On Zero-Shot Multi-Speaker Text-to-Speech Using Deep Learning

Pradnya Kandarkar

A Thesis
in
The Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montreal, Quebec, Canada

July 2023

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By : Pradnya Kandarkar
Entitled : On Zero-Shot Multi-Speaker Text-to-Speech Using Deep Learning

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

Master of Computer Science

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final Examining Committee:

_____ Chair
Dr. Ching Yee Suen

_____ Examiner
Dr. Eugene Belilovsky

_____ Thesis Supervisor
Dr. Mirco Ravanelli

Approved by _____
Dr. Leila Kosseim, Graduate Program Director

_____ 2023

_____
Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

On Zero-Shot Multi-Speaker Text-to-Speech Using Deep Learning

Pradnya Kandarkar

This thesis explores various aspects of zero-shot multi-speaker text-to-speech (TTS) synthesis using deep learning to create an effective system. A deep learning model for zero-shot multi-speaker TTS uses text and speaker identity as input to generate the respective output speech without fine-tuning for speakers not seen during training. The experiments consider a system with three main components: a speaker encoder network, a mel-spectrogram prediction network, and a vocoder network. A speaker encoder network captures the speaker identity in a fixed-sized speaker embedding. This speaker embedding is injected into a mel-spectrogram prediction network at one or more locations to generate a mel-spectrogram conditioned on the text and the speaker embedding. Finally, a vocoder network converts the mel-spectrogram into a waveform. All three components are trained separately. The speech synthesis aspects explored in the experiments include the speaker embedding injection method, speaker encoder network, speaker embedding injection location, and mel-spectrogram prediction network for the TTS system. The FiLM method from the visual reasoning field is adapted for the first time to inject speaker embeddings into the TTS workflow and compared against traditional methods. The significance of speaker embeddings is highlighted by comparing two well-established speaker embedding models. New combinations of speaker embedding injection locations are explored for two mel-spectrogram prediction networks. The best-performing model generates speech with naturalness ranging from fair to good, exhibits more than moderate speaker similarity, and shows potential for improvement. Additionally, the zero-shot multi-speaker TTS system is enhanced to generate fictitious voices.

Acknowledgments

I want to take a moment to express my sincere gratitude to everyone who has contributed to this work.

Above all, I extend my profound appreciation to my thesis supervisor, Dr. Mirco Ravanelli, for giving me the opportunity to study this captivating field. Throughout my academic journey, Dr. Ravanelli has been incredibly generous with his time, helping me grasp new concepts, engaging in critical analysis of ideas, and shaping the trajectory of this research. His expertise, guidance, and drive for innovation continue to motivate and inspire me.

I am also thankful to Dr. Cem Subakan for sharing his extensive knowledge and providing invaluable insights that helped improve the quality of this work at various stages.

I want to acknowledge the support and encouragement of my friends and fellow lab members, whose stimulating discussions greatly contributed to the development of this work.

I thank my family for being my source of strength.

Lastly, I wish to express my gratitude to all the participants in the user study conducted for this work, for contributing their time and effort to this research.

Thank you all!

Contents

List of Figures

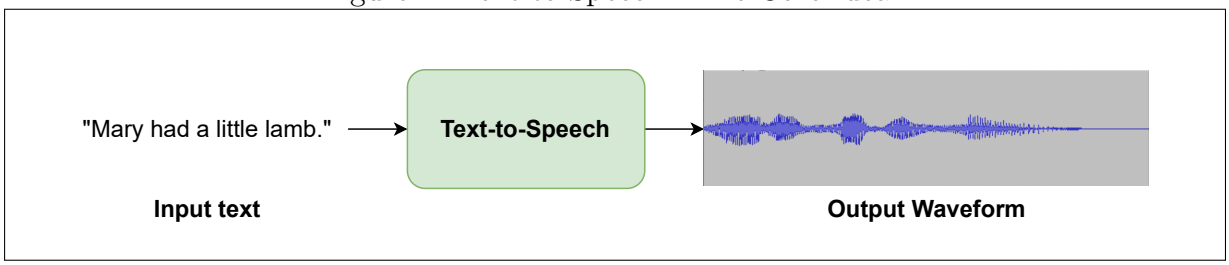
List of Tables

1 Introduction

1.1 What is TTS?

Text-to-Speech (TTS) technology takes written text as input and produces its corresponding spoken speech in the output, as shown in Figure 1.

Figure 1: Text-to-Speech - The Core Idea



TTS systems aim to imitate the way humans speak. The input text is processed using one or more models or algorithms to produce the output speech. An ideal TTS system should generate natural-sounding and expressive speech with a clear voice and pronunciation.

1.2 Applications

TTS technology has a multitude of potential applications, including:

1. Accessibility: Written content can be made accessible for individuals with visual impairments using TTS
2. Navigation systems: TTS is useful in providing directions to users while driving
3. Digital assistants: TTS can be used to equip digital assistants with a voice as they interact with and assist users in day-to-day tasks through spoken responses
4. Audiobooks: Expressive spoken versions of books or other written text can be created using TTS
5. Enhancing media production: A TTS system that can produce or adapt to different voices can be used to provide unique voices to characters in animated media, audiobooks, video games
6. Data augmentation for speech processing tasks: A multi-speaker TTS can be effective in data augmentation for speech processing tasks with low resources [3, 4]
7. Educational applications: Multilingual TTS technology with the ability to switch languages can be used for educational purposes, e.g., learning new languages

1.3 Historical Notes

Decades of research involving various approaches have contributed to the state-of-the-art TTS technology [5, 6].

1.3.1 Concatenative Synthesis (1980s - Early 2000s)

In earlier work, researchers explored variations of concatenative synthesis with unit selection [5, 7]. This approach relied on maintaining a database of small units of pre-recorded speech segments and combining them to create the desired waveform. The challenges of producing natural-sounding speech with this approach involve selecting the most appropriate units to form the final waveform and the boundary artifacts introduced when combining different speech segments. Additionally, it is limited by the available pre-recorded units in the dataset. Using pre-recorded samples implies predefined voices, styles, expressions, etc. Covering a broader range of these factors requires more memory, and creating a dataset that encompasses all the variations of these factors is not practical.

1.3.2 Statistical Parametric Speech Synthesis (2000s - 2010s)

Eventually, statistical parametric speech synthesis became the prevalent approach [5, 8]. Directly learning a mapping between text and speech is a difficult. Thus, the TTS problem is decomposed into multiple stages. The overall speech synthesis process involves first generating a sequence of speech features and then using a vocoder to produce the final speech waveform based on the intermediate speech features. Earlier work in statistical parametric speech synthesis focused on using Hidden Markov Models (HMMs) to generate the sequence of speech features. This approach attempts to learn data properties by optimizing model parameter values. Storing model parameter values requires significantly less memory than creating a dataset to maximize the coverage of speech variations. Additionally, statistical parametric speech synthesis models can be modified to change the voice, language, etc. The advantages introduced by this approach are its lesser memory requirements compared to the concatenative approach and its adaptability. Despite the advantages, earlier statistical parametric models were unable to synthesize speech that matched the naturalness and clarity of human speech.

1.3.3 Neural TTS (Late 2010s - Present)

Over time, deep neural networks have been adapted in statistical parametric speech synthesis to map input text to intermediate speech features [6, 9, 10, 11] and produce the final waveform using these features [12, 13].

One such model, WaveNet [12], was introduced as a generative vocoder. It produced waveforms that completed with human speech by utilizing intermediate speech and linguistic features. However, creating the input features for WaveNet required substantial domain expertise.

Meanwhile, Tacotron [9] presented a sequence-to-sequence architecture designed to map input text to magnitude spectrograms as intermediate features using a single neural network. This approach simplified the speech synthesis workflow by replacing intermediate linguistic and acoustic features with a spectrogram produced using a single network. However, Tacotron employed the Griffin-Lim algorithm as the vocoder to convert the spectrograms to waveforms. The waveforms generated with this approach contained artifacts, and the audio quality did not match that achieved with neural network-based vocoders like WaveNet.

Later, Tacotron2 [6] proposed a completely neural approach to speech synthesis, using the strengths of Tacotron and WaveNet. Tacotron2 uses an autoregressive recurrent sequence-to-sequence architecture to map the input text to a mel-spectrogram serving as the intermediate feature representation. This mel-spectrogram is used as the input to a modified WaveNet, which generates the final waveform.

Nowadays, a two-stage neural TTS architecture is a standard approach for TTS. It contains a mel-spectrogram prediction network followed by a neural network-based vocoder. This entirely neural approach produces high-quality speech that sounds natural.

1.4 Recent Progress

Advances in deep learning continuously contribute to the evolution of the TTS field.

1.4.1 End-to-End TTS Models

TTS models based on transformers [14], such as FastSpeech2 [15] and TransformerTTS [16], emerged to produce the mel-spectrogram in the two-stage TTS systems. FastSpeech2 is a non-autoregressive model designed for faster generation of high-quality speech compared to its autoregressive counterparts. TransformerTTS is another non-autoregressive model developed to improve the training and inference efficiency and modeling long-term dependencies.

Diffusion is another generative deep learning technology adapted for speech synthesis. Diff-TTS [17] utilized denoising diffusion for speech synthesis. The denoising diffusion framework used in Diff-TTS consists of two processes: the forward diffusion process and the reverse process. In the forward process, Gaussian noise is gradually added to the mel-spectrogram until it is transformed into pure Gaussian noise. This process is predefined and is assumed to be independent of the input text. On the other hand, the reverse process is dependent on the input text. It starts from the Gaussian noise distribution and gradually removes noise to recover a mel-spectrogram corresponding to the given input text. The reverse process is the process learned during training.

Meanwhile, HiFi-GAN [18] is a vocoder created using a Generative Adversarial Network (GAN) for efficient speech synthesis with high quality. It contains a generator and a discriminator in an adversarial training setup. The generator is trained to convert mel-spectrograms into high-quality synthesized speech that is indistinguishable from real speech. The discriminator is trained to distinguish between real and synthesized speech. After training, the generator is used for speech synthesis while the discriminator is discarded.

Conditional Variational Autoencoder with adversarial learning was used to combine the two stages of the TTS pipeline and create a single model that maps input text directly to the output waveform without using an intermediate feature representation, such as mel-spectrogram [19]. The authors of [19] named the approach Variational Inference with adversarial learning for end-to-end Text-to-Speech (VITS).

1.4.2 Further Advancements in TTS

Although the above advances can be used to generate high-quality speech, they can be extended with additional input to explore other aspects of TTS to enhance the capabilities and usefulness of TTS technology.

Effectively conveying emotions in a synthesized speech to make it more expressive has been a topic of interest for many decades. EmoDiff [20] is a recent diffusion-based TTS model that uses a soft label guiding technique for intensity controllable emotional TTS.

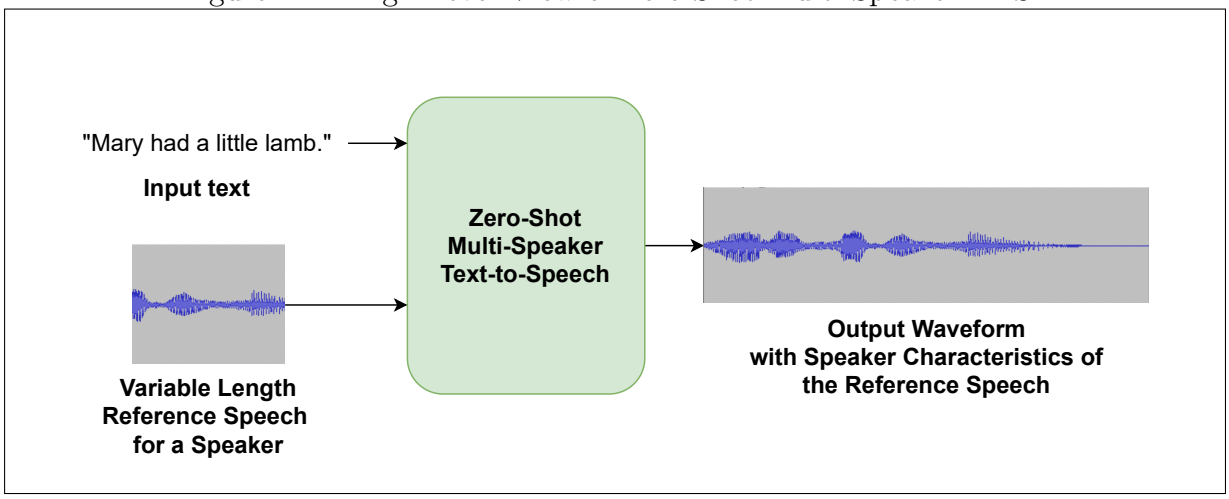
Multilingual TTS is another area being explored with the possibility of code-switching, i.e., changing the language of synthesized speech for some parts while maintaining the same voice [21]. This can enable using TTS technology to assist people in their preferred language or even aid them in learning a language.

There has also been a steady interest in multi-speaker TTS, which has promising versatile applications such as media production and data augmentation for speech-based tasks. Earlier multi-speaker TTS technology was limited to generating voices with the speaker identities available during training [22]. Zero-shot multi-speaker TTS, which is the focus of this work, further enhances the capabilities of TTS by allowing speech synthesis with the voices of unseen speakers without the need for specific training data for each individual. YourTTS [2] is a notable model that extends the VITS [19] model and combines multilingual and zero-shot multi-speaker TTS.

1.4.3 Towards Zero-Shot Multi-Speaker TTS

A zero-shot multi-speaker TTS system can use the speaker identity captured from a few seconds of audio to generate speech with the voice characteristics of that speaker for any given text [2, 23, 24, 25]. Figure 2 illustrates a high-level view of zero-shot multi-speaker TTS.

Figure 2: A High-Level View of Zero-Shot Multi-Speaker TTS



An effective zero-shot multi-speaker TTS system is advantageous in low-resource setups because it does not require fine-tuning for speakers not seen during training.

The text-to-speech synthesis field is making gradual progress towards effective zero-shot multi-speaker TTS systems. A common approach is to adapt a single-speaker TTS system to add speaker encoding capabilities using an external speaker embedding model [2, 23, 24, 25].

There are a few considerations when adding speaker encoding capabilities to TTS that can influence the outcome, including good speaker identity representation, the method used to incorporate speaker identity, and the location(s) where speaker identity information is added to the TTS model.

Adequately capturing speaker identity is important for zero-shot multi-speaker TTS. Various deep neural network-based speaker embedding models have been studied so far. Cooper et al. [24] studied the effect of the time-delay neural network [26] (TDNN)-based X-Vector [27] and Learnable Dictionary Encoding (LDE)-based speaker embeddings on the speaker similarity and naturalness of synthesized speech using an adapted version of Tacotron [9]. Xue et al. [25] proposed using ECAPA-TDNN [1], a model introducing enhancements to and outperforming other TDNN-based models for speaker verification on the VoxCeleb data. They used a pretrained ECAPA-TDNN model to extract speaker embeddings and FastSpeech2 to synthesize mel-spectrograms conditioned on the input text and speaker embeddings.

Along with obtaining good speaker representation using embeddings, how they are utilized is also important. A zero-shot multi-speaker TTS system can have speaker embeddings injected at one or more locations. Cooper et al. [24] try combinations of concatenating the speaker embeddings at different stages of an adapted Tacotron model. The best configuration observed in their experiments is concatenating the speaker embedding with the Tacotron encoder output and decoder prenet. Xue et al. [25] add speaker embeddings to the encoder output of the FastSpeech2 model. They do not test injecting speaker embeddings in multiple locations. YourTTS [2] is a single-stage zero-shot multi-speaker TTS model with speaker embeddings added in numerous places of the model. However, they do not provide an ablation study to show the contribution of injecting speaker embeddings in more than one place. It is worth noting that this criterion may be model-specific. Jia et al. [23] mention that passing speaker embeddings to the attention layer of Tacotron2 was sufficient to achieve convergence for the multi-speaker model. They do not report results with varying combinations of speaker embedding injection locations.

How speaker embeddings are injected into the TTS workflow can also affect the outcome. Concatenation and addition (or averaging) are traditional approaches when conditioning a TTS system on the input text and speaker embeddings. There is room to explore more effective speaker embedding injection methods. In recent developments, Yoon et al. [28] propose a sophisticated Speaker-Conditional Convolutional Neural Network (SC-CNN) which predicts convolution kernels from speaker embeddings and uses them to apply 1D convolutions to phoneme sequences for speaker conditioning. Meanwhile, Feature-wise Linear Modulation (FiLM) [29] is a general-purpose conditioning method for neural networks introduced in the context of visual reasoning. It can be adapted for zero-shot multi-speaker TTS.

Vocoder is the last piece of the TTS pipeline, relevant for two-stage systems. It is possible for zero-shot multi-speaker TTS systems to use a pretrained vocoder network, such as WaveNet [12] and HiFi-GAN [18], without any explicit modifications to incorporate speaker identity information [23, 24, 25].

Finally, a relatively unexplored area is extending the zero-shot multi-speaker TTS system to generate fictitious speaker voices. Conditioning the TTS model on data points randomly sampled from a continuous speaker embedding space can be used to generate random, unseen voices. Jia et al. [23] use uniformly sampled points from the surface of the unit hypersphere for this purpose.

1.5 Contribution

This work studies and contributes to various factors influencing the output of a zero-shot multi-speaker TTS system as mentioned below.

1. FiLM method, a general-purpose conditioning method adapted from computer vision, is used for the first time to inject speaker embeddings into the TTS workflow, and it is compared against the traditional methods of averaging and concatenating
2. X-Vector and ECAPA-TDNN speaker embedding models are used to highlight the importance of speaker encoders using Tacotron2 as the base TTS model extended for zero-shot multi-speaker TTS
3. FastSpeech2 architecture is enhanced with an additional component to improve the pace of the synthesized speech at inference time
4. The effectiveness of injecting speaker embeddings into one or more locations is examined for Tacotron2 and FastSpeech2
5. Lastly, a Gaussian Mixture Model (GMM) is used to model the distribution of speaker embeddings in an attempt to make the embedding space more continuous. This allows to randomly sample speaker embeddings from the GMM and generate speech with fictitious voices extending the capabilities of the zero-shot multi-speaker TTS model.

The performance of the best model configuration obtained from the experiments is compared against that of the YourTTS model, which has achieved state-of-the-art (SOTA) performance on the VCTK dataset for zero-shot multi-speaker TTS [2].

Additionally, this work aims to contribute to the SpeechBrain project [30] - an open-source conversational artificial intelligence toolkit built on PyTorch. This comprehensive toolkit covers various aspects of conversational AI, such as voice activity detection, speech enhancement, keyword spotting, speech recognition, spoken language understanding, dialogue, and text-to-speech. The SpeechBrain project aims to foster open research by providing pretrained models, training recipes, and training logs. The experiments conducted for this work use SpeechBrain for implementation. The code developed in this work is publicly available to the community through SpeechBrain to promote transparent and reproducible research in this field.

1.6 Thesis Outline

In Chapter 2, an overview of zero-shot multi-speaker TTS is provided, encompassing the speech synthesis pipeline and the various components employed in the experiments. Chapter 3 delves into the experimental setup, detailing the datasets utilized and the systematic approach taken to investigate the different facets of zero-shot multi-speaker TTS systems. The findings obtained from the experiments are presented and analyzed in Chapter 4. Finally, Chapter 5 offers concluding remarks based on the results.

2 Zero-Shot Multi-Speaker TTS

There are three main, independently trained components used for the zero-shot multi-speaker TTS experiments in this work:

1. A speaker encoder to capture the speaker identity from an input waveform using a fixed-sized speaker embedding
2. A mel-spectrogram prediction network that uses input text and speaker embedding to synthesize the output mel-spectrogram
3. A vocoder to convert the mel-spectrogram into the final output waveform

This chapter explains the workflow for zero-shot multi-speaker TTS, followed by component details.

2.1 Workflow for Zero-shot Multi-speaker TTS

The zero-shot multi-speaker TTS workflow at training and inference time can vary to some degree depending on the component models. Section 2.1.1 explains the training workflow used in this work, and Section 2.1.2 highlights the changes to the workflow at inference time.

2.1.1 Training Workflow

Figure 3: Training Workflow for Zero-Shot Multi-Speaker TTS

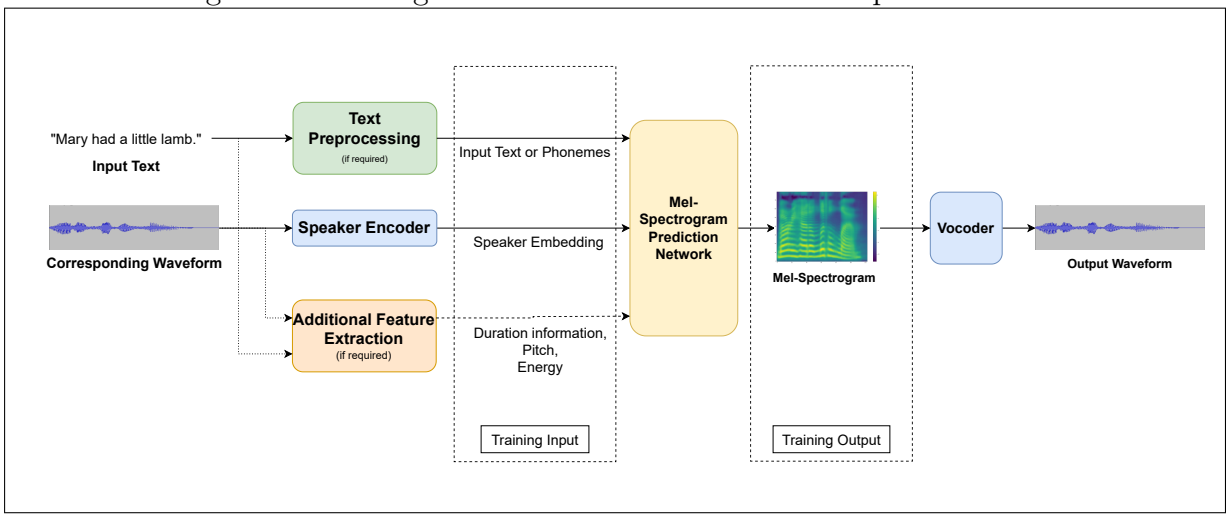


Figure 3 illustrates the training workflow for zero-shot multi-speaker TTS. The following key considerations are noteworthy:

1. The speaker embeddings are generated using an external, pretrained speaker encoder
2. The vocoder that converts a mel-spectrogram into the final waveform can be used without injecting speaker embeddings into it. Thus, a pretrained vocoder is used. Section 4.3.1 reports results supporting this.

3. Additional feature extraction is an optional step depending on the mel-spectrogram prediction network

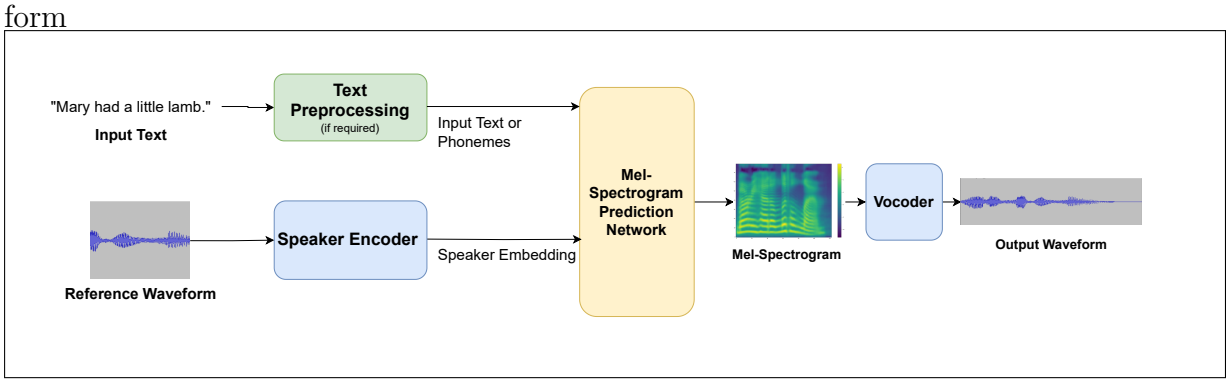
As pretrained speaker encoder and vocoder models are employed, the primary focus of training lies in the mel-spectrogram prediction network. It is trained to effectively utilize input text and speaker embeddings to predict the mel-spectrogram.

Training a mel-spectrogram prediction network requires transcribed speech data (text-utterance pairs). The target utterance (waveform) is processed to get the target mel-spectrogram. The input text may go through some preprocessing, such as grapheme-to-phoneme conversion. The target waveform is also used to compute a speaker embedding input using a speaker encoder. During training, the model learns to map the input text and speaker embedding to the target mel-spectrogram. Some models may require additional features during training, such as duration information for speech segments, pitch, and energy, which can be computed using the input text and the corresponding target waveform. The model learns to predict these additional features during training.

2.1.2 Inference Workflow

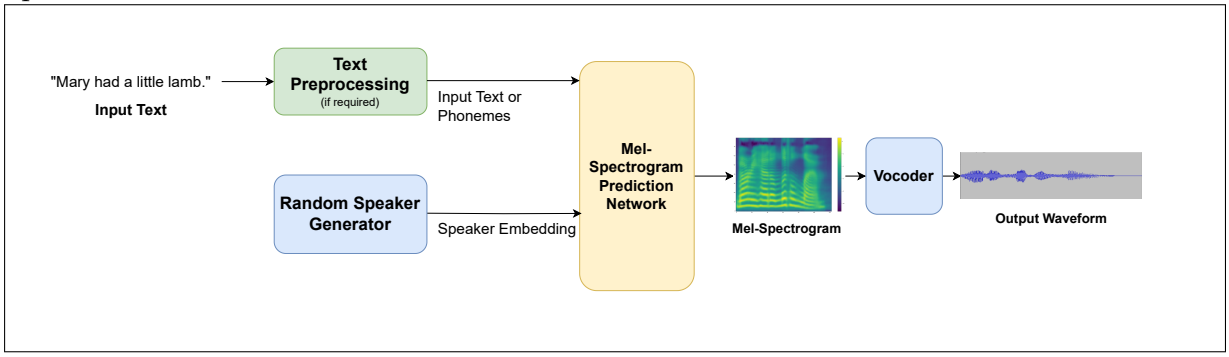
At inference time, along with input text, a speaker embedding computed for a reference waveform is used as the input to the TTS system, as shown in Figure 4. The preprocessed input text and speaker embedding are used to predict any additional features required by the mel-spectrogram prediction network while generating the final mel-spectrogram.

Figure 4: Inference Workflow for Zero-Shot Multi-Speaker TTS with a Reference Waveform



The speaker encoder can be replaced with a random speaker generator to generate fictitious speaker voices. It samples random speaker identities from a speaker embedding space. Figure 5 shows the workflow of the TTS system in this case.

Figure 5: Inference Workflow of Zero-Shot Multi-Speaker TTS Modified with a Random Speaker Generator



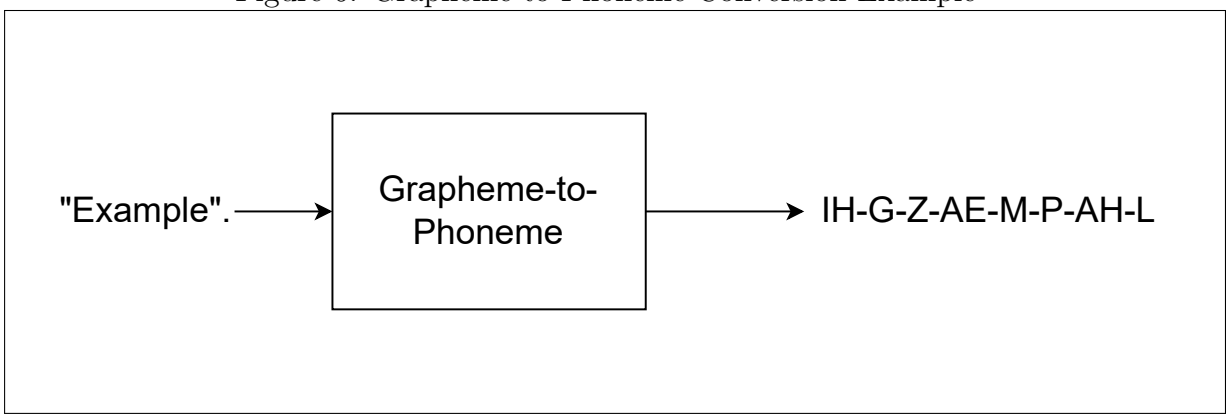
The following sections discuss the building blocks of the zero-shot multi-speaker TTS workflows described above.

2.2 Text Preprocessing

The main focus of text preprocessing in a TTS system can be converting input sentences to their phonetic transcription, also known as grapheme-to-phoneme conversion. A grapheme is the smallest unit in a writing system. Similarly, a phoneme is the smallest unit of speech. This step is not mandatory, and text can be used as input without converting it to a phoneme sequence. However, phonetic transcription is advantageous because it reduces the gap between input and output. A phonetic transcription represents how different words are pronounced when spoken. Sometimes, especially in English, words with similar spellings may have different pronunciations (e.g., “cough” and “dough”). In this case, using phonemes instead of graphemes can help improve the pronunciation of the synthesized speech. It also helps in the case of silent letters (e.g., “queue”).

The experiments for this work use the SoundChoice [31] grapheme-to-phoneme (G2P) model. Using the SoundChoice G2P model is a beneficial because it is trained to improve disambiguation when pronouncing identically spelled words. It operates on the sentence level rather than the word level, allowing it to use sentence context. The SoundChoice G2P model accepts input text (grapheme sequence), removes punctuation, and produces the corresponding output phoneme sequence. Figure 6 shows an example.

Figure 6: Grapheme-to-Phoneme Conversion Example



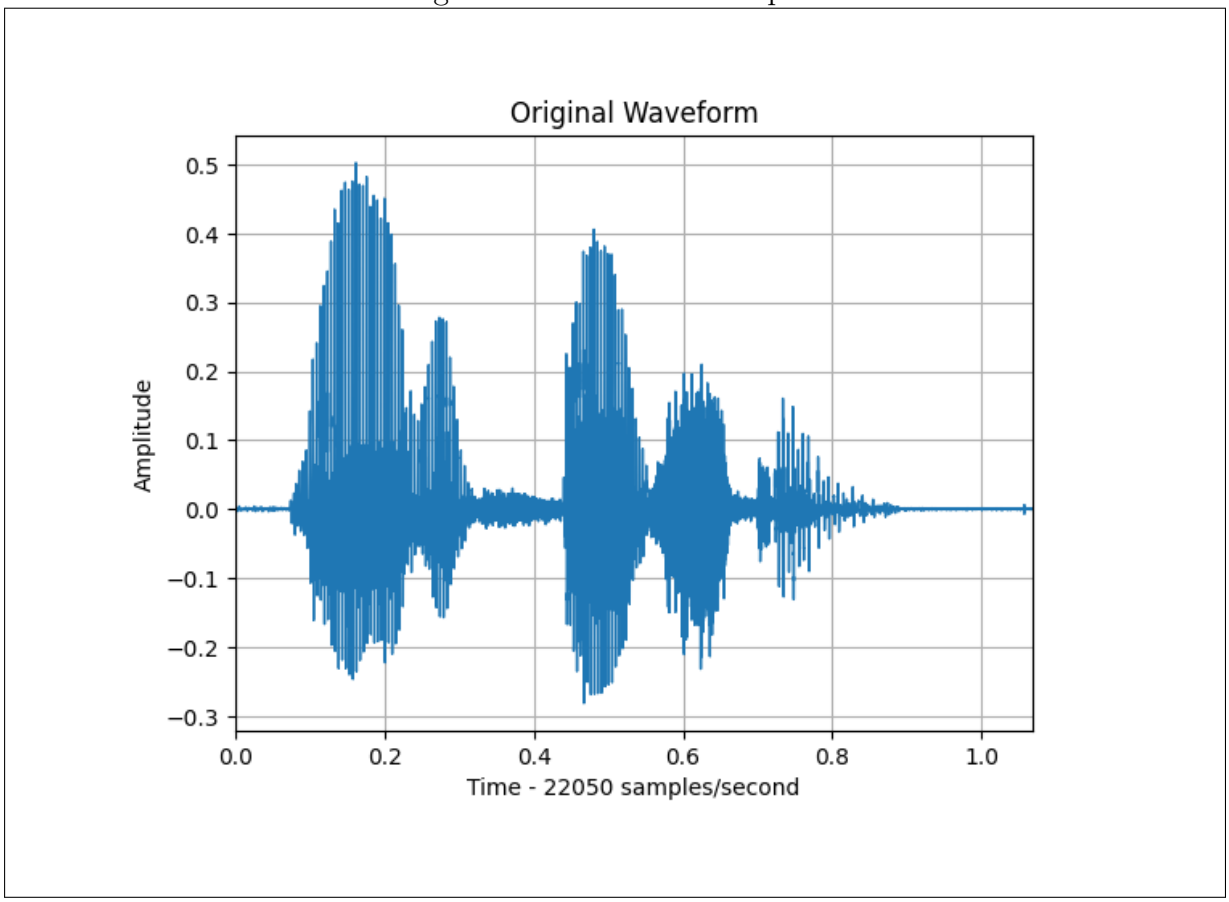
The output phoneme set used by SoundChoice G2P contains phonemes from ARPABET without stress indicators and a word separator token. ARPABET is a phonetic alphabet. It provides the building blocks of phonetic transcription. It represents each phoneme with 1 or 2 characters and a digit between 0 and 2 to indicate the stress used during pronunciation. Appendix A provides the entire set of phoneme tokens used in this work.

2.3 Mel-Spectrogram

The mel-spectrogram is a well-established way of representing speech signals [6, 18, 23, 24, 25]. Mel-spectrograms are fundamental in multi-speaker TTS because they provide a compressed representation of raw waveforms and encompass speaker information.

A speech signal is a non-periodic signal composed of a large number of single-frequency waveforms. This large number of frequency components results in speech signals having a very high dimensionality. Humans can perceive sounds with frequencies ranging from around 20 Hz to 20000 Hz. A speech signal with a sample rate of 22050 Hz contains 22050 samples per second. Figure 7 shows an example of a waveform with a sample rate of 22050 Hz.

Figure 7: Waveform Example



A spectrogram is a way to obtain a compressed representation of high-dimensional speech signals. It shifts the focus of speech representation from the time domain to the

frequency domain. A spectrogram uses a heatmap to show how the amplitudes of different frequency components of an audio signal change over time. It is obtained by processing the speech signal in overlapping windows of fixed size and hop length. A spectrogram frame is generated for every hop.

The spectrogram representation can be compressed further while retaining useful information. The human perception of the speech frequency spectrum is not linear. We are better at distinguishing between frequencies on the lower end of the spectrum than between frequencies on the higher end of the spectrum. The Mel scale is a perceptual scale of pitches designed to mimic the human auditory system's response to different frequencies. The Mel scale focuses more on the lower end of the frequency spectrum than the higher end. A mel-spectrogram is derived from a spectrogram by filtering and averaging the frequency axis using the Mel scale, providing a more accurate representation of speech perception by the human auditory system.

Figure 8 shows an example of a spectrogram-to-mel-spectrogram transformation by applying mel-filters to the frequency axis of the spectrogram. The colour dimension represents the variations in amplitude values of frequencies.

Figure 8: Spectrogram to Mel-Spectrogram Conversion Example

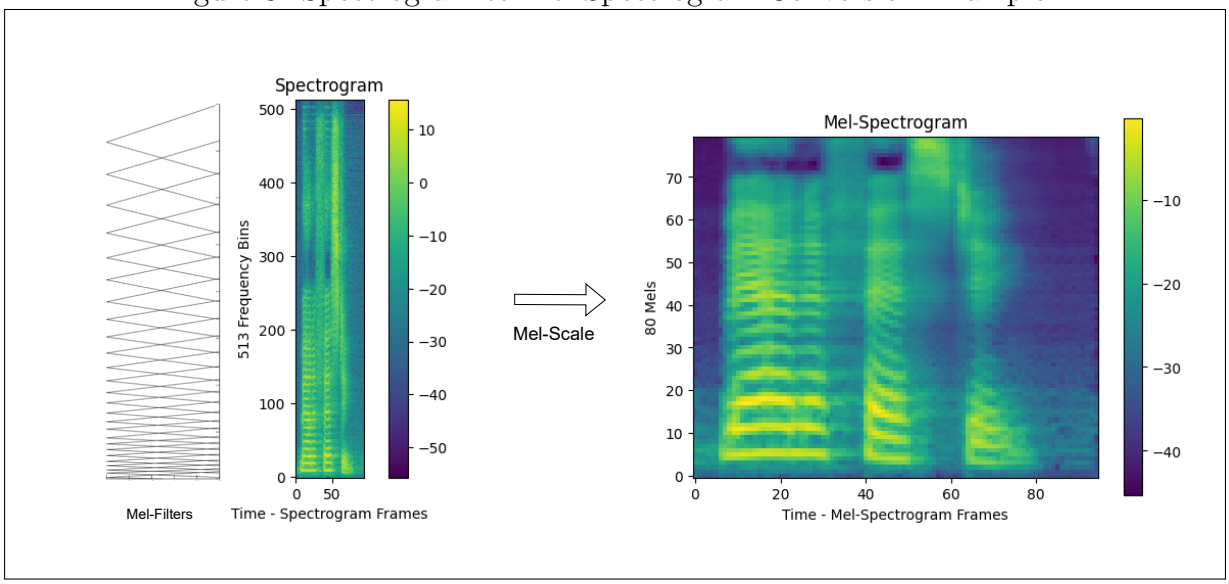
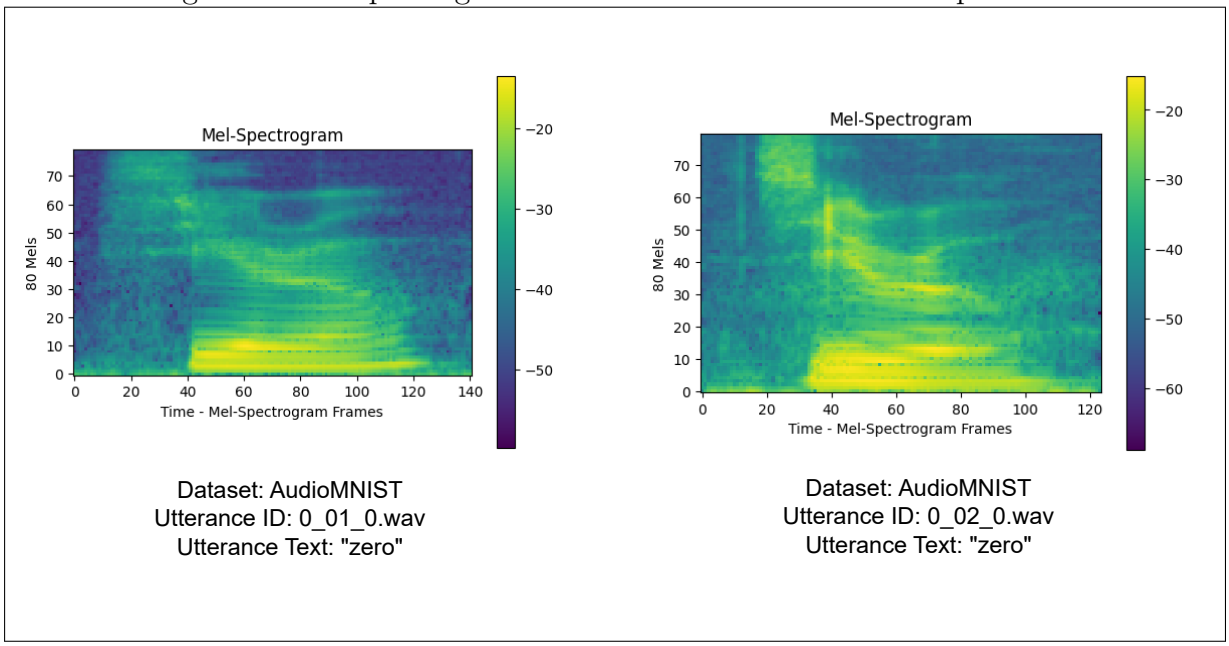


Figure 9 contains mel-spectrograms for the following two utterances selected from the AudioMNIST dataset [32]: "0_01_0.wav", "0_02_0.wav". Each utterance is for a different speaker and the same text, "zero". Some structural similarity can be seen between the two mel-spectrograms because of the same text condition. However, it is clear that the two mel-spectrograms are distinct. This verifies that mel-spectrograms are subject to the speaker.

Figure 9: Mel-Spectrogram for Same Text with Different Speakers



For the experiments in this work, mel-spectrograms are used in three different networks:

1. As input features for the speaker encoder model
2. As target values for the mel-spectrogram prediction model
3. As input features for the vocoder model

In all cases, mel-spectrograms are computed using torchaudio [33] with 80 mel channels (filters) and the sample rate for the model under consideration. Appendix C provides the parameter setup used to compute the mel-spectrograms with torchaudio.

2.4 Speaker Encoder

The speaker encoder captures the speaker identity from a variable-length audio waveform as a fixed-sized embedding. An ideal speaker encoder is robust against potential noise and words present in the input audio. The speaker encoders used for the experiments are trained separately on a discriminative speaker identification task using untranscribed speech data, with data augmentation including adding noise and reverberation to improve their robustness. Mel-spectrograms computed from the initial waveforms are used as the input features. The architecture used for training the models consists of a speaker encoder followed by a classifier. After training, the speaker classifier is discarded, and only the speaker encoder is used for zero-shot multi-speaker TTS.

X-Vector [27] and ECAPA-TDNN [1] are the models used to compute the speaker embeddings. These models are good choices for the speaker encoder component because of their excellent performance on speaker discriminative tasks [27, 1] and successful demonstrations of adopting discriminative speaker embedding models from prior works in zero-shot multi-speaker TTS [23, 24, 25].

2.4.1 X-Vector

The X-Vector architecture, shown in Figure 10, first extracts speaker representations at the frame level and aggregates them to operate at the segment level for speaker identification. The X-Vector workflow is as follows:

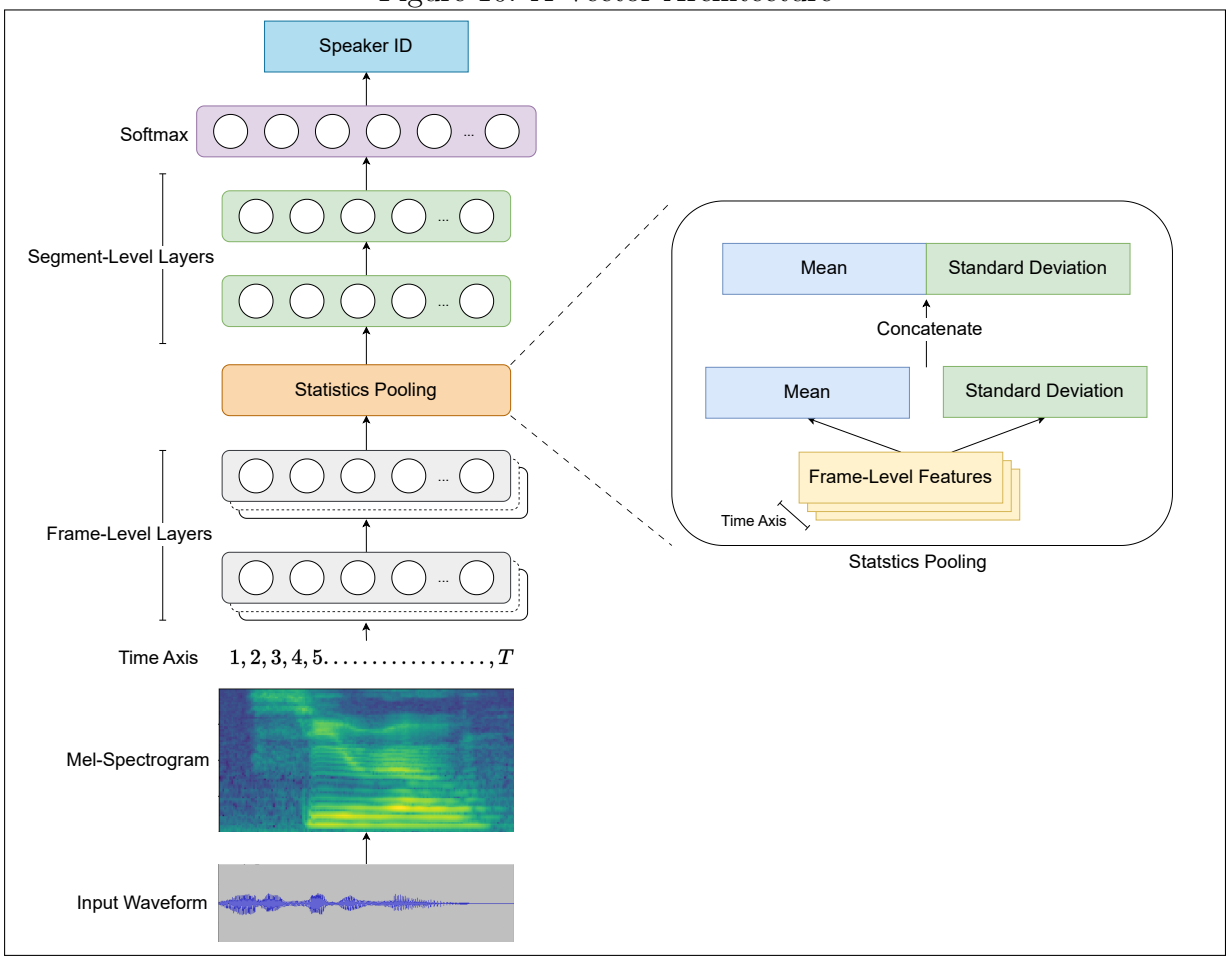
1. TDNN (Time Delay Neural Network) layers are used on feature frames of variable-length utterances.

2. The frame-level representations obtained are aggregated using statistics pooling to get segment-wide information. The statistics pooling layer accepts the output of the final frame-level layer as input and computes the mean and standard deviation along the time axis. The mean and standard deviations are concatenated, and the result is used as input for the segment-level layers.

3. Finally, linear transformations along with a final softmax layer are used to classify the speaker.

After training, the output of the first linear layer after statistical pooling is used as the fixed-size speaker embedding. The X-Vector speaker embedding used for the experiments has 512 dimensions.

Figure 10: X-Vector Architecture



2.4.2 ECAPA-TDNN

The ECAPA-TDNN architecture, shown in Figure 11, uses a workflow similar to the X-Vector architecture. However, it introduces the following enhancements over the TDNN-based architecture:

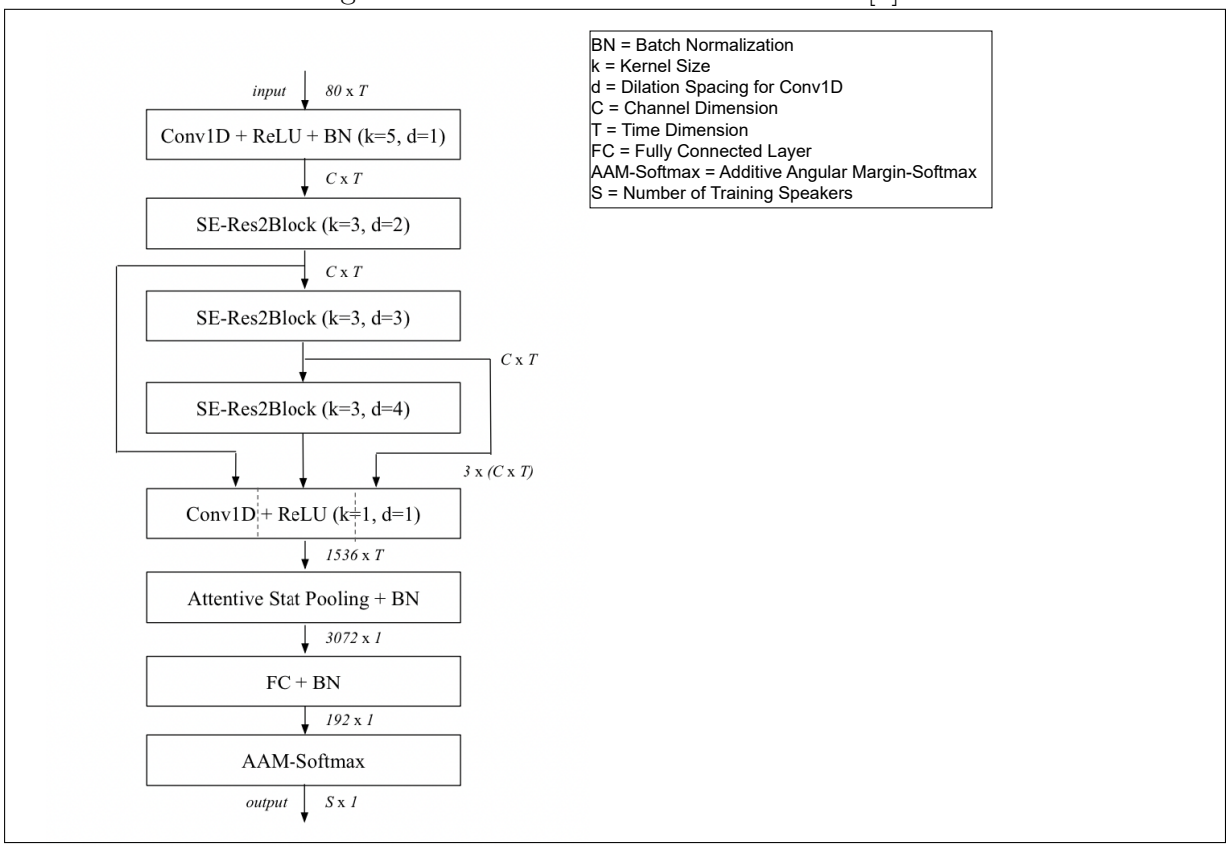
1. It introduces 1-Dimensional Squeeze-Excitation (SE) Res2Blocks in the initial frame-level layers to increase the temporal context. In order to achieve this, frame-level features are rescaled based on the global-level properties of the utterance.

2. It aggregates features at different levels of the model to leverage hierarchical features learned at various levels of complexity.

3. Finally, it uses attentive statistic pooling [34] for each channel. Attentive statistic pooling computes the weighted mean and standard deviation for each time step, as not every time step is equally important. The weights of different time steps are computed using an attention mechanism. The channel-specific frame attention used in ECAPA-TDNN allows the network to focus on different frames when computing statistics for different channels.

The output of the linear transformation applied after the attentive statistics pooling is used as the speaker embedding. ECAPA-TDNN uses Additive Angular Margin (AAM)-softmax [35, 36] as the training objective. The ECAPA-TDNN speaker embedding size used for the experiments is 192.

Figure 11: ECAPA-TDNN Architecture [1]



2.5 Additional Feature Extraction

Without considering the multi-speaker setup, TTS is a one-to-many problem. The same text spoken by a speaker at different instances may produce different mel-spectrograms due to various factors, including the duration of speech segments, pitch, and volume. While some models try to learn the intricate details implicitly, others try to use additional information, such as duration, pitch, and energy, that can affect the outcome of a TTS system.

This work considers two models for mel-spectrogram prediction: Tacotron2 [6] and FastSpeech2 [15]. Tacotron2 does not require any additional features. FastSpeech2 authors suggest that when using only text as input to predict the target mel-spectrograms, there is a big gap between input and output. FastSppech2 considers additional input features, such as the duration of pronunciation for a speech segment, pitch, and energy, to reduce the gap between input and output. During training, the target speech waveform is used to extract the additional features, and at the same time, separate predictor modules learn to predict these features. At inference time, the predicted values are used.

The following subsections for duration, pitch, and energy introduce the additional features used for FastSpeech2.

2.5.1 Duration

The duration information for the target speech is derived using the Montreal Forced Aligner (MFA) [37].

Figure 12: MFA Durations for LJSpeech Utterance, LJ001-0002

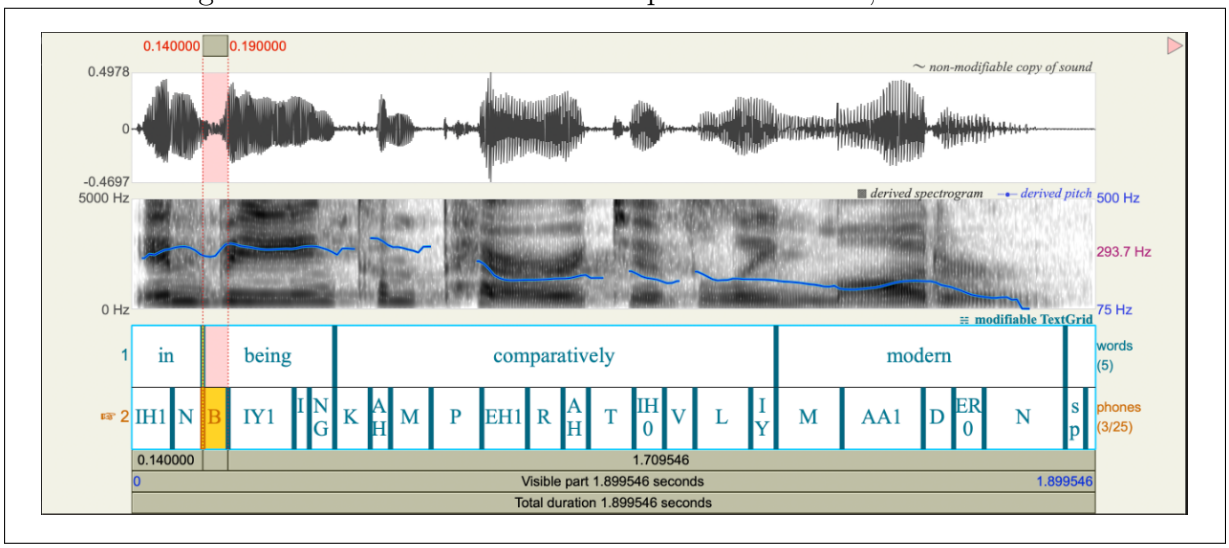


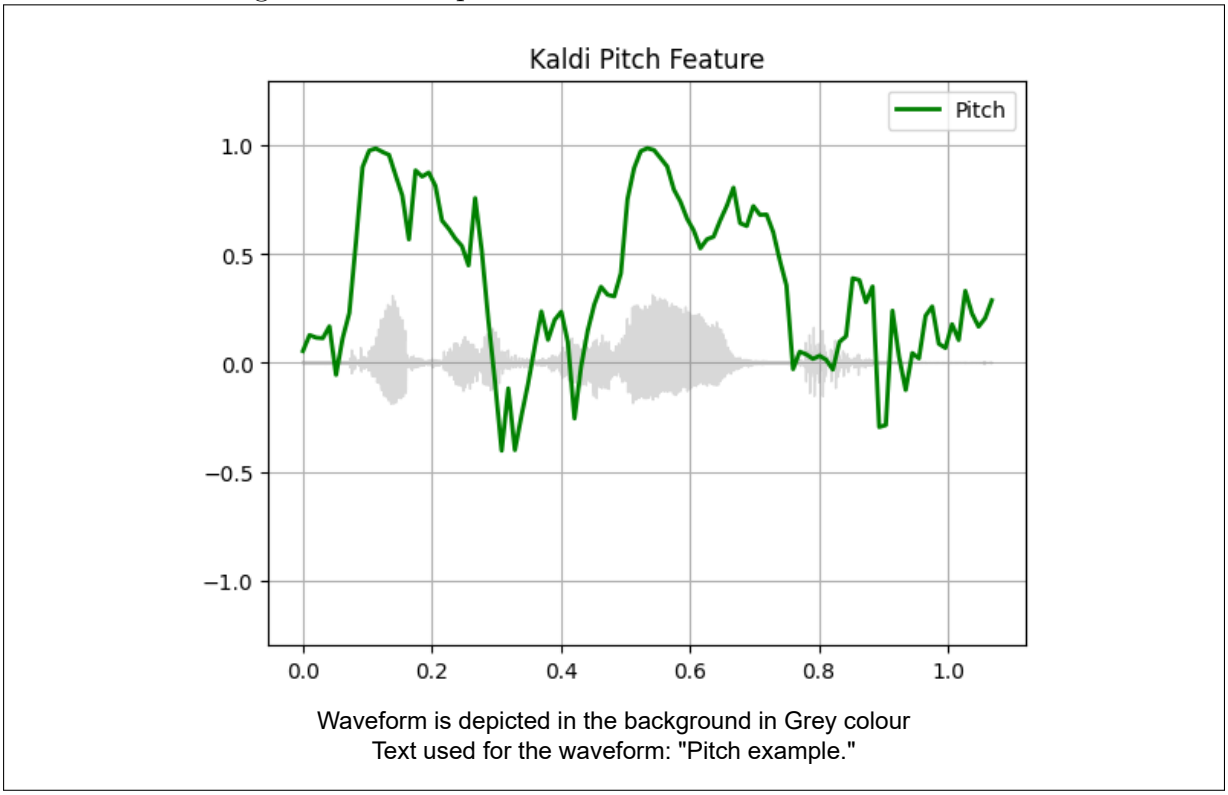
Figure 12 provides a visual representation of the durations extracted by MFA for utterance LJ001-0002 from the LJSpeech dataset. MFA generates durations at both the word and phoneme levels. This work uses phoneme durations. In Figure 12, the highlighted segment corresponds to the phoneme "B," displaying its start and end times in seconds above. These temporal values are then processed to obtain the starting and ending frame numbers in the mel-spectrogram for the respective phoneme. Consequently,

in the experiments, the duration of a phoneme in an utterance refers to the number of frames that correspond to the phoneme within the mel-spectrogram. It is worth noting that the phonemes produced by MFA contain stress indicators, which are removed before being used in the experiments. This is done to be aligned with the phoneme set used by SoundChoice G2P in other experiments and at inference time.

2.5.2 Pitch

Pitch is another factor that contributes to speech variations. It represents how high or low the voice sounds. The prosody of a speech is influenced by how pitch changes throughout the speech. Torchaudio's Kaldi Pitch feature is used to extract pitch information from the target speech waveform. Figure 13 shows an example of how the pitch information varies over time for a waveform. The pitch computation process used in this work ensures that each mel-spectrogram frame has a corresponding pitch value. Appendix C provides the parameter setup used for pitch computation with torchaudio.

Figure 13: Example of Pitch Variations for a Waveform



2.5.3 Energy

Energy affects the volume or loudness of speech. Higher energy values correspond to louder sounds, while lower energy values indicate softer sounds. It corresponds to the frame-level magnitude of mel-spectrograms. The energy computation process used in this work ensures that each mel-spectrogram frame has an associated energy value. For a mel-spectrogram frame, energy is computed as the L2-norm of the magnitude values of the frame. This is followed by mix-max normalization for energy values across all frames.

2.6 Mel-Spectrogram Prediction Network

The mel-spectrogram prediction network is a primary candidate for incorporating speaker encoding capabilities into a TTS system. It accepts a text or phoneme sequence as input and generates a mel-spectrogram representing its corresponding speech. As shown in Figure 9 on page 13, the content of a mel-spectrogram is subject to the speaker. With only text as input, the model does not have a structured way to add speaker-specific details to the mel-spectrogram. For a single-speaker TTS system, the standard approach is to train the model using the voice of only one speaker, and the speaker's voice characteristics are captured implicitly. The multi-speaker TTS models examined for this work are trained on text-utterance pairs for multiple speakers. In this case, it becomes imperative to provide speaker identity as input along with the input text. It is required for the model to learn how to map an input text with different speaker identities to different mel-spectrograms.

When injecting speaker identity into a mel-spectrogram prediction model, the range of choices depends on the model architecture. Two models are studied here: Tacotron2 [6] and FastSpeech2 [15].

2.6.1 Tacotron2

Tacotron2 is an autoregressive TTS model. It generates the mel-spectrogram one frame at a time. This subsection introduces the Tacotron2 architecture along with possible speaker embedding injection locations. Additionally, fundamental details about training Tacotron2 are provided.

Architecture

Figure 14 illustrates a modified Tacotron2 model architecture, including speaker embedding injection. The original Tacotron2 architecture consists of the following main components: an encoder, an attention network, a decoder, and a postnet.

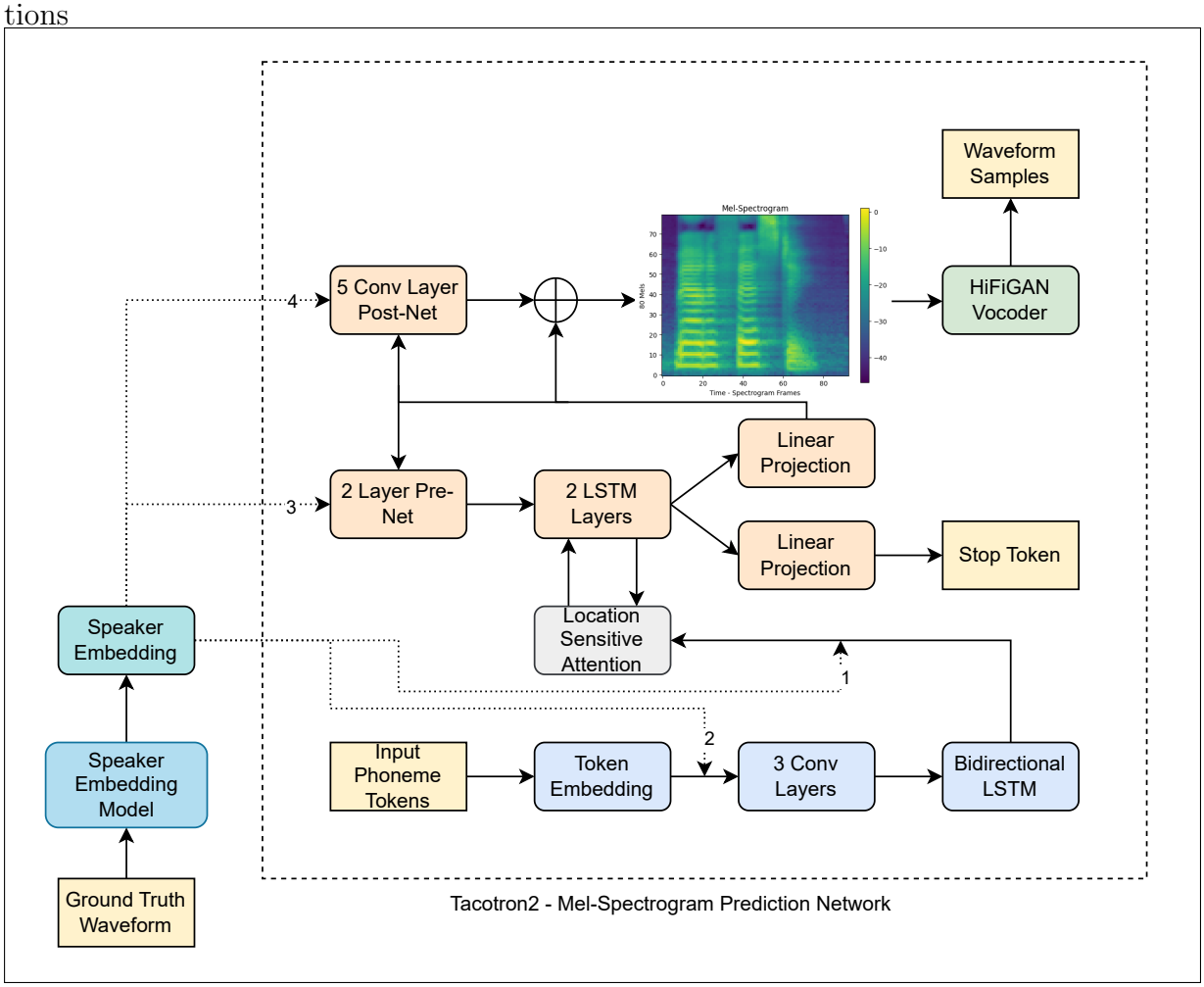
1. The encoder is responsible for converting a character or phoneme sequence into a hidden feature representation, which is learned as fixed-size embeddings. It consists of convolution layers followed by bidirectional LSTM.

2. The attention network accepts the output of the encoder as input. It summarizes the entire encoded sequence into a fixed-length context vector. This context vector is used for each decoding step.

3. The decoder operates as an autoregressive RNN, utilizing the encoded input sequence to predict the mel-spectrogram one frame at a time. The input at each time step is first passed through a prenet before being decoded. In addition to predicting the mel-spectrogram, the decoder also predicts a "stop token" that indicates whether the output sequence generation is complete. This "stop token" is particularly useful during inference, as it enables the model to dynamically stop producing the output when mel-spectrogram generation is complete.

4. The mel-spectrogram generated by the decoder goes through a postnet, which predicts a residual to add to the decoded mel-spectrogram to improve the overall reconstruction.

Figure 14: Multi-Speaker Tacotron2 with Numbered Speaker Embedding Injection Locations



Speaker Embedding Injection Locations

Given this architecture, there are a few potential locations for speaker embedding injection, as shown with numbered dotted lines in Figure 14. The following choices are explored:

1. Encoder output: One evident choice, also used by Jia et al. [23], is the encoder output before it is passed to the attention layer.

2. Encoder input: Speaker embeddings are combined with the input phoneme token embeddings before being passed through the entire Tacotron2 network.

3. Decoder prenet: Speaker embeddings are injected into the decoder input for every time step.

4. Postnet: Speaker embeddings are injected into the postnet input.

Training Tacotron2

The objective function for the Tacotron2 implementation in this work consists of the following components:

1. Mel-spectrogram loss: Mean Squared Error (MSE) is used for comparing the predicted and ground truth mel-spectrograms. MSE is computed for the mel-spectrogram generated before and after the postnet and summed.

2. Stop token loss: Binary Cross Entropy (BCE) is used for the "stop token" prediction.

3. Attention loss: Guided Attention Loss is used to force the attention matrices to be near-diagonal. This is particularly beneficial for sequence-to-sequence models like Tacotron2, where the output sequence is expected to be closely correlated with the input sequence.

During training, Tacotron2 employs teacher forcing for the decoder. It utilizes the correct values from the ground truth target mel-spectrograms instead of predicted values from previous time steps. However, during inference, when the target values are unknown, the model relies on predictions from previous time steps to generate the mel-spectrograms.

2.6.2 FastSpeech2

FastSpeech2 is one of the non-autoregressive TTS models that enjoy the benefits of faster mel-spectrogram generation. This subsection presents the original FastSpeech2 architecture, now extended to support speaker encoding, and explores various speaker embedding injection locations. Additionally, a new component is introduced to further improve the pace and naturalness of speech synthesis with FastSpeech2. Finally, essential training details for FastSpeech2 are provided.

Architecture

Figure 15 shows the FastSpeech2 architecture used for the experiments, with potential speaker embedding injection locations indicated using dotted lines. For Figure 15, subfigure a) depicts the overall architecture, and subfigure b) shows a more detailed view of the variance adaptor module. Without considering the multi-speaker aspect, the FastSpeech2 model has the following main components: an encoder, a variance adaptor, a decoder, and a postnet.

1. The encoder is based on multiple feed-forward transformer blocks, which contain a self-attention layer followed by a 2-layer 1D-convolution network. The encoder maps an input phoneme embedding sequence to its latent representation.

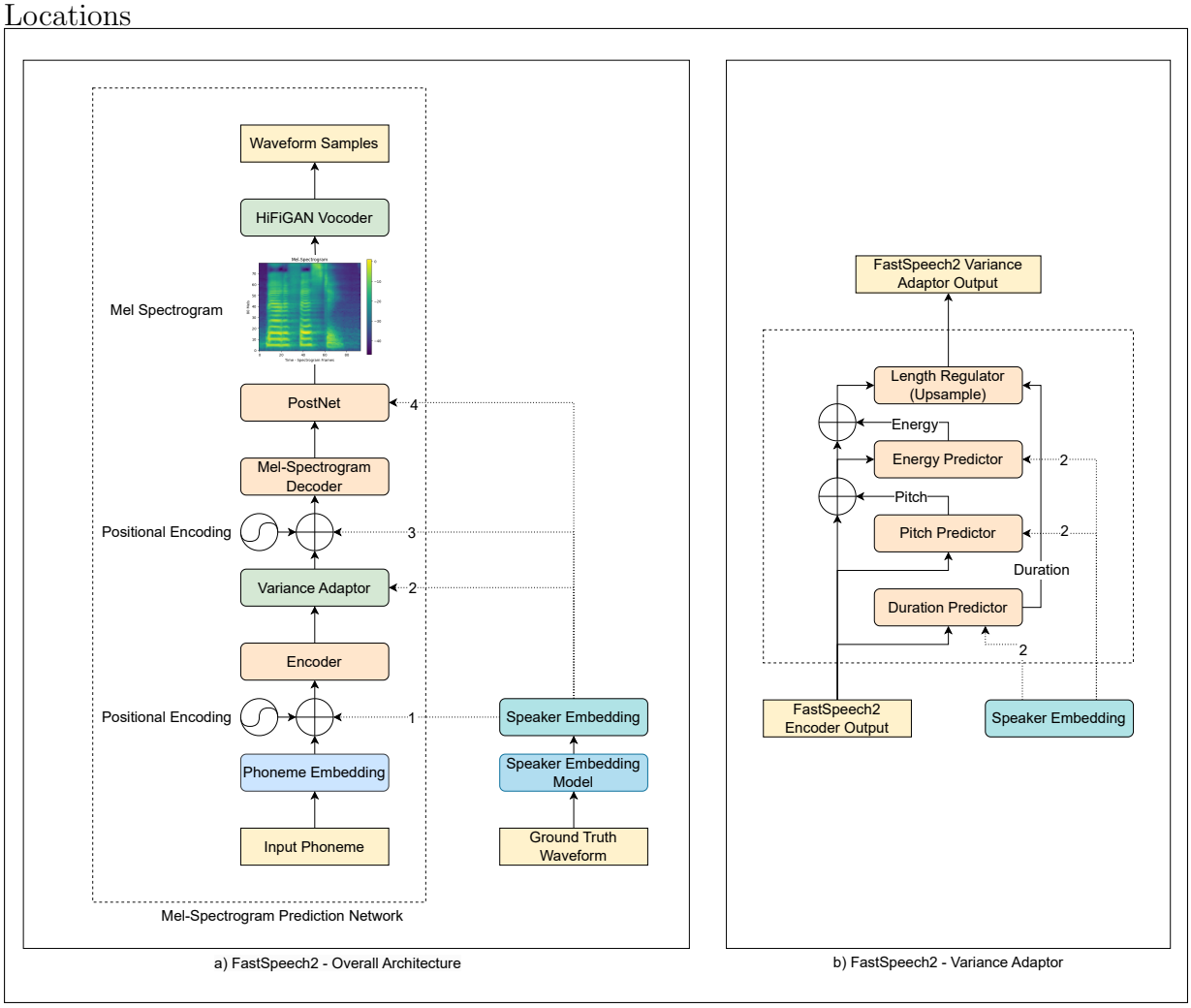
2. The encoder output is passed through the variance adapter module to augment the latent representations with variance information such as duration, pitch, and energy. Figure 15 - subfigure b) shows the workflow of the variance adaptor component. The variance adaptor has three different modules: a duration predictor, a pitch predictor, and an energy predictor. The duration predictor uses the ground truth phoneme durations extracted using MFA as the training target. It accepts a sequence of phoneme latent representations as input and learns to predict the phoneme durations. However, when training FastSpeech2, the predicted phoneme durations are discarded. The ground truth phoneme durations are combined with the phoneme latent representations. The output is passed to the next predictor module in the variance adapter. The pitch and energy predictor modules follow

the same approach for adding information to the latent representation. The model structure for the duration, pitch, and energy predictors is similar. It consists of a 2-layer 1D-convolutional network with ReLU activation, followed by layer normalization and a dropout layer, and a linear layer at the end to project the hidden states into the output sequence.

3. The modified latent representations are passed through the mel-spectrogram decoder to construct the mel-spectrogram. The decoder has the same feed-forward Transformer block-based structure as the encoder.

4. Similar to Tacotron2, the postnet accepts the decoded mel-spectrogram and predicts a residual to add to the decoded mel-spectrogram to improve the overall reconstruction.

Figure 15: Multi-Speaker FastSpeech2 with Numbered Speaker Embedding Injection Locations



Speaker Embedding Injection Locations

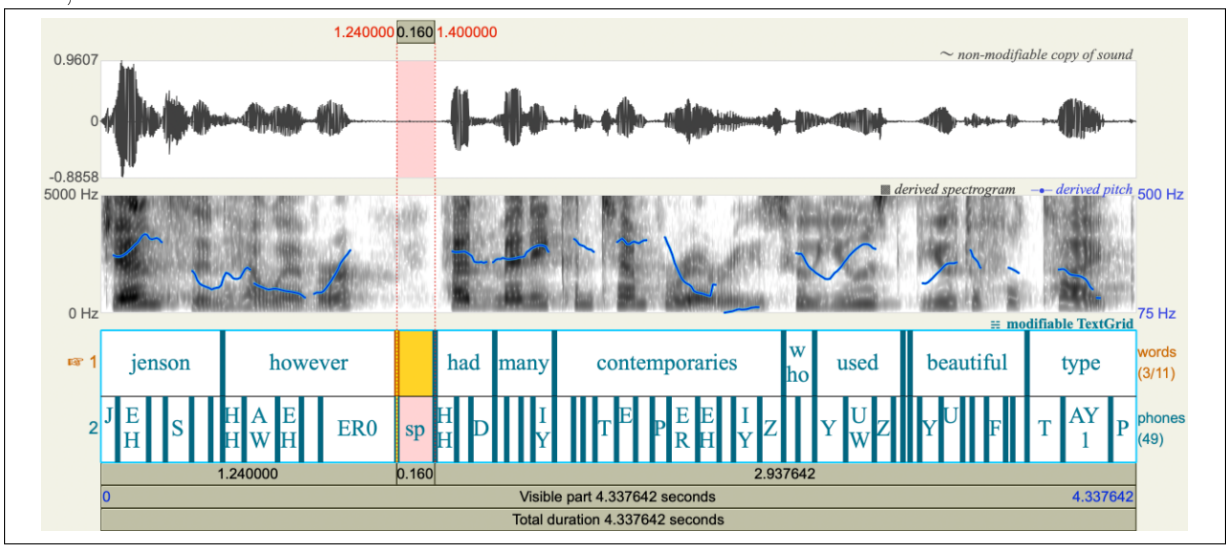
Figure 15 indicates the following speaker embedding injection locations for FastSpeech using dotted lines:

1. Encoder input: Speaker embeddings injected into the input to the encoder
2. Variance adaptor components: Speaker embeddings injected into the duration predictor, pitch predictor, and energy predictor inputs
3. Decoder input: Speaker embeddings injected into the decoder input
4. Postnet input: Speaker embeddings injected into the postnet input

Additional Component to Improve Speech Synthesis

In addition to the modifications necessary for integrating speaker embeddings, the FastSpeech2 architecture employed in this work uses an extra component called the silent phoneme predictor. This component is useful for effectively incorporating silent periods within speech, including pauses that occur during speaking. The significance of this supplementary component becomes evident when examining an input phoneme sequence alongside its corresponding target waveform. Figure 16 illustrates the phoneme durations extracted using MFA for utterance LJ001-0054 from the LJSpeech dataset. The highlighted section in Figure 16 represents a silent period, indicating the pause taken during the spoken utterance. Such pauses or silent periods play a crucial role in the naturalness of speech. To represent these silent periods in the audio, MFA utilizes silent phoneme tokens within the phoneme sequences. However, it is important to note that this information is only available during training, as the target speech waveform is not available during inference. Consequently, the silent phoneme tokens cannot be extracted using MFA in the inference phase.

Figure 16: MFA Phoneme Durations Highlighting the Silent Period in LJSpeech Utterance, LJ001-0054

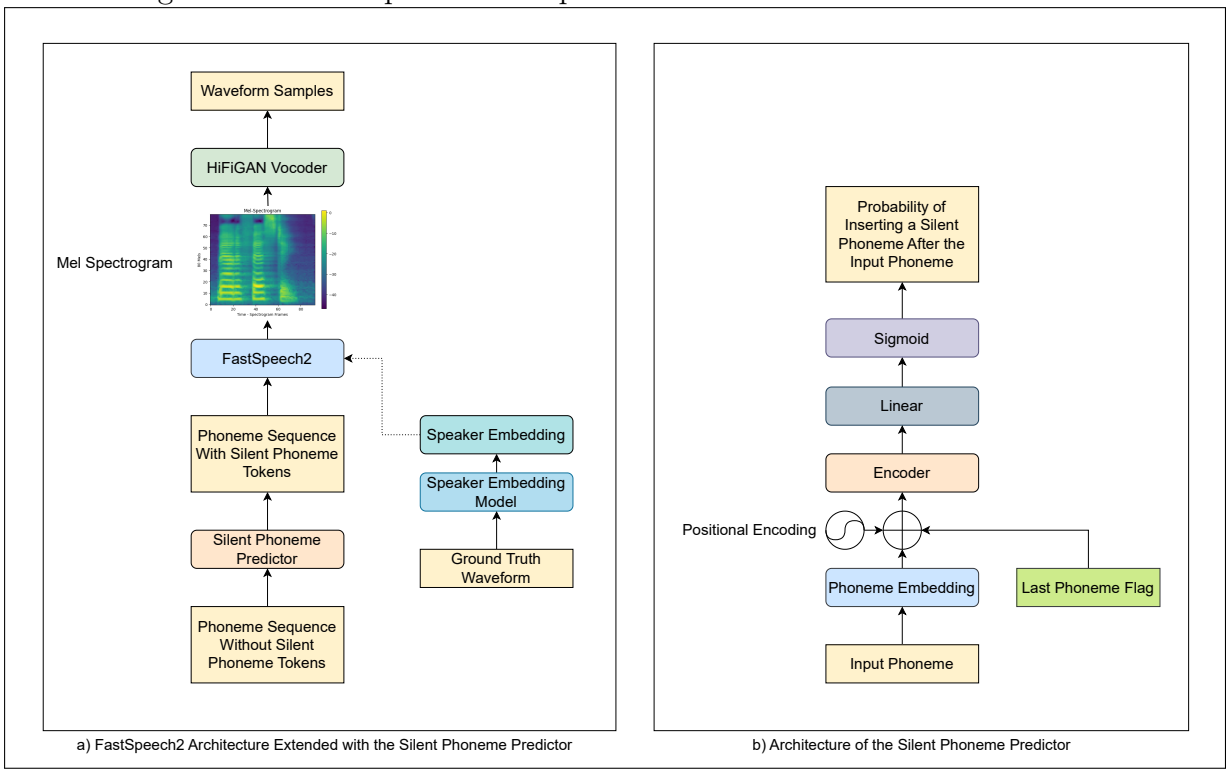


To address this challenge, a novel silent phoneme predictor component is introduced, which takes a phoneme sequence without silent phoneme tokens as input and predicts the appropriate positions for inserting silent phoneme tokens. Additionally, information about word boundaries is provided using a "last phoneme" flag, which indicates whether a phoneme is the last phoneme in the phonetic transcription of a word or not. The purpose

of this additional input is to avoid the insertion of silent phonemes at unexpected places, such as in the middle of the phonetic transcription of a word.

The silent phoneme predictor first processes the input sequence using an encoder with the same architecture as the FastSpeech2 encoder. However, it employs masked attention to mask future elements. The encoder output is then passed through a linear layer and sigmoid activation to estimate the probability of inserting a silent phoneme after each phoneme in the input sequence. Figure 17 illustrates the connection between the silent phoneme predictor and the original FastSpeech2 network, along with the architecture of the silent phoneme predictor. Appendix D provides more details about the configuration of the silent phoneme predictor.

Figure 17: Multi-Speaker FastSpeech2 with Silent Phoneme Predictor



Training FastSpeech2

During training, the silent phoneme predictor and the main FastSpeech2 network are jointly trained using the same dataset. The input phoneme sequence for training the silent phoneme predictor is constructed by excluding the silent phoneme tokens from the phoneme sequence extracted using MFA for duration information. The original phoneme sequence from MFA-extracted durations serves as the target value for the silent phoneme predictor. In the joint training process, the ground truth phoneme sequences from MFA-extracted durations are utilized as input for FastSpeech2. During inference, the input text is first converted into a phoneme sequence using SoundChoice G2P. This sequence does not contain silent phoneme tokens. It is subsequently passed through the silent phoneme predictor, which inserts the silent phoneme tokens at appropriate positions. The resulting phoneme sequence is then used as the input for the main FastSpeech2 network.

The objective function for the joint training of FastSpeech2 and the silent phoneme predictor implementation used in this work consists of the following components:

1. Mel-spectrogram loss: Mean Squared Error (MSE) is used for comparing the predicted and ground truth mel-spectrograms. MSE is computed for the mel-spectrogram generated before and after the postnet, similar to the Tacotron2 training objective.

2. SSIM loss: Structural Similarity Index Measure (SSIM) loss is defined as $(1 - SSIM)$. SSIM is used to measure the similarity between two images [38]. In this work, it is computed between the target and predicted mel-spectrograms.

3. Duration loss: MSE is computed using the predicted and target durations extracted using MFA.

4. Pitch loss: MSE is computed using the predicted and target pitch values.

5. Energy loss: MSE is computed using the predicted and target energy values.

6. Silent phoneme loss: Binary Cross Entropy (BCE) is used for the silent phoneme prediction.

2.7 Speaker Embedding Injection

The method of integrating speaker identity information into the TTS system can also influence the outcome. Three approaches are considered here. Averaging and concatenating are traditional approaches for speaker embedding injection. FiLM (Feature-wise Linear Modulation) [29] is a general-purpose conditioning method introduced for the visual reasoning field. FiLM is designed to adaptively modulate the workflow of a neural network computation based on a conditioning input. This work adapts the FiLM method to inject speaker embeddings into the TTS workflow and compares its effectiveness with the traditional approaches of averaging and concatenating.

The following subsections discuss each of the three speaker embedding injection methods.

2.7.1 Average

A speaker embedding is averaged with the hidden representation of every time step in a sequence. In order to support different speaker embedding models, speaker embeddings are resized using a linear transformation and repeated to match the dimensionality expected by the hidden representation.

2.7.2 Concatenate

A speaker embedding is concatenated with the hidden representation of every time step in a sequence. A linear transformation is applied to the concatenated vector to match the expected dimensionality of the next component.

2.7.3 FiLM

FiLM (Feature-wise Linear Modulation) layers affect the outcome of neural network computations by passing intermediate features through a simple, feature-wise affine transformation based on conditioning information. In the context of this work, FiLM allows linear transformations over an input speaker embedding to influence the TTS process of mapping the input phoneme sequence to the output mel-spectrogram.

Considering $F_{i,t,d}$ as the d dimensional intermediate features of the TTS network for i^th input phoneme sequence with t time steps and $s_{i,e}$ as its corresponding speaker embedding with e dimensions, the FilM [29] method involves the following steps.

First, functions h and g are applied to input speaker embedding $s_{i,e}$ to produce output $\gamma_{i,d}$ and $\beta_{i,d}$.

$$\gamma_{i,d} = h_d(s_{i,e})$$

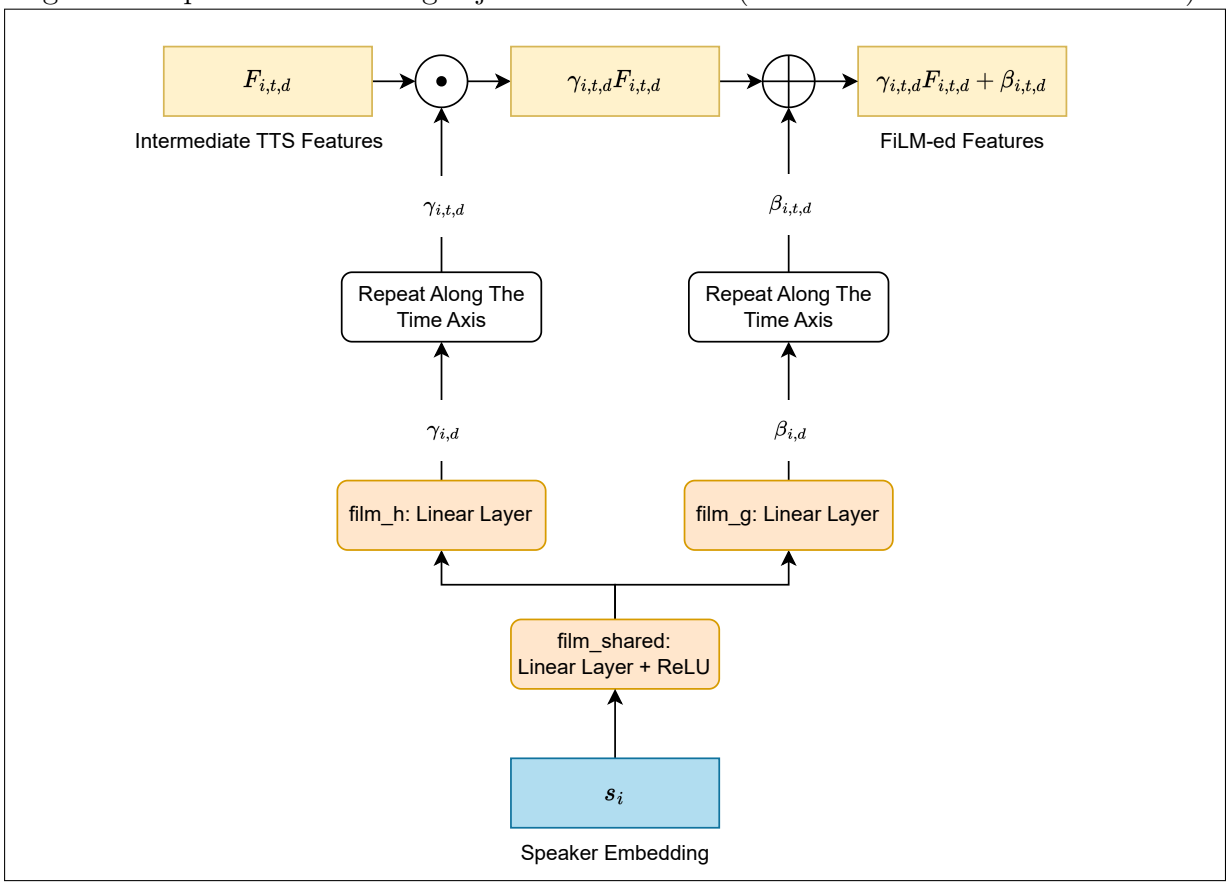
$$\beta_{i,d} = g_d(s_{i,e})$$

$\gamma_{i,d}$ and $\beta_{i,d}$ are repeated t times along the time step dimension to obtain $\gamma_{i,t,d}$ and $\beta_{i,t,d}$. This output is then used to modify the intermediate features at every time step using a feature-wise affine transformation.

$$FiLM(F_{i,t,d}|\gamma_{i,t,d}, \beta_{i,t,d}) = \gamma_{i,t,d}F_{i,t,d} + \beta_{i,t,d}$$

The Hadamard product is used for multiplying $\gamma_{i,t,d}$ and $F_{i,t,d}$.

Figure 18: Speaker Embedding Injection with FiLM (Feature-wise Linear Modulation)



The original work [29] includes the following information regarding FiLM and the functions h and g. They can be any function, such as neural networks. It is advantageous to share parameters between these functions for learning efficiency. The network to which the modulation is applied using the FiLM layer is referred to as the FiLM-ed network. Considering this information, Figure 18 illustrates how the FiLM method is used to inject speaker embeddings into the TTS system.

As shown in Figure 18, functions f and g share the "film_shared" layer. Function h consists of the "film_shared" and "film_h" layers. Function g contains the "film_shared" and "film_g" layers. Similar to the average and concatenation methods, the multiplication and addition operations are performed for every time step. The FiLM method implementation allows adaptively manipulating the TTS workflow by scaling the intermediate features up or down, negating them, thresholding them (with ReLU), etc., by applying scaling factors and offsets learned from the speaker embedding input. The same scaling factors and offsets are used for every time step to provide consistent speaker information throughout the sequence, which is important for maintaining the speaker identity across the generated speech.

Table 1 lists the input and output dimensions of the layers used for FiLM where e is the size of the speaker embeddings and d is the size of the intermediate features of the TTS network.

Table 1: Dimensions for FiLM Layers

Layer Name	**Description**	**Input Size**	**Output Size**
film_shared	Linear Layer + ReLU	e	$(e+d)/2$
film_h	Linear Layer	$(e+d)/2$	d
film_g	Linear Layer	$(e+d)/2$	d

2.8 Vocoder

Vocoder is the second stage model in a two-stage TTS system and converts intermediate representations, such as mel-spectrograms, into waveforms. Training a vocoder does not require transcribed data. However, for the experiments in this work, the transcribed TTS dataset used for training a mel-spectrogram network is also used for its corresponding vocoder training.

Pretrained HiFi-GAN [18] models are used as the vocoder for all experiments.

2.8.1 HiFi-GAN

HiFi-GAN has a Generative Adversarial Network (GAN) architecture, and it is designed to synthesize waveforms from mel-spectrograms.

Figure 19: HiFi-GAN - High-Level View

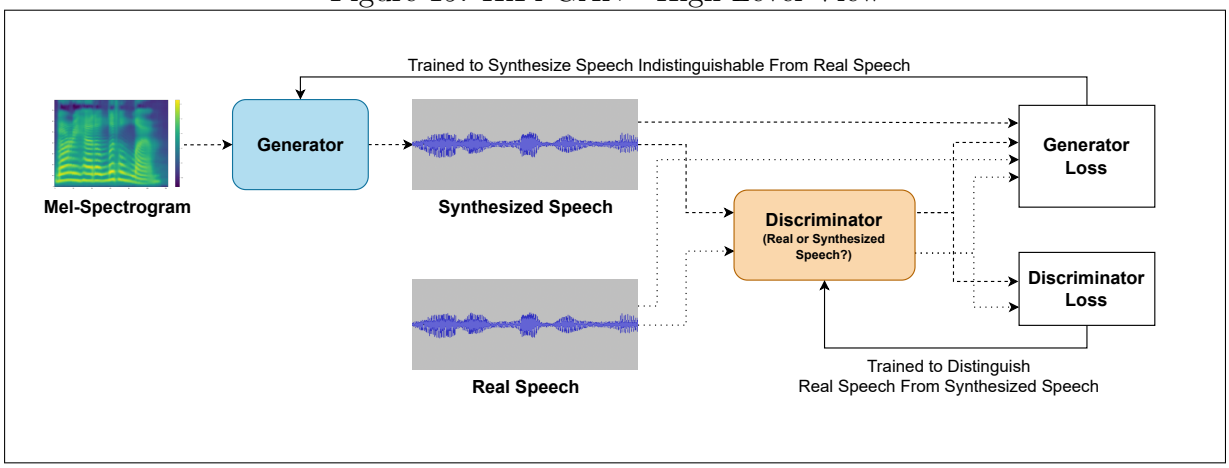


Figure 19 depicts a high-level view of HiFi-GAN. It contains two main components: a generator and a discriminator. The generator is responsible for producing high-quality audio comparable to real, ground truth audio. The goal of the discriminator is to distinguish between real speech and speech synthesized by the generator. They are trained adversarially. After training, the generator is used to convert mel-spectrograms into audio waveforms. The original HiFi-GAN [18] work demonstrates the generalization capability of HiFi-GAN to convert mel-spectrograms of unseen speakers into their corresponding waveforms. Zero-shot multi-speaker TTS is possible without injecting speaker embeddings into the HiFi-GAN architecture [25]. The HiFi-GAN model architecture used in the experiments does not have a separate input for speaker identity. Section 4.3.1 reports results supporting this decision.

2.9 Random Speaker Generator

The zero-shot multi-speaker TTS system is enhanced using a random speaker generator module, utilizing a Gaussian Mixture Model (GMM) to create a more continuous speaker embedding space. The GMM is trained on speaker embeddings extracted from a subset of the LibriTTS dataset, allowing it to learn the underlying structure of speaker embeddings. At TTS inference time, a random speaker embedding is sampled from this space. The randomly sampled speaker embedding is used as input along with some text to generate speech with fictitious speaker voices.

3 Experimental Setup

3.1 Datasets

All datasets are used for the experiments are publicly available: LJSpeech [39], LibriTTS [40], VCTK [41], VoxCeleb [42].

3.1.1 LJSpeech

LJSpeech is a single-speaker TTS dataset consisting of 13,100 text-utterance pairs in English. LJSpeech has approximately 24 hours of speech data at a sample rate of 22050 Hz and a maximum utterance duration of 10.10 seconds. For the experiments, LJSpeech is divided into 90% training and 10% validation data.

3.1.2 VCTK

VCTK is a multi-speaker TTS dataset. It has around 41.6 hours of speech with text-utterance pairs for 109 speakers. It is a high-quality dataset with a sample rate of 48 kHz. Following YourTTS [2], data for the following speaker IDs are reserved as test data: 225, 234, 238, 245, 248, 261, 294, 302, 326, 335, and 347. The remaining dataset is used for training and validation. All utterances were downsampled to the sample rate of 22050 Hz.

Any experiment involving performance comparison with the YourTTS model matches the sample rate and the data split of the YourTTS model.

3.1.3 LibriTTS

LibriTTS is also a multi-speaker TTS dataset containing about 585 hours of English speech with transcription for 2456 speakers. It uses a sample rate of 24 kHz. The dataset has "clean" subsets that contain utterances with a higher signal-to-noise ratio (SNR). It also has "other" subsets that contain utterances with low SNR. Unless specified otherwise, the experiments use the following train, validation, and test splits:

1. Train: train-clean-100 and train-clean-360
2. Validation: dev-clean
3. Test: Data for 10 speakers (5 male, 5 female speakers) from test-clean

All utterances were downsampled to a sample rate of 22050 Hz. Utterances with a duration greater than 10.10 seconds were excluded for training efficiency. In this work, 32 is the maximum batch size possible for training a zero-shot multi-speaker Tacotron2 model (with around 28.6 million trainable parameters) on the LibriTTS data mentioned above, including all utterances, using a single NVIDIA A100 GPU. The training time taken per epoch for this setup is approximately 3.85 hours. Excluding utterances with durations greater than 10.10 seconds allows increasing the batch size to 64. As a result, around 84% of the original number of utterances are used for training, and the time taken per epoch with this new setup is reduced to around 52.35 minutes. Details about the resultant LibriTTS data used for the experiments are provided in Table 2.

Table 2: LibriTTS Data Used for Training and Validation in the Experiments

LibriTTS Subset	**Total Duration (Hours)**	**Reduced Duration (Hours)**	**Total Number of Samples**	**Reduced Number of Samples**	**% of Samples Used**
train-clean-100	53.78	33.07	33,236	28,049	84.39
train-clean-360	191.29	115.26	116,500	97,405	83.60
dev-clean	8.97	5.93	5,736	4,964	86.54

It is important to acknowledge that while the VCTK dataset provides high-quality data, it is limited in terms of the number of speakers (109). As highlighted by the authors of YourTTS [2], zero-shot multi-speaker TTS models trained solely on VCTK struggle to generalize effectively to unseen speakers with significantly different voice characteristics. In contrast, the LibriTTS dataset, derived from LibriSpeech, has a larger number of speakers and draws its data from audiobooks, resulting in potential variations in intonation and speaking style even within the same speaker's utterances. This diversity encompasses a wider range of speaker characteristics found within the LibriTTS dataset. Consequently, the majority of the experiments conducted in this study make use of the LibriTTS dataset due to its enhanced speaker variety and characteristics.

3.1.4 VoxCeleb

The VoxCeleb dataset is used to train the speaker encoder models. It has two subsets: VoxCeleb1 and VoxCeleb2. Together, they contain untranscribed speech at a 16 kHz sample rate for 7000+ speakers, 1 million+ utterances, and 2000+ hours. After combining VoxCeleb1 and VoxCeleb2, the speaker encoder models were trained on data for 7205 speakers. VoxCeleb1 test data is used for selecting the best model.

3.2 Zero-Shot Multi-Speaker TTS Experiments

A step-wise approach is followed to explore various aspects of zero-shot multi-speaker TTS. At every step, possible implementation approaches for an aspect are compared, and the best one is selected before moving on to the next step.

3.2.1 Baseline

The baseline model has the following configuration:

1. Mel-spectrogram prediction model: Tacotron2
2. Speaker embedding model: ECAPA-TDNN
3. Speaker embedding injection method: Average
4. Speaker embedding injection location: Speaker embeddings averaged with the output of the Tacotron2 encoder before being passed to the attention layer

5. Dataset: LibriTTS with splits: train (train-clean-100, train-clean-360), validation (dev-clean)

6. Sample rate for the mel-spectrogram prediction model: 22050 Hz

7. Sample rate for the speaker embedding model: 16000 Hz

Following prior work [2], transfer learning is used to accelerate learning. First, a model is trained using LJSpeech for 91,500 steps. This model is used as the starting point for multi-speaker training using the LibriTTS dataset for 294,000 steps. This approach is used for all multi-speaker training experiments, unless specified otherwise.

3.2.2 Comparing Speaker Embedding Injection Methods

This step examines the impact of three speaker embedding injection methods: average, concatenation, and FiLM. The rest of the configuration is the same as the baseline.

3.2.3 Comparing Speaker Embedding Models

The best model from the previous step is used to compare the performance with the X-Vector and ECAPA-TDNN speaker embedding models.

3.2.4 Comparing Speaker Embedding Injection Locations

The potential locations for injecting speaker embeddings in a mel-spectrogram prediction model vary with the model architecture. This step has two stages. The first stage is model-specific and evaluates injecting speaker embeddings at one or more locations for a particular model. Two models for mel-spectrogram prediction are considered here: Tacotron2 and FastSpeech2. The second stage compares the observations for the two models.

For Tacotron2, the following combinations of injection locations are tried, as shown in figure 14:

1. Base: Speaker embeddings injected into the output of the Tacotron2 encoder before being passed to the attention layer.

2. DecPrenet: Base + Speaker embeddings injected into the input to the decoder prenet

3. EncDec: DecPrenet + Speaker embeddings injected into the encoder input

4. All: EncDec + Speaker embeddings injected into the postnet input

For FastSpeech2, the following combinations of injection locations are tried, as shown in figure 15:

1. Base: Speaker embeddings injected into the input to the encoder

2. Preds: Base + Speaker embeddings injected into the duration, pitch, and energy predictor inputs

3. Dec: Preds + Speaker embeddings injected into the decoder input
4. All: Dec + Speaker embeddings injected into the postnet input
5. BasePost: Base + Speaker embeddings injected into the postnet input

3.2.5 Comparing the Best Model with YourTTS

The performance of the best model configuration obtained from the previous step is compared with that of YourTTS, which achieved state-of-the-art performance for zero-shot multi-speaker TTS on the VCTK dataset.

3.2.6 Random Speaker Generation

A GMM is trained to model the speaker embedding distribution and make the speaker embedding space more continuous. Random speaker embeddings are sampled using this GMM to generate new fictitious voices. The speaker embeddings used to train the GMM are computed for the LibriTTS dev-clean dataset using the best speaker encoder model from previous experiments.

Appendix E provides details about important training hyperparameters for the best zero-shot multi-speaker model configuration obtained from the experiments.

4 Experiment Results

This section discusses the evaluation metrics, evaluation data, and results of the zero-shot multi-speaker TTS experiments.

4.1 Evaluation Metrics

The naturalness of speech and speaker similarity between the ground truth and synthesized speech are important qualities when evaluating a zero-shot multi-speaker TTS system. Following prior work [2, 24], the experiment results are examined using objective and subjective metrics. Speaker Embedding Cosine Similarity (SECS) is the objective metric used to evaluate and compare speaker similarity at various stages, along with informal listening tests to verify the intelligibility of the speech. The best configuration achieved from these experiments is evaluated using the Mean Opinion Score (MOS) obtained from conducting listening tests for naturalness and speaker similarity.

4.1.1 Objective Evaluation Metric

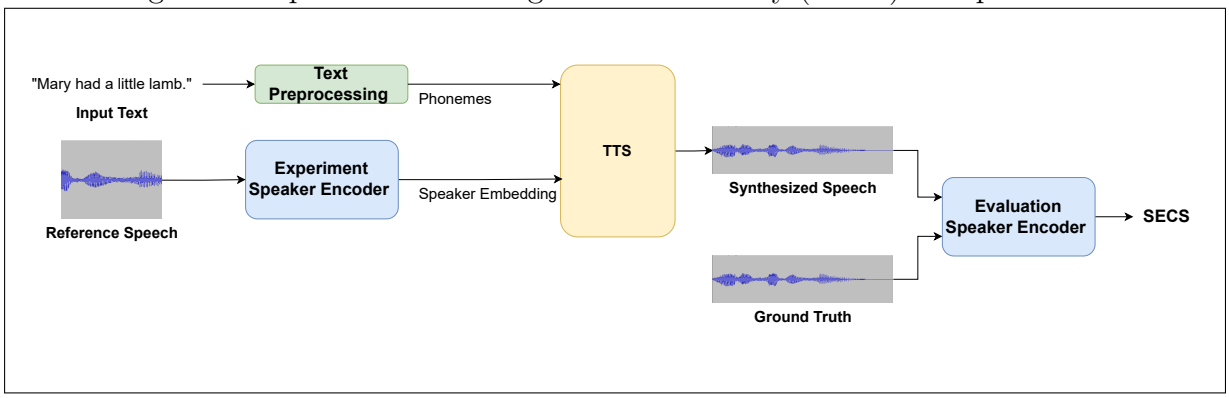
Speaker Embedding Cosine Similarity (SECS) is a standard measure for speaker similarity. It is defined as follows:

$$cosine\_similarity(X1, X2) = \frac{X1 \cdot X2}{||X1|| \cdot ||X2||}$$

The cosine similarity scale runs from -1 to 1, with a higher value indicating higher similarity between X1 and X2.

For objective evaluation of experiments, speaker embeddings are extracted for ground truth and synthesized speech. Cosine similarity is computed using an ECAPA-TDNN speaker encoder. Figure 20 shows the SECS computation workflow for the experiments. Since the evaluation speaker encoder is used on the final synthesized audio, it is not biased towards the speaker encoder used to extract speaker embeddings when training the TTS system.

Figure 20: Speaker Embedding Cosine Similarity (SECS) Computation



4.1.2 Subjective Evaluation Metrics

Subjective listening tests involving 21 participants are conducted to obtain the Mean Opinion Score (MOS) regarding the naturalness of synthesized speech and its speaker

similarity towards the ground truth. For the results presented, the term "MOS" refers to the naturalness score, and "Sim-MOS" refers to the speaker similarity score. As used in prior work [2, 23, 24], the values for the scores range from 1 to 5 and are explained using Table 3 and Table 4. All subjective scores are reported with a 95% confidence interval.

Table 3: Mean Opinion Score (MOS) - Mapping between Numerical Rating and Speech Naturalness

Numerical Rating	**Speech Naturalness**
5	Excellent
4	Good
3	Fair
2	Poor
1	Bad

Table 4: Similarity Mean Opinion Score (Sim-MOS) - Mapping between Numerical Rating and Speaker Similarity

Numerical Rating	**Speaker Similarity**
5	Extremely similar
4	Very similar
3	Moderately similar
2	Slightly similar
1	Not at all similar

4.2 Evaluation Data

This section discusses the factors considered while constructing the evaluation datasets and the two evaluation datasets used for the experiments.

4.2.1 Considerations for Evaluation Dataset Construction

A productive zero-shot multi-speaker TTS system should be able to perform well under the following conditions:

1. Seen speaker, unseen text
2. Unseen speaker, unseen text

Evaluating a model for these two conditions is a way to ensure that it generalizes well to unseen speakers and text without forgetting the speaker identities it has seen during training. The LibriTTS evaluation dataset, described below, is used for this purpose where applicable. When selecting the best model at every stage of the experiments, Speaker Embedding Cosine Similarity (SECS) scores are computed for these two conditions and their average, and the following priority order is used to select the best model:

1. Best average score
2. Best score for seen speakers, unseen text
3. Best score for unseen speakers, unseen text

Additionally, the performance of the best model obtained from the experiments is compared against that of the YourTTS model on the VCTK dataset. YourTTS has achieved state-of-the-art performance for zero-shot multi-speaker TTS on the VCTK dataset [2]. The VCTK test data from YourTTS experiments is used for this comparison. This data has samples only for the "unseen speaker, unseen text" condition.

The sample rate of the audio in both datasets is matched with the sample rate of the TTS experiment being evaluated to be as fair as possible during evaluation.

4.2.2 LibriTTS Evaluation Dataset

The LibriTTS evaluation dataset has a total of 100 text-utterance pairs as ground truth data. It has two subsets for the two evaluation conditions discussed in Section 4.2.1:

1. Seen speakers, unseen text
2. Unseen speakers, unseen text

During training, utterances with a duration greater than 10.10 seconds were excluded for training efficiency. Random speakers and their text-utterance pairs are sampled from this excluded data to construct evaluation data for the "seen speaker, unseen text" condition.

For the "unseen speaker, unseen text" condition, speaker data from the LibriTTS test-clean subset is used.

The subset for each condition has data for 5 male and 5 female speakers. For every speaker, a reference utterance is used to compute the reference speaker embedding, and 5 other text-utterance pairs are used as the ground truth. The reference speaker embedding is used as the input along with the text from ground truth text-utterance pairs to synthesize speech for a particular speaker identity. This synthesized speech is then evaluated against the corresponding ground truth utterance.

4.2.3 VCTK Evaluation Dataset

The VCTK evaluation dataset is constructed using VCTK test data from YourTTS, and it has data only for the "unseen speaker, unseen text" condition. It has data for 11 speakers (4 male, 7 female) representing different accents available in the VCTK dataset. The speaker IDs reserved for the evaluation dataset are 225, 234, 238, 245, 248, 261, 294, 302, 326, 335, and 347. Similar to the LibriTTS evaluation dataset, for every speaker, a reference utterance is used to compute the reference speaker embedding, and 5 other text-utterance pairs are used as the ground truth for SECS computation. Out of the 5 text-utterance pairs available for every speaker, 2 pairs are used for both MOS and Sim-MOS computation.

Appendix B lists more details about the speaker data used for constructing LibriTTS and VCTK evaluation datasets.

4.3 Results

The following subsections present results for a step-by-step examination of different factors of zero-shot multi-speaker TTS. Next, a comparison is made between the best model and YourTTS. Finally, an evaluation is conducted to assess the naturalness of speech using randomly generated speaker identities.

4.3.1 Baseline

Table 5: LibriTTS SECS Scores for Baseline

Experiment	**Seen Speaker, Unseen Text**	**Unseen Speaker, Unseen Text**	**Average**
Ground Truth Waveform	**1.000**	**1.000**	**1.000**
Ground Truth Mel-Spectrogram + Vocoder	0.985	0.984	0.984
Baseline	0.466	0.403	0.434

Table 5 compares the LibriTTS SECS (Speaker Embedding Cosine Similarity) scores for ground truth waveform, ground truth mel-spectrogram with vocoder, and the baseline for the zero-shot multi-speaker TTS experiments.

The first row of Table 5 contains cosine similarity scores of ground truth waveforms with respect to themselves. Predictably, these cosine similarity scores have the highest possible value of 1.

The second row of Table 5 contains the LibriTTS SECS scores for ground truth mel-spectrogram with a vocoder. For this, mel-spectrograms are computed for the ground truth waveform and passed through a vocoder to reconstruct the waveform. The reconstructed waveforms have a very high cosine similarity score with respect to the ground truth waveform. An informal listening test confirms that the reconstructed waveforms sound natural and very similar to the ground truth ones. Thus, speaker embeddings are not injected into the HiFi-GAN vocoder used in the experiments. This observation aligns with the findings of HiFi-GAN regarding the generalization capability of HiFi-GAN for unseen speakers [18].

The final row of Table 5 presents the LibriTTS SECS scores for the baseline model, where ECAPA-TDNN speaker embeddings are averaged with the Tacotron2 encoder output before passing it to the attention layer. Notably, the cosine similarity scores are higher for seen speakers compared to unseen speakers. This discrepancy is reasonable, as the model has been trained on numerous examples from seen speakers, allowing it to develop a stronger familiarity with their characteristics. It is expected that the model's

performance on seen speakers will be better than that on unseen speakers across the experiment results discussed in this work.

The findings from an informal listening test support these observations. The synthesized voices of seen speakers exhibit greater consistency, while those of unseen speakers display more variation across different utterances. Occasionally, there are voice modulations observed even within the same utterance. However, it is important to note that the LibriTTS corpus, derived from reading audiobooks, inherently includes voice modulation within utterances as speakers try to effectively convey different parts of the spoken text during storytelling. Overall, the model strives to produce intelligible speech, and there are no instances of completely unintelligible output, such as noise, during the evaluation.

4.3.2 Comparing Speaker Embedding Injection Methods

Table 6: LibriTTS SECS Scores Comparing Speaker Embedding Injection Methods

Experiment - Injection Method	**Seen Speaker, Unseen Text**	**Unseen Speaker, Unseen Text**	**Average**
Average (Baseline)	0.466	0.403	0.434
Concatenation	0.487	**0.428**	0.458
FiLM	**0.503**	0.424	**0.463**

The LibriTTS SECS scores in Table 6 demonstrate that the FiLM method outperforms the traditional ways of averaging and concatenating the speaker embeddings to inject them into the TTS system. Even though concatenating speaker embedding provides an SECS score better than that of FiLM for unseen speakers, its average score is lower than that of FiLM, and its score for seen speakers is worse compared to FiLM. The FiLM method is selected here considering the speaker similarity metric because it shows better generalization without forgetting the speaker identities seen during training, as discussed in 4.2.1. The better performance of FiLM could be attributed to the adaptive modulation capability it brings to the zero-shot multi-speaker TTS.

All experiments from this point on use FiLM as the injection method.

4.3.3 Comparing Speaker Embedding Models

To compare the speaker embedding models, Tacotron2 is combined with both the X-Vector and ECAPA-TDNN speaker embedding models. The independently trained X-Vector model achieves an error rate of 2.51% on the VoxCeleb1 test data, while the independently trained ECAPA-TDNN model achieves an error rate of 1.58% on the same dataset.

Table 7: LibriTTS SECS Scores Comparing Speaker Embedding Models

Experiment - Embedding Model	**Seen Speaker, Unseen Text**	**Unseen Speaker, Unseen Text**	**Average**
X-Vector	0.378	0.260	0.319
ECAPA-TDNN	**0.503**	**0.424**	**0.463**

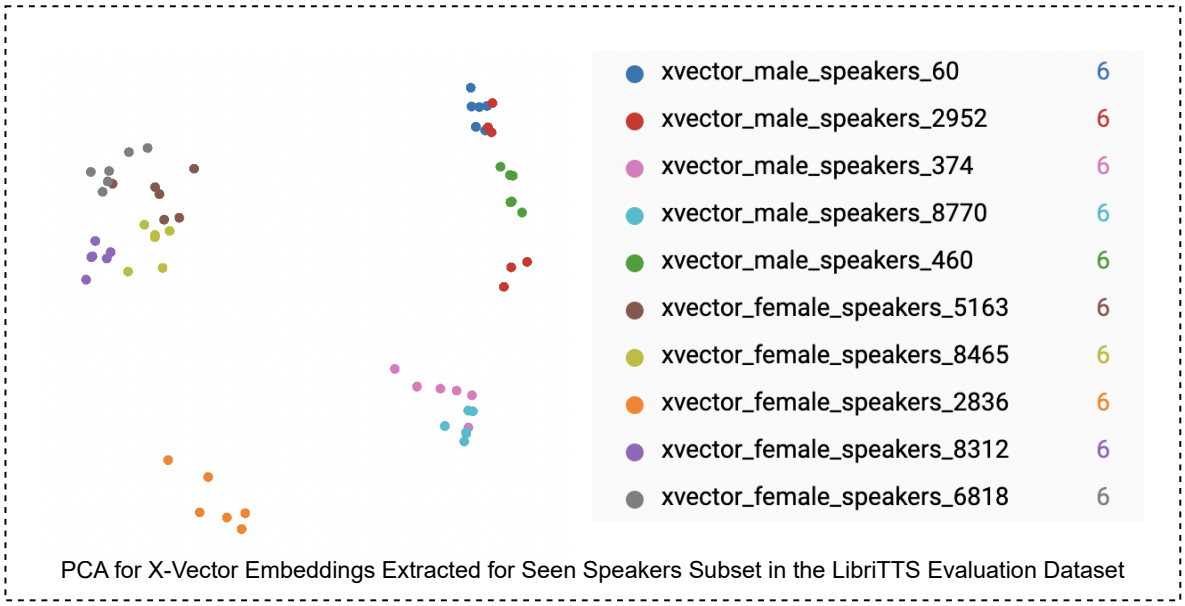
Based on the results of LibriTTS SECS (Speaker Encoding Similarity) presented in Table 7, it is evident that ECAPA-TDNN speaker embeddings surpass X-Vector speaker embeddings by a substantial margin. This can be attributed to the advancements introduced by the ECAPA-TDNN model over the TDNN architecture. The change in performance when changing only the speaker embedding model highlights the importance of good speaker embeddings in zero-shot multi-speaker TTS.

The effectiveness of the speaker encoding models can be further understood by examining the organization of speaker embeddings in the speaker embedding space. A good speaker encoder clusters embeddings from the utterances of the same speaker together while ensuring that embeddings from different speakers are separated into distinct clusters. Principal Component Analysis (PCA) visualization is used to illustrate this in Figure 21 and Figure 22. Figure 21 presents a comparison of the speaker embeddings extracted from the seen speakers of the LibriTTS evaluation dataset using both the X-Vector and ECAPA-TDNN speaker embedding models. Similarly, Figure 22 offers a comparison for the subset of unseen speakers from the LibriTTS evaluation dataset. In the visualizations, ECAPA-TDNN model exhibits superior capability in capturing speaker identities compared to the X-Vector model. These observations align with the error rates achieved by the models on the VoxCeleb1 test data and the LibriTTS SECS scores provided in Table 7.

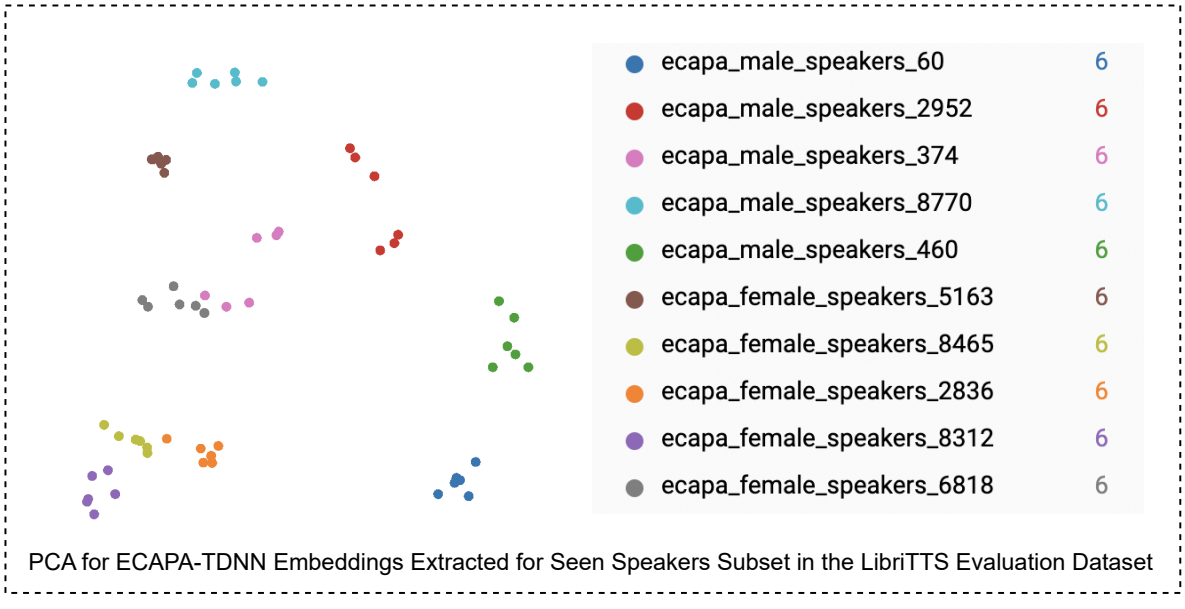
The results of this comparison step are consistent with the recent findings of Xue et al. [25] where ECAPA-TDNN performed better than X-Vector in the context of multi-speaker TTS using FastSpeech2 as the mel-spectrogram prediction network.

All experiments after this step use ECAPA-TDNN embeddings.

Figure 21: PCA Visualization for Seen Speaker Examples in LibriTTS Evaluation Data Using X-Vector and ECAPA-TDNN Models

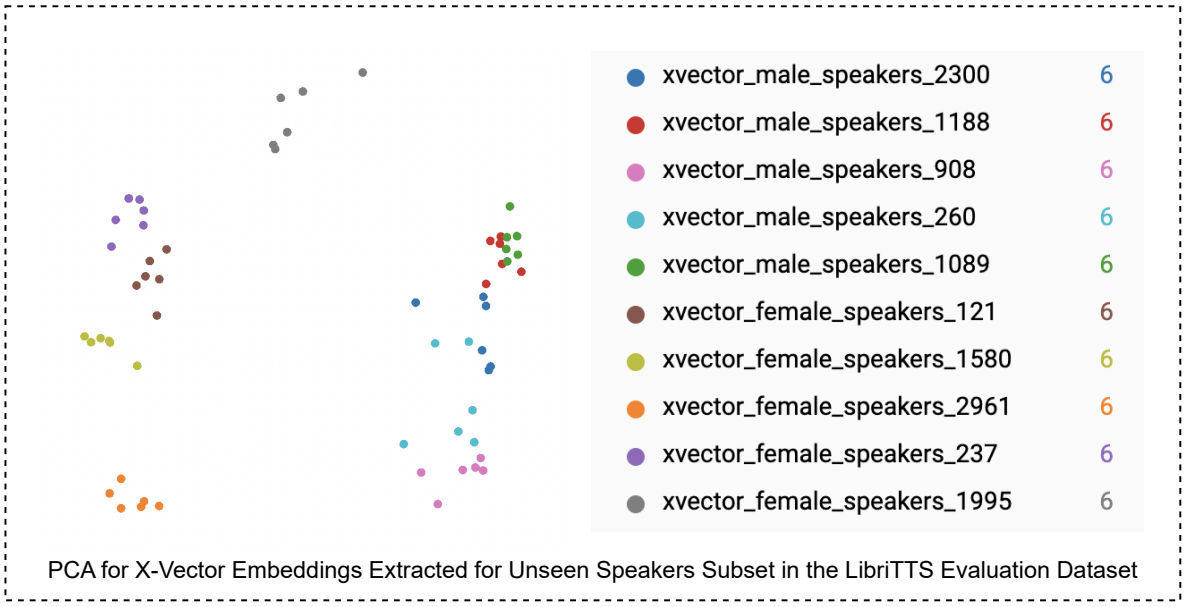


PCA for X-Vector Embeddings Extracted for Seen Speakers Subset in the LibriTTS Evaluation Dataset

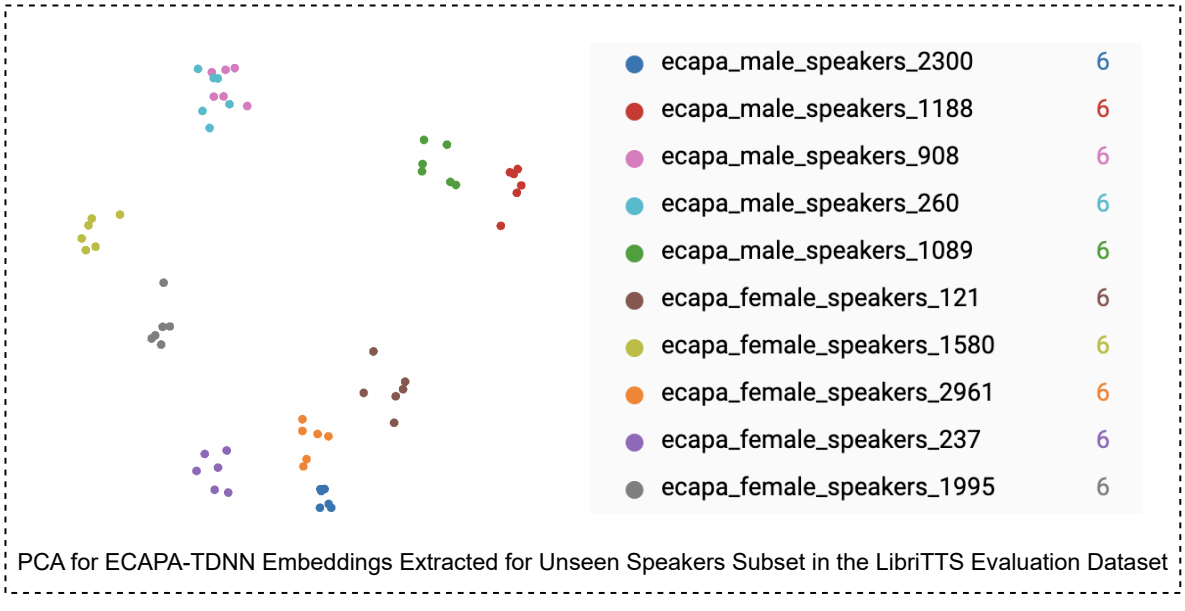


PCA for ECAPA-TDNN Embeddings Extracted for Seen Speakers Subset in the LibriTTS Evaluation Dataset

Figure 22: PCA Visualization for Unseen Speaker Examples in LibriTTS Evaluation Data Using X-Vector and ECAPA-TDNN Models



PCA for X-Vector Embeddings Extracted for Unseen Speakers Subset in the LibriTTS Evaluation Dataset



PCA for ECAPA-TDNN Embeddings Extracted for Unseen Speakers Subset in the LibriTTS Evaluation Dataset

4.3.4 Comparing Speaker Embedding Injection Locations

Two mel-spectrogram prediction models are used for this step: Tacotron2 and FastSpeech2. First, a model-specific comparison is performed to determine the best possible configuration of speaker embedding injection locations for that model. Afterward, the observations for Tacotron2 and FastSpeech2 are compared.

Table 8: LibriTTS SECS Comparing Speaker Embedding Injection Locations for Tacotron2

Experiment - Injection Locations for Tacotron2	**Seen Speaker, Unseen Text**	**Unseen Speaker, Unseen Text**	**Average**
Base	**0.503**	**0.424**	**0.463**
DecPrenet	0.459	0.372	0.416
EncDec	0.424	0.325	0.374
All	0.399	0.352	0.376

Table 8 shows the LibriTTS SECS scores for different speaker embedding injection location configurations for Tacotron2. The base case (Table 8, first row) provides the best performance. In the base case, speaker embeddings are injected in only one location; the FiLM method is used to inject the speaker embeddings into the output of the Tacotron2 encoder before being passed to the attention layer. Every row following the first row of Table 8 refers to an experiment where one additional location is used for speaker embedding injection. It can be seen that using additional locations for speaker embedding injection results in a performance decrease. When comparing the "EncDec" and "All" experiments, the average performance seems to improve with the additional injection location. However, it should be noted that, in this case, there is an increase in the unseen speaker performance at the expense of a decrease in the seen speaker performance on a similar scale. This is not desirable. Thus, no other combinations of speaker embedding injection locations with the base case are tried. Due to the overall negative impact seen after using locations other than the base case location, experiments using only those locations are not tried.

These results conform to the work of Jia et al. [23] where it is mentioned that injecting embeddings only into the input to the attention layer of Tacotron2 is sufficient to converge for different speakers, in contrast to the multi-location injection approach used in Deep Voice 2 [22]. However, Jia et al. [23] do not provide any studies comparing the effects of using one or more speaker embedding injection locations.

Table 9: LibriTTS SECS Comparing Speaker Embedding Injection Locations for FastSpeech2

Experiment - Injection Locations for FastSpeech2	**Seen Speaker, Unseen Text**	**Unseen Speaker, Unseen Text**	**Average**
Base	**0.287**	0.248	**0.268**
Preds	0.285	**0.249**	0.267
Dec	0.226	0.183	0.204
All	0.224	0.190	0.207
BasePost	0.283	**0.249**	0.266

The LibriTTS SECS scores noted in Table 9 for FastSpeech2 show that the experiment with the best average generalization without forgetting the speaker identities seen during training is the base case. In the base case, the speaker embeddings are injected only into the input to the encoder. The results for FastSpeech2 follow a trend similar to the one observed for Tacotron2 in Table 8.

For both models considered for this step, injecting speaker embeddings in only one place resulted in the best performance. However, this aspect may be model-specific, as Cooper et al. [24] conducted a similar study for an adapted version of the Tacotron model and found that injecting speaker embeddings into the encoder output and decoder prenet input resulted in the best outcome. In recent developments, YourTTS [2] achieved state-of-the-art results for zero-shot multi-speaker TTS on the VCTK dataset with a model architecture where speaker embeddings are injected into numerous locations.

Table 10: Best LibriTTS SECS for Tacotron2 and FastSpeech2

Experiment - Model	**Seen Speaker, Unseen Text**	**Unseen Speaker, Unseen Text**	**Average**
Tacotron2 - Base	**0.503**	**0.424**	**0.463**
FastSpeech2 - Base	0.287	0.248	0.268

Finally, Table 10 lists the best LibriTTS SECS scores achieved with multi-speaker Tacotron2 and FastSpeech2. It should be noted that this comparison is performed after training both models for the same number of epochs. It is not unexpected for different models to require different amounts of training for the same task on the same dataset. Thus, the performance of FastSpeech2 may improve with more training. Another factor to consider is the model capacity. Perhaps increasing the model capacity by changing the number of layers and their dimensionality could help improve the performance of both Tacotron2 and FastSpeech2. In this step, Tacotron2 is selected because of its higher scores.

The best model configuration from all the steps so far consists of ECAPA-TDNN embeddings injected into the Tacotron2 model using the FiLM method. This model configuration is referred to as zero-shot multi-speaker Tacotron2 from this point on. Appendix E provides important details about the training hyperparameters for the zero-shot multi-speaker Tacotron2 model.

4.3.5 Comparing the Best Model with YourTTS for VCTK

The performance of the zero-shot multi-speaker Tacotron2 model is compared to that of the YourTTS model, which achieved state-of-the-art performance for the VCTK dataset for zero-shot multi-speaker TTS [2]. A pretrained YourTTS model checkpoint trained on the VCTK dataset is used to generate the evaluation samples for YourTTS. YourTTS uses a sample rate of 16000 Hz. In order to be compatible with these YourTTS training factors, the zero-shot multi-speaker Tacotron2 model is trained on the VCTK dataset using the same data split and sample rate as YourTTS. The training is carried out for 2,644,040

steps to further improve the performance of the zero-shot multi-speaker Tacotron2 model (compared to the 294,000 training steps for the previous experiments). Informal listening tests were carried out throughout training to ensure the output quality was improving with an increasing number of steps.

Table 11: Comparing Zero-Shot Multi-Speaker Tacotron2 with YourTTS using VCTK Evaluation Dataset

Experiment	**SECS**	**MOS**	**Sim-MOS**
Ground Truth	1.000	**3.91±0.14**	NA
Zero-Shot Multi-Speaker Tacotron2	0.378	3.46±0.10	3.25±0.18
YourTTS	**0.498**	3.89±0.12	**3.66±0.16**

Table 11 shows that zero-shot multi-speaker Tacotron2 does not match the performance of YourTTS in terms of speech naturalness and speaker similarity. The speech naturalness expressed by subjective MOS results is the highest for the ground truth, closely followed by YourTTS. The objective SECS and subjective Sim-MOS results for speaker similarity complement each other in establishing YourTTS as the better model.

Closely observing the architectures of the two models being compared shows that the zero-shot multi-speaker Tacotron2 model has approximately 28.6 million parameters, while the number of parameters used for YourTTS is around 86.6 million. This indicates a greater capacity available for YourTTS, which could be a contributing factor to its superior performance. Increasing the capacity of the Tacotron2 by modifying the model architecture and increasing the dimensionality could potentially lead to better performance. Another factor to note here is the amount of training. The multi-speaker Tacotron2 model used for this comparison is trained for 2,644,040 steps. Further training the model could improve its performance in terms of pronunciation clarity, pace, robotic voice, etc., leading to the generation of natural speech with greater speaker similarity. Additionally, as shown in Table 7, the speaker embedding model can also make a difference for the speaker similarity of the generated speech. YourTTS uses the H/ASP model [43] as the speaker encoder, while the zero-shot multi-speaker Tacotron2 model uses ECAPA-TDNN speaker encoder. This is another aspect to consider when trying to understand the performance difference. Overall this examination indicates that even though the current configuration of zero-shot multi-speaker Tacotron2 does not meet the performance of the YourTTS model, there is potential for improvement.

4.3.6 Random Speaker Generation

Table 12: MOS for Random Speaker Generation Using VCTK Evaluation Data

Speaker Type	**MOS**
Real Speaker	**3.46±0.10**
Fictitious Speaker	3.45±0.18

Table 12 shows subjective naturalness MOS results for speech synthesized for real and fictitious speakers. The zero-shot multi-speaker Tacotron2 model is used to generate speech in both cases. For real speakers, samples available for MOS computation from the VCTK evaluation dataset are used to compute speaker embeddings. For fictitious speakers, a GMM is trained to model the distribution of ECAPA-TDNN speaker embeddings computed using the LibriTTS dev-clean dataset. The resultant speaker embedding space is used to sample random data points. The naturalness of the speech generated for fictitious speakers is comparable to that of real speakers. This promising result suggests that the zero-shot multi-speaker TTS system has the potential to successfully generate speech with realistic fictitious speaker voices using this approach. As the model's performance improves in terms of naturalness for real speakers, a corresponding enhancement in its performance for fictitious speakers can be expected.

5 Conclusion

In conclusion, this thesis comprehensively examines various aspects of zero-shot multi-speaker TTS to establish an effective system. The results underscore the significance of the speaker embedding injection method while adapting the FiLM method for zero-shot multi-speaker TTS. While the FiLM method currently employs simple linear transformations, future work could explore more sophisticated neural networks to enhance the model's performance. Furthermore, the study highlights the crucial role of high-performing speaker embeddings by demonstrating the impact of different speaker embedding models on the system's performance. Although injecting embeddings in a single location yields the best results in the models investigated here, outperforming any combination of multiple injection locations, it is important to note that this aspect may be sensitive to the model architecture. While the best-performing model obtained from the experiments does not reach the state-of-the-art in its current stage, there is potential for improvement by exploring variations of capacity, the amount of training, and speaker embeddings. Finally, the random speaker generator module successfully extends the capabilities of the zero-shot multi-speaker system to generate fictitious voices. While this thesis provides some valuable insights into zero-shot multi-speaker TTS, there are unexplored pathways that could lead to future advancements in the field. For instance, improving the objective functions by incorporating a term for speaker consistency and exploring controllable factors to modulate generated voices can significantly enhance the system. In fact, the knowledge gained from this study on the multi-speaker aspect can be applied to other facets of TTS, such as emotion, accent, and style, paving the way towards expressive TTS.

References

[1] B. Desplanques, J. Thienpondt, and K. Demuynck, "Ecapa-tdnn: Emphasized channel attention, propagation and aggregation in tdnn based speaker verification," *arXiv preprint arXiv:2005.07143*, 2020.

[2] E. Casanova, J. Weber, C. D. Shulby, A. C. Junior, E. Gölge, and M. A. Ponti, "Yourtts: Towards zero-shot multi-speaker tts and zero-shot voice conversion for everyone," in *International Conference on Machine Learning*. PMLR, 2022, pp. 2709–2720.

[3] E. Casanova, C. Shulby, A. Korolev, A. C. Junior, A. d. S. Soares, S. Aluísio, and M. A. Ponti, "Asr data augmentation in low-resource settings using cross-lingual multi-speaker tts and cross-lingual voice conversion," *arXiv preprint arXiv:2204.00618*, 2022.

[4] S. Ueno, M. Mimura, S. Sakai, and T. Kawahara, "Multi-speaker sequence-to-sequence speech synthesis for data augmentation in acoustic-to-word speech recognition," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6161–6165.

[5] P. Taylor, *Text-to-Speech Synthesis*, 1st ed. USA: Cambridge University Press, 2009.

[6] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. 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)*. IEEE, 2018, pp. 4779–4783.

[7] A. J. Hunt and A. W. Black, "Unit selection in a concatenative speech synthesis system using a large speech database," in *1996 IEEE international conference on acoustics, speech, and signal processing conference proceedings*, vol. 1. IEEE, 1996, pp. 373–376.

[8] A. W. Black, H. Zen, and K. Tokuda, "Statistical parametric speech synthesis," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, vol. 4. IEEE, 2007, pp. IV–1229.

[9] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio *et al.*, "Tacotron: Towards end-to-end speech synthesis," *arXiv preprint arXiv:1703.10135*, 2017.

[10] W. Ping, K. Peng, A. Gibiansky, S. O. Arik, A. Kannan, S. Narang, J. Raiman, and J. Miller, "Deep voice 3: Scaling text-to-speech with convolutional sequence learning," *arXiv preprint arXiv:1710.07654*, 2017.

[11] Y. Lee, J. Shin, and K. Jung, "Bidirectional variational inference for non-autoregressive text-to-speech," in *International conference on learning representations*, 2022.

[12] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[13] R. Prenger, R. Valle, and B. Catanzaro, "Waveglow: A flow-based generative network for speech synthesis," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3617–3621.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[15] Y. Ren, C. Hu, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "Fastspeech 2: Fast and high-quality end-to-end text to speech," *arXiv preprint arXiv:2006.04558*, 2020.

[16] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, "Neural speech synthesis with transformer network," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 6706–6713.

[17] M. Jeong, H. Kim, S. J. Cheon, B. J. Choi, and N. S. Kim, "Diff-tts: A denoising diffusion model for text-to-speech," *arXiv preprint arXiv:2104.01409*, 2021.

[18] J. Kong, J. Kim, and J. Bae, "Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 022–17 033, 2020.

[19] J. Kim, J. Kong, and J. Son, "Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech," in *International Conference on Machine Learning*. PMLR, 2021, pp. 5530–5540.

[20] Y. Guo, C. Du, X. Chen, and K. 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)*. IEEE, 2023, pp. 1–5.

[21] T. Nekvinda and O. Dušek, "One model, many languages: Meta-learning for multilingual text-to-speech," *arXiv preprint arXiv:2008.00768*, 2020.

[22] A. Gibiansky, S. Arik, G. Diamos, J. Miller, K. Peng, W. Ping, J. Raiman, and Y. Zhou, "Deep voice 2: Multi-speaker neural text-to-speech," *Advances in neural information processing systems*, vol. 30, 2017.

[23] Y. Jia, Y. Zhang, R. Weiss, Q. Wang, J. Shen, F. Ren, P. Nguyen, R. Pang, I. Lopez Moreno, Y. Wu *et al.*, "Transfer learning from speaker verification to multispeaker text-to-speech synthesis," *Advances in neural information processing systems*, vol. 31, 2018.

[24] E. Cooper, C.-I. Lai, Y. Yasuda, F. Fang, X. Wang, N. Chen, and J. Yamagishi, "Zero-shot multi-speaker text-to-speech with state-of-the-art neural speaker embeddings," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6184–6188.

[25] J. Xue, Y. Deng, Y. Han, Y. Li, J. Sun, and J. Liang, "Ecapa-tdnn for multi-speaker text-to-speech synthesis," in *2022 13th International Symposium on Chinese Spoken Language Processing (ISCSLP)*. IEEE, 2022, pp. 230–234.

[26] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Sixteenth annual conference of the international speech communication association*, 2015.

[27] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2018, pp. 5329–5333.

[28] H. Yoon, C. Kim, S. Um, H.-W. Yoon, and H.-G. Kang, "Sc-cnn: Effective speaker conditioning method for zero-shot multi-speaker text-to-speech systems," *IEEE Signal Processing Letters*, 2023.

[29] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[30] M. Ravanelli, T. Parcollet, P. Plantinga, A. Rouhe, S. Cornell, L. Lugosch, C. Subakan, N. Dawalatabad, A. Heba, J. Zhong, J.-C. Chou, S.-L. Yeh, S.-W. Fu, C.-F. Liao, E. Rastorgueva, F. Grondin, W. Aris, H. Na, Y. Gao, R. D. Mori, and Y. Bengio, "SpeechBrain: A general-purpose speech toolkit," 2021, arXiv:2106.04624.

[31] A. Ploujnikov and M. Ravanelli, "Soundchoice: Grapheme-to-phoneme models with semantic disambiguation," *arXiv preprint arXiv:2207.13703*, 2022.

[32] S. Becker, M. Ackermann, S. Lapuschkin, K.-R. Müller, and W. Samek, "Interpreting and explaining deep neural networks for classification of audio signals," *CoRR*, vol. abs/1807.03418, 2018.

[33] Y.-Y. Yang, M. Hira, Z. Ni, A. Chourdia, A. Astafurov, C. Chen, C.-F. Yeh, C. Puhrsch, D. Pollack, D. Genzel, D. Greenberg, E. Z. Yang, J. Lian, J. Mahadeokar, J. Hwang, J. Chen, P. Goldsborough, P. Roy, S. Narenthiran, S. Watanabe, S. Chintala, V. Quenneville-Bélair, and Y. Shi, "Torchaudio: Building blocks for audio and speech processing," *arXiv preprint arXiv:2110.15018*, 2021.

[34] K. Okabe, T. Koshinaka, and K. Shinoda, "Attentive statistics pooling for deep speaker embedding," *arXiv preprint arXiv:1803.10963*, 2018.

[35] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4690–4699.

[36] X. Xiang, S. Wang, H. Huang, Y. Qian, and K. Yu, "Margin matters: Towards more discriminative deep neural network embeddings for speaker recognition," in *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2019, pp. 1652–1656.

[37] M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner, and M. Sonderegger, "Montreal Forced Aligner: Trainable Text-Speech Alignment Using Kaldi," in *Proc. Interspeech 2017*, 2017, pp. 498–502.

[38] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[39] K. Ito and L. Johnson, "The lj speech dataset," https://keithito.com/LJ-Speech-Dataset/, 2017.

[40] H. Zen, V. Dang, R. Clark, Y. Zhang, R. J. Weiss, Y. Jia, Z. Chen, and Y. Wu, "Libritts: A corpus derived from librispeech for text-to-speech," *arXiv preprint arXiv:1904.02882*, 2019.

[41] C. Veaux, J. Yamagishi, and K. MacDonald, "CSTR VCTK Corpus: English multi-speaker corpus for CSTR voice cloning toolkit (version 0.92)," 2019.

[42] A. Nagrani, J. S. Chung, W. Xie, and A. Zisserman, "Voxceleb: Large-scale speaker verification in the wild," *Computer Science and Language*, 2019.

[43] H. S. Heo, B.-J. Lee, J. Huh, and J. S. Chung, "Clova baseline system for the voxceleb speaker recognition challenge 2020," *arXiv preprint arXiv:2009.14153*, 2020.

[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

A Phoneme Set

Following is the set of phoneme tokens used for the experiments in this work:

Table 13: Phoneme Set Used for the Experiments

Common Phonemes Tokens			
AA	AE	AH	AO
AW	AY	B	CH
D	DH	EH	ER
EY	F	G	HH
IH	IY	JH	K
L	M	N	NG
OW	OY	P	R
S	SH	T	TH
UH	UW	V	W
Y	Z	ZH	
SoundChoice G2P Specific Tokens			
‘ ’ - Space token to separate words			
FastSpeech2 Specific Tokens			
spn - Silent phoneme token to represent silent periods in an utterance			

B Evaluation Speaker Sets

Table 14: Evaluation Speaker Set for LibriTTS

Speaker Type (Seen/Unseen)	**Gender**	**Speaker ID**
Seen	Female	2836
Seen	Female	5163
Seen	Female	6818
Seen	Female	8312
Seen	Female	8465
Seen	Male	60
Seen	Male	374
Seen	Male	460
Seen	Male	2952
Seen	Male	8770
Unseen	Female	121
Unseen	Female	237
Unseen	Female	1580
Unseen	Female	1995
Unseen	Female	2961
Unseen	Male	260
Unseen	Male	908
Unseen	Male	1089
Unseen	Male	1188
Unseen	Male	2300

Table 15: Evaluation Speaker Set for VCTK Sourced from YourTTS [2]

Speaker Type (Seen/Unseen)	**Gender**	**Speaker ID**
Unseen	Female	p225
Unseen	Female	p234
Unseen	Female	p238
Unseen	Female	p248
Unseen	Female	p261
Unseen	Female	p294
Unseen	Female	p335
Unseen	Male	p245
Unseen	Male	p302
Unseen	Male	p326
Unseen	Male	p347

C Audio Parameters

This section provides the parameter values used for computing mel-spectrogram and pitch using torchaudio. For any parameters expected by torchaudio but not mentioned here, default torchaudio values are used.

Table 16: Parameter Values for Mel-Spectrogram Computation Using Torchaudio

Parameter	**Value**
Sample Rate	Experiment Sample Rate (22050 Hz or 16000 Hz)
Hop Length	256
Window Size	1024
Size of FFT	1024
Number of Mel Filterbanks	80
Minimum Frequency	0.0
Maximum Frequency	8000.0
Power	1
Normalized	False
Norm	"slaney"
Mel Scale	"slaney"

Table 17: Parameter Values for Pitch Computation Using Torchaudio

Parameter	**Value**
Sample Rate	Experiment Sample Rate (22050 Hz or 16000 Hz)
Frame Length (in milliseconds)	$(Size\ of\ FFT / Sample\ Rate) * 1000$
Frame Shift (in milliseconds)	$(Hop\ Length\ / Sample\ Rate) * 1000$
Minimum F0 to search for (Hz)	65
Maximum F0 to search for (Hz)	2093

D Configuration Details for the Silent Phoneme Predictor

The silent phoneme predictor, introduced to enhance the speech synthesis of FastSpeech2, processes an input sequence using an encoder based on multiple feed-forward transformer blocks, followed by a linear layer and sigmoid activation.

The encoder architecture of the silent phoneme predictor is the same as the FastSpeech2 encoder, with the exception of masked attention being used to mask future elements. The encoder contains 4 feed-forward transformer blocks. Each block contains a multi-headed self-attention layer followed by a 2-layer 1D-convolution network. The number of attention heads is 2. The dimension of phoneme embedding and the hidden size of self-attention are set to 256. The first convolution layer uses a kernel size of 9, an input size of 256, and an output size of 1024. The second convolution layer uses kernel size 1, input size 1024, and output size 256. The linear layer maps input of size 256 to output of size 1.

E Important Training Hyperparameters for the Best Model Configuration

This section provides important training hyperparameters for the best model configuration. It is the zero-shot multi-speaker Tacotron2 model with ECAPA-TDNN embeddings injected using the FiLM method.

E.1 Optimization Hyperparameters

A batch size of 64 is used to train the model on an NVIDIA A100 GPU.

An initial learning rate of 0.001 is used to train the model on the single-speaker LJSpeech dataset. When fine-tuning the checkpoint obtained from the single-speaker training using a multi-speaker dataset, the initial learning rate is reduced to 0.0001. An interval scheduler is used for learning rate annealing in both cases. The interval scheduler changes the learning rate to specific values after a predefined number of steps. For single-speaker training, the learning rate values change to 0.0005, 0.0003, and 0.0001 after 6000, 8000, and 10000 steps. For multi-speaker training, the learning rate values change to 0.00005, 0.00003, and 0.00001 after 6000, 8000, and 10000 steps.

The optimizer used for training is Adam [44] with a weight decay of 0.000006.

E.2 Architecture Modifications for Injecting Speaker Embeddings

The original Tacotron2 [6] architecture implemented with SpeechBrain [30] is used with minimal modifications to inject the speaker embeddings using the FiLM method. The ECAPA-TDNN speaker embeddings with size 192 are used with the zero-shot multi-speaker Tacotron2 model. After applying FiLM transformations as shown in Section 2.7.3, this size becomes 512 to match the size of the Tacotron2 encoder output (512) where the speaker embeddings are injected.