KANTIPUR ENGINEERING COLLEGE

(Affiliated to Tribhuvan University)

Dhapakhel, Lalitpur

[Subject Code: CT755]

A MAJOR PROJECT MID-TERM REPORT ON

NEPALI SPEECH RECOGNITION USING BILSTM AND RESNET

Submitted by:

Ankit Kafle [KAN076BCT012]

Dikshyanta Giri [KAN076BCT026]

Jenith Rajlawat [KAN076BCT034]

Nawaraj Shah [KAN076BCT044]

A MAJOR PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR IN COMPUTER ENGINEERING

Submitted to:

Department of Computer and Electronics Engineering

December, 2023

NEPALI SPEECH RECOGNITION USING BILSTM AND RESNET

Submitted by:

Ankit Kafle [KAN076BCT012]

Dikshyanta Giri [KAN076BCT026]

Jenith Rajlawat [KAN076BCT034]

Nawaraj Shah [KAN076BCT044]

Supervised by:

Prof. Dr. Subarna Shakya

Professor

Department of Electronics and Computer Engineering, IOE

Pulchowk Campus

A MAJOR PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR IN COMPUTER ENGINEERING

Submitted to:

Department of Computer and Electronics Engineering

Kantipur Engineering College

Dhapakhel, Lalitpur

December, 2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI: Artificial Intelligence

ASR: Automatic Speech Recognition

BILSTM: Bidirectional Long Short-Term Memory

CNN: Convolutional Neural Network

CTC: Connectionist Temporal Classification

E2E: End-To-End

GRU: Gated Recurrent Units

MFCCs: Mel-frequency cepstral coefficients

RESNET: Residual Networks

RNN: Recurrent Neural Network

CHAPTER 1
INTRODUCTION

1.1 Background

Speaking and writing are the two important things that help us to communicate among us. Deficient either in writing or speaking affects our daily activities. Most of the people in rural areas are able to speak properly but not able to write properly. Most communication technology (gadgets, mobiles, computers, etc) needs text as an input for their operation. To be familiar with the technology like Automatic Speech Recognition (ASR) can play a significant role [1]. Automatic Speech Recognition (ASR) is a technology that enables machines to convert spoken language into written text. It has evolved significantly over the years and has found applications in various fields, including telecommunications, transcription services, virtual assistants, voice-controlled systems, and more. The development of ASR systems has revolutionized human-computer interaction, allowing for more natural and intuitive communication.

Automatic Speech Recognition (ASR) is essential due to several significant reasons. Firstly, it enables a natural and intuitive means of interaction between humans and machines. By speaking commands or queries instead of typing or pressing buttons, users can engage with technology more seamlessly, providing convenience and accessibility. ASR also plays a crucial role in making digital content and services accessible to individuals with hearing or speech impairments, allowing them to participate in conversations, consume media, and access information through captioning and transcription services. Furthermore, ASR systems that support multiple languages foster effective communication across language barriers, promoting global connectivity and understanding. ASR significantly enhances efficiency and productivity in various domains, automating tasks like transcription and streamlining workflows. It forms the backbone of voice-controlled systems, enabling virtual assistants to accurately understand and execute user instructions. ASR facilitates data analysis and insights by processing and analyzing large speech datasets, enabling valuable information extraction and knowledge discovery. Additionally, ASR automates transcription, simplifying the creation of written records for information retrieval and documentation purposes. Lastly, ASR technology is vital for advancing human-machine interaction, allowing machines to understand and respond to spoken commands, leading to seamless communication and the development of intelligent virtual agents and interactive applications.

1.2 Problem Statement

The limited English language skills among certain Nepali-speaking populations hinder their ability to effectively execute tasks that require English communication. This language barrier emphasizes the need for an innovative and user-friendly Nepali speech recognition system that can accurately recognize Nepali speech, provide a textual representation, and convert it into English text. Such a system enables seamless interaction with English-based technologies, services, and information, thereby empowering individuals with limited English language knowledge to access a wide range of opportunities and resources.

1.3 Objectives

i To develop an Nepali Speech Recognition system.

ii To execute the commands based on the user's input voice.

1.4 Application Scope

Nepali speech recognition technology has diverse applications, including voice assistants which enables users to interact with devices, perform tasks, and access information using their voice in the Nepali language. The technology enhances efficiency, accessibility, and user experience in various domains, revolutionizing communication, transcription, language learning, customer service.Language localization is another area where Nepali speech recognition can be utilized, enabling voice input and commands in Nepali for software, applications, and websites, thereby enhancing accessibility for Nepali-speaking users.

1.5 Features

1. Nepali Speech Recognition
2. Voice-controlled virtual assistants

1.6 System Requirements

The system requirements for the project are as follows:

1.6.1 Development Requirements

1.6.1.1 Hardware Requirements

- PC with a specification of 16GB RAM and a 8th generation i7 processor

1.6.1.2 Software Requirement

- Google Colab
- Pycharm
- Jupyter Notebook

1.6.2 Deployment Requirements

1.6.2.1 Hardware Requirements

- Personal computer

1.6.2.2 Software Requirement

- Desktop Application

1.7 Feasibility Study

The feasibility study is one of the most important things to be considered for the project development. The feasibility study must be done for the different factors affecting the project. Here are some factors whose feasibility study should be done for our project.

1.7.1 Economic Feasibility

Economic feasibility attempts to weigh costs of developing and implementing a new system, against the benefits that would acquire from having the new system in a place. This feasibility study gives the top management the economic justification for the new system. A simple economic analysis which gives the actual comparison of costs and benefits is much more meaningful meaning in this case. In addition, this proves to be a useful point of reference to compare actual costs as the project progresses. There could be various types of intangible benefits on account of automation. These could include increased customer satisfaction, improvement in product quality, better decision making, and timeliness of information, expediting activities, improved accuracy of operations, better documentation and record keeping, faster retrieval of information, better employee morale

1.7.2 Technical Feasibility

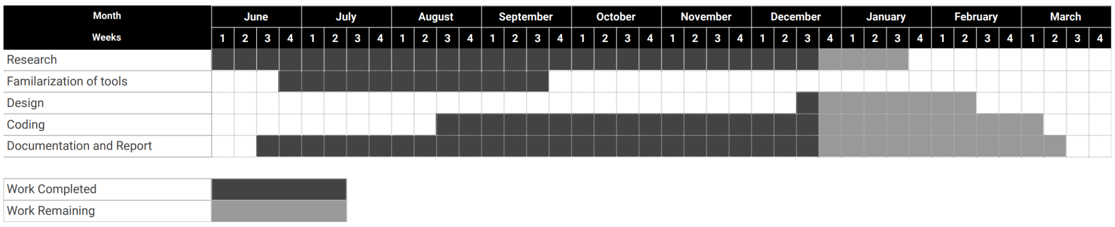
Since the proposed system utilizes software technologies and tools that are freely available, and the required technical skills are manageable, it demonstrates technical feasibility. There are many free machine learning libraries available for data analysis and predictions, along with proper documentation and courses. The hardware system in the project does not need to be highly powerful, but it requires normal computing capabilities. Additionally, the system server must be adequate and manageable for future needs. It is evident that both the hardware and software components meet the system's requirements. Therefore, it is clear that the proposed project is technically feasible.

1.7.3 Operational Feasibility

With the advancements in technology, operating any kind of system software or application is no longer challenging. Users only need to be somewhat familiar with the software system, supported by graphical explanations that can be easily understood over time with usage. This system places a strong emphasis on design-dependent parameters such as reliability, maintainability, supportability, usability, predictability, sustainability, affordability, and more. As a result, the project is operationally feasible.

1.7.4 Schedule Feasibility

Schedule feasibility is defined as the likelihood of a project being completed within its designated time limits and by the planned due date. If a project has a high probability of being completed on time, its schedule feasibility is considered high. Schedule feasibility ensures that a project can be finished before the technology becomes obsolete. Given the numerous features in our project, but with the potential for high-quality implementation, there is a very high probability that it will be completed on time.



Month	June				July				August				September				October				November				December				January				February				March			
Weeks	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Research	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	▒	▒	▒	▒									
Familiarization of tools				■	■	■	■	■	■	■	■	■	■	■	■																									
Design																											■	▒	▒	▒	▒	▒	▒	▒						
Coding											■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	▒	▒	▒	▒	▒	▒	▒	▒	▒	▒			
Documentation and Report			■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	▒	▒	▒	▒	▒	▒	▒	▒	▒	▒	▒		

Work Completed	■
Work Remaining	▒

Figure 1.1: Gantt Chart

CHAPTER 2
LITERATURE REVIEW

2.1 Related Research

The proposed model for Nepali speech recognition combines CNN, GRU, and CTC networks with MFCC feature extraction, showing promising potential for accurately transcribing spoken Nepali speech into written text [1]. This technology has the capacity to enhance interaction with communication devices and improve communication across various fields. Although the model achieves an 11% Word Error Rate (WER) using a dataset from Open Speech and Language Resources, further research and refinement are needed to enhance accuracy and real-world applicability. The 1D-CNN component captures high-level features from MFCC representations, while the GRU component, a type of recurrent neural network, excels in modeling sequential data, enabling the model to learn temporal dependencies and enhance recognition accuracy by capturing the context and structure of the Nepali language.

This study introduces a novel approach for developing a Nepali speech recognition model by utilizing a CNN-GRU architecture [2]. The Librispeech dataset is employed to collect the necessary training data, which is then subjected to preprocessing and feature extraction through the application of MFCC. The CNN-GRU model is leveraged for feature extraction and the construction of the acoustic model, while CTC serves the purpose of decoding. The evaluation of the proposed model is based on the Word Error Rate metric, which measures the accuracy of the transcribed text. The deep learning approach demonstrates substantial performance improvements in comparison to traditional methods by employing separate phonetic and linguistic constructs. The key components of the speech recognition system include front-end processing for feature extraction and model building, as well as back-end processing involving decoding using a lexicon and language model. In this study, MFCC is utilized for feature extraction, the CNN-GRU network is employed for acoustic modeling, and CTC is utilized for decoding. The MFCC features represent sequences of acoustic feature vectors, each capturing information from a small time window of the speech signal to address its non-stationarity. The neural network is trained on the available data to predict unseen data, with the CNN component summarizing and extracting useful features from the input, while the RNN component, in conjunction with CTC, contributes to the model building process.

This paper presents an end-to-end (E2E) deep learning model for automatic speech recognition (ASR) with a focus on transcribing Nepali speech into text. The model is trained and evaluated using the OpenSLR dataset, which provides audio and text data [3]. During dataset preprocessing, silent gaps present at the beginning and end of the audio files are clipped to enhance data quality. Mel Frequency Cepstral Coefficients (MFCCs) are utilized as audio features to input into the model. To achieve optimal performance, a combination of Bidirectional Long Short-Term Memory (LSTM), ResNet, and 1D Convolutional Neural Network (CNN) architectures are explored and compared. Different variations of LSTM, GRU, CNN, and ResNet are trained and tested, with the Bidirectional LSTM paired with ResNet and 1DCNN yielding the best results. For loss calculation during training, the Connectionist Temporal Classification (CTC) approach is employed, and CTC beam search decoding is utilized for character prediction. The project achieves a character error rate (CER) of 17.06 percent, outperforming other sequential models when evaluated on the OpenSLR dataset, thereby demonstrating its superiority in terms of test accuracy.

This study presents the development of a speech recognition system capable of transcribing audio input into text format [4]. The system utilizes a combination of the Long Short-Term Memory (LSTM) neural network and the Connectionist Temporal Classification (CTC) function. Performance evaluation based on the Word Error Rate (WER) metric reveals a WER of 40% without the CTC layer, which improves to 34.3% with the inclusion of CTC. The initial step involves extracting informative features from raw speech data. Subsequently, an acoustic model is generated, representing speech as a combination of different phonetic units. Over time, various methods have been employed to capture temporal audio data, with probabilistic approaches such as Hidden Markov Models (HMM) and Gaussian Mixture Models (GMM) utilized in the early stages of speech recognition. However, to address inherent issues such as gradient vanishing and bursting, recurrent neural networks have undergone advancements leading to the development of LSTM and GRU networks. The main stages of the automatic speech recognition (ASR) process encompass data preparation, feature extraction, model building, training, testing, and validation.

This paper introduces a new method of interacting with technical devices through lexical communication, leveraging voice assistants as a key technology [5]. Voice assistants, powered by AI, have the capability to recognize human speech and respond using integrated voices. The proposed voice assistant system described in this paper collects audio input from the microphone and converts it into text. The text is then processed using the Google Text-to-Speech (GTTS) engine to generate an audio file in the English language. The audio file is played back using the ”play sound” package in the Python programming language. The underlying principle of this system is Automatic Speech Recognition (ASR), which involves the device capturing audio from the microphone as the source. The recorded speech waveforms undergo acoustic analysis at three different levels: acoustic, pronunciation, and language modeling.

2.2 Related Works

One of the popular ASR software for Nepali is the Nepali ASR system developed by the Center for Speech and Language Technology (CSLT) at the University of Colorado Boulder [6]. This system is designed to recognize Nepali speech and convert it into written text. It has been trained on a large amount of Nepali speech data to improve its accuracy.

The Kaldi toolkit, a well-established and widely recognized open-source framework, has been a significant contribution to the field of automatic speech recognition (ASR) [7]. Developed in September 2011 by researchers at the Johns Hopkins University, Kaldi has gained prominence in academic and industrial domains. Its comprehensive set of tools and libraries make it a flexible and powerful platform for building state-of-the-art ASR systems. As we undertake the development of an ASR model for the Nepali language, the existence of the Kaldi toolkit serves as a valuable reference, showcasing its effectiveness in leveraging established techniques and providing a foundation for successful ASR implementations.

The combination of CNN, GRU, and CTC employed in Nepali speech recognition yielded a relatively high word error rate of 11% [1]. Therefore, for evaluating the performance, we have adopted the character error rate as the metric like our base paper [3], which resulted in a rate of 17.06%. To enhance the accuracy, we intend to leverage a wider training dataset from openslr, excluding numeric transcriptions and unnecessary silent gaps. This refined dataset will be used in conjunction with a hybrid architecture comprising BiLSTM, 1-D CNN, and ResNet.

Moving forward, we plan to incorporate a voice assistant feature by integrating the generated text from the ASR system. Additionally, as our project progresses and resources permit, we aim to explore the use of transformers, an alternative architecture to BiLSTM, to further enhance the accuracy of the ASR system.

CHAPTER 3
METHODOLOGY

3.1 Working Mechanism

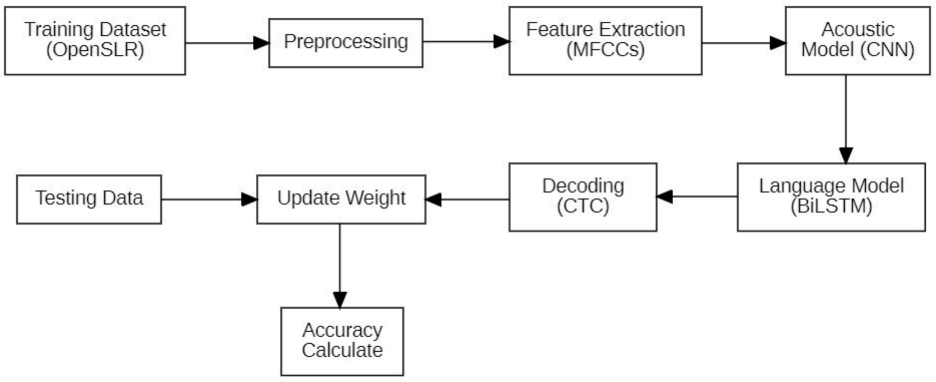


Figure 3.1: Block diagram of the Proposed ASR Training Model

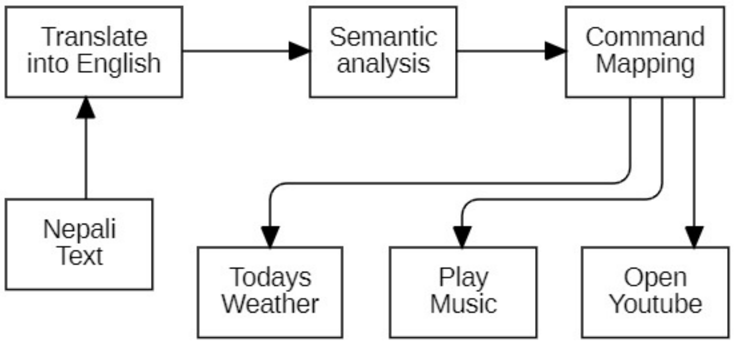


Figure 3.2: Block diagram of the Proposed ASR Implementation

1. Dataset: Two potential datasets will be used: High quality TTS data for Nepali (openslr.org/43)[8] and Large Nepali ASR training dataset (openslr.org/54) [9]. The first dataset contains female transcribed audio data for the Nepali language. It includes 2064 high-quality audio files from 18 different speakers, with multiple audio files per speaker having similar word and character patterns. However, the vocabulary and text sequences might not be diverse, leading to potential bias in the ASR model's predictions if used. On the other hand, the second dataset provides significant advantages with a larger vocabulary, more speakers, a greater number of audio files, and a more diverse sequence of characters and text. This dataset will be used for training and testing the ASR model. It consists of 157,905 audio clips from 527 unique speakers, sampled at 16KHz. The

dataset will be sourced from OpenSLR, a platform hosting speech and language resources for speech recognition. Before training, an initial cleansing process will be performed on the dataset by eliminating numeric transcriptions to prevent degradation of the model's overall performance. After removing these instances, approximately 143.6 hours of 148,188 audio clips will remain as the foundation for training and testing the model.

2. Preprocessing: The dataset will be acquired using OpenSLR, which will provide a collection of speech and language resources. Firstly, the dataset will undergo a cleansing process to eliminate numeric transcripts, as this type of data will play a minimal role and degrade the overall performance of the model. The majority of the dataset will contain large silent gaps, so the silent gaps will be clipped. This process will make the data more suitable for feature extraction and further processing, reducing potential errors.

   ALGORITHM 1: Clipping of silent gaps from both ends

```
wav ← sampled audio signal
Δ ← appropriate window length
INPUT: wav,Δ
PROCESS:
wavAvg ← Average( $|wav|$)
N ← Length(wav)

/* Removing the silent gap from the start */
for idx = 0, Δ , 2 Δ , . . . ,N - Δ do
    win ← wav[ idx : idx + Δ ]
    winAvg ← Avergae($|win|$)
    if $winAvg > wavAvg$ then
        wav ← wav[idx :]
        break
    end if
end for
```

```
/* Removing the silent gap from the end */
for idx = N - $\Delta$, N - 2 $\Delta$, . . . , 0 do
    win ← wav[idx : idx + $\Delta$]
    winAvg ← Avergae($|win|$)
    if $winAvg > wavAvg$ then
        wav ← wav[: idx]
        break
    end if
end for
OUTPUT: processed_wav ← wav
```

3. Feature Extraction: The extraction of the best parametric representation of acoustic signals will be an important task to produce better recognition performance. Mel Frequency Cepstral Coefficients (MFCCs) will be a powerful feature extraction mechanism that will generate coefficients [6]. This feature extraction mechanism will go through six stages:

   - Pre-emphasis: The signal will be processed through a filter that emphasizes higher frequencies, increasing the energy of the signal at higher frequencies.
   - Framing: The speech samples obtained will be segmented into smaller frames.
   - Windowing: Samples will be multiplied using a scaling function to smooth the signals near the edges.
   - Discrete Fourier Transform(DFT): Each frame's samples will be converted from the time domain into the frequency domain.
   - Mel Filter Bank Processing: The frequency range in the DFT spectrum is very large and wide. Mel filter banks will have a collection of bandpass filters over the mel scale. The FFT signal obtained will pass through the Mel scale filter bank, and each filter's output will be the sum of its filtered spectral components.
   - Discrete Cosine Transform(DCT): The log Mel spectrum obtained through Mel filter bank processing will be converted into the time domain using DCT. The result of this conversion will be called Mel Frequency Cepstral Coefficients (MFCCs). MFCCs will be highly favorable for signal process-

ing with RNN, DNN, and CNN. For human speech, 13 mel scales will be sufficient for extracting features from the signal. The equation involved in calculating the mel scale from the frequency in Hertz (f) will be given by

$$Mel(f) = 2595*\log(1+(f/700)) \quad (1)$$

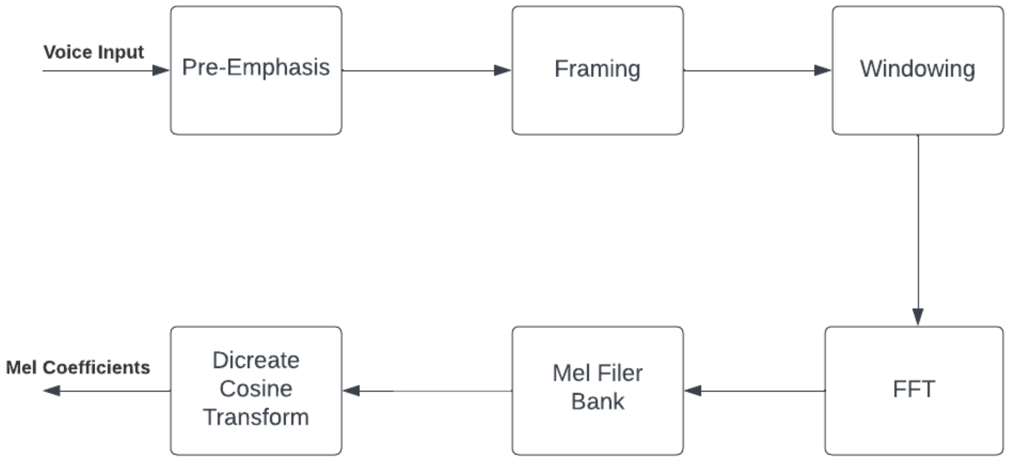


Figure 3.3: Block Diagram of MFCC

4. Acoustic and Language Model:
   - Residual Block(ResNet): The ASR Model will begin with the implementation of Residual Blocks. These blocks will utilize shortcut connections to add the input of a block into the output of our stacked layers. The residual block for our model will have an output of G(x)+x, where x will be the block's input and G(x) will be the output of stacked layers.
   - 1D-CNN: After residual blocks, the model will employ a 1D-CNN layer for localized feature extraction. The 1D-CNN will perform convolution operations on the temporal direction of the signal and will use trained weighted filters (kernels) to extract localized features. This process will help extract relevant features from the input audio.
   - Batch Normalization(BN): The output of the 1D-CNN layer will be passed through batch normalization. BN will normalize the output vector by using the mean and variance from the current batch. It will add stability and speed to the gradient descent process during model training.
   - Parametric ReLU: The normalized values from the BN layer will then be passed through the Parametric ReLU (PReLU) activation function. PReLU

will introduce non-linearity to the model by applying a rectified linear function with learnable parameters.

- Residual Learning: The output of the PReLU function will be added to the input of the residual block, creating a residual learning mechanism. This will allow the model to learn residual information and make improvements based on the difference between the input and output.

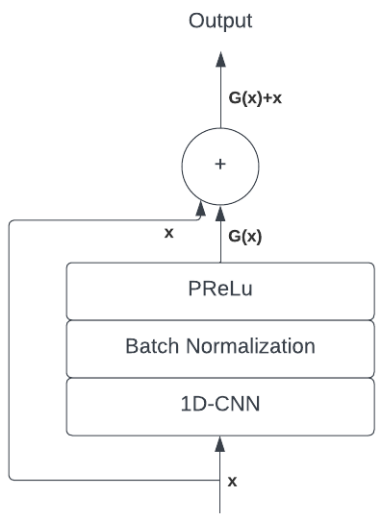


Figure 3.4: Diagram of Residual Learning

- RNN Layers: The output of the last residual block will serve as input to the stacked RNN layers (mainly BiLSTM). RNN will be a neural network that will use the previous time step's output as input for the current time step. RNNs will be suitable for sequential data and will be able to capture contextual information over time. The RNN layers will provide multiple levels of feature abstraction to generalize patterns in the data. Mainly, Bidirectional Long Short-Term Memory (BiLSTM) will be a variant of RNN used in our model. It will process the input sequence in both forward and backward directions simultaneously, capturing information from past and future contexts.

  Multiple BiLSTM layers can be stacked together to capture complex dependencies. It will enhance the ASR model's ability to understand and transcribe speech data accurately.
- Dense Layers and Softmax Output: The output of RNN layers (i.e., BiL-

STM layers) will be fed into the dense layers of the neural network, which will further process the features learned by RNNs. Finally, a softmax layer will be applied to obtain a probability distribution over the 66 unique characters. The softmax output can be represented as: Softmax_output = Softmax(Dense(RNN_output))

- CTC(Connectionist Temporal Classification) Loss: The CTC loss function will be used to compare the softmax output with the targeted transcriptions. The CTC loss will help handle the alignment problem between input audio and output characters. It will compute an alignment-free loss value using a blank token introduced during training and inference. The CTC loss can be calculated as: CTC_loss = -log(p($Y|X$)) , where p($Y|X$) will represent the probability of the target transcription Y given the input audio X.

  And, The objective function(i.e Probability p($Y|X$) will be the sum of all possible valid sequences. Mathematically,

  $$\mathrm{p}(Y|X)=\Sigma_{A \in A x,y} * \Pi^{\mathrm{T}}_{\mathrm{t=1}} \mathrm{p}_{\mathrm{t}} (a_{\mathrm{t}}|X) \qquad (2)$$

  where Ax,y will be the valid alignment of Y given X.

5. Decoding Algorithm: During prediction, the softmax outputs will need to be decoded to obtain a sequence of characters. The CTC beam search decoding method will be used in our model. CTC Beam Search will consider multiple alignments at each time step and will find the most probable output sequence. The process will involve tokenization, beam initialization, expansion, scoring, and pruning to select the final sequence. CTC Beam Search will effectively address the challenge of aligning audio inputs with output characters, resulting in accurate and contextually coherent transcriptions.

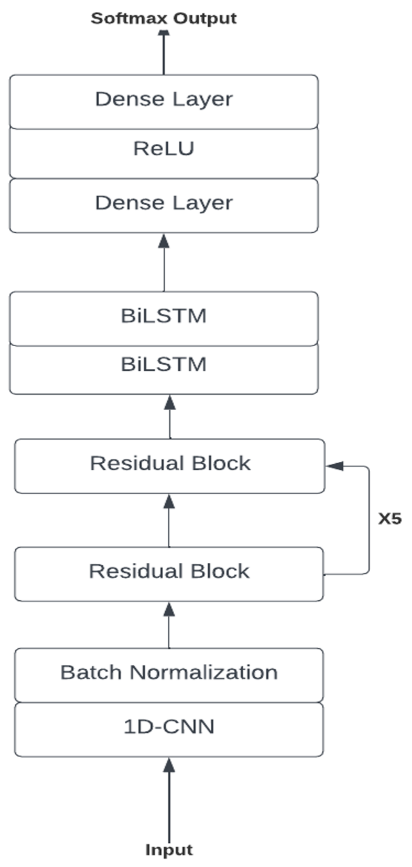


Figure 3.5: Diagram of ASR Model

6. Transcription Output:At the last stage of ASR modeling, the output text of the given speech/voice input will be obtained.
7. Language Translation and Commanding: We will utilize the Nepali text received from ASR (Automatic Speech Recognition) as input for Google Text To Speech (GTTS) to translate it into English. This translation process will be facilitated by the GTTS API, which we will employ for this purpose.
8. Semantic analysis and command mapping: Different expressions can be used to convey the same meaning. In this case, we will utilize the semantic analysis functionality of the Google API to associate those variations with specific actions. For example, commands such as ”open YouTube,” ”play music,” or ”perform calculations” will be mapped to their respective tasks using the semantic analysis feature.

3.2 UML Diagrams

Use Case Diagram

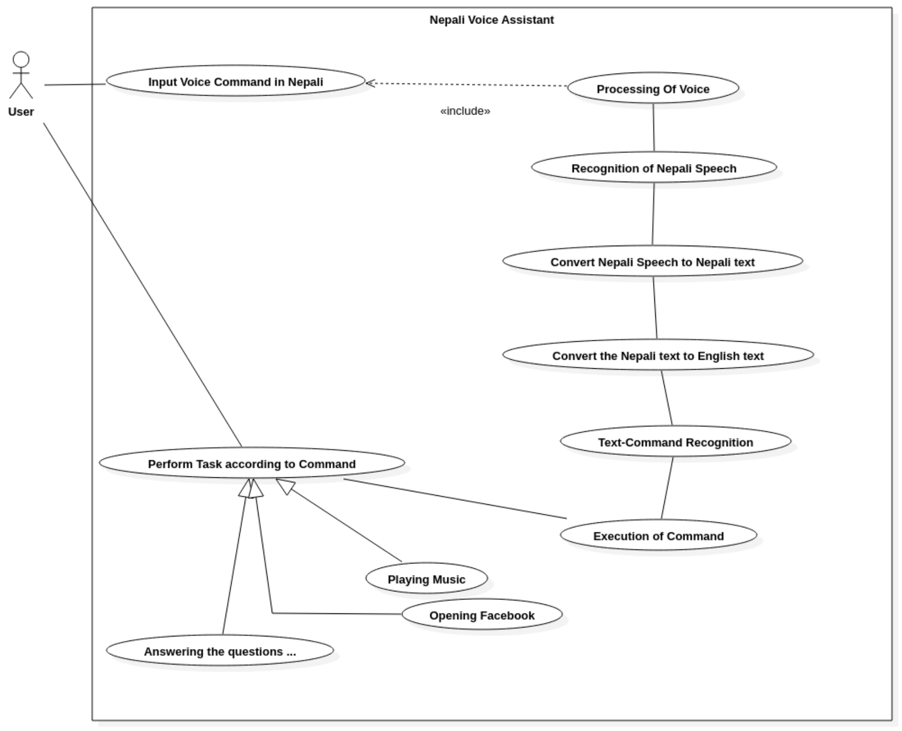

Nepali Voice Assistant
User
Input Voice Command in Nepali
«include»
Processing Of Voice
Recognition of Nepali Speech
Convert Nepali Speech to Nepali text
Convert the Nepali text to English text
Text-Command Recognition
Perform Task according to Command
Execution of Command
Playing Music
Opening Facebook
Answering the questions ...

Figure 3.6: Use Case Model of the Proposed System

Activity Diagram

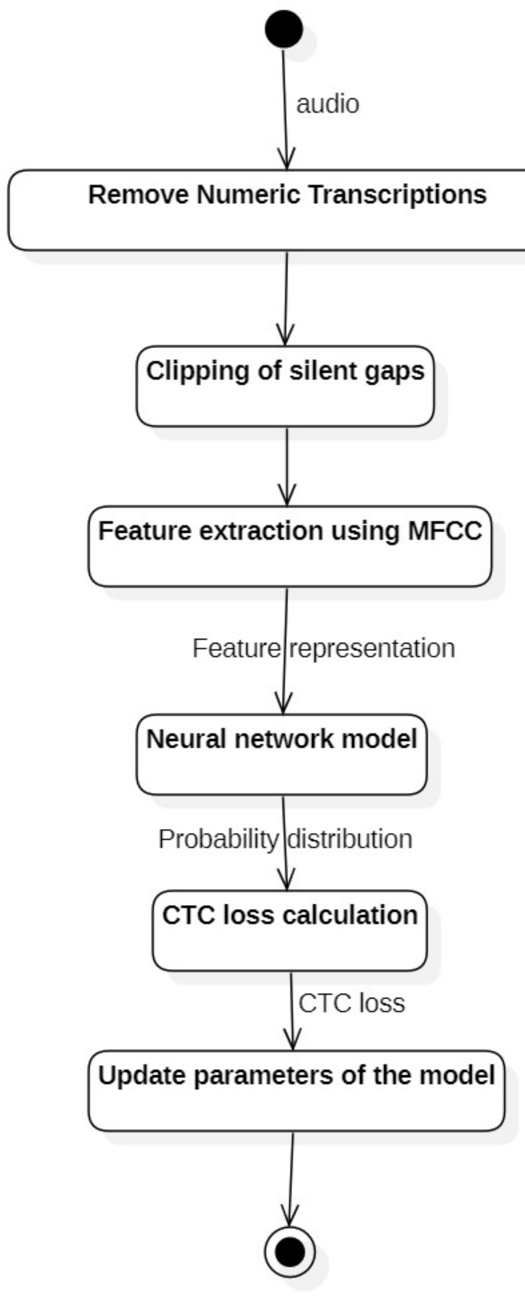


Figure 3.7: Activity Diagram of the Proposed System

3.3 Software Development Model

The incremental model is one of the easiest software development life cycle models to implement. It is suitable for scenarios where the initial or core software requirements are clearly defined, but the full set of features for the project is unknown. Additionally, the development company may choose not to provide the complete functionality of the software at once, but instead opt for periodic updates. Alternatively, clients may request functionality enhancements during the development process. In such cases, the incremental model is utilized.

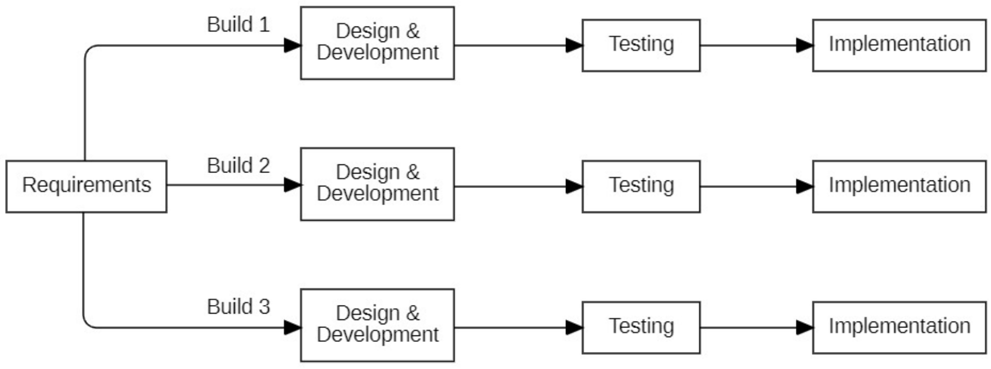


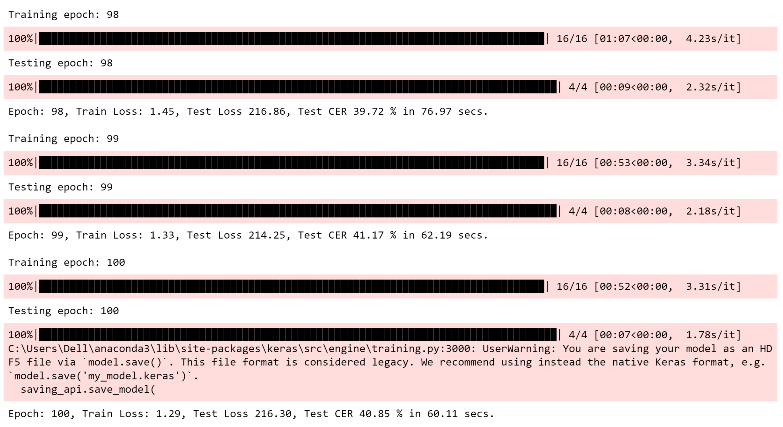
Figure 3.8: Incremental Model

- Build 1: In the initial phase, we proceed to utilize the voice input . This input is then processed using Automatic Speech Recognition (ASR) technology to convert it into text written in Nepali, conveying the equivalent meaning.
- Build 2: In the subsequent phase, we will take the Nepali text obtained as input from ASR and proceed to convert it into English text using the Google API. Additionally, we employ the Google API's semantic analysis feature to accurately associate the command with the corresponding task.
- Build 3: For the final version, we will create a desktop interface that includes the functionality to accept user voice as input for our system. Additionally, the interface will display the corresponding translated text for the user's input.

CHAPTER 4
EPILOUGE

4.1 Work Completed

We have implemented the conversion to MFCCs coefficient created by removing silent gaps and removing the unwanted devnagari transcription. Furthermore, we have written the code to implement the ASR model and Residual Block. The dataset we are using to train is taking excessive amount of time due to which we have just taken a sample of 44 audio files with epoch 100, the corresponding transcription to see the working of the model, plus taking 80% of dataset for training and 20% for test set.



```
Training epoch: 98
100%|████████████████████████████████████████████████████████████████████████| 16/16 [01:07<00:00,  4.23s/it]
Testing epoch: 98
100%|██████████████████████████████████████████████████████████████████████████| 4/4 [00:09<00:00,  2.32s/it]
Epoch: 98, Train Loss: 1.45, Test Loss 216.86, Test CER 39.72 % in 76.97 secs.
Training epoch: 99
100%|████████████████████████████████████████████████████████████████████████| 16/16 [00:53<00:00,  3.34s/it]
Testing epoch: 99
100%|██████████████████████████████████████████████████████████████████████████| 4/4 [00:08<00:00,  2.18s/it]
Epoch: 99, Train Loss: 1.33, Test Loss 214.25, Test CER 41.17 % in 62.19 secs.
Training epoch: 100
100%|████████████████████████████████████████████████████████████████████████| 16/16 [00:52<00:00,  3.31s/it]
Testing epoch: 100
100%|██████████████████████████████████████████████████████████████████████████| 4/4 [00:07<00:00,  1.78s/it]
C:\Users\Dell\anaconda3\lib\site-packages\keras\src\engine\training.py:3000: UserWarning: You are saving your model as an HD
F5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')`.
  saving_api.save_model(
Epoch: 100, Train Loss: 1.29, Test Loss 216.30, Test CER 40.85 % in 60.11 secs.
```

Figure 4.1: Training on the model

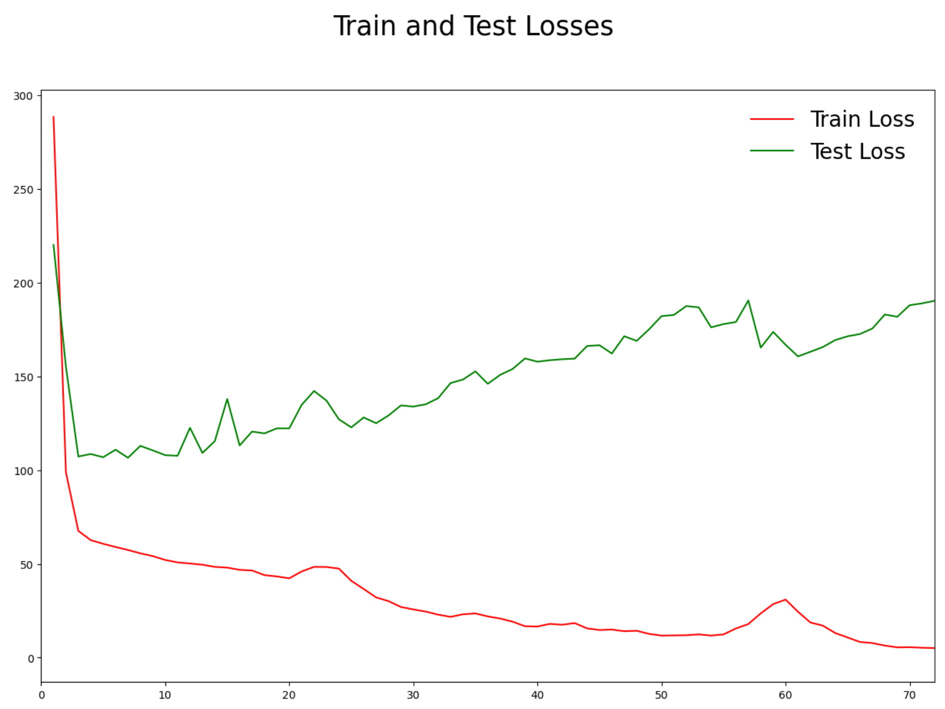
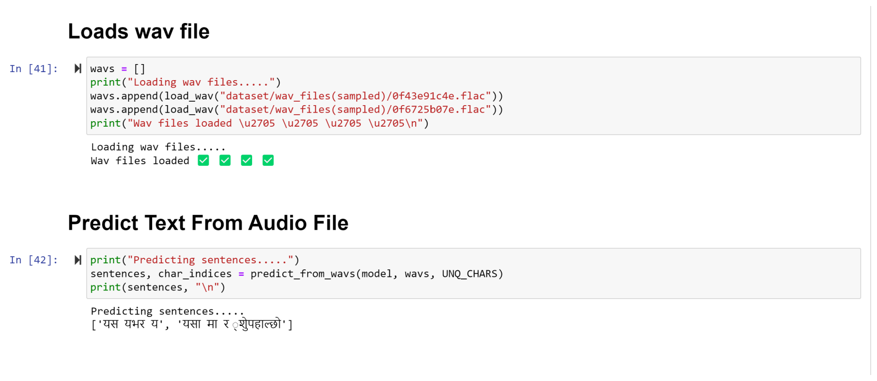


Figure 4.2: Train and Test Loss



Loads wav file

```
In [41]:  wavs = []
          print("Loading wav files.....")
          wavs.append(load_wav("dataset/wav_files(sampled)/0f43e91c4e.flac"))
          wavs.append(load_wav("dataset/wav_files(sampled)/0f6725b07e.flac"))
          print("Wav files loaded \u2705 \u2705 \u2705 \u2705\n")

          Loading wav files.....
          Wav files loaded ✅ ✅ ✅ ✅
```

Predict Text From Audio File

```
In [42]:  print("Predicting sentences.....")
          sentences, char_indices = predict_from_wavs(model, wavs, UNQ_CHARS)
          print(sentences, "\n")

          Predicting sentences.....
          ['यस यभर य', 'यसा मा र ्शेपहाल्छो']
```

Figure 4.3: Predicting Audio Output

4.2 Work Remaining

The Build 2 and Build 3 part are the remaining part, plus the training with much larger dataset of 148k(audio,text) to improve the accuracy must be done with proper deployment of the model.

REFERENCES

[1] B. Bhatta, B. Joshi, and R. K. Maharjhan, “Nepali speech recognition using cnn, gru and ctc,” in *Proceedings of the 32nd Conference on Computational Linguistics and Speech Processing (ROCLING 2020)*, 2020, pp. 238–246.

[2] B. Joshi, B. Bhatta, S. P. Panday, and R. K. Maharjan, “A novel deep learning based nepali speech recognition,” in *Innovations in Electrical and Electronic Engineering: Proceedings of ICEEE 2022, Volume 2*. Springer, 2022, pp. 433–443.

[3] M. Dhakal, A. Chhetri, A. K. Gupta, P. Lamichhane, S. Pandey, and S. Shakya, “Automatic speech recognition for the nepali language using cnn, bidirectional lstm and resnet,” in *2022 International Conference on Inventive Computation Technologies (ICICT)*. IEEE, 2022, pp. 515–521.

[4] R. Shrestha, B. Joshi, and S. Sharma, “Nepali speech recognition using lstm-ctc,” 2021.

[5] S. Subhash, P. N. Srivatsa, S. Siddesh, A. Ullas, and B. Santhosh, “Artificial intelligence-based voice assistant,” in *2020 Fourth world conference on smart trends in systems, security and sustainability (WorldS4)*. IEEE, 2020, pp. 593–596.

[6] T. C. for Asian Studies, “Tibet himalaya initiative,” https://www.colorado.edu/tibethimalayainitiative/Resources-and-Partners, 2016.

[7] Kaldi, “kaldi-asr/kaldi,” https://github.com/kaldi-asr/kaldi, 2011.

[8] K. Sodimana, K. Pipatsrisawat, L. Ha, M. Jansche, O. Kjartansson, P. D. Silva, and S. Sarin, “A Step-by-Step Process for Building TTS Voices Using Open Source Data and Framework for Bangla, Javanese, Khmer, Nepali, Sinhala, and Sundanese,” in *Proc. The 6th Intl. Workshop on Spoken Language Technologies for Under-Resourced Languages (SLTU)*, Gurugram, India, Aug. 2018, pp. 66–70. [Online]. Available: http://dx.doi.org/10.21437/SLTU.2018-14

[9] O. Kjartansson, S. Sarin, K. Pipatsrisawat, M. Jansche, and L. Ha, “Crowd-Sourced Speech Corpora for Javanese, Sundanese, Sinhala, Nepali, and

Bangladeshi Bengali," in *Proc. The 6th Intl. Workshop on Spoken Language Technologies for Under-Resourced Languages (SLTU)*, Gurugram, India, Aug. 2018, pp. 52–55. [Online]. Available: http://dx.doi.org/10.21437/SLTU.2018-11