Stress Transfer in Speech to Speech Machine Translation System

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in
Computer Science and Engineering
by Research

by

Sai Akarsh C
2019111017
`sai.akarsh@research.iiit.ac.in`

International Institute of Information Technology
Hyderabad - 500 032, INDIA
June, 2024

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Stress Transfer in Speech to Speech Machine Translation System" by Sai Akarsh C, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date	Adviser: Prof. Anil Kumar Vuppala

To my family

Acknowledgements

Like life, research is not just about reaching the destination but is also about the journey. And a journey is never straight and has its ups and downs. We need good people around us, not just to cherish the moments of success but also for support and encouragement during moments of failure. This journey would not have been possible without the help of many amazing people around me and I am incredibly grateful to each one of them.

First and foremost, I want to express my sincere gratitude to my adviser, Prof. Anil Kumar Vuppala, for motivating me towards research and for the opportunity to do an MS under his guidance. I am thankful to him for his knowledge in the field and the support he has given. This work would not have been possible without his motivation and his mindset to strive for excellence. His approach towards research which is systematic and thorough, helped me find my footing in this field and provided an excellent base for my research work. His strict values helped keep my work on track and gave me the confidence to do any work, however hard it might seem. I express my gratitude and respect to Prof. Yegnanarayana Bayya, a great teacher who taught us the concepts of speech signal processing and amazed us with the marvels of signals.

I want to thank all my lab members, who share this journey of research with me, in making research enjoyable. First, I would like to thank Raghu Vamshi sir, a person who is a pillar of support not just for me, but also for the entire lab. He is someone you can always rely on and is ready to help others, be it doubts, or discussions regarding research, friends, family or even future plans. Sorry for all the tough times we gave you. Thank you Priyanka ma'am, who is always there when you need to talk to someone and supports you no matter what. She is always lively and fun and makes the lab more than just research. Thank you, Snehal, for all the Formula 1 you got into me and for the late-night karting and ranting. Thank you, Nayan, for the insights you give irrespective of the work, and your ready-to-explain nature. Thank you, Anindita and Anuprabha ma'am for their support and presence in the lab, without whom it always feels empty. Thank you, Pramod, for keeping the lab chill and giving gym advice whenever needed. I would also like to thank everyone else whom I met here - Kowshik, Shivaang, Hasvitha, Subhankar, Srihari, Anirudh, Jhansi, Meenakshi, Kumar, Haala, Aditya, Kesav and others. I would also like to thank our Interns Sreemaee, Saketh, Subhangi, Rakesh and Dharma for all the memories and fun we had, despite it being for a short time. Thank you all for the memorable times and help in my journey.

As they say, your college life is defined by the friends you make there. I have met many smart and caring people on this campus, each with their own unique traits and persona. I want to thank my gang -

Nikit, Gokul, Kawshik, Gayatri, Tanvi and Ahana, for bearing with me for an incredible five years. That's a long time, and we seven have done so much together that it is heartbreaking to see this part of our lives coming to a conclusion. Each of us is so different yet we managed to stick around to have the best time of our lives. Thank you for all the fun times we had, for sharing the pain of eating mess food, ranting about research and people, and for always being there for each other. I would also like to thank my pool gang - Akhilesh, Anir and Sankalp, with whom I played nearly every day. Having to wait for the evening to come quickly, so that we can open the warehouse and play pool is a memory I will always remember and look back on.

Finally, I would like to thank my family. Thank you Mom and Dad for all the love and support you gave. Thank you, my twin sis, for literally being my other half and for keeping up with my annoying nature and still be amazing.

I would like to thank all the others whom I could not mention here, but you were just as important to my life and journey.

IIIT-H has taught me that with the right guidance, anyone is capable of research if you keep your mind to it. So keep this in mind, future Akarsh, that everything is within your reach, you just need to strive hard for it.

Abstract

The proliferation of online educational resources has revolutionized learning access globally. Many companies have started investing in online education as the idea of a better future without the need to go to top colleges but still receive a good education is very appealing to the Indian population. However, English is the ”Lingua Franca” of India and most of the online and offline information use it. But India being a land of diverse languages cannot cater to the entire population only in English. To allow and enable students from different regions of India to use online education to their advantage, a practical solution is required to tackle this. Speech-to-Speech Machine Translation (SSMT) holds promise in bridging this gap by translating educational content automatically. This is possible due to the recent advancement in machine learning, speech processing and most importantly the availability of powerful hardware. In SSMT, only the contextual information is retained when converting speech into text and information about the speaker and any prosodic elements are lost. Therefore the generated output speech is monotonous and machine-like that lacks emotion and depth. This speech devoid of prosody makes the converted education content not engaging and potentially hinders learning. There is a need for a system that can use available information in the source speech such as prosody and emotion and generate speech in the target language along with these prosodic cues and emotional state. Despite some work in literature, the unique challenges of incorporating prosody, especially stress or emphasis in Indian language-based SSMT remain largely unexplored.

This thesis addresses the limitations and needs for an SSMT system capable of transferring prosodic stress cues from Indian English to Hindi, especially in lecture-mode educational content. The importance of stress in educational content is that it would draw the attention of the student to properly focus on a concept when the instructor puts stress on it. For preserving stress in the generated speech, a model that can detect stress in the source language needs to be built. This work curates an open-source stress region annotated dataset in Indian English, as an accurate stress annotated dataset is very hard to come by, especially in the field of research. This dataset consists of lecture mode educational content by various speakers collected from an open-source online platform. Several acoustic, spectral and temporal features correlated to stress were extracted from the dataset to train different machine-learning models for predicting stressed regions in Indian English speech. This work also proposes Stress-Net a deep neural network (DNN) that gives better performance and generalizability. As most Text-to-speech (TTS) systems do not deal with stress as a conditioning input, this work proposes modifications to existing TTS architectures like FastPitch and YourTTS to add stress using the source speech as a reference. Different components of the SSMT pipeline work together and generate stress cues that condition the TTS when generating speech. Stress cues are information about the location of the words that are stressed and a quantitative measure of the amount of stress relative to the rest of the words in the sentence. Stress introduction using this information is achieved through variance modifiers, allowing the TTS to be trained on a generic neutral speech corpus without a TTS-specific stressed speech dataset. This work produces a comprehensive Indian English-Hindi SSMT system equipped to translate educational content from Indian English to Hindi while preserving stress to enhance user engagement. This work includes the curation of a stress-annotated Indian English dataset, the creation of stress detection models, the integration of these stress detection models into the SSMT pipeline and the modification modules required by TTS for an acceptable level of stress addition. This paves the way for more effective online learning experiences in India's diverse linguistic landscape.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Prosody and Stress in Speech

The rise of online learning platforms has opened up educational opportunities worldwide. This trend is particularly relevant in India, where access to quality education can be challenging. However, a major barrier exists – the dominance of English in online educational materials. India, a nation rich in diverse languages, requires solutions that cater to its multilingual population. Speech-to-Speech Machine Translation (SSMT) offers a promising approach by automatically translating spoken educational content. SSMT consists of three stages: converting source speech to text, translating the text into the target language, and converting the translated text into speech. When the source speech is converted into text, only the contextual information is retained. Information about the speaker, the emotion present, and any prosodic elements involved are all lost. Therefore, when the final speech is generated from the remaining information, which is text, it is devoid of prosody and does not remotely sound like the original speech. This monotonous machine-like speech that lacks emotion and depth makes the converted education content not engaging. This leads to user disengagement and potentially hinders learning outcomes, making such a system less effective for the task. There is a need for a system that can use available information in the source speech, such as prosody and emotion, and generate speech in the target language along with these prosodic cues and emotional states. Such systems would have a wide range of applications, from movie dubbing to real-time speech translation for communication and, in this case, spreading knowledge to a much wider audience and engaging them without compromising on the quality and depth of the content. Most works in literature deal with improving the naturalistic quality of SSMT and not preserving prosody. A few studies have explored prosody integration in SSMT, but those works focus on European language pairs [1, 2] and do not consider Indian languages. Therefore, the unique challenges of incorporating prosody, especially stress or emphasis in Indian language-based SSMT, remain largely unexplored.

Prosody plays a critical role in human communication, influencing how listeners perceive the intended meaning and emotional state of the speaker. By understanding the intonation, stress, and rhythm of speech, we can improve the quality of speech applications in several ways. Prosody can help identify natural breaks in speech, allowing for segmentation into sentences or word groups. This segmentation aids in aligning recognized words with the correct parts of the audio. Speech communication is a complex process that conveys meaning not only through the content of the spoken words but also through the way those words are delivered. Prosody can help distinguish between words that sound similar but have different meanings. For instance, variations in pitch and loudness can distinguish between a statement, a question, or a sarcastic remark. Understanding stressed words can provide clues about the speaker's intent and improve recognition of keywords. The emphasis placed on certain words or syllables can highlight key information and influence the overall interpretation of an utterance. Prosody refers to the elements of speech that influence how we say things rather than what we say. It's like the music of language. However, replicating this ability in machines, particularly in the context of speech-to-speech machine translation (SSMT), presents a significant challenge.

One of the most crucial elements of prosody is stress. Stressed syllables are pronounced with greater emphasis, often through variations in pitch, duration, or loudness. These variations act as a spotlight, drawing the listener's attention to specific words or phrases and influencing the overall interpretation of the utterance. Especially when information is being conveyed, the positioning of the stressed region often completely changes the meaning of the sentence. Stress detection in speech, therefore, becomes a vital step in capturing the speaker's intended meaning and emotional state. Our work focuses on this element of prosody which is stress for improving the quality or naturalness of the generated speech. This work first explores stress detection in speech by analyzing various speech features, such as pitch contours, and energy levels and employs machine learning techniques to identify stressed regions from them. More complex features such as Mel Frequency Cepstral Coefficient (MFCC), Shifted Delta Coefficient (SDC) and Mel Spectrogram are also explored for identification. These features contain rich temporal and contextual information that proves to be useful for stress identification. In speech processing, feature extraction and engineering are the most important steps. Therefore initially, the models used for stress region classification in Indian English as explained in Section 3.4.1 were kept simple to understand the impact of the features. In the follow-up work, a DNN called Stress-Net as explained in Section 3.4.2 comprising of Time Delay Neural Network (TDNN) and Transformers was proposed which uses similar features to identify stress regions with better performance and generalizability in Indian English. All these models require a well-curated highly accurate stress annotated dataset, and these are very hard to come by, especially in the field of research. Therefore, this work as explained in Section 3.2 also curates an open-source stress region annotated dataset in Indian English hand-marked by speech annotators that consist of educational content spoken by various speakers collected from an open-source online platform. This allows the training of these models on Indian English in the domain of online education and also opens doors for applications that require a deeper understanding of spoken language, including sentiment analysis, speaker identification, and speech-to-speech machine translation.

1.2 Stress in SSMT

In an SSMT system, the speech generated should possess some level of prosody. This thesis tries to address transferring one prosodic feature in SSMT: stress. This involves giving the Text-to-Speech (TTS) system, which generates synthetic speech from textual input, the ability to incorporate stress when generating speech. The recent surge in research within the fields of Machine Learning (ML) and Artificial Intelligence (AI), fueled by the remarkable advancements in Deep Neural Networks (DNNs) and the exponential growth of Graphics Processing Unit (GPU) hardware capabilities, has revolutionized the field of speech synthesis. The past few years have witnessed a wave of groundbreaking models that have redefined the landscape of speech synthesis. Pioneering models like Tacotron [3], Tacotron 2 [4], FastPitch [5], FastSpeech [6], FastSpeech 2 [7] and Deep Voice [8] have pushed the boundaries of what was previously thought achievable. These models leverage the power of DNNs to learn complex relationships between text and speech, enabling them to synthesize speech with unprecedented naturalness, fluency, and control over prosodic features. With the introduction of Non-Autoregressive models like [6, 7, 5], the inference time has greatly improved and can achieve real-time inference with minimal hardware requirements. The ability of these DNN models to deal directly with characters [4, 9] allows them to bypass the need for a well-curated lexicon. This breakthrough allowed for the creation of speech synthesis models for nearly a thousand languages [10, 9], as the labour and the skill-intensive job of lexicon creation isn't required anymore. Most speech synthesis systems are capable of adding some amount of prosody to make it sound natural. Still, such additions only rely on text and cannot be conditioned on scenarios. There has also been a lot of work in adding prosody to the synthesized speech via conditioning which includes transferring prosody through style tokens [11], through self-supervised learning [12] and even conditioning via diffusion [13, 14]. These works use reference audio to detect prosody and use that as a conditioning feature when generating speech. But without any previous works on stress, combined with the fact that other works that generate speech from a reference lack control, there is a need for a TTS that can generate speech with stress in it.

This work proposes a novel way of incorporating stress in the speech generated by TTS systems. By adopting stress cues, existing TTS models with few modifications can condition the generated speech to have emphasis with fine-grained control. Stress cues, composed of scaling factors as explained in 3.5.2, help in capturing the essence of stress in the source speech and give a way for the TTS to condition the speech on them. This work shows that a stress-aware speech synthesis model can be trained without the need for a stress-specific TTS dataset. This thesis discusses how changes were made to existing architectures in Sections 4.3 and 4.4 to add stress to the target speech. When integrated into SSMT systems, these advancements hold immense potential for the future of human-computer interaction and cross-lingual communication. In this work, a stress-annotated lecture-mode speech Indian English dataset was curated, stress detection models were proposed to detect stressed regions in speech, and TTS models were improved to handle stress incorporation using stress cues from the source speech. All of these components together enable the SSMT systems to generate translated speech that conveys the semantic meaning and captures the stress nuances present in the original utterance. This paves the way for more natural, engaging, and effective communication across language barriers.

1.3 Objective and Scope of the Thesis

This thesis aims to improve the quality and naturalness of generated translated speech in Indian English to Hindi SSMT system using prosodic stress cues. The source speech has prosodic and speaker information along with contextual information. Only contextual information is retained by Automatic Speech Recognition (ASR) systems, and therefore, important information that can help in SSMT is lost. This work tries to preserve one such information: stress. The primary objective is to develop a framework that translates the spoken content and incorporates stress information from the source language speech into the target-generated speech. By doing this, we aim to create a more natural and emotionally rich translation experience, enabling nuanced communication across languages and improving the depth and reach of online education for students in India. This thesis explores stress detection in speech, a stress dataset for that task and investigates integrating this information using TTS. The performance of the proposed framework will be evaluated through subjective human evaluation and objective speech quality metrics. The scope of the thesis is as follows:

- At the dataset level, a stress-annotated speech corpus in the Indian English education domain is curated for research purposes.
- At the feature level, acoustic features extracted from the dataset are investigated for stress detection in Indian English lecture-mode speech.
- At the model level, models for stress detection and speech synthesis models with stress-aware conditioning are proposed for enabling stress transfer in SSMT.

1.4 Thesis Outline

Overall, this thesis is divided into five chapters, for each of which a quick description is given below:

- **Chapter 1** highlights the need of a Speech-to-Speech Machine Translation that is capable of preserving stress. It also presents the challenges in building such a system and the method employed by this thesis to overcome them.
- **Chapter 2** presents the related work done in the past for Stress Detection, Speech Synthesis and SSMT with prosody transfer. It also presents some related work and datasets this thesis uses to build the SSMT pipeline.
- **Chapter 3** presents a stress-annotated speech corpus in Indian English and discusses the feature extraction and classifiers used for Stress Detection.

- **Chapter 4** presents the Speech Synthesis system responsible for stress conditioning and its feature extraction. It also presents the overall SSMT pipeline and its different components.
- **Chapter 5** presents conclusions and future scope of this work.

Chapter 2

Related Work

2.1 Introduction

Transferring stress from the source speech to the target speech in SSMT requires multiple components working together seamlessly. SSMT typically consists of Automatic Speech Recognition (ASR) to convert speech to text, Machine Translation (MT) to translate the text into the desired language, and finally Text-to-Speech (text) to synthesize speech from the translated text. This thesis primarily works on stress detection and improvements in TTS. Therefore we need to address the other components that are essential for this task. This chapter discusses previous works in stress detection and the motivation for building our own, different architectures that exist in TTS, standard datasets used for training and evaluating them, the metrics and methods of evaluation, previous works in transferring prosody in SSMT and finally discusses other works used in this thesis for ASR, MT and its word aligner.

2.2 Works on Stress Detection

Stress detection or emphasis detection is a decently explored topic in literature, but almost all of them deal with English or other European languages. Different methods were employed for stress detection in speech. Sonority, which represents the carrying power of sounds in words or longer utterances plays a key role in the method proposed by [15] for automatic syllable-level stress detection. Their method computes feature contour by combining sonority-motivated cues with sub-band short-time energy contours, unlike a traditional short-time energy contour. Experiments were conducted with ISLE corpus [16] for 23 German and 23 Italian intermediate learners of English. Even [17] worked on the same corpus, where they have considered a representation learning approach jointly with VAE and DNN for automatic syllable-level stress detection. They used acoustic and context-based features derived from the method proposed by [18]. An unsupervised approach for automatic word prominence detection is presented in [19] using spontaneous speech data from switchboard corpus [20] that can robustly distinguish between content word and function word classes. The algorithm scores prominence by combining various acoustic feature sets, incorporating both spectral and temporal features extracted from the correlation envelope of the word signal along with part of speech (POS) as a linguistic correlate for word prominence. The spectral and temporal features offer direct insights into the location of syllable nuclei, spectral intensity, and the dynamic patterns of pitch throughout an utterance. The study in [21] examines different pitch-based measures to determine their effectiveness in detecting prominence. It introduces a parametric scheme for modelling pitch plateaus, which surpassed traditional local pitch statistics in prominence detection performance. The hypothesis driving the usage of POS is that words perceived as more prominent in spoken language often belong to specific POS categories, such as content words. The results in [21] show that content words tend to have higher prominence scores compared to function words. A similar approach was followed in [22] where the analysis integrated both prosodic information, encompassing factors like the area under the F0 curve, voiced-to-unvoiced ratio, F0 peak/valley amplitude, and their locations, as well as lexico-syntactic information. In [23], emotionally stressed regions are identified by analyzing the strength of excitation (SoE) and fundamental frequency (F0) of speech signals to develop an emotion recognition system, utilizing the Berlin emotion speech (EMO-DB) database. This study employs an 11-dimensional feature vector to extract differenced prosody features, comprising mean, maximum, and minimum variations of pitch, pitch dynamics, and energy, alongside the duration ratio and pitch standard deviation.

Most of the works mentioned use non-Indian datasets for the task. In the case of lecture-mode education content, the meaning of stress or emphasis depends on the content being told and how it affects the meaning. Therefore there is a lack of work in literature that deals with stress detection in Indian English especially in lecture-mode educational content. The works in literature deal with syllable level detection, or other modes of stress like emotional stress. These models are not generalizable or are restricted to syllable level. By making the detection on the frame level, we can choose to convert this to either word or syllable level without having a prior restriction.

2.3 Speech Synthesis

Speech synthesis (Text-to-Speech, TTS) has witnessed significant advancements in recent years. Prior methods relied on concatenating pre-recorded speech segments, limiting flexibility and naturalness. This landscape has shifted dramatically due to progress in deep learning and GPU hardware capabilities. Sophisticated Auto-Regressive (AR) models [3, 4], Non-Auto-Regressive (NAR) models [6, 7, 5] and End-to-End (E2E) NAR models [9, 24] have emerged, leveraging deep neural networks to capture intricate relationships between text and speech. This enables the generation of highly natural and expressive speech with greater control over prosodic features. Moreover, the introduction of NAR models led to significant efficiency through parallelisation, allowing for real-time high-quality speech synthesis. Additionally, most models can work directly with characters, eliminating the need for extensive lexicon creation. This opens doors for speech synthesis applications in a wider range of languages. In essence, deep learning has transformed speech synthesis, making it more natural, efficient, and applicable to a broader range of languages.

2.3.1 Datasets

Datasets specific to speech generation are of very high audio quality and require special equipment to be recorded. Speakers are expected to record their speech in a balanced tone, without any disfluencies or stutters, and preferably in a recording studio. These tight requirements enable the data to be of very high quality without any text-audio mismatch, which is essential when training TTS systems. However, the downside of such high requirements is that it's quite hard and expensive to create them in the desired amount. Therefore, there are not a lot of open-source datasets that researchers can use for benchmarking and training their models. A few datasets [25, 26, 27, 28] are often used for benchmarking the quality of TTS in research papers. In this section, a few English datasets that are used for benchmarking and the Indian datasets that are used for training are discussed briefly.

2.3.1.1 LJ Speech

The LJ Speech Dataset [25] is a publicly available resource for training and evaluating Text-to-Speech (TTS) systems. This dataset consists of 13,100 short audio clips, featuring a single American female speaker reading excerpts from non-fiction books published between 1884 and 1964. The accompanying transcripts ensure proper alignment between the spoken audio and the corresponding textual content. The audio clips range from 1 to 10 seconds, with a total recording time of approximately 24 hours. The recordings were done in a recording studio and under strict quality control, resulting in a gold-standard benchmarking dataset used in nearly every TTS paper. This allows for a relatively easier comparison between different architectures as they are using the same dataset for evaluation.

The significant advantage of the LJ Speech Dataset lies in its public domain status. Both the source texts and the audio recordings are freely available for use without copyright restrictions. This eliminates potential licensing hurdles and allows researchers to openly share and adapt the dataset for various TTS research endeavours. It is part of the LibriVox project, recorded by a single speaker and has read speeches from non-fiction books. The availability of a large corpus of speech recordings paired with corresponding transcripts makes the LJ Speech Dataset a valuable resource for researchers developing and evaluating TTS models. The single-speaker nature of the dataset allows for a focused analysis of the model's ability to capture and reproduce prosodic variations within a consistent voice.

2.3.1.2 VCTK

The VCTK Corpus [26] is a comprehensive speech dataset designed for training and evaluating Text-to-Speech (TTS) systems. This publicly available resource features speech recordings from 109 native English speakers with diverse accents. Each speaker reads approximately 400 sentences, including excerpts from a newspaper, the Rainbow Passage, and an elicitation paragraph specifically designed to capture the speaker's unique accent. The newspaper text selection process employed a greedy algorithm to ensure a varied distribution of sentences across speakers, while the Rainbow Passage and elicitation paragraph remained constant for all participants.

The VCTK Corpus offers several advantages for TTS research. First, it provides a rich collection of speech data encompassing a broad range of English accents. This allows researchers to develop and evaluate TTS models capable of generating natural-sounding speech across diverse speaker demographics. Additionally, the standardized recording setup ensures consistency in audio quality, facilitating focused analysis of model performance. Finally, the freely available nature of the VCTK Corpus removes licensing barriers and promotes collaboration within the TTS research community. It's important to note that while the majority of recordings utilized high-quality equipment (DPA 4035 microphone and Sennheiser MKH 800 microphone) with a 96kHz sampling rate, technical issues affected the recordings of two speakers (p280 and p315) [26]. These recordings were downsampled to 48 kHz and underwent manual endpointing to ensure consistency with the remainder of the dataset.

2.3.1.3 IndicTTS

The IndicTTS Corpus [29] is a valuable resource designed specifically for training and evaluating Text-to-Speech (TTS) systems for Indian languages. This dataset addresses the need for high-quality speech corpora in these languages, where research efforts have historically lagged behind other regions. Developed through a collaborative effort, the IndicTTS Corpus offers a collection of speech recordings and corresponding text transcriptions encompassing 13 major Indian languages: Assamese, Bengali, Bodo, Gujarati, Hindi, Kannada, Malayalam, Manipuri, Marathi, Odia, Rajasthani, Tamil, and Telugu.

The corpus construction process meticulously considered various factors crucial for high-quality TTS systems. The corpus incorporates carefully chosen text samples suitable for TTS training. A diverse group of speakers were selected to represent a range of pronunciations and speaking styles within each language (except Bodo, which features only a female speaker). The corpus captures natural pronunciation variations to enhance the overall realism of the synthesized speech. Standardized recording procedures were implemented to ensure consistent audio quality across the dataset. The text underwent cleaning and correction to handle out-of-vocabulary words and maintain data integrity. Encoding, sampling rate, and channel configuration were carefully chosen to ensure high-fidelity audio recordings.

These considerations resulted in a comprehensive dataset exceeding 272 hours of transcribed speech recordings. Each language, except Bodo, features recordings from both male and female speakers, with a minimum of 8 hours of transcribed data per speaker. Additionally, all audio samples were downsampled to a consistent rate of 22.05 kHz, and utterances exceeding 20 seconds were removed for consistency. Preprocessing steps were applied to the text data to further enhance its suitability for TTS training. The IndicTTS Corpus serves as a cornerstone for research and development efforts in Indian language TTS. Its comprehensive design, diverse speaker representation, and focus on high-quality recordings make it a valuable asset for researchers seeking to create natural-sounding and speaker-specific TTS systems for a wide range of Indian languages. For these reasons, this dataset was used for training the TTS in SSMT. The above two datasets mentioned were used only for benchmarking the TTS for quality comparison.

2.3.2 Deep Neural Networks

This thesis considers two DNN-TTS systems as baselines for adding stress. The first system is a relatively simple transformer-based encoder-decoder model while the other is a more complex normalizing flow-based variational inference model.

2.3.2.1 FastPitch

FastPitch [5] is a non-autoregressive parallel Text-to-Speech (TTS) system. It deviates from traditional autoregressive models that sequentially generate speech. Instead, FastPitch leverages a parallel architecture, enabling significantly faster inference speeds. This translates to much faster than real-time speech synthesis, a critical advantage for practical applications. A key strength of FastPitch is its ability to generate high-quality speech conditioned on predicted pitch contours. This essentially means the model can be influenced by variations in pitch, allowing for more expressive and natural-sounding output. Additionally, FastPitch offers the capability to manipulate these pitch contours, providing a valuable tool for generating speech that resembles controlled voice modulation. This forms the basis for stress or prosodic emphasis control in the generated speech and opens doors for applications such as emotional and emphasis TTS.

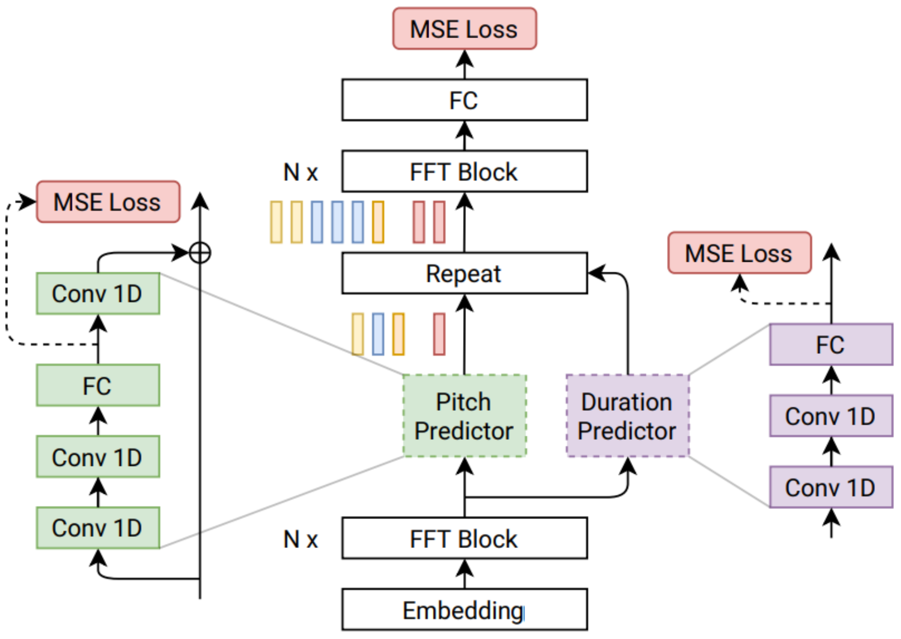


Figure 2.1: An overview of the components in FastPitch TTS Architecture. The transformer text embeddings are used to predict the pitch and duration variances using a convolution stack. Then the final embeddings are used to predict the mel spectrogram using another transformer. *Source: [5]*

The core of FastPitch as seen in Figure 2.1 revolves around a Transformer-based encoder-decoder structure. The Transformer[30], a well-established neural network architecture, excels at capturing long-range dependencies within sequential data like text. In FastPitch, the encoder processes the input text, extracting relevant information and generating intermediate representations. These representations are then utilized for two crucial tasks: predicting pitch contours and durations. Pitch contours define the variations in pitch throughout the speech on the text token level, while durations determine the number of frames per text token. By predicting these features on a text token level, FastPitch provides granular control over the prosodic aspects of the synthesized speech. A stack of 1D convolutions is used for predicting the pitch contour and durations, followed by a linear layer to get the respective embeddings. Following the prediction of pitch and duration, the combined text and pitch representations are upsampled to the frame level. This aligns the information with the frame rate of the target speech signal. The upsampling process utilizes the predicted durations, ensuring that each element in the sequence corresponds to the correct time frame in the synthesized speech.

Finally, the processed representations are fed into another Transformer block, which converts them into a mel spectrogram representation. A mel spectrogram is a compressed representation of the audio signal that captures the human auditory system's perception of pitch. The mel spectrogram then serves as the input for a WaveGlow [31] vocoder, a separate neural network model trained to reconstruct the actual audio waveform from the mel spectrogram representation. The training loss involves the reconstruction loss for the predicted mel spectrogram using Mean Square Error (MSE) loss, and also the MSE loss for the predicted pitch and duration values.

The design choices in FastPitch contribute significantly to its effectiveness. The use of a parallel architecture enables fast inference, while the Transformer's capability allows for efficient capture of textual information. Additionally, the separate prediction of pitch contour and durations provides fine-grained control over prosody. These factors combined make FastPitch an attractive choice for tasks like stress synthesis, where precise control over pitch is essential for effective communication.

2.3.2.2 YourTTS

YourTTS [24] is a state-of-the-art end-to-end TTS system known for its ability to generate high-quality speech. YourTTS builds upon the foundation established by VITS [9] and introduces several key modifications to achieve zero-shot multi-speaker TTS. A central modification involves the use of a larger text encoder. This enhanced encoder empowers YourTTS to extract more intricate information from the input text, leading to a more comprehensive understanding of the content. Additionally, YourTTS employs a separate speaker encoder specifically designed to process speaker information. This dedicated component allows YourTTS to achieve a higher degree of speaker fidelity, meaning the synthesized speech can be tailored to resemble the characteristics of a particular speaker, even without prior training on that specific speaker's voice. To model a latent representation from the speech spectrogram, YourTTS utilizes a combination of a posterior encoder and a normalizing flow [32, 33]. The posterior encoder analyzes the spectrogram and extracts a latent representation that captures the underlying structure of the speech as seen in Figure 2.2. The normalizing flow then refines this representation, ensuring it adheres to a specific probability distribution.

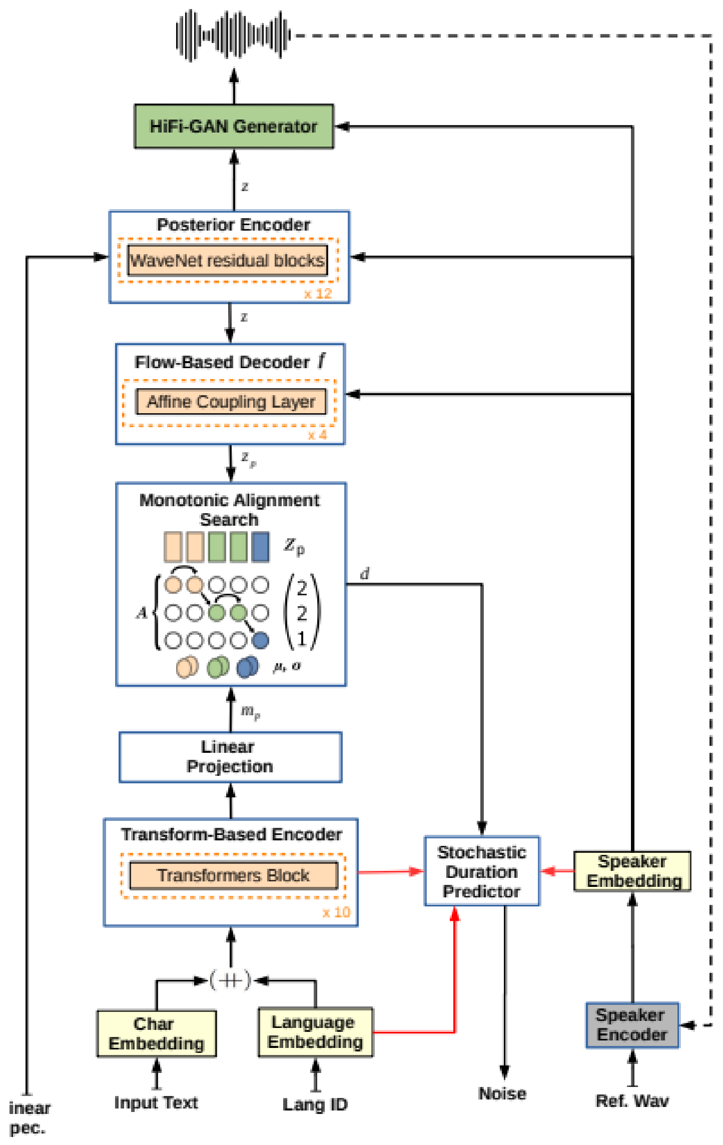


Figure 2.2: The architecture of YourTTS during training. Monotonic Alignment Search is used to learn the durations on the fly and is then used to train the Stochastic Duration Predictor. *Source: [24]*

A crucial aspect of YourTTS lies in its ability to perform Monotonic Alignment Search (MAS) [34, 35] between the latent representations and embeddings sampled from a normal distribution of the text embeddings. This alignment process essentially bridges the gap between the textual content and the corresponding speech characteristics. The learned alignment is then used to train a Stochastic Duration Predictor (SDP) [9] in parallel. This SDP leverages the text embeddings to directly predict the frame count of the synthesized speech, eliminating the need for separate duration modelling stages. SDP also makes the latent embeddings partly generative, allowing a bit of variability in the generated speech. However the MAS is only used during training, and in inference, we use the trained SDP to generate the durations as seen in Figure 2.3.

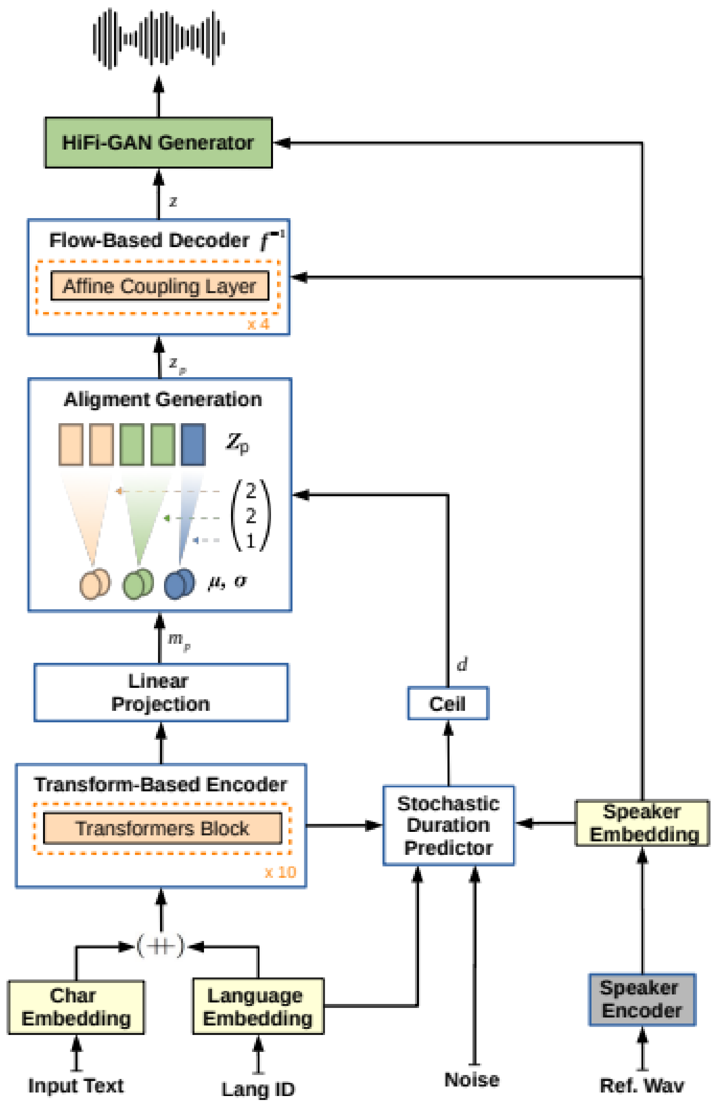


Figure 2.3: The architecture of YourTTS during inference. The Stochastic Duration Predictor is used for generating the duration for every text token. *Source: [24]*

Finally, YourTTS incorporates a HiFiGAN decoder [36] operating concurrently with the other components. The HiFiGAN decoder directly generates the audio waveform from the latent representation, bypassing the need for an external vocoder. This streamlined approach contributes to the overall efficiency of the YourTTS system. In essence, YourTTS leverages a larger text encoder, a dedicated speaker encoder, a posterior encoder with a normalizing flow, and a HiFiGAN decoder to achieve high-fidelity speech synthesis with zero-shot multi-speaker capability. This combination of architectural elements makes YourTTS a powerful tool for applications requiring natural-sounding and speaker-specific speech generation. Therefore this architecture was chosen for experimentation to generate stress in speech.

2.3.3 Metrics

When scoring a speech synthesis system, the naturalness and quality of speech need to be given utmost priority. Doing that algorithmically is not trivial, hence subjective metrics are often used when dealing with speech and its synthesis. Objective metrics such as Mel Cepstral Distortion (MCD) [37] are often not used due to such metrics' inability to capture the naturalness of the speech. Naturalness is a quality derived from human perception, hence a human evaluator is needed to evaluate the system for such qualities. However such evaluations have some limitations as well. An evaluator's subjective perceptions and preferences can influence their scores, and the chosen sample set may not represent the full range of speech variations the TTS system might encounter. Additionally, potential biases can arise from the order of presentation or comparisons with other systems presented at the same time. Despite these limitations, subjective metrics are a crucial tool for assessing the overall quality and user acceptance of TTS systems.

Mean Opinion Score (MOS) [38] is a subjective metric employed to assess the overall quality of Text-to-Speech (TTS) systems. It relies on human perception rather than objective measurements, providing insights into how natural and pleasant the synthesized speech sounds to listeners. The process typically involves selecting a representative sample of audio clips generated by the TTS system. These clips are then presented to a group of human evaluators, often through online crowdsourcing platforms. Each evaluator listens to the samples and assigns a score on a predefined scale, typically ranging from 1 (very bad) to 5 (very good), with finer gradations possible. Finally, the MOS is calculated by averaging the individual scores provided by all the evaluators for each audio clip. The benefit of MOS is that it can be modified based on the use case. For example [24] uses Sim-MOS, a grading based on similarity rather than naturalness. In our work, we too use different versions of MOS to perform evaluation based on the nature of the task.

webMUSHRA (Multiple Stimuli with Hidden Reference and Anchor) [39] is another subjective metric which is very similar to MOS and adheres to the same protocols for evaluation. It is a web-based methodology that builds upon the traditional MUSHRA standard for evaluating the perceived quality of audio samples. It is entirely conducted online, eliminating the need for specialized software or hardware thus streamlining the evaluation process. Usually, a high-quality reference audio sample representing the ideal speech content serves as the benchmark for comparison. The scoring range used is generally from 0 to 100, with 0 representing the lowest perceivable quality and 100 representing the highest quality, indistinguishable from the reference. The collected ratings are then statistically analyzed to assess the overall quality of the evaluated speech samples.

2.4 Speech-to-Speech Machine Translation

Speech-to-Speech Machine Translation (SSMT) systems are used to convert speech from one language to another language. Ensuring the faithful transfer of prosody from the source to the target language is essential for producing natural and human-like translations. Traditional speech translation systems overlook paralinguistic cues, but there has been a recent focus on systems that transfer linguistic content along with emphasis information. In contrast to conventional SSMT systems [40, 41, 42, 43], proposed methods to translate paralinguistic information, focusing specifically on stress/emphasis where source speech was English and target speech was Japanese. The approach in [40] involves the use of the Linear Regression Hidden Semi-Markov Model (LR-HSMM) to estimate a sequence of real-numbered emphasis values for each word in an utterance. Numerous studies have been conducted within the linguistic domains of Spanish, English, and Catalan such as [2, 1] reflecting a comprehensive exploration of prosody transfer techniques between these languages. Moreover, recent research initiatives have broadened their focus to incorporate the intricate linguistic environment prevalent in the Indian context [44]. Prosody often varies based on the context of the conversation, such as asking a question, making a statement, or expressing surprise. The inability to transfer prosodic features accurately results in robotic or unnatural-sounding translations, affecting the overall intelligibility and expressiveness of the translated speech.

They are of two types: cascaded and end-to-end. Our work only deals with cascaded systems composed of Automatic Speech Recognition, Machine Translation and Text-to-Speech systems. In this section, the individual components are discussed briefly (except TTS which has already been covered in Section 2.3). An English-to-Indian language SSMT pipeline work is also discussed to see how all the components go together.

2.4.1 Automatic Speech Recognition

Automatic Speech Recognition (ASR) bridges the gap between spoken and written language by converting spoken speech into text. This technology underpins numerous applications that have become integrated into our daily lives, including virtual assistants like Siri and Alexa, speech-to-text software, automated captioning systems, and voice search functionalities.

2.4.1.1 Whisper

Whisper [45], an open-source automatic speech recognition (ASR) system developed by OpenAI, has garnered significant attention for its state-of-the-art performance on various benchmarks. This success hinges on its deep learning architecture, specifically a transformer-based encoder-decoder model. Transformers excel at capturing long-range dependencies within sequences, making them well-suited for tasks like speech recognition where understanding the context across extended audio segments is crucial. Whisper's architecture leverages this strength to achieve highly accurate recognition even in challenging conditions characterized by background noise or diverse speaking styles (accents, dialects).

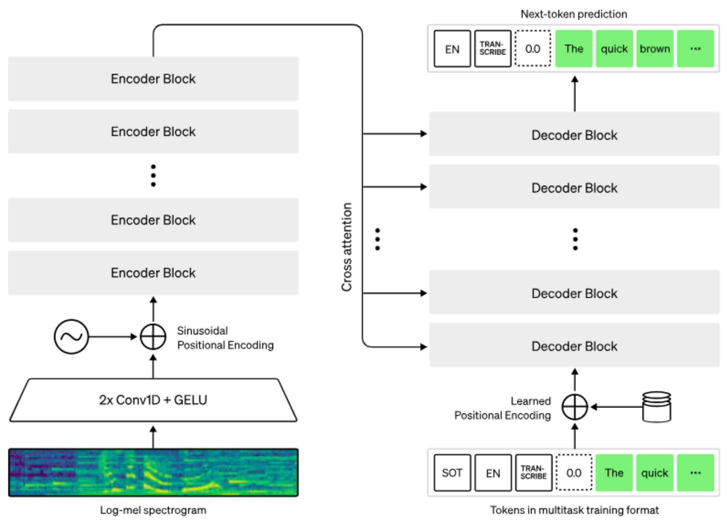


Figure 2.4: The architecture of Whisper as shown by OpenAI [45]: It uses a transformer-based encoder-decoder to convert speech into text.

A critical factor in Whisper's effectiveness is the sheer scale and diversity of its training data. OpenAI trained Whisper on a massive dataset exceeding 680,000 hours of multilingual and multitask supervised data collected from the web. This rich training corpus allows Whisper to not only transcribe speech in multiple languages but also translate it into English. Furthermore, the multitasking nature of the training data equips Whisper with a deeper understanding of the nuances of spoken language, leading to more robust performance.

2.4.1.2 WhisperX

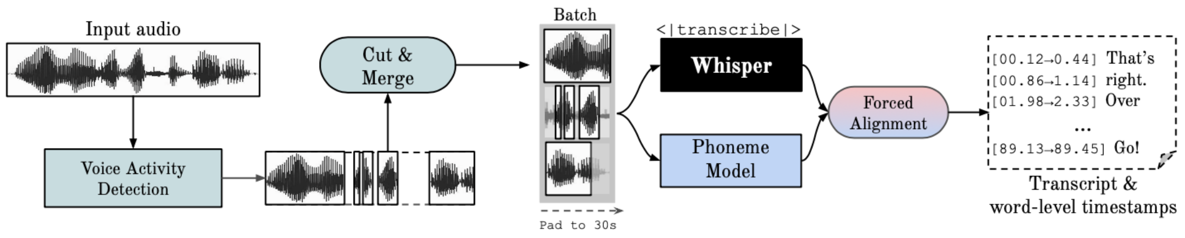


Figure 2.5: The pipeline used in WhisperX: It is composed of a VAD, the Whisper ASR model and finally a forced aligner *Source: [46]*

While Whisper demonstrates exceptional performance, its deep learning architecture demands significant computational resources. Building upon the core principles of Whisper, WhisperX offers a streamlined architecture that trades off some computational complexity for faster processing speeds.

WhisperX leverages Whisper's core functionality but adds three key preprocessing stages. An external Voice Activity Detection (VAD) model is employed to pre-segment the input audio. This isolates active speech regions, removing silence or background noise, and improves overall efficiency. It also uses Speaker Diarization from [47] along with VAD. It is a process of partitioning human speech into homogeneous segments according to the identity of each speaker. Then the VAD segments are further cut and merged into chunks of approximately 30 seconds, ensuring boundaries fall on minimally active speech regions. This allows for batched processing of the audio chunks using Whisper, maximizing the utilization of available computational resources. Following Whisper's transcription, an external phoneme recognition model based on [48] is used for forced alignment. This process refines the timing information, resulting in accurate word-level timestamps within the transcript. Its ability to deliver high-accuracy ASR while operating efficiently along with word-level timestamps and speaker diarization makes it a good choice in this thesis.

2.4.2 Machine Translation

Machine translation (MT) is a field that tackles the challenge of automatically translating text from one language to another. This technology has revolutionized communication by breaking down language barriers and fostering cross-lingual understanding. In recent years, the emergence of transformers [30], a novel neural network architecture, has significantly advanced the capabilities of machine translation models. These transformers have played a huge role in AI, pushing the boundaries of what machines can achieve in understanding and manipulating human language. The success of transformer-based models in machine translation has further fueled research into large language models (LLMs) like GPT-3 [49]. These LLMs are trained on massive amounts of text data, enabling them to not only translate languages but also perform a variety of other human-like tasks with impressive fluency and comprehension.

2.4.2.1 Opus-MT

The Helsinki NLP Opus MT [50] project offers a valuable resource for researchers, developers, and users seeking an open-source solution for machine translation (MT). Built upon the Open Parallel Corpus (Opus), a vast collection of multilingual text and speech data, Opus MT provides a suite of pre-trained models and tools designed for efficient and customizable machine translation tasks. Its open-source nature allows researchers and developers full access and the ability to modify the models and tools, pushing the boundaries of both translation accuracy and overall performance. Opus MT supports a diverse range of languages, catering to a wider spectrum of translation needs making it beneficial for applications requiring translation between less common languages. It also offers pre-trained models for various language pairs while also allowing for fine-tuning on specific datasets. It's important to acknowledge that its performance might not always match that of cutting-edge commercial MT systems, which often leverage proprietary training data and techniques. Nevertheless, Opus MT remains a compelling choice for transparent, customizable, and freely available MT software.

2.4.2.2 SimAlign

SimAlign [51] is a high-quality word alignment tool that uses static and contextualized embeddings and does not require parallel training data. It emerges as a powerful tool for word alignment in SSMT as it addresses a critical challenge: establishing precise correspondences between words in the source and target languages. Accurate word alignment is essential for tasks like vocabulary selection and reordering during translation. SimAlign tackles this challenge by leveraging pre-trained language models (LMs) such as mBERT [52]. This approach eliminates the need for parallel training data, a traditional hurdle for word alignment methods. It also incorporates both static and contextualized word embeddings from the pre-trained LM, enabling it to capture the semantic similarities between words across languages. The open-source nature of SimAlign allows the exploration of word alignment techniques and greater flexibility and applicability to a wider range of tasks.

2.5 Summary

The existing work in the different components that make up SSMT allows us to isolate the problem and focus on preserving stress. By using existing models and architectures proven to work to the highest standard, building a pipeline capable of transferring stress becomes feasible. Rather than building everything from scratch, which would be time and labour-intensive, this work focuses on building stress detection models and improving TTS models for stress capacity. With the right stress detection model and TTS capable of stress conditioning, along with the advances in ASR and MT, we should be able to build a pipeline that can preserve stress in the target speech.

Chapter 3

Stress Detection in Speech

3.1 Introduction

Stress, prominence or emphasis is a prosodic feature of speech accompanied by pitch and loudness variations. It can provide clues about the speaker's intent and improve recognition of keywords. For example, it can distinguish between a statement, a question, or a sarcastic remark depending on which words are stressed. Understanding the emphasis placed on certain words or syllables can highlight key information and influence the overall interpretation of an utterance. This work deals with transferring stress from the source speech to the target speech for a better and more engaging synthetic speech. This involves the detection of stress in the source speech which is in Indian English. This chapter explains the need for curating a stress dataset, the motivation behind the features used for classification, the different machine-learning models and deep-learning models used for the classification task, the post-processing step employed to make it more useful in the SSMT pipeline and lastly the results and insights obtained from the many experiments done.

3.2 Stress Dataset Construction

Existing speech datasets often lack the detail and nuance required for accurate stress detection in specific domains. This thesis tries to address this gap by creating a stress-annotated dataset specifically designed for Indian English spoken in educational video lectures.

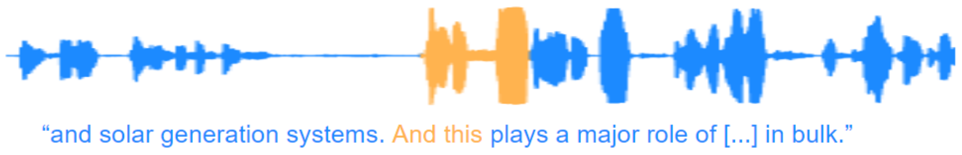

"and solar generation systems. And this plays a major role of [...] in bulk."

Figure 3.1: A speech signal with the stressed highlighted in *orange*. The transcript is just for reference and is not used when marking.

We leverage the quality and diversity of the IIITH-IED dataset [53], consisting of high-quality recordings of NPTEL [54] lectures. This dataset offers a diverse range of speakers and speaking styles, along with naturally occurring disfluencies common in spoken language. These disfluencies are crucial to consider, as our stress detection model needs to be robust and not misinterpret them as stressed regions. It needs to understand the difference between stress and disfluency and therefore having them in the dataset helps it learn this. In the context of lectures, instructors naturally emphasize key points and concepts by stressing specific words. These stressed regions often occur at sentence beginnings or endings, and exhibit characteristics such as prolonged duration and potential pauses. Depending on which part of the sentence is being stressed, the meaning that is being conveyed can also change. Hence the stress in speech needs to be put appropriately. To capture these nuances, the dataset was manually annotated by carefully instructed personnel who understood the nuances of speech. Annotators relied on both acoustic cues (pitch, duration) and visual indicators from the waveform, along with their understanding of the lecture content, to identify stressed regions. Annotators were also given instructions for patterns to look for when trying to identify the stressed region. This helps them to avoid giving false positives and get better agreed-upon regions that are consistent among the annotators. Figure 3.1 illustrates a sample with a highlighted stressed region (orange).

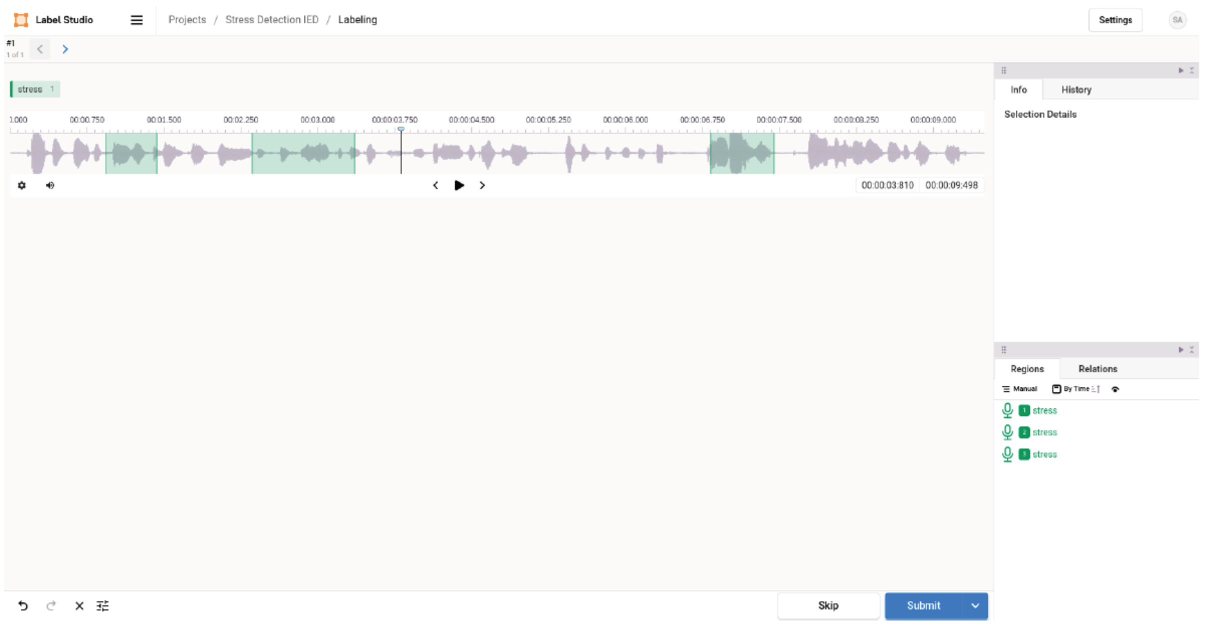


Figure 3.2: Label Studio software was used for annotating the datasets. The stressed regions in the speech signal were hand-marked and the corresponding timestamps were saved.

Label Studio [55], an open-source data labelling platform, facilitated the annotation process. Annotators designated stressed regions within audio files and generated timestamps for each, as shown in Figure 3.2. Any audio file could have multiple stressed regions or none at all. To account for the subjective nature of stress and emphasis perception, the audio data was distributed among ten annotators, ensuring each sample received labels from at least three different individuals. Then each annotator loaded their respective data into Label Studio and hand-annotated all their allotted speech samples. A custom interface was created as seen in Figure 3.2 to make it easier and faster for the annotators. By making sure that every sample is given to at least 3 different annotators, we end up getting different perspectives of what stress is for each sample. We employed Fleiss' Kappa inter-annotator agreement metric [56] to reconcile these multiple annotations and generate the final stress labels. Using this method, the best overlapping regions with the highest confidence can be found and used as the final timestamps. This step is necessary as stress is a very subjective perception. What can be qualified as stress depends on person to person. Hence a very detailed instruction set was created and given to the annotators to follow, which gives some guidelines to follow for considering stress. This along with the agreement metric ensures that the final timestamps noted down have the least possible bias from the annotators.

This effort has resulted in a stress-annotated dataset specifically tailored to Indian English spoken in NPTEL video lectures. The dataset consists of 3329 audio files, each lasting between 8 and 12 seconds, accumulating to nearly 10 hours of lecture mode speech. To ensure speaker diversity, the dataset features recordings from 30 male and 30 female speakers, each contributing approximately 10 minutes of data. The audio data is sampled at a rate of 16000 Hz. It's important to note that each audio file can have multiple stress annotations or none at all. This resource fills a critical gap in the landscape of stress detection datasets and paves the way for further advancements in this area of speech processing research.

3.3 Feature Extraction

For detecting stress in speech, we need to identify relevant features that can do the task effectively. The identified features are Pitch, Energy, Mel Frequency Cepstral Coefficient (MFCC), and Shifted Delta Coefficient (SDC). The reason for selecting these features is mainly based on the observations made by looking at the collected data. It is observed that most stressed regions are often accompanied by higher pitch and energy. By understanding how the pitch and energy contours are changing, we can get clues regarding stress in that region. The calculation of pitch and energy contours are discussed in Section 4.2.2. The motivation and calculation of the other two features are given in this section.

3.3.1 Mel Frequency Cepstral Coefficient

Mel-Frequency Cepstral Coefficients (MFCCs) have emerged as a go-to feature extraction technique in speech processing tasks. MFCCs aim to capture the characteristics of the human vocal tract system while aligning with how humans perceive sound [57]. They capture the nuances in the frequency domain, especially by factoring in the Mel filter banks that account for human frequency sensitivity. MFCC coefficients contain information about the rate changes in the different spectrum bands, which is important as stress production is not associated with just one frequency but rather a range of bands. By understanding how different frequency bands are changing, it will be easier to determine the advent of stress.

The speech signal is first divided into short frames (typically 20-40 milliseconds) to account for time-varying frequencies within the signal. A windowing function 4.1 is then applied to each frame to mitigate spectral leakage, an artefact that can occur during the transformation process. Then the power spectrum of each frame is computed. It is calculated by first obtaining the spectrum using FFT 4.1, and then applying a square to it to get the power spectrum. This spectrum represents the distribution of energy or power across different frequencies. MFCCs use the Mel scale 4.2, which approximates human auditory perception. A bank of filters spaced on the Mel scale is applied to the power spectrum, emphasizing frequencies that humans perceive more distinctly. The energies obtained from the Mel-scale filtering stage are compressed using a logarithmic function. This step further aligns with human perception, as human ears tend to perceive changes in intensity on a logarithmic scale. Finally, a Discrete Cosine Transform (DCT) is applied to the Mel-scaled and log-compressed filterbank outputs. The DCT helps to decorrelate the filterbank energies and extract essential features related to the shape of the spectral envelope. Typically, the first 12-20 coefficients obtained from the DCT are retained as MFCC features. In our work, we use the first 13 coefficients of the DCT as MFCCs, as we are interested in the lower frequency range where most human speech activity is present. We also find the first and second differential of the coefficients to get 13 extra values for each. These values model the change in 13 MFCCs and the change in the rate of change of MFCCs (akin to velocity and acceleration) respectively. By combining all these features, we end up with a 39-dimensional MFCC feature per frame of the speech signal.

MFCCs offer several advantages for stress classification. They effectively capture the spectral variations of speech that are often associated with stress, such as changes in pitch and formant frequencies. Additionally, the Mel scale filtering aligns the features with human auditory perception, making them robust to variations in speaker characteristics.

3.3.2 Shifted Delta Cepstral Coefficient

Shifted Delta Cepstra (SDC) [58, 59] represents another feature extraction technique employed in speech processing tasks. SDC focus on capturing long-term temporal information within the signal by looking at the information ahead and before the current frame. SDC is based on some cepstra capable of capturing the frequency information, and hence they capture long-term temporal information about changes in frequency. This ability to see in the future is what gives SDC an advantage over traditional MFCC. SDC They achieve this by incorporating information from multiple frames surrounding the current frame of analysis. The SDC feature extraction involves four parameters: N, d, P, k. N parameter represents the number of cepstral coefficients computed for each frame. These coefficients capture the spectral characteristics of the speech signal. d signifies the time advance and delay used in the delta computation. The delta coefficients capture the rate of change of the cepstral features over time. P defines the time shift between consecutive blocks. k specifies the number of blocks used to compute the

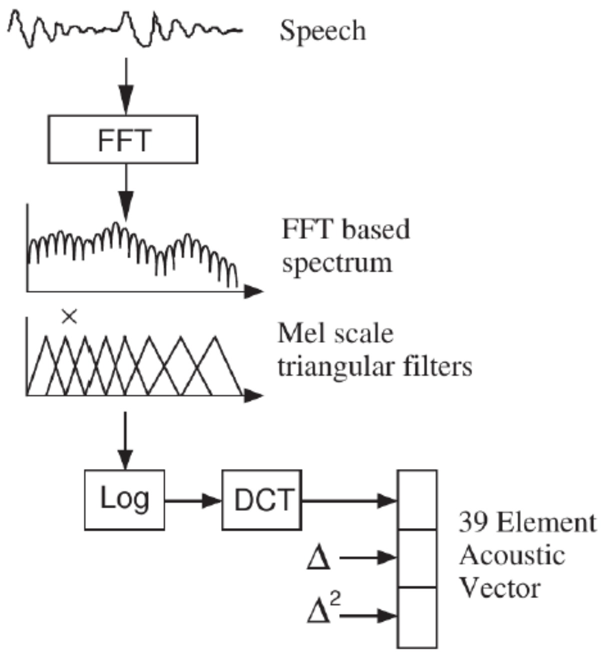


Figure 3.3: MFCCs are calculated from a speech signal using FFT and Mel scale triangular filters. The 39-dimensional vector is obtained from the first and second-order differentials of the original 13 features we get.

final SDC feature vector. Essentially, it dictates how many surrounding frames are incorporated into the feature extraction process. Let $x(t)$ be the speech signal, and $c(t)$ be the appropriate cepstra of $x(t)$. Then delta cepstra $\delta(t)$ is,

$$\begin{gathered}\delta_j(t) = c_j(t+d) - c_j(t-d) \\ j = 0, 1, 2, ..., N-1\end{gathered} \tag{3.1}$$

where, $c_j(r)$ is the j^{th} cepstral coefficient of frame at time step r. Once the delta cepstra is calculated as the difference between two cepstra, SDC is obtained by concatenating multiple such delta cepstra over multiple frames.

$$\begin{aligned}SDC(t) &= \delta_j(t + (i-1)P) \\ &= c_j(t + (i-1)P + d) - c_j(t + (i-1)P - d) \\ i &= 1, 2, 3, ...K\end{aligned} \tag{3.2}$$

By concatenating the delta coefficients from multiple frames around the current frame (as defined by k and P), SDC features capture a wider temporal context compared to conventional cepstral or delta-cepstral features. For instance, with a larger k value, SDC features can incorporate information

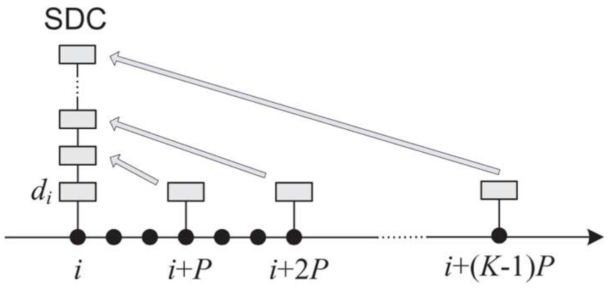


Figure 3.4: SDCs are calculated using cepstral features from multiple frames to get a higher temporal resolution. *Source: [60]*

from speech segments both preceding and following the current frame, potentially providing a more comprehensive representation of the speech dynamics. While requiring a larger number of parameters compared to conventional cepstral features, SDCs offer the benefit of capturing long-term dependencies within the speech signal. This characteristic can be particularly advantageous in tasks where the stress patterns may unfold over a longer duration, potentially making SDC features a valuable tool for stress classification, speaker identification, and other speech-processing applications.

3.4 Stress Prediction

Several models are used to predict the presence of stress using the above-explained features. The features extracted have some logical correlation to the task at hand, and these models will try to establish a relationship between the features and stress labels. The labels that are marked in the dataset are the start and end time stamps of stressed regions. We first apply framing or windowing on the entire signal to get these labels on the frame level. Then if a particular frame is present in any of the marked regions, it is labelled as 1 (stressed frame) otherwise 0 (not stressed frame). By doing this we get both features and labels at the frame level. All the features were first mean-variance normalized, and different subsets of features were considered to get the best performance. We concatenate the features on the frame level and then stack up those features from frames before and after to give wider contextual information. These stacked features are treated as an individual sample and the majority stress label of the group is chosen as the overall label. It is observed that the stressed regions are very few when compared to the non-stressed regions in speech, therefore the labels we get for the samples are also very skewed. The data imbalance needs to be tackled as the frames corresponding to stressed regions are much fewer than that of non-stressed regions. This would lead the model to be biased and hence is not desirable. We used SMOTE (Synthetic Minority Oversampling Technique) [61] based oversampling on the data to balance it. This technique fills the gaps in the under-represented class and tries to get a balance between both classes.

3.4.1 Machine Learning Models

Several machine-learning models were used for the task of stress classification. Machine-learning models are quite simple compared to deep-learning models but they outperform especially in the cases of fewer data points. But this comes with the issue of overfitting and a relatively simple understanding of the data. Despite these issues, machine-learning models learn the meaning of stress in speech and perform quite well in the frame-level classification task. In this section, only the best-performing models are discussed as the other models become irrelevant in the overall picture.

3.4.1.1 Random Forest Classifier

Random forests are a powerful ensemble learning method employed in supervised machine learning tasks for both classification and regression. This technique leverages the collective strength of multiple decision trees to achieve superior performance compared to individual trees. During training, a random forest constructs a collection of decision trees. Each tree is built using a random subset of features from the original data, promoting diversity within the ensemble. When making predictions, a random forest aggregates the outputs from all its constituent trees. In classification tasks, the most frequently predicted class by the individual trees becomes the final output of the forest. This approach helps to reduce variance and improve the overall accuracy of the model.

They are effective in handling high-dimensional data as only a subset of features is used to train each tree. This characteristic allows them to work well with a large number of features. The use of bagging (training each tree on a random subset of data) helps to prevent overfitting and improves the generalization ability of the model. These advantages make random forests a popular choice for various machine-learning applications.

3.4.1.2 Support Vector Machines

Support Vector Machines (SVM) is a powerful machine-learning technique for classification tasks. They excel at identifying the optimal hyperplane that separates data points belonging to distinct classes. In simpler terms, SVMs aim to construct a decision boundary that maximizes the margin between the two classes. This margin refers to the distance between the hyperplane and the closest data points from each class, also known as support vectors. These support vectors play a crucial role in defining the decision boundary. The equation for a hyperplane in SVM classification can be expressed as:

$$w^T * x + b = 0 \tag{3.3}$$

Here, w represents a weight vector that defines the orientation of the hyperplane in the feature space, x represents a data point, and b represents the bias term that determines the position of the hyperplane relative to the origin. The goal of an SVM is to find the w and b that maximize the margin, which can be intuitively understood as maximizing the absolute value of the decision function for the support vectors.

By focusing on these closest data points, SVMs achieve a robust decision boundary that is less susceptible to noise or outliers in the data.

Support Vector Classifier (SVC) is a type of SVM that offers various kernel functions to handle non-linearly separable data. The default kernel in SVC is linear, but kernels like RBF (Radial Basis Function) can be used to transform the data into a higher-dimensional space where a linear separation becomes possible. This flexibility allows SVMs to effectively handle a wider range of classification problems.

3.4.1.3 Label Propagation

Label Propagation [62] emerges as a semi-supervised learning technique for classification tasks. Unlike traditional supervised learning methods that require labelled data for all training instances, label propagation leverages the power of both labelled and unlabeled data. This approach is particularly advantageous when acquiring labelled data can be expensive or time-consuming. The core idea behind label propagation lies in the construction of a similarity graph between data points. This graph represents the relationships between data points based on their features. The labels from the limited set of labelled data points are then propagated throughout the graph, iteratively influencing the labels of unlabeled points based on the similarity between them. Points with high similarity are more likely to share the same class label. This propagation process continues until stable labelling is achieved for all data points.

Compared to methods like SVMs or random forests, which require fully labelled data for training, label propagation offers the advantage of utilizing both labelled and unlabeled data. This can be particularly beneficial in scenarios where labelled data is scarce. Additionally, label propagation can implicitly capture complex relationships between data points through the similarity graph, potentially leading to improved classification performance in certain situations. However, it's important to note that the effectiveness of label propagation heavily relies on the quality of the initial labelled data and the construction of the similarity graph.

3.4.2 Stress-Net

With a limited amount of work in stress detection, it is clear that deep learning methods have not been capitalized for this task. The machine learning models do give good accuracy but as a trade-off are not very generalizable. They especially fail in the cases of background noise, new speakers or low-intensity speech. The speciality of deep neural networks is their generalizability in such situations. Thus by leveraging Time Delay Neural Network's (TDNN) [63] ability to capture temporal and contextual information, we follow [64, 30] to build a neural network that is capable of detecting stress called *StressNet* as seen in Figure 3.5.

TDNNs are used in speech processing tasks, particularly in automatic speech recognition (ASR) systems. Within an ASR system, the acoustic model plays a crucial role in converting the speech signal into a phonetic representation. TDNNs excel in this task by effectively leveraging the temporal nature

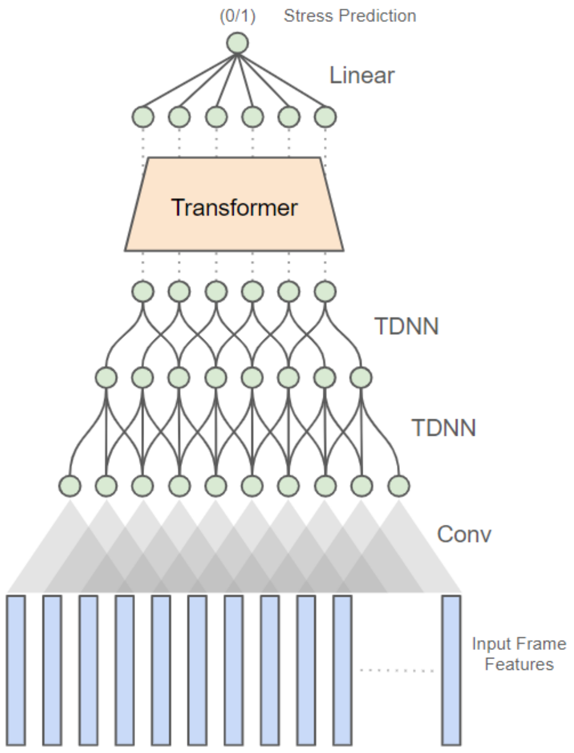


Figure 3.5: Proposed StressNet neural network to detect stress using frame level features. The TDNN context range shown is just for reference.

of the speech signal. In essence, a TDNN can be viewed as a one-dimensional convolutional neural network (CNN) without pooling layers [65] but with dilations. These dilations allow the network to capture context from the speech signal at varying time scales within each hidden layer. This capability is particularly advantageous for speech feature extraction as they can factor in their surrounding acoustic context. By selectively skipping frames very close to the target frame (which likely share high similarity) and focusing on context frames further out, TDNNs can effectively extract a more abstract representation from the raw speech signal.

StressNet takes a stack of frame-level features at a time as input. They are sent through a stack of 1D convolutions to capture short-range temporal information [4] followed by a stack of TDNNs with different contextual frames and dilations [64] with ReLU activation and 1D batch normalization. This will give a better receptive field and act as a feature extractor for stress markers. This is followed by a Transformer [30] and a linear layer with Sigmoid activation to give a single value prediction that denotes whether the group of input frames are stressed or not. A group of frames are categorized as stressed if most frames are labelled stressed.

3.5 Post-processing in Stress Prediction

3.5.1 Frame-Level to Word-Level Prediction

The models discussed above are designed to predict stress in speech on the frame level. That is, if the model is given features for a single frame, it will determine whether the frame is stressed or not. This method gives us better results and high resolution for identifying the boundaries. However since this model is part of a bigger SSMT pipeline, it needs to be seamlessly integrated into it. As covered later, the information exchange regarding stress between the source and target language is done on the word level.

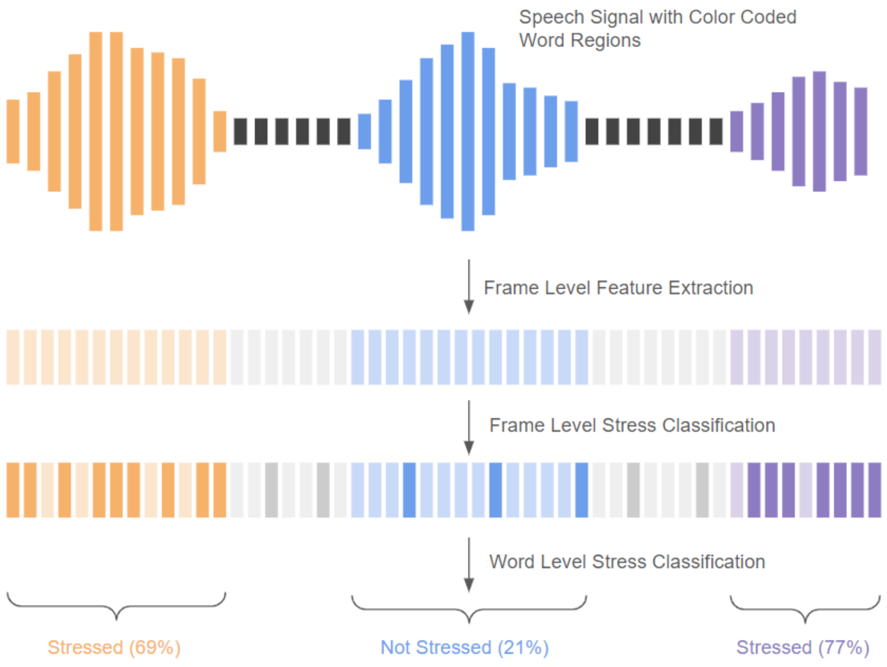

Speech Signal with Color Coded Word Regions
Frame Level Feature Extraction
Frame Level Stress Classification
Word Level Stress Classification
Stressed (69%)
Not Stressed (21%)
Stressed (77%)

Figure 3.6: The speech signal is converted to frame-level features. Using these features, frame-level labels are obtained from the models. Finally using word alignments from the ASR, word-level labels are obtained.

ASR systems are used to convert speech to text. There are many techniques for achieving this, but nearly all involve speech and text alignment. This is because we need to identify which speech segment corresponds to what text. By figuring out the most probable text from the given speech segment, we get both the speech transcript and alignment between them. Usually, these alignments are between the speech signal frames and the text token, where each text token is aligned to a bunch of frames. We take this additional alignment information from the ASR and use it in the post-processing stage of stress detection. Once the frame level labels are calculated using the stress classification models, we can use the frame level-to-text token alignments to get word-level classification for stress. This is done by first identifying which frames correspond to a word using the ASR alignments, and then looking at each of the predicted frame labels. If most of the frames corresponding to the word have been identified as stressed then the entire word is labelled as stressed. Using this method as seen in Figure 3.6, we can achieve word-level stress detection with very high accuracies. This word-level stress prediction in the source speech is later used to figure out the word-level stress in the target-language speech using other components in the SSMT pipeline.

3.5.2 Scaling Factors

Another important part of the post-processing module involves extracting relevant information from the speech signal that can describe the stress cues. Once we obtain the information about which words have been stressed in the source language, we also need additional information about what the stress looks like. This is because stressing a word is not always a fixed action and entirely depends on the situation and person speaking. To address this, we introduce what are called scaling factors.

The pitch and energy values are calculated in the region corresponding to every stressed word. Then we calculate scaling factors that describe how the pitch and energy values in the stressed region are, compared to the rest of the signal. Dividing the values of pitch and energy in the stressed region with their corresponding values in the total signal gives the scaling factors. The idea here is that since every person speaks differently, rather than using the absolute values of pitch and energy to denote the stress we instead use relative values. This helps mitigate some level of speaker-dependent characteristics and gives us better generalizability. These scaling factors are quite useful in SSMT as they serve as a metric for the TTS system to figure out how to induce stress. Using the final target-language stressed word information and the corresponding scaling factors, which are given as a pair of information, the TTS system modifies the pitch and energy variances of the synthesized speech to add stress in the appropriate locations.

3.6 Experimental Setup

The setup consists of multiple machine-learning models and one deep-learning model called StressNet. All the models were trained on the stress dataset collected, which is explained in Section 3.2. The evaluation for these models is performed on the eval split of the stress dataset. The models used are explained in Sections 3.4.1 and 3.4.2 respectively. The stress dataset consists of nearly 10 hours of lecture-mode speech in the context of educational video lectures from [53]. It features recordings from 60 speakers (30 male and 30 female), where each speaker has around 10 minutes of speech. These 10-minute recordings are further split into smaller audio files of 8 to 12 seconds, with a sampling rate of 16000 Hz. The dataset comprises 3329 audio files where each audio file is identified using a unique ID and has information about the speaker and the stressed region labels. Each audio sample can have multiple stressed regions or none at all. There are a total of 3807 occurrences of stressed regions with an average duration of about 0.53 seconds. Features such as F0, energy, Mel Frequency Cepstral Coefficients (MFCC) and Shifted Delta Coefficients (SDC) were extracted on the frame level and are used as input to

StressNet. 13 MFCC, 52 SDC features were extracted using a frame length of 1024 and hop length of 256. The SDC features were calculated using $d = 1, p = 5, k = 3$ as they gave the best results. The F0 and energy contours are averaged on the frame level, and all the features are mean-variance normalized and stacked to get the final frame-level features. The model takes a stack of frame-level features as input at once and predicts if the majority of frames given are stressed or not. The stressed regions in speech are smaller than the non-stressed regions in speech. Therefore the stressed frames will be much fewer when compared to the unstressed frames leading to an imbalance in the data. This is addressed using the Synthetic Minority Oversampling Technique (SMOTE) [61], a technique that oversamples under-represented classes to balance the data. The top three performing machine-learning models are considered which are explained in Section 3.4.1. Label Propagation Algorithm (LPA) with radial basis function kernel and 7 neighbours, Random Forest Classifier (RFC) with 100 trees in the ensemble and Support Vector Classifier (SVC) with a radial basis function kernel and penalty factor of 0.8 were used as the machine-learning models. StressNet, a DNN composed of TDNN and Transformers as explained in Section 3.4.2, was trained for 150 epochs on a single NVIDIA A100 80GB Tensor Core GPU with a batch size of 256.

3.7 Results and Discussion

The performance of detecting stress in speech is done using binary classification accuracy and F1 score. The frame level labels from ground truth and the models are compared and the scores are calculated. Post-processing is a part of the stress detection model, therefore to understand how effective that step is, another metric is used: **Post Accuracy**, meaning accuracy after post-processing. This score involves finding the word-level labels from both ground truth and the predictions and comparing them at the word level. The conversion of frame-level labels to word-level labels is explained in Section 3.5.1.

The performance of the machine-learning models is shown in Table 3.1. It shows the **F1 Score, Accuracy** and **Post Accuracy** of the binary classification of stress labels. The experiments were carried out on two feature sets: (F0, Energy) and (F0, Energy, MFCC, SDC) using different classifiers. It can be seen that LPA having a window size of 7 with the addition of the post-processing stage improves the accuracy and f1 score for both feature sets with an absolute improvement of around 2-4$\%$. RFC gave better results compared to LPA and SVC in the case of all features when the window size is 5 with an absolute improvement of around 5$\%$. SVC gave better results compared to LPA and RFC in both cases when the window size is 3 with an absolute improvement of around 2-3$\%$. Post accuracy is more relevant for SSMT, therefore it is given the highest priority when comparing the models.

The performance of StressNet is calculated using the **F1 Score** and the **Post Accuracy** of the stress label predictions. In the context of SSMT, word-level post accuracy is more relevant than frame-level accuracy. Hence, the comparison gives the most importance to post accuracy. Post accuracy is calculated after converting frame-level stress predictions to word-level stress predictions and then compared to ground truth stressed words. The frame-level features are stacked with different sizes and are given

Features	Models	Window Size = 3			Window Size = 5			Window Size = 7		
		Accuracy	**F1**	**Post Accuracy**	**Accuracy**	**F1**	**Post Accuracy**	**Accuracy**	**F1**	**Post Accuracy**
F0 and Energy	LPA	60.6	60.45	71.11	68.9	68.77	**81.85**	75.3	75.16	**84.18**
	RFC	58.2	58.06	69.52	68.6	67.59	76.75	72.8	71.79	82
	SVC	61.1	60.5	**73.35**	67.2	66.11	80.3	71.5	70.46	83.83
F0, Energy, MFCC, SDC	LPA	63.1	61.09	75.28	62.2	58.52	74.57	82.3	80.77	**90.36**
	RFC	64.3	63.28	77.37	71.1	70.3	**81.22**	78.3	78.27	85.29
	SVC	66.8	66.79	**78.42**	70.1	70.09	80.12	79.2	79.2	83.82

Table 3.1: Evaluation of machine-learning stress detection models with different features and varying window sizes.

together as input. The model then predicts if the frames are stressed or not, but since it gives only one value, it is considered to be denoting if the majority of frames are stressed or not. Experiments were done using different window sizes and different sets of input features to see which combination yields better results. The best machine-learning stress detection model, which is the LPA model is used as the baseline for a comparative understanding and was compared in different window sizes and feature spaces as seen in Table 3.2. The LPA model was re-implemented for comparison against StressNet, therefore the values seen in Table 3.1 and 3.2 might differ a bit, but the relative trends are preserved. Longer window sizes were taken as it was clear that StressNet was not giving good results when dealing with smaller window sizes as they contain less information.

Models (Features)	Window Size = 7			Window Size = 10			Window Size = 15		
	Accuracy	**F1**	**Post Accuracy**	**Accuracy**	**F1**	**Post Accuracy**	**Accuracy**	**F1**	**Post Accuracy**
LPA (F0 and Energy)	75.3	73.16	**84.18**	76.21	74.25	83.76	77.33	76.11	84.41
StressNet (F0 and Energy)	67.49	65.34	76.22	83.21	77.54	**87.54**	80.66	80.12	**88.23**
LPA (F0, Energy, MFCC, SDC)	82.3	80.77	**90.36**	82.53	82.31	**91.32**	81.22	80.54	91.02
StressNet (F0, Energy, MFCC, SDC)	78.12	77.31	88.19	83.44	81.12	90.12	86.01	85.48	**94.34**

Table 3.2: Comparison between LPA and StressNet stress detection models with different features and varying window sizes.

StressNet consistently gives better results, especially in the case of $window\ size = 15$ with all features used. It can be seen that StressNet outperforms LPA with a higher window size, which could be attributed to the temporal capability of TDNN. LPA however outperforms for lower window size, which could mainly be because the amount of information given to StressNet is not sufficient for learning more advanced patterns as it is a neural network. StressNet can still give comparable results in such scenarios, which could be due to the short-term capacity of the Convolutions used. However, the overall highest accuracy and F1 scores are higher in the case of StressNet, showing that it can outperform the best machine-learning model in the task of stress detection in speech. These results prove that the machine-learning models do give good results for the task of stress detection, but the deep-learning model StressNet can outperform them.

Statistical significance testing was also performed for analysis of the stress dataset. One-way ANOVA (Analysis of Variance) which tests the null hypothesis that two or more groups have the same population mean was used for this task. The test gives us two values: F-Statistic, which tells if there is a significant difference among the groups and P-Value, a number that tells us if the differences we see are real or just due to random chance. If the P-Value is less than 0.05 then we would reject the null hypothesis.

In our testing, all the P-Values were almost zero (¡ 10-8). This means the null hypothesis can be rejected. We compare the F-Statistic directly to understand how different the groups are as seen in Table 3.3. The groups considered are stressed frame features and unstressed frame features. We also analyze how the window size for frame stacking affects this test.

Window Size	**F Statistic for Feature**			
	F0	**Energy**	**MFCC**	**SDC**
1	73.12	99.02	35.78	80.87
3	87.54	62.10	76.19	82.69
5	78.04	96.64	52.82	85.45
7	106.56	96.31	114.12	91.94

Table 3.3: One-way ANOVA for stress dataset

3.8 Limitations

While the models give good accuracy and reliability, it can be seen that they usually overfit the source dataset and style. Despite efforts to make them more generalizable like StressNet, these models generally under-perform if the speech sample provided is from a different setting. The changes in microphone quality, surroundings, accent, and speaking rate are some factors that affect the performance of the models. The data is recorded using a similar microphone [53] and in the same setting, and any changes in these factors can affect the model as they have a strong correlation to what the model learns. Since the models are trained in Indian English, they become accent-sensitive and usually don't work with the same accuracy on other accents or languages. In this work, it was a deliberate decision but we can extend this model to be more generalizable. By improving the data available for research, we can overcome this obstacle in the future.

3.9 Summary

In this chapter, a stress-annotated lecture-mode speech corpus in Indian English and models that can detect stress in speech are discussed. The post-processing module required to integrate the stress detection model into SSMT is also discussed. The following are the observations from this chapter. Features such as F0, energy, MFCC and SDC are useful for detecting stress. The Label Propagation Algorithm outperforms other machine-learning models in stress detection. DNN-based StressNet outperforms all the other models in the task of stress detection. Having a larger window size for the feature stack is shown to give a better performance. Using the post-processing module, converting the frame-level predictions into word-level predictions improves the quality of the predictions.

Chapter 4

Stress Transfer using Text-to-Speech

4.1 Introduction

Text-to-Speech (TTS) systems are capable of generating speech from text and other cues depending on the task. But most TTS in literature [4, 5, 6, 7, 9, 24] only work with text as input, that is a neutral TTS. They learn how the text and corresponding speech signal are related to each other and figure out how to generate speech from any arbitrary text provided. The generated output has some prosodic elements to it, but these are learnt from the source data itself. This means that if the dataset itself was neutral, the generated speech would be neutral as well and would have a similar tone as that of the source data. In this case, the system does not understand the concept of prosody but rather directly tries to map the text and speech signal which inherently has prosody in it. Therefore, to have control over how the prosodic elements appear in the output speech, these systems cannot be used directly. Modifications are required to allow the TTS to condition the generated speech on external cues rather than just text.

The fidelity of Speech-to-Speech Machine Translation (SSMT) hinges not just on semantic accuracy, but also on capturing the speaker's intent conveyed through subtle prosodic cues like stress. While existing systems excel at translating meaning, they often stumble when replicating the emotional weight and emphasis embedded in the source speech. In this work, we design and propose an SSMT system that is capable of preserving stress in the source speech and conveying it in the target speech. This task requires new components to be added to handle stress information effectively. Some of the components have been explained in detail like Stress Detection in Section 3.4.1 and 3.4.2 and Stress Detection Post-processing in Section 3.5.1 and 3.5.2. Other components in this pipeline need to be addressed to make the entire system work. For this, we need a TTS system capable of using the stress cues as a conditioning input. A way of modifying the existing neutral TTS systems is required so that they can add stress to the generated speech. In this section, the modifications and additions made to these systems are discussed for them to generate stress. As explained in the previous Section 3.5.2, the stress cues or stress information is given as a pair of stressed words and their corresponding scaling factors. This section also covers the overall construction of the SSMT pipeline that integrates all the components required to perform this task.

4.2 Feature Extraction

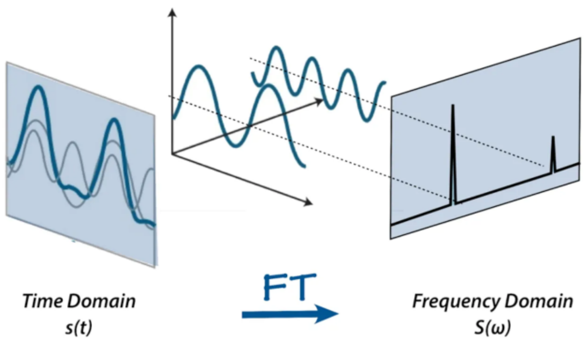
Extracting spectral and temporal features from raw speech data is a crucial step in Text-to-Speech (TTS) systems. This extraction transforms the speech waveform, representing air pressure variations over time, into a more manageable and informative format suitable for speech synthesis. Two primary feature extraction techniques used in TTS are Linear spectrograms and Mel spectrograms. These spectrograms are used over waveform to represent speech as they are much more manageable in terms of the size required to store the signal and they capture the frequency information of the signal. Extraction of prosodic information in a speech signal such as pitch and energy is also discussed in this section. This is because the stress in speech is heavily dependent on these prosodic contours, and to add stress in speech we need to use pitch and energy contours.

4.2.1 Linear and Mel Spectrogram

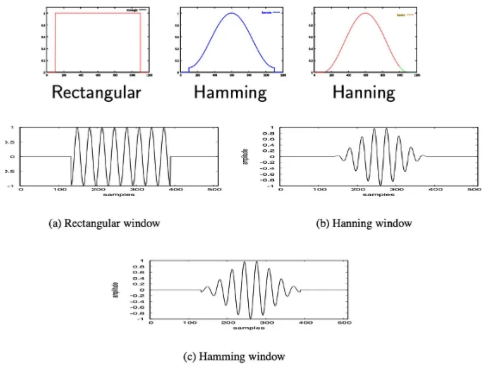
Spectrograms offer a visual representation of a signal's frequency content over time. In the context of speech, they depict how energy is distributed across different frequencies within a short audio segment. Audio signals are essentially combinations of pure tones at various frequencies. However, recordings only capture the amplitude of the combined signal at each time point. For quite a long time, understanding the frequency information from a time domain signal was not a trivial task. But that changed when the Fourier transformation was invented. The Fourier transform is a mathematical tool that decomposes this complex signal into its constituent frequencies and corresponding amplitudes. This process essentially translates the signal from the time domain (how it changes over time) to the frequency domain (what frequencies are present), resulting in a representation known as a spectrum. With the power of the Fourier transform, a diverse set of problems were solved which were previously not achievable. But this power comes with a huge computation cost, that is this algorithm works on $O(N^2)$ making it practically useless when dealing with large-scale speech processing. Therefore a variant of the Fourier Transform called the Fast Fourier Transform (FFT) [66], a hardware-optimized algorithm was invented, which changed not only the signal processing domain but the entire scientific community. This algorithm takes N discrete points representing a signal and returns N values that denote the intensity of each frequency bin corresponding to the frequency range (the frequency range of a given signal is determined by the sampling rate using the Nyquist Sampling Theorem [67]). This algorithm did this in $O(N \cdot log(N))$ time, meaning for processing one second of audio sampled at 16kHz, it was almost 4000 times faster than the normal Fourier transform. Such incredible optimization allows us to compute features in the blink of an eye! The equation given below denotes the Discrete FFT algorithm,

$$X(k) = \frac{1}{N} \sum_{n=0}^{N-1} x(n) \cdot e^{-j\frac{2\pi}{N}kn} \tag{4.1}$$

But spectrum using the Fourier transform can calculated only for a periodic signal, whereas a speech signal is quasi-periodic. Therefore we use Short Term Fourier Transform (STFT) where FFT is

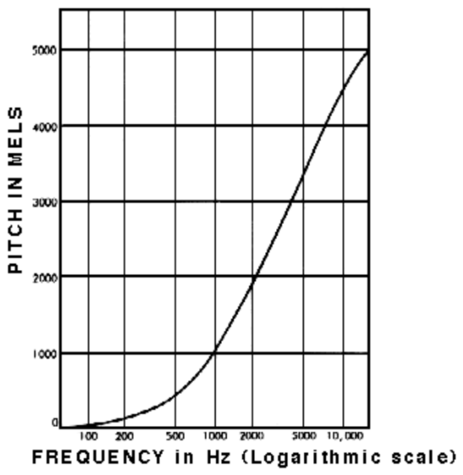


(a) Time to Frequency Domain

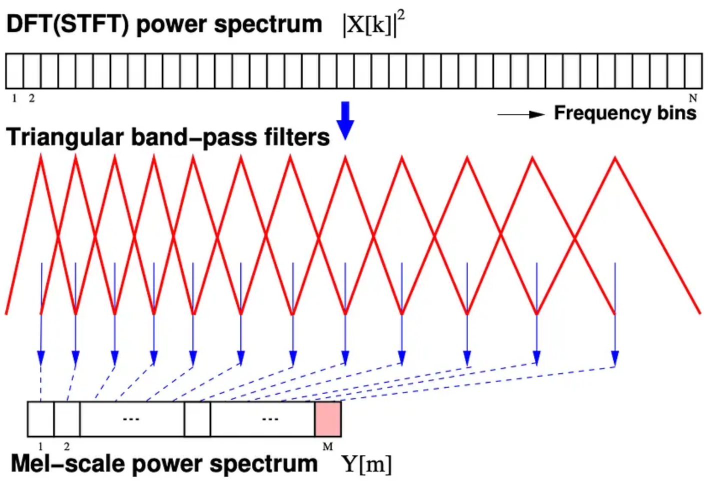


(b) Window Shapes

Figure 4.1: (a) Depicts the modality change of the signal from time domain to frequency domain. (b) Depicts the different window shapes available when dividing the signal into small slices, called frames.



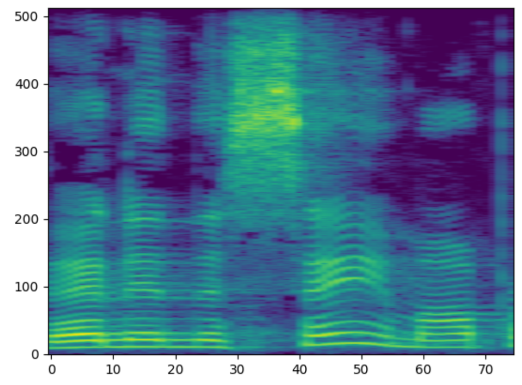
(a) Mel Scale Transformation


DFT(STFT) power spectrum |X[k]|²
Frequency bins
Triangular band–pass filters
Mel–scale power spectrum Y[m]

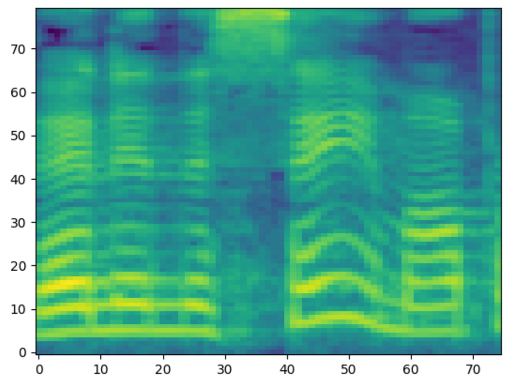
(b) Mel Filter Banks

Figure 4.2: Mel Scale Transformation for converting Linear frequency range into Mel frequency range *Source: [68]*; The Mel Filter Banks that convert regular power spectrum into Mel-scale power spectrum using triangular band-pass filters.

applied on a small frame of the speech signal, usually around 20-30 ms. To aggregate this information more effectively over the entire signal, the spectrums for each frame are stacked into what are called spectrograms. Imagine the spectrum as a snapshot of all the frequencies present at a specific moment. We obtain a sequence of these snapshots by calculating the spectrum for short, consecutive slices of the audio signal. Stacking these snapshots together, with frequency on the y-axis, time on the x-axis and intensity as a colour gradient, creates a spectrogram. This visual representation allows us to see



(a) Linear Spectrogram



(b) Mel Spectrogram

Figure 4.3: The linear spectrogram gives each frequency bin equal importance. Mel spectrogram gives more preference to the lower frequency range which can be seen as the stretching of light green formant contours.

how the frequency content of the audio signal changes over time, providing valuable insights into the characteristics of the sound. Linear spectrogram is the normal version of this where each frequency range is given equal importance.

While informative, Linear spectrograms don't account for how humans perceive sound. Our sensitivity to changes in pitch varies across the frequency spectrum. We are more attuned to variations at lower frequencies compared to higher ones. Mel spectrograms address this by incorporating a Mel scale which approximates human auditory perception as seen in Figure 4.2. This non-linear transformation warps the linear frequency scale according to the Mel scale transformation formula, resulting in a spectrogram with higher resolution at lower frequencies and lower resolution at higher frequencies. This spectrogram is called the Mel spectrogram and it better reflects how humans perceive the pitch of speech sounds. Figure 4.3 shows the difference in frequency distribution between a Linear and Mel spectrogram.

4.2.2 Pitch and Energy Contours

Pitch contour corresponds to the fundamental frequency (F0) of the speaker's voice, essentially representing the perceived "highness or lowness" of the voice. It provides insights into the speaker's vocal range and how pitch varies throughout the utterance. This information proves valuable in TTS applications, as evidenced by its use in models like FastPitch [5]. FastPitch leverages pitch contour as one of the predicted variances, enabling fine-grained control over the pitch of synthesized speech. As stress in speech is usually accompanied by a higher pitch tone it is useful to get pitch information as an extra conditioning feature. In our implementation, we calculate the pitch contour at the frame level using the Librosa library's [69] pYIN algorithm [70, 71]. However, since the encoder of both TTS models works on the text token level, the pitch contour also needs to be adjusted for this. To get this feature on the text token level, the frame-level pitch values are averaged at each text token level, thus giving each text token a single pitch value. This allows the model to predict pitch variations aligned with the linguistic structure of the input text.

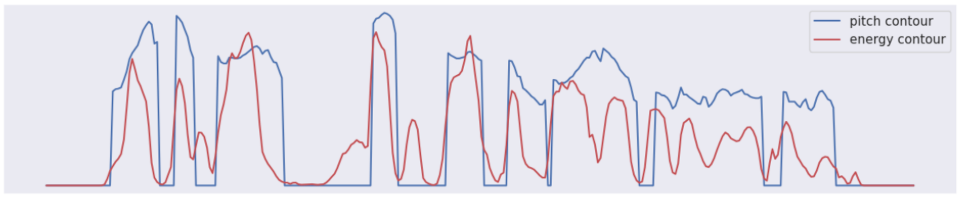


Figure 4.4: The pitch contour and energy contour plotted for a speech sample

Energy contour, on the other hand, reflects the loudness or overall energy present in the speech waveform. Visually, it manifests as regions of higher amplitude corresponding to higher energy segments. This is useful for stress as a higher energy tone usually accompanies it. Energy contour calculation typically involves computing the mean squared value of the speech signal. Similar to pitch contour, energy contour is initially calculated on a frame-by-frame basis. However, to align with the text tokens, these frame-level values are averaged to obtain an energy level associated with each text token. By incorporating predicted energy contours, the TTS system gains better control over the loudness of the generated speech, allowing for more natural and expressive synthesis. The energy of a signal $x(n)$, denoted by $E(x(n))$, that is composed of n samples is given as,

$$E(x(n)) = (1/n) * \sum_{i=1}^{n} x[i]^2 \tag{4.2}$$

4.3 Stress synthesis using FastPitch

The base FastPitch TTS model as explained in Section 2.3.2.2 is a neutral text-to-speech architecture. It predicts the pitch contour internally to give more fine-grained control of the pitch of the generated speech. However, this control is not sufficient to generate stress in the output speech but is a great starting point. Modifications were made to FastPitch architecture as seen in 4.5 to condition the synthesized speech on stress cues.

The model takes both text and stress cues as input rather than just text. The text is given as character tokens to the model, where the input text is first normalized using different techniques and then fragmented into characters. The stress cues are given as a list of stressed words and their corresponding scaling factors 3.5.2, generated using the stress detection model and post-processing for word labels. An energy prediction block is also added to the architecture. This is because as explained in Section 4.2.2 stressed regions of speech have a strong correlation with both pitch and energy. Therefore having an energy prediction block will enhance the quality of the stress and give more control over the addition of stress. The model now has two variance predictors, pitch and energy, but it is still a neutral TTS. This is because both the variances are predicted based on text, and then a spectrogram is predicted based on these variances. Till now we haven't added the stress cues in the architecture yet. Therefore a Pitch-Duration-Energy (PDE) Modifier block as shown in Figure 4.6 is introduced, which modifies the predicted variances using the stress cues. As it is known that stress is associated with high pitch and energy compared to the rest of the speech, we can use this information to our advantage when modifying the variances. We have access to information regarding which word has to be stressed in the target language (explained in Section 4.5), and its scaling factor, and therefore we can directly modify the variance contour accordingly. We first identify the region in the contour corresponding to a stressed word, and we multiply that region with the appropriate scaling factor associated with the stressed word. Using this method, the contours are scaled based on how the stress has appeared in the source speech. This is a critical assumption, where we assume that if a word has been stressed in the source speech, then the corresponding translated word in the target speech also needs to be stressed at the same scale. This allows us to train the TTS on neutral speech-text datasets that are already available, avoiding the whole hassle of creating a large clean custom speech dataset just for TTS. Since TTS requires very high-quality clean speech data, procuring such amounts of stressed data is not trivial and our method overcomes this issue by using the PDE Modifier block only during inference.

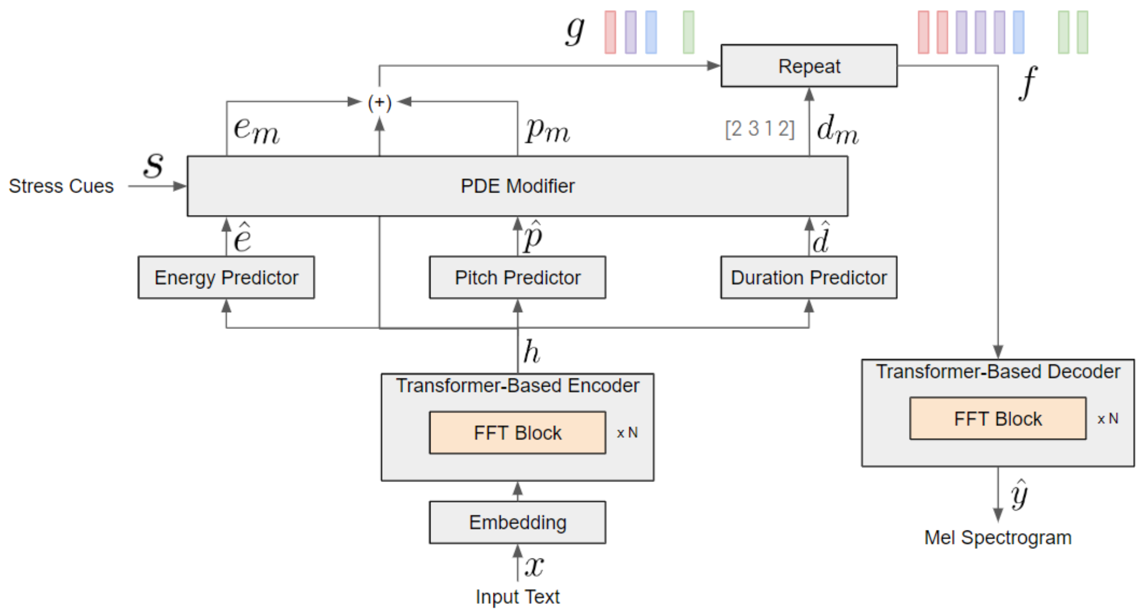


Figure 4.5: In the proposed design, both pitch and energy contour variances are predicted. This allows better modelling of stress. The predicted variances are modified by the PDE modifier to add stress.

Let x be input character tokens and s be input stress cues. The normalized text character tokens are first converted to indices using a predefined map. The token embeddings for the text indices are generated using an embedding layer, as we know the entire character space from the mapper. The token embeddings are passed to a feed-forward Transformer (FFT) encoder stack [30] that produces the hidden representation $h = \text{FFT}_e(\text{Embedding}(x))$. This hidden representation h of the text embeddings is considered as the encoder output and contains linguistic information and content in the text. This is then used to predict each input character token's duration $\hat{d}$, pitch $\hat{p}$ and energy $\hat{e}$ using a variance predictor. The variance predictor [5] is composed of a 1D CNN stack and a fully connected layer. The pitch, duration and energy values are then sent to the PDE Modifier block, where each variance is modified based on the stressed word region and the scaling factors provided as explained above.

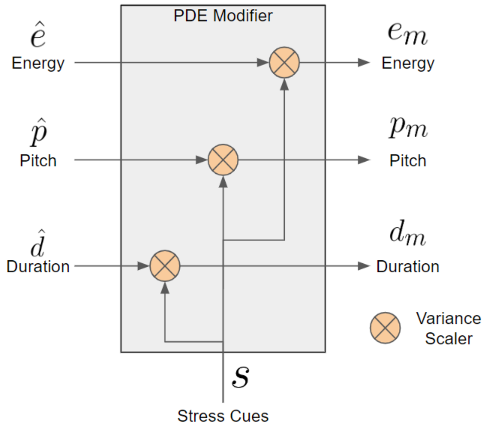


Figure 4.6: The PDE modifier uses the scaling factors and scales the predicted variances for all the corresponding stressed word regions.

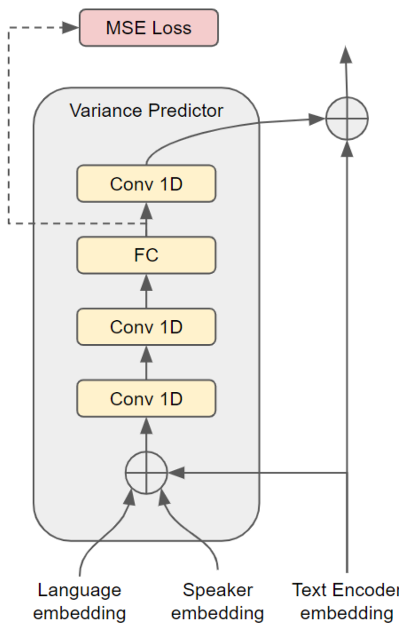
Next, the modified pitch p_m and energy e_m are projected to match the dimensionality of the encoder output h using an embedding layer. This allows us to combine the pitch and energy information to the encoder output to affect the generated speech appropriately. We combine all these embeddings using the vector addition operation, $g = h + P_{emb}(p_m) + E_{emb}(e_m)$. The resulting embedding g is at the resolution of the encoder output which itself is at the resolution of input text tokens. Therefore we need to upsample these embeddings to go from character level in text to frame level in speech. This can be done using the predicted duration values, as the duration predicted for each character denotes the number of frames it corresponds to in the final speech. The embeddings g are discretely upsampled using the modified durations d_m to give f which is now at the resolution of output frames. This upsampling is done by repeating the embedding of one character by the number of frames it corresponds to and then stacking them together. This converts the embeddings to the spectrogram space as we are dealing at the frame level. f is then passed to an FFT decoder stack FFT_d to produce the output Mel spectrogram sequence $\hat{y}$. During training $\hat{y}$ is generated using the variances predicted from the text. But during inference, it is generated using the modified variances from the PDE Modifier block 4.6. Since the model is trained to predict spectrogram from a given pitch and energy contour, it will be able to generate speech from the modified variances as well. Thus we incorporate stress in the synthesized speech. Since the PDE

Modifier block is used only during inference, the training loss is based on the mean-square error (MSE) of the predicted spectrogram and ground truth spectrogram, similar to that of FastPitch. The training loss also involves the MSE of the predicted variances and ground truth variances.

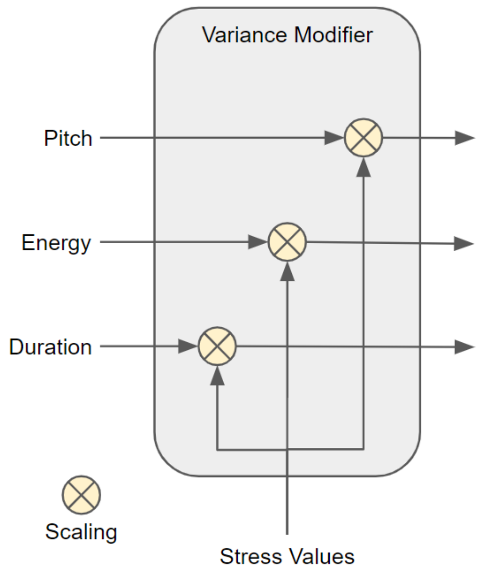
$$\mathcal{L} = \|\hat{y} - y\|_2^2 + \alpha\|\hat{p} - p\|_2^2 + \gamma\|\hat{e} - e\|_2^2 + \delta\|\hat{d} - d\|_2^2 \tag{4.3}$$

4.4 Stress synthesis using YourTTS

The base YourTTS model as explained in Section 2.3.2.2 is neural end-to-end text-to-speech architecture. It uses normalizing flows and Monotonic Alignment Search (MAS) for learning a mapping between input text and speech. This architecture doesn't have pitch or energy predictors but has a duration predictor in the form of a Stochastic Duration Predictor (SDP). But all three variances are essential for stress generation as discussed above. This TTS was chosen despite these issues because of its reliability and quality. This architecture is a state-of-the-art model that can perform voice cloning reliably through speaker disentanglement. Therefore, similar modifications were made in this architecture to account for variance prediction and allow the addition of stress.



(a) Variance Predictor



(b) Variance Modifier

Figure 4.7: (a) Pitch and Energy Variance Predictor composed of 1D convolution stack. (b) Variance Modifier or PDE Modifier that scales the contours based on scaling factors

The proposed design adds variance predictors for pitch and duration to give the capability of stress synthesis. This addition is non-trivial as it was originally designed for a different purpose. In the case of

FastPitch, the pitch predictor is already present and adding an energy predictor is straightforward. Hence, identifying the proper location in the information flow for adding variance information is crucial. After experimentation, results have shown that adding variance predictors right after the text encoder generates a stable output as compared to adding those in other locations. The variance predictors as seen in Figure 4.7 comprise a stack of 1D convolution layers with ReLU activation in between, following the work of [5]. These variance predictors calculate the values of pitch and energy on the text token level. They predict these values using the text encoder output, which itself is at the text token level. The prediction is conditioned on speaker and language embeddings, which is the main feature of YourTTS as this allows us proper disentanglement. The predicted variance contour is then used to generate embeddings of text encoder output dimension using a projection layer composed of 1D convolutions and is added to the text encoder embeddings as a skip connection [5, 72]. This includes the variance contour information into the embeddings and serves as the conditioning factor when generating speech. The predicted text token level variance values are compared to the ground truth text token level variance contour using MSE loss.

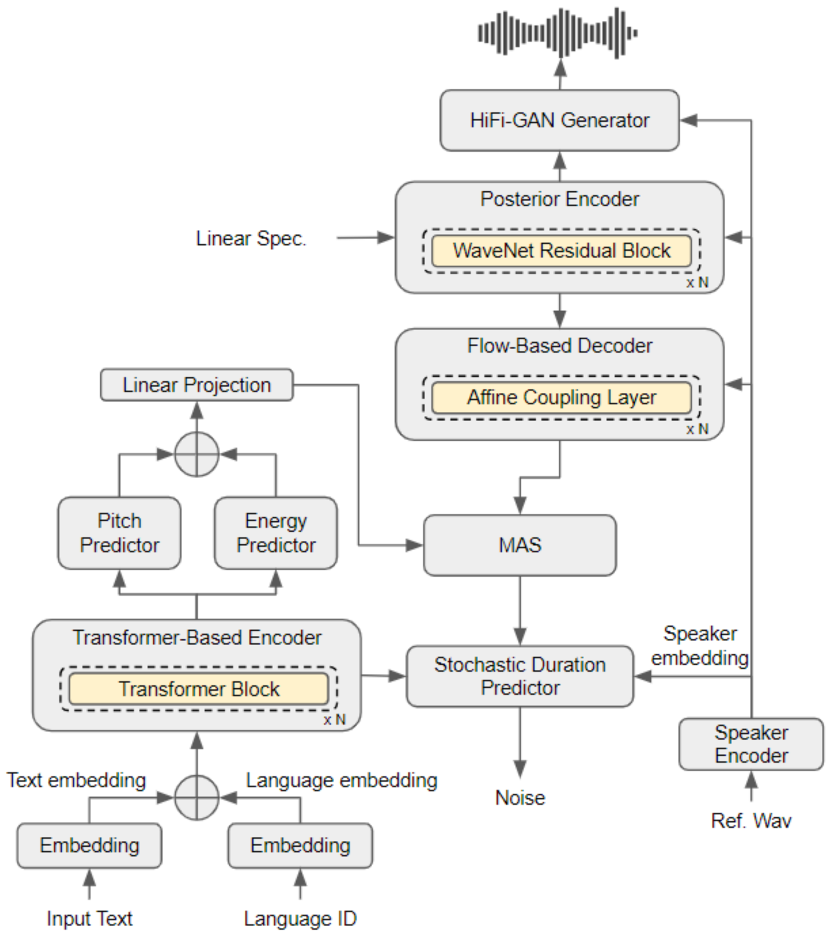


Figure 4.8: The architecture of YourTTS during training. Monotonic Alignment Search is used to learn the durations on the fly and is then used to train the Stochastic Duration Predictor.

During training as seen in Figure 4.8, the text encoder embeddings are combined with the pitch and energy embeddings. The resultant embeddings are used by the Monotonic Alignment Search (MAS) [34] to align it with speech embeddings given by the normalized flow. MAS in short searches for the best correspondence between the encoded speech and the characteristics of the spoken text, ensuring the generated speech aligns well with the intended content. The design of MAS enforces a monotonic relationship between the latent speech representation from the normalized flow and the text embedding from the transformer encoder. Monotonicity ensures a smooth and natural progression of speech features as the model traverses the latent space corresponding to the text. The monotonic alignments learnt are used to train the Stochastic Duration Predictor (SDP) [9] in parallel during training of the TTS. The SDP is a flow-based generative model that addresses the challenge of natural speech rhythm variation in TTS, also referred to as a one-to-many problem. It predicts each text token's duration by sampling it from random noise that is conditioned on the provided text input. SDP employs residual blocks with dilated and depth-separable convolutions for efficient parameterization. It utilizes neural spline flows within the coupling layers to achieve increased expressiveness in capturing token duration variations while maintaining parameter efficiency compared to traditional methods. Since the token duration is learnt within the system via MAS it does not require an external forced aligner to get the ground truth duration like in [7, 6]. Since the modifications made are generic, the training can be done using a neutral speech corpus eliminating the need for a high-quality curated dataset suitable for stress synthesis.

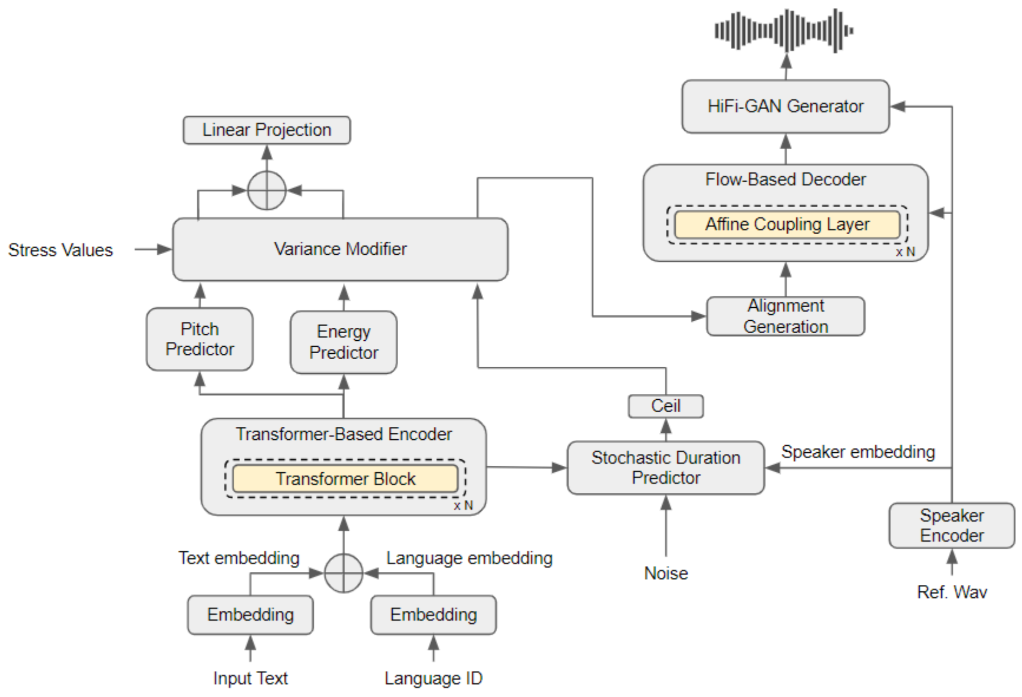


Figure 4.9: The architecture of YourTTS during inference. The Stochastic Duration Predictor is used for generating the duration for every text token.

Once the model is trained, the SDP replaces the MAS and the variance modifier is added to the architecture as seen in Figure 4.9. The SDP learnt to predict text token duration from sampled noise conditioned on the text input. The ground truth for this was learnt using MAS during training, but once the SDP is trained on these values MAS is no longer required. The variance modifier is responsible for manipulating the predicted variances based on the stress values provided in the form of stressed words and their corresponding scaling factors 3.5.2. These scaling factors determine how much the region corresponding to a stressed word needs to be modified to achieve the same stress as seen in the source speech. The variance modifier uses this information and scales the pitch and energy variances of all the stressed regions correspondingly. Once the predicted variances are modified, they are up-sampled using the duration predicted by the SDP by repeating the corresponding embeddings. The process of variance modifier is very similar to that of the PDE modifier explained in the previous section. The final upsampled text encoder outputs with variance embeddings are projected into the latent space through the inverted normalized flow. This space corresponds to a learnt space modelling a Variational Autoencoder (VAE) from a linear spectrogram to a waveform. These latent representations are finally converted to waveform using the HiFiGAN [36] decoder which is part of the VAE that was trained in parallel. Since the HiFiGAN vocoder is not external but was trained in parallel, the quality of the generated speech is superior. The overall architecture gives the ability to artificially induce stress in synthesized speech with high quality and flexibility along with zero-shot speaker capabilities and fast inference time.

4.5 Combined SSMT pipeline

In previous sections, the many components that go into making an SSMT pipeline work have been discussed along with the new proposed changes for incorporating stress in it. In this section, we discuss how the different components go together to give the final working system. For setting up the SSMT, components such as Automatic Speech Recognition (ASR), Machine Translation (MT) and Text-to-Speech (TTS) are required. We additionally add the stress detection model (with its post-processing module) and the MT word alignment model to give the SSMT the ability to preserve stress in the source speech and translate it into the generated target speech. The SSMT pipeline as seen in Figure 4.10 is designed for Indian English to Hindi to improve the quality of online education for students in India.

The ASR model converts the source spoken Indian English into English text. The model used here is WhisperX [46] which is based on OpenAI's Whisper [45] English large-v2 model. These models are explained in detail in Section 2.4.1.2 and Section 2.4.1.1 respectively. The ASR model, WhisperX gives both English transcription and English speech and text word-level forced alignment with high confidence scores.

The input English Speech is sent to the Stress Detection model as explained in Section 3.4.1, 3.4.2. This model gives frame-level stress prediction for the speech signal, where we get information regarding whether a particular frame of speech has stress in it or not. Then the post-processing module converts the frame-level prediction into word-level prediction using the word alignments obtained from the ASR model as explained in Section 3.5.1. Therefore we have information regarding whether a particular word in the source speech is stressed or not.

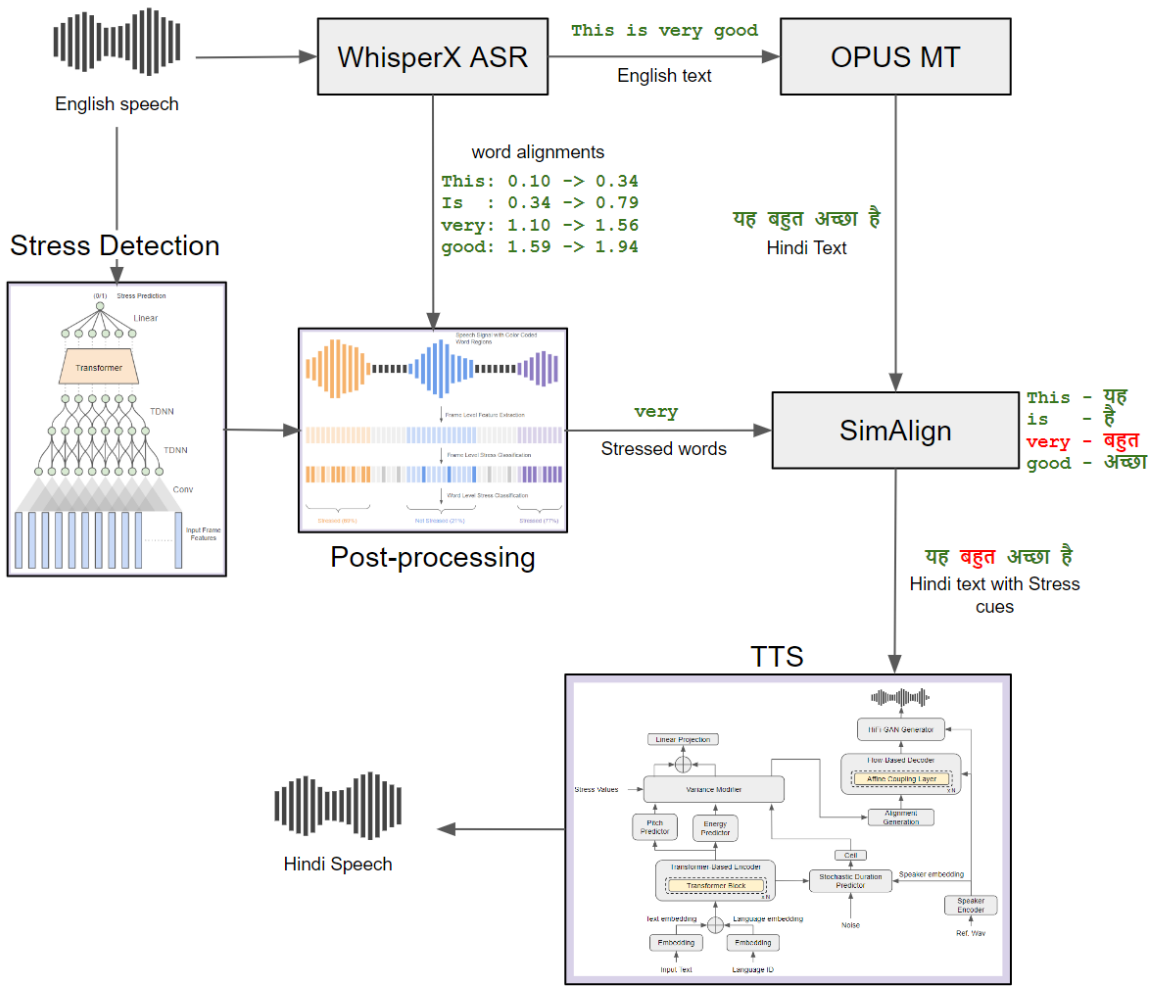


Figure 4.10: The final SSMT pipeline which is capable of transferring stress in source to target speech

The MT model converts English text into Hindi text. The model used here is the open-source Helsinki OPUS English-Hindi MT model [50]. This model is explained in detail in Section 2.4.2.1. Once we get the Hindi transcript, we need to figure out which Hindi words are stressed. Here we make a critical assumption: **if a particular word has been stressed in the source speech, then its corresponding translated word in the target speech is also stressed and will be at the same scale**. This assumption is necessary as it allows the system to be feasible. Even with the assumption, the SSMT pipeline gives decent quality, where the human evaluator is not able to notice the difference. Therefore we use the MT word aligner Simalign [51] to map the translated Hindi words to their corresponding English words. The working of Simalign is explained in Section 2.4.2.2, where this model gives a many-to-many mapping between English and Hindi. This means that a single Hindi word could correspond to multiple English words or multiple Hindi words could correspond to a single English word or multiple Hindi words could correspond to multiple English words. Therefore this task is non-trivial, and we simply extend the assumption that whatever words in English were stressed, **all** the corresponding Hindi words are also stressed. We also forward the scaling factors from English-stressed words to their corresponding Hindi-stressed words based on the assumption we made. By combining the Hindi stressed words with their scaling factors, we get the stress cues information that is given to the TTS system.

The TTS model converts the Hindi text into Hindi speech by conditioning it on the stress cues to add stress to the speech. The above two discussed TTS models are used in this work. They are modified versions of FastPitch 4.3 and YourTTS 4.4 are used to allow to consider stress addition. To train the TTS models, we have used the latest version of the Hindi Male speaker's database from the ULCA Bhashini IndicTTS [29] which is explained in Section 2.3.1.3. The Hindi version of the models was used in the SSMT pipeline which takes both the stress cues and Hindi text and generates the proper stressed Hindi speech. We also trained these models on English datasets [25, 26] as explained in Sections 2.3.1.1 and 2.3.1.2 respectively, for benchmarking the TTS to get an understanding of the relative quality.

4.6 Experimental Setup

The setup includes the TTS models along with the SSMT pipeline. For setting up the SSMT, other components such as Automatic Speech Recognition (ASR), Machine Translation (MT) and MT word aligner are required. For this, WhisperX [46] ASR model as explained in Sections 2.4.1.2 and 2.4.1.1 was used to convert the source speech into text. WhisperX is built on OpenAI's Whisper [45] English large-v2 model but it gives both transcription and word-level forced alignment. The source English speech is then sent to the stress detection model to identify the stressed regions on the frame level, and post-processing is done by comparing these regions with the word alignments to get word-level stress prediction. The scaling factors are calculated for the word-level regions in the stress detection post-processing. Then the English transcription is converted to Hindi using the open source Helsinki OPUS [50] MT model which is explained in Section 2.4.2.1. The English word-level stress prediction is converted to Hindi word-level using the MT word aligner SimAlign [51] which is explained in Section 2.4.2.2. The aligner also converts the scaling factors from the source language to the target language. Finally, the Hindi word-level stress predictions along with their scaling factors constitute the stress values which are given as input to the TTS system. The TTS models were trained using an NVIDIA A100 80GB with batch size 128. For training the TTS system, AdamW optimizer [73] with betas 0.8 and 0.99, weight decay of 0.01 and an initial learning rate of 0.0002 with exponential decay was used. Mixed precision was also used, and no blank tokens were kept between the text tokens. The models were trained up to 500k iterations which took around 3 days on the GPU.

4.7 Results and Discussion

In this section, we explain the different evaluations done, the metrics chosen and the observations made from these evaluations for different models. We perform evaluations on both the TTS and SSMT models to understand how the changes made in the TTS affect its quality and how the overall SSMT performs with the new additions.

For evaluating TTS, we generally use subjective metrics. This is because the TTS is rated based on the naturalness and quality of the speech generated. The naturalness of speech is a subjective idea that denotes how natural humans perceive the generated speech. Therefore we use Mean-Opinion-Score (MOS) [38], a human-evaluated metric used for evaluating TTS systems as explained in Section 2.3.3. The ratings have been crowdsourced from 15 raters, where each rater gives a score between 1 and 5 for randomly selected samples, where 1 denotes bad quality and 5 denotes good quality. We denote the FastPitch-based stress TTS as FTTS, and the YourTTS-based stress TTS as YTTS. We evaluate the quality of the TTS using three standard datasets: LJSpeech [25], VCTK [26], IndicTTS [29]. LJSpeech and VCTK are English datasets as explained in Sections 2.3.1.1, 2.3.1.2, and we use the Hindi male dataset from IndicTTS corpus which is explained in Section 2.3.1.3. Each model has two versions: only the Pitch variance predictor (P), and both the Pitch and Energy variance predictors (PE). For both models, we conduct two experiments: without variance modifier (WOVM), and with variance modifier (VM). When using a variance modifier, we also select a word and give it stress manually. This is done to see how much distortion is produced in the speech if any. From these experiments, we get the base quality of the TTS and also understand how the quality changes when stress is added to the speech.

Model	**MOS**		
	LJSpeech	**VCTK**	**IndicTTS (Hindi)**
FTTS (P-WOVM)	4.12 ± 0.04	4.09 ± 0.06	4.04 ± 0.05
FTTS (P-VM)	3.97 ± 0.05	3.88 ± 0.04	3.76 ± 0.06
FTTS (PE-WOVM)	4.22 ± 0.04	4.19 ± 0.05	4.13 ± 0.06
FTTS (PE-VM)	3.90 ± 0.04	3.92 ± 0.06	3.71 ± 0.07
YTTS (P-WOVM)	4.29 ± 0.05	4.24 ± 0.07	4.15 ± 0.07
YTTS (P-VM)	4.12 ± 0.07	4.08 ± 0.06	3.87 ± 0.05
YTTS (PE-WOVM)	**4.42 ± 0.05**	**4.38 ± 0.05**	**4.29 ± 0.06**
YTTS (PE-VM)	4.08 ± 0.06	4.08 ± 0.03	4.02 ± 0.06

Table 4.1: Comparison between FastPitch and YourTTS models in different setups.

These are the observations made from Table 4.1: The base TTS models without a variance modifier (WO-VM) perform better than their counterparts, with a variance modifier (VM). This indicates that the downside of adding stress using a variance modifier is that the quality of the generated speech drops a bit. This is due to the distortion that might have occurred due to the manipulation of the variance contours. We can also observe that when using both the variances (PE), the quality increases. This indicates that having multiple variance predictors captures more nuances in speech, and thus helps in generating better quality speech. However, it has the opposite effect when using it with a variance modifier (PE-VM), where the models underperform when compared to having one variance with a variance modifier (P-VM). This indicates that the naturalness goes down as we are manipulating multiple variance contours instead of one, thus increasing the distortion. The main takeaway from the table is that YourTTS (YTTS) consistently performs better than FastPitch (FTTS), which validates the results obtained in [24]. This implies that YourTTS with its flow modelling is able to perform better than Transformers-based FastPitch in the problem of speech synthesis as well as **stress-based speech synthesis**.

Similar to TTS, we use a subjective metric to evaluate the SSMT pipeline as both give speech as their output. The automated evaluation for SSMT is non-trivial, and hence a subjective evaluation survey was conducted, where 15 participants, fluent in both English and Hindi were asked to give performance metrics based on different factors. Each participant is given a few randomly selected samples, each sample containing the outputs for all the SSMT components. It consists of the original English speech, the English ASR text, the translated Hindi text, the generated Hindi speech with variance modifier and the generated Hindi speech without variance modifier. We define two performance metrics for the task: MOS_n which denotes how natural the generated speech is, and MOS_s which denotes how well the stress was incorporated. The rater has to use all the intermediate outputs as a reference along with the input and output to give these MOS scores. The rater uses the original speech and synthesized speech without stress as a reference to gauge how well the stress has been added to the synthesized speech with stress. This is used to calculate the MOS_s score, where 0 means that no stress was transferred and 5 means that all the stress regions were transferred. The SSMT pipeline using the machine-learning-based stress detection with FastPitch-based TTS is denoted as FSSMT, and the pipeline using StressNet with YourTTS-based TTS is denoted as YSSMT. Each TTS model has two versions: only the Pitch variance predictor (P), and both the Pitch and Energy variance predictors (PE). These scores are subjective in nature, hence they represent an approximation of the performance or quality.

These are the observations made from Table 4.2: in both naturalness and stress transfer, YSSMT has better MOS scores. This denotes that not only is YourTTS better than FastPitch in terms of quality of speech, but it can also generalize and add stress to speech with more impact. The increase in MOS_s can also be attributed to improvement in stress detection using StressNet, which outperforms machine-learning-based stress detection models as seen in Section 3.7.

Model	MOS_n	MOS_s
FSSMT (P)	3.75 ± 0.05	3.45 ± 0.04
FSSMT (PE)	3.95 ± 0.03	3.55 ± 0.05
YSSMT (P)	3.91 ± 0.04	3.52 ± 0.03
YSSMT (PE)	**4.12 ± 0.04**	**3.95 ± 0.04**

Table 4.2: Comparison between FastPitch and YourTTS-based SSMT models in different setups.

4.8 Limitations

Despite the models generating speech with stress that is perceivable, it is usually unclear without a reference. If the samples without the variance modifier and with it are shown together, it is quite easy to spot the improvements. This implies that efforts need to go into making such additions more noticeable. Forcibly increasing the scaling factors does not seem to help. During training, the TTS models have learnt to work for a certain range of variance contours, so when we drastically change these values, the TTS models have to deal with unfamiliar values and usually end up adding noise to the speech or distorting it. Hence, we need to work towards making the TTS models more sensitive to such changes and having better speech quality. This issue can also be solved by designing a TTS that is meant to handle stress. Therefore such systems could be more effective when adding stress.

4.9 Summary

In this chapter, a stress-aware TTS that is capable of conditioning the generated speech on stress cues is discussed. We show how existing models can be modified to achieve this task, which has the additional benefit of training using neutral TTS data without expensive stress-based TTS data. The feature extraction for training the TTS is also discussed. We show how the different components of ASR, MT, Aligner, Stress detection model and TTS go together to create a Speech-to-Speech Machine Translation (SSMT) System that is capable of preserving stress in the source speech and transferring it into the target speech. We show that using the modified version of YourTTS is better in quality when compared to the modified version of FastPitch. We also show that despite the slight decrease in the quality of speech generated, the stress addition can be perceived in the generated speech from the SSMT.

Chapter 5

Conclusion and Future Scope

In this thesis, we have captured the essence of stress in the source speech and transferred the emphasis to the target speech in Indian English to Hindi Speech-to-Speech Machine Translation System. The significant contributions of this thesis are as follows: At the dataset level, a stress-annotated speech corpus in the Indian English education domain is curated for research purposes. At the feature level, acoustic features extracted from the dataset are investigated for stress detection in Indian English-spoken speech. At the model level, models for stress detection and speech synthesis models with stress-aware conditioning are proposed for enabling stress transfer in SSMT.

5.1 Conclusion

The conclusions drawn from the thesis are as follows:

- Pitch, Energy, Mel Frequency Cepstral Coefficients and Shifted Delta Coefficients play a vital role in detecting stress in speech.
- Label Propagation machine-learning algorithm outperforms other machine-learning models in stress detection.
- Deep-learning model StressNet outperforms all other models in the task of stress detection due to its complex structure and better temporal reach.
- Models using all four features and having a larger window size for the input feature stack are shown to have a better performance when detecting stress.
- When converting word-level predictions into frame-level predictions, the performance of detection improves. This indicates that this operation helps mitigate noise predictions.
- Existing TTS models can be modified to achieve stress incorporation in speech. This gives the additional benefit of just requiring available neutral TTS data without any stress-based TTS data which are not available.

- The modified version of YourTTS is better in quality when compared to the modified version of FastPitch when generating speech by incorporating stress.
- Stress detection model combined with the modified TTS architectures and other components like ASR, MT, and MT aligner make it possible to build an SSMT that is capable of preserving stress in the source speech and transferring it into the target speech.

5.2 Future Scope

Some directions for future work are as follows:

- The performance of the stress detection models can be improved for a general scenario by training it using more generic stress data.
- The performance of the stress detection models has to be studied in the presence of noise, background speaker, background music and other conditions.
- Studies should be done on more advanced features specifically engineered for stress detection.
- Further work needs to be done to improve the quality of the TTS. Other TTS architectures should be explored for stress incorporation.
- Designing new TTS architectures specifically to incorporate prosody or stress needs to be studied in the context of SSMT.
- Works need to be done to replace the supervised scaling factors with an unsupervised method for stress tokens when generating speech for better quality.

Related Publications

1. **Sai Akarsh C**, Vamshiraghusimha Narasinga, Anindita Mondal and Anil Vuppala. **Attempt Towards Stress Transfer in Speech-to-Speech Machine Translation**. Accepted as a full paper in the proceedings of the International Conference on Signal Processing and Communications 2024 (SPCOM 24).
2. **Sai Akarsh C**, Vamshiraghusimha Narasinga, Anindita Mondal, Priyanka Kommagouny and Anil Vuppala. **Improving Stress Transfer in Speech-to-Speech Machine Translation**. Under review as a full paper at Interspeech 2024.
3. **Sai Akarsh C**, Vamshiraghusimha Narasinga, and Anil Vuppala. Demo Application for **Stress Transfer in Speech-to-Speech Machine Translation**. Under review as a Show&Tell paper at Interspeech 2024.

Bibliography

[1] Wen-Chin Huang, Benjamin Peloquin, Justine Kao, Changhan Wang, Hongyu Gong, Elizabeth Salesky, Yossi Adi, Ann Lee, and Peng-Jen Chen. A holistic cascade system, benchmark, and human evaluation protocol for expressive speech-to-speech translation. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[2] PD Aguero, Jordi Adell, and Antonio Bonafonte. Prosody generation for speech-to-speech translation. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 1, pages I–I. IEEE, 2006.

[3] Yuxuan Wang, RJ Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, et al. Tacotron: Towards end-to-end speech synthesis. *arXiv preprint arXiv:1703.10135*, 2017.

[4] Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, et al. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4779–4783. IEEE, 2018.

[5] Adrian Łańcucki. Fastpitch: Parallel text-to-speech with pitch prediction. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6588–6592. IEEE, 2021.

[6] Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech: Fast, robust and controllable text to speech. *Advances in neural information processing systems*, 32, 2019.

[7] Yi Ren, Chenxu Hu, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech 2: Fast and high-quality end-to-end text to speech. *arXiv preprint arXiv:2006.04558*, 2020.

[8] Sercan Ö Arık, Mike Chrzanowski, Adam Coates, Gregory Diamos, Andrew Gibiansky, Yongguo Kang, Xian Li, John Miller, Andrew Ng, Jonathan Raiman, et al. Deep voice: Real-time neural text-to-speech. In *International conference on machine learning*, pages 195–204. PMLR, 2017.

[9] Jaehyeon Kim, Jungil Kong, and Juhee Son. Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech. In *International Conference on Machine Learning*, pages 5530–5540. PMLR, 2021.

[10] Vineel Pratap, Andros Tjandra, Bowen Shi, Paden Tomasello, Arun Babu, Sayani Kundu, Ali Elkahky, Zhaoheng Ni, Apoorv Vyas, Maryam Fazel-Zarandi, Alexei Baevski, Yossi Adi, Xiaohui Zhang, Wei-Ning Hsu, Alexis Conneau, and Michael Auli. Scaling speech technology to 1,000+ languages. *arXiv*, 2023.

[11] Yuxuan Wang, Daisy Stanton, Yu Zhang, RJ-Skerry Ryan, Eric Battenberg, Joel Shor, Ying Xiao, Ye Jia, Fei Ren, and Rif A Saurous. Style tokens: Unsupervised style modeling, control and transfer in end-to-end speech synthesis. In *International conference on machine learning*, pages 5180–5189. PMLR, 2018.

[12] Zhao-Ci Liu, Zhen-Hua Ling, Ya-Jun Hu, Jia Pan, Yun-Di Wu, and Jin-Wei Wang. Speech synthesis with self-supervisedly learnt prosodic representations.

[13] Yiwei Guo, Chenpeng Du, Xie Chen, and Kai Yu. Emodiff: Intensity controllable emotional text-to-speech with soft-label guidance. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[14] Haobin Tang, Xulong Zhang, Jianzong Wang, Ning Cheng, and Jing Xiao. Emomix: Emotion mixing via diffusion models for emotional speech synthesis. *arXiv preprint arXiv:2306.00648*, 2023.

[15] Chiranjeevi Yarra, Om D Deshmukh, and Prasanta Kumar Ghosh. Automatic detection of syllable stress using sonority based prominence features for pronunciation evaluation. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5845–5849. IEEE, 2017.

[16] ES Atwell, PA Howarth, and DC Souter. The isle corpus: Italian and german spoken learner's english. *ICAME Journal: International Computer Archive of Modern and Medieval English Journal*, 27:5–18, 2003.

[17] Jhansi Mallela, Prasanth Sai Boyina, and Chiranjeevi Yarra. A comparison of learned representations with jointly optimized vae and dnn for syllable stress detection. In *International Conference on Speech and Computer*, pages 322–334. Springer, 2023.

[18] Chiranjeevi Yarra, Manoj Kumar Ramanathi, and Prasanta Kumar Ghosh. Comparison of automatic syllable stress detection quality with time-aligned boundaries and context dependencies. In *SLaTE*, pages 79–83, 2019.

[19] Dagen Wang and Shrikanth Narayanan. An unsupervised quantitative measure for word prominence in spontaneous speech. In *Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, volume 1, pages I–377. IEEE, 2005.

[20] John J Godfrey, Edward C Holliman, and Jane McDaniel. Switchboard: Telephone speech corpus for research and development. In *Acoustics, speech, and signal processing, ieee international conference on*, volume 1, pages 517–520. IEEE Computer Society, 1992.

[21] Dagen Wang and Shrikanth Narayanan. An acoustic measure for word prominence in spontaneous speech. *IEEE transactions on audio, speech, and language processing*, 15(2):690–701, 2007.

[22] Taniya Mishra, Vivek Kumar Rangarajan Sridhar, and Alistair Conkie. Word prominence detection using robust yet simple prosodic features. In *Interspeech*, pages 1864–1867, 2012.

[23] Krishna Gurugubelli, KNRK Raju Alluri, Anil Kumar Vuppala, et al. Differenced prosody features from normal and stressed regions for emotion recognition. In *2018 5th International Conference on Signal Processing and Integrated Networks (SPIN)*, pages 821–825. IEEE, 2018.

[24] Edresson Casanova, Julian Weber, Christopher D Shulby, Arnaldo Candido Junior, Eren Gölge, and Moacir A Ponti. Yourtts: Towards zero-shot multi-speaker tts and zero-shot voice conversion for everyone. In *International Conference on Machine Learning*, pages 2709–2720. PMLR, 2022.

[25] Keith Ito and Linda Johnson. The lj speech dataset. 2017.

[26] Christophe Veaux, Junichi Yamagishi, Kirsten MacDonald, et al. Superseded-cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit. 2016.

[27] Thorsten Müller and Dominik Kreutz. Thorsten-voice dataset 2021.02. 2021. Please use it to make the world a better place for whole humankind.

[28] Kishore Prahallad, Anandaswarup Vadapalli, Naresh Elluru, Gautam Mantena, Bhargav Pulugundla, Peri Bhaskararao, Hema A Murthy, Simon King, Vasilis Karaiskos, and Alan W Black. The blizzard challenge 2013–indian language task. In *Blizzard challenge workshop*, volume 2013, 2013.

[29] Arun Baby, Anju Leela Thomas, NL Nishanthi, TTS Consortium, et al. Resources for indian languages. In *Proceedings of Text, Speech and Dialogue*, 2016.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[31] Ryan Prenger, Rafael Valle, and Bryan Catanzaro. Waveglow: A flow-based generative network for speech synthesis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3617–3621. IEEE, 2019.

[32] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[33] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[34] Jaehyeon Kim, Sungwon Kim, Jungil Kong, and Sungroh Yoon. Glow-tts: A generative flow for text-to-speech via monotonic alignment search. *Advances in Neural Information Processing Systems*, 33:8067–8077, 2020.

[35] Colin Raffel, Minh-Thang Luong, Peter J Liu, Ron J Weiss, and Douglas Eck. Online and linear-time attention by enforcing monotonic alignments. In *International conference on machine learning*, pages 2837–2846. PMLR, 2017.

[36] Jungil Kong, Jaehyeon Kim, and Jaekyoung Bae. Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. *Advances in neural information processing systems*, 33:17022–17033, 2020.

[37] Robert Kubichek. Mel-cepstral distance measure for objective speech quality assessment. In *Proceedings of IEEE pacific rim conference on communications computers and signal processing*, volume 1, pages 125–128. IEEE, 1993.

[38] Robert C Streijl, Stefan Winkler, and David S Hands. Mean opinion score (mos) revisited: methods and applications, limitations and alternatives. *Multimedia Systems*, 22(2):213–227, 2016.

[39] Michael Schoeffler, Sarah Bartoschek, Fabian-Robert Stöter, Marlene Roess, Susanne Westphal, Bernd Edler, and Jürgen Herre. webmushra—a comprehensive framework for web-based listening tests. *Journal of Open Research Software*, 6(1):8, 2018.

[40] Quoc Truong Do, Tomoki Toda, Graham Neubig, Sakriani Sakti, and Satoshi Nakamura. Preserving word-level emphasis in speech-to-speech translation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(3):544–556, 2016.

[41] Quoc Truong Do, Sakriani Sakti, and Satoshi Nakamura. Sequence-to-sequence models for emphasis speech translation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(10):1873–1883, 2018.

[42] Quoc Truong Do, Sakriani Sakti, Graham Neubig, and Satoshi Nakamura. Transferring emphasis in speech translation using hard-attentional neural network models. In *Interspeech*, pages 2533–2537, 2016.

[43] Takatomo Kano, Sakriani Sakti, Shinnosuke Takamichi, Graham Neubig, Tomoki Toda, and Satoshi Nakamura. A method for translation of paralinguistic information. In *Proceedings of the 9th International Workshop on Spoken Language Translation: Papers*, pages 158–163, 2012.

[44] Shivam Mhaskar, Vineet Bhat, Akshay Batheja, Sourabh Deoghare, Paramveer Choudhary, and Pushpak Bhattacharyya. Vakta-setu: A speech-to-speech machine translation service in select indic languages. *arXiv preprint arXiv:2305.12518*, 2023.

[45] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. In *International Conference on Machine Learning*, pages 28492–28518. PMLR, 2023.

[46] Max Bain, Jaesung Huh, Tengda Han, and Andrew Zisserman. Whisperx: Time-accurate speech transcription of long-form audio. *arXiv preprint arXiv:2303.00747*, 2023.

[47] Hervé Bredin. pyannote.audio 2.1 speaker diarization pipeline: principle, benchmark, and recipe. In *Proc. INTERSPEECH 2023*, 2023.

[48] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460, 2020.

[49] Luciano Floridi and Massimo Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:681–694, 2020.

[50] Jörg Tiedemann and Santhosh Thottingal. Opus-mt–building open translation services for the world. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 479–480, 2020.

[51] Masoud Jalili Sabet, Philipp Dufter, François Yvon, and Hinrich Schütze. Simalign: High quality word alignments without parallel training data using static and contextualized embeddings. *arXiv preprint arXiv:2004.08728*, 2020.

[52] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[53] Sparsh Garg, Utkarsh Mehrotra, Gurugubelli Krishna, and Anil Kumar Vuppala. Towards a database for detection of multiple speech disfluencies in indian english. In *2021 National Conference on Communications (NCC)*, pages 1–6. IEEE, 2021.

[54] NK Sheeja. Open educational resources in india: A study of nptel and its usage. *Library Herald*, 56(1):122–129, 2018.

[55] Maxim Tkachenko, Mikhail Malyuk, Andrey Holmanyuk, and Nikolai Liubimov. Label Studio: Data labeling software, 2020-2022. Open source software available from https://github.com/heartexlabs/label-studio.

[56] Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

[57] Md Sahidullah and Goutam Saha. On the use of distributed dct in speaker identification. In *2009 Annual IEEE India Conference*, pages 1–4. IEEE, 2009.

[58] Pedro A Torres-Carrasquillo, Elliot Singer, Mary A Kohler, Richard J Greene, Douglas A Reynolds, and John R Deller Jr. Approaches to language identification using gaussian mixture models and shifted delta cepstral features. In *Interspeech*, volume 2, pages 89–92. Citeseer, 2002.

[59] Kshitiz Kumar, Chanwoo Kim, and Richard M Stern. Delta-spectral cepstral coefficients for robust speech recognition. In *2011 IEEE International conference on acoustics, speech and signal processing (ICASSP)*, pages 4784–4787. IEEE, 2011.

[60] Wei-Qiang Zhang, Liang He, Yan Deng, Jia Liu, and Michael T Johnson. Time–frequency cepstral features and heteroscedastic linear discriminant analysis for language recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(2):266–276, 2010.

[61] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[62] Dengyong Zhou, Olivier Bousquet, Thomas Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. *Advances in neural information processing systems*, 16, 2003.

[63] Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. A time delay neural network architecture for efficient modeling of long temporal contexts. In *Interspeech*, pages 3214–3218, 2015.

[64] Shakeel A Sheikh, Md Sahidullah, Fabrice Hirsch, and Slim Ouni. Stutternet: Stuttering detection using time delay neural network. In *2021 29th European Signal Processing Conference (EUSIPCO)*, pages 426–430. IEEE, 2021.

[65] Svetlana Tchistiakova. Time delay neural network. *Time Delay Neural Network Blog post*, 2019.

[66] E Oran Brigham. *The fast Fourier transform and its applications*. Prentice-Hall, Inc., 1988.

[67] HJ Landau. Sampling, data transmission, and the nyquist rate. *Proceedings of the IEEE*, 55(10):1701–1706, 1967.

[68] Jon H Appleton and Ronald Perera. The development and practice of electronic music. *(No Title)*, 1975.

[69] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *SciPy*, pages 18–24, 2015.

[70] Matthias Mauch and Simon Dixon. pyin: A fundamental frequency estimator using probabilistic threshold distributions. In *2014 ieee international conference on acoustics, speech and signal processing (icassp)*, pages 659–663. IEEE, 2014.

[71] Alain De Cheveigné and Hideki Kawahara. Yin, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, 111(4):1917–1930, 2002.

[72] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[73] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.