i=0}^{n} w_i x_i$ over its inputs, which can also be rewritten as the dot product $w^T x$. After this linear operation, an activation function is applied to provide non-linearity. The choice of activation function can vary and is generally referred to as $f(\cdot)$. By adjusting the bias weight, the neuron is able to translate the activation function. If no translation is needed, the bias weight is simply set to zero. Applying the activation function on the weighted sum leads to following equation to calculate the activated output a in a neuron:

$$\begin{aligned} a &= f\left(\sum_{i=0}^{n} w_i x_i\right) \\ &= f\left(w^T x\right) \end{aligned} \tag{3.1}$$

This output a can be the final output of the NN, or it can be passed on to the next neuron to serve as new input. It can be regarded as a new feature learned by the

neuron based on the already existing features **x**. As an NN consists of a series of these neurons grouped into several layers which transfer their activated output to each other, the NN will learn a hierarchy of features which get adjusted by altering the weights associated with the neurons, and which gradually get more complex as they are a mix of previously learned features. This allows for an NN to create potentially better predictions than more classical machine learning approaches which only work on the original features within the data. An illustration of a single neuron can be found in figure 3.4

An NN may be divided into three types of layers. The input layer is the one that receives the data, and its inputs represent the dataset's original characteristics. The features are sent from this layer to a hidden layer. A hidden layer is made up of multiple neurons that add up the activated weighted total of their inputs before passing it on to the next layer. Another concealed layer or an output layer might be the following layer. The output layer is similar to a hidden layer, but as it is the last layer within the network, it produces the predictions. For a regression problem, only one neuron is needed that outputs a single value $\hat{y}\epsilon R$. For classification problems, the number of neurons is equal to the number of classes K within the input data. Per sample, the neurons give back a probability vector $\hat{y}\epsilon R^K$ representing how sure the NN is that the fed data belongs to each class. This vector's probabilities are all in the [0, 1] range and add up to 1. One neuron in the output layer suffices for binary classification, which is a specific instance. Figure 3.5 depicts a fully linked DNN with three hidden layers. The architecture of the network refers to the entirety of the number and types of layers, their number of neurons and activation functions, as well as how they are interconnected.

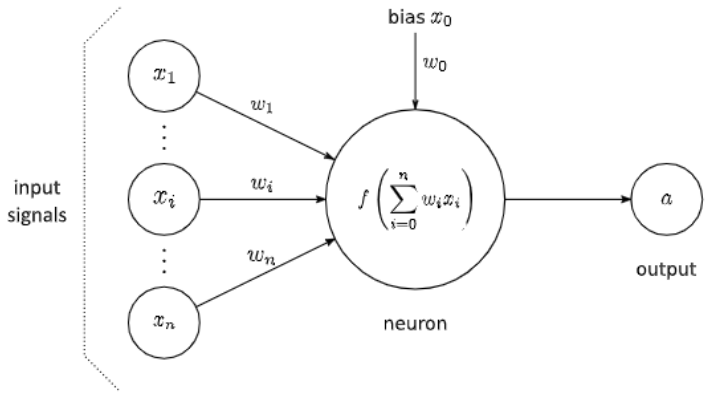


Figure 3.4: Visual representation of a single neuron in a NN.

$$\begin{aligned} a_{j,k+1} &= f\left(\sum_{i=0}^{n} w_{(i,k),(j,k+1)}.a_{i,k}\right) \\ &= f\left(w^T_{k,(j,k+1)} a_k\right) \end{aligned} \tag{3.2}$$

Equation(3.1) can now be redefined for the output $a_{j,k+1}$ of a single neuron j within hidden layer $k+1$ with $w_{(i,k)(j,k+1)}$ the connection weight from neuron i in layer k to neuron j in layer $k+1, a_{i,k}$ the output of neuron i in layer k, $w_{k,(j,k+1)} =$

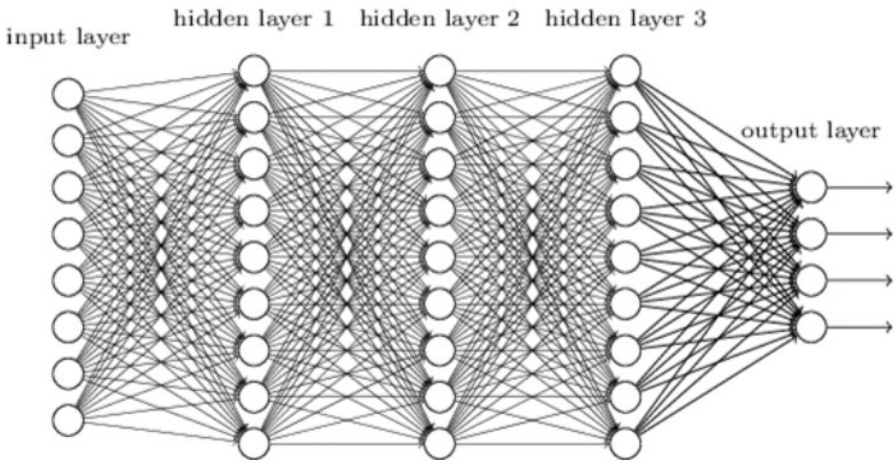


Figure 3.5: Fully connected feedforward DNN with three hidden layers Ho [16].

$[w_{(0,k),(j,k+1)}, ..., w_{(n,k),(j,k+1)}]^T$ the column vector holding all the weights of the connections coming from then neurons in layer k into neuron j in layer *k+1*, and $a_k = [a_{0,k}, ..., a_{n,k}]^T$ the column vector holding all the outputs from the n neurons in layer k. Note that in the last two column vectors $w_{(0,k),(j,k+1)}$ and $a_{0,k}$ are special cases that correspond to respectively the variable weight and fixed value of the bias of neuron j in layer *k + 1*.

As the output vector a_k is needed to calculate the output $a_{j,k+1}$ of a single neuron j within hidden layer *k+1*, this output vector should also be defined. This is done by associating the current layer with the output of the previous layer, so that the output vector a_{k+1} for a hidden layer *k+1* is given by:

$$a_{k+1} = f\left(w_{k,k=1}^T a_k\right) \tag{3.3}$$

with $w_{(i,k),(j,k+1)}$ the connection weight from neuron i in layer k to the j_{th} neuron in layer *k+1*, $a_{i,k}$ the output of neuron i in layer k, $a_k = [a_{0,k}, ..., a_{n,k}]^T$ the column vector holding all the outputs from the n neurons in layer k, and $W_{k,k+1}$ an $n \times m$ weight matrix associated with the biases of the m neurons in layer $k + 1$ and the connections from the n neurons in layer k going into the m neurons in layer $k + 1$ or can be prepestend in a matrix form.

With $w_{(i,k),(j,k+1)}$ the weight of the connection from neuron i in layer k to neuron j in layer k + 1, and $w_{k,(j,k+1)} = [w_{(0,k),(j,k+1)}, ..., w_{(n,k),(j,k+1)}]^T$ the column vector holding all the weights of connections coming from the n neurons in layer k into the j^{th} neuron in layer k+1. The first row of this matrix holds the bias weights for the m neurons in layer k + 1.

Equation(3.3) can be used to associate the output layer $k = L + 1$ and its prediction outputs $\hat{y}$ with all the previous layers, up until the first hidden layer $k = 1$ whose output depends on the input vector x. The network is then represented as a composition of a series of activation functions, such that:

$$\begin{aligned}\hat{y} &= f_{L+1}(WL, L + 1^T a_L) \\ &= f_{L+1}(WL, L + 1^T f_L(WL - 1, L^T a_{L-1})) \\ &= f_{L+1}(WL, L + 1^T f_L(WL - 1, L^T ... f_{k+1}(W_{k,k+1}^T f_k(W_{k-1,k}^T ... f_1(W_{0,1}^T x))))) \\ &= h(x, W)\end{aligned} \tag{3.4}$$

with f_k the activation function used in the k^thlayer of the network, and $W = [W_{0,1}, \cdots, W_{k,k+1}, \cdots, W_{L,L+1}]$ the matrix holding all the weight matrices associated with each layer. A DNN can thus be regarded as implementing a function $\hat{y} = h(x, W)$ that maps a set of inputs x to a set of outputs $\hat{y}$, controlled by a matrix W holding the adjustable weight and bias weight parameters. As each layer needs the previous one to calculate its outputs, data flows through the network in a feedforward manner. No connections are found between neurons within the same layer or across non-consecutive layers, although special network structures exist with feedback loops such as recurrent neural networks (RNNs). Initially, the weights of the NN are set to random values, and the algorithm alters them by comparing the final predictions $\hat{y}$ to the true values y. This comparison is done by the use of a cost or loss function $J(W) = L(W) = L(y, \hat{y})$ which expresses the importance of the errors that are made. The cost function is what the algorithm needs to minimise in order to come closer to the expected output. As the only variable values in the cost function are the weights W, an optimal weight matrix W^* exists which will result in the smallest loss possible. It is found by minimising the loss function:

$$W^* = arg_W min\iota(W) \tag{3.5}$$

The optimal predication vector $\hat{y}^*$ is then defined by:

$$\hat{y}^* = h(x, W^*) \tag{3.6}$$

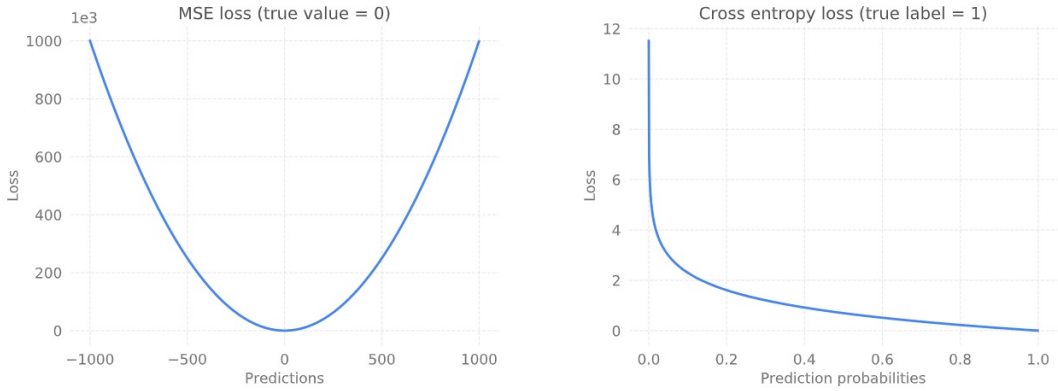
3.3.3 Loss functions and their minimisation

The MSE function and the cross-entropy function are the two most essential and often utilized loss functions. They have the following formulae and are employed in regression and classification issues, respectively:

$$\iota_M SE(W) = \frac{1}{n-1} \sum_{i=0}^{n-1} (y_i - \hat{y_i})^2 \tag{3.7}$$

$$\iota_c ross - entropy(W) = -\sum_{i=0}^{n-1} \sum_{c=1}^{M} y_{i,c} \log \hat{y_{i,c}} \tag{3.8}$$

with n the total number of samples present in the dataset, M the number of classes with in the dataset, $y_{i,c}$ a binary indicator showing if class c is the correct classification for sample i, and $\hat{y_{i,c}}$ the predicted probability of sample i belonging to class c. The MSE loss will result in a high loss when the predicted value is far away from the true value, and the cross-entropy loss punishes uncertain prediction probabilities. Plots for both functions are given in figure 3.6. Wher both functions rapidly rise to a higher loss value when the predication values get further away from the true value.



(a) MSE loss function

(b) Cross-entropy loss function

Figure 3.6: Loss plots for the MSE and cross-entropy loss functions.

The minimum value of a loss function is reached when the predicted values $\hat{y}$ are equal to the true values y. However, analytically solving equation (3.7) is computationally impossible due to the large number of parameters present in a DNN. Therefore, an algorithm called gradient descent is applied to find a good approximation to the true minimal value of the loss function, and its associated approximation of the optimal weight matrix W^*.

Various variants of the gradient descent algorithm exist, such as gradient descent with momentum or an adaptive learning rate. These variations address several problems with the base algorithm and the choice of which variant to use depends on the problem at hand. Their core mechanisms are similar to the base algorithm of stochastic gradient descent discussed in the next paragraph, and these variants are therefore not discussed further.

Stochastic gradient descent and backpropagation

Stochastic gradient descent is an optimisation algorithm designed to find the minimum of a given function. It does this by calculating the function's negative gradient in a certain point, updating the function's parameters accordingly, and evaluating the function again in the same point but with its newly set parameters. This results in a new point with a lower function value than the initial one. The loop repeats itself by the calculation of the negative gradient in the newly found point, and goes on until no or only small changes occur in the values of these newly calculated points. The gradient descent algorithm is thus slowly descending the function in small steps in order to reach the lowest value, while continuously updating the parameters of the function.

A typical NN consists of millions of weight parameters, resulting in a high dimensional space in which the loss function exists. The initial point for a certain step in the gradient descent algorithm can be defined by the weight matrix W_0, and its gradient by $\nabla\iota(W_0)$. Every entry in this gradient matrix indicates how the loss value is influenced if only that certain entry is modified, while the whole gradient

matrix describes the curvature of the loss function around the point W_0. By taking the negative gradient $-\nabla\iota(W_0)$, one goes against this curvature and descend in the high dimensional space. The gradient descent algorithm thus goes from an initial weight matrix W_0 to a point that is slightly lower by descending along its gradient, resulting in a new weight matrix W_1:

$$W_1 = W_0 - \gamma.\nabla\iota(W_0) \tag{3.9}$$

with γ the learning rate. The learning rate controls the size of the steps that the gradient descent algorithm takes while descending along the gradient. If γ is set to a large value, the algorithm will tend to overshoot the minimum value, potentially leading to an infinite loop. A small value for γ leads to in a slow convergence towards the lowest value, resulting in an algorithm that takes a long time to finish. The learning rate can be evaluated by plotting the error on the training set during training, where a good learning rate results in a steady descend towards zero loss. A learning rate that is set too high will result in a loss that stays high, while a learning rate that is set too low will result in a loss that descends very slowly towards zero.

As seen in equation(3.4), the predictions $\hat{y}$ of a NN are the product of a series of activation functions.Calculating the gradient of the loss function and updating the weights is therefore a complex operation, and both are done by the use of a technique called backpropagation. This technique is based on the chain rule, where the derivative of a composition of functions can be calculated by the product of their derivatives. As the deepest layer of the DNN is the one that depends upon all the previous ones, backpropagation goes through the network in a backwards manner and the first layer is the last one updated. Each activation function should also be differentiable in every point, as otherwise no gradient can be determined.

Vanishing and exploding gradients

Two problems that commonly occur when training DNNs are vanishing and exploding gradients. Both lead to a network that fails to learn meaningful features of the data.

The vanishing gradient problem arises when the gradient of the loss function gets close to zero. This is due to the chain rule used in the backpropagation algorithm, where the derivative of a layer is equal to the multiplication of the derivatives of all the following layers. Small derivative values that occurred in the last layers get multiplied while backpropagating through the network, leading to even smaller derivative values in the first layers. These first layers then fail to get meaningful updates to their weights and biases, resulting in a network where no learning occurs in those first layers. As these layers are essential in recognising the core features in the data, this results in a network with poor prediction abilities. Small derivative values are typically seen when activation functions are used where a large input range is mapped onto a small output range. The simplest solution is therefore choosing an activation function where the input data is not mapped onto a closed output range, but onto an unbounded one. Batch normalisation is also frequently used, where the data that a hidden layer receives is normalised and thus mapped onto

a smaller input range before its output is calculated. A more complex approach is the use of residual connections in the network. While normally each layer passes its output to the next layer, residual connections can skip one or more layers and pass their output to a layer that is more than one step away from them. This results in a smaller chain of multiplications of small derivatives, leading to an overall larger gradient value for the entire loss function.

Exploding gradients refer to gradients that get uncontrollably large, again due to the chain rule where the multiplication of large values eventually leads to even larger values in the first hidden layers. Large gradient values result in large updates to the weight parameters, and in large weights in general. These make the network unstable, such that a small variation in the input data will lead to large differences in the output. The network will be sensitive to noise in the input data, and fails to output meaningful predictions. In the worst case, the exploding gradients lead to an overflow in the loss or weight values, resulting in not-a-number (NaN) values which completely stop the learning process. Apart from changing the model's architecture, gradient clipping and weight regularisation can be applied to solve the exploding gradient problem. Gradient clipping does this by mapping the calculated gradients back to a smaller range, or cutting off gradients that are too large by setting them back to a smaller absolute value. The weights are then calculated with smaller gradients, leading to smaller weights than when calculated with the non-clipped gradients. While gradient clipping solves the problem on the gradient level, weight regularisation still allows large gradients but will punish the network for having large weights. To achieve this, a regularisation term is added to the cost function J(W), such that $J(W) = \iota(W) + \lambda.\phi(W)$, with λ the regularisation parameter that indicates the amount by which large weights are penalised, and $\phi(W)$ the regularisation function. The regularisation term outputs a higher value for larger weights, resulting in a higher cost value. This way, the network is forced to keep the weights small in order to minimise the errors. For the regularisation function, typically the L1-or L2-norm are used, or a combination of both (referred to as elastic net). The L1-norm regularisation term is calculated by taking the sum overall the absolute values of the entries in the weight matrix W. Due to its derivative, it introduces a sparse weight matrix where the majority of the weights are equal to zero. Because of this, the L1-norm is able to perform feature selection by setting the weights associated with non-useful features to zero. It is robust to outliers, but will not be able to generate complex models. The L2-norm on the other hand, takes the squared value of all the entries in the weight matrix W and sums them up, generating complex models where weights are never set to zero, but only to very small absolute values. All features are thus still taken into account and no feature selection is performed. It is not robust to outliers as the squared value of the weights will stress the outliers even more.

3.3.4 Activation Functions

Activation functions define the output of a neuron, and are typically non-linear. For easy clarification, equation(3.2) for the activated output $a_{i,k}$ for a neuron i in layer k is redefined as $a_{i,k} = \int(z_{i,k})$, with $\int$ the activation function, and $z_{i,k} = w^T_{k-1,(i,k)}a_{k-1}$ the non-activated value of neuron i in layer k. Some commonly used activation

functions and their derivatives are plotted in figure 3.7. The blue lines indicate the activation function itself, while the dotted blue lines are their derivatives. For all sub figures, the x-axis is equal to $a_{i,k}$, and the y-axis to $z_{i,k}$.

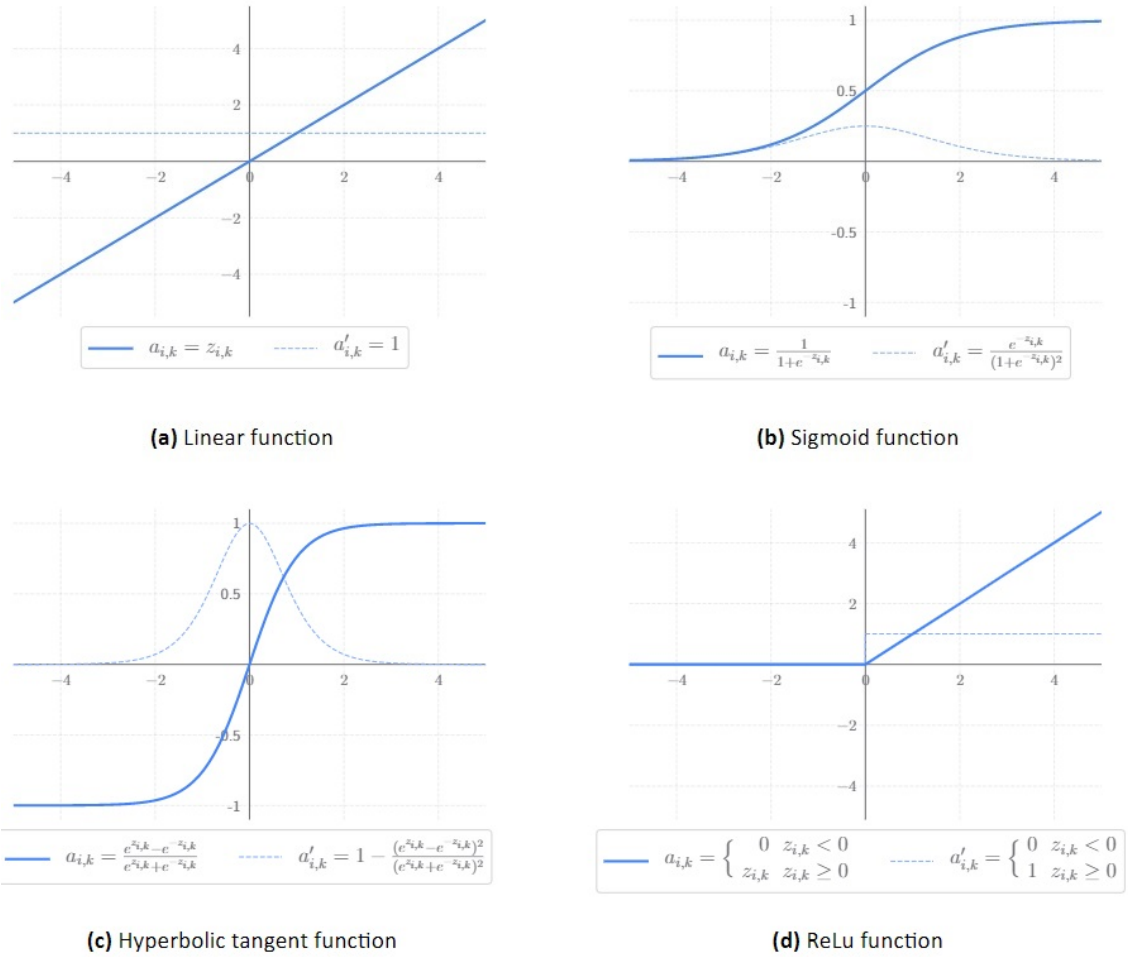
The first activation function, the linear function (figure 3.7a), is typically not used in the hidden layers of a NN. This is due to three main reasons. A derivative that is equal to a constant value will result in a backpropagation that makes no progress in updating the weights of the network. Secondly, when only linear activation are used, the final output of the network will be a linear combination of its input, reducing the NN to a simple linear regression model that lacks the power to handle complex input data. The last reason is the unconstrained nature of the output range of the linear function. It can produce large values which only get larger when propagated through the further network, eventually leading to uncontrollably large calculations. However, the linear function has its use in regression problems, where only the output layer of the network has a linear activation as here the predicted values need to be unconstrained.

The sigmoid (figure 3.7b) and hyperbolic tangent (figure 3.7c) functions solve the problems that come with the linear function. They are able to introduce non-linearity in the NN, have a non-constant function as derivative, and map large inputs back to small outputs due to their constrained nature. The sigmoid function is however prone to vanishing gradients, and is not centered around zero.This latter results in gradients that go too far in either the positive or negative direction, making optimisation harder when the sigmoid function is used. The hyperbolic tangent does not suffer from a harder optimisation as its values are centered around zero. It however does not solve the problem of vanishing gradients. The currently preferred activation function to use in hidden layers is the rectified linear unit (ReLu) function (figure 3.7d). It has a six times faster convergence than the hyperbolic tangent function due to its formula being simpler in nature, and does not suffer from vanishing gradients. However, it can introduce dead neurons, where neurons that are not activated will never be updated again during backpropagation. This can be solved by replacing the zero value for negative values by a linear function with as light slope. This solution is referred to as the leaky ReLu. Note that the derivative for the ReLu function should be undefined in $a_{i,k} = 0$, but is instead set to 1 in order to avoid problems with gradient descent.

The last important activation function is the softmax function, which is used in the output layer during classification tasks. This function will turn scalar values into probabilities for each of the n classes. Each probability lies in the [0,1] interval, and the sum overall n probabilities is equal to 1. Its formula is given by:

$$a_{i,k} = \frac{e^{z_{i,k}}}{\sum_n e^{z_{i,k}}} \tag{3.10}$$

with $z_{n,k} = w_{k-1,(n,k)}^T a_{k-1}$ the non-activated value of the n-th neuron in layer k, and $\sum_n e^{z_{n,k}}$ the sum overall the non-activated values of the n neurons in layer k.



(a) Linear function

(b) Sigmoid function

(c) Hyperbolic tangent function

(d) ReLu function

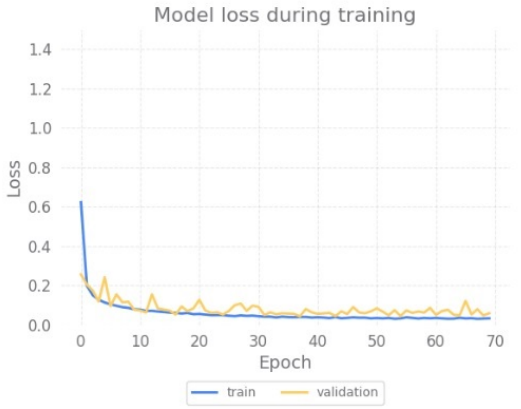
Figure 3.7: Activation functions typically used in NNs.

3.3.5 Recognising and solving over- and underfitting

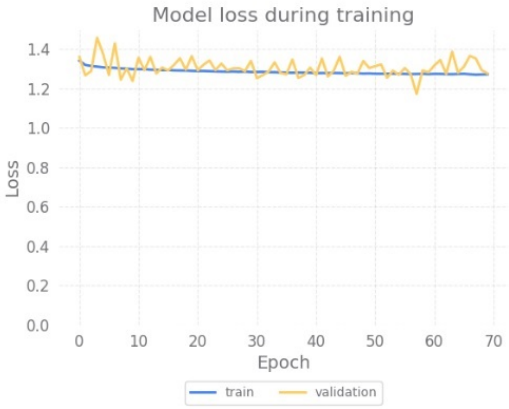
Both over-and underfitting refer to a model that is not able to generalise well on unseen data, and can be recognised by looking at the evolution of the loss value during model training. Under ideal circumstances, the loss of both the training and validation set should below (figure 3.8a). When one or both of these losses has a significantly high value, under-or overfitting occurs.

A high training loss means the model is underfitting. It is accompanied by a high validation loss, as the model fails to capture the structure in the training data and thus will not generalise well to unseen data either (figure 3.8b). Overfitting is encountered when a model has a high validation loss, but a low training loss (figure 3.8c). It happens when a model learns too much detail or random noise in its training data. These learned details and noise are however not present in unseen data, resulting in a model that captures the training data nearly perfectly yet outputs poor predictions for unseen data.

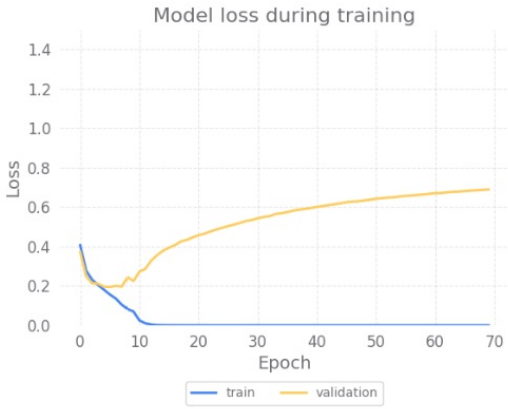
Underfitting is a problem that can easily be solved by extending the network and introducing more parameters that can capture the complexity of the input data. Overfitting on the other hand is a more complex problem that requires more thor-



(a) Ideal situation where no under- or overfitting occurs. The NN has a nearly perfect fit to the data at hand.



(b) Underfitting on the dataset. The NN fails to capture the structure in the training data, and thus cannot generalise well to the validation data either.



(c) Overfitting on the dataset. The NN learns to pick up tiny details in the training set, resulting in a poor fit on the validation set which lacks those same details.

Figure 3.8: Three different DNN architectures run on the same dataset, with the train and validation loss plotted after training for 70 epochs.

ough techniques to reduce it.

The easiest solution to reduce overfitting is simply to gather more data. This is however not an option in most of the cases and data augmentation is then a viable alternative. With data augmentation, new samples are created by slightly altering the original ones. In the case of images as input data, a variety of transformations exist such as flipping, translation, and rotation. The size of the dataset is increased by a factor equal to the number of transformations that were performed. If the dataset is small enough t of it into a computer's memory, the augmentation can be done offline by applying it before training takes place. However, if the data set is too large, real-time or online augmentation is used, where the augmentation is applied on the batches that are fed to the network during training. Another simple technique is to reduce the size of the network. A larger network equals more parameters, resulting in a network that is able to pick up more detail and noise than a smaller one. By making the network smaller, the model is forced to shift its

main focus back to patterns that actively contribute to the task at hand. A smaller network size can be achieved by removing hidden layers, or by reducing the number of neurons in the different layers.

More advanced methods to solve overfitting are early stopping and dropout. Early stopping stops the training process before overfitting can occur. This is done by monitoring a certain metric of the validation set, such as its loss. Several early stopping schemes exist, such as monitoring if the loss keeps increasing over a number of epochs or if the absolute loss increase is equal or bigger than a certain value. When the applied scheme is triggered and training stops, the model with the last most optimal loss value on the validation set is then set as the final model. When the dropout method is used, the output of randomly chosen neurons is set to zero during training. This helps with overfitting as neurons in a network will become codependent on each other during training. By dropping some of them, the others neurons are forced to learn meaningful features on their own again, resulting in a more robust network. The chance that a neuron is ignored during training is equal top p, with p a hyperparameter of the dropout layer. During testing no neurons are set to inactive, but every neuron's output is reduced by a factor pin order to account for the missing activation during the training phase.

3.3.6 Hyperparameters

Hyperparameters are parameters of the network that are chosen by the scientist, and are set before training takes place. Examples are the number of hidden layers, the total number of neurons in each layer, activation functions, and number of epochs. Hyperparameters either determine the network size and structure (model parameters) or indicate how the network is trained (optimiser parameters). The performance of a model can be optimised by tweaking the hyperparameters. This can be done manually or by an automatic search. While the former requires thorough understanding on how deep learning works and is labour-intensive, the latter comes with a high computational cost to loop through a high number of parameter combinations.

3.4 Convolutional Neural Networks

A fully connected DNN gives rise to a rapidly exploding number of parameters. This is especially trouble some when the input data has three or more dimensions, such as images (3D, 2D and three colour channels) and videos (4D, 3D and numerous frames), as one neuron in a fully connected layer would have as many connections as the element-wise multiplication of the dimensions of the input data. For example, an input image of size $100\times100\times3$ would lead to $100x100x3 = 30,000$ connections for each neuron. CNNs deal with this problem by constricting the number of connections a neuron has between two consecutive layers. This connection to only a small subset of neurons in the previous layer is called the receptive field of a neuron, and greatly limits the number of parameters present in the network. Learning takes place by looking at smaller and simpler patterns in the data, which are later assembled into bigger and more complex ones in the deeper layers. A typical CNN has an architecture similar to that of a normal DNN, but has two extra layers called

the convolutional and pooling layer stacked between the input layer and the fully connected layers. The neurons of these layers are stacked in a 3D manner, as opposed to the typical 2D arrangement seen in normal DNNs. These three dimensions are referred to as width, height and depth, and the data that a layer receives or produces are respectively called the input or output volume. An overview of an example CNN architecture is given in figure 3.9. It consists of two consecutive series of a convolution and max-pooling layer, followed by two fully-connected layers.

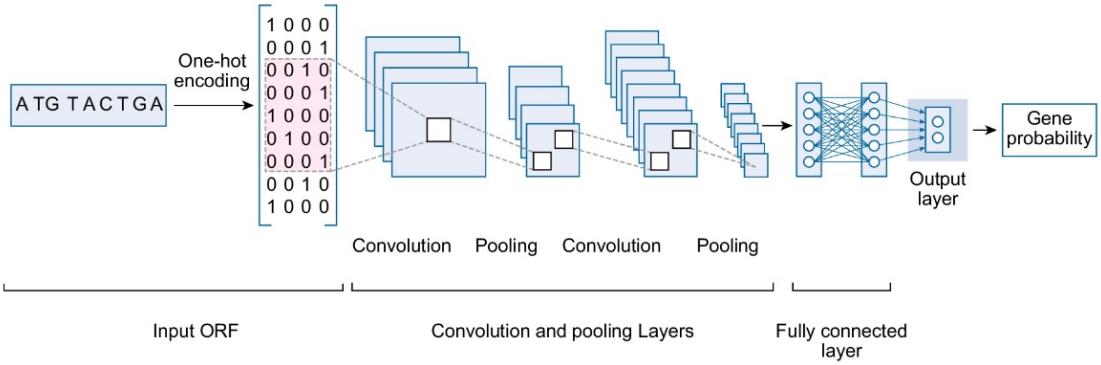


Figure 3.9: High level overview of a CNN for use with Spectrogram data type mentioned by Al-Ajlan et al. [18].

3.4.1 Convolutional layer

Most of the core computations of a CNN are all done in this layer. Here, a matrix window that is small in width and height but goes through the full depth of the input volume slides in small steps across the entire width and height of the input volume. This window is called a filter or kernel and has a size F that is seen as a hyperparameter of the convolutional layer. Everyone of its elements can be adjusted independently from the other elements in the window, and can be regarded as the weights of the layer. For every slide the window does, it computes the dot product between its own entries and the seen input, and outputs thus a single value for that exact position. As it slides over the input volume, a series of single values is outputted, resulting in a 2D feature map of the 3D input volume. Several of these filters can be applied to the same input volume which all produce a 2D feature map, and everyone of these maps recognises other patterns in the data. The feature maps are stacked along the depth dimension, creating the new 3D output volume. This output volume will thus not always have the same width and height dimensions as its associated input volume, and its depth K is equal to the total number of filters applied to the input volume. This number K is also considered a hyperparameter of the convolutional layer.

The width and height dimensions of the output volume are controlled by two other hyperparameters called the stride S and the zero-padding P. The stride S refers to the size of the steps that are taken when a filter is sliding over the input volume. When the step size is 1, then the filter moves from one entry in the input volume to the other consecutively. When the stride is set to a larger number, the filter will skip some entries, resulting in a smaller output volume. The zero-padding

hyperparameter P indicates if an extra border of zeros is added around the input volume and how wide that border is. By adding padding, a filter can also be applied at the edges and corners of an input volume. If no padding is added, the seen tries cannot be used as they lack certain neighbouring values needed to compute the dot product. This way, the original dimensions of the input volume can be preserved or even expanded. The stride S and zero-padding P hyperparameters can be used together with the size F of the filter and the width and height dimensions of the input volume V to calculate the width and height dimensions of the output volume W:

$$W = \frac{V - F + 2P}{S} + 1 \tag{3.11}$$

Every entry in the output volume can be regarded as the output of a single neuron. This neuron only has connections with the neurons in its immediate vicinity, namely the neurons whose output values were used in the calculation of the dot product with the kernel. This reduced number of connections along the first two dimensions is called the receptive field of the neuron and is equal to the size F of the kernel. While this kernel sees only a small part of the input volume along these dimensions, it goes through the full depth of the input volume. This means a neuron has as many connections along the depth axis as the depth of the original input volume. Its total number of connections along all dimensions is then equal to the element-wise multiplication of the width and height size F of the kernel and the depth of its received input. Also note that each neuron has as many weights as it has connections plus 1, as a bias still has to be added. If one now looks back at the example of an input image with size 100×100×3, the convolutional layer that directly follows the input layer will receive an input volume with the exact same dimensions as the original data. When a filter with size 2×2 is applied, a single neuron in that layer will then have only $2x2x3 = 12$ connections, instead of the 30,000 in a fully connected NN. Another intervention is needed however to reduce the number of parameters in a convolutional layer. To illustrate this, we will calculate the size of the output volume of the convolutional layer in the above example with astride S = 1, a padding of P = 0, and K=128 applied filters. Using equation (3.12), the output volume W is equal to:

$$\begin{aligned} W &= \frac{100 - 2 + 2 + 0}{1} + 1 \\ &= 99 \end{aligned} \tag{3.12}$$

The output volume thus has a dimension of 99×99×128. As each output is associated with a neuron, the number of neurons in this convolution layer is equal to $99x99x128 = 1,254,528$. As previously calculated, each neuron has 12 connections with 13 accompanying weights. This finally results in a total parameter number of $1,254,528x13 = 16,308,864$ for just this one single convolutional layer, which would quickly lead to overfitting. To solve this problem, parameter sharing is applied. The idea behind this is that if a feature is useful to calculate at one position of the input volume, then it will also be useful to calculate that exact same feature at another position of the input volume. Every neuron that is part of the same feature map can thus share the same parameters, resulting in K unique sets of weights and

biases. In the given example, this means that there would only be 128 different sets of weights and biases, where each set consists of 13 parameters, resulting in a total number of parameters of $128x13 = 1664$. This parameter sharing scheme can also be relaxed if the network has to learn different features on each side of its input.

After the output volume of a convolutional layer is calculated, it is typically activated by the ReLu function before it gets passed to the following layer. To illustrate the inner workings of a convolutional layer, a visual example is given in figure 3.10. Where blue indicates the 3D input volume, where the third dimension (depth=3) is illustrated as a stack of 2D inputs. Red indicates the filters (size F=3x3), and green the 3D output volume (depth = 2; equal to the number of filters used). The filter W0 is applied on the full depth of the input volume (high lighted in blue). The values are multiplicated element wise, summed up, and offset with a bias b0. This results in the highlighted green output, which is found in the first slice of the depth stack. When filter W1 is used, its result will be found within the second slice of the output volume depths tack.

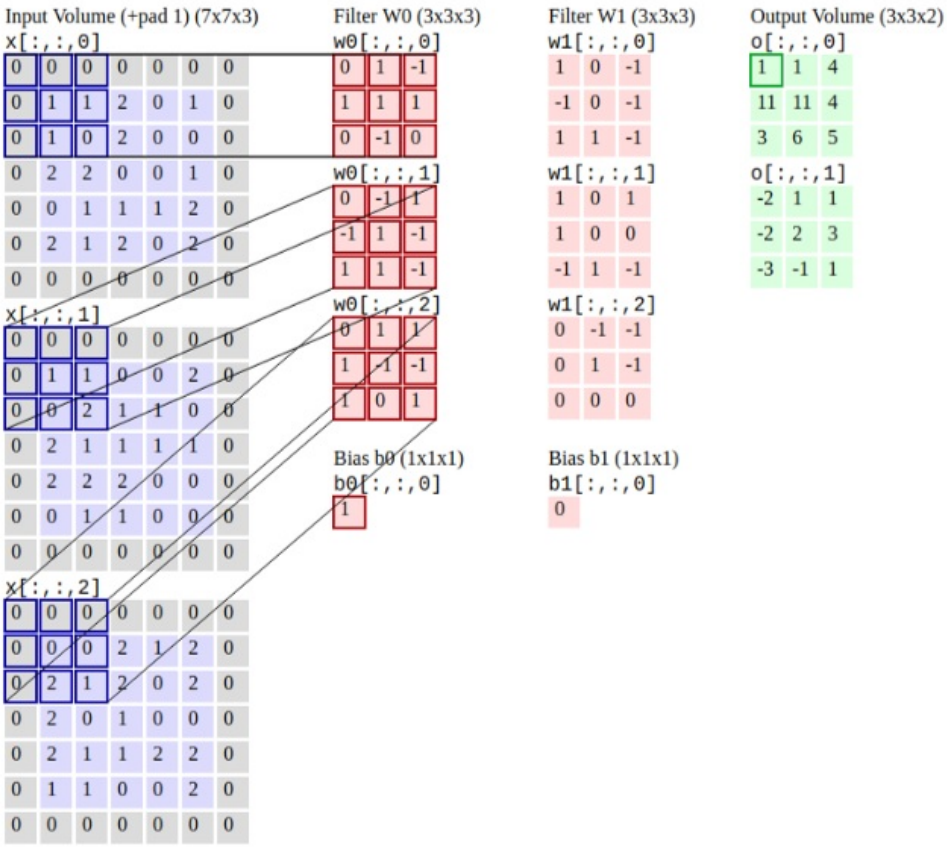


Figure 3.10: Inner workings of a convolution layer by Karpathyetal [66].

3.4.2 Pooling Layers

After one or more consecutive convolution layers, a pooling layer is added to reduce the output volume along its width and height dimension. This is done to reduce the number of parameters in the network which consequently also combats overfitting.

In this layer, a kernel slides over every feature map and applies a function to its input. This function can be the average or L2-norm but most commonly, the max function is used where only the maximum value overall its seen inputs is retained. Note that the pooling kernel does not go through the full depth of its input volume, and is instead applied on every feature map separately. The depth dimension of the input volume is therefore not changed.
A pooling layer has no parameters associated with it as it only applies a fixed function. It however consists of two hyperparameters, namely the size F of the kernel and its stride S. Using these hyperparameters together with the input volume V, as light variation of equation (3.12) is used to calculate the size of the output volume W:

$$W = \frac{V - F}{S} + 1 \tag{3.13}$$

A visualisation of how a max-pooling layer works can be seen in figure 3.11. A single depth slice extracted from the input volume with a height and width equal to 4x4 is illustrated on the left. The max-pool kernel of size 2x2 and with stride 2 is applied onto the depth slice. Each colour block indicates an application of the max-pool kernel. This results in the output volume on the right, where the result of each applied kernel operation is visualised by its accompanying colour.

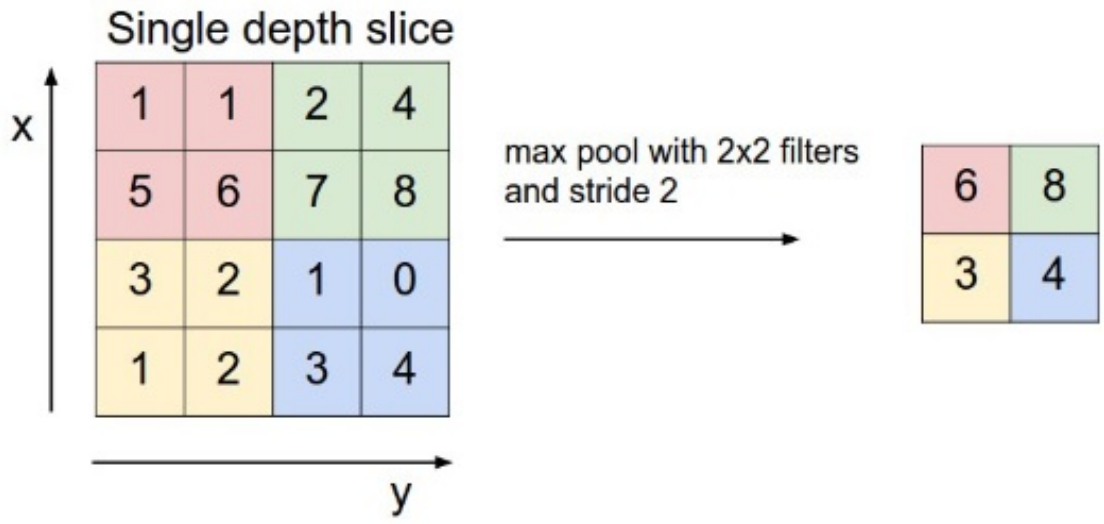


Figure 3.11: Visualisation of application of a max pooling layer onto a single depth splice as mentioned by Karpathyetal [66].

3.4.3 Following Layers

A CNN always ends with one or more of the classical fully connected hidden layers. The neurons in these fully connected layers are arranged in a 1D manner, as opposed to the 3D arrangement in the pooling or convolutional layers. To ensure the neuron connections between the last convolutional layer and the first fully connected layer, a flatten layer is added. This layer takes the 3D output from the last convolutional layer and reads it one feature map at a time. While reading a feature map, all values are concatenated, resulting in one big vector. The other feature maps are

added to the same vector after the values of the previous maps. This eventually results in a vector with a length equal to the element-wise multiplication of the three dimensions of the output volume of the convolutional layer. The following fully connected layer takes in the vector-output of the flatten layer, and passes it to the next fully connected layers. When the last fully connected layer is reached, a prediction is made by the network and outputted.

Chapter 4

Related Work

At the time of this research working we have reviewed more than 50 research papers associated with Bangla speech. We have tried our best to describe a few of the latest working conducted with bangla speech through Natural Language Processing (NLP) techniques, classification of Regional Bangla Language, artificial Bangla Speech Classification, classification of speakers identity through Bangla Speech and lastly the type and context of Bangla speech dataset available for researches currently.

4.1 Existing Work done with Bangla Language

The authors in the paper [36] used neural network to detect hate speech in social media and received an accuracy score of 77%. While Rahut, Riffat and Ridma [33] reported an accuracy score of 98.16% in classification of abusive words in Bangla speech by using VGG-16. Sharmin et al. in [24] built a Deep CNN model to classify Bangla spoken digits classification and achieved an accuracy of 98%. Alam et al. [35] uses YouTube Comment, Bangla news portal and Bangla e-library datasets to develop a fine-tune multilingual transformer models for Bangla text classification tasks; sentimental analysis, emotion detection, news categorization and authorship attribution. The authors received a score higher than previous accuracy by 5-29%.

4.2 Existing Work Related to Classification of Regional Bangla Language

Recently a detailed work was published on the challenges and opportunities for Bangla language speech recognition [25] stated that, for any system to recognize the features in Bangla speech have to first understand the construction of Bangla language structure grammatically and phonetically to built a flawless ASR and TTS system. The authors also defines the language dependent and independent challenges faced by previous researchers for Bangla speech recognition. Mridha et al. underlines the importance of a large grammatical and phonetic database for creating a flawless ASR systems that produces clean acoustic artificial Bangla speeches.

As a typical Bangla sentence is constructed in the pattern; subject followed by object and then verb. Additionally, auxiliary verbs are not used in a Bangla language sentence and the preposition are placed at the front of a noun or else the use of noun-equivalent words have to be used during the construction of regional Bangla language sentence. To build a flawless automatic speech recognition (ASR) [32][33][36] and text-to-speech (TTS) [6][7][8][13] systems for Bangla language one has to use a database with an extensive vocabulary and phoneme patterns of Bangla language. The extensive jargon is observed to be missing from the public or private databases available for Bangla speech. Hence, one of the many reason researches carried out in the past decade on recognition features in Bangla speech have failed to investigate regional language during Bangla speech feature classification is due to the limitation in the database.

A comprehensive work can be found on Bangla speech recognition; [33] classification of Bengali accent using different deep learning (DL) techniques. The authors did not build or use the broad corpus dataset required when distinguishing regional dialect. They retrieved characteristics such as Chroma Features, RollOff, MFCCs, ZCR, RMSE, Spectral Centroid, and Spectral Bandwidth from data gathered from nine areas. They used a Random Forest-based model with an accuracy score of 86% to feed the data into.

In their paper "Accent identification of Telugu speech using prosodic and formant characteristics" [50], K. Mannepalli and V. Rajesh employed predetermined features such as pitch, energy, power spectral density, short-time energy, and intensity extracted using COLEA and PRAAT to input into a Nearest Neighbor Classifier (NNC), reaching 72 % accuracy in categorizing Telegu regional language in three separate accents spoken in Southern India. Rather of utilizing NNC to classify, the authors of [51] presented a method that uses the Gaussian Mixture Model (GMM) and the Support Vector Machine (SVM). By mapping an utterance to a high-dimensional vector, they produced a GMM supervector. Many studies employed SVM, which is frequently used for categorizing data corresponding to a high-dimensional vector space. The works of [52]–[55] are likewise similar to these two aforementioned techniques.

In their paper "Deep Learning-based Mandarin Accent Identification for Accent Robust ASR," F. Weninger and Yang Sun propose a somewhat different Deep Learning-based method. In paper [56] They were able to successfully categorize 15 distinct geographical locations in China based on accents, despite the fact that some of them were not even mutually comprehensible. They proposed employing the bLSTM (bidirectional Long Short-Term Memory) accent classifier to swiftly transition between two alternative ASR models, standard and accented, depending on the current circumstance. They had collected 135k utterances from 466 speakers (84.6 hours). The goal of employing bLSTM was to capture the longer-term acoustic background in each syllable, purportedly increasing accent recognition.

The paper "Accent Detection and Speech Recognition for Shanghai-Accented Mandarin" takes a much more probabilistic technique. In paper [55] Accentedness (the degree of variation from the conventional accent) was divided into three categories

using MFCC and GMM. They also distinguished between two types of speakers: normal and accented. Finally, in order to choose the best model for a specific speaker, they calculated the MAP (Maximum a posterior) of several models. In their trial, using MAP with traditional techniques resulted in a 1 to 1.4% absolute reduction in character mistake rate (CER).

One of the key reasons for the ASR system's greater mistake rate when dealing with accented speech is that the speaker may be slightly mispronouncing the provided word. Some Microsoft researchers devised a far more sophisticated approach in their paper "Accent Issues in Large Vocabulary Continuous Speech Recognition." In paper [54] They created a novel adaptation approach called Pronunciation Dictionary Adaption, which is essentially a dictionary that captures the pronunciation changes caused by a speaker's mispronunciation (mispronunciation) for an accent by giving the system a little quantity of adaptation data. Given that the system had 3 to 5 utterances accessible for each unique speaker, the character error rate (CER) of the system was 13.2% - 13.6%.

4.3 Existing Works related to Artificial Bangla Speech Classification

Autoencoder neural networks for unsupervised learning of sparse and temporal hierarchical features to audio signals have long been used [10][13][18]. The authors of [10] used a convolutional autoencoder with Mel Frequency Energy Coefficients (MFEC) data to detect anomalies in synthesized speech. Their method yielded results better than the baseline. Yuxuan et al. [4] proposed a deep learning based approach to develop a Bangla speech synthesizer without the any frontend preprocessor and Grapheme-to-Phoneme (G2P) converter. It was able to synthesize Bangla speech at 3.79 Mean Opinion Score (MOS) on a5.0 scale as subjective evaluation and 0.77 Preceptual Evaluation of Speech Quality (PESQ) score.

Modern TTS systems, like any new technology, may be utilized for nefarious purposes. To construct a speech model for a target person, DNN-based TTS systems might be used. The malicious actor might use this model to carry out a variety of spoofing attacks, including impersonation and/or circumventing automatic speaker verification systems. Researchers have been researching ways to recognize synthetic speech in an attempt to reduce the risk of such assaults. With growing worry about the harmful use of such technology, researchers from around the world organized the ASVSpoof2 challenge, in which they released a dataset of actual and spoofed voices in the hopes that the community would be able to figure out how to distinguish between the two. Several studies have been published that provide strategies for detecting faked speech. The majority of the proposed solutions are based on extraction of frequency features utilizing HMM and GMM models.

The dataset does not include the most up-to-date state-of-the-art TTS technology, despite the fact that this challenge represents a watershed point in the synthetic speech recognition area. In addition, we identified a need for a new dataset that includes the most recent TTS solutions to reflect our current speech synthesis envi-

ronment, as there are existing ways in the literature that achieve excellent accuracy on the ASVSpoof dataset. As synthetic speech generating systems get more complicated, it will be necessary to investigate increasingly complex solutions (such as Deep Neural Networks) for synthetic speech detection.

4.4 Existing Works related to Classification of Age and Gender from Audio Speech

Previous researchers did extensive work on developing various methods for recognizing gender and age of speakers in Bangla Speech. Researchers focused on two important parts; identifying the optimal features and building a model that is able to recognize those features over various types of speech language. They achieved an accuracy of 90% for gender and age classification. Anvarjon et al. [8] build an end-to-end CNN with a multi-attention module, to extract the spatial and temporal salient features, to recognize age and gender from speech signals. Authors of [9] used multi-layers perceptron (MLP) to learn the gender and speaker identity features and it has outperformed the CNNs model [8] during the experimental process, accuracy score of 92%, due to the use of MFECs input data.

On the other hand, Gomez et al. [41] uses Saarbrucken database to build a age-dependent pathology detector by employing the sustained vowels from the database. The study uses two controlled group in the experiments: elderly and adults. The study also uses Mel frequency cepstal coefficients for characterization and Gaussian Mixture for classification. The paper contributes to the area of effectively recognizing the age from normal and pathological voices.

Orken et al [7] uses two neural architecture for speaker identification with Mel-frequency Cepstral Coefficients (MFCC) data type. Through experimentation Multi-layers preceptron (MLP) outperforms CNNs, while using z-score and Gramian matrix transformation and max-min normalization of MFCC.

4.5 Existing dataset for Bangla Speech

Rezaul et al. [42] has built the largest Bangla language word embedding models to this date, teh dataset is called "BengFastText". They have used 250,000,000 articles during the creation of this dataset.The dataset consist of 5 parts; expressing hate, commonly used topics, opinions for hate speech detection, document classification, and sentiment analysis. They have also build Multi-channel Convolutional-LSTM (MConv-LSTM) network to predict these classes. They were able to achieve 82.25%, 90.45% and 92.30% F1-scores for sentiment analysis, hate speech detection, and document classification via 5-fold cross-validation tests. S. Mavaddati [43], developed a generative incoherent models that learns using sparse non-negative matrix factorization and the atom correction step as a post-processing method to classify the gender and age of the speaker through audio signal. He has used MFCC as input data format. The models performs exceptional in the presence of background noise

compared to earlier methods.

Gutkin eet al. [27] have developed an TTS system to address the issue of limited-resources for Bangla language speech datasets faced by researchers. They have Long Short-Term Memory Recurrent Neural Network (LSTM-RNN) and Hidden Markon Model (HMM) approaches as statistical technique to construct multi-speaker acoustic models. Over the data collected through crowdsourcing from multiple speaker and applying text normalization system of the closest relate able language (Hindi) for linguistic front-end for Bangla speech.

As a result authors of [25] stated the importance of having large grammatical database with regional phonetic variation. While constructing a flawless ASR or TTS system in Bangla language and the findings of the previous work [33] influenced our proposed model for Bangla audio classification regional language in Bangla speech depending on the acoustic features and detecting artificial Bangla speech from audio signal in this paper. The suggested method uses stacked of fully connected convolutional autoencoder with a MLELMs to selectively focus on important information from the MFECs and efficiently recognize the features; synthesized/original audio, dialect, age, and gender of the speaker. The model also outperforms accuracy scores for both dialect, age and gender achieved by previous researchers.

Chapter 5

Dataset Collection

To design a system capable of classifying dialect of the speaker in Bangla speech and distinguishing synthesized voice from original voice, a dataset is first built with recorded Bangla speech with seven regional language and synthesized Bangla speech audio file created with the help of Text-to-Speech (TTS) system [13]. However, according to recent researches, at least 20 hours of speech data is required for constructing a robust TTS system.

Hence, within Brac University, 30 hours of Bangla speech dataset is made with seven different regional languages used in Bangladesh in this paper. Detail description of the dataset can be found in the following sections. For the testing and training purposes, the authors of this papers combined the datasets of 13 hours of Bangla voice data previously published by Brac University [7], 3 hours of Bangla speech released by Google with the created dataset. Later with the help of help of TTS, synthesized regional Bangla speech is generated. As the dataset contains large speech data with related transcription, TTS is effectively able to generate the synthesized audio Bangla regional speech.

For English speech VoxCeleb [3] dataset is used to generate synthesized English speech for the proposed model. The VoxCeleb dataset has roughly 2000 hours of 100,000 phrases taken from YouTube videos by 1,251 celebrities of American, European and Asian dialects and from various age groups.

5.1 Text Composition

The dataset is build with a phonetically balanced text corpus as suggested by the authors [3]. Text data gathered from variety of source. While ensuring the corpus contained every conceivable Bangla punctuation [13] for each of the seven regional language in Bangladesh; Khulna, Bogra, Rangpur, Sylhet, Chittagong, Noakhali, and Mymensingh. Finally, the dataset for Bangla speech includes almost 75,000 utterances. Table 5.1 shows a sample of the Bangla speech data. According to [13], nonstandard terms needs to be translated into their standard pronunciation utilizing text normalization procedure to reduce ambiguities in Bangla synthetic speech. The process of transforming an unpronounceable text into a pronounceable form is also taken into consideration when creating the dataset. The process of compositing the regional language is discussed in the following section.

Table 5.1: Summary of Bangla speech data

Total duration of speech (hours)	21:16:19
Total number of sentences	100,057
Average duration of words each sentence (seconds)	7.81
Total number of words	2,759,421
Number of words of Khulna region	385,714
Number of words of Bogra region	265,482
Number of words of Rangpur region	276,348
Number of words of Sylhet region	348,788
Number of words of Chittagong region	475,428
Number of words of Noakhali region	425,482
Number of words of Mymensingh region	582,179
Total unique words	85,500
Maximum words in a sentence	10
Minimum words in a sentence	5
Average words in a sentence	5.45

Table 5.2: Sample Words as per Sylhet regional language used for text recognizer. In total 85,500 words used for the the seven regional area.

ID	Bangla	English	English/Bangla Abbreviated
01	পুয়	Son	powa
02	পোলা	Boy	pola
03	মানুশর	Man	manusher

We have used text recognizers created by [69] to understand Bangla language. As no current text recognizer understand Bangla literature, we had to adopted the Bangla/English abbreviation words, commonly know as the Banglish text used in by people on the daily bases, to train the model. Bangla Speech data was created on 50 unique sentence. Each participants recorded 10 sentences from the list. Each bangla word were given an ID number , as shown in Table 5.2.

Later these words were used to create sentences to further train recognizer. Each sentence had 5 to 10 words. Sample is shown in Table 5.3. As the common words or stop words like ”the, an, a, and, to, etc” used in English literature are not used in Bangla literature as a separate term but combined to the object or the verb in the sentence.

We have also recorded a few details of the speaker and recording to create the label that would be identify each audio signals. For example;

Table 5.3: Sample Words as per Sylhet regional language used for text recognizer.

ID	Bangla	English	English/Bangla Abbreviated
01	আমার বয়স পঁচিশ	I am **25** years old	amar byosh pochis
02	পোলাটা স্কুলে খুশী	**The boy** is **happy** in **school**	Polata eskule khusi

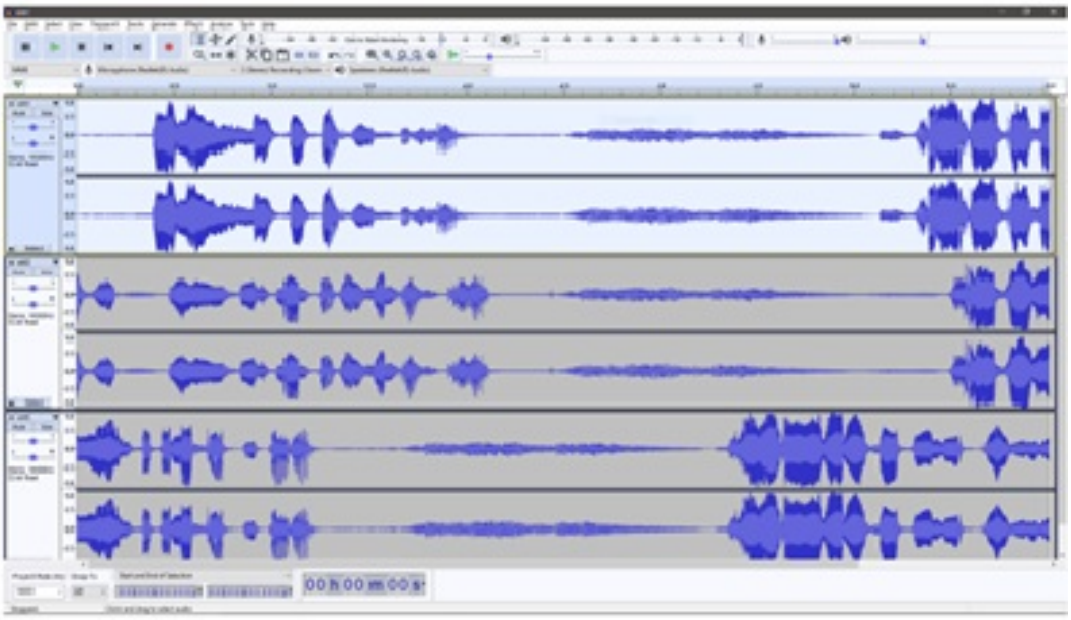


Figure 5.1: Before cleaning the audio file

- Age
- Gender
- Language dialect
- Condition of the recorded environment; for instance the source of the noise if any in the room.
- Recorded device specifications: mobile, microphone, etc.
- Time and date of the recorded session.
- ID generated in 5 characters. The first character is M or F that would stand for Male or Female. Second character 0 for Bangla speech or 1 for English speech. Third character participant number that starts from 1 to 50. Lastly, fourth and fifth character is the sentence ID.

The following wave file settings were kept throughout the recording session:

- Audio sample rate : 22 kHz
- Bit rate : 16
- Single channel mono

The 22 kHz sample rate was chosen for this project because it delivers more accurate high frequency information and separates the element location into 78245 potential values.

The audio files were than striped in to separate one sentence one audio file manually by using Audacity software. the files were saved in .wav format. The files were

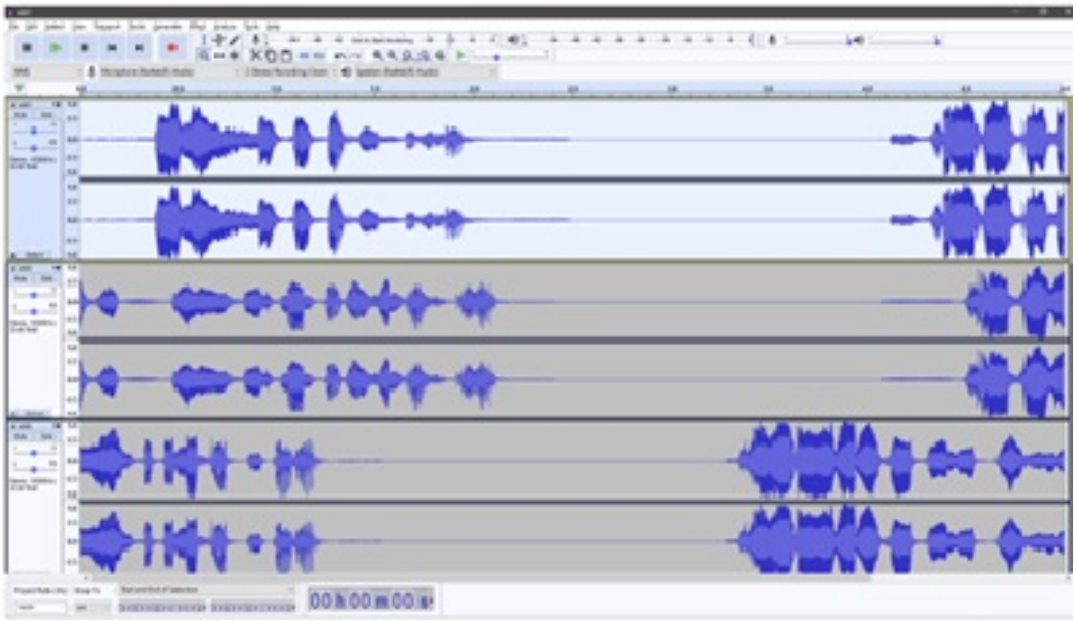


Figure 5.2: After cleaning the audio file

named as speech, sentence, word and speaker id.

For instance: 01010103.wav means;

- Speaker Id: 01
- Sentence Id: 01
- Word Id/s: 01
- Speech Id: 03

Problems faced after gathering the data. The audio samples in the formal accent were in various formats. None of them were in '.wav' format. Used Librosa to convert all the file to .wav format. Few of the recording had significant amount of background noise and extra music which were not part of the speech. Used Audacity software to clean the noises. As shown in figure 5.1 and figure 5.2. However, this had caused few samples to be discarded, because it was not possible to denoise the audio sample without causing significant loss in the speech.
For proper MFEC 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 MFEC features. Used pydub module to check the length.
Final dataset, figure 5.3, after removing noise and speech data that could not be cleaned without losing speech. We had a total of 10,000 audio files. 7,334 original speech and 2,666 synthesized. Figure below shows the visual for the distribution of the sample.

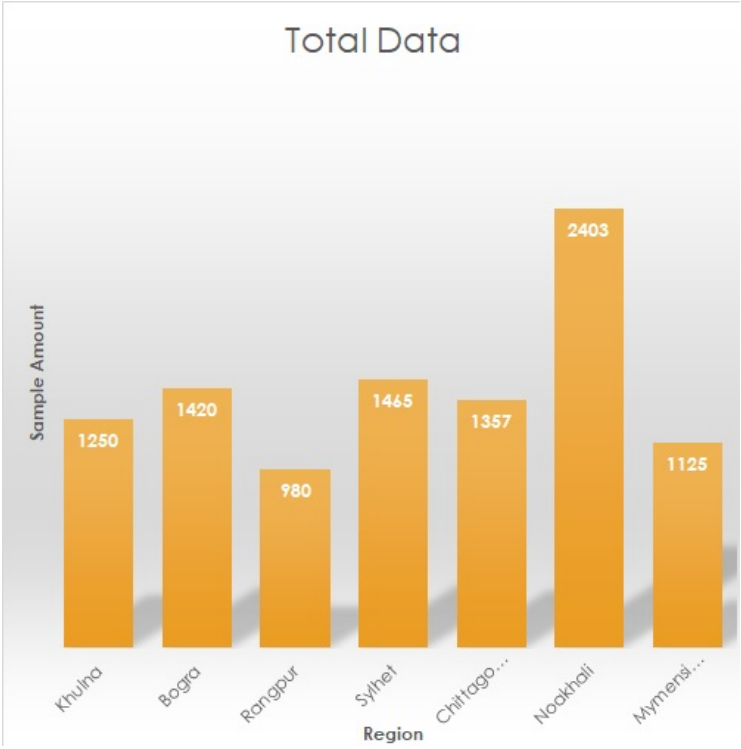


Figure 5.3: Total distribution of the samples region-wise.

5.2 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:

$$AE_t = \max_{k=t.K}^{(t+1).K-1} s(k) \tag{5.1}$$

Here, s(k) refers to the amplitude calculated at the k_{th} 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. Figure 5.4 shows the sample amplitude envelope for Bogra region speech.

5.3 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:

$$ZCR_t = \frac{1}{2} \sum_{k=t.K}^{(t+1).K-1} |(s(k)) - (s(k+1))| \tag{5.2}$$

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()

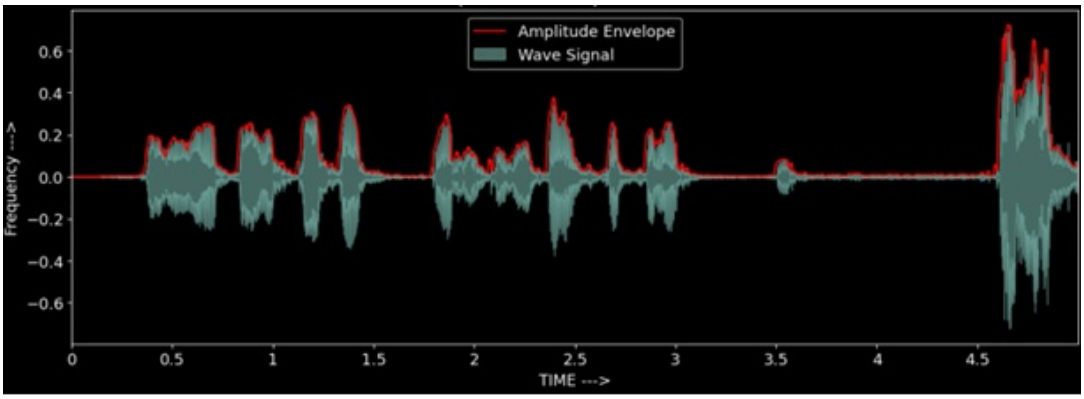


Figure 5.4: Amplitude envelope sample for Bogra region

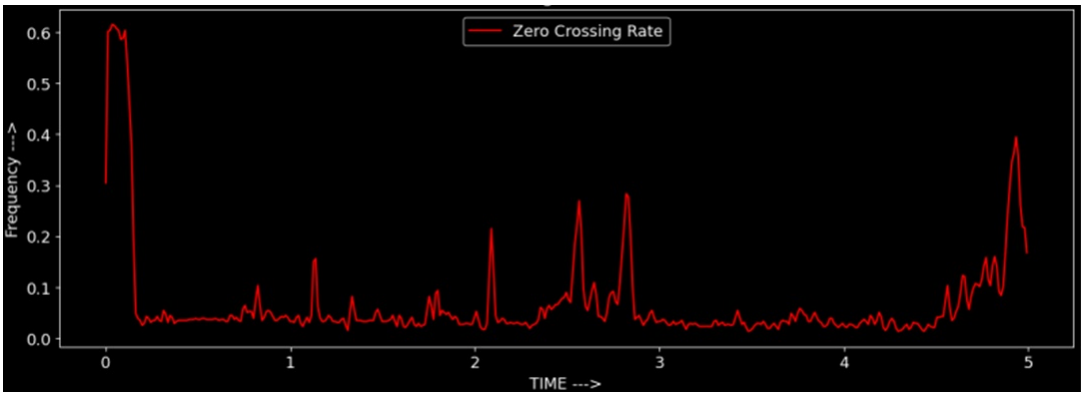


Figure 5.5: Zero crossing rate sample for Bogra region

function as such: $s(k) > 0 \rightarrow +1, s(k) < 0 \rightarrow -1 and s(k) = 0 \rightarrow 0$
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. Figure 5.5 shows the sample Zero crossing rate for Bogra region speech.

5.4 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:

$$RMS_t = \sqrt{\frac{1}{K} \sum_{k=t.K}^{(t+1).K-1} s(k)^2} \tag{5.3}$$

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. Figure 5.6 shows the sample rate mean square error for Bogra region speech.

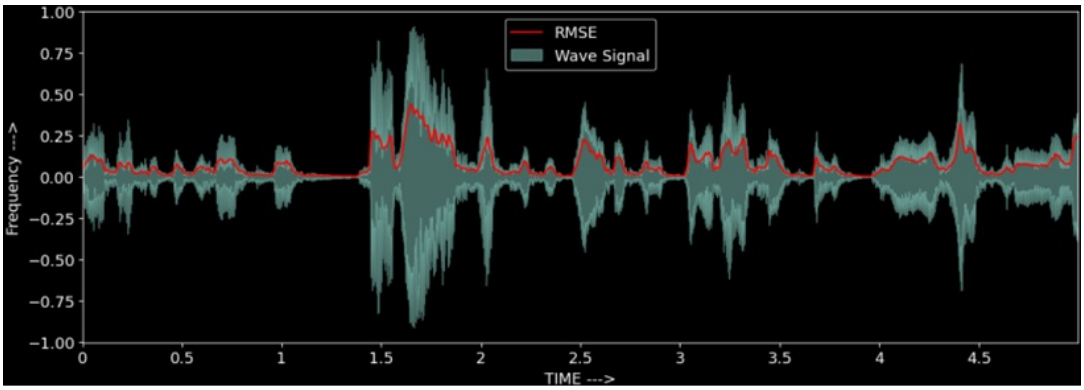


Figure 5.6: Rate mean square error sample for Bogra region

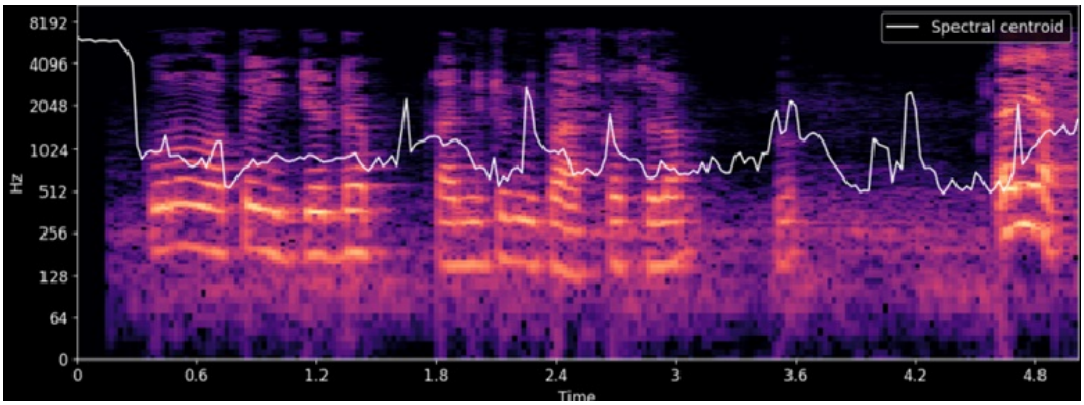


Figure 5.7: Spectral Centroid sample for Bogra region

5.5 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. Figure 5.7 shows the sample spectral centroid for Bogra region speech. Where the white line represents the mean frequency of the particular frame.

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:

$$SC_t = \frac{\sum_{n=1}^{N} m_t(n).n}{\sum_{n=1}^{N} m_t(n)} \tag{5.4}$$

This concept is similar to RMSE whereas during RMSE the calculated mean is Amplitude and in this case the mean is Frequency. This feature can help us determine the difference between the accents using the variety of frequency bins that can found in each regional accent.

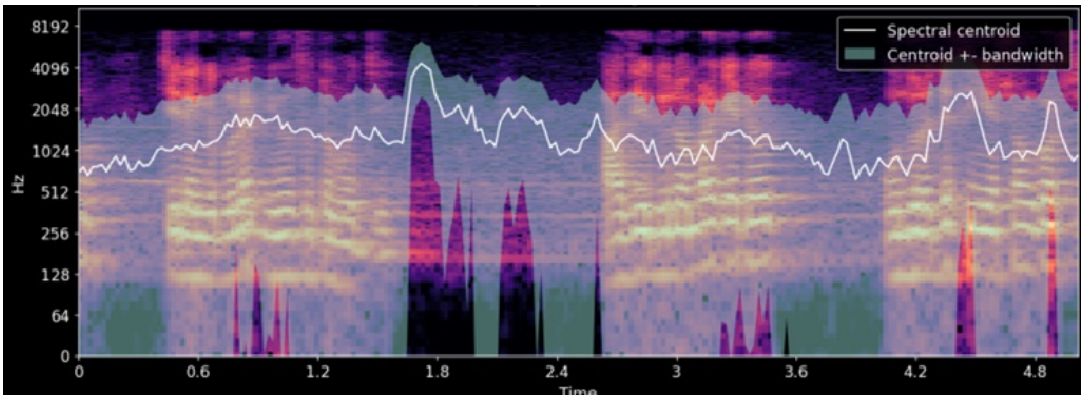


Figure 5.8: Spectral Bandwidth sample for Bogra region

5.6 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 bethought 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. Figure 5.8 shows the sample spectral bandwidth for Bogra region speech.

$$BW_t = \frac{\sum_{n=1}^{N} |n - SC_t|.m_t(n)}{\sum_{n=1}^{N} m_t(n)} \tag{5.5}$$

From the formula we can clearly see the similarities with the Variance formula. Here, $mt(n))$ 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.
We can see the distribution of the above mentioned features across the seven region through the below graphs. Khulna region have the highest amplitude envelope, meaning the loudness of the speech is the highest compared to the rest. While Rangpur as the highest energy distribution across all sample. On the other hand we can see that no region have similar frequency bins, spectral centriod, which is helpful to distinguish the regional language.

5.7 MFEC

Mel-Frequency Energy Coefficients is connected to the concept of mel-spectrogram. We're basically employing a mel scale, which is a perceptually meaningful pitch scale. It may provide up to 39 features depending on how it is implemented; we use 13 features for our job.

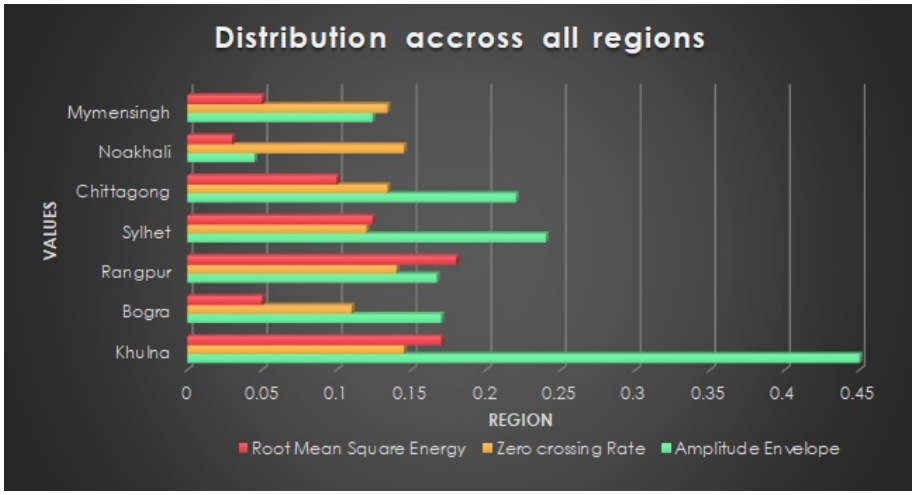


Figure 5.9: Distribution of amplitude envelope, zero crossing rate and root mean square error feature across the seven regions.

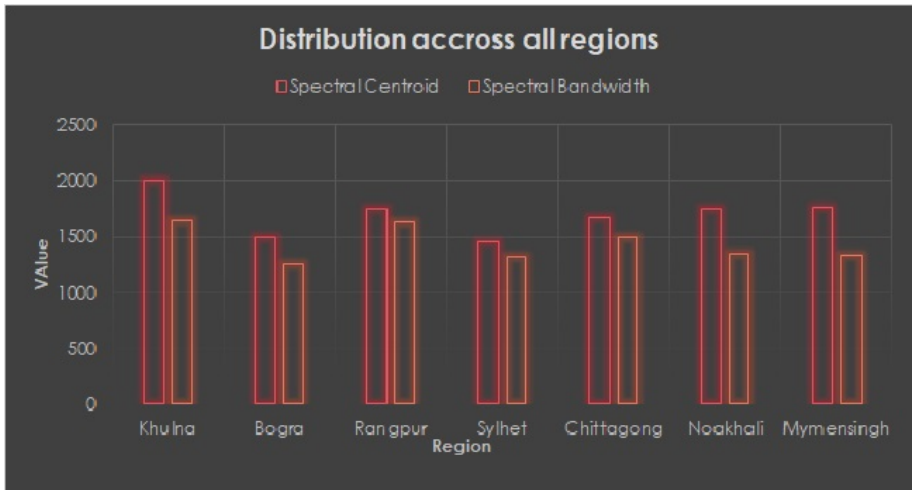


Figure 5.10: Distribution of spectral centroid and spectral bandwidth feature across the seven regions.

To understand the MFECs we would first need to understand the **Cepstrum** mathematically equation;

$$C(x(t)) = F^{-1}[\log(F(x(t)))] \tag{5.6}$$

The discrete Fourier transform logic is then employed. *x(t)* is the time domain signal. This will give us the frequency domain spectrum of the signal. We acquire Log Amplitude Spectrum by taking the log of the spectrum. Finally, we compute Cepstrum using the inverse Fourier transform.

It is a mapping between the frequency or pitch of a pure tone that we hear to the frequency or pitch that it actually has. We utilize Mel-Scale because we humans are considerably better at detecting slight changes in pit chat low frequencies than high frequencies. We can better match our features to what humans hear by using this scale. We compute the mel-frequency of a frequency using the following formula:

$$M(f) = 1125ln(1 + \frac{f}{100}) \tag{5.7}$$

later to convert to frequency from mel-scale we use;

$$M^{-1}(m) = 700(\exp(\frac{m}{1125}) - 1) \tag{5.8}$$

Discrete Fourier Transform to each frame is applied through the below equation. h(n) denotes the hamming window.:

$$S_i(k) = \sum_{n=1}^{N} s_i(n)h(n)\exp(\frac{-j2\pi kn}{N}) \tag{5.9}$$

Therefore, to sum it up, we started with 16KHz sampled audio signal, Then applied MFEC steps; Waveform, Discrete Fourier Transform (DFT), Log-Amplitude Spectrum, and lastly Mel-Scaling. The fame size used is 2000 samples per frame, and the hop length was decided by Librosa to generate 15 features. Let t(n) be the time domain signal, After the framing the whole signal we get $t_i(k)$ where i range is from 1 to 300. k denotes the frame number. $P_i(k)$ denotes the power spectrum of frame i. Power-spectrum is denoted by the the Periodogram estimate:

$$P_i(k) = \frac{1}{N}|S_i(k)|^2 \tag{5.10}$$

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 50000. Fast Fourier Transform were applied 2048 times. Hop length were 512 samples that was generated from the 22050 samples of each audio segment of 1 second. Ceiling value was 22050/512 which is 44 segments. Hence a total of $44x13 = 572$ MFEC features extracted from 1 second audio sample. Following figure shows the detailed work of extracting MFEC feature. Various audio features were extracted from the samples and using "librosa" and using the python module "matplotlib" libaray the features were visualized. An initial CSV file was created to store the mean values of 18 extracted features. Sample CSV file ca be seen in figure 5.11

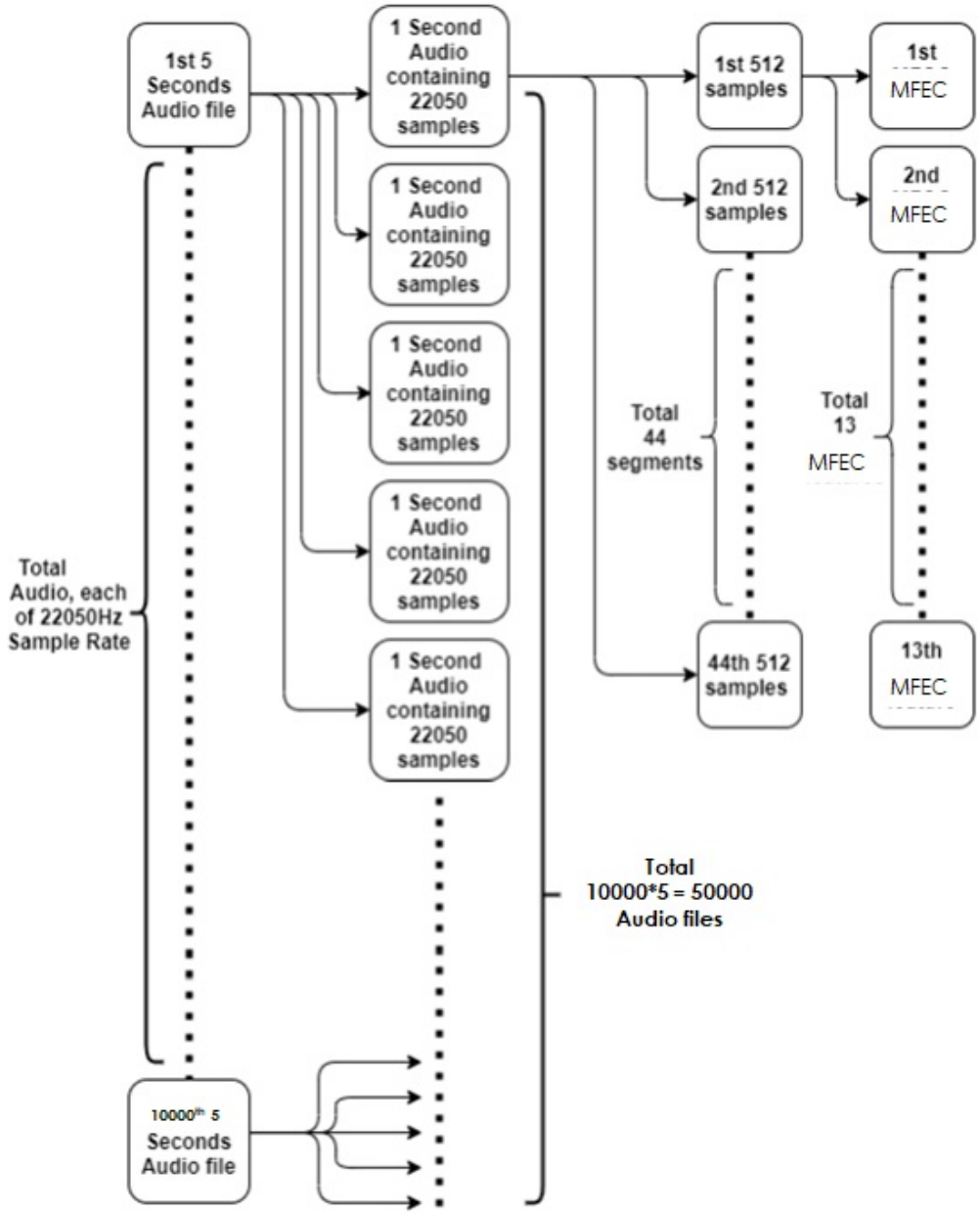
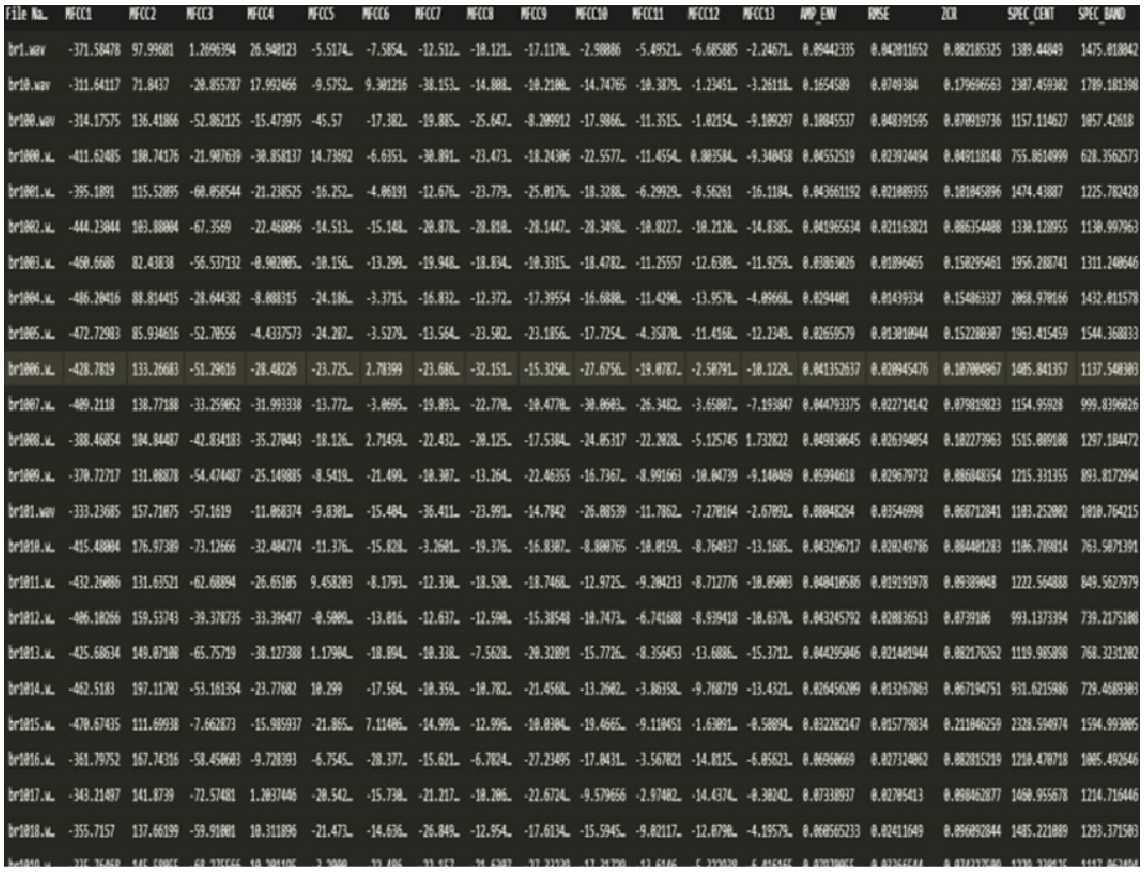


Figure 5.11: Audio segmentation and MFEC feature extraction process.



File Na..	MFCC1	MFCC2	MFCC3	MFCC4	MFCC5	MFCC6	MFCC7	MFCC8	MFCC9	MFCC10	MFCC11	MFCC12	MFCC13	AMP_ENV	RMSE	ZCR	SPEC_CENT	SPEC_BAND
br1.wav	-371.58478	97.99681	1.2696394	26.948123	-5.5174..	-7.5854..	-12.512..	-10.121..	-17.1178..	-2.98886	-5.49521..	-6.685885	-2.24671..	0.89442335	0.042011652	0.082185325	1389.44849	1475.818042
br10.wav	-311.64117	71.8437	-20.855787	17.992466	-9.5752..	9.301216	-38.153..	-14.888..	-10.2108..	-14.74765	-10.3879..	-1.23451..	-3.26118..	0.1654589	0.0749384	0.179696563	2387.459302	1789.181398
br100.wav	-314.17575	136.41866	-52.862125	-15.473975	-45.57	-17.382..	-19.885..	-25.647..	-8.209912	-17.9866..	-11.3515..	-1.02154..	-9.109297	0.10845537	0.048391595	0.070919736	1157.114627	1057.42618
br1000.w..	-411.62485	180.74176	-21.987639	-30.858137	14.73692	-6.6353..	-30.891..	-23.473..	-18.24386	-22.5577..	-11.4554..	0.080584..	-9.340458	0.04552519	0.023924494	0.049110148	755.8614999	628.3562573
br1001.w..	-395.1891	115.52895	-60.058544	-21.238525	-16.252..	-4.06191	-12.676..	-23.779..	-25.0176..	-18.3288..	-6.29929..	-8.56261	-16.1184..	0.043661192	0.021089355	0.10104589 6	1474.43887	1225.782428
br1002.w..	-444.23044	183.88804	-67.3569	-22.468896	-14.513..	-15.188..	-20.878..	-28.818..	-20.1447..	-28.3498..	-10.8227..	-10.2128..	-14.8385..	0.041965634	0.021163821	0.086354488	1330.128955	1130.997963
br1003.w..	-460.6686	82.43838	-56.537132	-0.982005..	-10.156..	-13.299..	-19.948..	-18.834..	-10.3315..	-18.4782..	-11.25557	-12.6388..	-11.9258..	0.03863026	0.01096465	0.158295461	1956.288741	1311.240646
br1004.w..	-486.20416	88.814415	-28.644382	-8.888315	-24.186..	-3.3715..	-16.832..	-12.372..	-17.39554	-16.6888..	-11.4298..	-13.9578..	-4.09668..	0.0294401	0.01439334	0.154863327	2068.978166	1432.011578
br1005.w..	-472.72983	85.934616	-52.70556	-4.4337573	-24.287..	-3.5279..	-13.564..	-23.582..	-23.1856..	-17.7254..	-4.35878..	-11.4168..	-12.2348..	0.02659579	0.013010044	0.152288987	1963.415459	1544.368833
br1006.w..	-428.7819	133.26683	-51.29616	-28.48226	-23.775..	2.78399	-23.686..	-32.151..	-15.3258..	-27.6756..	-19.0787..	-2.58791..	-10.1228..	0.041352637	0.020945476	0.107004967	1685.841357	1137.548303
br1007.w..	-469.2118	138.77188	-33.259052	-31.993338	-13.772..	-3.0695..	-19.893..	-22.778..	-10.4778..	-30.8603..	-26.3482..	-3.69807..	-7.193847	0.044793375	0.022714142	0.079819823	1154.95928	999.8396026
br1008.w..	-388.46854	104.84487	-42.834183	-35.270443	-10.126..	2.71459..	-22.432..	-20.125..	-17.5384..	-24.85317	-22.2028..	-5.125745	1.732822	0.049830645	0.026394054	0.102273963	1515.089108	1297.184472
br1009.w..	-370.72717	131.88878	-54.474487	-25.149885	-8.5419..	-21.499..	-10.307..	-13.264..	-22.46355	-16.7367..	-8.991663	-10.84739	-9.140469	0.05994618	0.029670732	0.086848354	1215.331355	893.8172994
br101.wav	-333.23685	157.71875	-57.1619	-11.068374	-9.8301..	-15.484..	-36.411..	-23.991..	-14.7842	-26.88539	-11.7862..	-7.270164	-2.67892..	0.08848264	0.03546998	0.068712841	1183.252002	1010.764215
br1010.w..	-415.48804	176.97389	-73.12666	-32.484774	-11.376..	-15.828..	-3.2601..	-19.376..	-16.8387..	-8.880765	-10.0158..	-8.764937	-13.1685..	0.043296717	0.020249786	0.084401283	1186.709814	763.5871391
br1011.w..	-432.26086	131.63521	-62.68894	-26.65105	9.458203	-8.1793..	-12.338..	-18.528..	-18.7468..	-12.9725..	-9.284213	-8.712776	-10.85083	0.048410586	0.019191978	0.09389048	1222.564888	849.5629979
br1012.w..	-486.18266	159.53743	-39.378735	-33.396477	-8.5009..	-13.816..	-12.637..	-12.598..	-15.38548	-18.7473..	-6.741688	-8.939418	-10.6378..	0.043245792	0.020836513	0.0739186	993.1373394	739.2175188
br1013.w..	-425.68634	149.07188	-65.75719	-38.127388	1.17984..	-10.894..	-10.338..	-7.5628..	-20.32891	-15.7726..	-8.356453	-13.6886..	-15.3712..	0.044295846	0.021401944	0.082176262	1119.985898	768.3231202
br1014.w..	-462.5183	197.11702	-53.161354	-23.77682	10.299	-17.564..	-10.358..	-10.782..	-21.4568..	-13.2982..	-3.88358..	-9.768719	-13.4321..	0.026456209	0.013267863	0.067194751	931.6215986	729.4689303
br1015.w..	-470.67435	111.69938	-7.662873	-15.985937	-21.865..	7.11486..	-14.999..	-12.996..	-10.8304..	-19.4665..	-9.110451	-1.63091..	-0.58894..	0.032282147	0.015779834	0.211046259	2328.594974	1594.993895
br1016.w..	-361.79752	167.74316	-58.458683	-9.728393	-6.7545..	-28.377..	-15.621..	-6.7024..	-27.23495	-17.8431..	-3.567821	-14.8125..	-6.85621..	0.06568669	0.027324062	0.082815219	1210.478718	1085.492646
br1017.w..	-343.21497	141.8739	-72.57481	1.2837406	-20.542..	-15.758..	-21.217..	-10.206..	-22.6724..	-9.579656	-2.97482..	-14.4374..	-0.38242..	0.07338937	0.02705413	0.090462877	1460.955678	1214.716446
br1018.w..	-355.7157	137.66199	-59.91001	10.311896	-21.473..	-14.636..	-26.849..	-12.954..	-17.6134..	-15.5945..	-9.02117..	-12.8798..	-4.19579..	0.060565233	0.02411649	0.096092044	1485.221089	1293.371503

Figure 5.12: All features of the audio sample stored in CSV format.

5.8 Speech Data

After completing the text corpus preparation, 50 individuals is requested, 25 males and 25 females between the ages of 17 and 54, to record their voice using their mobile phones. To improve the quality of the speech for the TTS synthesis model, each audio file gathered from participants is preprocessed using Audacity software. The audio data is captured at 48 kHz and ranged in duration from 0.7 to 40 seconds. Audio clips that contained sentences with words more than 15 or less or equal to 3 is removed. The audio files is than striped into one sentence wav file. Later the three step described by Jia et al. [11] is used to generate the synthesized Bangla language audio files. 8,466 original Bangla speech wav files were used for the TTS system that generated a total of 4,776 synthesis voices wav file. Therefore, a total of 13,242 audio clips dataset were used to detect synthesis or original voice, dialect of each audio file through the proposed method. To control the dialect, pronunciation of Bangla words is used in the manner the citizens of the specific region speaks. Table 5.4. shows a sample of the speech data used to create Bangla dataset with variation for dialect detection. Certain regions in Bangladesh have few words that are pronounced similarly while others are very different.

5.8.1 Real and Synthesized Speech

Collecting a considerable volume of computer-generated speech as well as genuine human speech is the initial stage in training a Synthetic Speech Detection system.

Table 5.4: Text Corpus for Bangla speech of the sampled English language sentence "A man has two sons" in Regional Bangla Language and Bangla/English abbreviation

District	Bangla/English Abbreviation	Bangla Language
Khulna	Ek jon mainsher duto chawal chil	অ্যাক জন মানশির দুটো ছাওয়াল ছিল।
Bogra	Yak jhoner duta beta achil	য়্যাক ঝনের দুটা ব্যাটা আছিল।
Rangpur	Ak jon mansher duikna beta achil	এক জন ম্যানশের দুইক্না ব্যাটা আছিল্।
Sylhet	Ak manusher dui powa achilo	এক মানুশর দুই পুয়া আছিল্।
Chittagong	Eguya manusher dui powa achilo	এগুয়া মানশের দুয়া পোয়া আছিল্
Noakhali	Akjoner dui hout achil	একজনের দুই হুত আছিল।
Mymensingh	Yeak joner dui put achil	য়্যাক জনের দুই পুৎ আছিল্

We built a dataset for our study that includes more than 84,000 synthetic utterances and more than 111,000 genuine utterances. Despite the fact that earlier researchers have created datasets comprising both actual and synthetic utterances [57][54], the focus of this study is on the most up-to-date speech synthesis methods based on neural network architectures. We also look at commercial technologies that may be used to produce synthetic speech, in addition to open-source systems.

Collecting Real Utterances

We need to assure a range of recording methods, a variety of speaker genders, a variety of speaker ages, a variety of accents, and even a variety of microphones utilized for recording to collect authentic utterances. This variant is necessary to prevent a situation in which an algorithm learns patterns in the training audio rather than the true distinctions between a synthetic and a genuine speech, resulting in poor upper formation in unknown data. To find voice datasets that may be used in this study, a thorough search was conducted. We found and gathered utterances from a variety of open source speech datasets as well as other real-life speech sources including TED Talks and Youtube video. Along with the audio speech recorded within the intuition.

All large utterances (above 10 seconds) were broken into 10-second maximum utterances once the audio was collected from the data sources. The split was done using the SoX audio processing application, which can identify silences in the audio and truncate the utterance in between phrases to maintain the naturalness of the speech files. After that, we'll get the final for-original dataset, which will be utilized later.

Collecting Synthetic Utterance

Before beginning to gather synthetic audio, considerable study was carried out to identify the most up-to-date speech synthesis approaches, such as Text-To-Speech (TTS) or Voice Conversion (VC) systems. Several open source and commercial systems were discovered throughout this investigation, including:

- DeepVoice 3 [44]
- Neural Voice Cloning [45]

- Baidu TTS [46]
- Microsoft Azure TTS [47]
- Amazon AWS Polly [48]
- Google Cloud TTS with Wavenet [49]

The next phase was to select phrases that would be utilized as input for TTS systems once the synthetic speech generators had been discovered. A dataset of English phrases [7] was utilized to achieve a good spread of grammatical phrase forms, whereas the text corpus established before was used for Bangla speech. There are almost 152,000 English phrases in the English phrase dataset. As a consequence, there are more than 105,000 phrases in a variety of grammatical patterns in the phrase collection. The phrase list was then split into 40 phrase buckets and dispersed across the TTS systems at random.

Following the identification of TTS systems and the generation of a list of English phrases, the next step was to run each TTS system with a set of phrases to retrieve the created synthetic speech. The utterance extraction procedure varies depending on the TTS system and will be detailed in depth in the next sections, but in general, free source TTS systems were run locally, while commercial tools were utilized through HTTP APIs. As a result, we have a collection of synthetic utterances that will be pre-processed and used in the training process.

5.9 Prepossessing Dataset

In this research unlabeled dataset is used to determine whether the proposed method is able to detect the features accurately. Noise form audio clips were remove and trimmed the silent sections. In order to have a static input zero-padding were added or trimmed it further to make a length of 10 second. In a speech spectrogram, different frequencies that are received by the human hear can be observed on the time axis. mel frequency cepstral coefficient (MFCCs) were used popularly as input data format to process audio features. It is made from the mel-cepstrum representation of a sound and are commonly utilized to analyze significant audio aspects [15]. However, because it applies the discrete cosine transform (DCT) to the logarithm of the filter banks' outputs, the MFCC features become decorrelated, making CNN unsuitable for non-local feature processing. As a result, MFECs is employed in this research as it does not require the DCT technique and calculate log-energies directly from the filter-bank energies. It delivers high-accuracy audio categorization results. With the consideration of 65ms analysis frames with a 50% overlap, 130 log mel-band energy characteristics from the magnitude spectrum for each MFEC data is received. Finally, the mel-spectrogram is segmented into 35 column data with a hop size of roughly 100 milliseconds every second. Since speech data represented in two-dimensional are fit for convolutional models. 25% of the dataset is used for testing purposes and 75% for training.

5.9.1 Audio Normalization

After collecting both synthetic and actual data and completing the for-original dataset, the next step was to pre-process the data so that it could be utilized by machine learning techniques. The following are the pre-processing processes, which were completed in the following order:

- Filetype Conversion:
  Because the files came from a variety of various data sources, the first step in the pre-processing procedure is to convert them all to the same file-type. All of the files were converted to the WAV filetype because it is the most prevalent format in machine learning and digital audio processing.

- Volume Normalization:
  Because each voice source has its unique volume settings, it's critical to equalize the level of all utterances to avoid volume being a differentiate. The volume of all utterances, both synthetic and actual, was set to 0dB.

- Sample-Rate Normalization:
  The majority of TTS systems create audio at 16kHz sample rate, but the bulk of genuine audio was captured at 48kHz sample rate. All audio samples were down sampled to 16kHz in order to save training time. Given that human voice has a frequency range of 300Hz to 5000Hz, down-sampling to 16kHz should not result in significant audio quality loss, as a 16kHz sample rate allows for frequencies up to 8kHz.

- Channel Mixing:
  Because most TTS systems output audio in a single channel (mono) while most actual audio has two channels (stereo), all two-channel files were transformed to a single channel using channel mixing, which means merging two audio tracks into a mono track by scaling each track by 0.5 and adding the signals to result in a single track.

- Silence Removing:
  Synthetic utterances had roughly 0.5 seconds of quiet at the beginning and conclusion of each statement, but genuine utterances had a more random silence pattern, according to early research. We took the silence out of the beginning and finish of each statement to eliminate any quiet bias.

- Gender Balancing:
  Female voices dominated the synthetic utterances, but male speakers dominated the real audio. Downsampling was used to balance the dataset in order to eliminate any gender bias during training and classification. As a consequence, a gender-balanced dataset was produced.

- Class Balancing:
  The dataset contains more genuine speech than synthetic ones after gender balancing. The dataset was downsampled to guarantee a 50/50 mix of synthetic and actual utterances in order to create a class balanced dataset.

5.9.2 Length Normalization

Early investigations revealed that synthetic utterances were significantly shorter than genuine utterances. Because this might be a source of bias in the dataset, all utterances lasting more than 2 seconds were shortened, while those lasting less than 2 seconds were eliminated.

5.10 Dataset division

The dataset was separated into three sections, as is customary in machine-learning research: training, validation, and testing:

- Training: Machine learning models were trained on 75% of the dataset. Gender and social status are both represented.
- Validation: 5% of the dataset was used to test the machine learning models' accuracy. Gender and social status are both represented. During the training phase, the validation utterances are hidden.
- Generalization Testing: This section contains 20% of the dataset. Only genuine voices and synthetic voices from one unknown algorithm (Google TTS Wavenet). Gender and social status are both represented. It's used to assess if the trained model can generalize and recognize genuine sounds that aren't visible.

Chapter 6

Proposed Model and Experimentation

The proposed method uses stacked convolutional autoencoder (SCAE) and MLELM framework to detect dialect, original/synthesized voice, and gender/age from MFEC speech input data. Through experimentation with various type of DL models, the best yielded model is a fully connected SCAE with MLELM for soft classification and score approximation for classes. To handle the spatial structure in an audio signal, convolutional autoencoder is used as it benefits in term of computational complexity, performance and retrain the hidden relationship between feature of the data. The features are then transferred to two MLELM, where the first machine predicts the soft labels and second machine connects the hard labels to the soft labels. Based on the anticipated scores for the classes, hard labels are assigned to the unseen data. Detailed description of the proposed model is presented in the subsequent sections and shown in figure 6.1.

6.1 Multi-label data representation

For the given number of classes C, each data instance $F_i = f_{i1}, f_{i2}, ..., f_{in}$, i= 1,..., N is paired with a vector of multiple outputs $O_i = o_{i1}, o_{i2}, ..., o_{iC}$ [21]. At any given time, the instances might belong to more than one class. The values in a vector are given in the binary format, 1 if the sample belongs to the class category and 0 if it does not have any features similar to the class category. As a result several class labels can be applied at once, which is not able to be done in signal-label data. Each class category combination label is called a label-set. Further discussion about the representation of the multiple labels are given in the following sections.

6.2 Stacked Deep Convolutional Autoencoder

Autoencoder are artificial neural networks that have several hidden layers with few nodes than the input and the output is expected to have a similar number of input. On the other hand Convolutional Neural Networks consist of three parts; convolutional layers, pooling layers and fully connected layers. In contrast to the AE structure, input, hidden and output layers. The input and output layers have n nodes for S samples in the autoencoder, each with feature vector $\mathbf{F}_i$ for $i = 1, ..., S$ and

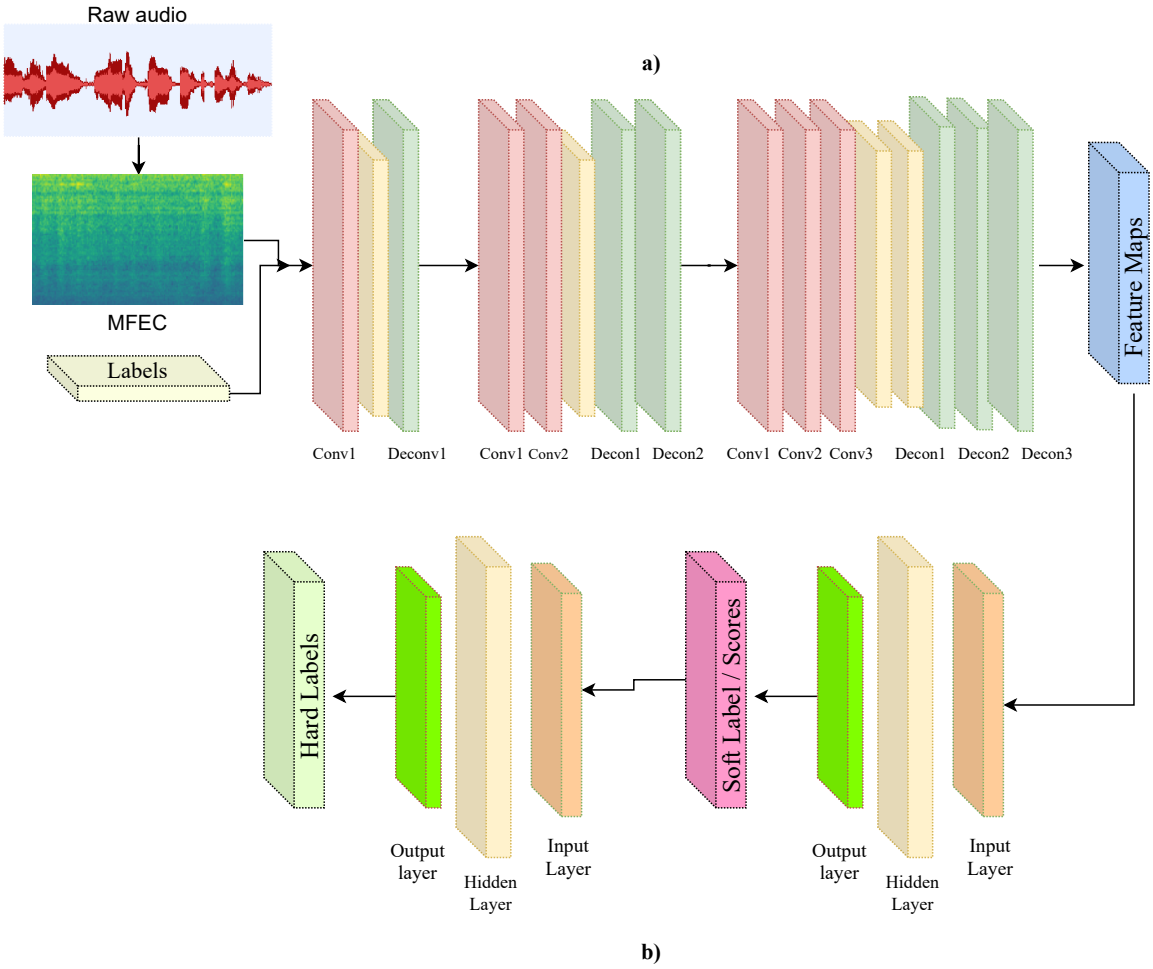


Figure 6.1: Architecture of Proposed Method consist of two parts. a) SCAE b) MLELMs

$\mathbf{F}_i \epsilon R^n$. The autoencoders work in unsupervised manner compared to the the feed forward network. An autoencoder input and output, $\mathbf{F}$= $f_1, f_2, f_3, ..., f_n$. and then in the encoder potion it converted n-dimensional data input to n'dimension. n' is smaller to n, to compress the input data to smaller dimension. Later in the decoded part of autoencoder, the decoded features in the n'-dimensional form be converted back n-dimension, which is decompressing the decoded features for the output nodes. The encoder maps the input $\mathbf{F}$ to a set of hidden nodes H= $h_1, h_2, h_3, ..., h_n$. The node h_j output is computed as

$$h_j = \varphi(\sum_{i=1}^{n} w_{ij} f_i + b_j), \tag{6.1}$$

where φ represent the transfer function in the encoder section, i start from 1. w_{ij} is the weight between the f_i and h_j, and the b_j stands for the bias.

$$f'_k = \varrho(\sum_{j=1}^{n'} w_{jk} h_j + b'_k). \tag{6.2}$$

In the decoded position function maps the H from encoded representation to es-

Table 6.1: Proposed Method Detailed architecture

Layer Names	Architecture	Feature Map size	Parameters
Conv1	32x128x1	32x64x32	29K
Maxpool	(Max2x2)	32x32x32	
Bottleneck-Conv	32x32x32	8x8x16	768K
Deconv1	8x8x16	32x64x32	
unpool	Max(2x2)	32x128x1	
Conv1	32x128x1	32x32x64	228K
Maxpool	(Max2x2)	16x16x128	
Conv2	16x16x128	8x8x256	12K
Maxpool	(Max2x2)	4x4x512	
Bottleneck-Conv	4x4x512	2x2x124	221K
Deconv1	2x2x124	8x8x256	
unpool	Max(2x2)	16x16x128	
Deconv2	16x16x128	32x32x64	
unpool	Max(2x2)	32x128x1	
Conv1	32x128x1	32x64x32	30K
Maxpool	(Max2x2)	32x32x64	
Conv2	32x32x64	16x16x128	250K
Maxpool	(Max2x2)	8x8x128	
Conv3	8x8x128	4x4x256	12K
Maxpool	(Max2x2)	4x4x512	
Bottleneck-Conv1	4x4x512	4x4x256	885K
Bottleneck-Conv2	4x4x256	2x2x124	885K
Deconv1	2x2x124	4x4x256	
unpool	Max(2x2)	8x8x128	
Deconv2	8x8x128	16x16x128	
unpool	Max(2x2)	32x32x64	
Deconv3	32x32x64	32x64x32	
unpool	Max(2x2)	32x128x1	

timated **F'**. Hence, the output of the node f'_k for the k^{th} position is as stated in Equation 2. ϱ act as the transfer function in the decoded side, j begins from 1, w_{ij} is the weight connection value for the h_j and f'_k nodes and b'_k is the bias for the k^{th} node in the decoder. Similar to the multi-layer perceptron the weights are updated through the iterative training of the autoencoders through backpropagation.

The proposed model includes convolutional layers with ReLu activation functions followed by max pooling layers in the encoder section. The use of a convolutional autoencoder is applied to the model for better computational complexity and performance [18]. The components in the encoder part map the input vector to the lower dimensional hidden representation through the use of nonlinear transform. Then, the reverse transform is reconstructed from the hidden representation to the original audio input signal in the decoded part of the model. Reverse transform serves as the new representation of input sample for another convolutional autoencoder and so on to form SCAE, which is constructed similar to a Stacked Autoencoder (SAE). All the structures in the model for both encoded and decoded parts are kept symmet-

rical to find a series of low-dimensional hierarchical features in the data [18][19][20].

In the final bottleneck position, a convolutional layer is used to obtain a vector by flattening all the units and passing it on to an embedded layer which is a fully connected low dimensional unit. Resulted in the 2D input data to be now transformed into lower dimensional features. The feature vector is then later used by the multi-label extreme learning machines. To reduce the re-construction error the parameters of the decoder and encoder were updated. Table 6.1. provides the architecture of SCAE.

6.3 Extreme Learning Machine (ELM)

ELM is an efficient, compact and sophisticated single layer feed forward neural network that performs the classification task in an systematic and express manner [39]. The MLELM architecture is composed of 3 layers: input, hidden, and output. Here the input samples are denoted as $\mathbf{F}$ and $\mathbf{F}\ \epsilon R^n$, class labels are represented as $\boldsymbol{Y}$ where $\mathbf{Y}\ \epsilon R^E$. The input layer have I number of nodes, hidden layers have D and output layers have E number of nodes. The weights for the input layer to hidden nodes are represented by ω, while the weights from hidden layer to output is denoted by ϖ

Unlike most artificial neural networks (ANNs), the weights associated with the input layer and the biases in MLELM network are randomly initialized and are not updated later. Learning from the data input takes place only in the hidden layer in the network that is reflected in the wights of the hidden layer. ϑ is the activation function for the hidden nodes. h_j hidden node, output from the hidden layers are calculated as follows

$$h_j = \vartheta(\sum_{i=1}^{n} \omega_{ij} f_i + b_j), \tag{6.3}$$

where, ω_{ij} represents the connection weight between f_i and h_j, and b_j is the bias. As a result the output node o_k

$$o_k = \sum_{D}^{j=1} h_j \varpi_{jk}, \tag{6.4}$$

ϖ_{jk} represent the weight between the h_j and o_k. Once the MLELM model obtains the weight matrix of ϖ, it is considered to have learned iteratively from the training phase. The model than undergoes through testing phase and later class predication.

The topology of the ELM network [21] to perform multi-label classification and score prediction is used in the proposed system. The encoded features obtained from the SCAE are used as input and class labels are provided as output from the multi-label extreme learning machines.

In a data set with a signal-label, the sample is allocated with the highest values that correspond to the class-label. However, with multi-label data, based on the score

achieved, numerous class labels might be assigned to one sample. The threshold setting determines the hard multi-labels. If the anticipated value exceeds the threshold, the class is considered relevant, and the label is 1; otherwise, it is 0. The drawbacks of this strategy are that a low threshold might assign numerous labels, while a high threshold will apply labels to data instances, resulting in misclassification.

In comparison to the amount of input nodes employed in MLELM, the ELM requires a larger number of hidden nodes to learn efficiently from the data. The amount of features in input data to MLELM has been decreased as a result of use of SCAE. As a consequence, the weight matrix will be compact and the concealed layer will be small. The soft classification label scores for that particular class were built after the weight matrix is obtained. The next MLELM model takes them as input and predicts the original target labels as output. Random initialization of input weights and biases is done in the second MLELM. We used a second MLELM to avoid using a specific threshold to forecast classes, as is done in a standard ELM. Using a calibrated threshold, the final score is transformed to hard class labels.

6.4 Feature Extraction

The input data is provided to the SCAE to perform feature extraction in the training phase. To minimize the Mean Squared Error (MSE) between the input data and the reconstructed output, the encoder and decoder were trained with a learning rate of 0.001 and Adam Optimizer. The training process is scheduled for 200 epochs but, after no change in validation loss for 10 epochs the training process is hauled and the best saved model is selected and used for the testing process. In the convolutional autoencoder, the number of layers and decreased number of features are specified. The model is iteratively trained until it recognizes the input completely. The encoded feature is delivered to the next step after it has been retrieved from the SCAE network.

6.5 Prediction of Soft Class

The encoded characteristics generated from the SCAE are fed into the multi-label ELM network for soft class prediction. The number of hidden layers in MLELM is determined by input nodes. It operates in batch mode, that is accepting all of the input instances and learning at once. Once MLELM network has learnt the weighs ϖ from the hidden layer, feature encoded training data $\mathbf{a}$ are fed back into the MLELM network to create class scores.

$$o'_k = \sum_{D}^{j=1} h_j \varpi_{jk}, \tag{6.5}$$

the predicted score is calculated through the above equation, ϖ_{jk} weight matrix between hidden node h_j and output node o_k. The result of this is a output layer where all nodes contains the soft classification score for the respective class. After obtaining the projected score, it is transferred to the second MLELM network, which improves the prediction by matching the class scores to the true class labels. The

weights of the hidden layer are likewise learned by the second MLELM network in a single run.

6.6 Testing Stage

After training the SCAE and multi-label extreme learning machine networks, the test data fed into the SCAE network independently. Unsupervised sets of encoded features were constructed and then fed to the MLELM networks. The first MLELM model creates individual class scores for each of the test patterns, which are then input into the second MLELM model. The soft class scores are then mapped to actual class labels, and the test data's hard class labels are determined as a result.

Chapter 7

Result and Discussion

All networks were implemented using Python programming language on various GPUs; GeForce RTX 3090, Radeon RX 590 and NVidia GeForce RTX 3070. The model was tested on unseen data, which are new formed Bangla phrases from the unique Bangla/English abbreviation words. For example the new English sentence from the trained words "Man is happy with son", the abbreviate sentence would be "Ak manusher khusi niye powa ", Bangla sentence " এক মানুশর খুশি নিয়ে পুয় "

To assess the effectiveness of the proposed model SCAE-MLELMs, we compared it to two other existing models, dense autoencoder (DAE) and convolutional autoencoder (CAE), and employed AUC and pAUC performance measures. Statistical parameters were used of the model [8] and the ground truth to assess the performance and robustness of SCAE-MLELMs for categorization of audio type, dialect, gender, and age.

Confusion matrix is used as it displays the true positive (TP) value, the number of positively predicted samples that properly match the true positive labels, false negative (FN) value, which is the number of positively predicted samples that do not match the positive ground truth labels. True negative (TN) samples are those that were accurately predicted as negative and had real values, whereas false positive (FP) samples are those that were forecasted as negative but had actual labels that were positive. To understand the number of positively predicted labels for test data accuracy score is measured.

Additionally, recall (R), precision (P), F1-score (FS) measurements were used to understand the effectiveness of the model, as accuracy score alone is not sufficient enough to measure a model effectiveness and performance. Following sections provides discussion, confusion matrix and tables with values of the matrix obtained from the models for each specific type of classes and its categories, correlation of age with dialect class classification, comparison among datasets and exiting algorithms. The use of Bangla Speech and English speech datasets were used to train and test the model.

$$Recall = \frac{TruePos}{TruePos + FalseNeg} \tag{7.1}$$

$$Precision = \frac{TruePos}{TruePos + FalsePos} \tag{7.2}$$

$$F1 - score = \frac{2 * TruePos}{2 * TrueNeg + FalsePos + FalseNeg} \tag{7.3}$$

$$Accuracy = \frac{TruePos + TrueNeg}{TruePos + TrueNeg + FalsePos + FalseNeg} \tag{7.4}$$

7.1 Bangla Speech

7.1.1 Type of Audio

The type of audio classification for the Bangla speech datasets is a two class category problem; original or synthesized speech. The proposed method recognizes the actual Bangla voices from generated voices at a high rate. The highest values obtained for precision, recall and F1-score for Bangla speech are 91%, 94%, 93% respectively with a mean accuracy for recognition is 93%, as observed in Table 7.1. Confusion matrix obtained from the model prediction for type of audio classification problem is present in figure 7.1 for Bangla speech. The best category-wise accuracy for Bangla speech is 94% for original. Low rate of false classification for either categories.

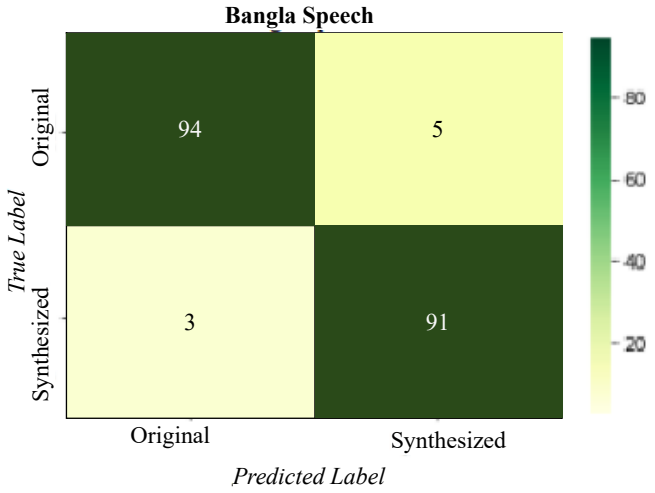


Figure 7.1: Confusion Matrices of Type of Audio for Bangla Speech.

Table 7.1: Classification Results for Type of Audio for Bangla Speech; precision (P), recall (R), f1-score (FS) by using the SCAE-MLELMs model.

Class Group	P	R	FS
Original	0.90	0.94	0.93
Synthesized	0.91	0.92	0.91
Accuracy		**0.93**	

7.1.2 Dialect

The dialect classification for Bangla speech is a seven category classification problem; Khulna, Bogra, Rangpur, Sylhet, Chittagong, Noakhali, Mymensingh. The highest values obtained for precision, recall and F1-score for Bangla speech is 83%, 78%, 72% respectively as observed in table 7.2. The mean accuracy for recognition is 75% for Bangla speech.

Table 7.2: Classification Results of Dialect for both Bangla and English Speech precision (P), recall (R), f1-score (FS) by using the SCAE-MLELMs model.

Class Group	P	R	FS
Khulna	0.67	0.44	0.64
Bogra	0.78	0.66	0.72
Rangpur	0.83	0.52	0.65
Sylhet	0.80	0.54	0.58
Chittagong	0.66	0.44	0.87
Nokhali	0.72	0.78	0.70
Mymensingh	0.85	0.36	0.64
Accuracy		**0.85**	

These results show that the bigger the variation in dialect type, the better the recognition rate. Because all of the regional languages are spoken in Bangla language in the Bangla speech dataset, it is difficult to identify the input audio from each other at a high pace. Confusion matrix obtained from the model prediction for dialect classification problem is present in figure 7.2 for Bangla speech. In Bangla speech M, N, C, S, R, B, K stands for Mymensingh, Noakhali, Sylhet, Rangpur, Bogra, Khulna, respectively. The best category-wise accuracy for Bangla speech is achieved by Noakhali (78%) followed by Bogra (66%). However, the proposed model confuses the prediction for Bogra with Rangpur and Sylhet with Chittagong, 23% and 32% respectively, falsely predicted. One of the main reasons for this confusion is the similarity of the acoustic features, frequency and intensity is similar between the words used in those regional parts [29].

7.1.3 Dialect and Age correlations classification

The results indicate that the more one ages the higher the recognition rate for the speaker's dialect. As children have a more smooth acoustic features compared to the high pitch-shift in adult tone. As stated by Hanjun, Nichole, and Charles [25], aging has physiological change that impact the processing of auditory feedback in the brain. Hence, considering the age feature maps when predicting the dialect of a speaker, yields far better accuracy percentage than considering predicting classes alone. Due to which low false prediction can be observed for dialect between Sylhet-Chittagong and Bogra-Rangpur regional languages.

The age and dialect correlation classification for Bangla speech is a fourteen class classification problem; Child-Adult; Khulna-Bogra-Rangpur-Sylhet-Chittagong-Noakhali-Mymensingh. The highest values obtained for precision, recall and F1-score for

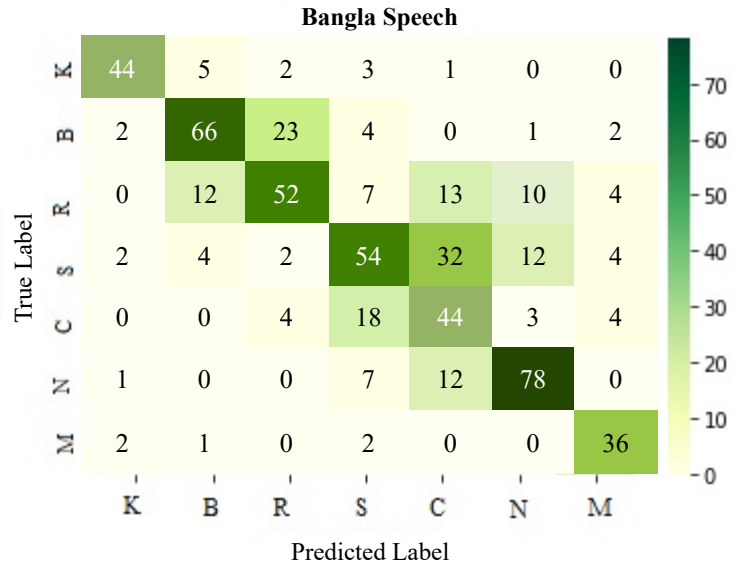


Figure 7.2: Confusion Matrices of dialect for Bangla Speech.

Bangla speech 90%, 78%, 76% respectively, as observed in table 7.3. The mean accuracy for recognition is 92% for Bangla speech. Confusion matrix obtained from the model prediction for dialect classification problem is present in figure 7.3 for both speeches. In Bangla speech; CK, CB, CR, CS, CC, CN, CM, AK, AB, AR, AS, AC, AN, AM stands for Child-Khulna, Child-Bogra, Child-Rangpur, Child-Sylhet, Child-Chittagong, Child-Noakhali, Child-Mymensingh, Adult-Khulna, Adult-Bogra, Adult-Rangpur, Adult-Sylhet, Adult-Chittagong, Adult-Noakhali, Adult-Mymensingh, respectively. 34% of Child-Rangpur, 23% of the Child-Chittagong and 21% of Child-Sylhet were falsely classified to Child-Bogra, Child-Sylhet and Child-Chittagong respectively. Due to the smooth acoustic frequency in a child voice making it hard for the model to recognize the words spoken in the speech. We could improve this results by increasing the number of input data from these respective classes during the training stage.

7.1.4 Other feature extraction classification

Age

The age range in Bangla Speech is different for each age group. The child age group is between 12 to 20, while the adult age group is 30 to 50 years. The highest values obtained for precision, recall and F1-score for Bangla speech are 89%,95%, 92%, respectively, as observed in table 7.4. These indicate that the greater the difference in age range the higher the recognition rate. The mean accuracy for Bangla speech is 95%.

Confusion matrix obtained from the model prediction for age classification problem is present in figure 7.10 for Bangla speech. The best category-wise accuracy for Bangla speech is obtained by the Child group, 95%. Although 15% of the class was falsely classified as Adult. One of the main reasons for this confusion is the similarity of the acoustic pitch features, the fundamental frequency is similar between the ages

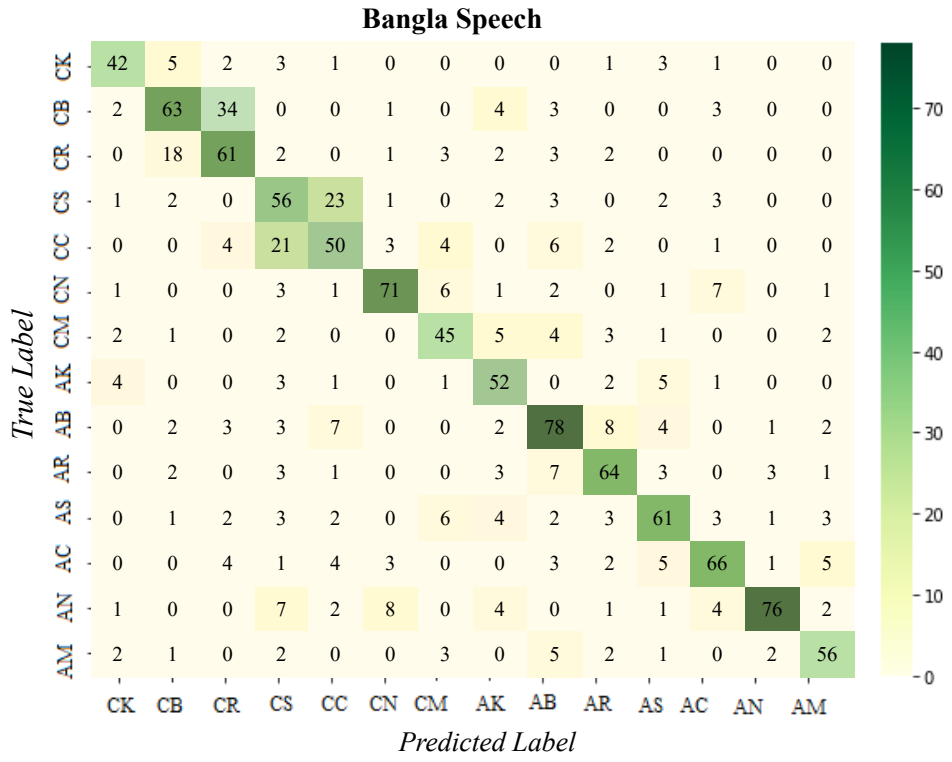


Figure 7.3: Confusion Matrices of dialect and age correlation for Bangla Speech

[25,26]. As MFEC data log-energies of the audio signals, the frequency features for certain children that are transitioning into maturity phase have high acoustic feature similar an adult.

Gender

The gender classification for the Bangla speech dataset is two class category problems; male and female. The highest values obtained for precision, recall and F1-score for for Bangla speech are 85%,94%, 93%, as observed in table 7.5. The mean accuracy for recognition is 92% for Bangla speech. Confusion matrix obtained from the model prediction for age classification problem is present in figure 7.4 for Bangla speech. The best category-wise accuracy for Bangla speech is achieved by male category, 87%. The proposed model has 10% false predictions for male class compared to the the female class.

7.2 English Speech

7.2.1 Type of Audio

The type of audio classification for the English speech dataset is a two class category problem; original or synthesized voice. The proposed method recognizes the synthesized English voices from actual voices at a high rate. The highest values obtained for precision, recall, F1-score and mean accuracy for English speech are 94%, 97%, 94%, 96%, respectively, as observed in Table 7.6. Confusion matrix obtained from the

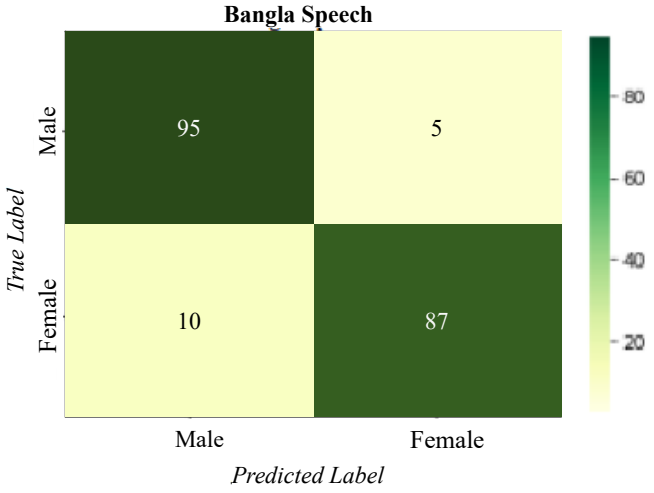


Figure 7.4: Confusion Matrices of Gender for Bangla Speech

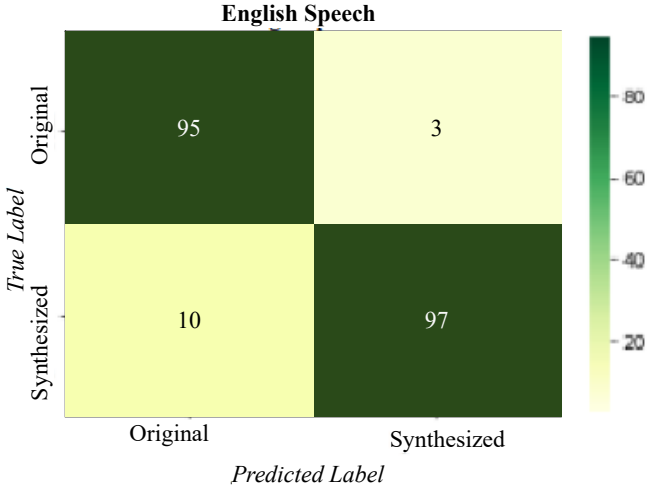


Figure 7.5: Confusion Matrices of Type of Audio for English Speech

model prediction for type of audio classification problem is present in figure 7.5 for English speech. 10% of the original class label were falsely predicated as synthesized voices.

7.2.2 Dialect

The dialect classification for English speech is a three category classification problem; Asian (Bangladesh, Indian, Pakistan, China, Korean), American and European (United Kingdom, Germany, Russia). The highest values obtained for precision, recall and F1-score for English speech is 81%, 88% and 85%, respectively as observed in table 7.6. The mean accuracy for recognition is 81% for English speech.
These results show that the bigger the variation in dialect type, the better the recognition rate. In English speech, where the dialects are considerably varied, making the work of recognition fairly simple. Confusion matrix obtained from the model prediction for dialect classification problem is present in figure 7.6 for English

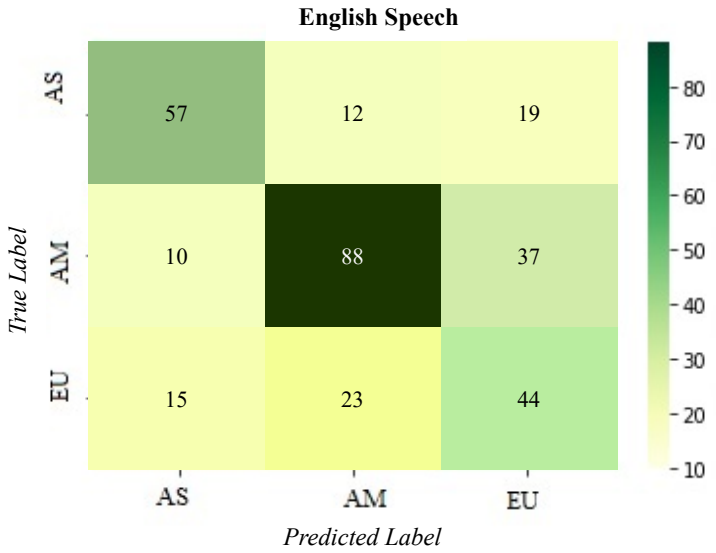


Figure 7.6: Confusion Matrices of dialect for English Speech

speech. In English speech European, American and Asian is denoted with keywords EU, AM, AS, respectively. The best category-wise accuracy for English speech is achieved by American (88%) followed by Asian (57%). However, the proposed model confused the prediction for American with European and visa-versa, 23% falsely predict of American to European and 37% falsely predicted for European for American. One of the main reasons for this confusion is the similarity of the acoustic features, frequency and intensity is similar between the words used in those regional parts [29].

7.2.3 Dialect and Age correlations classification

The results indicate that the more one ages the higher the recognition rate for the speaker's dialect. As children have a more smooth acoustic features compared to the high pitch-shift in adult tone. As stated by Hanjun, Nichole, and Charles [25], that aging has physiological change that impact the processing of auditory feedback in the brain. Hence, consider the age feature maps when predicting the dialect of a speaker, yields far better accuracy percentage than considering predicting classes alone. Due to which low false prediction can be observed for dialect between American and European speakers. As well as the confusion between close age groups 30s and 40s is also reduced when considering two class labels.
For English Speech dataset is a twelve category classification problem; 20s-30s-40s-50s; Asian-American-European. The highest values obtained for precision, recall and F1-score for English speech are 83%, 87%, 86% respectively, as observed in table 7.8. The mean accuracy for recognition is 87% for English speech. Confusion matrix obtained from the model prediction for dialect classification problem is present in figure 7.7 for English speech. In English speech; 2AS, 2AM, 2EU, 3AS, 3AM, 3EU, 4AS, 4AM, 4EU, 5AS, 5AM, 5EU stands for 20s-Asian, 20s-American, 20s-European, 30s-Asian, 30s-American, 30s-European, 40s-Asian, 40s-American, 40s-European, 50s-Asian, 50s-American and 50s-European, respectively.

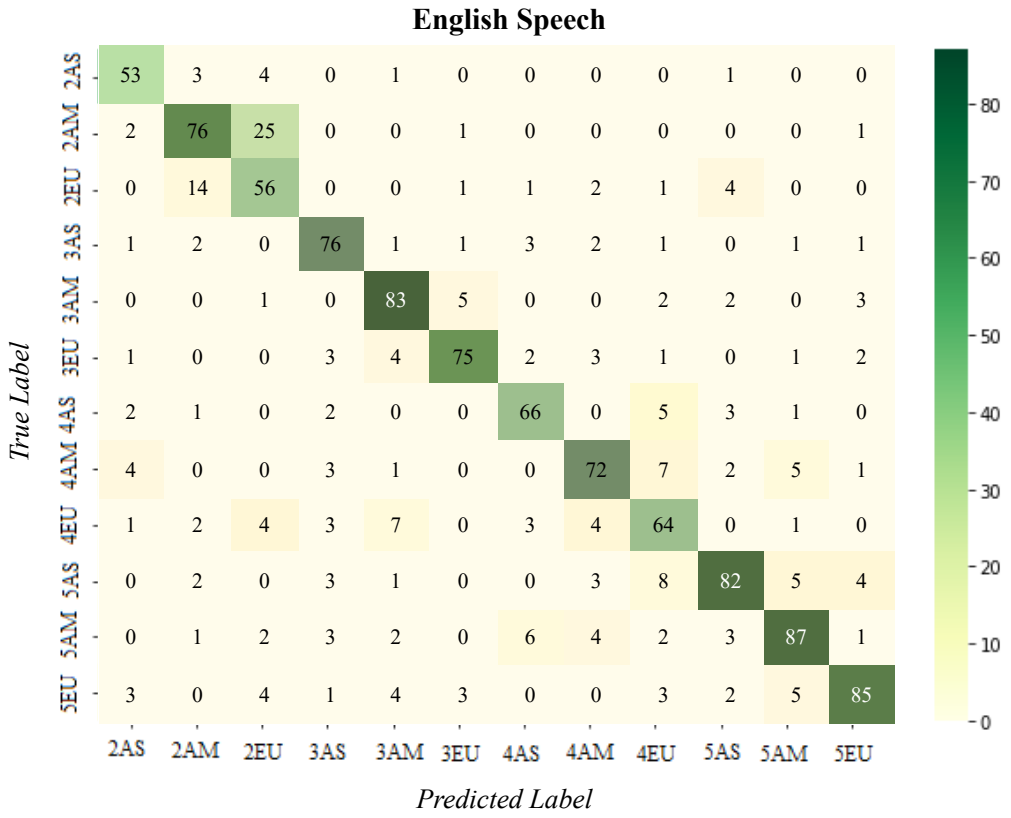


Figure 7.7: Confusion Matrices of dialect and age correlation for English Speech

7.2.4 Other Feature Extraction classification

Age

The age classification for the English Speech datasets is four class classification problem; 20s, 30s, 40s, and 50s. Difference between the classes is 10 years. The highest values obtained for precision, recall and F1-score for English speech are 88%, 85%, 86% respectively, as observed in table 7.9 These indicate that the greater the difference in age range the higher the recognition rate. The mean accuracy for recognition is 82% for English speech.
Confusion matrix obtained from the model prediction for age classification problem is present in figure 7.8 for English speech. The best category-wise accuracy for English speech is achieved by twenties (85%) followed by fifties (81%). However, the accuracy rates for thirties and forties 75% and 67%, respectively. The proposed model confuses the prediction for thirties age group with forties and via-versa, 25% and 45%, respectively, falsely predicted. One of the main reasons for this confusion is the similarity of the acoustic pitch features, the fundamental frequency is similar between the ages [25][26]. AS MFEC data log-energies of the audio signals, the frequency features for the two age groups are quite similar.

Gender

The gender classification for English speech dataset is two class category problems; male and female. The highest values obtained for precision, recall and F1-score for

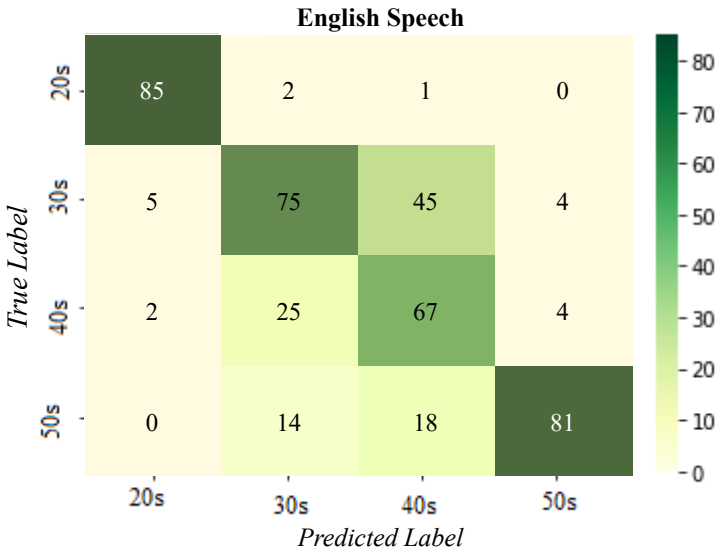


Figure 7.8: Confusion Matrices of Age for English Speech

for English speech are 96%, 98%, 96% respectively, as observed in table 7.10. The mean accuracy for recognition is 96% for English speech. Confusion matrix obtained from the model prediction for age classification problem is present in figure 7.9 for English speech. The best category-wise accuracy for English speeches is achieved by male category, 98%. The proposed model has low false predictions for English speech dataset.

7.3 Comparison among existing datasets

The number of Convolutional Autoencoders in the SCAE-MLELMs model is varied to evaluate the accuracy in predicting the class label for the test data and to determine the influence input data format had on the suggested system. Spectrograms and MFECs is chosen as the two input data types. As they are the most often used data format in audio recognition studies. The dataset mentioned in the earlier sections is utilized for Bangla speech. While for English Speech, freely available datasets from Google AudioSet and VoxCeleb is utilized. Table 7.11 shows the categorization accuracy in the specifics of the experiment with spectrogram as data input format and Table 7.12 for MFECs data format.

Four CAE combinations with MLELMs is tested upon; Model 1 employs only one CAE network with MLELMs, which is used as a baseline model to assess the efficiency of the suggested methods. Models 2, 3, and 4 contain three, four, and six CAE networks, respectively, followed by MLELM networks; a comprehensive architectural description can be found in this paper's proposed model section. Model 4 gives the maximum classification accuracy for all classes labels for both types of speeches for all datasets in spectrogram data format. Detecting the prominent aspects of an audio stream in a spectrogram requires a greater number of convolutional autoencoders. Whereas MFECs data offers the maximum classification accuracy using Model 2, since its log mel-energy properties are more easily discernible. Additionally, the model leans towards overfitting as the number of CAE network is increased for

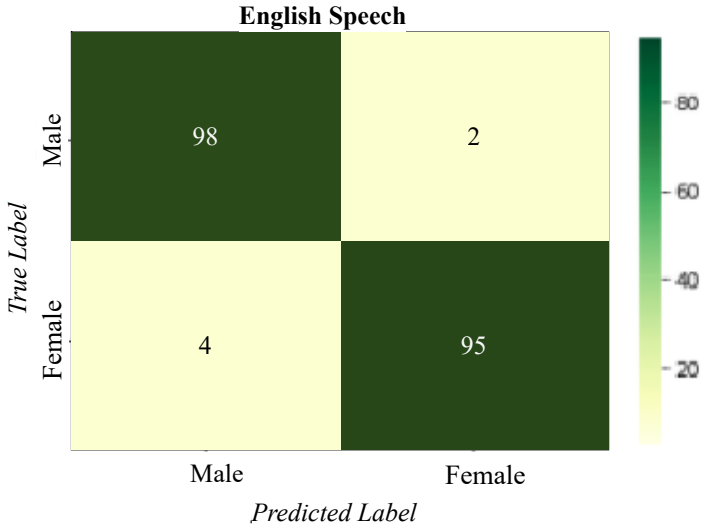


Figure 7.9: Confusion Matrices of Gender for English Speech

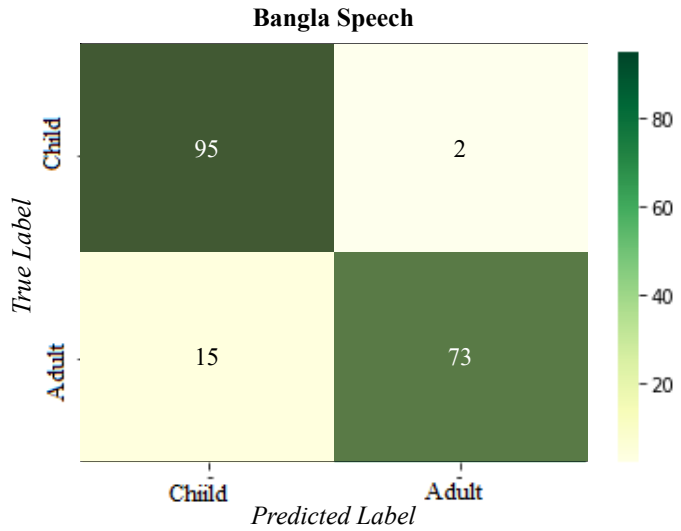


Figure 7.10: Confusion Matrices of Age for Bangla Speech

MFECs input data format, classification accuracy of Model 3, 4 compared to model 2. Furthermore, across all sorts of datasets of Bangla and English speech, dialect and type of audio has the greatest predicted accuracy.

7.4 Comparison among existing Algorithms

To assess the performance and efficiency of the system, it is compared with the existing models developed by Ribeiro [10], a Deep CNN model, and Tursunov [8], a multi-attention module CNN model, for each classification category. The performance and robustness of the techniques using the AUC and pAUC measurements were compared. Table 7.13 shows the AUC and pAUC values for each class category with spectrogram input data format. While Table 7.14 shows the MFEC data format for both Bangla and English audios. The average AUC and pAUC matrix for each

Table 7.3: Classification Results of dialect and age correlation for Bangla Speech, precision (P), recall (R), f1-score (FS) by using the SCAE-MLELMs model.

Class Group	P	R	FS
Child-Khulna	0.75	0.42	0.57
Child-Bogra	0.68	0.63	0.72
Child-Rangpur	0.76	0.61	0.58
Child-Sylhet	0.60	0.56	0.64
Child-Chittagong	0.66	0.50	0.65
Child-Nokhali	0.79	0.71	0.67
Child-Mymensingh	0.88	0.45	0.75
Adult-Khulna	0.83	0.52	0.64
Adult-Bogra	0.86	0.78	0.74
Adult-Rangpur	0.89	0.64	0.75
Adult-Sylhet	0.79	0.61	0.68
Adult-Chittagong	0.73	0.66	0.75
Adult-Nokhali	0.83	0.76	0.67
Adult-Mymensingh	0.90	0.56	0.76
Accuracy		**0.92**	

Table 7.4: Classification Results of Age for Bangla and English Speech precision (P), recall (R), f1-score (FS) by using the SCAE-MLELMs model.

Class Group	P	R	FS
Child	0.89	0.95	0.81
Adult	0.66	0.73	0.92
Accuracy		**0.95**	

class category for both data formats demonstrate that SCAE-MLELMs model outperforms the current model for both speeches.

Furthermore, when compared to spectrogram data formats, MFEC data formats had greater AUC and pAUC values for all model types. When compared to other classes, the Deep CNN model [10] has the lowest AUC and pAUC performance values for dialect class labels. When compared to other classes, the Deep CNN model [10] has the lowest AUC and pAUC performance values for dialect class labels. When compared to approaches developed in article [10], the Multi-attention module CNN model [8] produced a few top results for a few classification labels; age, gender, and audio type. Due to its single-label model structure and lack of ability in learning characteristics that integrate age and dialect in audio frequency pattern, existing approaches have difficulty distinguishing dialect in speech of any language. As addressed in the suggested model, employing multi-label extreme learning machine networks. Furthermore, the existing methods do not perform as well in Bangla speech audio input as they do in English speech. The suggested system's performance is consistent across both speech languages.

Table 7.5: Classification Results of Gender for Bangla Speech precision (P), recall (R), f1-score (FS) by using the SCAE-MLELMs model.

Class Group	P	R	FS
Male	0.82	0.94	0.93
Female	0.85	0.87	0.90
Accuracy		**0.92**	

Table 7.6: Classification Results of Type of Audio for English Speech, precision (P), recall (R), f1-score (FS) by using the SCAE-MLELMs model.

Class Group	P	R	FS
Original	0.94	0.95	0.94
Synthesized	0.93	0.97	0.93
Accuracy		**0.96**	

Table 7.7: Classification Results of Dialect for English Speech, precision (P), recall (R), f1-score (FS) by using the SCAE-MLELMs model.

Class Group	P	R	FS
Asian	0.61	0.57	0.70
American	0.76	0.88	0.76
European	0.81	0.44	0.85
Accuracy		**0.81**	

Table 7.8: Classification Results of dialect and age correlation for English Speech, precision (P), recall (R), f1-score (FS) by using the SCAE-MLELMs model.

Class Group	P	R	FS
20s-Asian	0.54	0.53	0.57
20s-American	0.71	0.76	0.68
20s-European	0.53	0.56	0.65
30s-Asian	0.63	0.76	0.70
30s-American	0.80	0.83	0.74
30s-European	0.70	0.75	0.86
40s-Asian	0.61	0.66	0.69
40s-American	0.56	0.72	0.66
40s-European	0.73	0.64	0.85
50s-American	0.81	0.87	0.79
50s-European	0.83	0.85	0.79
Accuracy		**0.87**	

Table 7.9: Classification Results of Age for English Speech, precision (P), recall (R), f1-score (FS) by using the SCAE-MLELMs model.

Class Group	P	R	FS
20s	0.76	0.85	0.72
30s	0.88	0.75	0.87
40s	0.87	0.67	0.70
50s	0.68	0.81	0.76
Accuracy		**0.82**	

Table 7.10: Classification Results of Gender for English Speech precision (P), recall (R), f1-score (FS) by using the SCAE-MLELMs model.

5Class Group	P	R	FS
Male	0.96	0.98	0.96
Female	0.94	0.94	0.93
Accuracy		**0.96**	

Table 7.11: Classification Accuracy (%) of the four different SCAE-MLELMs architecture on different datasets with input format as **spectrogram**; Brac University previous and self-built Bangla Speech dataset and Google Audio-Set and VoxCeleb for English speech dataset is used during the experiment. Numbers in bold represent the highest classification accuracy.

Model No.	**Bangla Speech**			**English Speech**					
	Brac University			**Google AudioSet**			**VoxCeleb**		
	Audio type	Dialect	Gender/Age	Audio type	Dialect	Gender/Age	Audio type	Dialect	Gender/Age
1	76	75	84	67	79	76	87	78	84
2	74	86	90	78	81	73	90	82	86
3	87	78	89	84	84	90	89	90	92
4	**94**	**93**	**92**	**95**	**94**	**93**	**95**	**94**	**93**

Table 7.12: Classification Accuracy (%) of the four different SCAE-MLELMs architecture on different datasets with input format as **MFECs**; Brac University previous and self-built Bangla Speech dataset and Google Audio-Set and VoxCeleb for English speech dataset is used during the experiment. Numbers in bold represent the highest classification accuracy.

Model No.	**Bangla Speech**			**English Speech**					
	Brac University			**Google AudioSet**			**VoxCeleb**		
	Audio type	Dialect	Gender/Age	Audio type	Dialect	Gender/Age	Audio type	Dialect	Gender/Age
1	84	87	89	87	88	86	91	92	92
2	**95**	**94**	**94**	**97**	**96**	**94**	**95**	**95**	**93**
3	78	84	90	84	83	91	90	90	91
4	76	79	76	78	76	79	81	82	86

Table 7.13: Performance Results of exiting methods; Ribeiro [10]: Deep CNN, and Tursunov [8]; Multi-attention module CNN model for spectrogram data type

Class	Speech	**Ribeiro.A [10]**		**Tursunov.A [8]**		**SCAE-MLELMs**	
		AUC(%)	pAUC(%)	AUC(%)	pAUC(%)	AUC(%)	pAUC(%)
Audio Type	Bangla	67.57	55.24	78.16	64.24	91.24	87.12
	English	87.45	73.15	89.78	82.57	92.75	86.74
Dialect	Bangla	52.47	42.18	61.41	59.78	89.75	83.12
	English	60.42	57.48	68.48	55.46	89.76	86.14
Gender	Bangla	69.40	55.94	84.15	77.42	92.00	87.45
	English	89.73	72.49	91.42	96.87	91.42	89.48
Age	Bangla	78.69	63.54	86.44	78.62	89.76	82.62
	English	83.45	73.44	81.46	77.48	89.41	86.42

Table 7.14: Performance Results of exiting methods; Ribeiro [10]: Deep CNN, and Tursunov [8]; Multi-attention module CNN model for MFECs data type

Class	Speech	**Ribeiro.A [10]**		**Tursunov.A [8]**		**SCAE-MLELMs**	
		AUC(%)	pAUC(%)	AUC(%)	pAUC(%)	AUC(%)	pAUC(%)
Audio Type	Bangla	76.15	65.73	87.41	78.24	92.11	89.12
	English	79.25	64.65	89.88	72.11	93.70	84.97
Dialect	Bangla	66.48	54.18	76.48	62.18	92.48	93.17
	English	68.42	56.38	81.42	75.40	90.45	82.75
Gender	Bangla	72.11	65.19	86.12	73.14	91.75	83.46
	English	89.73	72.14	88.25	79.48	91.10	80.74
Age	Bangla	83.45	72.38	83.47	78.34	90.29	89.26
	English	85.16	76.34	80.42	79.28	88.49	86.77

Chapter 8

Conclusion

In this paper, a dataset was prepared with seven regional Bangla language speech and stacked convolution autoencoder followed by multi-label extreme learning machines model for classification of synthesized voices and regional Bangla languages using MFECs data format was proposed. The model, is able to extract essential features and classify unsupervised data (new bangla/english abbrevation word phase that was previously not trained upon) appropriately. The SCAE identifies relevant features required for the class label and produces detailed feature maps from the given input data. While the MLELM networks in the suggest method learns from the training data to produce multi-label classification in one-pass. Two MLELM networks was used because the first performs soft classification scores and soft labels. While the later MLELM network matches the soft label to hard labels. To evaluate the performance, efficiency and robustness of the system extensive training and testing was performed. The suggested method outperforms the existing algorithms (Ribeiro [10], a Deep CNN model, and Tursunov [8], a multi-attention module CNN model) with an accuracy score of 91%, 89%, 89%, 92% for synthesised/ original audio type, dialect, age, and gender classification, respectively for Bangla Speech, for the spectrogram input data type. While for MFECs input format the accuracy scores are synthesised/original audio type, 92%, dialect, 92%, age 90%, gender 91%. As a result, MFECs data input format are more reliable when tasked to recognize relevant salient feature from audio inputs. The proposed model is also able to improve the classification accuracy score for dialect class to 95%, by using the detailed feature maps produced from the SCAE, that produces the correlated acoustic features patterns between age and dialect class. As aging have a physiological change that impact the processing of auditory feedback in the brain. Hence with the help of MLELM networks the multi-label data was used to created correlated feature maps of the data. The model also achieves highest accuracy score against the existing models for English speech dataset. 93%, 94% 88% and 91% for synthesised/original audio type, dialect, age, gender classification, respectively, for MFECs. The proposed method can be applied to the concept of ASR, TTS and speech recognition and processing tasks, like customer care, health care devices and many more in the future.

8.1 Limitation and Future Work

There are many additional research topics that could be explored, for instance;

- Raw-audio classifiers: The translation of audio into spectrograms formed the basis for the majority of our deep learning research. However, as demonstrated in articles such as Wavenet [40], it is feasible to directly feed raw audio into neural networks without first converting it to spectrograms. This was extensively researched for speech synthesis. However, to the best of our knowledge, raw-audio was never employed as a classifier for the synthetic speech recognition challenge. This might improve classification accuracy while decreasing pre-processing time (since spectrograms are not needed).

- Regular Convolutional Networks: Regular convolutional networks were used in the experiments in this study (squared convolution filters). However, because the spectrogram is a sequence of frequency-magnitude measurements, a classifier based on temporal convolutional networks [34] might be used to classify the data.

- In our research, we developed distinct algorithms for recognizing "original" synthetic speech and rerecorded synthetic speech. One intriguing research issue would be to develop a unified model capable of correctly detecting a synthetic utterance regardless of whether it is re-recorded or not.

- A heterogeneous rerecorded dataset: We only used one type of speaker and microphone to rerecord our utterances. Using a wide range of recording/playback devices in a wide range of recording rooms would be a fascinating experiment. This would result in a more diverse rerecorded dataset and a more broad synthetic speech detection model.

- An in-depth investigation at the differences in accuracy between audio representations: As we discovered in our tests, some audio representations work well in one experiment but not in another. Understanding the cause of this occurrence, as well as whether or not audio compression has a part in this fluctuation, would be a fascinating study issue.

- Since we created models for detecting synthetic voice, one may construct an application (such as a browser plugin) that can identify if synthetic audio is being played in a web page. This would assist the community by informing individuals whether the audio they are listening to is synthetic or genuine, minimizing the possibility of successful impersonation assaults.

Bibliography

[1] F. Alam, S. M. Habib, D. A. Sultana, and M. Khan, “Development of annotated Bangla speech corpora”, Project: Bangla Language Processing, vol 9, pp. 125-132, 2010.

[2] A. Nagrani, J. S. Chung, W. Xie, A. Zisserman, Voxceleb: Large-scale speaker verification in the wild, Dataset: Computer Science and Language, 2019

[3] P.-E. Honnet, A. Lazaridis, P. N. Garner, and J. Yamagishi, “The siwisfrench speech synthesis database - design and recording of a high quality french database for speech synthesis”, Journal Idiap, Tech. Rep., 2017.

[4] Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc Le, Yannis Agiomyrgiannakis, Rob Clark, Rif A. Saurous, "Tacotron: Towards End-to End Speech Synthesis, Book: INTERSPEECH, 2017.

[5] Boyang Zhang, Jared Leitner, and Sam Thornton, "Audio Recognition using Mel Spectrograms and Convolution Neural Networks".

[6] Anvarjon Tursunov, Mustaqeem, Joon Yeon Choeh, and Soonil Kwon, "Age and Gender Recognition Using a Convolutional Neural Network with a Specially Designed Multi-Attention Module through Speech Spectrograms", Journal MDPI sensors, 2021, https://doi.org/10.3390/s21175892

[7] Orken Mamyrbayev, Alymzhan Toleu, Gulmira Tolegen Nurbapa Mekebayev| (2020) "Neural architectures for gender detection and speaker identification, Cogent Engineering", Journal Cogent Engineering, vol no.7:1, pp-1727168, DOI: 10.1080/23311916.2020.1727168.

[8] Luis Miguei Matos, Pedro Jose Pereira, Andre Ferrelra, Puulo Cortez, "Deep Dense and Convolutional Autoencoders for Unsupervised Anomaly Detection in Machine Condition Sounds", 2020, Project: EASY RIDE PROJECT: Intelligent Mobility.

[9] Ye Jia, Yu zhang, Ron J. Weiss, Quan Wang, Jonathan Shen, Fei Ren, Zhifeng Chen, Patrick Nguyen, Ruoming Pang, Ignacio Lopez Moreno, Yonghui Wu, "Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis", 2 Jan 2019.

[10] "MIT Deep Learning Genomics - Lecture11 - PCA, t-SNE, autoencoder embedings", 2020, Youtube, Manolis Kellis, https://www.youtube.com/watch?v=Qh6cAXJJxd4

[11] T. Islam Pial, S. Salim Aunti, S. Ahmed and H. Heickal, "End-to-End Speech Synthesis for Bangla with Text Normalization,", CSII, 2018, pp. 66-71, doi: 10.1109/CSII.2018.00019.

[12] Turchenko, Volodymyr Luczak, Artur. "Creation of a deep convolutional auto-encoder in Caffe", Conference: IDAACS, 2017, 651-659. 10.1109/IDAACS.2017.8095172, https://github.com/NervanaSystems//examples/autoencoder.py

[13] G. Sharma, K. Umapathy, and S. Krishnan, “Trends in audio signal feature extraction methods”, Journal Applied Acoustics, vol.158, p. 107020, 2020.

[14] Nervana Systems/Neon, Convolutional autoencoder example network for MNIST data set., 2015, https://github.com/NervanaSystems//examples/autoencoder.py

[15] Seyfio□lu, M. S., □zbayo□lu, A. M., G□rb□z, S. Z. (2018). "Deep convolutional autoencoder for radar-based classification of similar aided and unaided human activities.", IEEE Transactions on Aerospace and Electronic Systems, 54, (4), 1709–1723.

[16] Xifeng Guo, Xinwang Liu, En Zhu, and Jianping Yin. "Deep Clustering with Convolutional Autoencoders". Lecture Notes in Computer Science, (including subseries Lecture Notes in Artificial Intelligence and Lecture, Notes in Bioinformatics), 10635 LNCS:373–382, 2017.

[17] Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong, Cai, and Heng Huang. "Deep Clustering via Joint Convolutional Autoencoder Embedding and Relative Entropy Minimization". Proceedings of the IEEE International Conference on Computer Vision, 2017-October:5747–5756,

[18] Berniker, Max Kording, Konrad. (2015). "Deep networks for motor control functions. Frontiers in computational neuroscience", Journal Frontiers in Computational Neuroscience, Vol no. 9. 2015.8

[19] Hou, W.; Dong, Y.; Zhuang, B.; Yang, L.; Shi, J.; Shinozaki, T. "Large-Scale End-to-End Multilingual Speech Recognition and Language Identification with Multi-Task Learning.", In Proceedings of the INTERSPEECH 2020, Shanghai, China, 25–29 October2020; pp. 1037–1041.

[20] Sajjad, M.; Kwon, S. "Clustering-based speech emotion recognition by incorporating learned features and deep BiLSTM", Journal IEEEAccess 2020, 8, 79861–79875.

[21] Law, A., Ghosh, A., (2019). "Multi-label classification using a cascade of stacked autoencoder and extreme learning machines.", Journal Neurocomputing, 358, 222–234.

[22] Rahman, S., Kabir, F., Huda, M. N., (2016). "Automatic gender identification system for Bengali speech.", EICT 2015, 549–553.

[23] Hassan, F., Khan, M. S. A., Kotwal, M. R. A., Huda, M. N., (2012). "Gender independent Bangla automatic speech recognition.", ICIEV 2012, 144–148.

[24] Sharmin, R., Rahut, S. K., Huq, M. R, 2020, "Bengali Spoken Digit Classification: A Deep Learning Approach Using Convolutional Neural Network.", Journal Procedia Computer Science, 171, 1381–1388.

[25] Liu H, Russo N.M, Larson C.R., "Age-related differences in vocal responses to pitch feedback perturbations: a preliminary study.", Journal Acoust Soc Am., 2010 Feb, 127(2):1042-6. doi: 10.1121/1.3273880. PMID: 20136225; PMCID: PMC2830265.

[26] Mridha, M.F., Ohi, A.Q., Hamid, M.A. et al. "A study on the challenges and opportunities of speech recognition for Bengali language", Artificial Intelligence Review, 2021.

[27] Gutkin, A., Ha, L., Jansche, M., Pipatsrisawat, K., Sproat, R. (n.d.). "TTS for Low Resource Languages: A Bangla Synthesizer", 2016 - 10^{th} International Conference on Language Resources and Evaluation, pp.2005-2010.

[28] F. Y. Sadeque, S. Yasar and M. M. Islam, "Bangla text to speech conversion: A syllabic unit selection approach,", 2013 International Conference on Informatics, Electronics and Vision (ICIEV), 2013, pp. 1-6, doi: 10.1109/ICIEV.2013.6572593.

[29] Firoj Alam and Promila Kanti Nath and Mumit Khan, "Text to speech for Bangla language using festival", 2010, Project: Bangla Language Processing.

[30] G. Muhammad, Y. A. Alotaibi and M. N. Huda, "Automatic speech recognition for Bangla digits," 2009 12th International Conference on Computers and Information Technology, 2009, pp. 379-383, doi: 10.1109/ICCIT.2009.5407267.

[31] M. Asfak-Ur-Rahman, M. R. A. Kotwal, F. Hassan, S. Ahmmed and M. N. Huda, "Gender effect cannonicalization for Bangla ASR,", 2012 15th International Conference on Computer and Information Technology (ICCIT), 2012, pp. 179-184, doi: 10.1109/ICCITechn.2012.6509701.

[32] Gutkin, A., Ha, L., Jansche, M., Kjartansson, O., Pipatsrisawat, K., Sproat, R., 2016, "Building Statistical Parametric Multi-speaker Synthesis for Bangladeshi Bangla.", Journal Procedia Computer Science, 81, 194–200. https://doi.org/10.1016/j.procs.2016.04.049

[33] Rahut, Shantanu Sharmin, Riffat Tabassum, Ridma, 2020, "Bengali Abusive Speech Classification: A Transfer Learning Approach Using VGG-16", Conference: 2020 Emerging Technology in Computing, Communication and Electronics (ETCCE), Dhaka, 10.1109/ETCCE51779.2020.9350919.

[34] Badhon, S M Nobel, Md. Habibur Rupon, Farea Abujar, Sheikh., 2021, "Bengali Accent Classification from Speech Using Different Machine Learning and Deep Learning Techniques", Book: Soft Computing Techniques and Applications, pp.503-513, 10.1007/978-981-15-7394-1-46.

[35] Alam, T., Khan, A., Alam, F. (n.d.)., "Bangla Text Classification using Transformers", 2000, Project: Bangla Language Processing

[36] M. M. Jam and H. Sadjedi, “Identification of hearing disorderly multi-band entropy cepstrum extraction from infant’s cry”, International Conference on Biomedical and Pharmaceutical Engineering, 2009, pp. 1–5.

[37] Amit Kumar Das and Abdullah Al Asif and Anik Paul and Md. Nur Hossain, "Bangla hate speech detection on social media using attention-based recurrent neural network", Journal of Intelligent Systems, No.1, Vol.30, 2021, pp 578-591, doi:10.1515/jisys-2020-0060, doi:10.1515/jisys-2020-0060

[38] A. Torfi, S. M. Iranmanesh, N.M. Nasrabadi, and J. M. Dawson, “3d convolutional neural networks forcross audio-visual matching recognition”, IEEE Access, vol.5, pp.22 081–22 091, 2017

[39] G.B.Huang, Q.Y.Zhu, C.K.Siew, "Extreme learning machine: theory and applications", Journal Neurocomputing, 70 (1–3), 2006, pp489–501.

[40] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Ko-ray Kavukcuoglu. "WaveNet:A Generative Model for Raw Audio.", September 2016.

[41] J-A Gomez-Garcia, L Mow-Velazquez, J-L Godino-Llorente and G Castellanos-Dominguez, "Automatic age detection in normal and pathological voice".

[42] Md. Rezaul Karim, Bharathi Raja Chakravarthi, John P. McCrae and Michael Cochez, "Classification Benchmarks for Under-resourced Bengali Language based on Multichannel Convolutional-LSTM Network", , 2020.

[43] S. Mavaddati, "Voice-based Age and Gender Recognition Based on Learning Generative Sparse Models", International Journal of Engineering, September 2018, ISSN: 24237167.

[44] Wei Ping, Kainan Peng, Andrew Gibiansky, Sercan O. Arik, Ajay Kannan,Sharan Narang, Jonathan Raiman, and John Miller. "Deep Voice 3: ScalingText-to-Speech with Convolutional Sequence Learning.", Book International Conference on Learning Representation, October 2017

[45] Sercan O. Arik, Jitong Chen, Kainan Peng, Wei Ping, and Yanqi Zhou. "Neural Voice Cloning with a Few Samples.", Curran Associates, Inc, Book Advances in Neural Information Processing Systems, February 2018.vol:31

[46] https://www.home-assistant.io/components/tts.baidu/

[47] https://azure.microsoft.com/en-ca/services/cognitive-services/text-to-speech/

[48] https://aws.amazon.com/polly/ view on Jan 2022

[49] https://cloud.google.com/text-to-speech/ viewed on Dec 2021

[50] K. Mannepalli, P. N. Sastry, and V. Rajesh, “Accent detection of telugu speech using prosodic and formant features,”, Conf. on Signal Processing and Communication Engineering Systems, 2015, pp. 318–322. doi:10.1109/SPACES.2015.7058274.

[51] S. Zhang and Y. Qin, “Semi-supervised accent detection and modeling,”, IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 7175–7179. doi:10.1109/ICASSP.2013.6639055.

[52] J. Hansen and L. Arslan, “Foreign accent classification using source generator based prosodic features,” , Conf. Acoustics, Speech, and Signal Processing, vol. 1, 1995, 836–839 vol.1. doi: 10 . 1109 /ICASSP.1995.479824.

[53] C. Teixeira, I. Trancoso, and A. Serralheiro, “Accent identification,” , ICSLP ’96, vol. 3, 1996, 1784–1787 vol.3. doi: 10.1109/ICSLP.1996.607975.

[54] L. W. Kat and P. Fung, “Fast accent identification and accented speech recognition,” IEEE ICASSP99 (Cat. No.99CH36258), vol. 1, 1999, 221–224 vol.1.

[55] T. Chen, C. Huang, E. Chang, and J. Wang, “Automatic accent identification using gaussian mixture models,” in IEEE Automatic Speech Recognition and Understanding, 2001. ASRU ’01., pp. 343–346.

[56] F. Weninger, Y. Sun, J. Park, D. Willett, and P. Zhan, “Deep Learning Based Mandarin Accent Identification for Accent Robust ASR,” in Proc. Interspeech 2019, pp. 510–514.

[57] Y. Zheng, R. Sproat, L. Gu, I. Shafran, H. Zhou, Y. Su, D. Jurafsky, R. Starr, and S.-Y. Yoon, “Accent detection and speech recognition for shanghai accented mandarin.,” Jan. 2005, pp. 217–220.

[58] C. Huang, T. Chen, and E. Chang, “Accent issues in large vocabulary continuous speech recognition: Special double issue on chinese spoken language technology,” International Journal of Speech Technology, vol. 7, Jan. 2004.

[59] NgA (2019),Neural Networks and DeepLearning, Coursera., https://www.coursera.org/learn/neural-networks-deep-learning

[60] Ng A (2018), What is Machine Learning? - Introduction, Coursera., https://www.coursera.org/lecture/machine-learning/what-is-machine-learning-Ujm7v

[61] NgA, KatanforooshK (2018), CS230 DeepLearning, http://cs230.stanford.edu/

[62] Bishop CM (2006), Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, Berlin

[63] Bengio Y (2009), Learning Deep Architectures for AI., Technical report

[64] Jones N (2014), Computer science: The learning machines. Nature 505: 146–148

[65] Karn U (2016) An Intuitive Explanation of Convolutional Neural Networks., https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

[66] Karpathy A, Li F F, Johnson J (2016), Stanford University CS231n: Convolutional Neural Networks for Visual Recognition., http://cs231n.stanford.edu/2016/

[67] Talwalkar A (2016) CS190.1x | Scalable Machine Learning., https://courses.edx.org/courses/BerkeleyX/CS190.1x/1T2015/576601d4282341f99ac7956718cc2301/

[68] Chollet F (2017), Deep learning with Python

[69] Shammur Absar Chowdhury, "Implementation of speech recognition system for Bangla Empathy and Affective Scene in Conversation View project Bangla Language Processing View project",