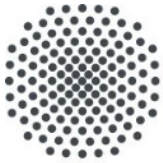


University of Stuttgart

Institute for Natural Language Processing (IMS)

Pfaffenwaldring 5b
70569 Stuttgart

Master Thesis

Deep Learning for Voice Cloning

Soumyadeep Bhattacharjee

Study Program :	M.Sc. Computer Science
Examiner :	Professor Dr. Ngoc Thang Vu
Advisor :	Dr. Bac Nguyen Cong
Start Date :	01.12.2020
End Date :	01.08.2021

Abstract

Voice cloning is the artificial simulation of the voice of a specific person. We investigate various deep learning techniques for voice cloning and propose a cloning algorithm that generates natural-sounding audio samples using only a few seconds of reference speech from the target speaker. We follow a transfer learning approach from a speaker verification task to text-to-speech synthesis with multi-speaker support. The system generates speech audio in the voices of different speakers, including those that were not observed during the training process, i.e. in a Zero-shot setting. We encode speaker-dependent information using latent embeddings, allowing other model parameters to be shared across all speakers. By training a separate speaker-discriminative encoder network, we decouple the speaker modeling step from speech synthesis. Since the networks have different data requirements, decoupling allows them to be trained on independent datasets. Using an embedding-based approach for voice cloning improves speaker similarity when utilized for zero-shot adaptation to unseen speakers. Furthermore, it minimizes computational resource requirements and could be beneficial for use-cases that require low-resource deployment.

Acknowledgments

I would like to convey my sincere gratitude to my supervisor Dr. Bac Nguyen Cong at the Sony Europe B.V. for guiding me throughout the course of my research.

I would also like to express my gratitude to my examiner, Professor Dr. Thang Vu at the Institute for Natural Language Processing, University of Stuttgart, for overseeing my thesis and steering me in the right direction.

Finally, I would like to thank all the members of the AI for Speech and Sound Group at Sony for providing their valuable feedback throughout my thesis.

List of Abbreviations

AI	Artificial Intelligence
DL	Deep Learning
HMM	Hidden Markov Model
GMM	Gaussian Mixture Model
RBM	Restricted Boltzmann Machine
CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory
RNN	Recurrent Neural Network
BLSTM	Bidirectional Long Short-Term Memory
LUT	Lookup Table
ZSL	Zero-shot Learning
TTS	Text-To-Speech
SV2TTS	Speaker Verification to Multispeaker Text-To-Speech Synthesis
UMAP	Uniform Manifold Approximation and Projection for Dimension Reduction
MOS	Mean Opinion Score
VCTK	Voice Cloning Toolkit

Contents

Chapter 1

Introduction

Voice cloning is the artificial simulation of the voice of a specific person. It can prove to be an extremely useful feature in today's digital world with an increased demand for voice-based services. Voice cloning can be used as a mode of communication for people who lost their voice, in call centers for automated answering systems, transfer of voices across languages, generating speech from text in low resource settings, or for merely customization purposes in voice assistants. Other applications include narration of audio-books, podcasts, or adding post-production voice-overs for characters in movies or video games. Recent research in voice cloning has uncovered new synthetic speech generation techniques that can closely mimic a targeted human voice. Although an average listener cannot distinguish the difference between a real and a synthesized voice, there is still potential for development in terms of naturalness.

1.1 Motivation

Training a single-speaker TTS model is effectively a type of voice cloning, but it has the limitation of not generalizing to multiple speakers. A naïve approach to achieve multi-speaker voice cloning would be to train separate TTS models for each speaker. Natural speech synthesis necessitates training on a large number of high-quality <audio,transcript> pairs, and supporting multiple speakers typically requires tens of minutes of training data per speaker [1]. It is however impractical to record a large amount of high-quality data for a large number of speakers.

Recent approaches in multi-speaker TTS models recognize new speakers based on an embedding of the target voice. This is usually a low-dimensional embedding that is obtained by using the speaker encoder model, which uses reference speech as input [1, 2, 3]. A TTS model trained using this approach requires lesser

data and is faster and computationally less expensive to train than a separate TTS model for each speaker.

In this project, our interest lies in integrating the voice of any speaker with little or no change to the base model, making it less computationally expensive and more data efficient. We follow a transfer learning approach from a speaker verification task as demonstrated in the SV2TTS model[3]. We decouple the speaker modeling step from speech synthesis by training a separate speaker-discriminative encoder network. This network captures the speaker characteristics in a lower-dimensional space, independently of the synthesizer, which is then trained by conditioning it on the speaker embedding generated by the encoder network. Decoupling the networks allows them to be trained on independent datasets, reducing the need for high-quality multi-speaker training data.

1.2 Objective

The objective of this master thesis is two-fold. First, we would like to explore different deep learning techniques for voice cloning. As a final goal, we would like to develop a suitable voice cloning algorithm, which takes only a few seconds of reference speech and generates natural-sounding audio samples. The algorithm should be able to generate speech audio in the voice of different speakers, including those unseen during training, that is, in a zero-shot setting.

In summary, the following research questions will be investigated in this Master thesis:

- How can a voice cloning model be developed, using only a few seconds of reference audio signal to simulate the voice of a target speaker?
- Can the model generalize to unseen speakers in a Zero-shot setting?
- What suitable representation can be utilized to capture voice characteristics without requiring additional data?

As a final step, we evaluate the performance of the proposed model and provide an analysis of the results along with potential ways of improving the model performance.

1.3 Thesis Outline

The thesis will be organized in the following manner.

- **Chapter 1: Introduction**

  Introduces the topic of voice cloning highlighting the motivation and objectives of the Master thesis project.

- **Chapter 2: Evolution of Speech Synthesis**

  Provides a brief history of speech synthesis techniques, from mechanical synthesis methods through modern deep-learning-based systems.

- **Chapter 3: Related Work**

  Highlights the methodologies used in previous research work in the domain of voice cloning.

- **Chapter 4: Background**

  Elaborates on the theoretical understanding required for the numerous concepts utilized in the thesis.

- **Chapter 5: Voice Cloning - Methodology**

  Provides a detailed study and analysis of the methods implemented in our Voice cloning model.

- **Chapter 6: Resources**

  Highlights the dataset and tools used in the project.

- **Chapter 7: Experiments and Results**

  Elaborates the experiments performed and provides an analysis of the results.

- **Chapter 8: Conclusion**

  Concludes the work with a discussion on the current method and potential enhancements for future work.

Chapter 2

Evolution of Speech Synthesis

Humans have been in a continuous endeavor to synthesize human voice for centuries; the first attempt being made in the 18^{th} century. A historical overview may be beneficial in understanding how modern systems work and how they have evolved to their current form. The history of synthesized speech is explored in this chapter, from the earliest mechanical efforts to the methods that form the foundation for the state-of-the-art synthesizers used in practice today. Several key milestones in synthesis-related methodologies and techniques will also be briefly examined.

Naturalness and intelligibility are the most significant characteristics of a speech synthesis system. Naturalness refers to how closely the output resembles human speech, whereas intelligibility refers to how easily the output can be comprehended. A speech synthesis system should ideally produce speech audio that is both intelligible and natural-sounding.

2.1 Mechanical Synthesis

Around 1750, Leonhard Euler developed a physical understanding of sound waves. He questioned how speech could emerge from air flow through the tract and vocal folds related to the tonal qualities of the letter when it was spoken. He speculated that some kind of musical instrument might be possible to produce similar sounds and combined together into comprehensible words, resulting in the first ideation of a mechanical synthesizer of human speech. Following the ideas of Euler, Prof. Kratzenstein at the University of Copenhagen explained the physiological differences between the five long vowels ($/a/, /e/, /i/, /o/, /u/$) and created an apparatus to produce them artificially in 1779. He created acoustic resonators that resembled the human vocal tract and activated them with vibrating reeds similar to those used in musical instruments.

Wolfgang von Kempelen presented the "Acoustic-Mechanical Speech Machine" in Vienna a few years later, capable of producing single sounds along with certain sound combinations. A pressure chamber simulating the lungs, a vibrating reed acting as vocal cords, and a leather tube replicating the vocal tract action were the key components of the machine. He could make varied vowel sounds by changing the shape of the leather tube. Four distinct restricted channels were used to simulate consonants, which were controlled by the fingers. He also used a model of a vocal tract with a hinged tongue and movable lips for plosive sounds. The vocal tract, a chamber between the vocal cords and the lips, was considered to be the primary site of acoustic articulation, according to his research [4, 5].

Around the mid-nineteenth century, Charles Wheatstone built his famous version of von Kempelen's speaking machine. Slightly more sophisticated, it could create vowels as well as the majority of consonant sounds. Some sound combinations, as well as whole words, were also feasible to generate. All passageways were closed, and vowels were created with a vibrating reed. Resonances were altered by distorting the leather resonator in the same way that von Kempelen's mechanism did. Using turbulent flow through an appropriate passage with reed-off, consonants including nasals were produced.

Inspired by Wheatstone, Alexander Graham Bell built a speech machine of this kind together with his father in the late 1800's. Research and tests on mechanical and semi-electrical vocal system analogs were conducted until the 1960s, without noticeable success.

2.2 Electrical Synthesis

The advancement in electrical engineering around the beginning of the 20th century allowed vocal sounds to be synthesized using electrical techniques. The Voice Operating Demonstrator (abbreviated as the VODER), created by Homer Dudley, debuted during the 1939 World Fair in New York and was the first device recognized as a speech synthesizer. A wrist bar was used to pick a voicing or noise source, and a foot pedal was used to alter the fundamental frequency. The source signal was fed through ten bandpass filters, the output levels of which could be adjusted by the fingers. Although the voice quality and intelligibility were poor, the ability to create artificial speech was clearly established.

The scientific community grew increasingly interested in speech synthesis with the demonstration of VODER. Finally, it was demonstrated that understandable speech may be generated artificially. VODER's core structure and concept are quite similar to current systems that use the source-filter-model of speech. In 1951, Franklin Cooper and his colleagues at Haskins Laboratories produced a Pattern Playback synthesizer, which transformed recorded spectrogram patterns into sounds, either as they were or as they were chosen to be modified.

2.3 Concatenative Synthesis

The concatenation (or joining together) of pre-recorded voice fragments is the basis of Concatenative Speech Synthesis (abbreviated as CSS). In general, concatenative synthesis yields the most natural-sounding synthetic speech. As long as adequate voice data is available for development, this method has the benefit of producing extremely natural-sounding speech. The downside is that all of the speech segments used must be pre-recorded, limiting the option of speaker voices and other verbal expression modifications. There are three subcategories of concatenative synthesis – Unit Synthesis, Diphone Synthesis, and Domain-Specific Synthesis [6] .

In unit selection synthesis, large datasets of recorded speech are employed. During the database building process, each recorded voice is separated into one or more of the following categories: phones, diphones, half-phones, morphemes, syllables, phrases, words, and sentences. Typically, segmentation is performed using a specially adapted speech recognizer in "forced alignment" mode, with human correction using visual representations such as waveforms and spectrograms. Since only a minimal amount of digital signal processing (DSP) is applied to the recorded voice, unit selection delivers the most realistic results.

A diphone, in phonetics, is the phone transition between a pair of neighboring phones in a sentence. Diphone synthesis works by storing all of a language's diphones in a tiny speech database. The voice database in diphone synthesis only has one example of each diphone. Digital signal processing techniques like linear predictive coding, PSOLA, or MBROLA, as well as more recent procedures like pitch modulation in the source domain using the discrete cosine transform, superimpose the intended prosody of a phrase on these minimal units at runtime. Diphone synthesis is negatively impacted by the sonic glitches of concatenative synthesis and the robotic-sounding nature of formant synthesis.

Another type of concatenative synthesis is domain-specific synthesis, which joins the pre-recorded phrases and words to generate whole utterances. It is used when the output of the system is confined to a single domain, as in applications like weather reports [7]. These systems are not general-purpose and can only produce pre-recorded words and phrases since they are restricted by the words and sentences in their databases. However, merging words within naturally spoken language may still pose challenges unless numerous variations are taken into account.

2.4 HMM based Synthesis

While concatenative synthesis may produce natural-sounding synthetic voice, the technique is limited by the voice corpus characteristics utilized in the process of unit selection. Only pre-recorded segments, like diphones, can be utilized for voice production in concatenative systems resulting in a lack of flexibility.

Another TTS technique that has been quite prominent in the realm of voice technology is statistical parametric voice synthesis (SPSS) [8]. It tackles the principal constraint of concatenative systems, by producing speech using statistical language models (also known as an acoustic model), rather than depending on pre-recorded units. These statistical models are machine-learned from speech corpus consisting of audio recordings and the respective text transcripts. Statistical probability models help determine the word sequence which is most likely to occur based on probability values of a phoneme (the fundamental sound unit) being uttered.

A complete SPSS system includes modules for text analysis, feature extraction, and waveform creation. The traditional approach is based on a HMM-GMM architecture[9] for feature extraction and a vocoder for waveform generation. The parameters of a speech unit, such as the spectrum, phoneme duration, and the fundamental frequency (F0), are statistically modeled and generated using HMMs based on the maximum likelihood criterion.

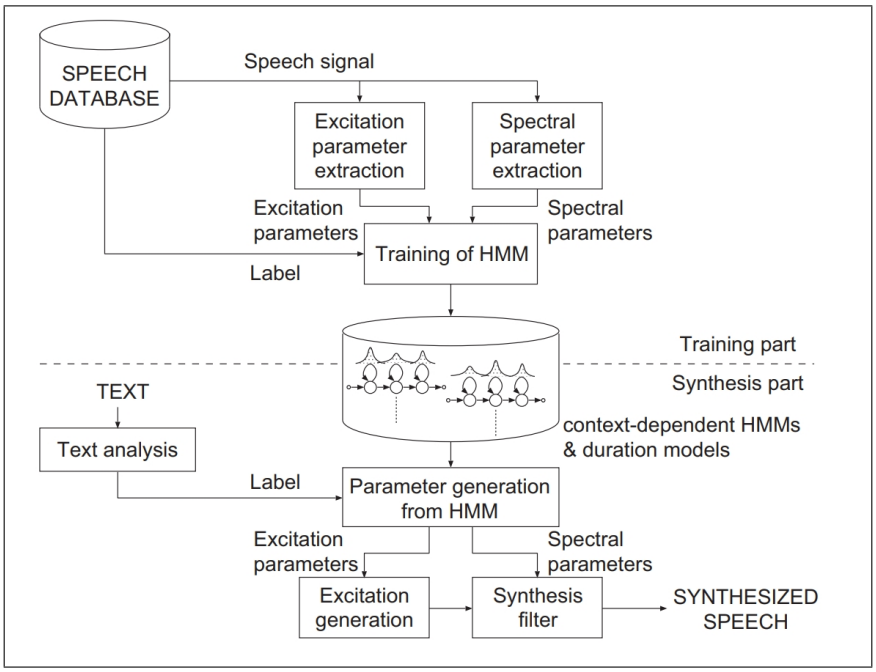


Figure 2.1 The training and synthesis components of a typical HMM-based speech synthesis system depicted as a block diagram. This figure is extracted from Black et al.,2007 [8].

The advantages of HMM-based generation include the capacity to easily alter voice qualities and to adapt it to a variety of languages with minimal modification. It has a tiny footprint and can be used to synthesize a wide range of speaking styles or speech with emotion. The quality of speech generated is the most significant drawback of the HMM-based generation synthesis technique over the unit selection approach. Over-smoothing, modeling accuracy, and the Vocoder are the three factors that appear to impair speech quality.

2.5 Deep Learning based Synthesis

Deep learning and end-to-end approaches have significantly contributed to recent improvements in voice synthesis, which have been used to improve a wide range of applications like chatbots and conversational AI. Deep learning-based methods for voice synthesis can operate on massive datasets of *<speech, text>* pairings to identify effective feature representations that bridge the gap between speech and the corresponding text transcript. Unlike HMM-based methods that

map linguistic features to probability densities of speech parameters using several decision trees, DL-based methods directly map linguistic features into acoustic features using deep neural networks, which are particularly efficient at learning intrinsic data features. Numerous models have been proposed in the long lineage of studies that use Deep Learning techniques for speech synthesis, some of which will be highlighted in this section.

The first is the Restricted Boltzmann Machine, which is commonly used to model the spectral envelopes of acoustic parameters. A deep belief network (DBN) with multi-distribution is a method for simulating the simultaneous distribution of context information and auditory characteristics. It uses three forms of RBMs to represent the continuous spectral, multi-space F0, and discrete voiced/unvoiced (V/UV) parameters simultaneously. A DBN is used to represent the joint probability of syllables and speech characteristics, therefore reducing over-smoothing in HMM-based speech synthesis [10].

Another approach is to employ a BLSTM-RNN, which is a modified version of a bidirectional recurrent neural network. It substitutes units in the BRNN's hidden layers with LSTM memory blocks, with which it may store information for long and short time contextual dependencies, both in the forward and backward directions. To predict the acoustic parameters in speech synthesis using a BLSTM model, we must first transform the input text into a feature vector, and then pass it to the BLSTM model which can then map the input features to acoustic features. The output consists of static features like spectral envelope, fundamental frequency, gain, and Unvoiced/Voiced decision flags, or simply a spectrogram, which are then passed to a Vocoder to generate the speech waveform[11].

TTS systems usually comprises of a text analysis front-end, an acoustic model and a speech-synthesizer. These components are trained independently and rely on broad domain expertise which is not only cumbersome but can lead to propagation and accumulation of errors from each individual component. End-to-end approaches in speech synthesis, that combine all the constituent components into a single integrated framework, are increasingly being used in voice synthesis in order to overcome this issue. WaveNet, a deep generative model of raw audio waveforms, was suggested by DeepMind in September 2016, exhibiting the ability to generate raw waveforms using acoustic features such as spectrograms or mel-spectrograms. Char2wav, a model for producing raw waveforms in an end-to-end fashion, was suggested in early 2017. To produce acoustic features directly from the input text, Facebook and Google proposed the VoiceLoop and Tacotron ar-

chitectures, respectively. It had the capability to train a speech synthesis model end-to-end with text and audio pairings, avoiding the laborious feature engineering steps. Google later proposed Tacotron2, another end-to-end speech synthesis system that integrated the WaveNet vocoder with the updated Tacotron architecture.

Deep Learning end-to-end methods in Speech Synthesis have garnered attention in the research community due to its powerful model capacity allowing it to capture latent information within the data with minimal feature engineering. Unlike concatenative synthesis, it has a smaller footprint as it eliminates the need to maintain large databases. However, most end-to-end methods are slow during inference due to their auto-regressive nature. Lack of sufficient data can lead to generation of output speech that is not robust and has the problem of a flat prosody if not conditioned properly.

Chapter 3

Related Work

As discussed in Section 2.5, the interest in the development of end-to-end trainable TTS models have been substantial. These architectures do not rely on feature engineered intermediate representations, but are directly trainable from large datasets of <*text, audio*> pairs [12, 13].

In 2017, Wang et al. introduced Tacotron, a sequence-to-sequence model combined with the attention mechanism[14] capable of creating a spectrogram from a character sequence, further decreasing the reliance on domain expertise. Tacotron used the Griffin-Lim method as the vocoder to convert the predicted spectrograms to audio waveforms.

The arrival of WaveNet (van den Oord et al., 2016) marked a significant advancement in TTS. Given a raw signal input, WaveNet uses a deep convolutional neural network architecture to synthesize an audio output autoregressively, one sample at a time. This is done by sampling a signal value from a softmax distribution, which is then encoded using the μ-law companding and quantized to 256 potential values. WaveNet uses stacked one-dimensional dilated causal convolutions. Unlike standard convolution which does not take into account the direction of convolution, causal convolution guarantees that the output at time step t is derived only from the inputs until time t–1. The exponentially increasing dilation factor with every layer depth allows for a very broad receptive field and increased non-linearity required to produce raw audio.[15] By blending Tacotron's prosody with WaveNet's audio quality, Tacotron 2 achieved naturalness close to that of human speech. It replaced the Griffin-Lim with WaveNet as a vocoder to invert spectrograms which were generated by an encoder-decoder architecture similar to Tacotron. However, it was not designed to provide multi-speaker support [16].

Gibiansky et al. enhanced the Neural TTS architecture with a trainable low-dimensional speaker embedding to generate a multi-speaker TTS. They demon-

strated that this single neural TTS model can learn hundreds of distinct voices, achieving high audio quality while simultaneously preserving the speaker identities. Deep Voice 3[17] proposed a simple, fully convolutional encoder-decoder architecture which provided high efficiency during training along with fast model adaptation, providing multi-speaker support for over 2400 speakers from the LibriSpeech dataset.

These methods were only capable of learning a restricted range of speaker embeddings and hence only allowed voice synthesis of target speakers seen during training. As a novelty, VoiceLoop[17] instead suggested a new architecture based on a memory buffer of a fixed size, which could generate voices from speakers that were not observed while training. However, it required tens of minutes of transcribed speech during the enrollment of a new speaker in order to obtain good results.

Recent developments have utilized few-shot speaker adaptations where just a few seconds of untranscribed speech audios per speaker could be used to generate a cloned voice corresponding to that speaker[2]. Extending Deep Voice 3, Arik et al., 2018, proposed a method to achieve Voice Cloning from a Multi-Speaker Generative Model. It highlights two approaches to Voice cloning, namely Speaker Adaptation and Speaker Encoding, and provides a comparison between the two. The speaker adaptation method is similar to VoiceLoop, where the model parameters, as well as the speaker embeddings are fine-tuned on a small adaptation data. The speaker encoding method uses a neural network to generate a speaker embedding vector from a spectrogram. The naturalness of speech and similarity to the target speaker is slightly better for the Speaker adaptation method, whereas the speaker encoding approach performs better in terms of cloning time and memory footprint, making it beneficial for low-resource deployment.

While the existing multi-speaker TTS systems relied on speaker embeddings stored in Lookup Tables, Nachmani et al. [18] extended the VoiceLoop to remove the need for LUTs. This was done by introducing an additional speaker encoding network to predict the target speaker embedding. The encoding network is jointly trained along with the synthesizer network using a triplet loss to ensure that the embeddings generated from the utterances by the same speaker are closer in the latent embedding space than those computed from other speakers. In addition, other consistency losses are used to ensure that the synthesized speech encodes to an embedding similar to that of the adaptation utterance.

A similar network was introduced by Skerry-Ryan et al. [19], to transfer target prosody to the synthesized speech. A latent embedding is learned in the space of prosody by a reference encoder, obtained from a reference acoustic signal containing the prosody to be replicated. A Tacotron synthesizer is then conditioned on this learned embedding space, resulting in synthesized audio incorporated with prosody resembling that of the reference signal, even if different speakers are used in the reference and synthesized speech.

Wang et al. [20] introduced Global Style Tokens, a set of style embeddings that learn to simulate a wide variety of acoustic expressiveness. It is jointly trained within Tacotron without the use of any explicit labels. Global Style Tokens provide a wide range of notable outcomes, regulating the synthesis by changing the pace and style of speech, irrespective of the textual content. They may as well be used for style transfer, which involves mimicking the speaking style corresponding to a reference audio clip over an entire corpus of text. Global Style Tokens paved the way for highly scalable and robust speech synthesis, as it learns to factorize speaker identification and noise when trained on unlabeled, noisy data.

Jia et al. explored a Transfer Learning method from Speaker Verification to Multipeaker TTS, similar to the speaker encoding models in [2, 18]. Unlike [18], which utilized a speaker-discriminative network all of whose components were trained jointly, SV2TTS proposed transfer learning from a pre-trained speaker verification model. A separate network independently trained on a large dataset of untranscribed audio from 18k speakers is utilized for a speaker verification task, using a state-of-the-art generalized end-to-end loss [21].

Chapter 4

Background

4.1 Sequence-to-Sequence Models

Deep Neural Networks, despite their power and flexibility, can only be used to solve problems where the inputs and the corresponding targets can be meaningfully encoded with vectors of fixed dimensionality. This is a severe limitation, as many sequential tasks like machine translation and speech recognition are best described with sequences whose durations are not known in advance.

Sutskever et al., 2014 [22], proposed a Sequence-to-Sequence approach, which is an end-to-end sequence learning approach with minimal assumptions about the sequence structure. As part of this strategy, a multilayered Long Short-Term Memory (LSTM) is utilized to map the input sequence to a single-dimension vector before decoding the target sequence from the vector with another LSTM [22].

4.2 Zero-Shot Learning

Zero-shot learning (ZSL) is an approach in machine learning aimed at classifying objects whose instances have not been encountered during the training procedure, i.e. in the event of occurrence of disjoint training and test classes. This type of learning is usually inspired by observations on how humans learn and recognize entirely unseen classes when given a high-level description of the object. Thus, it appears plausible to believe that humans can transfer their knowledge between activities that are comparable but not identical (multi-task learning).

The motivation behind ZSL is to address the tasks in which adequate labeled

data is not available in the training dataset to cover all of the classes to be categorized, as well as in multi-task problems wherein the available training data does not provide samples of outputs that are desirable for certain tasks to be accomplished[23].

Zero-shot methods associate the observed and unobserved classes by using some type of auxiliary information that encodes the discriminative features of objects. For instance, if an AI has been trained to detect horses but has never observed a zebra, it may still detect a zebra if it also knows that zebras appear like striped horses. This is possible when it is provided with a series of animal images to be classified along with supplementary textual descriptions of the appearance of the animals. ZSL has garnered a lot of attention in the fields of computer vision, machine perception, and natural language processing.

There are two common approaches used to solve the zero-shot recognition problems.

- **Embedding Space approach:** The fundamental idea of this method is to use a projection function trained using deep neural networks to map semantic features of the data (image, text or audio) into a shared fixed-dimensional embedding space. During training, the aim is to find a projection function from visual space to semantic space (that is, word vectors or semantic embedding) using information from the observed categories.

  During inference, the input features from an unseen category are fed into the trained model, and an embedding is obtained which summarizes the input feature into a compact representation that describes the corresponding class. In order to perform the classification on the predicted output of the network, nearest-neighbor search or a similarity scoring metric is used in the latent embedding space to identify the most similar result. The class of the input feature is predicted as the class which corresponds to the nearest embedding in the latent space.

- **Generative model approach:** The purpose of the generative technique is to utilize semantic attributes to generate features for unseen categories. In most cases, a conditional generative adversarial network (cGAN) is employed, which generates features based on the semantic attribute corresponding to a given class.

4.3 Attention

Given the richness of human language, automatic or machine translation is probably one of the most complex tasks tackled by artificial intelligence. Neural machine translation, proposed by Kalchbrenner and Blunsom, 2013 [24], is an approach to solve this task. In contrast to the traditional systems based on phrase-based translations [25] consisting of multiple smaller sub-components, neural machine translations attempt to train one single neural network which reads in a sentence and provides the correct translation.

A majority of the models proposed for translation follow an encoder-decoder based architecture [22, 26], constituting of an encoder and a decoder for each language, or involving the use of a language-specific encoder, the outputs of which are then compared [27]. The encoder neural network learns a meaningful representation from the source sentence and compresses all the information into a fixed-length encoded vector. The decoder network then reads the encoded vector and outputs a translation. The entire encoder-decoder for a certain language pair is jointly trained to maximize the probability of obtaining a correct translation for a source sentence.

The fact that a neural network must be able to compress all of the essential information from a source phrase into a single fixed-length vector is a serious drawback of the encoder–decoder architecture. Long sentences, particularly those that are longer than those in the training corpus, may be challenging for the neural network to adapt to. Cho et al., 2014 [26] demonstrated that when the length of an input phrase rises, the performance of a simple encoder–decoder degrades rapidly.

To address this bottleneck caused by the fixed-length encoded vector, Bahdanau et al., 2014 [14] proposed an extension to the encoder–decoder, referred to as the Attention model, that learns an alignment during translation. The suggested approach determines a set of locations in a source phrase where the most significant information is concentrated. Contrary to the basic encoder–decoder model, it does not make an attempt to encode the entire input sentence into one single fixed-length vector. Instead, the input sentence is encoded into a series of vectors. A subset of these vectors are adaptively chosen during the decoding step of the translation. This liberates the model from having to compress the information contained in a variable-length source sentence, into a fixed-size vector. This helps the model to better adapt to longer sentences[14].

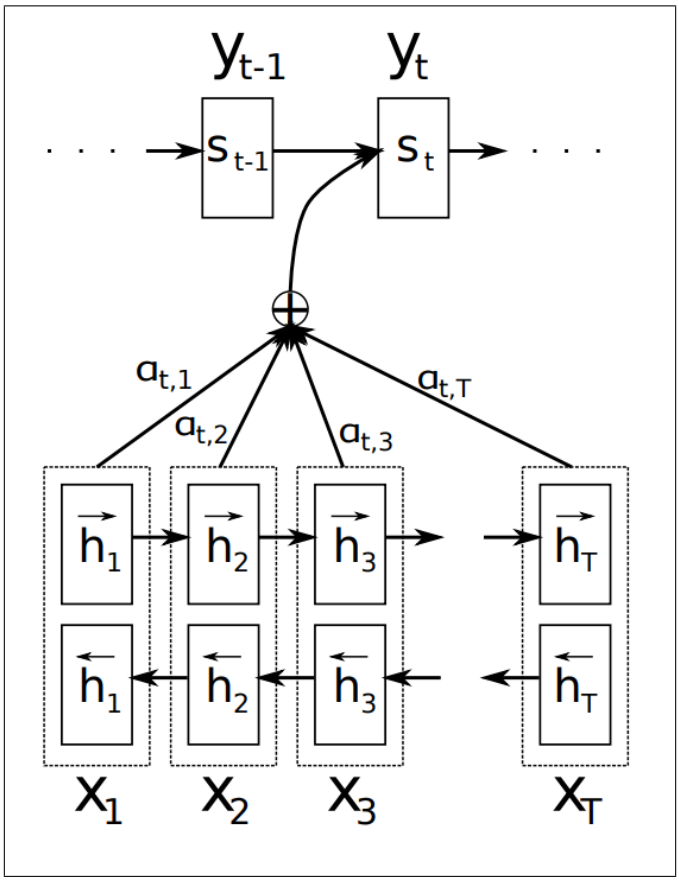


Figure 4.1 Illustration of the Attention mechanism generating the t-th target word y_t provided with a source sentence $(x_1, x_2, \ldots, x_T)$. This figure is extracted from Bahdanau et al., 2014 [14].

The context vector c_i is computed from a sequence of annotations

$$(h_1, h_2, \ldots, h_{T_x})$$

These annotations are generated as encoder outputs at each time-step of the input sentence. Each annotation h_i contains information about the entire input sequence, primarily focusing on the region surrounding the i-th word of the input sentence. The context vector is then calculated as the weighted sum of the annotations:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{4.1}$$

The weight a_{ij} corresponding to annotation h_j is computed as:

$$\alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T_x} \exp\left(e_{ik}\right)} \tag{4.2}$$

where,

$$e_{ij} = a(s_{i-1}, h_j)$$

is an alignment model which scores how well the encoded inputs around position j match with the current decoder output at position i.

4.4 Autoregressive Models

The word autoregressive is derived from the literature pertaining to time-series models, in which the observations from prior time-steps are utilized to predict values at the present time-step. Deep autoregressive architectures have witnessed a lot of success in recent architectures like PixelCNN[28] and WaveNet[15].

Given a dataset $\mathcal{D}$ with n-dimensional data-points $\boldsymbol{x}$, a sequential technique is used to sample from an autoregressive model. We sample x_1 first, then x_2 conditioned on the previously sampled x_1, then x_3 conditioned on both x_1 and x_2. This continues until we sample x_n conditioned on all the previously sampled $x_{<n}$.

Sequential sampling may be a costly procedure especially for applications that generate high-dimensional data in real-time, as in audio synthesis. This is the primary reason that makes autoregressive networks like WaveNet[15] extremely slow during inference. Several other papers like Parallel WaveNet[29] have proposed methods to sidestep the use of this expensive sampling process using Inverse Autoregressive Flows to allow for more efficient sampling [29].

4.5 Teacher Forcing

Recent advances image captioning and machine translation show that recurrent neural networks may be taught to create sequences of tokens given some input. The recurrent network models data through a fully-observed directed graphical model, decomposing the distribution over the discrete time-sequence $y_1, y_2, \ldots . y_T$ into an ordered product of conditional distributions across tokens.

$$P\left(y_1, y_2, \ldots y_T\right) = P\left(y_1\right) \prod_{t=1}^{T} P\left(y_t \mid y_1, \ldots y_{t-1}\right) \tag{4.3}$$

The most prevalent training technique is by far the concept of maximum probability. The utilization of a teacher signal $d(t)$ in place of the predicted output $y(t)$ is a noteworthy technique often employed in dynamic supervised learning problems. This teacher signal $d(t)$, whenever present, is applied in the subsequent calculation of the behavior of the network. This method of training is often referred to in RNN literature as Teacher forcing [30] because the ground-truth samples y_t are reintroduced to the model as the teacher signal. It is a form of feedback or conditioning employed to predict the future outputs, which forces the predictions of the RNN to remain closer to the original ground truth sequence.

Teacher Forcing allows for faster convergence. The predictions for a model are usually quite dismal at the early phases of training. Without employing this technique, a series of incorrect predictions will alter the hidden states of the model, accumulating and amplifying the errors, making it harder for the model to learn something constructive.

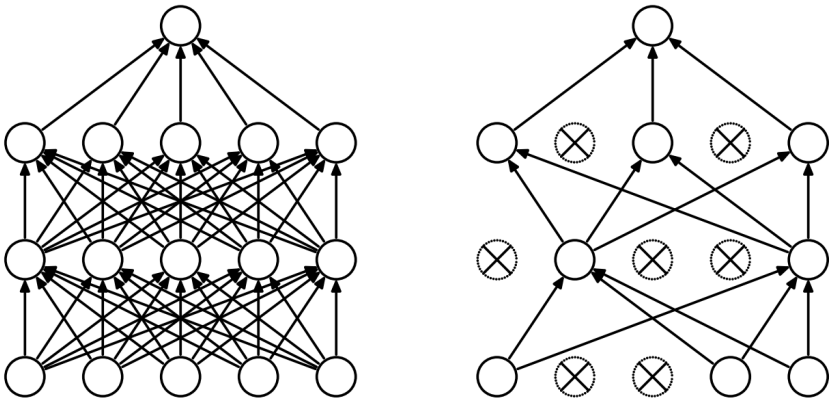
However, during inference, ground-truth tokens are not available. Hence, the previous unknown token is substituted by the token predicted by the model itself. This mismatch between the training and inference phases might result in small prediction errors that build up rapidly along the produced sequence. Bengio et al. proposed a curriculum learning approach to gradually transition from an entirely guided training scheme utilizing the true previous token, towards a less guided scheme that mostly uses the generated token instead. Experiments on a variety of sequence prediction problems demonstrate that this method improves performance significantly[31].

4.6 Dropout

With the rising complexities of the problems that Machine Learning systems must answer, big networks with a large number of parameters, are becoming more commonplace. These deep neural networks, despite their immense capability, are prone to heavy overfitting, resulting in poor generalization. To avoid this, predictions from numerous large neural networks could be combined during

inference. However, deep networks are particularly sluggish to execute, making this approach extremely challenging. Srivastava, et al., 2014, proposed Dropout, an approach to resolve the problem of overfitting in neural networks[32].

During training, neurons (together with their connections) are dropped at random from the neural network. In a standard neural network, excessive co-adaptation among individual units develop during learning through back-propagation. This becomes relevant only for the training data and the model fails to generalize to data that it has not encountered before. By introducing uncertainty to the presence of a particular neuron in a hidden layer, random dropout disrupts these co-adaptations. Dropout adds a new hyperparameter to the equation: the probability p of retaining a neuron. The intensity of dropout is controlled by this hyperparameter. If $p = 1$, there will be no dropout. A lower value of p implies more dropout. For concealed units, typical values for p range from 0.5 to 0.8. For the input layers, The type of input influences the choice of the value of p.



(a) Standard Neural Net (b) After applying dropout.

Figure 4.2 A Neural Network Model **(a)** Without dropout **(b)** With dropout applied. The units that are crossed have been dropped. This figure is extracted from Srivastava et al., 2014 [32].

This method has been identified to improve the performance of neural nets in a broad range of domains like object classification, speech recognition, document classification, digit recognition, etc. On ImageNet, SVHN, MNIST, and CIFAR-100, methods applying dropout produce state-of-the-art results. On other data sets, dropout significantly enhanced the performance of regular neural nets.

4.7 Zoneout

As seen by the frequent usage of early stopping and the success of approaches like dropout and its recurrent versions, regularizing neural nets can greatly enhance performance. Recurrent Neural Networks (RNNs) use an input-dependent transition operator to fold incoming observations into their hidden state. This enables them to create a fixed-length representation of any arbitrary-length sequence. The use of the same transition operator at different time-steps in a sequence can, however, make an RNN's dynamics sensitive to slight changes in the hidden state.

Krueger et al., 2017, introduced Zoneout in a bid to improve the robustness of RNN's to perturbations in the hidden state, so as to regularize the transition dynamics [33]. It is a simple regularizer for RNNs that stochastically preserves the activations of hidden units.

Similar to dropout, zoneout introduces noise during the training step. However, rather than setting the activation of certain units to 0, as in dropout, zoneout randomly substitutes the activation values of some units with their activations from the preceding time-step. At test time, the expectation of the random noise was employed, just as it was in dropout. This yields a straightforward regularization method that can be used for any RNN architecture and can be theoretically extended for use in any model whose state changes over time. When compared to dropout, zoneout is attractive since it maintains information flow over the network in both forward and backward directions. This helps to tackle the problem of vanishing gradients[33].

Zoneout outperforms numerous alternative regularizers to produce results competitive with state-of-the-art on the Text8 and Penn Treebank datasets, as well as state-of-the-art on pMNIST. Although trying out several zoneout values can help fine tune zoneout for each task, low zoneout probability values (0.05 - 0.2) on states consistently improve the performance of existing models.

Chapter 5

Voice Cloning - Methodology

5.1 Overview

Our implementation of Voice Cloning is based on "Transfer Learning from Speaker Verification to Multispeaker Text-to-Speech Synthesis", referred to as SV2TTS (Jia et al., 2018). Some modifications have been made to the baseline architecture, the most significant being the replacement of the WaveNet based Vocoder by WaveGlow. The primary idea is to offer faster inference speeds, without sacrificing on performance. SV2TTS illustrates a zero-shot voice cloning framework, requiring reference speech of only 5 second duration. The SV2TTS paper combines three prior works from Google, namely, the Generalized End-to-End (GE2E) Loss [21], Text-to-Speech model based on Tacotron [13] and a Vocoder based on WaveNet [15].

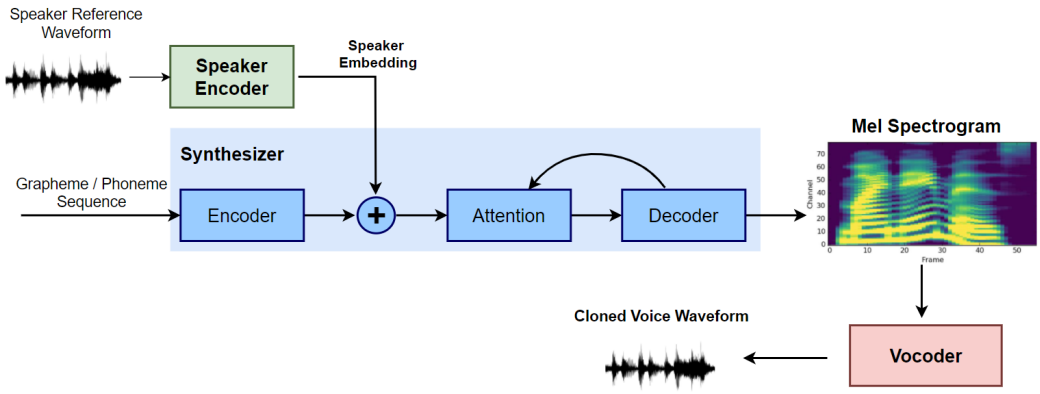


Figure 5.1 The SV2TTS architecture represented as a block diagram. The green, blue and red blocks denote the Encoder, Synthesizer and Vocoder units respectively.

The combined framework, as shown in Figure 5.1, consists of a three-stage pipeline, where each stage corresponds to the models in the order described:

- A speaker encoder network that generates a latent embedding from a short utterance corresponding to a particular speaker. This latent embedding is innately a compact representation of the voice of the speaker. Similar to the idea of word embeddings, the latent embeddings for similar voices will be closer together in that N-dimensional latent space. The speaker encoder model is trained using the GE2E loss based on a speaker verification task, which is described in Wan et al., 2017 [21] and Heigold et al., 2015 [34].

- A Text-to-Speech Synthesizer which when conditioned on the speaker embedding, generates a spectrogram from the text transcript provided. This model is based on the popular Tacotron 2 architecture [16].

- A vocoder, based on WaveNet [15], that generates an audio waveform from the synthesized spectrograms. We have used WaveGlow as the vocoder in our implementation to allow faster inference.

A nice characteristic of this framework is that the individual models can be trained independently and, if needed, on inherently distinct datasets as well. For the speaker encoder model, it is essential to make it robust to noise, while accurately capturing the various traits of the human voice. To achieve this, the encoder needs to be trained on a large dataset comprising of diverse speakers. The dataset need not be transcribed, as the linguistic content of the speech is not of particular importance while interpreting the voice characteristics. This follows as a result of the encoder being trained using the GE2E loss, requiring nothing but the speaker identity. The problem to be solved thus becomes a speaker verification task. However, instead of explicitly classifying speakers, we use the latent embedding generated by the model as a meaningful, concise representation of the speaker's voice. This embedding is thus an appropriate candidate for conditioning the synthesizer on a voice. For the synthesizer and the vocoder, transcripts are required, and the quality of the generated audio can be expected to be only as good as that of the data used in training. High-quality audios and annotated datasets are thus preferable while training the Synthesizer and Vocoder networks. However, for the speaker encoder, there is also no strict requirement on the level of noise in the audios, and a noisy dataset should technically make the model robust to the presence of any noise in the reference speech provided during inference.

5.2 Problem Definition

Let us consider a dataset containing fixed size utterances of multiple speakers grouped by the speaker IDs. The utterances are all in the waveform domain from which log-mel spectrograms are computed to be used as features for the encoder.

The j$^{\text{th}}$ utterance from the i$^{\text{th}}$ speaker is denoted as $\mathbf{u}_{ij}$, and the corresponding mel-spectrogram is denoted as $\mathbf{x}_{ij}$. The encoder $\mathcal{E}$ computes an embedding $e_{ij} = \mathcal{E}(\mathbf{x}_{ij}; \mathbf{w}_{\mathcal{E}})$ from the utterance $\mathbf{u}_{ij}$ ($\mathbf{w}_{\mathcal{E}}$ being the parameters of the encoder). The centroid embedding $\mathbf{c_i}$ is calculated for each speaker as the average of all the utterance embeddings.

With the speaker embeddings generated, the Synthesizer $\mathcal{S}$ can be used to predict an approximate of $\mathbf{x}_{ij}$, provided with the speaker embedding $\mathbf{c}_i$ and the text transcript $\mathbf{t}_{ij}$ for the utterance $\mathbf{u}_{ij}$. Thus, we obtain $\hat{\mathbf{x}}_{ij} = \mathcal{S}(\mathbf{c}_i, \mathbf{t}_{ij}; \mathbf{w}_{\mathcal{S}})$, where $\mathcal{S}$ is parametrized $\mathbf{w}_{\mathcal{S}}$.

A vocoder $\mathcal{V}$, defined by the parameters $\mathbf{w}_{\mathcal{V}}$, can now be used to approximate the utterance $\mathbf{u}_{ij}$ from the mel-spectrogram $\hat{\mathbf{x}}_{ij}$ predicted by $\mathcal{S}$. This is then denoted as $\hat{\mathbf{u}}_{ij} = \mathcal{V}(\hat{\mathbf{x}}_{ij}; \mathbf{w}_{\mathcal{V}})$.

Training this entire framework in a true end-to-end approach would require the minimization of the following loss function directly in the waveform domain:

$$\min_{\mathbf{w}_{\mathcal{E}}, \mathbf{w}_{\mathcal{S}}, \mathbf{w}_{\mathcal{V}}} L_{\mathcal{V}} \left(\mathbf{u}_{ij}, \mathcal{V} \left(\mathcal{S} \left(\mathcal{E} \left(\mathbf{x}_{ij}; \mathbf{w}_{\mathcal{E}} \right), \mathbf{t}_{ij}; \mathbf{w}_{\mathcal{S}} \right); \mathbf{w}_{\mathcal{V}} \right) \right) \tag{5.1}$$

However, this end-to-end technique poses multiple challenges and is therefore not suitable for the task of Voice Cloning [35].

- It necessitates training all the three models (Encoder $\mathcal{E}$, Synthesizer $\mathcal{S}$ and Vocoder $\mathcal{V}$) on the same dataset, which should ideally fulfill the desired criteria for all three models. The encoder requires a large number of speakers in the dataset, without the need for transcripts. However, the synthesizer requires a *<text, audio>* pair in order to predict the mel-spectrograms.
- The tolerance level for the amount of noise present in the audio also varies for the Encoder and the Synthesizer. While the Encoder network can generalize well on noisy audio, the Synthesizer would ideally require cleaner input speech.

- The Tacotron synthesizer requires substantial training time before it starts generating meaningful alignments. Thus, the convergence of the entire model might be a difficult task to accomplish.

Hence, it would be reasonable to assume that training the models separately on different datasets would result in better generalization in a zero-shot setting.

The encoder, in contrast to the vocoder and synthesizer, does not have any labels for training. The only label of interest is the Speaker Id's for the speech audios used in a batch. This task can be portrayed as capturing a meaningful representation for the voice of a speaker obtained from the utterances of the speaker. Using an autoencoder could be challenging for this task, as the upsampling model would become aware of the textual content of the audio, rather than just the characteristics relevant to the speaker's voice.

The GE2E Loss (described in Section 5.3.2) helps solve this task, and train the Speaker Encoder network separately without any dependence on the Synthesizer. Having the pre-trained Speaker Encoder, the synthesizer can now be trained to predict the mel-spectrograms directly from the speaker embeddings and transcripts of the input speech audio, such that the loss function $L_{\mathcal{S}}$ is in the time-frequency domain.

$$\min_{\mathbf{w}_{\mathcal{S}}} L_{\mathcal{S}} \left(\mathbf{x}_{ij}, \mathcal{S} \left(\mathbf{e}_{ij}, \mathbf{t}_{ij}; \mathbf{w}_{\mathcal{S}} \right) \right) \tag{5.2}$$

The Vocoder can now be trained on the ground-truth spectrograms or on the predicted spectrogram by the Synthesizer network [35].

$$\min_{\mathbf{w}_v} L_{\mathcal{V}} \left(\mathbf{u}_{ij}, \mathcal{V} \left(\mathbf{x}_{ij}; \mathbf{w}_{\mathcal{V}} \right) \right) \text{ or } \min_{\mathbf{w}_v} L_{\mathcal{V}} \left(\mathbf{u}_{ij}, \mathcal{V} \left(\hat{\mathbf{x}}_{ij}; \mathbf{w}_{\mathcal{V}} \right) \right) \tag{5.3}$$

On noisier datasets, using the spectrograms generated by the synthesizer to train the vocoder can improve the performance of the model [3].

5.3 Speaker Encoder

The speaker encoder aims to produce a latent, fixed-dimensional embedding that would then be used to condition the synthesizer network on a reference speech audio pertaining to the preferred target speaker. To achieve good generalization across different speakers, the latent embedding needs to capture the various characteristics of human voice, regardless of its background interference or phonetic content. A speaker-discriminative model is described in numerous papers (Heigold et al., 2015; Wan et al., 2017; Jia et al., 2018) [34, 21, 3]. In our implementation, we train the speaker encoder model on a transcript-independent speaker verification task as described in the SV2TTS paper.

5.3.1 Model Architecture

The encoder model comprises of a 3-Layer Bidirectional LSTM with 128 hidden nodes followed by a fully connected (Projection Layer) of 64 units. The final embedding is the L2-normalized output of the last layer. The input audios are broken down into 1.6 second utterances from which 40-channel log-mel spectrograms with a 10ms step and a 25ms window width are computed and passed as input to the model. The training dataset only includes the speaker labels without any need for transcripts. The network then converts the series of log-mel spectrogram frames into a fixed-dimensional latent embedding vector, termed as the d-vector [34].

5.3.2 Generalized End-to-End loss

The encoder network is trained on a speaker verification task and aims to optimize a GE2E loss, such that embeddings of utterances from the same speaker are closer together in the embedding space, having higher cosine similarity, while those belonging to different speakers are farther apart with lower cosine similarity.

Speaker verification is an application in biometrics in which the identity of an individual is validated through their voice. Before verification can be performed, a speaker needs to enrol his voice from a few short utterances. This allows a template to be created for an individual by deriving their speaker embedding. During verification, the user validates himself with a short input utterance from which the system generates an embedding and compares it with the enrolled speaker embeddings. The user is granted access to the device if the similarity crosses a certain predefined threshold. The Generalized End-to-End Loss helps

in simulating this process and optimizing the model.

The training is carried out in the form of a batch containing N speakers, and M utterances from each speaker in the corresponding batch, resulting in a total of $N * M$ utterances per batch. The feature vector $\mathbf{x}_{ij}$ $(1 \leq i \leq N, 1 \leq j \leq M)$ represents the feature extracted from the j$^{\text{th}}$ utterance of the i$^{\text{th}}$ speaker. The feature vectors are passed as input to the Speaker Encoder network whose architecture is defined in Section 5.3.1. Embeddings $\mathbf{e}_{ij}$ for M utterances of fixed duration, corresponding to N speakers are computed, where e_{ij} $(1 \leq i \leq N, 1 \leq j \leq M)$ represents the embedding extracted from the j$^{\text{th}}$ utterance of the i$^{\text{th}}$ speaker. We denote the output of the Speaker Encoder network as $f(\mathbf{x}_{ij}; \mathbf{w})$ where $\mathbf{w}$ represents all of the parameters of the neural network (including the LSTM layers as well as the linear layer). The embedding vector (d-vector) is computed as the L2 normalized output from the Speaker Encoder network:

$$\mathbf{e}_{ij} = \frac{f\left(\mathbf{x}_{ij}; \mathbf{w}\right)}{\left\| f\left(\mathbf{x}_{ij}; \mathbf{w}\right) \right\|_2} \tag{5.4}$$

A centroid embedding for each speaker is derived as:

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

.

A similarity matrix $S_{ij,k}$ is generated which is the result of a two by two comparison of all embeddings e_{ij} against the centroid embeddings for each speaker within a batch $\mathbf{c}_k (1 \leq k \leq N)$. This is used to compute a scaled cosine similarity:

$$\mathbf{S}_{ij,k} = w \cdot \cos\left(\mathbf{e}_{ij}, \mathbf{c}_k\right) + b = w \cdot \mathbf{e}_{ij} \cdot \left\|\mathbf{c}_k\right\|_2 + b$$

where w,b are learned parameters. These parameters are initialized to (10,-5) as suggested in the original paper. This process is depicted in Figure 5.2 illustrating the features, embedding vectors and the similarity matrix obtained from multiple speakers, portrayed using distinct colors.

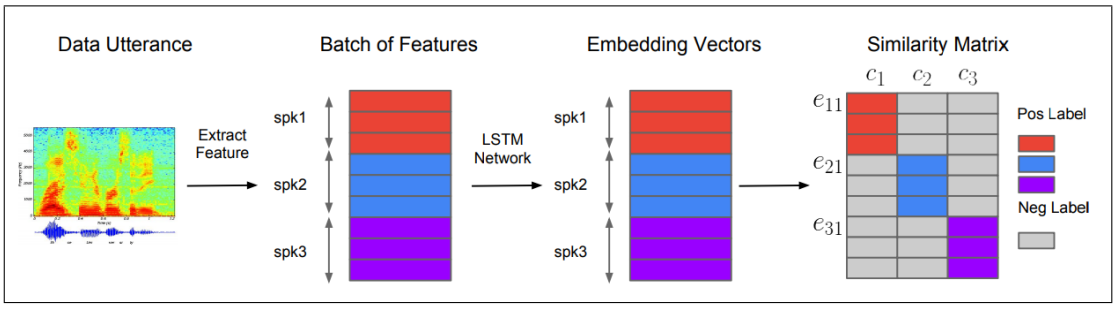


Figure 5.2 An overview of the GE2E loss calculation using a similarity matrix. Utterances from different speakers are marked with distinct colors. This figure is extracted from Wan et al., 2017 [21].

Ideally, the embedding from all utterances of the i$^{\text{th}}$ speaker should be similar to the centroid embedding ($\mathbf{c}_i$) of that particular speaker, while simultaneously being dissimilar from the centroids of other speakers. As demonstrated in Figure x, the similarity values of the colored cells along the diagonal of the similarity matrix should be large, while those of the remaining gray cells being small.

Additionally, e_{ij} needs to be removed while computing the true speaker centroid as this helps avoid trivial solutions and makes the training stable[21]. The similarity matrix is thus defined as follows:

$$\mathbf{S}_{ij,k} = \begin{cases} w \cdot \cos\left(\mathbf{e}_{ij}, \mathbf{c}_i^{(-j)}\right) + b & \text{if} \quad i = k \\ w \cdot \cos\left(\mathbf{e}_{ij}, \mathbf{c}_k\right) + b & \text{otherwise} \end{cases} \tag{5.5}$$

where $\mathbf{c}_i^{(-j)}$ are the exclusive centroids, defined as :

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

The final Generalized End-to-End loss L_G is thus the sum of all losses over all the cells of the similarity matrix ($1 \leq i \leq N, 1 \leq j \leq M$):

$$L_G(\mathbf{x}; \mathbf{w}) = L_G(\mathbf{S}) = \sum_{i,j} L\left(\mathbf{e}_{ij}\right) \tag{5.6}$$

We use N = 64 and M = 12 as parameters for the batch size, resulting in 768 features per batch. As per our experiments, a greater number of speakers per batch results in better generalization. However, it should be noted that the time complexity to compute the similarity matrix is $O(N^2M)$. Thus, this parameter should not be too large such that it substantially slows down the training, nor too small such that it affects the generalizing capability of the model. The ideal value should be one corresponding to the maximum batch size that fits on the GPU memory.

During inference, an arbitrary length speech utterance is divided into 1.6 second windows with a 50% overlap. The network then operates individually on each window, and the final utterance embedding is calculated by taking the average of all outputs. This method is illustrated in Figure 5.3.

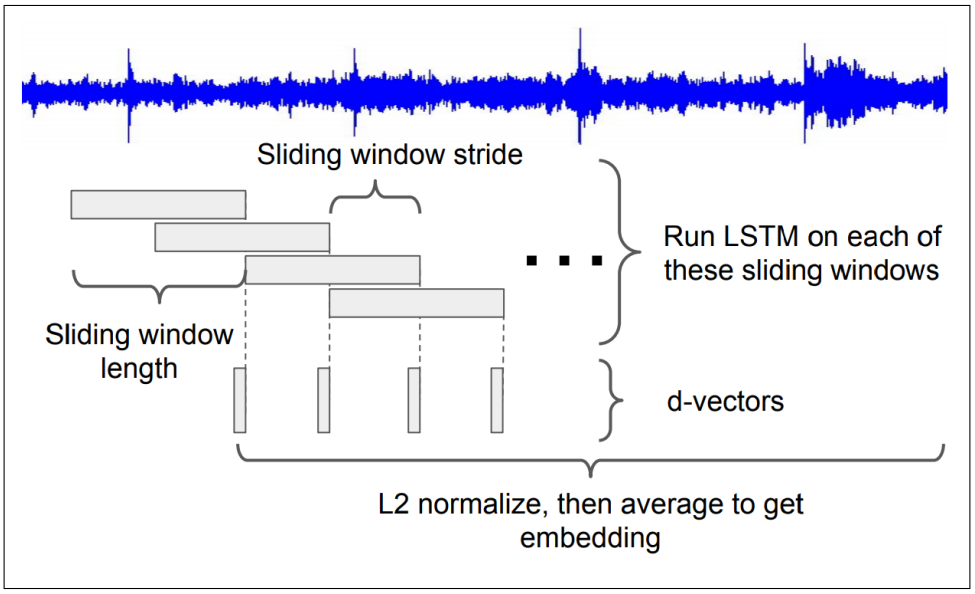


Figure 5.3 Sliding window approach used during inference. This figure is extracted from Wan et al., 2017 [21].

5.4 Synthesizer

To synthesize mel-spectrograms from text transcripts, we use the Tacotron2 architecture excluding the Wavenet. Tacotron2 is an end-to-end generative TTS model that accepts a character sequence as input and produces the corresponding spectrogram as output. Since Tacotron2 is frame-based, inference takes far less time than sample-level autoregressive approaches. Unlike earlier methods, it does not require the need for hand-crafted linguistic features or components like an HMM aligner and yet it can achieve high-quality audio nearer to original human speech[13].

The architecture includes a recurrent sequence-to-sequence encoder-decoder architecture bridged by an attention mechanism. In order to support multiple speakers, modifications to the architecture are performed following a scheme similar to Deep Voice 2 [1]. The speaker embedding vector corresponding to the target speaker, obtained from the Speaker encoder network needs to be integrated into the Synthesizer network in order to transfer characteristics of the target voice. This is done by concatenating the embedding vector with the encoder output of the synthesizer network at every time-step. It has been observed that simply passing the concatenated embeddings to the attention layer is sufficient for the model to converge across various speakers.

5.4.1 Intermediate Feature Representation

As an intermediate feature representation, mel-frequency spectrogram, a low-level acoustic representation is utilized. This representation can be easily computed from time-domain waveforms and is smoother than waveform representations. Since it is phase invariant within each frame, it is easier to train with a squared error loss. Mel-spectrograms summarize the frequency information with fewer dimensions. It is computed by employing a non-linear transform to the short-time Fourier transform (STFT) frequency axis, inspired by the functioning of the human auditory system. According to research, audio frequencies are not perceived on a linear scale by humans. Variances in lower frequency ranges are more easily detected than those at higher frequencies. An auditory frequency scale like the Mel-scale highlights lower-frequency features, which are vital for speech intelligibility, while de-emphasizing the details in high-frequencies, that are mostly dominated by fricatives and other noise bursts and do not require high-fidelity modeling [16].

Linear spectrograms are lossy representations that lack phase information. Mel spectrograms are more compact and discard a lot more information, posing a difficult inverse problem. However, when linguistic and acoustic features are considered, the mel spectrogram is a simpler, low-level representation of an audio signal. As a result, neural vocoders trained on mel-spectrograms should be able to generate audio with ease.

5.4.2 Model Architecture

The synthesizer architecture consists of an encoder-decoder network connected by attention. The encoder takes the input character sequence and converts it to a latent feature representation. Instead of solely using the encoder output for decoding, we concatenate the speaker embedding to every frame of the encoder output as done in the SV2TTS paper. This makes the synthesis process "speaker-aware", augmenting the generated mel-spectrograms with the desired voice characteristics.

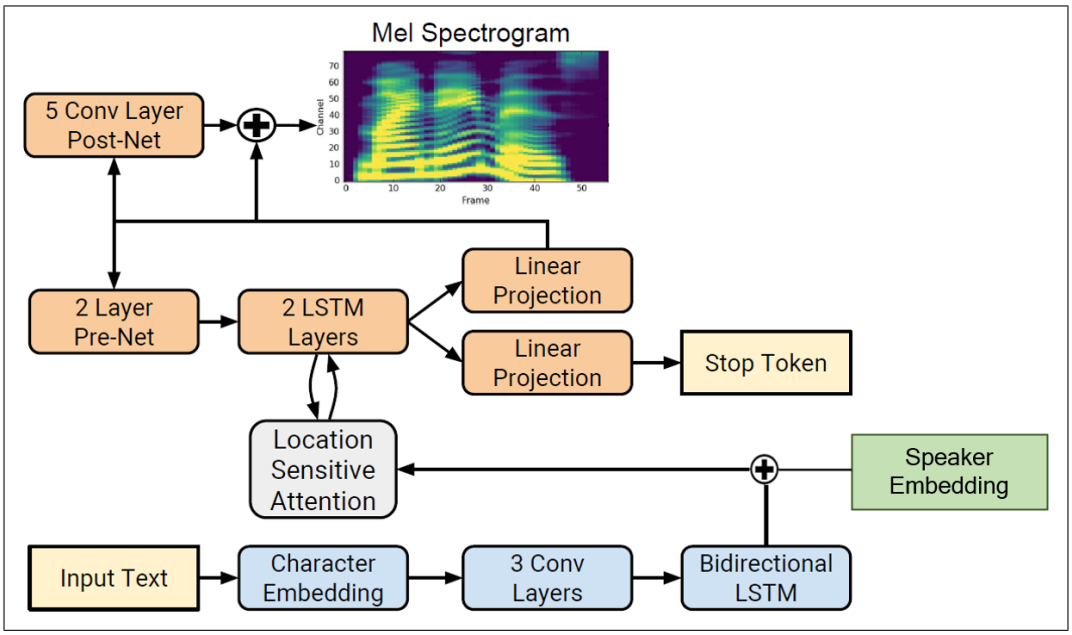


Figure 5.4 The Tacotron2 architecture conditioned on speaker-embeddings. The blue and orange blocks represent the encoder and the decoder, respectively. This figure was modified and adapted from (Shen et al., 2017)[16].

This concatenated representation is then used by the decoder to predict a spectrogram. A learned 512-dimensional character embedding is used to represent input characters, which is then passed through a stack of three convolutional layers before batch normalization [36] and ReLU activation. Each convolutional layer has 512 filters, each of which spans 5 characters and has a 5 x 1 shape. The long-term context (like N-grams) corresponding to the input character sequence is modelled by the convolutional layers.

Unlike the original Tacotron, Tacotron2 employs simpler blocks for both the encoder and decoder. A vanilla LSTM and convolutional layers are used instead of the "CBHG" stacks and GRU layers. To generate the encoded features, the output of the final convolutional layer is passed into a single bi-directional [37] LSTM [38] layer with 512 units, i.e., 256 units in each direction.

For each decoder output step, the encoder output is fed into an attention network, which summarizes the entire encoded sequence as a fixed-length context vector. The location-sensitive attention mechanism [39] adds cumulative attention weights from previous decoder time steps as an extra feature to the additive attention mechanism [14]. This urges the model to progress through the input in a consistent manner, reducing the risks of failure modes in which the decoder repeats or ignores some sub-sequences. The inputs and location features are projected to 128-dimensional hidden representations, and subsequently, the attention probabilities are calculated. A total of 32 1-dimensional convolution filters are used to compute the location features, each having a length of 31.

The decoder part of the synthesizer, is a recurrent neural network that predicts a mel-spectrogram one frame at a time from the encoded input sequence, in an autoregressive manner. A pre-net with two fully connected layers having 256 hidden units with ReLU, is used as an information bottleneck layer for the prediction generated from the preceding time-step. As mentioned by the authors, the pre-net layer is necessary for learning attention[16]. The attention context vector and the pre-net output are concatenated and passed through a stack of two unidirectional LSTM layers, each having 1024 units. To predict the target spectrogram frame, the concatenation of the LSTM output and the attention context vector is projected through a linear transform. Finally, the predicted mel spectrogram is fed into a convolutional post-net with 5 layers, which determines a residual to add to the prediction and improve the entire reconstruction. Each post-net layer is made up of 512 filters with a shape of 5 x 1. All layers except the final layer are subjected to batch normalization and tanh activations. To aid convergence, the sum of mean squared errors (MSE) before and after the

post-net is minimized.

Aside from the spectrogram frame prediction, we must also ensure that the model knows when to stop the decoding process. To predict whether the output sequence has terminated, the concatenation of decoder LSTM output and attention context is projected down to a scalar and passed through a sigmoid activation function. This "stop token" prediction is used during inference to enable the model to dynamically decide when to stop, instead of always generating frames for a fixed amount of time. The generation is deemed to be complete at the frame for which the predicted value crosses the threshold of 0.5.

Regularization is performed on all the convolutional and LSTM layers of the network using dropout[32] and zoneout[33] with probabilities of 0.5 and 0.1 respectively. Dropout is applied (with probability 0.5) only to the layers in the pre-net of the decoder. This is done to create output variation during inference.

Target spectrogram features for the synthesizer are calculated from 50ms windows with a 12.5ms step, then passed through an 80-channel mel-scale filter-bank. This is followed by a log dynamic range compression. The loss function used is a L_2 loss on the predicted spectrogram without the introduction of any other loss terms that are based upon the speaker embeddings. The input text transcripts also pass through a few pre-processing steps before being fed to the model. The Unicode text is first normalized based on canonical decomposition, removing accents by filtering out characters which belong to the "Non-spacing Mark" (Mn) category. Extra white-spaces are removed, while numbers and abbreviations are substituted by their full textual forms. The text is then padded to a maximum length using a special character "∼". A vocabulary is then constructed from all the lower and uppercase alphabets of the English language along with punctuation marks, white-space, and the stop token. Finally, the text is converted to a numeric encoding based on the indices of the corresponding characters in the dictionary. This representation is then ready to be passed as input to the synthesizer.

5.5 Vocoder

Most TTS systems consist of two modules: the Synthesizer and the Vocoder. The first module converts textual information into acoustic features while the Vocoder generates speech samples from the previously generated acoustic information.

For the generation of acoustic parameters of speech, traditional vocoder techniques typically used a source filter model [40, 41, 42]. Voiced/unvoiced segments, fundamental frequency (F0), band aperiodicities and spectral envelope were used to define the speech parameters and the Griffin-Lim algorithm used spectral representation to produce speech [43]. However, errors in parameter estimation limited the speech quality of such vocoders. Using a direct waveform modelling method, the naturalness of vocoders has recently been greatly improved. WaveNet and other neural vocoders use a generative autoregressive model to reconstruct waveforms from intermediate acoustic information with minimum loss in the quality of speech [15].

Tacotron employs the Griffin-Lim algorithm [43] for phase estimation, coupled with an inverse short-time Fourier transform to vocode the magnitude spectrograms. Griffin-Lim generates typical artifacts and inferior audio quality than systems like WaveNet, therefore it was used in Tacotron as a placeholder to be substituted by future neural vocoder approaches. WaveNet is used as the vocoder in Tacotron2 as well as in SV2TTS. Since its inception, WaveNet has been at the forefront of deep learning tasks in audio synthesis, and it continues to be the industry standard when it comes to voice naturalness in TTS. However, it is extremely slow in terms of inference time, as it relies on the generation of audio samples sequentially, one sample at a time. Numerous subsequent approaches proposed improvements to bring the generation of audio close to or faster than real-time, with little or no impact on the quality of the generated speech [29, 44, 45, 46].

In our implementation, we choose to employ WaveGlow as a vocoder. WaveGlow is a flow-based network that converts mel spectrograms to high-quality speech. WaveGlow, which eliminates the necessity for auto-regression, combines ideas from Glow[47] and WaveNet[15] to provide fast audio synthesis without compromising on quality. It contains a single network trained on a single cost function that aims to maximize the likelihood of the training data, making the training process simple and robust. Furthermore, the authors also claimed that a PyTorch implementation of WaveGlow produced samples at a rate of 520 kHz

for a 10 second speech, which was faster than any other approaches they tested. Griffin-Lim, for example, synthesized sounds at a rate of 507kHz, while Parallel WaveNet reported a rate of 500kHz. WaveNet generated speech at 0.11 kHz, which was much slower than real-time [46].

Other spectrogram inversion techniques like parallel WaveNet and Clarinet are also capable of synthesizing audios at a rate of more than 500kHz. However, unlike WaveGlow, these models are more difficult to train and use. To increase audio quality or solve difficulties with mode collapse, both require compound loss functions [29, 48]. Furthermore, Parallel WaveNet and Clarinet require two networks - a student and a teacher network. Inverse Auto-regressive Flows (IAF) are used in the student networks for both Parallel WaveNet and Clarinet. Due to the complexity of training these models towards convergence, these approaches are difficult to implement and deploy. Faster inference speeds combined with a simple, reproducible model were the primary motivations behind the use of WaveGlow as the vocoder of choice in our implementation.

5.5.1 Model Architecture

WaveGlow [46] being a generative neural network model, uses samples from a simple distribution to generate audios. An example of a simple distribution could be a zero-mean Gaussian distribution having the same number dimensionality as that of our intended output. Samples from this simple distribution are passed through a sequence of layers to transform them into the desired distribution. In our case, the distribution of the audio samples is modeled by conditioning them on a mel-spectrogram.

$$\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{z}; 0, \boldsymbol{I}) \tag{5.7}$$

$$\boldsymbol{x} = \boldsymbol{f}_0 \circ \boldsymbol{f}_1 \circ \dots \boldsymbol{f}_k(\boldsymbol{z}) \tag{5.8}$$

The model is trained by minimizing the negative log-likelihood of the data. Since the use of any arbitrary neural network would be intractable for this task, it is addressed using flow-based networks [47], which ensure an invertible mapping for the neural network. Using a change in variables, the likelihood may be determined directly by restricting each layer to be bijective.

$$\log p_\theta(\boldsymbol{x}) = \log p_\theta(\boldsymbol{z}) + \sum_{i=1}^{k} \log \left| \det \left(\boldsymbol{J} \left(\boldsymbol{f}_i^{-1}(\boldsymbol{x}) \right) \right) \right| \tag{5.9}$$

$$\boldsymbol{z} = \boldsymbol{f}_k^{-1} \circ \boldsymbol{f}_{k-1}^{-1} \circ \dots \boldsymbol{f}_0^{-1}(\boldsymbol{x}) \tag{5.10}$$

$\log p_\theta(\boldsymbol{z})$ is the log-likelihood of the spherical Gaussian which imposes a penalty on the transformed sample's l_2 norm . The second term is formed due to the change of variables, where **J** refers to the Jacobian. This term rewards any layer during the forward pass that increases the volume of the space. It also prevents a layer from multiplying the **x** terms by zero in order to optimize the l_2 norm. This sequence of transformations can be referred to as a "normalizing flow"[49]

Figure 5.5 depicts the WaveGlow model architecture, which is similar to the Glow approach[47]. Groups of 8 audios are used as vectors for the forward pass over the network, which is referred to as the "squeeze" operation[47]. These vectors are then processed through a series of "steps of flow", which constitutes an invertible 1 x 1 convolution accompanied by an affine coupling layer.

Affine Coupling Layer

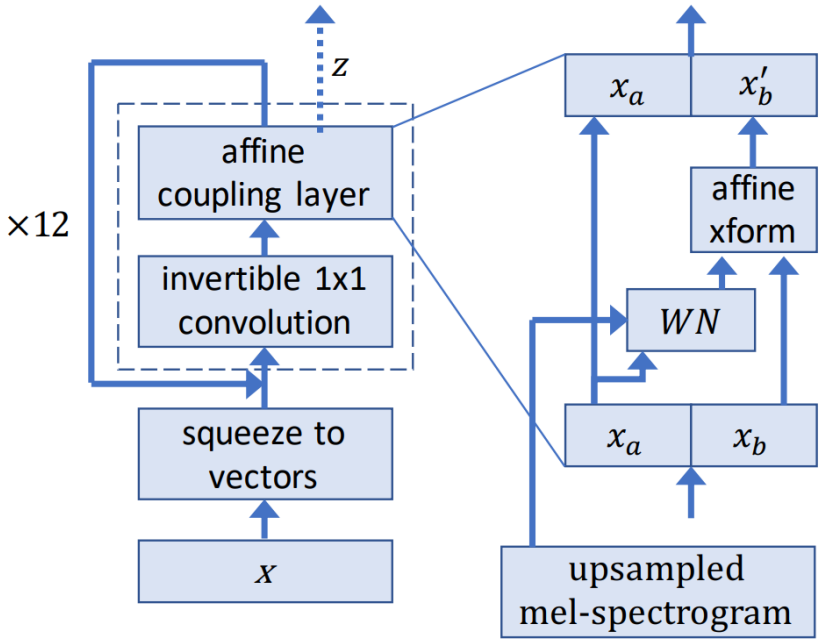


Figure 5.5 Waveglow model architecture. This figure is extracted from Prenger et al., 2019 [46]

Coupling layers are commonly used in the construction of invertible neural networks [47, 50]. In the WaveGlow model, an affine coupling layer [50] is employed. Half the channels are utilized as inputs, which generate additive and multiplicative terms. The remaining channels are then scaled and translated.

$$\boldsymbol{x}_a, \boldsymbol{x}_b = split(\boldsymbol{x}) \tag{5.11}$$

$$(\log \boldsymbol{s}, \boldsymbol{t}) = WN\left(\boldsymbol{x}_a, \textit{mel-spectrogram}\right) \tag{5.12}$$

$$\boldsymbol{x}_b\prime = \boldsymbol{s} \odot \boldsymbol{x}_b + \boldsymbol{t} \tag{5.13}$$

$$\boldsymbol{f}^{-1}_{coupling}\left(\boldsymbol{x}\right) = concat\left(\boldsymbol{x}_a, \boldsymbol{x}_b\prime\right) \tag{5.14}$$

In this case, it is possible for W N() to be any transformation and it need not be invertible as the coupling layer already ensures that the entire network remains invertible. This is due to the fact that the channels passed as inputs to WN ($\mathbf{x}_a$ in this case), are transmitted through to the layer's output unmodified. To invert the network, s and t can be computed from the output $\mathbf{x}_a$ and then inverting $\mathbf{x}_b\prime$ to calculate $\mathbf{x}_b$ by recomputing $WN\left(\boldsymbol{x}_a, \textit{mel-spectrogram}\right)$. In our model, $WN()$ is composed of dilated convolutional layers with gated-$tanh$ non-linearities along with residual and skip connections. The WN architecture resembles WaveNet [15] and Parallel WaveNet [29], with the exception that the convolutions are not causal and have three taps. The mel-spectrogram is also included in the affine coupling layer to condition the generated result based on the input. As in WaveNet, the up-sampled mel-spectrograms are inserted before the gated-$tanh$ non-linearities of every layer.

The s term in an affine coupling layer alters the volume of the mapping. A change of variables term is also added to the loss term that penalizes the model for affine mappings that are non-invertible.

$$\log \left|\det \left(\boldsymbol{J}\left(\boldsymbol{f}_{\textit{coupling}}^{-1}\left(\boldsymbol{x}\right)\right)\right)\right| = \log \left|\boldsymbol{s}\right| \tag{5.15}$$

1×1 Invertible Convolution

Channels in the same half of the affine coupling layer never directly alter one another. This would be a major limitation if information was not mixed across channels. Similar to Glow[47], WaveGlow adds an invertible 1×1 convolution layer before each of the affine coupling layers, thereby allowing mixing of information across channels. The weights (W) for these convolutions are initialized to be orthonormal and thus, invertible. The log-determinant of the Jacobian ($\mathbf{J}$) of this transformation is added as an extra term to the loss function due to the change of variables, ensuring that the convolutions remain invertible as the training advances.

$$\boldsymbol{f}_{\textit{conv}}^{-1} = \boldsymbol{W}\boldsymbol{x} \tag{5.16}$$

$$\log \left|\det \left(\boldsymbol{J}\left(\boldsymbol{f}_{\textit{conv}}^{-1}(\boldsymbol{x})\right)\right)\right| = \log \left|\det \boldsymbol{W}\right| \tag{5.17}$$

When all the terms resulting from all of the coupling layers are added together, we obtain the final likelihood as follows:

$$\begin{aligned}\log p_\theta(\boldsymbol{x}) = &- \frac{\boldsymbol{z}(\boldsymbol{x})^T \boldsymbol{z}(\boldsymbol{x})}{2\sigma^2} \\ &+ \sum_{j=0}^{\#\ coupling} \log \boldsymbol{s}_j(\boldsymbol{x},\ \textit{mel-spectrogram}\ ) \\ &+ \sum_{k=0}^{\#\ conv} \log\det|\boldsymbol{W}_k|\end{aligned} \tag{5.18}$$

The first term originates from the log-likelihood of a spherical Gaussian, σ^2 being the estimated variance of the Gaussian distribution, while the rest of the terms result from the change of variables.

During inference z values are randomly sampled from a Gaussian distribution and passed through the network. To invert the 1×1 convolutions, it is only required to invert the weight matrices. The mel-spectrograms are still included at all of the coupling layers, but the affine transforms are now inverted. The loss guarantees that all the inverses exist [46].

$$\boldsymbol{x}_a = \frac{\boldsymbol{x}_a\prime - \boldsymbol{t}}{\boldsymbol{s}} \tag{5.19}$$

Chapter 6

Resources

6.1 Data

We use the LibriTTS audio corpus as the training data for our Speaker Encoder ("clean" and "other" set), Synthesizer ("clean" set) as well as the Vocoder ("clean" set). LibriTTS is inherently a modified version of the LibriSpeech corpus, suited for Text-to-Speech applications.

The audiobooks in the LibriVox project [18] are used to create the LibriSpeech corpus [17]. This corpus contains 982 hours of speech data from 2,484 speakers. In terms of gender and speaker time, it is supposed to be fairly balanced. It can also be used for both non-commercial and commercial uses because it is published under a non-restrictive license. Despite the fact that this corpus was created for automatic speech recognition (ASR) study, it has been used in a number of text-to-speech (TTS) tasks due to its interesting qualities, such as a large volume of data, non-restrictive license, and a wide range of speakers.

A new corpus called "LibriTTS" was introduced [51] to solve several undesirable aspects of LibriSpeech that make it challenging for TTS applications. The LibriTTS corpus is based on the LibriSpeech corpus' source materials (MP3 from LibriVox and texts from Project Gutenberg) and is released under the same non-restrictive license.

- The audio files have a sampling rate of 24kHz and any audio with a sampling rate less than 24kHz is removed.
- Using Google's proprietary sentence splitting engine, book-level texts are split into sentences and then the audio is split at the corresponding sentence boundaries.
- A normalized version of the original text transcripts, normalized using Google's text normalization engine, are included in the corpus.

6.2 Tools

- **Neural Network Libraries (NNabla)**:

  Nnabla by Sony, is an open-source library for machine learning tasks. The NNabla Library can utilize both static and dynamic graph paradigms. The usage of a dynamic computation graph allows for the creation of a flexible runtime network. It also supports multi-gpu execution for distributed learning. All of the models in this project have been developed and trained using NNabla.

- **Librosa:**

  Librosa is a Python package that can be used to analyze audio and music. It contains the fundamental components required to construct information retrieval systems for music. All of the audio pre-processing and spectrogram generation tasks used in this project have been performed using librosa.

- **Audacity:**

  Audacity is a free and open-source digital audio recording and editing program compatible with most operating systems. We utilized Audacity to collect and extract audio samples of popular personalities from YouTube videos along with some of the spectrogram visualizations of synthesized audios.

- **React and Firebase:**

  React is a free, open-source front-end JavaScript library for building User Interface components for web applications, whereas Firebase provides a real-time cloud-hosted database service. We used React and Firebase Database to design the MOS evaluation platform (Figure 7.9) and store the results, respectively.

Chapter 7

Experiments and Results

7.1 Speaker Encoder

The authors of the SV2TTS paper integrated numerous noisy datasets to create a huge corpus of speech with quality comparable to what would be expected in a real-world context. LibriSpeech [52], VoxCeleb1 [53], VoxCeleb2 [54], and an internal dataset have been used in the training. LibriSpeech is a collection of audiobooks totaling 1000 hours of audio from 2400 speakers, divided into two sets, namely "clean" and "other." The clean set reportedly contains cleaner speech than the other set, despite the fact that some parts of the clean set still contain a lot of noise [51]. VoxCeleb1 and VoxCeleb2 contain audio clips derived from celebrity YouTube videos, mostly in an interview context. VoxCeleb1 has approximately 1.2k speakers, whereas VoxCeleb2 has approximately 6k. Since non-English speakers are present in both of these datasets, we did not consider them for training the encoder. Without further experimentation, it is difficult to ascertain whether having non-English speakers impacts the encoder training process. The authors also make no mention of any specifics in this regard.

To train the Speaker Encoder network, we combine speakers from the LibriTTS "clean" and "other" sets. Speakers with fewer than 20 minutes of total audio are removed. Audio samples that are less than 1.6 seconds in length are also rejected, as our model requires a minimum of 1.6 seconds to generate a speaker embedding. All audio samples greater than 1.6s are clipped into equal sized distinct utterances upto the maximum multiple of 1.6. It is also ensured that all the speakers have the same number of utterances by removing extra samples. We now arrange the dataset into batch sizes having 'N' speakers per batch, each having 'M' utterances of 1.6 seconds. This arrangement needs to be followed to facilitate the construction of the similarity matrix from which the GE2E loss is computed.

We choose N=64, and M=12 resulting in a $batch\_size = N * M = 768$. It should also be noted that the total number of speakers in the dataset must be an integral multiple of the number of speakers used in a batch (N). Similarly, the total utterances per speaker must also be an integral multiple of the number of utterances per speaker (M) in a batch.

7.1.1 Speaker Embedding space

In our implementation of the Speaker Encoder, we use 1536 speakers for training. An embedding dimensionality of 64 is used to encode the speaker characteristics. Any higher dimensionality leads to overfitting and a poor capacity to cluster unseen speakers. To verify the functionality of the speaker encoder, we selected a set of 5 utterances from 8 notable personalities from YouTube videos and passed it on to the network.

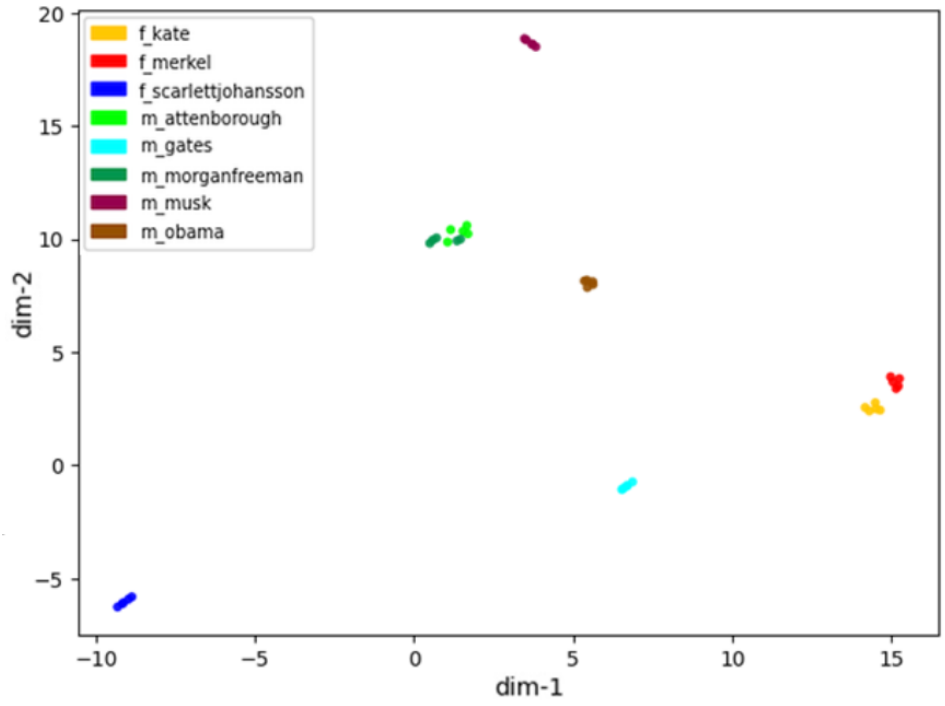


Figure 7.1 UMAP projection of the utterances of various speakers demonstrating the formation of tight clusters. Utterances from the same speaker appear as dots of the same color.

The Speaker Encoder generates a 64-dimensional vector for each utterance, which we project down to a 2-D space for visualization using a UMAP projection. UMAP, like t-SNE, is a dimension reduction technique that can be used to visualize high-dimensional data [55]. As seen in Figure 7.1, the utterances for a particular speaker form tight clusters within themselves, while being well separated from other speakers with different voice characteristics. The overlaps between the two male speakers 'm_attenborough' and 'm_morganfreeman' could be attributed to the speakers having similar vocal qualities (both have deep voices) or to the encoder not learning the characteristics of a diverse enough set of human voices. This could be addressed by including many more speakers in the dataset and simultaneously increasing the embedding dimensionality to capture all the essential features in a human voice.

If we were to be convinced that the Speaker Encoder actually learns a good representation of voices, then it should also capture the differences between male and female voices without being explicitly trained to do so. It should be emphasized that the speaker encoder is only trained on utterances and speaker IDs within a batch, with no additional information regarding gender.

Figure 7.2 Gender-based embeddings demonstrating distinct cluster formations. Red points denote Female speakers, while black represents Males.

However, as can be observed in Figure 7.2, the UMAP projection for the embeddings of the utterances from 500 Male (in black) and 500 Female (in red) speakers obtained from the LibriTTS Clean set, form two separate clusters based on the gender of the speakers. A few utterances appearing in the wrong cluster could be attributed to the high intra-utterance variations within the LibriTTS dataset, where the speakers often mimic the voices of other characters or add dramatic effects to speech[3].

We train the speaker encoder network for 800 epochs. Training on a single Nvidia GeForce 2080 Ti GPU requires approximately 8 hours to complete. To monitor the progress of the training process, we evaluate how well the model can cluster utterances from the same speakers, while simultaneously isolating them from the utterances of different speakers. We sample 12 speakers with 12 utterances from a validation set after every 20 epochs, compute the utterance embeddings, and then project them in 2-D space using UMAP.

We utilize the Silhouette Score to measure the quality of the generated clusters and accordingly guide the training process. The Silhouette score has a value ranging from -1 to 1. If the score is 1, a cluster is dense and well-separated from the other clusters. A number close to 0 denotes overlapping clusters with samples that are very close to the decision boundaries of adjoining clusters, while a negative score implies that the samples were assigned to the incorrect clusters.

The Silhouette score is computed for each sample from each cluster. Utterances having the same 'speaker_id' are considered to be part of the same cluster. We use early stopping to end the training when the Silhouette score exceeds 0.9 on the validation data. Since embeddings from different speakers are expected to be more apart in the latent space than embeddings from the same speakers, cluster formations of utterances from the same speaker are likely to appear as the training advances. This trend can be observed in Figure 7.3, which depicts the UMAP projections and the cluster formations over epochs.

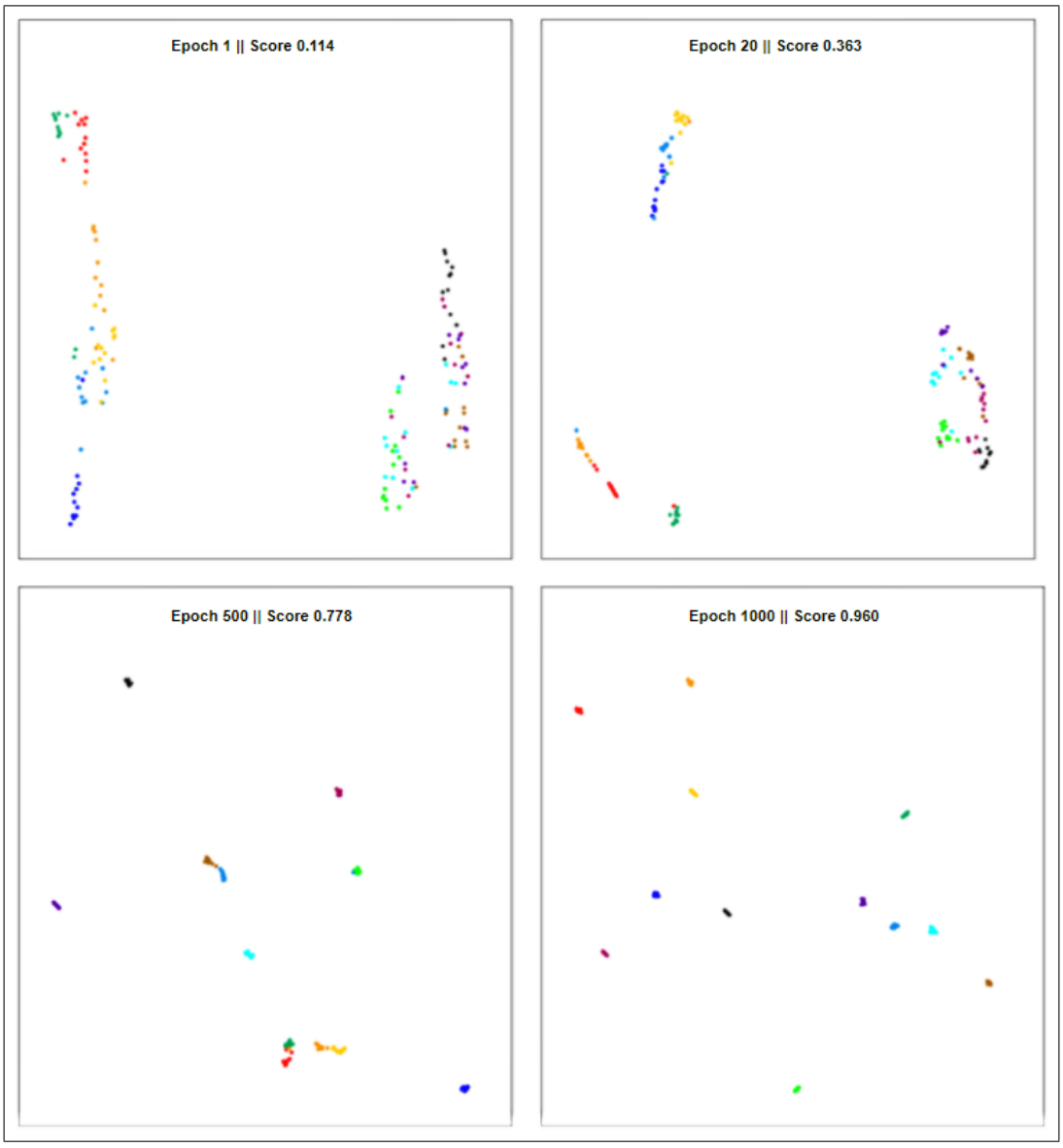


Figure 7.3 Tight clusters form with improved Silhouette scores as the training proceeds over epochs.

The cluster formations evaluated in terms of the Silhouette score are shown in Figure 7.4. As the training progresses, we notice that the scores for both the Training and Validation sets improve.

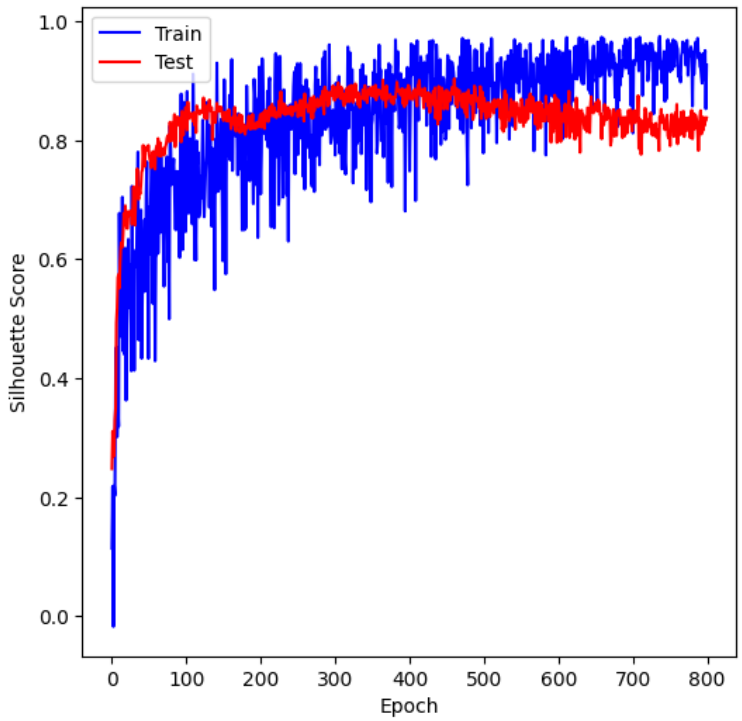


Figure 7.4 A graph demonstrating the increase of Silhouette Scores for Training and Validation dataset as the training proceeds over epochs.

7.1.2 Impact on the number of Speakers

According to the authors of SV2TTS, the ability of the speaker encoder to generalize across a wide variety of speakers likely depends on its ability to learn a good representation of a speaker. This implies that if the training dataset contains a large number of speakers, the generalization potential will be higher. Therefore, they explored the effect of the speaker encoder training set on synthesis quality by making use of other additional training sets and measuring the corresponding similarity and naturalness scores.

SE Training Set	Speakers	Embedding Dim	Naturalness	Similarity	SV-EER
LS-Clean	1.2K	64	3.73 ± 0.06	2.23 ± 0.08	16.60%
LS-Other	1.2K	64	3.60 ± 0.06	2.27 ± 0.09	15.32%
LS-Other + VC	2.4K	256	3.83 ± 0.06	2.43 ± 0.09	11.95%
LS-Other + VC + VC2	8.4K	256	3.82 ± 0.06	2.54 ± 0.09	10.14%
Internal	18K	256	4.12 ± 0.05	3.03 ± 0.09	5.08%

Table 7.1 Impact of Speaker diversity on the quality of Voice cloning. The results are obtained from Jia et al., 2018 [3].

As observed from the results in Table 7.1, naturalness and similarity improve significantly as the number of training speakers increases. Along with the subjective evaluations, the objective EER results also improve. The implications of these findings for multi-speaker TTS training are significant. Since no transcripts are required, the data requirement for the speaker encoder is much lower than for TTS training, which requires high quality, transcribed data. The audio quality in case of the speaker encoder training can also be lower than that needed for TTS training. This demonstrates that natural sounding voice cloning can be obtained by combining a speaker encoder network with a TTS network, even when the networks have different input data requirements.

In our implementation, we have combined the LibriTTS Clean and Other sets to create a dataset of 1536 speakers after filtering out speakers with less than 20 minutes of audio duration, as well as removing audio clips with a duration lesser than the required threshold of 1.6 seconds. We were unable to produce a dataset as large as the one used in the original approach since we did not have access to the internal dataset utilized by the authors of SV2TTS. However, we noted the generalization capability to increase on multiple speakers when we increased the number of speakers from 896 (purely from the Clean set) to 1536 (combination of Clean and Other sets), although no subjective evaluation was performed between the two.

7.2 Synthesizer

In the SV2TTS paper, the authors re-segmented the audio into shorter utterances by employing an ASR model to force align the audio to the transcript and breaking segments on silence. In our implementation, force-alignment and silence removal was redundant as we have trained the Synthesizer on the LibriTTS dataset where the audio and their text transcripts were automatically aligned, segmented into utterances and filtered to remove noise. LibriTTS also includes punctuation marks, that greatly help in learning prosody [51].

The SV2TTS approach uses phonemes sequences as input unlike the original Tacotron model that used graphemes. The authors claimed that it allowed for faster convergence and eliminated potential pronunciation errors caused by the model's failure to learn how to pronounce certain rare words. Our implementation of the modified synthesizer was trained directly on character embeddings. eliminating the need for an extra grapheme to phoneme conversion step. This is primarily motivated based on the deductions by Perquin et al., 2020, demonstrating that a Tacotron model trained directly on characters rather than phonemes performs similarly to a system trained on phonemes in a well-curated French dataset. This type of system embeds characters in such a way that phoneme information is encoded. As a result, Tacotron's embedded character representations could be used for a variety of purposes, including grapheme-to-phoneme conversion and control of synthesized pronunciation [56].

Our implementation of the modified Synthesizer (concatenated with the Speaker Embeddings) generates a smooth and clean alignment, that almost forms a diagonal line. To better understand what it signifies, we must note that brighter colors in the graph indicate that the decoder focuses its attention on the associated input character at that particular decoding time-step.

In a nutshell, the encoder checks each input character individually and generates status vectors. The decoder then evaluates all status vectors and generates spectrogram frames sequentially. Simply expressed, good alignment requires the decoder to output the spectrogram frame corresponding to "X" as a result of focusing on the vector created by the encoder when reading the character "X". The diagonal line occurs when frames are generated by focusing attention to the correct input characters in order.

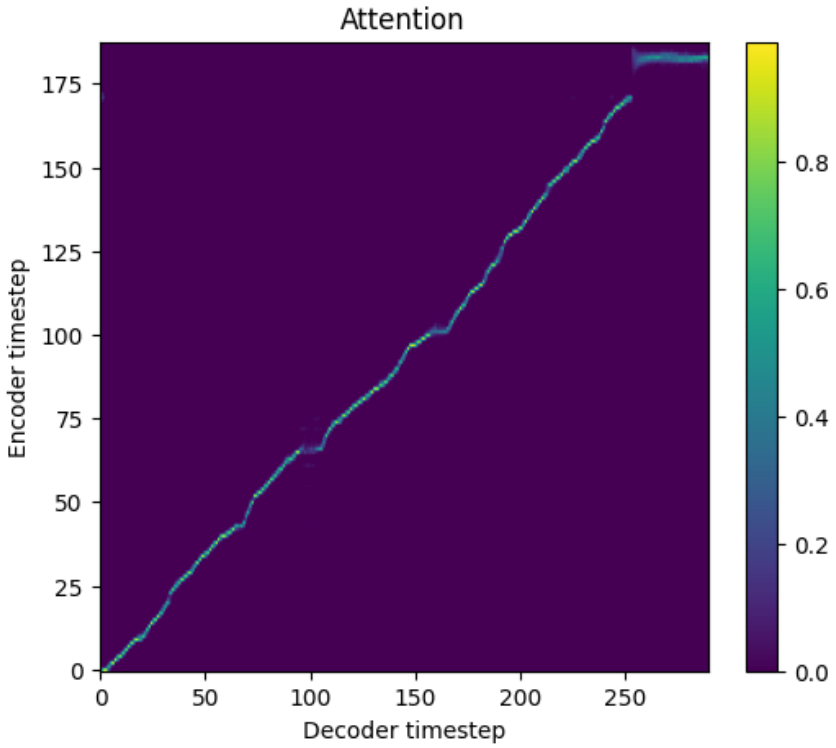


Figure 7.5 Alignment Graph observed during the training of the synthesizer

Speaker embeddings are used to condition the synthesizer during training in SV2TTS. However, we use the utterance embeddings of the same target utterance for both training and inference. This is a more natural choice as single utterances have lower intra-variation, as their scope is limited to a single sentence at most [35]. The authors acknowledge that there are frequently large variations in tone and pitch within the utterances of a single speaker in the dataset, as they imitate different characters in the books (Jia et al., 2018, Appendix B). Because of this, it is expected that embedding utterances will give us a more accurate representation of speaker voices than speaker embeddings will. Utterance embeddings are also used during inference since only a single reference speech signal is provided for voice cloning and a speaker (centroid) embedding would not be possible to compute.

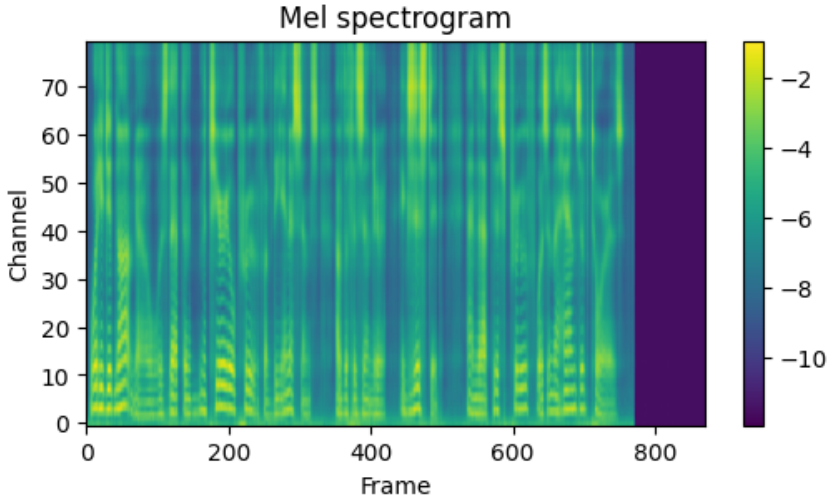


Figure 7.6 A sample mel-spectrogram generated by the synthesizer

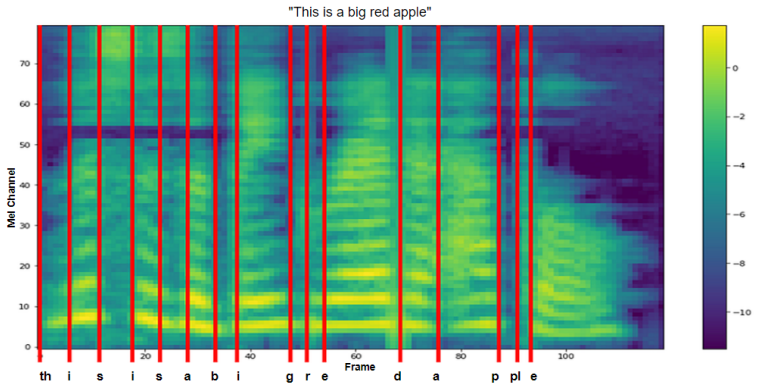
A meaningful embedding is only possible if the utterance is sufficiently long. The optimal duration of reference speech was found to be 5 seconds, as evident from the experiments conducted by the authors (Table 7.2).

	1 sec	2 sec	3 sec	5 sec	10 sec
Naturalness (MOS)	4.28 ± 0.05	4.26 ± 0.05	4.18 ± 0.06	4.20 ± 0.06	4.16 ± 0.06
Similarity (MOS)	2.85 ± 0.07	3.17 ± 0.07	3.31 ± 0.07	3.28 ± 0.07	3.18 ± 0.07
SV-EER	17.28%	11.30%	10.80%	10.46%	11.50%

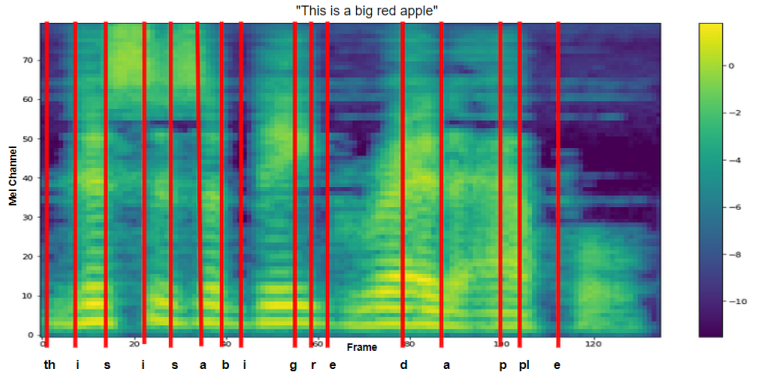
Table 7.2 Impact of reference speech duration on the quality of Voice cloning. The results are obtained from Jia et al.,2018 [3].

The network was trained on 4 Nvidia GeForce 2080 Ti GPUs for 1500 epochs with a batch size of 32. The training requires approximately 9 days to complete. The replication factor is set to 3 (number of decoder outputs per step). We observed that with smaller output per decoder step, the model either performed poorly or failed to converge. The L_2 loss between the predicted and ground truth mel-spectrograms is used as the loss function. The model is trained in Ground Truth Aligned (GTA) mode (teacher-forcing), where the pre-net is provided with the previous frame of the ground truth spectrogram rather than the predicted one. As the prosody and pitch of the generated spectrogram are aligned with the ground truth, a shared context between the prediction and the ground truth

can be established, allowing for faster convergence.



(a) Speaker 1 (Female)



(b) Speaker 2 (Male)

Figure 7.7 Mel-Spectrograms of audios synthesized on two different target speakers **(a)** Female **(b)** Male. The red lines are manually added to demonstrate the alignment between the spectrogram and the text transcript. Differences in the speaking rate and the fundamental frequency between the two speakers can be observed.

Despite being modified with speaker embeddings, our implementation of the synthesizer retains the fundamental features of Tacotron. The synthesizer reproduces prosodic qualities such as punctuation sensitivity, stress on specific syllables based on word context, and pronunciation of out-of-vocabulary terms. Figure 7.7 shows the mel-spectrograms generated for the speech synthesized on the embeddings of a male and a female speaker on the same phonetic content, demonstrating clear differences between the speaking rate and the fundamental frequency. This indicates that the synthesizer is capable of modeling speaker

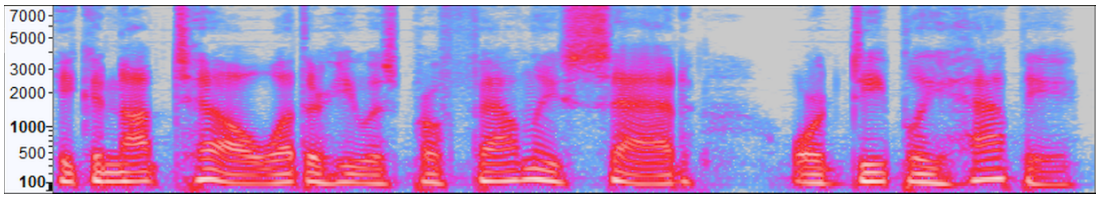
characteristics based on the speaker embeddings it is conditioned on.

7.3 Vocoder

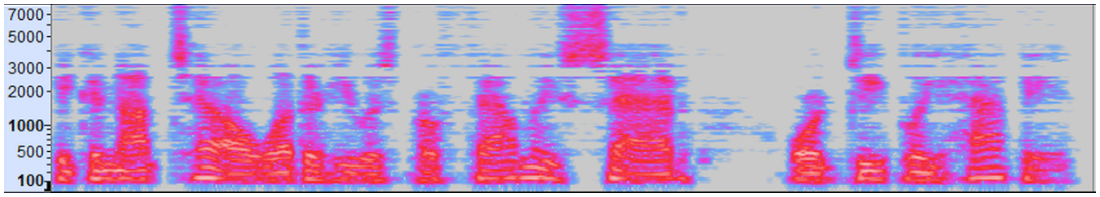
Our WaveGlow implementation consists of 12 coupling layers along with 12 invertible 1x1 convolutions. The coupling layer networks WN, as explained in Section 5.5.1, each includes 8 dilated convolution layers, 512 channels utilized as residual connections, and 256 channels for skip connections. After every four coupling layers, we output two of the channels. The network was trained on 4 Nvidia GeForce 2080 Ti GPUs for 1500 epochs with a batch size of 8 and a learning rate of 1×10^{-4}, using randomly picked clips of 8000 samples. The training requires approximately 12 days to complete.

The model was trained on two different datasets, one being a single-speaker LJSpeech [57] and the other being the LibriTTS clean dataset, to perform a comparative analysis between the two. This was also essential to identify the reason behind the poor quality of the synthesized audio on certain speakers in the test dataset. Our final implementation of the vocoder is trained on the LibriTTS Clean set so as to support multiple speakers.

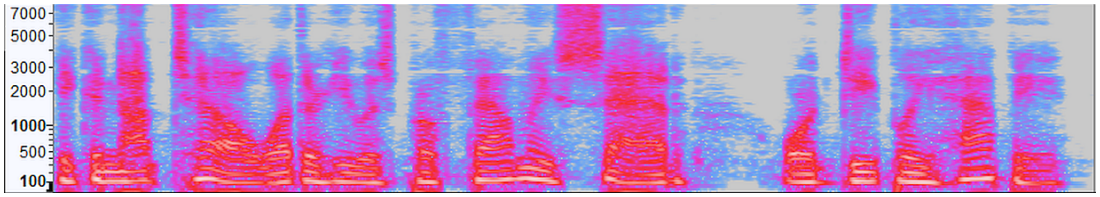
To compare the difference in generated audios, we observed the mel-spectrograms resulting from the single-speaker and multi-speaker vocoders. We take a sample audio from the test set and generate the corresponding mel-spectrogram. This is then passed through both the vocoders to reconstruct the original audio. The single-speaker Vocoder was less noisy with the lower frequency bands not being represented well. It also failed to generalize to a diverse set of speakers, especially for male voices. This behavior is expected as the LJSpeech consists of audios recorded from a single female speaker. In the case of the multi-speaker Vocoder, the issue of generalization is somewhat resolved, although some mid-range frequencies are not prominent and it introduces noise at the fricative sounds ($/s/, /z/, /v/$). We also analyzed the mel-spectrogram for the synthesized audio on the same transcript corresponding to the embedding from the same speaker. As demonstrated in Figure 7.8, the formants beyond the third are not distinct and are smoothed out. The fricative sounds also appear to be elongated when compared to the mel-spectrogram from the original audio.



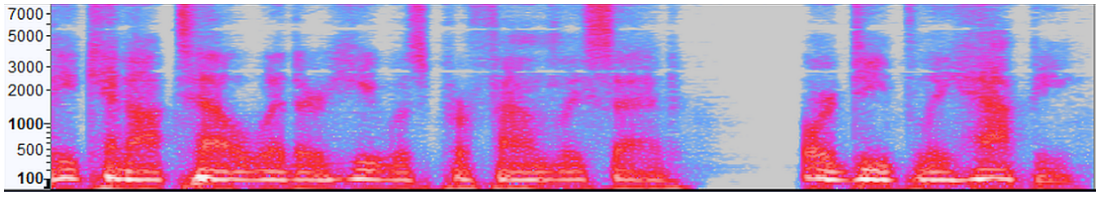
(a) Original audio



(b) Single-speaker vocoder



(c) Multi-speaker vocoder



(d) Synthesized audio

Figure 7.8 Mel spectrograms corresponding to the **(a)** Original audio **(b)** Reconstruction of the original audio by a single-speaker vocoder **(c)** Reconstruction of the original audio by a multi-speaker vocoder **(d)** Audio synthesized on the speaker embedding of the same speaker using the same transcript

7.4 Results and Analysis

To evaluate the performance of our model, we use the Mean Opinion Score (MOS) as indicated in the original paper. Although the SV2TTS study used crowd-sourced evaluation for different experiments, organizing such an evaluation method was beyond our means. Instead, we created a web platform using ReactJS and Firebase as a cloud-hosted database to store the ratings in order to conduct the evaluation. The user is instructed to assess the Naturalness and Similarity of a synthesized audio corresponding to a reference speech by a specific

speaker. The term "Naturalness" refers to how natural the synthesized audio sounds in comparison to human speech, whereas "Similarity" refers to how similar the voice of the cloned sample is to the speaker from the reference speech. The scale ranges from 1 (Least Similar/Natural) to 5 (Extremely Similar/Natural) in increments of 0.5. We use audio samples from 10 speakers in the evaluation, 5 of which are seen during training, whereas the remaining are unseen, mostly collected from YouTube videos of popular personalities. The result obtained from 20 participants has been provided in Table 7.3. The confidence interval is wider in our result owing to the lesser number of participants in the evaluation compared to that in the SV2TTS study.

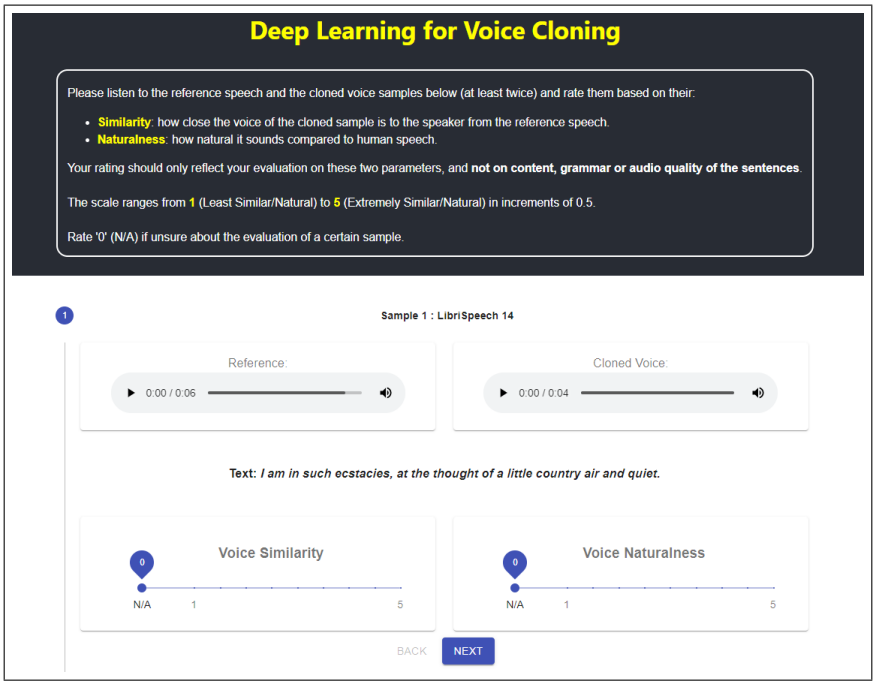


Figure 7.9 MOS Evaluation platform

We compare our results with the first two rows of Table 7.3 as it has a similar number of speakers with the same embedding dimensionality (64) as used in our implementation. This is justified as the model performance is impacted by the number of speakers used, as highlighted in Section 7.1.2. As can be seen from Table 7.3, our model achieves a better Similarity score than the original approach, although the Naturalness is significantly lower. There could be several reasons for this :

- Our implementation of the Speaker Encoder included 1.5K speakers from both the LibriTTS Clean and Other sets. Most audios from the LibriTTS-

Other set contains significant background noise. While the noisy data can help generalize and improve the cloning capability, it can adversely impact the naturalness of the synthesized audio. This pattern is evident from the scores of SV2TTS represented in the first two rows of Table 7.3. Switching from the Clean to Other set resulted in an increase of the mean Similarity by 0.04 whereas the mean Naturalness decreased by 0.13.

- The Synthesizer was trained on the LibriTTS Clean set, which is also not entirely devoid of noise [43]. There are significant differences in the tone and style of LibriSpeech utterances, even when the speaker is the same. There are some instances in which the speaker attempts to imitate another person's voice of a different gender. This is highlighted in Appendix B of the SV2TTS paper which demonstrates how some speakers have considerably lower naturalness scores as a result of the noise level in the recordings. Experiments on the VCTK dataset revealed that it is more consistent both in terms of similarity and naturalness.

- Our implementation of the WaveGlow vocoder was trained on the LibriTTS dataset using the ground truth mel-spectrograms of the audios. However, the authors mention that this approach of training the Vocoder on the ground truth mel-spectrograms worked well in case of the VCTK dataset, which has relatively cleaner audio. Since LibriSpeech is noisier, using the spectrograms generated by the synthesizer for training the vocoder would have been beneficial.

Method	SE Training Set	Speakers	Embedding Dim	Naturalness	Similarity
SV2TTS	LS-Clean	1.2K	64	3.73 ± 0.06	2.23 ± 0.08
SV2TTS	LS-Other	1.2K	64	3.60 ± 0.06	2.27 ± 0.09
SV2TTS	LS-Other + VC	2.4K	256	3.83 ± 0.06	2.43 ± 0.09
SV2TTS	LS-Other + VC + VC2	8.4K	256	3.82 ± 0.06	2.54 ± 0.09
SV2TTS	Internal	18K	256	4.12 ± 0.05	3.03 ± 0.09
Proposed Method	LibriTTS-Clean + Other	1.5K	64	2.60 ± 0.15	2.84 ± 0.15

Table 7.3 MOS Evaluation Result

Despite identifying the likely explanations for the result, we were unable to test the above-mentioned improvement possibilities due to the time limits imposed by the extensive training period required for both the vocoder and the synthesizer. However, future work in this direction might significantly improve the naturalness of the synthesized audios and the cloning capability of the model.

Chapter 8

Conclusion and Future Work

This thesis investigates several deep learning techniques for voice cloning. The primary focus is to achieve voice cloning in a Zero-shot setting with a short reference speech signal of the target speaker. We build upon the SV2TTS model [3] and apply certain modifications to the baseline architecture. This approach follows a transfer learning strategy from speaker verification to multi-speaker text-to-speech synthesis.

8.1 Discussion

The proposed approach exhibits significant voice cloning capabilities, as evident from the MOS results, with the mean similarity score exceeding that of the original method. As described in Section 7.1, this has been possible due to the speaker encoder network, which has demonstrated the ability to capture the characteristics of a diverse set of voices fairly well. The mean naturalness score, on the other hand, is lower than the original method, indicating that there is room for improvement. The low naturalness could be attributed to the presence of background noise in some of the synthesized voices.

From our experiments, we identified that some of the noise (mostly at fricatives) and artefacts originates from the multi-speaker vocoder. Small errors in the generated spectrogram also result in artificial, metallic noise in the cloned audio, as also encountered in VoiceLoop [58]. A proper spectrogram synthesis relies heavily on the selection of a clean dataset. In our implementation, the synthesizer learns to reproduce noise from the LibriTTS-clean dataset which contains low-moderate background noise, in spite of being labeled as "clean". Generalization across unseen voices is dependent on the amount of speakers (Table 7.1) used to train the speaker encoder and is not affected by the vocoder or the synthesizer.

8.2 Future Work

Given the shortcomings highlighted in the existing implementation of the model, a reasonable extension of this study would be to apply the necessary changes and re-evaluate the performance for potential improvement. A clean, multi-speaker dataset is of primary importance to train the synthesizer and vocoder networks. As for the speaker encoder, a large dataset with diverse speakers is required to capture all the intricacies of the human voice, so as to allow for better generalization for unseen speakers. Training a gender-dependent model can also improve the similarity for unseen speakers [59].

Another improvement could be in terms of inference speed. While we have replaced the WaveNet for WaveGlow as the vocoder to improve the audio generation speed, the autoregressive nature of the synthesizer prevents the model from attaining real-time inference speeds. Replacing the synthesizer with non-autoregressive models like FastSpeech [60] or FastSpeech2 [61] could help generate synthesized voices for real-time applications.

While our approach of using a speaker-embedding approach for voice cloning can produce promising results in terms of retaining speaker-specific characteristics in the cloned speech, it does not allow control over certain aspects of speech. Some of these aspects that are not fully captured by the speaker-specific embedding or contained in the text-transcript include speaking rate, intonation, emotion, and emphasis. Recent techniques, such as the Mellotron [62], have been able to achieve expressive voice synthesis by extending the concept of style representations by conditioning the TTS model with rhythm, pitch, and style tokens. Combining these approaches into a unified end-to-end system can achieve expressive voice cloning for real-time applications.

Although voice cloning can be a valuable tool in today's voice-driven digital ecosystem, it can also be exploited for identity theft and other malevolent objectives. Apart from the improvement of existing methods in voice cloning, any future work must also take into consideration ways to distinguish synthetic speech from real speech to allow fair use of this technology.

References

[1] Sercan Arik, Gregory Diamos, Andrew Gibiansky, John Miller, Kainan Peng, Wei Ping, Jonathan Raiman, and Yanqi Zhou. Deep voice 2: Multi-speaker neural text-to-speech, 2017.

[2] Sercan O Arik, Jitong Chen, Kainan Peng, Wei Ping, and Yanqi Zhou. Neural voice cloning with a few samples. *arXiv preprint arXiv:1802.06006*, 2018.

[3] Ye Jia, Yu Zhang, Ron J Weiss, Quan Wang, Jonathan Shen, Fei Ren, Zhifeng Chen, Patrick Nguyen, Ruoming Pang, Ignacio Lopez Moreno, et al. Transfer learning from speaker verification to multispeaker text-to-speech synthesis. *arXiv preprint arXiv:1806.04558*, 2018.

[4] Dennis H Klatt. Review of text-to-speech conversion for english. *The Journal of the Acoustical Society of America*, 82(3):737–793, 1987.

[5] Manfred R Schroeder. A brief history of synthetic speech. *Speech Communication*, 13(1-2):231–237, 1993.

[6] Lawrence R Rabiner and Ronald W Schafer. *Introduction to digital speech processing*, volume 1. Now Publishers Inc, 2007.

[7] LF Lamel, JL Gauvain, B Prouts, C Bouhier, and R Boesch. Generation and synthesis of broadcast messages. In *Proc. ESCA-NATO Workshop on Applications of Speech Technology*, pages 207–210. Citeseer, 1993.

[8] Alan W Black, Heiga Zen, and Keiichi Tokuda. Statistical parametric speech synthesis. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 4, pages IV–1229. IEEE, 2007.

[9] Keiichi Tokuda, Yoshihiko Nankaku, Tomoki Toda, Heiga Zen, Junichi Yamagishi, and Keiichiro Oura. Speech synthesis based on hidden markov models. *Proceedings of the IEEE*, 101(5):1234–1252, 2013.

[10] Shiyin Kang, Xiaojun Qian, and Helen Meng. Multi-distribution deep belief network for speech synthesis. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8012–8016. IEEE, 2013.

[11] Yuchen Fan, Yao Qian, Feng-Long Xie, and Frank K Soong. Tts synthesis with bidirectional lstm based recurrent neural networks. In *Fifteenth annual conference of the international speech communication association*, 2014.

[12] Jose Sotelo, Soroush Mehri, Kundan Kumar, Joao Felipe Santos, Kyle Kastner, Aaron Courville, and Yoshua Bengio. Char2wav: End-to-end speech synthesis. 2017.

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

[14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[15] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[16] Jonathan Shen, Ruoming Pang, Ron J Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, Rj Skerrv-Ryan, et al. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4779–4783. IEEE, 2018.

[17] Wei Ping, Kainan Peng, Andrew Gibiansky, Sercan O Arik, Ajay Kannan, Sharan Narang, Jonathan Raiman, and John Miller. Deep voice 3: Scaling text-to-speech with convolutional sequence learning. *arXiv preprint arXiv:1710.07654*, 2017.

[18] Eliya Nachmani, Adam Polyak, Yaniv Taigman, and Lior Wolf. Fitting new speakers based on a short untranscribed sample. In *International Conference on Machine Learning*, pages 3683–3691. PMLR, 2018.

[19] RJ Skerry-Ryan, Eric Battenberg, Ying Xiao, Yuxuan Wang, Daisy Stanton, Joel Shor, Ron Weiss, Rob Clark, and Rif A Saurous. Towards end-to-end prosody transfer for expressive speech synthesis with tacotron. In *international conference on machine learning*, pages 4693–4702. PMLR, 2018.

[20] Yuxuan Wang, Daisy Stanton, Yu Zhang, RJ-Skerry Ryan, Eric Battenberg, Joel Shor, Ying Xiao, Ye Jia, Fei Ren, and Rif A Saurous. Style tokens:

Unsupervised style modeling, control and transfer in end-to-end speech synthesis. In *International Conference on Machine Learning*, pages 5180–5189. PMLR, 2018.

[21] Li Wan, Quan Wang, Alan Papir, and Ignacio Lopez Moreno. Generalized end-to-end loss for speaker verification. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4879–4883. IEEE, 2018.

[22] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[23] Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. Zero-data learning of new tasks. In *AAAI*, volume 1, page 3, 2008.

[24] Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1700–1709, 2013.

[25] Philipp Koehn, Franz J Och, and Daniel Marcu. Statistical phrase-based translation. Technical report, UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL REY INFORMATION SCIENCES INST, 2003.

[26] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[27] Karl Moritz Hermann and Phil Blunsom. Multilingual models for compositional distributed semantics. *arXiv preprint arXiv:1404.4641*, 2014.

[28] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. *arXiv preprint arXiv:1606.05328*, 2016.

[29] Aaron Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George Driessche, Edward Lockhart, Luis Cobo, Florian Stimberg, et al. Parallel wavenet: Fast high-fidelity speech synthesis. In *International conference on machine learning*, pages 3918–3926. PMLR, 2018.

[30] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[31] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *arXiv preprint arXiv:1506.03099*, 2015.

[32] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[33] David Krueger, Tegan Maharaj, János Kramár, Mohammad Pezeshki, Nicolas Ballas, Nan Rosemary Ke, Anirudh Goyal, Yoshua Bengio, Aaron Courville, and Chris Pal. Zoneout: Regularizing rnns by randomly preserving hidden activations. *arXiv preprint arXiv:1606.01305*, 2016.

[34] Georg Heigold, Ignacio Moreno, Samy Bengio, and Noam Shazeer. End-to-end text-dependent speaker verification. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5115–5119. IEEE, 2016.

[35] Corentin Jemine et al. Master thesis: Real-time voice cloning. 2019.

[36] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[37] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[38] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[39] Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. *arXiv preprint arXiv:1506.07503*, 2015.

[40] Eric Moulines and Francis Charpentier. Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones. *Speech communication*, 9(5-6):453–467, 1990.

[41] Hideki Kawahara, Ikuyo Masuda-Katsuse, and Alain De Cheveigne. Restructuring speech representations using a pitch-adaptive time–frequency smoothing and an instantaneous-frequency-based f0 extraction: Possible role of a repetitive structure in sounds. *Speech communication*, 27(3-4):187–207, 1999.

[42] Masanori Morise, Fumiya Yokomori, and Kenji Ozawa. World: a vocoder-based high-quality speech synthesis system for real-time applications. *IEICE TRANSACTIONS on Information and Systems*, 99(7):1877–1884, 2016.

[43] Daniel Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on acoustics, speech, and signal processing*, 32(2):236–243, 1984.

[44] Tom Le Paine, Pooya Khorrami, Shiyu Chang, Yang Zhang, Prajit Ramachandran, Mark A Hasegawa-Johnson, and Thomas S Huang. Fast wavenet generation algorithm. *arXiv preprint arXiv:1611.09482*, 2016.

[45] Nal Kalchbrenner, Erich Elsen, Karen Simonyan, Seb Noury, Norman Casagrande, Edward Lockhart, Florian Stimberg, Aaron Oord, Sander Dieleman, and Koray Kavukcuoglu. Efficient neural audio synthesis. In *International Conference on Machine Learning*, pages 2410–2419. PMLR, 2018.

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

[47] Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*, 2018.

[48] Wei Ping, Kainan Peng, and Jitong Chen. Clarinet: Parallel wave generation in end-to-end text-to-speech. *arXiv preprint arXiv:1807.07281*, 2018.

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

[50] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[51] Heiga Zen, Viet Dang, Rob Clark, Yu Zhang, Ron J Weiss, Ye Jia, Zhifeng Chen, and Yonghui Wu. Libritts: A corpus derived from librispeech for text-to-speech. *arXiv preprint arXiv:1904.02882*, 2019.

[52] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

[53] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. Voxceleb: a large-scale speaker identification dataset. *arXiv preprint arXiv:1706.08612*, 2017.

[54] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman. Voxceleb2: Deep speaker recognition. *arXiv preprint arXiv:1806.05622*, 2018.

[55] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[56] Antoine Perquin, Erica Cooper, and Junichi Yamagishi. Grapheme or phoneme? an analysis of tacotron's embedded representations. *arXiv e-prints*, pages arXiv–2010, 2020.

[57] Keith Ito. The lj speech dataset https://keithito. com, 2017.

[58] Yaniv Taigman, Lior Wolf, Adam Polyak, and Eliya Nachmani. Voiceloop: Voice fitting and synthesis via a phonological loop. *arXiv preprint arXiv:1707.06588*, 2017.

[59] Erica Cooper, Cheng-I Lai, Yusuke Yasuda, Fuming Fang, Xin Wang, Nanxin Chen, and Junichi 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)*, pages 6184–6188. IEEE, 2020.

[60] Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech: Fast, robust and controllable text to speech. *arXiv preprint arXiv:1905.09263*, 2019.

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

[62] Rafael Valle, Jason Li, Ryan Prenger, and Bryan Catanzaro. Mellotron: Multispeaker expressive voice synthesis by conditioning on rhythm, pitch and global style tokens. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6189–6193. IEEE, 2020.

Declaration

I hereby declare that the work presented in this thesis is entirely my own. I did not use any other sources and references that the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted hard copies.

Soumyadeep Bhattacharjee

01.08.2021