TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS

A
MAJOR PROJECT REPORT
ON

SHRUTI - A NEPALI BOOK READER

SUBMITTED BY:
PRABIN PAUDEL (PUL075BCT060)
RAHUL SHAH (PUL075BCT063)
RANJU G.C. (PUL075BCT064)
SUPRIYA KHADKA (PUL075BCT090)

SUBMITTED TO:
DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING

May, 2023

Page of Approval

TRIBHUVAN UNIVERSIY
INSTITUTE OF ENGINEERING
PULCHOWK CAMPUS
DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

The undersigned certifies that they have read and recommended to the Institute of Engineering for acceptance of a project report entitled **”Shruti - A Nepali Book Reader”** submitted by **Prabin Paudel**, **Rahul Shah**, **Ranju G.C.**, **Supriya Khadka** in partial fulfillment of the requirements for the Bachelor’s degree in Electronics & Computer Engineering.

....	....
Supervisor	External examiner
Dr. Basanta Joshi	**Dr. Bal Krishna Bal**
Assistant Professor	Associate Professor
Department of Electronics and Computer Engineering,	Department of Computer Science & Engineering,
Pulchowk Campus, IOE, TU.	Kathmandu University (KU)

Date of approval:

Copyright

Acknowledgement

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible. We consider it our privilege to express a voice of gratitude and respect to all those who guided us and inspired us in the completion of this project.

Primarily, we would like to express our sincere gratitude towards the **Department of Electronics and Computer Engineering, IOE Pulchowk Campus**, for providing us with this great opportunity to sharpen our programming and analytical skills by assigning us the task of designing a furnished solution to a real-world problem in the form of a major project, which has enhanced our knowledge and broadened our horizon.

This project would not have been possible without the continuous guidance and support of our project supervisor **Dr Basanta Joshi.** We owe a great debt of gratitude to him for his invaluable suggestions and constructive criticisms, which have contributed significantly to our work.

We would also like to express our sincere gratitude and appreciation to **LIS Nepal** for providing us with the opportunity to work in their space and for their guidance throughout the project. The supportive environment, resources, and expertise that LIS provided were invaluable to the growth and development of this project.

We would also like to express our heartfelt gratitude to our supervisors whose constant guidance and encouragement helped us complete this project. And, of course, nothing could have come true without the support of our family and friends for their constant external support.

Authors:
Prabin Paudel
Rahul Shah
Ranju G.C.
Supriya Khadka

Abstract

The use of audiobook technology in the classroom has long been a viable instructional intervention for struggling readers. Shruti, an AI-generated Nepali book reader, is an application that generates a voice for the book. It is a text-to-speech(TTS) system that takes an input book in a PDF format. The PDF is extracted to text using Optical Character Recognition(OCR) and sent to the text-to-speech pipeline. The speech synthesis acts in two phases: spectrogram generation and vocoder output. The text is extracted, preprocessed, tokenized and sent to the modified Tacotron2 model for generating Mel spectrograms. The output in the form of Mel spectrograms is sent to the HifiGAN vocoder, which produces the sound. The synthesized sample of speech attained a Mean Opinion Score of 4.04 on the basis of naturalness, when audio samples were subjected to 28 volunteers. This sound is post-processed as a final output. The model has been deployed and integrated with a mobile application.

Keywords: *Text-to-Speech, Tacotron2, OCR, Mel-spectrogram, vocoder, HifiGAN*

Contents

List of Figures

List of Tables

List of Abbreviations

CNN	Convolutional Neural Network
ESNOLA	Epoch Synchronous Non-Overlap Add Method
GPU	Graphics Processing Unit
K-12	Kindergarten through 12th grade
MOS	Mean Opinion Score
NLTK	Natural Language ToolKit
OCR	Optical Character Recognition
OpenSLR	Open Speech and Language Resources
PDF	Portable Document File
RNN	Recurrent Neural Network
TDPSOLA	Time Domain Pitch Synchronous Overlap Add Method
TTS	Text-to-Speech
URL	Uniform Resource Locator
Vocoder	Voice Encoder

1. Introduction

1.1 Background

Audiobooks are recordings of a book that we listen to on any media-consuming device. It can be an exact word-for-word version of books or an abridged version. Historically, audiobooks have been described as the by-product of printed books but numerous research studies have found that audiobooks when concurrently used with reading, can have significant improvement in a student's fluency, vocabulary, comprehension, and motivation. This is because reading and listening are symbiotic[1]. Listening to the text removes the constraint of word recognition and decoding, and instead allows the student to focus on the content. Additionally, audiobooks can help improve fluency and pronunciation. Listening to a skilled narrator read a book can provide a model for students on how to properly pronounce words and phrases. This can also be helpful for language learners who may be struggling with pronunciation.

1.2 Topic Choice and its importance

The literacy rate of Nepal was around 66 % in 2018, which is quite low. This was much lower in rural areas with an average percent of around 62% compared to 84% of urban areas. Similarly, this difference exists among gender with a literacy rate of 71% in males and 45% in females.[2] Literacy of today's children directly affects the next generation. So, necessary changes should be employed to improve the reading ability of the general public.

We are looking for strategies to engage children in the practice of reading to bring about that change. One area of growth is in the use of technology where students at various skill levels can benefit from targeted technology-infused literacy strategies. One of the prominent strategies is the use of audiobooks. Audiobooks have been used as a practical intervention strategy for struggling adolescent students since books were recorded. A study has found that the most significant activity for increasing reading skills and comprehension for children is to read to them aloud[3].

1.3 Problem Statement

Finding pleasure in the reading activity starts at a young age. However, struggling learners find reading to be pure drudgery. Reading for pleasure is not just for fun rather it helps in the overall development of personnel like opening up different perspectives and infusing skill-sets that include better communication and conflict resolution. Disliking the act of reading is an urgent issue to solve because discovering joy from reading is essential to comprehension. Students who are not able to comprehend their textbooks or other forms of literature in various content areas will not learn the concepts and vocabulary.

Additionally, the traditional method of creating audiobooks involves hiring a voice actor to read the book aloud, which can be expensive and time-consuming. Moreover, the availability of audiobooks in Nepali languages and for diverse genres is limited. This poses a problem for individuals who prefer listening to reading and for students who struggle with reading comprehension.

1.4 Proposed Solution

Shruti, an AI-generated book reader, is an application that generates audio for the book. Researchers have found that using audiobook technology in the classroom has long been a viable instructional intervention for struggling readers[4]. This project aims to develop a deep learning-based TTS model and a pipeline for generating audiobooks automatically from PDF. It is not merely a traditional TTS system, but a system integrated with a mobile application. With this technology, there are even more options for educators to get the technology into the hands of the reader.

Audiobooks have proven to be an effective tool for enhancing learning outcomes at the K-12 level in Nepal. We aim to build on this success by extending the reach of audiobooks to all those who embrace this innovative form of learning.

1.5 Objective

The main objective of our project is to build a natural-sounding Nepali Book Reader that will enhance the reading experience for users and make it more accessible to those who struggle with traditional reading methods.

1.6 Scope

With literature becoming more and more complicated, audiobooks can help struggling readers gain knowledge from books. It can assist visually impaired students, students with reading disorders such as dyslexia or even illiterate people by reading the book to them.

Listening to audiobooks while reading the book simultaneously leads to faster learning ability since it enables students to focus more on the plot rather than word-by-word decoding. There are findings that the technology is also effective for teaching English as a second language. It also enables the reader to associate certain words with sounds as represented by a fluent model. As a result, it improves the learner's phonological and comprehension abilities while making the tale more enjoyable.

1.7 Application

Some major applications of Shruti would be:

- It can assist in reading books to people who struggle to read whether because of physical impairments, disorders or illiteracy.
- It helps to comprehend the book better increasing the enjoyment provided by the book.
- It can also be used to read the content of the website allowing for easy navigation through the site.
- Shruti can be modified to read content through multiple voices.This can be utilized to create voices for gaming or other virtual characters.

2. Literature Review

Text to Speech(TTS) synthesis began as early as in the 1770s and gained attraction in the 1960s when Noriko Umeda and his companions developed the first full text-to-speech system for English. In that system, the speech was fairly understandable, but far from current standards.[5] The TTS systems gained newer heights with other approaches like concatenative, articulatory and parametric. Although they are considered to be traditional methods now after the emergence of technologies like WaveNet[6], Tacotron[7] and Tacotron2[8], they paved the path for Speech Synthesis.

2.1 Related Works

The TTS synthesis technology has evolved leaps and bounds for languages like English, which has abundant resources, but it is still difficult to develop a natural-sounding TTS system for under-resourced languages like Nepali. One of the earliest attempts in the field of generating Nepali Speech was seen in the form of phonetic lexicons done by a group of scientists from Columbia University, Carnegie Mellon University and Cepstral LLC. [9] In this work, the bootstrapping algorithm method proceeded by automatically building Letter-to-Sound (LTS) rules from a small set of the most commonly occurring words in a large corpus of a given language. The approach was tested for three languages: English, German and Nepali. An accuracy of 94.6% was noted, which was the highest among the three languages. The authors have attributed this advantage to the Nepali language, which has orthography and pronunciation that are more closely related.

One of the first attempts from Nepal in building a Nepali Text to Speech system was Bhasa Sanchar, which used a concatenative approach with the Festival TTS system.[10]. A concatenative speech synthesis approach produces sound by combining recorded sound clips. Any spoken word can be created from the recorded speech by breaking it down into smaller components. There are three main kinds of concatenative synthesis: Unit Selection Synthesis, Diphone Synthesis and Domain-specific Synthesis. Bhasa Sanchar was based on the unit selection synthesis method. Unit selection synthesis uses large databases of recorded speech and creates a database from the recorded utterances. [11] It was an initial attempt but lacked a lot of refinements. It was observed that long input to the engine caused unusual output which was messy, lacking proper intonation (too fast or too slow reading), and thus sounding unnatural.[12]

An improvement to Bhasa Sanchar was developed in 2017 with modifications in the general architecture, by adding a tokenizer to split long input text into manageable small tokens and a post-processing module to concatenate the output in accordance with the input.[12] The qualitative evaluation of this modified TTS system is done by Mean Opinion Score(MOS). Mean opinion score (MOS) is a rating, done by human listeners, of how good the synthesized utterances are, usually on a scale from 1–5.[13] MOS showed an overall improvement of 6 per cent in terms of naturalness and intelligibility, and the result of comprehension and diagnostic rhyme tests increased by 12 per cent and 10 per cent respectively. Bhasa Sanchar also collaborated with the Sambaad project, which had initially been developing TTS for Nepali using the Festival and Festvox systems[1], to create easy accessibility technological accessibility for non-literates.[14]

There have been multiple attempts in improving the naturalness of voice in Nepali TTS systems. The Nepali TTS system trained on the Flite TTS engine gives the facility of changing pitch and speed, but it sounds robotic and is not completely noise free. [15] The Festival system used with concatenation to synthesize Nepali speech from the text displayed on the screen also has issues such as overlaps and echoes. [11] Other Nepali TTS systems built with concatenative approaches employing Epoch Synchronous Non-Overlap Add Method (ESNOLA) and Time Domain Pitch Synchronous Overlap Add Method (TDPSOLA) are also in existence, but the have issues of their own.

ESNOLA Nepali TTS system uses signal dictionaries having raw sound signals representing parts of phonemes as a speech database.[16] As this system uses a partneme for concatenation, it makes the size of the speech dictionary quite small. There also might be some disturbance in the concatenation point of the synthetic speech signal.[17] TDPSCOLA Nepali TTS system has a small speech dictionary making it less accurate and error-prone. Another problem seen with concatenative approaches(in both ESNOLA and TDPSCOLA) is that it requires a lot of feature engineering.[17]

The Nepali TTS system using a formant approach and FreeTTS synthesizer was developed in 2018.[18] Formant approach is a rule-based synthesis method, which synthesizes speech using additive synthesis and an acoustic model taking multiple parameters.[19] Mostly, these system adopts the open-source text-to-speech translation frameworks such as Festival[20] and its variants to perform Nepali TTS. However, these

[1] `(http://www.cstr.ed.ac.uk/projects/festival/)`

systems still need an improvement in various speech synthesis aspects.[21]

In the field of multilingual text-to-speech systems for South Asian[22] and Indian languages [23], Nepali TTS has been mentioned as a component. Nepali is considered one of the Indian languages, and as a result, many research studies have incorporated it into multilingual setups. However, the amount of work done on Nepali TTS is not extensive, and the focus is not solely on Nepali.

With the rise of systems like WaveNet, Tacotron and Deep Voice, the world has been moving towards more data-efficient and natural-sounding TTS research areas. WaveNet and Tacotron are neural text-to-speech mechanisms, which work end-to-end and can learn directly from text-to-speech pairs. WaveNet is an auto-regressive and fully probabilistic model for predicting the distribution of speech signals. Since it consists of convolutional units instead of recurrent ones, it is more efficient in terms of computation.[6] Tacotron TTS is an end-to-end generative text-to-speech model that takes a character sequence as input and outputs the corresponding spectrogram.[7] The update to the Tacotron model, Tacotron 2, is considered to be one of the most successful TTS models with a MOS score of 4.526 in North American English.[24] Tacotron 2 works on the principle of superposition of two deep neural networks; one that converts text into a spectrogram, and the other that converts the elements of the spectrogram to corresponding sounds.[8]

Although WaveNet and Tacotron have not been extensively used in Nepali TTS, there is a base model for Nepali TTS synthesis using Neural WaveNet Vocoder. This system pre-processes the Nepali Speech dataset from OpenSLR and develops the Nepali TTS model using the Attention Based Recurrent Sequence to Sequence model with WaveNet vocoder.[25] The quality evaluation of this system by Mean Opinion Score resulted in a score of 2.78, which is even lower than the MOS obtained from concatenative approaches.[12] That is due to less size of the dataset, noise in the dataset and lack of resources. The score will definitely increase with the betterment of these factors. [26]

2.2 Related Theory

2.2.1 PDF Parser/Scraping

PDF Parsing is the process of extracting important information from PDF documents and PDF Parser is a tool capable of parsing the necessary PDF. PDF Parsers are generally used to extract large amounts of data from PDFs or from a batch of PDF files. PDF Parser can distinguish between different elements in a PDF such as text, table, images, metadata, and hyperlinks and extract any specific element. It can even recognise different fonts embedded in the document. It reads what's in the document and hence, is fast and accurate. PDF Parsers are an optimal solution for intelligently scraping PDFs.

Since the parsing process is code-driven, different logics can be implemented in order to extract required content anywhere in the document. This allows one to create a framework for extracting data from multiple sources even though the documents may vary from each other.

Some of the PDF parsing Python libraries are:

- PyMuPDF
- PyPDF2
- PDFMiner

2.2.2 Optical Character Recognition (OCR)

Optical Character Recognition refers to the conversion of images consisting of texts or characters within it, such as handwritten or typed, into machine-encoded text. An OCR program can extract as well as repurpose data from a scanned document, a photo of a document or any images consisting of textual data. It extracts the content of one character or a word at a time, joins them to form a sentence, and thus generates a textual representation of the original source.

It generally supports different file types such as jpg, jpeg, png, and pdf as well as other formats. The output stream can be a plain text stream or a group of characters. However, an improved OCR can represent the data in a layout parallel to the original input.

Some of the Python libraries for OCR are:

- PyTesseract

- EasyOCR

There are 2 basic types of core OCR algorithms, which are described below:

1. **Pattern matching:** Also known as "pattern recognition", "image correlation" or "matrix matching", it compares an image to a stored glyph on a pixel-by-pixel basis. This method requires isolation of the glyph from the image and the stored glyph should be similar in font as well the scale. It is highly efficient in typewritten text but doesn't work in texts whose font it is not trained on.

2. **Feature extraction:** It decomposes any glyphs into a number of features such as lines, closed loops, line direction and intersection. These features reduce the dimensionality of the representation making the detection of the character computationally efficient. The reference characters are stored in an abstract vector-like representation which are compared with the glyph's features. Most modern OCR software as well as intelligent handwriting recognition systems use this technique.

2.2.3 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN) are a type of artificial neural network that is commonly used for image classification tasks. They are particularly effective in detecting patterns and spatial relationships in images, making them well-suited for tasks such as object recognition and image segmentation.

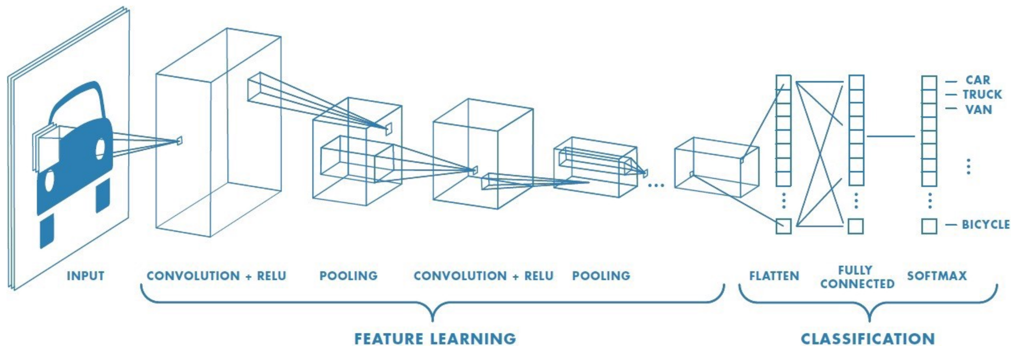


Figure 2.1: CNN Network

Architecture

CNNs are made up of several layers, each of which performs a specific function. The three main types of layers are:

- Convolutional Layers
- Pooling Layers
- Fully Connected Layers

Convolutional layers apply a set of filters to the input image, which is used to extract features such as edges and shapes. Pooling layers downsample the feature maps produced by the convolutional layers, reducing their size while retaining the most important information. Finally, fully connected layers use feature maps to classify the input image.

Training

Training a CNN involves feeding it a large dataset of labelled images, and adjusting the weights of the network in order to minimize the difference between its predicted outputs and the true labels. This is typically done using an optimization algorithm such as stochastic gradient descent.

Applications

CNNs have a wide range of applications, including image classification, object detection, image segmentation, facial recognition and natural language processing.

2.2.4 Recurrent Neural Networks(RNNs)

Recurrent Neural Networks (RNNs) are a type of artificial neural network where connections between nodes form a directed graph along a sequence. This allows the network to exhibit temporal dynamic behaviour. In contrast to feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs.

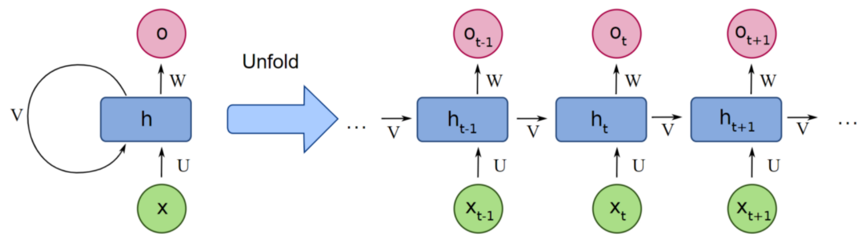


Figure 2.2: RNN Network

Architecture

The basic architecture of an RNN consists of a single layer of processing nodes, where each node is connected to the previous node with a directed edge. The output of each node is a function of its current input and its previous state (output). This creates a feedback loop that allows the network to maintain a memory of its previous inputs.

Training

Training an RNN involves adjusting the weights of the connections between nodes to minimize a loss function. This is typically done using backpropagation through time (BPTT), which is a variant of backpropagation specifically designed for training recurrent networks.

Applications

RNNs have been successfully applied to a wide range of tasks involving sequential data, such as speech recognition, natural language processing, and time series prediction. They have also been used in generative models, where the network is trained to generate new sequences of data that are similar to a training set.

2.2.5 Incremental Learning

Incremental learning, also known as online learning or lifelong learning, is a machine learning paradigm in which the model learns incrementally from new incoming data without forgetting previously learned knowledge. In other words, the model is capable of adapting to new information while maintaining its previous knowledge.

Need for Incremental Learning

Traditional machine learning algorithms rely on the availability of large amounts of labeled data, which is often expensive and time-consuming to acquire. Moreover, these algorithms require the entire dataset to be available at once for training. However, in many real-world scenarios, the data arrives incrementally, in a stream, or in batches. In such cases, it is not feasible to retrain the model from scratch each time new data is available. Hence, there is a need for incremental learning, which allows the model to learn from new data without sacrificing its previous knowledge.

Challenges in Incremental Learning

Incremental learning poses several challenges, such as catastrophic forgetting, which occurs when the model forgets previously learned knowledge while learning new information. Other challenges include managing the trade-off between exploiting the new data and exploring new areas of the input space, avoiding overfitting, and maintaining model stability over time.

Approaches to Incremental Learning

There are various approaches to incremental learning, including incremental update of model parameters, incremental expansion of model structure, and rehearsal-based approaches. Some popular algorithms for incremental learning include Online Gradient Descent, Elastic Weight Consolidation, and Deep Adaptive Network.

Applications of Incremental Learning

Incremental learning has various applications in domains such as computer vision, natural language processing, and recommendation systems. For example, in computer vision, incremental learning can be used to improve object recognition over time by learning from new examples of objects. In natural language processing, incremental learning can be used to improve machine translation by learning from new text data. In recommendation systems, incremental learning can be used to update user preferences over time.

2.2.6 Short Time Fourier Transform (STFT)

The Short Time Fourier Transform (STFT) is a time-frequency analysis technique used to represent a signal in the time-frequency domain. It is based on the Fourier Transform (FT) and involves breaking down a signal into small time segments and applying the Fourier Transform to each segment.

Windowing

To obtain a smooth spectral representation of a signal, a windowing function is applied to each time segment before the Fourier Transform is computed. The windowing function is a weighting function that tapers the signal towards zero at the beginning and end of each segment to avoid spectral leakage.

Mathematical Formulation

Given a discrete-time signal $x[n]$, the STFT can be computed using the following equation:

$$X(m, \omega) = \sum_{n=0}^{N-1} x[n]w[n-m]e^{-j\omega n}, \tag{2.1}$$

where m is the index of the time segment, ω is the frequency index, $w[n]$ is the windowing function, and N is the length of the signal.

Parameters

The STFT involves several parameters that affect the analysis, including:

- **Window length**: The length of the windowing function applied to each time segment.
- **Filter length**: The length of the filter used to compute the Fourier Transform of each segment.
- **Hop length**: The number of samples by which the window is shifted to obtain the next segment. A smaller hop length results in a higher time resolution but lower frequency resolution, while a larger hop length results in a lower time resolution but higher frequency resolution.

Applications

The STFT is widely used in various applications, including audio signal processing, image processing, and speech recognition. It is particularly useful for analyzing non-stationary signals, where the frequency content of the signal varies over time.

2.2.7 TTS

TTS stands for Text-to-Speech, which is a technology that converts written text into synthesized speech. The TTS technology can be used in various applications such as audiobooks, navigation systems, virtual assistants, and accessibility tools for people with disabilities.

In a typical TTS system, the input text is first processed by a language analysis module that identifies the structure and pronunciation of the words and sentences. This processed text is then sent to a speech synthesis module that generates speech waveform from the text. The speech synthesis module can use various techniques such as concatenative synthesis, formant synthesis, and statistical parametric synthesis to generate the speech waveform.

TTS systems typically use an encoder-decoder architecture, either based on LSTMs or Transformers. The encoder maps the input text into an intermediate representation, and the decoder generates the corresponding speech. This approach is similar to that used in machine translation. The TTS task can be broken down into two components:

- Spectrogram Prediction Model: The first component maps from strings of letters to mel spectrographs, which are sequences of mel spectral values over time. This is done using an encoder-decoder model, which takes the input text as an input and generates the corresponding spectrogram as an output.

- Vocoder: The second component maps from mel spectrograms to waveforms, which is called vocoding. This is done using a vocoder, which takes the spectrogram as an input and generates the corresponding waveform as an output. The quality of the vocoder can significantly impact the overall quality of the TTS system.

The standard encoder-decoder algorithms for TTS are computationally intensive, so research is focused on ways to speed them up. Recently, there have been advances in using parallel processing and attention mechanisms to improve the performance of TTS systems.

2.2.8 Text Preprocessing

Text preprocessing is the process of cleaning, normalizing, and transforming raw text data into a format that is suitable for analysis and modelling by machine learning algorithms. Text data is often unstructured and noisy, containing various forms of noise such as punctuation, capitalization, stop words, and spelling errors. Text preprocessing is necessary to remove these noise components and extract useful features that can be used for analysis and modelling. Text preprocessing typically involves several steps, including:

Text normalization

The process of converting the input text into a standardized format, such as converting numbers to words, expanding abbreviations, and converting contractions to their full forms.

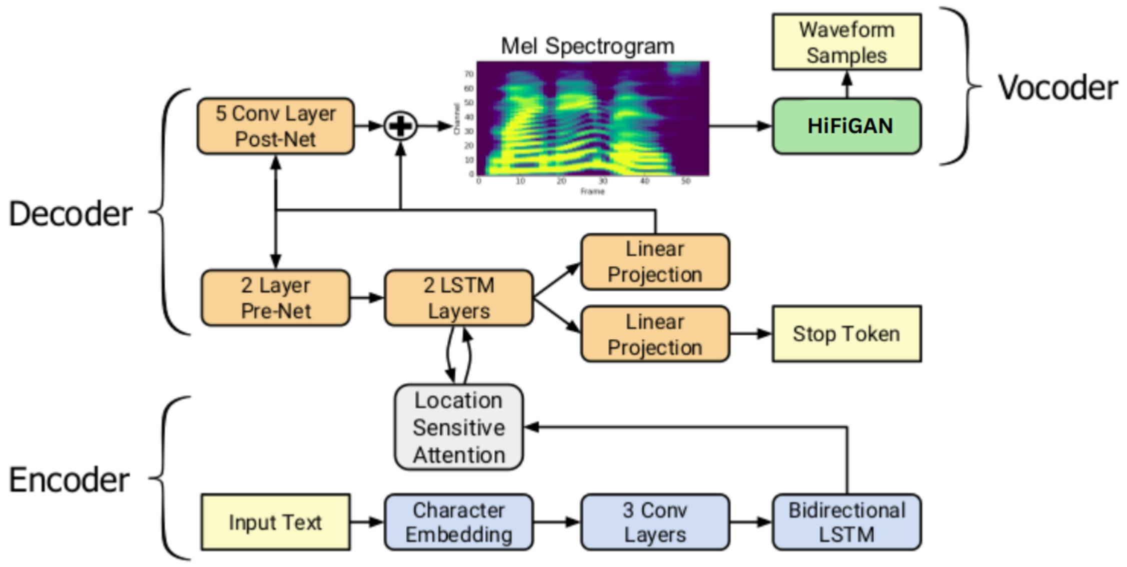
Punctuation and Symbol Handling

The process of identifying and handling punctuation marks and special symbols in the input text, such as commas, question marks, and emoticons.

Text processing for TTS is an important step in the TTS pipeline, as it directly affects the quality and naturalness of the synthesized speech. Effective text processing can improve the accuracy and fluency of the synthesized speech, making it more pleasant and easier to understand for the listener.

2.2.9 Tacotron 2

Tacotron 2 is a state-of-the-art neural network architecture for text-to-speech (TTS) synthesis. It was developed by researchers at Google's DeepMind and Google Brain teams and was first introduced in a paper published in 2017.



[27]

Figure 2.3: Block Diagram of Tacotron 2 System Architecture

The Tacotron 2 model is based on a deep neural network that uses a sequence-to-sequence architecture to generate speech from input text. It utilizes an encoder-decoder architecture with an attention mechanism.

Encoder

The encoder is responsible for taking a sequence of letters as input and producing a hidden representation that can be used by the attention mechanism in the decoder. Tacotron2's encoder consists of the following steps:

- **Character Embeddings:** In this section, the input text is first transformed into a sequence of embeddings, which are vector representations of each character in the text. Each embedding has a fixed size (512 dimensions in Tacotron2) and is learned during training. These embeddings capture the phonetic and semantic information of the text.

- **Convolutional Layers:** The character embeddings are passed through a stack of 3 convolutional layers with 512 filters each. Each filter has a shape of 5x1, meaning it spans 5 characters horizontally and only 1 character vertically. The output of each filter is a feature map that represents the presence of certain features in the text, such as phonemes or syllables. The purpose of these layers is to model the larger context of the text and capture longer-range dependencies.

- **Bi-directional LSTM (BiLSTM):** The output of the convolutional layers is fed into a bidirectional LSTM (BiLSTM), which processes the sequence in both forward and backward directions. The BLSTM produces a sequence of hidden states, which capture the context of each character in the text. The forward and backwards hidden states are concatenated at each time step to create a final encoding of the text.

Decoder

The decoder is responsible for generating the mel-spectrogram output from the hidden representation produced by the encoder. Tacotron2's decoder consists of the following steps:

- **Decoder Pre-net:** The output of the attention mechanism is passed through a small feedforward neural network called the pre-net, which has two fully connected layers with ReLU activations. The purpose of the pre-net is to reduce the dimensionality of the input to the decoder LSTM and to provide more meaningful input to the LSTM.

- **Decoder LSTM:** The pre-net output is fed into a two-layer LSTM, which produces a sequence of hidden states that are used to predict the mel-spectrogram. The decoder LSTM takes as input the concatenation of the pre-net output, the previous attention context vector, and the previously predicted mel-spectrogram. The hidden states of the decoder LSTM capture the temporal dependencies of the speech signal.

- **Stop Token Prediction:** In addition to predicting the mel-spectrogram, Tacotron2 also predicts a binary stop token at each time step, which indicates whether the decoder should stop generating output. The stop token is predicted by passing the decoder LSTM output through a linear layer followed by a sigmoid activation. If the stop token is predicted, the decoder stops generating output, otherwise, it continues until a maximum time step is reached.

One of the key innovations of Tacotron 2 is the use of a multi-speaker model, which enables the system to generate speech in the voices of different speakers. This is achieved by training the model on speech data from multiple speakers and using a speaker embedding network to encode information about the desired speaker into the input features.

Tacotron 2 also uses a technique called the attention mechanism to improve the alignment between the input text and the generated speech. The attention mechanism allows the decoder network to selectively focus on different parts of the input sequence when generating each output frame.

2.2.10 Location Based Attention Network

Location-based attention network is a type of attention mechanism used in sequence-to-sequence models such as those used in text-to-speech (TTS) systems.

The idea behind location-based attention is to provide the model with information about where to focus its attention when generating each frame of the mel spectrogram. This is achieved by adding an extra input to the attention mechanism that encodes the current position in the input sequence, known as the ”location” vector. This location vector is a learned parameter that is updated during training along with the other parameters of the model.

The location-based attention network uses this location vector to compute a set of attention weights that determine how much attention to give to each input frame when generating the output. The attention weights are computed using a softmax function applied to a score function that takes as input the previous hidden state of the decoder and a context vector computed from the encoder outputs.

By incorporating information about the location of the input sequence into the attention mechanism, location-based attention can help the model better align the input and output sequences, resulting in better-quality synthesized speech. It is particularly useful in TTS systems where the length of the input sequence can vary widely, as it allows the model to adapt its attention strategy to the length of the input sequence.

2.2.11 Melspectogram

A Melspectrogram is a visual representation of the power spectrum of an audio signal over time. It is a type of spectrogram, which is a visual representation of the spectrum of frequencies of a signal as it varies with time.

Calculation

The Melspectrogram is calculated by first taking the Short-Time Fourier Transform (STFT) of the audio signal, which is the Fourier Transform of a short window of the signal at a time. The magnitude of the STFT is then squared and passed through a filter bank that is designed to mimic the non-linear frequency response of the human ear. The filter bank is typically made up of triangular filters that are spaced evenly on the Mel scale, which is a perceptual scale of pitches judged by listeners to be equal in distance from one another.

Applications

The Melspectrogram is widely used in audio signals processing tasks such as speech recognition, music genre classification, and audio event detection. It is particularly useful in tasks that require capturing the spectral content of audio signals while ignoring some of the noise and other variations that are not as relevant to the task at hand.

Parameters

The Melspectrogram can be customized by adjusting several parameters such as the window length, filter length, and hop length. The window length determines the size of the window used in the STFT, the filter length determines the number of filters in the filter bank, and the hop length determines the step size used in sliding the window over the signal.

2.2.12 Fine-tuning

Fine-tuning a text-to-speech (TTS) model involves training the existing pre-trained model on new data specific to the desired application or language. The process of fine-tuning allows the TTS model to adjust its parameters and learn from the new data to generate more accurate and natural-sounding speech.

The pre-trained TTS models are trained on large datasets of speech data with various linguistic features and variations in speech patterns. However, these models may not perform well when it comes to specific applications, such as generating speech in a particular language, accent, or voice. This is where fine-tuning comes into play, allowing the model to adapt to the new data and generate more accurate and natural-sounding speech.

Fine-tuning a TTS model typically involves feeding the model with new training data specific to the desired application or language. The training data includes transcripts of text, recorded speech, and linguistic features specific to the language or accent. The model is then trained using the backpropagation algorithm to adjust its parameters and minimize the difference between the predicted speech and the actual speech.

During the fine-tuning process, the TTS model's architecture and hyperparameters can also be modified to optimize its performance on the specific task. The fine-tuning process typically requires a large amount of data and computing resources to train the model effectively. However, the benefits of fine-tuning can be significant, resulting in a more accurate and natural-sounding speech in the desired application or language.

2.2.13 Weights and Biases

Weights and Biases, shortly Wandb is a platform that provides tools for tracking, visualizing, and managing machine learning experiments. It is designed to help data scientists and machine learning engineers streamline their workflow and collaborate more effectively on machine learning projects. Wandb provides a range of features to facilitate machine learning experimentation.

2.2.14 Vocoder

A vocoder, short for "voice encoder", is a type of digital signal processor that analyzes and synthesizes human speech and other sounds. It is used in a wide range of applications, including telecommunications, music production, and speech synthesis.

The primary function of a vocoder is to analyze an audio signal and separate it into two components: the spectral envelope, which contains information about the frequency content of the signal, and the excitation signal, which contains information about the sound source and the timing of the sound events.

The spectral envelope is typically represented as a set of linearly-spaced frequency bands, with each band corresponding to a different part of the spectrum. The excitation signal is usually represented as a series of time-domain waveforms or frequency-domain spectra, depending on the specific vocoder algorithm being used. Once the audio signal has been separated into these two components, they can be processed independently and then recombined to create a synthesized version of the original audio signal. This allows for a wide range of creative effects, including pitch shifting, formant shifting, and the creation of robotic or otherworldly vocal effects.

There are many different types of vocoders available, each with its own strengths and weaknesses. Two popular types of vocoders are HiFiGAN and WaveNet.

Hifi-Gan

HiFi-GAN is a state-of-the-art generative adversarial network (GAN) based model for speech synthesis, developed by researchers at the University of Montreal and MILA. The goal of HiFi-GAN is to generate high-quality speech with a high level of fidelity and naturalness, while also being computationally efficient.

HiFi-GAN builds upon the principles of traditional GANs, which use a generator and a discriminator network to learn to generate realistic samples of data. In the case of speech synthesis, the generator network is trained to generate speech waveforms from input features such as Mel-spectrograms, while the discriminator network is trained to distinguish between real and synthetic speech signals.

What sets HiFi-GAN apart from other speech synthesis models is its use of a multi-resolution approach, which allows the model to capture both the low-frequency and high-frequency components of speech signals. This is achieved through the use of a hierarchical generator architecture, where multiple generator networks are trained at different resolutions, with each network focusing on a different frequency range. Additionally, HiFi-GAN incorporates a number of optimization techniques, such as the use of a residual generator and a multi-scale discriminator, which help to improve the quality and efficiency of the model.

Waveglow

Waveglow is a powerful generative model for synthesizing high-quality speech from text, developed by researchers at NVIDIA. It is a type of neural vocoder, which is a model that converts acoustic features into speech waveforms.

Waveglow is built on the principles of generative flow-based models, which use a series of invertible transformations to generate samples from a simple probability distribution. In the case of Waveglow, these transformations are used to generate speech waveforms from input spectrogram representations of the speech signal.

Waveglow uses a modified version of the Glow architecture, which is a type of flow-based model that uses affine coupling layers and invertible 1x1 convolutions to model the probability distribution. The model is trained using a combination of maximum likelihood estimation and an adversarial training objective, which helps to improve the quality and naturalness of the synthesized speech.

One of the key advantages of Waveglow is its ability to generate high-quality speech with a relatively small number of model parameters. This makes it computationally efficient and practical for use in real-world applications. Additionally, Waveglow can be trained in a variety of languages and dialects, making it a versatile tool for speech synthesis.

2.2.15 Mean Opinion Score

Mean Opinion Score (MOS) is a commonly used subjective evaluation metric for speech quality in text-to-speech systems. MOS is a numerical score between 1 and 5 that represents the perceived quality of the speech output by human evaluators. A MOS score of 1 represents the worst possible quality, while a score of 5 represents the best possible quality.

To evaluate the quality of a text-to-speech system using MOS, a group of human evaluators are asked to listen to a set of speech samples generated by the system and rate the quality of each sample on a scale of 1 to 5. The scores are then averaged to produce an overall MOS score for the system.

MOS is a useful metric because it provides a way to measure the subjective quality of speech output in a standardized and quantitative way. However, it is important to note that MOS scores can be influenced by factors such as the experience and expertise of the evaluators, the characteristics of the speech samples being evaluated, and the conditions under which the evaluations are conducted. Therefore, it is important to use MOS scores in conjunction with other objective evaluation metrics to get a comprehensive picture of the performance of a text-to-speech system.

2.2.16 Audio Processing

After text-to-speech (TTS) conversion, the generated audio signal may require some additional processing to improve its quality or to make it suitable for specific applications. Here are some common audio processing techniques used after TTS:

- **Noise Reduction :** It involves applying a filter to the synthesized speech signal after the TTS conversion to remove any noise or distortion introduced during the conversion process.

- **Loudness Normalization :** Loudness normalization is a technique used to adjust the overall loudness of the synthesized speech to a specific level. This technique ensures that the loudness of the speech is consistent across different devices and environments.

- **Pitch Modification :** Pitch modification is a technique used to change the fundamental frequency of the synthesized speech. This technique is often used to create variations in the speech output or to match the speech to a specific application.

- **Voice Conversion :** Voice conversion is a technique used to modify the characteristics of the synthesized speech to make it sound like a different speaker. This technique involves analyzing the speech signal of the target speaker and then modifying the synthesized speech signal to match the target speaker's characteristics.

2.2.17 Python Packages for Audio Processing

NoiseReduce

noisereduce is a Python library for reducing noise in audio signals. It uses spectral subtraction, a common method for noise reduction.

Spectral subtraction works by estimating the noise spectrum and then subtracting it from the noisy signal spectrum to obtain an estimate of the clean signal spectrum. The clean signal is then reconstructed by taking the inverse Fourier transform of the estimated clean signal spectrum.

Normalize

effects.normalize is a function that is used to normalize an audio signal. Normalization is the process of adjusting the amplitude of an audio signal to bring its maximum level to a specified value without altering the relative amplitudes of other parts of the signal. It is a common technique used to ensure that the audio signal does not clip or distort when played back at a higher volume.

The normalize function in Python's *pydub* library scales the amplitude of the audio signal to a specified peak amplitude. The function takes an audio segment as input and returns the normalized segment. The normalization process involves finding the maximum amplitude of the audio signal, and then scaling the signal by a factor that will bring the maximum amplitude to the specified peak level.

2.2.18 Server and Applications

Flask

Flask is a micro web framework written in Python. It is designed to be simple and lightweight, making it an ideal choice for small to medium-sized web applications. Flask allows developers to quickly and easily create web applications by providing a set of tools and libraries that handle the common tasks associated with web development, such as routing, templating, and database integration.

One of the key features of Flask is its flexibility. It is designed to be modular and extensible, which means that developers can easily add or remove functionality as needed. This makes it a popular choice for building RESTful APIs, web services, and other types of web applications.

React Native

React Native is a framework for building mobile applications using JavaScript and React. It allows developers to write code once and deploy it across multiple platforms, including iOS and Android. This can save a significant amount of time and effort, as developers do not need to write separate codebases for each platform.

One of the key features of React Native is its ability to create a native user interface using a combination of JavaScript and native code. This means that the user interface of a React Native app looks and feels like a native app, rather than a web app. Additionally, React Native provides a large set of pre-built components that can be used to quickly build complex user interfaces.

3. System Design

3.1 Requirement Specification

3.1.1 Functional Requirements

Functional requirements help to ensure that the system provides the necessary features and capabilities to meet the needs of the users and the business.

System Side

- The system should be able to take input text in the Nepali language and convert it into an audio file.
- The system should convert numbers to words for better voice synthesis.
- The system should allow users to control various parameters of speech synthesis, such as pitch, speed, and volume.
- The system should be able to process large volumes of text quickly and efficiently.

User's Side

- The user should be able to upload a Nepali text file into the system for conversion.
- The user should be able to preview the synthesized audio before downloading it.
- The user should be able to control various parameters of speech synthesis, such as pitch, speed, and volume.

3.1.2 Non-Functional Requirements

Non-functional requirements describe the quality characteristics or attributes of the system. They are the attributes of the system that can be measured, evaluated or tested, and they are not directly related to the system's functionality. Non-functional requirements specify how well the system should perform, rather than what it should do.

- **Performance :** The system must be able to handle a certain amount of load or number of users without slowing down or crashing.
- **Availability :** The system should be available to users at all times. This means that the system should have high uptime and minimal downtime.
- **Fault Tolerant :** The system should be able to continue operating even in the event of a failure or error. This means that the system should have built-in redundancy and failover mechanisms to ensure that critical functionality is always available.
- **Scalability :** The system must be able to handle growth and expansion without needing significant changes or upgrades.
- **User Friendly :** The system should be easy to use and navigate, with a simple and intuitive interface. This is important for the Nepali Book Reader, as users of all ages and backgrounds may be using the system.
- **Compatibility :** The system must be compatible with various operating systems, browsers, and devices.

Non-functional requirements are important to ensure that the system not only functions properly but also meets the needs and expectations of its users.

3.2 Hardware and Software Requirement

3.2.1 Hardware Requirements

- **Server :** A server is required to host the TTS model and serve the audio files to the user. The server should have a stable internet connection and enough processing power to handle multiple requests simultaneously.
- **RAM :** The server should have at least 8GB of RAM to handle the TTS model and other processes.
- **Storage :** The server should have enough storage to store the TTS model and audio files. At least 100GB of storage is recommended.
- **PCs:** Users will require a PC to access the web-based application. The PC should have a stable internet connection and a modern web browser such as Google Chrome or Mozilla Firefox.

3.2.2 Software Requirements

- **Operating System :** The system should support a Unix/Linux operating system as the required software is mostly compatible with it.
- **TTS Pre-trained Model :** The project requires a pre-trained TTS (Text-to-Speech) model. The model should be compatible with the chosen GPU and its dependencies.
- **Audio processing software :** The project requires audio processing software to perform various operations such as noise reduction, normalization, and segmentation.
- **Android/Web browser :** The project requires a mobile application or a web browser to run the Nepali Book Reader.
- **Other Utilities :** The project may require other utilities such as text editors, version control systems, and IDEs (Integrated Development Environments) for development and testing.

3.2.3 Network Requirement

- **Internet Connection :** An internet connection is necessary for the application to function properly.

3.3 Feasibility Assessment

Feasibility assessment is an important aspect of any project, as it helps to determine the viability and practicality of the project. In the case of the Nepali book reader, several aspects need to be evaluated to determine its feasibility, such as technical, operational, economic, legal, and scheduling feasibility.

3.3.1 Technical Feasibility

The technical feasibility of the project refers to the ability to implement the necessary hardware, software, and other technologies required to develop and deploy the system.

For this project, the hardware and software requirements have already been identified, and it has been determined that they are available and accessible. However, the development of the TTS model and its fine-tuning may require significant computing resources, which could be a challenge for smaller organizations or individuals.

3.3.2 Operational Feasibility

Operational feasibility refers to whether the proposed system will be used effectively and efficiently after it is implemented. The system should be user-friendly, meet the needs of users, and not require too much training or support. The system's success depends on whether users find it easy to use and whether it meets their needs. Therefore, user testing and feedback will be essential during the development process to ensure the system's operational feasibility.

3.3.3 Economic Feasibility

Economic feasibility refers to whether the project is financially viable and can be completed within the budget and resources available. The project's cost must be justified by the benefits it will provide.

In the case of the Nepali Book Reader, the primary cost will be the development of the TTS model and the associated hardware and software requirements. However, the potential benefits of the system, such as increased accessibility to literature for visually impaired or struggling readers, justify the costs.

3.3.4 Legal Feasibility

Legal feasibility refers to whether the proposed system complies with all applicable laws, regulations, and standards. This includes issues such as data privacy and security, copyright infringement, and accessibility compliance. It will be important to ensure that all literature used in the system is obtained legally and that the system complies with all applicable accessibility guidelines.

3.3.5 Scheduling Feasibility

Scheduling feasibility refers to whether the proposed project can be completed within the desired timeframe. This includes factors such as the availability of resources, potential delays, and time required for testing and feedback.

Scheduling feasibility may challenging as the project involves multiple stages, such as data collection, TTS model training, and software development. However, with proper planning and resource allocation, the project can be completed within a reasonable timeframe.

3.4 UML Diagrams

3.4.1 Use Case Diagram

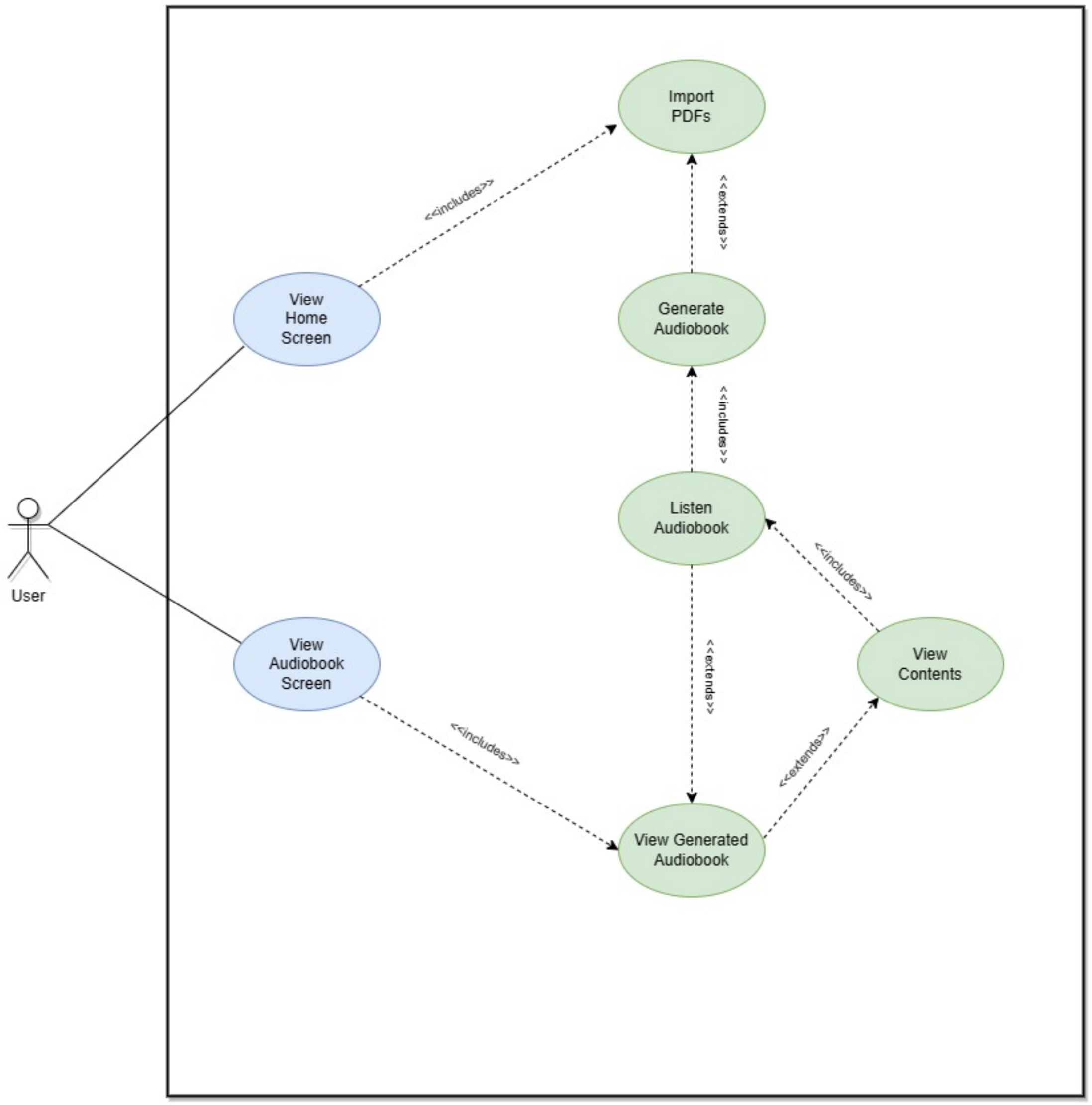


Figure 3.1: Use Case Diagram

3.4.2 Sequence Diagram

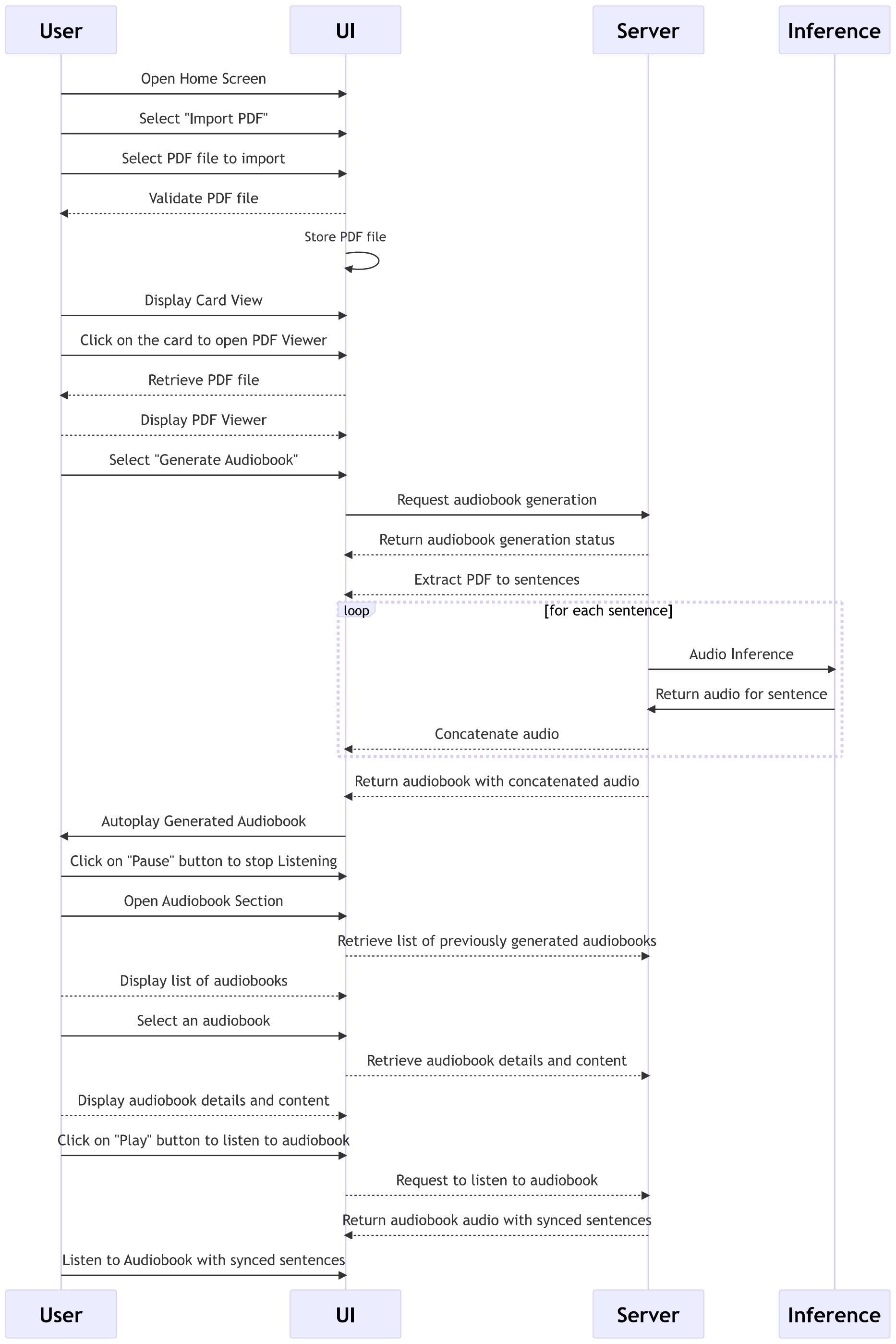

User
UI
Server
Inference
Open Home Screen
Select "Import PDF"
Select PDF file to import
Validate PDF file
Store PDF file
Display Card View
Click on the card to open PDF Viewer
Retrieve PDF file
Display PDF Viewer
Select "Generate Audiobook"
Request audiobook generation
Return audiobook generation status
Extract PDF to sentences
loop
[for each sentence]
Audio Inference
Return audio for sentence
Concatenate audio
Return audiobook with concatenated audio
Autoplay Generated Audiobook
Click on "Pause" button to stop Listening
Open Audiobook Section
Retrieve list of previously generated audiobooks
Display list of audiobooks
Select an audiobook
Retrieve audiobook details and content
Display audiobook details and content
Click on "Play" button to listen to audiobook
Request to listen to audiobook
Return audiobook audio with synced sentences
Listen to Audiobook with synced sentences
User
UI
Server
Inference

Figure 3.2: Sequence Diagram

4. Methodology

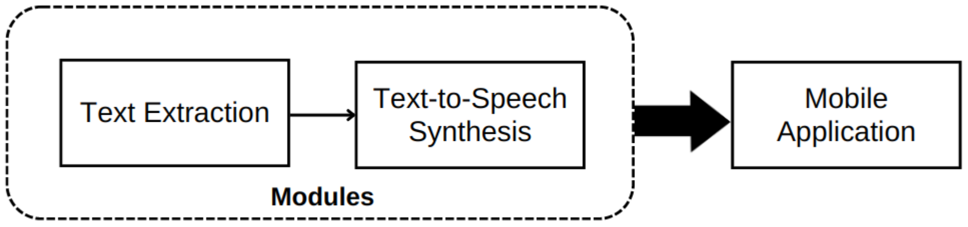


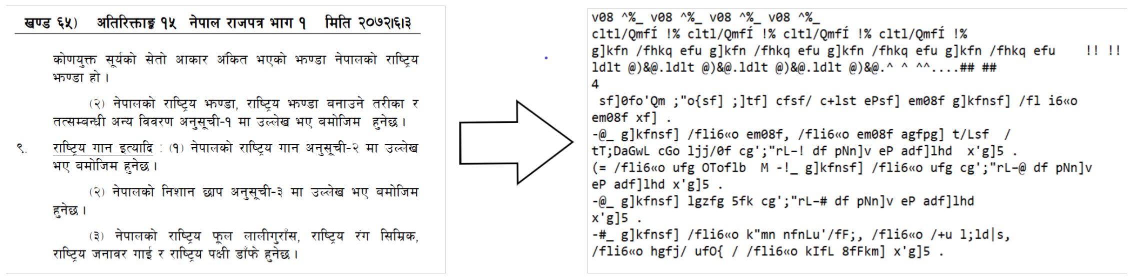
Figure 4.1: Overall System Block Diagram

The overall system can be classified into three components: Text Extraction Module, Text-to-Speech Synthesis Module and Mobile Application. The user inputs a book in the PDF format. The text is extracted from the PDF, preprocessed, tokenized and sent to the Speech Synthesis Module. Text-to-Speech synthesis works in two phases: Melspectrogram Generation and Vocoder Output. A modified version of Tacotron2 has been used for Melspectrogram generation, and HifiGAN and WaveGLOW have been used to generate a vocoder output. The data has been recorded to generate natural-sounding audio with prosody and intonation. A server has been set up to integrate all the different modules, which are bundled together in a mobile application.

4.1 Text Extraction

The first step in the process is to extract the content from the PDF file. The text extraction process extracts content from the PDF to Unicode text format which is processed and then fed to the TTS model for speech generation.

There are mainly 2 methods of text extraction, PDF Parsing and OCR. PDF Parsing is a very fast and accurate method. Parsers can extract required elements from the PDF such as text, image, and metadata. However, if the required data is within an image, it can only extract the image but not the content within the image. Moreover, PDF Parsers are not trained in Nepali fonts. If a PDF is written in Nepali Unicode, parsers can accurately extract the content but Nepali fonts such as Preeti, Sagarmatha, Mangal, Kanchan etc are not Unicode compatible. Hence, a parser extracts the English letter correspondence of the Nepali character which in turn has to be converted to Nepali content through one-to-one mapping, increasing error on the final result.



खण्ड ६५)   अतिरिक्ताङ्क १५   नेपाल राजपत्र भाग १   मिति २०७२।६।३

कोणयुक्त सूर्यको सेतो आकार अंकित भएको झण्डा नेपालको राष्ट्रिय झण्डा हो।

(२) नेपालको राष्ट्रिय झण्डा, राष्ट्रिय झण्डा बनाउने तरीका र तत्सम्बन्धी अन्य विवरण अनुसूची-१ मा उल्लेख भए बमोजिम हुनेछ।

९. राष्ट्रिय गान इत्यादि : (१) नेपालको राष्ट्रिय गान अनुसूची-२ मा उल्लेख भए बमोजिम हुनेछ।

(२) नेपालको निशान छाप अनुसूची-३ मा उल्लेख भए बमोजिम हुनेछ।

(३) नेपालको राष्ट्रिय फूल लालीगुराँस, राष्ट्रिय रंग सिम्रिक, राष्ट्रिय जनावर गाई र राष्ट्रिय पक्षी डाँफे हुनेछ।

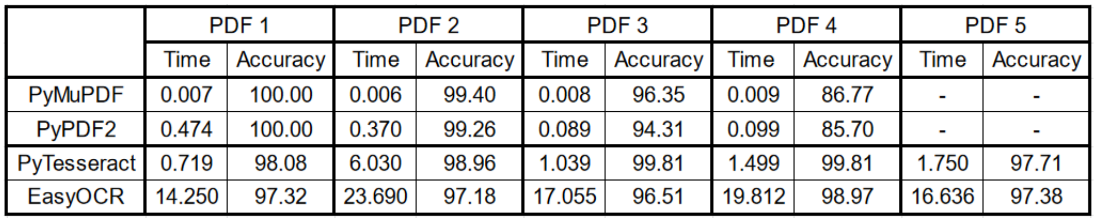
```
v08 ^%_ v08 ^%_ v08 ^%_ v08 ^%_
cltl/QmfÍ !% cltl/QmfÍ !% cltl/QmfÍ !% cltl/QmfÍ !%
g]kfn /fhkq efu g]kfn /fhkq efu g]kfn /fhkq efu g]kfn /fhkq efu    !! !!
ldlt @)&@.ldlt @)&@.ldlt @)&@.ldlt @)&@.^ ^ ^^....## ##
4
 sf]0fo'Qm ;"o{sf] ;]tf] cfsf/ c+lst ePsf] em08f g]kfnsf] /fl i6«o
em08f xf] .
-@_ g]kfnsf] /fli6«o em08f, /fli6«o em08f agfpg] t/Lsf  /
tT;DaGwL cGo ljj/0f cg';"rL-! df pNn]v eP adf]lhd  x'g]5 .
(= /fli6«o ufg OToflb  M -!_ g]kfnsf] /fli6«o ufg cg';"rL-@ df pNn]v
eP adf]lhd x'g]5 .
-@_ g]kfnsf] lgzfg 5fk cg';"rL-# df pNn]v eP adf]lhd
x'g]5 .
-#_ g]kfnsf] /fli6«o k"mn nfnLu'/fF;, /fli6«o /+u l;ld|s,
/fli6«o hgfj/ ufO{ / /fli6«o kIfL 8fFkm] x'g]5 .
```

Figure 4.2: Extraction of Unicode Incompatible Font

Most of these problems are addressed by OCR which extracts the content visually. Although OCR requires an intermediate step of creating an image of the PDF, it is not limited to digitally created PDFs and can even extract the content from scanned PDFs. So, even though this method is comparatively slower than parsing, it is not dependent on the Unicode compatibility of the font as well as the type of PDF.

EasyOCR and PyTesseract are the widely used Python OCR libraries supporting the Devanagari script. EasyOCR utilizes GPU in order to accelerate character recognition. However, without GPU, it defaults to CPU and is significantly slower than PyTesseract.

PyTesseract, an abbreviation of Python-tesseract, is a Python wrapper for Google's Tesseract OCR Engine. It is based on Long Short Term Memory (LSTM) Neural Network and is one of the most accurate open-source OCR engines. It supports a total of 116 languages including Nepali as well as Devanagari script & can read all image types supported by the Pillow library.



	PDF 1		PDF 2		PDF 3		PDF 4		PDF 5	
	Time	Accuracy	Time	Accuracy	Time	Accuracy	Time	Accuracy	Time	Accuracy
PyMuPDF	0.007	100.00	0.006	99.40	0.008	96.35	0.009	86.77	-	-
PyPDF2	0.474	100.00	0.370	99.26	0.089	94.31	0.099	85.70	-	-
PyTesseract	0.719	98.08	6.030	98.96	1.039	99.81	1.499	99.81	1.750	97.71
EasyOCR	14.250	97.32	23.690	97.18	17.055	96.51	19.812	98.97	16.636	97.38

Figure 4.3: Comparison table of different extraction methods

Different methods are compared on different types of PDFs. PDF 1 and 2 contain Nepali Unicode characters and hence PDF Parsers such as PyMuPDF and PyPDF2 have high extraction accuracy. PDF 3 and 4 are written using Unicode-incompatible

Nepali fonts which have to be translated to Nepali after the extraction process, reducing the overall accuracy. The content in PDF 5 is embedded in an image. So, PDF Parsers cannot extract the content at all. Overall, the time taken by Parsers is significantly lower than the OCR but depending on the type of PDF, the output accuracy fluctuates. On the other hand, OCRs on average have greater extraction time but almost constant accuracy as shown in the graph below:

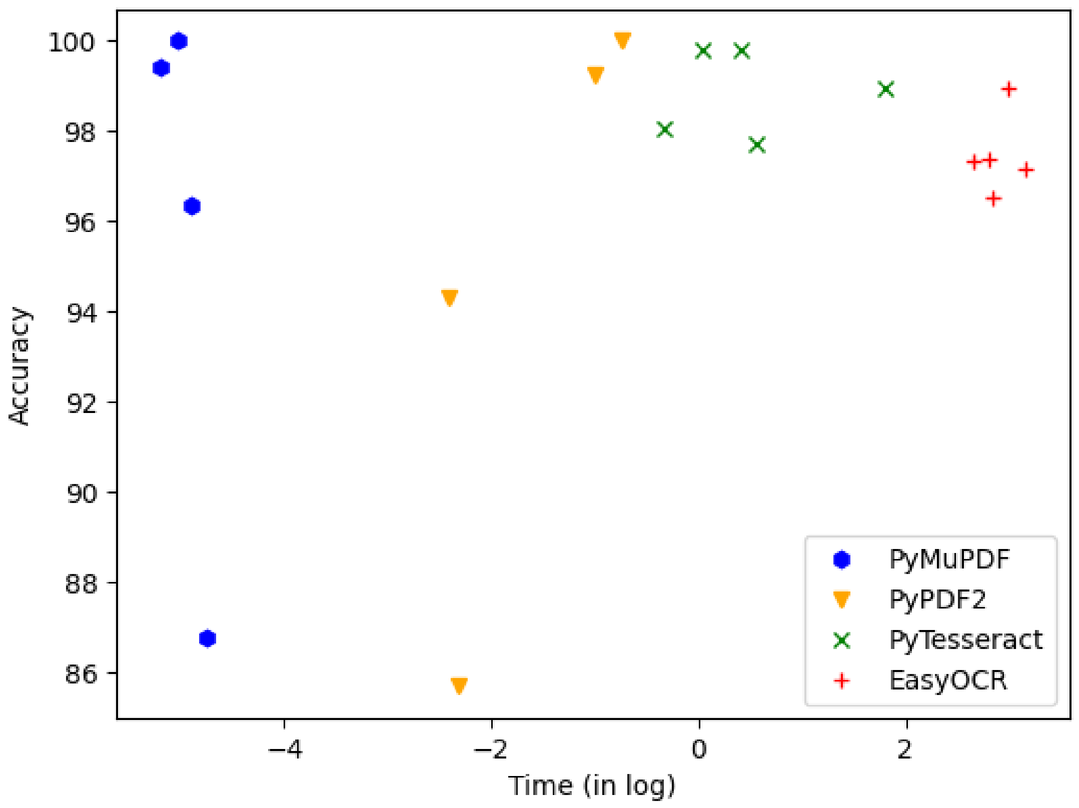


Figure 4.4: Comparison graph of different extraction methods

Hence, weighing the time and accuracy aspect of the extraction, PyTesseract is considered to be the most suitable library for extracting the PDF in this project.

PyTesseract only reads images, not PDFs. So, the pages have to be converted to images to be fed to the OCR engine. Hence python PDF library PyMuPDF is used to generate an image of the page which is processed by PyTesseract. The extracted data is stored as a string of Nepali Unicode characters which may consist of unnecessary characters or in the incorrect form to be passed into the model. This string is processed as necessary and finally fed to the TTS model.

4.2 Text-to-Speech Synthesis

The preprocessed text obtained from the text extraction module is sent to the text-to-speech synthesis module. The TTS module is made up of two distinct components: the Spectrogram Prediction component and the Vocoder component. The output in the form of Mel spectrogram is obtained from the first component. The second component maps from mel spectrograms to waveforms. Generating waveforms from intermediate representations like spectrograms is called vocoding and this second component is called a vocoder.

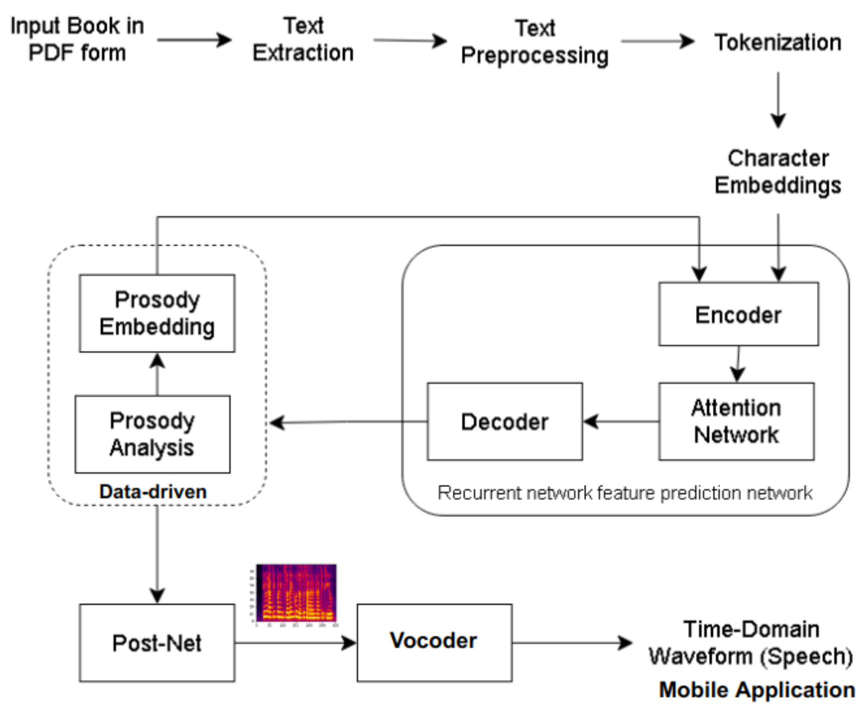


Figure 4.5: Text-to-Speech System Block Diagram

4.2.1 Spectrogram Prediction

The first component of the text-to-speech synthesis task is spectrogram prediction. Spectrogram Prediction task uses an encoder-decoder with an attention model, which maps a string of letters to mel spectrographs: sequences of mel spectral values over time. The encoder's job is to take a sequence of letters and produce a hidden representation representing the letter sequence, which is then used by the attention mechanism in the decoder.

Character Embedding

Each word token is initially transformed into a character embedding. Character Embedding in this project is handled as a part of the Tacotron2 system architecture. It handles the OOV(Out of Vocabulary) words and uses one-dimensional CNN to find numeric representations of words by looking at their character-level composition. The scanners can focus on several characters at a time, and the information from different scanners can collectively represent a word.

Feature Prediction Network

The character embeddings are fed into an attention-based recurrent sequence-to-sequence feature prediction network. The spectrogram's sequence is predicted by the sequence-to-sequence network. The attention network for converting or mapping character embeddings to a spectrogram is part of the feature prediction network together with an encoder and decoder network.

- **Encoder Network**: The encoder network encodes every input grapheme to a 512-dimensional character embedding. These are then passed through a stack of 3 convolutional layers, each containing 512 filters with a shape 5×1, i.e. each filter spanning 5 characters, to model the larger letter context. The output of the final convolutional layer is passed through a biLSTM to produce the final encoding.

- **Attention Network**: Encoded output is consumed by the attention network which summarizes the full-encoded sequence as a fixed-length context vector for each decoder output step. It's common to use a slightly higher quality (but slower) version of attention called location-based attention, in which the computation of the α values (vectorized dot-product attention scores) makes use of the α values from the prior time state. [28]

- **Decoder Network**: The decoder network (autoregressive recurrent neural network - RNN) uses the attention network's output and extrapolates from the hidden feature representation the spectrogram's sequence. The predicted mel spectrum from the prior time slot is passed through a small pre-net as a bottleneck. This prior output is then concatenated with the encoder's attention vector context and passed through 2 LSTM layers. The output is used in two ways.

  1. It is passed through a linear layer, and some output processing, and sent to 5 convolution layer post-net, where prediction is done one frame at a time.

2. It is passed through another linear layer to a sigmoid to make a "stop token prediction" decision about whether to stop producing output.

Post-Net

After the decoding phase, the mel-spectrogram is fed into the Post-Net. It is a stack of convolution layers, which can capture the features from both past and future contexts in sequence.

4.2.2 Waveform Generation

For the waveform generation, the spectrogram is then fed into Vocoder which creates time-domain waveforms (Speech). Vocoder is an abbreviation for Voice + Encoder. The human voice signal is extracted from the spectrogram and synthesized using the vocoder.

The spectrogram prediction encoder-decoder and the vocoder are trained separately. In our project, Tacotron2 is accompanied by two vocoder models HiFiGAN and WaveGlow. A combination of Glow and Wavenet, WaveGlow generates high-quality audio using mel-spectrograms. Hifi-GANs are Generative Adversarial Networks that are capable of efficiently and accurately synthesizing speech using given mel-spectrograms. The finetuned data and the final mobile application both use Hifi-GAN as the vocoder due to relatively noise-free speech synthesis.

4.3 Data Collection

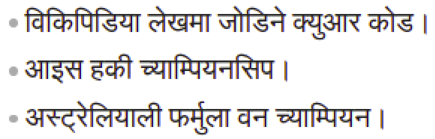
The quality and quantity of data are fundamental to any deep learning system. In TTS, it's less crucial to use multiple voices, and so basic TTS systems are speaker-dependent: trained to have a consistent voice, on much less data, but all from one speaker. [27] We collected data from three different sources: OpenSLR Dataset, Self Recorded Data, Youtube Audiobook Clippings

4.3.1 OpenSLR Female Dataset

We used *High-Quality TTS data for Nepali* from OpenSLR[29] as our primary data source. It contains 2.8 hours of data having 2064 sentences(47114 words) and high-quality audio clips of 1-13 seconds associated with each sentence from 19 female speakers.

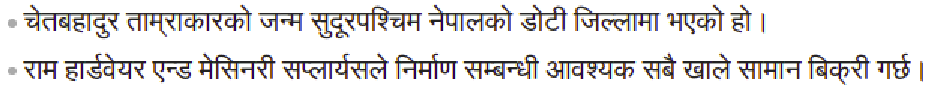
Data Insights

1. The data contains a lot of named entities like people's names, foreign transliterated/romanized words and country's and people's names



- विकिपिडिया लेखमा जोडिने क्युआर कोड।
- आइस हकी च्याम्पियनसिप।
- अस्ट्रेलियाली फर्मुला वन च्याम्पियन।

2. Many sentences are news articles like statements which might not favor our audiobook kind of speech.



- चेतबहादुर ताम्राकारको जन्म सुदूरपश्चिम नेपालको डोटी जिल्लामा भएको हो।
- राम हार्डवेयर एन्ड मेसिनरी सप्लार्यसले निर्माण सम्बन्धी आवश्यक सबै खाले सामान बिक्री गर्छ।

3. It contains many repetitive phrases.

4.3.2 Self-Recorded Data

The data obtained from OpenSLR did not meet the requirements of prosody. So, we recorded our audio clips in male and female voices. It contains 1.2 hours of data having 666 sentences(20690 words) and high-quality audio clips of 1-14 seconds associated with each sentence from 2 speakers.

We followed certain guidelines while recording the data:

1. Keeping the recording device at a fixed distance from the speaker
2. Keeping a fixed amount of silence before and after recording a clip
3. Maintaining the same volume throughout the clip
4. Keeping the voice low at the beginning

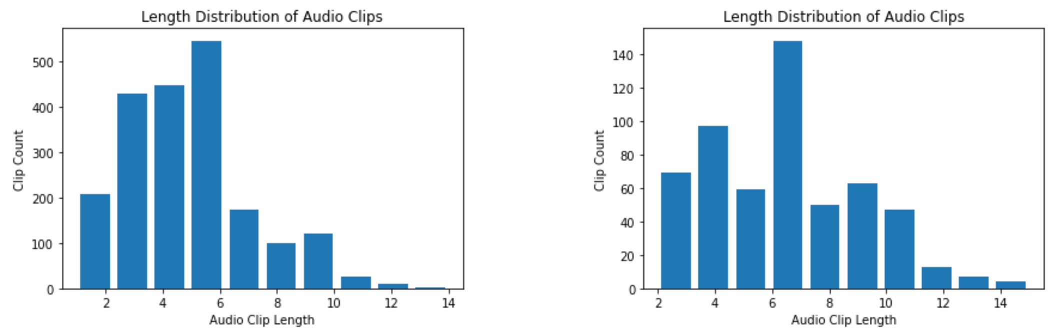
Data Insights

1. Text data obtained from online essay collections, news articles and *Shwet Bhairavi*[30] e-book.
2. 557 sentences female recording and 109 sentences male recording

4.3.3 OpenSLR and Self-Recorded Data Comparison

	OpenSLR	**Self-Recorded**
Speakers	19	2
Audio Samples	2064	666
Gender	All Female	Male and Female
Total Duration	2.8 hours	1.2 hours

Table 4.1: OpenSLR vs Self-Recorded Data



(a) OpenSLR Audio (b) Self-Recorded Audio

Figure 4.6: Audio Length Distribution Comparison

4.3.4 Youtube Audio Clippings

Audiobook Recordings from Youtube were also considered briefly as a part of our dataset. 160 audio clips of 1-25 seconds were obtained by processing Youtube videos of book recitations. The clips had heavy background music, which, even after audio processing, did not decrease significantly. The synthesized speech from those clips was very noisy and entirely unintelligible. Therefore, these clips are not included in our official training dataset.

4.4 Data Preprocessing

4.4.1 Audio Preprocessing

1. The file type was converted to a common format of .wav from different types like .flac, .m4a, .mp3 and .mp4
2. The audio clips were converted from a sampling rate of 48kHz to 22.05kHz.

3. The stereo was set to mono.
4. Silence was removed from the beginning and end if there was any.
5. The long silences and other noises(like music in audiobooks) were removed from the middle of the audio clips
6. Very long clips were clipped in between

4.4.2 Text Preprocessing

1. The numerals, years and dates are converted into text using a Python script.
2. Unwanted and unseen characters like symbols were removed
3. Lengthy sentences were broken down into smaller ones
4. Appropriate stop token was added manually at the end of each sentence.

Raw Text	Preprocessed Text
१०० खेल खेल्ने	एक सय खेल खेल्ने
१९६७ भारतमा	उन्नाइस सय सतसट्ठी भारतमा

Table 4.2: Text Preprocessing

4.5 Tools and Technologies Used

4.5.1 Text Extraction

- **Pytesseract** - used for extracting the content from the PDF
- **PyMuPDF** - for generating image of the PDF pages

4.5.2 Text-to-Speech Synthesis

- **ffmpeg** - Audio conversion
- **Torch** - a scientific computation network framework and a machine learning library
- **Weights and Biases** - for plotting the validation loss

4.5.3 Server Setup

- **flask** - Web framework in Python
- **pydub** - Python package for noise reduction and loudness normalization

4.5.4 Mobile Application

- **React Native** - for creating frontend of the application
- **Expo** - for building and running the application

4.6 TTS Training

4.6.1 Pretraining

The training of the model is done in two phases. The first phase is pretraining with English, which is followed by Nepali data. The first phase involves using an English model provided by Nvidia's official implementation of Tacotron-2. Due to inadequate training resources, we couldn't train the entire model from scratch. Hence, the next step of pretraining followed by finetuning is done above english the TTS system which was trained on the LJ speech corpus dataset.[31]

The second phase of pretraining is done with the high quality TTS dataset from OpenSLR. Initially, the training was done with a sampling rate of 48kHz for 60 epochs. But the voice obtained was unintelligible with a very high pitch. In order to overcome that issue, the training was done with a sampling rate of 22.05kHz which yielded much better results.

4.6.2 Finetuning

The result obtained from pretraining was understandable but lacked prosody. We adopted a data-driven approach for prosody management by recording the audio with suitable intonations. The method of incremental learning was adopted finetuning where input data was continuously fed to extend the model's vocabulary and pronunciation dictionary. The result obtained from HifiGAN was superior to WaveGLOW in comparison to the pretraining phase, hence only HifiGAN was used for finetuning.

4.7 Experimental Setup

4.7.1 Dataset

The dataset used for pretraining was OpenSLR Nepali Dataset. The dataset was split into the train-to-validation ratio of 90:10. For finetuning, all the collected datasets were used. In the pattern of incremental learning, the model training was ongoing with the recording process. The model was first trained with the female self-recorded dataset in batches. Other attempts to improve the result were done using the YouTube audio clippings and ASR data from OpenSLR[32]. Both of them, being noisy data, yielded very poor results. Finally, finetuning was done separately for male self-recorded voices.

4.7.2 Training Details

Pretraining

The pretraining was first done on 2064 sentences with a sampling rate of 48kHz. The training was done on top of an existing English pre-trained model at the beginning before a checkpoint was loaded. Once the checkpoints had been loaded and the initial setup had been completed, the model was updated every epoch while training. The model was trained for 60 epochs using the free version of Google Colab. The training-to-validation ratio was 90:10.

After getting unintelligible results from 48kHz, the same dataset was trained with a sampling rate of 22.05kHz. The model was trained for 72 epochs with a batch size of 4 and a learning rate of 0.001 using the pro version of Google Colab. The result was much better than the previous attempt, and its details are mentioned in the result section.

The encoder and decoder parameters of the architecture were not tampered with. The required modifications were done with the audio parameters. The window length (win_length) is kept at 1024. The filter length(filter_length), which is typically set to be the same as the win_length, is also 1024. The hop length(hop_length) is set to 256. The maximum wavelength value (max_wav_value) is set to 32768 and the number of mel channels (n_mel_channels) is set to 80.

Finetuning

The model and audio parameters for finetuning were the same as in the second phase of pretraining. Its training-to-validation ratio was, here too, 90:10. The model was trained for 68 epochs with batch size 2 using the free version of Google Colab. The learning rate was 0.001 for finetuning as well.

The finetuning was done on multiple datasets. The iteration using the female self-recorded voice produced the best results, therefore only its details are included, and the same voice was used in the mobile application

Pretrained Vocoder

Among the two kinds of models used in this project, we only trained the Mel spectrogram-generating model. We used pre-trained vocoder models due to the unavailability of huge training resources, but we have used two of them in order to have better means of comparison.

4.7.3 Audio Postprocessing

The self-recorded data is not entirely noise-free as it was not recorded in a studio environment. As a result, the finetuned data is not entirely noise free. Before sending it to the mobile application, it goes through some postprocessing as mentioned below.

1. The model can only generate 1-14 seconds of audio clips, which is not appropriate for an audiobook. So, the clips are concatenated in the final result.
2. Noise is removed from the generated audio using Python scripts.
3. The audio is finally normalized using Python scripts after the noise removal.

4.8 Application Development

Our project's aim goes beyond generating text-to-speech. We believe that research without practical application is not productive. Hence, we have developed a mobile application that incorporates both a text extraction module and a text-to-speech synthesis module.

4.8.1 Mobile Application

The mobile application was developed using React Native with Expo. React Native is a popular framework for building native mobile applications using JavaScript and allows for cross-platform development, making it easier to develop and maintain a single codebase for both iOS and Android platforms. Expo is a toolchain built around React Native to help with the development and deployment of React Native applications.

The application provides users with the ability to read PDF files and generate audiobooks with just a few taps. The main features of the application include:

- Reading PDF files: Users can upload and read PDF files directly within the application.
- Request audiobook generation: Users can make a request for an audiobook to be generated from the PDF file they are reading. The application then sends a request to the server to process the request.
- View previously generated audiobooks: Users can view a list of audiobooks that they have previously generated and listen to them directly within the application.

The application was designed with simplicity and ease of use in mind. Users can navigate the application easily and generate audiobooks with just a few taps. The use of React Native also allowed for the application to be developed quickly and efficiently, saving development time and resources.

4.8.2 Server Development

The server component of the Nepali Book Reader system was developed using the Flask web framework. The server is responsible for providing endpoints for the mobile application to communicate with, as well as integrating the text extraction and TTS model components. The server also acts as a database for storing the audio files generated by the TTS model.

Text Extraction Integration

To enable the TTS model to generate audio files from text, a text extraction component was integrated into the server. The text extraction component extracts the text from PDF documents uploaded by the user and stores it in a database. This extracted text is then passed to the TTS model for generating the corresponding audio files.

TTS Model Integration

The TTS model used in the Nepali Book Reader system is based on the Tacotron 2 architecture. The pre-trained model was fine-tuned using a dataset of Nepali speech recordings. The fine-tuned model was then integrated into the Flask server, allowing the mobile application to generate audio files for the extracted text.

API Endpoints

The server provides several API endpoints for the mobile application to communicate with. These endpoints include uploading PDF documents, generating audio files, and retrieving previously generated audio files.

Database

As our application only supports reading PDF files, we have implemented a simple database system using JSON files. Whenever a user uploads a PDF file, the server extracts the text from the file and stores it in a JSON file along with other relevant information like the file name, date of upload, etc.

For each uploaded PDF file, a separate JSON file is created, which acts as a database for that particular file. This database file is used by the server to generate the audio output using the TTS model.

We chose to use a JSON file for our database system as it is lightweight and easy to handle with Python. However, in future, we will be shifting to use a proper database management system like MySQL or PostgreSQL.

4.8.3 Speech Translation

The developed application is a speech translation application that translates spoken words from English to Nepali and Nepali to English. The development process involved three major components:

- Speech Recognition
- Language Translation
- Text to Speech Conversion

For speech recognition, we used the Google Speech-to-Text API, which is known for its high accuracy in converting spoken words to text. This API can detect and transcribe speech in more than 120 languages and dialects, including Nepali and English. After the speech was transcribed into text, we used the mtranslate, python package for language translation. mtranslate is a machine translation API that uses advanced machine learning algorithms to provide accurate translations between different languages. This API is open-source and free to use. Lastly, to convert the translated text back into speech, we used our Nepali text-to-speech model for Nepali language and gTTS for English.

5. Results and Discussion

5.1 Results

5.1.1 Training Results

The validation loss for pretraining was obtained to be 0.234 after 72 epochs. The best validation loss for finetuning was 0.358 after 68 epochs.

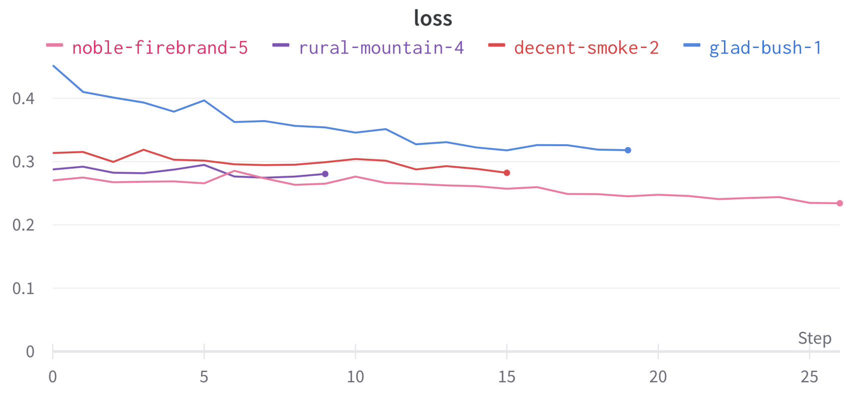


Figure 5.1: Validation Loss for Pretraining

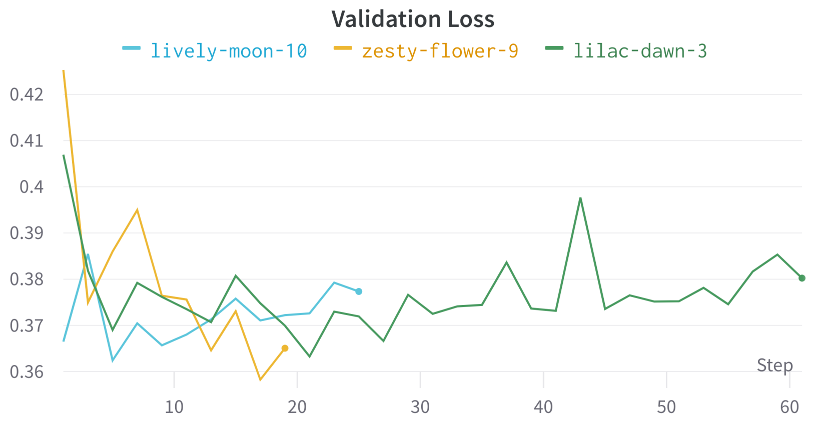


Figure 5.2: Validation Loss for Finetuning

The validation loss fluctuations during the finetuning process can be attributed to the non-studio environment in which the recordings were made. The presence of noise in the input signal resulted in an output that also contained noise, leading to the observed variability in the validation loss.

5.1.2 Mean Opinion Score

We evaluated the quality of synthesized utterances by playing a sentence to listeners and asking them to give a mean opinion score (MOS), a rating of how good the synthesized utterances are, usually on a scale from 1–5. Developing an effective automatic metric for evaluating speech synthesis is an ongoing research area, so MOS can be considered the best metric for speech evaluation as of now.[13]

We conducted the MOS evaluation in two phases. In both phases, a certain number of audio clips were given in a Google Form format to a certain number of volunteers. They scored on the basis of Accuracy and Naturalness on a scale from 1-5, 1 being the worst and 5 being the best.

MOS Phase 1

The main objective of this phase was to determine how much finetuning the model needed, and which vocoder model was better between WaveGlow and HifiGAN.

10 sample sentences were subjected to 18 volunteers using WaveGlow and HifiGAN each. The same sentence was produced using two different vocoders. 88.9% of the volunteers were of the age group 20-39 and 11.1% were below the age group 20. The score was averaged separately for accuracy and naturalness. The result is shown in the table below.

	Naturalness	**Accuracy**
HifiGAN	3.64	3.78
WaveGlow	3.47	3.68

Table 5.1: MOS Phase 1 Result: HifiGAN vs WaveGlow

As shown in the table, HifiGAN was a superior vocoder both in terms of accuracy and naturalness, hence it was used for finetuning.

MOS Phase 2: Finetuning

The next phase of MOS was evaluating the finetuned speech. For this, 28 volunteers were subjected to 12 audio clips and they had to judge on the basis of accuracy and naturalness with a score of 1-5.

60.7% volunteers were of the age 20-39, 25% were below 20 age group and 14.3% were of the age group 40-69. The result is shown below.

	Naturalness	**Accuracy**
HifiGAN	4.04	3.98

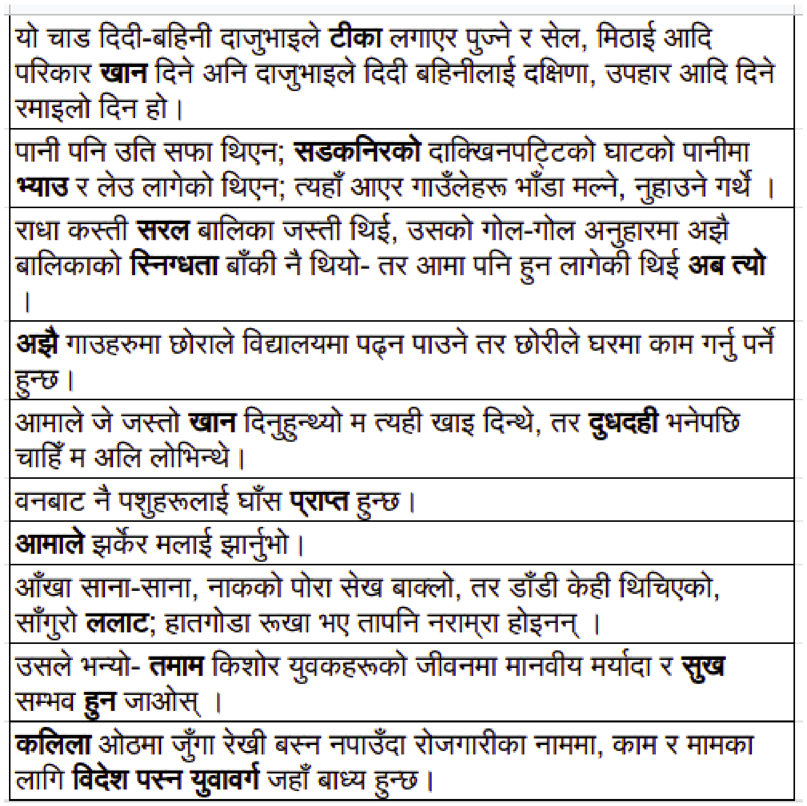
Table 5.2: MOS Finetuning Result

We obtained an improved result after finetuning both in Naturalness and Accuracy.

5.1.3 Accuracy Estimation

MOS, being a subjective measure, cannot be considered a robust mechanism for determining the accuracy of speech. To overcome that deficiency, we devised a mechanism for estimating accuracy by manual checking.

We listened to 80 audio clips with sentences having an average of 17 words. The error was an average of 2.4 per sentence. In the figure below, the highlighted words are the ones our model couldn't pronounce properly. We obtained an accuracy of **86%** by this method. This is still a subjective measure and can be improved a lot.



यो चाड दिदी-बहिनी दाजुभाइले **टीका** लगाएर पुज्ने र सेल, मिठाई आदि परिकार **खान** दिने अनि दाजुभाइले दिदी बहिनीलाई दक्षिणा, उपहार आदि दिने रमाइलो दिन हो।
पानी पनि उति सफा थिएन; **सडकनिरको** दाक्खिनपट्टिको घाटको पानीमा **भ्याउ** र लेउ लागेको थिएन; त्यहाँ आएर गाउँलेहरू भाँडा मल्ने, नुहाउने गर्थे ।
राधा कस्ती **सरल** बालिका जस्ती थिई, उसको गोल-गोल अनुहारमा अझै बालिकाको **स्निग्धता** बाँकी नै थियो- तर आमा पनि हुन लागेकी थिई **अब त्यो** ।
अझै गाउहरुमा छोराले विद्यालयमा पढ्न पाउने तर छोरीले घरमा काम गर्नु पर्ने हुन्छ।
आमाले जे जस्तो **खान** दिनुहुन्थ्यो म त्यही खाइ दिन्थे, तर **दुधदही** भनेपछि चाहिँ म अलि लोभिन्थे।
वनबाट नै पशुहरूलाई घाँस **प्राप्त** हुन्छ।
आमाले झर्केर मलाई झार्नुभो।
आँखा साना-साना, नाकको पोरा सेख बाक्लो, तर डाँडी केही थिचिएको, साँगुरो **ललाट**; हातगोडा रूखा भए तापनि नराम्रा होइनन् ।
उसले भन्यो- **तमाम** किशोर युवकहरूको जीवनमा मानवीय मर्यादा र **सुख** सम्भव **हुन** जाओस् ।
कलिला ओठमा जुँगा रेखी बस्न नपाउँदा रोजगारीका नाममा, काम र मामका लागि **विदेश पस्न युवावर्ग** जहाँ बाध्य हुन्छ।

Figure 5.3: Accuracy Estimation Sample

5.1.4 Melspectrogram Comparison

We also did a simple comparison of the melspectrograms generated by the human voice, and synthesized voice before finetuning and after finetuning.

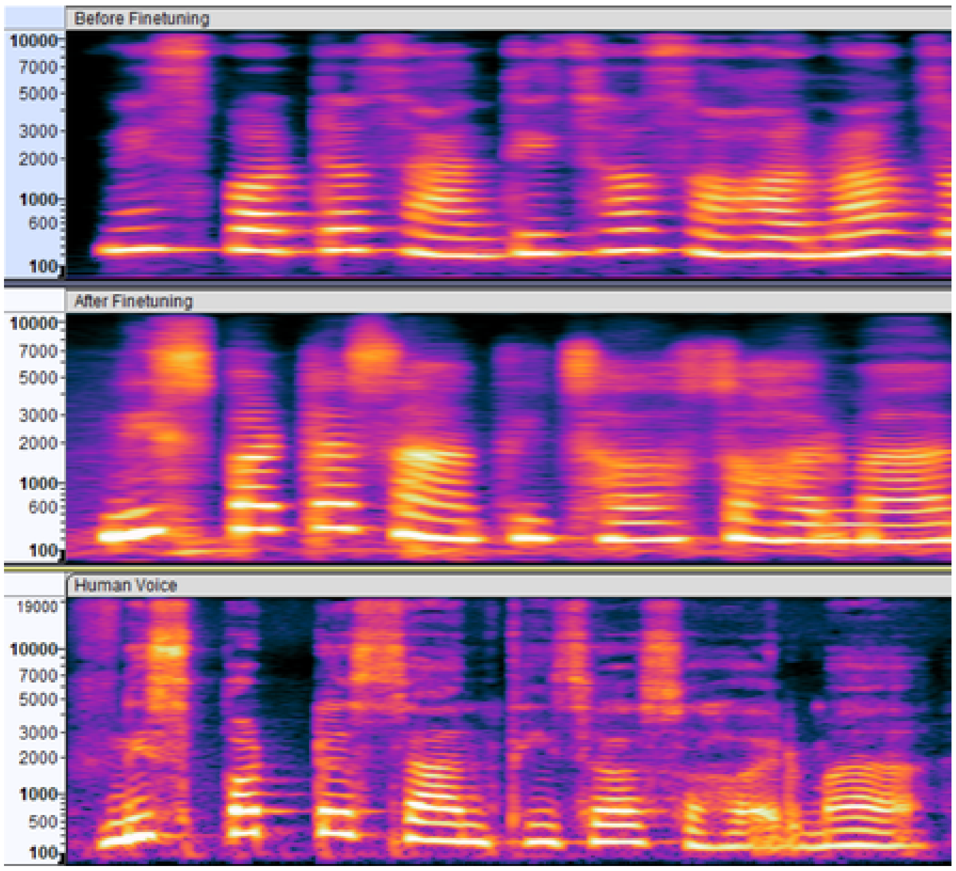


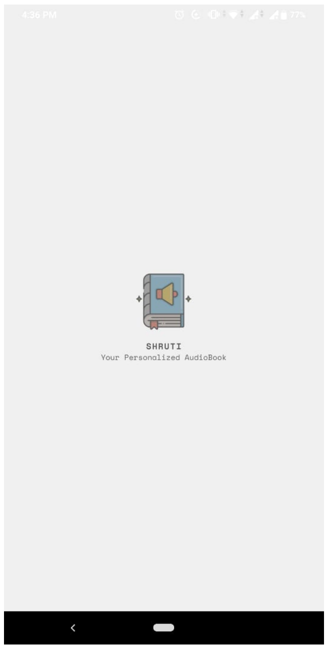
Figure 5.4: Melspectrogram Comparision

We notice the difference in naturalness from the diagram. The human voice is the most natural followed by the finetuned voice followed by the primary training synthesis.

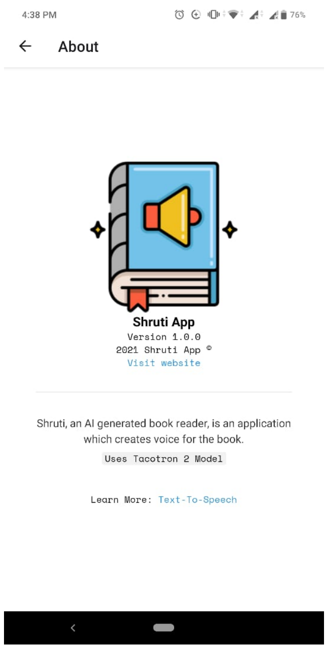
Before finetuning, we notice the frequencies don't diminish between the words. This causes the words to sound the same, and there is no difference in prosody and intonations. The entire sentence is monotone. After fine-tuning, we see that the gaps between the words is much clearer. And in the human voice, every word is clearly differentiable.

Moreover, the higher quality of voice has the frequency component even at the higher ranges whereas the lower quality of voice has few or broken components. This is evident when we compare the frequency range of Human Voice which reaches up to 20000Hz whereas the finetuned model does not. At the higher frequencies in the finetuned model, there is a broken frequency component.

5.2 Mobile Application Output



(a) Splash Screen

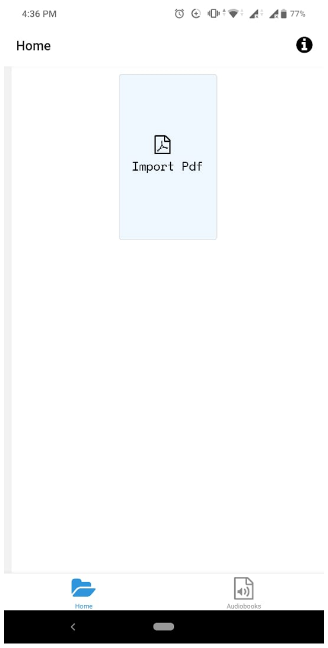


Shruti, an AI generated book reader, is an application which creates voice for the book.

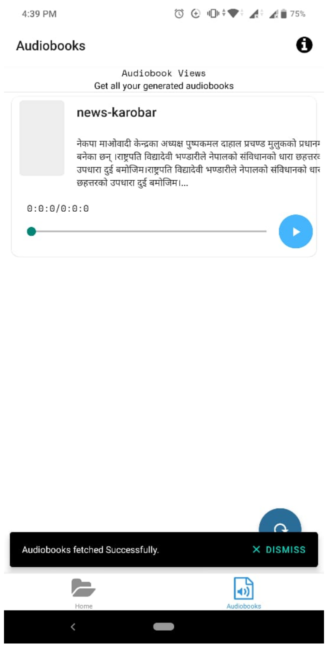
Uses Tacotron 2 Model

Learn More: Text-To-Speech

(b) About App



(c) Home Section



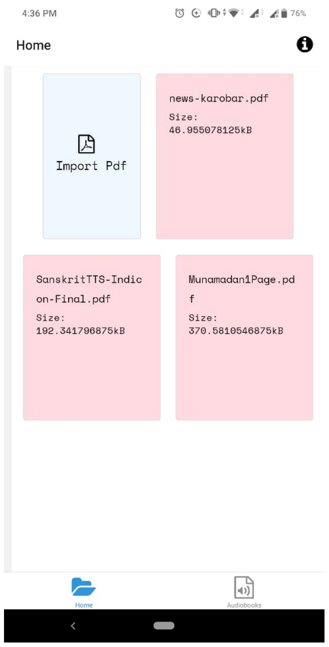
नेकपा माओवादी केन्द्रका अध्यक्ष पुष्पकमल दाहाल प्रचण्ड मुलुकको प्रधान
बनेका छन् ।राष्ट्रपति विद्यादेवी भण्डारीले नेपालको संविधानको धारा छहत्तर
उपधारा दुई बमोजिम।राष्ट्रपति विद्यादेवी भण्डारीले नेपालको संविधानको धा
छहत्तरको उपधारा दुई बमोजिम।...

0:0:0/0:0:0

Audiobooks fetched Successfully.

(d) Audiobook Section

Figure 5.5: Mobile Application Screens - 1


4:36 PM

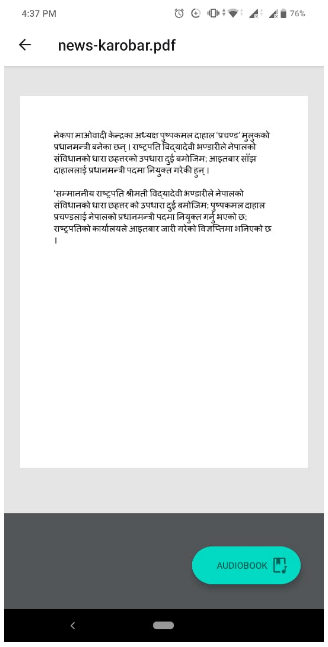
Home

news-karobar.pdf
Size:
46.955078125kB

SanskritTTS-Indicon-Final.pdf
Size:
192.341796875kB

Munamadan1Page.pdf
Size:
370.5810546875kB

(a) Imported PDF

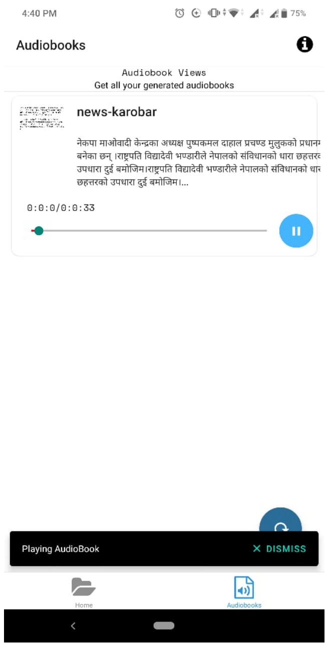

4:37 PM

news-karobar.pdf

नेकपा माओवादी केन्द्रका अध्यक्ष पुष्पकमल दाहाल 'प्रचण्ड' मुलुकको प्रधानमन्त्री बनेका छन् । राष्ट्रपति विद्यादेवी भण्डारीले नेपालको संविधानको धारा छहत्तरको उपधारा दुई बमोजिम; आइतबार साँझ दाहाललाई प्रधानमन्त्री पदमा नियुक्त गरेकी हुन् ।

'सम्माननीय राष्ट्रपति श्रीमती विद्यादेवी भण्डारीले नेपालको संविधानको धारा छहत्तर को उपधारा दुई बमोजिम; पुष्पकमल दाहाल प्रचण्डलाई नेपालको प्रधानमन्त्री पदमा नियुक्त गर्नु भएको छ; राष्ट्रपतिको कार्यालयले आइतबार जारी गरेको विज्ञप्तिमा भनिएको छ ।

(b) PDF Viewer


4:40 PM

Audiobooks

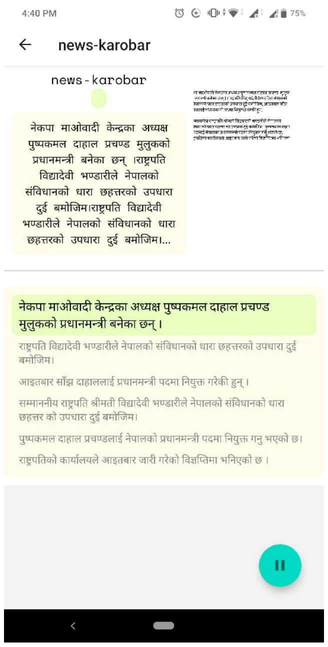
Audiobook Views
Get all your generated audiobooks

news-karobar

नेकपा माओवादी केन्द्रका अध्यक्ष पुष्पकमल दाहाल प्रचण्ड मुलुकको प्रधानम
बनेका छन् ।राष्ट्रपति विद्यादेवी भण्डारीले नेपालको संविधानको धारा छहत्तरव
उपधारा दुई बमोजिम।राष्ट्रपति विद्यादेवी भण्डारीले नेपालको संविधानको धार
छहत्तरको उपधारा दुई बमोजिम।...

0:0:0/0:0:33

(c) Audiobook Playing


4:40 PM

news-karobar

news-karobar

नेकपा माओवादी केन्द्रका अध्यक्ष पुष्पकमल दाहाल प्रचण्ड मुलुकको प्रधानमन्त्री बनेका छन् ।राष्ट्रपति विद्यादेवी भण्डारीले नेपालको संविधानको धारा छहत्तरको उपधारा दुई बमोजिम।राष्ट्रपति विद्यादेवी भण्डारीले नेपालको संविधानको धारा छहत्तरको उपधारा दुई बमोजिम।...

नेकपा माओवादी केन्द्रका अध्यक्ष पुष्पकमल दाहाल प्रचण्ड मुलुकको प्रधानमन्त्री बनेका छन् ।

राष्ट्रपति विद्यादेवी भण्डारीले नेपालको संविधानको धारा छहत्तरको उपधारा दुई बमोजिम।

आइतबार साँझ दाहाललाई प्रधानमन्त्री पदमा नियुक्त गरेकी हुन् ।

सम्माननीय राष्ट्रपति श्रीमती विद्यादेवी भण्डारीले नेपालको संविधानको धारा छहत्तर को उपधारा दुई बमोजिम।

पुष्पकमल दाहाल प्रचण्डलाई नेपालको प्रधानमन्त्री पदमा नियुक्त गनु भएको छ।

राष्ट्रपतिको कार्यालयले आइतबार जारी गरेको विज्ञप्तिमा भनिएको छ ।

(d) Sentence Syncing

Figure 5.6: Mobile Application Screens - 2

5.3 Discussion

Nepali TTS has been a challenging task due to the complexity of the Nepali language, such as the presence of consonant clusters, tone marks, and *halanta* (half-word). Many attempts have been made to develop a Nepali TTS system, but most lack quality and fluency. However, with recent advancements in deep learning techniques, we have obtained good results compared to other approaches.

Our TTS system uses the Tacotron 2 model, which has been widely used for TTS tasks in various languages. We found that Tacotron 2 produces a more natural and human-like speech compared to other TTS models we tested. The fine-tuning process was done by recording additional data in Nepali and using it to train the model. This helped to capture the unique features of the Nepali language, such as its tonal nature and the presence of half-words (*halanta*). The recorded data included a variety of sentences, including some with different tones and emphasis, to ensure that the model could generate speech with natural-sounding prosody. Despite the successful fine-tuning process, there were still some challenges. One of the major issues was handling punctuation marks, which is critical to the correct pronunciation and emphasis of Nepali words. The model sometimes failed to correctly interpret and render these marks, leading to incorrect articulation or unnatural pauses in the speech output.

Similarly, on the application side, the application was developed using React Native for the front end and Flask for the back end. The front end allows users to input Nepali text and choose the desired voice and speed. The backend then processes the text using the Tacotron 2 model and returns an audio file on the app. A major issue we encountered during the development process was related to multithreading. Since the Tacotron 2 model required significant computation time, we had to implement a multithreading mechanism to ensure that the application remained responsive while generating the audio file. This required careful management of resources and coordination between threads.

Overall, we are confident that the application has successfully achieved its intended goal of providing a user-friendly and accessible Nepali text-to-speech system. Although there may be some remaining issues that need to be addressed, we believe that with further development, our system has the potential to become a valuable and indispensable tool for the Nepali-speaking community.

6. Conclusion

Audiobook technology has been proven to be a useful instructional tool for struggling readers in classrooms. To further support the use of audiobooks in the classroom, we developed Shruti, an AI-generated Nepali book reader application that utilizes a text-to-speech system to generate a voice for books. The objective of the project was to provide a tool that makes reading more accessible and enjoyable for struggling readers in Nepal.

Shruti operates through the extraction of text from PDF files using Optical Character Recognition (OCR), which is then sent to the text-to-speech pipeline. The speech synthesis occurs in two phases: spectrogram generation and vocoder output. The extracted text is preprocessed, tokenized, and sent to a modified Tacotron2 model for generating Mel spectrograms. These spectrograms are then sent to the HifiGAN vocoder, which produces a synthesized speech output. The system has been designed to produce a natural-sounding voice that is easy to understand and engage with.

To evaluate the quality of the synthesized speech output, we conducted a Mean Opinion Score (MOS) test with 28 volunteers. The test was based on naturalness, with the synthesized speech output achieving a MOS of 4.04 out of 5. This high rating indicates that the system produces speech that is natural and easy to understand, further supporting its potential as a viable instructional intervention for struggling readers.

The Shruti system has been successfully deployed and integrated with a mobile application, allowing users to access audiobooks with ease. The application provides a user-friendly interface that enables users to search and select books, adjust reading speed, and listen to the synthesized speech output.

7. Limitations and Future Enhancement

7.1 Limitations

Although a working application that can read a PDF and generate its speech has been developed, many limitations were faced during the development and testing phase. Some of the drawbacks and limitations of this project are:

- **Insufficient Data :** While the Tacotron 2 model was used for generating Nepali TTS, we faced the limitation of insufficient data for training the model properly. As a result, the quality of generated audio is not as high as it could be with more data.

- **Limited availability of Nepali language resources:** Since Nepali is not a widely spoken language, there is limited availability of language resources such as text corpora, linguistic databases, and voice data. This can limit the accuracy and quality of the text extraction and TTS models, especially for less common words and phrases.

- **Lack of Nepali-focused text extractor:** There is a lack of Nepali-specific text extractors that can extract text from Nepali PDF files accurately. Therefore, the current system relies on an English text extractor, which can result in incorrect text extraction or missing text from Nepali PDFs.

- **Only supports PDF documents:** The application only supports PDF files, which means that users may face difficulties converting to other document formats such as Microsoft Word or Google Docs. This limits the usefulness of the application for users who work with different document formats.

- **Time required to generate audiobook:** The application takes some time to generate the audiobook, especially for longer documents. This can be inconvenient for users who need to generate speech for large documents quickly.

- **Difficulty in detecting and handling errors:** Due to the complexity of the TTS and text extraction models, it can be difficult to detect and handle errors during the audiobook generation process. This can lead to inaccuracies in the generated audiobooks and make it challenging to troubleshoot and fix errors.

- **Dependence on internet connection:** The application requires an internet connection to function properly since it relies on cloud-based services for audio generation and storage. This means that users may face difficulties in areas with poor or no internet connectivity, which limits the accessibility of the application.

7.2 Future Enhancements

Many aspects of this project can be further enhanced to improve efficiency as well as user experience. Some of the possible enhancements are:

- **Parallel Processing:** Multithreading can be used for text extraction and speech generation to speed up the audiobook generation process and reduce the waiting time for users.

- **Improved TTS Model:** The current TTS model was trained with limited data and fine-tuned using self-recorded data. A larger dataset with high-quality audio samples can be used to train the model for generating more natural-sounding speech.

- **Additional Voices:** Integrating both male and female voices into the application can provide more options for the users to choose from, making it more versatile and accessible.

- **Modulations:** Introducing voice modulations to the application, such as varying pitch and speed, can further enhance the user experience and provide more customization options.

- **Support for other Document Formats:** The application currently only supports PDF documents, but supporting other document formats like Word and HTML can expand its user base and usefulness.

- **Add a Separate Database:** The application currently stores all the data on the server, rather than that e can shift to a database like MongoDB.

- **Improve OCR:** Development of an algorithm to automatically detect and exclude non-textual content such as images and tables from the PDF file.

References

[1] Chris Wagar. *The Impact of Audiobooks on Reading Comprehension and Enjoyment.* PhD thesis, 06 2016.

[2] Basanta Dhakal. Statistical trends in literacy rates in nepal. *IOSR Journal of Applied Chemistry*, 11:71–77, 11 2018.

[3] Richard C. Anderson. *Becoming a nation of readers: The report of the commission on reading.* The National Institute of Education, 1985.

[4] Lorna M. Gilbert, Randy Lee Williams, and T. F. McLaughlin. Use of assisted reading to increase correct reading rates and decrease error rates of students with learning disabilities. *Journal of Applied Behavior Analysis*, 29(2):255–257, 1996.

[5] N. Umeda and R. Teranishi. The parsing program for automatic text-to-speech synthesis developed at the electrotechnical laboratory in 1968. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 23(2):183–188, 1975.

[6] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio, 2016.

[7] Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc Le, Yannis Agiomyrgiannakis, Rob Clark, and Rif A. Saurous. Tacotron: Towards end-to-end speech synthesis, 2017.

[8] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, RJ Skerry-Ryan, Rif A. Saurous, Yannis Agiomyrgiannakis, and Yonghui Wu. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions, 2017.

[9] Sameer R Maskey, Alan W Black, and Laura M Tomokiyo. Bootstrapping phonetic lexicons for new languages. 2004.

[10] Language technology kendra projects. [Date Accessed: July, 2022].

[11] Roop Bajracharya, Santosh Regmi, Bal Krishna Bal, and Balaram Prasain. Building a natural sounding text-to-speech system for the nepali language: research and development challenges and solutions. *Gipan*, 4:106–116, 12 2019.

[12] Rupak Raj Ghimire and Bal Krishna Bal. Enhancing the quality of nepali text-to-speech systems. In Alla Kravets, Maxim Shcherbakov, Marina Kultsova, and Peter Groumpos, editors, *Creativity in Intelligent Technologies and Data Science*, pages 187–197, Cham, 2017. Springer International Publishing.

[13] Dan Jurafsky and James H. Martin. *TTS Evaluation*. Pearson, 2022.

[14] Sagun Dhakhwa, Patrick AV Hall, Ganesh Bahadur Ghimire, Prakash Manandhar, and Ishwor Thapa. Sambad-computer interfaces for non-literates. *Lecture Notes in Computer Science*, 4550:721, 2007.

[15] Kaushal Subedi. Nepalispeech - convert written nepali text to speech.

[16] Bhusan Chettri and Krishna Shah. Nepali text to speech synthesis system using esnola method of concatenation. *International Journal of Computer Applications*, 62:24–28, 01 2013.

[17] Pratistha Malla. Nepali text to speech using time domain pitch synchronous overlap add method. Master's thesis, 2015.

[18] Krishna Bikram Shah, Kiran Kumar Chaudhary, and Ashmita Ghimire. Nepali text to speech synthesis system using freetts. *SCITECH Nepal*, 13(1):24–31, 2018.

[19] Dennis H Klatt. Software for a cascade/parallel formant synthesizer. *the Journal of the Acoustical Society of America*, 67(3):971–995, 1980.

[20] Paul Taylor, Alan W Black, and Richard Caley. The architecture of the festival speech synthesis system. In *The third ESCA/COCOSDA workshop (ETRW) on speech synthesis*, 1998.

[21] Tej Bahadur Shahi and Chiranjibi Sitaula. Natural language processing for nepali text: a review. *Artificial Intelligence Review*, pages 1–29, 2022.

[22] Isin Demirsahin, Martin Jansche, and Alexander Gutkin. A unified phonological representation of south asian languages for multilingual text-to-speech. 2018.

[23] Anand Arokia Raj, Tanuja Sarkar, Sathish Chandra Pammi, Santhosh Yuvaraj, Mohit Bansal, Kishore Prahallad, and Alan W Black. Text processing for text-to-speech systems in indian languages. In *Ssw*, pages 188–193, 2007.

[24] Papers with code - north american english benchmark (speech synthesis). [Date Accessed: July, 2022].

[25] Ashok Basnet. Attention and wavenet vocoder based nepali text-to-speech synthesis. Master's thesis, 2021.

[26] Ashok Banset, Basanta Joshi, and Suman Sharma. Deep learning based voice conversion network. page 1292 – 1298, 10 2021.

[27] Dan Jurafsky and James H. Martin. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition.* Pearson, 2022.

[28] Dan Jurafsky and James H. Martin. *TTS: Spectrogram Prediction.* Pearson, 2022.

[29] Keshan Sodimana, Knot Pipatsrisawat, Linne Ha, Martin Jansche, Oddur Kjartansson, Pasindu De Silva, and Supheakmungkol 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)*, pages 66–70, Gurugram, India, August 2018.

[30] Shwet bhairabi: Free download, borrow, and streaming.

[31] Keith Ito and Linda Johnson. The lj speech dataset. `https://keithito.com/LJ-Speech-Dataset/`, 2017.

[32] Oddur Kjartansson, Supheakmungkol Sarin, Knot Pipatsrisawat, Martin Jansche, and Linne 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)*, pages 52–55, Gurugram, India, August 2018.