SCHOOL OF SCIENCE AND ENGINEERING

# MULTI-SPEAKER TEXT-TO-SPEECH SYNTHESIZER

**Capstone Design Report**

April 2023

**Zakariae Jabbour**

Supervised by

**Dr. Violetta Cavalli-Sforza**

Capstone Report

**Student Statement:**

I, Zakariae Jabbour, applied ethics to the design process and in the selection of the final proposed design. I affirm that I have held the safety of the public to be paramount and have addressed this in the presented design wherever may be applicable.

[illegible]

Zakariae Jabbour

[illegible]

Approved by the Supervisor

[illegible]

Dr. Violetta Cavalli-Sforza

## Acknowledgments

I would like to take this opportunity to express my deep gratitude to my parents for their consistent support, compassion, encouragement, and concern during my academic career. This accomplishment would not have been possible without them and their effort in supporting me.

On top of that, I want to thank everyone who assisted with this project, whether directly or indirectly, no matter how small the contribution was.

Sincerely, thank you.

## List of Figures

**Figure 1:** MOS for the various TTS methods.

**Figure 2:** Example of a Mel-spectrogram.

**Figure 3:** The SV2TTS framework during inference.

**Figure 4:** The stages of training the model and how the parts interact.

**Figure 5:** The procedure of creating the similarity matrix during training.

**Figure 6:** Computing the embedding of a complete utterance.

**Figure 7:** Speaker verification EER and loss during training for the first 10k steps

**Figure 8:** Umap projection of utterance embeddings from randomly selected batches from the training dataset.

**Figure 9:** Altered Tacotron structure.

**Figure 10:** Concatenation of the speaker embedding and the output of the encoder.

## List of Acronyms and Abbreviations

**CNN:** Convolutional Neural Network

**DSP:** Digital Signal Processing

**EER:** Equal Error Rate

**FFT:** Fast Fourier Transform

**GE2E:** Generalized End-to-End

**GMM:** Gaussian Mixture Model

**HMM:** Hidden Markov Model

**LSTM:** Long Short-Term Memory

**MOS:** Means Opinion Score

**MPLG:** Maximum Likelihood Parameter Generation

**RNN:** Recurrent Neural Network

**STFT:** Short-Time Fourier Transform

**SV2TTS:** Speaker Verification to Text to Speech

**TTS:** Text-to-speech

**UMAP:** Uniform Manifold Approximation and Projection

# Table of Contents

**LIST OF FIGURES** .......... **3**

**LIST OF ACRONYMS AND ABBREVIATIONS** .......... **4**

**ABSTRACT** .......... **7**

**RESUME** .......... **8**

**1. INTRODUCTION** .......... **9**

**2. FEASIBILITY STUDY** .......... **11**

2.2 TECHNICAL FEASIBILITY .......... 11

2.3 ECONOMIC FEASIBILITY .......... 11

**3. LITERATURE REVIEW** .......... **12**

**4. TRANSFER LEARNING FROM SPEAKER VERIFICATION TO MULTISPACER TEXT-TO-SPEECH SYNTHESIS 15**

4.2 MEL-SPECTROGRAM .......... 15

4.3 MODEL OVERVIEW .......... 16

4.4 SPEAKER ENCODER .......... 19

*4.4.1 Model architecture .......... 19*

*4.4.2 Training and Dataset .......... 22*

*4.4.3 Results .......... 23*

4.5 SYNTHESIZE .......... 24

*4.5.1 Model Overview .......... 24*

*4.5.2 Implementation .......... 26*

*4.5.3 Data Pre-processing and Training .......... 27*

4.6 VOCODER .......... 29

**5. RESULTS** .......... **30**

**6. LIMITATIONS AND FUTURE WORK** .......... **31**

**7. STEEPLE ANALYSIS** .......... **32**

7.2 SOCIAL .......... 32

7.3 TECHNOLOGICAL .......... 32

7.4 ECONOMIC .......... 32

7.5 ENVIRONMENTAL .......... 32

7.6 POLITICAL .......... 32

7.7 LEGAL .......... 33

7.8 ETHICAL .......... 33

**8. INVESTIGATION ON RELATED ENGINEERING STANDARDS** .......... **34**

**9.CONCLUSIONS ........................................................................................................................36**

**REFERENCES ...........................................................................................................................37**

## Abstract

Deep learning has made considerable advances in the field of text-to-speech synthesis in recent years. A deep neural network is often trained using a huge dataset of professionally recorded speech from a single speaker, which makes giving the model a fresh voice expensive and time-consuming. Nevertheless, a recent study offered a three-stage model for cloning an unknown voice utilizing only a few seconds of reference speech and without the need for model retraining. The authorsreported impressive results in terms of natural-sounding speech.

This project’s objective is to implement this framework and try to generate speech audio in the voices of different speakers, including those unseen during training, effectively allowing us to imitate another voice based on a small sample of that person’s voice. The model was reasonably able to clone voices, albeit with some limitations like the length of the input sentence and the speed of speech in the sample.

***Keywords:*** *Deep Learning, Machine Learning, Neural Networks, Python, Text-to-Speech Synthesis.*

## Résumé

L'apprentissage profond a fait des progrès considérables dans le domaine de la synthèse vocale au cours des dernières années. Un réseau neuronal profond est souvent entraîné à l'aide d'un énorme ensemble de données de parole enregistrée professionnellement par un seul locuteur, ce qui rend l'introduction d'une nouvelle voix dans le modèle coûteuse et chronophage. Néanmoins, une étude récente a proposé un modèle en trois étapes pour cloner une voix inconnue en utilisant seulement quelques secondes de discours de référence et sans avoir besoin de réentraîner le modèle. Les auteurs ont fait état de résultats impressionnants en termes de sonorité naturelle de la parole.

Dans ce travail, nous essayons de mettre en œuvre ce cadre et de générer des sons vocaux avec les voix de différents locuteurs, y compris ceux qui n'ont pas été vus pendant la formation, ce qui nous permet d'imiter une autre voix sur la base d'un petit échantillon de la voix de cette personne. Le modèle était raisonnablement capable de cloner des voix, bien qu'avec certaines limitations comme la longueur de la phrase d'entrée et la vitesse de la parole dans l'échantillon.

***Mots-clés :*** *Apprentissage Profond, Apprentissage Automatique, Réseaux Neuronaux, Python, Synthèse Vocale.*

## 1. Introduction

Deep learning models are dominating the field of practical machine learning in many different applications. Text-to-speech, or TTS for short, is a method that generates synthetic voice from text input. It is like other methods, with one exception. Since their first appearance, deep models produce speech that sounds more realistic than that produced by traditional concatenative algorithms. Most current research efforts have been focused on improving the effectiveness of these deep models and making them sound more natural.

There is a severe lack of speech datasets that have been professionally recorded. Training data with comparable qualities are required to synthesize a realistic voice that is capable of exact pronunciation, dynamic intonation, and low levels of background noise. In addition, there is still a significant problem with the efficiency of the data in deep learning. In most cases, the training of a conventional text-to-speech model like Tacotron [11] will need a large amount of actual speech to be played back. However, the possibility of synthesizing speech using any human voice can be desired for a variety of reasons, including those that are just cosmetic as well as those that are beneficial. The results of many studies have led to the development of frameworks for voice conversion and voice cloning. Voice conversion is a sort of style transfer from one voice to another on a spoken segment, whereas voice cloning is the process of capturing features of the voice of a speaker and using them in a text-to-speech synthesizer on unseen text inputs.

The objective is to construct a fixed model that can take into account new voices using as little information as possible. The most common method is to teach a TTS model to generalize to new speakers by basing its training on an embedding of the voice that is to be cloned [20]. A speaker encoder model takes reference speech as its input in order to build a low-dimensional encoding for the speech. This technique is often more data-efficient than training a separate TTS model for each speaker. In addition, it is a great deal quicker and requires far less computing effort.

The proposed model has numerous applications, including the creation of a personalized voice assistant for individuals, enabling them to interact with technology using a familiar voice. It can also be used to produce multimedia content, such as videos, podcasts, and online courses, to provide narration or voiceovers. Furthermore, it can be utilized in digital marketing campaigns to deliver personalized messages to customers, such as voicemail and text messages. Entertainment applications, such as video games, virtual reality experiences, and interactive storytelling can also benefit from this technology. Lastly, voice cloning technology can be used to create a digital voiceprint of a person's voice, which can be used to synthesize speech after the person's death or loss of voice due to medical conditions.

# 2. FEASIBILITY STUDY

## 2.2 Technical Feasibility

Regarding the technical aspect, there are two components, the availability of data and hardware used to train the model. Regarding the data, there are many publicly accessible datasets, both with transcripts and without a transcript, to work with. We will be mainly working with LibriSpeech. Regarding the hardware we also have a couple of options: we can either rely on Google Collaboratory, the computer provided by the university for computationally demanding tasks, or I can use my personal computer since it has decent computational power. The model will be entirely implemented using Python as the main language to make use of the many libraries it provides for deep learning. It would also take inspiration from previous attempts at implementing the model and its components that are publicly available to use and modify by anyone since implementing everything from the ground up would not be feasible.

## 2.3 Economic Feasibility

Regarding the economic aspect, this project has minimum cost since it would rely on freely accessible datasets and open-source implementations, the only costs are the hardware and electricity costs required during the training. If the project is going to be successful, additional costs would be required to improve and finetune the model, design a user interface, and other requirements to make it a viable product in the market.

# 3. LITERATURE REVIEW

The history of TTS synthesis dates to the 1950s when researchers first began experimenting with speech synthesis. The first speech synthesizer called the "Vocoder," was developed in 1948 by Homer Dudley at Bell Labs. However, it was not until the 1960s and 1970s that digital signal processing (DSP) was introduced, leading to the development of more sophisticated TTS systems. One of the earliest TTS systems was the "Pattern Playback" developed in 1968 by the Artificial Intelligence Laboratory at MIT. Pattern Playback" TTS (text-to-speech) systems work by first analyzing the input text to determine the phonetic transcription of each word [1]. This is typically done using a combination of rule-based approaches and machine learning algorithms that have been trained on large datasets of spoken language.

The TTS system employs "concatenative synthesis" to produce speech after determining the phonetic transcriptions for each word [2]. The system chooses pre-recorded voice segments, referred to as "units," that match each phonetic transcription in this process, and then it combines them to produce the final output. The segments can be single phones, complete sentences, words, syllables, or even diphones. To build the desired sequence at runtime, the database's best chain of candidate units is chosen. The 1980s and 1990s saw a significant improvement in TTS synthesis technology through the development and improvement of rule-based TTS systems.

The 2000s witnessed further advancements in TTS synthesis technology, with the introduction of machine learning and artificial intelligence techniques. These systems used statistical models and neural networks to generate speech that sounded more natural.

To produce speech that sounds natural, statistical models are frequently used in modern voice synthesis technologies. To develop statistical models that can be utilized to produce new speech from written text, these models are trained on enormous datasets of recorded speech [4]. The Hidden Markov explain (HMM), one of the most used statistical models in speech synthesis, can explain the intricate relationships between various speech variables [3]. The Gaussian Mixture Model (GMM), another well-liked statistical model utilized in voice synthesis, accurately represents the distribution of acoustic characteristics in speech like **pitch**, **duration**, **spectral envelope**, and **formants** [7]. **Pitch** is the perceived fundamental frequency of a sound wave and is a crucial acoustic feature in speech. It is determined by the rate of vocal fold vibration and can vary across different speakers and contexts. **Duration** refers to the length of a speech sound and can vary depending on factors such as stress, emphasis, and intonation.

Longer durations are typically associated with stressed or emphasized syllables. The **spectral envelope** refers to the overall shape of the frequency spectrum of a speech sound and is determined by the vocal tract shape. It can be used to distinguish different phonemes. **Formants** are spectral peaks in the frequency spectrum of a speech sound and are determined by the resonant frequencies of the vocal tract. They are important for distinguishing between different vowel sounds.

Deep learning voice synthesis models have recently gained popularity, including Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs). RNNs, CNNs, LTSM (Long Short-Term Memory), and MPLG (Maximum Likelihood Parameter Generation) models which are trained using massive datasets of recorded speech, are used to model the link between the acoustic and linguistic aspects of speech. The effectiveness and naturalness of synthetic speech have significantly improved because of these models [5]. Statistical model-based TTS systems can produce speech that sounds more natural than rule-based TTS systems [6].

Speech synthesis evaluation often uses the Mean Opinion Score (MOS) as a subjective method for rating synthesized speech samples. MOS is determined by asking a group of listeners to rate the quality of synthesized speech on a scale of 1 to 5 or 1 to 10 and then averaging the scores.

MOS provides a quantitative measure of speech quality based on subjective listener perception [8]. (Shirali-Shahreza and Penn, 2018) suggest utilizing A/B testing instead of MOS to evaluate TTS systems. In A/B testing, subjects are asked to choose between two audio segments (a neutral vote is usually allowed). These investigations should involve hundreds of individuals to account for subject opinion variation.

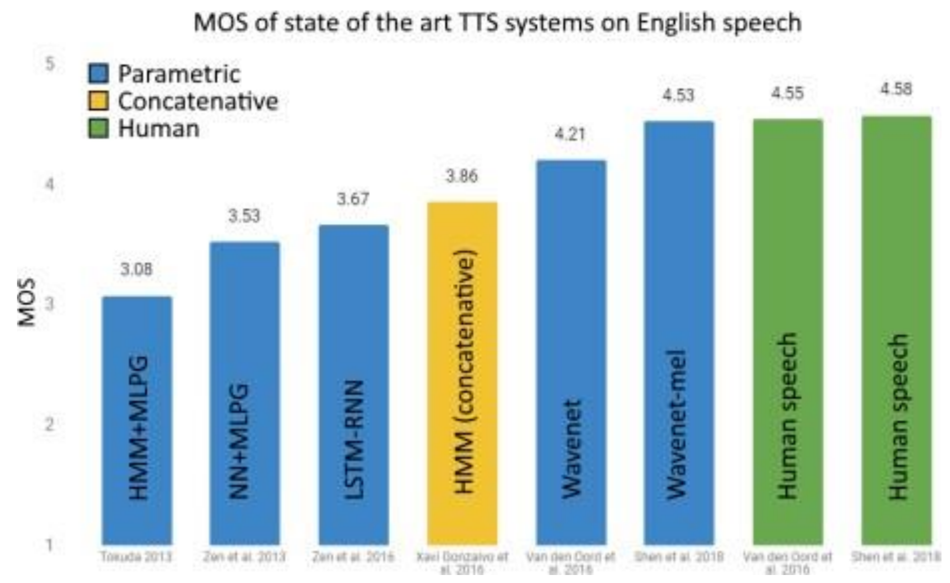


**Figure 1 MOS for the various TTS methods. Extracted from [13]**

# 4. TRANSFER LEARNING FROM SPEAKER VERIFICATION TO MULTISPACER TEXT-TO-SPEECH SYNTHESIS

## 4.2 Mel-Spectrogram

A Mel spectrogram is a representation of the frequency content of an audio signal, with emphasis on the spectral content that is most relevant to human hearing. They are to extract features from speech signals in our case. These features can be used to train machine learning models to recognize spoken words and phrases.

To create a spectrogram, we will apply FFTs (fast Fourier transforms) to small segments, referred to as frames, of the signal. This is known as an STFT (Short-Time Fourier Transformation). This enables us to preserve information about the audio signal's evolution over time. As we traverse the audio signal, the frames will begin to overlap. The hop length, which informs the function of how many sample rates to the right it should slide when creating the next frame, determines how far the frame travels.

The resulting spectrum is then transformed into a Mel-frequency spectrum by applying a filter bank that approximates the non-linear relationship between frequency and human perception of pitch. This filter bank is designed to map the frequency scale to a Mel scale, which is a scale of pitch perception that is based on the properties of the human ear. The filter bank typically consists of a series of triangular bandpass filters that are spaced according to the Mel-scale [16].

Finally, the Mel-frequency spectra for each frame are combined to create a spectrogram, which is a time-varying visualization of the frequency content of the audio signal. In a Mel spectrogram, the y-axis represents frequency in Mel units, while the x-axis represents time in seconds. The color of each pixel in the, or the value in each cell of the spectrogram indicates the amplitude or power of the corresponding frequency component then in decibels.

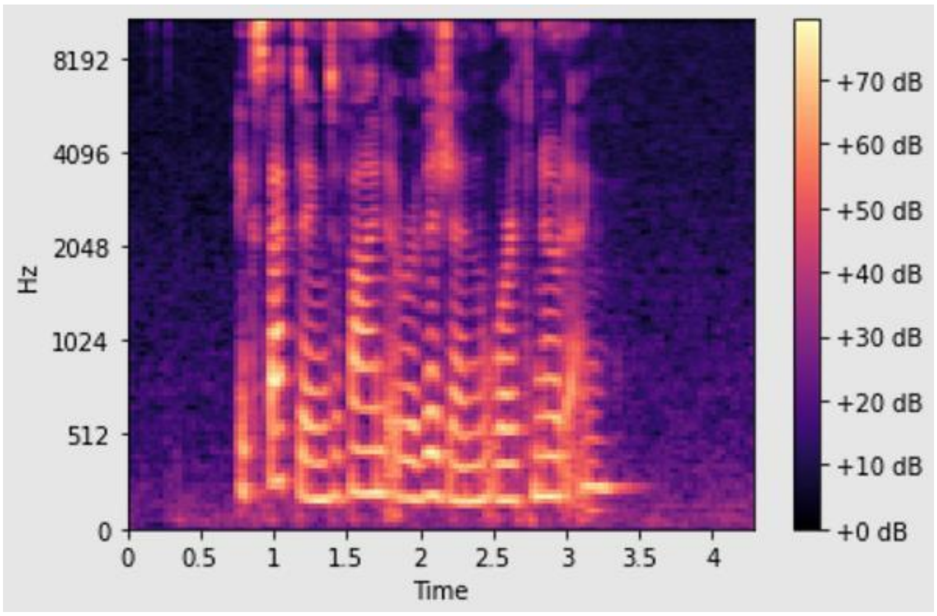


**Figure 2 Example of a Mel-spectrogram.**

## 4.3 Model Overview

This document refers to our method for real-time vocal replication as SV2TTS (speaker verification to text to speech) which is based on this study [9]. It allows for voice duplication with a small sample of speech from the person whose voice we are trying to clone, and the length could be as small as 5 seconds. This research is of Google's numerous Tacotron series5 publications. The SV2TTS research does not include any new concepts; rather, it is built upon other related Google works: the GE2E loss [10], Tacotron [11], and WaveNet [12]. The entire structure consists of a three-stage pipeline, with each stage corresponding to one of the previously described models. The three components of the model are:

- A speaker encoder that generates an embedding from a single speaker's brief speech. The embedding is a vector that can meaningfully represent the most important features of a person’s voice, such that comparable sounds are close together.[10] and [15] describe this model.
- A synthesizer that creates a spectrogram from text based on the embedding of a speaker. This model is based on the famous Tacotron 2 [14] with the removal of the Vocoder used in that paper and using a more efficient version.
- A more efficient vocoder that can produce audio waveform based on the synthesizer's spectrograms.

The speaker encoder is supplied with a small audio sample of the speech from the person whose voice we are trying to clone at inference time. It creates an embedding that will be used as an input to the synthesizer and will affect the spectrogram generated by conditioning it on that speaker's voice, and a sequence of text is fed into the synthesizer. The vocoder generates the speech waveform from the synthesizer's output. We will go into further detail in the upcomingsections.

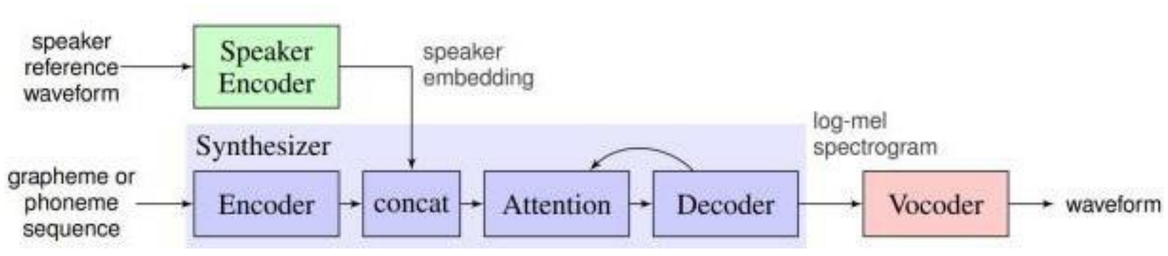


**Figure 3 The SV2TTS framework during inference. Figure extracted from [8].**

SV2TTS provides the flexibility of allowing all models to be trained individually on different datasets. The encoder needs a noise-resistant model that can capture many human speech traits. Thus, to train the encoder, a huge corpus of numerous speakers without noise requirements is best. The encoder is trained using GE2E (Generalized End-to-End) loss, which requires only speaker identity labels. GE2E's model learns to perform a speaker verification task, which isunrelated to voice copying. However, the task requires the network to output a relevant speakervoice embedding. The title of this research study, "Transfer Learning from Speaker Verification to Multispacer Text-To-Speech Synthesis," comes from the fact that this embedding is used for conditioning the synthesizer on a voice.

Even if every component of the framework is trained separately, it is required for the synthesizer to have meaningful embeddings from a trained encoder, and it is also necessary to use that encoder to produce the Mel- spectrograms that will be used to train the synthesizer [13]. This is true even if each component of the framework is trained individually. Figure 8 depicts how the training of each subsequent model relies on the prior model's performance. For the voice encoder to be able to produce meaningful embeddings on the dataset used by the synthesizer, it must be able to generalize to a sufficient degree. Additionally, the speaker encoder still needs to be capable to work in a zero-shot environment when it comes to inference time, meaning the model should be able to generalize to new and unseen speakers without requiring any additional training or fine-tuning. That is why we train it using a larger but less clean dataset since it has more variety when it comes to the number of speakers.

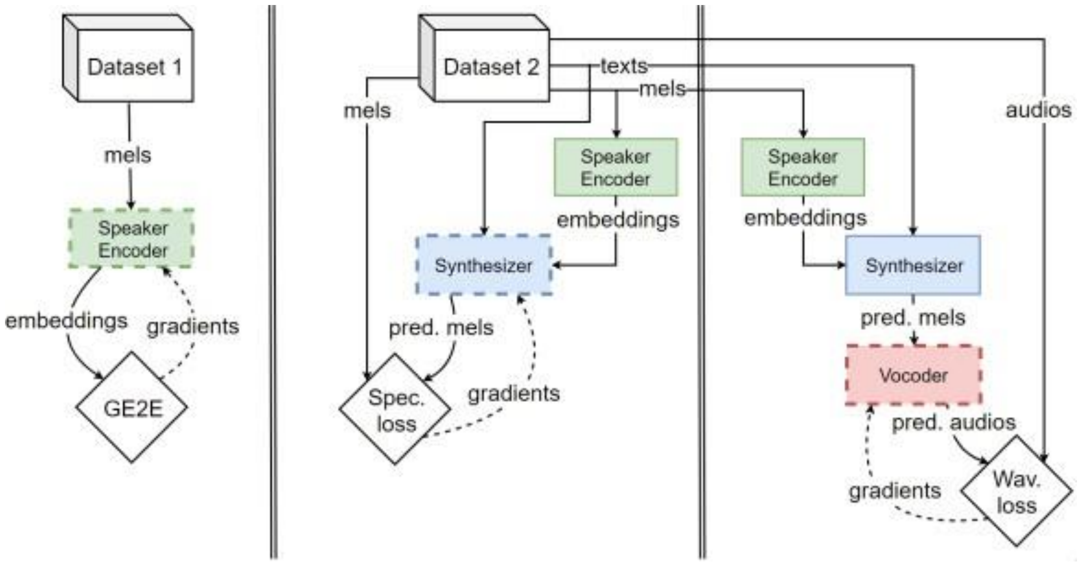


**Figure 4 The stages of training the model and how the parts interact. Dataset 1 is less clean and contains more utterances, while Dataset 2 has cleaner recordings but fewer utterances. Figure extracted from [9]**

## 4.4 Speaker encoder

The encoder model and the training process for it are both discussed in several different works[9][10]. We use the implementation provided by the GitHub user *CorentinJ*,[22] and we modify it to suit SV2TTS requirements.

### *4.4.1 Model architecture*

Training for the speaker encoder is originally for another task which is speaker verification. A common use of biometrics is speaker verification, which involves determining a person's identification by listening to their speech recordings. The speaker embedding of a person is derived from a handful of their utterances in order to construct a template for that individual.

GE2E loss is a loss function used in speaker verification systems to train the neural network to extract speaker embeddings, which can be used for speaker recognition tasks.

The main objective of the GE2E loss is to ensure that the embeddings generated for the same speaker are close to each other, while embeddings generated for different speakers are far apart in the embedding space. This is achieved by maximizing the cosine similarity between the embeddings of the same speaker and minimizing the cosine similarity between embeddings of different speakers.

At training time, the model computes the embeddings $\mathbf{e}_{ij}$ of M utterances of fixed duration from N speakers with ($1 \leq i \leq N$, $1 \leq j \leq M$), which we get after feeding the Mel spectrogram of the utterance to a LTSM. As an extra alteration of the network's last frame response, a linear layer is connected to the last LSTM layer. The output of the complete neural network is denoted as $f(x_{ji};\ w)$, where x represents the features (log-Mel filter bank energies) from a fixed-length segment for speaker **j** and utterance ***i***, and w represents all neural network parameters (including both LSTM layers and the linear layer). The L2 normalization of the network output is defined as the embedding vector. The L2 normalization process involves dividing each element of the vector by the L2 norm of the vector because it can improve the performance of the model and help avoid issues related to different scales of feature values.

$$\mathbf{e}_{ji} = \frac{f(\mathbf{x}_{ji}; \mathbf{w})}{||f(\mathbf{x}_{ji}; \mathbf{w})||_2}.$$

The similarity matrix $\boldsymbol{S_{ij,k}}$ is derived from a two-by-two comparison conducted on embeddings $\mathbf{e_{ij}}$ in the batch against each speaker embedding $\boldsymbol{c_k}$ ($1 \le k \le N$). This similarity is measured using the scaled cosine distance:

$$\mathbf{S}_{ji,k} = \begin{cases} w \cdot \cos(\mathbf{e}_{ij}, \mathbf{c}_i^{(-j)}) + b & \text{if} \quad i = k \\ w \cdot \cos(\mathbf{e}_{ij}, \mathbf{c}_k) + b & \text{otherwise.} \end{cases}$$

Where $\boldsymbol{c_i}$ A speaker embedding that we derive for each speaker by computing the centroid of the embedding of each utterance. Both *w* and *b* are learnable weights. The centroid is computed differently when the embedding of the utterance we are comparing belongs to the same speaker as the centroid, we exclude that embedding because it results in a bias towards the correct speaker independently of how accurate the model actually is. This centroid is referred as to exclusive centroid $\boldsymbol{c_i^{(-j)}}$. Thus, the centroid is computed as follows:

$$\mathbf{c}_i = \frac{1}{M} \sum_{j=1}^{M} \mathbf{e}_{ij}$$

And the exclusive centroid is computed using this formula:

$$\mathbf{c}_i^{(-j)} = \frac{1}{M-1} \sum_{\substack{m=1 \\ m \neq j}}^{M} \mathbf{e}_{im}$$

During the training, we want the embedding of each utterance to be similar to the centroid of all that speaker's embeddings, while at the same time, far from other speakers' centroids so we define the loss on each embedding by putting a SoftMax on each $\boldsymbol{S_{ij,k}}$, for k = 1, . . . , N that makes the output equal to 1 if k = j, otherwise makes the output equal to 0. It could be defined as:

$$L(\mathbf{e}_{ji}) = -\mathbf{S}_{ji,j} + \log \sum_{k=1}^{N} \exp(\mathbf{S}_{ji,k})$$

And the final GE2E loss LG is the sum of all losses over the similarity matrix ($1 \le j \le N$, and $1 \le i \le M$):

$$L_G(\mathbf{x};\mathbf{w}) = L_G(\mathbf{S}) = \sum_{j,i} L(\mathbf{e}_{ji}).$$

During training, each batch has N speakers and M utterances per speaker as parameters for the batch size, and all utterances must have the same length.

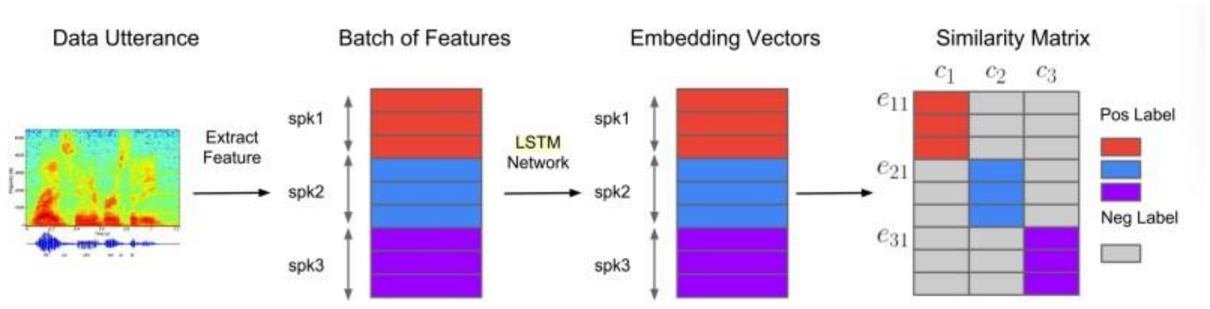


**Figure 5 The procedure of creating the similarity matrix during training. This figure is extracted from[10].**

At the time of inference, an utterance is divided into segments with a 50% overlap, and the encoder inputs each segment individually to the LTSM. After that, the resultant vectors are L2 normalized, and then they are averaged to obtain the speaker embedding that corresponds to the utterance. The L2 normalization process involves dividing each element of the vector by the L2 norm of the vector because it can improve the performance of the model and help avoid issues related to different scales of feature values.

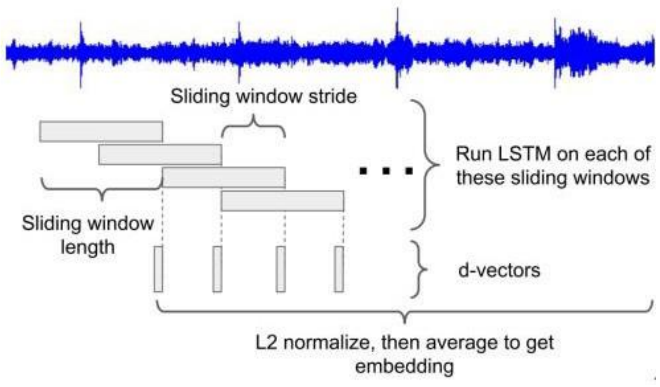

Sliding window stride
Run LSTM on each of these sliding windows
Sliding window length
d-vectors
L2 normalize, then average to get embedding

**Figure 6 Computing the embedding of a complete utterance. Figure extracted from[10].**

### *4.4.2 Training and Dataset*

The LibriSpeech corpus is a large, publicly available dataset of speech recordings, intended for use in speech recognition and related research. It consists of approximately 1,000 hours of English-language speech data, recorded from a wide variety of speakers and covering a diverse range of topics sampled at 16Khz and has 1.2 thousand speakers. It contains 'train_clean100', 'train_clean360', and ‘train_other500' subsets. We will be using the 'train_other500’ since it hasa larger number of utterances that are noisier compared to their clean counterparts but has moreutterances, 148,688 to be exact. For data preprocessing, when sampling fragments from complete utterances, we utilize the webrtcvad Python package to detect silent sections and remove them to avoid using mostly silent segments. Any silent segment between spoken words that exceeds 0.2 seconds is cut. 0.2 is a good choice because it retains a natural speech prosody according to, [13].

For the model parameters, we chose the parameters suggested in the SV2TTS paper. The model consists of a 3-layer LSTM with 768 hidden nodes and 256 projection units because the authors trained their own model for 50 million steps (albeit on a bigger dataset), it is impossible for usto reproduce the findings they obtained; as a consequence, we decided to train our model for atotal of just 1 million steps because our dataset is far smaller.

The inputs of the model are 40-channel Mel-spectrograms with a 25ms window width and a 10ms time step. The output is a 256-element vector containing the L2-normalized hidden state of the final layer, which has been normalized using the L2 transform, and we apply a ReLU layer prior to normalization in an effort to make embeddings sparser and therefore more interpretable.

We set the duration of the utterances in a training batch to 1.6 seconds, and the authors use N = 64 speakers and M = 10 utterances per speaker as parameters for the batch size and a learningrate of 1e-4.

### *4.4.3 Results*

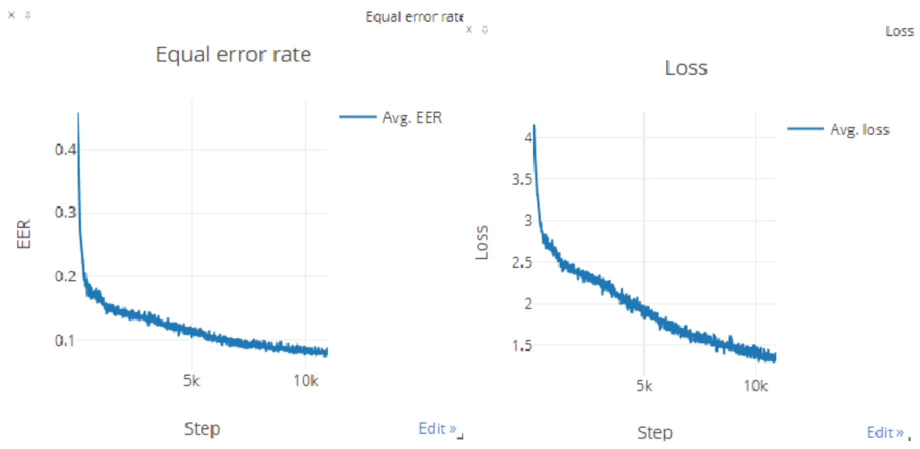


**Figure 7 Speaker verification EER and loss during training for the first 10 k steps**

We were able to get the model to converge early, since the loss is steadily decreasing, and the equal error rate (ERR)–the error rate at which the decision threshold of the classifier makes the same number of false positives (FP) and false negatives (FN)–is also low, which indicates better performance in matching an utterance with its speaker.

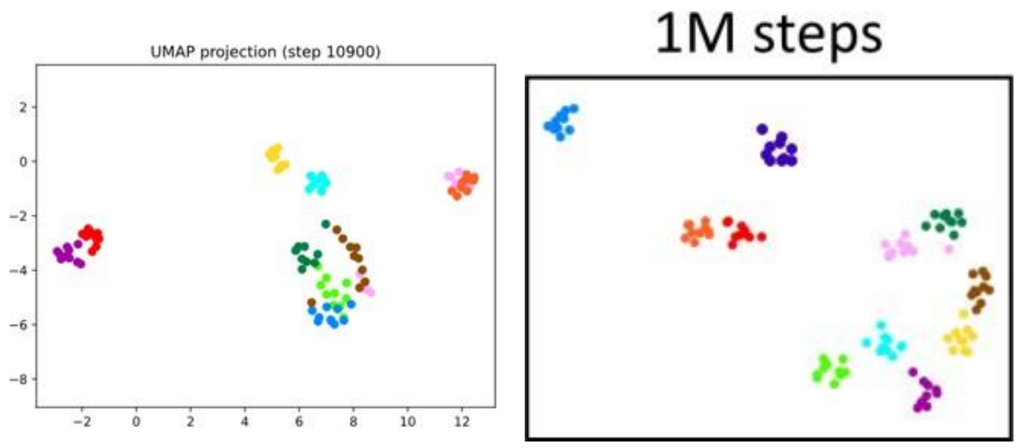


**Figure 8 Umap projection of utterance embeddings from randomly selected batches from the training dataset. A dot of the same color represents utterances from the same speaker.**

We take a sample of ten speakers, each of whom has ten utterances, calculate utterance embeddings, and then project those embeddings onto a two-dimensional space using UMAP

[17]. It is predicted that embeddings from various speakers would be wider apart than embeddings from the same speaker. Therefore, as training advances, clusters of utterances from the same speaker should appear.

## 4.5 Synthesize

The synthesizer is based on the Tacotron 2 model, but without  Wavenet vocoder, since we will be using another vocoder [12]. We use a publically available implementation of Tacotron 2 by Github user *Rayane-mamah* [23] from which we strip Wavenet and modify its parameters to suit SV2TTS.

### *4.5.1 Model overview*

Tacotron is a recurrent sequence-to-sequence model that can predict a Mel spectrogram from text based on the text itself. It has a structure known as an encoder-decoder, which is connected using a location-sensitive attention mechanism [18]. This structure should not be confused with the speaker encoder that SV2TTS employs. First, vector representations of the text's individualcharacters are inserted into the sequence. The text is here represented by a series of one-hot embedding of the English characters and some special characters like the space. The convolutional layers are responsible for capturing local information about the input text, while the LSTM layers capture the long-term dependencies between the input characters. The output of the encoder is a sequence of vectors that represent the phonetic content of the input text. A modification to the architecture is introduced by SV2TTS at this point: Every single vector that is output by the Tacotron 2 encoder has a speaker embedding added to the end of it.

The model is considered to be autoregressive due to the fact that the input of each decoder frame is concatenated with the output of the frame that came before it, and this combination is then routed via a pre-net. The purpose of the pre-net is to reduce the dimensionality of the input and to extract meaningful features that can be used to train the encoder more effectively. The pre-net consists of two fully connected layers with ReLU activation functions, followed by a dropout layer. The concatenated vector that is outputted by the prenet is eventually projected onto a single Mel spectrogram frame after first traveling through two layers of unidirectional LSTM. Another projection of the same vector to a scalar, known as the stop token, allows the network to predict on its own that it has to stop creating frames by emitting a value that is higher than a particular threshold. This is made possible by the fact that the stop token is a scalar. the predicted Mel spectrogram is passed through a 5-layer convolutional post-net which predicts a residual to add to the prediction to improve the quality of reconstructed Mel frames.

The decoder generates the output sequence one frame at a time, based on the previous frame and the current attention context vector. The attention mechanism provides the context information needed to generate each frame, by aligning the input sequence with the output sequence. During the decoding process, the decoder generates a sequence of hidden states that capture the information needed to generate the output frames. The decoder's hidden state at each time step is used as input to the attention mechanism, along with the previous attention contextvector. The attention mechanism then computes a set of attention weights that indicate the relative importance of each input frame for the current time step of the decoder. These attentionweights are used to compute a weighted sum of the encoder's hidden states, which forms the attention context vector.

The attention context vector is then concatenated with the current decoder hidden state and passed through a set of feedforward and convolutional layers to produce the output frame. This process is repeated for each frame in the output sequence, with the attention mechanism providing the alignment information needed to ensure that the output sequence accurately reflects the input sequences.

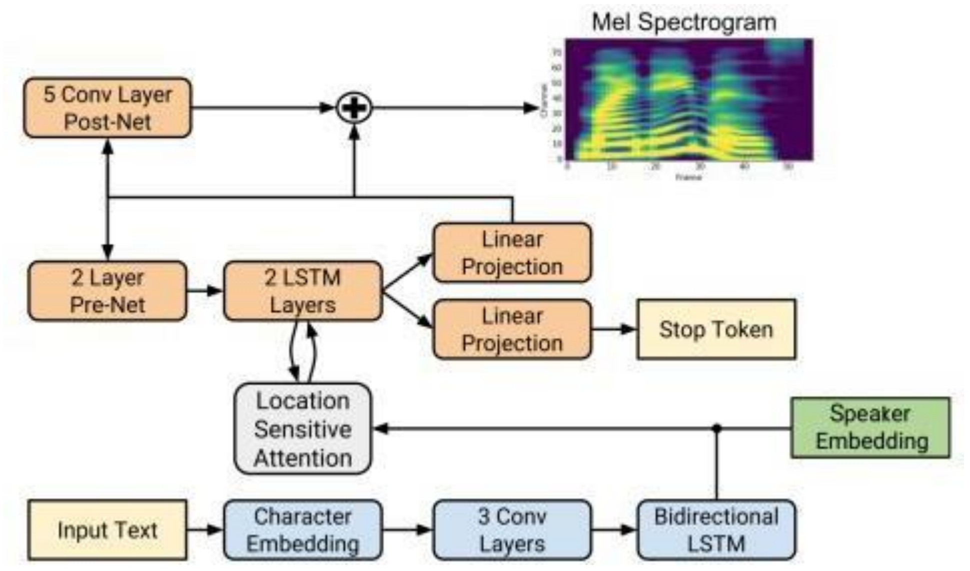
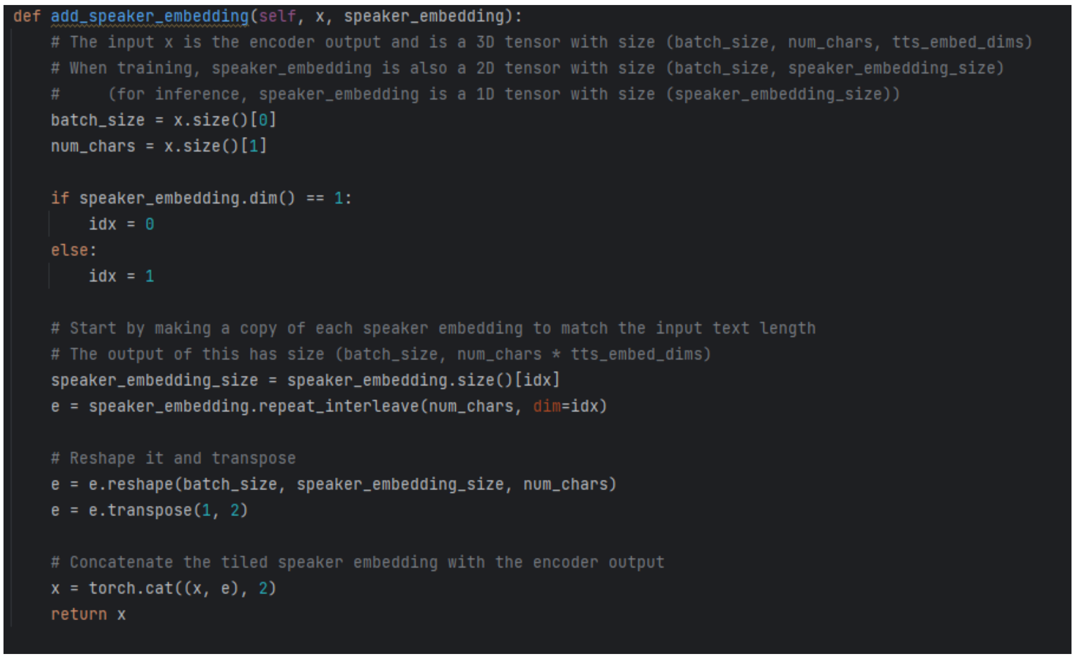


**Figure 9 Altered Tacotron structure. The orange blocks represent the decoder, and the blue blocks represent the encoder. This graphic was updated and taken from [21**

### *4.5.2 Implementation*

Compared to those utilized for the speaker encoder, the target Mel spectrograms for the synthesizer have more features. They have 80 channels and are computed from a 50 ms window with a 12.5 ms step.

The input characters are represented by a 512-dimensional character embedding that has been learned and is then run through a stack of three convolutional layers, each of which has 512 filters. To create the encoded features, the output of the final convolutional layer is fed into a single bi-directional LSTM layer with 512 units (256 in each direction). We then concatenate the speaker embedding to the output tensor.



```python
def add_speaker_embedding(self, x, speaker_embedding):
    # The input x is the encoder output and is a 3D tensor with size (batch_size, num_chars, tts_embed_dims)
    # When training, speaker_embedding is also a 2D tensor with size (batch_size, speaker_embedding_size)
    #     (for inference, speaker_embedding is a 1D tensor with size (speaker_embedding_size))
    batch_size = x.size()[0]
    num_chars = x.size()[1]

    if speaker_embedding.dim() == 1:
        idx = 0
    else:
        idx = 1

    # Start by making a copy of each speaker embedding to match the input text length
    # The output of this has size (batch_size, num_chars * tts_embed_dims)
    speaker_embedding_size = speaker_embedding.size()[idx]
    e = speaker_embedding.repeat_interleave(num_chars, dim=idx)

    # Reshape it and transpose
    e = e.reshape(batch_size, speaker_embedding_size, num_chars)
    e = e.transpose(1, 2)

    # Concatenate the tiled speaker embedding with the encoder output
    x = torch.cat((x, e), 2)
    return x
```

**Figure 10 Concatenation of the speaker embedding and the output of the encoder.**

The **add_speaker_embedding** function takes in two inputs: x, which is the encoder output, and **speaker_embedding**, which is the embedding vector for the speaker. This function concatenates the speaker embedding with the encoder output and returns the concatenated tensor.

The first step in the function is to get the batch size and the number of characters in the input sequence. The size of x is (**batch_size, num_chars, tts_embed_dims**) where **tts_embed_dims** isthe dimension of the encoder output.

Next, the function checks the dimension of the **speaker_embedding**. If the dimension is 1, it means that it is an embedding for a single speaker (which is the case during inference) and the

The function sets the index (**idx**) to 0. If the dimension is 2, it means that the embedding is for multiple speakers (which is the case during training), and the function sets the index to 1.

The speaker embedding is then tiled to match the length of the input text. This is done using the **repeat_interleave** function, which repeats the speaker embedding **num_chars** times along the specified dimension (which is either 0 or 1, depending on the dimensionality of the speaker embedding). This results in a tensor of size (**batch_size**, **num_chars** * **speaker_embedding_size**). The speaker embedding tensor is then reshaped and transposed to match the dimensions of the encoder output tensor. This is done using the reshape and transpose functions. The resulting tensor has size (**batch_size, num_chars, speaker_embedding_size**).

Finally, the speaker embedding tensor is concatenated with the encoder output tensor along the third dimension (dimension 2). This is done using the torch.cat function, which concatenates two tensors along the specified dimension. The resulting tensor has size (**batch_size, num_chars, tts_embed_dims + speaker_embedding_size**). Since our speaker embedding has 256 dimensions in our case and the encoder embedding also has 256 dimensions, the total output embedding has 512 dimensions.

We leave the other hyperparameters related to the attention mechanism and decoder as they are after we update the encoder output and dimension, and then proceed with data pre-processing and training.

### *4.5.3 Data Pre-processing and Training*

We will utilize the train-clean subset of our data set for the synthesizer and vocoder because it is much cleaner and having a clean synthesized voice is a criterion in the paper. Our approach does not process the input texts for pronunciation; instead, the characters are fed exactly as they are. We replace acronyms and numbers with their full textual equivalents, normalize whitespace by ensuring that there is only one space between each word and no more, and lowercase all our letters. After pre-processing, we get 290,729 utterances, which gives us 102 817,322 Mel frames, which is about 356.4 hours of speech. The maximum text character input is 257 characters, and the longest utterance is 898 frames, which is about 12 seconds.

We then use the encoder we trained in part 4.3 to produce a speaker embedding for each utterance in our dataset by following the procedure in Figure 6, and then produce their corresponding Mel-spectrograms. The transcript of each utterance is provided in a text file.

We use the same parameters used by Rayane-mama for training, which are different from the ones used for the paper, but I used them since I do not have access to the same hardware as the authors. For example, the authors use a batch size of 64 but we only use 12 to preserve memory since we are using our own machine to train the model. We train the model for 150 thousand steps and an exponentially decaying learning rate that goes from 1e-3 to 1e-5. Luckily, Rayane-mama implementation allows us to save the model parameters every 500 steps so we can have more flexibility while training.

## 4.6 VOCODER

Spectrograms are a type of visual representation of sound that show the frequencies present in a waveform over time. However, this transformation cannot be reversed since some information is lost, as the frequency resolution of the original signal is reduced. There is no unique inverse transformation function that can map a Mel spectrogram to an audio signal, but this is the role of the vocoder.

WaveNet serves as the vocoder in Tacotron2 and the SV2TTS paper. Since its debut, WaveNet has been at the forefront of deep learning for audio and continues to set the bar for voice naturalness in TTS. However, it is also renowned for having the slowest inference time of any real deep-learning architecture. Instead, [19] proposes a more efficient model which is WaveRNN that we will be using instead of WaveNet since it is fasterduring inference and is less demanding computationally. We will be using an open-source implementation provided by GitHub user *fatchord* [24].

We train the model using waveform and spectrogram tuples, with the spectrograms being generated from the trained synthesizer. And we leverage the user interface that he provides andmodify it so that we can load the audio sample that of the voice we want to clone and then inputthe text of the speech we want to generate.

## 5. RESULTS

As mentioned before, there is no objective  way to assess the quality of the synthesized speech, there are only subjective ways of assessing it. Methods like MOS require getting the opinion from multiple listeners and averaging the score they give, but due to time limitations we couldn’tconduct such a survey, so we will just provide our own opinion and observations on a couple of samples we generated.

Despite the occasional strange prosody, we found the outcomes to be satisfactory. The voice cloning ability of the framework is reasonably good, but it is not on par with approaches that make use of greater reference speech time.

To elaborate further, the unnatural prosody can be caused by various factors, such as incorrect stress patterns, inconsistent intonation, or the talking speed of the sample we are trying to clone. We also noticed that when a sentence is too short, the voice will stretch it out by pausing for a longer period of time, and when a sentence is too long, the voice will speed through it. Regarding the voice cloning ability of the framework, the similarity of the samples and synthesized speech indicates that it is reasonably good, but not on par with methods that use more reference speech time. This suggests that the framework may not yet have reached its full potential and may benefit from additional training data. Voice cloning typically requires a large amount of reference speech with a large variety of speaker data to capture the nuances of the speaker's voice and reproduce it accurately. Therefore, the framework may require additional training data to improve its voice cloning ability.

# 6. LIMITATIONS AND FUTURE WORK

Large datasets of recorded speech were used by the authors of the model we implemented in this project. The quantity and variety of the training data may be a limiting factor in the cloned voices' quality. The framework may be given more samples of various speech patterns from larger, more varied datasets, which may enable it to better capture the subtleties of a wider range of voices. Therefore, the voice cloning system may perform better as the dataset's quantity and diversity are increased.

It is crucial to understand that the language and accent being copied might affect how successful the voice cloning framework is. Since the study was done in a particular language, it may not be possible to generalize the findings to other languages and accents. For instance, the voice cloning system may encounter difficulties due to the diversity of speech patterns among languages. To handle the variation in speech patterns among languages, future work should entail expanding the framework to function with multiple languages and accents.

The issue of the dataset's lack of variety can be addressed, for example, by using data augmentation techniques. Data augmentation is the process of modifying already-existing data to produce new samples that more accurately represent differences in speech patterns, such as pitch, speed, and tone. The voice cloning system may learn to better capture the subtleties of many voices and enhance its performance by adding more variation to the dataset.

# 7. STEEPLE ANALYSIS

## 7.2 social

Social factors play an important role in the adoption and use of a text-to-speech synthesizer. Factors such as the acceptance of synthesized voices in society and the perceived usefulness of the technology for individuals with disabilities can influence the market demand for the product. Additionally, cultural, and linguistic differences may also impact the effectiveness of the synthesizer in different regions.

## 7.3 Technological

The text-to-speech synthesizer is a technological product, so advancements in related fields such as speech recognition, natural language processing, and machine learning can impact the quality and effectiveness of the synthesizer.

The increasing use of voice-enabled devices such as smart speakers and virtual assistants can also create opportunities for the integration of text-to-speech technology.

## 7.4 Economic

The cost of the text-to-speech synthesizer, as well as the cost of related software and hardware, can impact the market demand for the product.

Additionally, the cost-effectiveness of the synthesizer in comparison to other assistive technologies can influence its adoption and use.

## 7.5 Environmental

Environmental factors are not likely to have a direct impact on the text-to-speech synthesizer, but the product may have an indirect impact on the environment depending on the materials and manufacturing processes used. The most important thing to think about, however, remains to be the amount of energy and electronic waste generated by the hardware that was used to train the model.

## 7.6 Political

Concerns and debates may be generated by the project's political ramifications and potential exploitation of its technology, particularly in relation to the production of deep fakes and synthetic voices, with the goal of persuasion for a specific party or ideology.

## 7.7 Legal

Legal considerations such as data privacy, copyright, and accessibility regulations can impact the development, marketing, and use of the text-to-speech synthesizer.

## 7.8 Ethical

Ethical considerations such as the use of synthesized voices for impersonation or deception can impact the public perception and adoption of the technology.

The potential impact of technology on employment and job displacement is also an ethical consideration that should be considered.

## 8. INVESTIGATION ON RELATED ENGINEERING STANDARDS

The IEEE (Institute of Electrical and Electronics Engineers) has published several standards related to text-to-speech synthesis. Here are some of the most important ones:

IEEE 187 is a standard that defines the minimum requirements for the voice quality of a TTS system. It specifies the following criteria for a TTS system's voice quality:

- Naturalness: The TTS system's voice should sound as close to human speech as possible, with no obvious synthetic-sounding artifacts.
- Intelligibility: The TTS system's voice should be easy to understand, with accurate pronunciation and appropriate speed.
- Expressiveness: The TTS system's voice should convey appropriate emotion and tone, such as sarcasm or enthusiasm.

When implementing a TTS system, these criteria can be taken into consideration by selecting a high-quality voice that meets these requirements, and by ensuring that the TTS system's algorithms and settings produce natural, intelligible, and expressive speech The model we implemented did well when it comes to naturalness and intelligibleness but is lacking when it comes to expressiveness.

IEEE P1876 is a draft standard that provides guidelines for the prosody of TTS systems. Prosody refers to the rhythm, intonation, and stress patterns of speech. The P1876 standard specifies guidelines for the following aspects of prosody:

- Pitch: The TTS system's pitch should be consistent with the natural pitch of human speech, with appropriate variations for emphasis and intonation.
- Duration: The TTS system's duration should be consistent with the natural duration of human speech, with appropriate variations for emphasis and intonation.
- Intensity: The TTS system's intensity (loudness) should be consistent with the natural intensity of human speech, with appropriate variations for emphasis and intonation.

When implementing a TTS system, the guidelines in the P1876 standard can be taken into consideration by using algorithms and settings that produce appropriate pitch, duration, and intensity variations, to produce natural-sounding prosody, and the model succeeds in these aspects.

The IEEE standards ISO/IEC 23053:2022, titled "Framework for Artificial Intelligence (AI) Systems Using Machine Learning (ML)" provide a comprehensive framework for the development and deployment of AI systems that use machine learning. The standard outlines best practices and guidelines for the design, development, and evaluation of AI systems, with a particular focus on machine learning techniques.

In the context of a text-to-speech synthesis, the IEEE 23053 standard can be highly relevant. Text-to-speech systems use machine learning techniques, such as deep neural networks, to generate synthetic speech from input text. These systems must be carefully designed, trained, and evaluated to ensure that they produce high-quality, natural-sounding speech.

The data that developers use to train their models should be properly selected and pre-processed since this is one of the recommendations made by the standard. This will guarantee that the data is reflective of the real-world inputs that the system will come into contact with. The performance of the developers' systems should also be evaluated using rigorous evaluation criteria, such as the word error rate and subjective hearing tests, in accordance with the recommendations made by the standard.

Text-to-speech developers may guarantee that their systems are designed in a responsible and methodical manner and that their systems produce high-quality speech that satisfies the requirements of their users by adhering to the principles that are outlined in the IEEE 23053 standard. These recommendations were produced by the IEEE.

# 9.CONCLUSIONS

Overall, the text-to-speech synthesis capstone project was effective in constructing a high-quality and natural-sounding TTS system, and it was successfully finished. The overall quality of TTS is influenced by several important elements, including voice quality, which has been the subject of extensive research. Additionally, we have investigated several TTS synthesis methods and models, with a focus on neural network-based strategies. All three of the model's components, including the Mel spectrogram, synthesizer, and vocoderhave been fully implemented, and training and testing have been successful in achieving our set goals. As a stretch goal for such a project, collecting data, training models, and evaluating them is imperative in order to further improve the performance of the TTS system.

Nevertheless, the development of this project has been a significant undertaking, drawing upon extensive research into deep learning, neural networks, speech recognition, and text-to-speech synthesis. As someone with a keen interest in this field, I am excited to continue exploring the potential for future improvements, particularly for other languages. This project has been an invaluable opportunity to apply and expand upon my technical skills, and I look forward to contributing to advancements in this and other related fields.

## REFERENCES

[1] T. Dutoit, "An introduction to text-to-speech synthesis," in Springer Science & Business Media, 2001.

[2] E. Moulines and F. Charpentier, "Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones," in Speech communication, vol. 9, no. 5-6, pp. 453-467, Oct./Nov. 1990.

[3] P. Taylor, S. King, and R. Godby, "A comparison of HMM-based and unit-selection synthesis," in Proceedings of the International Conference on Text, Speech, and Dialogue, pp. 102-109, Springer, Sep. 2009, https://doi.org/10.1007/978-3-642-04207-2_14.

[4] Y. Bao, H. Huang, Y. Wang, and L. Deng, "Recent advances in statistical parametric speech synthesis," in Frontiers in Electronics, vol. 1, pp. 1-12, Jan. 2020, https://doi.org/10.3389/fel.2020.00001.

[5] L. Fan, H. Li, and Z. Zhang, "Deep learning in speech synthesis: A review," in Journal of Signal Processing Systems, vol. 93, no. 5, pp. 471-485, May 2021, https://doi.org/10.1007/s11265-020-01670-w.

[6] H. Zen et al., "LibriTTS: A Corpus Derived from LibriSpeech for Text-To-Speech," in IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019.

[7] L. Deng, X. Li, X. Huang, and K. Yao, "Recent advances in deep learning for speech research at Microsoft," in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 8604-8608, May 2013, https://doi.org/10.1109/ICASSP.2013.6639246.

[8] T. Raitio and A. Suni, "A review of methods for evaluation of speech synthesis quality," in Speech Communication, vol. 90, pp. 106-126, Apr. 2017, doi: 10.1016/j.specom.2017.02.001.

[9] Y. Jia, Y. Zhang, R. J. Weiss, Q. Wang, J. Shen, F. Ren, Z. Chen, P. Nguyen, R. Pang, I. Lopez-Moreno, and Y. Wu, "Transfer learning from speaker verification to multispeaker text-to-speech synthesis," CoRR, abs/1806.04558

[10] L. Wan, Q. Wang, A. Papir, and I. Lopez Moreno, "Generalized end-to-end loss for speaker verification," in IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017.

[11] Y. Wang et al., "Tacotron: A fully end-to-end text-to-speech synthesis model," in IEEE Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017, pp. 4779-4783, doi: 10.1109/ICASSP.2017.7962838.

[12] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio,"CoRR, vol. abs/1609.03499, 2016. [Online]. Available: http://arxiv.org/abs/1609.03499.

[13] A. Jemine, C. Promoteur, Louppe, and Gilles, “Master thesis: Automatic Multispeaker Voice Cloning.” Available: https://matheo.uliege.be/bitstream/2268.2/6801/5/s123578Jemine2019.pdf

[14] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. J. Skerry-Ryan, R. A. Saurous, Y. Agiomyrgiannakis, and Y. Wu, "Natural TTS Synthesis by conditioning wavelet on mel spectrogram predictions," CoRR, abs/1712.05884, 2017. Available: http://arxiv.org/abs/1712.05884.

[15] G. Heigold, I. Moreno, S. Bengio, and N. Shazeer, "End-to-end text-dependent speaker verification," CoRR, abs/1509.08062, 2015. Available: http://arxiv.org/abs/1509.08062.

[16] D. Stowell and M. D. Plumbley, "Automatic large-scale classification of bird sounds isstrongly improved by unsupervised feature learning," PeerJ, vol. 7, no. 12, e488, 2009, doi: 10.7717/peerj.488.

[17] L. McInnes and J. Healy, "Umap: Uniform manifold approximation and projection for dimension reduction," arXiv preprint arXiv:1802.03426, 2018. [Online]. Available: https://arxiv.org/abs/1802.03426.

[18] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models forspeech recognition," CoRR, vol. abs/1506.07503, 2015. [Online]. Available: https://arxiv.org/abs/1506.07503.

[19] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. Van Den Oord, S. Dieleman, and K. Kavukcuoglu, "Efficient Neural Audio Synthesis," arXiv preprint arXiv:1802.08435, 2018. [Online]. Available: https://arxiv.org/pdf/1802.08435v1.pdf.

[20] S. Arik, G. Diamos, A. Gibiansky, J. Miller, K. Peng, W. Ping, J. Raiman, and Y. Zhou,"Deep voice 2: Multi-speaker neural text-to-speech," 2017. [Online]. Available: https://arxiv.org/abs/1705.08947.

[21] Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang,Y., Skerry-Ryan, R., Saurous, Rif A, Agiomyrgiannakis, Y., & Wu, Y. (2017). Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions. ArXiv.org. https://arxiv.org/abs/1712.05884

[22] C. Jemine, "Real-Time Voice Cloning," GitHub. [Online]. Available: https://github.com/CorentinJ/Real-Time-Voice-Cloning. [Accessed: February 28, 2023].

[23] R. Mamah, "Tacotron-2," GitHub. [Online]. Available: https://github.com/Rayhane-mamah/Tacotron-2. [Accessed: March 21, 2023].

[24] F. Chollet, "WaveRNN," GitHub. [Online]. Available: https://github.com/fatchord/WaveRNN. [Accessed: April 15, 2023].