Minor Project Report on

SEQUENCE-TO-SEQUENCE NEPALI ASR WITH MFCC FEATURES AND LSTM-CTC MODEL

In partial fulfillment for the award of the degree of

Bachelor in Computer Engineering

Submitted By:

Hina Tamrakar [22070491]

Kajal Kumari Kushwaha [22070492]

Rabindra Yadav [22070498]

Shahil Shrestha [22070504]

Submitted to:

Department of Computer, Software and IT Engineering

Everest Engineering College

Sanepa-2, Lalitpur

July 13, 2025

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to all those who supported and contributed to the successful completion of this project. We are deeply thankful to our supervisors, Er. Pradip Paudel and Er. Narayan Sapkota, for their invaluable guidance, continuous encouragement, and constructive feedback throughout the development of this work.

We extend our sincere appreciation to Er. Shailesh Pandey and Er. Manish Karn for their valuable insights and suggestions, which greatly enriched our project. We are also grateful to the Department of Computer, Software and IT Engineering at Everest Engineering College, Sanepa-2, Lalitpur, for providing the necessary facilities and resources.

Finally, we extend our thanks to our peers for their support and collaboration throughout the course of this project.

This project was completed by the following team members:

Hina Tamrakar [22070491]

Kajal Kumari Kushwaha [22070492]

Rabindra Yadav [22070498]

Shahil Shrestha [22070504]

ABSTRACT

Nepali is a low-resource language that lacks large-scale annotated datasets and robust speech recognition tools. This creates a barrier to digital accessibility and communication for millions of native speakers that are not accustomed to digital systems. To address this issue, this project aims to develop a custom speech-to-text system for the Nepali language using BiLSTM networks and CTC loss function.

The system is designed to learn temporal acoustic patterns from Nepali speech using MFCCs as input features. The BiLSTM architecture then effectively captures long-range dependencies in audio signals. The CTC loss function enables alignment-free training between variable-length input speech and output text sequences. A dataset of 4,000 Nepali audio samples (about 5.5 hours of speech) with corresponding transcripts was collected and used for training and evaluation. The performance of the system was measured using CER, and the model achieved a CER of 0.009, indicating high transcription accuracy. The developed system demonstrates the potential for advancing ASR capabilities in underrepresented languages like Nepali and contributes valuable resources for future research in Nepali language technology.

Keywords: *Nepali ASR, Speech Recognition, BiLSTM, CTC, Pytorch, Torchaudio, NLP*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI – Artificial Intelligence

ASR – Automatic Speech Recognition

BiLSTM – Bi-Directional Long Short-Term Memory

CER – Character Error Rate

CNN – Convolutional Neural Network

CTC – Connectionist Temporal Classification

DCT – Discrete Cosine Transform

DNN – Deep Neural Network

FFT – Fast Fourier Transform

GMM – Gaussian Mixture Model

GRU – Gated Recurrent Unit

HMM – Hidden Markov Model

IPA – International Phonetic Alphabet

LSTM – Long Short-Term Memory

MFCC – Mel Frequency Cepstral Coefficients

ReLU – Rectified Linear Unit

RNN – Recurrent Neural Network

TTS – Text To Speech

VAD – Voice Activity Detection

WER – Word Error Rate

ZWJ – Zero Width Joiner

ZWNJ – Zero Width Non-Joiner

Chapter 1: INTRODUCTION

In recent years, speech recognition technology has rapidly transformed the way humans interact with digital systems. The world has seen significant advancements in ASR for global languages like English. However, low resource languages like Nepali remains under-represented due to the lack of high-quality speech datasets. To address this gap, this project seeks to develop an ASR model tailored for the Nepali language. It leverages MFCCs for feature extraction, a BiLSTM network for temporal modeling, and the CTC loss function for training. A collection of publicly available datasets is used to train the model. Ultimately, this project aspires to advance speech technology for Nepali and contribute to the growing field of low-resource language processing.

1.1.Background

With the growth of AI and machine learning, ASR systems have become increasingly accurate for resource-rich languages. However, the Nepali language remains a low-resource language with limited speech datasets and pretrained models. This limitation in Nepali ASR systems affects especially those people who cannot easily type or read, such as the elderly or the disabled people.

At present, an implementation of Nepali speech recognition using LSTM and CTC showed an average of 9.62% WER after training over 30 epochs [1]. This showed a significant potential for Nepali ASR despite the limited number of speech resources recorded from three male speakers. But, although it achieved promising results, the performance varied significantly depending on audio quality, clarity, and speaker variability.

Building on this foundation, our project introduces a speech-to-text system for Nepali, leveraging the BiLSTM architecture. The dataset used is publicly available, and audio is processed using MFCC which is fed into a BiLSTM model trained with CTC loss. This approach aims to improve the generalization, paving the way for better accessibility tools and further research in speech technology for underrepresented languages.

1.2.Problem Statement

In the context of Nepali ASR, there is a significant lack of publicly available datasets and language-specific models. The existing ASR systems are often based on multilingual or

generic architectures that fail to capture the unique phonetic and linguistic features of Nepali, resulting in poor performance and limited applicability. This technological gap restricts access to voice-enabled services in critical areas such as education, accessibility, and digital communication for Nepali-speaking communities. Therefore, there is a pressing need to develop a speech recognition system specifically designed for the Nepali language. Our project focuses on developing a Nepali ASR which is supported by modern deep learning techniques such as the BiLSTM model. By building an end-to-end ASR model tailored to the Nepali language, this project seeks to bridge the digital divide and enable inclusive access to voice-driven technologies.

1.3. Objective

The main objective of this project is:

- To build a model using BiLSTM architecture with CTC loss function for Nepali automatic speech recognition.

1.4. Scope

This project focuses on developing a Nepali ASR model for native speakers using publicly available datasets. It involves MFCC-based feature extraction and training a BiLSTM model with CTC loss using PyTorch. The scope is limited to speech-to-text transcription and does not include translation, speaker identification, or real-time deployment. The goal is to deliver a functional prototype within a controlled development environment.

1.5. Applications

Its areas of applications include:

- Education: It enables automatic transcription of lectures and supports voice-based learning tools.
- Accessibility: It assists visually impaired and physically challenged users through voice-controlled interaction.
- Language Technology: It lays the foundation for integrating Nepali ASR into digital assistants, captioning systems, and other voice-based tools.
- Linguistics and Research: It contributes to the Nepali linguistics sector by collecting and analyzing spoken data for language preservation and research

Chapter 2: LITERATURE REVIEW

Early speech recognition systems used statistical models like HMM and GMM. However, these models struggled with capturing temporal dependencies. With the rise of deep learning, LSTM and GRU networks improved sequence modeling in ASR systems. A notable contribution in Nepali ASR system [1] used LSTM networks with CTC loss. This system trained on 2,813 samples, achieving a WER of 40% for isolated words without CTC and as low as 9.62% with CTC after 30 epochs. While effective, the system faced limitations such as limited data and speaker variation, highlighting the need for larger, more balanced datasets and better noise handling. Our project builds upon this approach by focusing on continuous speech recognition using a BiLSTM-CTC architecture.

Another implementation used a hybrid model combining 1D-CNN, ResNet, and BiLSTM [2]. It was trained on a cleaned version of the OpenSLR dataset with initial 157,905 audio clips from 527 speakers reducing to 148,188 audio clips (143.6 hours). Using MFCC features and CTC loss, their best model achieved a 17.06% CER, outperforming simpler architectures. However, it faced challenges such as noisy data, limited speaker diversity, and decoding inefficiencies in real-world settings. While this model shows promise, our project differs by focusing on a streamlined BiLSTM-CTC architecture to better capture native speech patterns in a controlled environment.

The Nepali ASR model in [3] implemented an RNN-CTC model using a 67-character Nepali set for transcription. The model, trained on just two hours of audio from three male speakers, achieved a CER of 0.34% with a unigram language model. However, performance dropped to 0.52% on unseen speakers, revealing limitations in generalization and speaker dependence. While beam search decoding helped improve output coherence, the study highlights key challenges such as limited dataset size and diversity. Our project builds on this work by adopting a similar MFCC, BiLSTM and CTC architecture but addresses its limitations through a larger dataset and improved training conditions to enhance robustness and speaker-independence.

A model developed in [4] used a CNN-GRU-CTC architecture for Nepali ASR, with MFCCs for feature extraction, a 1D CNN for spatial feature learning, and a GRU network to model temporal dependencies, with CTC for alignment-free decoding. Trained on a clean, high-quality TTS dataset, the model achieved an impressive CER of 1.84%,

outperforming earlier RNN-CTC models. However, the model's strong performance was limited to clean, single-speaker data, making it less effective in noisy or multi-speaker environments. In contrast, our project focuses on training using a BiLSTM-CTC setup to better handle speaker and acoustic variability.

An end-to-end Nepali ASR system using MFCC features, CNN-GRU layers, and CTC decoding was developed in [5], that also included a language model. This model focused on converting Nepali speech to text without requiring phoneme-level alignments achieving WERs of 49.85%, 46.39%, and 52.89% for training, validation, and testing, respectively without using a language model. By integrating the uni-gram language model, the performance further improved with WERs of 35.40%, 37.50% and 39.72% on train, validation, and test data respectively. However, the model was tested in controlled, noise-free conditions, limiting real-world applicability. In contrast, our BiLSTM-CTC model captures bidirectional context more effectively and demonstrates greater robustness in handling natural Nepali speech and complex Devanagari script.

The Nepali ASR system in [6] leveraged MFCC features and a deep neural architecture combining CNN, ResNet, and BiLSTM layers to recognize and transcribe spoken Nepali. MFCCs captured essential spectral characteristics, while CNNs extracted local patterns, ResNet captured deep hierarchical features, and BiLSTM modeled temporal dependencies. CTC loss was used to align audio features with corresponding text without requiring pre-aligned data. The model, trained on approximately 157,000 audio samples with 1.55 million parameters over 47 epochs, achieved a character accuracy of 82.02%. However, the system faced limitations including moderate accuracy, potential lack of speaker diversity, absence of a language model or beam search for improved decoding, and untested robustness in noisy or real-world environments.

A study traced the shift from traditional HMM-GMM models to advanced architectures like DNNs, CNN-RNNs, and Transformers [7]. They noted that models such as CNN-GRU-CTC [4] and joint CTC-attention system [8] perform well depending on dataset quality and model design. Key challenges identified include the lack of diverse, balanced speech datasets, limited use of advanced language models, and the absence of open-source pre-trained models. The study also emphasized the gap between academic research and practical deployment in noisy, multi-speaker settings. Our project addresses some of these gaps by focusing on implementing a robust BiLSTM-CTC model.

One paper introduced a novel end-to-end approach to speech recognition RNNs, trained with CTC [9]. Unlike traditional systems that relied on HMM and pre-aligned data, their method allowed the model to learn both acoustic features and alignments directly from raw audio and transcriptions. Using BiLSTM networks, they achieved state-of-the-art results on the TIMIT dataset. However, the approach faced limitations in scaling to large-vocabulary tasks, required external decoding techniques like beam search for better accuracy, and posed computational challenges. This study laid the foundation for future end-to-end speech recognition models while highlighting the need for improved decoding and scalability.

An end-to-end ASR model for the endangered Tujia language used sample-based transfer learning from Mandarin [10]. By building a cross-lingual IPA-aligned corpus to address the lack of annotated Tujia data, and employing CNN for spatial feature extraction and BiLSTM networks for modeling temporal acoustic dependencies with CTC for alignment free recognition, the model achieved a 46.19% recognition error rate which was 2.11% better than training on Tujia alone. This demonstrated the benefits of cross-language training. However, the reliance on Mandarin limited phonetic generalization and resulted in a relatively high error rate. In contrast, our project uses a BiLSTM-CTC model trained directly on native Nepali speech, showing the strength of language-specific modeling.

The impact of Mel filter bank design on the MFCC extraction from resampled speech is explored in [11]. This is crucial for ASR systems operating at different sampling rates. The authors evaluate various strategies for adapting the filter bank when speech is down sampled, like from 16kHz to 8kHz. Experiments show that using a filter bank designed for the target sampling rate significantly improves MFCC consistency and recognition accuracy. The study underscores the importance of aligning preprocessing steps with feature extraction. A key limitation is the focus solely on MFCCs, without exploring other feature types or models.

In summary, while Nepali ASR research has produced several strong models with competitive error rates, their performance often depends heavily on dataset quality and controlled environments. Future progress hinges on developing richer datasets, improving language modeling, and making trained models openly available to the research community.

Chapter 3: METHODOLOGY

The methodology of this project outlines the overall approach used to develop the Nepali ASR system. It involves a series of steps including data collection and preprocessing, feature extraction, model design, training, and evaluation. The goal is to build an end-to-end system capable of accurately converting spoken Nepali into text. By following a structured and systematic process, the methodology ensures that each phase contributes effectively to the development of a functional ASR model tailored for the Nepali language.

3.1. Block Diagram

The model is designed with a sequential architecture that combines MFCC feature extraction, a BiLSTM network, and a CTC-based decoding mechanism. Figure 3.1 illustrates the basic system architecture.

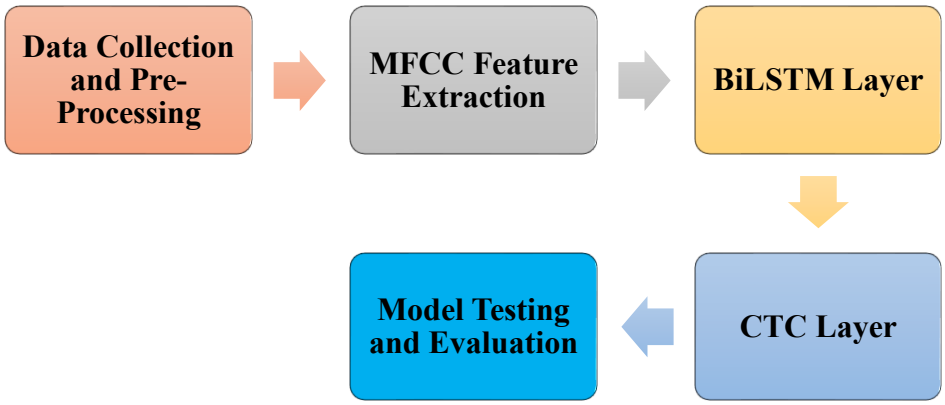

Figure 3.1: System Architecture

3.1.1. Data Collection and Preprocessing

The data collection involves obtaining the dataset from OpenSLR [12] which is publicly available. These datasets contain 4000 audio clips, with each clip being 5 seconds long. The audio clips are converted to mono channel, and resampled to 16kHz for simpler processing and sufficient clarity. Along with the audio clips, the transcription is also obtained and filtered out to get a .tsv file that has the transcriptions of the audio used. The collected data is then split into training and test sets in an 80-20 ratio for model training and evaluating performance metrics.

3.1.2. MFCC Feature Extraction

MFCCs are features that capture the important spectral features of speech by mimicking human auditory perception. It follows a series of steps involving pre-emphasis, framing, windowing, Fourier transform, Mel filter banks, and cepstral transformation. The Figure 3.2 shows the working of the MFCC extraction.

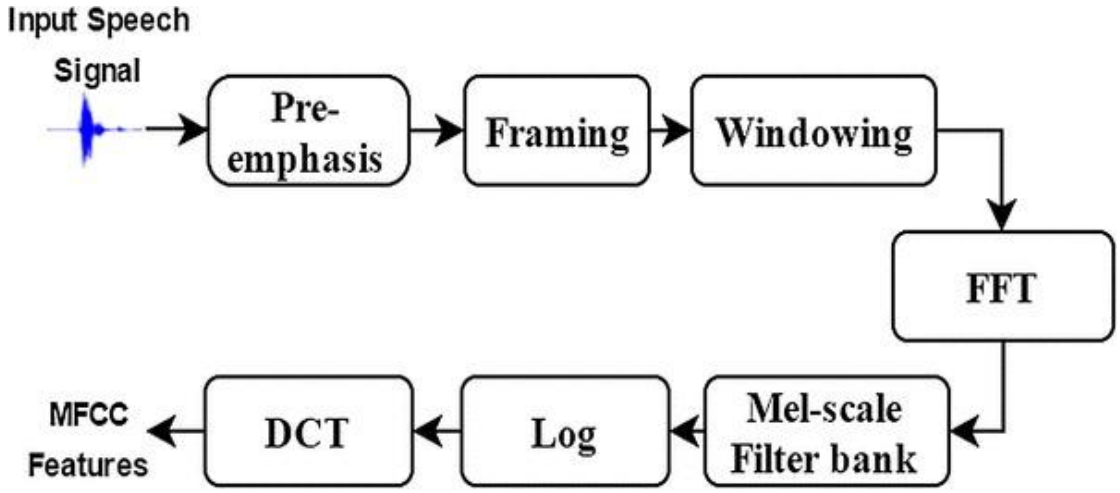

Figure 3.2: MFCC Architecture [13]

3.1.2.1.Pre-Emphasis

This step applies a high-pass filter to the audio signal to boost high-frequency components. It balances the frequency spectrum and improves the clarity of important speech features, especially in noisy environments.

$$y[n] = x[n] - \alpha \, . x[n-1] \tag{3.1}$$

where, x[n] is the input signal,

y[n] is the output signal,

α is the pre-emphasis coefficient typically between 0.95 and 0.97.

3.1.2.2.Framing

Speech signals are not stationary over long periods, so the signal is divided into short overlapping frames. This allows each segment to be treated as approximately stationary for further frequency analysis. Overlapping ensures smooth transitions and reduces information loss at frame edges.

3.1.2.3.Windowing

Each frame is multiplied by a Hamming window to minimize spectral leakage. This smooths the signal at the edges of each frame, reducing discontinuities that can introduce noise into the frequency analysis.

$$w[n] = 0.54 - 0.46 \,.\cos\left(\frac{2\pi n}{N-1}\right) \tag{3.2}$$

$$x_w[n] = x[n] \,.\, w[n] \tag{3.3}$$

where, w[n] is the Hamming window function,

$x_w[n]$ is the windowed signal,

N is the number of samples in a frame.

3.1.2.4.FFT

FFT is used to convert each time-domain frame into the frequency domain. This reveals how much energy is present at different frequency components in the speech signal, forming the basis for the Mel-scale analysis.

$$X[k] = \sum_{n=0}^{N-1} x_w[n].\, e^{-\frac{j2\pi kn}{N}} \tag{3.4}$$

$$P[k] = \frac{1}{N}\, |X[k]|^2 \tag{3.5}$$

where, X[k] is the FFT output,

P[k] is the power spectrum,

N is the number of FFT points.

3.1.2.5.Mel Filter Bank

The FFT output is passed through a set of triangular filters spaced on the Mel scale, which mimics the human ear's frequency sensitivity. This emphasizes important frequencies in the lower range, and compresses less important higher frequencies.

$$Mel\,(f) = 2595 \,.\log_{10}\left(1 + \frac{f}{700}\right) \tag{3.6}$$

$$S_m = \sum_{k=f_{m-1}}^{f_{m+1}} P[k] \,.\, H_m[k] \tag{3.7}$$

where, f is the frequency,

S_m is the energy for Mel filter m,

$H_m[k]$ is the triangular filter.

3.1.2.6.Logarithm

A logarithmic function is applied to the Mel filter bank energies to mimic human loudness perception. It also transforms multiplicative effects into additive ones, making the features easier to model and analyze.

$$\log S_m = \log(S_m) \tag{3.8}$$

5.1.2.7.DCT

DCT decorrelates the log Mel energies and concentrates the most useful information in the first few coefficients. These resulting values are the MFCCs, which effectively summarize the spectral shape of each speech frame.

$$c_n = \sum_{m=1}^{M} \log(S_m) \cdot \cos\left[\frac{\pi n}{M} \cdot (m - 0.5)\right] \tag{3.9}$$

where, n = 1, 2, …, L,

c_n is the MFCC coefficient n,

L is the number of MFCCs to retain,

M is the number of Mel filters.

3.1.3. BiLSTM Layer

The diagram in Figure 3.3 shows the internal workings of BiLSTM. BiLSTM is an extension of traditional LSTM network that, unlike conventional LSTMs, allow information to flow in both forward and backward directions. This allows them to capture more contextual information.

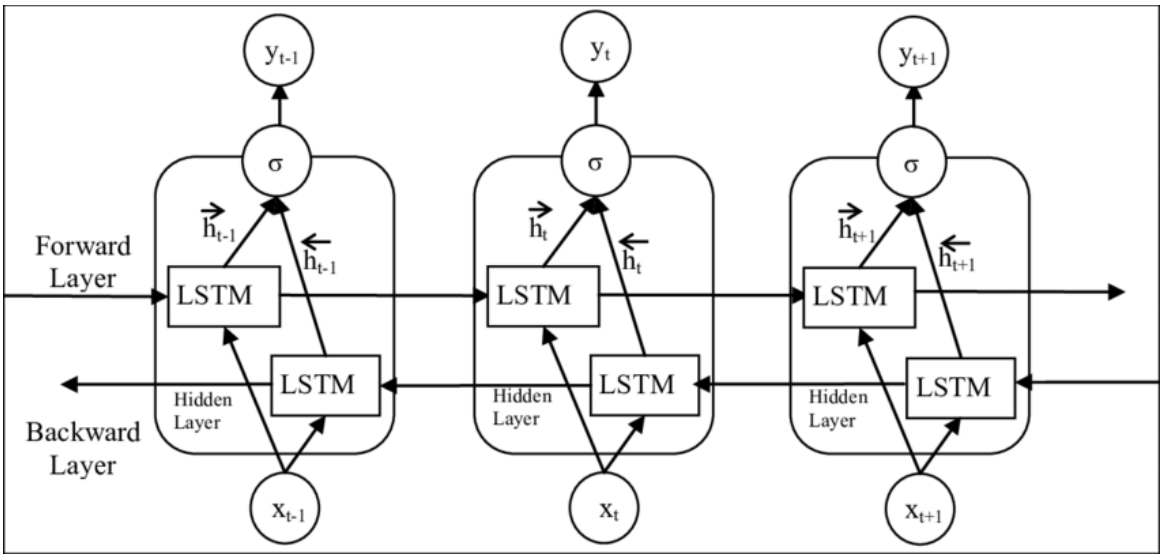

Figure 3.3: BiLSTM Architecture [14]

The BiLSTM network is used as the core of the acoustic model. When an audio signal is converted into a sequence of MFCC feature vectors, each vector represents a small slice of sound or a short window of the audio. To understand what sound or character this vector represents, it is not enough to look at it in isolation as speech is highly contextual, especially in languages like Nepali. The BiLSTM addresses this by using two LSTM layers: the forward LSTM and the backward LSTM.

An LSTM processes sequences by maintaining a cell state and gates to control the flow of information. This enables it to learn what to remember, what to forget and what to output at each time step. The gates include the forget gate, the input gate and the output gate.

The forget gate decides what part of the cell state to forget. It takes previous hidden state and current input and outputs a value between 0 and 1 for each part of the cell state. 0 means "completely forget" and 1 means "completely keep". The equation of forget gate is:

$$f_t = \sigma\left(W_f \,.\, [h_{t-1}, X_t] + b_f\right) f_t \tag{3.10}$$

where, W_f represents the weight matrix associated with the forget gate.

$[h_{t-1}, X_t]$ denotes the concatenation of current input and previous hidden state.

b_f is the bias with the forget gate.

σ is the sigmoid activation function.

The input gate decides how much of the new input should be added to the memory. The cell state is updated by combining the old memory and the new candidate values. The equation of input gate is:

$$i_t = \sigma\left(W_i \,.\, [h_{t-1}, X_t] + b_i\right) i_t \tag{3.11}$$

$$\hat{C}_t = \tanh\left(W_c \,.\, [h_{t-1}, X_t] + b_c\right) \tag{3.12}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \hat{C}_t \tag{3.13}$$

where, ⊙ denotes element-wise multiplication

tanh is activation function

The output gate controls how much of the memory should be passed to the next step. The updated cell state is passed through a tanh function and then multiplied by the output gate value. The equation of output gate is:

$$o_t = \sigma\left(W_o \,.\, [h_{t-1}, x_t] + b_o\right) \tag{3.14}$$

$$h_t = o_t \,.\, \tanh(C_t) \tag{3.15}$$

At each time step, the LSTM uses these gates to selectively update and carry forward useful information.

In BiLSTM, the forward LSTM processes the sequence from the beginning to the end of the audio (i.e. t = 1 → T), learning how the current sound depends on previous ones. The backward LSTM works in reverse by processing the sequence from the end to the beginning (t = T → 1), learning how the current sound is influenced by future sounds. At each time step t, the BiLSTM combines both perspectives:

$$h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}] \tag{3.16}$$

This concatenation of forward ($\overrightarrow{h_t}$) and backward ($\overleftarrow{h_t}$) hidden states gives the model a full context window of both, what has been said and what is about to be said.

For example, in Nepali, the word "कला" (art) and "कल" (yesterday) start similarly, but the difference becomes clear only with the following sound. The BiLSTM sees both directions and makes a better-informed prediction for each time step. The result is that BiLSTM can predict the most likely character at each moment, even when the input sound is affected by surrounding phonemes.

3.1.4. CTC Layer

The CTC layer then enables learning from the audio and its transcription, without requiring manual alignment between the two. In speech, the number of input frames, are often much longer than the number of output labels which are characters or words, and the exact timing of each phoneme or letter is typically unknown. The CTC layer solves this alignment problem by introducing a flexible mechanism that allows the model to learn how sequences of input frames map to the correct output sequence.

To make this possible, CTC uses a special blank token alongside the original label set, which helps the model handle time steps that do not correspond directly to output characters. It also allows for repeated predictions of the same character. For example, the word "कलम" might be predicted as a sequence like "—कक—ल—मम—", where the dashes represent the blank token. After prediction, CTC applies two operations: collapsing repeated characters and removing blanks. This transforms the raw output into the final transcription. In this case, the "—कक—ल—मम—" becomes "कलम".

Internally, the CTC layer calculates a loss by summing the probabilities of all possible valid alignments between the input and the target transcription. This enables the model to learn the most probable mapping from input to output, regardless of timing. At the decoding stage, the model generates a sequence of character probabilities for each frame. From this, the most likely transcription is produced using beam search decoding, which considers multiple possible output sequences to improve accuracy. The figure in Figure 3.4 shows a simple CTC decoding that converts a sequence of characters into a human understandable format.

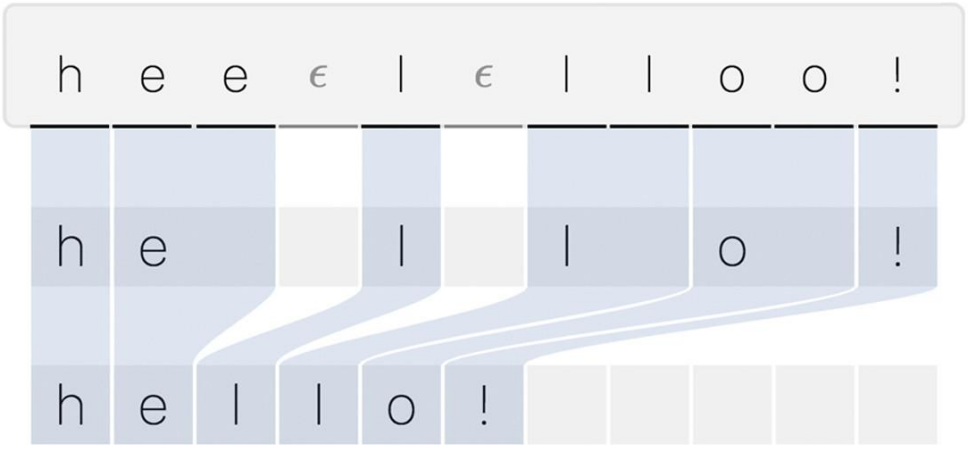

Figure 3.4: CTC Decoding [15]

3.1.5. Model Testing and Evaluation

The model testing and evaluation focuses on assessing how accurately the model transcribes unseen speech data. After training, the model is evaluated using a held-out test set containing audio files and their corresponding ground truth transcriptions. The performance is primarily measured using CER, which calculates the number of character insertions, deletions, and substitutions needed to convert the predicted transcription into the reference text, normalized by the total number of characters in the reference. CER is particularly useful for evaluating ASR systems for Nepali, where word boundaries and spelling consistency may vary, and fine-grained character-level accuracy is essential. A lower CER indicates better model performance.

3.2.System Workflow

The flowchart in Figure 3.5 illustrates the workflow of the Nepali ASR model. The process begins with audio input to the system which is pre-processed to prepare it for feature extraction. Then, the MFCC features are extracted and fed into the acoustic model. This model uses a BiLSTM network with CTC loss function. The model is trained on the training dataset after which it generates the text output.

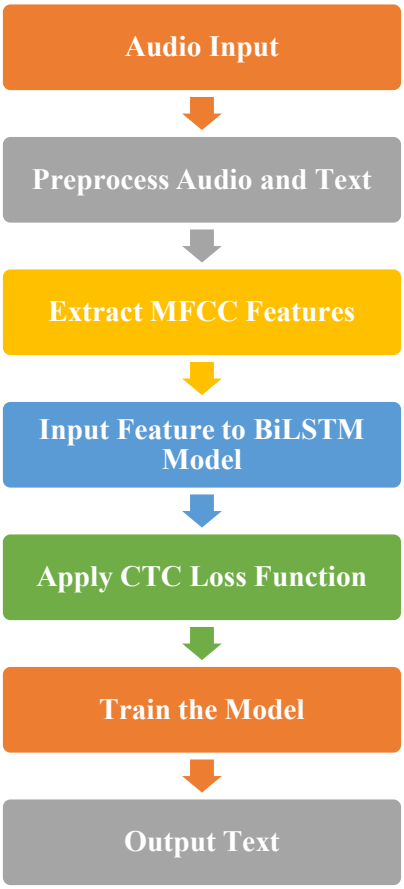

Figure 3.5: Flowchart of the System

3.2.1. Audio Input

A dataset consisting of 4000 Nepali audio files, with each audio clip being 5 seconds long, and their corresponding text transcriptions were uploaded to Google Drive. This included the publicly available datasets from OpenSLR as they were already preprocessed and could provide better results. The audio was mounted to Google Colab and loaded. These audio samples were divided into a batch size of 32, split into a ratio of 80-20 for training and testing and sent for preprocessing.

3.2.2. Audio and Text Preprocessing

The recorded audio files were preprocessed to prepare them for feature extraction. The audio was converted to mono channel and resampled if they weren't already in format. The audio samples were padded so all samples in the batch were of the same length, and then they were normalized. For the transcriptions, the length of each transcription was obtained. A vocabulary was defined for the system to map the audio to. This was then tokenized. The vocabulary consisted of the following 72 characters, along with a blank character and an unknown character:

‘ ’, ‘ँ’, ‘ं’, ‘ः’, ‘अ’, ‘आ’, ‘इ’, ‘ई’, ‘उ’, ‘ऊ’, ‘ऋ’, ‘ए’, ‘ऐ’, ‘ओ’, ‘औ’,

‘क’, ‘ख’, ‘ग’, ‘घ’, ‘ङ’, ‘च’, ‘छ’, ‘ज’, ‘झ’, ‘ञ’, ‘ट’, ‘ठ’, ‘ड’, ‘ढ’, ‘ण’,

‘त’, ‘थ’, ‘द’, ‘ध’, ‘न’, ‘प’, ‘फ’, ‘ब’, ‘भ’, ‘म’, ‘य’, ‘र’, ‘ल’, ‘व’, ‘श’, ‘ष’, ‘स’, ‘ह’,

‘०’, ‘१’, ‘२’, ‘३’, ‘४’, ‘५’, ‘६’, ‘७’, ‘८’, ‘९’,

‘ा’, ‘ि’, ‘ी’, ‘ु’, ‘ू’, ‘ृ’, ‘े’, ‘ै’, ‘ो’, ‘ौ’, ‘्’,

‘\u200c’, ‘\u200d’, ‘।’

The characters ‘\u200c’ and ‘\u200d’ represent ZWNJ and ZWJ respectively. They are invisible characters. ZWNJ prevents two characters from joining when they normally would, while ZWJ causes characters that are normally separated to be joined together. As Devnagari script forms conjunct characters, they help ensure no characters combine or separate when they should not.

3.2.3. MFCC Feature Extraction

The preprocessed audio was converted into MFCCs. MFCCs were chosen due to their effectiveness in capturing speech-relevant features while reducing noise and irrelevant frequency information. 13 MFCCs were extracted per frame where the first 12 coefficients capture the spectral features and the remaining 1 represents log energy. These feature vectors served as input to the acoustic model.

3.2.4. Acoustic Model

The acoustic model in this system was designed to convert sequences of audio features into sequences of textual representations. The architecture is illustrated in Figure 3.6 and described as follows:

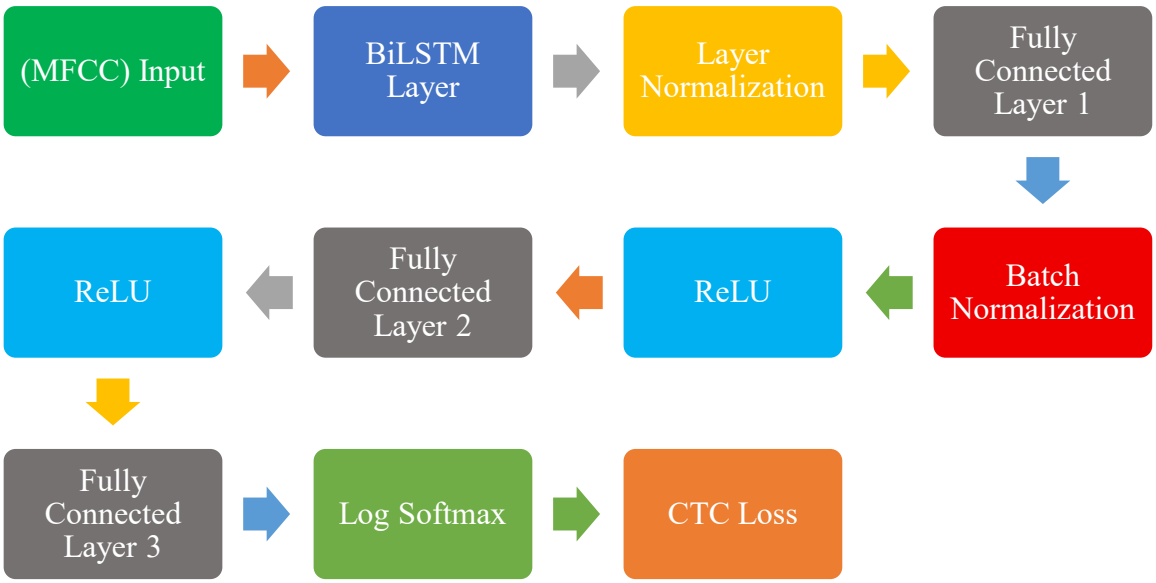

Figure 3.6: Acoustic Model Architecture for the System

3.2.4.1. BiLSTM Layer

The MFCCs were passed through a BiLSTM layer. This layer captured the temporal dependencies in the audio signal, making the model capable of understanding the context and sequence of spoken language.

3.2.4.2. Layer Normalization

The output of BiLSTM layer was normalized using Layer Normalization. This layer stabilized and accelerated training by standardizing the outputs across the features.

3.2.4.3. Fully Connected Layer 1

A fully connected layer, also called the dense layer, was applied to transform the normalized BiLSTM outputs into a suitable dimension for further processing.

3.2.4.4. Batch Normalization

After the first fully connected layer, batch normalization was used to further improve training stability and reduce internal covariate shift.

3.2.4.5. ReLU Activation

A ReLU activation function was applied to introduce non-linearity into the model, enabling it to learn complex patterns.

3.2.4.6. Fully Connected Layer 2

The activated output was fed into another fully connected layer, continuing the transformation of learned features.

3.2.4.7. ReLU Activation

A second ReLU activation was applied for further non-linear mapping.

3.2.4.8. Fully Connected Layer 3

The output was then passed to a third fully connected layer, preparing it for classification.

3.2.4.9. Log Softmax

A log softmax layer was used to convert the final output scores into log-probabilities across the output vocabulary (e.g., characters or phonemes).

3.2.4.10. CTC Loss

The CTC loss handled the alignment between input audio frames and output labels. It enabled the model to train without the need for pre-aligned input-output pairs.

3.2.5. Model Training

The model was trained on the MFCC feature sequences using the CTC loss. During training, the model learned to map audio features like loudness, pitch, phonemes to corresponding character sequences. The hyperparameters like batch size, learning rate and number of epochs were selected to ensure convergence.

3.2.6. Output

The output of the trained model was a sequence of readable and coherent Nepali characters that represented the transcribed text of the input audio. This result was evaluated to check how closely they matched to the ground truth, based on which the CER was calculated.

3.3. Hardware and Software Used

3.3.1. Hardware Used

The data collection and preprocessing required only a standard laptop with 8 GB RAM and basic processing capabilities, as it was sufficient. The audio data was recorded using an in-

built laptop microphone in a quiet environment to ensure acceptable quality for training. The model building required a machine with a modern CPU and a GPU to accelerate model training. The cloud-based GPU, Tesla T4 was used for this model. Since training and experimentation was conducted on Google Colab, the local hardware requirements are minimal.

Table 3.1: Specification of Google's Tesla T4 GPU

Graphics Processor		**Memory**	
GPU Name	TU104	Memory Size	16 GB
GPU Variant	TU104-895-A1	Memory Type	GDDR6
Architecture	Turing	Memory Bus	256 bit
Foundry	TSMC	Bandwidth	320.0 GB/s
Process Size	12 nm		
Transistors	13,600 million		
Density	25.0 M/mm^2		
Die Size	545 mm^2		
Chip Package	BGA-2228		

3.3.2. Software Used

The implementation used Python along with the PyTorch library for building and training the BiLSTM-CTC model. Additional tools like NumPy, pandas and Librosa were used to handle the audio. Google Colab provided a browser-based development environment with pre-installed libraries that assisted with the task of training the model and testing it. Google Drive was used for storing datasets and models, as well as the output, making the workflow efficient and accessible without the need for high local storage or setup.

Table 3.2: Software Requirements

Cateogry	**Specification / Packages**
OS	Windows 11 / Ubuntu 24.04.2 LTS
Deep Learning Framework	PyTorch 2.6.0 + CUDA 124

Audio Processing	Torchaudio 2.6.0, Librosa 0.11.0
Visualization & Monitoring	Matplotlib 3.10.0, Ipython.display 7.34.0, tqdm 4.67.1
Tokenizer	Character-level Tokenizer
Other Library	Pandas 2.2.2, Numpy 2.0.2

Chapter 4: RESULT AND ANALYSIS

The model was trained for 82 epochs with a batch size of 32, using the AdamW optimizer with a learning rate of 0.001. 13 MFCCs were extracted from the audio signals and fed into a two-layer BiLSTM network, followed by a CTC layer for alignment-free transcription. The following results were seen.

4.1.MFCC Features

MFCCs were used to convert speech signals into a compact feature representation that captures important phonetic information. Figure 4.2 shows the MFCC representation of a speech signal. The x-axis represents time frames, and the y-axis indicates the MFCC coefficients. The color intensity reflects the value of each coefficient, where darker regions represent lower values and brighter regions indicate higher values. This plot captures the dynamic acoustic features of the audio and serves as the input to the ASR model for recognizing phonetic patterns over time.

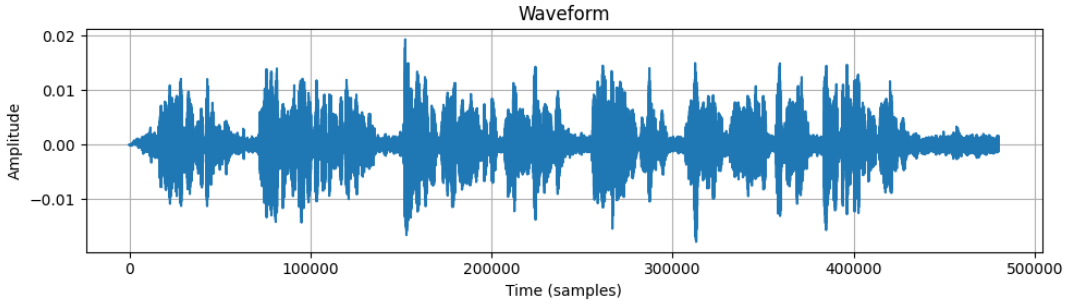

Figure 4.2: Representation of Audio in Time vs Amplitude Graph

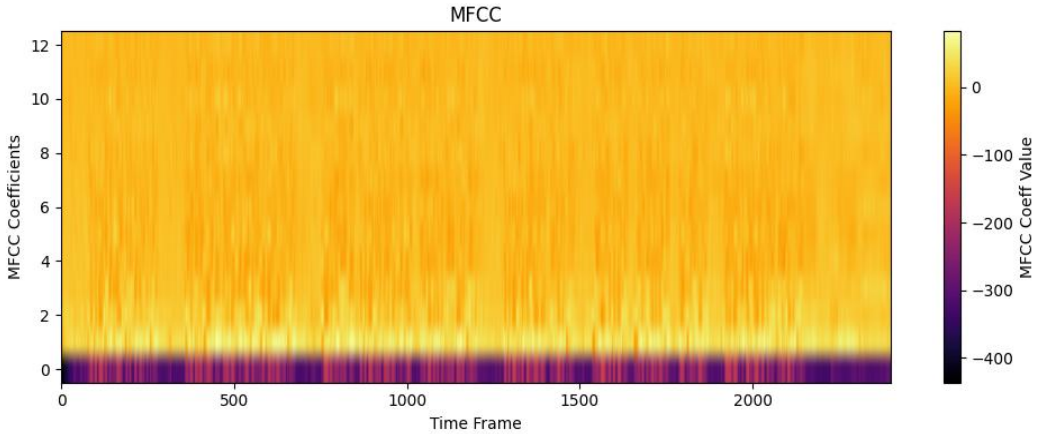

Figure 4.1: MFCC Representation

4.2. Training and Validation Performance

The model was trained on the training set, which comprised of 80% of the dataset and evaluated the testing set consisting of 20% of the dataset. The training and validation loss values were tracked over each epoch. Figure 4.3 shows a plot of the training and validation loss values over the epochs. The loss decreased steadily, showing effective learning. The training loss stabilized after approximately 51 epochs.

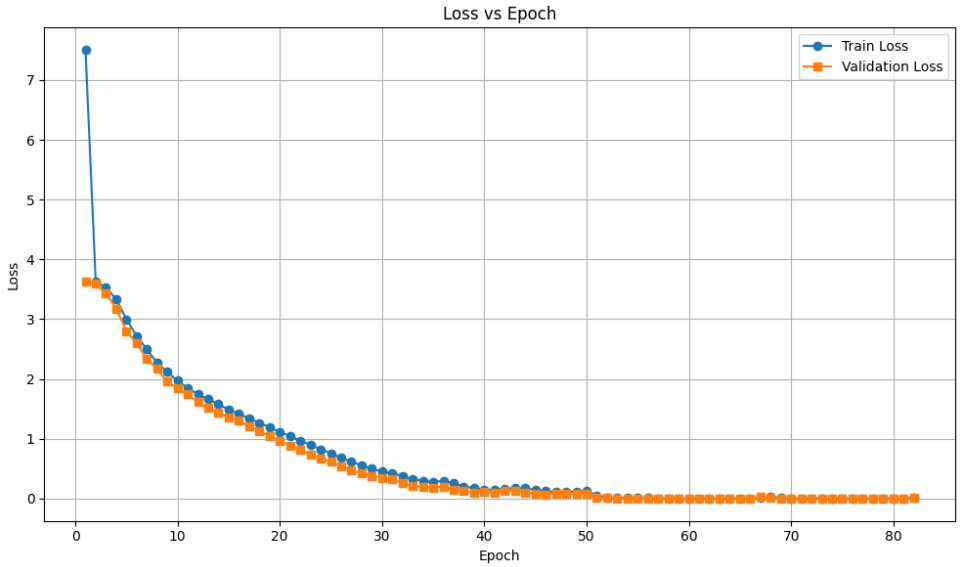

Figure 4.3: Plot of Train and Validation Loss over the Epochs

4.3. Evaluation Metric: CER

Nepali is a morphologically rich language. Nepali words can change significantly with small character-level modifications, and minor errors in spacing or inflection can lead to large penalties in WER. CER avoids such harsh penalties by evaluating character-level accuracy. This makes it more suitable for capturing subtle variations and partial correctness in the words. Due to this reason, CER was used as the evaluation metric. The model achieved the CER value of 0.009. Figure 4.4 shows a plot of CER for different epochs.

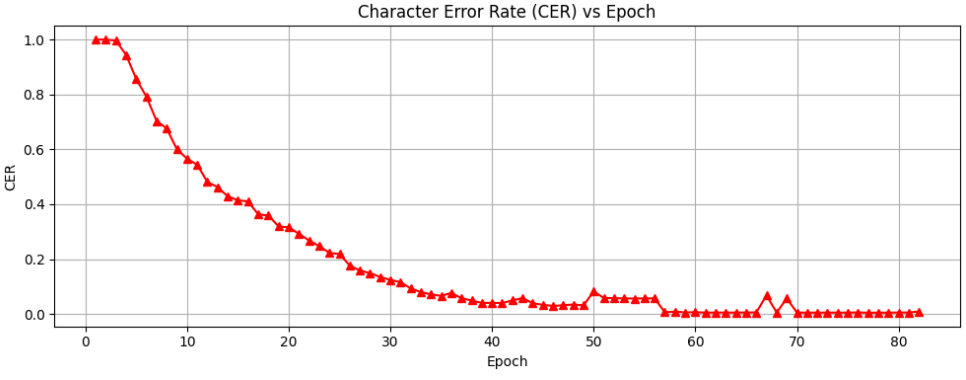

Figure 4.4: Plot of CER over the Epochs

4.4. Error Analysis

The analysis of prediction errors reveals several recurring patterns, primarily involving vowel matra misplacement or substitution. For example, the model predicted "नयँ" instead of "नयाँ" and "राखिएको" instead of "राखिएको". This is common due to the complex placement of matras in the Devanagari script. Character reordering was also observed, especially when multiple vowel signs were applied to the same base consonant. Some outputs showed missing characters or digits, such as the omission of "०" in "१९५०". Occasional insertions, like the trailing "क" in "माध्यमबाटक", and complete word-level substitutions, such as "छ" instead of "थियो", highlight the challenges faced by CTC-based decoding without language model support. These errors suggest the model handles overall structure well but still struggles with fine-grained character-level accuracy, particularly with complex vowel combinations.

4.5. Sample Predictions

Some sample outputs from the test set are shown below, with their corresponding CERs:

Table 4.1: Prediction and Ground Truth Comparison with the CER

Prediction	Ground Truth	CER
नेपाली काङ्ग्रेस पार्टीको	नेपाली काङ्ग्रेस पार्टीको	0.00
उपयुक्त नक्सा तयार	उपयुक्त नक्सा तयार	0.00
के थियो भने	के थियो भने	0.00
कष्ट कडा हुन्छ	कष्ट कडा हुन्छ	0.00
१९३४ को जनावरी	१९३४ को जनावरी	0.00
१४ वटा स्वतन्त्र	१४ वटा स्वतन्त्र	0.00
उहाँले मनोक्रान्तिको माध्यमबाटक	उहाँले मनोक्रान्तिको माध्यमबाट	0.03
ट्रेट्सको थाहा भएको छ।	ट्रेट्सको थाहा भएको थियो।	0.16

4.6. Discussion

The ASR system demonstrated strong performance on Nepali speech recognition. The BiLSTM-CTC model trained on MFCC features performed well in recognizing Nepali speech in clear and simple conditions, as shown by its low CER. However, the model struggled with complex character combinations, matras, and nasal forms common in the Devanagari script. Errors such as repeated or misplaced characters were more frequent in longer and irregular sentences. These issues highlight the model's limitations in capturing long-range context and linguistic structure, as the CTC loss alone does not incorporate deeper language understanding.

Chapter 5: CONCLUSION AND FUTURE ENHANCEMENT

5.1. Conclusion

This project demonstrates the successful implementation of an end-to-end Nepali ASR system by utilizing the MFCC features of the audio, a BiLSTM-based neural network, and the CTC loss function. The system was trained on a dataset comprising 4,000 audio samples (about 5.5 hours) from publicly available Nepali speech data, enabling the model to learn from real-world pronunciation patterns.

The BiLSTM architecture effectively captured both forward and backward temporal dependencies, which is crucial for accurately recognizing phonemes in the Devanagari script. The use of CTC loss allowed the model to learn alignments between speech and text without the need for frame-level labeling, thereby simplifying the training process while maintaining high accuracy.

The trained model achieved a low CER of 0.0090, reflecting strong performance. Sample predictions were largely accurate, and the training and validation loss curves showed steady convergence. These results demonstrate that the developed system can transcribe Nepali speech with high accuracy and serves as a solid foundation for further advancements in Nepali ASR and speech technology for low-resource languages.

5.2. Future Enhancements

Despite the encouraging results achieved, several areas offer potential for further improvement. A key enhancement involves expanding the dataset to include a broader range of speakers, incorporating variations in accents, age groups, and speaking styles. This would significantly improve the model's ability to generalize across diverse real-world scenarios. Additionally, applying data augmentation techniques, such as adding background noise, speed perturbation, and pitch shifting, can enhance the model's robustness in noisy or unpredictable environments.

To produce more fluent and contextually accurate transcriptions, character-level decoding could be improved by integrating a language model. Furthermore, experimenting with advanced architectures like Transformer-based models, attention mechanisms, or hybrid

CNN-BiLSTM networks may yield better performance, particularly on complex or extended speech segments.

Finally, deploying the trained model as a mobile or web-based application would greatly increase its accessibility and usability for Nepali speakers. These proposed enhancements aim not only to improve overall system accuracy and resilience but also to support the broader objective of fostering inclusive and effective voice-based technologies for low-resource languages.

REFERENCES

[1] R. Shrestha, B. Joshi, and S. Sharma, "Nepali Speech Recognition Using LSTM-CTC," in *10th IOE Graduate Conference (IOEGC)*, Kathmandu: Institute of Engineering, Tribhuvan University, Oct. 2021, pp. 170–174.

[2] 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, Jul. 2022, pp. 515–521. doi: 10.1109/ICICT54344.2022.9850832.

[3] P. Regmi, A. Dahal, and B. Joshi, "Nepali Speech Recognition using RNN-CTC Model," *Int J Comput Appl*, vol. 178, no. 31, pp. 1–6, Jul. 2019, doi: 10.5120/ijca2019918401.

[4] B. Bhatta, B. Joshi, and R. K. Maharjhan, "Nepali Speech Recognition Using CNN, GRU and CTC," in *32nd Conference on Computational Linguistics and Speech Processing (ROCLING 2020)*, Taipei: The Association for Computational Linguistics and Chinese Language Processing, Sep. 2020, pp. 238–246.

[5] B. Joshi, B. Bhatta, and R. K. Maharjan, "End to End based Nepali Speech Recognition System," *Journal of the Institute of Engineering*, vol. 17, no. 1, pp. 102–109, Apr. 2023.

[6] A. Kafle, J. Rajlawat, N. Shah, N. Paudel, and B. Thapa, "Advancements in Nepali Speech Recognition: A Comparative Study of BiLSTM, Transformer, and Hybrid Models," *International Journal on Engineering Technology*, vol. 2, no. 1, pp. 96–105, Dec. 2024, doi: 10.3126/injet.v2i1.72525.

[7] R. R. Ghimire, B. K. Bal, and P. Poudyal, "A Comprehensive Study of the Current State-of-the-Art in Nepali Automatic Speech Recognition Systems," in *International Conference on Technologies for Computer, Electrical, Electronics & Communication (ICT-CEEL 2023)*, Bhaktapur: Khwopa Engineering College/Khwopa College of Engineering, Oct. 2023, pp. 73–80.

[8] S. Regmi and B. K. Bal, "An End-to-End Speech Recognition for the Nepali Language," in *18th International Conference on Natural Language Processing (ICON)*, S. Bandyopadhyay, S. L. Devi, and P. Bhattacharyya, Eds., Silchar: NLP

Association of India (NLPAI), Dec. 2021, pp. 180–185. [Online]. Available: https://aclanthology.org/2021.icon-main.22/

[9] A. Graves and N. Jaitly, “Towards End-To-End Speech Recognition with Recurrent Neural Networks,” in *31st International Conference on Machine Learning*, PMLR, Jun. 2014, pp. 1764–1772. [Online]. Available: https://proceedings.mlr.press/v32/graves14.html

[10] C. Yu, Y. Chen, Y. Li, M. Kang, S. Xu, and X. Liu, “Cross-Language End-to-End Speech Recognition Research Based on Transfer Learning for the Low-Resource Tujia Language,” *Symmetry (Basel)*, vol. 11, no. 2, p. 179, Feb. 2019, doi: 10.3390/sym11020179.

[11] S. K. Kopparapu and M. Laxminarayana, “Choice of Mel filter bank in computing MFCC of a resampled speech,” in *10th International Conference on Information Science, Signal Processing and their Applications (ISSPA 2010)*, IEEE, May 2010, pp. 121–124. doi: 10.1109/ISSPA.2010.5605491.

[12] O. Kjartansson, S. Sarin, K. Pipatsrisawat, M. Jansche, and L. Ha, “Crowd-Sourced Speech Corpora for Javanese, Sundanese, Sinhala, Nepali, and Bangladeshi Bengali,” in *6th Workshop on Spoken Language Technologies for Under-Resourced Languages (SLTU 2018)*, ISCA: ISCA, Aug. 2018, pp. 52–55. doi: 10.21437/SLTU.2018-11.

[13] N. D. Londhe and G. B. Kshirsagar, “Chhattisgarhi speech corpus for research and development in automatic speech recognition,” *Int J Speech Technol*, vol. 21, no. 2, pp. 193–210, Jun. 2018, doi: 10.1007/s10772-018-9496-7.

[14] P. Bahad, P. Saxena, and R. Kamal, “Fake News Detection using Bi-directional LSTM-Recurrent Neural Network,” *Procedia Comput Sci*, vol. 165, pp. 74–82, 2019, doi: 10.1016/j.procs.2020.01.072.

[15] A. Hannun, “Sequence Modeling with CTC,” *Distill*, vol. 2, no. 11, Nov. 2017, doi: 10.23915/distill.00008.