Deep learning based voice cloning framework for a unified system of text-to-speech and voice conversion

by

Hieu-Thi Luong

Dissertation

submitted to the Department of Informatics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

SOKENDAI

The Graduate University for Advanced Studies, SOKENDAI
September 2020

Declaration

I declare that this thesis was composed by myself and that the work contained therein is my own, except where explicitly stated otherwise in the text.

(*Hieu-Thi Luong*)

Summary

Speech synthesis is the technology of generating speech from an input. While the term is commonly used to refer to text-to-speech (TTS), there are many types of speech synthesis systems which handle different input interfaces such as voice conversion (VC), which converts speech of a source speaker to the voice of a target, or video-to-speech, which generates speech from an image sequence (video) of facial movements.

This thesis focuses on the voice cloning task which is the developing of a speech synthesis system with an emphasis on speaker identity and data efficiency. A voice cloning system is expected to handle circumstance of having less than ideal data for a particular target speaker. More specifically, when we not have control over the target speaker, recording environment, or the quality and quantity of speech data. Such systems will be useful for many practical applications which involve generating speech with desired voices. However, it is also vulnerable to misuse which can cause significant damage to society by people with malicious intentions. By first breaking down the structures of conventional TTS and VC systems into common functional modules, we propose a versatile deep learning based voice cloning framework which can be used to create a unified speech generation system of TTS and VC with a target voice. Given such unified system, which is expected to have consistent performance between its TTS and VC modes, we can use it to handle many application scenarios that are difficult to tackle by just one or the other, as TTS and VC have their own strengths and weaknesses.

As this thesis is dealing with two major research subjects, which are TTS and VC, to provide a comprehensive narrative its content can be considered as comprising of two segments which tackle two different issues: (1) developing a versatile speaker adaptation method for neural TTS systems. Unlike VC in which existing voice cloning methods are capable of producing high-quality generated speech, existing TTS adaptation methods are lacking behind in performance and scalability. The proposed method is expected to be capable of cloning voices using either transcribed or untranscribed speech with varying amounts of adaptation data while producing generated speech with high quality and speaker similarity; (2) establishing a unified speech generation system of TTS and VC with highly consistent performance between two. To achieve this consistency, it is desirable to reduce the differences between the methodology and use the same framework for both systems. In addition to convenience, such system also has the ability to solve many unique speech generation tasks, as TTS and VC are operated under different application scenarios and complement each other.

On the first issue, by investigating the mechanism of a multi-speaker neural acoustic model, we proposed a novel multimodal neural TTS system with the ability to perform crossmodal adaptation. This ability is fundamental for cloning voices with untranscribed speech on the basis of the backpropagation algorithm. Comparing with existing unsupervised speaker adaptation methods which only involve a forward pass, a backpropagation-based unsupervised adaptation method has significant implication on performance as it allows us to expand the speaker component to other parts of the neural networks beside the speaker bias. This hypothesis is tested by using speaker scaling together with speaker bias, or the entire module as adaptable components. The proposed system unites the procedure of supervised and unsupervised speaker adaptation.

On the second issue, we test the feasibility of using the multimodal neural TTS system proposed previously to bootstrap a VC system for a particular target speaker. More specifically, the proposed VC system is tested on standard intra-language scenarios and cross-lingual scenarios with the experiment evaluations showing promising performance in both. Finally given the proof-of-concept provided by earlier experiments, the proposed methodology is incorporated with relevant techniques and components of modern neural speech generation systems to push performance of the unified TTS/VC system further. The experiments suggest that the proposed unified system has comparable performance with existing state-of-the-art TTS and VC systems, at the time this thesis was written, but higher speaker similarity and better data efficiency.

At the end of this thesis, we have successfully created a versatile voice cloning system which can be used for many interesting speech generation scenarios. Moreover, the proposed multimodal system can be extended to other speech generation interfaces or enhanced to provide controls over para-linguistic features (e.g., emotions). These are all interesting directions for future works.

Acknowledgements

Before I start my PhD study, I was not confident that doing research or pursuing a PhD degree was a road that I wanted to take. Even now when it is close to the end of my study I am still not sure if researching is what I want to do, but this journey has helped me recognize my ability and open many doors into the future.

This thesis would not be possible without my advisor, Professor Yamagishi Junichi, who taught me research methods and helped me write my very first published paper. More importantly, I have learned from him to try for better instead of settling with good enough in everything you do, which is a valuable lesson that will be with me whether in doing research or everyday life.

I would like to express my gratitude to Dr. Takaki Shinji who had helped me to start my research on speech synthesis during my time as an intern at National Institute of Informatics (NII), which is the inspiration for the content of this thesis. I also want to thanks my colleages at Yamagishi Lab and all my friends at NII for their help over the past several years, during which I spent more times than at my own house.

I want to thank the examiners of my intermediate evaluations, Professor Shinoda Koichi, Professor Toda Tomoki, Professor Satoh Shin'ichi, and Professor Inamura Tetsunari, for constructive comments and suggestions over the years which motivate me to push my research further.

I would like to thank Ms. Kuwahara Makiko, secretary of Yamagishi Lab, NII Education support team, and SOKENDAI for not only supporting my research but also helping me deal with big and small problems of living in Japan, especially in difficult time likes the ongoing pandemic.

Finally, I want to thank my mother, Le Thi Em, and my sisters who are always there to support me in my works and my life.

List of publications

The content of this thesis are based on following papers that were published or under-reviewed during the course of my PhD study.

Journal

1. Hieu-Thi Luong and Junichi Yamagishi, “NAUTILUS: a Versatile Voice Cloning System”,
IEEE/ACM Transactions on Audio, Speech, and Language Processing, May 2020 (Submitted)

2. Yi Zhao, Shinji Takaki, Hieu-Thi Luong, Junichi Yamagishi, Daisuke Saito, and Nobuaki Minematsu,
“Wasserstein GAN and waveform loss-based acoustic model training for multi-speaker text-to-speech synthesis systems using a WaveNet vocoder”,
IEEE Access, vol. 6, pp. 60 478-60 488, September 2018

Conference

1. Hieu-Thi Luong and Junichi Yamagishi, “Bootstrapping non-parallel voice conversion from speaker-adaptive text-to-speech”,
Proc. ASRU, pp. 200-207, December 2019

2. Hieu-Thi Luong, Xin Wang, Junichi Yamagishi, and Nobuyuki Nishizawa,
“Training Multi-Speaker Neural Text-to-Speech Systems using Speaker-Imbalanced Speech Corpora”,
Proc. INTERSPEECH, pp. 1303-1307, September 2019

3. Hieu-Thi Luong and Junichi Yamagishi, “Scaling and bias codes for modeling speaker-adaptive DNN-based speech synthesis systems”,
Proc. SLT, pp. 610-617, December 2018

4. Hieu-Thi Luong and Junichi Yamagishi, “Multimodal Speech Synthesis Architecture for Unsupervised Speaker Adaptation”,
   Proc. INTERSPEECH, pp. 2494-2498, September 2018

5. Hieu-Thi Luong, Xin Wang, Junichi Yamagishi, and Nobuyuki Nishizawa, “Investigating accuracy of pitch-accent annotations in neural network-based speech synthesis and denoising effects”,
   Proc. INTERSPEECH, pp. 37-41, September 2018

Technical report

The following paper is self-published but consists of important contents that is included in this thesis.

1. Hieu-Thi Luong and Junichi Yamagishi, “A Unified Speaker Adaptation Method for Speech Synthesis using Transcribed and Untranscribed Speech with Backpropagation”,
   arXiv preprint https://arxiv.org/abs/1906.07414, June 2019

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Voice cloning

Speech synthesis is the technology of generating speech from an input. In its narrow sense, speech synthesis is used to refer to text-to-speech (TTS) [1], which play an essential role in spoken dialog systems as a way for machines to communicate with humans. In its broader definition, speech synthesis can refer to all kinds of speech generation interfaces like voice conversion (VC) [2], video-to-speech [3, 4], et cetera [5]. Recent state-of-the-art (SOTA) speech synthesis systems can generate speech with natural sounding quality, some of which are indistinguishable from recorded speech [6]. Deep neural networks are used for various components of these speech synthesis systems such as acoustic models and neural vocoders. Many use a sequence-to-sequence (seq2seq) model to unfold a compact phoneme sequence to acoustic features in TTS [6, 7] or to handle the mismatch alignment of acoustic sequences in VC [8, 9, 10]. A neural vocoder which generates waveforms sample-by-sample [11, 12, 13] is also a staple of many high quality speech generation recipes [6, 14]. Generally speaking, a deep learning approach achieves high performance when training on a lot of data. For speech generation models, it means we need many hours of speech from a target speaker to train the model. This limits the ability to scale the technology to many different voices.

The term *voice cloning* is used to refer to a specific speaker adaptation scenario of TTS with untranscribed speech in several works [15, 16]. However in pop culture, it is loosely used to describe the technology which resembles VC. To decouple this confusion we can use *voice cloning* as an umbrella term to indicate any type of system which generates speech imitating the voice of a particular speaker. The main difference between *voice cloning* and *speech synthesis* is that the former puts an emphasis on the identity of the target speaker [17] while the

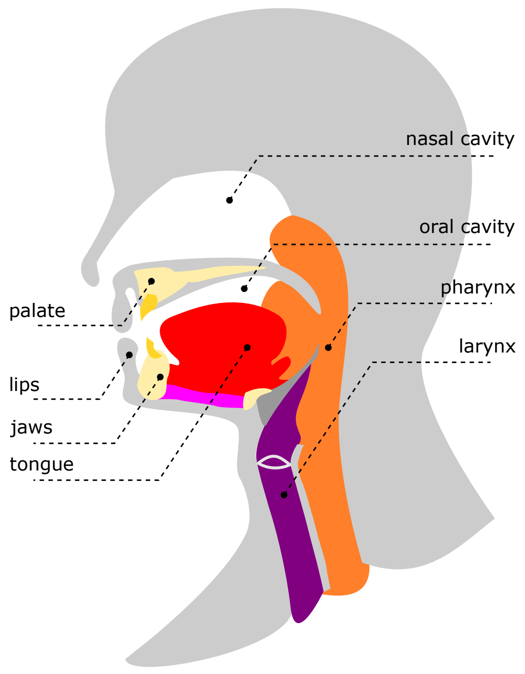


Figure 1.1: Human speech production system.

latter sometimes disregards this aspect for the naturalness [18]. Given this definition, a voice cloning system can be TTS, VC, or any type of speech generation system [4, 5].

The performance of a voice cloning system is judged on many aspects. As a speech generation system, the naturalness and the similarity to target speakers are important [6]. As a computer system, small memory footprint [19] and fast computing time [15, 20] are desirable for practical reasons. However, the defining property of a voice cloning system compared with a generic speech synthesis is its data efficiency as this would dictate the scalability [21]. While data efficiency can be casually interpreted as using as little data as possible [19], a better voice cloning system should not only work with extremely limited data situation but should also be able to take advantages of abundant speech data [21] when they become available and whether they are transcribed or not [22].

While TTS and VC are two systems functioning under different scenarios, they share a common goal of generating speech with a target voice [23]. If we define the TTS as a synthesis system generating speech using linguistic instructions obtained from written text [24], then the VC can be defined as a synthesis system generating speech using linguistic instructions extracted from a reference utterance. Given the similarity in objective but difference in operation contexts, TTS and VC complement each other and have their unique role when dealing with a particular speech generation task. If we can unite their methodology and create a versatile system with high consistence, it would greatly reduce the methodological complexity as well as potentially be used for many unique scenarios. This is also a main objective of this thesis.

1.2 Thesis outline

As illustrated in Figure 1.2, the chapters in this thesis tackle different aspects of two major issues. Chapter 2 and 3 give general background about deep learning, speech generation and establishing the basis of the multi-speaker neural TTS system as well as the supervised speaker adaptation based on it. Chapter 4, 5, and 6 tackle different aspects of the development of a novel versatile speaker adaptation method. Chapter 7, and 8 improve upon the methodology established in Chapter 6 to create a unified self-consistent voice cloning system for both TTS and VC. The content of each chapter in more details follows:

Chapter 2 provides the background of TTS and VC systems in the context of voice cloning. The general information about deep learning that is most relevant to the thesis is also explained.

Chapter 3 establishes the multi-speaker acoustic model which is the basis of novel methodology proposed in later chapters. It provides the fundamental information on speaker-adaptive acoustic models and supervised adaptation.

Chapter 4 proposes a novel unsupervised speaker adaptation model by performing crossmodal adaptation with multimodal neural TTS. The methods used to train such multimodal neural networks are also explained.

Chapter 5 introduces different types of speaker components, like speaker-dependent scaling and bias codes, and evaluates the their effect on performance of multi-speaker and speaker adaptation tasks.

Chapter 6 uses the observations presented in Chapter 4 and 5 to propose further enhancements inspired by variational autoencoder to increase the robustness and performance of the multimodal speaker-adaptive neural TTS.

Chapter 7 provides proof-of-concept that the same methodology presented in Chapter 6 is applicable for creating non-parallel VC systems in both standard intra-language as well as cross-lingual scenarios.

Chapter 8 incorporates the proposed method with functional components of SOTA speech generation systems to push the performance of the unified TTS/VC system on native and non-native speaker voice cloning scenarios.

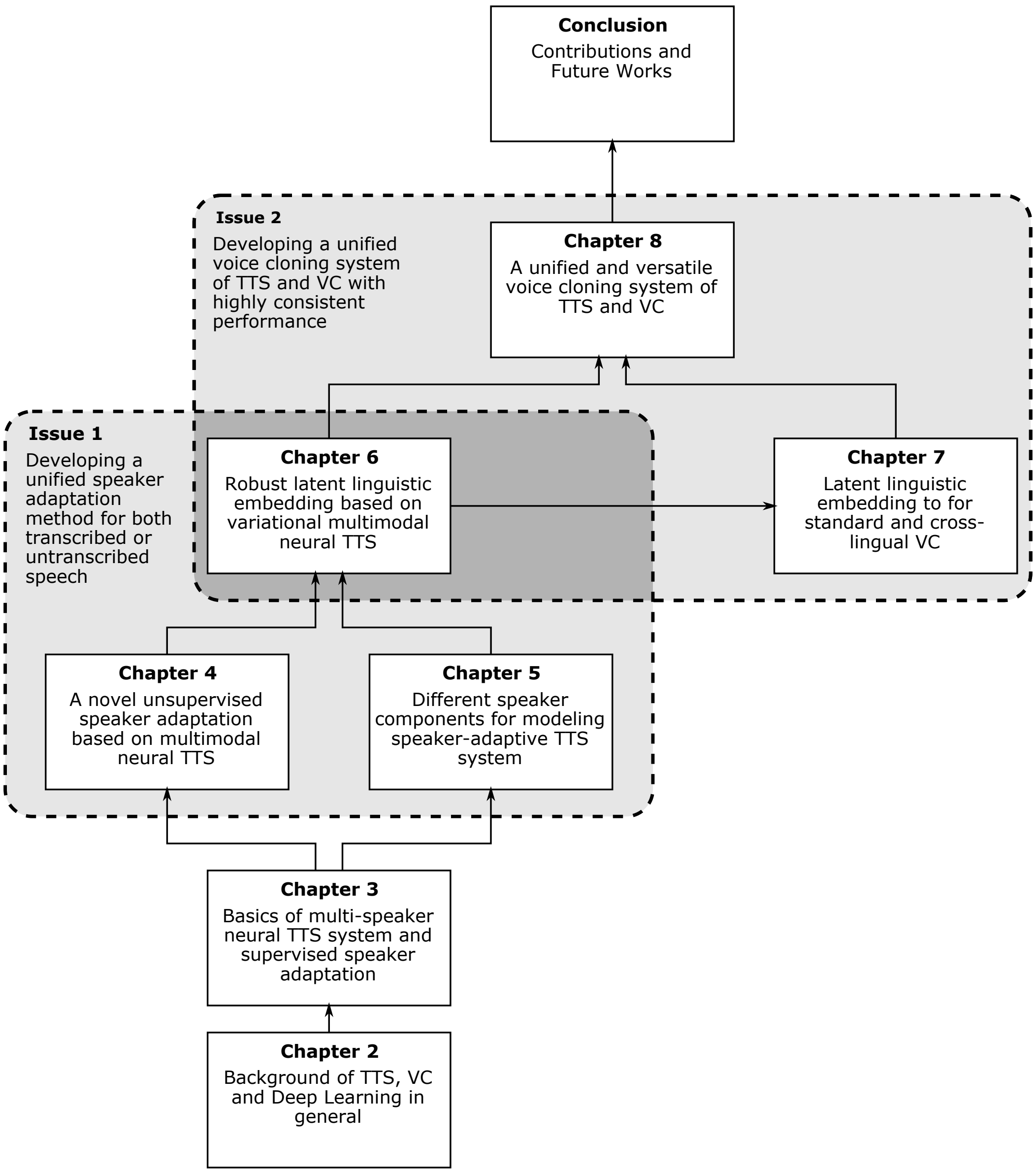

Conclusion
Contributions and Future Works
Issue 2
Developing a unified voice cloning system of TTS and VC with highly consistent performance
Chapter 8
A unified and versatile voice cloning system of TTS and VC
Issue 1
Developing a unified speaker adaptation method for both transcribed or untranscribed speech
Chapter 6
Robust latent linguistic embedding based on variational multimodal neural TTS
Chapter 7
Latent linguistic embedding to for standard and cross-lingual VC
Chapter 4
A novel unsupervised speaker adaptation based on multimodal neural TTS
Chapter 5
Different speaker components for modeling speaker-adaptive TTS system
Chapter 3
Basics of multi-speaker neural TTS system and supervised speaker adaptation
Chapter 2
Background of TTS, VC and Deep Learning in general

Figure 1.2: Outline of the thesis.

Chapter 2

Background

This chapter provides general background of TTS and VC in the context of cloning voices and introduces several related prior works. The information is framed in a way to highlight the similarity and complementary relation between TTS and VC. The fundamental deep learning methods that are most relevant to the work in this thesis are also explained in this chapter.

Section 2.1 and 2.2 review the prior work on TTS and VC respectively. Section 2.3 introduces the general deep learning techniques, while Section 2.4 puts them in the context of speech synthesis systems. Section 2.5 presents the objective and subjective metrics that are used throughout this thesis to evaluate the performance of speech generation systems.

2.1 Text-to-speech as voice cloning system

2.1.1 Text-to-speech system

A TTS system takes a text input and generates a speech utterance that conveys the information contained in the given words. Conceptually, a TTS system can be described as a transformation function (or a chain of functions as seen on Figure 2.1) which is defined by its parameters $\boldsymbol{\Theta}^{tts}$. The function takes in a text (or linguistic) input $\boldsymbol{x}$ and produces the speech (or acoustic) output $\tilde{\boldsymbol{y}}$ which is an approximation of the natural speech $\boldsymbol{y}$ of the target speaker:

$$\tilde{\boldsymbol{y}} = TTS(\boldsymbol{x}; \boldsymbol{\Theta}^{tts}) \tag{2.1}$$

A conventional pipeline of a TTS system is illustrated in Figure 2.1. The main challenge of TTS model is modeling the transformation of a compact text representation to a very dense speech waveform. Each module in the pipeline

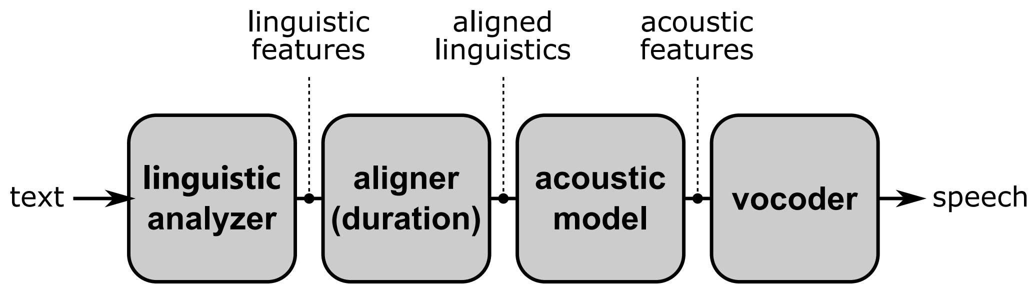


Figure 2.1: A conceptual representation of the conventional TTS pipeline.

transforms its input into an intermediate representation which helps bridging the gap between text and speech. More details about modules of the TTS system are explained in Section 2.4 together with that of the VC system. For a generic TTS system, the naturalness and the intelligibility of generated speech are the most important aspects as they directly affects the way the information is absorbed by listeners. While the speaker identity of generated utterances is also important, it is sometimes disregarded for the naturalness. For example, when the target language has limited resources [18] and the main goal is creating a serviceable TTS system.

A TTS system is typically trained on dozens of hours of transcribed speech [6, 25] of a particular target speaker. Due to the high requirement in quantity and quality, a professional voice actor is commonly commissioned to record such data in a controlled environment. This makes the conventional approach ill-fitting for voice cloning task in which we do not have control over the target speaker, recording environment, or the amount of available data.

2.1.2 Speaker adaptation for TTS systems

Cloning voices with a TTS system essentially means we want to create new voices using a small amount of data from the target speakers. With limited transcribed speech samples, we can adapt a pretrained model in a supervised manner. The initial model can be trained on data of a single speaker [26] or data pooled from multiple speakers [27, 28]. A simple fine-tuning produces a high quality model when the amount of data of the target speaker is small but sufficient (e.g. one hour) [21]. When data is extremely limited (e.g. one minute), we can restrict the tuning to a certain speaker component to prevent overfitting [28, 29, 21] or deploy additional regularization methods [30] to prioritize the reliable quality over speaker similarity. In summary, speaker adaptation methods transfer knowledge learned from abundant data of one or multiple speakers to reduce the data demand from a particular target speaker.

The costly part of a voice cloning process with the TTS system is data collecting stage, especially the transcript of speech. Theoretically speaking, the speaker characteristic should be self-contained in the utterances, so cloning voices with untranscribed speech should be a possibility. One practical approach is obtaining automatically annotated transcript using a SOTA automatic speech recognition (ASR) system [31]. However ASR-predicted transcripts contain wrong annotations which affects the performance of the adaptation process. Moreover this approach assumes a well-trained ASR system is obtainable for the target language which makes it impractical for low-resource languages [18] or performing cross-language speaker adaptation [32, 16]. Given the disentanglement ability of deep learning, another approach is training a speaker-adaptive acoustic model conditioned on a speaker representation extracted from speech [15, 19, 33]. The speaker representation can be *i-vector* [34], *d-vector* [35, 19], or *x-vector* [36] which are all byproducts of speaker recognition systems. This approach has computing advantage as it does not involve an optimization loop [15]. But its drawback is the low data scalability, in other words the similarity to the target speaker seems to stop improving when using more than a few seconds of speech [21].

2.1.3 Supervised adaptation with HMM-based TTS

Speaker adaptation for TTS systems is a long standing problem which has been developed along with advancements of the conventional data-abundant TTS problem. For traditional statistical parametric speech synthesis (SPSS) systems based on Hidden Markov Model (HMM), the speaker adaptation methods generally involve using a function to transform a single-speaker or an average-voice model to the adapted model which is better at explaining target speaker's limited transcribed speech data. Even though this thesis focuses on deep learning based systems, looking into the traditional framework provides a systematic overview about the motivations and the developments of speaker adaptation methods that are still relevant for contemporary systems.

An HMM is defined by its parameters which consist of initial state probabilities, the state transition probabilities and the output probability distributions as illustrated in Figure 2.2. Given an HMM-based acoustic model [37] with its parameter $\boldsymbol{\lambda}$ and a linguistic input sequence $\boldsymbol{x}_{1:T}$, the problem of generating speech from the acoustic model [38] is essentially obtain an acoustic feature sequence $\tilde{\boldsymbol{y}}_{1:T}$ which maximizes the output probability,

$$\tilde{\boldsymbol{y}}_{1:T} = \operatorname*{argmax}_{\boldsymbol{y}_{1:T}} p(\boldsymbol{y}_{1:T} | \boldsymbol{x}_{1:T}; \boldsymbol{\lambda}). \tag{2.2}$$

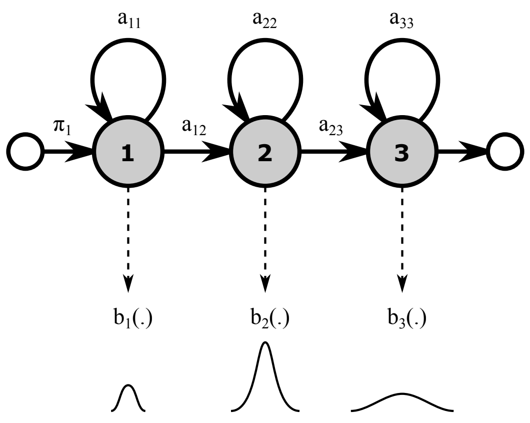
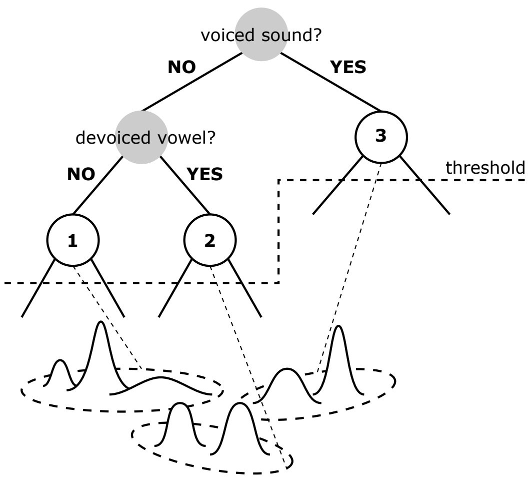


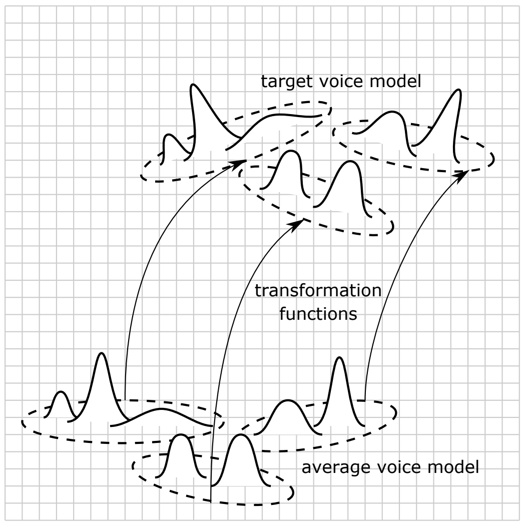
Figure 2.2: A three-state left-to-right HMM.

For speech synthesis, a continuous distribution is generally used for the output probabilities of HMM states. Specifically, a Gaussian distribution is normally used as the probability density function for both acoustic state output and duration state output. To generate sophisticated speech units which vary depending on text content, the acoustic model does not use phoneme but uses triphone/quinphone as well as augmenting linguistic contexts and prosody information as the input. However, it is impossible to have data to cover all possible linguistic contexts as there are too many variations some of which rarely occur in real-life situations. A decision tree which asks linguistic questions is used to cluster these variations and reduce the amount of parameters [39, 40].

Due to the fragmented nature of decision tree and the large amount of parameters, training an HMM-based acoustic model is not an option for speakers whose data is limited. In these cases, we can apply adaptation techniques to create an adapted model for target speakers by using a pretrained model of a different speaker with abundant data or an average-voice model trained on data of multiple speakers. In the other words, the fundamental of adaptation method is using a small amount of adaptation data of a target speaker to estimate a transformation function, which is applied to an initial model, instead of training the model's parameters themselves. A popular technique for speaker adaptation with HMM-based acoustic model is estimating a linear regression function to transform the initial model. The expressive power of the transformation function is one of the factor that has big impact on the performance of the adaptation method. Many works focused on tackling this aspect of speaker adaptation methods, for example:



(a) Decision tree context clustering



(b) Speaker adaptation

Figure 2.3: Decision tree clustering and speaker adaptation for TTS.

Maximum likelihood linear regression (MLLR): [41] given the mean $\boldsymbol{\mu}_i \in \mathbb{R}^d$ and covariance $\boldsymbol{\Sigma}_i \in \mathbb{R}^{d\times d}$ of the i-th multivariate output distribution of the initial model, MLLR adaptation technique transforms this distribution to one of the target speaker using an affine function:

$$\hat{\boldsymbol{\mu}}_i = \boldsymbol{A}^{(k)}\boldsymbol{\mu}_i + \boldsymbol{b}^{(k)}, \tag{2.3}$$

here $\boldsymbol{A}^{(k)} \in \mathbb{R}^{d\times d}$ and $\boldsymbol{b}^{(k)} \in \mathbb{R}^d$ belong to speaker k-th. Generally, these speaker-dependent parameters are estimated in maximum likelihood (ML) fashion with transcribed speech data of the target speaker.

Constrained MLLR (CMLLR): [42, 43] the above MLLR technique only alters the mean of the distribution which restrict its performance. Therefore, many works proposed the CMLLR technique which changes both the mean and the covariance of the distribution:

$$\hat{\boldsymbol{\mu}}_i = \boldsymbol{A}^{(k)}\boldsymbol{\mu}_i + \boldsymbol{b}^{(k)} \tag{2.4}$$

$$\hat{\boldsymbol{\Sigma}}_i = \boldsymbol{A}^{(k)}\boldsymbol{\Sigma}_i\boldsymbol{A}^{(k),\top}. \tag{2.5}$$

This technique is equivalent to applying the transformation on feature vectors themselves, thus it is also referred as feature-space MLLR (fMLLR) [44]. Depend on data situation of target speaker, we can estimate one transformation function for each distribution or a single function for the entire model to reduce the amount of adaptation parameters. For a more sophisticated technique, we

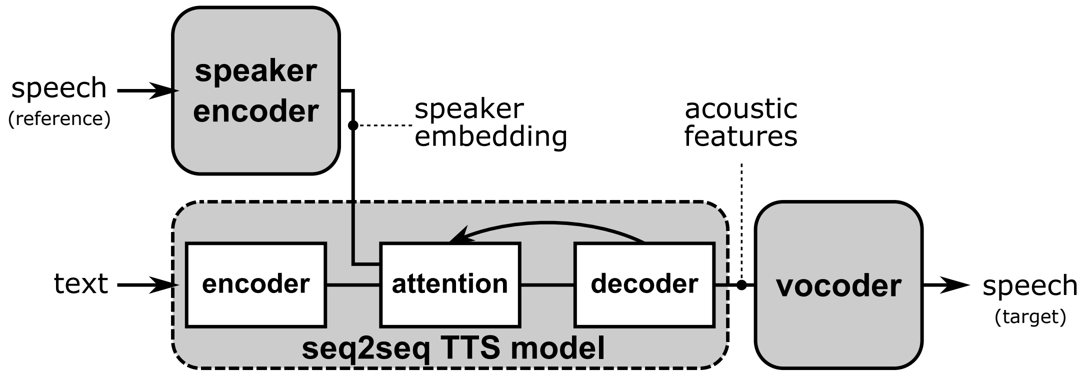


Figure 2.4: Speaker-adaptive TTS with speech-encoded speaker embedding.

can cluster distributions using tree structure [45, 30] to drastically reduce the amount of adaptation parameters as illustrated in Figure 2.3. The main incentive for proposing novel HMM-based speaker adaptation methods at that time was improving the balance between performance and overfitting resistance. For example, maximum a posterior (MAP) [46, 47] can be used instead of ML to reduce overfitting. The prior in MAP acts as an auxiliary regularization to prevent the adapted model from straying too far from the initial.

2.1.4 Unsupervised adaptation with d-vector

Unsupervised speaker adaptation in the context of TTS systems refers to methods that use untranscribed speech of target speakers to clone their voices. For deep learning based TTS systems, one prominent approach for unsupervised speaker adaptation with a small amount of speech data is using speech-encoded speaker embedding. This approach is dependent on the ability to decouple speaker characteristics from the linguistic model of the neural acoustic model. It works on the assumption that the speaker characteristic can be represented by a fixed-length embedding vector which is extracted from speech signal.

For example, Jia et. al. [19] proposed a speaker-adaptive TTS system which is capable of cloning voices with a few speech samples, which consists of three modules trained independently: a seq2seq TTS model, a neural vocoder, and a speaker encoder network as illustrated in Figure 2.4. The performance of this approach is dependent on two main assumptions. First, the speaker encoder is trained on a large-scale dataset with thousands of speakers. Second, the acoustic model has sufficient expressive power to decouple speaker characteristics and contain them in the speaker embedding. The results reported in [19] suggested that we can only enforce both of these two elements to some extent which makes the similarity of unseen target speakers to be worse than that of speakers in the initial training set.

The fact that the speaker encoder is trained independently from the TTS model makes its neural structure irrelevant. In the other words, the speaker encoder can be any type of system that takes the speech waveform and produces a respectable speech-encoded speaker embedding. The important part is that the embedding produced by the speaker encoder is "good" for the speaker-adaptive model, however there is no direct way to judge the quality of the embedding without trial and error on the neural TTS model itself. Many other works have evaluated similar systems with other types of speech-encoded speaker embedding [34, 35], with some types of embedding seeming to work better than others [35, 36] although the difference in performance is moderate in general.

2.2 Voice conversion as voice cloning systems

2.2.1 Voice conversion systems

A VC system modifies a utterances spoken by a source speaker to make it sound like it is spoken by a different target speaker while preserving the linguistic information [2, 48]. Similar to TTS, a VC system can also be described as a transformation function which is defined by its parameters $\boldsymbol{\Theta}^{vc}$. The function takes in speech of a source speaker $\boldsymbol{y}^{src}$ and produces the speech with a target voice $\tilde{\boldsymbol{y}}^{tar}$ while maintaining the linguistic content:

$$\tilde{\boldsymbol{y}}^{tar} = VC(\boldsymbol{y}^{src}; \boldsymbol{\Theta}^{vc}) \ . \tag{2.6}$$

A generic VC system by itself is complied with all requisite conditions of a voice cloning system defined in Section 1.1, in the other words speaker identity of generated utterances is the most important objective of VC. So a good voice cloning system for VC is the one that can generate high quality speech and use available data of the target efficiently.

The conventional VC approach is text-dependent, i.e. it expects the training data to be parallel utterances of source and target speakers [49, 50]. Given these utterances, VC can be modeled as a transformation from a deterministic speech input to a deterministic speech output. A conventional pipeline of a parallel VC system is illustrated in Figure 2.5 with several modules conceptually identical to that of the TTS system (see Figure 2.1). More details about each module is explained in Section 2.4. As both the input and output of VC system are speech,

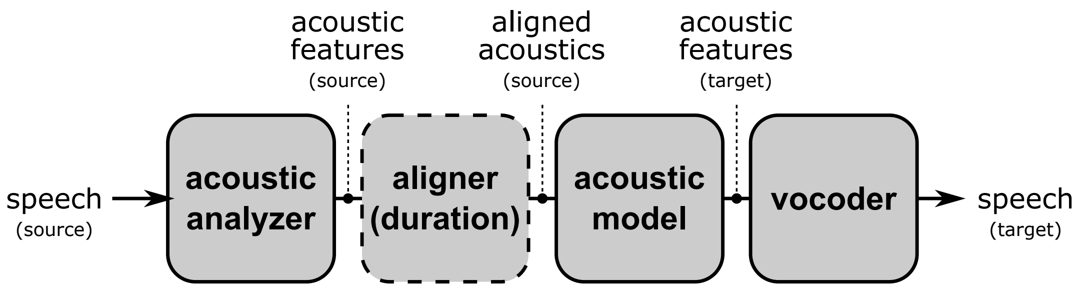


Figure 2.5: A conceptual representation of the conventional VC pipeline.

the main challenge is misalignment of two utterances in the training stage. The "aligner" module in this case can be techniques such as dynamic time warping (DTW) [51] or a seq2seq network [8, 9]. In the conversion stage, the aligner is not mandatory as we could use duration of source utterance directly. However, by doing so we miss the chance to further improve the speaker similarity by making the speaking rate of the generated utterance to be more similar to the target speaker [8, 52, 9]. It is dependent on the application scenarios to include or not include a duration/prosody converter.

As it is labor-intensive, and therefore expensive, to prepare parallel utterances, the parallel VC systems commonly have to build with as little as five minutes of speech data from a speaker [53]. This is inconvenient and also limits the performance of the VC systems in general.

2.2.2 Non-parallel VC systems

Many have worked on methodologies to train VC systems with non-parallel utterances to reduce the data demanding [54]. One approach is handling voice cloning at model-level by formulating a transformation function to adapt from pretrained models [55, 56] similar to the speaker adaptation of HMM-based TTS model described in Section 2.1.3. With the deep learning approach, the fundamental for non-parallel VC is learning a disentangled linguistic representation either implicitly or explicitly. For the implicit cases, Hsu et al. [57] used variational autoencoder (VAE) while Kameoka et. al. [54] used generative adversarial network (GAN) to train a *many-to-many* non-parallel VC system. These methods use multi-speaker data, conditional labels, and various forms of regularization to encourage the model to decouple linguistic content from speaker characteristics via a self-supervised training process. For the explicit cases, Sun et al. [58] used phonetic posteriorgrams (PPG) obtained from a SOTA ASR model to train an *any-to-one* non-parallel VC system. As an ASR model is speaker-independent, a PPG-based VC system can theoretically convert speech of arbitrary source

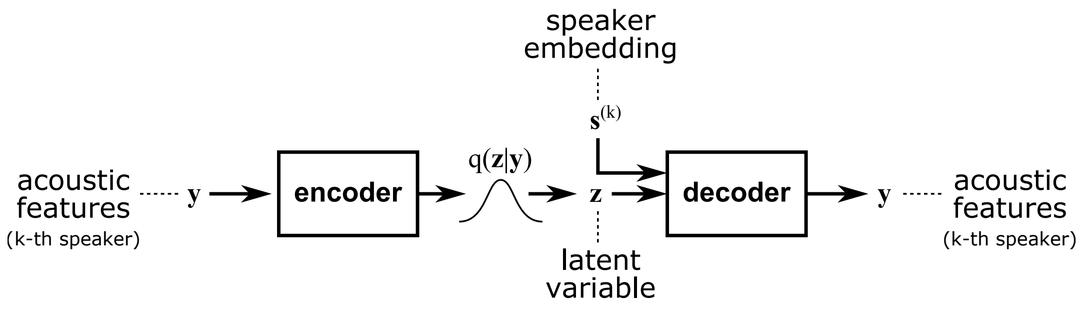

speaker embedding
s^(k)
q(z|y)
acoustic features (k-th speaker)
y
encoder
z
decoder
y
acoustic features (k-th speaker)
latent variable

Figure 2.6: Variational autoencoder based acoustic model for VC.

speakers to the target speaker.

Even though a typical VC system is only trained on speech data, recent works have suggested that using transcriptions of training data can further improve quality of generated samples [59, 10]. Having a solution in place to take advantage of additional resources whenever they become available provides a robust system which could tackle many diverse voice cloning scenarios.

2.2.3 Non-parallel VC with variational autoencoder

One of the main challenges for training a VC system is obtaining alignment between source and target speakers' utterances due to the duration and prosody mismatches. It is also the reason that non-parallel VC is more challenging than parallel VC as there is no alignment at all between utterances of source and target speakers. One deep learning based approach is using self-supervised training to learn a speaker-disentangled latent representation. A self-supervised neural network is trained with the same utterance used as input and output, so it does not have to deal with misalignment between source and target utterances.

The VAE-based VC system is one realization of this approach [57]. The general structure of the system is illustrated in Figure 2.6. In summary, the VAE-based acoustic model contains two modules, an encoder and a decoder. The encoder Enc, defined by its parameter ϕ, is designed to be independent from speakers and able to convert the speech input into a speaker-disentangled latent variable $\boldsymbol{z}$, or a distribution of it due to variational structure:

$$\boldsymbol{z} \sim Enc(\boldsymbol{y}; \phi) = q(\boldsymbol{z}|\boldsymbol{y}) \ . \tag{2.7}$$

On the other hand, the decoder Dec, defined by its parameters θ, takes the latent variable and a speaker representation $\boldsymbol{s}^{(k)}$ to reconstruct the speech input of the same k-th speaker in the training stage:

$$\tilde{\boldsymbol{y}} = Dec(\boldsymbol{z}, \boldsymbol{s}^{(k)}; \theta) \ . \tag{2.8}$$

In the conversion stage, we use a reference utterance of a source speaker but speaker embedding of the target to produce speech with same content as the reference but in the voice of the target speaker.

The viability and performance of the VAE-based VC system rely on two assumptions. First, the encoder can discard all speaker information while retaining the linguistic content even though no explicit constraint for such objective is deployed in the training stage. Second, a single speaker embedding is enough for the decoder to reconstruct speaker characteristics from a speaker-disentangled latent feature. In the fashion of vanilla VAE, we assume the latent variable has a simple prior distribution likes isotropic normal distribution which limits the expressiveness of latent space. This assumption acts as a regularization which forces the encoder to remove unnecessary information from speech, which we expect to be speaker characteristics as the decoder is already provided with such information through the speaker embedding.

The limitation of VAE-based systems is that we have to indirectly shape the latent space through regularization and constraints, such as putting a simple prior on the latent space, without the supervision of hard label or paired data. Therefore, we cannot be sure what kind of information the encoder discarded or what kind of information it retained. This is a basic property of self-supervised training method as it is dependant on the model to figure things out by themselves. To improve the self-supervised training, we can explicitly use an auxiliary speaker classifier to force the model to focus more on speaker characteristics [60], or incorporated GAN [61] and adversarial loss in general [54] to avoid the assumption of a simplistic prior distribution over the latent space.

2.2.4 Non-parallel VC with phonetic posteriorgram

The main principle of the VAE-based VC system described previously is that the model is somehow able to discover a speaker-disentangled latent feature containing linguistic information by itself. Based on this observation, instead of using a latent linguistic feature implicitly, we can use an explicit frame-based linguistic representation.

One of the most successful methods for building non-parallel VC systems is using PPG as a frame-based phonetic feature and training a TTS-like acoustic

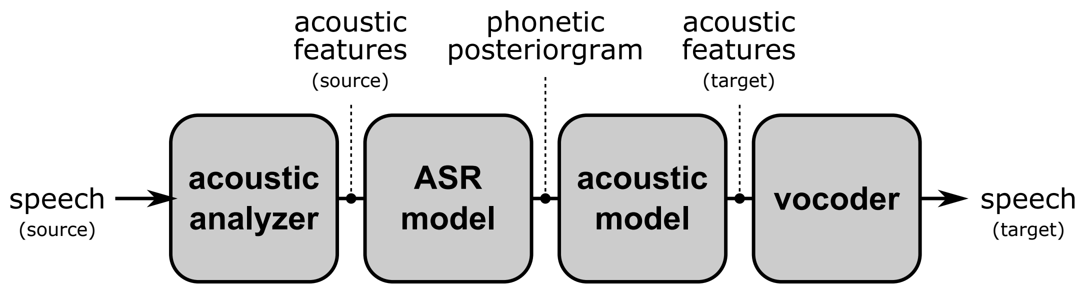


Figure 2.7: A conceptual representation of the PPG-based VC pipeline.

model [58]. The PPG, produced by a ASR system which is trained to classify words, phonemes or senones, contains phonetic and alignment information. Given the speaker-independent characteristic of ASR model, it is expected to generalize to unseen speakers and, therefore, able to be used for speakers who are not included in the training of ASR model.

Figure 2.7 illustrates the general pipeline of PPG-based VC systems. PPG is used to train a TTS-like acoustic model which transforms phonetic information extracted from an arbitrary source speaker's utterance to acoustic features with the target voice. In a way, PPG-based VC system is essentially a TTS model stacked on the top of an ASR model, but instead of text, an intermediate phonetic representation is used as text proxy. Auxiliary techniques that are generally used for TTS to improve the performance are also applicable to PPG-based VC systems. For examples, we can use a multi-speaker corpus to train an average model then adapt it to the target speaker to reduce data demand [62] or train a speaker-aware acoustic model by augmenting the PPG input with a speaker representation embedding [63].

Given a ASR model trained with thousands of hours of transcribed speech, the PPG-based approach produces high quality generated speech [14] and is able to maintain high consistency even when used for cross-lingual scenarios [64]. However, training SOTA ASR models is not an option for many languages due to the scarcity of speech data, especially transcribed speech. Moreover, a linguistic feature extractor, which is the ASR model in this case, trained independently from the acoustic model would prevent further improvement on performance, as many recent works have shown that an integrated and jointly trained model is the key for boosting performance with deep learning methods [65].

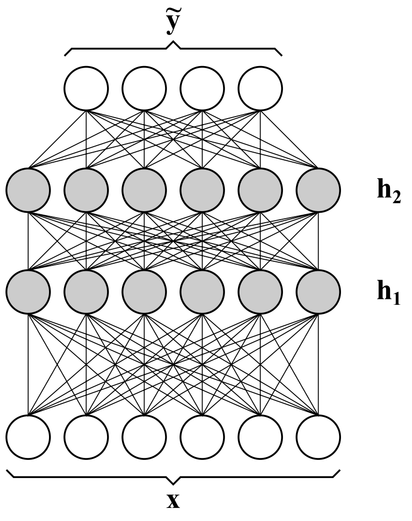


Figure 2.8: A simple deep feedforward neural network.

2.3 Deep learning

2.3.1 Neural network

The goal of neural network, or multilayer perceptron, is to approximate a function $\mathcal{F}^*$, which maps an input $\boldsymbol{x}$ to an output $\boldsymbol{y}$ by using a neural network $\mathcal{F}$ which is defined by its parameters $\boldsymbol{\Theta}$:

$$\tilde{\boldsymbol{y}} = \mathcal{F}(\boldsymbol{x}; \boldsymbol{\Theta}) \ . \tag{2.9}$$

The neural network $\mathcal{F}$ is a composition of L neural layers stacked on top of each others. A simple feedforward layer takes the input and transforms it into a latent feature using a non-linear transformation function:

$$\boldsymbol{h}_1 = f^{(1)}(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{c}_1) \ , \tag{2.10}$$

where $\boldsymbol{W}_1$ and $\boldsymbol{c}_1$ are weight and bias of the first hidden layer, and $f^{(1)}(.)$ is a non-linear activation function, such as sigmoid, hyperbolic tangent (tanh), rectified linear unit (ReLU), etc, used for the first layer. To increase expressive power of the model, we stack multiple hidden layers on the top of each others. The output of the current hidden layer is used as the input of the next layer:

$$\boldsymbol{h}_l = f^{(l)}(\boldsymbol{W}_l\boldsymbol{h}_{l-1} + \boldsymbol{c}_l) \ , \tag{2.11}$$

for $l \in [2, L-1]$. The last layer is responsible for mapping to the designated output. For a classifier, the output y is a category; while for a generator, the output is a desired target feature (e.g., image, speech, and video):

$$\tilde{\boldsymbol{y}} = f^{(L)}(\boldsymbol{W}_L \boldsymbol{h}_{L-1} + \boldsymbol{c}_L) \ . \tag{2.12}$$

As the output of the neural network, $\tilde{\boldsymbol{y}}$, is an approximation of the true target, $\boldsymbol{y}$; we want it to be as close to the true target as possible. This can be achieved by adjusting the network's parameters $\boldsymbol{\Theta}$ to better explain the training data using an optimization scheme. This is referred to as the training process of a neural network model. In general we refer to a feedforward neural network by using Equation 2.11. For advanced topics on neural network, please refer to the Deep Learning book [66].

2.3.2 Gradient descent and backpropagation algorithms

The neural network is trained with gradient descent algorithm. It iteratively adjusts the model's parameters to minimize a designated cost function:

$$E = Cost(\tilde{\boldsymbol{y}}, \boldsymbol{y}) \tag{2.13}$$

As the weights and biases of the network are the adjustable components, we calculate their gradients with respect to E:

$$\bigtriangledown E = (\frac{\partial E}{\partial W_{1,1}}, \frac{\partial E}{\partial c_{1,1}}, \frac{\partial E}{\partial W_{1,2}}, \frac{\partial E}{\partial c_{1,2}}, ..., \frac{\partial E}{\partial W_{l,j}}, \frac{\partial E}{\partial c_{l,j}}, ..., \frac{\partial E}{\partial W_{L,j}}, \frac{\partial E}{\partial c_{L,j}}) \tag{2.14}$$

Each parameter is updated with an increment as follow:

$$\triangle W_{l,j} = -\gamma \frac{\partial E}{\partial W_{l,j}} \tag{2.15}$$

where γ represents a learning rate which dictates the changing intensity in each training step. For classic (or batch) gradient descent, the gradient is calculated on the cost of all available training examples. However for a modern system which is trained on enormous amount of data, stochastic gradient descent (SGD) and mini-batch gradient descent, which use one or several examples in each step, are more popular approaches.

To calculate gradients for parameters which sit in lower layers of the network, we use the backpropagation algorithm. It applies the chain rule to calculate derivative of composite functions. Specifically, let x be a real number, and both f and g are real-valued functions. Suppose that $z = f^{(1)}(x)$ and $y = f^{(2)}(f^{(1)}(x)) =$

$f^{(2)}(z)$ then chain rules states that:

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial z} \frac{\partial z}{\partial x} \tag{2.16}$$

By applying the chain rule to neural networks, we are able to propagate the error from the output layer to hidden layers and calculate their corresponding gradients. Even though, in general, gradient descent and backpropagation are used together to train a neural network by jointly optimizing all parameters at each step, we can use them to adjust only certain parts of network while making the other parameters immutable. This is also fundamental for many backpropagation-based speaker adaptation methods based on neural models.

2.4 Deep learning for speech synthesis systems

Most of the recent advancements in te development of speech synthesis systems were achieved by replacing modules in the conventional pipelines (Figure 2.1 and 2.5) with neural-based counterparts trained with the backpropagation algorithm [6]. This approach reduces the methodology difference of each module and opens up the possibility to jointly optimize the entire system in an end-to-end (E2E) fashion which is important for boosting the performance [65]. This section describes the incorporation of deep learning into the speech synthesis pipeline.

2.4.1 Linguistic and acoustic analyzers

As text is a discrete and compact representation which lacks sophisticated linguistic information for speech generation while a speech waveform is high local temporal feature which contains thousands of data point per seconds, the linguistic and acoustic analyzer are used to transform the initial representations of text and speech into the ones that are better suited for the tasks at hand.

Linguistic analyzer

Traditionally, the linguistic analyzer of a TTS system extracts a set of handpicked language-dependent linguistic information which is deemed relevant for the text-to-speech mapping by experts. A linguistic toolkit, or front-end of TTS system, is developed to normalize and extract the information from the input text [67, 68]. However, recent works have shown that by letting the neural models figure out the relevant information by themselves we can improve the naturalness of the generated utterances [69, 65]. Figure 2.4 shows an example of this approach with the linguistic analyzer absorbed into a seq2seq TTS model. The text representation used in these model is normalized text or phonemes which greatly reduces the complexity of building a front-end system for a particular language. However, for certain languages (e.g., Japanese) augmenting the text representation with linguistic information which is deemed difficult to extract from text (e.g. pitch accent) seems to improve the performance [70, 71].

Most of the experiments in this thesis using off-the-shelf hand-crafted linguistic features of English and Japanese (details in Appendix B) to test the focused hypothesis of each chapter. In the very end, specifically the experiments in Chapter 8, phoneme input and a neural-based linguistic analyzer is used to boost the performance and move the system toward an end-to-end setup.

Acoustic analyzer

The type of acoustic used is dependent on the speech processing task. Generally, the speech signal is represented by prosodic and acoustic features but we do not always used both of them. For example, many ASR systems use exclusively mel-frequency cepstral coefficients (MFCC) as the acoustic input as pitch information is deemed irrelevant for English ASR systems. However, fundamental frequency (F0) seems to improve the recognition performance of many pitch-accented languages such as Vietnamese [72] and Cantonese [73]. For speech synthesis, Mel-generalized cepstral coefficient (MGC) and F0 are two main acoustic features for speech synthesizing. With recent end-to-end setup, it is desirable to use a single representation for both prosodic and acoustic features, so many have used spectrogram or mel-spectrogram [6] as the representation of speech.

Most of the experiments in this thesis use a parametric vocoder to synthesize speech with F0 and MGC acoustic features. For the experiments that synthesize speech with WaveNet vocoder, the mel-spectrogram is used as the acoustic feature. For the speech input of STS stack, either raw waveform or mel-spectrogram is used as the speech representation.

2.4.2 Sequence aligner and seq2seq model

One of the challenges of training a speech synthesis system is the misalignment of input and output sequences, it is relevant to both TTS and VC. As seen on Figure 2.1 and 2.5, we need an "aligner" module to handle this problem. In the inference stage, the aligner is used to determine the duration of generated speech in the case of TTS; while the in case of VC, generated speech can use the duration of the reference utterance, although a duration model dependent on the

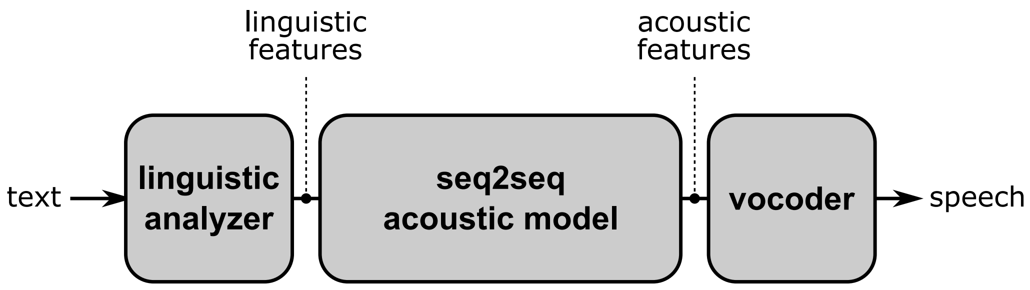


Figure 2.9: A conceptual representation of a seq2seq TTS pipeline.

target speaker might boost the speaker similarity further. The two main ways to approach the duration modeling is treat it as an model separated from the rest of the speech synthesis system or as an integrated model right from the start.

Separated duration modelling approach

In the conventional setup of neural speech synthesis models, we use an external duration model to align the input and output sequences. This duration model can be human annotations or automatic systems. In the case of TTS, we can use an HMM-based acoustic model to force align the linguistic and acoustic sequences. In the case of VC, we use Dynamic Time Warping (DTW) to align parallel utterances of source and target speakers. For the inference we can train a separated duration model to produce the necessary prosody information for generation or use duration of a reference utterances. Modeling duration information separately increases the complexity of the speech synthesis system but provides more control over prosody of generated speech. The separated duration approach is used throughout this thesis.

Integrated duration modelling approach (seq2seq model)

Many recent works have pushed forward an integrated neural speech synthesis model to reduce the complexity of the system. Specifically, many use seq2seq model to integrate the duration model into the acoustic model which is convenient and significantly reduces the complexity of the speech generation systems. For TTS, the linguistic analyzer is generally integrated into the seq2seq model [69]. The system described in Section 2.1.4 is an example (Figure 2.4). Although, the main problem solved by the seq2seq model is the misalignment between input and output so an integrated linguistic analyzer is not an obligation [65] as shown in Figure 2.9. For VC, a seq2seq model can help transform the speaking rate (duration) to that of the target along with the acoustic feature which further improves the similarity to the target speaker [8, 52]. In general, the development direction of deep learning approaches is replacing all the modules with neural-based components which simplify the setup and unify the methodology.

2.4.3 Neural acoustic model

The major part of this thesis focuses on improving the acoustic model of TTS and VC systems. Generally speaking, the acoustic model is the module which add the speaker characteristic of the target speaker from either a speaker-independent linguistic features or a source-speaker-dependent acoustic features. Most of the chapters will evaluate a particular aspect of a simple acoustic model while using off-the-shelf setups for other modules. Chapter 8 is where all proposed techniques are presented in conjunctions with relevant components of a modern speech synthesis system. The use of deep learning for acoustic models is straightforward when the input and output of it are assumed to be aligned sequences thanks to the prior and the subsequent modules in the pipeline. The neural acoustic model is essentially a trainable function which uses neural networks to model the target transformation [74, 75]. The parameters of the neural networks, $\mathbf{\Theta}^{tts}$ and $\mathbf{\Theta}^{vc}$, are trained with labeled or paired speech samples using gradient descent and back-propagation in a supervised fashion as described in Section 2.3.2. By framing the acoustic model of TTS and VC systems this way, they are no different from any other neural networks so it is easier to take advantage of available knowledge on representation learning and deep learning as a whole to tackle the voice cloning task.

2.4.4 Waveform generation

For a speech synthesis system, the outputs of the acoustic model are acoustic features instead of raw waveforms to reduce the complexity of the text-to-speech (or speech-to-speech) transformations, as a waveform is a high frequency and high temporal correlation feature. To synthesize waveforms from these features, a module referred as a "vocoder" is used. There are two main approaches for synthesizing speech waveforms from acoustic features, the conventional parametric approaches and data-driven neural vocoders.

Parametric vocoders

Parametric vocoders are designed based on an assumption (e.g., source-filter model). Given a acoustic feature sequence, contains F0 and spectral features, the parametric vocoder generates a segment of a waveform using a spectra frame then overlaps and adds all segments based on F0 information. Two commonly used vocoders for speech synthesis are STRAIGHT [76] and WORLD [77]. Parametric vocoders can be considered as an immutable and data-independent module.

Neural vocoders

With recent advances in deep learning, many works have tried to replace each component with a neural network based counterpart. One breakthrough of this trend is the proposal of neural vocoder system which is capable of generating waveforms one sample at a time. Leading in this field is WaveNet model [11] proposed by DeepMind, followed by many systems proposed to tackle similar systems like WaveGlow [12] and neural-source-filter (NSF) [13]. Neural vocoders can generate speech with better naturalness than the parametric vocoders thanks to their ability to generate sample-by-sample. However as a trained model, the performance of neural vocoders are dependant on their training data as any deep learning model.

2.5 Speech synthesis evaluation metrics

Speech generation systems are generally evaluated subjectively with human perception. However subjective measurements are expensive and take a very long time to prepare, so objective metrics are used as a complement to provide a quick feedback for the experiment developments.

2.5.1 Objective metrics

Depending on the setup of the speech synthesis model, the suitable objective metrics vary from experiment to experiment.

Mel-cepstral distortion (MCD)

For the acoustic model which outputs mel-cepstral features, we use mel-cepstral distortion [78] as an objective evaluation. The generated and the natural sequences are presumed to be aligned (i.e., same length):

$$MCD(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = \frac{\alpha}{|\boldsymbol{S}|} \sum_{\substack{t=1 \\ t \in \boldsymbol{S}}}^{T} \sqrt{\sum_{d=s}^{D} (y_t^d - \tilde{y}_t^d)^2} \tag{2.17}$$

$$\text{where } \alpha = \frac{10\sqrt{2}}{\ln 10} \approx 6.14185 \text{ and } s \in [1, 2] \tag{2.18}$$

The expression $t \in \boldsymbol{S}$ keeps speech segments while excluding silence frames from the calculation. In our experiments, the speech or silence frame is decided by linguistic information. The first dimension of mel-cepstral can be included or excluded. We set $s = 2$ which excludes the first dimension from the calculation.

F0 root mean squared error (F0 RMSE)

Fundamental frequency is another relevant feature that could provide information about similarity between generated and natural speech. We calculate F0 RMSE and use it as another point of reference when the two sequences are presumed to be aligned:

$$RMSE(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = \sqrt{\frac{1}{|\boldsymbol{V}|} \sum_{\substack{t=1 \\ t \in \boldsymbol{V}}}^{T} (y_t - \tilde{y}_t)^2} \tag{2.19}$$

As the mismatch between unvoiced frames of generated and natural F0 sequences can make the F0 RMSE unreliable, we only calculate the errors on frames which are marked as voiced by both sequences ($t \in \boldsymbol{V}$).

Mean squared error (MSE)

Mean squared error is used as loss function to train many acoustic model. For these cases we can also use MSE for objective evaluation. Specifically, we use MSE as an objective metric for two aligned mel-spectrogram sequences:

$$MSE(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = \frac{1}{T} \sum_{t=1}^{T} \sum_{d=1}^{D} (y_t^d - \tilde{y}_t^d)^2 \tag{2.20}$$

Optionally we can calculate MSE only on speech segments similar to MCD to produce more focused results.

Word error rate (WER)

Previous objective measurements presume aligned feature sequences, which limits their usage on experiment scenarios. We could use an ASR system to calculate the word error rate of the generated speech given the text reference:

$$WER(\text{reference}, \text{hypothesis}) = 100\,\frac{\#\text{substituted} + \#\text{deleted} + \#\text{inserted}}{\#\text{reference}} \tag{2.21}$$

The assumption here is that a generated speech sample with smaller word error has better pronunciation than the one with bigger error. While this might be true to some extent, we need to consider the training nature of ASR models. Even though they are trained on large-scale multi-speaker corpora, these ASR models are far from universal. Therefore, they might bias to certain setups of speech generation systems. So we need to treat the WER as a reference point instead of ultimate goal, the same for other objective metrics.

2.5.2 Subjective metrics

As the sole point of speech synthesis systems is communicating with human, human perception is the most important metric to evaluate speech synthesis systems. However, human perception is subjective and affected by the relative context so for each experiment the subjective test needs to be carefully prepared. The perceptive survey conducted in my thesis consists of mean opinion score (MOS) measure, in which participants have to give a rating in the range of 1 to 5 (or 4 in some cases), and AB questions, in which participants have to choose a superior option between choices.

Quality evaluation

All quality questions given in subjective survey of this thesis are evaluating a very generic sense of quality. The participants are asked to judge the naturalness in a general sense without providing any further instruction or scenario context. This ignored the nuance of human perception but allows a uniform framework across multiple experiments. For the MOS question, listeners are asked to judge a presented speech sample in 5-point scale: 1="Completely unnatural", 2="Mostly unnatural", 3="Equally natural and unnatural", 4="Mostly natural", and 5="Completely natural". For the AB question, listeners are asked to pick the more natural one out of two speech samples presented.

Similarity evaluation

Similar to the quality evaluation, generic similarity questions are used throughout this thesis even though, in some cases, the scenario context is extremely relevant (e.g., cross-language, non-native speaker). For the MOS question, listeners are asked to rate their confidence that two presented samples are spoken by the same speaker on a 4-point scale: 1=”Different (sure)”, 2=”Different (not sure)”, 3=”Same (not sure)”, and 4=”Same (sure)”. For certain experiments, a 5-point scale is used for similarity which inserts 3=”Can’t decide” into the scale. For the AB question, listeners are asked to pick one between two presented samples which has higher chance to be spoken by the same speaker as a reference utterance.

Chapter 3

Supervised adaptation with speaker-adaptive neural TTS

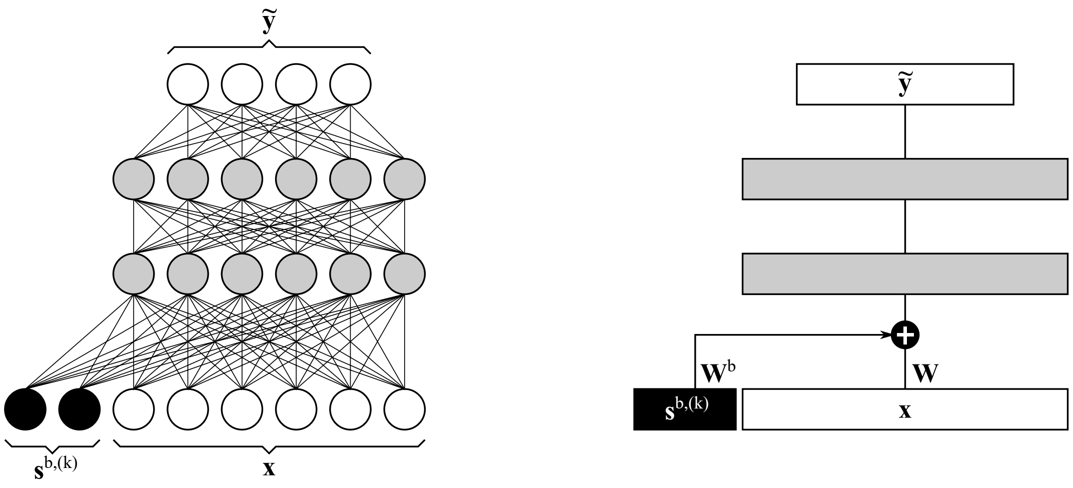
This chapter establishes the basics of the multi-speaker neural acoustic model and speaker adaptation method. Several experiments were conducted to provide initial observations about the fundamentals of these tasks. The content in this chapter is based on the paper [79] that I published before the start of the PhD program, but it is necessary to include in this thesis to provide the context and motivations for the other chapters.

Section 3.1 presents the fundamentals of the multi-speaker acoustic model based on neural network. Section 3.2 explains the supervised speaker adaptation method by tuning with the backpropagation algorithm. Section 3.3 introduces different types of speaker embedding which can be used as the speaker component in a multi-speaker neural acoustic model. Section 3.4 describes the experiments and highlights the relevant observations as well as remaining challenges which are the foundations for hypothesises that are tackled in later chapters.

3.1 Multi-speaker TTS with bias codes

3.1.1 Motivation for multi-speaker model

When the available data of a target speaker is insufficient to train a high quality speaker-dependent neural TTS system, we can combine data from multiple speakers and train a multi-speaker TTS model instead [28, 80]. The multi-speaker model can generate more stable speech waveforms than those of the speaker-dependent model [28, 81] when the amount of the target speaker's data is limited [19]. The multi-speaker TTS is especially important for low-resource languages



(a) Details illustration (b) Simplified illustration

Figure 3.1: Multi-speaker neural acoustic model.

in which it is difficult to gather a large quantity of data from a single speaker [18]. In general, simply combining all available data of all speakers is sufficient, however, many works have shown that complement data pruning policy [82, 83, 84] or techniques to handle data imbalance [81, 85] can further improve the naturalness of generated speech for all training speakers.

3.1.2 Modeling multi-speaker model with bias codes

By augmenting the conventional linguistic inputs of DNN-based acoustic models with auxiliary features, collectively referred to as *input codes*, which include speaker codes as well as encodings of other labelled attributes, such as gender and age, we can train a multi-speaker model from a speech corpus comprised of hundreds of different speakers, each contributing as little as ten minutes of speech. The training process of a neural multi-speaker model is identical to its single-speaker model with gradient descent and backpropagation algorithm.

A simple realization with feedforward layers of the neural multi-speaker model is illustrated in Figure 3.1. Basically, we append a speaker embedding $\boldsymbol{s}^{b,(k)} \in \mathbb{R}^{Q\times 1}$ which represents k-th speaker into every frames of the linguistic input $\boldsymbol{x}$. This also creates additional weight parameters in the first hidden layer. Comparing with the regular hidden layer described by Equation 2.10, the first hidden layer of neural multi-speaker model can be written as follow:

$$\boldsymbol{h}_1 = f^{(1)}(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{c}_1 + \boldsymbol{W}^b\boldsymbol{s}^{b,(k)}) \,, \tag{3.1}$$

where $\boldsymbol{W}^b \in \mathbb{R}^{m\times Q}$ is the additional weight created by the augmenting of the speaker embedding. Within a single utterance, the linguistic information changes every frame while the speaker embedding remains the same which makes $\boldsymbol{W}^b\boldsymbol{s}^{b,(k)}$ frame-independent. This essentially means all parameters of the network are shared among the training speakers except the bias of the first hidden layer. The first layer can be rewritten as follows:

$$\boldsymbol{h}_1 = f^{(1)}(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{c}_1 + \boldsymbol{b}^{(k)}) \tag{3.2}$$

$$\boldsymbol{h}_1 = f^{(1)}(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{c}_1^{(k)}) \ , \tag{3.3}$$

where $\boldsymbol{b}^{(k)} = \boldsymbol{W}^b\boldsymbol{s}^{b,(k)}$ is a speaker-specific bias projected from the embedding vector. In the speech generation stage, we use the speaker embedding vector of the desired speaker to change the bias of the first hidden layer. This, in turn, changes the entire transformation modeled by neural network sufficiently to change the voice of generated speech. The method is simple yet effective and does not depend on the network architecture [80, 86].

3.2 Adaptation with transcribed speech

Similar to the multi-speaker model, the motivation of supervised speaker adaptation is creating a TTS system with voices of target speakers whose data is limited. The difference between the multi-speaker task and the adaptation task is that the latter is expected to create models for speakers who are not included in the training set, which is faster than training a multi-speaker model from scratch as it only needs to fine-tune with a limited amount of data of the target.

While there are many different methods of supervised adaptation proposed which tackle different aspects of the adaptation process, they all share the same principle which is using transcribed speech and the backpropagation algorithm to optimize the model to better explain the new data at hand. The supervised adaptation methods with neural model can be categorized into two main types, we either tune all the network's parameters or just a specific part of it.

3.2.1 Adapting by fine-tuning a pretrained network

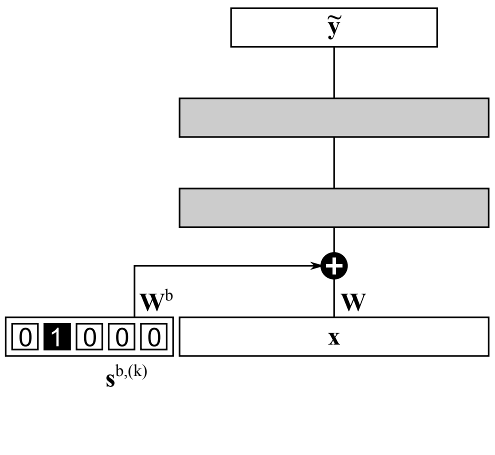
This approach is exactly as its name suggests, we simply optimize a pretrained model using the new data of the target speaker [87]. The basic concept is that even though the voice of target speaker is different from the ones used to train the initial model, the network can still transfer the knowledge it learned about linguistics to the target. The initial model can be trained on data of a single speaker [26] or data pooled from multiple speakers [27, 28]. This fine-tuning approach produces a high quality model when data of target speaker is sufficient but becomes overfitting easily when the amount of data is limited. The main research incentive for this approach is to develop novel techniques to balance improvement of speaker similarity and model overfitting, which causes unstable performance. For example, we can deploy some form of regularization [88] or a tactic to stop the adaptation process early [21]. This approach is not the focus of this chapter, but will be revisited later in Scenario B of the experiment section in Chapter 8.

3.2.2 Adapting by estimating new speaker embedding

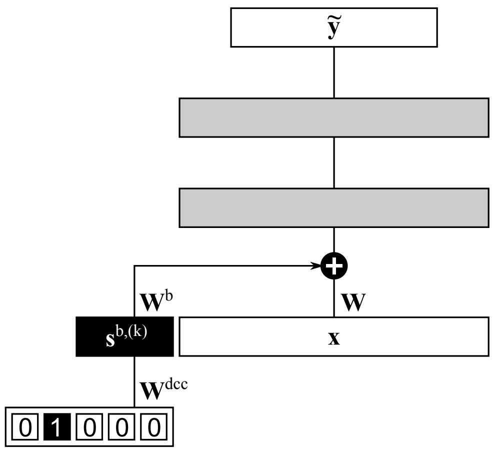
When data is extremely limited (e.g. one minute), we can restrict the tuning to a certain part of the network to prevent overfitting [28, 89]. This approach can be seen as a type of regularization applied to the adaptation process. By limiting the amount of adaptable parameters, we essentially discourage the adapted model to stray too far from the initial model. More elaborate discussion on this subject is presented in Chapter 5. Specifically, given the multi-speaker model described in Section 3.1.2, we can use transcribed speech and backpropagation to estimate a new speaker embedding for an unseen target speaker. The multi-speaker model shows that changing the speaker embedding input can change the voice of the generated utterances, which suggests the feasibility of this method. While the intuition is clear, there are several concerns about this premise. First, the back-propagation is an optimizing algorithm and it can be stuck at a local maximum so the best performance is not guaranteed. Second, while the embeddings of the speakers in the training set produce speech with consistent linguistic content and speaker characteristics, it is not a given for the unseen embedding as this assumes a continuity and completeness characteristic of the speaker embedding space. This approach is the main focus of this chapter. The objective is not create a high performance voice cloning method for TTS yet but to understand the mechanism and nature of speaker-adaptive neural acoustic model.

3.3 Speaker embedding types

To understand the nature of speaker embedding space we use different types of embedding vectors as speaker codes and observe their effects on the performance of multi-speaker and adaptation tasks.



(a) Speaker one-hot vector



(b) Speaker discriminant condition code

Figure 3.2: Different types of speaker embedding.

3.3.1 One-hot vector

The most simple speaker embedding type is one-hot vector. It is defined as follows: If there are N speakers in the training set, the *one-hot vector* for the k-th speaker is $\boldsymbol{s}^{b,(k)} = (s_1^{b,(k)}, s_2^{b,(k)}, ..., s_N^{b,(k)})$ where each value $s_i^{b,(k)}$ is 0 when $i \neq k$ and 1 when $i = k$. Figure 3.2a illustrates a simple speaker one-hot vector used in a multi-speaker model. As speaker bias $\boldsymbol{b}^{(k)}$ of the first hidden layer is projected from the embedding $\boldsymbol{b}^{(k)} = \boldsymbol{W}^b \boldsymbol{s}^{b,(k)}$, a one-hot vector would make the speaker biases of training speakers independent from each other. For this reason, we refer to one-hot vector based speaker embedding as *speaker (full) bias* for the rest of this thesis.

3.3.2 Random vector

To understand the speaker latent space of the multi-speaker model, we can use a randomized vector as the speaker embedding. Speaker random vector is a vector with predetermined dimension Q. The *random vector* for the k-th speaker is $\boldsymbol{s}^{b,(k)} = (s_1^{b,(k)}, s_2^{b,(k)}, ..., s_Q^{b,(k)})$ where $s_i^{b,(k)}$ are non-zero random values. This scheme is also a simple method to reduce or expand the size of the speaker code. The random vector is used to test several hypotheses. First, whether or not a unique but meaningless vector can be used for a multi-speaker model. Second, how forcing the speaker biases to share a latent space (due to non-zero, continuous values) would affect the performance of the multi-speaker and the speaker adaptation tasks.

3.3.3 Discriminant condition code

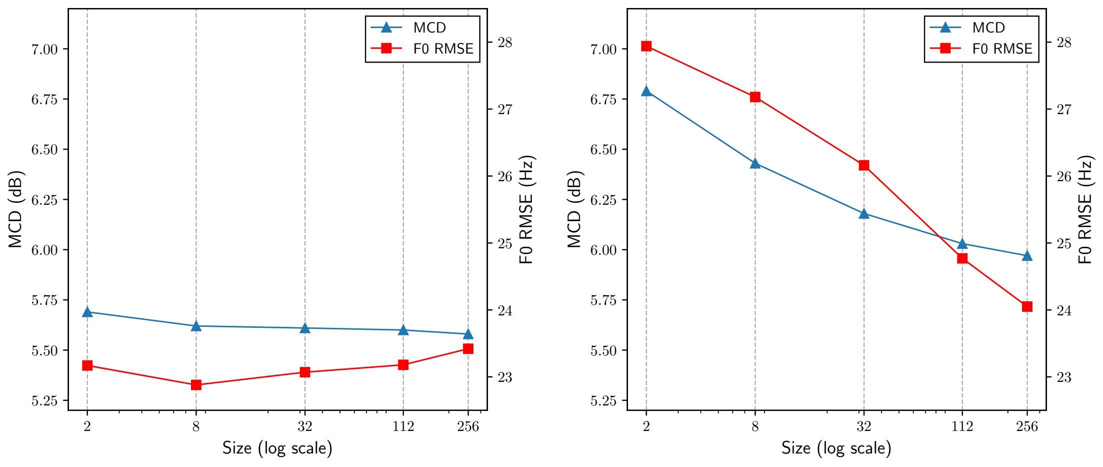
While a random vector is convenient for testing the underlining nature of the embedding space, we do not expect it to produce high performance as the random vector is meaningless therefore lacks useful information about speaker. A more sophisticated approach is training a continuous speaker embedding along the the model, which is referred to as discriminant condition code (DCC) in several literature [90, 91]. Watts et al. [92] uses a similar ideal to distinguish between sentences. Figure 3.2b shows a simple representation of the multi-speaker model with speaker discriminant condition code. Discriminant codes constitute hidden-unit activation obtained by projecting one-hot speaker codes into Q dimensions using a matrix $\boldsymbol{W}^{\text{dcc}} \in \mathbb{R}^{Q \times N}$. This means that the continuous speaker vector $\boldsymbol{s}^{b,(k)}$ is jointly trained and "stored" in the DCC matrix $\boldsymbol{W}^{\text{dcc}}$. The discriminant code approach can also be interpreted as a bottleneck layer which forces the model to learn a dense and compact latent space. For this reason, we refer to it simply as *(speaker) bias code* for the rest of this thesis.

3.3.4 Speech-encoded vector

Instead of jointly training the speaker embedding along with the acoustic model, we can use an external module to extract the speaker representation. Specifically, we want a speaker encoder which is capable of extracting the speech-encoded speaker vector from a given speech sample. Section 2.1.4 described a specific realization of this approach. Speaker recognition systems are generally used as the speaker encoder module as we expect them to produce a meaningful vector with relevant information about speaker characteristics [34, 36]. The main advantage of speech-encoded speaker embedding is its ability to perform speaker adaptation with untranscribed speech. This speaker embedding type is not evaluated in this chapter, but will be revisited in the experiment section of Chapter 8.

3.4 Experiments

To establish the basic understanding about the speaker-adaptive neural acoustic model, we first examine the performance of multi-speaker and adaptation tasks of the speaker-adaptive acoustic model using different types of speaker embedding. The main goal of the following experiments is establishing the relative performance gap between the multi-speaker and adaptation rather than the absolute performance of speech synthesis systems. Therefore, a simple feed-forward neural



(a) Multi-speaker task (b) Adaptation task

Figure 3.3: The effect of the speaker bias code size on the objective measurement of the multi-speaker and adaptation tasks.

network with five non-linear layers is used for the multi-speaker acoustic model. The speaker embedding is added at the input layer along with the linguistic features. More details about the model configuration can be found in the Appendix C.1. Japanese linguistic input and the dynamic multi-task acoustic features for the STRAIGHT vocoder are used in this experiment.

The experiments are conducted with the Japanese voice bank corpus (JVB, Appendix A). The train and test sets of *jvb.small.base* are used to train the multi-speaker model and evaluate the multi-speaker task. The *jvb.small.target* sets are used for evaluating the adaptation task. The corpus also includes age and gender information of speakers which are used as auxiliary input in the experiment. These information can be controlled to manipulate the speaker characteristics. However, only multi-speaker and adaptation tasks are discussed in this section. The information for the speaker, age, and gender manipulation task can be found in the original paper [79].

3.4.1 Effect of speaker embedding size on performance

The advantage of discriminant bias code over a simple one-hot vector is its adjustable size. By concentrating the speaker parameters into a small size vector, it potentially can be used for disentangling and manipulating speech characteristics [79]. To have understanding about the relation between the bias codes and the acoustic model we observe the effect of its size on the objective performance of an acoustic model.

Table 3.1: Objective evaluations of multi-speaker and adaptation tasks for model trained with different types of speaker embedding.

Task	Embedding type	Speaker code	MCD	F0 RMSE
		bias	100 utt.	100 utt.
multi-AVG	average vector	112	7.64	52.06
multi-OHV	one-hot vector	112	5.58	23.43
multi-RND112	random vector	112	5.60	23.48
multi-RND008	random vector	8	5.60	23.06
multi-DCC112	discriminant code	112	5.60	23.18
multi-DCC008	discriminant code	8	5.62	22.88
adapt-AVG	average vector	112	7.49	53.90
adapt-OHC	one-hot vector	112	6.01	23.54
adapt-RND112	random vector	112	6.46	24.98
adapt-RND008	random vector	8	6.49	29.44
adapt-DCC112	discriminant code	112	6.03	24.77
adapt-DCC008	discriminant code	8	6.43	27.18

Figure 3.3a shows the objective performance of the multi-speaker task. The general trend that can be seen is that a bigger vector gives better MCD than a smaller one, while it is more complicated in case of F0 RMSE. However, the improvement over MCD is slowing down pretty quickly. The most interesting observation here is that a 2-unit or 8-unit bias code seems to be enough to create multi-speaker acoustic model with 112 different voices. Figure 3.3b shows the objective performance of the speaker adaptation task. Generally speaking, both measurements benefit from bigger bias codes. However comparing with the multi-speaker task, the adaptation is generally not as good. These results suggest that the amount of adaptable parameters is highly relevant to performance of the adaptation.

Observation 1. *The size of the bias code does affect the performance of speaker-adaptive model and bigger is generally better. However, the improvement on multi-speaker task is marginal and unsteady, while it is more notable in the case of the speaker adaptation task.*

3.4.2 Effect of speaker embedding type on performance

This experiment evaluates performance of the first three types of speaker embedding, described in Section 3.3, on objective performance of multi-speaker and adaptation tasks. The experiment setting details and objective results are listed in Table 3.1. The size of the one-hot vector depended on the amount of speakers in the training set, which is 112 in this scenario. The size of other embedding types are either set at 112, to remove the mismatch with one-hot vector, or 8, representing a small speaker embedding vector.

The *average vector* is essentially the *one-hot vector* model with the value of each unit set at $\frac{1}{N}$, with $N = 112$ in this scenario. The average model (average voice) provides a naive anchor point to check if the multi-speaker and adaptation tasks with the speaker embedding are working correctly by observing the objective measurement. Interestingly, neither the embedding type nor the size of speaker vector has a significant impact on the multi-speaker task. This suggests that neural model able to compensate for the meaningless speaker vector by using the rest of its parameters. However, in the adaptation case, the models trained with random vectors and 8-unit discriminant code have significantly worse objective evaluations than the rest. One hypothesis for this phenomenon is that as the adaptation stage does not have the rest of the network's parameters at its disposal like the training stage, the usefulness of the speaker embedding is critical for its performance.

Observation 2. *Neither the size nor the embedding type of the speaker code has a significant impact on objective evaluations of the multi-speaker task. However, the adaptation task greatly benefits from a bigger and better (meaningful) speaker representation.*

Between multi-speaker and speaker adaptation tasks, the latter is worse than the former as expected. Balancing between the resistance to overfitting and overall performance is one of the most important aspects of speaker adaptation method, which is also a recurring theme throughout this thesis.

3.4.3 Manipulating speaker characteristic

The multi-speaker and adaptation tasks can be interpreted as speaker manipulation with a very specific agenda, which is producing a generated voice close to a target speaker by using a respectable speaker embedding vector. To further test the ability to manipulate speaker characteristics we can interpolate the speaker embedding from the value of one speaker to another within a single utterance. This experiment is not evaluated but only observed. The speech samples for such manipulation can be found at `www.hieuthi.com/papers/icassp2017/`. Interestingly, the generated utterances maintain the desired linguistic content with the speaker characteristic smoothly transitioning from one speaker to another. This suggests that the speaker embedding space is a continuous latent space.

Observation 3. *Given a speaker embedding space jointly trained with the acoustic model, gradually changing speaker code values produces a fictitious voice while maintaining the linguistic content. It suggests that the speaker embedding space is continuous, consistent and somewhat complete.*

While speaker manipulation is an interesting experiment by itself, the above observation has an important implication on the speaker adaptation methodology introduced in this chapter. If the speaker latent space is continuous and complete then there is no ‘wrong’ embedding. In the other words every embedding in this space guarantees generated speech samples with consistent quality. So for speaker adaptation tasks, the remaining problem is finding the embedding which produces speeches sounds most similar to the voice of target speaker.

Chapter 4

Unsupervised speaker adaptation with multimodal neural TTS

This chapter proposes a novel speech synthesis system using multimodal neural network architectures. The multimodal structure allows the system to perform unsupervised speaker adaptation with untranscribed speech by utilizing its cross-modal adaptation ability. By carefully factorizing and training the initial model, the unsupervised adapted model can generate speech with high speaker similarity and marginal difference in performance compared to a supervised adapted model.

Section 4.1 explains the motivation of speaker adaptation using untranscribed speech and introduces existing approaches. Section 4.2 describes the novel multimodal neural TTS and explains the mechanism for performing unsupervised speaker adaptation. Section 4.3 introduces strategies for training multimodal neural structures to achieve designated goals. Section 4.4 presents the experiments which test the feasibility of the proposed method.

4.1 Adaptation with untranscribed speech

Between supervised speaker adaptation and multi-speaker tasks, the former is a faster process as we do not train the entire model from scratch. However, it do not provide any additional benefits in terms of data efficiency as both tasks require transcribed speech from the target speaker. Practically speaking, speech of a speaker can be collected passively, but its transcriptions require high attention to create which makes its collection process labour-intensive.

In theory, as speaker characteristics are self-contained within speech utterances we should be able to adapt without using transcription. The speaker adaptation methods that work with untranscribed speech, which are referred as unsupervised speaker adaptation, will able to clone thousands of voices quickly and cheaply. Moreover removing the transcription obligation will greatly increase the amount of data of the target speaker which is usable for the adaptation process. As unsupervised speaker adaptation, for a neural TTS model in particular, is a rewarding task, there are many methods proposed to tackle this problem. At the start of my PhD research, the existing methods could be grouped into two main approaches which will be briefly explained next.

4.1.1 Adapting with ASR-predicted transcription

One simple and practical approach to tackle an unsupervised speaker adaptation task is obtaining automatically annotated transcriptions using a SOTA ASR system [31]. Given ASR-predicted transcriptions, the speaker adaptation can be performed in the same fashion as the transcribed case (see Section 3.2). However ASR-predicted transcriptions are expected to contain more wrong annotations than manually-annotated one, which would certainly affect the performance of the adaptation process. Moreover this approach assumes that a well-trained ASR is obtainable for the target language, which is impractical for many low-resource languages [18]. It also removes the possibilities of performing cross-language speaker adaptation [32, 16].

4.1.2 Adapting with text-independent speaker embedding

Given the disentanglement ability of deep learning, another approach for unsupervised speaker adaptation is training a speaker-adaptive model conditioned on a speaker representation extracted from speech [15, 33]. The speaker representation can be *i-vector* [34], *d-vector* [35, 19], or *x-vector* [36], which are all byproducts of speaker recognition systems. Section 2.1.4 describes in details a particular realization of this approach. This approach has the computational advantage as it does not involve the optimization loop [15]. The drawback is that it does not scale with the amount of adaptation data. In the other words, the speaker similarity seems to stop improving when using more than a few seconds of speech [21].

4.2 Multimodal speech synthesis system

The unsupervised speaker adaptation with ASR-predicted transcription (Section 4.1.1) is more versatile as it does not assume the structure of the speaker component and the speaker transformation. Therefore, we want a novel unsupervised adaptation method which has similar advantages but without any of its drawbacks mentioned above.

The key difference between the approach described in Section 4.1.1 and the approach described in Section 4.1.2 is that the adaptation process of the former is performed with the backpropagation algorithm, the same as supervised adaptation, while the latter only involves the forward pass. The multimodal neural TTS system is proposed based on this observation. The problem that it addresses is "can we have a method with similar traits to the ASR-based approach but skipping the text generating step and performing adaptation directly to reduce the limitations of ASR-based approach?". The main idea is to jointly train an ASR-like module with the acoustic model using the same multi-speaker corpus. At the same time we published our multimodal neural TTS system [93], Karita et. al. [94] proposed a similar structure for E2E ASR. However, the motivation and the specific method are quite different as they use multimodal structure to enable the semi-supervised training instead of unsupervised speaker adaptation as in our case.

4.2.1 Interchangeable text and speech encoders

Given the multi-speaker model described in Section 3.1.2, its mechanism can be described as follows: the speaker component (speaker bias in this case) alters the function modeled by neural layers to transform speaker-independent linguistic features to speaker-dependent acoustic features with the voice of a particular target. In other words, the hidden features before the speaker component are speaker-independent while the hidden features after it are speaker-dependent. Based on this assumption we can split the acoustic model into two modules: a text encoder and a speech decoder. The underlining network structure is unchanged, only the theoretical description is revised.

The original acoustic model is now described as the text-to-speech stack which comprises of the text encoder and the speech decoder. Output of the text encoder, $\boldsymbol{z}$, is called latent linguistic embedding (LLE) from now on. The new TTS transformation can be written as follows:

$$\boldsymbol{z}^T = TEnc(\boldsymbol{x}; \phi^T) \tag{4.1}$$

$$\tilde{\boldsymbol{y}}^T = SDec(\boldsymbol{z}^T; \theta^{\mathrm{S/core}}, \theta^{\mathrm{S/spk},(k)}) \tag{4.2}$$

The text encoder $TEnc$ encodes a text/linguistic sequence $\boldsymbol{x}$ to a continuous

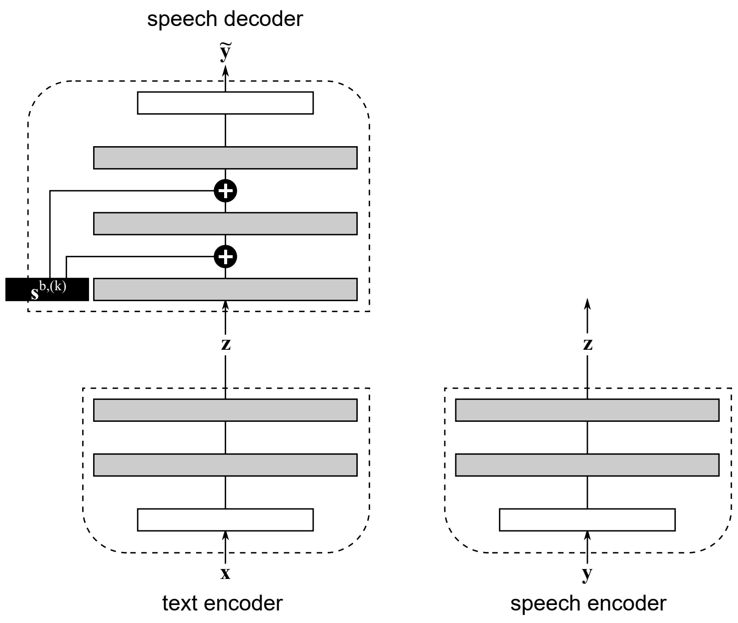


Figure 4.1: The multimodal speaker-adaptive speech synthesis system.

latent representation $\boldsymbol{z}$. The text encoder is a neural network structure defined by its parameter ϕ^T which is speaker-independent and shared among all training speakers. The speech decoder $SDec$ consumes the LLE sequences produced by the text encoder and transforms it into speech/acoustic sequences with characteristics of the target speaker. A speech decoder is defined by its parameters $\theta^{\mathrm{S/core}}$ and $\theta^{\mathrm{S/spk,(k)}}$, which are the speaker-independent and speaker-dependent parameters (e.g. speaker bias), respectively. By changing the speaker-dependent parameters $\theta^{\mathrm{S/spk,(k)}}$ we change the voice of the generated utterances. This is a factorized and generalized mathematical description of the multi-speaker model described in Section 3.1.2. As the TTS stack is just a typical neural acoustic model, it can be trained by optimizing a designated loss function between generated and natural speech features:

$$\mathrm{loss}_{tts} = Cost(\tilde{\boldsymbol{y}}^T, \boldsymbol{y}) \ . \tag{4.3}$$

Next, we introduce a new speaker-independent module, the speech encoder $SEnc$, which is defined by its parameters ϕ^S, that can extract LLE from a given utterance. Its purpose is to be used as substitute for text encoder and to create a speech-to-speech (STS) stack:

$$\boldsymbol{z}^S = SEnc(\boldsymbol{y}; \phi^S) \tag{4.4}$$

$$\tilde{\boldsymbol{y}}^S = SDec(\boldsymbol{z}^S; \theta^{\mathrm{S/core}}, \theta^{\mathrm{S/spk,}(k)}) \ . \tag{4.5}$$

The speech encoder can be interpreted as an ASR model, but instead of text or phonemes it extracts an unobservable/hidden linguistic representation. Typically we can train such STS model in a self-supervised manner by optimizing a designated cost function just like the TTS stack:

$$\text{loss}_{sts} = Cost(\tilde{\boldsymbol{y}}^S, \boldsymbol{y}) \ . \tag{4.6}$$

However, if we train the STS stack separated from the TTS stack, we cannot guarantee a consistent latent space between the two which is the main objective of the multimodal neural TTS. The remaining challenge of the proposed method is how to train all modules to create a consistent latent space. The training methods are discussed in Section 4.3.

4.2.2 Crossmodal adaptation with untranscribed speech

Assuming that we able to obtain a well-trained multimodal multi-speaker model as illustrated in Figure 4.2.1, we are then able to use the STS stack to perform unsupervised speaker adaptation with backpropagation. Specifically, the STS stack is used to estimate a new set of speaker-dependent parameter $\theta^{\text{S/spk},(r)}$ of the unseen r-th target speaker. It is essentially the same as supervised adaptation approach described in Section 3.2.2 which is evaluated in the previous chapter.

$$\text{loss}_{adapt} = \text{loss}_{sts} \ . \tag{4.7}$$

The obtained speaker-dependent parameter $\theta^{\text{S/spk},(r)}$ is then used in the TTS stack to generate speech with the voice of the new target. The main hypothesis is that if the linguistic latent space of the TTS and STS stacks are consistent with each other, we can reuse the parameter "found" (with backpropagation) in one stack for the other, hence the name crossmodal adaptation.

4.2.3 Autoencoder for shaping latent space

The STS stack has a structure similar to an autoencoder, which transforms a speech utterance $\boldsymbol{y}$ to itself. Autoencoder [66] is a neural network trained to copy its input to its output. However generally we are not interested in the copying task but hope the training process will create a latent feature $\boldsymbol{z}$ that has useful properties. Specifically in our case, we want $\boldsymbol{z}$ to contain linguistic information but none of the speaker characteristic. To achieve this, we can use an undercomplete autoencoder by make $\boldsymbol{z}$ have smaller dimensions than $\boldsymbol{y}$ or deploy auxiliary regularization in the training stage.

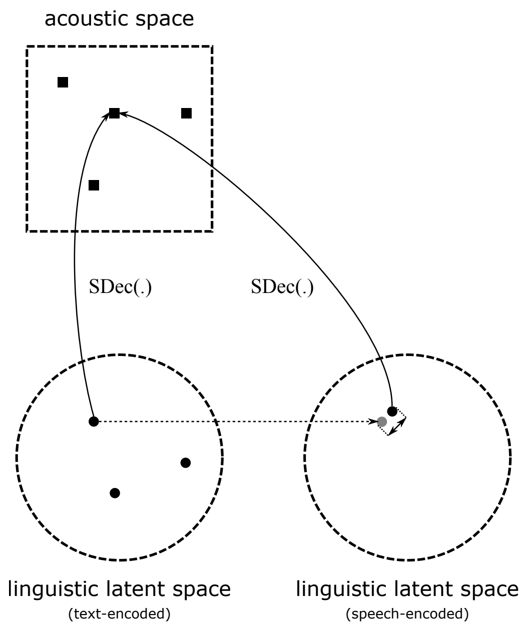


Figure 4.2: Latent and acoustic spaces of the multimodal TTS.

However the proposed multimodal system as a whole is technically not an autoencoder, as the “auto” part in its name implies the network is trained in a self-supervised manner which is not the case. The proposed multimodal learning methods, which will be described in the next section, use both speech and its transcription to train all modules of the system. In the other words, the entire network is still trained in a supervised fashion, it is just that both speech and text are unconventionally placed at the input. Nevertheless, due to the loose use of these terminologies, the jointly training method, which will be explained in the following, can also be interpreted as a regularization method for autoencoder model.

The main objective for using a multimodal structure is that we want to train a speech-encoded latent space that approximates the text-encoded latent space as illustrated in Figure 4.2. To achieve this goal we want to ensure the continuity of the latent space. In the other words, we want the speech decoder to be able to transform latent points which are close in the latent space into acoustic features which are close in the acoustic space. This is one motivation for the development of multimodal learning methods introduced in the following section.

4.3 Multimodal learning methods

This section describes several multimodal learning methods which are techniques used to train neural structures with multiple interchangeable encoders. The term multimodal is used to differentiate with multitask which is a neural structure with multiple decoders (or output features). However, similar to many deep learning concepts, these terms are used loosely unless they are important to the context. Most of the methods introduced next were used in several other works without being given a name or emphasis on the multimodal structure. This section is an attempt to provide a systematic and consistent overview on the subject.

4.3.1 Step-by-step training

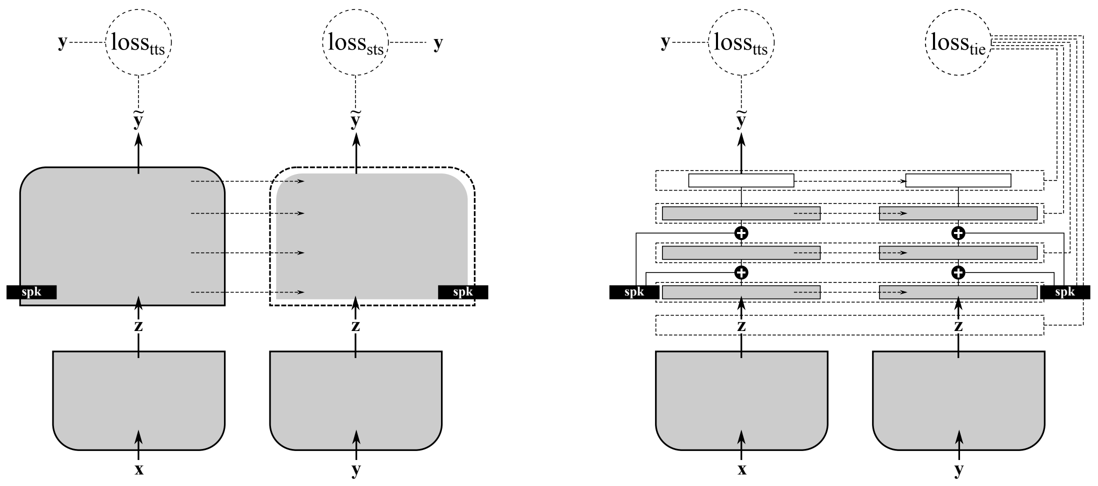
A simple strategy for training a multimodal neural structure is optimizing one stack until convergence then switching to the other. Specifically for our case, we first train the TTS stack using loss_{tts}, then replace the text encoder with the speech encoder and train the speech encoder using loss_{sts} while freezing parameters of the speech decoder. As our objective is to obtain the speech encoder that can substitute the text encoder, we want to train the text encoder first. The orders of training and the immutable parameters are dependent on the particular task [95]. Although this strategy is reasonable, we do not expect it to lead to sufficient results as it treats the speech encoder as an afterthought.

4.3.2 Stochastic training

The step-by-step training method creates an order of importance for each stack as only the first stack has all parameters at its disposal. To remove this order we can train all modules stochastically [96]. Specifically, we train all the modules concurrently by randomly switching input data, encoders, and the optimizing loss functions [97, 98]. The switching of modules in each training step changes the optimizing landscape constantly which makes it harder for the model to converge, or it might not converge at all. Although this strategy would work for our task, due to the large number of factors to be compared in our experiment, it is not investigated in this thesis.

4.3.3 Joint-goal training

Given the limitations of previous training methods, we want a method that can jointly train all modules together in each step which is the main motivation for joint-goal method. Specifically, we treat the TTS and STS stacks as two separated networks which share the parameters of the speech decoder as illustrated in Figure 4.3a. To recreate a priority order for the optimizing objective we can use weighting



(a) Joint-goal (b) Tied-layer

Figure 4.3: The joint-goal and tied-layer multimodal learning methods.

hyperparameters to emphasize or de-emphasize a particular goal:

$$\text{loss}_{train} = \text{loss}_{tts} + \alpha \ \text{loss}_{sts} \tag{4.8}$$

where α is a weighting parameter for the loss function of the less important or less reliable goal. The joint-goal training strategy functions similarly to multitask learning [99]. By weighting the sum of the losses while sharing the weights of the common layers, we expect that the models are aware of the additional but less important goal of the second stack. This acts as a method of regularization to help preserve some weights in the common layers to process the secondary input.

This training strategy assumes that by forcing the output of two stacks to approximate the same target, the network will automatically discover a shared latent space with the continuity characteristic.

4.3.4 Tied-layer training

As the joint-goal method only focuses on the output of the entire stacks, we should consider putting constraints on the hidden layers as well. In particular we want to the hidden layers that are common between TTS and STS stacks to generate latent features that are close to each other as illustrated in Figure 4.3b. For this purpose, we use a "distance" function to measure the differences:

$$\text{loss}_{train} = \text{loss}_{tts} + \beta \ \text{loss}_{tie} \ , \tag{4.9}$$

where β is a weighting parameter of the latent tying loss which forces the hidden layers of the two stacks to be close to each other. For the tied-layer loss, loss_{tie}, we can decide which layer and how many layers we want to put this constraint on. The optimal setup is dependent on each particular task:

$$\text{loss}_{tie} = \sum_{\substack{l=1 \\ l \in \boldsymbol{F}}}^{L} \text{distance}(\boldsymbol{h}_l^{tts}, \boldsymbol{h}_l^{sts}) \ , \tag{4.10}$$

where L is the number of hidden layers, $\boldsymbol{h}$ represents the outputs of a hidden layer, and $\boldsymbol{F}$ is the set of the layers that we want to includes in the *tied-layer* loss, it could be a single layer or all shared hidden layers between the two stacks. The "distance" functions can be cosine distance or Euclid distance.

In contrast with the joint-goal strategy, this strategy assumes that by forcing the latent spaces to be close to each other the outputs of the TTS and STS stacks will also become similar to each other.

4.3.5 Joint-goal and tied-layer training

The main purpose of multimodal learning methods is training the speech encoder to approximate the text encoder. To achieve this we want to enforce the continuity of shared latent space as discussed in Section 4.2.3. Interestingly, the joint-goal and tied-layer strategies focus on different aspect of this problem and they are complementary to each other:

$$\text{loss}_{train} = \text{loss}_{tts} + \alpha\ \text{loss}_{sts} + \beta\ \text{loss}_{tie} \ . \tag{4.11}$$

The above loss function explicitly describes what we referred to as the continuity of shared latent space. Specifically, the tied-layer loss (loss_{tie}) forces the speech-encoded latent feature to be close to the text-encoded latent feature, while the goal losses (loss_{tts} and loss_{sts}) make sure these latent points will produce the same/similar acoustic features.

4.4 Experiments

The purpose of following experiments is to test the feasibility of using the multimodal neural TTS for unsupervised speaker adaptation. A 5-layer feedforward network similar to the previous chapter is used for the acoustic model, with the first two hidden layers assigned to text encoder and the rest are assigned to speech

Table 4.1: Acoustic models trained with different multimodal learning methods.

Model	Method	Speaker-aware	Loss weights α	β
VL	vanilla	all layers	-	-
SS	step-by-step	last 2 layers	-	-
JG	joint-goals	last 2 layers	0.5	-
TL	tied-layers	last 2 layers	-	1.0
JT	JG+TL	last 2 layers	0.2	0.2

Table 4.2: Objective evaluations of multimodal TTS on multi-speaker task.

Strategy	Speaker code bias	MCD $\sim$400 utt.	F0 RMSE $\sim$400 utt.
multi-VL	128	5.85	15.1
multi-SS	128	5.71	15.0
multi-JG	128	5.57	14.6
multi-TL	128	5.79	15.3
multi-JT	128	5.63	14.8

decoder. The speech decoder take raw waveform input instead of acoustic features to avoid the complication with multiple acoustic features output. Details can be found in Appendix C.1. The English linguistic input used in these experiments is described in Appendix B. Table 4.1 described the configuration for multimodal neural TTS. For the tied-layer loss, constraint is only put on first common hidden layer. For the speaker modeling, a single speaker embedding is shared between multiple layers instead of using one for each.

The voice cloning toolkit English corpus (VCTK, Appendix A) is used for experiments. The train and test sets of *vctk.small.base* are used for training and evaluating the multi-speaker models, respectively. The *vctk.small.target* are used for the adaptation task. The experiments also test the effect of augmenting a single bias code on multiple speaker-aware layers.

4.4.1 Effect of multimodal learning methods

Even though the multimodal setup does not change the essential components of the TTS acoustic model (the stack comprising text encoder and speech decoder), it does alter the training process. This experiment evaluates the effect of different training methods on the performance of the original multi-speaker acoustic model. Table 4.2 shows objective evaluation of the multi-speaker task. The multi-speaker models of VL and SS are trained with the same conventional TTS loss (Equation 4.3), with the only difference coming from their speaker-aware layers. The ob-

Table 4.3: Objective evaluations of multimodal TTS on supervised and unsupervised speaker adaptation tasks.

Task	Adaptation	Speaker code	MCD		F0 RMSE	
		bias	10 utt.	320 utt.	10 utt.	320 utt.
VL-su	supervised	128	6.48	6.23	13.88	12.32
SS-su	supervised	128	6.15	6.04	13.30	13.47
JG-su	supervised	128	6.03	5.89	13.63	12.74
TL-su	supervised	128	6.12	6.00	12.93	11.59
JT-su	supervised	128	6.05	5.90	13.73	13.69
SS-un	unsupervised	128	6.30	6.24	13.40	13.83
JG-un	unsupervised	128	6.17	6.10	13.72	12.58
TL-un	unsupervised	128	6.13	6.10	15.36	12.21
JT-un	unsupervised	128	6.16	6.00	13.73	12.90

jective results show no significant difference between their performance. In fact, differences between all training methods are all marginal. This suggests that the proposed multimodal training methods do not hurt performance of multi-speaker task, but enhances them slightly in certain cases.

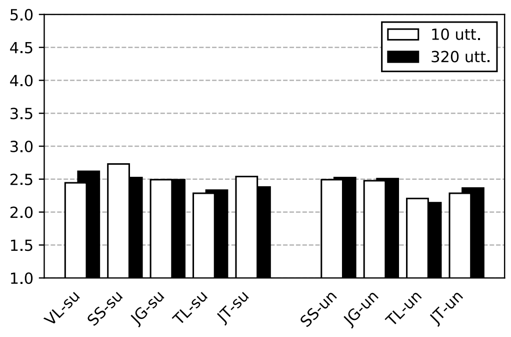
Observation 4. *Given a complementary network setup, the multimodal learning method able to create an extra functional module without affecting negatively on performance of the original multi-speaker task. In several cases, it even improves the performance of the original task.*

It is concluded that the multimodal training method does not have negative impact on the initial multi-speaker model, and provides an auxiliary speech encoder that could be used for performing unsupervised speaker adaptation. The behavior of multimodal learning is quite similar to multitask learning, in the sense that by jointly optimizing multiple relevant objectives, it seems to improve performance of all objectives involved.

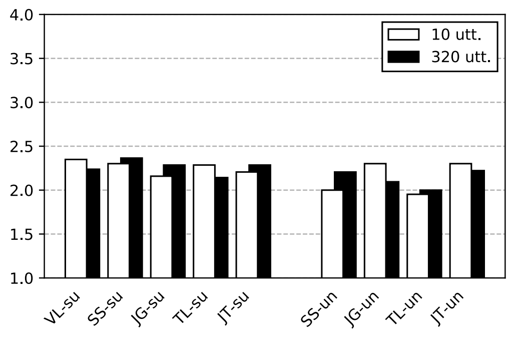
4.4.2 Performance of crossmodal speaker adaptation

Given the extra speech encoder, we can now perform unsupervised speaker adaptation by using untranscribed speech to estimate a new speaker bias code for the unseen target speaker (see Section 3.2.2). The same multimodal system can perform adaptation with transcribed and untranscribed speech, as the text and speech encoders are expected to be interchangeable.

Table 4.3 shows the objective evaluations of adaptation tasks. VL-su and SS-su are essentially the supervised adaptation method investigated in Section 3.4.2 with slightly different speaker component setups. Interestingly, VL-su are slightly



(a) Quality



(b) Similarity

Figure 4.4: Subjective evaluations of multimodal TTS on supervised and unsupervised speaker adaptation tasks.

worse than SS-su which suggests that sharing a single speaker code across multiple layers acted as a constraint which restricts the adaptation process. There are no significant differences between supervised adaptation of the model trained with multimodal learning methods, this reassures that multimodal learning methods do not affect the behavior of the original models.

The important results are the performance of unsupervised speaker adaptation. In general, the unsupervised adaptation is slightly worse than its supervised counterpart as expected, but the difference is marginal. Among the unsupervised systems SS-un stands out with the worst performance. This suggests that the jointly training methods are better than step-by-step training. However the quality and similarity of generated speech is still low in general as shown in the subjective results presented in Figure 4.4. Generally speaking, the unsupervised strategies are worse than supervised strategies in both measurements as expected. Among the unsupervised strategies *JG-un* and *JT-un* are promising with comparable performance with supervised baselines. It suggests that given a complementary setup, the unsupervised strategy is potentially able to reach the same performance of the supervised strategy.

Observation 5. *By restricting the flexibility of the speaker component, a multimodal neural TTS can perform adaptation with untranscribed speech using backpropagation algorithm in a similar manner and performance as the adaptation with transcribed speech.*

While the performance of the unsupervised speaker adaptation and the supervised adaptation are still low, the results of the current experiment provide a proof-of-concept for feasibility of the crossmodal speaker adaptation method which is the foundation for further improvements introduced in later chapters.

Chapter 5

Modeling speaker-adaptive TTS with scaling and bias codes

This chapter investigates different ways to incorporate a speaker component into neural network by reviewing prior works on the subject for both ASR and TTS acoustic models. The principles are then applied for modeling speaker-adaptive TTS models in the form of speaker scaling and bias codes. Through the experiments with the scaling and bias codes we can better understand the relation of linguistic feature, acoustic features and speaker transformation.

Section 5.1 systematically reviews different ways to model speaker transformation within a neural acoustic model. Section 5.2 introduces a method of using scaling and bias codes as speaker component for speaker-adaptive neural TTS and present the hypothesis about the non-linear and linear nature of the speaker transformation. Section 5.3 describes the experiments and relevant observations about the subject.

5.1 Neural speaker component

The speaker component is a crucial aspect of a speaker adaptation methodology as it directly affects the speaker footprint and performance of the adapted model. As explained in Section 3.2, we can adapt by simply tuning the entirely of a pretrained network. However due to the large amount of parameters but limited amount of data, it is vulnerable to overfitting. Regularization or heuristic training policy can be used to alleviate this problem. Instead of regularization, we can also limit adaptable parameters to a specific part of the network. The speaker component can be just one [28] or several layers [26]. These layers can be further factorized [100] to discourage the adapted model of the target speaker from straying too far from the initial state. In this section, we categorize these factorized methods on the type of transformation they model within a single layer of the neural network.

5.1.1 Speaker layer

Due to the hierarchy structure of a neural network, we can have one specific layer as speaker-dependent:

$$\boldsymbol{h}_l = f^{(l)}(\boldsymbol{W}_l^{(k)}\boldsymbol{h}_{l-1} + \boldsymbol{c}_l^{(k)}) \tag{5.1}$$

These speaker layers are strategically placed at input [101], output [97] or in-between the hidden layers [26] depending on the task. The weights and biases can be factorized in various ways to further reduce the speaker footprint [100, 26].

5.1.2 Speaker weight

Instead of using the whole layer, we can use the weight as speaker dependant while the bias is common for all speakers:

$$\boldsymbol{h}_l = f^{(l)}(\boldsymbol{W}_l^{(k)}\boldsymbol{h}_{l-1} + \boldsymbol{c}_l) \tag{5.2}$$

The motivation behind this is by reducing the amount of speaker-dependent parameters it would reduce the risk of overfitting for speakers with less data. The amount of parameters can be further reduced by using other techniques. Singular value decomposition (SVD) bottleneck [102] factorizes the full matrix $\boldsymbol{W}_l^{(k)} \in \mathbb{R}^{m\times m}$ into products of several low-rank matrices

$$\boldsymbol{W}_l^{(k)} = \boldsymbol{U}_l\boldsymbol{A}_l^{(k)}\boldsymbol{V}_l \tag{5.3}$$

where $\boldsymbol{U}_l \in \mathbb{R}^{m\times n}$, $\boldsymbol{V}_l \in \mathbb{R}^{n\times m}$ and $\boldsymbol{A}_l^{(k)} \in \mathbb{R}^{n\times n}$. By setting $n \ll m$ the speaker specific parameters become much less numerous than using the square matrix $\boldsymbol{W}_l^{(k)}$. Similarly, in the cluster adaptive training (CAT) method proposed by Tan et al. [103], the speaker weight is estimated based on an interpolation between several canonical matrices and hence the interpolation coefficients $\boldsymbol{\lambda}_l^{(k)} \in \mathbb{R}^p$ are speaker-specific parameters:

$$\boldsymbol{W}_l^{(k)} = \sum_{i=1}^{p} \lambda_{l,i}^{(k)}\boldsymbol{W}_{l,i} \tag{5.4}$$

where $\boldsymbol{W}_l^{(k)}$ depends on the canonical set $\boldsymbol{M}_l = \{\boldsymbol{W}_{l,1}, ..., \boldsymbol{W}_{l,p}\}$. Factorized hidden layer (FHL) [104] exercises a similar concept, modeling the speaker weight as a subspace over a finite set of canonical matrices.

5.1.3 Speaker bias

The layer bias by itself has also been proven to be an effective speaker-specific parameter [80, 79].

$$\boldsymbol{h}_l = f^{(l)}(\boldsymbol{W}_l \boldsymbol{h}_{l-1} + \boldsymbol{c}_l^{(k)}) \tag{5.5}$$

In practice we model a speaker bias by augmenting the input or hidden layer(s) with a one-hot vector representing the speaker [86] while still keeping an extra common bias:

$$\boldsymbol{h}_l = f^{(l)}(\boldsymbol{W}_l \boldsymbol{h}_{l-1} + \boldsymbol{c}_l + \boldsymbol{b}_l^{(k)}) \tag{5.6}$$

where $\boldsymbol{b}_l^{(k)} \in \mathbb{R}^{m \times 1}$ is a speaker-specific bias projected from the speaker one-hot vector. We could factorize the speaker bias further by using a continuous vector to represent the speaker instead of using the discrete one-hot vector.

$$\boldsymbol{b}_l^{(k)} = \boldsymbol{W}_l^b \boldsymbol{s}_l^{b,(k)} \tag{5.7}$$

This arbitrary-sized bias code $\boldsymbol{s}_l^{b,(k)} \in \mathbb{R}^{q \times 1}$ can be obtained by jointly training with the model [91] or be immutable and assigned with a meaningful embedding obtained with an external system (e.g. i-vector [105]). Whether the bias code is mutable or immutable, the training process still optimizes $\boldsymbol{W}_l^b \in \mathbb{R}^{m \times q}$ which is a universal subspace matrix for all speakers in the training stage.

5.1.4 Speaker scaling

Instead of factorizing the existing weight or bias of a layer, using an additional speaker dependant diagonal matrix as a scaling operation has also been proven to be useful for speaker modeling. For example learning hidden unit contribution (LHUC) [106] uses a speaker-dependent vector $\boldsymbol{a}_l^{(k)} \in \mathbb{R}^{m \times 1}$ to adjust the output of the hidden layers:

$$\boldsymbol{h}_l = \boldsymbol{a}_l^{(k)} \circ f^{(l)}(\boldsymbol{W}_l \boldsymbol{h}_{l-1} + \boldsymbol{c}_l). \tag{5.8}$$

If we look at it from the perspective of the next layer, $\boldsymbol{a}_l^{(k)}$ is basically a diagonal scaling matrix:

$$\boldsymbol{h}_{l+1} = f^{(l)}(\boldsymbol{W}_{l+1} \mathrm{diag}(\boldsymbol{a}_l^{(k)}) \boldsymbol{h}_l + \boldsymbol{c}_{l+1}) \tag{5.9}$$

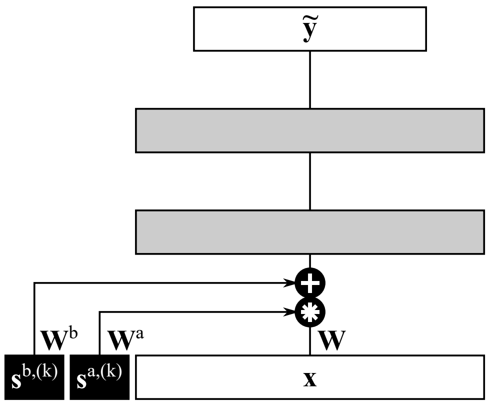


Figure 5.1: Scaling and bias codes for speaker-adaptive neural TTS.

where diag is the operation of changing an $m \times 1$ vector into a diagonal $m \times m$ matrix. Just like the speaker weight and bias, speaker scaling can be factorized further using the subspace approach. Samarakoon et al. [107] proposed subspace LHUC, in which $\boldsymbol{a}_l^{(k)}$ is projected from a vector $\boldsymbol{s}_l^{a,(k)} \in \mathbb{R}^{p\times 1}$ of arbitrary size p by using a SI matrix $\boldsymbol{W}_l^a \in \mathbb{R}^{m\times p}$:

$$\boldsymbol{a}_l^{(k)} = 2 \times \sigma(\boldsymbol{W}_l^a \boldsymbol{s}_l^{a,(k)}) \tag{5.10}$$

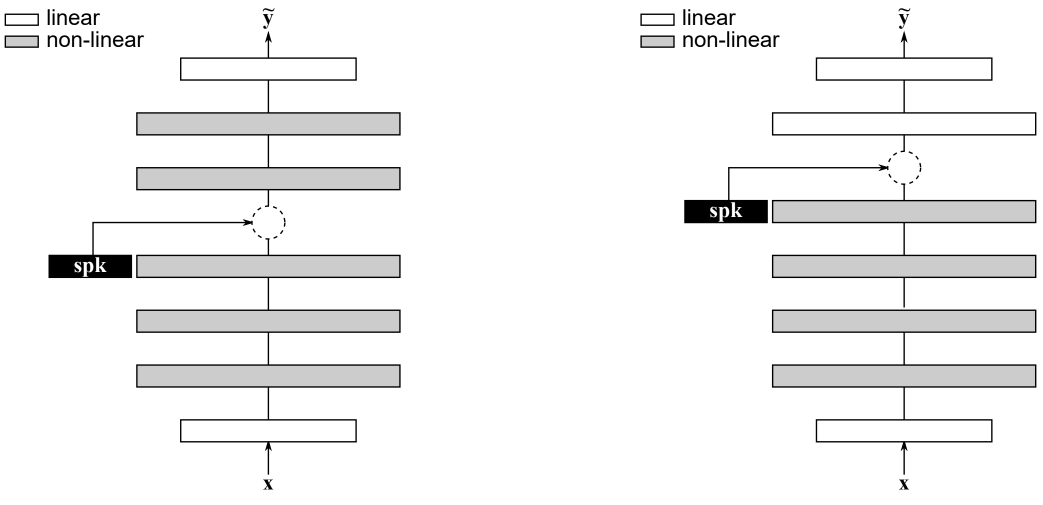
Other variations of subspace speaker scaling are investigated in [29] and [108].

5.2 Scaling and bias codes for neural TTS

Speaker bias by itself is enough to model a speaker-adaptive speech synthesis system as suggested by the experiments in previous chapters. However, there is a significant gap in performance between the multi-speaker and the speaker adaptation task. To improve the performance of adaptation task while maintaining its resistance to overfitting, we want to enhance the expressive power of the speaker components while maintaining the amount of adaptable parameters.

5.2.1 Scaling and bias codes as speaker components

Based on the prior works presented in Section 5.1, we can increase the expressive power of the speaker transformation but not significantly increase the amount of speaker-dependent parameters by using speaker scaling in conjunction with



(a) non-linear injection point (setup1) (b) linear injection point (setup2)

Figure 5.2: The non-linear and linear injection points of the speaker components.

speaker bias:

$$\boldsymbol{h}_l = f^{(l)}(\text{diag}(\boldsymbol{a}_l^{(k)})\boldsymbol{W}_l\boldsymbol{h}_{l-1} + \boldsymbol{c}_l + \boldsymbol{b}_l^{(k)}) \tag{5.11}$$

To further reduce the amount of adaptation parameters, we can apply the subspace approach to factorize the speaker scaling and bias into scaling and bias codes by using universal speaker projection matrices which are shared among speakers in the training set:

$$\boldsymbol{a}_l^{(k)} = \boldsymbol{W}_l^a \boldsymbol{s}_l^{a,(k)} \ , \tag{5.12}$$

$$\boldsymbol{b}_l^{(k)} = \boldsymbol{W}_l^b \boldsymbol{s}_l^{b,(k)} \ , \tag{5.13}$$

where $\boldsymbol{s}_l^{a,(k)} \in \mathbb{R}^{p\times 1}$ and $\boldsymbol{s}_l^{b,(k)} \in \mathbb{R}^{q\times 1}$ are scaling and bias codes of the l-th layer and have arbitrary size p and q. These codes are projected into a speaker scaling and bias $\boldsymbol{a}_l^{(k)}$, $\boldsymbol{b}_l^{(k)}$ by using the SI matrices $\boldsymbol{W}_l^a \in \mathbb{R}^{m\times p}$ and $\boldsymbol{W}_l^b \in \mathbb{R}^{m\times q}$. This reduces the number of parameters to be fine-tuned in the adaptation stage.

5.2.2 Linear and non-linear setups

The mechanism of multi-speaker neural TTS is that the speaker component changes the transformation function modeled by the neural network, so that the network is capable of transforming a speaker-independent latent feature to a speaker-dependent feature. The power to model complex transformation functions is the product of hierarchy and non-linear structure of the neural network.

The question is does the non-linear transformation benefits the speaker modeling of the multi-speak neural TTS. Due to the hierarchy structure, the speaker

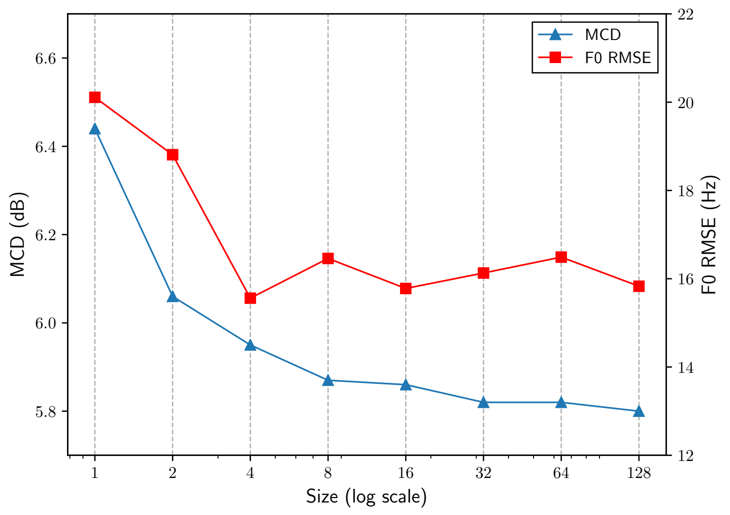


Figure 5.3: The effect of the speaker scaling code size on the objective measurement of the multi-speaker task.

component essentially changes the transformation of the function modeled by layers placed above it, while it has no effect on the function modeled by the layers placed below it. To understand the relationship between linguistic input and acoustic output we test the effect of injection point for speaker components on the performance. Figure 5.2a represents a scenario in which the speaker component changes a non-linear function as it is placed below two layers with non-linear activation functions, while Figure 5.2b represents a scenario in which the speaker component changes a linear function.

5.3 Experiments

The purpose of the following experiments is testing the ability to use a scaling code by itself or in conjunction with bias code as the speaker component. The same 5-layer feedforward network used in previous chapters is reused. Details can be found in Appendix C.1. The experiments in this chapter test the focused hypothesis on both English and Japanese TTS systems. The linguistic features used for each language can be found in Appendix B. The configuration of the speaker component is dependent on each particular experiment.

We perform the experiments on [corpus].*medium* sets of VCTK and JVB corpus to make sure the finding is consistent across languages. Both the multi-speaker and adaptation tasks are evaluated on the [corpus].*medium.target* sets of these corpora to provide a consistent condition and allow a direct comparison between multi-speaker and adaptation tasks.

Table 5.1: The objective evaluations for multi-speaker task of speaker-adaptive acoustic model using scaling and bias codes.

Task	Speaker code		MCD (dB)	F0 RMSE (Hz)
	scaling	bias	320 utt.	320 utt.
multi-setup1-bias	-	64	5.81	16.93
multi-setup1-scale	64	-	5.82	16.49
multi-setup1-bias+scale	32	32	5.90	16.42
multi-setup2-bias	-	64	6.53	17.68
multi-setup2-scale	64	-	5.85	16.38
multi-setup2-bias+scale	32	32	5.87	17.08

5.3.1 Effect of scaling code size on multi-speaker task

The first experiment is conducted with VCTK corpus to test the effect of varying scaling code size on the multi-speaker task. The *setup1* network is used as the acoustic model. The objective evaluation is presented in Figure 5.3. This corresponds to experiments which test the effect of the varying bias code size presented in Figure 3.3a. In general, objective results of both F0 RMSE and MCD are getting better the bigger the scaling code is. However the performance is quickly converges. This result is similar to the bias code case and proves that a scaling code by itself can be used as the speaker component of a multi-speaker acoustic model.

Observation 6. *Scaling code by itself can be used as the speaker component of a multi-speaker acoustic model. The size of the scaling code affects the performance of multi-speaker task in a similar manner to the bias code with a bigger vector generally performing better than a smaller one.*

5.3.2 Scaling and bias codes for English TTS

Next, we test performance of scaling and bias codes on both multi-speaker and adaptation tasks of the English TTS system. We first evaluate the effect of scaling and bias codes with different network setups on a multi-speaker task. The combined size of scaling and bias codes is kept at 64 as listed in Table 5.1. All strategies show marginal difference but *multi-setup2-bias*, which has the worst score in MCD while maintaining a competitive F0 RMSE evaluation. This result suggests that the relationship between linguistic and F0 can be modeled by linear function while it is not the case between linguistic and spectral feature. This observation is perpetuated in the setup of many VC systems which transform F0 of a source to a target speaker's using a linear function instead of non-linear

Table 5.2: The objective evaluations for adaptation task of speaker-adaptive acoustic model using scaling and bias codes.

Task	Speaker code		MCD (dB)		F0 RMSE (Hz)	
	scaling	bias	10 utt.	320 utt.	10 utt.	320 utt.
multi-setup1-bias	-	64	6.17	5.81	22.20	16.93
adapt-setup1-bias	-	64	6.29	6.27	21.84	22.39
adapt-setup1-scale	64	-	6.19	6.16	21.13	19.49
adapt-setup1-bias+scale	32	32	6.26	6.26	22.21	21.38
adapt-setup2-bias	-	64	6.48	6.47	15.88	14.78
adapt-setup2-scale	64	-	6.19	6.19	19.99	17.99
adapt-setup2-bias+scale	32	32	6.29	6.26	17.35	15.82

function.

Table 5.2 shows the objective evaluation of the adaptation task with different amounts of adaptation data. The *multi-setup1-bias* is included as the upper bound. We can see that the multi-speaker task greatly benefits from increasing amounts of data while the adaptation task does not. The *adapt-setup2-bias* stands out from the rest with the worst MCD and the best F0 RMSE similar to its multi-speaker counterpart. Between other adaptation systems, the ones which include a scaling code seem to be slightly better than the baseline with just the bias code *adapt-setup1-bias*. This confirms the potential of using scaling codes for speaker-adaptive acoustic models.

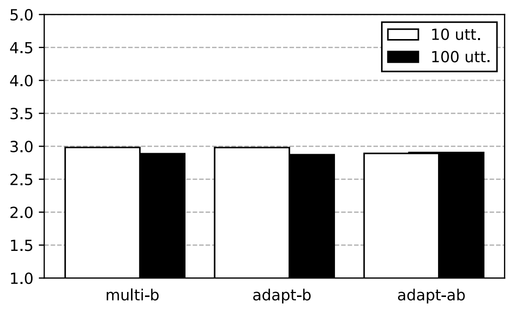
Observation 7. *The speaker component which is most suitable for modeling a transformation between a linguistic feature and an acoustic feature is dependent on the nature of the acoustic feature. Therefore, using a multitask network to model all acoustic features together may have a negative impact on the performance.*

5.3.3 Scaling and bias codes for Japanese TTS

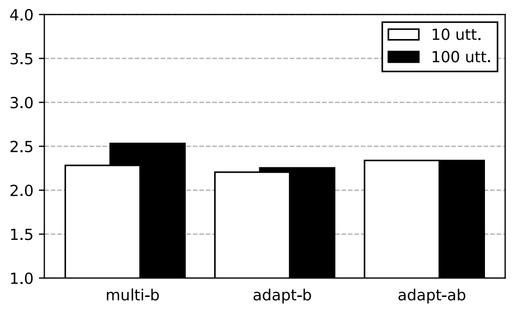
In this experiment, we want to test our hypothesis on scaling and bias again with a Japanese TTS system. As Japanese is a pitch-accent language [109], in which pitch plays a bigger role on the meaning conveyed than English, it is interesting to know if the same conclusion is still substantial. Only a few selected strategies are recreated for the Japanese TTS experiment which includes the multi-speaker task baseline with bias code *multi-b* which is short for *multi-setup1-bias*, the adaptation task baseline with bias code *adapt-b* which is short for *adapt-setup1-bias* and the proposed adaptation strategy with scaling and bias code *adapt-ab* which is short for *adapt-setup2-bias+scale*. Their objective evaluations are listed in Table 5.3 with the 10-utterance and 100-utterance scenarios. Between the adaptation

Table 5.3: Objective evaluations for selected strategies with Japanese corpus.

Strategy	Speaker code		MCD (dB)		F0 RMSE (Hz)	
	scaling	bias	10 utt.	100 utt.	10 utt.	100 utt.
multi-b	-	64	5.23	5.07	27.76	25.99
adapt-b	-	64	5.32	5.27	28.25	27.17
adapt-ab	32	32	5.25	5.23	25.96	25.90



(a) Quality



(b) Similarity

Figure 5.4: Subjective evaluations for selected strategies with Japanese corpus.

baseline *adapt-b* and the proposed strategy *adapt-ab*, the latter has slightly better scores in all data points with noticeable improvement on F0 RMSE metric. There is marginal difference between multi-speaker and adaptation strategies in 10-utterance case, but the multi-speaker strategy *multi-b* is significantly better than the adaptation strategies in the 100-utterance case.

Figure 5.4 presents subjective evaluation of the Japanese generated speech samples. Interestingly, there is not much difference in the quality score between strategies. In similarity measurement, however, the multi-speaker strategy with 100 utterances has the best performance as expected. The strategy with scaling and bias codes are consistently better than the adaptation baseline with just bias code even though the amount of adaptable parameters is identical.

Observation 8. *By using speaker scaling code along with speaker bias code to model the speaker transformation, we can improve performance of speaker adaptation with the speaker-adaptive neural TTS without having to increase the amount of adaptable parameters.*

The experiments in this chapter show that the key to improving the performance of speaker adaptation is improving the expressive power of the speaker component. Even though the performance of the speaker adaptation task is improved by using scaling code along with bias code, it is still worse than the multi-speaker task baseline.

Chapter 6

Training a robust speaker-adaptive multimodal neural TTS

As Chapter 4 proposes a novel unsupervised speaker adaptation method and Chapter 5 investigates relevant aspects of the adaptation process, this chapter addresses their remaining limitations and provides solutions to improve upon the established method. Specifically, we introduce the variational multimodal neural TTS which greatly increases the consistency of the text-encoded and speech-encoded linguistic latent spaces. This consistency allows us to relax constraints on speaker components which, in turn, greatly improve performances of both supervised and unsupervised speaker adaptation tasks.

Section 6.1 addresses the remaining limitations of methods introduced in Chapter 4 and Chapter 5. Section 6.2 describes the enhanced system which is inspired by variational autoencoder. Section 6.3 introduces a new hypothesis about performance of the speaker adaptation approach by finetuning the entire speaker-adaptive module. Section 6.4 presents the experiment setups and evaluations.

6.1 Limitations of the multimodal neural TTS

6.1.1 The discontinuity of the linguistic latent space

Chapter 4 experiments suggest that even if we train the multimodal system with the *joint-goals* method, which constrains the output, and the *tied-layers* method, which constrains the hidden layers, the continuity of the latent space is only achievable to some extent. In simpler words, a small difference in the latent space can produce vastly different output which also means it is not reliable to use the speech encoder as a substitute of the text encoder. It is because of the scarcity nature of data, as we train a continuous latent space with a limited amount of training examples.

Due to this reason, a key element for a serviceable crossmodal speaker adaptation method, introduced in Chapter 4, is restricting the amount of adaptable parameters. More specifically, even if the speech encoder failed to approximate the text encoder, as long as their latent space is correlated the adaptation method can still function properly. The correlation in this context means that a speaker embedding which improves the designated loss in the STS stack improves the same loss function in the TTS stack.

However, if we are limiting the amount of adaptable parameters, there are very few ways we can improve performance further; using speaker scaling and bias codes, introduced in Chapter 5, is one approach. So to open up more possibilities to improve the performance, we need to improve the continuity of the shared latent space and train a speech encoder that is better at approximating the text encoder.

6.1.2 The restrictive nature of speaker codes

The experiments conducted in Chapter 5 show that by adding a speaker scaling along with the speaker bias, we can create more sophisticated speaker transformations. These results suggests that the key to improve the performance of the speaker adaptation task is increasing the expressive power of the adaptable speaker component, which means we need to enhance the type of transformation and increase the amount of adaptable parameters.

However, doing so makes the adaptation process become more vulnerable to overfitting which is the reason we restrict the amount of adaptable parameters in the first place. In summary, to improve the performance of speaker adaptation task, we need to increase the expressive power of adaptable components while deploying different types of strategies to prevent overfitting without having to reduce the amount of adaptable parameters.

6.2 Variational multimodal neural TTS

We first address the discontinuity problem described in Section 6.1.1 by introducing a modification to the multimodal neural TTS, which is inspired by VAE [110].

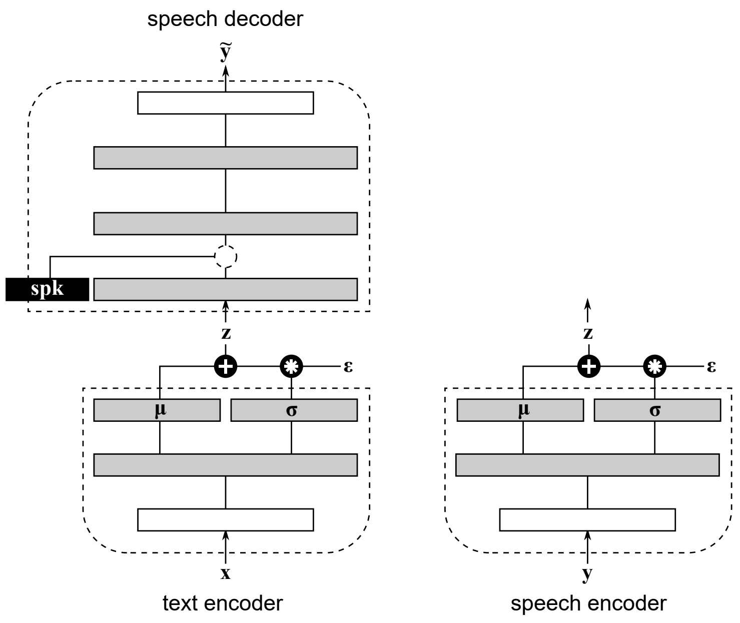


Figure 6.1: The variational multimodal speaker-adaptive speech synthesis system.

The main concept is instead of encoding the input into a single LLE sample (as in Section 4.2), the encoders output a distribution over the latent space.

6.2.1 Variational latent linguistic embedding

The text encoder is modified to output a distribution of $\boldsymbol{z}$ given the input $\boldsymbol{x}$ to reflect the new concept. The speech decoder then consumes a sample $\boldsymbol{z}^T$, which is sampled from such distribution, and transforms it into speech feature:

$$\boldsymbol{z}^T \sim TEnc(\boldsymbol{x}; \phi^T) = p(\boldsymbol{z}|\boldsymbol{x}) \tag{6.1}$$

$$\tilde{\boldsymbol{y}}^T = SDec(\boldsymbol{z}^T; \theta^{\text{S/core}}, \theta^{\text{S/spk},(k)}) \tag{6.2}$$

The same modification is applied to the speech encoder which changes the STS transformation as follows:

$$\boldsymbol{z}^S \sim SEnc(\boldsymbol{y}; \phi^S) = q(\boldsymbol{z}|\boldsymbol{y}) \tag{6.3}$$

$$\tilde{\boldsymbol{y}}^S = SDec(\boldsymbol{z}^S; \theta^{\text{S/core}}, \theta^{\text{S/spk},(k)}) \tag{6.4}$$

The proposed modified network is not trainable with backpropagation as it is, we need to make several practical assumptions to make the training tractable. We assume the linguistic latent spaces have isotropic Gaussian distribution and makes the encoders output mean ($\boldsymbol{\mu}$) and standard deviation ($\boldsymbol{\sigma}$) of such distributions. The reparameterization trick is, then, applied so we could train the network with

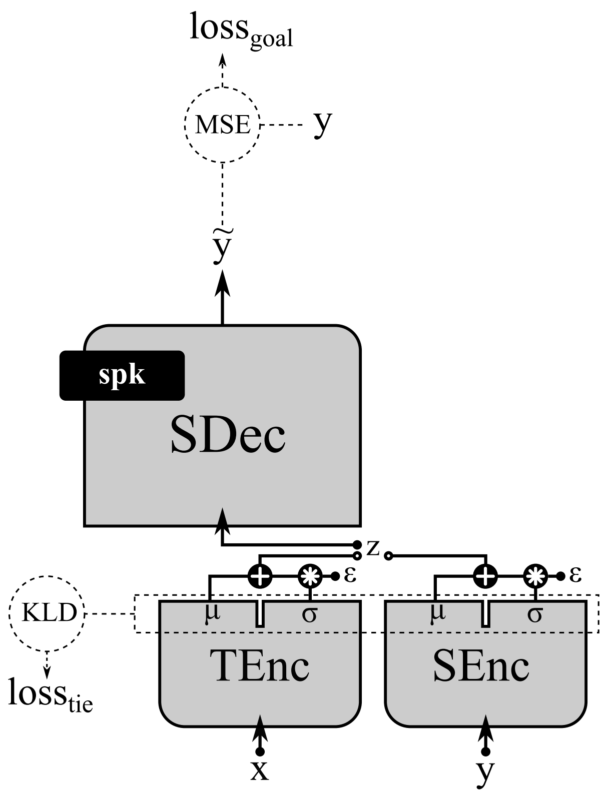


Figure 6.2: The training stage of variational multimodal system.

backpropagation as usual:

$$\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, 1) \tag{6.5}$$

These configurations are not entirely novel but inspired by the typical setup of the variational autoencoder network.

6.2.2 Kullback-Leibler divergence as latent tying loss

Conceptually, there is no difference between the variational multimodal neural TTS and the original model proposed in Chapter 4. Therefore, the same training methods introduced previously (Section 4.3) are usable for the new system. Specifically we can use the *joint-goal* and the *tied-layer* methods to jointly train the entire multimodal network. However, the *tied-layer* method is focused on in this chapter as a special distance function can be used with the variational neural structure, and we want to investigate its performance and behavior in detail:

$$\text{loss}_{train} = \text{loss}_{tts} + \beta \, \text{loss}_{tie} \tag{6.6}$$

Instead of using distortion between hidden layers as loss_{tie}, we can use Kullback-Leibler divergence (KLD) on the encoders' latent distribution outputs as the latent tying loss:

$$\text{loss}_{tie} = L_{KLD}(TEnc(\boldsymbol{x}), SEnc(\boldsymbol{y})) \tag{6.7}$$

The motivation for the setup is identical to the standard multimodal neural TTS proposed in Chapter 4, the text encoder and speech decoder are trained with a typical TTS objective while the speech encoder is trained to approximate the text encoder so it could be used as a substitution. By jointly training the entire multimodal network with the combined loss, we encourage the model to find the optimal representation for all criteria.

6.2.3 Crossmodal adaptation with variational LLE

The adaptation process with variational multimodal neural TTS is also identical to that of the standard multimodal system (Section 4.2.2). Specifically, the STS stack is used to estimate a new set of speaker-dependent parameters, $\theta^{\text{S/spk},(r)}$, for the r-th target unseen speaker using untranscribed speech:

$$\text{loss}_{adapt} = \text{loss}_{sts} \ . \tag{6.8}$$

However we do need to apply the reparameterization trick just like in the training stage. The main hypothesis about variational multimodal neural TTS is that as we expect the shared linguistic latent space to become more consistent thanks to the new modifications, we could increase the amount of adaptable parameters in the speech decoder without having to worry about mismatch between the speech-encoded and text-encoded latent spaces.

6.2.4 Variational autoencoder for shaping latent space

The proposed variation multimodal neural TTS addresses the sparsity of the latent space by replacing the AE-like structure (see Section 4.2.3) with VAE-like structure. The original purpose of VAE network [111] is enabling content generation by organizing the latent space into a simple distribution which could be sampled from. However for speech synthesis, VAE is generally used as an enhanced version of AE with the capacity to learn better latent features. The VAE-based VC system introduced in Section 2.2.3 is one example. The advantages of VAE over AE when used for shaping latent space is that it provides better continuity and completeness (over a designated trivial prior distribution).

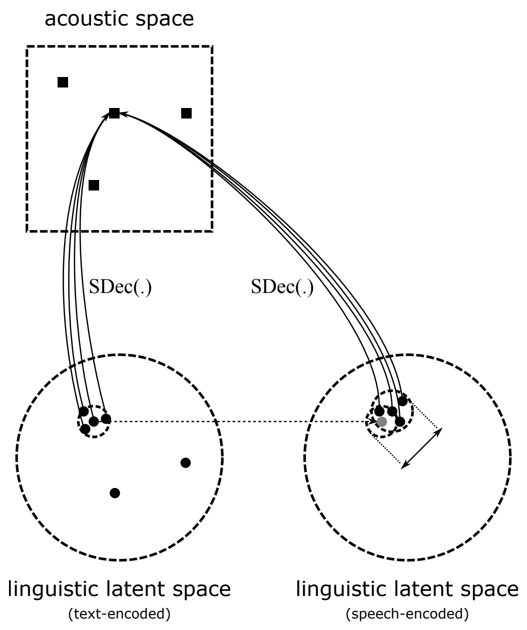


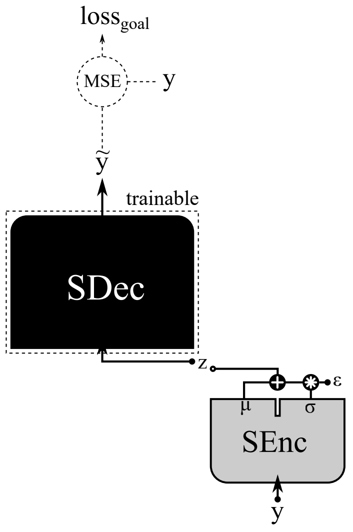
Figure 6.3: Latent and acoustic space of the variational multimodal TTS.

The incorporating of VAE-like structure into the proposed multimodal network is motivated by its ability to enhance the continuity of latent space. The completeness of latent space is not yet a concern in this work, therefore we do not assume a trivial prior distribution over the latent space like regular VAE setup. Intuitively, there are several reasons a VAE-like structure would help create a latent space with better continuity and train a speech encoder better at approximating text encoder. First, the reparameterization trick can be regarded as an artificial samples generation which increases the amount of training examples. Moreover, these artificial latent samples are complied with the continuity policy (latent samples belonged to same distribution will produce the same acoustic feature). Second, by forcing the latent space to have a simple distribution form (e.g. isotropic Gaussian) we can apply the latent tying loss over densities instead of individual samples which helps shape the speech-encoded latent space to approximate the text-encoded latent space.

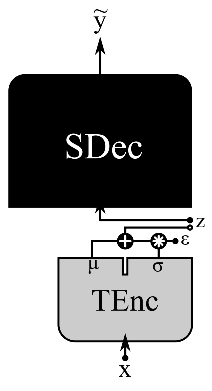
6.3 Adapting by finetuning entire module

6.3.1 Speaker-awareness the initial model

Besides the speaker component, the state of the initial model also affects the performance of the adapted model [30]. More specifically, we can classify the initial model either as speaker-aware or speaker-unaware. A speaker-unaware



(a) Adaptation

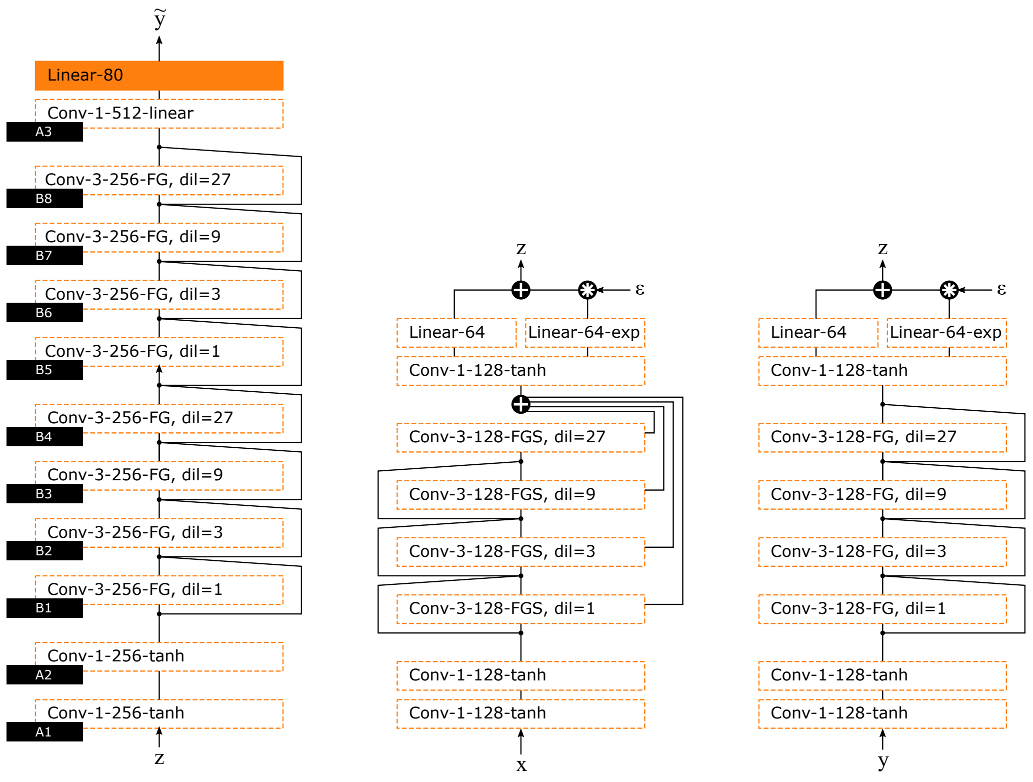


(b) Inference

Figure 6.4: Adapting by finetuning the entire speaker-adaptive speech decoder.

model is trained without information about the speaker. This sort of model includes the conventional SI model of speech recognition and the single-speaker [75, 87] or average-voice model [18, 62] of TTS and VC systems. A speaker-aware model is trained with the speaker components integrated into the initial model. For speech recognition, it is generally known as speaker-adaptive training (SAT) [105]. For speech synthesis, it is the multi-speaker model [80].

The speaker components introduced in Section 5.1 can be used for both speaker-aware and speaker-unaware setups. For example, Fan et al. [28] train a multi-speaker model with a speaker output layer, capable of adapting to an unseen speaker by fine-tuning a new layer for the target. Meanwhile Huang et al. [26] add new layers for unseen speakers on top of a pretrained speaker-unaware single-speaker model. Similar to the LHUC method, Swietojanski et al. [106] proposed to add speaker parameters on top of the SI model for adaptation. By constrast, in a more recent publication, they introduced a speaker adaptive training method for LHUC (SAT-LHUC) [112] which adds LHUC parameters right from the training stage. The speaker awareness or unawareness of the initial model does not change the structure of the adapted model, but changes the representation learned by the hidden layers. Training a speaker-aware model encourages the model to disentangle speaker characteristics (style) from the linguistic information (content) which is expected to improve the adaptation performance [112].



(a) Speech decoder (b) Text encoder (c) Speech encoder

Figure 6.5: Architecture of the variational multimodal neural TTS system using residual filter-gate convolution layers.

6.3.2 Tuning speaker-adaptive layers to target speaker

Recent studies [21, 113] have shown that fine-tuning the entire network along with the speaker embedding is better than fine-tuning just the speaker-embedding. Given a speaker-aware layer with the speaker bias code defined by Equation 3.1, finetuning the entire layer will create a speaker-dependent layer which can be defined as follows:

$$\boldsymbol{h}_1 = f^{(1)}(\boldsymbol{W}_1^{(k)}\boldsymbol{x} + \boldsymbol{c}_1^{(k)} + \boldsymbol{W}_1^{b,(k)}\boldsymbol{s}^{(k)}) \ , \tag{6.9}$$

where all parameters of the layer are now dependent on the target speaker. However it is redundant to have $\boldsymbol{c}_1^{(k)} + \boldsymbol{W}_1^{b,(k)}\boldsymbol{s}^{(k)}$ to model a speaker bias as a single vector $\boldsymbol{c}_1^{(k)} \in \mathbb{R}^{m\times 1}$ can perform the same job.

Based on the above observations, we propose a new adaptation strategy in which we fine-tune entire layers of an initial speaker-adaptive network by first removing all speaker components like speaker scaling $\mathrm{diag}(\boldsymbol{a}_l^{(k)})$ and speaker bias $\boldsymbol{b}_l^{(k)}$. Liu et al. [14] used a similar strategy to adapt a multi-speaker Wavenet vocoder [114] to unseen speakers with limited data. We hypothesize that a

Table 6.1: Speaker adaptation strategies with the variational multimodal systems.

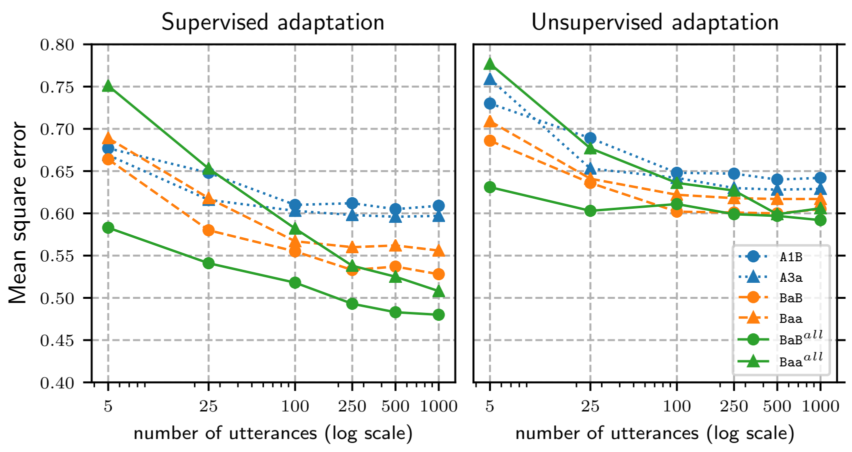
Strategy	Speaker layer(s)	Speaker code scaling	Speaker code bias	Adaptable parameters
`A1B`	A1	-	full	256x1
`A3a`	A3	128	128	256x1
`BaB`	B[1-8]	-	full	512x8
`Baa`	B[1-8]	64	64	256x8
`BaB`^{all}	B[1-8]	-	full	speech decoder
`Baa`^{all}	B[1-8]	64	64	speech decoder

speaker-aware model with all speaker-specific parameters stripped is a good initialization for the speaker adaptation.

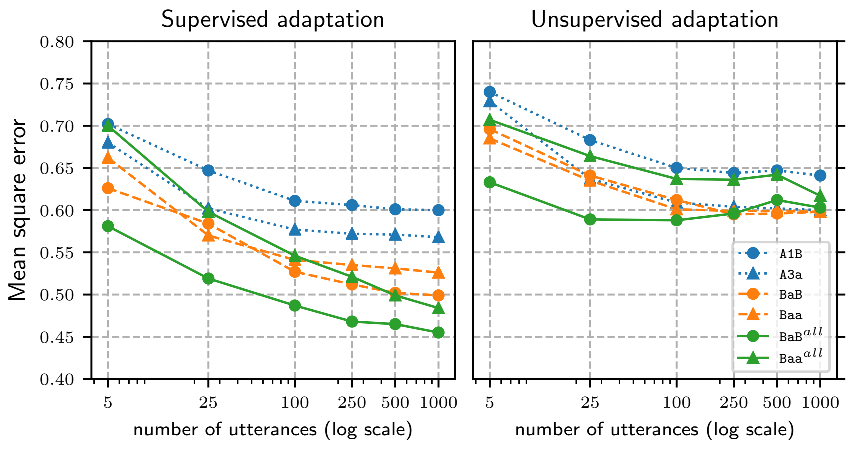
6.4 Experiments

The following experiments test the performances of variational multimodal neural TTS system on supervised and unsupervised speaker adaptation tasks. The medium-sized network with time-domain one-dimensional convolution layer is used extensively in the network as illustrated in Figure 6.5. While the concepts of speaker scaling and speaker bias are defined in the context of a simple feed-forward layer, they can easily be extended to more complex neural layers like recurrent neural network (RNN) or convolution neural network (CNN). Details about the architecture can be found in Appendix C.2. The text representation is aligned Japanese linguistic which listed in Appendix B and speech representation uses mel-spectrograms. By using a single acoustic feature instead of multiple acoustic features and their dynamics, it reduces the complication with speaker transformation which is discussed in Chapter 5.

The initial multi-speaker model is trained with *jvb.medium.base* (Appendix A). Unlike previous experiments which test performance on many target speakers, the following experiments focus on just two Japanese, one male and one female in the *single.ja.base* set, whom have more than one thousand utterances so we could investigate the effect of the amount of data on adaptation performance. Table 6.1 describes the speaker adaptation strategies that will be evaluated in the following experiments. `A1B` and `A3a` are strategies with speaker scaling and bias at a single layer near input or output. `BaB` and `Baa` are strategies with speaker scaling and bias at multiple hidden layers. `BaB`^{all} and `Baa`^{all} use the same initial model as `BaB` and `Baa` but we fine-tune the entire speech decoder after removing all speaker components in the adaptation stage.



(a) Female



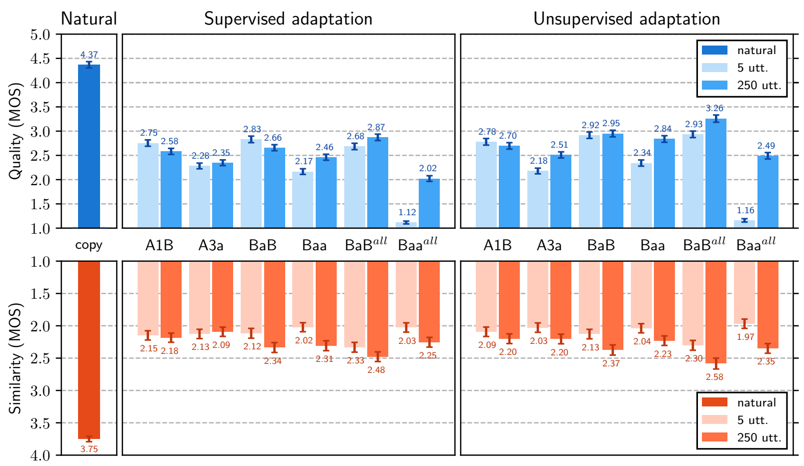
(b) Male

Figure 6.6: Objective evaluations for supervised and unsupervised speaker adaptation tasks of the variational multimodal systems.

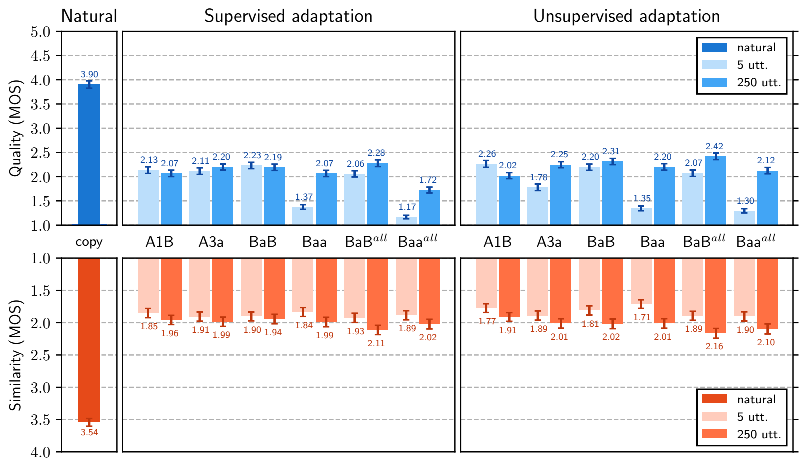
6.4.1 Objective evaluation on adaptation by finetuning

As the mel-spectrogram is used as the acoustic feature, mean square error is utilize for objective metric. Figure 6.6 shows the results of supervised and unsupervised adaptation over different amounts of adaptation data. The number of utterances used for adaptation ranged from 5 to 1000 utterances. We can see that `A3a` is slightly better than `A1b` at most data points, which reassures the benefit of using speaker scaling along with speaker bias. Surprisingly, `BaB`^{all} outperforms all other strategies, while `Baa`^{all} shows poor results when the amount of data is limited. The pattern is consistent between male and female speakers.

For the unsupervised adaptation task, `BaB`^{all} is also the best strategy. However, adding more adaptation data seems to worsen the objective results. There is still a observable gap between the objective performances of the supervised and unsupervised adaptation, but the best proposed strategy `BaB`^{all} surpasses the



(a) Female



(b) Male

Figure 6.7: Subjective evaluations for supervised and unsupervised speaker adaptation tasks of the variational multimodal systems.

baseline `A1B` in both supervised and unsupervised tasks.

Observation 9. *Given a good initial pretrained model, adaptation by finetuning the entire module not only produces the best performance when data is plenty but also does not cause overfitting when data is limited. The tricky part is finding a good initial model for a particular setup.*

6.4.2 Subjective evaluation on adaptation by finetuning

We conducted subjective surveys on the supervised and unsupervised adaptation tasks. A speaker-independent WaveNet vocoder conditioned on mel-spectrogram is used to generate speech. The neural vocoder was trained on *jvb.medium.base* without tuning to target speaker so we can observe the difference in performance of the adapted acoustic model. To reduce the number of systems that the participants had to evaluate, we only used models adapted with 5 and 250 utterances. A copy synthesis system was also included as a reference.

The mean values of the quality and similarity tests are shown in Figure 6.7. Several inter-speaker and inter-strategy trends are: the quality of the male speaker is lower than the female speaker; when more data becomes available the similarity score increases for most strategies, while the quality sometimes decreases; strategies utilizing both speaker scaling and bias got worse results than those utilizing only speaker bias, despite their better objective results; the supervised and unsupervised adaptations strategies gave similar results. Generally speaking, $\texttt{BaB}^{all}$ is the best strategy for both supervised and unsupervised adaptation tasks, especially in the 250-utterance case. The most surprising outcome is that unsupervised adaptation of $\texttt{BaB}^{all}$ outperforms its supervised counterpart, even though the objective results indicated the opposite.

Observation 10. *The crossmodal unsupervised adaptation strategy with variational multimodal neural TTS has better subjective results than its supervised counterpart even though the objective results suggest the opposite. This highlights the limitation of using objective metrics to evaluate generation tasks.*

The experiment results in this chapter have shown that the proposed variational multimodal neural TTS system has a highly consistent linguistic latent space shared between the text encoder and speech encoder. This allows us to fine-tune the entire speech decoder with STS stack then use it in the TTS stack. The performance of this adaptation strategy is improved significantly when using more adaptation data even if the data is untranscribed speech.

Chapter 7

Latent linguistic embedding for any-to-one VC

The previous chapters establish the fundamentals of the proposed voice cloning method for neural TTS systems. Even though the quality of generated speech presented in their experiment sections are not yet comparable to other SOTA systems, we will put the TTS investigation on hold and return to it later in Chapter 8. In this chapter, we move on to the second issue of this thesis which is developing a unified voice cloning system of TTS and VC. Specifically, we investigate the feasibility of using the latent linguistic embedding, obtained from the training process of the multimodal neural TTS, for non-parallel VC systems. As LLE shares many traits with PPG, which are a popular phonetic features used to train non-parallel VC systems (Section 2.2.4), it hints at the potential of using LLE for the same task. Moreover, this chapter also highlights the complementary nature of TTS and VC systems, the ability of VC to convert speech in unseen languages being one example.

Section 7.1 explains the complementary nature of TTS and VC systems and points out the motivation for a unified method. Section 7.2 describes the procedure to create a VC system with a target speaker's voice using LLE obtained from the training of a multimodal neural TTS system. Section 7.3 introduces different cross-lingual scenarios that the proposed VC system can handle. Section 7.4 lays out the experimental setups and results.

7.1 Complementarity of TTS and VC

TTS and VC can be seen as different interfaces for generating speech with a target voice. The difference in the input interface is not trivial but has important implications for their operation scenarios. More specifically, as text input is easy to create and modify, TTS is capable of generating a large amount of speech automatically and cheaply. However it is unable to generate speech when the desired linguistic instructions cannot be represented in the expected written form (e.g. when text is written in foreign languages). On the other hand, speech used as a reference input for VC is more time-consuming and expensive to create, but the system can be straightforwardly extended to an unseen language.

TTS and VC systems with the voice of a particular speaker can be used together to handle a single task. For example, in the case of using speech synthesis for video games, most of the time we would want to use TTS to generate dialog of characters as it is cheaper and easier to edit, however when the conveying content is more complex, expressive or unable to be represented by written text we would want to use VC to generate speech with the same character's voice. In this example we assume that the quality and, more importantly, speaker similarity between TTS and VC is highly consistent. To create TTS and VC systems with consistent performance, it is desirable to use the same method or system for both as each separated module is a chance to cause friction in performance.

7.2 Latent linguistic embedding for VC

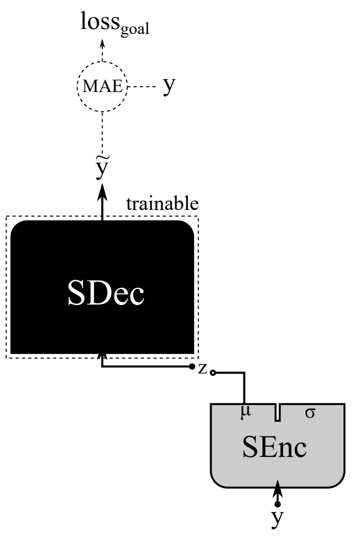
Given the variational multimodal neural TTS introduced in Chapter 6, this section establishes the framework to transfer knowledge learned by TTS, in the form of latent linguistic embedding, to VC and create an any-to-one non-parallel VC system for a target speaker.

7.2.1 Training the multimodal neural TTS

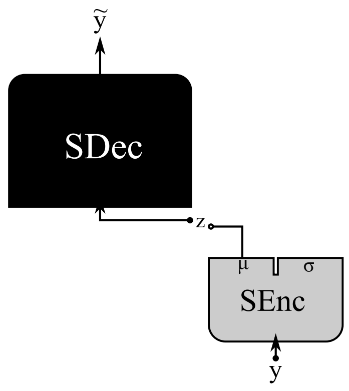
The first step is training a robust multimodal TTS system and the latent linguistic embedding. This step is identical to the training stage defined in Section 6.2 which requires transcribed speech data of multiple speakers:

$$\text{loss}_{train} = \text{loss}_{tts} + \beta\,\text{loss}_{tie}\ , \tag{7.1}$$

where loss_{tts} is the distortion between output of the TTS stack and the natural speech features, and loss_{tie} is the Kullback-Leibler divergence (KLD) between the output of the text encoder and output of the speech encoder. We can use mean-square error (MSE) or mean-absolute error (MAE) as the speech feature distortion function. The previous chapter used MSE while this chapter uses MAE just for



(a) Adaptation



(b) Inference

Figure 7.1: Creating VC system by finetuning speech decoder to target speaker's untranscribed speech data.

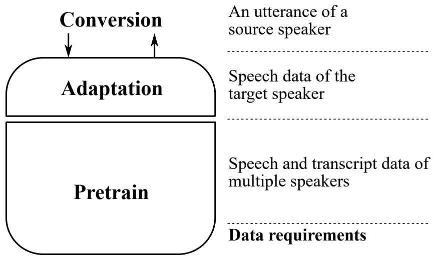
observation. In the training step, the speech encoder is trained to transform acoustic features to the linguistic representation LLE while the acoustic decoder is trained to become a good initial model for speaker adaptation.

7.2.2 Adapting to the target speaker of VC

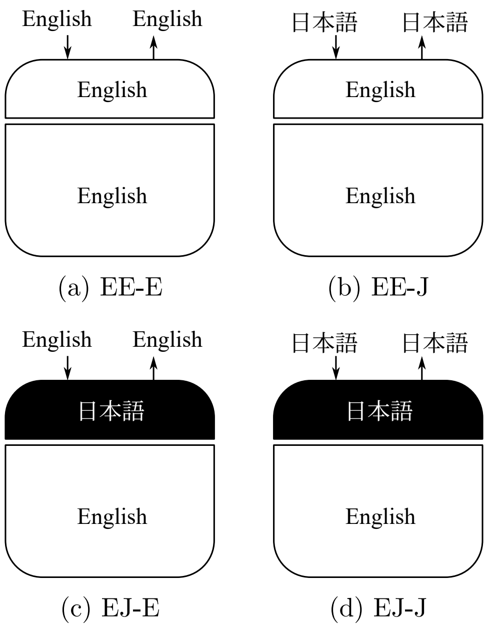
For VC, we only need the speech encoder and the speech decoder trained previously. They make a complete STS stack. To create a VC system with a particular voice, we can clone voices of the targets using the their untranscribed speech data. It is identical to the unsupervised crossmodal speaker adaptation for TTS introduced in the previous chapter. The adaptable parameters of the speech decoder are fine-tuned using the following loss function:

$$\text{loss}_{adapt} = \text{loss}_{sts} \ . \tag{7.2}$$

Specifically, we remove all speaker components $\theta^{\text{S/spk},(r)}$ and then fine-tune remaining parameters $\theta^{\text{S/core},(r)}$ to the r-th target speaker as described in Section 6.3.2. A new tactic, named *mean-value LLE*, is introduced in this step. Basically, instead of sampling $\boldsymbol{z}^S$ from a distribution (Equation 6.4), we assign the mean vaue to it. The motivation is that by removing the stochastic sampling process, the model can learn fine-grain details instead of a generalizing representation. Figure 7.1a illustrates the adaptation process used for VC which is slightly different to its counterpart in the previous chapter (Figure 6.4a).


Conversion
Adaptation
Pretrain
An utterance of a source speaker
Speech data of the target speaker
Speech and transcript data of multiple speakers
Data requirements

The asymmetric nature of data requirements for each step of the procedure for building a LLE-based VC system.



(a) EE-E (b) EE-J (c) EJ-E (d) EJ-J

Figure 7.2: Different scenarios (by examples) of using LLE-based VC system with an unseen language. English plays the role of the abundant language (seen and with transcript) while Japanese plays the role of the low-resource language (unseen and without transcript).

7.2.3 Converting speech of arbitrary source speakers

The adapted speech decoder is used together with the speaker-independent speech encoder as an any-to-one VC system to convert speech of arbitrary source speakers to the voice of the target as illustrated in Figure 7.1b. The system does not need to train on or adapt to the speech data of source speakers which is similar to PPG-based VC system.

7.3 Cross-lingual voice conversion scenarios

The procedure to create a VC system with a target voice described in Section 7.2 has an interesting characteristic which is the asymmetric nature in terms of data requirements for each step. While the TTS training step requires a transcribed multi-speaker corpus, the adaptation only requires a small amount of untranscribed speech from the target speaker. Moreover the model does not need to train on data of source speakers and hypothetically can convert utterances of arbitrary speakers out of the box. This asymmetric nature in terms of data requirements can be taken advantages of to build VC systems for low-resource languages or handle various cross-lingual scenarios. Even though there are many works dealt with cross-lingual VC, the way different types of cross-lingual scenarios are described is imprecise and interchangeable which fails to highlight their distinctions and their practical application. Therefore, we explain these cross-lingual scenarios by using real examples as shown in Figure 7.2. These scenarios are differentiable due to the asymmetric nature of data requirements for creating a VC system with a target voice using our framework.

7.3.1 Standard intra-language scenario

This scenario is represented by the EE-E scenario. We adapt the pretrained English model to an English speaker and use it to convert English utterances. Generally speaking the entire framework from start to finish operates within a single language. It is essentially the typical intra-language scenario of a voice conversion system.

7.3.2 Cross-language voice conversion scenario

This scenario is represented by the EE-J scenario. We adapt the pretrained English model to an English speaker but use it to convert Japanese utterances. Generally speaking, the VC system is used to generate speech from linguistic instructions that it did not see before. The ability to generate speech for content that is difficult to represent in the expected written form (a foreign language in this case) is one advantage of VC over TTS. This scenario is referred to as cross-language voice conversion in several publications [115, 116].

7.3.3 Cross-language speaker adaptation scenario

This scenario is represented by the EJ-E scenario. We adapt the pretrained English model to a Japanese speaker and use it to convert English utterances. In this scenario we want to build a VC system in the abundant language; however, the speech data of the target speaker is only available in a low-resource unseen language. This is sometime referred to as cross-lingual voice conversion [64, 117], however we call it cross-language speaker adaption to distinguish it from EE-J. Unlike other scenarios involving the unseen language, cross-language speaker adaptation is relevant for both VC [64] and TTS [118, 119]. However we will not evaluate cross-language speaker adaptation for TTS systems in this thesis.

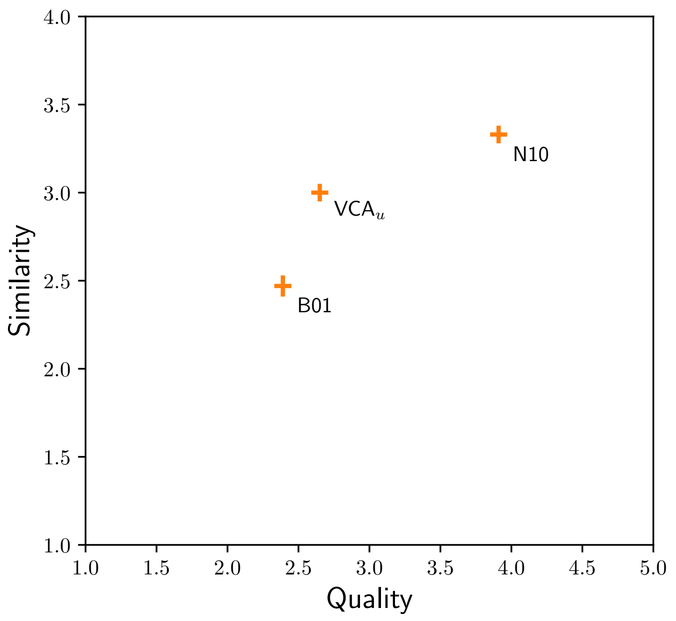


Figure 7.3: Subjective evaluation of the reenactment of Voice Conversion Challenge 2018 SPOKE task for the LLE-based VC system.

7.3.4 Bootstrapping for low-resource language scenario

This scenario is represented by the EJ-J scenario. We adapt the pretrained English model to a Japanese speaker and use it to convert Japanese utterances of unseen source speakers. In this scenario we essentially bootstrap a VC system for a low-resource language from a pretrained model of an abundant one. The written form of the target language is not used in the training, the adaptation or the conversion stages. This scenario is sometime referred as text-to-speech without text, which is the main topic of the Zero Resource Speech Challenge 2019 [120]. Even though our scenario has the same objective as the challenge, the approach is a little different. The participants of the challenge are encouraged to develop intra-language unsupervised unit discovery methods, which are more difficult [121, 122, 123]. Our framework is bootstrapped from an abundant language, which is a more practical approach [124, 125].

7.4 Experiments

The following experiments test the performance of LLE-based VC system on the intra-language scenario as well as the cross-lingual ones. The medium-sized network, same as previous chapter, is used as acoustic model (Appendix C.2). The speaker biases are placed in layers B5, B6, B7 and B8 (see Figure 6.5). The text representation is aligned English linguistic which is listed in Appendix B and

Table 7.1: Detailed same-gender and cross-gender subjective results of the reenactment of VCC2018 SPOKE task.

(a) Quality

	F-F	F-M	M-F	M-M	ALL
N10	3.91	3.96	3.85	3.93	3.91
B01	3.10	2.12	1.84	2.49	2.39
VCA$_u$	2.72	2.77	2.41	2.68	2.65

(b) Similarity

	F-F	F-M	M-F	M-M	ALL
N10	3.18	3.55	3.14	3.48	3.33
B01	2.74	2.87	2.21	2.04	2.47
VCA$_u$	2.92	3.13	2.70	3.26	3.00

speech representation is mel-spectrogram. The WaveNet vocoder is finetuned to the target speaker before it is used to generate the speech waveform.

The initial multi-speaker neural TTS system is trained with *vctk.medium.base* (Appendix A). To test the feasibility of using LLE for non-parallel VC, the target speaker of the Voice Conversion Challenge 2018 (VCC2018) SPOKE task, *vcc2018.spoke.target*, is used for the standard intra-language scenario. For cross-lingual scenarios, two bilingual speakers spoke English and Japanese in the *bilingual.enja.target* dataset are used as the target speakers. Details about the data conditions can be found in Appendix A.

7.4.1 LLE for standard voice conversion

This experiment follows the guidelines of VCC2018 [53] to build systems for the four target speakers of the SPOKE task and converting evaluation utterances of the four source speakers. The proposed LLE-based VC system (VCA$_u$) is compared with B01 [126], which is the baseline of VCC2018, and N10 [14], which is the best system based on the subjective evaluation. The listening test is setup to ask participants to judge the quality and similarity of utterances generated by the VCA$_u$, B01 and N10 systems.

The general results can be seen in Figure 7.3. While it is not as good as the best system N10, our system is slightly better than the baseline B01 in terms of quality and significantly better in terms of speaker similarity. One should note that B01 is a strong baseline, it is ranked 3rd in quality and 6th in speaker similarity at the VCC2018 [53]. This validates our framework for VC. Detailed results can be found in Table 7.1, which shows consistent performance of the LLE-based VC system across same-gender and cross-gender pairs.

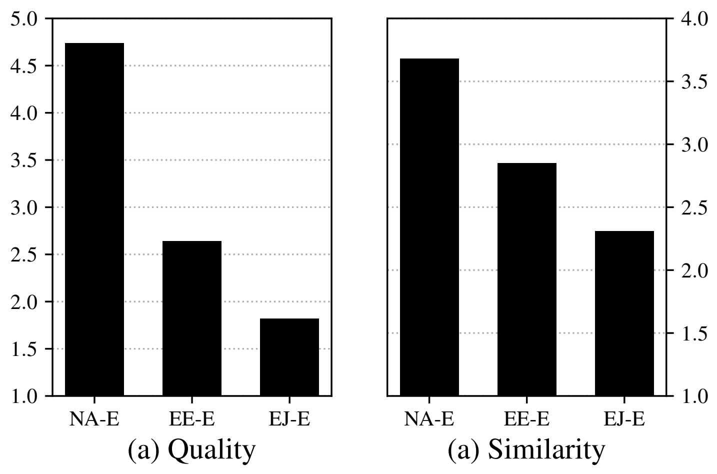


Figure 7.4: Subjective results for cross-language speaker adaptation scenario. All results are statistically significant.

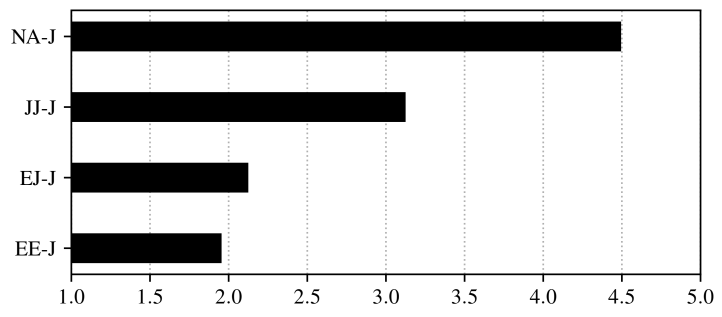
Observation 11. *By following a similar procedure for cloning voices with multi-modal neural TTS, we can create an any-to-one VC system with promising quality and speaker similarity. Moreover, the converting performance is consistent between same-gender and cross-gender scenarios.*

7.4.2 LLE for cross-language speaker adaptation

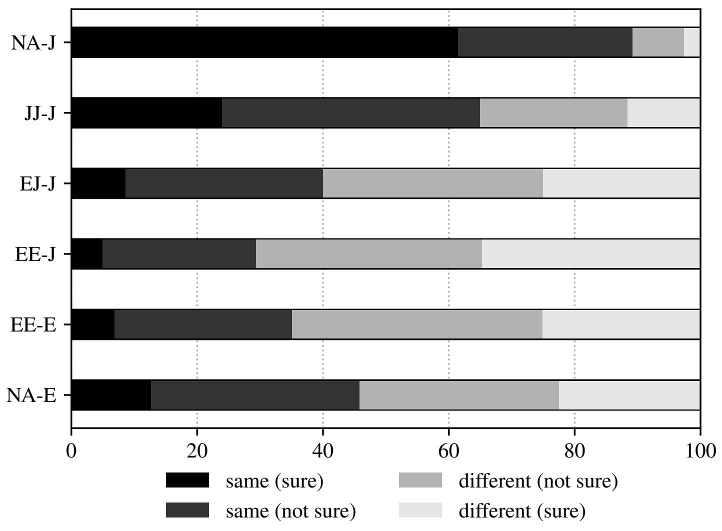
Next, we evaluate our system in the scenario of cross-language speaker adaptation (EJ-E). For this task we developed the VC systems using the speech data of the bilingual target speakers. We compare EJ-E with EE-E, which is the standard intra-language of English, and a reference system NA-E, which is of held-out natural English utterances.

The native English speakers who participated in the survey for the previous task are asked to evaluate the cross-language speaker adaptation task as well. In summary, each scenario is judged 544 times for quality and another 544 times for similarity. The quality and similarity results are illustrated in Figure 7.4 and are statistically significant between all scenarios. The cross-language speaker adaptation EJ-E is generally worse than EE-E as one would expect but the results show the feasibility to use the model for cross-language speaker adaptation.

Observation 12. *The LLE-based VC system can be adapted using speech in a foreign unseen language of a target speaker without having to train on the speech or text of the language. The performance is acceptable which suggests that the linguistic latent space is somewhat universal.*



(a) Quality



(b) Similarity

Figure 7.5: Subjective results for low-resource language scenarios. All results are statistically significant.

7.4.3 LLE for low-resource language voice conversion

In the last experiment, we test the performance of the proposed system on converting speech of an unseen (low-resource) language. This consists of the EE-J and EJ-J scenarios. We use JJ-J, the abundant language scenario of Japanese, as the upper-bound and NA-J, the held-out natural Japanese utterances, as the reference.

The subjective quality evaluation is illustrated in Figure 7.5a. Our standard Japanese system JJ-J achieved a relatively better score than the English counterpart EE-E, although technically they should not be compared directly. The cross-language converting scenarios, EJ-J and EE-J, are a lot worse than the standard scenario JJ-J; as expected EJ-J has a slightly better score than EE-J. The result for the similarity test is illustrated in Figure 7.5b. For the four systems that convert Japanese speech, their similarity result trend is the same as their quality result trend. For the similarity test, we also included two extra systems that produce English speech, EE-E and NA-E. In these two cases the listener would listen to an English utterance contributed by EE-E or NA-E and a natural Japanese utterance of the same speaker (as the target speakers are bilingual) and judge their similarity. Interestingly the similarity result of NA-E is a lot worse than that of NA-J even though they both contain natural speech spoken by the same speaker in real life. This suggests that utterances spoken by the same speaker in different languages might not have consistent characteristics.

Observation 13. *The proposed LLE-based VC system can convert speech of an unseen language without having to train on speech or text of the particular language. Generally speaking, it is able to convert linguistic instructions that it has not been trained on and cannot be represented in the expected written form.*

Even though the performances in the unseen language scenarios are not as good as the standard scenario, the subjective results present a proof-of-concept for using the proposed system in cross-lingual scenarios. The experimental conditions in this chapter are strictly configured to emulate the extremely limited data scenarios of an unseen language. In practice, the performances of cross-lingual scenarios are expected to significantly improve by using multi-lingual corpus which included the target language [64, 127]. This is also an interesting topic that we will evaluate more in the future.

Chapter 8

A versatile voice cloning system of TTS and VC

As previous chapters establish and evaluate different components or aspects of a voice cloning system, the experimental systems are kept simple so the investigation can isolate and focus on behaviors of a particular component. This chapter takes the generalized observations obtained in previous chapters and integrates the proposed method into a SOTA versatile voice cloning framework by considering all major and minor complications of creating one [128].

Section 8.1 introduces relevant components and techniques that are essential for high-quality speech generation systems which are overlooked in previous chapters. Section 8.2 redefines the voice cloning framework as a unified system of TTS and VC. Section 8.3 proposes the standard strategy for cloning voices with untranscribed speech. Section 8.3 introduces an alternative cloning strategy to use when adaptation data is transcribed speech. Section 8.5 introduces two experiment scenarios to test performance of the proposed system by comparing with third-party SOTA TTS and VC systems. Section 8.6 provides further analysis and discussion on the setup and training process of the proposed system.

8.1 Components of modern TTS/VC systems

8.1.1 Jointly tuned neural vocoder

A neural vocoder which generates waveforms sample-by-sample [11, 12, 13] is a staple of many SOTA speech synthesis systems [6, 14]. Unlike the conventional vocoder, a neural vocoder is trained with speech data and backpropagation algorithm in the same manner as the neural acoustic model. A downside of a trained model is that it is more difficult to obtain a universal vocoder but the upside is that it can be integrated deeper with the neural acoustic model through finetuning with backpropagation [129, 130].

8.1.2 Autoregressive speech synthesis

A standard neural acoustic model learns the mapping between a linguistic input and the acoustic output while disregarding the temporal dependence between neighboring acoustic feature frames. A better model should take the temporal correlation between the target acoustic sequences into account. The autoregressive neural acoustic model has shown to significantly improve the naturalness of generated speech [131]. The autoregressive neural acoustic model takes previous generated frames into account for the generation of the current, which allows the network to model more sophisticated and complex functions. However an autoregressive structure also increases the inference time significantly as it generates one sample at a time. Due to this reason, we did not use it in previous experiments.

8.1.3 Data-driven linguistic representation

As discussed in Section 2.4.1, in the conventional TTS system setup, the phonetic information is augmented with engineered language-specific linguistic information. While for E2E system [6] this information is expected to be learned by the model. Watts *et al.* [65] suggest that the learned linguistic context is one aspect that contributes to the quality of E2E system. Moreover it also reduces the burden of expert knowledge required to create hand-crafted relevant linguistic information for a particular language. In previous chapters, we use aligned linguistic as text representation; to boost the performance further we use a neural module to learn linguistic information from a phoneme sequence.

8.2 A versatile voice cloning system

The core concept of our proposal is the latent linguistic embedding (LLE) which is used as a stand-in for text when transcription is difficult to obtain. The architecture of our multimodal system resembles the model proposed by Karita *et al.* [132]; however, they focus on the performance of ASR system instead of speaker adaptation. While the emphasis on linguistic latent features is similar to the PPG-based VC system proposed by Sun *et al.* [58], their phonetic representation extractor is trained independently from the VC model while our linguistic latent features are jointly trained with the speech generation model.

The main components of the proposed voice cloning system are presented in Figure 8.1. The text-speech multimodal network is essential for our methodology, while the neural vocoder is optional, but it is included as it is a necessary for generating high-quality speech [6, 14]. The proposed multimodal system contains four modules, which are encoders and decoders of either text, $\boldsymbol{x}$, or speech, $\boldsymbol{y}$. In combinations of the encoder and decoder, they can perform four transformations: text-to-speech (TTS), speech-to-speech (STS), speech-to-text (STT), and text-to-text (TTT). Combining these modules into a single system is not just for convenience but serves an important purpose. The speech encoder helps the TTS system adapt with untranscribed speech as discussed in Chapter 6 while the text encoder helps the VC system disentangle linguistic representation from speaker characteristics as discussed in Chapter 7. The text decoder is the new addition in this chapter. While Karita *et al.* [132] use a similar combination for speech recognition, we focus on speech generation tasks and the text decoder is used exclusively as an auxiliary regularizer in the training stage.

8.2.1 Training the text-speech multimodal system

Our methodology is designed around the training of a speaker-disentangled LLE, $\boldsymbol{z}$. The LLE in our setup plays the same role as the PPG proposed for VC [57]. However, the LLE is jointly trained with the speech generation modules and contains linguistic information as a whole (instead of phoneme). There are several way to train a multimodal neural network. The modalities of such system can be trained stochastically [97], step-by-step [95], or jointly together [93, 132] as discussed in Section 4.3. To jointly train a multimodal system using either *joint-goal* or *tied-layer* method is enough [93, 22] but as they are complementary we can use them together:

$$\begin{aligned} \text{loss}_{train} &= \text{loss}_{goals} + \beta \, \text{loss}_{tie} \\ &= \text{loss}_{tts} + \alpha_{sts} \, \text{loss}_{sts} + \alpha_{stt} \, \text{loss}_{stt} \\ &\quad + \beta \, \text{loss}_{tie} \ , \end{aligned} \tag{8.1}$$

where the loss_{tts} in Equation 8.1 is a TTS loss defined by the text encoder and speech decoder and is used as the anchor to adjust other hyperparameters. loss_{sts} is an STS loss defined by the speech encoder and speech decoder and we de-emphasize loss_{sts} with a weighting parameter, α_{sts}. loss_{stt} is an STT loss defined by the speech encoder and text decoder. Even though the speech-to-text task

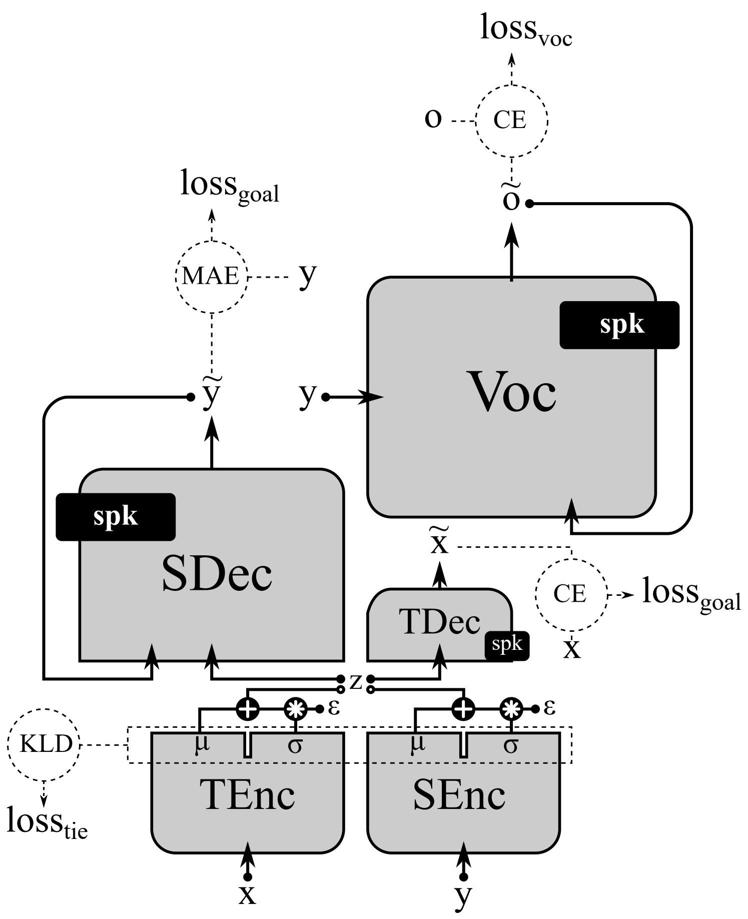


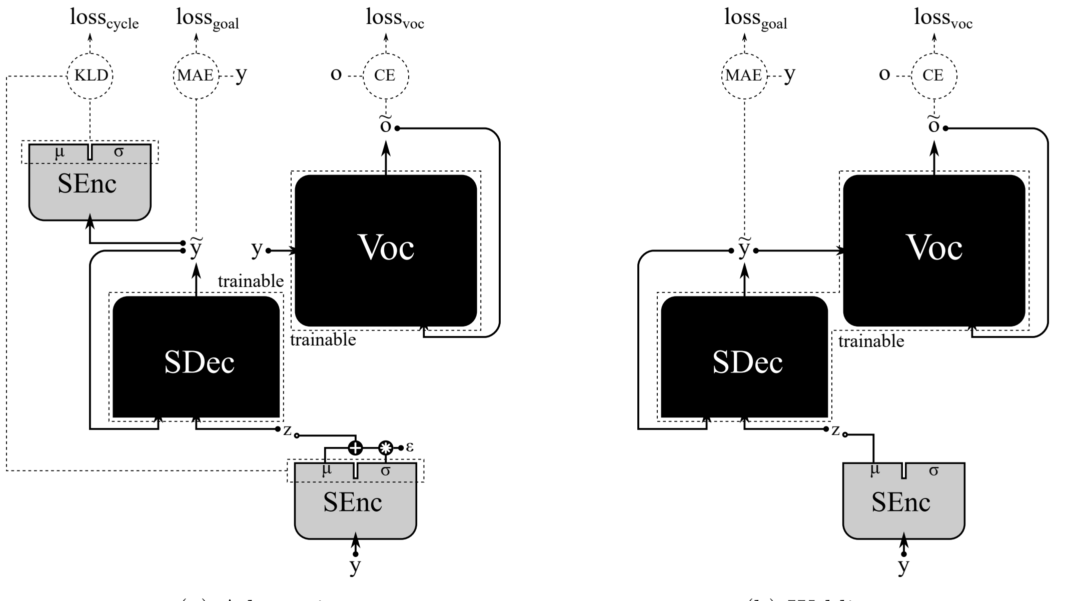
Figure 8.1: The multimodal speaker-adaptive speech synthesis system.

is not a target one, its loss is also included to encourage the latent space to focus more on phonemes (but not entirely). Some other works have shown that an auxiliary phoneme classifier helps in boosting the quality of speech generation systems in general [10]. A TTT loss defined by the text encoder and text decoder, loss_{ttt}, is not included as we do not think that it helps. The last term loss_{tie} is for the tied-layer constraint. In each training step, we calculate each term of the loss_{train} using a transcribed sample then process to optimize all parameters in a supervised manner. Our system can also benefits from semi-supervised strategy [132], but we only focus on supervised training in this work.

For the tied-layers loss, we calculated the symmetric Kullback-Leibler divergence between the output of the text and speech encoders instead of asymmetric one as in previous chapters:

$$\text{loss}_{tie} = \frac{1}{2}\, L_{KLD}(TEnc(\boldsymbol{x}), SEnc(\boldsymbol{y})) + \frac{1}{2}\, L_{KLD}(SEnc(\boldsymbol{y}), TEnc(\boldsymbol{x})) \qquad (8.2)$$

The setup is decided so that we could obtain a consistent latent space between



(a) Adaptation (b) Welding

Figure 8.2: The standard strategy for cloning voices using untranscribed speech.

the text and speech encoders. Another important aspect is random sampling at the output of the encoders. Thanks to the noise added by the sampling process of the LLE in the training stage, the text and speech decoders are trained in a denoising fashion. This, in turn, makes the speech decoder robust to unseen samples, which is helpful for speaker adaptation.

To push the speech generation system toward an E2E setup, we include a neural vocoder to generate a waveform from the acoustic representation instead of using a conventional vocoder. In the training stage, the neural vocoder is trained separately from the rest of the system on natural speech samples:

$$\text{loss}'_{train} = \text{loss}_{voc} \tag{8.3}$$

We used an autoregressive WaveNet [11] conditioned on mel-spectrogram and trained on multi-speaker corpus as the neural vocoder in this paper. However, our voice cloning procedure is applicable to any type of neural vocoder.

8.3 Cloning voices with untranscribed speech

8.3.1 Adapting to target speaker

We first remove all speaker components and then fine-tune remaining parameters of the speech decoder using the following loss:

$$\text{loss}_{adapt} = \text{loss}_{sts} + \beta \, \text{loss}_{cycle} \tag{8.4}$$

The speech distortion loss_{sts} by itself is enough for the adaptation [22], but we still add a linguistic cycle consistent term loss_{cycle} to try improve the performance. The loss_{cycle} is KL-divergence between LLE distribution of natural speech and reconstructed speech:

$$\text{loss}_{cycle} = \frac{1}{2}\, L_{KLD}(SEnc(\boldsymbol{y}), SEnc(\tilde{\boldsymbol{y}})) + \frac{1}{2}\, L_{KLD}(SEnc(\tilde{\boldsymbol{y}}), SEnc(\boldsymbol{y})) \tag{8.5}$$

Even though both loss_{sts} and loss_{cycle} are tried to force the reconstructed feature to be closed to the natural, they focus on different aspects: loss_{sts} is either l_1 or l_2 frame-based hard distortion of the acoustic features, while loss_{cycle} focuses on the linguistic content with soft divergence. We adapt the neural vocoder in a similar manner using its goal loss:

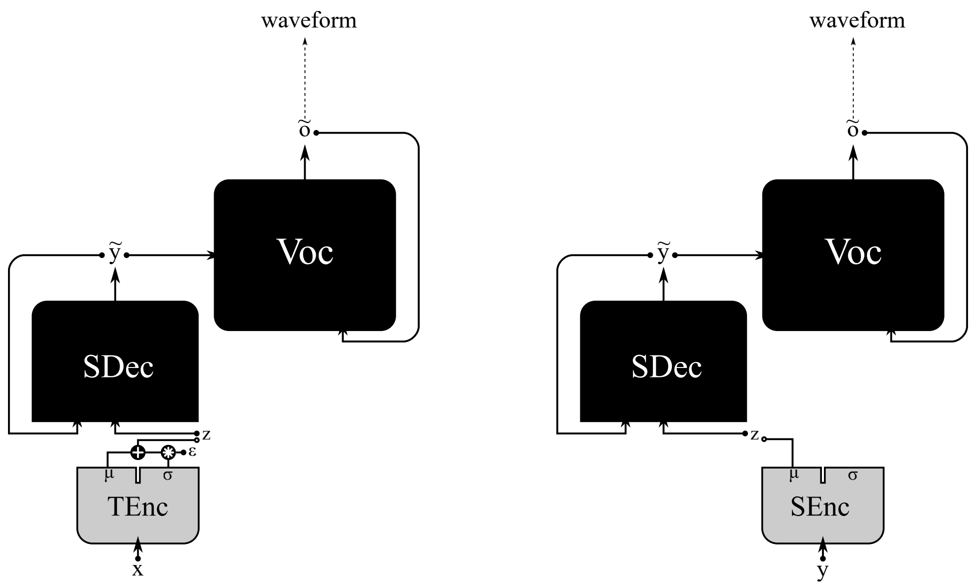
$$\text{loss}_{adapt} = \text{loss}_{voc} \tag{8.6}$$

As the neural vocoder is only dependent on speech, it can be used in an unsupervised adaptation strategy. This is a simple yet effective approach [14].

8.3.2 Welding the speech decoder and the neural vocoder

Even though tuning the acoustic model and the neural vocoder separately produce sufficient quality [14], there are still mismatches between the generated features and the natural features used to train the vocoder. For text-to-speech systems, Zhao et al. [129] fine-tuned the acoustic model with the losses propagated from neural vocoder; while Ping et al. [25] jointly train them together. For voice conversion, due to the duration mismatch between source and target utterances, Huang et al. proposed to fine-tune WaveNet vocoder using reconstructed acoustic features of the target [130].

Motivated by them, we deploy a "welding" strategy, illustrated in Figure 8.2b, that conducts fine-tuning by using the reconstructed features of the target speaker in a similar way to Huang's approach [130], but, for both the speech decoder and



(a) TTS inference (b) VC inference

Figure 8.3: TTS and VC inferences using adapted voice cloning system.

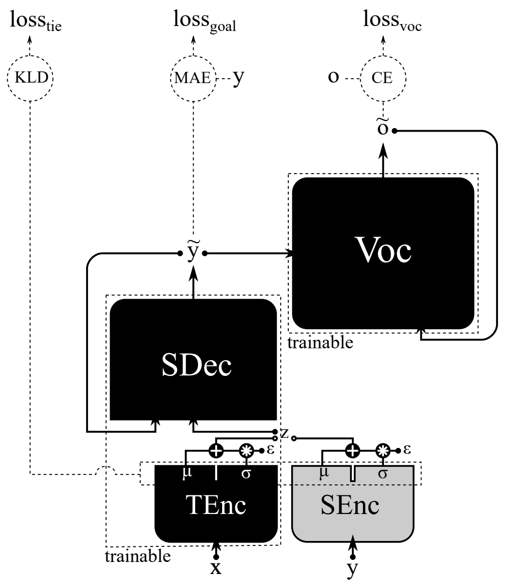
neural vocoder like Ping's method [25] based on the loss function below:

$$\text{loss}_{weld} = \text{loss}_{sts} + \gamma\ \text{loss}_{voc} \tag{8.7}$$

where loss_{sts} is included to preserve the acoustic space even after the welding process as the speech decoder is assumed to be autoregressive in the domain. Two practical tactics are further introduced for this step. 1) *mean-value LLE*: to let the acoustic model learn fine-grained details, we remove the sampling process from the speech encoder and use the mean value instead. 2) *mix-in*: as losses propagating from the neural vocoder can overpower the speech decoder [129], we propose a mix-in tactic, inspired by drop-out, to ease this problem. Specifically the output of the speech decoder is randomly mixed with natural frames by a percentage to reduce the amount of losses propagated back.

8.3.3 Generating speech with TTS/VC stacks

Even though we use the speech encoder to tune the speech decoder and neural vocoder in the adaption and welding steps, the text encoder can utilize these tuned modules without any further adjustment in inference (See Figure 8.3) thanks to the consistency between the latent spaces of the text and speech encoders. As our cloning method tunes entire modules, the more data available, the better the performance.



(a) Adaptation (supervised alternative)

Figure 8.4: The alternative strategy for cloning voices using transcribed speech.

8.4 Cloning voices with transcribed speech

Instead of using exactly the same setup as the unsupervised strategy, for the supervised strategy we first tune the speech decoder and text encoder together using the transcribed speech since using transcriptions is expected to be beneficial for the TTS system.

Step 1 - Adapting (supervised alternative): The supervised strategy for the adapting step is illustrated in Figure 8.4a. We adapt both the speech decoder and text encoder using the following function.

$$\text{loss}_{adapt} = \text{loss}_{tts} + \alpha\ \text{loss}_{sts} + \beta\ \text{loss}_{tie} \tag{8.8}$$

The optimizing loss is similar to that used in the training stage (Equation 8.1). We use loss_{sts} and loss_{tie} to maintain the linguistic latent space for VC. The *welding* and *inference* steps are the same as the unsupervised strategy.

8.5 Experiments

The end goal of voice cloning systems is generating speech with high quality and similarity to the voice of a target speaker when we do not have control over their data situation. However we do have more control over the data used to train the initial model. Therefore, to evaluate performance of voice cloning systems we designed two specific experiment scenarios which center around the target

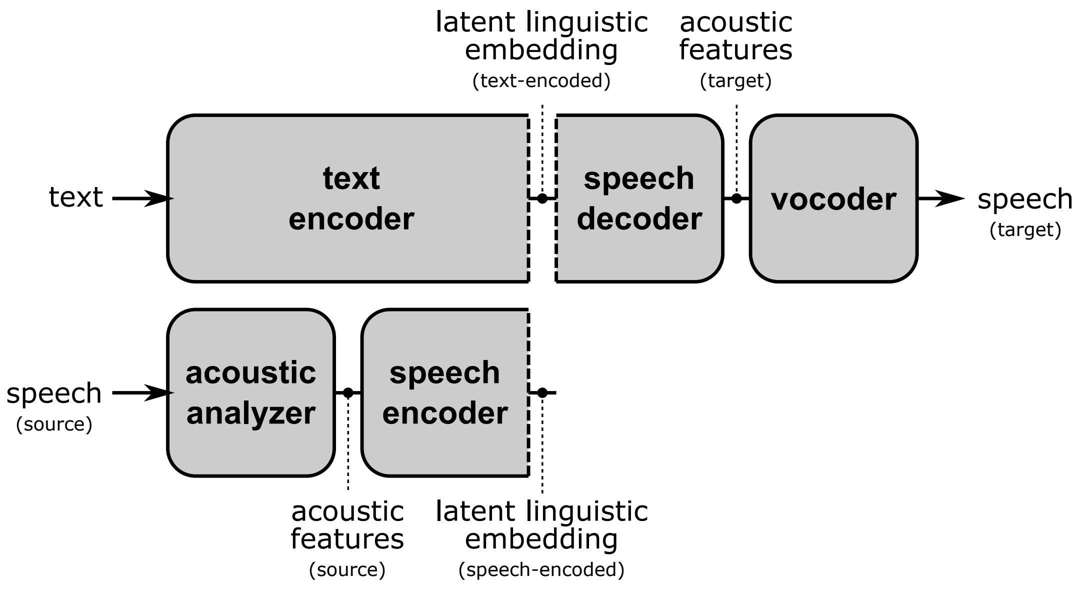


Figure 8.5: A conceptual representation of TTS/VC pipeline of NAUTILUS.

speakers. The first scenario focuses more on VC and cloning voices with untranscribed speech, while the second scenario focuses more on TTS and performance of the supervised and unsupervised speaker adaptation strategies. Unlike previous chapters which evaluate individual techniques, these scenarios treat the proposed system as a whole and compare it with third-party SOTA TTS and VC systems. Our system can seamlessly switch between the TTS and VC modes with several modules shared between the two as illustrated in Figure 8.5.

The particular realization used in the following experiments is called NAUTILUS whose details are given in Appendix C.3. It incorporates several modern components of speech synthesis systems likes autoregressive and learned linguistic representation to improve the naturalness of generated speech as discussed in Section 8.1. The signature component of E2E system, which is the integrated alignment/duration model (see Section 2.4.2), is not used in the current setup to maintain absolute control over duration of generated speech which is convenient for creating a matching condition between generated speech of the TTS and VC systems. The text-speech multimodal system is first trained with *libritts.clean460.base* to take advantage of the diverse linguistic contents, then trained on the *vctk.big.base*, with a particular sampling rate dependant on the target speaker's data, to take advantage of high-quality studio-recorded speech.

Table 8.1: Target speakers of scenario A.

Speaker	Database	Gender	Accent	Quantity	Duration
VCC2TF1	VCC2018	female	American	81 utt.	5.2 min
VCC2TF2	VCC2018	female	American	81 utt.	5.0 min
VCC2TM1	VCC2018	male	American	81 utt.	5.2 min
VCC2TM2	VCC2018	male	American	81 utt.	5.3 min

8.5.1 Cloning voices with untranscribed speech

In the first scenario, scenario A, we tested the ability to clone voices by using a small amount of untranscribed speech (about five minutes). A system showing good performance under this scenario is expected to have the capability to clone thousands of voices efficiently and cheaply. We also use this scenario to test the consistence between performances of TTS and VC modes of the proposed system.

Scenario description

We re-enacted the SPOKE task of Voice Conversion Challenge 2018 (VCC2018) [53] for this scenario. The original goal of the task was to build VC systems for four target English speakers (two males and two females) using 81 utterances (Table 8.1). These systems were used to convert the speech of four source speakers (two males and two females) into each of the target voices. We followed the VCC2018 guidelines [53] faithfully with one extension – we evaluated TTS systems and VC systems at the same time. These TTS systems were required to train on the untranscribed speech of the target speakers. In the inference stage, transcriptions of source utterances were used to generate speech with TTS systems. As there were only 35 unique sentences, we generated each sentence twice. In summary, each TTS system produced 70 utterances for each target speaker while each VC system produced 140 utterances. We split each VC system into two entities, one for same-gender conversion denoted by the superscript "=" and the other for cross-gender denoted by "×".

Evaluated systems

We evaluated the following TTS and VC systems in scenario A:

- **XV**: a speaker-adaptive E2E TTS system using the x-vector [15, 19, 36]. It was used as the third-party unsupervised TTS baseline. We used the *libritts.tacotron2.v1* model and the speaker-independent WaveNet vocoder *libritts.wavenet.mol.v1* which were trained on the LibriTTS corpus to realize this approach. Both are available at the ESPnet [133] repository[1]. As the x-vector is utterance-based, we randomly picked five utterances (about ten seconds) from the training pool of target speakers to extract the x-vector each time we generated an utterance. It is a realization of the unsupervised adaptation method with speech-encoded speaker embedding discussed in Section 2.1.4.

- **N10**: the winner of the VCC2018 SPOKE task. N10 contains a PPG-based acoustic model [58] and a fine-tuned WaveNet vocoder [14]. It uses a speaker-independent ASR model trained on hundreds of hours of labeled data to extract PPG from speech. N10 clones voice without using the speech data of source speakers. This is a SOTA VC system based on PPG which is discussed in Section 2.2.4.

- **N13\N17** (**NR**): the runners up of the VCC2018 SPOKE task in terms of quality and similarity, respectively. To reduce the amount of systems, we treat them as one (denotes as NR) and use N13 in the quality evaluation while using N17 [134] in the similarity evaluation.

- **VCA$_\mathrm{u}$**: VC mode of the NAUTILUS system which was adapted to target speaker by using the unsupervised strategy described in Section 8.3. The letter "*A*" as in "*any-to-one*" indicates that the model is not trained on source speakers. The word *unsupervised* means that the cloning is performed with untranscribed speech in the context of our current work. It is operated at 22.05 kHz to be compatible with the target speakers.

- **TTS$_\mathrm{u}$**: TTS mode of the NAUTILUS system which was adapted by using the unsupervised strategy. As we did not train an automatic duration model, we used the duration extracted from the same-gender source speakers to generate speech from text. This means that TTS$_\mathrm{u}$ shares the same duration model as VCA$_\mathrm{u}^{=}$ (and other same-gender VC systems). This reduces the difference in experimental conditions between them and allows us to make more insightful observations.

- **T00** and **S00**: natural utterances of the target and source speakers used as references, respectively.

[1]https://github.com/espnet/espnet

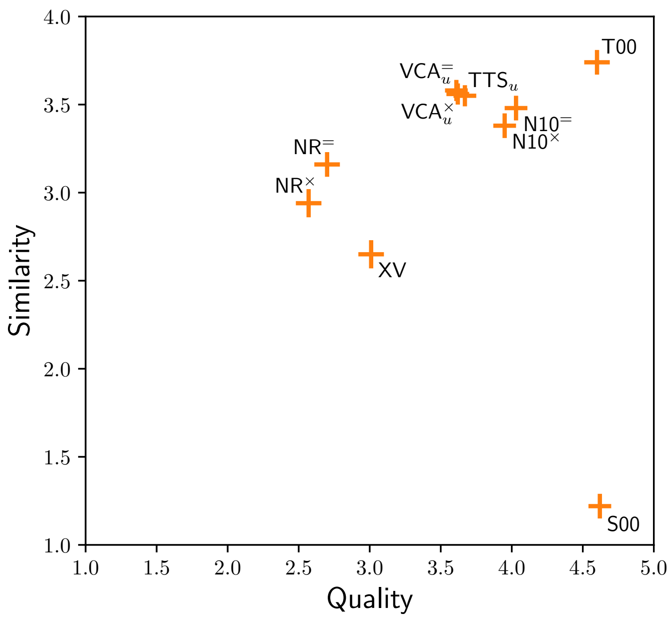
Table 8.2: The word error rate for objective evaluation of scenario A.

System	Target speakers (%WER)			
	VCC2TF1	VCC2TF2	VCC2TM1	VCC2TM2
XV	3.25	2.98	3.66	10.57
$N10^{=}$	9.21	7.99	11.79	9.89
$N10^{\times}$	9.62	11.52	8.67	9.21
$N13^{=}$	23.31	21.68	31.57	27.37
$N13^{\times}$	32.25	24.80	21.41	26.96
$N17^{=}$	25.47	24.39	33.47	23.71
$N17^{\times}$	38.08	31.44	35.23	25.88
$VCA_u^{=}$	25.34	26.02	27.37	25.75
$VCA_u^{\times}$	30.62	27.51	23.71	22.63
TTS_u	7.72	8.40	6.23	7.18
	Source speakers (%WER)			
	VCC2SF3	VCC2SF4	VCC2SM3	VCC2SM4
S00	5.69	4.88	5.69	7.32

Scenario evaluation

Twenty-eight native English speakers participated in the subjective test for scenario A. Listeners were asked to answer 18 quality and 22 similarity questions in each session. In summary, each system was judged 560 times for each measurement, while natural speech systems (T00 and S00) were judged 280 times. The objective and subjective evaluation results are shown in Table 8.2 and Figure 8.6 with many interesting observations. a) XV had better quality but worse similarity than the runners-up of VCC2018, while it received the lowest WER for certain speakers. One possible explanation is that the utterances generated by XV had the characteristics of the speakers in LibriTTS corpus instead of those of the target speakers, which makes its utterances more compatible with ASR model trained on LibriSpeech. The subjective evaluation of the XV speech samples supports this speculation. b) Our systems had high scores in both subjective measurements. Interestingly our TTS system has lower WER than our VC systems. c) Even though we had a lower score for quality than N10, the similarity seems to be higher. d) Our TTS and VC systems had highly consistent results, while there was a gap between the same-gender and cross-gender N10 subsystems. This was perpetuated by extra similarity evaluations between the generated systems presented in Figure 8.6. The similarity between our TTS_u and $VCA_u^{=}$ systems was higher than that of between TTS_u and $N10^{=}$.

Comparing with the similar experiment in Section 7.4.1, the performance of the NAUTILUS has been significantly improved over the medium-sized neural



Extra similarity evaluations
TTS$_u$ / VCA$_u^=$: mean=3.68, lower=3.63, upper=3.73
TTS$_u$ / N10$^=$: mean=3.41, lower=3.35, upper=3.48

Figure 8.6: The subjective evaluations of scenario A.

acoustic network setup thanks to the newly added components.

Scenario conclusion

Even though the naturalness of our voice cloning system was slightly worse than N10 (again the best system at VCC2018), generally speaking it has achieved performance that is comparable to SOTA systems considering the difference in experimental conditions (e.g., the amount of data used in the training stage). More importantly, our system can seamlessly switch between TTS and VC modes with high consistency in terms of speaker characteristics. This is a desirable trait that would be useful for many applications.

8.5.2 Capturing unique speaker characteristics

As mentioned earlier, the way voice cloning is differentiated from speech synthesis is that it should prioritize capturing the unique characteristics of target speakers. While it is easy for listeners to grasp general global characteristics (e.g., average pitch), it is more difficult to notice local subtle traits (e.g., pronunciation of particular words) with just a single reference utterance. We could use famous individuals as targets [17], but this assumes that listeners would be familiar with them. In scenario B, we therefore used non-native speakers as targets to highlight

Table 8.3: Target speakers of scenario B.

Speaker	Database	Gender	Accent/L1	Quantity	Duration
p294	VCTK	female	American	325 utt.	11.2 min
p345	VCTK	male	American	325 utt.	11.0 min
MF6	EMIME	female	Mandarin	145 utt.	10.2 min
MM6	EMIME	male	Mandarin	145 utt.	11.3 min

their unique characteristics. This is convenient for subjective evaluation as native speakers can generally spot their distinctiveness without any explanation about the linguistic aspect of it [135]. In simple words, the goal of scenario B was to reproduce the accent of non-native speakers. This scenario is closely related to reducing accents [136, 137] or controlling accents [16] tasks.

Scenario description

The target speakers for this scenario included two American and two non-native English speakers who use Mandarin as their native language. Each speaker had about 10 minutes of speech as listed in Table 8.3. As the base model was trained with native speakers of English, the speakers from the VCTK corpus represented the standard easy task while the speakers from the EMIME corpus [138] represented difficult and unique target speakers. The evaluated systems were required to be built with either the transcribed or untranscribed speech of the targets. Twenty common sentences from the VCTK corpus were used for the evaluations. Each sentence was generated twice by each TTS system, which totaled 40 utterances. In the case of VC, one female (p299) and one male (p311) with a general American accent included in the training pool are used as source speakers.

Evaluated systems

The following TTS and VC systems were used for the evaluation in scenario B:

- **XV**: the same x-vector system in scenario A is reused as the unsupervised baseline of TTS.

- **FT**: a fine-tuned E2E TTS system was used as the supervised baseline. We used *ljspeech.tacotron2.v3*, implemented with ESPnet [139], as the initial model. It was trained with 24 hours of the transcribed speech of a female speaker from the LJSpeech corpus [140]. An initial WaveNet vocoder was also trained with the same corpus. When cloning voices, we fine-tuned both acoustic and vocoder models with the transcribed speech of the targets. This system represented a simple supervised approach by fine-tuning

Table 8.4: The word error rate for objective evaluation of scenario B.

System	Target speakers (%WER)			
	VCTK-p294	VCTK-p345	EMIME-MF6	EMIME-MM6
NAT*	6.09	8.69	56.24	43.39
XV	3.50	24.05	5.33	3.81
FT	13.39	20.09	57.53	42.01
VCM$_u$	22.22	24.05	27.70	27.09
VCM$_s$	23.29	24.81	29.07	29.53
TTS$_u$	8.37	9.74	13.39	14.92
TTS$_s$	9.28	10.05	36.38	38.20
	Source speakers (%WER)			
	VCTK-p299	VCTK-p311	-	-
SRC**	5.64	6.51	-	-

*calculated on all training utterances of target speakers.
**calculated on natural utterances of source speakers.

a well-trained single speaker model [31]. It is a realization of the supervised speaker adaptation discussed in Section 3.2.1.

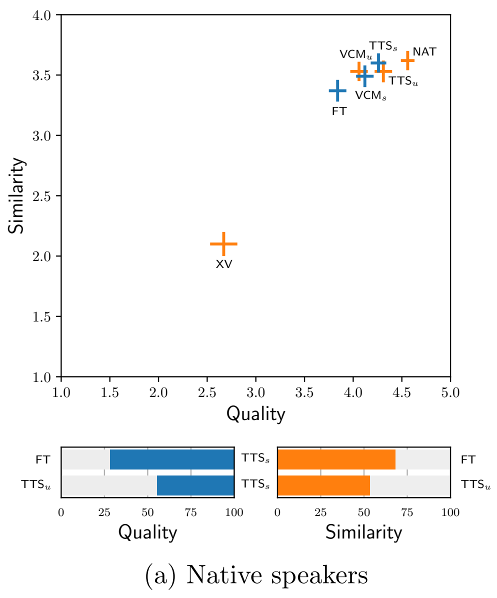
- **VCM$_u$**: VC mode of the NAUTILUS system which was adapted to target speaker by using the unsupervised strategy described in Section 8.3 using untranscribed speech. The letter "M" as in "many-to-one" indicates that the source speakers were included in the training pool of the base model. The system was operated in 24kHz.

- **VCM$_s$**: VC mode of the NAUTILUS system which was adapted to target speaker by using the alternative supervised strategy described in Section 8.4 using transcribed speech. The supervised strategy is more relevant to TTS, but we still included its VC counterpart.

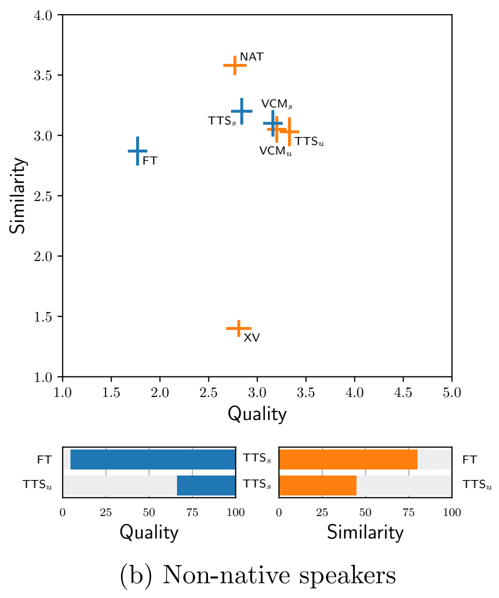
- **TTS$_u$**: TTS mode of the NAUTILUS system which was adapted by using the unsupervised strategy. The duration is extracted from the source speakers of VC. This means our TTS and VC systems share the same duration model.

- **TTS$_s$**: TTS mode of the NAUTILUS system which was adapted by using the alternative supervised strategy.

- **NAT**: the natural utterances of the target speakers.



(a) Native speakers



(b) Non-native speakers

Figure 8.7: Subjective evaluations of scenario B. The lines indicate 95% confidence interval.

Scenario evaluation

Thirty-two native speakers took part in our subjective evaluation for scenario B. As the participants were native English speakers living in Japan and many work as English teachers, we expected that they could quickly pick up on the non-native accents. Each session had 18 quality and 18 similarity questions that contain utterances of both native and non-native speakers. Besides the standard MOS tests, we also included several AB tests in this scenario. In summary, each system was evaluated 640 times for each assessment. The objective evaluation result are listed in Table 8.3, and the subjective evaluation results are shown in Figure 8.7. Here the results of native and non-native speakers are shown separately.

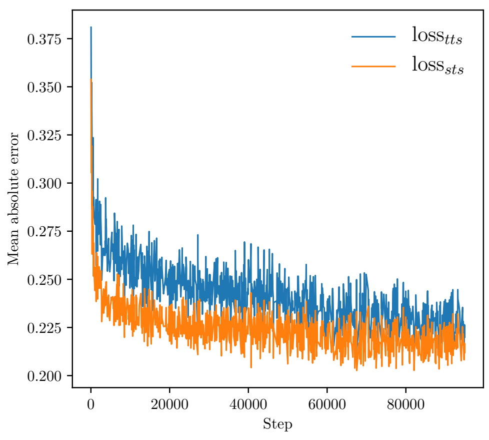
For the standard case with native target speakers, the subjective results show high MOS scores for most systems as shown in Figure 8.7a. The new results here are comparisons between supervised and unsupervised approaches. Comparing the XV and FT systems, which represent unsupervised and supervised TTS baselines, we see that the fine-tuned one was significantly better than the speaker embedding one as it benefited from all ten minutes of data. Similar to scenario A, XV system has better WER than FT for many targets. Among our systems, the difference between the supervised and unsupervised strategies was marginal, but they were all better than the supervised baseline FT. One hypothesis is that our approaches are less sensitive to overfitting thanks to the multi-speaker corpus, speaker factorization and denoising training while FT has a higher possibility of overfitting when using ten minutes of speech [31, 95]. These observations are also supported by AB-preference tests (See the bottom part of Figure 8.7a).

For the challenging case with non-native target speakers, the subjective results revealed more interesting tendencies as shown in Figure 8.7b. This scenario not only revealed the robustness of the voice cloning methods but also the listeners' behaviors. First, we can see that our systems had higher similarity scores than the TTS baselines, FT and XV. The differences between our supervised and unsupervised strategies was more profound in the non-native cases. TTS$_s$ seemed to have higher similarity than TTS$_u$. Next, we see that the natural speech of the non-native speakers (NAT) had lower quality scores than its native counterpart. This would be because our native listeners perceived the "quality" of speech with strong non-native accents as low. As a result, the quality and similarity measurements in this case was no more a positive correlation. Even a negative correlation was found for the subjective results of the TTS baselines, FT and XV, indicating that higher-quality speech corresponded to less accented speech and hence lower speaker similarity to non-native target speakers. This highlights the pros and cons of these adaptation methods. Interestingly, WER of TTS$_s$ was worse than that of TTS$_u$ while the natural speech (NAT) had the worst score of all in the non-native case. This can be interpreted as that TTS$_s$ produces pronunciation which is more similar to the natural speech than TTS$_u$, which means TTS$_s$ is better at capturing non-standard speaker characteristics.

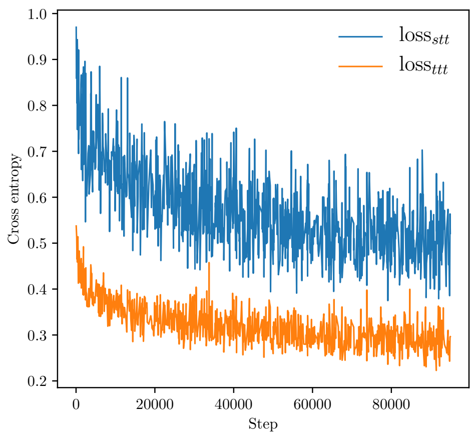
In summary, the proposed system had higher speaker similarity than the baseline systems. Our TTS system, in particular, benefited from the supervised strategy although the improvement was relatively small. Regarding the TTS$_u$ and the other two VC systems that had slightly better quality than the natural speech, we suspect that this is due to the reduced/lack of accents of their generated speech. This hints at potential uses for other accent-related tasks [136].

Scenario conclusion

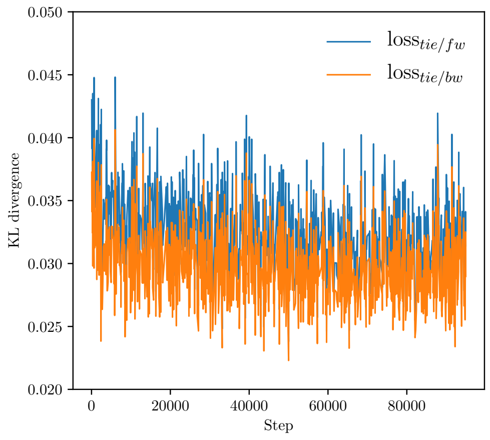
The subjective results have shown that the fine-tuning approach is better at capturing unique speaker characteristics than the speaker embedding approach when data are sufficient. Our systems, in particular, achieved high performance for native speakers as well as non-native speakers. Moreover our cloning strategy can be adjusted to take advantage of the transcriptions if they are available. In the meantime, the experiment also points out the limitations of the subjective evaluation. While the current quality and similarity questions work well for native



(a) Speech generation goals



(b) Phoneme classification goals



(c) Latent space tying

Figure 8.8: Training curve of different loss terms when jointly train the text-speech multimodal system.

speakers, listeners' judgements were biased when they needed to evaluate the voices of non-native speakers.

8.6 Analysis and Discussion

This section looks into several aspects of the proposed model to provide more insight on its behaviors. The information is provided as it is without any evaluation for readers to use as references and form their own opinions on the subject.

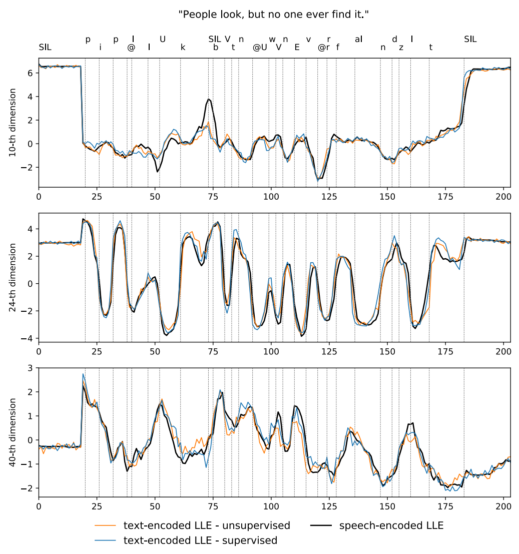
8.6.1 Training robust linguistic latent spaces

The linguistic latent spaces obtained in the initial training stage have critical effect on the performance of the proposed system, as the rest of the voice cloning procedure functions on the assumption that LLE is a speaker-disentangled linguistic feature. Therefore, the training of the text-speech multimodal system must be carefully designed to guarantee that objective. If we only consider the text encoder and the speech decoder, then the proposed system is just a multi-speaker TTS model which lacks the ability to adapt with untranscribed speech. By adding a speaker-independent speech encoder, we provide a backdoor for unsupervised speaker adaptation, which is the topic explored in our previous publications [93, 22]. If we only consider the speech encoder and speech decoder, then it is not much different from a VAE-based multi-speaker non-parallel VC system [57]. However, to avoid the weakness of self-supervised models, which is the dependence on regularization to shape the latent space indirectly, we jointly trained the STS stack with the text encoder and transcribed speech in a supervised fashion. This ensures that the latent spaces will contain linguistic information which in turn guarantees a moderately high level of performance for VC [141].

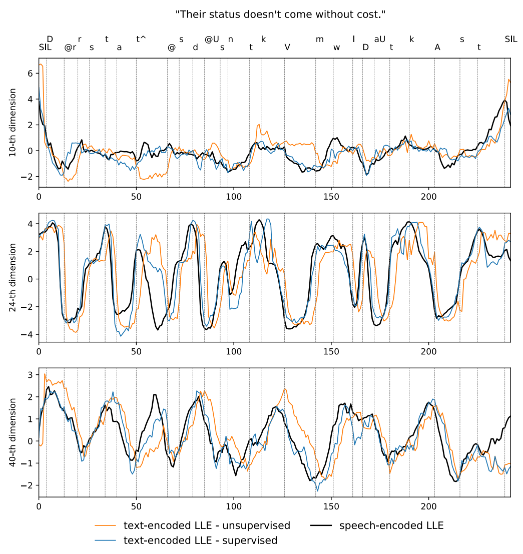
By jointly training the text and speech encoder, we help the speech encoder to learn a speaker-disentangled representation, as it is forced to approximate the text encoder, which is speaker-independent by nature. Figure 8.8a shows the training curves of the TTS and STS goals, both of which descend over time and gradually converge to each other. However in practice, we have to de-emphasize loss$_{sts}$ with a weighting parameter and observe the progress of the training curves, as there is a risk that the training will focus on optimizing loss$_{sts}$ and abandon loss$_{tts}$ completely, as there is always an easy and uninteresting solution to the autoencoder task. In summary, a robust and speaker-disentangled latent linguistic representation is guaranteed by strategic placement of speaker components, joint training of the TTS and STS stacks, and use of a large-scale transcribed multi-speaker corpus. Furthermore, the tied-layer loss is used in conjunction with the join-goal losses to encourage a consistent representation between text-encoded and speech-encoded latent spaces. Figure 8.8c shows the forward and backward KL divergence learning curves, which reveals that a small gap still exist between the two. Finally, the text decoder was used to force the LLE to focus more on phonetic information by adding loss$_{stt}$ to the optimizing loss. Interestingly, even though loss$_{ttt}$ is not optimized, it is still better than loss$_{stt}$, as can be seen in Figure 8.8b.‘

8.6.2 LLE of the supervised and unsupervised models

As mentioned in earlier sections, the architecture of the NAUTILUS system used in this paper is not an E2E system, which is inconvenient for practical appli-



(a) Native speaker as target - p294



(b) Non-native speaker as target - MF6

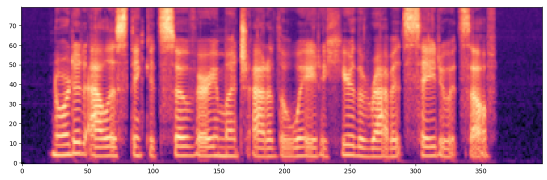
Figure 8.9: Examples of 64-dimensional LLE sequences generated by the text and speech encoders of models adapted using either the supervised or unsupervised cloning strategy.

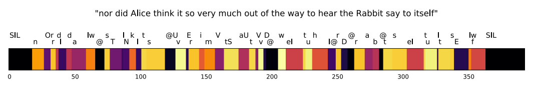
cations but it allows us to have more control over the duration of the generated utterances. In this section, we look into the linguistic latent spaces of the adapted models to understand the behaviors of the supervised and unsupervised cloning strategies. Figure 8.9 shows selected dimensions of the 64-dimensional LLE sequences generated by either the text or speech encoder of models adapted to either p294 or MF6. For each target speaker, we used speaker-independent speech encoder and an utterance not included in their adaptation data to generate a speech-encoded LLE sequence and then used either the supervised or unsupervised text encoder and the phoneme (text) and duration information of the same utterance to generate text-encoded LLE sequences. This arrangement guarantees the LLE sequences generated from the encoders are aligned, which helps to highlight differences between the supervised and unsupervised text encoders.

Even though we referred to the outputs of both the text and speech decoder as LLE, they actually represent slightly different concepts. The speech-encoded LLE represents the sound spoken in an utterance input, while the text-encoded LLE represents the sound that we want to generate from a symbolic phoneme input. Figure 8.9a shows all three LLE sequences are well-aligned to each other in the case of p294. This suggests that the unsupervised speaker-independent text encoder was able to correctly map the symbolic phoneme to the actual spoken sound when the target is a native English speaker, which left little room for the supervised strategy to improve upon. It is expected as the text and speech encoders were initially trained on transcribed speech of a large-scale native speaker corpus. In contrast, Fig. 8.9b shows clear misalignments between the LLE sequences; the text-encoded LLE sequence of the supervised model seems to align to the speech-encoded LLE sequence better than its unsupervised counterpart. From this figure, we can see that the supervised strategy adjusted the text encoder to map the symbolic phoneme to the actual (wrong) sound spoken by MF6, which helps to improve the speaker similarity but degrades the quality (or pronunciation) of generated utterances. The latent spaces of the models adapted to p345 and MM6, while not presented in this paper, also show similar patterns.

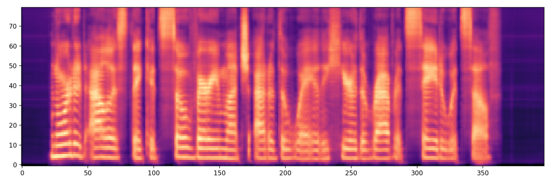
8.6.3 Mel-spectrogram of the generated speech

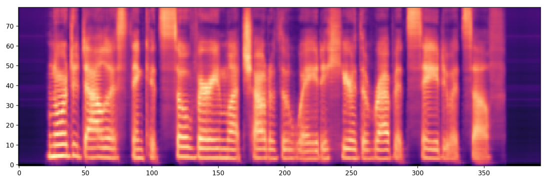
The input features of TTS and VC modes as well as their respective generated mel-spectrogram in different stages of the unsupervised speaker adaptation process are presented in Figure 8.10. More coherent time-domain patterns can be seen on samples generated by the speech decoder that goes through the "welding" stage. As expected, the WaveNet vocoder adds fine-grained details to the the generated speech as it generates waveform one sample at a time.

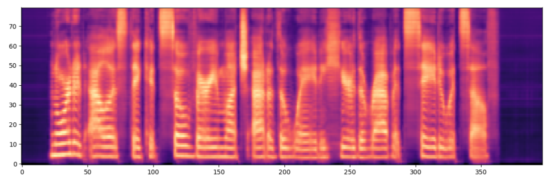

(a) Input - Source utterance

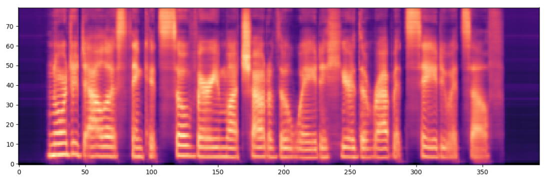


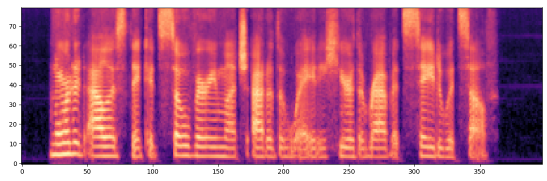
(b) Input - Phonemes and durations

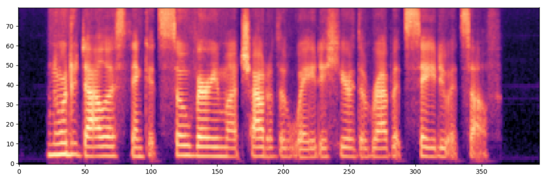

(c) Speech decoder - Adaptation (VC)

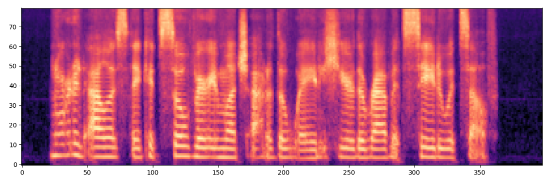

(d) Speech decoder - Adaptation (TTS)


(e) Speech decoder - Welding (VC)


(f) Speech decoder - Welding (TTS)


(g) WaveNet vocoder - Adaptation (VC)


(h) WaveNet vocoder - Adaptation (TTS)


(i) WaveNet vocoder - Welding (VC)

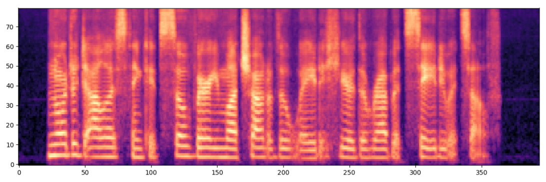

(j) WaveNet vocoder - Welding (TTS)

Figure 8.10: Examples of mel-spectrogram in each stage of the proposed unsupervised voice cloning procedure.

Chapter 9

Conclusions

9.1 Contributions

In this thesis, we have described a novel deep learning based voice cloning framework which can be used for both TTS and VC. The method is first split into several aspects, which are tackled separately in each chapter, before represented as a unified framework. This narrative setup helps provide more insights about each individual technique so they can be applied to similar problems. In summary, the four major contributions of this thesis are as follow:

1) Proposing the crossmodal speaker adaptation method to adapt with untranscribed speech and backpropagation algorithm. The proposed fine-tuning based approach avoids the shortcoming of existing speaker embedding based approach, which is the lack of data scalability, and unifies the techniques used for supervised and unsupervised speaker adaptation.

2) Systematically investigating different types of speaker components to use in a speaker-adaptive neural acoustic model, and concluding that the best type of speaker component is highly dependent on the network architecture as well as the linguistic and acoustic representation. In the other words, the experiments can be treated as a guideline for designing a speaker-adaptive acoustic model.

3) Providing a proof-of-concept that the proposed latent linguistic embedding can be used for standard and cross-lingual voice conversion. The method has similar data efficiency as PPG-based systems and can be utilized for low-resource languages. Having the same framework for both TTS and VC systems allows better integration for using both of them to handle a particular task.

4) Integrating the main principles into a large-scale neural network and producing SOTA performance by incorporating modern deep learning components of a speech generation system. Besides high quality synthetic speech, the proposed

system also has a high speaker consistency when switching between TTS and VC which is a useful for many practical aapplications.

9.2 Future works

At the end of this thesis, we essentially create a highly versatile voice cloning framework. As a speech generation system, improving the general naturalness and the similarity to target speakers is a continuing process. However given the unique characteristic of the proposed system, the following two research directions are interesting for future works:

1) A speech generation system is essentially a system which manipulates the linguistic content of generated speech, while a voice cloning is a system which manipulates the speaker characteristics. Recent end-to-end approaches pursue a simple system without having to explicitly model all requisite information. In contrast, our system, while also reduces the requirements for building a speech generation system, carefully models different modality to allow a high degree of control over generated speech utterances. Throughout this thesis, speech is assumed to be a product of linguistic content and speaker characteristics. However, speech is a complex, multimodal feature. For future work, we will focus on controlling other para-linguistic features of speech or increasing the degree of control over existing ones. A simple example is controlling emotions of the utterances or controlling fine-grained details of a speaker, like accents.

2) The TTS and VC modes of the proposed system can be interpreted as different input interfaces to control the output generated speech. Given the factorized and multimodal structure of it, the proposed voice cloning framework can be extended to other less popular speech generation tasks like video-to-speech. Besides for purpose of convenience, merging these tasks into the base TTS/VC system also opens up the possibility to improve performance for low-resource uncommon tasks by combining available data of TTS or VC corpora and executing semi-supervised training.

Bibliography

[1] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura, "Speech parameter generation algorithms for hmm-based speech synthesis," in *Proc. ICASSP*, 2000, pp. 1315–1318.

[2] Y. Stylianou, O. Cappé, and E. Moulines, "Continuous probabilistic transform for voice conversion," *IEEE Trans. Speech & Audio Process.*, vol. 6, no. 2, pp. 131–142, 1998.

[3] T. L. Cornu and B. Milner, "Reconstructing intelligible audio speech from visual speech features," in *Proc. INTERSPEECH*, 2015, pp. 3355–3359.

[4] D. Michelsanti, O. Slizovskaia, G. Haro, E. Gómez, Z.-H. Tan, and J. Jensen, "Vocoder-based speech synthesis from silent videos," *arXiv preprint arXiv:2004.02541*, 2020.

[5] G. Krishna, C. Tran, Y. Han, and M. Carnahan, "Speech synthesis using eeg," in *Proc. ICASSP*, 2020, pp. 1235–1238.

[6] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerry-Ryan, R. A. Saurous, Y. Agiomyrgiannakis, and Y. Wu, "Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions," in *Proc. ICASSP*, 2018, pp. 4779–4783.

[7] N. Li, S. Liu, Y. Liu, S. Zhao, and M. Liu, "Neural speech synthesis with transformer network," in *Proc. AAAI Conf. AI*, vol. 33, 2019, pp. 6706–6713.

[8] H. Miyoshi, Y. Saito, S. Takamichi, and H. Saruwatari, "Voice conversion using sequence-to-sequence learning of context posterior probabilities," *Proc. INTERSPEECH*, pp. 1268–1272, 2017.

[9] K. Tanaka, H. Kameoka, T. Kaneko, and N. Hojo, "AttS2S-VC: Sequence-to-sequence voice conversion with attention and context preservation mechanisms," in *Proc. ICASSP*, 2019, pp. 6805–6809.

[10] J. Zhang, Z. Ling, and L.-R. Dai, “Non-parallel sequence-to-sequence voice conversion with disentangled linguistic and speaker representations,” *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 28, pp. 540–552, 2019.

[11] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, “WaveNet: A generative model for raw audio,” *arXiv preprint arXiv:1609.03499*, 2016.

[12] R. Prenger, R. Valle, and B. Catanzaro, “Waveglow: A flow-based generative network for speech synthesis,” in *Proc. ICASSP*, 2019, pp. 3617–3621.

[13] X. Wang, S. Takaki, and J. Yamagishi, “Neural source-filter waveform models for statistical parametric speech synthesis,” *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 28, pp. 402–415, 2019.

[14] L.-J. Liu, Z.-H. Ling, Y. Jiang, M. Zhou, and L.-R. Dai, “Wavenet vocoder with limited training data for voice conversion,” in *Proc. INTERSPEECH*, 2018, pp. 1983–1987.

[15] S. Arik, J. Chen, K. Peng, W. Ping, and Y. Zhou, “Neural voice cloning with a few samples,” in *Proc. NIPS*, 2018, pp. 10 040–10 050.

[16] Y. Zhang, R. J. Weiss, H. Zen, Y. Wu, Z. Chen, R. Skerry-Ryan, Y. Jia, A. Rosenberg, and B. Ramabhadran, “Learning to speak fluently in a foreign language: Multilingual speech synthesis and cross-language voice cloning,” *Proc. INTERSPEECH*, pp. 2080–2084, 2019.

[17] J. Lorenzo-Trueba, F. Fang, X. Wang, I. Echizen, J. Yamagishi, and T. Kinnunen, “Can we steal your vocal identity from the internet?: Initial investigation of cloning Obama’s voice using GAN, WaveNet and low-quality found data,” in *Proc. Odyssey*, 2018, pp. 240–247.

[18] A. Gutkin, L. Ha, M. Jansche, K. Pipatsrisawat, and R. Sproat, “TTS for low resource languages: A bangla synthesizer,” in *Proc. LREC*, 2016, pp. 2005–2010.

[19] Y. Jia, Y. Zhang, R. J. Weiss, Q. Wang, J. Shen, F. Ren, Z. Chen, P. Nguyen, R. Pang, I. L. Moreno, and Y. Wu, “Transfer learning from speaker verification to multispeaker text-to-speech synthesis,” *arXiv preprint arXiv:1806.04558*, 2018.

[20] H. Tachibana, K. Uenoyama, and S. Aihara, "Efficiently trainable text-to-speech system based on deep convolutional networks with guided attention," in *Proc. ICASSP*, 2018, pp. 4784–4788.

[21] Y. Chen, Y. Assael, B. Shillingford, D. Budden, S. Reed, H. Zen, Q. Wang, L. C. Cobo, A. Trask, B. Laurie, C. Gulcehre, A. van den Oord, O. Vinyals, and N. de Freitas, "Sample efficient adaptive text-to-speech," *arXiv preprint arXiv:1809.10460*, 2018.

[22] H.-T. Luong and J. Yamagishi, "A unified speaker adaptation method for speech synthesis using transcribed and untranscribed speech with backpropagation," *arXiv preprint arXiv:1906.07414*, 2019.

[23] S. Takamichi, "Acoustic modeling and speech parameter generation for high-quality statistical parametric speech synthesis," Ph.D. dissertation, Nara Institute of Science and Technology, 2016.

[24] P. Taylor, *Text-to-speech synthesis.* Cambridge university press, 2009.

[25] W. Ping, K. Peng, and J. Chen, "Clarinet: Parallel wave generation in end-to-end text-to-speech," in *Proc. ICLR*, 2019, pp. 1–15.

[26] Z. Huang, H. Lu, M. Lei, and Z. Yan, "Linear networks based speaker adaptation for speech synthesis," in *Proc. ICASSP*, 2018, pp. 5319–5323.

[27] J. Yamagishi and T. Kobayashi, "Average-voice-based speech synthesis using HSMM-based speaker adaptation and adaptive training," *IEICE T. Inf. Syst.*, vol. 90, no. 2, pp. 533–543, 2007.

[28] Y. Fan, Y. Qian, F. K. Soong, and L. He, "Multi-speaker modeling and speaker adaptation for DNN-based TTS synthesis," in *Proc. ICASSP*, 2015, pp. 4475–4479.

[29] H.-T. Luong and J. Yamagishi, "Scaling and bias codes for modeling speaker–adaptive DNN–based speech synthesis systems," in *Proc. SLT*, 2018, pp. 610–617.

[30] J. Yamagishi, T. Kobayashi, Y. Nakano, K. Ogata, and J. Isogai, "Analysis of speaker adaptation algorithms for HMM-based speech synthesis and a constrained SMAPLR adaptation algorithm," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 17, no. 1, pp. 66–83, 2009.

[31] K. Inoue, S. Hara, M. Abe, T. Hayashi, R. Yamamoto, and S. Watanabe, “Semi-supervised speaker adaptation for end-to-end speech synthesis with pretrained models,” in *Proc. ICASSP*, 2020, pp. 7634–7638.

[32] Y.-N. Chen, Y. Jiao, Y. Qian, and F. K. Soong, “State mapping for cross-language speaker adaptation in tts,” in *Proc. ICASSP*, 2009, pp. 4273–4276.

[33] S. Takaki, Y. Nishimura, and J. Yamagishi, “Unsupervised speaker adaptation for DNN-based speech synthesis using input codes,” in *Proc. APSIPA*, 2018, pp. 649–658.

[34] Z. Wu, P. Swietojanski, C. Veaux, S. Renals, and S. King, “A study of speaker adaptation for DNN-based speech synthesis,” in *Proc. INTERSPEECH*, 2015, pp. 879–883.

[35] R. Doddipatla, N. Braunschweiler, and R. Maia, “Speaker adaptation in DNN-based speech synthesis using d-vectors,” in *Proc. INTERSPEECH*, 2017, pp. 3404–3408.

[36] E. Cooper, C.-I. Lai, Y. Yasuda, F. Fang, X. Wang, N. Chen, and J. Yamagishi, “Zero-shot multi-speaker text-to-speech with state-of-the-art neural speaker embeddings,” *arXiv preprint arXiv:1910.10838*, 2019.

[37] B. H. Juang and L. R. Rabiner, “Hidden markov models for speech recognition,” *Technometrics*, vol. 33, no. 3, pp. 251–272, 1991.

[38] K. Tokuda, T. Kobayashi, and S. Imai, “Speech parameter generation from hmm using dynamic features,” in *Proc. ICASSP*, vol. 1. IEEE, 1995, pp. 660–663.

[39] S. J. Young, J. J. Odell, and P. C. Woodland, “Tree-based state tying for high accuracy acoustic modelling,” in *Proc. ARPA Human Language Technology Workshop*, 1994, pp. 307–312.

[40] T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura, “Simultaneous modeling of spectrum, pitch and duration in hmm-based speech synthesis,” in *Proc. EUROSPEECH*, 1999, pp. 2374–2350.

[41] C. J. Leggetter and P. C. Woodland, “Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models,” *Computer speech & language*, vol. 9, no. 2, pp. 171–185, 1995.

[42] V. V. Digalakis, D. Rtischev, and L. G. Neumeyer, "Speaker adaptation using constrained estimation of gaussian mixtures," *IEEE Trans. Speech Audio Process.*, vol. 3, no. 5, pp. 357–366, 1995.

[43] M. J. Gales, "Maximum likelihood linear transformations for HMM-based speech recognition," *Computer speech & language*, vol. 12, no. 2, pp. 75–98, 1998.

[44] D. Povey and G. Saon, "Feature and model space speaker adaptation with full covariance gaussians," in *Proc. ICSLP*, 2006.

[45] J. Yamagishi, "Average-voice-based speech synthesis," Ph.D. dissertation, Tokyo Institute of Technology, 2006.

[46] J.-L. Gauvain and C.-H. Lee, "Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains," *IEEE Trans. Speech & Audio Process.*, vol. 2, no. 2, pp. 291–298, 1994.

[47] O. Siohan, T. A. Myrvoll, and C.-H. Lee, "Structural maximum a posteriori linear regression for fast hmm adaptation," *Computer Speech & Language*, vol. 16, no. 1, pp. 5–24, 2002.

[48] A. Kain and M. W. Macon, "Spectral voice conversion for text-to-speech synthesis," in *Proc. ICASSP*, 1998, pp. 285–288.

[49] Y. Stylianou, O. Cappe, and E. Moulines, "Statistical methods for voice quality transformation," in *Proc. EUROSPEECH*, 1995, pp. 447–450.

[50] T. Toda, A. W. Black, and K. Tokuda, "Voice conversion based on maximum-likelihood estimation of spectral parameter trajectory," *IEEE Trans. Audio, Speech, Language Process.*, vol. 15, no. 8, pp. 2222–2235, 2007.

[51] M. Abe, S. Nakamura, K. Shikano, and H. Kuwabara, "Voice conversion through vector quantization," *J. Acoust. Soc. Jpn. (E)*, vol. 11, no. 2, pp. 71–76, 1990.

[52] C.-c. Yeh, P.-c. Hsu, J.-c. Chou, H.-y. Lee, and L.-s. Lee, "Rhythm-flexible voice conversion without parallel data using cycle-GAN over phoneme posteriorgram sequences," in *Proc. SLT*, 2018, pp. 274–281.

[53] J. Lorenzo-Trueba, J. Yamagishi, T. Toda, D. Saito, F. Villavicencio, T. Kinnunen, and Z. Ling, "The voice conversion challenge 2018: Promoting development of parallel and nonparallel methods," in *Proc. Odyssey*, 2018, pp. 195–202.

[54] H. Kameoka, T. Kaneko, K. Tanaka, and N. Hojo, "StarGAN-VC: Non-parallel many-to-many voice conversion using star generative adversarial networks," in *Proc. SLT*, 2018, pp. 266–273.

[55] Y. Chen, M. Chu, E. Chang, J. Liu, and R. Liu, "Voice conversion with smoothed GMM and MAP adaptation," in *Proc. EUROSPEECH*, 2003, pp. 2413–2416.

[56] T. Toda, Y. Ohtani, and K. Shikano, "Eigenvoice conversion based on gaussian mixture model," in *Proc. INTERSPEECH*, 2006, pp. 2446–2449.

[57] C.-C. Hsu, H.-T. Hwang, Y.-C. Wu, Y. Tsao, and H.-M. Wang, "Voice conversion from non-parallel corpora using variational auto-encoder," in *Proc. APSIPA*, 2016, pp. 1–6.

[58] L. Sun, K. Li, H. Wang, S. Kang, and H. Meng, "Phonetic posteriorgrams for many-to-one voice conversion without parallel data training," in *Proc. ICME*, 2016, pp. 1–6.

[59] M. Zhang, X. Wang, F. Fang, H. Li, and J. Yamagishi, "Joint training framework for text-to-speech and voice conversion using multi-source Tacotron and WaveNet," *Proc. INTERSPEECH*, pp. 1298–1302, 2019.

[60] H. Kameoka, T. Kaneko, K. Tanaka, and N. Hojo, "ACVAE-VC: Non-parallel many-to-many voice conversion with auxiliary classifier variational autoencoder," *arXiv preprint arXiv:1808.05092*, 2018.

[61] C.-C. Hsu, H.-T. Hwang, Y.-C. Wu, Y. Tsao, and H.-M. Wang, "Voice conversion from unaligned corpora using variational autoencoding wasserstein generative adversarial networks," in *Proc. INTERSPEECH*, 2017, pp. 3364–3368.

[62] X. Tian, J. Wang, H. Xu, E. S. Chng, and H. Li, "Average modeling approach to voice conversion with non-parallel data." in *Proc. Odyssey*, 2018, pp. 227–232.

[63] Y. Saito, Y. Ijima, K. Nishida, and S. Takamichi, "Non-parallel voice conversion using variational autoencoders conditioned by phonetic posteriorgrams and d-vectors," in *Proc. ICASSP.* IEEE, 2018, pp. 5274–5278.

[64] Y. Zhou, X. Tian, H. Xu, R. K. Das, and H. Li, "Cross-lingual voice conversion with bilingual phonetic posteriorgram and average modeling," in *Proc. ICASSP*, 2019, pp. 6790–6794.

[65] O. Watts, G. E. Henter, J. Fong, and C. Valentini-Botinhao, "Where do the improvements come from in sequence-to-sequence neural TTS?" in *Proc. SSW*, 2019.

[66] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning.* MIT press, 2016.

[67] A. Black, P. Taylor, R. Caley, R. Clark, K. Richmond, S. King, V. Strom, and H. Zen, "The festival speech synthesis system, version 1.4.2," *Unpublished document available via http://www.cstr.ed.ac.uk/projects/festival.html*, 2001.

[68] K. Oura, S. Sako, and K. Tokuda, "Japanese text-to-speech synthesis system: Open jtalk," in *Proc. ASJ*, 2010, pp. 343–344.

[69] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. Le, Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous, "Tacotron: Towards end-to-end speech synthesis," *Proc. INTERSPEECH*, pp. 4006–4010, 2017.

[70] Y. Yasuda, X. Wang, S. Takaki, and J. Yamagishi, "Investigation of enhanced tacotron text-to-speech synthesis systems with self-attention for pitch accent language," in *Proc. ICASSP*, 2019, pp. 6905–6909.

[71] T. Fujimoto, K. Hashimoto, K. Oura, Y. Nankaku, and K. Tokuda, "Impacts of input linguistic feature representation on japanese end-to-end speech synthesis," in *Proc. SSW10*, 2019, pp. 166–171.

[72] H.-T. Luong and H.-Q. Vu, "A non-expert kaldi recipe for vietnamese speech recognition system," in *Proc. WLSI/OIAF4HLT*, 2016, pp. 51–55.

[73] P. Ghahremani, B. BabaAli, D. Povey, K. Riedhammer, J. Trmal, and S. Khudanpur, "A pitch extraction algorithm tuned for automatic speech recognition," in *Proc. ICASSP.* IEEE, 2014, pp. 2494–2498.

[74] M. Narendranath, H. A. Murthy, S. Rajendran, and B. Yegnanarayana, “Transformation of formants for voice conversion using artificial neural networks,” *Speech communication*, vol. 16, no. 2, pp. 207–216, 1995.

[75] H. Zen, A. Senior, and M. Schuster, “Statistical parametric speech synthesis using deep neural networks,” in *Proc. ICASSP*, 2013, pp. 7962–7966.

[76] H. Kawahara, I. Masuda-Katsuse, and A. 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*, vol. 27, no. 3-4, pp. 187–207, 1999.

[77] M. Morise, F. Yokomori, and K. Ozawa, “WORLD: a vocoder-based high-quality speech synthesis system for real-time applications,” *IEICE T. Inf. Syst.*, vol. 99, no. 7, pp. 1877–1884, 2016.

[78] J. Kominek, T. Schultz, and A. W. Black, “Synthesizer voice quality of new languages calibrated with mean mel cepstral distortion,” in *Proc. SLTU*, 2008, pp. 63–68.

[79] H.-T. Luong, S. Takaki, G. E. Henter, and J. Yamagishi, “Adapting and controlling DNN-based speech synthesis using input codes,” in *Proc. ICASSP*, 2017, pp. 4905–4909.

[80] Y. Zhao, D. Saito, and N. Minematsu, “Speaker representations for speaker adaptation in multiple speakers’ BLSTM-RNN-based speech synthesis,” in *Proc. INTERSPEECH*, 2016, pp. 2268–2272.

[81] J. Latorre, J. Lachowicz, J. Lorenzo-Trueba, T. Merritt, T. Drugman, S. Ronanki, and V. Klimkov, “Effect of data reduction on sequence-to-sequence neural TTS,” in *Proc. ICASSP*, 2019, pp. 7075–7079.

[82] F.-Y. Kuo, S. Aryal, G. Degottex, S. Kang, P. Lanchantin, and I. Ouyang, “Data selection for improving naturalness of TTS voices trained on small found corpuses,” in *Proc. SLT*, 2018, pp. 319–324.

[83] E. Cooper, X. Wang, A. Chang, Y. Levitan, and J. Hirschberg, “Utterance selection for optimizing intelligibility of TTS voices trained on ASR data.” in *Proc. INTERSPEECH*, 2017, pp. 3971–3975.

[84] K.-Z. Lee, E. Cooper, and J. Hirschberg, "A comparison of speaker-based and utterance-based data selection for text-to-speech synthesis," in *Proc. INTERSPEECH*, 2018, pp. 2873–2877.

[85] H.-T. Luong, X. Wang, J. Yamagishi, and N. Nishizawa, "Training multi-speaker neural text-to-speech systems using speaker-imbalanced speech corpora," *Proc. INTERSPEECH*, pp. 1303–1307, 2019.

[86] N. Hojo, Y. Ijima, and H. Mizuno, "DNN-based speech synthesis using speaker codes," *IEICE T. Inf. Syst.*, vol. 101, no. 2, pp. 462–472, 2018.

[87] Z. Kons, S. Shechtman, A. Sorin, R. Hoory, C. Rabinovitz, and E. Da Silva Morais, "Neural TTS voice conversion," in *Proc. SLT*, 2018, pp. 290–296.

[88] D. Yu, K. Yao, H. Su, G. Li, and F. Seide, "KL-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition," in *Proc. ICASSP*, 2013, pp. 7893–7897.

[89] K. Li, J. Li, Y. Zhao, K. Kumar, and Y. Gong, "Speaker adaptation for end-to-end CTC models," in *Proc. SLT*, 2018, pp. 542–549.

[90] S. Xue, O. Abdel-Hamid, H. Jiang, L.-R. Dai, and Q. Liu, "Fast adaptation of deep neural network based on discriminant codes for speech recognition," *IEEE/ACM T. Audio Speech*, vol. 22, no. 12, pp. 1713–1725, 2014.

[91] O. Abdel-Hamid and H. Jiang, "Fast speaker adaptation of hybrid NN/HMM model for speech recognition based on discriminative learning of speaker code," in *Proc. ICASSP*, 2013, pp. 7942–7946.

[92] O. Watts, Z. Wu, and S. King, "Sentence-level control vectors for deep neural network speech synthesis," in *Proc. INTERSPEECH*, 2015, pp. 2217–2221.

[93] H.-T. Luong and J. Yamagishi, "Multimodal speech synthesis architecture for unsupervised speaker adaptation," in *Proc. INTERSPEECH*, 2018, pp. 2494–2498.

[94] S. Karita, S. Watanabe, T. Iwata, A. Ogawa, and M. Delcroix, "Semi-supervised end-to-end speech recognition." in *Proc. INTERSPEECH*, 2018, pp. 2–6.

[95] W.-C. Huang, T. Hayashi, Y.-C. Wu, H. Kameoka, and T. Toda, "Voice transformer network: Sequence-to-sequence voice conversion using transformer with text-to-speech pretraining," *arXiv preprint arXiv:1912.06813*, 2019.

[96] L. Kaiser, A. N. Gomez, N. Shazeer, A. Vaswani, N. Parmar, L. Jones, and J. Uszkoreit, "One model to learn them all," *arXiv preprint arXiv:1706.05137*, 2017.

[97] B. Li and H. Zen, "Multi-language multi-speaker acoustic modeling for LSTM-RNN based statistical parametric speech synthesis." in *Proc. INTERSPEECH*, 2016, pp. 2468–2472.

[98] V. Wan, Y. Agiomyrgiannakis, H. Silen, and J. Vit, "Google's next-generation real-time unit-selection synthesizer using sequence-to-sequence LSTM-based autoencoders," in *Proc. INTERSPEECH*, 2017, pp. 1143–1147.

[99] R. Caruana, "Multitask learning," in *Learning to learn.* Springer, 1998, pp. 95–133.

[100] Y. Zhao, J. Li, and Y. Gong, "Low-rank plus diagonal adaptation for deep neural networks," in *Proc. ICASSP*, 2016, pp. 5005–5009.

[101] J. Neto, L. Almeida, M. Hochberg, C. Martins, L. Nunes, S. Renals, and T. Robinson, "Speaker-adaptation for hybrid HMM-ANN continuous speech recognition system," in *Proc. EUROSPEECH*, 1995, pp. 2171–2174.

[102] J. Xue, J. Li, D. Yu, M. Seltzer, and Y. Gong, "Singular value decomposition based low-footprint speaker adaptation and personalization for deep neural network," in *Proc. ICASSP*, 2014, pp. 6359–6363.

[103] T. Tan, Y. Qian, M. Yin, Y. Zhuang, and K. Yu, "Cluster adaptive training for deep neural network," in *Proc. ICASSP*, 2015, pp. 4325–4329.

[104] L. Samarakoon and K. C. Sim, "Factorized hidden layer adaptation for deep neural network based acoustic modeling," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 24, no. 12, pp. 2241–2250, 2016.

[105] G. Saon, H. Soltau, D. Nahamoo, and M. Picheny, "Speaker adaptation of neural network acoustic models using i-vectors." in *Proc. ASRU*, 2013, pp. 55–59.

[106] P. Swietojanski and S. Renals, "Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models," in *Proc. SLT*, 2014, pp. 171–176.

[107] L. Samarakoon and K. C. Sim, "Subspace LHUC for fast adaptation of deep neural network acoustic models." in *Proc. INTERSPEECH*, 2016, pp. 1593–1597.

[108] X. Cui, V. Goel, and G. Saon, "Embedding-based speaker adaptive training of deep neural networks," in *Proc. INTERSPEECH*, 2017, pp. 122–126.

[109] H.-T. Luong, X. Wang, J. Yamagishi, and N. Nishizawa, "Investigating accuracy of pitch-accent annotations in neural network-based speech synthesis and denoising effects," *Proc. INTERSPEECH*, pp. 37–41, 2018.

[110] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," *arXiv preprint arXiv:1512.09300*, 2015.

[111] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[112] P. Swietojanski and S. Renals, "SAT-LHUC: Speaker adaptive training for learning hidden unit contributions." in *Proc. ICASSP*, 2016, pp. 5010–5014.

[113] Y. Deng, L. He, and F. Soong, "Modeling multi-speaker latent space to improve neural TTS Quick enrolling new speaker and enhancing premium voice," *arXiv preprint arXiv:1812.05253v2*, 2018.

[114] T. Hayashi, A. Tamamori, K. Kobayashi, K. Takeda, and T. Toda, "An investigation of multi-speaker training for WaveNet vocoder," in *Proc. ASRU*, 2017, pp. 712–718.

[115] M. Abe, K. Shikano, and H. Kuwabara, "Statistical analysis of bilingual speaker's speech for cross-language voice conversion," *J. Acoust. Soc. Am.*, vol. 90, no. 1, pp. 76–82, 1991.

[116] M. Mashimo, T. Toda, K. Shikano, and N. Campbell, "Evaluation of cross-language voice conversion based on GMM and Straight," in *Proc. EUROSPEECH*, 2001, pp. 361–364.

[117] S. S. Rallabandi and S. V. Gangashetty, "An approach to cross-lingual voice conversion," in *Proc. IJCNN*, 2019.

[118] Y.-J. Wu, Y. Nankaku, and K. Tokuda, “State mapping based method for cross-lingual speaker adaptation in hmm-based speech synthesis,” in *Proc. INTERSPEECH*, 2009, pp. 528–531.

[119] L. Sun, H. Wang, S. Kang, K. Li, and H. M. Meng, “Personalized, cross-lingual tts using phonetic posteriorgrams.” in *Proc. INTERSPEECH*, 2016, pp. 322–326.

[120] E. Dunbar, R. Algayres, J. Karadayi, M. Bernard, J. Benjumea, X.-N. Cao, L. Miskic, C. Dugrain, L. Ondel, A. W. Black, L. Besacier, S. Sakti, and E. Dupoux, “The zero resource speech challenge 2019: TTS without T,” *arXiv preprint arXiv:1904.11469*, 2019.

[121] S. Feng, T. Lee, and Z. Peng, “Combining adversarial training and disentangled speech representation for robust zero-resource subword modeling,” *arXiv preprint arXiv:1906.07234*, 2019.

[122] A. T. Liu, P.-c. Hsu, and H.-y. Lee, “Unsupervised end-to-end learning of discrete linguistic units for voice conversion,” *arXiv preprint arXiv:1905.11563*, 2019.

[123] R. Eloff, A. Nortje, B. van Niekerk, A. Govender, L. Nortje, A. Pretorius, E. Van Biljon, E. van der Westhuizen, L. van Staden, and H. Kamper, “Unsupervised acoustic unit discovery for speech synthesis using discrete latent-variable neural networks,” *arXiv preprint arXiv:1904.07556*, 2019.

[124] S. Sitaram, S. Palkar, Y.-N. Chen, A. Parlikar, and A. W. Black, “Bootstrapping text-to-speech for speech processing in languages without an orthography,” in *Proc. ICASSP*, 2013, pp. 7992–7996.

[125] P. K. Muthukumar and A. W. Black, “Automatic discovery of a phonetic inventory for unwritten languages for statistical speech synthesis,” in *Proc. ICASSP*, 2014, pp. 2594–2598.

[126] K. Kobayashi and T. Toda, “sprocket: Open-source voice conversion software,” in *Proc. Odyssey*, 2018, pp. 203–210.

[127] T. Tu, Y.-J. Chen, C.-c. Yeh, and H.-y. Lee, “End-to-end text-to-speech for low-resource languages by cross-lingual transfer learning,” *arXiv preprint arXiv:1904.06508*, 2019.

[128] H.-T. Luong and J. Yamagishi, “NAUTILUS: a versatile voice cloning system,” *arXiv preprint arXiv:2005.11004*, 2020.

[129] Y. Zhao, S. Takaki, H.-T. Luong, J. Yamagishi, D. Saito, and N. Minematsu, "Wasserstein GAN and waveform loss-based acoustic model training for multi-speaker text-to-speech synthesis systems using a wavenet vocoder," *IEEE Access*, vol. 6, pp. 60 478–60 488, 2018.

[130] W.-C. Huang, Y.-C. Wu, H.-T. Hwang, P. L. Tobing, T. Hayashi, K. Kobayashi, T. Toda, Y. Tsao, and H.-M. Wang, "Refined WaveNet vocoder for variational autoencoder based voice conversion," in *Proc. EUSIPCO*, 2019, pp. 1–5.

[131] X. Wang, S. Takaki, and J. Yamagishi, "An autoregressive recurrent mixture density network for parametric speech synthesis," in *Proc. ICASSP*, 2017, pp. 4895–4899.

[132] S. Karita, S. Watanabe, T. Iwata, M. Delcroix, A. Ogawa, and T. Nakatani, "Semi-supervised end-to-end speech recognition using text-to-speech and autoencoders," in *Proc. ICASSP*, 2019, pp. 6166–6170.

[133] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N.-E. Y. Soplin, J. Heymann, M. Wiesner, N. Chen *et al.*, "ESPnet: End-to-end speech processing toolkit," *Proc. INTERSPEECH*, pp. 2207–2211, 2018.

[134] Y.-C. Wu, P. L. Tobing, T. Hayashi, K. Kobayashi, and T. Toda, "The NU non-parallel voice conversion system for the voice conversion challenge 2018." in *Proc. Odyssey*, 2018, pp. 211–218.

[135] A. C. Janska and R. A. Clark, "Native and non-native speaker judgements on the quality of synthesized speech," in *Proc. INTERSPEECH*, 2010.

[136] S. Aryal and R. Gutierrez-Osuna, "Can voice conversion be used to reduce non-native accents?" in *Proc. ICASSP*, 2014, pp. 7879–7883.

[137] Y. Oshima, S. Takamichi, T. Toda, G. Neubig, S. Sakti, and S. Nakamura, "Non-native text-to-speech preserving speaker individuality based on partial correction of prosodic and phonetic characteristics," *IEICE Trans. Inf. & Syst.*, vol. 99, no. 12, pp. 3132–3139, 2016.

[138] M. Wester and H. Liang, "The EMIME mandarin bilingual database," 2011, http://hdl.handle.net/1842/4862.

[139] T. Hayashi, R. Yamamoto, K. Inoue, T. Yoshimura, S. Watanabe, T. Toda, K. Takeda, Y. Zhang, and X. Tan, "ESPnet-TTS: Unified, reproducible, and integratable open source end-to-end text-to-speech toolkit," in *Proc. ICASSP*, 2020, pp. 7654–7658.

[140] K. Ito, "The LJ speech dataset," 2017, https://keithito.com/LJ-Speech-Dataset/.

[141] H.-T. Luong and J. Yamagishi, "Bootstrapping non-parallel voice conversion from speaker-adaptive text-to-speech," *Proc. ASRU*, pp. 200–207, 2019.

[142] C. Veaux, J. Yamagishi, and K. MacDonald, "CSTR VCTK corpus: English multi-speaker corpus for CSTR voice cloning toolkit," 2017, http://dx.doi.org/10.7488/ds/1994.

[143] H. Zen, V. Dang, R. Clark, Y. Zhang, R. J. Weiss, Y. Jia, Z. Chen, and Y. Wu, "LibriTTS: A corpus derived from LibriSpeech for text-to-speech," in *Proc. INTERSPEECH*, 2019, pp. 1526–1530.

[144] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 3, pp. 328–339, 1989.

[145] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proc. INTERSPEECH*, 2015, pp. 3214–3218.

[146] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The Kaldi speech recognition toolkit," in *Proc. ASRU*, 2011.

[147] H. Zen and H. Sak, "Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis," in *Proc. ICASSP*, 2015, pp. 4470–4474.

Appendix A

Speech corpora

There are two types of speech corpora used throughout this thesis: multi-speaker corpus used to train initial based models and evaluate multi-speaker tasks, and the target speaker corpus which used exclusive for evaluating voice cloning which data may or may not include transcription. Table A.1 lists the multi-speaker corpus, and Table A.2 listed corpus that used exclusively for evaluating speaker adaptation/voice cloning. The corpora are partitioned into base and target datasets with vary amount of data to uniform the data experiment conditions as listed in Table A.3.

Table A.1: The multi-speaker speech corpora used for training the initial base model and evaluation.

Japanese Voicebank (JVB)	Language:	Japanese
	Speakers:	61 male, 194 female, 255 in total
	Utterances:	∼145 utterances per speaker
	Sampling rate:	48kHz
	Notes:	NII internal corpus
Voice cloning toolkit (VCTK) [142]	Language:	English
	Speakers:	46 male, 62 female, 108 in total
	Utterances:	∼375 utterances per speaker
	Sampling rate:	48kHz
	Notes:	
LibriTTS [143]	Language:	English
	Speakers:	more than one thousand speakers
	Utterances:	varying
	Sampling rate:	24kHz
	Notes:	

Table A.2: The corpora contains speakers used exclusively for evaluation speaker adaptation.

Japanese targets	Language:	Japanese
	Speakers:	1 male, 1 female
	Utterances:	1000 utterances per speaker
	Sampling rate:	48kHz
	Notes:	
Voice Conversion Challenge 2018 [53] (VCC2018)	Language:	English
	Speakers:	2 male, 2 female
	Utterances:	81 utterances per speaker
	Sampling rate:	22,05kHz
	Notes:	
English/Japanese bilingual	Language:	English and Japanese
	Speakers:	1 Male 1 Female
	Utterances:	400 utterances per speaker
	Sampling rate:	48kHz
	Notes:	internal, bilingual speakers
EMIME Mandarin [138]	Language:	English
	Speakers:	1 Male 1 Female
	Utterances:	145 utterances per speaker
	Sampling rate:	48kHz
	Notes:	L2 speakers

Table A.3: The datasets split into the *base* and *target* sets used throughout the thesis.

Dataset	Speakers			Train utterances		Test utterances		Note
	Male	Female	Total	Each	Total	Each	Total	
jvb.small.base	56	56	112	100	11200	10	1120	
jvb.small.target	9	14	23	100	-	10	230	
jvb.medium.base	51	184	235	~148	34713	-	-	
jvb.medium.target	10	10	20	100	-	10	200	
vctk.small.base	24	20	44	~375	16910	10	440	
vctk.small.target	4	3	7	320	-	10	70	
vctk.medium.base	31	41	72	~375	26785	-	-	
vctk.medium.target	4	4	8	320	-	15	120	
vctk.big.base	42	58	100	~375	37052	-	-	
vctk.big.american.target	1	1	2	320	-	20	40	p294 and p345
libritts.clean460.base	638	592	1230	vary	140777	-	-	clean100 + clean360
single.ja.target	1	1	2	1000	-	100	200	F009 and M007
vcc2018.spoke.target	2	2	4	81	-	-	-	
bilingual.enja.target	1	1	2	400E/400J	-	-	-	
emime.mandarin.target	1	1	2	145	-	-	-	MM6 and MF6

Appendix B

Linguistic features

Linguistic features used for TTS system are obtained using an automatic annotator (i.e. front-end) then converted into feature vectors and used as input of acoustic models. Details of linguistic information included in linguistic feature are presented in Table B.2 and B.3 for Japanese and English respectively. In case of the NAUTILUS system, only the current phoneme is used as the linguistic input, as it learns the linguistic context by itself.

Table B.1: Linguistic feature vector configurations.

Language	Dimension	Toolkit
English	367	Flite
Japanese	389	OpenJTalk [68]

Table B.2: Japanese linguistic features.

Type	linguistic information
Phoneme	previous previous phoneme identity
	previous phoneme identity
	current phoneme identity
	next phoneme identity
	next next phoneme identity
Mora	difference between accent location and position of current mora position of current mora in AP (forward/backward)
Word	part-of-speech (POS) of previous/current/next word
	inflected forms of previous/current/next word
	conjugation type of previous/current/next word
Accent phrase (AP)	number of moras in previous/current/next AP
	accent type of previous/current/next AP
	is previous/current/next AP interrogative
	is there a pause between current and previous/next AP
	position of current AP in BG by AP/mora (fw/bw)
Breath group (BG)	number of mora in previous/current/next BG
	number of AP in previous/current/next BG
	position of current BG by BG/AP/mora (fw/bw)
Utterance	number of BG/AP/mora in utterance
Frame	position of current frame in utterance (fw/bw)
	number of frames in utterance

Table B.3: English linguistic features.

Type	linguistic information
Phoneme	previous previous phoneme identity
	previous phoneme identity
	current phoneme identity
	next phoneme identity
	next next phoneme identity
Syllable	number of phoneme in previous/current/next syllable
	is previous/current/next syllable has lexical stress? (stressed)
	is previous/current/next syllable has pitch-accent? (accented)
	position of current syllable in word/phrase (fw/bw)
	number of stressed syllables precede/follow current in phrase
	number of accented syllables precede/follow current in phrase
	distance from previous/next stressed syllable
	distance from previous/next accented syllable
Word	POS of previous/current/next word
	number of syllables in previous/current/next word
	position of current word in phrase (fw/bw)
	number of content words precede/follow current in phrase
	distance from previous/next content word
Phrase	number of syllables in previous/current/next phrase
	number of words in previous/current/next phrase
	position of current phrase in utterance (fw/bw)
Utterance	number of syllables/words/phrases in utterance
Frame	position of current frame in utterance (fw/bw)
	number of frames in utterance

Appendix C

Neural network architectures

This appendix provides details on the neural network architectures used in experiment sections of the chapters in the main content.

C.1 Small-sized neural acoustic model

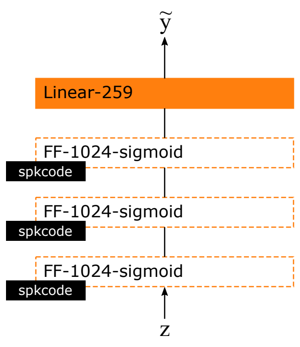
A small-sized neural network, which consists of several feed-forward layers, is used as acoustic model in Chapter 3, 4 and 5 to test the feasibility and behavior of the focused technique.

The neural acoustic model (text encoder and speech decoder) takes aligned linguistic features (dependent on language) and transforms them into acoustic features, which include 60-dimensional mel-cepstral coefficients, 25-dimensional band-limited aperiodicities, interpolated logarithm fundamental frequencies, and their dynamic counterparts. A voiced/unvoiced binary flag is also included. The output is 259-dimensional multitask feature. WORLD spectral analysis [77] was used to extract spectra from 16-bit waveform signals with sampling frequency of 48kHz with a 25 ms window length and 5 ms shift.

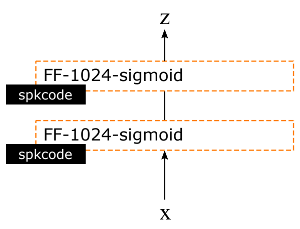
Most of the layers in this setup are feedforward layer with *sigmoid* activation function, unless stated otherwise:

$$\boldsymbol{h}_l = \sigma(\boldsymbol{W}_l \boldsymbol{h}_{l-1} + \boldsymbol{c}_l) \ , \tag{C.1}$$

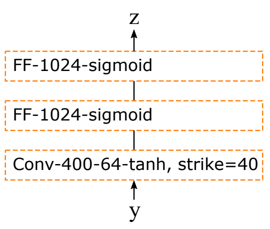
These layers can contain speaker components, like speaker scaling and bias, dependent on the particular experiment of the chapters. The speech encoder used in Chapter 4 takes in raw waveform, sampled at 16 kHz, as the input. The first layer is one-dimensional convolution layer transform the waveform sequence to the same length as the acoustic output by setting width of convolution to 400 and stride to 80. The convolution layer has 64 filters. The output of the convolu-



(a) Speech decoder



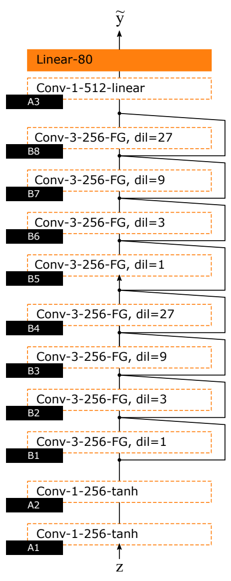
(b) Text encoder



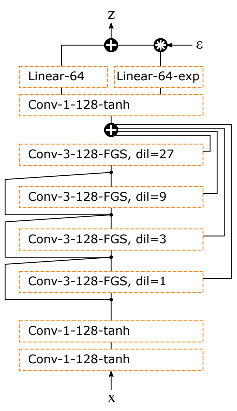
(c) Speech encoder

Figure C.1: Details of the small-sized neural acoustic model.

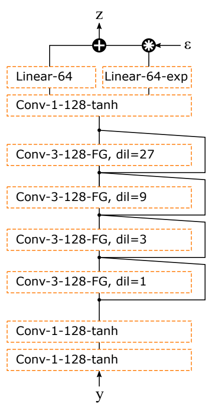
tion layer was then fed into a 1024-dimensional feedforward layer just before the common layers part as shown in Figure C.1.



(a) Speech decoder



(b) Text encoder



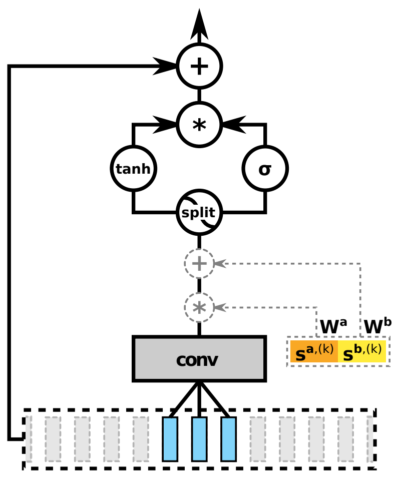
(c) Speech encoder

Figure C.2: Details of the medium-sized neural acoustic model.

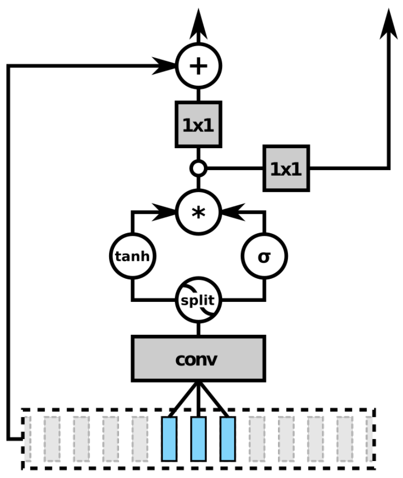
C.2 Medium-sized neural acoustic model

A medium-sized neural network, which consists of one-dimensional convolution layers, is used as acoustic model in Chapter 6 and 7 to evaluate the performance of TTS and VC cloning protocol.

The neural acoustic model (text encoder and speech decoder) takes aligned linguistic features (depending on languages) and transforms them into 80–dimensional mel-spectrograms. The features are extracted from a 25ms window and shifted in steps of 5ms over speech waveform with varying sampling rate depended on the particular experiments. The layers in the speech decoder which can contain speaker component are marked with an identity tag as shown in Figure C.2. The input of the speech encoder is the same mel-spectrogram as the speech decoder output, which make the STS stack a typical variational autoencoder network.



(a) Residual unit



(b) Skip connection unit

Figure C.3: Filter-gate residual and skip connection layer.

C.2.1 Filter-gate activation function

To able to train a very deep network, we use residual and skip connection along with one-dimensional (time-domain) convolution layers with non-linear filter-gate activation function:

$$\boldsymbol{h}_l = \tanh(\boldsymbol{W}_l^f * \boldsymbol{h}_{l-1} + \boldsymbol{c}_l^f) \odot \sigma(\boldsymbol{W}_l^g * \boldsymbol{h}_{l-1} + \boldsymbol{c}_l^g) \tag{C.2}$$

where $\boldsymbol{c}_l^f$, $\boldsymbol{c}_l^g \in \mathbb{R}^{m\times 1}$ are bias of the filter and gate, while $\boldsymbol{W}_l^f$ and $\boldsymbol{W}_l^g$ are filters of convolutions layers used for the filter and the gate respectively. We can increase the captured temporal context, without increase the amount of parameters, by using dilated filters instead of dense filters. The speaker and bias components can also be added into these layers:

$$\begin{aligned}\boldsymbol{h}_l = \tanh(&\operatorname{diag}(\boldsymbol{a}_l^{f,(k)})\boldsymbol{W}_l^f * \boldsymbol{h}_{l-1} + \boldsymbol{c}_l^f + \boldsymbol{b}_l^{f,(k)}) \\ &\odot \sigma(\operatorname{diag}(\boldsymbol{a}_l^{g,(k)})\boldsymbol{W}_l^g * \boldsymbol{h}_{l-1} + \boldsymbol{c}_l^g + \boldsymbol{b}_l^{g,(k)})\end{aligned} \tag{C.3}$$

where the filter and gate have their own scaling vectors $\boldsymbol{a}_l^{f,(k)}$, $\boldsymbol{a}_l^{g,(k)} \in \mathbb{R}^{m\times 1}$ and bias vectors $\boldsymbol{b}_l^{f,(k)}$, $\boldsymbol{b}_l^{g,(k)} \in \mathbb{R}^{m\times 1}$. Just as in the case of the feedforward layer we can factorize the speaker vectors into smaller speaker scaling and bias codes:

$$\boldsymbol{a}_l^{f,(k)} = \boldsymbol{W}_l^{a,f}\boldsymbol{s}_l^{a,(k)} \quad \text{and} \quad \boldsymbol{a}_l^{g,(k)} = \boldsymbol{W}_l^{a,g}\boldsymbol{s}_l^{a,(k)} \tag{C.4}$$

$$\boldsymbol{b}_l^{f,(k)} = \boldsymbol{W}_l^{b,f}\boldsymbol{s}_l^{b,(k)} \quad \text{and} \quad \boldsymbol{b}_l^{g,(k)} = \boldsymbol{W}_l^{b,g}\boldsymbol{s}_l^{b,(k)} \tag{C.5}$$

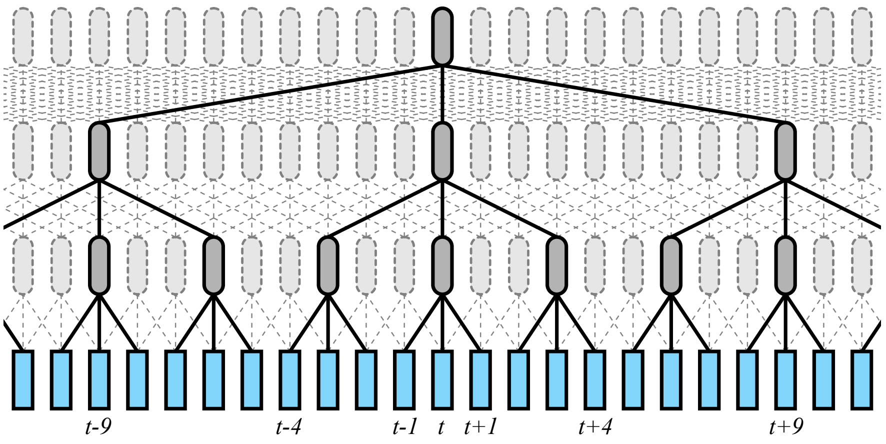


Figure C.4: Temporal contexts captured using a stack of non-overlapping dilated convolution layers.

where the scaling code $\boldsymbol{s}_l^{a,(k)} \in \mathbb{R}^{p\times 1}$ and bias code $\boldsymbol{s}_l^{b,(k)} \in \mathbb{R}^{q\times 1}$ are shared between the filter and the gate.

C.2.2 Dilated one-dimensional convolution layer

We use dilated one-dimensional convolution layer extensively for our system as it is an easy way to capture temporal context and train faster than recurrent layer. The dilated convolution layer is a variation of a time delay neural network (TDNN) [144, 145]. Our version is most similar to the one used in the WaveNet model [11], except that it does not have the causal part. We use blocks of dilated convolution layers to capture both left and right non-overlapping contexts by setting the dilation rate in order of 1, 3, 9 and 27 as illustrated in Figure C.4. In the proposed NAUTILUS system, the causal and non-causal layers are used together to model different functional modules of the system.

C.3 NAUTILUS

The architecture of the proposed system, NAUTILUS, used in experiment section of Chapter 8 is shown in Figure C.5. The text representation $\boldsymbol{x}$ is a phoneme sequence and the speech representation $\boldsymbol{y}$ is mel-spectrogram. The features are extracted from a 50ms window and shifted in steps of 12.5ms over speech waveform with varying sampling rate depended on the particular experiments.

The phoneme alignments of each corpus were extracted using an ASR model trained on the same corpus using the KALDI toolkit [146]. For evaluated utterances, the model trained on the LibriTTS corpus is used to extract their phoneme alignments.

C.3.1 Text encoder

the text encoder transforms a compact phoneme sequence $\boldsymbol{x}$ into the LLE sequence $\boldsymbol{z}$, which has the same length as the acoustic sequence. Our specifications for the text encoder are illustrated in Figure C.5c. The input phoneme sequence is represented as one-hot vectors. As engineered linguistic features are no longer provided, *tenc-linguistic-context* is used to learn the linguistic context. This is a direct imitation of Tacotron 2 [6] but with quasi-RNN used in place of the standard RNN to speed up the training. The attention mechanism is essential in a E2E setup to unroll the phoneme sequence; our setup, however, uses an explicit duration/alignment module called "*tenc-alignment*" in training and inference to have direct control over the generated sample prosody.[1] The coarse linguistic features, then, go through several dilated convolution layers called "*tenc-latent-context*" to capture the local context and smooth out the coarseness. *tenc-latent-context* has essentially the same design as the acoustic models used in our prior work [22], which used residual, skip connection and filter-gate function (Figure 4a in [22]) to help the gradient flow:

$$\boldsymbol{h}_l = \tanh(\boldsymbol{W}_l^f \boldsymbol{h}_{l-1} + \boldsymbol{c}_l^f) \odot \sigma(\boldsymbol{W}_l^g \boldsymbol{h}_{l-1} + \boldsymbol{c}_l^g) \, , \tag{C.6}$$

where $\boldsymbol{h}_l$ is the output of the l-th layer, and $\boldsymbol{W}_l^f$, $\boldsymbol{W}_l^g$, $\boldsymbol{c}_l^f$, and $\boldsymbol{c}_l^f$ are the weights and biases for filters and gates. The output of the text encoder consists of the mean and standard deviation of a text-encoded LLE sequence.

[1]The *tenc-alignment* could be replaced with attention mechanism for convenience, and this could also potentially improve the quality further [65].

C.3.2 Speech decoder

the speech decoder takes in an LLE sequence $\boldsymbol{z}$ to generate a respective acoustic sequence $\widetilde{\boldsymbol{y}}$ with a particular voice. It is essentially a multi-speaker speech synthesis model and there are three components that significantly affect the performance: temporal context capturing [147], autoregressive mechanism [131, 65], and speaker modeling [29]. *sdec-context-blk* captures LLE temporal context by using time-domain convolution (1dconv) layers, which also contain speaker biases in their filters and gates (Figure 4b in [22]):

$$\boldsymbol{h}_l = \tanh(\boldsymbol{W}_l^f \boldsymbol{h}_{l-1} + \boldsymbol{c}_l^f + \boldsymbol{b}_l^{f,(k)}) \odot \sigma(\boldsymbol{W}_l^g \boldsymbol{h}_{l-1} + \boldsymbol{c}_l^g + \boldsymbol{b}_l^{g,(k)}) \,, \tag{C.7}$$

where $\boldsymbol{b}_l^{f,(k)}$ and $\boldsymbol{b}_l^{f,(k)}$ are the speaker biases of k-th speaker in the training speaker pool. The effective type of speaker component depends on the network structure as well as the acoustic features [29]. We previously found that speaker biases work the best for our setup [22].

An autoregressive mechanism is introduced to improve the overall naturalness. *sdec-prenet* is responsible for the autoregressive dependency that captures the past outputs using causal layers. This is a direct imitation of the *AudioEnc* proposed by Tachinaba *et al.* [20]. The layers in *sdec-prenet* use the highway function in the same way as [20] as follows:

$$\boldsymbol{h}_l^f = \boldsymbol{W}_l^f \boldsymbol{h}_{l-1}, \tag{C.8}$$
$$\boldsymbol{h}_l^g = \sigma(\boldsymbol{W}_l^g \boldsymbol{h}_{l-1}), \tag{C.9}$$
$$\boldsymbol{h}_l = \boldsymbol{h}_l^f \odot \boldsymbol{h}_l^g + \boldsymbol{h}_{l-1} \odot (1 - \boldsymbol{h}_l^g) \tag{C.10}$$

The linguistic context and the past-state token are fed into more causal layers before being transformed into the acoustic features. The architecture of the speech decoder is shown in Figure C.5a. We use the mean absolute error (MAE) as the loss function for the speech generation goals. In the adaptation stage, speaker biases are removed from the speech decoder.

C.3.3 Speech encoder

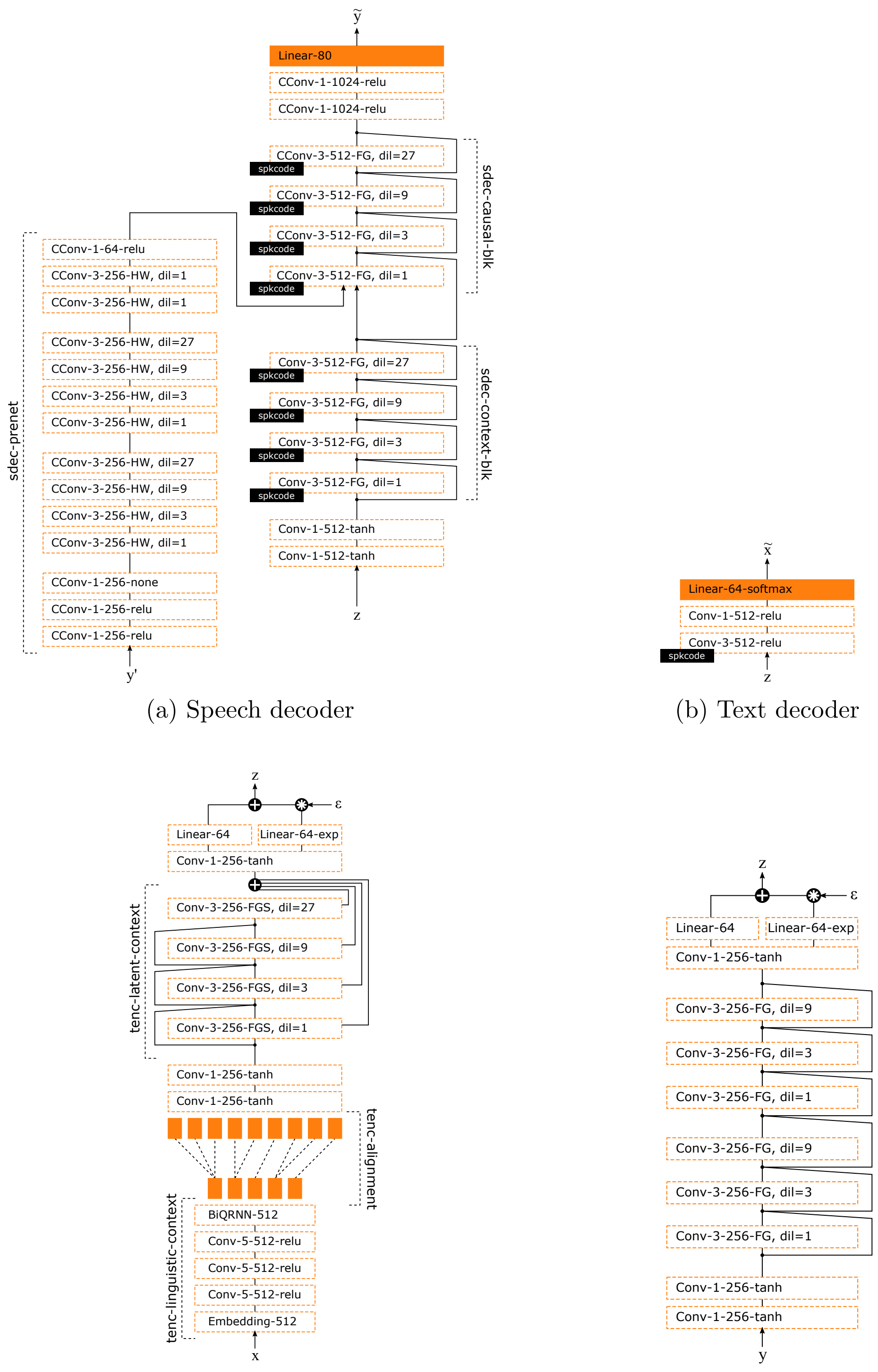
the speech encoder extracts the LLE $\boldsymbol{z}$ from a given acoustic sequence $\boldsymbol{y}$ while stripping unnecessary information (i.e. speaker characteristics). It is similar to an ASR model as the output needs to be independent from training speakers, and the model needs to be generalized to unseen targets. We have no strong preference for speech encoder specification and simply use several dilated layers to capture the local context as illustrated in Figure C.5d.

C.3.4 Text decoder

the text decoder takes an LLE sequence $\boldsymbol{z}$ and predicts the phoneme posterior $\widetilde{\boldsymbol{x}}$ at each frame. This is a new component introduced in this work compared with previous ones [22]. Unlike other modules that would be reused in various stages, the shallow text decoder is included in the training only and acts as an auxiliary regularizer. Its purpose is forcing the latent linguistic embedding to focus more on phoneme information, which we found important for generating utterances with clear pronunciation. The balance between phoneme and other linguistic information is adjustable using the joint-goal weight α_{stt} and the representative power of the text decoder itself. This is why we use a couple of layers only to model the text decoder (Figure C.5b). The cross-entropy criterion is used as the loss function of the phoneme classifier.

C.3.5 WaveNet vocoder

An auto-regressive WaveNet model conditioned on a mel-spectrogram [114, 6, 14] is used as the neural vocoder of our setup. WaveNet is trained on either 22.05kHz or 24kHz speech depending on the scenarios. Waveform amplitudes are quantized by using 10-bit μ-law. The network consists of 40 dilated causal layers containing speaker biases. Both the residual and skip channels are set at 128. This is a typical setup for WaveNet [11]. In the adaptation stage, speaker biases are removed before fine-tuning.



(a) Speech decoder

(b) Text decoder

(c) Text encoder

(d) Speech encoder

Figure C.5: Details network architecture of the proposed NAUTILUS system.

Appendix D

Generated speech samples

This appendix provides links to access speech samples generated by models trained for the experiment sections of the main chapters.

- Chapter 3: speech samples for preliminary experiments on multi-speaker, speaker adaptation and speaker manipulation by gradually changing speaker code within an utterances.
  http://www.hieuthi.com/papers/icassp2017/
- Chapter 4: speech samples for the proof-of-concept experiments on performing unsupervised speaker adaptation with multimodal neural TTS.
  http://www.hieuthi.com/papers/interspeech2018/
- Chapter 5: speech samples for experiments use speaker scaling and speaker bias to model speaker transformation.
  http://www.hieuthi.com/papers/slt2018/
- Chapter 6: speech samples for supervised and unsupervised adaptation with variational multimodal neural TTS.
  https://nii-yamagishilab.github.io/sample-tts-unified-adaptation/
- Chapter 7: speech samples for voice conversion system using latent linguistic embedding for both standard intra-language and cross-lingual scenarios.
  https://nii-yamagishilab.github.io/sample-vc-bootstrapping-tts/
- Chapter 8: speech samples for high performance unified TTS and VC system for a standard "easy" scenario with native English speaker and unique "difficult" scenario with L2 non-native English speaker.
  https://nii-yamagishilab.github.io/sample-versatile-voice-cloning/