Silent voice:
Harnessing Deep Learning for Lip-Reading in Bangla

by

Munia Shaheen
23241102
Akib Zabed Ifti
23341129
Ariful Hassan
20301259
Junaed Hossain
23241107

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

Department of Computer Science and Engineering
Brac University
January 2024

Declaration

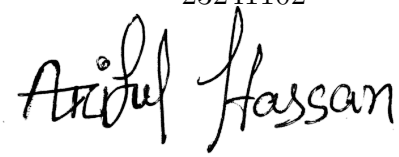
It is hereby declared that

1. The thesis submitted by 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:

Munia Shaheen
23241102

Akib Zabed Ifti
23341129



Ariful Hassan
20301259

Junaed Hossain
23241107

Approval

The thesis project titled "Silent voice: Harnessing Deep Learning for Lip-Reading in Bangla" submitted by

1. Munia Shaheen (23241102)
2. Akib Zabed Ifti (23341129)
3. Ariful Hassan (20301259)
4. Junaed Hossain (23241107)

Of Fall, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 09, 2024.

Examining Committee:

Supervisor:
(Member)

Dr. Muhammad Iqbal Hossain
Associate Professor
Department of Computer Science and Engineering
BRAC University

Co-Supervisor:

Rafeed Rahman
Lecturer
Department of Computer Science and Engineering
BRAC University

Program Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:
(Chair)

Sadia Hamid Kazi
Associate Professor
Department of Computer Science and Engineering
BRAC University

Abstract

Understanding speech just through lip movement is known as lipreading. It is a crucial component of interpersonal interactions. The majority of the previous initiatives attempted to address the English lipreading issue. However, our goal is to build up a deep neural network for the Bangla language that can produce comprehensible speech from silent videos just by capturing the speaker's lip movements. Despite the fact that there is research on this topic in various languages, Bangla does not currently have a study or a suitable corpus to conduct research. Therefore, our research aims to investigate the performances of the state-of-the-art deep learning models on the Bangla corpus and build our own deep learning model suitable for Bangla lip-reading.

Keywords: Lipreading. Deep learning, Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Lip feature extraction, Lip region localization.

Nomenclature

LR: Lip Reading

BLR: Bangla Lip Reading

DL: Deep Learning

CNN: Convolutional Neural Network

RNN: Recurrent Neural Network

ASR: Automatic Speech Recognition

VSR: Visual Speech Recognition

BSR: Bangla Speech Recognition

BVLDB: Bangla Visual Lipreading Dataset

BLSD: Bangla Lipreading Speech Dataset

WER: Word Error Rate

CER: Character Error Rate

Accuracy: Accuracy of lip reading predictions

F1-score: F1-score for lipreading accuracy

Acknowledgement

Firstly, all praise to the Great Allah for whom our thesis has been completed without any major interruption. Secondly, to our supervisor Dr. Mohammad Iqbal Hossain, and co-supervisor Mr. Rafeed Rahman for their constant support and advice in our work. And finally, all the people who contributed to our dataset and gave us their consent to use their videos. Without their support, our research would not be possible.

Table of Contents

List of Figures

Chapter 1

Introduction

Lipreading is the technique of comprehending what someone is saying only by their lip movement. Humans have long been able to read lips manually. However, automating lipreading is a significant goal since human lipreading performance is weak and constrained. Initially, systems for reading lips were created using traditional machine-learning techniques. Deep learning applications' prominence, particularly in recent years, has caused this topic to be studied more than in the past. The main components of automated lip-reading include face recognition, lip localization, feature extraction, classifier training with corpus, and word/sentence recognition by lip movement.

Nevertheless, due to challenges in the extraction of spatiotemporal features, generalization across speakers, and environmental noise, lip-reading is an extremely difficult task. Additionally, there are a lot of homophones, which implies that the same lip movement can produce a variety of different characters. Furthermore, some of the phonemes are created inside the throat and mouth and cannot be identified by simply observing a speaker's lips. Also, lip-reading from video without audio or text data depends on a variety of factors, including lighting, recording distance, and the speaker's speaking style. The endeavor becomes more challenging by external noises, mumbling sounds, and guttural sounds. Therefore, lipreading is a challenging task for both humans and machines.

Even though lip reading is a challenging technique to process, it is essential in a variety of fields, including accessibility, noise-sensitive communication, human-computer interaction, multimodal communication, security, and numerous more fields. Lip reading and voice recognition technologies can greatly improve accessibility for the deaf. Additionally, in noisy situations where speech could be hard to hear, such as crowded public spaces or industrial settings, lip reading technology might improve communication. These technologies enable individuals to understand speech even in challenging acoustic environments, improving the effectiveness and precision of communication in commonplace applications. Recent years have seen an increase

in enthusiasm for studies on the security of merging lipreading with biometric data, such as biological fingerprints or facial recognition. Studies that use lip movement monitoring for individuals include printing messages to smartphones in loud circumstances using visual data instead of auditory data and implementing various security measures utilizing visual silent passwords [13], [18], [30]. The applications of lip-reading technology are mentioned below:

- Helping hearing-impaired people.
- Transcribing old silent movies or videos.
- Facilitating the alert for public safety.
- Improving audio in existing videos.
- Enabling video conversation in noisy or silent places, like libraries.
- Identifying utilizing biometrics.
- Synthesizing simultaneous voice from multiple speakers.
- Enhancing automatic speech recognition's overall
- Dictating information or messages into a phone while surrounded

1.1 Problem Statement

In today's world lip-reading is needed in various fields and both humans and machines are able to do some sort of lip reading. However, human lipreading performance is poor and limited. For example, studies showed that people with hearing impairments only attain an accuracy of 17±12% for a small subset of 30 monosyllabic words and 21±11% for 30 complex words. Thus, automating lipreading is a key objective. While classical machine learning techniques can be used to read lips, they are slower and less accurate than deep learning techniques. These days, with the use of deep learning, it is liable to translate lip motions into relevant words.
Even though there has been little research on lipreading in English, currently, there is no known research on lipreading in the Bangla language. Even there is no known corpus in Bangla suitable for this task. There are 265 million native and non-native speakers of the Bangla language worldwide, making it the fifth most widely used language. A significant majority of them are unable to speak English. Because of this, it will be challenging for them to use present technologies that require English. In our literature review, we will see most of the research used on English lip-reading becomes useless when trained on different language datasets. Therefore, research in lip reading in the Bangla language is crucial for the development of this field.

1.2 Research Contribution

In this research, we designed and created a corpus for Bangla language which can be used in further research or any kind of work which needs speaking Bangla. Research for lip reading in Bangla is very low and the available ones lack enough data or are not updated. We propose a polished dataset with different speakers with the same words and different speakers with different accents. Also, we proposed models which are more preferable for detecting Bangla language rather than using other language models. The main contribution of our work is described below:

1. Creating a dataset for Bangla language

   Creating a corpus that can be used in further research in the Bangla language. The dataset contains 20 Bangla words which have been spoken by 65 different speakers. Each word has been spoken 200 times.

2. Finding the suitable feature extraction method for the Bangla corpus.

   Enhancing the accuracy and efficiency of our Bangla lip reading required finding the right feature extraction method for the Bangla corpus. This contribution enhances the system's overall performance and establishes a solid basis for linguistic analysis, which increases the system's capacity to comprehend and interpret Bangla speech patterns.

3. Examining the results of the current best models used in the English corpus using our Bangla corpus.

   Previous English models, such as LipNet [11], Lip in the Wild [13], and others that primarily used English corpora, impacted our decision-making process. The Bangla corpus that we created was used for our lip reading. However, in comparison study, the existing models performance was not satisfactory.

4. Establishing the ideal lip-reading model for the Bangla language.

   Developing the best lip-reading model for the Bangla language required a lengthy testing and evaluation process. We applied numerous methodologies and systematically analyzed their efficacy, drawing on insights from past models such as LipNet and Lip in the Wild. The model that demonstrated the highest accuracy and efficacy in the context of Bangla lip reading was chosen as the foundation for our system, assuring optimal performance and relevance to the specific linguistic peculiarities of the Bangla language.

1.3 Thesis Structure

Our thesis starts with an introduction in which we stated the main purpose and contribution of our research. The next chapter is a complete "Literature Review," which covers academic works in English, Urdu, Chinese, Turkish, German, and Bangla. The "Proposed Model" chapter includes our working plan, dataset details, and a number of pre-processing stages, including face detection, video editing, and grayscale conversion. Multiple models including LipNet and Autoencoder-decoder and multiple custom models are also implemented and the performance of these models is critically examined in a subsequent "Analysis" chapter. Finally, we compare and contrast how effective they are and the last chapter of our thesis summarises our conclusions and future works.

Chapter 2

Literature Review

In this literature review, we will be discussing the previous research in this domain of lip-reading. Most of the research used English speakers in their corpus. However, there are few types of research in German, Urdu, Korean, Chinese, and Turkish which gave state-of-the-art performance in their own language datasets. In our literature review, we will review lip-reading in English, Urdu, Chinese, Turkish, and German. As there is no known research on lip-reading in Bangla, we will review a few papers on Bangla [1] voice recognition systems on the existing technologies.

2.1 Language: English

The first hand-segmented phone-based sentence-level lip-reading research [2] was released in 1997 and used Markov models (HMMs) in a small dataset. Later, using an HMM in conjunction with hand-engineered features in the IBM ViaVoice dataset, another researcher conducted the first sentence-level audiovisual speech recognition [3] . Additionally, the authors train an LDA-transformed version in an HMM/GMM system [14] using the GRID corpus, outperforming the prior state-of-the-art with an accuracy of 86.4%. However, in these automated methods, extraction of motion information and generalization across speakers are both regarded as a big issue [10] .

In automated lip reading, deep learning is needed as such tasks necessitate intensive preprocessing for an image or video frame extraction or other forms of manually created vision pipelines[5] [8]. However, deep learning is seldom used in lipreading and only performs classification, not sentence-level sequence prediction in most of the research.

The paper [11], is the first known sentence-level lipreading model that uses the GRID corpus [7] consisting of 1000 sentences of audio and video by 34 speakers.

In this paper, the DLib face detector, iBug face landmark predictor [9], and affine transformation were used for preprocessing. Here, they introduced a new architecture called, LipNet. In the LipNet architecture, the input is processed by 3 layers of STCNN (Spatiotemporal convolutional neural networks) which is followed by a spatial max-pooling layer. Two Bi-GRUs then follow the retrieved features. Then at each time step, a linear transformation is applied before performing a softmax over the vocabulary augmented by the CTC (Connectionist Temporal Classification) blank, followed by the CTC loss. The authors evaluated the performance of the LipNet model using the word error rate (WER) and character error rate (CER). The LipNet model is the first sentence-level lip reading that gives 95.2% accuracy at the sentence level outperforming a human lipreading baseline and exhibiting better performance than the word-level state-of-the-art in the GRID corpus. However, this paper has some limitations, the model only works for sentences consisting of 6 words and to improve their research they need larger datasets.

The paper [15], used the same GRID corpus [7] used in the paper LipNet [11]. To begin with, they trained an autoencoder of audio files. A CNN-LSTM architecture is connected to a decoder of a pre-trained autoencoder, while the main network uses a 7-layer 3D convolutional structure to extract spatiotemporal information from the input video sequence. The core lip reading network was trained with batches of 32, and data augmentation was carried out by either flipping images horizontally or adding reasonable amounts of Gaussian noise to the data. For evaluation, they compared their result with Vid2Speech [16] in terms of PESQ(Perceptual Evaluation of Speech Quality) [4], Corr2D, and STMI(Spectro Temporal Modulation Index) [6] and gave better results. This paper showed ways to improve the audio quality for better prediction with a 98% correlation. However, they only used the GRID dataset, which contained only 6 sentences with a small word probability, to evaluate their model. For future work, more train data needs to be gathered, speech reconstructions need to include emotions, and an end-to-end framework needs to be built.

Lip Reading in the wild [12] recognises the words being spoken by a talking human face, given only the video without the audio. The main problem they faced in this paper is collecting the database. They have collected the dataset from TV broadcasts and collected over a million word instances, spoken by over a thousand different people. They took the videos and converted it to frames for preprocessing. After that, using face detection they separated the lip region and using OCR Recognition they collected the subtitles from the video and then aligned them with the frames from the video. They have used CNN architecture for this paper. They worked on the VGG-M model and modified it for their paper. They used four

different models and two of them are 2d and the other two of them are 3d. The models are: EF, EF-3, MT, MT-3. They got the highest accuracy for the MT model which gives the top-1 accuracy for 65.4% and top-10 accuracy for 92.3%. Lastly, the preference tends to be working with audio data combined with video for future work.

Next, in the paper [31], the author introduces a novel method for text identification from a silent lip movement video where they used the GRID corpus [7]. Here, A different technique for data augmentation is applied to the dataset for each epoch: by randomly inserting photos from the dataset video and adding modest quantities of Gaussian noise, we can simulate real-world conditions. The auditory MFCC features are converted in the architecture to a variable-length sequence of video frames via the visual-to-audio feature architecture. Second, the text information is distinguished from the audio feature by the architecture of the audio feature to text. A 7-layer 3D convolutional network (CNN) is used to recover the spatial and temporal characteristics of the video stream. Their research revealed that the model suggested in this article is capable of 92.76% validation accuracy. This paper has the same limitations as the first two papers as they used GRID corpus.

This paper [22], proposes a deep-learning technique for speech enhancement. A wellestablished GRID [7] [22] and ChiME3 [22], [19] corpus have been used as datasets for this method. The authors used a speech enhancement framework to extract audio features from a noisy speech by Enhanced Visually-Derived Wiener Filter(EVWF). Then they enhanced this wiener filter by proposing LSTM-based filter bank estimation. After that, they worked on a regression model for lip reading where they used LSTM & MLP. From a video, they extracted audio and frames, and with the help of hamming window and object tracker, lip cropper they managed to get audio and video features successfully. With one frame the LSTM model MSE of 0.092. While, with 18 frames MSE of 0.058 has been achieved. Again, With one frame MLP-based lip reading model achieved MSE of 0.199 only and 0.209 MSE while working with eighteen frames. LSTM can safely find out audio features which are clean but vanilla neural network models can not accomplish this task as smoothly as LSTM. At low SNR the model can give a standard pl-score but at high SNR, it outperforms but still works better than conventional speech enhancement approaches. However, the result and accuracy could be much higher, a context-aware AV algorithm can be used also and the dataset is much smaller to train as they have selected fewer frames while training.

The research [27] conducted small research where the two steps of the traditional lipreading method are feature extraction and categorization. In this paper[27], they

did not mention the dataset, however, they used Speech recognition using a Convolutional Neural Network. This model is trained to operate on large amounts of multi-dimensional data. After that, Dynamic lip videos, CNN, HMM, and decoding are used to extract features and appropriate words from HMM models. To compare artificial neural networks for voice identification, CNN and HMM were utilized for visual feature extraction and frame-level features, respectively. With an accuracy rate of 80words from a series of photographs of the lip region. RNN has certain drawbacks like explosive issues and it cannot process for a long sequence, which is the main restriction of this paper. RNN training is a challenging task as well. Research could lead to a speaker-independent recognition system in the future.

In this paper [24], the authors worked only with the micro-content of the English language which is the alphabet. They built the dataset having 20 letters and more than 2700 recorded videos which are a maximum of 2 seconds long. They prepared the data by converting videos into frames, detecting human faces, detecting mouths from the faces, and selecting the keyframes. The model used in this paper is CNN and stochastic gradient descent (SGD) where CNN is used for recognizing letters and also extracting features. In this paper [24], VGG19 pre-trained CNN has been used containing sixteen convolution layers, five max-pooling layers, and more than two fully connected layers. Finally, after testing, models can predict those twenty alphabets 95% of the time for the testing set whereas an accuracy of 98% for the testing set. Letters that have similarities with other letters were conducted with an accuracy of less than 99% and the letters of distinguished features managed to have over 99% accuracy. The authors only worked with letters only, word or sentence prediction from videos was not their concern in this dataset. Also, the dataset is much smaller for training which consisted of just 20 alphabets but at the same time they worked with so many videos so this could be much more in number.

In contrast to earlier published approaches for lip reading that are based on sequence to-sequence architectures, a research paper [25] based on Deep Convolutional Neural Networks produced a hybrid lip-reading (HLR-Net) model from a video. In this paper GRID dataset [7] is used to conduct sentence-level lip reading. Here, HLR-Net was compared to LipNet, LCANet, and A-CTC models for unseen and overlapped speakers. The suggested model is composed of a CNN model, an attention layer, and a CTC layer. The proposed method is based on the attention deep learning model and converts a video of lip movement into frames using OpenCV before extracting the mouth component with the Dlip package. While the decoder employs attention, fully connected, activation function layers, the encoder makes use of inception, gradient, and bidirectional GRU layers. With a CER of 4.9

The authors of [28] attempted to compare all currently available deep learning architectures and additional techniques offered by other researchers in order to determine which could offer the best performance. The suggested research project uses the MIRACLVC1 [18] dataset, which contains 3000 occurrences of words and phrases, with 150 examples in each folder. 10 words and 10 phrases are spoken by five men and ten women in each piece of data and each word/phrase was spoken 10 times. The Dlib facial detector and Dlib mouth detector were used in this research to extract the image's mouth and identify the speaker's face. AlexNet, VGG16, Inception V1, ResNet50, Auto-Encoder, Q-ADBN, and EfficientNetBO are used to train the model. EfficientNetBO will serve as the model's foundation in this design, to which the Attention block and LSTM layer are added to produce a finer performance. The research had an accuracy of 80.133% using EfficentNetBO and 86.67% with EfficientNet and Attention Block. Finally, using LSTM on top of that gives an accuracy of 91.13%. For their future work, lip movement detection can be improved to include additional words and phrases from different speakers.

In this paper [30], the authors developed their model by extracting the mouth area and segmenting the area from a frame by using a hybrid model. For this, they have used the LRW dataset [21] from BBC TV broadcasts having audiovisual speech segments. Each video has 29 frames from which 22 frames have been selected. Their suggested method of segmentation has been split into two different stages. First, detecting the mouth and then applying an improved hybrid model. The models they used for lip reading are Bi-GRU which has two hidden layers and 2D CNN. First, they wrap the entire CNN input model in the TD function, then pass it into the Bi-GRU layer for extracting comprehensive information from previous layers' characteristics. Afterward, the outputs have been infiltrated into a pooling layer, and then all the outputs are connected into a softmax function to produce every word's possibilities. Moreover, by using ReLU activation functions two convolutional layers have been added. That's how they omit the problem of vanishing gradient. After 6 epochs loss approaches 0 in training data and 0.4 in testing data. On the other hand, with segmentation lip reading gives 90.385% accuracy and without segmentation, it gives 85% accuracy. So, their proposed architecture is doing great work on classifying lip motion sequences on words. However, in this paper, they have not worked with sentences. Also, the authors have not mentioned the sequence of the words

2.2 Language: Urdu

Few studies on lip reading in other languages have been published. The paper[20] shows lip-reading for the Urdu language that uses a corpus containing ten words. As there was no available corpus for Urdu lip-reading, the authors created a corpus of 10 words repeated 10 times by 10 participants. The recorded videos are preprocessed by cropping them to a standard size, applying the Viola-Jones face detector, and a mouth detector, and then cropping each video. To train the dataset, at first, they used LipNet [11], which failed to train due to CTC loss. Then they used RNN and LSTM to train the model on Urdu words and digits and got better accuracy from the LSTM model which is 62% and 72% respectively for words and digits. To show how effectively audio-visual lipreading performed in noisy conditions, the researchers in this study individually trained two different models for audio and video. However, they were unable to combine both networks. Additionally, they presented a small corpus of Urdu words which worked as a classification task with no sequence. Finally, they hope to create a model in the future that can predict text sequence and is akin to lipreading.

2.3 Language: Chinese

The first sentence-level Chinese lipreading was introduced in the paper [23]. In this study, they created their own dataset from CCTV News and Logic Show, labeling it with Hanyu Pinyin (a phonetic interpretation of Chinese), and end up with 349 classes and 1705 characters. They present a unique two-step network for sentence-level Mandarin lipreading where predicting the probability of Hanyu Pinyin is the first step. A max-pooling layer is applied after a 3D convolution that suggests a series of frames as input. Then, more visual features are extracted using DenseNet. The visual features are processed using a two-layer resBi-LSTM, which is subsequently accompanied by linear and softmax layers. The complete network is then trained using the CTC loss function. The second phase involves using a stack of multi-head attention to translate Hanyu Pinyin into Chinese characters. The suggested network has an absolute 13.91% and 14.99% rise in Hanyu Pinyin and Chinese character accuracy compared to LipNet [11], whereas ResNet [17] technique has an increase of 4.68% and 4.74%. The problem that remains is that for different words Hanyu Pinyin might be the same, but Chinese characters would be different due to the different contexts which indicates that Hanyu Pinyin is unable to capture context.

2.4 Language: Turkish

In the paper [29], two new datasets were created for the Turkish language, one with 111 words and the other with 113 sentences. In the paper[29], the Media-pipe framework determines the lip position and removes it from the image. Using a convolutional neural network (CNN), features are first retrieved to perform lip detection and then the classification process is completed using bidirectional long short-term memory(Bi-LSTM). The results of the experiment demonstrate that, for words and sentences, respectively, ResNet-18 and Bi-LSTM pair offer the best results, with accuracy scores of 84.5% and 88.55%. However, the limitation of the paper is that each speaker is positioned at 1.5 m in these photos, and the video clips taken to generate the dataset were captured in identical lighting and ambient conditions.

2.5 Language: German

In this paper[31], the researchers worked on the German language. They took the "Lip Reading in the Wild" dataset created by Chung Zisserman(2016). First of all, they used English for comparison and transfer learning. Then, they brought up the issue of copyright and data protection in Germany. The dataset consists of MPEG4 format videos which are 1.16 seconds long. In the processing part, they cropped the videos into 96*96 pixels only focusing on the lips of the speaker using the Glips model. As the videos have audio too, the video and audio files were stored in separate files and synchronized in the last step. For the subtitle part, they used WebMaus to synchronize with the audio. They did two experiments in this paper [19]. The validation accuracies of GLips and LRW in experiment 1 are respectively 78.4% and 77.8%. The validation accuracies of the transfer-learned models were 82.2% and 81.6% respectively. The LRW network's average validation accuracies in experiment 2 were 79.2% and 80.4%, respectively. The average validation accuracies of the GLips networks were 78.4% and 79.6%, respectively. The fact that learning was successfully transferred between the two languages suggests that there are aspects of lip reading in both datasets that are linguistically independent and can be applied to different languages. We assume that the difference between the models trained on GLips and LRW lies in the quality of the data because the LRW-trained networks perform better.

2.6 Language: Bangla

In the Bangla language, there are numerous words or symbols that sound remarkably similar. In this paper [26], they investigated the DeepSpeech network for recognizing individual Bengali speech samples. To overcome this, they used ASR which is an algorithm to detect individual characters, words, and sentences. In this paper [26], almost two thousand speech samples which are mostly Bengali real numbers, and a total of 115 unique words are distributed around the data samples. The authors first tried to approximate the phone alignment in individual speech samples using a statistical approach to build their input set. Then, they fed these input samples to a two-layer unidirectional LSTM network and trained their model. But the problem here is recognizing words instead of full sentences. At first, to remove all the noise and distortion, they used Fourier transform on all individual audio samples and extracted highly 13 distinctive nonlinear mel frequency cepstral coefficients (MFCCs) from them. Next, they used Bi-Directional LSTM with CTC loss function for training their data and tested a Bengali real word speech dataset which gives an 8.20% WER and a 3% CER. The limitation of the paper is that it only works on word-level detection, each data sample has eight examples in the dataset and the dataset is very small.

In this paper [27], the authors suggested a convolutional neural network (CNN) architecture for Bangla short speech detection. As the dataset for Bangla is not widely available, they made their own dataset for Bangla short speech commands. Their dataset contains 65,000 samples, considering one second for each word. They took Banglish as their data and then converted them to proper Bangla. They normalized their inputs before training their model with MFCC inputs. They then fed the raw audio data into a similar CNN architecture, which they call the raw model. Prior to training their data, they also pre-trained their model. They used three models for training and testing their dataset. The models are: MFCC, Raw, and Transfer and their training accuracies are respectively 85.44%, 69.08%, and 68.06%. Finally, their testing accuracies are respectively 74.01%, 71.44%, and 73%. In comparison to deep neural networks (DNN), they observed in their research that CNN architecture can lower the error rate on the TIMIT phone recognition dataset by 6-10%. Additionally, they conducted several tests utilizing the limited weight sharing (LWS) and full weight sharing (FWS) schemes and reported that LWS is more efficient since it can recognize feature patterns in many frequency bands. In this research, the main limitation was the dataset was relatively small. Finally, they proposed to expand the words in the dataset in the future so that they can find a better result from the used model MFCC.

In the research [25], they took the dataset from Google Bangla Speech Corpus which has a total duration of 229 hours with 508 native speakers and 50k unique words. Also, test Corpus: 10.6 million unique words and 602.5 million global words, 48.56 million sentences. In this paper[25], they claimed that their CTC-based Bangla ASR end-to-end deep CNN system surpasses the RNN-based DS2 system. For the first time in Bangla ASR research used deep learning techniques to analyze character-wise mistake rates in order to evaluate the corpus's quality. The CNN-based models, which frequently use acoustic characteristics in voice recognition and speaker identification, have been built using MFCC features. They used CTC-CNN and Deep Speech 2 models to train their data. The Deep Speech 2 model is an RNN -CTC-based model. The word error rate in the CNN-CTC model is 36.12% in the Deep Speech 2 model is 40.91%. The advantages of their model are they have a very large dataset, the error rate gets 10% when the characters are found more than 100k times, and characters that are used frequently and have better performance. However, there are some limitations too. Characters found less than 1k times, found 50% error. The characters found less than 1k times, found 50% error. The letter 'R' arrived only 7 times. In this situation, the DS2 model gets 71.4% accuracy. On the other hand, the CNN model could not recognize the word at all. The same pronounced words like 2% cannot be differentiated by this model. The 10 Bangla digits are not very often used in this dataset. So, it has less accuracy than others. In the future, they will work with a large high-order N-gram model to improve Bangla LVCSR.

Chapter 3

Proposed Model

3.1 Working Plan

Our study seeks to evaluate the effectiveness of the latest deep learning algorithms on the Bangla corpus and develop our own deep learning model appropriate for lip-reading Bangla. For this, at first, we need a Bangla corpus for lip-reading. As there is no known Bangla corpus for lip-reading, we will collect some Bangla videos suitable for our task and make our corpus. Secondly, we will use preprocessing to extract the features for our task. After that, we will split our dataset into train and test data. To train our data, we will use some of the models with the highest success used in lip-reading in other languages, in our own dataset. If those architectures give an accuracy of less than 90, then will try to build a unique model which might provide better accuracy for lip-reading in Bangla.

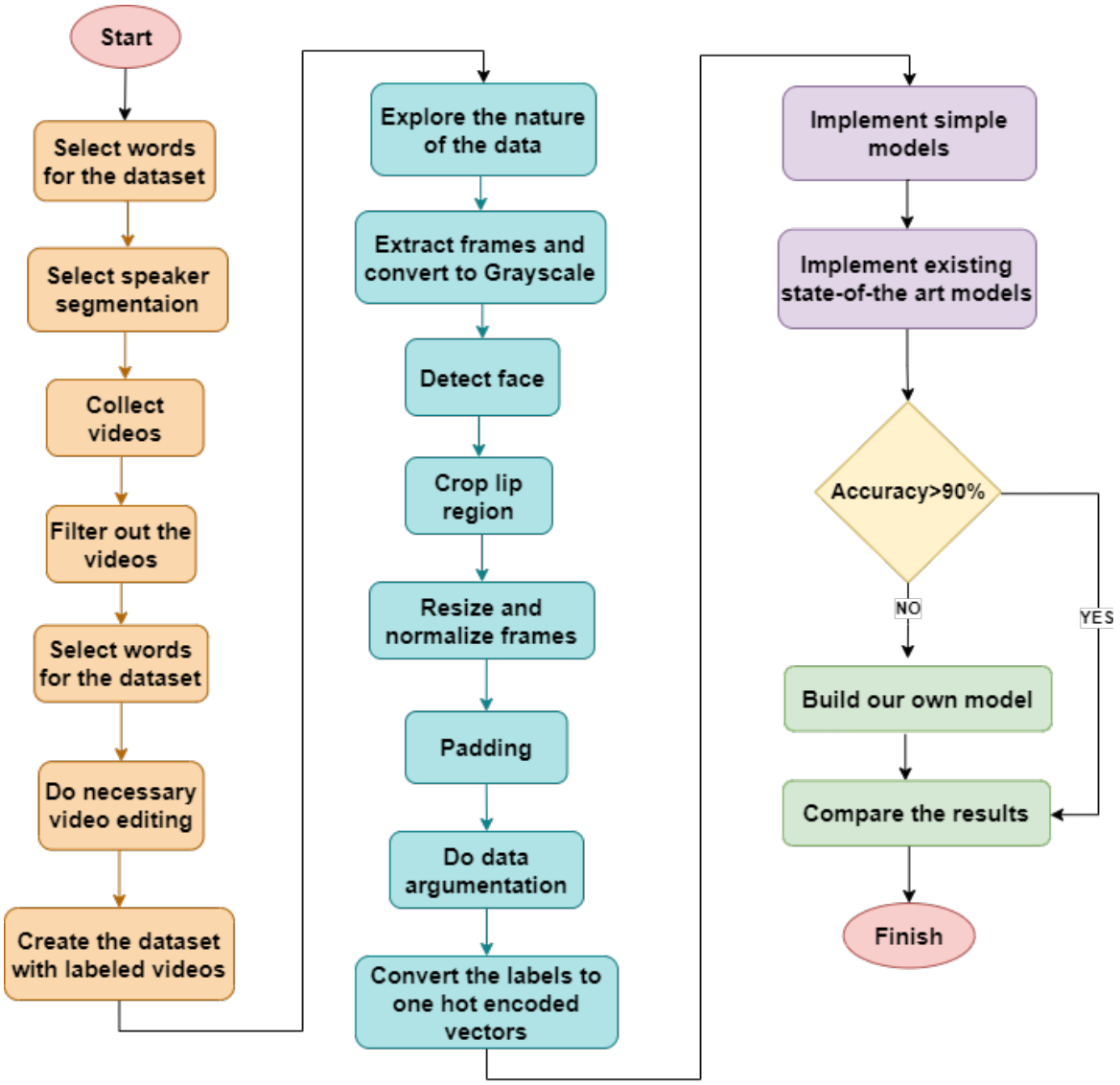


Figure 3.1: Working Plan

3.2 Dataset

3.2.1 Data collection

As there was no available video corpus for Bangla language lip reading, we have created our own corpus. The process of collecting video data from numerous individuals recording themselves speaking specific words typically involves several key steps. Firstly, a well-defined set of words or phrases is chosen for the participants to speak. Next, a diverse group of participants is recruited, ensuring variability in age, gender, and other relevant factors to capture a wide range of linguistic and vocal characteristics.

For our data collection, we have selected 20 Bangla words and these words were pronounced by 65 different speakers. The words are:

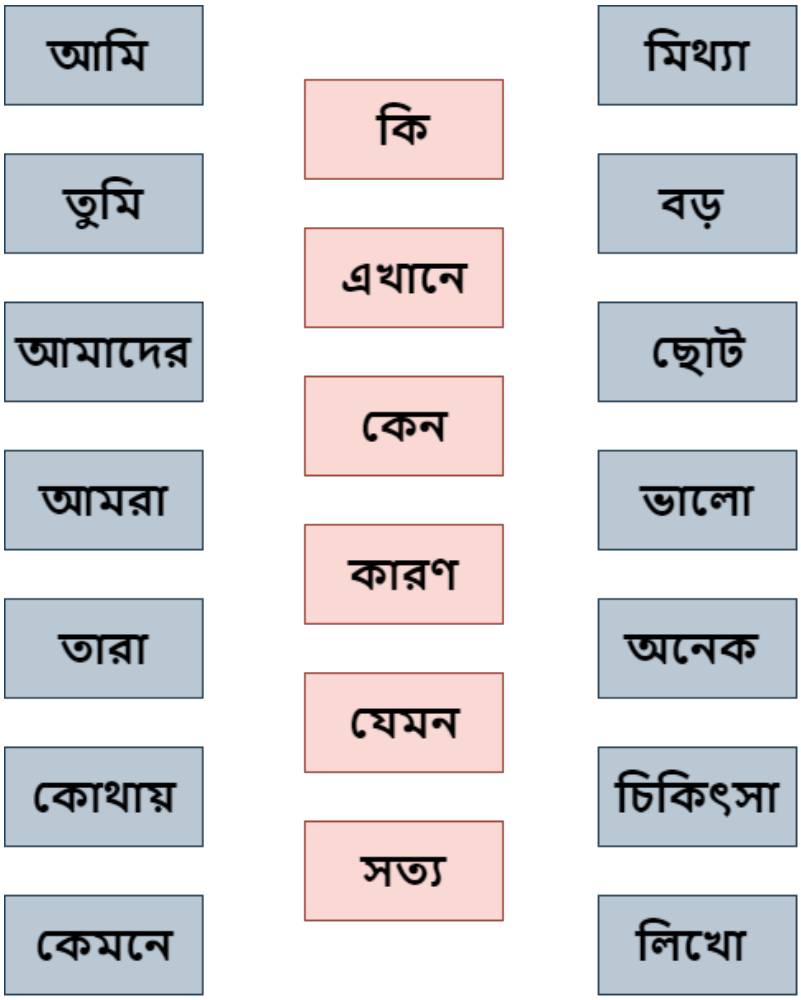


Figure 3.2: Selected words for dataset

Some of the Bangla words have the same lip movement like: “Je”, “She” and “Ke”, that is why selected words on a small scale which have distinct pronunciation and unique lip movements than the other one. To collect speakers pronouncing these 20 words is difficult. As in Bangladesh, there are numerous accents and pronunciations. Thus, we have selected the people of Dhaka as our target group of speakers. However, when we were recording these videos we saw that even the educated people born and raised in Dhaka pronounce the same word differently. Some words are pronounced differently like “amra” and “amora” by different speakers. This makes our prediction extremely hard.

Participants have been instructed clearly with a sample video guideline to record their videos. Also video resolutions have been specified for them to record their videos. Devices have been provided to record their video. Otherwise, they may use their own devices for recording and then provide them via email or google drive link. After completion of video collection, they have been quality checked and processed where the unusable which did not show face properly or the lighting is not suitable

to detect lip region have been filtered out. The remaining videos have been stored, organized and labeled in a structured manner to save them on a database for analysis. The dataset can serve different purposes such as research in linguistics, voice recognition technology and other kinds of research which require both video and audio or one of them. Privacy concerns have been measured and participants have consented to use the data for ethical and legal purposes.

3.2.2 Creating Dataset

After collecting the video files from the speakers, they have been sorted labeled by the work that is spoken by the user. Labeling and sorting the videos helps to categorize and identify the data efficiently.

Figure 3.3: Visual of dataset

3.2.3 Exploratory Data Analysis:

In the context of our research, data visualization played a vital role in providing a comprehensive understanding of the modular structure of our dataset. To demonstrate how the videos are distributed throughout the 20 distinct folders (each with a unique Bengali title) displayed in Figure 3.4 . In particular, we used a bar plot with its name. With each folder holding precisely 200 videos, this visualization demonstrated the distribution of our dataset in a simple and understandable manner.Bengali terms were used to label the folders, which not only enriched our dataset culturally but also demonstrated our dedication to maintaining linguistic variety in data gathering and annotation. This visualization acts as a key building block for our research, assisting with the first investigation and comprehension of the structure of our dataset, which is fundamental to subsequent analyses and insights drawn from this unique collection of videos.

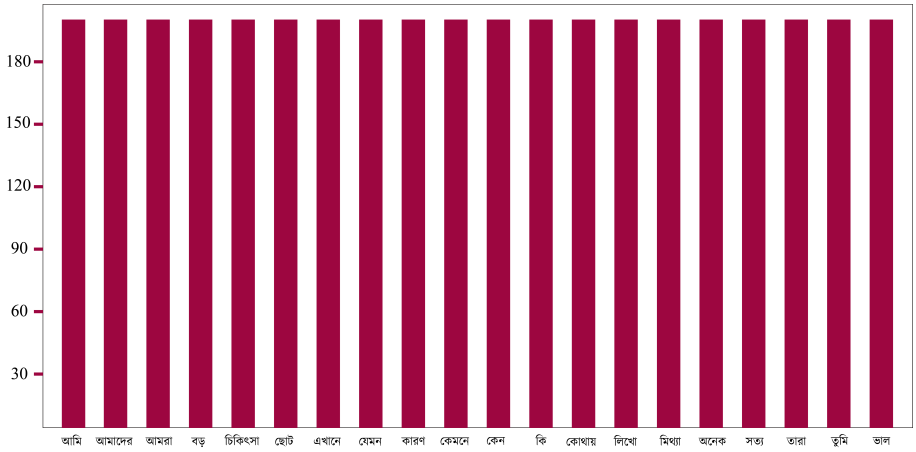


Figure 3.4: Number of videos(Balanced)

Here in Figure 3.4 we can see that all the folders are equally distributed with 200 videos. A sample has been shown in our dataset in 3.5. There we can see a bunch of pictures of different people which are a particular frame from their videos. There are 4000 video samples and the average length of those videos is 2 seconds shown in Figure 3.6.

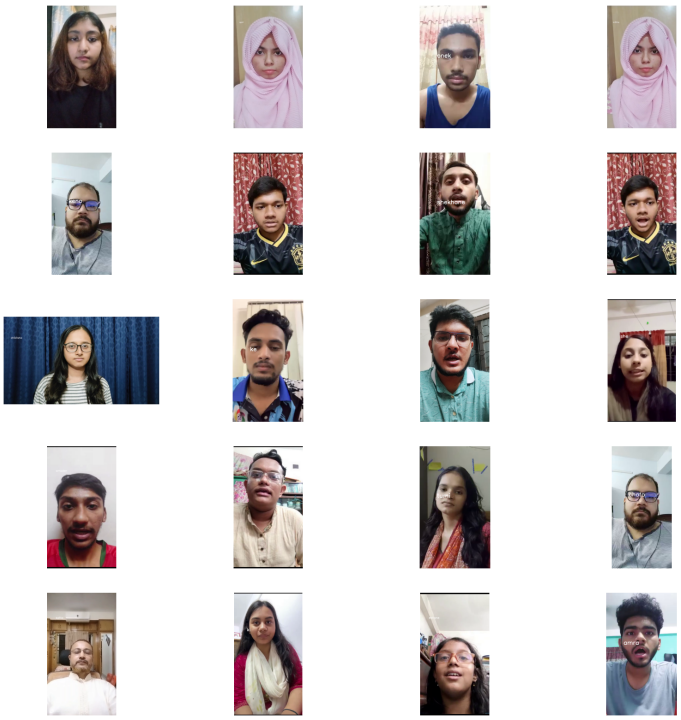


Figure 3.5: Sample frames of different videos

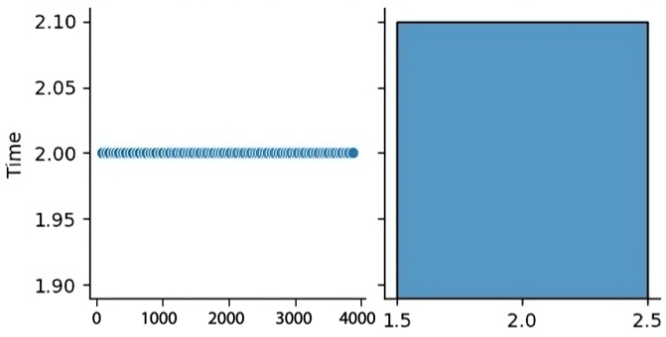


Figure 3.6: Average length of videos

3.3 Pre-Processing

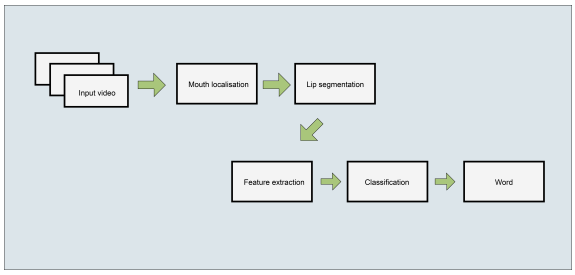


Figure 3.7: Lip-reading processs

3.3.1 Video editing

When collecting videos from speakers, it's not uncommon to encounter variations in lighting conditions and video file sizes. To address these challenges and ensure the quality and consistency of the dataset, video editing becomes a necessary part of the data preparation process. Videos captured under poor lighting conditions can result in dark, grainy footage that hinders visual clarity. In such cases, video editing software can be employed to adjust brightness, contrast, and color balance to enhance the overall visual quality. This ensures that all videos in the dataset are visually consistent and suitable for analysis or presentation.

However, we manipulated the pixel value in each frame and to do this we used some video editing softwares. Moreover, we managed the large size video by doing some actions like trimming and cutting unnecessary parts of the video, such as long pauses or redundant sections and irrelevant content of the video.

3.3.2 Extract Frames and Grayscale

Lipreading is the process of identifying spoken words and phrases by evaluating the movement and form of the speaker's lips over time.Frames can be extracted such that the video can be divided into discrete time steps or moments, with each frame corresponding to a different lip visual configuration. The dynamic of speech, phonemes, and lip motions must all be captured through this temporal segmentation.
Using a Python video processing library like OpenCV, we were able to turn a video into frames. First, we open the video file using the library's video capture function. Then, we read each frame sequentially from the video using a loop. Once we have a frame, we save it as an image file. The number of frames per second (fps) in the video determines the frame rate, which can be used to control how many frames we extract per second. By iterating through all the frames in this manner, we can

effectively convert the video into a sequence of individual image frames. In our case, we took a constant number of 30 frames for each video.
Along with converting videos into frames, we converted our videos into grayscale using the 'cv2.COLOR_BGR2GRAY'. Grayscale images contain only intensity information, as opposed to color images that contain red, green, and blue (RGB) channels. By converting to grayscale, we simplify the data and reduce computational complexity, making it easier and faster to detect faces.

3.3.3 Face Detection & Crop Lip Region

Lip reading focuses on the lip movement of a human face which requires only the lip region. Other than lip region - face, eyes, and expression are not required for lip reading. We removed all the unnecessary regions from the frames and cropped the lip region from the image. For this, we first detected the face and then cropped the lips.

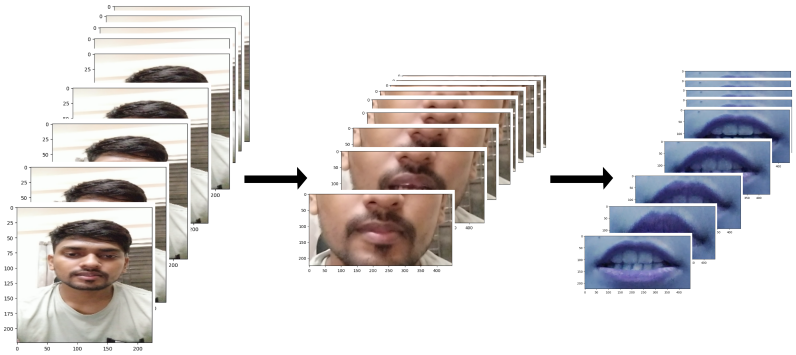


Figure 3.8: Lip segmentation

It begins by using the 'dlib' library to initialize a face landmark predictor and a frontal face detector. The 'face_detector', identifies the location of faces in the input frame.
Once a face is detected, the 'landmark_predictor' is employed to obtain the facial landmarks, which are specific points on the face, including those around the lips. In this case, points 48 to 59 correspond to the lip landmarks. These lip landmarks' coordinates are extracted and used to determine the position and size of the lip region within the face.
To crop the lip region effectively, we code to scale it inward by a factor specified as 'lip_region_scale'. This scaling factor allows adjustment of the size of the extracted lip region. The final 'lip_region' is created by using the computed coordinates (lip_x, lip_y) as the top-left corner and dimensions (lip_width, lip_height) for the region of interest within the face. This 'lip_region' contains the lips of the detected face.

3.3.4 Resize & Normalize Frames

The videos we have collected from the speaker are from different angles from the camera. Some have uplifted the camera, and some have captured a little from the side. As a result, the lip regions have different sizes in every video or in every frame. Also, every speaker was recorded from a different distance from the camera. This is also a big reason for the size of the frames not to be equal on everything. The cropped frames that have been collected are not equal in every frame or for every speaker. To solve this, all the images have been resized to a fixed height and width of 200,100.

When working with images, it's common to ensure that pixel values fall within a certain range so we used normalization. Normalization typically involves scaling pixel values so that they lie between 0 and 1. By dividing each pixel value for every frame in 255, we are rescaling the pixel values so that they are in the range [0, 1]. This is useful because many image processing and machine learning algorithms work better when pixel values are within this range.

3.3.5 Padding

Working with the videos of the speakers, every speaker has recorded their videos from different devices that have different frame rates. Having a different number of frames in the dataset can affect the model's accuracy. It's important to maintain consistency in frame counts for training and testing to achieve more reliable and accurate results. Thus, we have set 30-frame standards for our video inputs.

For this, we count the number of frames in each video. If the frame length is more than 30, then it calculates how many frames should be added at regular intervals to create sequences of the desired length then it will add frames at least one frame at a time to maintain consistency. However, if the number of frames for a video is less than 30, it pads with duplicate frames. At first, it calculates the number of frames needed to pad on both sides of the frames to reach the desired length. It then adds duplicate frames to the end and to the beginning to achieve the padding. The purpose of this padding is typically to ensure that all sequences or frames have the same length, which can be important when working with neural networks or other machine learning models that require fixed input sizes.

3.3.6 Data Augmentation

For complicated tasks like lip reading, we need an enormous amount of data. Thus we used data augmentation to create variations of the original video frames to enhance the diversity of the training dataset. For the augmentation part argumenta-

tion has been used which is a built-in library in python. To modify the lip images in a variety of ways to increase the training dataset's resilience and diversity, it is a great approach. However, for complex tasks like lip reading, where the number of visual inputs can enhance the model's capacity for generalization and the overall performance.

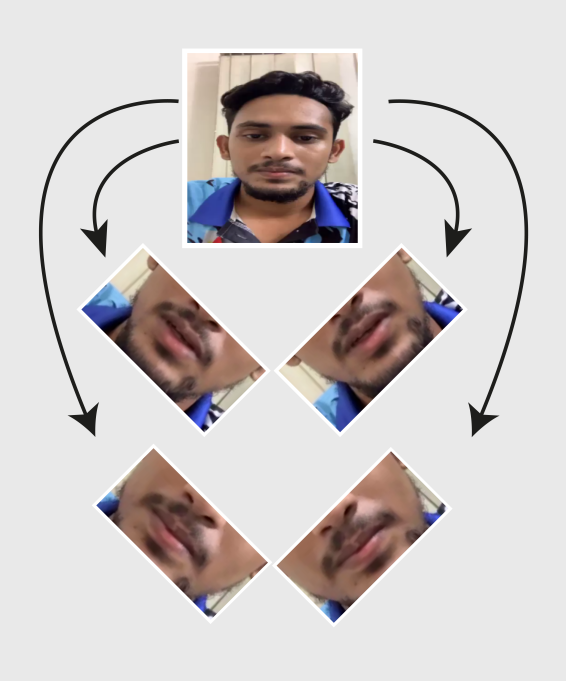


Figure 3.9: Rotated frame of cropped lip clip

In our research a customized augmentation process has been applied as the generated frames from all the videos were in grayscale format and after tuning the features augmentation has been enabled thoroughly. After that shifts in both the horizontal and vertical axes were applied to mimic different orientations, and also established a rotation range of 20 degrees to simulate different angles. Furthermore, providing the ability to flip the photographs horizontally, which broadens the range of viewpoints included in the training set. When these adjustments are used, the grayscale frames get subtle differences that successfully simulate various situations and viewpoints of the same subject. The number of training instances is effectively expanded by adding these modified copies to the original frames, which are essentially slightly altered replicas of the original frames. This method improves the training dataset's robustness and adaptability, which is important for tasks like lip reading that depend on subtle visual variations.

3.3.7 Convert the categorical values

In our dataset, the labels were the 20 words we selected for our classification. These labels were in categorical values and we converted categorical labels to one-hot encoded labels to make them suitable for training, ensuring compatibility, distinct class representation, and proper loss calculation.

3.4 Models

3.4.1 Dataset Implementation

Our dataset is a Bangla video corpus of different speakers pronouncing Bangla words. Our target is to predict what a person is saying based only on their lip movement. It is a video classification task. For this task, a CNN-RNN architecture is needed due to the multifaceted nature of video data. To implement this, we try our dataset on two state-of-the-art models that we found during our literature review and analyze its results.

3.4.2 Architecture

Convolutional Neural Network (CNN):
A Convolutional Neural Network (CNN) is a type of deep neural network designed to process data with a grid-like topology, such as images. CNNs are particularly adept at extracting spatial features from visual inputs, making them a staple in many computer vision applications, including image and video analysis.

Recurrent Neural Network (RNN):
A Recurrent Neural Network (RNN) is a type of artificial neural network in which node connections create a directed graph along a time sequence. This structure enables RNNs to demonstrate temporal dynamic behavior and sequences of inputs using their internal state. RNNs are meant to recognise patterns in data sequences such as text, speech, video, or time series, as opposed to typical neural networks, which presume that all inputs (and outputs) are independent of each other. An RNN's primary concept is to employ sequential information, retaining a type of internal memory that captures information about what has been analyzed . RNNs, in essence, have a type of memory that stores information about prior steps, making them excellent for jobs where we have to classify a sequence of images in our case these are the frames from videos.

CNN-RNN:

Convolutional neural networks known as CNN are used to extract spatial features from video frames, while recurrent neural networks known as RNN are used to model temporal correlations in video sequences. This type of neural network model is known as a CNN-RNN architecture. It efficiently captures both static and dynamic information in videos for use in video classification tasks. Applications like action identification, video summarization, and scene understanding, where capturing both static and dynamic data is required for successful categorization, this combination of spatial and temporal processing architecture is used.

CNN-RNN Working Procedure:

CNN:

A CNN is a structure made of several layers, each of which is designed to identify a distinct feature. Deeper layers are able to recognise complicated patterns, while the initial layers are usually responsible for identifying basic elements like edges and textures.

Convolutional layers compute the dot product between the filter values and the input pixels by moving very small filters systematically across the input picture (or video frame). Through this method, a feature map that highlights particular features is created. A non-linear activation function, usually Rectified Linear Unit (ReLU), is applied following convolution. This stage gives the model non-linearity, which enables it to recognise patterns that are hard to identify. Convolutional layers are followed by pooling layers, which aid in shrinking the feature maps' spatial extent.

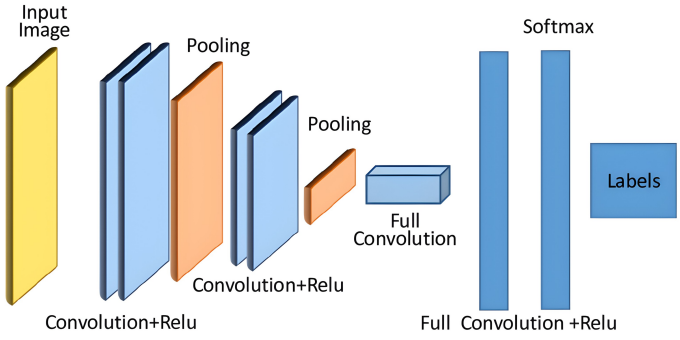


Figure 3.10: A generic CNN-RNN architecture

However, CNNs are capable of efficiently analyzing individual video frames in order to extract significant spatial data. These features could be particular textures, items, or shapes that are included in the frame.

RNN:

The basic structure of Recurrent Neural Networks (RNNs) is that the network's output is influenced by its internal memory of past inputs. The output of one step is transmitted back into the network as an input for the next stage in an RNN. Because of this sequential approach, the RNN takes into account both the current input and the knowledge it has gained from earlier inputs at each stage of the sequence. Because of this approach, RNNs can retain some sort of internal memory, which makes them very useful for involving sequential input.

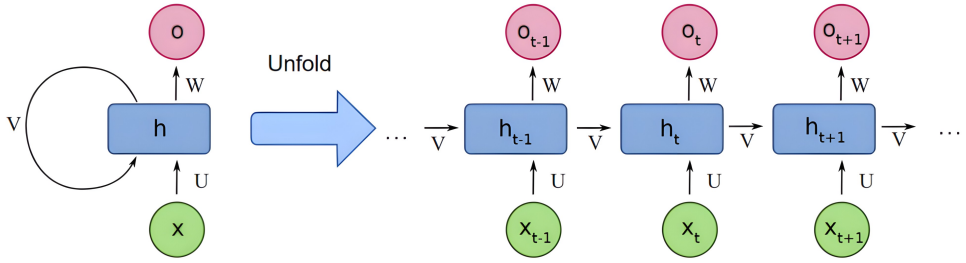


Figure 3.11: RNN Architecture

As we know video is a combination of images which are called frames, RNN can process these frames and extract information in a way that it can relate each and every information with the prior information and it is good at capturing and monitoring the temporal dependencies from the separate frames. As opposed to examining each frame independently, an RNN analyzes a video by taking into account all of the frames that came before it. An RNN can identify the movement pattern over these frames and effectively capture the temporal dependencies. Because the accuracy of the identification and classification of the video depends on the sequence and evolution of frames, this makes RNNs particularly appropriate for jobs involving the video classification.

CNN-RNN:

CNN-RNN is a great architecture for classifying videos and images. A CNN model is used to extract high-level features from video pictures. The features are subsequently passed through an RNN layer, and the RNN layer's output is sent to a fully connected layer to produce the classification result. Moreover, the CNN part evaluates the video frames and its features, different aspects from the frames which are present. Although the RNN part does something different here. It understands the sequence and time progression of the frames extracted from the videos. The first step is to assess the frames that are taken from the several videos that CNN has analyzed. It excels at picking up on small details. RNN, on the other hand,

remembers the prior frames and their relationship. This aids in its comprehension of the video's action. That is how the CNN-RNN architecture is able to comprehend the entire video in addition to seeing what's in each frame.

Firstly, to evaluate the frames which are extracted from the collected videos evaluated by CNN. It is great at noticing details like shapes, colors, and objects. Imagine CNN as a detective inspecting each frame to see what's there - a car, a tree, a person, etc. After the CNN has checked all the frames, the RNN part takes over.

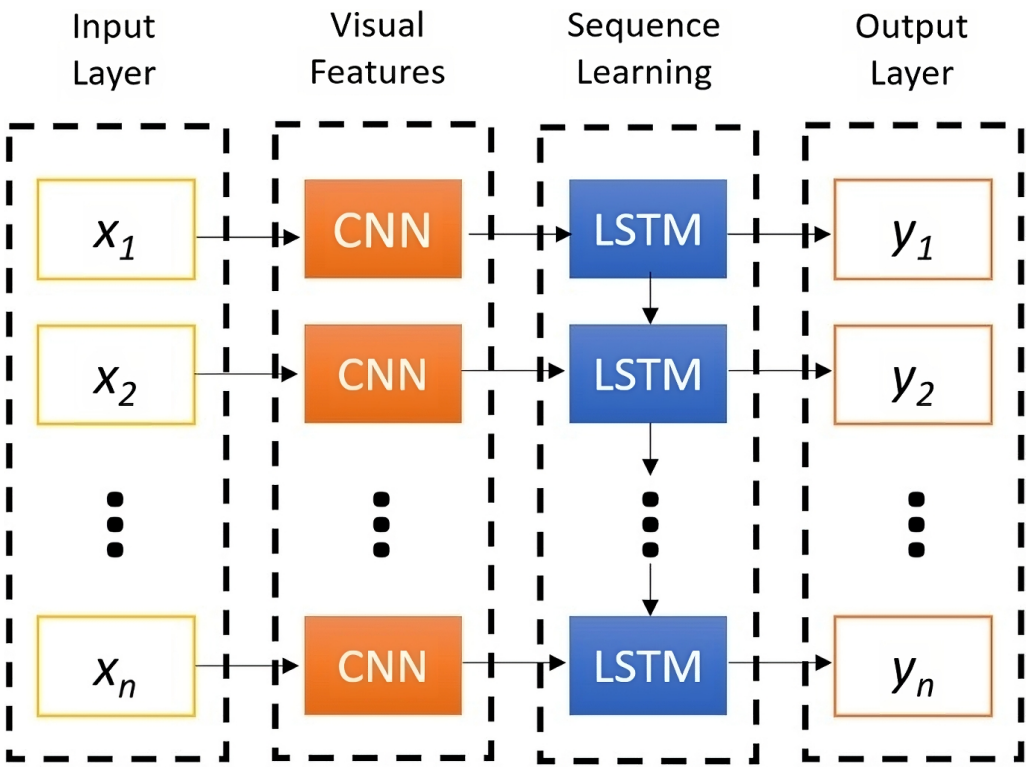


Figure 3.12: CNN-RNN architecture

The RNN is like a storyteller. It doesn't just look at the frames one by one; instead, it remembers what happened in the previous frames and thinks about how they all connect. This helps it understand the action or the story in the video. Is the person in the video walking, running, or jumping? The RNN figures this out by looking at the sequence of movements over time. By combining CNN and RNN, this architecture can not only see what's in each frame but also understand the whole story of the video.

In simple terms, the CNN-RNN architecture works by having the CNN part look at each frame for details and the RNN part puts these frames together to understand the sequence or the story in the video. This makes it a powerful tool for tasks like video classification, where you need to know both what's happening in each frame and how it all fits together over time.

Limitations without CNN-RNN

Lip-reading, a form of video classification, primarily involves understanding and interpreting the movement of lips over time to decipher spoken words. This task can be challenging and typically benefits from a combined CNN-RNN architecture. However, it is theoretically possible to approach lip-reading with only CNNs or RNNs, but each comes with its limitations:

1. Using Only CNNs: CNNs are excellent at extracting spatial features from images, such as the shape and position of lips in a frame. For lip-reading, a CNN could analyze individual frames to detect lip positions and shapes. However, the major limitation here is that CNNs, by themselves, are not adept at capturing temporal dynamics – the sequence and timing of lip movements across frames. Since speech involves a series of lip movements over time, a CNN alone might struggle to accurately interpret spoken words from lip movements.

2. Using Only RNNs: RNNs are designed to handle sequential data and could, in theory, be used to interpret the temporal sequence of lip movements. However, without the spatial feature extraction capabilities of CNNs, an RNN would have difficulty processing raw video frames. RNNs are more suited to scenarios where the input data is already in a time-series format, like audio data. For lip-reading, an RNN alone would be challenged to extract meaningful information from raw video frames.

In summary, while it's possible to attempt lip-reading with only CNNs or RNNs, each on its own has significant limitations. CNNs struggle with temporal sequencing, but they are capable of analyzing the visual features in every frame. RNNs are not designed to process raw visual data efficiently. However, they can only handle sequencing and using CNNs extracting spatial features from each frame and RNNs interpreting the temporal sequence of these features, the combined CNN-RNN architecture combines the best aspects of each, making it more effective for lip-reading tasks.

3.4.3 Model: LipNet

Definition

As the first lipreading model to achieve end-to-end sentence-level sequence prediction, LipNet [11]is a revolutionary method. The purpose is to understand text from a speaker's lip movement. Then, a task that is divided into two parts, prediction and visual feature design. This architecture differs from other models that simply

carried out word classification in such a way that it can simultaneously learn spatiotemporal visual characteristics and a sequence model.

Architecture

Three layers of spatiotemporal convolutions follow each other in the LipNet design, and each is then followed by spatial max-pooling. In order to extract relevant details from the video input, these layers are necessary. Two bidirectional Gated Recurrent Units (Bi-GRUs), which are essential for effectively aggregating the output of the spatiotemporal convolutions, receive the processed features after that. Also, for the connectionist temporal classification (CTC) loss, a softmax function over a larger vocabulary is applied after a linear transformation of each time step of the GRU output.

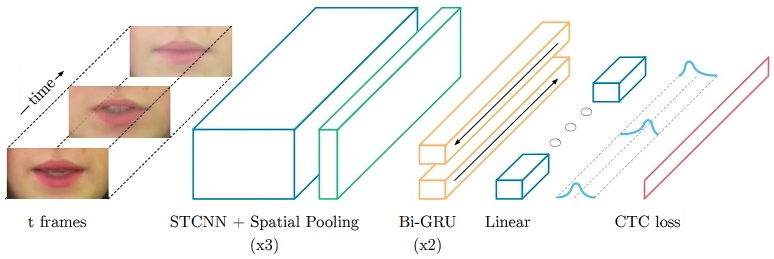


Figure 3.13: LipNet Architecture

The design of this model is the combination of recurrent and convolutional neural network components, which allows it to process input and output sequences of varying lengths efficiently. ReLU activation functions are used in all layers of LipNet, and the CTC loss is used to train the model from beginning to end. So, sentence-level lipreading tasks LipNet's great accuracy and efficiency can be attributed to its complex architecture.

3.4.4 Model: Autoencoder-decoder

Definition

The LIP2AUDSPEC [15] model uses autoencoder-decoder, a deep learning network to turn silent lip movement videos into speech. It combines CNNs and LSTM networks to understand lip shapes and movements over time and then uses an autoencoder to reconstruct the speech audio from these visual cues. Essentially, it reads lips from videos and translates them into audible speech.

Architecture

The architecture comprises a combination of Convolutional Neural Networks (CNN), a Long Short-Term Memory (LSTM) network, and fully connected layers. The CNN

layers are used to extract spatial features from the input video sequence. Following this, the LSTM layer encodes temporal dependencies. Finally, the fully connected layers work on these features to generate the bottleneck features of the audio corresponding to the silent lip movements in the video. These features are then decoded using the pre-trained autoencoder to reconstruct the auditory spectrogram.

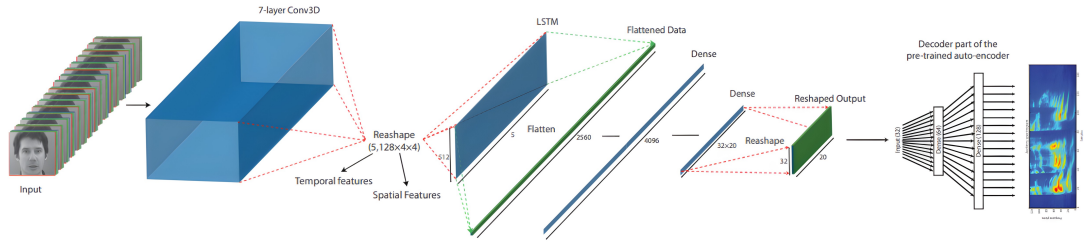


Figure 3.14: Autoencoder-Decoder

3.4.5 Model: Custom Model 1

Definition

This model is a deep neural network designed for lip reading in video classification tasks with 20 classes. It consists of multiple layers, including convolutional layers for feature extraction, recurrent LSTM layers for temporal modeling, and dense layers for classification.

Architecture

Here we used CNN-RNN architecture similar to the LipNet model. However we have used LSTM for modeling temporal sequences and maintaining context over time. Here 'TimeDistributed' is used in each layer to ensure that convolutional and pooling operations are applied consistently and independently to each frame in the video sequence. This maintains a uniform architecture, facilitates the capture of temporal information through subsequent LSTM layers, and simplifies input handling for sequences.

3.4.6 Model: Custom Model 2: ConvLSTM

Definition

This model uses a variant of the Convolutional Long Short-Term Memory (ConvLSTM) network, an extension of the traditional LSTM model, which is adept at handling sequential data with spatial dimensions. A ConvLSTM layer is an LSTM (Long Short-Term Memory) layer that has convolutional structures for multiple transitions. This suggests that the ConvLSTM layer was designed to concurrently

handle the temporal and spatial correlations within the data. Since LSTM input data is 1-D can not be used with spatial sequence data, such as sets of radar scans, videos that is why ConvLSTM was developed with 3-D input data.

Architecture

Both a ConvLSTM layer and a sequence of a convolutional layer followed by flattening and then an LSTM layer offers two different approaches to deal with spatiotemporal data in neural networks.

In ConvLSTM, the spatial information is preserved throughout the LSTM processing. This is particularly useful for tasks like video processing or any sequence prediction task where both the spatial features (shape, texture, etc.) of individual frames and the temporal dynamics between frames are important.

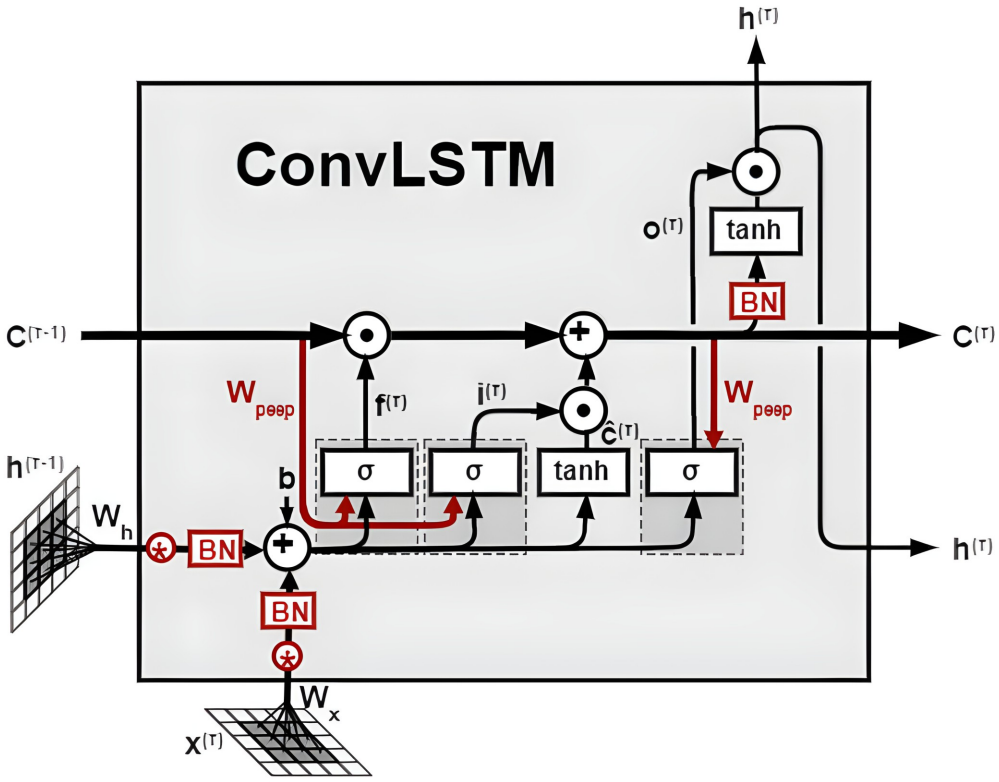


Figure 3.15: Convolutional-LSTM architecture

Moreover, ConvLSTM layers are generally more parameter-efficient when dealing with high-dimensional input data, as they do not require the data to be flattened before processing.

Consequently, in the Convolutional Layer followed by the Flattening and LSTM Layer, when the output of the convolutional layers is flattened, the spatial structure

of the feature maps is lost. This means the LSTM layers do not have access to the original spatial relationships of the features. This approach might be more suitable for tasks where spatial features are important but can be effectively summarized before temporal processing. For example, in some time-series prediction tasks where each time step involves analyzing an image to extract certain features, and the sequence of these features is what's crucial.

In summary, the choice between these two approaches depends on the specific requirements of the task at hand. ConvLSTM is generally more suited for problems where maintaining the spatial structure of the data throughout the processing is important, while the sequential approach of convolutional layers followed by LSTM might be more appropriate when the spatial features can be condensed into a vector form without losing significant information before analyzing the temporal dynamics.

Chapter 4

Analysis

4.1 LipNet

Implementation

Here we tried to implement this architecture on our Bangla corpus. Our model takes as input video data with dimensions (30, 100, 200, 1), which represents videos with 30 frames, each frame having a size of 100 x 200 pixels in grayscale. We used Keras, a high-level neural networks API, with a TensorFlow backend, for building and training the model. Here's a brief explanation of the model:

1. Input Layer (TimeDistributed + Conv2D):
   This layer takes a sequence of 2D frames as input (e.g., video frames). Then the TimeDistributed wrapper applies the subsequent convolutional layers to each time step independently and Conv2D performs 2D convolution on each frame to extract spatial features using 32 filters, a (3, 3) kernel, 'relu' activation, and 'same' padding.

2. MaxPooling2D Layer:
   This is applied after each Conv2D layer. It is performed with a (2, 2) pool size, reducing spatial dimensions and retaining important features.

3. Dropout Layer:
   It is applied after each MaxPooling2D layer and helps to prevent overfitting by randomly setting a fraction (20%) of input units to zero during training.

4. Flatten Layer:
   It converts the output from the convolutional layers into a 1D vector which is necessary before feeding the output into recurrent layers.

5. Bidirectional GRU Layers: Bidirectional GRU layers capture temporal dependencies in both forward and backward directions. The first GRU layer has

256 units and *'return_sequences=True'*, meaning it returns sequences of hidden states. The second GRU layer also has 256 units but has *'return_sequences=False'*, producing a single output vector.

6. Fully Connected (Dense) Layer:
   A dense layer with 512 units and 'relu' activation is then used which adds non-linearity to the features extracted by the previous layers.

7. Output Layer (Dense):
   This is the final dense layer with 'softmax' activation. It produces class probabilities for multi-class classification (20 classes in this case). Each unit corresponds to a class, and the softmax function converts raw scores into probability scores. Categorical cross-entropy is used as the loss function, which is standard for multi-class classification tasks.

time_distributed_30 istributed)	(TimeD (None, 30, 98, 198, 32)
time_distributed_31 istributed)	(TimeD (None, 30, 49, 99, 32)
time_distributed_32 istributed)	(TimeD (None, 30, 47, 97, 64)
time_distributed_33 istributed)	(TimeD (None, 30, 23, 48, 64)
time_distributed_34 istributed)	(TimeD (None, 30, 21, 46, 128)
time_distributed_35 istributed)	(TimeD (None, 30, 10, 23, 128)
time_distributed_36 istributed)	(TimeD (None, 30, 29440)
lstm_1 (LSTM)	(None, 30, 128)
dropout_18 (Dropout)	(None, 30, 128)
lstm_2 (LSTM)	(None, 128)
dense_12 (Dense)	(None, 256)
dropout_19 (Dropout)	(None, 256)
dense_13 (Dense)	(None, 20)

Figure 4.1: Implemented LipNet model details

Overall, this model combines convolutional layers for spatial feature extraction, bidirectional GRU layers for temporal modeling, and fully connected layers for classification. The dropout layers help prevent overfitting during training.

Result

Figure-4.2 shows the evaluation graphs for the proposed Lip-net model used in our

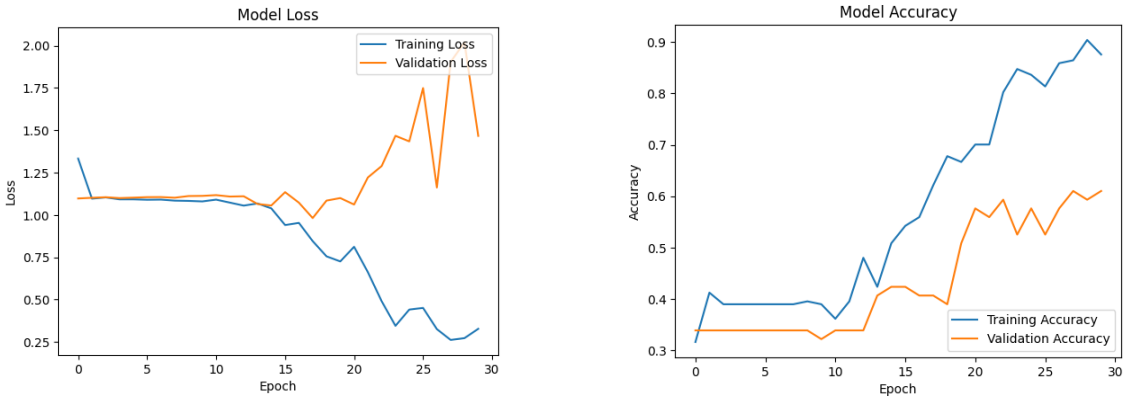


Figure 4.2: Evaluation graph for Lip-Net

research. In this the 'Model Loss' is plotted on the first graph; the blue line shows that the training loss has dropped drastically and is still decreasing which is a sign that the model is learning from the training set. Because the model performs well on training data but poorly on unseen validation data, the validation loss, represented by the orange line, begins to fluctuate and grow towards later epochs, possibly indicating overfitting. The 'Model Accuracy' is represented by the second graph, where the training accuracy is represented by the blue line. This line climbs significantly across the epochs, showing better performance on the training set. On the other hand, the orange line represents the validation accuracy, which shows a slower rate of rise and notable volatility, suggesting less steady performance on the validation set.

4.2 AutoEncoder-Decoder

Implementation

Our research primarily focuses on correctly identifying the spoken words, not generating audio. Hence the decoder is not used in our case. Now in this part layers and hyperparameters we used in this model are explained.

1. Convolutional Layers:
   There are 7 convolutional layers in this model. Each convolutional layer uses a 3D convolution operation (Conv3D). The kernel size for each convolutional

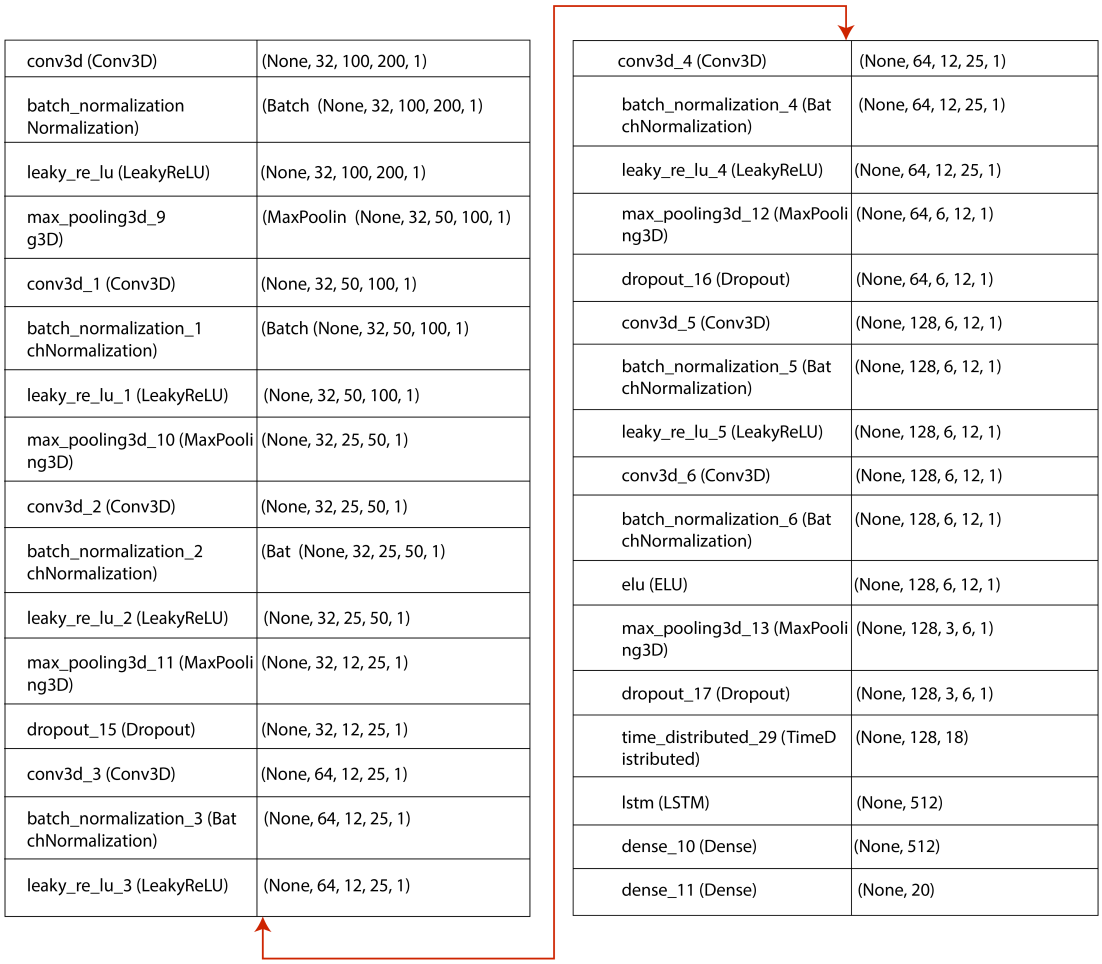
layer is [3, 3, 3]. The number of filters starts at 32 and increases to 128 as we go deeper into the network. The "channels_first" data format is used, which means the input shape is expected to be (channels, depth, height, width). Batch normalization is applied after each convolutional layer to stabilize and speed up training. LeakyReLU activation is used after each convolutional layer except for the last one, where ELU (Exponential Linear Unit) activation is used. L2 regularization with a regularization strength of 0.0005 is applied to the convolutional layers.

2. MaxPooling Layers:
   There are 4 max-pooling layers in this model. Max-pooling is performed in the spatial dimensions (width and height) with a pool size of (2, 2, 1), where the third dimension (depth) remains unchanged. Max-pooling helps reduce the spatial dimensions and capture the most important features.

3. Dropout Layers:
   There are 3 dropout layers with a dropout rate of 25% (0.25). Dropout is used for regularization to prevent overfitting.

4. TimeDistributed Layer:
   This layer wraps a Flatten operation, making it TimeDistributed, which means it's applied to each time step of the input sequence. It helps flatten the spatial features obtained from the convolutional layers, preparing them for input to the recurrent (LSTM) layer.

5. LSTM Layer:
   There is a single LSTM layer with 512 units. The return_sequences parameter is set to False, which means it produces a 2D output (not a sequence), suitable for feeding into a fully connected layer. LSTM layers are commonly used for sequence modeling and can capture temporal dependencies in the data.



conv3d (Conv3D)	(None, 32, 100, 200, 1)
batch_normalization Normalization)	(Batch (None, 32, 100, 200, 1)
leaky_re_lu (LeakyReLU)	(None, 32, 100, 200, 1)
max_pooling3d_9 g3D)	(MaxPoolin (None, 32, 50, 100, 1)
conv3d_1 (Conv3D)	(None, 32, 50, 100, 1)
batch_normalization_1 chNormalization)	(Batch (None, 32, 50, 100, 1)
leaky_re_lu_1 (LeakyReLU)	(None, 32, 50, 100, 1)
max_pooling3d_10 (MaxPooli ng3D)	(None, 32, 25, 50, 1)
conv3d_2 (Conv3D)	(None, 32, 25, 50, 1)
batch_normalization_2 chNormalization)	(Bat (None, 32, 25, 50, 1)
leaky_re_lu_2 (LeakyReLU)	(None, 32, 25, 50, 1)
max_pooling3d_11 (MaxPooli ng3D)	(None, 32, 12, 25, 1)
dropout_15 (Dropout)	(None, 32, 12, 25, 1)
conv3d_3 (Conv3D)	(None, 64, 12, 25, 1)
batch_normalization_3 (Bat chNormalization)	(None, 64, 12, 25, 1)
leaky_re_lu_3 (LeakyReLU)	(None, 64, 12, 25, 1)

conv3d_4 (Conv3D)	(None, 64, 12, 25, 1)
batch_normalization_4 (Bat chNormalization)	(None, 64, 12, 25, 1)
leaky_re_lu_4 (LeakyReLU)	(None, 64, 12, 25, 1)
max_pooling3d_12 (MaxPooli ng3D)	(None, 64, 6, 12, 1)
dropout_16 (Dropout)	(None, 64, 6, 12, 1)
conv3d_5 (Conv3D)	(None, 128, 6, 12, 1)
batch_normalization_5 (Bat chNormalization)	(None, 128, 6, 12, 1)
leaky_re_lu_5 (LeakyReLU)	(None, 128, 6, 12, 1)
conv3d_6 (Conv3D)	(None, 128, 6, 12, 1)
batch_normalization_6 (Bat chNormalization)	(None, 128, 6, 12, 1)
elu (ELU)	(None, 128, 6, 12, 1)
max_pooling3d_13 (MaxPooli ng3D)	(None, 128, 3, 6, 1)
dropout_17 (Dropout)	(None, 128, 3, 6, 1)
time_distributed_29 (TimeD istributed)	(None, 128, 18)
lstm (LSTM)	(None, 512)
dense_10 (Dense)	(None, 512)
dense_11 (Dense)	(None, 20)

Figure 4.3: Implementation of Autoencoder Decoder model

6. Fully Connected Layers:
   There is a dense (fully connected) layer with 512 units and ReLU activation following the LSTM layer. This layer helps further process the features extracted by the previous layers.

7. Output Layer:
   The final layer is a dense layer with a softmax activation function. The number of units in this layer corresponds to the number of target classes, which is 20 in this case. Categorical cross-entropy is used as the loss function, which is appropriate for multi-class classification tasks. Moreover, the Adam optimizer is used with a learning rate of 0.0001 (lr=0.0001).

Result

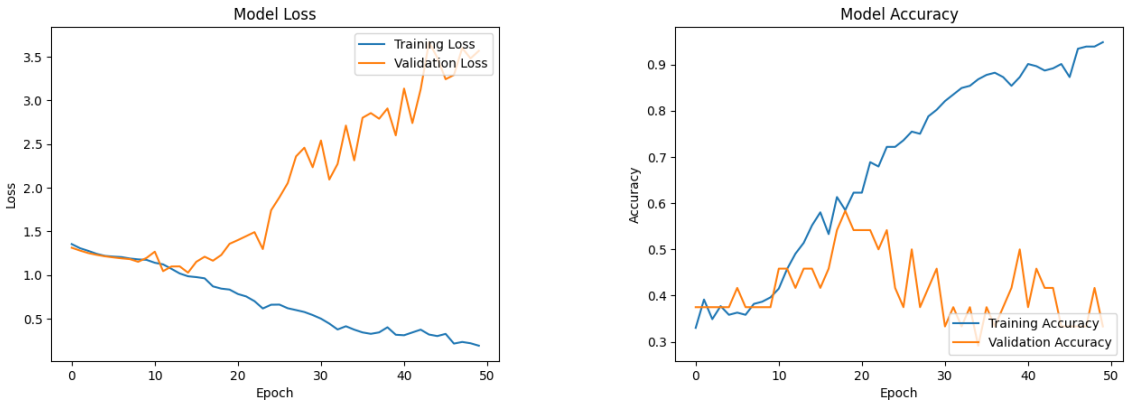


Figure 4.4: Evaluation graph for Auto Encoder-Decoder

Figure-4.4 shows The evaluation graph for the model "Encoder-Decoder" where the 'Model Loss' graph shows a steadily declining training loss. On the other hand, an upward trajectory and instability in the validation loss point to overfitting. The training accuracy is steadily increasing, as seen by the 'Model Accuracy' graph, indicating that the model gets better with time on the training set. Even though it is getting better, the validation accuracy is inconsistent and typically lower than the training accuracy, which suggests that the model might not be effectively generalizing to new data.

4.3 Custom Model - 1

Implementation

1. Convolutional and MaxPooling layer:
   The first layer in this model is a TimeDistributed(Conv2D) layer. It applies a convolutional operation with 32 filters and a (3, 3) kernel size to each frame of the input video. This operation introduces non-linearity through the ReLU activation function. The layer has 32 learnable parameters (32 filters * (3 * 3 + 1 bias term)).
   Following the convolutional layer, there is a TimeDistributed(MaxPooling2D) layer with a (2, 2) pool size. This layer performs max-pooling for each frame separately, reducing spatial dimensions, and has no learnable parameters.

   The next TimeDistributed(Conv2D) layer has 64 filters and a (3, 3) kernel size. Again, ReLU activation is applied to each frame. This layer introduces 18,496 parameters (64 filters * (3 * 3 * 32 previous channels + 1 bias term)).

Another TimeDistributed(MaxPooling2D) layer with a (2, 2) pool size follows, performing frame-wise max-pooling without additional parameters.

Subsequently, there's a TimeDistributed(Conv2D) layer with 128 filters and a (3, 3) kernel size. ReLU activation is applied to each frame, and this layer has 73,856 parameters (128 filters * (3 * 3 * 64 previous channels + 1 bias term)). A third TimeDistributed(MaxPooling2D) layer with (2, 2) pool size reduces spatial dimensions again without any learnable parameters.

2. Flatten Layer
   The TimeDistributed(Flatten()) layer flattens the output from the convolutional layers for each frame, converting it into a 1D vector for input into the subsequent LSTM layers. This layer has no learnable parameters.

3. LSTM layer
   Next, there is an LSTM layer with 128 units and 'return_sequences=True'. This layer captures temporal dependencies across frames, and the 'return_sequences=True' ensures it outputs sequences of data. It has 131,584 parameters. A Dropout layer with a dropout rate of 0.5 follows, which helps prevent overfitting. This layer has no learnable parameters. The second LSTM layer is identical to the previous one, with 128 units, but it doesn't return sequences. It has the same number of parameters, 131,584.

4. Dense Layer
   Then a Dense layer with 256 units and ReLU activation follows, capturing high-level features from the LSTM output. It has 33,024 parameters. Another Dropout layer with a 0.5 dropout rate is used to regularize the model, followed by the final Dense layer with 3 units (assuming 3 classes) and a softmax activation function for classification. This layer has 771 parameters.

In summary, this model combines convolutional layers to extract spatial features from video frames, LSTM layers to capture temporal dependencies, and fully connected layers for classification. The hyperparameters and parameters of each layer determine the model's architecture and the number of learnable weights in each layer, while activations introduce non-linearity into the model.

time_distributed_30 (TimeDistributed)	(None, 30, 98, 198, 32)
time_distributed_31 (TimeDistributed)	(None, 30, 49, 99, 32)
time_distributed_32 (TimeDistributed)	(None, 30, 47, 97, 64)
time_distributed_33 (TimeDistributed)	(None, 30, 23, 48, 64)
time_distributed_34 (TimeDistributed)	(None, 30, 21, 46, 128)
time_distributed_35 (TimeDistributed)	(None, 30, 10, 23, 128)
time_distributed_36 (TimeDistributed)	(None, 30, 29440)
lstm_1 (LSTM)	(None, 30, 128)
dropout_18 (Dropout)	(None, 30, 128)
lstm_2 (LSTM)	(None, 128)
dense_12 (Dense)	(None, 256)
dropout_19 (Dropout)	(None, 256)

Figure 4.5: Custom_Model_1

Result

The evaluation graphs for our custom model (nCM-1) in figure-4.6. Here the learning trajectory over a period of thirty epochs of the custom model. The 'Model Loss' graph shows that while the validation loss first falls but then fluctuates, indicating potential overfitting or instability in the validation phase, the training loss constantly reduces, indicating effective learning. The 'Model Accuracy' graph shows that validation accuracy is increasing as well, but it is doing so with more instability. Training accuracy is increasing consistently,

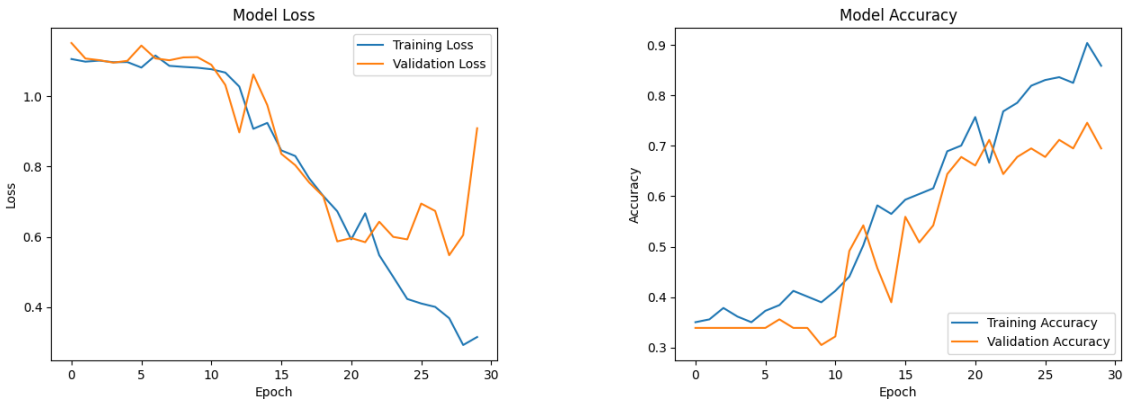


Figure 4.6: Evaluation graph for nCM-1

4.4 Custom Model - 2

Implementation

This model combines Convolutional LSTM (ConvLSTM) layers which utilize the VGG16 model's architecture as inspiration. The usage of ConvLSTM layers indicates that it is intended to operate with video data. Let's dissect the main elements and how they work:

1. Type of Model: It is a sequential model so that the layers are created in a straight line. There is exactly one input tensor and one output tensor for each layer.

2. Shape of input: The input shape is defined as (30, 100, 200, 1), which is equivalent to 30 time steps of grayscale images of 100 by 200 pixels (1 channel).

3. ConvLSTM2D Layers: Multiple ConvLSTM2D layers with varying numbers of filters (32, 64, 128, 256) and kernel sizes (3, 3) are used in the model. The same padding and the tanh activation function are applied.

4. TimeDistributed with MaxPooling2D Layers: The TimeDistributed layer is used with MaxPooling2D layers following a few ConvLSTM2D layers. For lowering the number of parameters and the spatial dimensions (height and width) of the model, this combination allows the model to apply max pooling to each frame individually, contributing in the control of overfitting.

5. GlobalAveragePooling3D Layer: Here in this model it lowers the output's dimensionality from the earlier layers.

6. Dense and Dropout Layers: The model then consists of 512 and 1024 unit fully connected (Dense) layers, each of these layers followed by a Dropout layer with a 0.5 dropout rate. By randomly setting a certain amount of the input units

to 0 at each training update, the regularization technique known as "dropout" helps avoid overfitting.

7. Output Layer: The final Dense layer indicates that the model is designed for a twenty-class classification problem since it contains three units with a softmax activation function.

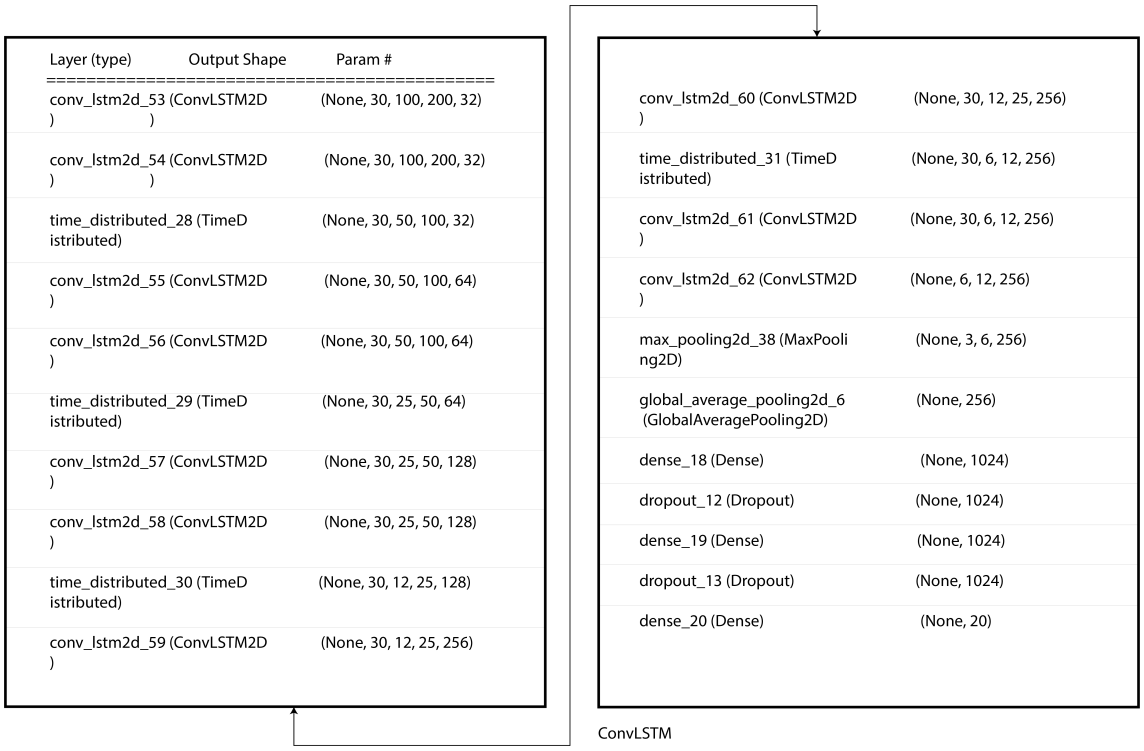


Figure 4.7: Custom_Model_2 (ConvLSTMl)

Result

Figure 4.8 shows evaluation graphs of the 2nd Custom model. In the "Model

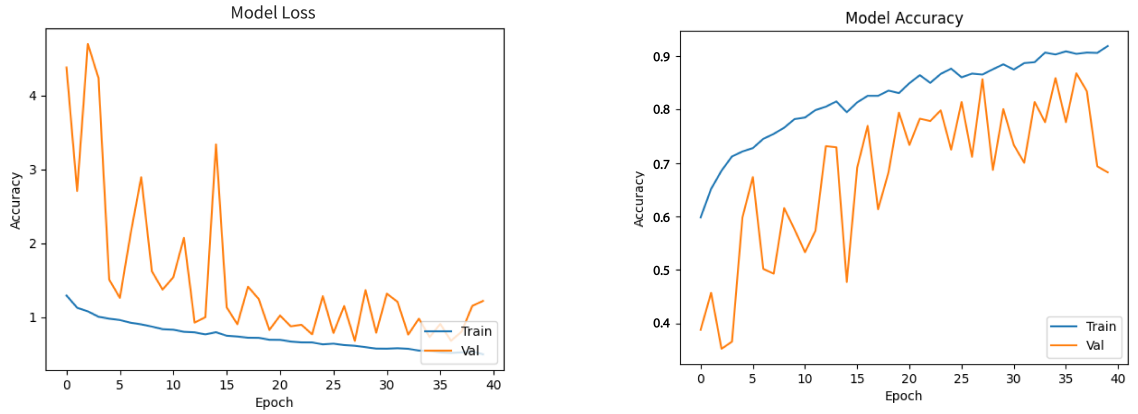


Figure 4.8: Evaluation graph for nCM-2

Accuracy" graph, training accuracy steadily increases over 40 epochs. Validation

accuracy, on the other hand, fluctuates, which may indicate that the model is overfitting or that more effective generalization methods are required.
As might be predicted, the loss graph shows a diminishing loss over time. Nevertheless, the 'validation' curve exhibits extreme volatility and unexpectedly high values.

4.5 Model Comparison

Various categorization reports were produced in this research to assess the performance of the suggested model. These reports provide a thorough evaluation by using multiple metrics. Accuracy shows the percentage of all predictions that have been generated correctly. It gives a clear and concise indicator of the model's accuracy rate for every class. Whereas, the F1 score is the harmonic mean of recall and accuracy, where recall is the ratio of all actual positives (including false negatives) to all true positives and precision is the ratio of true positives to all positives (including false positives).

Proposed Architecture	Accuracy	F1-Score	Top-1	Top-10
Lip-Net	62%	64%	62%	88.02%
Auto Encoder-Decoder	49.65%	46%	49.65%	66%
Custom Model 1	70.86%	77%	70.86%	95.89%
Custom Conv-LSTM	76.24%	78.11%	76.24%	98.88%

Figure 4.9: Model Comparison Table

The 4.9 displays the comparison between various architectures that have been proposed for lip-reading Bangla words. These designs are assessed using multiple criteria, including accuracy, F1-Score, top-1 accuracy, and top-10 accuracy.

Firstly, Lip-Net performs moderately, exhibiting a 62% accuracy and a comparable Top-1 Accuracy, indicating that it successfully selects the correct word 62% of the time. With a somewhat higher F1-Score of 64%, it shows a respectable balance between recall and precision. The Top-10 Accuracy is significantly higher at 88.02%, meaning that 88.02% of the time the correct word is among the top 10 predictions

made by the model.

Secondly among all the 4 architectures, Auto Encoder-Decoder performs poorly, with an accuracy of 49.65%, matching F1-Score and Top-1 Accuracy. Though still less than other models, the Top-10 Accuracy is 66%, which is comparatively better than its Top-1 Accuracy.

Again, with an accuracy of 70.86% and the highest F1-Score of 77%, Custom Early-Fusion shows a notable increase. This indicates a solid balance between recall and precision and, consequently, an efficient classification of the correct word. Additionally high are the Top-1 and Top-10 Accuracies, particularly the latter at 95.89%, which shows that the right word is nearly always among the top 10 guesses.

Finally, with an F1-Score of 78.11% and the maximum accuracy of 76.24%, Custom Conv-LSTM performs the best overall. Its Top-1 Accuracy is consistent with the overall accuracy, and its remarkable Top-10 Accuracy of 98.88% is the greatest of all the architectures and indicates that the correct word is almost always among the first 10 predictions.

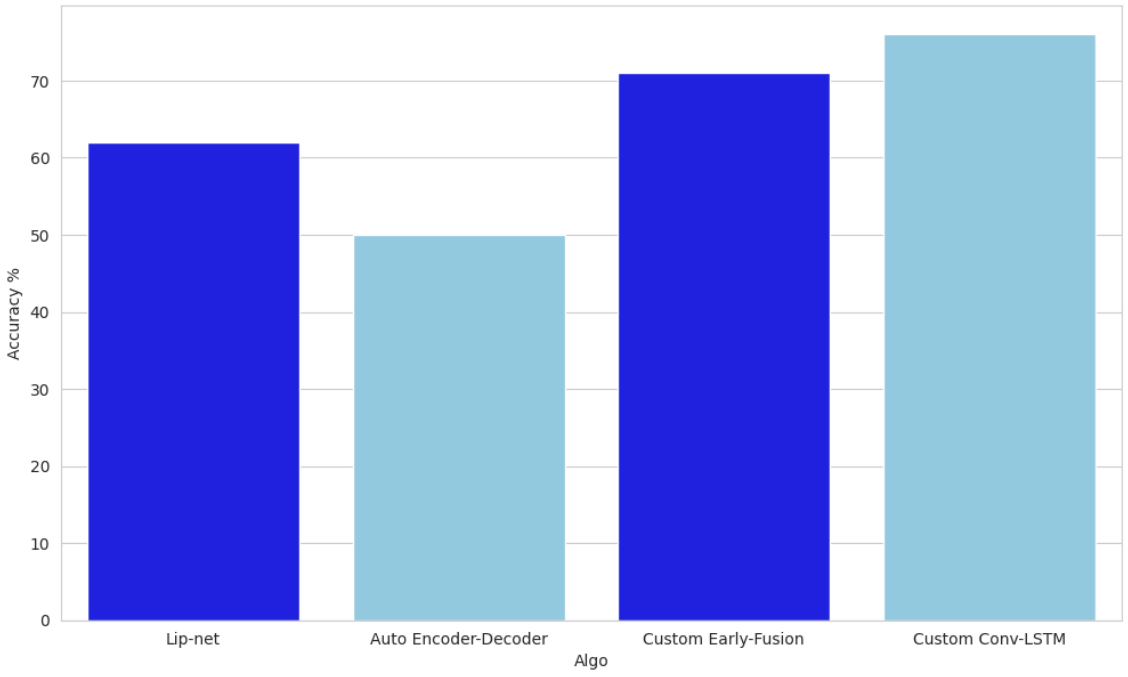


Figure 4.10: Model Comparison Bar

These evaluations show that the Custom Conv-LSTM architecture stands out to be the most effective model for word level lipreading in Bangla according to our research. For practical applications, it is highly dependable since it not only delivers

the best probability of accurately recognising the correct word as its first choice (Top-1 Accuracy), but it also offers the greatest assurance that the correct word will be within its top 10 predictions (Top-10 Accuracy). The model maintains an excellent balance between precision and recall, which is crucial in applications where both false positives and false negatives have large costs, as further indicated by the high F1-Score.

Chapter 5

Conclusion

In conclusion, lipreading is the skill of understanding a speaker's words solely from the movement of their lips which is crucial in many different disciplines like accessibility, noise-sensitive communication, human-computer interaction, multimodal communication, security, and a great deal more. Despite the fact that the English language has been the subject of numerous studies in this area, the Bangla language has not yet been the subject of any studies. CNN, Bi-LSTM, Bi-GRU, and CTC were frequently employed for English lipreading. However, after reviewing some studies, we discovered that the lipreading models utilized for English were inadequate for other languages. Therefore, it is imperative that we carry out research on Bangla lip-reading. However, there isn't yet a corpus of Bangla that can be used for lipreading. So, for our research a small video corpus has been created, where at most 200 videos for all the 20 words collected from different speakers. Then after necessary preprocessing of the dataset 4 different models have been applied into the features of every video keeping their labels aside which are basically the target variables. Finally, one of the custom models stands out among the others in terms of the accuracy and correctly classified 78% each time it has been executed into our test data.

However, due to lack of sufficient videos in the dataset the model performance were below 90%. Hence, in our future work, we will keep increasing the Bangla video corpus for greater accuracy.

Bibliography

[1] D. Easton and M. Basala, "Perceptual dominance during lipreading," *Perception & Psychophysics*, vol. 32, no. 6, pp. 562–570, 1982.

[2] J. Goldschen, O. N. Garcia, and E. D. Petajan, "Continuous automatic speech recognition by lipreading," in *Motion-Based Recognition*, Springer, 1997, pp. 321–343.

[3] G. Neti, J. Potamianos, J. Luettin, *et al.*, "Audio visual speech recognition," IDIAP, Tech. Rep., 2000.

[4] A. Rix, J. Beerends, M. Hollier, and A. Hekstra, "Perceptual evaluation of speech quality (pesq), an objective method for end-to-end speech quality assessment of narrowband telephone networks and speech codecs," ITU-T Recommendation, Tech. Rep., 2001.

[5] T. F. Matthews, J. A. Cootes, J. Bangham, S. Cox, and R. Harvey, "Extraction of visual features for lipreading," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, pp. 198–213, 2002.

[6] M. Elhilali, T. Chi, and S. Shamma, "A spectrotemporal modulation index (stmi) for assessment of speech intelligibility," *Speech communication*, vol. 41, no. 2, pp. 331–348, 2003.

[7] J. Cooke, S. Barker, D. Cunningham, and X. Shao, "An audio-visual corpus for speech perception and automatic speech recognition," *The Journal of the Acoustical Society of America*, vol. 120, no. 5, pp. 2421–2424, 2006.

[8] M. Zhao, M. Barnard, and M. Pietikainen, "Lipreading with local spatiotemporal descriptors," *IEEE Transactions on Multimedia*, vol. 11, no. 7, pp. 1254–1265, 2009.

[9] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, "300 faces in-the-wild challenge: The first facial landmark localization challenge," in *IEEE International Conference on Computer Vision Workshops*, 2013, pp. 397–403.

[10] G. Zhou, G. Zhao, X. Hong, and M. Pietikainen, "A review of recent advances in visual speech decoding," *Image and Vision Computing*, vol. 32, no. 9, pp. 590–605, 2014.

[11] Y. M. Assael, "Lipnet: End-to-end sentence-level lipreading," *arXiv preprint arXiv:1611.01599*, 2016.

[12] J. S. Chung, A. Senior, O. Vinyals, and A. Zisserman, "Lip reading sentences in the wild," *arXiv preprint arXiv:1611.05358*, 2016. [Online]. Available: https://www.robots.ox.ac.uk/~vgg/publications/2016/Chung16/chung16.pdf.

[13] S. Chung and A. Zisserman, "Lip reading in the wild," in *Asian Conference on Computer Vision*, 2016.

[14] S. Gergen, S. Zeiler, A. H. Abdelaziz, R. Nickel, and D. Kolossa, "Dynamic stream weighting for turbodecoding-based audiovisual asr," in *Interspeech*, 2016, pp. 2135–2139.

[15] H. Akbari, "Lip2audspec: Speech reconstruction from silent lip movements," *arXiv preprint arXiv:1710.09798*, 2017.

[16] A. Ephrat and S. Peleg, "Vid2speech: Speech reconstruction from silent video," *arXiv preprint arXiv:1701.00495*, 2017.

[17] T. Stafylakis and G. Tzimiropoulos, "Combining residual networks with lstms for lipreading," in *conference of the international speech communication association*, 2017, pp. 3652–3656.

[18] T. Afrouas, J. Chung, and A. Zisserman, "Deep lip reading: A comparison of models and an online application," in *ArXiv*, 2018.

[19] "Bangla short speech commands recognition using convolutional neural networks," *IEEE Conference Publication — IEEE Xplore*, 2018.

[20] M. Faisal, "Deep learning for lip reading using audio-visual information for urdu language," *arXiv preprint arXiv:1802.05521*, 2018.

[21] L. McQuillan, "Is lip-reading the secret to security?" *Biometric Technol Today*, vol. 2019, pp. 5–7, 2019.

[22] B. Purkaystha, M. Nahid, and M. Islam, "End-to-end bengali speech recognition using deepspeech," *ResearchGate*, 2019.

[23] X. Chen, J. Du, and H. Zhang, "Lipreading with densenet and resbi-lstm," *Signal, Image and Video Processing*, vol. 14, no. 5, pp. 981–989, 2020.

[24] N. Ali, M. Abdulmunem, and A. Ali, "Constructed model for micro-content recognition in lip reading based deep learning," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 5, pp. 2557–2565, 2021.

[25] "Hlr-net: A hybrid lip-reading model based on deep convolutional neural networks," 2021.

[26] A. Samin, M. Kobir, S. Kibria, and M. Rahman, "Deep learning based large vocabulary continuous speech recognition of an under-resourced language bangladeshi bangla," *Acoustical Science and Technology*, vol. 42, no. 5, pp. 252–260, 2021.

[27] B. Soundarya, R. Krishnaraj, and S. Mythili, "Visual speech recognition using convolutional neural network," *IOP Conference Series: Materials Science and Engineering*, vol. 1084, no. 1, p. 012 020, 2021.

[28] "A novel method for lip movement detection using deep neural network," *Journal of Scientific & Industrial Research*, vol. 81, no. 06, 2022.

[29] U. Atila and F. D. Sabaz, "Turkish lip-reading using bi-lstm and deep learning models," *Engineering Science and Technology, an International Journal*, vol. 35, p. 101 206, 2022.

[30] M. Miled, M. Messaoud, and A. Bouzid, "Lip reading of words with lip segmentation and deep learning," *Multimedia Tools and Applications*, vol. 82, no. 1, pp. 551–571, 2022.

[31] G. Schwiebert, C. Weber, L. Qu, H. Siqueira, and S. Wermter, "A multimodal german dataset for automatic lip reading systems and transfer learning," Tech. Rep., 2022.