Speech Representation Learning for Voice Conversion

Seyed Hamidreza Mohammadi

A DISSERTATION SUBMITTED TO THE FACULTY OF CENTER FOR SPOKEN LANGUAGE UNDERSTANDING WITHIN THE OREGON HEALTH & SCIENCE UNIVERSITY SCHOOL OF MEDICINE IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE AND ENGINEERING

February 2019

Document compiled on

February 21, 2019

The dissertation "Speech Representation Learning for Voice Conversion" by Seyed Hamidreza Mohammadi has been examined and approved by the following Examination Committee:

Alexander Kain

Associate Professor

Thesis Advisor

Xubo Song

Professor

Steven Bedrick

Assistant Professor

John Paul Hosom

Research Technologist

This dissertation is dedicated to Ava.

Acknowledgement

I would like to express my sincere gratitude to my advisor, Dr. Alexander Kain for his continuous support during my Ph. D. study and being a visionary teacher for me. In addition, I want to thank Dr. Jan van Santen for his support in my dissertation committee and wish him a healthy recovery. I also would like to thank the other members in my committee: Steven Bedrick, Xubo Song, and John-Paul Hosom for their interest in my work and their insightful discussion and comments.

My sincere thanks also goes to all of the current and former faculty at CSLU for their important input and motivation. I want to also thank my fellow students, both past and present, for making CSLU a great place to work, especially Joel Adams, Meysam Asgari, Alireza Bayesteh, Russ Beckley, Brian Bush, Shiran Dudy, Andrew Fowler, Maider Lehr, Rebecca Lunsford, Archana Machireddy, Eric Morley, Masoud Rouhizadeh, Ethan Slefridge, Golnar Sheikhshab, Brian Snider, Guillaume Thibault, and Mahsa Yarmohammadi. Thanks goes to Peter Heeman for his helpful directions and dedicating time for directing graduate students, as well as Patricia Dickerson for her great administrative support.

Finally, this dissertation would not have been possible without the love, support, and patience of my wife Mahsa. I want to also thank my parents for all their sacrifices to support me. I also want to thank my brothers Alireza and Saeed for being okay siblings. Finally, I want to thank my parent-in-laws, Reza and Golsa for their support.

This work was partially supported by NSF grants. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NSF.

Contents

List of Tables

List of Figures

Abstract

Speech Representation Learning for Voice Conversion

Seyed Hamidreza Mohammadi

Doctor of Philosophy

Center for Spoken Language Understanding, Oregon Health & Science University

Thesis Advisor: Alexander Kain

February 2019

The sound of a person's voice is an important factor in human communication. Voice Conversion (VC) is a technology that modifies a *source* speaker's speech utterance to sound as if it has been spoken by a *targ*et speaker. VC offers a number of useful applications. For example, personalizing a text-to-speech system to speak with a new voice with minimal amount of data, or mimicking the voice of another individual when dubbing a movie in another language.

In this dissertation, we consider new approaches in the design of VC systems. We propose techniques for learning speech representations with some characteristics that facilitate building systems for VC. In a first approach, we propose to learn artificially-enforced similar representations from both source and target speakers' speech features. This allows the encoding of source speaker features to a representation which can then be used to decode the target speech features. We name this architecture *joint autoencoder*. We investigate the behaviors of this model through objective and subjective evaluations.

In a second approach, we propose to learn decoupled speech representations. Specifically, we focus on decoupling speaker identity and linguistic content. The model is initially trained to decouple the representations from a multi-speaker parallel corpus. We then apply the trained model to seen or unseen speakers to compute the speaker identity representation. We can replace the speaker identity of an input utterances with that of a target speaker's to achieve VC. We name this model *decomposing autoencoder*. We investigate the effectiveness of this model through objective and subjective evaluations.

A third contribution is proposing to learn deep autoencoder models using two proposed algorithms: a *progressive training* algorithm, and a *deeply supervised* algorithm. Traditionally, greedy layer-wise training was used to training deep autoencoders. As the name suggests, each level is trained in isolation, starting from the first level towards the deepest level. Alternatively, all levels

can be trained at once. In our proposed progressive algorithm, to train the autoencoder at a certain level, we train that level in isolation, and then re-train all the previous layers and current layer at once. This process is repeated until the desired depth is reached. This algorithm can be used as a general algorithm to train deep autoencoders instead of greedy layer-wise training or training all layers at once. In our proposed deeply supervised algorithm, we propose to use an architecture in which all autoencoder levels have their loss contribute to the total loss, hence all deep autoencoder levels are trained simultaneously. This is motivated by the idea behind deeply supervised networks in which each layer has a connection to the output, hence helping the error to propagate to the lower layers of the network. Through objective evaluations, we show that the proposed algorithms learn more useful representations compared to a baseline.

Chapter 1

Introduction

1.1 Definition of voice conversion

Voice transformation (VT) refers to the various modifications one may apply to human-produced speech [261, 184]; specifically, VT aims to modify one or more aspects of the speech signal while retaining its linguistic information. *Voice conversion* (VC) is a special type of VT whose goal is to modify a speech signal uttered by a *source* speaker to sound as if it was uttered by a *target* speaker, while keeping the linguistic content unchanged [37]. In other words, VC modifies speaker-dependent characteristics of the speech signal, such as spectral and prosodic aspects, in order to modify the perceived speaker identity while keeping the speaker-independent information (linguistic content) the same. There is also another class of voice transformation called *voice morphing* where the voices of two speakers are blended to form a virtual third speaker [25]. VT approaches can be applied to solve related problems, such as changing one emotion into another [137], improving the intelligibility of speech [127], or changing whisper/murmur into speech without modifying speaker identity and linguistic content. For more information regarding applications, please see Section 1.2. In this work, we will focus on studies pertaining to VC systems, since the majority of important milestones of the VT field have been studied in the VC literature.

There are various ways to categorize VC methods. One factor is whether they require *parallel* or *non-parallel* recordings during their training phase. Parallel recordings are defined as utterances that have the same linguistic content, and only vary in the aspect that needs to be mapped (speaker identity, in the VC case) [192]. A second factor is whether they are *text-dependent* or *text-independent* [272]. Text-dependent approaches require word or phonetic transcriptions along with the recordings. These approaches may require parallel sentences recorded from both source and target speakers. For text-independent approaches, there is no transcription available, therefore these approaches require finding speech segments with similar content before building a conversion

function [269]. A third factor is based on the language that source and target speakers speak. Language-independent or *cross-language* VC assumes that source and target speakers speak in different languages [267, 313]. Finally, another category of VC systems is one-to-one versus many-to-one and many-to-many systems. *One-to-one* systems assume the availability of both source and target speaker data (either parallel, or non-parallel) and expects source speaker as input during conversion. *Many-to-one* systems are able to only use target data, and expect arbitrary speakers as input during conversion. Similarly, *many-to-many* systems can handle arbitrary speakers as input during conversion, and can convert to unseen (during training) speakers as target.

1.2 The applications of voice conversion

VT and VC techniques can be applied to solve a variety of applications. We list some of these applications in this section:

Transforming speaker identity: The typical application of VT is to transform speaker identity from one source speaker to a target speaker, which is referred to as VC [35]. For example, a high-quality VC system could be used by dubbing actors to assume the original actor's voice characteristics. VT methods can also be applied for singing voice conversion [319, 332, 44, 141].

Transforming speaking type: VT can be applied to transform the speaking type of a speaker. The goal is to retain the speaker identity but to transform emotion [101, 102, 290, 159], speaking style [187, 79], speaker accent [14], or speaker character [228], as well as de-identification [118, 16] and delexicalization [124]. Prosodic aspects are considered a more prominent factor in perceiving emotion and accent, thus some studies focus on prosodic aspects [137, 287, 133, 114, 17, 102, 159, 338, 340].

Personalizing Text-to-Speech (TTS) systems: A major application of VC is to personalize a TTS systems to new speakers, using limited amounts of training data from the desired speaker (typically the end-user if the TTS is used as an augmentative and alternative communications device) [122, 46, 265]. Another option is to create a TTS system with new emotions [137, 317, 115, 318, 152].

Speech-to-Speech translation: The goal of these systems is to translate speech spoken in one language to another language, while preserving speaker identity [336, 23]. These systems are

usually a combination of automatic speech recognition (ASR), followed by machine translation. Then, the translated sentence is synthesized using a TTS system in the destination language, followed by a cross-language VC system [49, 276, 215, 274].

Biometric voice authentication systems: VC presents a threat to speaker verification systems [224]. Some studies have reported on the relation between the two systems and the vulnerabilities that VC poses for speaker verification, along with some solutions [9, 352, 40, 348].

Speaking- and Hearing-aid devices: VT systems can potentially be used to help people with speech disorders by synthesizing more intelligible or more typical speech [127, 98, 307, 360, 5, 284, 297, 125]. VT is also applied in speaking-aid devices that use electrolarynx devices [21, 199, 200]. Similar approaches can be used to increase the intelligibility of speech, especially in noisy environments with application to increasing the performance of future hearing-aid devices [187, 145, 80]. Other applications are devices that convert murmur to speech [298, 198, 307], or whisper to speech [191, 311].

Telecommunications: VT approaches have been used to reconstruct wide-band speech from its narrowband version [222]. This can enhance speech quality without modifying existing communication networks. Spectral conversion approaches have also been successfully used for speech enhancement [193].

1.3 Problem Definition

Background and Context

Voice conversion and voice transformation share many common techniques. They have numerous useful applications as listed in the previous sub-section. There has been numerous signal-processing based, exemplar-based, statistical, and neural approaches proposed to perform voice conversion. These approaches, however, still do not produce acceptable quality for the general public. Moreover, these approaches perform mapping directly on the acoustic features. More research towards improving speech quality and speaker similarity are required. An extensive subjective evaluation was performed during the 2016 Voice Conversion Challenge (VCC), with multiple submitted systems [342]. It was concluded that "there is still a lot of work to be done in voice conversion, it is not a solved problem. Achieving both high levels of naturalness and a high degree of similarity to a target speaker — within one VC system — remains a formidable task" [342].

Scope and Relevance

This study focuses on proposing various autoencoder architectures for performing voice conversion. Autoencoders are neural-network architectures that are useful for learning representations from data. We leverage the properties of a parallel corpus to find *similar* representations from two or more voices. We enforce this similar representation learning by imposing constraints in either the cost function or the architecture of the neural network. These similar representations allow us to perform useful modifications in the representation to achieve voice transformation.

Problem Statement

The voice conversion quality and similarity is still not acceptable to a wider audience, and the mapping is typically performed directly between low-level acoustic features which might not be an appropriate domain for transformation.

Objective

The aim of this research is to propose methodologies to learn a more abstract representation domain compared to the acoustic feature domain, and perform the voice conversion mapping in that representation domain. The motivations is that these representations allow higher quality conversion, and more flexibility in mapping different voices.

Thesis statement

Methodologies that learn meaningful representations from the speech acoustic features in order to perform the mapping in the representation domain perform better than comparable baselines.

1.4 Motivation of dissertation

In this dissertation, we focus on two main tasks: one-to-one parallel VC, and many-to-many non-parallel (non-parallel in the sense that the final system does not require parallel training data on new source and target speakers, but initial parallel training data is still required for building a general-purpose model) VC. In this dissertation, we aim to find unified (joint) representations for both source and target voices, and simultaneously, reduce the distance between the learned source and target representations. The typical one-to-one VT system assumes the model receives source features as input and target features as output, and imposes some closeness criterion between the original target and the predicted target. In contrast, we propose to learn the abstract representation

for each source and target distributions, and simultaneously reduce the distance between the learned representations. We posit that the closeness criterion tries to force the model to learn a speaker-independent representation which we can use to generate either source or target features. We investigate the behavior of the predicted features using the traditional and proposed approaches and show the benefits of the proposed approach.

For solving the many-to-many non-parallel VC task, signal-processing based approaches have been proposed which find similar acoustic or phonetic units between the source and target speech corpora and align them. Then a model learns to map aligned source to target features. In this dissertation, we propose an approach which decomposes arbitrary speaker speech features to its underlying factors, namely the linguistic message and speaker identity. This disentanglement allows numerous applications. For VC, the speaker identity factor can be manipulated (e. g. replaced), so that the linguistic content stays the same but the speaker identity of the reconstructed speech features are modified. These linguistic or speaker identity vectors can be potentially also used for tasks such as speaker identification and speech recognition.

1.5 Contribution of this dissertation

In this dissertation, we focus on two main tasks: one-to-one parallel VT, and many-to-many non-parallel VC. We focus on representation learning approaches to learn useful speech representations, to be applied for modifying speaker identity.

For training deep autoencoders, we propose two training approaches: progressive training and deeply-supervised training. In progressive training, each layer is trained in isolation first, and then all the layers are jointly trained together. We start from the lowest layer and iterate until the desired depth is reached. In deeply-supervised training, we propose to use an architecture in which all autoencoder levels have their loss contribute to the total loss, hence all levels are trained simultaneously. This is motivated by the idea behind deeply supervised networks in which each layer has a connection to the output, hence helping the error to propagate to the lower layers of the network. We show that we gain significant improvements in several speech processing tasks.

In the one-to-one parallel VT task, models typically use the mean squared error between the predicted and original target features. In contrast, we propose a joint-autoencoder architecture in which the model builds two separate autoencoders for source and target speakers, and simultaneously tries to minimize the representation layer distance between them.

In this dissertation, we propose a model which is able to decompose linguistic and speaker identity factors from speech features. We propose an autoencoder which learns to decompose the

factors in the representation layer. To train this decomposing autoencoder, we propose a siamese architecture, where in addition to reconstruction costs, other costs are imposed on linguistic and speaker identity representations. We also propose a modification of this approach where we impose additional constraints on the model architecture.

The following are accepted conference and journal articles:

1. S. H. **Mohammadi**, A. Kain, Siamese Autoencoders for Speech Style Extraction and Switching Applied to Voice Identification and Conversion, **Interspeech**, 2017 [185].

2. S. H. **Mohammadi**, A. Kain, An overview of voice conversion systems, **Speech Communication**, 2017 [184].

3. S. H. **Mohammadi**, A. Kain, A Voice Conversion Mapping Function based on a Stacked Joint-Autoencoder, **Interspeech**, 2016 [183].

4. S. H. **Mohammadi**, A. Kain, Semi-supervised Training of a Voice Conversion Mapping Function using Joint-Autoencoder, **Interspeech**, 2015 [182].

5. M. S. Elyasi Langarani, J. van Santen, **S. H. Mohammadi,** A. Kain, Data-driven Foot-based Intonation Generator for Text-to-Speech Synthesis, **Interspeech,** 2015 [149].

6. S. H. **Mohammadi**, A. Kain, Voice Conversion Using Deep Neural Networks With Speaker-Independent Pre-Training, **Spoken Language Technology (SLT)**, 2014 [181].

7. S. H. **Mohammadi**, A. Kain, Transmutative Voice Conversion, **ICASSP**, 2013 [180].

8. S. H. **Mohammadi**, A. Kain, J. van Santen, Making Conversational Vowels More Clear, **Interspeech**, 2012 [187].

9. E. Morley, E. Klabbers, J. van Santen, A. Kain, **S. H. Mohammadi**, Synthetic F0 Can Effectively Convey Speaker ID in Delexicalized Speech, **Interspeech,** 2012 [190].

The following are planned submissions covering some of the contributions of the dissertation:

1. S. H. **Mohammadi**, A. Kain, Training deep autoencoders using a deeply supervised training, **to be submitted to a ML journal or conference**.

2. S. H. **Mohammadi**, A. Kain, Learning Speaker and Linguistic Encodings of Speech using Decomposing Autoencoders, **to be submitted to a ML journal or conference**.

3. S. H. **Mohammadi**, A. Kain, Joint Deep Autoencoders for High-Dimensional Regression and the Application to Voice Conversion, **to be submitted to a ML journal or conference**.

1.6 Dissertation outline

In Chapter 2, we present the literature review of VC systems. We present the methodologies, definitions, evaluation approaches, databases, and explore the challenges. In Chapter 3, we present deep autoencoders and various algorithms to train them. In Chapter 4, we present Joint Autoencoders as acoustic models, applied to one-to-one parallel VC. In Chapter 5, we present Decomposing Autoencoder which disentangle speaker-related and linguistic-related information and investigate its effect on phoneme recognition, speaker identification, and many-to-many voice conversion. Finally, we summarize and discuss the contributions in Chapter 6.1 and present possible future directions in Chapter 6.2.

Chapter 2

Literature Review

2.1 Introduction

Voice transformation refers to the various modifications one may apply to human-produced speech [261]; specifically, VT aims to modify one or more aspects of the speech signal while retaining its linguistic information. *Voice conversion* is a special type of VT whose goal is to modify a speech signal uttered by a *source* speaker to sound as if it was uttered by a *target* speaker, while keeping the linguistic content unchanged [37]. In other words, VC modifies speaker-dependent characteristics of the speech signal, such as spectral and prosodic aspects, in order to modify the perceived speaker identity while keeping the speaker-independent information (linguistic content) the same. There is also another class of voice transformations called *voice morphing* where the voices of two speakers are blended to form a virtual third speaker [25]. VT approaches can be applied to solve related problems, such as changing one emotion into another [137], improving the intelligibility of speech [127], or changing whisper/murmur into speech without modifying speaker identity and linguistic content. For more information regarding applications, please see Section 1.2. In this work, we will focus on studies pertaining to VC systems, since the majority of important milestones of the VT field have been studied in the VC literature.

An overview of a typical VC system is presented in Figure 1 [58]. In the training phase, the VC system is presented with a set of utterances recorded from the source and target speakers (the training utterances). The speech analysis and mapping feature computation steps encode the speech waveform signal into a representation that allows modification of speech properties. Source and target speakers' speech segments are aligned (with respect to time) such that segments with similar phonetic content are associated with each other. The *mapping* or *conversion* function is trained on these aligned mapping features. In the conversion phase, after computing the mapping

features from a new source speaker utterance, the features are converted using the trained conversion function. The speech features are computed from the converted features which are then used to synthesize the converted utterance waveform.

There are various ways to categorize VC methods. One factor is whether they require *parallel* or *non-parallel* recordings during their training phase. Parallel recordings are defined as utterances that have the same linguistic content, and only vary in the aspect that needs to be mapped (speaker identity, in the VC case) [192]. A second factor is whether they are *text-dependent* or *text-independent* [272]. Text-dependent approaches require word or phonetic transcriptions along with the recordings. These approaches may require parallel sentences recorded from both source and target speakers. For text-independent approaches, there is no transcription available, therefore these approaches require finding speech segments with similar content before building a conversion function [269]. A third factor is based on the language that source and target speakers speak. Language-independent or *cross-language* VC assumes that source and target speakers speak in different languages [267, 313]. Because of the differences in languages, some phonetic classes may not correspond to each other, resulting in problems during mapping. To solve this issue, a combination of non-parallel, text-independent approaches can been used. Another important factor for VC categorization is the amount of the training data that is available. Typically, for larger training data, conversion functions that memorize better are more effective; however, for smaller training data, techniques that generalize better are more preferable.

Some investigators have studied the contributions of several speech features such as of pitch, formant frequencies, spectral envelope and others to speaker individuality [173, 147]. The three most relevant factors were found to be average spectrum, formants, and the average pitch level. As a result, the majority of VC systems aim to modify short-time spectral envelopes and the pitch value. In this study, we present the spectral and prosodic mappings that have been proposed for VC in Sections 2.5 and 2.6, respectively. We also review prominent approaches for evaluating the performance of VC systems in Section 2.7.

2.2 Speech Features

As shown in Figure 2.1, in order to perform voice conversion, analysis/synthesis of the speech signal is necessary. The goal is to extract speech features that allow a good degree of modification with respect to the acoustic properties of speech. Most techniques work on the frame-level (or frame-by-frame), defined as short time segments (~20 milliseconds), in which the length of the frame is chosen so that it satisfies the assumption that the speech signal is stationary (the statistical parameters

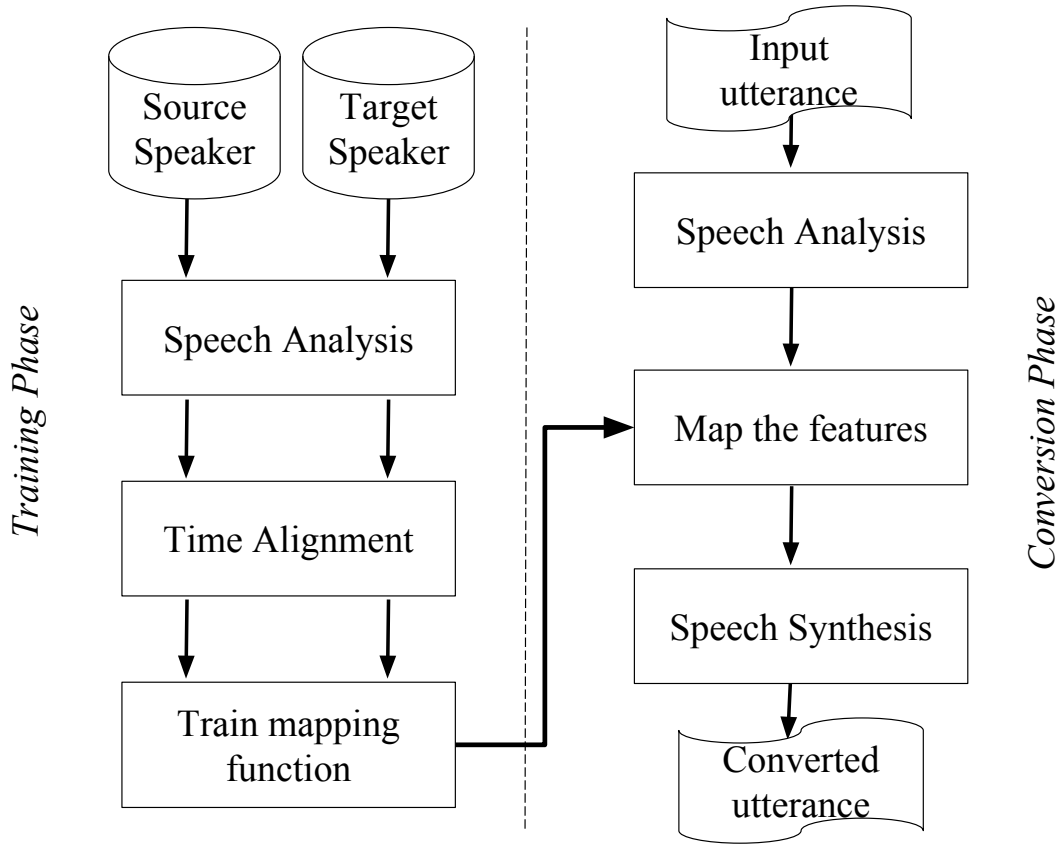


Figure 2.1: Training and conversion phases of a typical VC system [58]

of the signal over time are fixed) in that frame. The frame can be of fixed length throughout the analysis or it can have a length relative to the pitch period of the signal (pitch-synchronous analysis).

Speech models can be broadly categorized into source-filter models and signal-based models. In source-filter models, speech is modeled as a combination of an excitation or source signal (representing the vocal cords, not to be confused with the source speaker), and a spectral envelope filter (representing the vocal tract). The model assumes that speech is produced by passing an excitation signal (related to vocal cord movements and frication noise) through the vocal tract (represented by a filter), or, in other words, filtering the excitation signal with the vocal tract filter. The excitation signal and filter are assumed to be independent of each other. Two prominent filter models are commonly used: all-pole and log-spectrum filters. Linear predictive coding (LPC) is an implementation of all-pole models, and mel-log spectrum approximation (MLSA) is an implementation of log-spectrum filters [112]. SPTK is a publicly available toolkit that provides linear predictive

and MLSA analysis/synthesis [110]. When estimating the spectral envelope, the pitch periods present in the speech signal can show up as harmonics (sharp peaks and valleys) in the spectral envelope. This phenomenon can be problematic when performing any further modifications to the spectrum, since the presence of pitch information in the spectrum would fail the assumption of the independence of source signal and filter. In an attempt to alleviate the interference between signal periodicity and the spectrum, *STRAIGHT* proposes a pitch-adaptive time-frequency spectral smoothing [134], which was later extended to *TANDEM-STRAIGHT* to provide a unified computation of spectrum, fundamental frequency, and aperiodicity [135]. The advantage of a smooth spectrum is that it provides a representation that is easier to model and manipulate. *CheapTrick* and WORLD propose some improvements over TANDEM-STRAIGHT [188, 189]. The excitation signal can be modeled in various ways. A simple implementation is the pulse/noise model in which the voiced speech segments are modeled using a periodic pulse and the unvoiced speech segments are modeled using noise. More complex excitation signal models such as glottal excitation models [36, 334, 42, 232, 3], residual signals [123, 273, 370, 47, 225], mixed excitation [218, 216], and band aperiodicity [95, 32] have been used.

Signal-based analysis/synthesis approaches model the speech signal by not making any restrictive assumptions (such as the independence of source signal and filter); hence they usually have higher quality. The downside is that they are less flexible for modification. A simple analysis/synthesis technique is pitch-synchronous overlap-add (PSOLA) [196]. PSOLA uses varying frame sizes related to the fundamental frequency (F_0) to create short frames of the signal, keeping the signal in time-domain. PSOLA allows for prosodic transformations of pitch and duration. Linear Predictive PSOLA adds the ability to perform simple vocal tract modifications [326]. Harmonic plus noise models (HNM) assume that the speech signal can be decomposed into harmonics (sinusoids with frequencies relevant to pitch). HNMs generate high quality speech but they are not as flexible as source-filter models for modification, mainly because of the difficulty of dealing with phase [260]. The Ahocoder is a publicly available toolkit that provides high-quality HNM synthesis [60]. Speech signals can also be represented as a sum of non-stationary modulated sinusoids; this has shown to significantly improve the synthesized speech quality in low-resource settings [2].

2.3 Mapping Features

One might directly use speech analysis output features for training the mapping function. More commonly, the speech features are further processed to allow better representation of speech. As shown in Figure 2.1, following the speech analysis step, the mapping features are computed from

the speech features. The aim is to obtain representations that allow for more effective manipulation of the acoustic properties of speech.

2.3.1 Local Features

Local features represent speech in short-time segments. The following features are commonly utilized to represent local spectral features:

Spectral envelope: the logarithm of the magnitude spectrum can be used directly for representing the spectrum. Because of the high dimensionality of these parameters, more constrained VC mapping functions are commonly used [326, 267, 180]. The frequency scale can be warped to Mel- or Bark-scale, which are frequency scales that emphasize perceptually relevant information. Recently, due to the prevalence of neural network techniques and their ability to handle high-dimensional data, these features are becoming more popular. Spectral parameters have high inter-correlation.

Cepstrum: a spectral envelope can be represented in the cepstral domain using a finite number of coefficients computed by the Discrete Cosine Transform of the log-spectrum. Commonly, mel-cepstrum (MCEP) is used in the literature [111]. Cepstral parameters have low inter-correlation.

Line spectral frequencies (LSF): manipulating LPC coefficients may cause unstable filters, which is the reason that usually LSF coefficients are used for modification. LSFs are more related to frequency (and formant structure), and they also have better quantization and interpolation properties [221]. These properties make them more appropriate when statistical methods are used [120]. LSF parameters have high inter-correlation. These parameters are also known as Line spectral pairs (LSP).

Formants: formant frequencies and bandwidths can be used to represent a simplified version of the spectrum [178, 375, 240, 75]. They represent spectral features which are of high importance to speaker identity; however, because of their compact nature, they can result in low speech quality during more complex acoustic events in cases when the formants are modified or synthesized.

The local pitch features are typically represented by F_0, or alternatively by logarithm of F_0 which is considered to be more perceptually relevant.

2.3.2 Contextual Features

Most of the mapping functions assume frame-by-frame processing. Human speech is highly dynamic over longer segments and the frame-by-frame assumption restricts the modeling power of the mapping function. Ideally, speech segments with similar static features but different dynamic features should not be treated the same. Techniques that add contextual information to the features are proposed: appending multiple frames, appending delta (and delta-delta) features, and event-based encodings. Appending multiple frames forms a new super-vector feature [353, 30, 182] on which the mapping function is trained. This new multi-frame feature would allow the mapping function to capture the transitions within the short (but longer than a single frame) segments, since the number of neighboring frames that are appended is chosen in a way that meaningful transitional information is present within the segment. In another approach, appending delta and delta-delta features has been proposed [68]; this allows the mapping function to also consider the dynamic information in the training phase [48]. Moreover, during computing speech features from the converted features, this dynamic information can be utilized to generate a local feature trajectory that considers both static and dynamic information [303]. Event-based approaches decompose local feature sequence into event targets and event transitions to effectively model the speech transition. Temporal decomposition (TD) decomposes local feature sequence into event targets and event functions [211, 212, 210]. The event functions connect the event targets through time. Similarly, the asynchronous interpolation model (AIM) proposes to encode local feature sequence by a set of basis vectors and connection weights [126]. The connection weights connect the basis vectors through time to model feature transition. The main difficulty with the event-based approaches is to correctly identify event locations in the sequence.

Analogous to spectral parameterization, contextual information can be added to the local pitch features as well. More meaningful speech units such as syllables can be considered to encode contextual information. We present pitch parametrization and mapping approaches in more detail in Section 2.6.

In addition to these techniques that explicitly encode the speech dynamics, some mapping functions implicitly model dynamics from a local feature sequence. Examples of these implicit dynamic models are hidden Markov models (HMMs) and recurrent neural networks (RNNs). These models typically encompass a concept of *state*. The state that the model is currently in is determined by the previously seen samples in the sequence, hence allowing the model to capture context. We will mention these approaches at the end of their relevant spectral mapping subsections in Section 2.5.

2.4 Time-alignment

As shown in Figure 2.1, VC techniques commonly utilize parallel source-target feature vectors for training the mapping function between source and target features. The most common approach uses recordings of a set of *parallel* sentences (sentences including the same linguistic content) from both source and target speakers. However, the source and target speakers are likely to have different-length recordings, and have dissimilar phoneme durations within the utterance as well. Therefore, a time-alignment approach must be used to address the temporal differences. Manual or automatic phoneme transcriptions can be utilized for time alignment. Most often, a dynamic time warping (DTW) algorithm is used to compute the best time alignment between each utterance pair [1, 121], or within each phoneme pair. The final product of this step is a pair of source and target feature sequences of equal length. The DTW alignment strategy assumes that the same phonemes of the speakers have similar features (when using a particular distance measure). This assumption however is not always true and might result in sub-optimal alignments, since the speech features are typically not speaker-independent. For improving the alignment output, one can iteratively perform the alignment between the target features and the converted features (instead of source features), followed by training and conversion, until a convergence condition is satisfied. There are various methods that perform time alignment in different conditions, depending on the availability of parallel recordings, the availability of phonetic transcription, the language of the recordings, and whether the alignment is implicit in training or is performed separately. An overview of some time-alignment methods is given in Table 2.1.

More complicated approaches are required for non-parallel alignment. One set of alignment methods use transcribed, non-parallel recordings for training purposes. For alignment, a unit-selection text-to-speech (TTS) system can be built and used to synthesize the same sentences for both source and target speakers [49]. The resulting speech is completely aligned, since the duration of the phonemes can be specified to the TTS system beforehand [345]. These approaches usually require a relatively large number of training utterances and they are usually more suited for adapting an already trained parametric TTS system to new speakers/styles. These approaches, however, are text-dependent. For text-independent, non-parallel alignment, a unit-selection approach that selects units based on input source features is proposed to select the best-matching source-target feature pairs [274]. The INCA algorithm [55, 58] iteratively finds the best feature pairs between the converted source and the target utterances using a nearest neighbors algorithm, and then trains the conversion on those pairs. This process is iterated until the converted source converges and stops changing significantly.

Researchers have studied the impact of frame alignment on VC performance, specifically the situation where one frame aligns with multiple other frames (hence making the source-target feature relationship not one-to-one), and approaches to reduce the resulting effects were proposed [195, 92, 74, 82]; notably, some studies suggested to filter out the source-target training pairs that are unreliable, based on a confidence measure [316, 239].

2.5 Spectral modeling

This section discusses the mappings that are used for VC task to learn the associations between the spectral mapping features. We assume that the mapping features are aligned using one of the techniques described in Section 2.4. In addition, we assume that the training source and target speaker features are sequences of length N represented by $\mathbf{X}^{train} = [\mathbf{x}_1^{train}, \ldots, \mathbf{x}_N^{train}]$ and $\mathbf{Y}^{train} = [\mathbf{y}_1^{train}, \ldots, \mathbf{y}_N^{train}]$, respectively, where each element is a D-dimensional vector $\mathbf{x}^\top = (x_1, \ldots, x_D)$. Each element of the sequence represents the feature computed in a certain frame, where the features can be any of the mapping features described in Section 2.3. The goal is to build a feature mapping function $\mathcal{F}(\mathbf{X})$ that maps the source feature sequence to be more similar to the target speaker feature sequence, as shown in Equation 2.1. At conversion time, an unseen source feature $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_{N^{test}}]$ of length N^{test} will be passed to the function in order to predict target features,

$$\mathcal{F}(\mathbf{X}) = \hat{\mathbf{Y}} = [\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_{N^{test}}] \tag{2.1}$$

Traditionally we assume that the mappings are performed frame-by-frame, meaning that each frame is mapped independent of other frames,

$$\hat{\mathbf{y}} = \mathcal{F}(\mathbf{x}) \tag{2.2}$$

however, more recent models consider more context to go beyond frame-by-frame mapping, which are mentioned at the end of their relevant subsections.

In Figure 2.2, we devise a toy example to show the performance of some conversion techniques. We utilize 40 sentences from a male (source) and a female (target) speaker from the Voice Conversion Challenge corpus (refer to Section 2.7). We extract 24^{th}-order MCEP features and use principal component analysis (PCA) on both speaker's data to reduce the dimensionality to two for easier two-dimensional visualization. The yellow and green dots represent source and target training features. The input data, represented as magenta, is a grid over the source data distribution in the top row, and the feature sequence of a word uttered by the source speaker (excluded from

the training data) in the bottom row. The original target and converted features are represented as blue and red, respectively.

2.5.1 Codebook mapping

Vector quantization (VQ) can be used to reduce the number of source-target pairs in an optimized way [1]. This approach creates M code vectors based on hard clustering using vector quantization on source and target features separately. These code vectors are represented as $\mathbf{c}_m^x$ and $\mathbf{c}_m^y$ for source and target speakers, for $m = [1, \ldots, M]$, respectively. At conversion time, the closest centroid vector of the source codebook is found and the corresponding target codebook is selected

$$\mathcal{F}_{\mathrm{VQ}}(\mathbf{x}) = \mathbf{c}_m^y, \tag{2.3}$$

where $m = \arg_{\eta=[1,M]} \min d(\mathbf{c}_\eta^x, \mathbf{x})$. The VQ approach is compact and covers the acoustic space appropriately since a clustering approach is used to determine the codebook. However, this simple approach still has the disadvantage of generating discontinuous feature sequences. This phenomenon can be solved by using a large M but this requires a large amount of parallel-sentence utterances. The quantization error can be reduced by using a fuzzy VQ, which uses soft clustering [251, 12, 316]. For an incoming new source mapping feature, a continuous weight w_m^x is computed for each codebook based on a weight function. The mapped feature is calculated as a weighted sum of the centroid vectors

$$\mathcal{F}_{\text{fuzzy VQ}}(\mathbf{x}) = \sum_{m=1}^{M} w_m^x \mathbf{c}_m^y, \tag{2.4}$$

where $w_m^x = weight(\mathbf{c}_m^x, \mathbf{x}^{new})$. This weight function can be computed using various methods, including Euclidian distance [251], phonetic information [252], exponential decay [11], vector field smoothing [84], and statistical approaches [156]. Simple VQ is a special case of fuzzy-VQ in which only one of the vectors is assigned the weight value of one, and the rest have zero contribution.

Alternatively, to allow the model to capture more variability and reduce quantization error, a difference vector between the source and target centroids can be stored as codebook (VQ-DIFF) and added to the incoming mapping feature [172]

$$\mathcal{F}_{\text{VQ-DIFF}}(\mathbf{x}) = \mathbf{x} + (\mathbf{c}_m^y - \mathbf{c}_m^x). \tag{2.5}$$

Similar to fuzzy-VQ, a soft-clustering extension can be applied. For associating the source and

target codebooks vectors, the joint-density (JD) can be modeled, in which the source and target vectors are first stacked and then the joint codebook vectors are estimated using the clustering algorithm. As a result, the computed source-target codebook vectors will be associated together. In Figures 2.2b and 2c JDVQ and JDVQ-DIFF conversions are applied to the toy example data. As can be seen in the figure, the JDVQ-DIFF is able to generate samples that were not present in the target training data, however, JDVQ can not make this extrapolation. JDVQ exhibits high quantization error. Both JDVQ and JDVQ-DIFF are prone to generating discontinuous feature sequences.

2.5.2 Mixture of Linear Mappings

Valbret et al. [326] proposed to use linear multivariate regression (LMR) for each code vector. In this approach, the linear transformation is calculated based on a hard clustering of the source speaker space

$$\mathcal{F}_{\text{LMR}}(\mathbf{x}) = \mathbf{A}_m\mathbf{x} + \mathbf{b}_m, \tag{2.6}$$

where $m = \arg_{\eta=[1,M]} \min d(\mathbf{c}^x_\eta, \mathbf{x})$, and $\mathbf{A}_m$ and $\mathbf{b}_m$ are regression parameters. This method, however, suffers from discontinuities in the output when the clusters change between neighboring frames. To solve this issue, an idea similar to fuzzy-VQ is proposed, but for linear regression. The previous equation then changes to

$$\mathcal{F}_{\text{weighted LMR}}(\mathbf{x}) = \sum_{m=1}^{M} w^x_m(\mathbf{A}_m\mathbf{x} + \mathbf{b}_m), \tag{2.7}$$

where $w^x_m = weight(\mathbf{c}^x_m, \mathbf{x})$. Various approaches have been proposed to estimate the parameters of the mapping function. Kain and Macon [121] proposed to estimate the joint density of the source-target mapping feature vectors in an approach called joint-density Gaussian mixture model (JDGMM). A joint feature vector $\mathbf{z}_t = [\mathbf{x}^\top_t, \mathbf{y}^\top_t]^\top$ is created, and a Gaussian mixture model (GMM) is fit to the joint data. The parameters of the weighted linear mapping are estimated as

$$\mathbf{A}_m = \Sigma^{xy}_m \Sigma^{xx-1}_m, \mathbf{b}_m = \mu^y_m - \mathbf{A}_m\mu^x_m, w^x_m = P(m|\mathbf{x}^{new}), \tag{2.8}$$

where Σ^{xy}_m, Σ^{xx}_m, μ^x_m, μ^y_m, and $P(m|\mathbf{x})$ are the m^{th} training cross-covariance matrix, source covariance matrix, source mean vector, target mean vector, and conditional probability of cluster m given input $\mathbf{x}$, respectively. Stylianou et al. [262] proposed a similar formulation as Equation 2.7, however the GMM mixture components are estimated on source feature vectors only, rather than

the joint feature vectors. Additionally, instead of computing the cross-covariance matrix and the target means directly from the joint data, they are computed by solving a matrix equation to minimize the least squares error via

$$\mathbf{A}_m = \Gamma_m \Sigma_m^{xx-1}, \mathbf{b}_m = \mathbf{v}_m - \mathbf{A}_m \mu_m^x, w_m^x = P(m|\mathbf{x}^{new}), \tag{2.9}$$

where Γ and $\mathbf{v}$ are the mapping function parameters which are estimated by solving a least squares optimization problem. In the case of JDGMM, $\Gamma = \Sigma_m^{xy}$ and $\mathbf{v} = \mu_m^y$, which are computed from the joint distribution. JDGMM has the advantage of considering both the source and the target space during training, giving opportunity for more judicious allocation of individual components. Furthermore, the parameters of the conversion function can be directly estimated from the joint GMM and thus a potentially very large matrix inversion problem can be avoided. The derivation of the mapping function parameters are derived similar to Equation 2.8. GMM approaches are compared in [174]. In Figure 2.2d and 2.2e, the JDGMM conversion for $M = 8$ with diagonal covariance and $M = 4$ with full covariance matrices are applied to the toy example data, respectively. Both approaches result in smoother trajectories compared to JDVQ methods. The full covariance matrix seems to capture the distribution of the target speaker better.

One major disadvantage of GMMs is the requirement of computing covariance matrices [174]. If we assume a full covariance matrix, the number of parameters is on the order of m multiplied by the square of the dimension of the features. If we don't have sufficient data (which is usually the case in VC), the estimation might result in *over-fitting*. To overcome this issue, diagonal covariance matrices are commonly used in the literature. Due to the assumption of independence between the individual vector components, diagonal matrices might not be appropriate for some mapping features such as LSFs or the raw spectrum. To propose a middle ground between diagonal and full covariance matrices, some studies use a mixture of factor analyzers, which assumes that the covariance structure of the high-dimensional data can be represented using a small number of latent variables [324]. There also exists an extension of this approach that utilizes non-parallel a priori data [350]. Another study proposes to use partial least squares (PLS) regression in the transformation [94]. PLS is a technique that combines principles from principal component analysis (PCA) and multivariate regression (MLR), and is most useful in cases where the feature dimensionality of $\mathbf{x}_t^{train}$ and $\mathbf{y}_t^{train}$ is high and the features exhibit multicollinearity. The underlying assumption of PLS is that the observed variable $\mathbf{x}_t^{train}$ is generated by a small number of latent variables $\mathbf{r}_t$ which explain most of the variation in the target $\mathbf{y}_t^{train}$, in other words $\mathbf{x}_t^{train} = \mathbf{Q}\mathbf{r}_t + \mathbf{e}_t^x$ and $\mathbf{y}_t^{train} = \mathbf{P}\mathbf{r}_t + \mathbf{e}_t^y$, where $\mathbf{Q}$ and $\mathbf{P}$ are speaker specific transformation matrices and $\mathbf{e}_t^x$ and $\mathbf{e}_t^y$ are

residual terms. Solving $\mathbf{Q}$ and $\mathbf{P}$, and extending the model to handle multiple weighted regressions, result in the computation of regression parameters $\mathbf{A}_m$, $\mathbf{b}_m$, and w_m^x, as detailed in [94]. The approach is later extended to use kernels and dynamic information, in order to capture non-linear relationships and time-dependencies [95].

Various other approaches to estimate regression parameters have been proposed. In the Bag of Gaussian model (BGM) [234], two types of distributions are present. The basic distributions are GMMs, but the approach also uses some complex distributions to handle the samples that are far from the center of their distribution. Other approaches based on Radial Basis Functions (RBFs) [341, 214] and Support vector regression (SVR) [150, 257] have also been proposed; these use non-linear kernels (such as Gaussian or polynomial) to transform the source mapping features to a high-dimensional space, followed by one linear mapping in that space. Finally, some approaches are physically motivated mappings [362, 376] and local linear transformations [231].

One effect of over-fitting, mentioned earlier, is the presence of discontinuity in the generated features. For example, if the number of parameters is high, the converted feature sequence might be discontinuous. For solving this phenomenon, post-filtering of the posterior probabilities [33] or the generated features themselves [303, 94] has been proposed. Another known effect of GMM-based mappings is generating speech with a muffled quality. This is due to averaging features that are not fully interpolable, which results in wide formant bandwidths in the converted spectra. For example, LSF vectors can use different vector components to track the same formant, and thus averaging across such vectors produces vectors that do not represent realistic speech. This problem is also known as *over-smoothing*, since the converted spectral envelopes are typically smoothened to a degree where important spectral details become lost. The problem can be seen in Figure 2.2c where the predicted samples fall well within the probability distribution of the target features and fail to move to the edges of the distribution, thus failing to capture the variability of the target features. To solve this issue, some studies have proposed to post-process the converted features. A selection of post-processing techniques is given in Table 2.2.

Another framework for solving the VC problem is to view it as a noisy channel model [245]. In this framework, the output is computed from the conditional maximum-likelihood $\mathcal{F}_{\text{noisy-channel}}(\mathbf{x}) = \texttt{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x})$, where the conditional probability is defined using Bayes' rule $P(\mathbf{y}|\mathbf{x}) = P(\mathbf{x}|\mathbf{y})P(\mathbf{y})$. The conditional probability $P(\mathbf{x}|\mathbf{y})$ represents the channel properties and is trained on the parallel source-target data, whereas $P(\mathbf{y})$ represents the target properties and is trained on the non-parallel target speaker data. Finally, the problem reduces to decoding of the target features given the observed features, the channel properties, and the target properties. In another framework, the idea of separating style from content is explored using bilinear models [229, 230]. For the VC task, style

is the speaker identity and content is the linguistic content of the sentence. In this method, two linear mappings are performed, one for style and one for content. During conversion, the speaker identity information of the input utterance is replaced with the target speaker identity information computed during training.

In order to better model dynamics of speech, various approaches such as HMMs have been proposed [138, 48, 365, 372]. These approaches consider some context when decoding the HMM states but the final conversion is usually performed frame-by-frame. Another approach is to append dynamic features (delta and delta-delta, i. e. velocity and acceleration, respectively [68]) to the static features [48], as described in Section 2.3. A very prominent approach called maximum likelihood parameter generation (MLPG) [310] has been used for generating feature trajectory using dynamic features [303]. MLPG can be used as a post-processing step of a JDGMM mapping. It generates a sequence with maximum likelihood criterion given the static features, the dynamic features, and the variance of the features. This approach is usually coupled with GV to increase the variance of the generated feature sequence. Ideally, MLPG needs to consider the entire trajectory of an utterance to generate the target feature sequence. This property is not desirable for real-time applications. Low-delay parameter generation algorithms without GV [197] and with GV [306] have also been proposed. Recently, considering the modulation spectrum of the converted feature trajectory (as a feature correlated with over-smoothing) has been proposed, which resulted in significant quality improvements [279]. Incorporating parameter generation into the training phase itself has also been studied [369, 54]. Some data-driven post-filters are also proposed [131].

2.5.3 Neural network mapping

Another group of VC mapping approaches use artificial neural networks (ANNs). ANNs consist of multiple layers, each performing a (usually non-linear) mapping of the type $\mathbf{y} = f(\mathbf{W}\mathbf{x}+\mathbf{b})$ where $f(\cdot)$ is called the activation function that can be implemented as a sigmoid, tangent hyperbolic, rectified linear units, or linear function. A shallow (two-layered) ANN mapping can be defined as

$$F_{ANN}(\mathbf{x}) = f_2(\mathbf{W}_2 f_1(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2)), \tag{2.10}$$

where W_i, b_i, and f_i represent the weight, bias and activation function for the i^{th} layer, respectively. ANNs with more than two layers are typically called deep neural networks (DNNs) in the literature. The input and output size are usually fixed depending on the application. (For VC, the input and output size are the source and target mapping feature dimensions.) However, the size of the middle layer and activation function are chosen depending on the experiment and data distributions. The

first layer activation function is almost always non-linear and the activation function of the last layer is linear or non-linear, depending on the design. If the last layer is linear, the ANN approach can be viewed as an LMR approach, with the difference that the linear regression is applied on a data space that is mapped non-linearly from the mapping feature space, and not directly on the mapping features (similar to RBF and SVR). The weights and biases can be estimated by minimizing an objective function, such as mean squared error, perceptual error [327], or sequence error [357].

ANNs are a very powerful tool, but the training and network design is where most care needs to be exercised since the training can easily get stuck in local minima. In general, both GMMs and ANNs are universal approximators [296, 100]. The non-linearity in GMMs stems from forming the posterior-probability-weighted sum of class-based linear transformations. The non-linearity in ANNs is due to non-linear activation functions. Laskar et al. [151] compare ANN and GMM approaches in the VC framework in more detail. In Figure 2.2f, the ANN conversion for a hidden layer of size 16 is applied to the toy example data. The ANN trajectory is performing similar to JDGMM with full covariance matrix, which is expected since both are universal approximators.

The very first attempt for using ANNs utilized formant frequencies as mapping features [209], i. e. the source speaker's formant frequencies were transformed towards target speaker's formant frequencies using a ANN followed by a formant synthesizer. Later, Makki et al. [169] successfully mapped a compact representation of speech features using ANNs. A more typical approach used a three-layered ANN to map mel-cepstral features directly [43]. Various other ANN architectures have been used for VC [236, 88]: Feedforward architectures [43, 15, 162, 181, 213, 246], restricted Boltzmann machines (RBMs) and their variations [29, 351, 204, 158], joint architectures [29, 182, 183], and recurrent architectures [201, 263, 237]. More generative models such as Generative adversarial network (GAN) models [129, 130, 131, 34] and variational autoencoders [22, 103, 105, 106, 104, 186, 99, 247] are also investigated for VC. Recently, CycleGANs have been proposed for non-parallel voice conversion [129, 66, 38]. These approaches assume an A and B domain (in VC, A can be defined as source and B as target). The model is comprised of two networks: $A2B$ and $B2A$ in which $\hat{\mathbf{x}} = B2A(A2B(\mathbf{x}))$ and $\hat{\mathbf{y}} = A2B(B2A(\mathbf{y}))$. Some constraints are applied in order to force the networks to learn the intended mappings.

Traditionally, DNN weights are initialized randomly; however, it has been shown in the literature that deep architectures do not converge well due to a vanishing gradient and the likelihood of being stuck in a local minimum solution [72]. A regularization technique is typically used to solve this issue. One solution is pre-training the network. DNN training converges faster and to a better-performing solution if their initial parameter values are set via pre-training instead of

random initialization [53]. This especially important for the VC task since the amount of training data is typically smaller compared to other tasks such as ASR or TTS. Stacked RBMs are used to build speaker-dependent representations of cepstral features for source and target speakers before DNN training [202, 207, 205]. Similarly, layer-wise generative pre-training using RBMs for VC has been proposed [30, 31]. Mohammadi and Kain [181] proposed a speaker-independent pre-training of the DNN using multiple speakers other than source and target speakers using de-noising stacked autoencoders. This approach is later extended to use speakers that sound similar to source and target speakers to pre-train the DNN using joint-autoencoders [182]. In a related study, using multiple speakers as source for training the DNN was proposed [163]. Alternatively, other regularization techniques such as dropout [259] and using rectified linear units [73] have shown to be successful.

For capturing more context, Xie et al. [357] proposed a sequence error minimization instead of a frame error minimization to train a neural network. The architecture of RNNs allows the network to learn patterns over time. They implicitly model temporal behavior by considering the previous hidden layer state in addition to the current frame [203, 201, 263, 176, 160, 346]. Sequence-to-sequence (seq2seq) models have been proposed in which the source feature sequence is encoded into a small embedding and then decoded into the target feature sequence [177, 130]. Convolutional seq2seq models have also been proposed; they have shown the ability to perform conversion on not only the voice characteristics but also the pitch contour and duration of the input speech [128]. Attention seq2seq models have also been used for VC with the property of converting acoustic features and duration simultaneously [285].

2.5.4 Dictionary mapping

One of the simplest mapping functions is a look-up table that has source features as entry keys and target features as entry values. For an incoming feature, the function will look up to find the most similar key based on a distance criterion, e. g. an objective distortion measure $d(\cdot)$ similar to one described in Section 2.7.1. In other words, it will look for the nearest neighbor of the incoming source feature and select its corresponding entry value

$$\mathcal{F}_{\text{lookup}}(\mathbf{x}) = \mathbf{y}_t^{train}, \tag{2.11}$$

where $t = \arg_{\tau=[1,T]} \min d(\mathbf{x}_\tau^{train}, \mathbf{x})$. A major concern is that the similarity of the source feature does not necessarily guarantee similarity in neighboring target features. This phenomenon will

cause discontinuities between the generated target parameter sequence. One approach to overcome the discontinuity of the target feature sequence is to assign a weight to all target feature vectors (computed based on the new source feature vector), which will generate a smoother feature sequence. This category of approaches is called *exemplar-based* VC in the literature [356, 6, 355, 4, 373] and the mapping function is given by

$$\mathcal{F}_{\text{exemplar}}(\mathbf{x}) = \sum_{t=1}^{T} w_t^x \mathbf{y}_t^{train}, \tag{2.12}$$

with $w_t^x = \omega(\mathbf{x}_t^{train}, \mathbf{x})$, where $\omega(\cdot)$ can potentially be any distortion measure. A generic objective distortion measure might result in over-smoothing, since many frames may be assigned non-zero weights and will thus be averaged (unless the mapping features are completely interpolable). Commonly, non-negative matrix factorization (NMF) techniques have been used to compute sparse weights. The goal of NMF is to compute an activation matrix $\mathbf{H}$ which represents how well we can reconstruct $\mathbf{x}$ by a non-negative weighted addition of all $\mathbf{x}_t^{train}$ vectors, such that $\mathbf{X} = \mathbf{X}^{train}\mathbf{H}$. The activation matrix $\mathbf{H}$ is calculated iteratively [356]. NMF computes a non-negative weight for each entry in the table, which results in the mapping function

$$\mathcal{F}_{\text{NMF}}(\mathbf{X}) = \mathbf{Y}^{train}\mathbf{H}. \tag{2.13}$$

This relatively sparse weighting over all vectors results in smooth generated feature sequences while reducing over-smoothing. This approach however has the disadvantage of computational complexity, which might not be suitable for some applications. To address this issue, computing the activation matrix in a more compact dimension has been proposed [356]. The NMF methods are also inherently well-suited for noisy environments [280, 281, 170]. Several other extensions of NMF approaches have been proposed, such as mapping the activation matrix [8], many-to-many VC [7], including contextual information [354, 356, 20], and local linear embeddings [347].

Another approach to combat discontinuities in the generated features is to take the similarity of the target feature sequence into consideration by using a unit-selection (US) paradigm. US approaches make use of a target cost (similar to a table look-up distortion measure) and a concatenation cost (to ensure the neighboring target features are most similar to each other). Since the units are frames, this method is also referred to as frame-selection (FS). The goal is to find the best sequence of indices of training target vectors $\mathbf{S} = [s_1, \ldots, s_N]$ which minimizes the following

cost function [248, 322, 157]:

$$\begin{aligned}\mathcal{F}_{\text{FS}}\left(\mathbf{X}\right) = \quad & \arg_{\mathbf{S}=[s_1,\ldots,s_{N^{test}}]}\min \\ & \textstyle\sum_{n=1}^{N^{test}} \alpha \cdot target(\mathbf{x}_{s_n}, \mathbf{x}_n^{new}) + \\ & (1-\alpha)\cdot concatenation(\mathbf{y}_{s_n}, \mathbf{y}_{s_{n-1}})\end{aligned} \tag{2.14}$$

where α is used for adjusting the tradeoff between fitting accuracy and the spectral continuity criterion. Since there is an exponential number of permutation of index sequences, a dynamic programming approach such as Viterbi is used to find the optimal target sequence. This can be used for aligning frames before any other type of training, or directly used as a mapping function.

The US/FS approach can be adjusted to create text-independent, non-parallel VC systems [274, 275]. In this variation, a vector $\tilde{\mathbf{x}}_n^{cmp}$ is compared to a target training vector in the dictionary to compute the target cost

$$\begin{aligned}\mathcal{F}_{\text{US}}\left(\mathbf{X}\right) = \quad & \arg_{\mathbf{S}=[s_1,\ldots,s_{N^{test}}]}\min = \\ & \textstyle\sum_{n=1}^{N^{test}} \alpha \cdot target(\mathbf{y}_{s_n}, \tilde{\mathbf{x}}_n^{cmp}) + \\ & (1-\alpha)\cdot concatenation(\mathbf{y}_{s_n}, \mathbf{y}_{s_{n-1}})\end{aligned} \tag{2.15}$$

where $\tilde{\mathbf{x}}_n^{cmp}$ is either the input source vector (in the absence of any parallel data) [274] or a naive conversion to target (in the presence of real or artificial parallel data) [45]. As mentioned in Section 2.4, these techniques can be used for parallelizing the training data as well.

Combinations and variants of US/FS approaches combined with other mapping approaches have been proposed, such as: dictionary mapping [67], codebook mapping [138, 64], frequency warping [254, 321], GMM mapping [48], segmental GMM [81], k-histogram [323], exemplar-based VC [354], and grid-based approximation [20]. These approaches have some limitations, specifically they can generate discontinuous features. Helander et al. [90] studied the coverage of speech features when using FS as a mapping for VC, and concluded that a small number of training utterances (which is typical in VC tasks) is not adequate for representing the speaker space. Sparse representation of phonetic features for voice conversion with and without parallel data is also investigated [39].

2.5.5 Frequency warping mappings

The estimation of linear regression parameters described in Section 2.5.2 is typically unconstrained; this can lead to over-fitting. There exist a class of constrained mapping methods which are physically motivated [376]. One common motivation is that two different speakers have different formant

frequencies and bandwidths, and different energies in each frequency band. Thus, for conversion, a constrained mapping only allows manipulation of formant location/bandwidths and energy in certain frequency bands. This reduces over-fitting, while allowing the use of high-dimensional mapping features, which is beneficial since vocoders that utilize high-dimensional speech features (e. g. harmonic vocoders) usually have higher speech quality compared to more compact vocoders (e. g. LSF vocoders). The motivation behind the frequency warping (FW) methods is that a mapping of a source speaker spectrum to a target speaker spectrum can be performed by warping the frequency axis, to adjust the location and bandwidth of the formants, and applying amplitude scaling, to adjust the energy in each frequency bands [56, 59, 78]; this approach is more physically interpretable than an unconstrained mapping. Although these approaches can be implemented as constrained linear transformations (for certain features, such as cepstral features), we dedicate a separate chapter to them due to their slightly different motivation.

In a first attempt, Valbret et al. [325] proposed to warp the frequency axis based on precomputed warping functions between source and target, using log-spectral features. The source speaker spectral tilt is subtracted before warping and the target speaker spectral tilt is added after warping. Some studies directly model and manipulate formant frequencies and bandwidths [178, 312, 253, 76] so that they match the target formants, as shown in Figure 2.3. Maeda et al. [168] proposed to cluster the acoustic space into different classes (similar to VQ) and perform a non-linear frequency warping on the STRAIGHT spectrum for each class. Later, Sündermann et al. [267] studied various vocal tract length normalization (VTLN) approaches that were used in ASR to perform VC, including piecewise linear, power, quadratic, and bilinear VTLN functions. Erro et al. [61] extended this VTLN approach to multiple classes and proposed an iterative algorithm to estimate the VTLN parameters. Přibilová and Přibil [233] experimented with various linear and non-linear warping functions, with application to TTS adaptation. Erro and Moreno [56] proposed weighted frequency warping (WFW) to perform a piecewise linear frequency warping in each mixture components of a GMM, weighted by the posterior probability. It is worth noting that they used a pitch-asynchronous harmonic model (a high-quality vocoder) and performed phase manipulation to achieve high quality speech. Toda et al. [299] proposed to convert the source spectrum using a GMM and then warp the source spectrum to be similar to the converted spectrum with the aim of keeping the spectral details intact.

Other than the formant frequency locations, the average energy of the spectral bands is also an important factor in speaker individuality. Previously, this has been partly taken care of by subtracting source spectral tilt before frequency warping and adding the target spectral tilt. In an extension of WFW work, it was shown that in addition to frequency warping, an energy correction

filter is required to increase the flexibility of the mapping function [59]. Tamura et al. [283] proposed a simpler amplitude scaling by adding a *shift* value to the converted vector. In another extensive study, *amplitude scaling* in addition to frequency warping was proposed to add more degrees of freedom to the mapping [77, 78].

Some frequency warping functions can be reformulated as a weighted linear mapping approach [227]. The linear mappings are usually constrained, so that the mapping is less prone to over-fitting. However, the constraints will limit the ability to mimic very different voices. Erro et al. [62] studied this idea using bilinear warping function (similar to the VTLN approach) and constrained amplitude scaling, and extended it [63].

Numerous extensions of the FW approach have been proposed, such as in combination with GMMs [57, 376, 180, 293], dictionary-based methods [254, 321, 292], maximizing spectral correlation [291], equalizing formant frequencies as pre-processing step [179], and combination with exemplar-based approach [294].

2.5.6 Adaptation techniques

In this section, we describe the techniques that are used when only limited or non-parallel training data are available. These approaches typically utilize the mappings or models learned from some pre-defined set of speakers to aid the training of the conversion mapping. Most of these approaches use a mixture of linear mappings, however, the ideas could be generalized to other approaches such as neural networks.

Adaptation techniques perform mean adaptation on the means of GMM mixture components that are trained on the source speaker [33] to move the GMM means towards the target speaker's probability distribution. Mouchtaris et al. [194] proposed an adaptation technique for non-parallel VC, in which a JDGMM is trained on a pre-defined set of source and target speakers that have parallel recordings. For building the mapping function using non-parallel recordings, the means and covariances of the GMMs are adapted to the new source and target speakers. In a neural network-based approach, a semi-supervised learning approach is proposed in which first speakers that sound similar to the source and target speakers are used for training the network, and then the pre-trained neural network is further trained using the source and target speaker data [182]. In another study, an adaptive RBM approach was proposed in which it is assumed that features are produced from a model where phonological information and speaker-related information are defined explicitly. During conversion, the phonetic and speaker information are separated and the speaker information is replaced with that of the target's [206].

Another scheme for voice conversion is to utilize the conversions built on multiple pre-stored speakers (different from the target speaker) to create the mapping function. A first attempt called speaker interpolation generates the target features using a weighted linear addition (interpolation) of multiple conversions towards multiple other pre-defined target speakers, by minimizing the difference between the target features and the converted features [116, 117]. The interpolation coefficients are estimated using only one word from the target speaker.

The eigenvoice VC (EVC) approach represents the joint probability density similar to the conventional GMM, except that the target means are defined as [302, 217]

$$\mu_m^y = \mathbf{G}_m w + \mathbf{g}_m \tag{2.16}$$

where $\mathbf{g}_m$ is the bias vector and the matrix $\mathbf{G}_m = [\mathbf{g}_m^1, \ldots, \mathbf{g}_m^J]$ consists of basis vectors $\mathbf{g}_m^j$ for the m^{th} mixture. The total number of basis vectors is J and the target speaker identity is controlled with the J-dimensional weight vector $w = [w^1, \ldots, w^J]^\top$. For a given target speaker, a weight is computed and assigned to each eigenvoice; the weight represents the eigenvoice's contribution to generating features [302, 217]. In the traditional eigenvoice approach, weights are estimated during training and are fixed during conversion. For lowering the computational cost, a multistep approach has been proposed [171]. For further improving the robustness of this approach to the amount of adaptation data, a maximum-a-posteriori adaptation approach has also been proposed [286]. The eigenvoice approach has also been extended to *many-to-one* VC, where the target speaker is always the same but the source speaker can be an arbitrary speaker with minimal adaptation data [304]. Finally, *one-to-many* eigenvoice VC based on a tensor representation of the space of all speakers has been proposed [244]. *Many-to-many* conversion has also been proposed in which the goal is to perform a conversion using an arbitrary source speaker to an arbitrary target speaker with minimal parallel [219] and non-parallel data [220].

2.5.7 Other mappings

Various other mapping approaches have been proposed. The K-histogram approach is a non-parametric approach which defines the mapping via the cumulative distribution function (CDF) of the source followed by an inverse CDF of the target [323]

$$\mathcal{F}_{\text{K-Histogram}}(\mathbf{x}) = CDF_y^{-1}(CDF_x(\mathbf{x})) \tag{2.17}$$

A Gaussian processes (GP) approach has also beens proposed [226, 359]. GPs are kernel-based,

non-parametric approaches that can be viewed as distribution over functions, which relieves the need to specify the parametric form beforehand. For example, it is possible to define how to describe the mean and covariance functions [226]. Another non-parametric approach based on topological maps has been proposed which estimates the joint distribution of the spectral space of source and target speakers [242, 320]. The topological map is a type of a neural network where each node is topologically located on a 2D map in a grid-like fashion. In the training step, the value of these nodes are learned. For each node in the source speaker map, a corresponding node in the target speaker map is computed. This correspondence is used to map an incoming source vector to a target vector. This approach has some similarities to the VQ method.

2.6 Prosodic modeling

Most of the VC literature focuses on mapping spectral features, despite the fact that prosodic aspects (pitch, duration, spectral balance, energy) are also important for speaker identity [96, 190]. For modeling duration, a global speaking rate adjustment is not sufficient since it has been observed that phoneme durations differ somewhat arbitrarily between source and target speakers [13]. Modeling duration using decision trees [232] and duration-embedded HMMs has been studied [345].

The most common method to transform pitch is to globally match the average and standard deviation of the pitch contour. Pitch can be converted by mapping the log-scaled F_0 using a linear transformation

$$\hat{F_0^y} = \frac{\sigma^y}{\sigma^x}(F_0^x - \mu^x) + \mu^y \tag{2.18}$$

where μ and σ represent mean and standard deviation of the log-scaled F_0 [27]. Several studies have looked into modeling F_0 and spectral features jointly [52, 83, 358]; this has shown improvements for both spectral and F_0 conversions. Conversely, predicting pitch values from the target speaker spectrum using a GMM has also been studied [51].

When we use simple linear mapping techniques, such as globally changing the speaking rate or adjusting the pitch mean and variance, the supra-segmental information is not modified effectively. Prosody modeling is a complex problem that depends on linguistic and semantic information. As an example, the emphasis that speakers put on certain speech units (such as words) does not necessarily have a similar pattern for other speakers depending on the context and high level information. In VC tasks, this high level information is typically not available. ASR can be used to automatically compute textual information, but the error that it is likely to introduce

may become a detrimental factor for prosodic mapping performance. Pitch modeling for VC has been studied on different acoustic/linguistic levels: frame-level, syllable-level, and utterance-level. Moreover, various pitch representations have been used, such as F_0 contour, the discrete cosine transform (DCT) of the F_0 contour, the Wavelet transformation of the F_0 contour, and other compact parameterizations of the F_0 contour [166]. In order to model the dynamics of the pitch contour in frame-level representations, mapping F_0 using multi-space probability distribution (MSD) HMMs [366] and LSTM networks [32] have been proposed. Syllable-level representations model the pitch movements at the syllable level, which is a more meaningful representation for modeling pitch events. The most prominent pitch conversion approaches for VC are presented in Table 2.3. Some studies investigate some of these approaches in more detail [349, 255, 256] .

2.7 Performance evaluation

When evaluating the performance of VC systems several aspects can be evaluated:

Speaker similarity: Answers the question of "How similar is the converted speech to the target?". This is also known as conversion accuracy or speaker individuality.

Speech quality: This describes the quality of the generated speech with respect to naturalness and audible artifacts.

Speech intelligibility: Assesses the intelligibility of the generated speech. This is a lesser-studied aspect in the VC literature

In experimental voice conversion evaluations, a distinction is often made between intra-gender conversion (female-to-female or male-to-male) and inter-gender conversion (female-to-male or male-to-female).

A standard corpus for VC evaluation does not exist. Several databases have been used for VC including TIMIT [69], VOICES [120], CMU-Arctic [144], MOCHA [344], and the MSRA mandarin corpus [370]. Recently, the VC Challenge (VCC) 2016 prepared a standard dataset for a VC task, which has the potential to become the standard for VC studies [308]. More recently, the VC Challenge (VCC) 2018 prepared standard parallel and non-parallel datasets for VC tasks [165]. The VCtools available in the Festvox toolkit [10] can be used for implementing baseline VC techniques such as GMM and MLPG/GV processing. Another useful toolbox is SPROCKET which uses GMM and diff approach [140].

It has been shown that the performance of the system depends on the selection of source speaker. Turk and Arslan [315] has studied the problem of automatic source speaker ("donor")

selection from a set of available speakers that will result in the best quality output for a specific target speaker's voice. This problem is also studied by proposing a selection measure [84, 85].

In the following subsections, we study the objective and subjective measures used for evaluating VC performance.

2.7.1 Objective evaluation

For evaluating VC performance objectively, a parallel-sentence corpus is required. First, the conversion and the associated target utterances have to be time-aligned. The difference between the converted speech and target can then be calculated using various general spectral difference measures. An example is the log-spectral distortion (in dB), which can be computed on unwarped, or warped (using the mel or Bark scale) spectra [262]. The most prominent measure used in the VC literature is the mel-cepstrum distance (mel-CD), also measured in dB

$$\text{mel-CD}(\mathbf{y}, \hat{\mathbf{y}}) = (10/\ln 10)\sqrt{2(\mathbf{y} - \hat{\mathbf{y}})^\top (\mathbf{y} - \hat{\mathbf{y}})} \tag{2.19}$$

where $\mathbf{y}$ and $\hat{\mathbf{y}}$ are target and converted MCEP feature vectors, respectively.

The mel-CD is suitable for evaluating preliminary experiments, defining training criterions, and validation purposes, but not for evaluating the final system regarding quality due to the low correlation with human perception [268]. Other objective speech quality assessment techniques exist [243]. These measures aim to have higher correlation with human judgment. Recently, an automatic voice conversion evaluation strategy was proposed, wherein both speech quality and speaker similarity were automatically computed [107]. The speaker similarity score was computed using a speaker verification method. These scores were shown to have higher correlation with subjective scores. However, optimizing mapping functions based on these criterions is more difficult, due to their the complex mathematical formulation.

2.7.2 Subjective Evaluation

Unfortunately, objective evaluations do not necessarily correspond to human judgments. Thus, in most studies, subjective evaluations are performed; during such evaluations human listeners asses the performance of the VC system. The listeners should ideally perform their task in ideal listening environments; however, statistical requirements often necessitate a large number of listeners. Therefore, listeners are often hired that perform the task through a crowd-sourcing website such as Amazon Mechanical Turk (AMT).

The mean opinion score (MOS) test is an evaluation using 5-scale grading. Both the speech quality and similarity to the target voice can be evaluated. The grades are as follows: 5=excellent, 4=good, 3=fair, 2=poor, 1=bad. The project TC-STAR proposes a standard perceptual MOS test as a measure of both quality and similarity [276].

The comparative MOS (CMOS) can also be used to directly compare the speech quality of two VC techniques. The listener is asked to choose the better sounding utterance. The measure is computed as the percentage where each techniques is selected over the other. The grading can also be 5-scale as follows: 5=definitely better, 4=better, 3=same, 2=worse, 1=definitely worse. This would give a good indication of any improvements. However, the absolute quality score is not calculated, making it difficult to judge the closeness to ideal quality (natural speech).

The ABX test is often used in comparing similarity between converted and target utterances. In this test, the listeners hears a pair of utterances A and B, followed by hearing a given utterance X, and have to decide is whether X is closer to A or B. The A and B utterances are uttered by source and target speakers but the ordering that the listener hears them is randomized. The measure is computed as the percentage of correct assignment of X to the target speaker. The main problem with interpreting ABX scores is that the subjects do not have the option to answer that the sentence X is not similar to either A or B [167]. For example, given A="mosquito", B="zebra", X="horse", subjects may be forced to equate B with X; however, B is still very dissimilar from X.

The ABX test can compare two VC techniques directly by setting X, A, and B to the target utterance, first VC, and second VC. This measure is computed for each VC technique as the percentage of the utterances for which that technique has been chosen as closer to the target utterance. The MOS and ABX scores of various VC techniques have been published [167].

Another technique for testing similarity is to do use the CMOS for same-different testing [120]. In this test, listeners hear two stimuli A and B with *different* content, and were then asked to indicate wether they thought that A and B were spoken by the *same*, or by two *different* speakers, using a five-point scale comprised of $+2$ (definitely same), $+1$ (probably same), 0 (unsure), -1 (probably different), and -2 (definitely different). One stimulus is the converted sample and the other is a *reference* speaker. Half of all stimuli pairs are created with the reference speaker identical to the target speaker of the conversion (the "same" condition); the other half were created with the reference speaker being of the same gender, but *not* identical to the target speaker of the conversion (the "different" condition). There has to be careful consideration in picking the proper speaker for the different condition.

Finally, the Multiple Stimuli with Hidden Reference and Anchor (MUSHRA) test has been proposed to evaluate the speech quality of multiple stimuli. In this test, the subject is presented

with a reference stimulus and multiple choices of test audio (stimuli), which they can listen to as many times as they want. The subjects are asked to score the stimuli according to a 5-scale score. This test is especially useful if one wants to test multiple system outputs in regards to speech quality.

As with all subjective testing, there is a lot of variability in the responses and it is highly recommended to perform proper significant testing on any subjective scores to show the reliability of improvements over baseline approaches. For crowd-sourcing experiments, it is best to incorporate certain sanity checks to exclude listeners that are performing below a minimum performance threshold or inconsistently. A possible implementation of these recommendations is to include obviously good/bad stimuli in the experiment, and to duplicate a small percentage of trials.

An extensive subjective evaluation was performed during the 2016 VCC, with multiple submitted systems [342]. It was concluded that "there is still a lot of work to be done in voice conversion, it is not a solved problem. Achieving both high levels of naturalness and a high degree of similarity to a target speaker — within one VC system — remains a formidable task" [342]. The average quality MOS score was about 3.2 for top submissions. The similarity average score was around 70% correctly identified as target for top submissions. Due to the high number of entries, techniques to compare and visualize the high number of stimuli, such as multidimensional scaling, were utilized [342, 343].

method	**parallel recording**	**phonetic transcription**	**cross-language**	**implicit in training**
DTW [1]	yes	no	no	no
DTW including phonetics [121]	yes	yes	no	no
Forced alignment [13, 364]	yes	forced alignment	no	no
Time Sequence Matching [208]	yes	no	no	yes
TTS with same duration [49, 345]	no	yes	no	no
ASR-TTS with same duration [363, 288]	no	ASR	no	no
Model Alignment [371]	no	no	yes	yes
Unit-selection alignment [13, 270, 55, 271]	no	no	yes	no
Iterative (INCA) [55, 58]	no	no	yes	no
Unit-selection VC [274, 275]	no	no	yes	yes
Model Adaptation [194, 155]	no	no	no	yes
Inherent in model [86, 106, 186, 99, 247, 129, 158, 295]	no	no	yes	yes

Table 2.1: Overview of time-alignment methods for VC

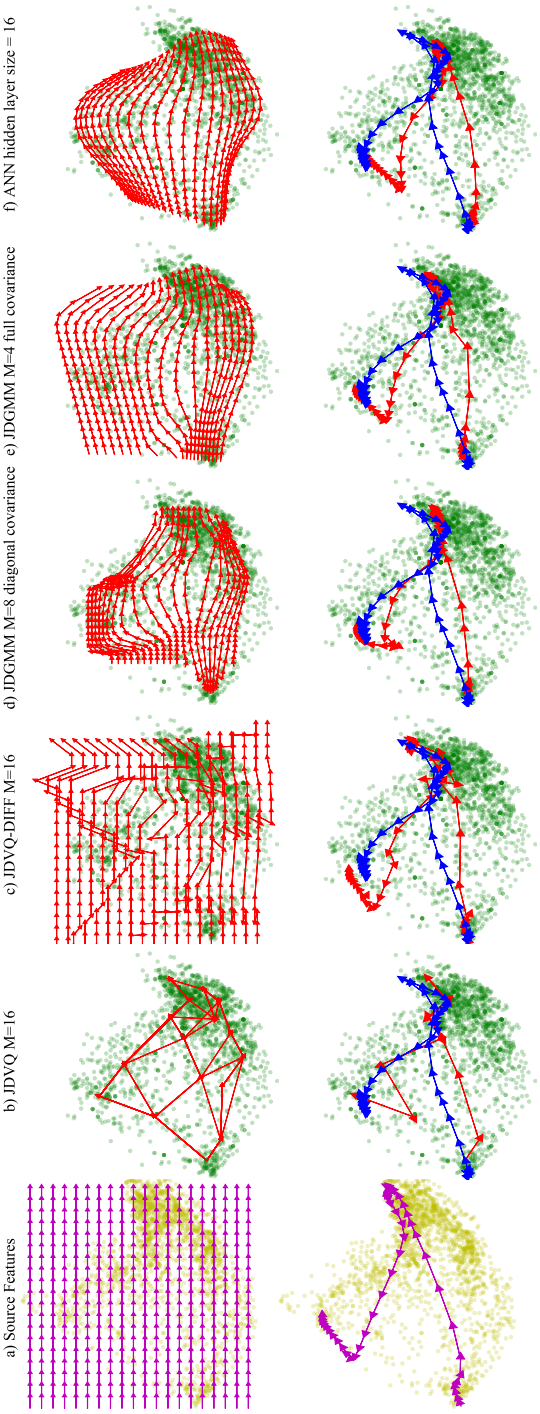


Figure 2.2: A toy example comparing JDVQ, JDVQ-DIFF, JDGMM, and ANN. The x- and y-axis are first and second dimensions of PCA, respectively. Color codes for source, target, input, original target, and converted samples are represented as yellow, green, magenta, blue, and red, respectively. The top row shows an example with a grid as input and the bottom row shows an example with a real speech trajectory as input.

Method	Description
Global Variance(GV) [301, 19, 109]	Adjusts the variance of generated features to match that of target's
ML parameter generation [303]	Maximizes the likelihood during parameter generation using dynamic features
MMI parameter generation [108]	Maximizes the mutual information during parameter generation using dynamic features
Modulation Spectrum [278]	Adjusts the spectral shape of the generated features
Monte Carlo [93]	Minimizing the conversion error and the sequence smoothness together
L2-norm [258]	Sharpens the formant peaks in spectrum
Error Compensation [333]	Models error and compensate for it
Residual addition [132]	Maps the envelope residual and adds it to the GMM-generated spectrum

Table 2.2: Post-processing techniques for reducing the over-smoothing

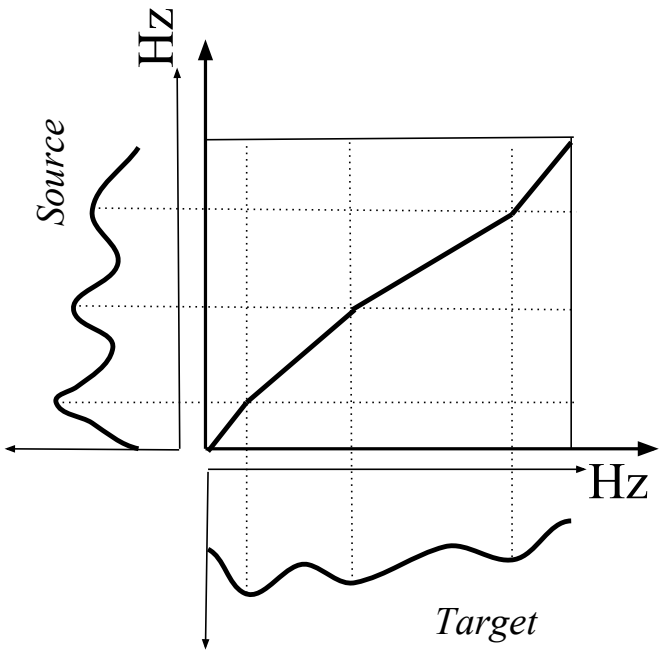


Figure 2.3: Frequency warping function based on source/target formant frequencies

Method	**level**	**pitch representation**	**other info**	**mapping function**
Mean and variance matching [27]	frame-level	F_0 contour	-	linear
Predicting from spectrum [51]	frame-level	F_0 contour	spectrum	weighted linear
Joint modeling with spectrum[52, 83, 358]	frame-level	F_0 contour	spectrum	weighted linear
Histogram equalization [349]	frame-level	F_0 contour	-	histogram equalization
MSD-HMM [366]	frame-level	F_0 contour	spectrum	weighted linear
LSTM [32, 176]	frame-level	F_0 contour	spectrum	LSTM
Syllable-based codebook [238]	syllable-level	F_0 contour	syllable boundary	codebook mapping
Syllable-based MLLR [164]	syllable-level	F_0 contour	syllable boundary	MLLR adaptation
Syllable-based CART [97]	syllable-level	DCT	syllable boundary	CART
Syllable-based weighted linear [329]	syllable-level	DCT	syllable boundary	weighted linear
Hierarchical modeling of F0 [249]	utterance-level	Wavelet transform [277]	-	KPLS [95]
Contour codebook + DTW [27, 113]	utterance-level	F_0 contour	-	codebook mapping
Weighting contours [314, 113]	utterance-level	F_0 contour	-	weighting codebooks
SHLF parametrization [71]	utterance-level	Patterson [223]	-	piecewise linear
OSV parametrization [26]	utterance-level	Offset, Slope and Variance	-	linear

Table 2.3: An overview of pitch mapping methods for VC

Chapter 3

Deep Autoencoders

3.1 Introduction

Artificial neural network (ANN) architectures used for regression tasks typically utilize a feed-forward architecture to learn and map input features to output features. Specifically, let *input* feature matrix $\mathbf{X}_{M\times D_x} = [\mathbf{x}_1, \ldots, \mathbf{x}_M]^\top$, where $\mathbf{x} = [x_1, \ldots, x_{D_x}]^\top$, represent M examples of D_x-dimensional input feature vectors, and *output* feature matrix $\mathbf{Y}_{M\times D_y} = [\mathbf{y}_1, \ldots, \mathbf{y}_M]^\top$, where $\mathbf{y} = [y_1, \ldots, y_{D_y}]^\top$, represent the *associated* (example-by-example) output feature vectors. A K-layer, fully-connected deep neural network (DNN) mapping of input $\mathbf{x}$ is a succession of (non-)linear transformations

$$\hat{\mathbf{y}} = \mathcal{F}_{\text{DNN}}(\mathbf{x}) = (f^K \circ f^{K-1} \circ \cdots \circ f^1)(\mathbf{x}) \tag{3.1}$$

where $\circ$ represents function composition, $f^k = s^k(\mathbf{W}^k\mathbf{x}+\mathbf{b}^k)$ represents one layer, and $\mathbf{W}^k$, $\mathbf{b}^k$, and s^k represent the weight matrix, bias vector, and activation function for the k^{th} layer, respectively. The model parameters $\Theta = \{\mathbf{W}^k, \mathbf{b}^k\}_{k=[1,\ldots,K]}$ are learned by minimizing a loss function $L(\mathbf{y}, \hat{\mathbf{y}})$ during training.

An autoencoder is a type of artificial neural network typically used for unsupervised learning of efficient encodings (a.k.a. representations) [18]. These encodings can be used for dimensionality reduction or for simplifying the feature distribution. A basic AE *encodes* an input $\mathbf{x}$ using a hidden representation, $\mathbf{h} = f(\mathbf{x}) = s_{\text{enc}}(\mathbf{W}\mathbf{x}+\mathbf{b})$, and then reconstructs the input by *decoding* the hidden representation via $g(\mathbf{h}) = s_{\text{dec}}(\mathbf{V}\mathbf{h}+\mathbf{c})$, and thus

$$\hat{\mathbf{x}} = \mathcal{F}_{\text{AE}}(\mathbf{x}) = (g \circ f)(\mathbf{x}) \tag{3.2}$$

where $\mathbf{W}$, $\mathbf{b}$, s_{enc} represent the encoding weight matrix, bias vector, and activation function, respectively; analogously, $\mathbf{V}$, $\mathbf{c}$, and s_{dec} are the decoding weight matrix, bias vector, and activation

function. An autoencoder architecture is depicted in Figure 3.1. The decoding weight $\mathbf{V}$ can optionally be set to the transpose of the encoding weight $\mathbf{W}^\top$. Model parameters are initialized with random values. Minimizing the reconstruction loss, $L(\mathbf{x}, \hat{\mathbf{x}})$, forces the network to learn useful representation patterns during training. To assist in learning, the input to the network is corrupted with noise during training [18]. In this study, we use the mean-squared error (MSE) loss

$$L(\mathbf{x}, \hat{\mathbf{x}}) = (\mathbf{x} - \hat{\mathbf{x}})^2. \tag{3.3}$$

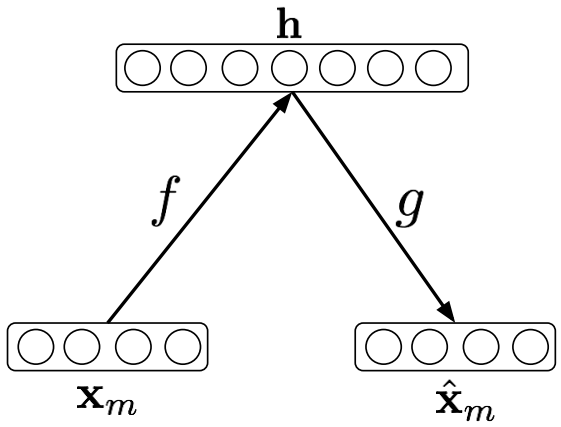


Figure 3.1: Autoencoder architecture

3.2 Formulation

An AE can be constructed to contain two more more levels of hidden representations; this type of architecture is called a deep autoencoder (DAE), and its depth is defined by the number of levels K:

$$\hat{\mathbf{x}} = \mathcal{F}_{\text{DAE}}(\mathbf{x}) = (g^1 \circ g^2 \circ \cdots \circ g^K \circ f^K \circ f^{K-1} \circ \cdots \circ f^1)(\mathbf{x}) \tag{3.4}$$

The parameters of the model are represented by $\Theta = \{\mathbf{W}^k, \mathbf{b}^k, \mathbf{c}^k\}_{k=[1,\ldots,K]}$. DAEs have been shown to learn more meaningful encodings, as with most deep neural network architectures [153]. We study the training of the architecture in various ways: greedy (GRD), simultaneous (SML), progressive (PRG), and deeply supervised (DPS). The implementations for greedy training, simultaneous training, and the proposed progressive training are shown in Algorithm 3.1.

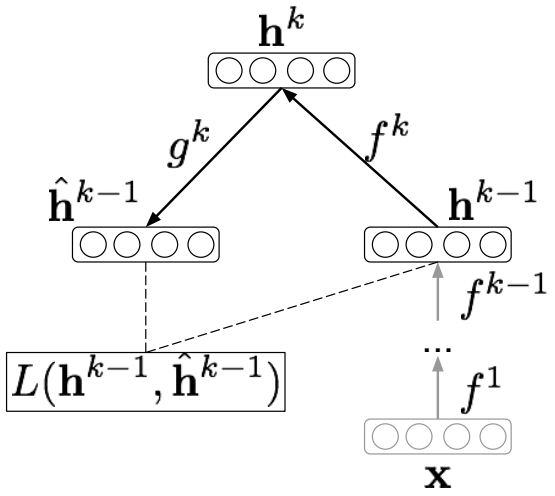


Figure 3.2: Deep autoencoder greedy training.

3.2.1 Greedy training

Training a deep architecture is also challenging, due to the vanishing gradient problem, being prone to local minima, and other issues [72]. As a result, a training method using stacked autoencoders (SAE) has been proposed, in which the first level AE is trained, the input is encoded, and then the next-level AE is trained on the encoded input [335]. This process is iterated until the desired depth is reached. We refer to this training method as "greedy" and is depicted in Figure 3.2. The k^{th} level hidden encodings are represented using $\mathbf{h}^k$.

3.2.2 Simultaneous training

Recently, Zhou et al. [374] proposed that training all levels simultaneously can achieve a better representation, given enough iterations and proper regularization. In their proposed architecture, the network is trained similarly to a one-level AE by minimizing the reconstruction criterion, with the difference that the encoder and decoder parts are deep rather than a single level. We refer to this training method as "simultaneous" and is depicted in Figure 3.3. The network parameters can be initialized with random values, or from a DAE previously trained with the greedy method. Similar to the one-level AE, SML uses a single reconstruction loss $L(\mathbf{x}, \hat{\mathbf{x}})$ between the original input and the reconstructed input.

Algorithm 3.1 Deep autoencoder training algorithm.

procedure `Train-DAE`
Input: $\mathbf{u}$, Θ', *type* ▷ types: GRD, SML, PRG
Output: Θ
$\Theta \leftarrow \Theta'$ ▷ Initialize parameters
$\mathbf{u}^0 \leftarrow \mathbf{u}$
if *type* $= SML$ **then** ▷ train SML
    $\Theta^{1\ldots K} \leftarrow$ `Minimize-Loss`$(\Theta^{1\ldots K}, \mathbf{u}^0)$
else ▷ train GRD/PRG
    $k \leftarrow 1$
    while $k \leq K$ **do**
        $\Theta^k \leftarrow$ `Minimize-Loss`$(\Theta^k, \mathbf{u}^{k-1})$
        if *type* $= PRG$ and $k > 1$ **then**
            $\Theta^{1\ldots k} \leftarrow$ `Minimize-Loss`$(\Theta^{1\ldots k}, \mathbf{u}^0)$
        if $k < K - 1$ **then**
            $\mathbf{u}^k \leftarrow$ `Encode`$(\Theta^{1\ldots k}, \mathbf{u}^0)$
        $k \leftarrow k + 1$
return Θ

3.2.3 Progressive training

In this study, we propose creating a DAE as follows: We first construct a shallow AE. Then, we add the next level greedily. Next, we train the entire network using simultaneous training. This process of adding one level and then training all levels simultaneously is iterated until the desired depth is reached. We refer to this training method as "progressive".

3.2.4 Deeply supervised training

In another approach, we propose to use an architecture in which all autoencoder levels have their loss contribute to the total loss, hence all DAE levels are trained jointly. This is motivated by the idea behind deeply supervised networks [154] in which each layer has a connection to the output, hence helping the error to propagate to the lower layers of the network. The architecture is shown in Figure 3.4. The loss is given by

$$Loss = L(\mathbf{x}, \hat{\mathbf{x}}^1) + L(\mathbf{x}, \hat{\mathbf{x}}^2) + \cdots + L(\mathbf{x}, \hat{\mathbf{x}}^K). \tag{3.5}$$

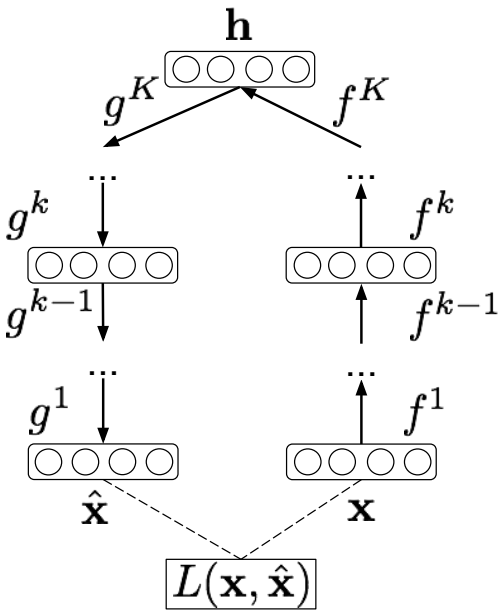


Figure 3.3: Deep autoencoder simultaneous training.

3.3 Experiment: Reconstruction

In a first experiment, we set out to compare and contrast the performance of the different training approaches, using the TIMIT speech corpus which contains read speech for acoustic-phonetic studies requiring comprehensive phonetic coverage. We used the recommended training and testing data split. For each utterance, 40^{th}-order mel-cepstra (MCEP) features (excluding the energy coefficient, thus the dimensionality D=39) were extracted every 5 ms, using the Ahocoder toolkit [60]. We chose to model 11 frames (the current frame plus 10 preceding frames), for a total of $11 \times 39 = 429$ features per frame. All AEs were regularized using a denoising approach, which has been shown to result in more robust and general feature representations [335]. Denoising operates by corrupting the inputs with random noise (in our case Gaussian noise) while reconstructing clean input targets. For the first-level AE, s^1_{enc} was a sigmoidal transfer function, and s^1_{dec} was linear, and the number of hidden nodes is 50. For all higher level activation functions s^k, $k > 1$, we used sigmoids for both encoding and decoding. We used the mean-squared error (MSE) as the loss function. For each level, we extended the architecture by adding an AE with 50 hidden nodes. The goal was to keep the encoding dimension constant (50 in this experiment) across the configurations under examination.

For evaluating the first-order moments of the reconstructed features, we report the reconstruction error in form of the average mel-cepstral distortion (mel-CD in dB) between the original and

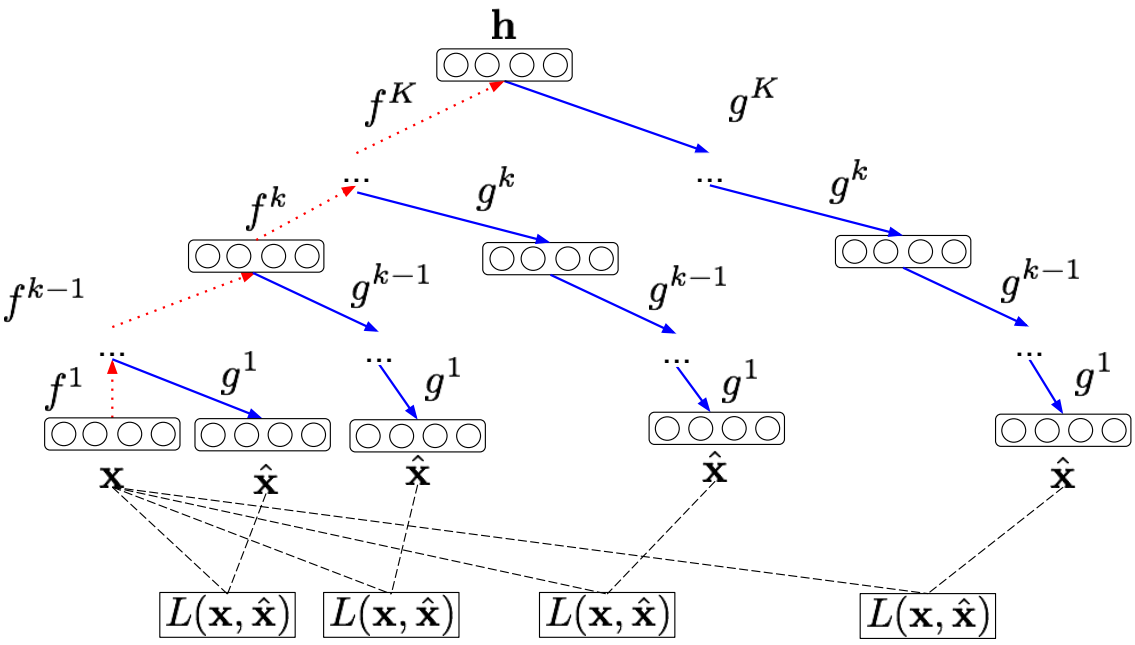


Figure 3.4: Deep autoencoder deeply supervised training

reconstructed MCEP vectors for the current frame:

$$E_1(\mathbf{x}, \hat{\mathbf{x}}) = \frac{10}{\log 10} \sqrt{2 \sum_{d=1}^{D} (\mathbf{x}_d - \hat{\mathbf{x}}_d)^2} \tag{3.6}$$

For evaluating the second-order moments of the reconstructed features, we based the measurement on the average ratios of the standard deviation of each reconstructed feature component over the standard deviation of the associated original feature component:

$$E_2(\mathbf{x}, \hat{\mathbf{x}}) = (1 - \frac{1}{D} \sum_{d=1}^{D} \frac{\sigma_{\hat{y}_d}}{\sigma_{y_d}})^2 \tag{3.7}$$

Note that in this measure, the standard deviation of the original features are always in the denominator. Also, in this study, the standard deviation of the original features is always larger than the standard deviation of the reconstructed features.

Table 3.1 shows the results. The reconstruction error for GRD constantly increased as depth was increased. The reason for this increase is that each added level introduces a local reconstruction error, causing an increase in the total reconstruction error. It is important to note that the ultimate goal is to learn useful features, which may not necessarily correlate with reconstruction error. Furthermore, initializing the network using an already GRD-trained architecture and then

Level(s) \ Method	GRD	SML	GRD→SML	PRG	DPS
1	3.81 (0.38)				
2	4.33 (0.45)	4.12 (0.45)	3.81 (0.42)		4.22 (0.45)
3	4.50 (0.48)	4.59 (0.48)	3.88 (0.47)	3.80 (0.42)	4.31 (0.46)
4	4.61 (0.50)	4.74 (0.49)	3.98 (0.48)	3.75 (0.43)	4.36 (0.48)

Table 3.1: Errors of four training methods applied to variable-depth AE training methods. Shown are the first moment error E_1, defined as the mel-cepstral distortion, and the second-order moment $\sqrt{E_2}$, the standard deviation ratio error, in parenthesis.

continuing training with SML (represented as GRD→SML) shows better convergence of E_1 and E_2 measures. The PRG algorithm performed consistently better in both E_1 and E_2 error measures over all other training algorithms. Note that for depth 2 both GRD→SML and PRG are identical. Furthermore, we observe that GRD→SML and PRG perform better than DPS in terms of both E_1 and E_2. This might be due to the fact that in the DPS algorithm, the loss of the lower level DAEs might be competing with the loss of the highest level DAE resulting in a higher error measure. The DPS approach can be further refined so that each level is assigned a dedicated multiplicative regularizing hyper-parameter, α^k, associated with that level's reconstruction loss, as a function of the current training epoch. This allows more flexibility in training the model. For example, this makes it possible to first focus on lower levels by first assigning higher values to the lowest level loss, and gradually move the focus towards higher levels' loss functions. The loss would be as follows, where i represents the iteration number:

$$Loss = \alpha^1(i)L(\mathbf{x}, \hat{\mathbf{x}}^1) + \alpha^2(i)L(\mathbf{x}, \hat{\mathbf{x}}^2) + \cdots + \alpha^K(i)L(\mathbf{x}, \hat{\mathbf{x}}^K). \tag{3.8}$$

The original proposed DPS approach can be thought of as a special case of this more general proposition, with $\alpha^k(i) = 1$ for all i and k.

3.4 Experiment: Phoneme classification

To evaluate the quality of the learned features, we designed a phoneme recognition experiment. We used the TIMIT corpus and its suggested training/testing split to train and evaluate the model. We exclude all SA sentences which have the same content for all speakers. We selected 10% of the training speakers for validation purposes. We compressed the 61 unique phonemes present in the TIMIT corpus transcriptions to 39 using a Kaldi script[1]. For the experiment, we also computed the frame-to-frame transition probabilities between phonemes from the training data and performed a Viterbi search on the posterior probabilities of the phoneme classifier during testing. We used

[1] https://github.com/kaldi-asr/kaldi/blob/master/egs/timit/s5/conf/phones.60-48-39.map

Feature	# of classifier DNN layers	GRD	SML	PRG	DPS
DAE 3-level	1	64.23%	63.89%	64.46%	64.48%
	2	68.14%	67.88%	68.75%	68.79%
	3	69.11%	69.02%	69.56%	69.61%
DAE 4-level	1	64.34%	64.09%	64.93%	65.02%
	2	68.57%	67.97%	68.75%	68.88%
	3	69.30%	69.16%	**69.81%**	**69.95%**
MCEPs	1	60.73%			
	2	67.82%			
	3	**68.97%**			

Table 3.2: Phoneme classification accuracy of feed-forward DNNs on various features: MCEPs, DAE-SML, DAE-GRD, DAE-PRG, and DAE-DPS.

a phoneme bigram to model the transition probabilities. The goal is to reduce the jumps that might happen due to selecting the phoneme with maximum probability by using the transition probabilities.

Similar to the previous subsection, we use MCEP features. We use the representation of the models trained using different layers and algorithms. For building the classification model, we used feed-forward DNNs with ReLU activation functions and dropout of 0.5 at each layer. We trained the model using the cross entropy loss function and the Adam optimization algorithm with a learning rate of 0.001 as suggested by the Keras documentation[2]. We trained the model for 100 iterations and selected the model with the lowest validation error. We report the frame-wise phoneme classification accuracy [136]. The results are shown in Table 3.2. The results show a consistent improvement when representations with deeper DAEs are used, regardless of the classifier. Furthermore, there is a consistent improvement in performance of the same classifier when PRG is used over GRD and SML. Similarly, there is a consistent improvement in performance of the same classifier when DPS is used over GRD and SML Finally, the DAE representation features perform better than MCEPs. We compare the best-performing DNN operating on DAE-PRG and also DAE-DPS versus MCEPs using Chi-square test with Yates's correction. The test shows a significant difference between the distribution of the phoneme classification scores with the two-tailed p value less than 0.005 for both comparisons. Furthermore, we conduct a similar experiment between best-performing PRG and DPS but do not find significant difference between the scores.

[2]https://keras.io/optimizers/#adam

Chapter 4

Joint Autoencoder

4.1 Introduction

In this chapter, we propose an architecture for regression that involves *two* autoencoders (AEs), one representing the input and another representing the output distribution. Each AE is tasked with learning an efficient encoding for each data distribution, using a reconstruction loss function; however, in addition, the two encodings are also encouraged to be *similar* by imposing an *additional* loss function. We refer to the resulting architecture as a joint autoencoder (JAE). The final regression function is constructed by encoding the input features using the input autoencoder and then decoding these values using the output decoder. In addition, we extend the idea of encouraging joint representations to *stacked* and *deep* joint autoencoders. The motivation for using deeper architectures is that it enables finding joint representations between input and output at deeper representation levels, which are likely to represent more abstract phenomena. Another important advantage of the proposed architecture is that it provides us with the ability to initialize the mapper using unsupervised input and output data. Similar architectures have been studied in multi-view deep learning techniques which have focused on extracting useful representations from multiple data streams, which enable learning representations that are more linearly separable, thus helping classification tasks [339]. We focus on the generative aspects of the joint architecture and study the properties of the predicted data.

We also propose a novel level-progressive training approach for training deep autoencoders. In this training scheme, rather than training all levels simultaneously, or by stacking autoencoders in a greedy fashion, we propose an algorithm in which the deep autoencoder of depth K is trained by starting from a shallow architecture and progressing the levels one-by-one, while performing simultaneous weight updating for all previous layers. This process is iterated until the desired depth K is reached. We experimentally show that this approach results in better first- and second-order

moments of reconstructed features.

4.2 Formulation

4.2.1 Joint Autoencoder

The joint autoencoder (JAE) consists of two AEs, one learning the input feature representations and the other learning the output feature representations. These two AEs are architecturally separate, i. e. they do not have any node connections with each other. The encoding and decoding functions for the input and output autoencoders can be represented as

$$\begin{array}{rclcl} \mathbf{h}_x & = & f_x(\mathbf{x}) & = & s_{\text{enc}}(\mathbf{W}_x\mathbf{x} + \mathbf{b}_x) \\ \hat{\mathbf{x}} & = & g_x(\mathbf{h}_x) & = & s_{\text{dec}}(\mathbf{W}_x{}^\top\mathbf{h}_x + \mathbf{c}_x) \\ \mathbf{h}_y & = & f_y(\mathbf{y}) & = & s_{\text{enc}}(\mathbf{W}_y\mathbf{y} + \mathbf{b}_y) \\ \hat{\mathbf{y}} & = & g_y(\mathbf{h}_y) & = & s_{\text{dec}}(\mathbf{W}_y{}^\top\mathbf{h}_y + \mathbf{c}_y) \end{array} \tag{4.1}$$

where $\mathbf{W}$, $\mathbf{b}$, $\mathbf{c}$ are the weights, hidden layer bias, and visible layer bias, respectively. The encoding and decoding activation functions are represented by s_{enc} and s_{dec}. The parameters of the model are represented by $\Theta = \{\mathbf{W}_x, \mathbf{b}_x, \mathbf{c}_x, \mathbf{W}_y, \mathbf{b}_y, \mathbf{c}_y\}$. The hidden layer encodings represented by $\mathbf{h}_x$ and $\mathbf{h}_y$ are the input and output hidden layer values, respectively. If this architecture is trained to minimize the reconstruction cost for input and output separately, $\mathbf{h}_x$ and $\mathbf{h}_y$ are likely to be uncorrelated. The core idea of the JAE is to maximize the *similarity* of the encoding values between $\mathbf{h}_x$ and $\mathbf{h}_y$, in addition to the goal of reconstruction. Therefore, we modify the standard training loss function to include the loss between the hidden layer encodings, i. e.

$$\begin{array}{l} L\left(\mathbf{x}, \hat{\mathbf{x}}, \mathbf{y}, \hat{\mathbf{y}}, \mathbf{h}_x, \mathbf{h}_y; \alpha\right) = \\ \quad (1-\alpha) \quad \cdot \quad (L_r(\mathbf{x}, \hat{\mathbf{x}}) + L_r(\mathbf{y}, \hat{\mathbf{y}})) + \\ \qquad \alpha \quad \cdot \quad L_d(\mathbf{h}_x, \mathbf{h}_y) \end{array} \tag{4.2}$$

where L_r is the reconstruction loss function and L_d is the distortion loss function between the hidden values of the two otherwise separate AEs, with α denoting the jointness of the input and output autoencoders; this jointness factor controls the tradeoff between reconstruction performance and the degree of encoding similarity. The JAE architecture is depicted in Fig. 4.1. The input and output autoencoders in the JAE architecture can be initialized randomly, or from previously trained autoencoders on unsupervised data. Using the trained JAE architecture, we can directly construct a mapping function for transforming input features to output features. First, the hidden

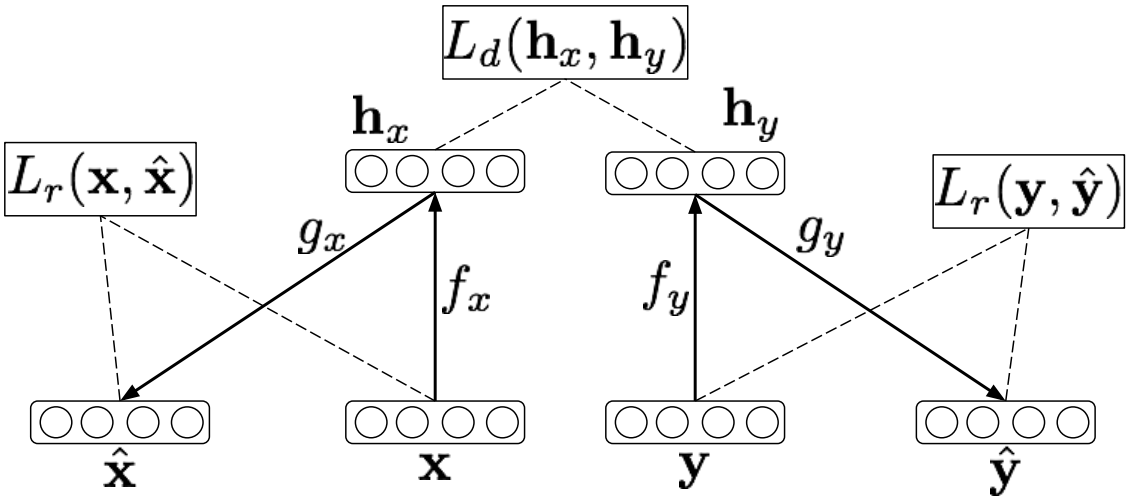


Figure 4.1: Joint autoencoder architecture. The autoencoder maps $\mathbf{x}$ and $\mathbf{y}$ to $\mathbf{h}_x$ and $\mathbf{h}_y$, and then reconstructs $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$. Reconstruction error is measured by $L_r(\mathbf{x}, \hat{\mathbf{x}})$ and $L_r(\mathbf{y}, \hat{\mathbf{y}})$ and the distortion error is measured by $L_d(\mathbf{h}_x, \mathbf{h}_y)$.

encodings are computed using the input encoder. These encodings are then decoded using the target decoder to obtain the output. The mapping function is essentially a feed-forward network where the parameters are trained using the JAE architecture:

$$\hat{\mathbf{y}} = \mathcal{F}_{JAE}(\mathbf{x}) = g_y\left(f_x(\mathbf{x})\right) \tag{4.3}$$

4.2.2 Joint Deep Autoencoder

JAEs can be composed of AEs with deep architectures. We refer to these architectures as joint deep autoencoders (JDAE) since they include multiple levels. Each level can be assigned different α values, which represents the tradeoff between reconstruction performance and encoding similarity. Previously we found that if lower levels are assigned lower α values, we achieve better VC performance [183]. In this study, we focus on studying the architectures with joining performed at the top-most level only. The JDAE computes

$$\begin{array}{rcccl}
\mathbf{h}_x^k & = & f_x^k(\mathbf{h}_x^{k-1}) & = & s_{\text{enc}}^k(\mathbf{W}_x^k\mathbf{h}_x^{k-1} + \mathbf{b}_x^k), \\
\mathbf{h}_y^k & = & f_y^k(\mathbf{h}_y^{k-1}) & = & s_{\text{enc}}^k(\mathbf{W}_y^k\mathbf{h}_y^{k-1} + \mathbf{b}_y^k), \\
\hat{\mathbf{h}}_x^{k-1} & = & g_x^k(\hat{\mathbf{h}}_x^k) & = & s_{\text{dec}}^k(\mathbf{W}_x^{k\top}\hat{\mathbf{h}}_x^k + \mathbf{c}_x^k), \\
\hat{\mathbf{h}}_y^{k-1} & = & g_y^k(\hat{\mathbf{h}}_y^k) & = & s_{\text{dec}}^k(\mathbf{W}_y^{k\top}\hat{\mathbf{h}}_y^k + \mathbf{c}_y^k)
\end{array} \tag{4.4}$$

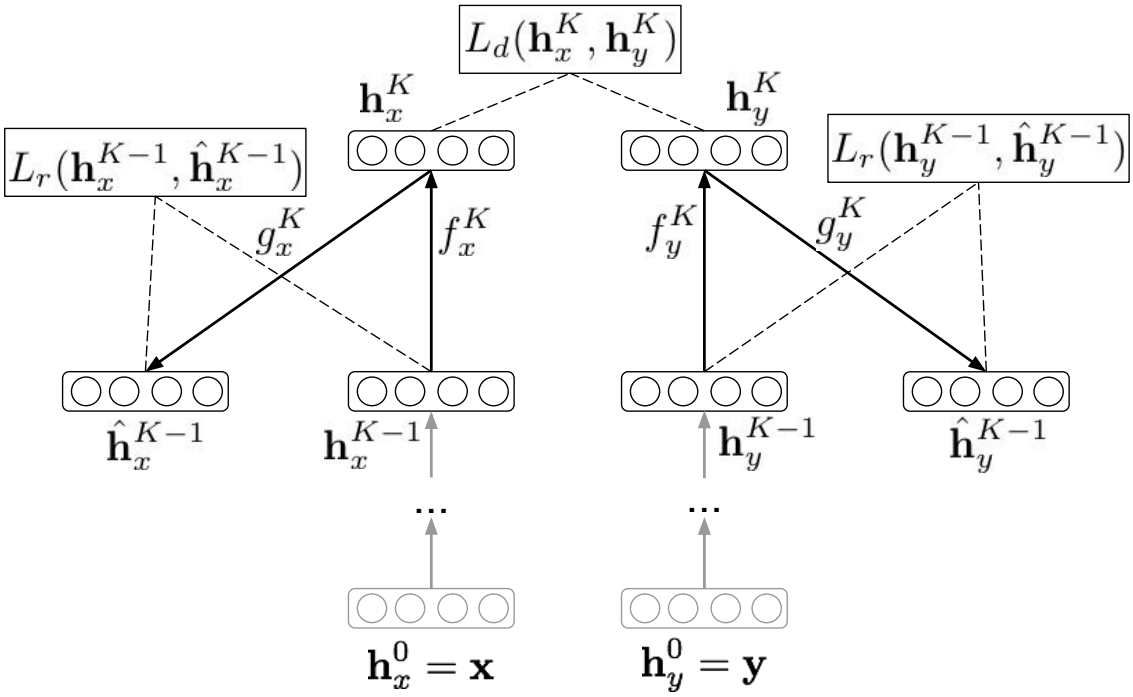


Figure 4.2: Joint deep autoencoder greedy training. After training a first-level joint autoencoder (see Figure 4.1), its learned encoding function is used on input and output features. The resulting representation is used to train a second-level joint autoencoder to learn a second-level encoding function. From there, the procedure can be repeated.

with $k = 1, \ldots, K$, $\mathbf{h}_x^0 = \mathbf{x}$, $\mathbf{h}_y^0 = \mathbf{y}$, $\hat{\mathbf{x}} = \hat{\mathbf{h}}_x^0$, and $\hat{\mathbf{y}} = \hat{\mathbf{h}}_y^0$. The parameters of the model are $\Theta = \{\mathbf{W}_x^k, \mathbf{b}_x^k, \mathbf{c}_x^k, \mathbf{W}_y^k, \mathbf{b}_y^k, \mathbf{c}_y^k\}_{k=[1,\cdots,K]}$, where superscript k represents the k^{th}-level of an AE with K levels. The JDAE encoding and decoding functions for input x (and analogously output y) are defined as $\mathbf{h}_x = f_x(\mathbf{x}) = (f_x^K \circ f_x^{K-1} \circ \cdots \circ f_x^1)(\mathbf{x})$ and $\hat{\mathbf{x}} = g_x(\mathbf{h}_x) = (g_x^1 \circ g_x^2 \circ \cdots \circ g_x^K)(\mathbf{h}_x)$, respectively. Similar to Equation 4.3, the input hidden encoding from the incoming input is calculated using source JDAE encoder, and these encoding values are decoded using the output JDAE decoder, i. e.

$$\hat{\mathbf{y}} = \mathcal{F}_{\text{JDAE}}(\mathbf{x}) = (g_y^1 \circ g_y^2 \circ \cdots \circ g_y^K \circ f_x^K \circ f_x^{K-1} \circ \cdots \circ f_x^1)(\mathbf{x}) \tag{4.5}$$

The architecture can be trained in various ways similar to Section 3.2: greedy, simultaneous, and progressive.

4.2.2.1 Greedy training

The JDAE architecture with greedy training is depicted in Figure 4.2. Each autoencoder level is trained individually with the relevant loss function, starting from training the lowest-level and encoding the parameters, and iterating the training/encoding process until all levels are trained, using the loss function

$$\begin{array}{rcl} L\left(\mathbf{h}_x^{k-1}, \hat{\mathbf{h}}_x^{k-1}, \mathbf{h}_y^{k-1}, \hat{\mathbf{h}}_y^{k-1}, \mathbf{h}_x^{k}, \mathbf{h}_y^{k}; \alpha^k\right) = & & \\ (1-\alpha^k) & \cdot & \left(L_r(\mathbf{h}_x^{k-1}, \hat{\mathbf{h}}_x^{k-1}) + L_r(\mathbf{h}_y^{k-1}, \hat{\mathbf{h}}_y^{k-1})\right) + \\ \alpha^k & \cdot & L_d(\mathbf{h}_x^{k}, \mathbf{h}_y^{k}) \end{array} \tag{4.6}$$

where α^k is zero for all $k < K$, and a non-zero value (see Section 4.3.2.1) for $k = K$.

When unsupervised data is available, a greedy level-wise approach can be used to build the stacked autoencoder level-by-level. First, the input and output AEs in each level can be initialized by their relevant unsupervised data, and then the JAE is trained on the supervised data. For the next level, in addition to encoding the input and output values, the input and output unsupervised corpus is also encoded using current level AEs. The next level is trained similarly, and the process is iterated until the desired depth is reached. The implementation of the greedy level-wise JDAE training (GRD) is shown in Algorithm 4.1. The argument $\mathbf{u}$ represents unsupervised data used in the initialization step.

4.2.2.2 Simultaneous training

A greedy training stacks multiple AEs to build a deep representation of feature space. A new level of the DAE is trained independently from its previous levels. This training scheme is also referred to as greedy level-wise training. In Section 3.3, we showed that training all the levels simultaneously results in better feature representation [374]. We posit that the same improvement is possible for our proposed joint models; specifically, we extend the greedy JDAE training so that the parameters of all the levels are trained simultaneously in each iteration.A local optimum of Θ is found by minimizing the following loss function

$$\begin{array}{rcl} L\left(\mathbf{x}, \hat{\mathbf{x}}, \mathbf{y}, \hat{\mathbf{y}}, \mathbf{h}_x, \mathbf{h}_y; \alpha\right) = & & \\ (1-\alpha) & . & (L_r(\mathbf{x}, \hat{\mathbf{x}}) + L_r(\mathbf{y}, \hat{\mathbf{y}})) + \\ \alpha & . & L_d(\mathbf{h}_x, \mathbf{h}_y) \end{array} \tag{4.7}$$

It is worthwhile to note that all the parameters of all levels are optimized simultaneously, rather than in a greedy level-wise fashion. The implementation of simultaneous JDAE training (SML) is

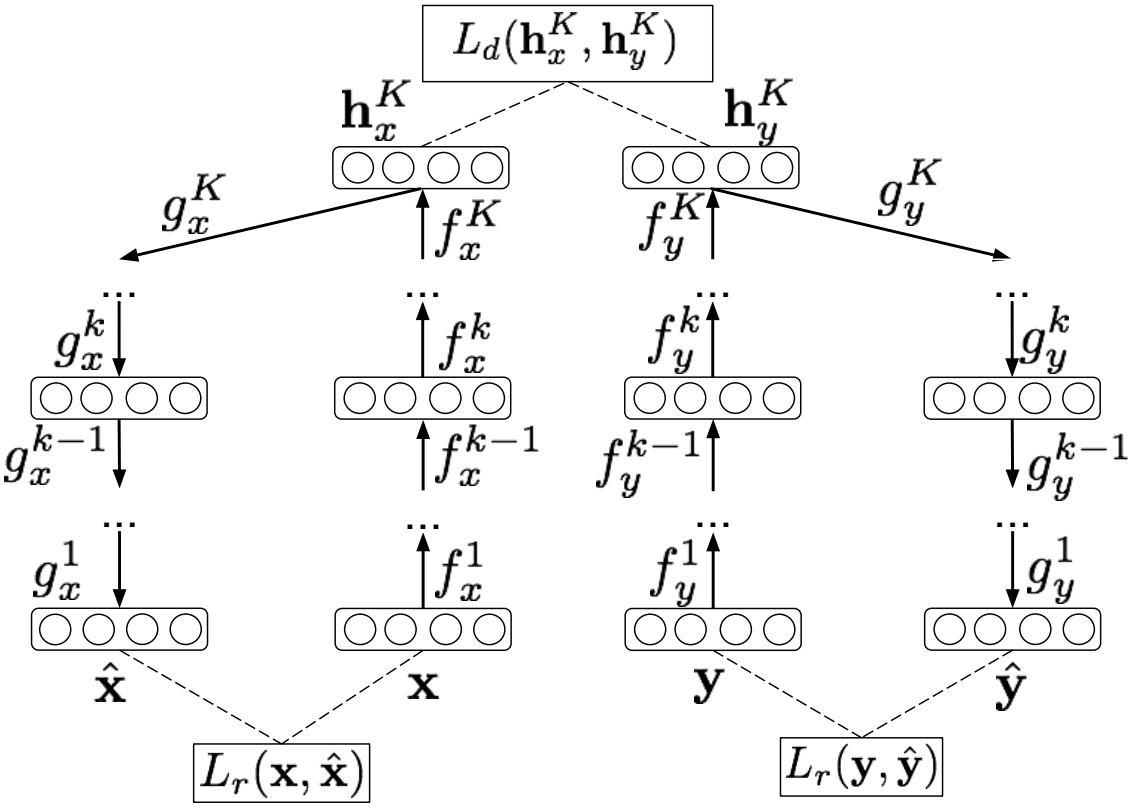


Figure 4.3: JDAE simultaneous training. All the level parameters are optimized simultaneously, rather than level-wise.

shown in Algorithm 4.1. The simultaneous JDAE architecture is depicted in Figure 4.3.

4.2.2.3 Progressive training

Similar to Section 3.2, in which we proposed a progressive training scheme for learning the DAE weights, we also propose a progressive training scheme for JDAE architecture. In this architecture, the first level representation is learned. Then the second level JAE, followed by a simultaneous learning of all the parameters up to the second level. Similarly, the third level JAE is leaned, followed by a simultaneous learning of all the parameters up to the third level. This process is iterated until the desired depth is reached. The implementation of progressive JDAE training (PRG) is shown in Algorithm 4.1.

4.2.2.4 Deeply supervised training

We also use a deeply supervised method similar to the method proposed in Section 3.2.4 to train the JDAE. We define the loss function for each of the source and target DAEs by the deep supervised training approach. Hence, all DAE levels have reconstruction loss present in the total loss function. All the parameters are then learned simultaneously. We hypothesize that similar to training a single DAE, the joint DAE also benefits from the densely supervised training approach since the errors are propagated more easily to the lower levels.

4.3 Experiment: Voice Conversion

We applied the proposed architectures to the Voice Conversion (VC) problem, which is particularly well-suited as it involves a regression with high-dimensional input and output data. Specifically, a VC system is a speech generation system which converts speech produced by a *source* speaker to sound similar to that of a *target* speaker's. Most commonly, a speech utterance is analyzed frame-by-frame, then extracted source speaker features are mapped to more closely resemble likely target speaker features, and finally a new output speech waveform is synthesized. Due to the high dimensionality of the features, the mapping is typically achieved using a non-linear regression function, which must be trained on time-aligned source and target features from existing parallel or artificially parallelized [e. g. 58] speech utterances. The speech features used for speech analysis/synthesis and mapping typically do not have good interpolation properties [184] which poses challenges for approaches that perform addition operations on the features. Human speech also exhibits inconsistency over several renditions of the same sentence [120] which results in one-to-many relationships in the parallel training data [92], further complicating the mapping function learning process. During training, a feed-forward architecture learns to directly predict the target speech features from the source speech features. This training criterion typically results in averaged features (higher E_2), since the algorithm is attempting to minimize a loss function which is not typically highly correlated with perceptual quality. These challenging issues are all factors in over-averaging of converted features which leads to decreased naturalness of the generated speech. We hypothesize that the JDAE architecture will lead to higher variance in the output features (lower E_2), and thus increased perceived speech quality. In this section, we detail various training configurations and investigate the effects of several hyper-parameters on objective evaluation results. Finally, we report on the outcome of a subjective listening test.

4.3.1 Training

We used the VC challenge 2016 corpus [308] as speech data, consisting of 162 parallel sentences. We split the sentences into training (100), validation (31), and objective testing (31) sentences for the objective experiments. We utilize the 54 subjective testing set as proposed in the corpus for subjective experiments. As speech features, we used 40^{th}-order MCEPs (excluding the energy coefficient, dimensionality D=39), extracted using the Ahocoder toolkit [60] with a 5 ms frame shift. We chose to model 11 frames (the current frame plus 10 preceding frames), for a total of 11×39=429 features per frame. For simplicity, we only consider four conversions: SM1→TF1, SF2→TM1, SF1→TF2, and SM2→TM2 (two inter- and two intra-gender conversions). For the first-level AE, s_{enc} is a sigmoidal transfer function, and s_{dec} is linear. For all higher levels, we used sigmoids for both s_{enc} and s_{dec}. We let L_r and L_d equal the mean-squared error. Additionally, we used a Gaussian corruption function to corrupt the inputs to the network during training.

We initialized the network parameters using the following schemes:

Random Weights are initialized to small random values.

Speaker independent Alternatively, we first trained the networks using general-purpose, multi-speaker unsupervised data; in this case we used all of the training set speakers of the TIMIT corpus [69].

Speaker dependent The joint architectures also allow speaker dependent initialization. The source and target (D)AEs can be initialized using two separate unsupervised data corpora, each selected to be relatively similar to the source and target data distributions [182]; in this case we separated TIMIT into male and female sets and use them to initialize the source and target (D)AEs based on their gender.

We investigate the value of various measures for a 2-level JDAE and speaker independent initialization over training iterations; specifically, we show the values of L_d, L_r for source and target speakers, E_1, and E_2 in Figure 4.4. As training progresses, we observe that E_2 goes down for the JDAE whereas it goes up for the DNN. We use DNNs as baseline since it is the state-of-the-art approach that uses a closeness criterion on the output. There exists more advance models with sequence modelling capabilities, however we intended to keep the architectures simple to assess our proposition with fewer dependent variables (such as hyper-parameters).

4.3.2 Objective Evaluation

4.3.2.1 Finding the optimum jointness factor

To experiment with the effect of the jointness factor α on mapping accuracy, we varied the α value from 0 to 1 in 0.1 increments. We also added points 0.01 and 0.99 as extreme border cases. We used 100 training sentences to train a JDAE architecture with two levels and speaker independent initialization. As the baseline, we report DNN performance. We report average E_1 of all four conversion pairs for DNN and the JDAE mapper trained with the PRG scheme. As shown in Figure 4.5-a, the optimum value was between 0.6 and 0.7. For $\alpha > 0.8$, the JDAE mapper quickly became unstable regarding mapping accuracy since the reconstruction errors for source and target DAEs started performing poorly. For $\alpha = 1.0$, the network was only trying to minimize the encoding similarity, resulting in very high reconstruction and thus also mapping errors (which are outside of the plot with values of > 30 dB).

It is possible to select the jointness factor based on both E_1 and E_2, allowing control over the first- and second-order behavior of the mapper. First, we normalized the mel-CD plot, $E_1(\mathbf{y}, \hat{\mathbf{y}})$, to contain values between zero and one, by subtracting 4.0 dB and dividing the result by $10.47-4.00$, obtaining E_1'. The value 4.00 was based on the estimated rendition error that speakers naturally have between different utterance renditions of the same sentence [120], and we estimated 10.47 as the pre-existing mel-CD error between source and target speakers. We then found the best alpha by solving

$$\alpha = \text{argmin}_\alpha \Big[(1-\beta) \cdot E_1'(\mathbf{y}, \mathcal{F}_{\mathbf{JDAE}}(\mathbf{x}; \alpha)) + \beta \cdot E_2(\mathbf{y}, \mathcal{F}_{\mathbf{JDAE}}(\mathbf{x}; \alpha)) \Big] \tag{4.8}$$

where the value of β acts as a control knob for selecting a system that controls the tradeoff between first-order and second-order accuracy. For lower β values, we favor systems with lower mel-CD; conversely, for higher β values, we favor systems that generate features with higher standard deviation. For β values of 0.0, 0.25, 0.50, and 0.75, the corresponding α values are 0.70, 0.50, 0.2, and 0.1, respectively. For the following experiments, we used a jointness factor of $\alpha = 0.2$, which corresponds to $\beta = 0.5$, favoring first- and second-order accuracy equally.

4.3.2.2 Investigating different architectures and training configurations

We report the conversion accuracy of a male to female conversion (SM1→TF1) in Table 4.1. We experimented with using 20 or 100 training sentences to see the training size effect. As a reference,

the mel-CD between the source and target testing samples SM1 and TF1 (i. e. the pre-existing error without mapping) is 10.47 dB. We investigate two fundamental regression approaches and their variations: the feed-forward (DNN, SNN) and joint AE (JAE, JDAE-GRD, JDAE-SML, JDAE-PRG) architectures. We experimented with three different depths ($K = 1, 2, 3$) and various training configurations. The results of the experiments are presented in Table 4.1. For $K = 1$, the JAE performed worse than the SNN regarding E_1 in all initialization scenarios; however, it performed significantly better regarding E_2. For $K = 2$ and $K = 3$, comparing the different JDAE architectures, we found PRG>GRD>SML regarding E_1, and PRG resulted in either a similar or better E_2, compared to SML and GRD. The initialization of the DNN with the speaker independent model trained using PRG or GRD did not differ significantly, but both outperformed random initialization as expected. The improvement is even more noticeable for the smaller training set, since proper initialization is more likely to help when there are fewer training data available.

Compared to speaker independent initialization, speaker dependent initialization improved performance for all joint architectures in all training configurations with respect to both E_1 and E_2. This suggests that selecting data distributions that are closer to individual speaker data distributions is beneficial during initialization. More precise data selection, compared to a mere male-female splitting, may be even more beneficial. Finally, we also investigated the effect of training size in more detail. In Figure 4.7, we show E_1 of the test set corresponding to changes in training size. For a higher number of training sentences, both E_1 and E_2 improve for the JDAE, whereas only E_1 improves for the DNN.

4.3.2.3 Investigating first- and second-order properties

We also studied the second-order properties of the generated features in a component-wise manner. We used the JDAE-PRG architecture with $K = 2$ levels and speaker independent initialization. Figure 4.8 shows E_1 (representing the first order properties) and E_2 (representing the second order properties). We observed that the E_2 values are always lower for the JDAE, which are more important for reducing the muffling of generated speech [301]. There still remains a gap to achieve a perfect E_2 (no muffling). Figure 4.9 shows the spectrogram of conversions of $K = 2$ level DNN and JDAE-PRG with speaker-independent initialization. We observed that the DNN-generated spectrogram is more blurry, the result of over-averaging. The JDAE-generated spectrogram shows sharper peaks and increased spectral detail. Furthermore, we investigated the DNN performance errors by adjusting the standard deviation of generated features in a post-processing way. We adjusted the standard deviation of generated features toward the standard deviation of training features, by subtracting the mean value of features and then normalizing it to one, and finally

Config \ Init		Random		Speaker independent		Speaker dependent	
Training sentences		20	100	20	100	20	100
Architecture	Level(s)						
JAE	1	9.11(0.01)	8.04(0.04)	8.16(0.24)	7.83(0.22)	8.10(0.26)	7.73(0.25)
JDAE-GRD	2	8.97(0.42)	7.46(0.41)	7.85(0.45)	7.49(0.46)	7.64(0.44)	7.42(0.45)
JDAE-SML	2	7.82(0.47)	7.39(0.44)	7.55(0.45)	7.30(0.41)	7.35(0.44)	7.24(0.41)
JDAE-PRG	2	7.67(0.43)	7.38(0.41)	7.50(0.45)	7.27(0.41)	7.32(0.43)	7.21(041)
JDAE-GRD	3	8.80(0.55)	7.46(0.53)	7.66(0.56)	7.35(0.57)	7.49(0.54)	7.28(0.56)
JDAE-SML	3	7.81(0.62)	7.47(0.57)	7.42(0.53)	7.25(0.49)	7.35(0.51)	7.20(0.49)
JDAE-PRG	3	7.55(0.61)	7.26(0.54)	7.39(0.52)	7.22(0.50)	7.28(0.51)	7.17(0.49)
SNN	1	6.77(0.64)	6.53(0.64)	6.76(0.64)	6.53(0.64)	n/a	n/a
DAE-GRD→DNN	2			6.80(0.62)	6.52(0.65)	n/a	n/a
DAE-PRG→DNN	2	7.13(0.75)	6.73(0.81)	6.78(0.62)	6.51(0.65)	n/a	n/a
DAE-GRD→DNN	3			6.79(0.64)	6.54(0.66)	n/a	n/a
DAE-PRG→DNN	3	8.42(0.90)	7.34(0.79)	6.76(0.63)	6.51(0.66)	n/a	n/a

Table 4.1: Objective measures for different architectures, training sentences, and initializations for M2F. The scores represent E_1, mel-CD, with $\sqrt{E_2}$ in parenthesis.

	Level 2	Level 3
JDAE-PRG	7.38(0.41)	7.26(0.54)
JDAE-DPS	7.37(0.41)	7.22(0.50)
DNN	6.73(0.81)	7.34(0.79)

Table 4.2: Objective measures comparing JDAE-PRG-$\alpha = 0.2$, and JDAE-DPS starting from random initialization. The scores represent E_1, mel-CD, with $\sqrt{E_2}$ in parenthesis.

multiplying it with the standard deviation of training features. For each feature element the following is performed

$$\hat{\mathbf{x}}_{adjusted-std} = ((1-\tau) \times std(\hat{\mathbf{x}}) + \tau \times std(\mathbf{x}_{train})) \times (\hat{\mathbf{x}} - mean(\hat{\mathbf{x}}))/std(\hat{\mathbf{x}})$$

in which τ controls the amount of adjustment. In Figure 4.10, we show both E_1 and E_2 for DNN with adjusted standard deviation and contrast it to DJAE. We use JDAE-PRG with $K = 2$ and DNN of the same architecture for this experiment.

4.3.2.4 Investigating performance of DPS versus PRG

We investigated the performance of DPS versus PRG approaches. We compared $K = 2$ and $K = 3$ level architectures with random initialization. The results are shown in Table 4.2. We did not find a statistically significant difference (by performing a t-test) between PRG and DPS.

4.3.2.5 Investigating the effect of jointness factor in progressive training

We investigated the effect of various jointness factors in progressive training. During progressive training, we grow the network from shallow layers to deeper layers. In JDAE, we have the option to either impose a jointness factor in building the shallower layers, or not impose any jointness factor in the shallow network training and only impose the jointness factor only in the deepest layers.

We used the JDAE-PRG architecture with $K = 3$ levels and random initialization and trained on 100 training sentences. We used a varying jointness factor from $\alpha = 0.2$ to $\alpha = 0.4$. We consider three different jointness factor assignments: JDAE-PRG-repr, JDAE-PRG-all, and JDAE-PRG-linear:

JDAE-PRG-repr we only applied the α to the deepest representation layer and other levels have no jointness.

	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$
JDAE-PRG-repr	7.44(0.61)	7.26(0.54)	7.15(0.51)	7.03(0.49)
JDAE-PRG-all	7.46(0.58)	7.29(0.50)	7.17(0.48)	7.06(0.46)
JDAE-PRG-linear	7.43(0.57)	7.26(0.50)	7.15(0.49)	7.01(0.47)

Table 4.3: Objective measures comparing applying jointness factor to all levels equally, all levels incrementally, or only deepest level in JDAE-PRG. The scores represent E_1, mel-CD, with $\sqrt{E_2}$ in parenthesis

JDAE-PRG-all we applied α to all levels throughout progressive training.

JDAE-PRG-linear we applied jointness factor to all levels throughout progressive training, however we increase the jointness factor from 0 for first level to the target α for the representation level. This shows the effect when we increase the jointness from no jointness to the desired jointness, which seems a reasonable hypothesis since the lower layers are learning their own data distribution and might perform better if less jointness is imposed. For deeper levels, we can impose more jointness since they are learning more abstract information and would be joint more appropriately.

We show the E_1 and E_2 scores in Table 4.3. We observe that JDAE-PRG-linear results in a better E_1 and E_2 tradeoff behavior in most cases. This shows that, growing the jointness from low values for lower levels towards higher values for deeper levels is more effective than only applying jointness on the deepest level or using a constant jointness for all levels.

4.3.2.6 Investigating effects of similarity measures

In this experiment, we aim to compare several similarity loss functions that is applied to the hidden representation. We compared: Squared Error (SE), Absolute Error (AE), Kullback-Leibler (KL) divergence, and cosine proximity (CP).

We used the JDAE-PRG architecture with $K = 3$ levels and random initialization and train on 100 training sentences. We used a jointness factor of $\alpha = 0.2$. We show the E_1 and E_2 scores in Table 4.4. The scores show better performance regarding E_1 for SE and best performance for E_2 for AE. We speculate that the reason SE performs better in E_2 is possibly due to learning representations which are more sparse.

Loss	Error
SE	7.26(0.54)
AE	7.75(0.43)
KL	7.28(0.50)
CP	7.89(0.48)

Table 4.4: Objective measures comparing various similarity measures applied to hidden layer. The scores represent E_1, mel-CD, with $\sqrt{E_2}$ in parenthesis

4.3.3 Subjective Evaluation

To subjectively evaluate voice conversion performance, we performed two perceptual tests. The first test measured speech quality, designed to answer the question "how natural does the converted speech sound?", and the second test measured speaker similarity, designed to answer the question "how accurately does the converted speech mimic the target speaker?". The listening experiments were carried out using Amazon Mechanical Turk, with participants who had approval ratings of at least 95% and were located in North America. Both perceptual tests used three trivial-to-judge trials, added to the experiment to exclude unreliable listeners from statistical analysis. No listeners were flagged as unreliable in our experiments.

4.3.3.1 Speech Quality Test

To evaluate the speech quality of the converted utterances, we conducted a Comparative Mean Opinion Score (CMOS) test. In this test, listeners heard two stimuli A and B with the *same* content, generated using the *same* source speaker, but in two *different* processing conditions, and were then asked to indicate wether they thought B was better or worse than A, using a five-point scale comprised of +2 (much better), +1 (somewhat better), 0 (same), −1 (somewhat worse), −2 (much worse). We utilized three processing conditions: DNN, JDAE, and VOCOD, where the VOCOD condition is a MCEP-vocoder re-synthesis, serving as a reference of maximum achievable quality (Re-synthesis quality is considered nearly transparent compared to the original audio). Hence, we ran three comparison pairs: VOCOD→DNN, VOCOD→JDAE, DJAE→DNN. We randomized the order of stimulus presentation, both the order of A and B, as well as the order of the comparison pairs. The experiment was administered to 40 listeners with each listener judging 111 (108+3) sentence pairs. The results are shown in Figure 4.11. The VOCOD→DNN, VOCOD→JDAE, and JDAE→DNN mean score values (with 95% confidence intervals) are -1.666 ± 0.134, -1.508 ± 0.151, and -0.741 ± 0.155, respectively. We performed planned one-sample t-tests with a mean of zero and

achieved $p < 0.0001$ for all comparison pairs, showing statistically significant differences between all processing conditions. Notably, JDAE is performing significantly better in regards to speech quality compared to DNN generated audio. Also, the VOCOD→JDAE drop is smaller than that of VOCOD→DNN, showing that the listeners perceived JDAE conversions to be closer to the purely vocoded condition.

4.3.3.2 Speaker Similarity Test

To evaluate the speaker similarity of the converted utterances, we conducted a same-different speaker similarity test [119]. In this test, listeners heard two stimuli A and B with *different* content, and were then asked to indicate wether they thought that A and B were spoken by the *same*, or by two *different* speakers, using a five-point scale comprised of +2 (definitely same), +1 (probably same), 0 (unsure), −1 (probably different), and −2 (definitely different). One of the stimuli in each pair was created by one of the two conversion methods, and the other stimulus was a purely MCEP-vocoded condition, used as the *reference* speaker. Half of all pairs were created with the reference speaker identical to the target speaker of the conversion (expecting listeners to reply "same", ideally); the other half were created with the reference speaker being the source speaker (expecting listeners to reply "different"). The experiment was administered to 40 listeners, with each listener judging 75 (72+3) sentence pairs. The results are shown in Figure 4.12. The DNN-same, DNN-diff, JDAE-same, and JDAE-diff mean score values (with 95% confidence intervals) are 0.0387±0.23, −1.323±0.17, 0.604±0.21, and −1.427±0.17, respectively. We performed planned one-sample t-tests with a mean of zero and found statistically significant differences for DNN-diff, JDAE-same, and JDAE-diff compared to chance (all with $p < 0.001$). Furthermore, we performed a two-sample t-test between DNN-same and JDAE-same and found the difference to be statistically significant ($p < 0.01$). These results show that listeners perceived JDAE conversions to better mimic the target speaker than when using the DNN. We also performed a two-sample t-test between DNN-diff and JDAE-diff, but did not find a statistically significant difference. Finally, we negated the DNN-diff and JDAE-diff scores and compared them to the DNN-same and JDAE-same scores, respectively. This type of comparison discovers which task is more difficult: de-identifying the source speaker, or mimicking the target speaker. Both t-test results for DNN and JDAE show statistically significant differences ($p < 0.0001$) between negated "diff" vs. "same" scores, confirming the intuition that is is easier to de-identify than to mimic.

Algorithm 4.1 Joint deep autoencoder training algorithm.

procedure `Train-JDAE`
Input: $\mathbf{u}_x$, $\mathbf{u}_y$, $\mathbf{x}$, $\mathbf{y}$, α , Θ', *type*
Output: Θ
$\Theta \leftarrow \Theta'$
$\mathbf{h}_x^0 \leftarrow \mathbf{x}$; $\mathbf{h}_y^0 \leftarrow \mathbf{y}$
$\mathbf{u}_x^0 \leftarrow \mathbf{u}_x$; $\mathbf{u}_y^0 \leftarrow \mathbf{u}_y$
if $type = SML$ **then** ▷ simultaneous training
    if $\mathbf{u}_x \neq \emptyset$ **then**
        $\Theta_x^{1\ldots K} \leftarrow$ `Minimize-Loss-AE`$(\Theta^{1\ldots K}, \mathbf{u}_x)$
        $\Theta_y^{1\ldots K} \leftarrow$ `Minimize-Loss-AE`$(\Theta^{1\ldots K}, \mathbf{u}_y)$
    $\Theta^{1\ldots K} \leftarrow$ `Minimize-Eq12`$(\Theta^{1\ldots K}, \mathbf{x}, \mathbf{y}, \alpha^K)$
else ▷ greedy/progressive training
    $k \leftarrow 1$
    while $k \leq K$ **do**
        if $\mathbf{u}_x \neq \emptyset$ **then**
            $\Theta_x^k \leftarrow$ `Minimize-Loss-AE`$(\Theta_x^k, \mathbf{u}_x^{k-1})$
            $\Theta_y^k \leftarrow$ `Minimize-Loss-AE`$(\Theta_y^k, \mathbf{u}_y^{k-1})$
        $\Theta^k \leftarrow$ `Minimize-Eq5`$(\Theta_x^k, \Theta_y^k, \mathbf{h}_x^{k-1}, \mathbf{h}_y^{k-1}, \alpha^k)$
        if $type = PRG$ and $k > 1$ **then**
            $\Theta^{1\ldots k} \leftarrow$ `Minimize-Eq12`$(\Theta^{1\ldots k}, \mathbf{x}, \mathbf{y}, \alpha^k)$
        if $k < K - 1$ **then**
            $\mathbf{h}_x^k, \mathbf{h}_y^k \leftarrow$ `Encode`$(\Theta^k, \mathbf{x}, \mathbf{y})$
            if $\mathbf{u}_x \neq \emptyset$ **then**
                $\mathbf{u}_x^k, \mathbf{u}_y^k \leftarrow$ `Encode`$(\Theta^k, \mathbf{x}, \mathbf{y})$
        $k \leftarrow k + 1$
return Θ

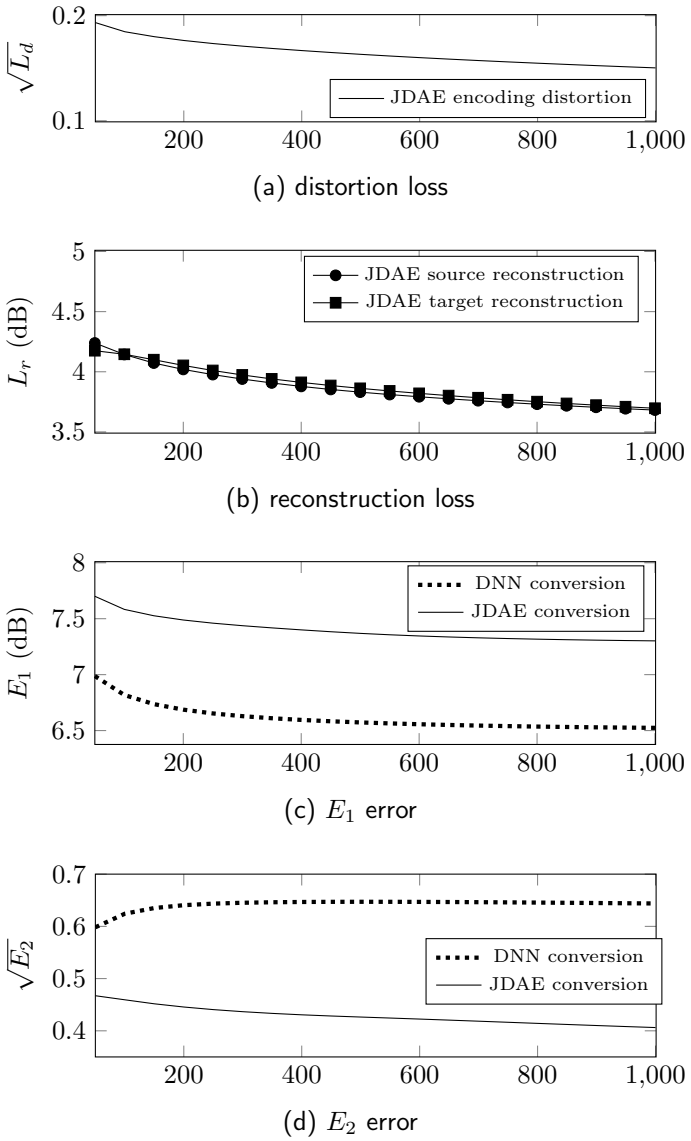


Figure 4.4: Various measures over 1000 training iterations of a JDAE with $\alpha = 0.2$, for the M2F case, using 100 sentences.

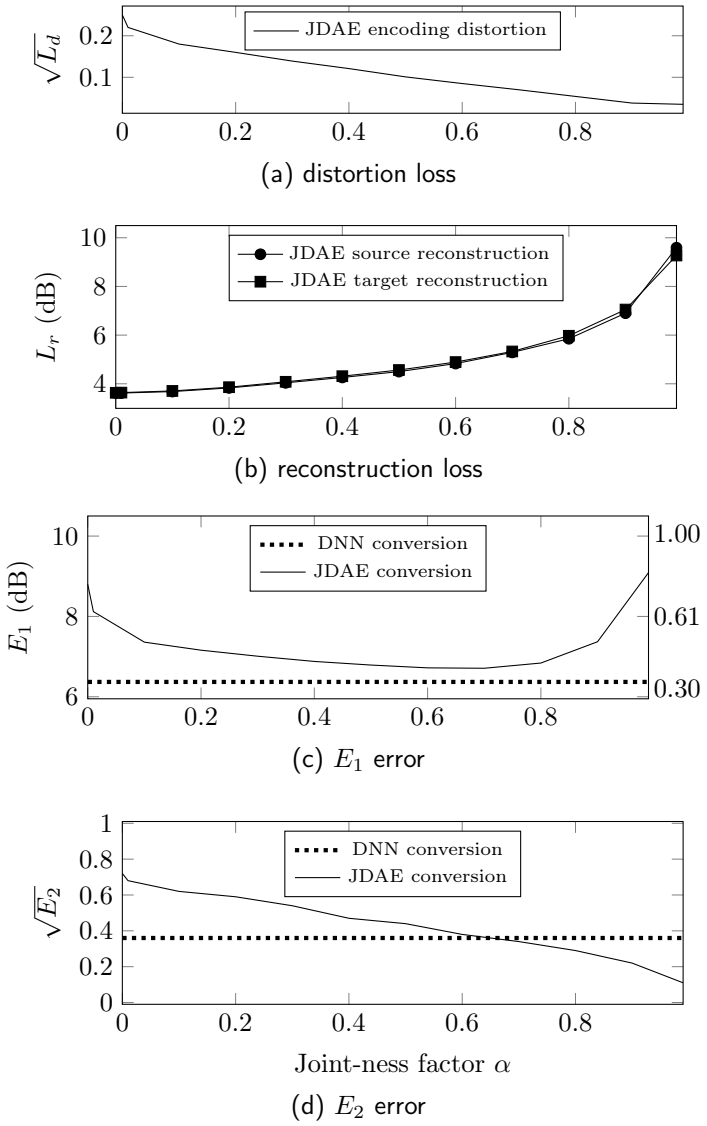


Figure 4.5: Varying the value of the joint-ness factor from 0 to 0.9 by 0.1 increments, points 0.01 and 0.99 are also included. First row represents the encoding distortion between the joint autoencoders. Second row shows the reconstruction error for source and target speakers. Third and fourth row show the mapping function E_1 (with raw value shown on the left y-axis and normalized value on the right y-axis) and E_2 performance, respectively. We used 100 training sentences and trained JDAE using PRG algorithm.

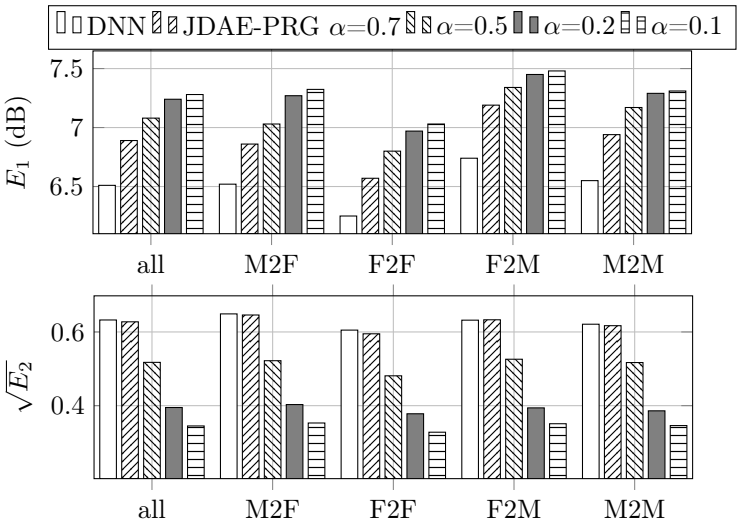


Figure 4.6: The objective measures for different conversions

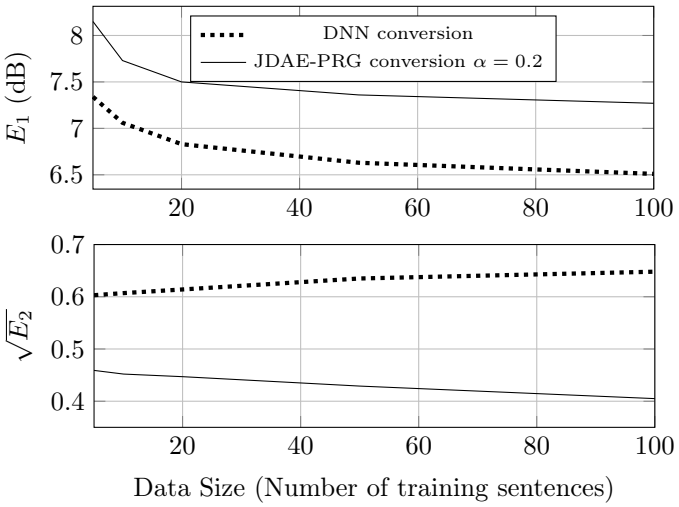


Figure 4.7: Varying the number of training sentences from 5 to 100

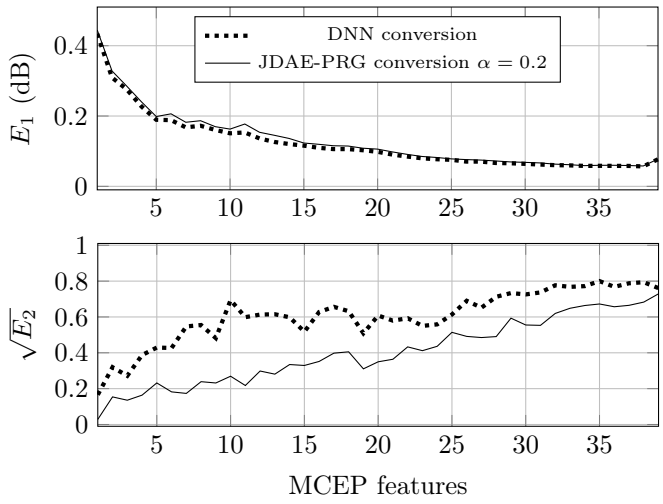


Figure 4.8: Component-wise E_1 and $\sqrt{E_2}$ for 39 MCEPs

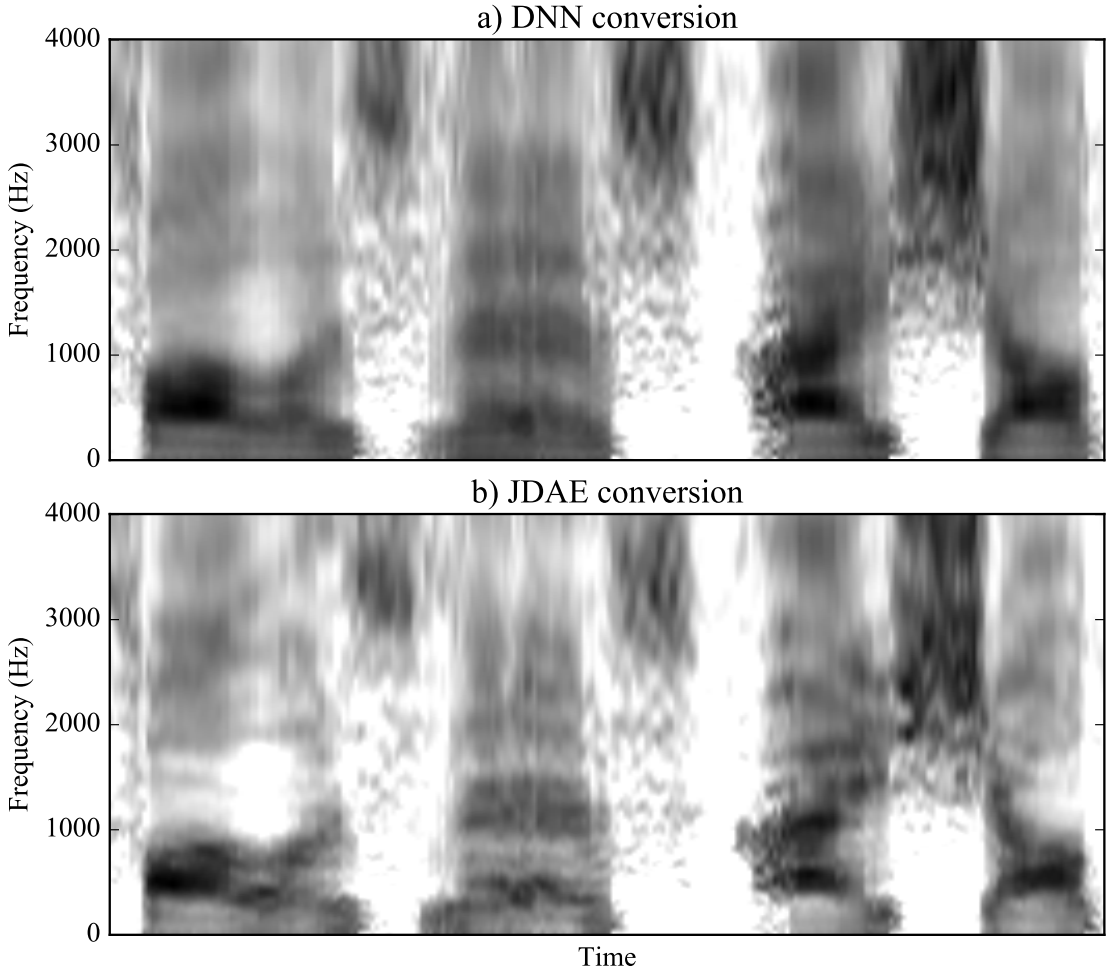


Figure 4.9: Converted MCEP spectrograms using DNN and JDAE

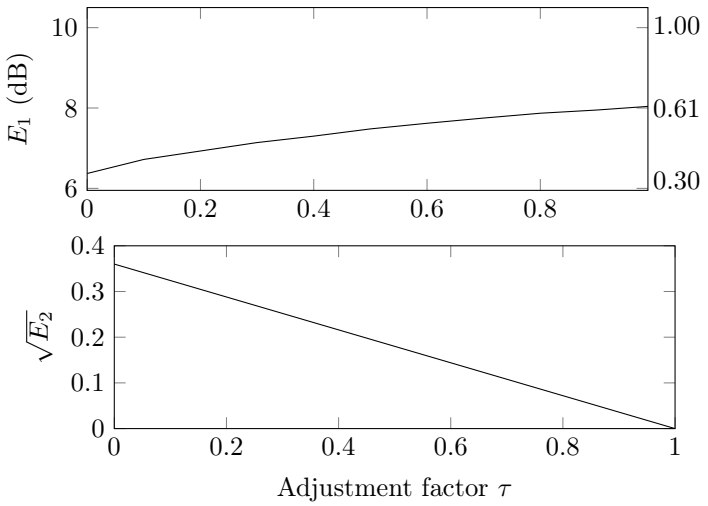


Figure 4.10: E_1 and E_2 for DNN features with adjusted standard deviation

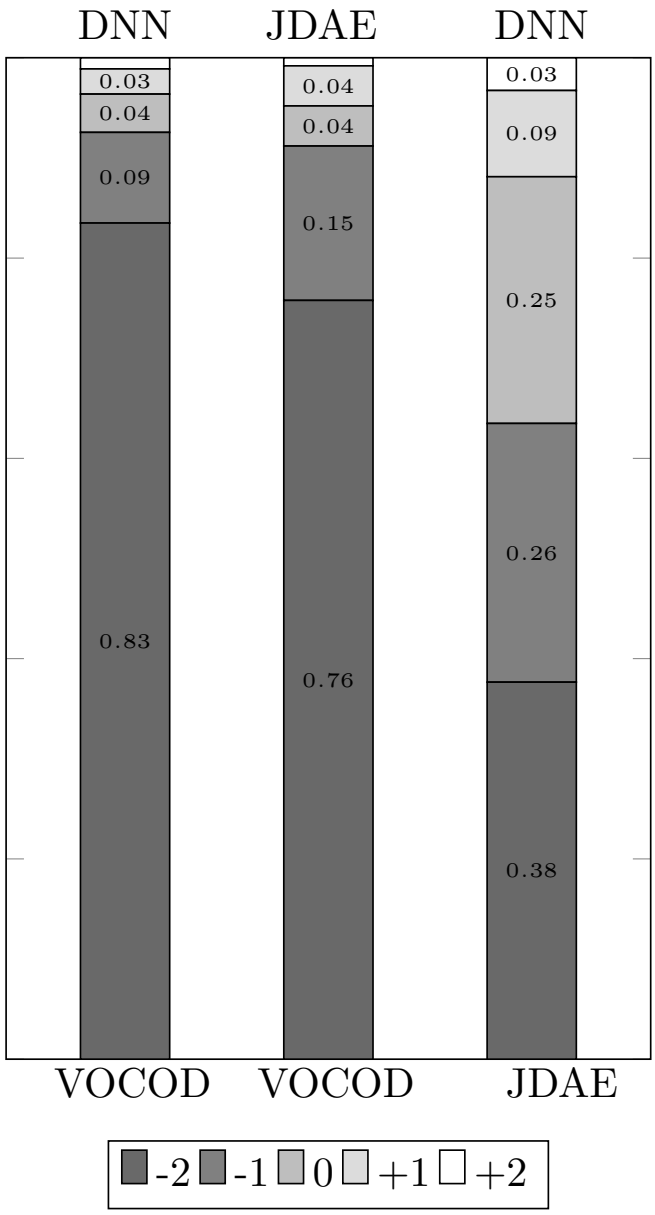


Figure 4.11: Speech quality preference test results comparing the two processing conditions. Negative scores favor the processing condition shown on the bottom, whereas positive scores favor the one shown on top.

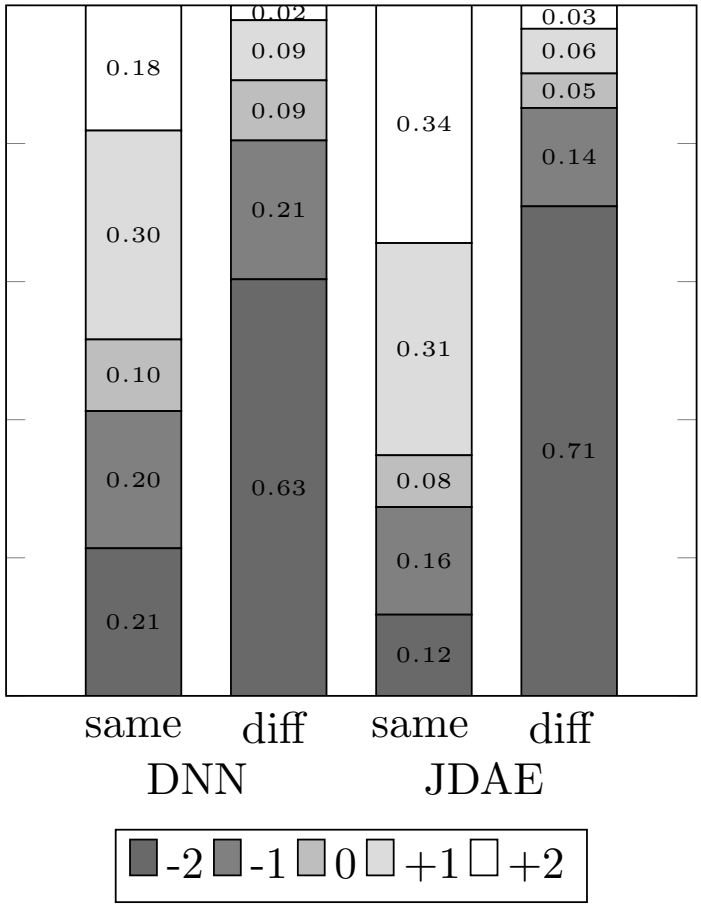


Figure 4.12: Speaker similarity test results comparing converted vs. target ("same") for the mimic task, and converted vs. source speaker ("diff") for the de-identification task. The negative and positive scores show the degree of confidence in that the samples are from different speakers or the same speaker, respectively.

Chapter 5

Decomposing Autoencoders

5.1 Introduction

Some categories of data can be modeled as a composition of style and content information that are independent of each other. Style and content decomposition has been the focus of numerous studies [289, 50, 337, 106, 311]. Speech can be thought to be (at a minimum) composed of speaker identity and linguistic content information. The availability of methods that *decompose* input speech into its underlying information embeddings can facilitate several use-cases. In certain applications, we are only interested in one of the information embeddings. For example, for text-independent speaker recognition [241], we are interested in the speaker identity embedding and do not require the knowledge of the linguistic content, whereas in speech recognition [235], we are interested in the linguistic content embedding which ideally should not be entangled with speaker-related information. The availability of methods that can *recompose* the decomposed embeddings to reconstruct the input data can facilitate several other use-cases. For example, VC [184] can be formulated as modifying the style of an speech utterance, in this case the speaker characteristics, while keeping the content --- the linguistic message --- unchanged; thus an utterance produced by a source speaker will be perceived as if it had been spoken by the target speaker. The goal of this study is to investigate neural network architectures that are able to achieve style and content decomposition and recomposition.

As described in detail in Chapter 3, an autoencoder is a type of artificial neural network typically used for unsupervised learning of efficient encodings, or representations [18]. Representations are learned by requiring the network to reconstruct the data in the presence of non-linear functions and noise corruption in the architecture; in this manner the network learns to extract useful patterns from the data in order to be able to encode and reconstruct them [335]. When modeling speech data, the representation typically contains a combination of several factors such as linguistic

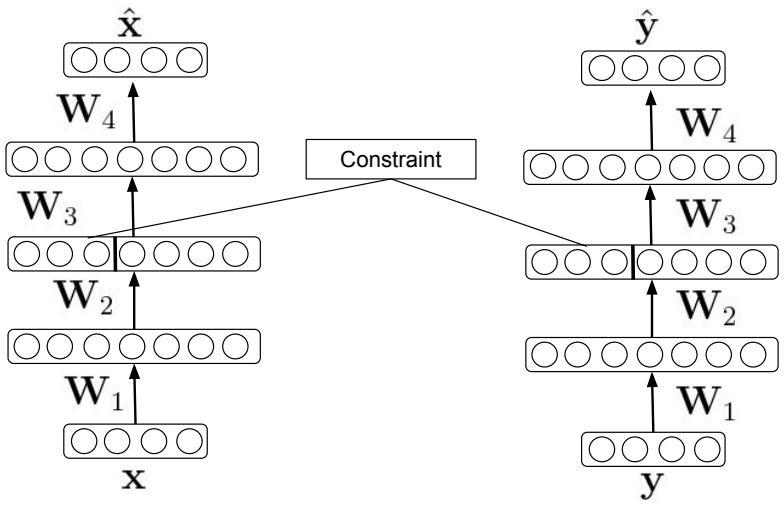


Figure 5.1: An exampleSiamese architecture

content, speaker identity, dialect, and emotional state. In this study, we aim to create an architecture that decouples speaker identity (defined as style hereon) and linguistic content (defined as content hereon) representations. We assume the availability of a database with different speakers uttering the same sentences, i. e. a database that has identical content information for different pre-determined styles. We model the hidden representation vector as the concatenation of a style vector and a content vector. This key concept leads to an architecture we term decomposing autoencoder (DcAE).

Siamese architectures have typically been defined as two tied-weight networks with two different inputs that are constrained to reconstruct their respective inputs. Furthermore, a constraint between the shared segment of a hidden layer in the two sub-networks is added to the loss function [24]. An example of a siamese architecture is depicted in Figure 5.1. As can be seen in the figure, both networks (which have tied weights) are reconstructing two separate samples from data. A constraint is also applied in a part of the middle layer in order to impose some constraint (which is use-case dependent); without this additional constraint the inputs could be simply presented as two distinct training samples, as is the case with common network architectures. We propose to learn the DcAE representations by using a multi-siamese DcAE architecture during training. In contrast to traditional Siamese techniques [24], we use multiple copies of the network rather than two copies. Furthermore, we impose constraints on all parts of the representation layer rather than only imposing constraints on a specific part of the representation layer. For each style, we construct a DcAE network, and then impose constraints between the style-specific sub-networks'

style and content representations. We propose two training approaches to train the network. In a first approach, we use a similarity constraint which is imposed on the content of the same frame of different speakers, while also simultaneously using a similarity constraint which is imposed on the styles of all frames by each of the speakers. This reflects the hypothesis that all content representations should be similar to each other, and that speaker representations should be similar over all frames of an individual speaker. In a second approach, we impose the constraints by modifying the training such that the content representations of each frame are mean-pooled over all speakers before decoding, and the style representations of each speaker are mean-pooled over all frames before decoding. The mean-pooling of content over speakers is performed by adding all the corresponding content representations and dividing the outcome by the number of speakers. This mean-pooled representation is then broadcasted to the decoding part of each speaker's autoencoder. Mean-pooling across style is performed similarly, but over frames of each speaker, and then broadcasted to the decoding part of that speaker's autoencoder. Mean-pooling content representations of a particular frame index prevents the content representation from learning any speaker-specific information; conversely, mean-pooling the style over all frames of a speaker removes the variability of the style representation.

To validate the model, we perform several visualizations, a phoneme classification experiment in Section 5.3, and a speaker recognition experiment in Section 5.4. Finally, we present the many-to-many VC application and experiments in Section 5.5.

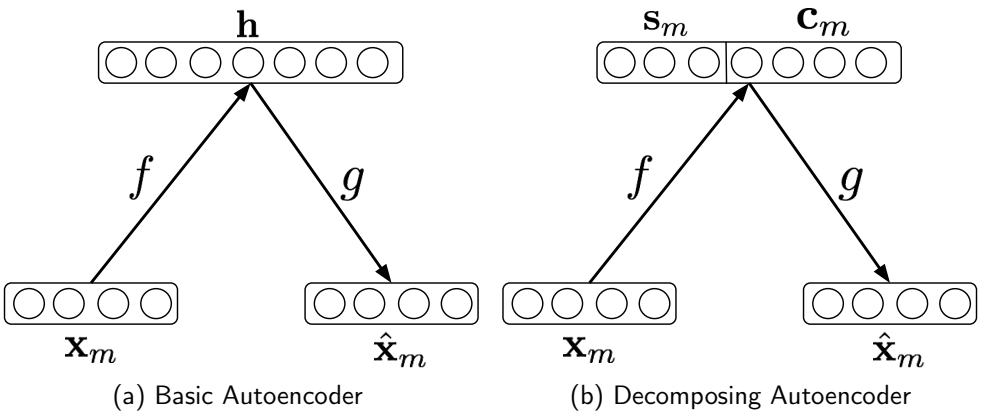
5.2 Formulation

5.2.1 Preliminaries

We will use the following notation: Let $\mathbf{X}^k_{M\times D} = [\mathbf{x}^k_1, \ldots, \mathbf{x}^k_M]^\top$, where the m^{th} content observation $\mathbf{x}^k_m = [x_1, \ldots, x_D]^\top$ represents the k^{th} data stream out of K parallel (i. e. content-aligned) data streams of M observations of D-dimensional training feature vectors.

- $m \in M$: Represents the observation index
- $k \in K$: Represents the style index
- $d \in D$: Represents the content feature index

For any given m, $\{\mathbf{x}^k_m\}$ are vectors with *distinct* style k (out of a total of K styles) and identical content. We omit the m subscript for ease of notation, unless explicitly specified. In this study, speech style represents speaker identity, and content represents the linguistic message.



(a) Basic Autoencoder (b) Decomposing Autoencoder

Figure 5.2: Basic Autoencoder versus proposed Decomposing Autoencoder

5.2.2 Decomposing autoencoder

A basic autoencoder (AE) is composed of an *encoder* $f(\cdot)$ and a *decoder* $g(\cdot)$. The network encodes the input to the hidden representation $\mathbf{h}_m = f(\mathbf{x}_m)$, and then reconstructs the input by decoding the hidden representation, $\hat{\mathbf{x}}_m = g(\mathbf{h}_m)$. Minimizing the reconstruction loss, $L_r(\mathbf{x}_m, \hat{\mathbf{x}}_m)$, forces the network to learn useful representation patterns during training, as shown in Figure 5.2a. To force the network to learn more generalized representations, the input is corrupted by adding noise; the resulting training method is called *denoising* autoencoders [335]. We extend the basic AE to force the network to learn reconstruction, but, in addition, learn the decomposition of style and content. The latter goal is achieved by splitting the hidden representation vector $\mathbf{h}_m$ into two sub-vectors $\mathbf{s}_m$ and $\mathbf{c}_m$, which represent style and content, respectively; i. e. $\mathbf{h}_m = [\mathbf{s}_m, \mathbf{c}_m]$, as shown in Figure 5.2b. For training the DcAE, we propose to use a Siamese architecture with certain constraints as described in the next two subsections. We describe constraining by a specific loss function and mean-pooling in Sections 5.2.3 and 5.2.4, respectively.

5.2.3 Decomposing Autoencoder constrained by loss function

For training the DcAE, we propose a novel Siamese autoencoder (SiAE) architecture, in which K AEs (one for each available style in the training data) have *shared* parameters, but operate on inputs that are parallel in content, but distinct in style. Specifically, the SiAE is composed of multiple copies of the encoder, mapping the k^{th} data stream to style and content representations $[\mathbf{s}_m^k, \mathbf{c}_m^k] = f(\mathbf{x}_m^k)$, and multiple copies of the decoder, reconstructing the data from style and content representations $\hat{\mathbf{x}}_m^k = g([\mathbf{s}_m^k, \mathbf{c}_m^k])$. This training approach, shown in Figure 5.3, learns to reconstruct the training data using a reconstruction loss function. However, importantly, the

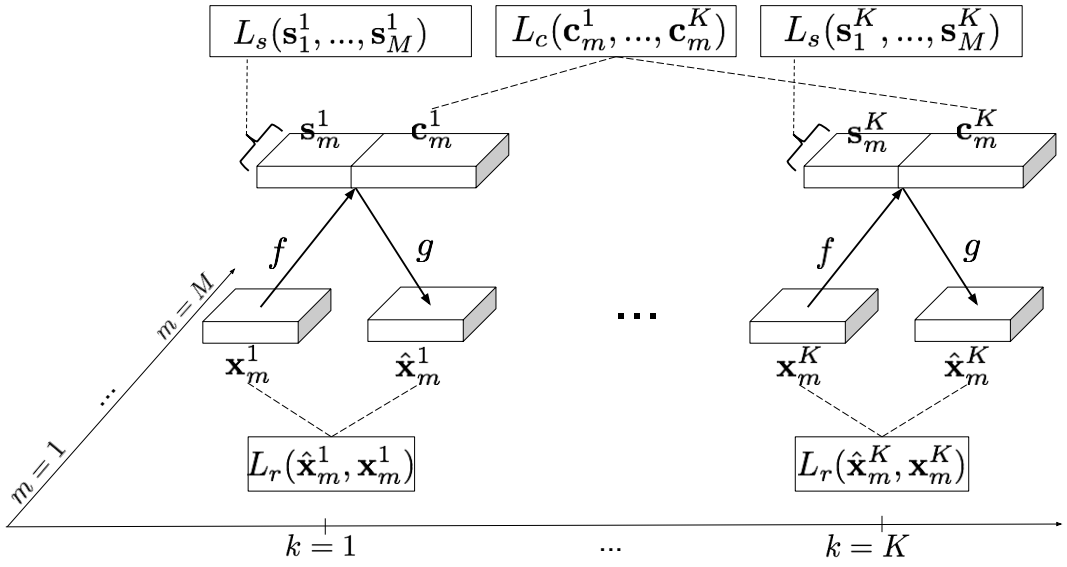


Figure 5.3: Siamese Autoencoders with loss imposed on style and content along their relevant factors

SiAE also allows constraints on both style and content by means of *additional* loss functions that are applied to the hidden representations $\mathbf{s}_m$ and $\mathbf{c}_m$, resulting in the total loss function (for a single observation) $L_r + L_c + L_s$, where $L_r = \sum_{k=1}^{K} L(\mathbf{x}_m^k, \hat{\mathbf{x}}_m^k)$, L_c, and L_s are the reconstruction, content, and style loss functions, respectively. Note that the SiAE architecture is used during training only, resulting in a nearly-standard AE architecture during testing: the DcAE.

We consider learning content by constraining the content representation using a *similarity* measure.

$$L_c = \sum_{m=1}^{M} \sum_{k=1}^{K} \sum_{l=k+1}^{K} L(\mathbf{c}_m^k, \mathbf{c}_m^l) \tag{5.1}$$

since all data streams have the same content.

We consider learning distinct styles by using a *similarity* measure over data samples. We posit that the average distance between all styles of a speaker's data samples should be minimized while learning the speaker's style space.

$$L_s = \sum_{k=1}^{K} \sum_{i=1}^{M} \sum_{j=i+1}^{M} L(\mathbf{s}_i^k, \mathbf{s}_j^k) \tag{5.2}$$

We call this architecture SiAEloss, shown in Figure 5.3.

5.2.4 Decomposing Autoencoder constrained by mean-pooling

In this section, we describe constraining the Siamese autoencoders to learn to distinguish style and content by *mean-pooling* hidden style and contents across their relevant factors. We propose to modify the style representation by mean-pooling the computed styles of all samples in a batch *before decoding*. The reason is that we want to compute unique style values for each style that are invariant to the content. Thus, we modify the decoder of the autoencoder to use the mean-pooled over content observations style:

$$\bar{\mathbf{s}}^k = \frac{1}{M} \sum_{m=1}^{M} \mathbf{s}_m^k \tag{5.3}$$

For training, we consider a large batch M gradient approach since the more samples we have during parameter update for style, the more stable and less variable the styles will be. We also propose to modify the content representation by mean-pooling all K content representations of different styles:

$$\bar{\mathbf{c}}_m = \frac{1}{K} \sum_{k=1}^{K} \mathbf{c}_m^k \tag{5.4}$$

We reconstruct the input by feeding the mean-pooled style and content to the decoder:

$$\hat{\mathbf{x}}_m^k = g([\bar{\mathbf{s}}^k, \bar{\mathbf{c}}_m]) \tag{5.5}$$

Note that we do not impose any explicit constraints in the loss function on either style or content, since the mean-pooling itself acts as a form of constraint. We merely use the reconstruction loss L_r. We call this architecture SiAEpool, shown in Figure 5.4.

5.2.5 Related work

5.2.5.1 Siamese Networks

Similar to our proposed architecture, Siamese deep networks have been proposed previously by Chen et al. [28]; in their work the architecture consisted of two Siamese feed-forward networks in which only the speaker-specific part of a middle layer was constrained. The goal of that study was to compute a speaker representation from non-parallel data, and not content representation. However, in our study we propose using multiple Siamese representation learning networks, and constrain content encodings by using a parallel training speech corpus which contains multiple speakers speaking the same text material, thus providing training data with both content and style factors.

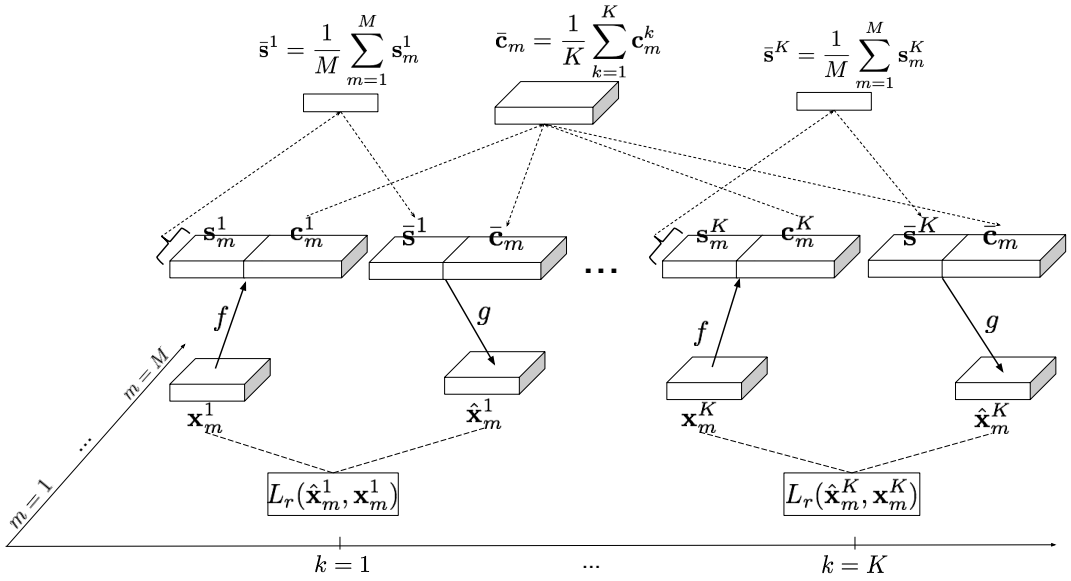


Figure 5.4: Siamese Autoencoders with mean-pooled style and content along their relevant factors

Other studies used tri-Siamese autoencoders for phone and speaker embedding learning [367, 368]; however, these approaches were supervised using phoneme and speaker labels. In contrast, we propose to learn both speaker and linguistic content without labels, using the parallel structure of the corpus (which is a form of supervision). We also propose to extend the Siamese architecture to *multiple* copies of the autoencoders, one per style, rather than merely two or three copies.

5.2.5.2 Multi-view Learning

The proposed work has some similarities to multi-view learning [339], in which a shared representation is learned from multiple data streams. Their approach hypothesized that all of their data have the same or highly correlated latent representation which generates the different views of the data. In our study, we have a similar assumption, in that we assume these different data stream observations are generated from the same content latent representation. That is the reason for choosing to mean-pool the content representations in one of the training approaches. One difference of this technique as compared to our proposition is that our work assumes shared parameters for all data views and also explicitly models the uniqueness of the different views (style) compared to multi-view approaches which have separate parameters for each data stream. The multi-view learning goal is typically to achieve good quality latent representations from multiple data modalities, whereas here we focus on how to decouple different factors of the data and modifying and recomposing the factors to achieve feature modification (voice conversion in this study).

5.2.5.3 Factorized Hierarchical Variational Autoencoders

A recently proposed approach learns disentangled and interpretable representations using variational autoencoders [106]. The authors propose to explicitly formulate the sequential information within a factorized hierarchical graphical model. The proposed model imposes sequence-dependent priors and sequence-independent priors to different sets of latent variables. This approach is able to decompose speech factors from a multi-speaker speech corpus without requiring parallel (same linguistic content) sentences from the speakers. In contrast, our approach requires parallel sentences from the speakers. In addition, they impose speaker dissimilarity constraints on the speaker style representation and the convergence is very sensitive to the contribution of this constraint to the loss function. Furthermore, the variational autoencoder approach has limitations in learning to neutralize any recording environment differences in a training speaker's recordings since it is unsupervised. However, our proposed approach can neutralize any variations in a training speaker's recording environment since it is imposing speaker embedding similarity constraints *explicitly* and can potentially result in more accurate speaker embeddings.

5.2.5.4 Discriminative Autoencoders

A recently proposed approach learns phonetic and speaker representations using discriminative autoencoders [106, 361]. The goal of the study is to improve the performance of phoneme classification in ASR. The authors propose to impose a supervised phonetic loss on the phonetic representation (as is done in typical DNN-based ASR acoustic models), and additionally, impose a reconstruction loss (as in typical autoencoder models), and a supervised and an unsupervised speaker loss on a newly introduced speaker representation. They show that when the speaker discrimination losses are added to the loss function, the model performs better in terms of phone error rate. They do not study the properties of the learned representations or the reconstructed features.

5.2.6 Training

During training, we sample K speakers from the total available speakers. We also sample a batch of M samples from the training samples. Note that sampling of both speakers and the data batch are both due to computational constraints. For example, we would have a computational graph that grows quadratic in terms of K. The time and space complexity is linear in terms of M for SiAEpool, and quadratic for SiAEloss in terms of M.

For training the models, we used the VCTK corpus which includes speech data uttered by 109 native speakers of English with various accents [330]. Each speaker read approximately 24

parallel sentences, selected from the Rainbow Passage and an elicitation paragraph. Due to the unavailability of all of the parallel sentences for all of the speakers, we excluded file 001 and 015 from consideration, resulting in 82 speakers with 22 parallel sentences. The audio materials were downsampled to 16 kHz with 16-bit precision for further processing to match further experiments. We used 39^{th} order Mel-Cepstrum (MCEP) features extracted by the World vocoder [189]. We used 5 ms frame shifts. We used a 0.005 learning rate and a value of 0.9 for momentum. All input features were mean and standard deviation normalized using the computed values from the training data. The representation layer used a sigmoidal activation function. The output decoding layer used a linear activation function. All other hidden layers used ReLU activation functions. We trained for 1000 iterations and selected the network with the smallest validation error.

5.3 Experiment: Phoneme Recognition

To evaluate the quality of the learned embeddings, we designed a phoneme recognition experiment. We compared the performance of using raw MCEP features versus using the embeddings computed using DcAE. We used the TIMIT corpus and its suggested training/testing split to train and evaluate the model. We excluded all SA sentences which have the same content for all speakers. We compressed the 61 unique phonemes present in TIMIT corpus transcriptions to 39 using a Kaldi script[1]. For this experiment, we also computed the transition probabilities between phonemes from training data and performed a Viterbi search on the posterior probabilities of the phoneme classifier during testing to smooth them. We used a phoneme bigram to model the transition probabilities. The goal is to reduce jumps that might happen due to selecting the phoneme with maximum probability without respecting the transition probabilities. We considered the following systems:

- **MCEP**: A feed-forward DNN of size [40,500,500,39] with ReLU activation functions trained using a cross-entropy loss function on frame-wise labeled data. The inputs are MCEP features. The labels are one-hot vectors representing vowel identity. The design is similar to Section 3.4.
- **LOSS-C/POOL-C**: A feed-forward DNN of size [$|C|$,500,500,39] where $|C|$ is the content embedding size. The DcAE was trained using both the DcAE-LOSS and the DcAE-POOL approach. This system evaluated how much content information is retained in the content embedding.

[1] https://github.com/kaldi-asr/kaldi/blob/master/egs/timit/s5/conf/phones.60-48-39.map

Features	Accuracy
MCEP	68.97%
LOSS-C	64.03%
POOL-C	66.52%
LOSS-S	27.30%
POOL-S	23.94%
Chance	6.03%

Table 5.1: Percentages of correctly classified frames in the phoneme recognition experiment

- **LOSS-S/POOL-S**: A feed-forward DNN of size [$|S|$,500,500,39] where $|S|$ is the style embedding size. The DcAE was trained using both the DcAE-LOSS and the DcAE-POOL approach. This system evaluated how much content information remained in the style embedding.

- **Chance**: A randomly-assigned label according to the training label priors.

We used $|C| = 32$ and $|S| = 32$. The results are shown in Table 5.1. The results show that the content embedding computed using the POOL approach preserves phonetic information better than the LOSS approach. This is also reaffirmed by comparing POOL-S and LOSS-S, since POOL-S is preserving less phonetic information. However, POOL-C and LOSS-C are not retaining all phonetic-related information, since raw MCEPs are still outperforming them. One cause of this phenomenon might be misalignments in the time-alignment step of the parallel corpus. The differences in pronunciation or mispronunciations might result in the DcAE being forced to learn similar content embeddings for phonetically different acoustic segments. This might cause the network to not perform optimally in regards to content embedding and/or rely on the speaker embedding to learn the differences in contents of different data streams. Interestingly, it appears that there is still some phonetic information present in style encodings since classification accuracy is significantly higher than chance.

For validating the architecture further, we visually inspect the content embeddings of the DcAE. We use the 22 sentences of the four test speakers from the VCTK corpus that have parallel utterances. We perform a principal component analysis (PCA) over all of the speaker content encodings, depicted in Figure 5.5. As shown in the figure, the distribution of these encodings for different speakers are similar; moreover the three emphasized time-synchronous data points fall in the same regions for different speakers, further validating this visually. To objectively assess the closeness of the content features by different speakers, we computed the PCA of the features,

mean and variance normalized them, and finally computed the root mean square error (RMSE) of the features over all speakers in a pair-wise manner. We computed this measure for DcAE-LOSS content encodings, DcAE-POOL content embeddings, and MCEP features to be 0.243, 0.237, and 1.578, respectively. The order or magnitude difference demonstrates that the content encoding values of the speakers are closer to each other compared to MCEP features, showing that the similarity constraint is successfully imposing the content encodings to learn similar encodings.

5.4 Experiment: Speaker Recognition

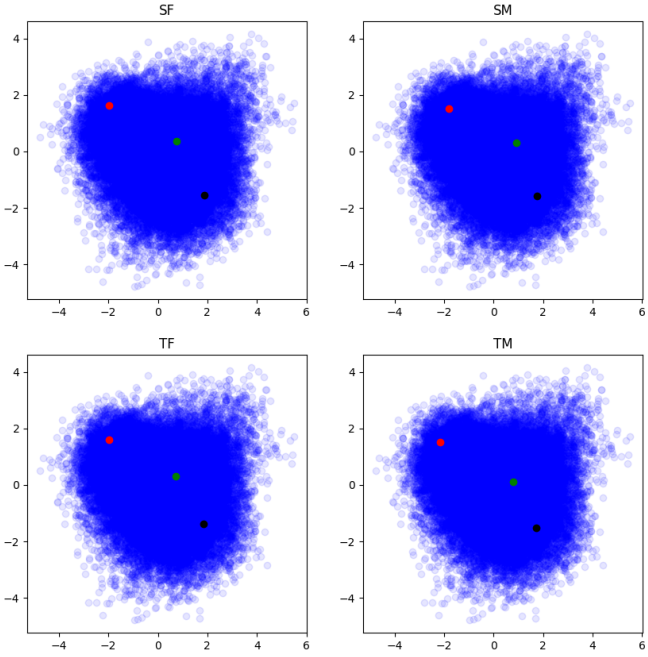
We set up a speaker recognition experiment to investigate the quality of the learned speaker embedding. We selected a closed-set speaker recognition task in which we only queried within-set speakers. We used test speakers from the TIMIT database [69] to perform this experiment. For each speaker, we used one sentence for enrollment and one sentence for testing (randomly chosen from non-SA sentences). We report the Equal Error Rate (EER). We considered the following systems:

I-Vector This system is an I-vector approach for learning the I-vector for each speaker as the baseline. I-vectors are a low dimensional subspace of the mean supervectors of a Gaussian mixture model (GMM). The GMM is the universal background model (UBM). An I-vector system $s = m + Tw$ recreates speaker supervector s using a speaker-independent vector m and a set of low-dimensional total variability factors w to represent each utterance. Each variability factor w controls an eigendimension of the total variability matrix T, and these w are known as the I-vectors. The m and T values are computed from training data, and a new w is obtained during testing for an input utterance [41]. We used the Cosine distance criterion to find the closest speaker.

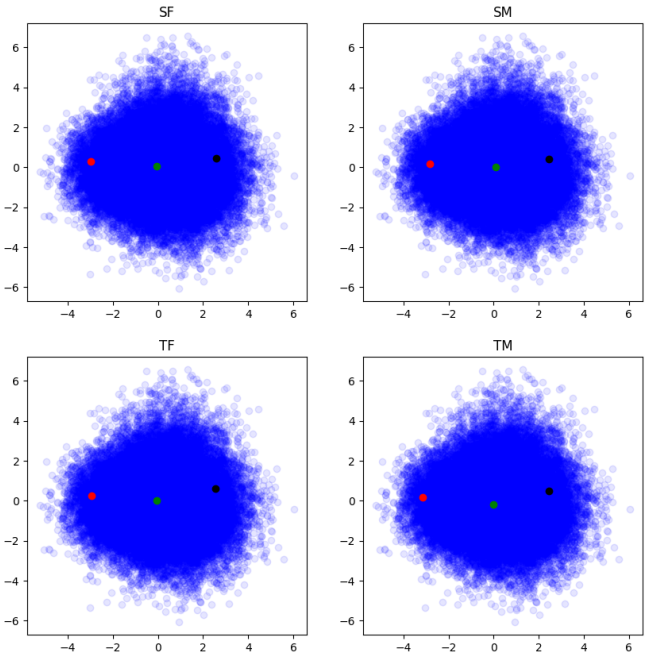
LOSS-S/POOL-S This system is the DcAE trained using the DcAEloss/DcAEpool approach. We used the Cosine distance criterion to compare the speaker embeddings of the the input speaker against the enrolled speakers in order to find the closest match.

LOSS-C/POOL-C Similar to the above condition, except the goal is to evaluate the amount of speaker information that remains in the content embedding.

We also investigated Linear Discriminant Analysis (LDA) to reduce the dimensionality of the I-vector and DcAE embeddings. The results, shown in Table 5.2, show that, with LDA applied, POOL-S had similar performance to the I-vector approach. Furthermore, the POOL approach



(a) DcAE-LOSS computed content embeddings



(b) DcAE-POOL computed content embeddings

Figure 5.5: 2D PCA plots of computed content encodings of the test frames of 4 testing speakers using DcAE-LOSS (top) and DcAE-POOL (bottom). Three time-synchronous data points are emphasized.

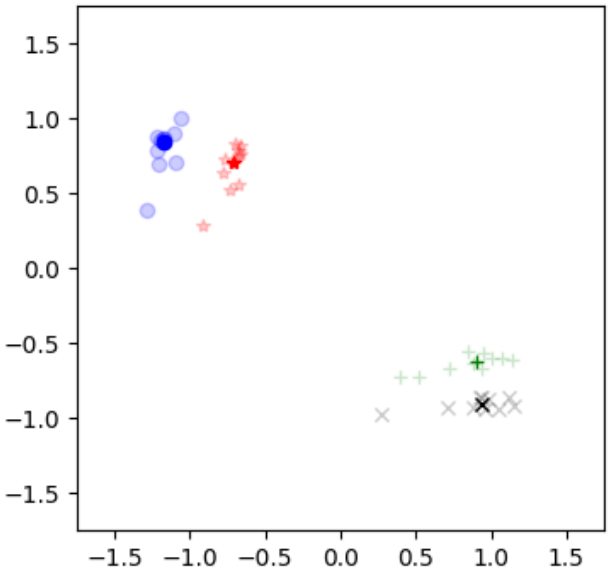
Features	Dimension	Raw	LDA (12 dim)
I-vector	50	90.08%	93.74%
	100	90.58%	93.9%
POOL-S	16	91.81%	93.65%
	32	91.73%	93.82%
LOSS-S	16	91.24%	92.81%
	32	91.37%	92.75%
POOL-C	16	18.8%	23.34%
	32	18.48%	24.29%
LOSS-C	16	23.20%	26.21%
	32	22.55%	26.70%

Table 5.2: Percentages of correctly classified utterances in the speaker recognition experiment

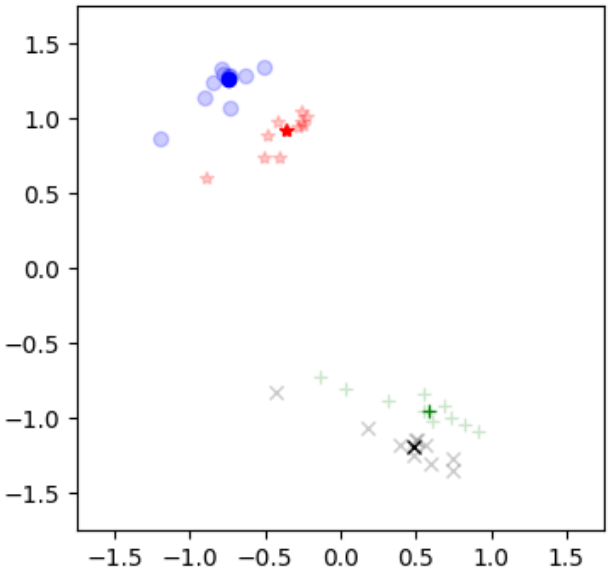
once again performed better than the LOSS approach when style embedding is used. When the content embedding is used, LOSS performed better, suggesting that the LOSS approach is not separating style and content as well as the POOL approach. These results verify the effectiveness of DcAE in separating style and content. The POOL approach is providing a method that provides an embedding with information with identical performance as I-vectors, with the benefit of allowing modification and feature reconstruction to achieve VC (the next experiment). We also visually studied the computed style embeddings of the DcAE. In Figure 5.6 we show the 2D PCA plot of computed styles from the utterances of the 4 testing speakers using DcAE-LOSS (left) and DcAE-POOL (right) techniques. It is evident that the embeddings of each speakers are forming their own cluster. Furthermore, the same gender speakers are closer to each other and further apart from the other gender. Finally, the within-class distances in the speaker clusters formed using the DcAE-POOL model seem to be more concentrated as compared to the DcAE-LOSS approach, a desired feature. However, there is still variability present in the computed styles which shows there is room for improvement in the model and training.

5.5 Experiment: Many-to-Many Voice Conversion

We performed a VC experiment which aimed to modify the voice of a given utterance to sound like a target speaker. We used four conversion conditions: M2M, M2F, F2M, and F2F. We considered the following methods:



(a) DcAE-POOL computed style embeddings



(b) DcAE-LOSS computed style embeddings

Figure 5.6: 2D PCA plot of computed styles from the utterances of the 4 testing speakers using DcAE-LOSS (left) and DcAE-POOL (right) techniques. The bold dots are computed from 10 utterances and the transparent dots are computed from 1 utterance. SF: Red, SM: Green, TF: Blue, TM: Black.

DNN As our baseline, we used a feed-forward DNN of size [39, 500, 500, 39] with ReLU activation functions trained using a mean-squared error loss function on frame-wise aligned data.

LOSS/POOL As our proposed system, we used the DcAE, trained using the SiAE approach. We used theDcAE to compute the style of the target speaker from the training data. We then used the DcAE to decouple content and style embeddings of a given utterance, and, while keeping the content embedding the same, replaced the style with that of the target's. Note that the DNN approach works only for a given source-target pair, but the DcAE method works for any arbitrary input and enrolled target speaker.

For performing VC, we compute the average speaker embedding $\bar{\mathbf{s}}$ from the training utterance(s) of source and target speakers. For a given input utterance, we compute $\mathbf{c}_{inp}$ and $\mathbf{s}_{inp}$ of the input utterance. There are two approaches one can take to perform voice conversion: We can either replace $\mathbf{s}$ values of the source speaker with the average $\bar{s}_t$ from the target speaker as $\mathbf{s}_{cnv} = \bar{\mathbf{s}}_t$, or we can compute a difference vector between source and target averages $\bar{\mathbf{s}}_{dif} = \bar{\mathbf{s}}_t - \bar{\mathbf{s}}_s$. This difference vector is added to $\mathbf{s}_{inp}$ from the input utterance as $\mathbf{s}_{cnv} = \mathbf{s}_{inp} + \bar{\mathbf{s}}_{dif}$, leaving observational variances in $\mathbf{s}_{inp}$ intact. The speech features are obtained by decoding with the DcAE decoder using $\mathbf{s}_{cnv}$ and $\mathbf{c}_{inp}$. By performing some informal experiments, we found that the first, or replacement approach resulted in more muffled generated speech than the second, or difference vector approach. Due to this reason, for the VC experiment in Section 5.5 we used the difference vector approach.

Furthermore, we consider the following four speaker enrollment conditions:

- ss-ts: source is seen (in-training), target is seen (in-training)
- ss-tu: source is seen, target is unseen
- su-ts: source is unseen, target is seen
- su-tu: source is unseen, target is unseen

In the unseen source or target conditions, we exclude the two source or target speakers, one per gender, from the training data, as we intended for the network to see the same number of speakers. For example, in the su-tu case, we exclude four speakers from the training data. To compensate for the excluded speakers, we added two source or target substitute speakers to the training data. From our 82 speakers, we selected four test speakers, and four substitute speakers. We used the rest of the 74 speakers as training data. We selected the first 10 parallel utterances from the speakers for learning the voice mappings and embeddings.

Model	MelCD
AE	4.54
LOSS	4.97
POOL	5.10

Table 5.3: Feature reconstruction of various models, we report the results using melCD (dB)

5.5.1 Objective experiments

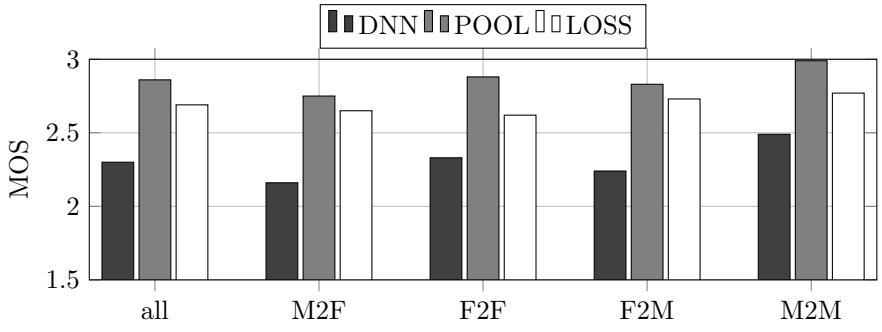
We first investigated the ability of the model to reconstruct the features. We compared DcAE-LOSS and DcAE-POOL feature reconstruction with an unconstrained AE; the resulting melCD [303] is shown in Table 5.3. We used the su-tu speaker enrollment condition and averaged over all conversion conditions. We compared the models to a regular AE with similar architecture to those in Section 5.3 and the representation size of $|C| + |S|$. We observed that the unconstrained AE achieved lower reconstruction error which is due to it being trained in an unconstrained way. The LOSS approach achieved a lower feature reconstruction error which might be due to the LOSS constraints being less restrictive compared to that of the POOL approach. This reason is also consistent with the lower performance of LOSS compared to POOL in the phoneme recognition and speaker recognition tasks. Furthermore, we performed objective experiments for the VC task to observe the effect of mapping. We report the melCD in Table 5.4. The results show that in the ss-ts case, both LOSS and POOL approaches performed better than the baseline DNN. However, when one of the speakers is unseen, either DcAE approaches perform worse than the DNN approach. It is important to note that DNN can not handle unseen speakers at all, and the input has to always be the source speaker. Increasing the number of parallel training data might help learning more speaker-independent representations and improve the VC scores. Finally, POOL consistently outperformed LOSS which is consistent with phoneme recognition and speaker recognition performance results, again validating that POOL is able to compute better quality embeddings compared to LOSS. We performed planned t-test and notably find statistically significant differences in the following comparisons: LOSS-ss-ts vs. DNN, POOL-ss-ts vs. DNN, LOSS-ss-ts vs. POOL-ss-ts, LOSS-ss-ts vs. LOSS-su-tu, and POOL-ss-ts vs. POOL-su-tu.

Model	ss-ts	ss-tu	su-ts	su-tu
LOSS	8.58	8.78	8.82	9.07
POOL	8.37	8.60	8.67	8.95
DNN		8.72		

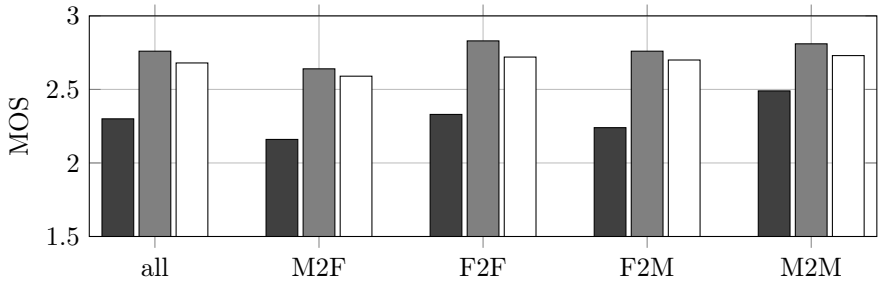
Table 5.4: Voice conversion objective measure, we report the results using melCD (dB)

5.5.2 Subjective experiments

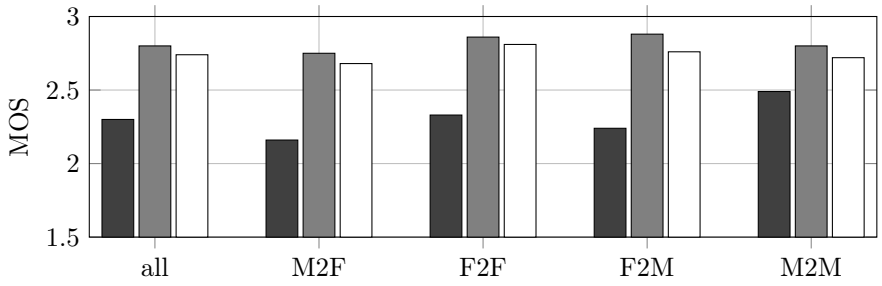
We performed subjective experiments to see the effect of mapping using the DcAE. We considered the following conditions: DNN, LOSS (all four seen/unseen conditions), POOL (all four seen/unseen conditions). Each condition has four gender conversion conditions. We used 10 utterances to compute the target speaker embeddings. We used 36 test utterances from the source speaker, and thus generated $(4 + 16 + 16) \times 36 = 1296$ stimuli. We computed the source speaker embedding from each individual input utterance. We added three high quality, non-modified audio samples to the listening test to find and exclude any unreliable speakers. We performed a speech quality experiment to assess the naturalness of the VC systems for the above-mentioned system architectures. We used 36 listeners for each experiment; each listener was presented 36+3=39 stimuli. We played a stimuli for the listeners and asked them to rate the quality from 1 (very bad) to 5 (very good), the Mean Opinion Score (MOS). We used four conversion pairs: SM1→TF1, SF2→TM1, SF1→TF2, and SM2→TM2. The quality results are shown in Figure 5.7. The four speaker enrollment conditions are shown in the subfigures. We performed planned t-tests to assess the statistical significance of the differences. The results show that both LOSS and POOL average scores significantly outperform the DNN in all four enrollment conditions. We found POOL scores to be consistently, but only marginally, better than LOSS. We also performed a speaker similarity experiment by playing two stimuli for a listener, one target and the other converted. The listeners are asked to score from 1 (the utterances are definitely from different speakers) to 5 (the utterances are definitely from the same speaker). We used the 1296 stimuli from the previous experiment. We also included the same number of converted utterances but with different same-gender target speakers. Hence, we created 1296 × 2=2592 stimuli pairs. Additionally, we added three same-speaker non-modified audio samples to the listening test to find and exclude any unreliable speakers. We used 36 listeners, each judging 36x2+3=75 stimuli pairs. We report the scores of stimuli with converted and original target speakers. The speaker similarity results are shown in Figure 5.8. The four speaker enrollment conditions are shown in the subfigures. We performed planned t-tests to assess the statistical significance of the differences. The results show



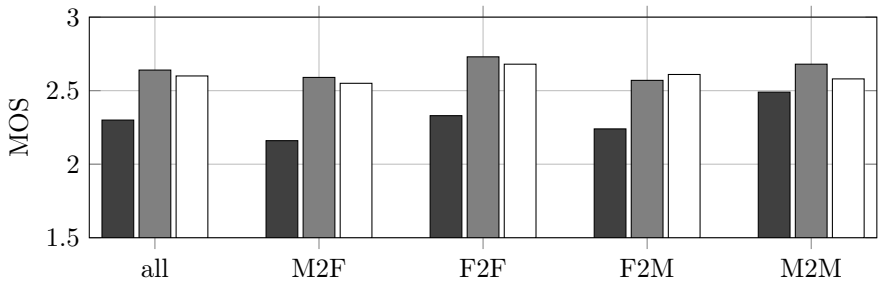
(a) ss-ts: source is seen, target is seen



(b) ss-tu: source is seen, target is unseen



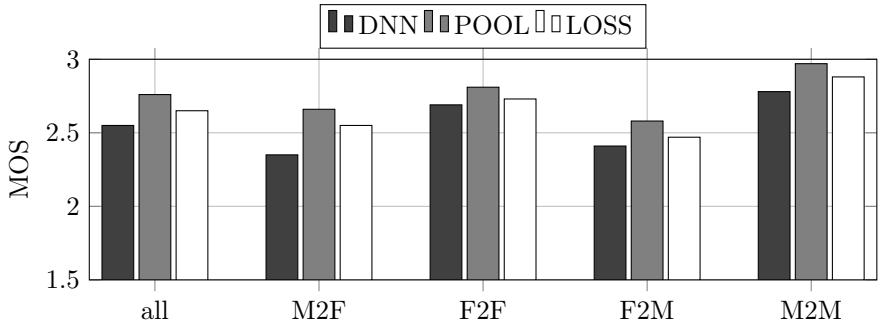
(c) su-ts: source is unseen, target is seen



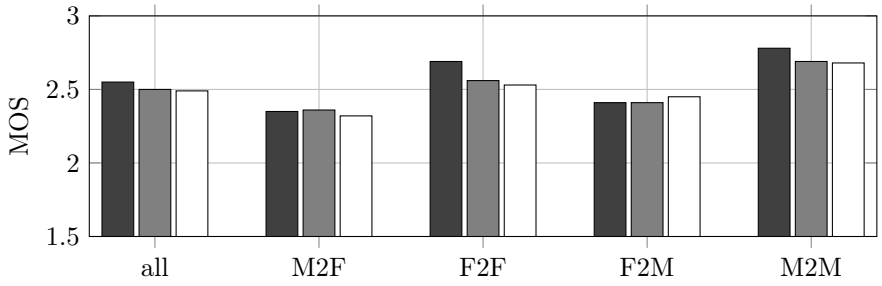
(d) su-tu: source is unseen, target is unseen

Figure 5.7: Speech quality MOS scores

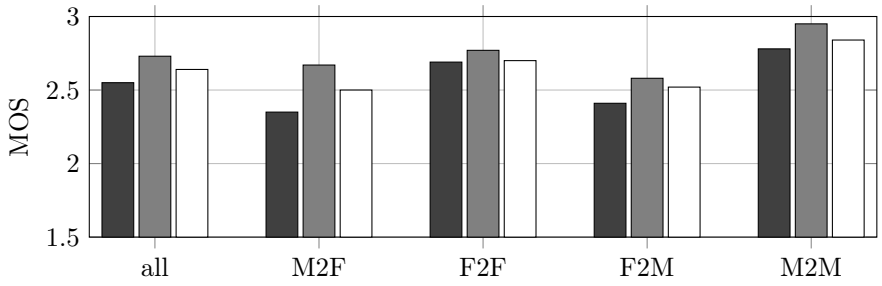
that when a target is seen, POOL significantly outperformed the DNN. We also found marginal but consistent improved performance of LOSS over DNN, and POOL over LOSS. In the ss-tu case, performance is not significantly different. In the su-tu case, the DNN is performing marginally but consistently better than LOSS and POOL. We hypothesize that by adding more training speakers, we could improve the speaker embedding quality and close the gap between enrollment conditions. Furthermore, we see that POOL and LOSS approaches both have consistent quality improvements, which provides a tradeoff for choosing the method based on the use-case. Finally, it is important to note that the DcAE can perform with even a single utterance of either source or target speakers, whereas the DNN is not able to perform reasonably given less than ten utterances. The reason is that the DNN has to be trained using data that has full phonetic coverage, whereas the DcAE can only use data with partial phonetic coverage to learn the speaker embedding and then perform embedding modification to achieve conversion.



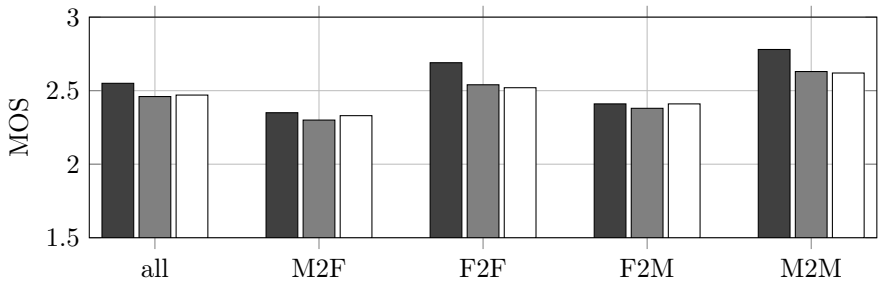
(a) ss-ts: source is seen, target is seen



(b) ss-tu: source is seen, target is unseen



(c) su-ts: source is unseen, target is seen



(d) su-tu: source is unseen, target is unseen

Figure 5.8: Speaker similarity MOS scores

Chapter 6

Summary and Future directions

6.1 Discussion of Contributions

- Deep autoencoder training: We proposed two training approaches: progressive training and deeply-supervised training. In progressive training, each new layer is trained in isolation first, and then all the layers are jointly trained together. We start from the lowest layer and iterate until the desired depth is reached. In deeply-supervised training, we proposed to use an architecture in which all autoencoder levels have their loss contribute to the total loss, hence all levels are trained simultaneously. This is motivated by an idea from deeply supervised networks in which each layer has a connection to the output, hence helping the error to propagate to the lower layers of the network. We show that we gain significant improvements in phoneme recognition and voice conversion.

- **Joint-representation learning**: In the one-to-one parallel VT task, we proposed a joint-autoencoder architecture in which the model builds two separate autoencoders for source and target speakers, and simultaneously tries to minimize the representation layer distance between them. Using objective experiments, we showed that the generated features have closer variance to original target speaker features. This results in a perception of less muffling in the converted speech.

- **Decomposing representations**: We proposed a model which is able to decompose linguistic and speaker identity factors from low-level acoustic features. Specifically, we proposed an autoencoder which learns to decompose the factors in the representation layer. To train this decomposing autoencoder, we utilize a siamese architecture, where in addition to reconstruction costs, specific other costs are imposed on linguistic and speaker identity representations. We also proposed a modification of this approach where we imposed the constraints directly

as part of the model architecture and only consider reconstruction loss.

6.2 Future Work

6.2.1 Future work of thesis contributions

In investigating the contributions of this dissertation, we studied models that use fully-connected feed-forward layers. A possible future direction is to use sequence modeling such as recurrent or convolutional models to capture context in speech feature sequence. Some future works for the contributions of the dissertation are listed below:

- Deep autoencoder training: We proposed two training approaches for deep autoencoders: progressive training and deeply-supervised training. For the deeply-supervised approach, we can explore the possibility of assigning different weights to different level cost function. Furthermore, the weights assigned to different levels' loss functions can be adjusted based on the iteration number during the training. For example, lower levels can be assigned higher weight initially, and gradually as training progresses, that level's loss function weight is decreased. Similarly, higher levels can he assigned lower loss function weights initially, and gradually, as the training progresses, the weight is increased. One can see the similarities with the progressive training approach if weights are set to 0 or 1; in other words, the progressive approach is equivalent to the deeply-supervised training with binary loss function weights.

- **Joint-representation learning**: We proposed a joint-autoencoder architecture in which the model builds two separate autoencoders for source and target speakers, and simultaneously tries to minimize the representation layer distance between them. A possible next step is to use *multiple* speakers rather than just one source and one target speaker. This allows us to learn a joint representation for several speakers, allowing multi-way VC. Furthermore, if enough speakers are enrolled, we can learn a speaker-independent representation.

- **Decomposing representations**: We proposed a model which is able to decompose linguistic and speaker identity factors from low-level acoustic features. Different noise or environmental conditions could also be considered as an extra representation in addition to the speaker and linguistic representations.When considering only linguistic and speaker representations, an identical speaker who is speaking in different noise conditions might have very different speaker representation and may be considered as different speakers. Adding a representation for the noise or channel might allow the model to factor this out. The data

could be artificially created by adding various noise types and intensities to the corpus.

6.2.2 Continuing Challenges of VC systems

Many unsolved problems exist in the area of VC. Some of them have been identified in previous studies [35, 147, 268, 261, 167]. As concluded in the VC Challenge 2016, there is still a significant performance gap between the current state-of-the-art performance levels and end-user expectations [308]. There are a lot of similarities between components of VC and statistical TTS systems, since both aim to generate speech features and synthesizing waveforms [161]. Consequently, some of the challenges and issues are shared in both systems.

Analysis/Synthesis issues: One major VC component that limits the quality of the generated speech is the analysis/synthesis aspect. STRAIGHT is a high-quality vocoder, but compared to natural speech, there is a still a quality gap [135]. Recently, new high-quality vocoders were proposed, such as Ahocoder [60] and VOCAINE [2], both of which have shown improvements in statistical TTS. Recently, several first attempts for direct waveform modeling using neural networks for statistical parametric TTS were proposed [309, 142, 65, 328, 89, 282]. These efforts may be a first step towards a new scheme for speech modeling/modification; however, the situation in VC is different since we have access to a valid source speaker utterance, which potentially allows copying certain aspects of speech without modifications. Some initial approaches of applying Wavenet to VC have been investigated [143].

Feature Interpolation issues: To represent spectral envelopes, various features are used, such as spectral magnitude, all-pole representations (LSFs, LPCs), and cepstral features. One major issue with these features is that interpolating two spectral representations may not result in spectral representations that are generated by the human vocal tract. For example, when using cepstra, if we interpolate two different vowel regions, the outcome would sound as if the two original sounds are occurring at the same time, and not as a single sound that lies perceptually between the two initial vowels. This limitation is one of the reasons for over-smoothing when multiple frames are averaged together. A spectral representation that represents meaningful features are formants locations and bandwidth. The two major problems of this representation is that formant extraction is still an unsolved problem, especially in noisy environments, and the inability of formants alone to represent finer spectral details.

One-to-many issues: The one-to-many problem in VC happens when two very similar speech segments of the source speaker have corresponding speech segments in the target speaker

that are not similar to each other. As a result, the mapping function usually averages the generated features in order to be similar to both target speech segments. Some studies have attempted to solve this problem [195, 92, 82].

Over-smoothing issues: In most VC approaches, the feature mapping is a result of averaging many parameters which results in over-smoothed features. This phenomenon is a symptom of the feature interpolation issue and one-to-many issue in combination. This effect reduces both speech quality and speaker similarity. Many approaches, such as global variance, have been proposed to increase the variability of the spectrum. Approaches like dictionary mapping and unit-selection don't suffer as much since they retain raw parameters and the feature manipulation is minimal; however, they typically require a larger training corpus and might suffer from discontinuous features, resulting in audible discontinuities in the generated speech waveform.

Prosodic mapping issues: For converting prosodic aspects of speech, various methods have been proposed. However, most of them simply adjust basic global statistics, such as average and standard deviation. The conversion is usually performed in the frame-level domain. As mentioned in the previous sections, these naive modifications can not effectively convert supra-segmental features. There are some challenges to modeling prosody for parametric VC. The main challenge is the absence of certain high-level features during conversion, which significantly affect human prosody. These features might be linguistic features (such as information about phonemes and syllables), or more abstract features (such as sarcasm and emotion). For TTS systems, textual information is available during conversion, which facilitates predicting prosodic features from more prosodically relevant representations such as syllable-level or word-level information. Especially foot-level information modeling might be helpful for conversion [148]. These types of data, extracted from the input text, are not available to a stand-alone VC system, but could be extracted using ASR systems with some degree of error. The main challenge is to transform pitch contours by considering more context than one frame at a time, i. e. segmentally.

6.2.3 Future Directions for VC systems

In the previous section, we presented several challenges that current VC technology faces. In this section, we list some future research directions.

Non-Parallel VC: Most studies in the literature use parallel corpora. However, to make VC systems easier to use, building transformation systems from non-parallel corpora is essential.

The reason is that average users are hesitant to record numerous speech prompts with specific contents, which might be laborious for some. Several attempts to create non-parallel VC systems have been reported, including but not restricted to: INCA iterative alignment [58], RBMs [206], and CycleGANs [129, 66].

Text-dependent VC: VC systems that utilize phonetic information are another research area. One example is to use phoneme identity before clustering the acoustic space [146, 331]. Using phonetic information to identify classes using a CART model instead of spectral information has also been proposed [48]. These systems could use the output of ASR to help the effectiveness of VC. These systems would likely use a combination of techniques from ASR, VC and parametric TTS [265, 264, 266].

Database size: An important research direction is capturing the voice using very limited recordings. Some studies propose methods for dealing with limited amounts of data [85, 324, 175, 91, 229, 283, 245, 359, 70]. Utilizing additional unsupervised data has been proposed; for example, techniques that separate phonetic content and speaker identity are an elegant approach [229, 245, 206, 185, 186, 106, 38, 139].

Modeling dynamics: Typically, most VC systems focus on performing transformations frame-by-frame. One approach to this consists of adding dynamic information to the mapping features. Event-based approaches seem to be a good representation since they decompose a sequence into events and transitions, and these can be individually modeled. However, detection of event locations is a challenging task and requires more research. Additionally, some models such as HMMs, RNNs, and Seq2Seq implicitly model the speech dynamics from a sequence of local features. Typically, these models have higher number of parameters compared to frame-by-frame models. These sequence mapping approaches appear to be a major future direction.

Prosody Modeling: Developing more complex prosody models that can capture speakers' intonation and segmental duration in an effective way is an important research direction. Most of the literature performs simple linear transformations of the pitch contour (typically in log domain) [349] and the speaking rate. Developing more sophisticated prosody models would enable the capture of complex prosodic patterns and thus enable more effective transformations. Seq2seq models have shown promise in their ability to simultaneously model acoustic, pitch and duration parameters [128, 285].

Many-to-one and many-to-many conversion: In practice, most VC systems can only convert

speech from the source speaker that they have been trained on. A more practical approach is to have a system that converts speech from anybody to the target speaker. Several attempts to accomplish this have been studied [304, 266, 86, 186, 34, 250, 87]. We propose an approach which enables us to decompose different factors of the speech data and allows modification of and reconstruction from these factors. This is a unique property of our approach, seen only in one other recent work [106].Our proposed approach allows us to potentially consider other factors such as recording environments as well.

Articulatory features: Most of the current literature studies the VC problem from a perceptual standpoint. However, it may be worthwhile to approach the problem from a speech production point of view. Several attempts to model and synthesize articulatory properties of the human vocal tract have been proposed [300, 305]. These approaches have some limitations, such as being speaker-dependent, or requiring hard-to-collect data such as Magnetic Resonance Imaging 3D images, electromagnetic articulography, and X-rays. Overcoming these limitations would open up an important set of tools for articulatory conversion and synthesis.

Perceptual optimization: The optimizations that are performed in statistical methods during learning source-target feature mapping function typically optimize criterions that are not highly correlated with human perception. An attempt at performing perceptual error optimization for DNN-based TTS has been proposed [327]; similar approaches could be adopted to VC.

Real-world situations: Most of the corpora used in the literature are recorded in clean conditions. In real-world situations, speech is often encountered in noisy environments. Attempts to perform VC on these noisy data would result in even more distorted synthesized speech. Creating corpora for these situations and developing noise-robust systems are an essential step to allowing VC systems to become mainstream.

Bibliography

[1] Masanobu Abe, Satoshi Nakamura, Kiyohiro Shikano, and Hisao Kuwabara. Voice conversion through vector quantization. In *Proceedings of the ICASSP*, 1988.

[2] Yannis Agiomyrgiannakis. VOCAINE the vocoder and applications in speech synthesis. In *Proceedings of the ICASSP*, 2015.

[3] Yannis Agiomyrgiannakis and Olivier Rosec. ARX-LF-based source-filter methods for voice modification and transformation. In *Proceedings of the ICASSP*, 2009.

[4] R. Aihara, R. Ueda, T. Takiguchi, and Y. Ariki. Exemplar-based emotional voice conversion using non-negative matrix factorization. In *Proceedings of the APSIPA*, Dec 2014. doi: 10.1109/APSIPA.2014.7041640.

[5] Ryo Aihara, Ryoichi Takashima, Tetsuya Takiguchi, and Yasuo Ariki. Individuality-preserving voice conversion for articulation disorders based on non-negative matrix factorization. In *Proceedings of the ICASSP*, 2013.

[6] Ryo Aihara, Toru Nakashika, Tetsuya Takiguchi, and Yasuo Ariki. Voice conversion based on non-negative matrix factorization using phoneme-categorized dictionary. In *Proceedings of the ICASSP*, 2014.

[7] Ryo Aihara, Testuya Takiguchi, and Yasuo Ariki. Many-to-many voice conversion based on multiple non-negative matrix factorization. In *Proceedings of the INTERSPEECH*, 2015.

[8] Ryo AIHARA, Tetsuya TAKIGUCHI, and Yasuo ARIKI. Activity-mapping non-negative matrix factorization for exemplar-based voice conversion. In *Proceedings of the ICASSP*, 2015.

[9] Federico Alegre, Asmaa Amehraye, and Nicholas Evans. Spoofing countermeasures to protect automatic speaker verification from voice conversion. In *Proceedings of the ICASSP*, 2013.

[10] Gopala Krishna Anumanchipalli, Kishore Prahallad, and Alan W Black. Festvox: Tools for creation and analyses of large speech corpora. In *Workshop on Very Large Scale Phonetics Research, UPenn, Philadelphia*, 2011.

[11] Levent M Arslan. Speaker transformation algorithm using segmental codebooks (STASC). *Speech Communication*, 28(3):211–226, 1999.

[12] Levent M Arslan and David Talkin. Voice conversion by codebook mapping of line spectral frequencies and excitation spectrum. In *Proceedings of the EUROSPEECH*, 1997.

[13] Levent M Arslan and David Talkin. Speaker transformation using sentence HMM based alignments and detailed prosody modification. In *Proceedings of the ICASSP*, 1998.

[14] Sandesh Aryal, Daniel Felps, and Ricardo Gutierrez-Osuna. Foreign accent conversion through voice morphing. In *Proceedings of the INTERSPEECH*, 2013.

[15] Elias Azarov, Maxim Vashkevich, Denis Likhachov, and Alexander Petrovsky. Real-time voice conversion using artificial neural networks with rectified linear units. In *Proceedings of the INTERSPEECH*, 2013.

[16] Fahimeh Bahmaninezhad, Chunlei Zhang, and John Hansen. Convolutional neural network based speaker de-identification. In *Proc. Odyssey*, 2018.

[17] Roberto Barra, Juan Manuel Montero, J Macias-Guarasa, J Gutiérrez-Arriola, Javier Ferreiros, and José Manuel Pardo. On the limitations of voice conversion techniques in emotion identification tasks. In *Proceedings of the INTERSPEECH*, 2007.

[18] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35 (8):1798–1828, 2013.

[19] Hadas Benisty and David Malah. Voice conversion using gmm with enhanced global variance. In *Proceedings of the INTERSPEECH*, 2011.

[20] Hadas Benisty, David Malah, and Koby Crammer. Sequential voice conversion using grid-based approximation. In *Proceedings of the IEEEI*, 2014.

[21] Ning Bi and Yingyong Qi. Application of speech conversion to alaryngeal speech enhancement. *IEEE Transactions on Speech and Audio Processing*, 5(2):97–105, 1997.

[22] Merlijn Blaauw and Jordi Bonada. Modeling and transforming speech using variational autoencoders. In *Proc. Interspeech*, 2016.

[23] Antonio Bonafonte, Harald Höge, Imre Kiss, Asunción Moreno, Ute Ziegenhain, Henk van den Heuvel, Horst-Udo Hain, Xia S Wang, and Marie-Neige Garcia. TC-STAR: Specifications of language resources and evaluation for speech synthesis. In *Proceedings of the LREC*, 2006.

[24] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. In *Advances in Neural Information Processing Systems*, pages 737–744, 1994.

[25] Pedro Cano, Alex Loscos, Jordi Bonada, Maarten De Boer, and Xavier Serra. Voice morphing system for impersonating in karaoke applications. In *Proceedings of the ICMC*, 2000.

[26] Tim Ceyssens, Werner Verhelst, and Patrick Wambacq. On the construction of a pitch conversion system. In *Proceedings of the EUSIPCO*, page 2002.

[27] David T Chappell and John HL Hansen. Speaker-specific pitch contour modeling and modification. In *Proceedings of the ICASSP*, 1998.

[28] Ke Chen and Ahmad Salman. Extracting speaker-specific information with a regularized siamese deep network. In *Advances in Neural Information Processing Systems 24*, pages 298–306, 2011.

[29] Ling-Hui Chen, Zhen-Hua Ling, Yan Song, and Li-Rong Dai. Joint spectral distribution modeling using restricted boltzmann machines for voice conversion. In *Proceedings of the INTERSPEECH*, 2013.

[30] Ling-Hui Chen, Zhen-Hua Ling, and Li-Rong Dai. Voice conversion using generative trained deep neural networks with multiple frame spectral envelopes. In *Proceedings of the INTERSPEECH*, 2014.

[31] Ling-Hui Chen, Zhen-Hua Ling, Li-Juan Liu, and Li-Rong Dai. Voice conversion using deep neural networks with layer-wise generative training. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 22(12):1859–1872, 2014.

[32] Ling-Hui Chen, Li-Juan Liu, Zhen-Hua Ling, Yuan Jiang, and Li-Rong Dai. The USTC system for voice conversion challenge 2016: Neural network based approaches for spectrum, aperiodicity and F0 conversion. In *Proceedings of the INTERSPEECH*, 2016.

[33] Yining Chen, Min Chu, Eric Chang, Jia Liu, and Runsheng Liu. Voice conversion with smoothed GMM and MAP adaptation. In *Proceedings of the EUROSPEECH*, 2003.

[34] Yuya Chiba, Akinori Ito, and Takahiro Shinozaki. Voice conversion from arbitrary speakers based on deep neural networks with adversarial learning. In *Advances in Intelligent Information Hiding and Multimedia Signal Processing: Proceedings of the Thirteenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 2017.

[35] DG Childers, B Yegnanarayana, and Ke Wu. Voice conversion: Factors responsible for quality. In *Proceedings of the ICASSP*, 1985.

[36] Donald G Childers. Glottal source modeling for voice conversion. *Speech communication*, 16 (2):127–138, 1995.

[37] Donald G Childers, Ke Wu, DM Hicks, and B Yegnanarayana. Voice conversion. *Speech Communication*, 8(2):147–158, 1989.

[38] Ju-chieh Chou, Cheng-chieh Yeh, Hung-yi Lee, and Lin-shan Lee. Multi-target voice conversion without parallel data by adversarially learning disentangled audio representations. *Proc. Interspeech*, 2018.

[39] Berrak Çişman, Haizhou Li, and Kay Chen Tan. Sparse representation of phonetic features for voice conversion with and without parallel data. In *Proc. ASRU*, 2017.

[40] Maria Joana Ribeiro Folgado Correia. Anti-spoofing: Speaker verification vs. voice conversion. *Master's Thesis, Instituto Superior Técnico*, 2014.

[41] Najim Dehak, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 2011.

[42] Arantza Del Pozo and Steve Young. The linear transformation of lf glottal waveforms for voice conversion. In *Proceedings of the INTERSPEECH*, 2008.

[43] Srinivas Desai, Alan W Black, B Yegnanarayana, and Kishore Prahallad. Spectral mapping using artificial neural networks for voice conversion. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(5):954–964, 2010.

[44] Hironori Doi, Tomoki Toda, Tomoyasu Nakano, Masataka Goto, and Satoshi Nakamura. Singing voice conversion method based on many-to-many eigenvoice conversion and training

data generation using a singing-to-singing synthesis system. In *Proceedings of the APSIPA*, 2012.

[45] Thierry Dutoit, A Holzapfel, Matthieu Jottrand, Alexis Moinet, Javier Perez, and Y Stylianou. Towards a voice conversion system based on frame selection. In *Proceedings of the ICASSP*, 2007.

[46] Helenca Duxans. *Voice Conversion applied to Text-to-Speech systems*. PhD thesis, Universitat Politecnica de Catalunya, Barcelona, Spain, 2006.

[47] Helenca Duxans and Antonio Bonafonte. Residual conversion versus prediction on voice morphing systems. In *Proceedings of the ICASSP*, 2006.

[48] Helenca Duxans, Antonio Bonafonte, Alexander Kain, and Jan Van Santen. Including dynamic and phonetic information in voice conversion systems. In *Proceedings of the ICSLP*, 2004.

[49] Helenca Duxans, Daniel Erro, Javier Pérez, Ferran Diego, Antonio Bonafonte, and Asunción Moreno. Voice conversion of non-aligned data using unit selection. In *TC-STAR WSST*, 2006.

[50] Ahmed Elgammal and Chan-Su Lee. Separating style and content on a nonlinear manifold. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2004.

[51] Taoufik En-Najjary, Olivier Rosec, and Thierry Chonavel. A new method for pitch prediction from spectral envelope and its application in voice conversion. In *Proceedings of the INTERSPEECH*, 2003.

[52] Taoufik En-Najjary, Olivier Rosec, and Thierry Chonavel. A voice conversion method based on joint pitch and spectral envelope transformation. In *Proceedings of the INTERSPEECH*, 2004.

[53] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660, 2010.

[54] D Erro, A Alonso, L Serrano, D Tavarez, I Odriozola, Xabier Sarasola, Eder Del Blanco, J Sanchez, I Saratxaga, Eva Navas, et al. Ml parameter generation with a reformulated mge

training criterion—participation in the voice conversion challenge 2016. In *Proceedings of the INTERSPEECH*, 2016.

[55] Daniel Erro and Asunción Moreno. Frame alignment method for cross-lingual voice conversion. In *Proceedings of the INTERSPEECH*, 2007.

[56] Daniel Erro and Asunción Moreno. Weighted frequency warping for voice conversion. In *Proceedings of the INTERSPEECH*, 2007.

[57] Daniel Erro, Tatyana Polyakova, and Asunción Moreno. On combining statistical methods and frequency warping for high-quality voice conversion. In *Proceedings of the ICASSP*, 2008.

[58] Daniel Erro, Asunción Moreno, and Antonio Bonafonte. INCA algorithm for training voice conversion systems from nonparallel corpora. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(5):944–953, 2010.

[59] Daniel Erro, Asunción Moreno, and Antonio Bonafonte. Voice conversion based on weighted frequency warping. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(5): 922–931, 2010.

[60] Daniel Erro, Iñaki Sainz, Eva Navas, and Inma Hernáez. Improved HNM-based vocoder for statistical synthesizers. In *Proceedings of the INTERSPEECH*, 2011.

[61] Daniel Erro, Eva Navas, and Inma Hernáez. Iterative MMSE estimation of vocal tract length normalization factors for voice transformation. In *Proceedings of the INTERSPEECH*, 2012.

[62] Daniel Erro, Agustín Alonso, Luis Serrano, Eva Navas, and Inma Hernáez. Towards physically interpretable parametric voice conversion functions. In *Advances in Nonlinear Speech Processing*, pages 75–82. Springer, 2013.

[63] Daniel Erro, Agustin Alonso, Luis Serrano, Eva Navas, and Inma Hernaez. Interpretable parametric voice conversion functions based on gaussian mixture models and constrained transformations. *Computer Speech & Language*, 30(1):3–15, 2015.

[64] Mahdi Eslami, Hamid Sheikhzadeh, and Abolghasem Sayadiyan. Quality improvement of voice conversion systems based on trellis structured vector quantization. In *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

[65] Bo Fan, Siu Wa Lee, Xiaohai Tian, Lei Xie, and Minghui Dong. A waveform representation framework for high-quality statistical parametric speech synthesis. *arXiv preprint arXiv:1510.01443*, 2015.

[66] Fuming Fang, Junichi Yamagishi, Isao Echizen, and Jaime Lorenzo-Trueba. High-quality nonparallel voice conversion based on cycle-consistent adversarial network. *Proc. ICASSP*, 2018.

[67] Kei Fujii, Jun Okawa, and Kaori Suigetsu. Highindividuality voice conversion based on concatenative speech synthesis. *World Academy of Science, Engineering and Technology*, 2: 1, 2007.

[68] Sadaoki Furui. Speaker-independent isolated word recognition using dynamic features of speech spectrum. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 34(1): 52–59, 1986.

[69] John S Garofolo, Lori F Lamel, William M Fisher, Jonathon G Fiscus, and David S Pallett. DARPA TIMIT acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1. *NASA STI/Recon Technical Report N*, 93:27403, 1993.

[70] Mostafa Ghorbandoost, Abolghasem Sayadiyan, Mohsen Ahangar, Hamid Sheikhzadeh, Abdoreza Sabzi Shahrebabaki, and Jamal Amini. Voice conversion based on feature combination with limited training data. *Speech Communication*, 67:113–128, 2015.

[71] Ben Gillett and Simon King. Transforming f0 contours. In *Proceedings of the EUROSPEECH*, 2003.

[72] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256, 2010.

[73] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Aistats*, 2011.

[74] Elizabeth Godoy, Olivier Rosec, and Thierry Chonavel. Alleviating the one-to-many mapping problem in voice conversion with context-dependent modelling. In *Proceedings of the INTERSPEECH*, 2009.

[75] Elizabeth Godoy, Olivier Rosec, and Thierry Chonavel. Speech spectral envelope estimation through explicit control of peak evolution in time. In *Proceedings of the ISSPA*, 2010.

[76] Elizabeth Godoy, Olivier Rosec, and Thierry Chonavel. On transforming spectral peaks in voice conversion. In *Proceedings of the SSW*, 2010.

[77] Elizabeth Godoy, Olivier Rosec, and Thierry Chonavel. Spectral envelope transformation using DFW and amplitude scaling for voice conversion with parallel or nonparallel corpora. In *Proceeding of the INTERSPEECH*, 2011.

[78] Elizabeth Godoy, Olivier Rosec, and Thierry Chonavel. Voice conversion using dynamic frequency warping with amplitude scaling, for parallel or nonparallel corpora. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(4):1313–1323, 2012.

[79] Elizabeth Godoy, Maria Koutsogiannaki, and Yannis Stylianou. Assessing the intelligibility impact of vowel space expansion via clear speech-inspired frequency warping. In *Proceedings of the INTERSPEECH*, 2013.

[80] Elizabeth Godoy, Maria Koutsogiannaki, and Yannis Stylianou. Approaching speech intelligibility enhancement with inspiration from lombard and clear speaking styles. *Computer Speech & Language*, 28(2):629–647, 2014.

[81] Hung-Yan Gu and Sung-Fung Tsai. Improving segmental gmm based voice conversion method with target frame selection. In *Proceedings of the ISCSLP*, 2014.

[82] S. Hamidreza Mohammadi. Reducing one-to-many problem in Voice Conversion by equalizing the formant locations using dynamic frequency warping. *ArXiv e-prints*, October 2015.

[83] Zdeněk Hanzlíček and Jindrich Matoušek. F0 transformation within the voice conversion framework. In *Proceedings of the INTERSPEECH*, 2007.

[84] Makoto Hashimoto and Norio Higuchi. Spectral mapping method for voice conversion using speaker selection and vector field smoothing. In *Proceedings of the EUROSPEECH*, 1995.

[85] Makoto Hashimoto and Norio Higuchi. Training data selection for voice conversion using speaker selection and vector field smoothing. In *Proceedings of the ICSLP*, 1996.

[86] Tetsuya Hashimoto, Hidetsugu Uchida, Daisuke Saito, and Nobuaki Minematsu. Parallel-data-free many-to-many voice conversion based on dnn integrated with eigenspace using a non-parallel speech corpus. *Proc. Interspeech*, 2017.

[87] Tetsuya Hashimoto, Daisuke Saito, and Nobuaki Minematsu. Many-to-many and completely parallel-data-free voice conversion based on eigenspace dnn. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2019.

[88] Jon Magnus Momrak Haug. Voice conversion using deep learning. Master's thesis, NTNU, 2017.

[89] Tomoki Hayashi, Akira Tamamori, Kazuhiro Kobayashi, Kazuya Takeda, and Tomoki Toda. An investigation of multi-speaker training for wavenet vocoder. In *Proc. ASRU*, 2017.

[90] Elina Helander, Jani Nurminen, and Moncef Gabbouj. Analysis of lsf frame selection in voice conversion. In *Proceedings of the SPECOM*, 2007.

[91] Elina Helander, Jani Nurminen, and Moncef Gabbouj. Lsf mapping for voice conversion with very small training sets. In *Proceedings of the ICASSP*, 2008.

[92] Elina Helander, Jan Schwarz, Jani Nurminen, Hanna Silen, and Moncef Gabbouj. On the impact of alignment on voice conversion performance. In *Proceedings of the INTERSPEECH*, 2008.

[93] Elina Helander, Hanna Silén, Joaquin Míguez, and Moncef Gabbouj. Maximum a posteriori voice conversion using sequential monte carlo methods. In *Proceedings of the INTERSPEECH*, 2010.

[94] Elina Helander, Tuomas Virtanen, Jani Nurminen, and Moncef Gabbouj. Voice conversion using partial least squares regression. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(5):912–921, 2010.

[95] Elina Helander, Hanna Silén, Tuomas Virtanen, and Moncef Gabbouj. Voice conversion using dynamic kernel partial least squares regression. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(3):806–817, 2012.

[96] Elina E Helander and Jani Nurminen. On the importance of pure prosody in the perception of speaker identity. In *Proceedings of the INTERSPEECH*, 2007.

[97] Elina E Helander and Jani Nurminen. A novel method for prosody prediction in voice conversion. In *Proceedings of the ICASSP*, 2007.

[98] DOI Hironori, Keigo Nakamura, TODA Tomoki, Hiroshi Saruwatari, and Kiyohiro Shikano. Esophageal speech enhancement based on statistical voice conversion with gaussian mixture models. *IEICE TRANSACTIONS on Information and Systems*, 93(9):2472–2482, 2010.

[99] Tuan Vu Ho and Masato Akagi. Nonparallel dictionary-based voice conversion using variational autoencoder with modulation-spectrum-constrained training. *Journal of Signal Processing*, 2018.

[100] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[101] Chi-Chun Hsia, Chung-Hsien Wu, and Te-Hsien Liu. Duration-embedded bi-HMM for expressive voice conversion. In *Proceedings of the INTERSPEECH*, 2005.

[102] Chi-Chun Hsia, Chung-Hsien Wu, and Jian-Qi Wu. Conversion function clustering and selection using linguistic and spectral information for emotional voice conversion. *IEEE Transactions on Computers*, 56(9):1245–1254, 2007.

[103] Chin-Cheng Hsu, Hsin-Te Hwang, Yi-Chiao Wu, Yu Tsao, and Hsin-Min Wang. Voice conversion from non-parallel corpora using variational auto-encoder. In *Proc. APSIPA*, 2016.

[104] Chin-Cheng Hsu, Hsin-Te Hwang, Yi-Chiao Wu, Yu Tsao, and Hsin-Min Wang. Voice conversion from unaligned corpora using variational autoencoding wasserstein generative adversarial networks. *arXiv preprint arXiv:1704.00849*, 2017.

[105] Wei-Ning Hsu, Yu Zhang, and James Glass. Learning latent representations for speech generation and transformation. *Proc. Interspeech*, 2017.

[106] Wei-Ning Hsu, Yu Zhang, and James Glass. Unsupervised learning of disentangled and interpretable representations from sequential data. In *Advances in neural information processing systems*, 2017.

[107] Dong-Yan Huang, Lei Xie, Yvonne Siu, Wa Lee, Jie Wu, Huaiping Ming, Xiaohai Tian, Shaofei Zhang, Chuang Ding, Mei Li, Quy Hy Nguyen, Minghui Dong, and Haizhou Li. An automatic voice conversion evaluation strategy based on perceptual background noise distortion and speaker similarity. In *Proceedings of the SSW*, 2016.

[108] Hsin-Te Hwang, Yu Tsao, Hsin-Min Wang, Yih-Ru Wang, Sin-Horng Chen, et al. A study of mutual information for GMM-based spectral conversion. In *Proceedings of the INTERSPEECH*, 2012.

[109] Hsin-Te Hwang, Yu Tsao, Hsin-Min Wang, Yih-Ru Wang, and Sin-Horng Chen. Incorporating global variance in the training phase of gmm-based voice conversion. In *Proceedings of the APSIPA*, 2013.

[110] S Imai, T Kobayashi, K Tokuda, T Masuko, K Koishida, S Sako, and H Zen. Speech signal processing toolkit (SPTK), version 3.3, 2009.

[111] Satoshi Imai. Cepstral analysis synthesis on the mel frequency scale. In *Proceedings of the ICASSP*, 1983.

[112] Satoshi Imai, Kazuo Sumita, and Chieko Furuichi. Mel log spectrum approximation (MLSA) filter for speech synthesis. *Electronics and Communications in Japan (Part I: Communications)*, 66(2):10–18, 1983.

[113] Zeynep Inanoglu. Transforming pitch in a voice conversion framework. *Master's Thesis, St. Edmunds College, University of Cambridge*, 2003.

[114] Zeynep Inanoglu and Steve Young. A system for transforming the emotion in speech: combining data-driven conversion techniques for prosody and voice quality. In *Proceedings of the INTERSPEECH*, pages 490–493, 2007.

[115] Zeynep Inanoglu and Steve Young. Data-driven emotion conversion in spoken english. *Speech Communication*, 51(3):268–283, 2009.

[116] Naoto Iwahashi and Yoshinori Sagisaka. Speech spectrum transformation by speaker interpolation. In *Proceedings of the ICASSP*, volume 1, pages I–461. IEEE, 1994.

[117] Naoto Iwahashi and Yoshinori Sagisaka. Speech spectrum conversion based on speaker interpolation and multi-functional representation with weighting by radial basis function networks. *Speech Communication*, 16(2):139–151, 1995.

[118] Qin Jin, Arthur R Toth, Tanja Schultz, and Alan W Black. Speaker de-identification via voice transformation. In *Proc. ASRU*, 2009.

[119] A. Kain. *High Resolution Voice Transformation*. PhD thesis, OGI School of Science & Engineering at Oregon Health & Science University, 2001.

[120] Alexander Kain. *High resolution voice transformation*. PhD thesis, Oregon Health & Science University, 2001.

[121] Alexander Kain and Michael W Macon. Spectral voice conversion for text-to-speech synthesis. In *Proceedings of the ICASSP*, 1998.

[122] Alexander Kain and Michael W Macon. Text-to-speech voice adaptation from sparse training data. In *Proceedings of the ICSLP*, 1998.

[123] Alexander Kain and Michael W Macon. Design and evaluation of a voice conversion algorithm based on spectral envelope mapping and residual prediction. In *Proceedings of the ICASSP*, 2001.

[124] Alexander Kain and Jan PH van Santen. Frequency-domain delexicalization using surrogate vowels. In *Proc. Interspeech*, 2010.

[125] Alexander Kain and Jan Van Santen. Using speech transformation to increase speech intelligibility for the hearing-and speaking-impaired. In *Proceedings of the ICASSP*, 2009.

[126] Alexander Kain and Jan PH van Santen. Unit-selection text-to-speech synthesis using an asynchronous interpolation model. In *Proceedings of the SSW*, 2007.

[127] Alexander B Kain, John-Paul Hosom, Xiaochuan Niu, Jan PH van Santen, Melanie Fried-Oken, and Janice Staehely. Improving the intelligibility of dysarthric speech. *Speech communication*, 49(9):743–759, 2007.

[128] Hirokazu Kameoka, Kou Tanaka, Takuhiro Kaneko, and Nobukatsu Hojo. Convs2s-vc: Fully convolutional sequence-to-sequence voice conversion. *arXiv preprint arXiv:1811.01609*, 2018.

[129] Takuhiro Kaneko and Hirokazu Kameoka. Parallel-data-free voice conversion using cycle-consistent adversarial networks. *Proc. EUSIPCO*, 2017.

[130] Takuhiro Kaneko, Hirokazu Kameoka, Kaoru Hiramatsu, and Kunio Kashino. Sequence-to-sequence voice conversion with similarity metric learned using generative adversarial networks. In *Proc. Interspeech*, pages 1283–1287, 2017.

[131] Takuhiro Kaneko, Shinji Takaki, Hirokazu Kameoka, and Junichi Yamagishi. Generative adversarial network-based postfilter for stft spectrograms. In *Proceedings of Interspeech*, 2017.

[132] Yongguo Kang, Zhiwei Shuang, Jianhua Tao, Wei Zhang, and Bo Xu. A hybrid gmm and codebook mapping method for spectral conversion. In *Affective Computing and Intelligent Interaction*, pages 303–310. Springer, 2005.

[133] Yongguo Kang, Jianhua Tao, and Bo Xu. Applying pitch target model to convert f0 contour for expressive mandarin speech synthesis. In *Proceedings of the ICASSP*, 2006.

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

[135] Hideki Kawahara, Masanori Morise, Toru Takahashi, Ryuichi Nisimura, Toshio Irino, and Hideki Banno. TANDEM-STRAIGHT: A temporally stable power spectral representation for periodic signals and applications to interference-free spectrum, f0, and aperiodicity estimation. In *Proceedings of the ICASSP*, 2008.

[136] Kazuya Kawakami. *Supervised Sequence Labelling with Recurrent Neural Networks*. PhD thesis, PhD thesis. Ph. D. thesis, Technical University of Munich, 2008.

[137] Hiromichi Kawanami, Yohei Iwami, Tomoki Toda, Hiroshi Saruwatari, and Kiyohiro Shikano. GMM-based voice conversion applied to emotional speech synthesis. In *Proceedings of the EUROSPEECH*, 2003.

[138] Eun-Kyoung Kim, Sangho Lee, and Yung-Hwan Oh. Hidden markov model based voice conversion using dynamic characteristics of speaker. In *Proceedings of the EUROSPEECH*, 1997.

[139] Tomi Kinnunen, Lauri Juvela, Paavo Alku, and Junichi Yamagishi. Non-parallel voice conversion using i-vector plda: Towards unifying speaker verification and transformation. In *Proceedings of the ICASSP*, 2017.

[140] Kazuhiro Kobayashi and Tomoki Toda. sprocket: Open-source voice conversion software. In *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, pages 203–210, 2018.

[141] Kazuhiro Kobayashi, Hironori Doi, Tomoki Toda, Tomoyasu Nakano, Masataka Goto, Graham Neubig, Sakriani Sakti, and Satoshi Nakamura. An investigation of acoustic features for singing voice conversion based on perceptual age. In *Proceedings of the INTERSPEECH*, 2013.

[142] Kazuhiro Kobayashi, Tomoki Toda, Graham Neubig, Sakriani Sakti, and Satoshi Nakamura. Statistical singing voice conversion based on direct waveform modification with global variance. In *Proceedings of the INTERSPEECH*, 2015.

[143] Kazuhiro Kobayashi, Tomoki Hayashi, Akira Tamamori, and Tomoki Toda. Statistical voice conversion with wavenet-based waveform generation. In *Proc. Interspeech*, volume 2017, pages 1138–1142, 2017.

[144] John Kominek and Alan W Black. The CMU arctic speech databases. In *Proceedings of the SSW*, 2004.

[145] Maria Koutsogiannaki and Yannis Stylianou. Simple and artefact-free spectral modifications for enhancing the intelligibility of casual speech. In *Proceedings of the ICASSP*, 2014.

[146] Arun Kumar and Ashish Verma. Using phone and diphone based acoustic models for voice conversion: a step towards creating voice fonts. In *Proceedings of the ICME*, 2003.

[147] Hisao Kuwabara and Yoshinori Sagisak. Acoustic characteristics of speaker individuality: Control and conversion. *Speech communication*, 16(2):165–173, 1995.

[148] Mahsa Sadat Elyasi Langarani and Jan van Santen. Speaker intonation adaptation for transforming text-to-speech synthesis speaker identity. In *Proceedings of the ASRU*, 2015.

[149] Mahsa Sadat Elyasi Langarani, Jan van Santen, Seyed Hamidreza Mohammadi, and Alexander Kain. Data-driven foot-based intonation generator for text-to-speech synthesis. In *Proc. of INTERSPEECH*, 2015.

[150] Rabul Hussain Laskar, Fazal Ahmed Talukdar, Rajib Bhattacharjee, and Saugat Das. Voice conversion by mapping the spectral and prosodic features using support vector machine. In *Applications of Soft Computing*, pages 519–528. Springer, 2009.

[151] RH Laskar, D Chakrabarty, FA Talukdar, K Sreenivasa Rao, and K Banerjee. Comparing ANN and GMM in a voice conversion framework. *Applied Soft Computing*, 12(11):3332–3342, 2012.

[152] Javier Latorre, Vincent Wan, and Kayoko Yanagisawa. Voice expression conversion with factorised HMM-TTS models. In *Proceedings of the INTERSPEECH*, 2014.

[153] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[154] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Artificial Intelligence and Statistics*, 2015.

[155] Chung-Han Lee and Chung-Hsien Wu. Map-based adaptation for speech conversion using adaptation data selection and non-parallel training. In *Proceedings of the INTERSPEECH*, 2006.

[156] Ki-Seung Lee. Statistical approach for voice personality transformation. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(2):641–651, 2007.

[157] Ki-Seung Lee. A unit selection approach for voice transformation. *Speech Communication*, 60:30–43, 2014.

[158] Ki-Seung Lee. Restricted boltzmann machine-based voice conversion for nonparallel corpus. *IEEE signal processing letters*, 2017.

[159] Bingjie Li, Zhongzhe Xiao, Yan Shen, Qiang Zhou, and Zhi Tao. Emotional speech conversion based on spectrum-prosody dual transformation. In *Proceedings of the ICSP*, 2012.

[160] Runnan Li, Zhiyong Wu, Yishuang Ning, Lifa Sun, Helen Meng, and Lianhong Cai. Spectro-temporal modelling with time-frequency lstm and structured output layer for voice conversion. *Proc. Interspeech*, 2017.

[161] Zhen-Hua Ling, Shi-Yin Kang, Heiga Zen, Andrew Senior, Mike Schuster, Xiao-Jun Qian, Helen M Meng, and Li Deng. Deep learning for acoustic modeling in parametric speech generation: A systematic review of existing techniques and future trends. *Signal Processing Magazine, IEEE*, 32(3):35–52, 2015.

[162] Li-Juan Liu, Ling-Hui Chen, Zhen-Hua Ling, and Li-Rong Dai. Using bidirectional associative memories for joint spectral envelope modeling in voice conversion. In *Proceedings of the ICASSP*, 2014.

[163] Li-Juan Liu, Ling-Hui Chen, Zhen-Hua Ling, and Li-Rong Dai. Spectral conversion using deep neural networks trained with multi-source speakers. In *Proceedings of the ICASSP*, 2015.

[164] Damien Lolive, Nelly Barbot, and Olivier Boeffard. Pitch and duration transformation with non-parallel data. In *Proceedings of the Speech Prosody*, 2008.

[165] Jaime Lorenzo-Trueba, Junichi Yamagishi, Tomoki Toda, Daisuke Saito, Fernando Villavicencio, Tomi Kinnunen, and Zhenhua Ling. The voice conversion challenge 2018: Promoting development of parallel and nonparallel methods. *arXiv preprint arXiv:1804.04262*, 2018.

[166] Zhaojie Luo, Jinhui Chen, Tetsuya Takiguchi, and Yasuo Ariki. Emotional voice conversion with adaptive scales f0 based on wavelet transform using limited amount of emotional data. *Proc. Interspeech*, 2017.

[167] Anderson F Machado and Marcelo Queiroz. Voice conversion: A critical survey. *Proceedings of the SMC*, 2010.

[168] Noriyasu Maeda, Hideki Banno, Shoji Kajita, Kazuya Takeda, and Fumitada Itakura. Speaker conversion through non-linear frequency warping of straight spectrum. In *Proceedings of the EUROSPEECH*, 1999.

[169] Behrooz Makki, SA Seyedsalehi, Nasser Sadati, and M Noori Hosseini. Voice conversion using nonlinear principal component analysis. In *Proceedings of the CIISP*, 2007.

[170] Kenta Masaka, Ryo Aihara, Tetsuya Takiguchi, and Yasuo Ariki. Multimodal voice conversion using non-negative matrix factorization in noisy environments. In *Proceedings of the ICASSP*, 2014.

[171] Tsuyoshi Masuda and Makoto Shozakai. Cost reduction of training mapping function based on multistep voice conversion. In *Proceedings of the ICASSP*, 2007.

[172] Hiroshi Matsumoto and Yasuki Yamashita. Unsupervised speaker adaptation from short utterances based on a minimized fuzzy objective function. *Journal of the Acoustical Society of Japan (E)*, 14(5):353–361, 1993.

[173] Hiroshi Matsumoto, Shizuo Hiki, Toshio Sone, and Tadamoto Nimura. Multidimensional representation of personal quality of vowels and its acoustical correlates. *IEEE Transactions on Audio and Electroacoustics*, 21(5):428–436, 1973.

[174] Larbi Mesbahi, Vincent Barreaud, and Olivier Boeffard. Comparing GMM-based speech transformation systems. In *Proceedings of the INTERSPEECH*, 2007.

[175] Larbi Mesbahi, Vincent Barreaud, and Olivier Boeffard. Gmm-based speech transformation systems under data reduction. In *Proceedings of the SSW*, 2007.

[176] Huaiping Ming, Dongyan Huang, Lei Xie, Jie Wu, and Minghui Dong1and Haizhou Li. Deep bidirectional lstm modeling of timbre and prosody for emotional voice conversion. In *Proceedings of the INTERSPEECH*, 2016.

[177] Hiroyuki Miyoshi, Yuki Saito, Shinnosuke Takamichi, and Hiroshi Saruwatari. Voice conversion using sequence-to-sequence learning of context posterior probabilities. *arXiv preprint arXiv:1704.02360*, 2017.

[178] Hideyuki Mizuno and Masanobu Abe. Voice conversion algorithm based on piecewise linear conversion rules of formant frequency and spectrum tilt. *Speech Communication*, 16(2): 153–164, 1995.

[179] Seyed Hamidreza Mohammadi. Reducing one-to-many problem in voice conversion by equalizing the formant locations using dynamic frequency warping. *arXiv preprint arXiv:1510.04205*, 2015.

[180] Seyed Hamidreza Mohammadi and Alexander Kain. Transmutative voice conversion. In *Proceedings of the ICASSP*, 2013.

[181] Seyed Hamidreza Mohammadi and Alexander Kain. Voice conversion using deep neural networks with speaker-independent pre-training. In *Proceedings of the SLT*, 2014.

[182] Seyed Hamidreza Mohammadi and Alexander Kain. Semi-supervised training of a voice conversion mapping function using a joint-autoencoder. In *Proceedings of the INTERSPEECH*, 2015.

[183] Seyed Hamidreza Mohammadi and Alexander Kain. A voice conversion mapping function based on a stacked joint-autoencoder. In *Proceedings of the INTERSPEECH*, 2016.

[184] Seyed Hamidreza Mohammadi and Alexander Kain. An overview of voice conversion systems. *Speech Communication*, 2017.

[185] Seyed Hamidreza Mohammadi and Alexander Kain. Siamese autoencoders for speech style extraction and switching applied to voice identification and conversion. *Proc. Interspeech*, 2017.

[186] Seyed Hamidreza Mohammadi and Taehwan Kim. Investigation of using disentangled and interpretable representations for one-shot cross-lingual voice conversion. *Proc. Interspeech*, 2018.

[187] Seyed Hamidreza Mohammadi, Alexander Kain, and Jan PH van Santen. Making conversational vowels more clear. In *Proceedings of the INTERSPEECH*, 2012.

[188] Masanori Morise. Cheaptrick, a spectral envelope estimator for high-quality speech synthesis. *Speech Communication*, 67:1–7, 2015.

[189] Masanori Morise, Fumiya Yokomori, and Kenji Ozawa. World: a vocoder-based high-quality speech synthesis system for real-time applications. *IEICE Transactions on Information and Systems*, 2016.

[190] Eric Morley, Esther Klabbers, Jan PH van Santen, Alexander Kain, and Seyed Hamidreza Mohammadi. Synthetic f0 can effectively convey speaker id in delexicalized speech. In *Proc. of INTERSPEECH*, 2012.

[191] Robert W Morris and Mark A Clements. Reconstruction of speech from whispers. *Medical Engineering & Physics*, 24(7):515–520, 2002.

[192] Athanasios Mouchtaris, Jan Van der Spiegel, and Paul Mueller. Non-parallel training for voice conversion by maximum likelihood constrained adaptation. In *Proceedings of the ICASSP*, 2004.

[193] Athanasios Mouchtaris, Jan Van der Spiegel, and Paul Mueller. A spectral conversion approach to the iterative wiener filter for speech enhancement. In *Proceedings of the ICME*, 2004.

[194] Athanasios Mouchtaris, Jan Van der Spiegel, and Paul Mueller. Nonparallel training for voice conversion based on a parameter adaptation approach. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(3):952–963, 2006.

[195] Athanasios Mouchtaris, Yannis Agiomyrgiannakis, and Yannis Stylianou. Conditional vector quantization for voice conversion. In *Proceedings of the ICASSP*, 2007.

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

[197] Takashi Muramatsu, Yamato Ohtani, Tomoki Toda, Hiroshi Saruwatari, and Kiyohiro Shikano. Low-delay voice conversion based on maximum likelihood estimation of spectral parameter trajectory. In *Proceedings of the INTERSPEECH*, 2008.

[198] Mikihiro Nakagiri, Tomoki Toda, Hideki Kashioka, and Kiyohiro Shikano. Improving body transmitted unvoiced speech with statistical voice conversion. In *Proceedings of the INTERSPEECH*, 2006.

[199] Keigo Nakamura, Tomoki Toda, Hiroshi Saruwatari, and Kiyohiro Shikano. A speech communication aid system for total laryngectomies using voice conversion of body transmitted artificial speech. *The Journal of the Acoustical Society of America*, 120(5):3351–3351, 2006.

[200] Keigo Nakamura, Tomoki Toda, Hiroshi Saruwatari, and Kiyohiro Shikano. Speaking-aid systems using gmm-based voice conversion for electrolaryngeal speech. *Speech Communication*, 54(1):134–146, 2012.

[201] T. Nakashika, T. Takiguchi, and Y. Ariki. Voice conversion using rnn pre-trained by recurrent temporal restricted boltzmann machines. *IEEE/ACM Transactions on Audio,*

Speech, and Language Processing, 23(3):580–587, March 2015. ISSN 2329-9290. doi: 10.1109/TASLP.2014.2379589.

[202] Toru Nakashika, Ryoichi Takashima, Tetsuya Takiguchi, and Yasuo Ariki. Voice conversion in high-order eigen space using deep belief nets. In *Proceedings of the INTERSPEECH*, 2013.

[203] Toru Nakashika, Tetsuya Takiguchi, and Yasuo Ariki. High-order sequence modeling using speaker-dependent recurrent temporal restricted boltzmann machines for voice conversion. In *Proceedings of the INTERSPEECH*, 2014.

[204] Toru Nakashika, Tetsuya Takiguchi, and Yasuo Ariki. Sparse nonlinear representation for voice conversion. In *Proceedings of the ICME*, 2015.

[205] Toru Nakashika, Tetsuya Takiguchi, and Yasuo Ariki. Voice conversion using speaker-dependent conditional restricted boltzmann machine. *EURASIP Journal on Audio, Speech, and Music Processing*, 2015(1):1–12, 2015.

[206] Toru Nakashika, Tetsuya Takiguchi, and Yasuhiro Minami. Non-parallel training in voice conversion using an adaptive restricted boltzmann machine. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(11):2032–2045, 2016.

[207] Toru Nakashika, Toru, Tetsuya Takiguchi, Tetsuya, and Yasuo Ariki, Yasuo. Voice conversion based on speaker-dependent restricted boltzmann machines. *IEICE Transactions on Information and Systems*, 97(6):1403–1410, 2014.

[208] Yoshihiko Nankaku, Kenichi Nakamura, Tomoki Toda, and Keiichi Tokuda. Spectral conversion based on statistical models including time-sequence matching. In *Proceedings of the SSW*, 2007.

[209] M Narendranath, Hema A Murthy, S Rajendran, and B Yegnanarayana. Transformation of formants for voice conversion using artificial neural networks. *Speech communication*, 16(2): 207–216, 1995.

[210] Binh Phu Nguyen. *Studies on spectral modification in voice transformation.* PhD thesis, Japan Advanced Institute of Science and Technology, 2009.

[211] Binh Phu Nguyen and Masato Akagi. Spectral modification for voice gender conversion using temporal decomposition. *Journal of Signal Processing*, 2007.

[212] Binh Phu Nguyen and Masato Akagi. Phoneme-based spectral voice conversion using temporal decomposition and gaussian mixture model. In *Proceedings of the ICCE*, 2008.

[213] Jagannath Nirmal, Mukesh Zaveri, Suprava Patnaik, and Pramod Kachare. Voice conversion using general regression neural network. *Applied Soft Computing*, 24:1–12, 2014.

[214] JH Nirmal, Suparva Patnaik, and Mukesh A Zaveri. Voice transformation using radial basis function. In *Proceedings of the TITC*, pages 345–351. Springer, 2013.

[215] Jani Nurminen, Victor Popa, Jilei Tian, Yuezhong Tang, and Imre Kiss. A parametric approach for voice conversion. In *TCSTAR WSST*, pages 225–229, 2006.

[216] Jani Nurminen, Jilei Tian, and Victor Popa. Voicing level control with application in voice conversion. In *Proceedings of the INTERSPEECH*, 2007.

[217] Yamato Ohtani. *Techniques for improving voice conversion based on eigenvoices*. PhD thesis, Nara Institute of Science and Technology, 2010.

[218] Yamato Ohtani, Tomoki Toda, Hiroshi Saruwatari, and Kiyohiro Shikano. Maximum likelihood voice conversion based on GMM with STRAIGHT mixed excitation. In *Proceedings of the INTERSPEECH*, 2006.

[219] Yamato Ohtani, Tomoki Toda, Hiroshi Saruwatari, and Kiyohiro Shikano. Many-to-many eigenvoice conversion with reference voice. In *Proceedings of the INTERSPEECH*, 2009.

[220] Yamato Ohtani, Tomoki Toda, Hiroshi Saruwatari, and Kiyohiro Shikano. Non-parallel training for many-to-many eigenvoice conversion. In *Proceedings of the ICASSP*, 2010.

[221] Kuldip K Paliwal. Interpolation properties of linear prediction parametric representations. In *Proceedings of the EUROSPEECH*, 1995.

[222] Kun-Youl Park and Hyung Soon Kim. Narrowband to wideband conversion of speech using gmm based transformation. In *Proceedings of the ICASSP*, 2000.

[223] David John Patterson. *linguistic approach to pitch range modelling*. PhD thesis, Edinburgh University, 2000.

[224] Bryan L Pellom and John HL Hansen. An experimental study of speaker verification sensitivity to computer voice-altered imposters. In *Proceedings of the ICASSP*, 1999.

[225] Winston S Percybrooks and Elliot Moore. Voice conversion with linear prediction residual estimaton. In *Proceedings of the ICASSP*, 2008.

[226] Nicholas CV Pilkington, Heiga Zen, Mark JF Gales, et al. Gaussian process experts for voice conversion. In *Proceedings of the INTERSPEECH*, 2011.

[227] Michael Pitz and Hermann Ney. Vocal tract normalization equals linear transformation in cepstral space. *Speech and Audio Processing, IEEE Transactions on*, 13(5):930–944, 2005.

[228] Teeraphon Pongkittiphan. Eigenvoice-based character conversion and its evaluations. Master's thesis, The University of Tokyo, 2012.

[229] Victor Popa, Jani Nurminen, and Moncef Gabbouj. A novel technique for voice conversion based on style and content decomposition with bilinear models. In *Proceedings of the INTERSPEECH*, 2009.

[230] Victor Popa, Jani Nurminen, Moncef Gabbouj, et al. A study of bilinear models in voice conversion. *Journal of Signal and Information Processing*, 2(02):125, 2011.

[231] Victor Popa, Hanna Silen, Jani Nurminen, and Moncef Gabbouj. Local linear transformation for voice conversion. In *Proceedings of the ICASSP*, 2012.

[232] A Pozo. *Voice source and duration modelling for voice conversion and speech repair*. PhD thesis, University of Cambridge, 2008.

[233] Anna Přibilová and Jiří Přibil. Non-linear frequency scale mapping for voice conversion in text-to-speech system with cepstral description. *Speech Communication*, 48(12):1691–1703, 2006.

[234] Yu Qiao, Tong Tong, and Nobuaki Minematsu. A study on bag of gaussian model with application to voice conversion. In *Proceedings of the INTERSPEECH*, pages 657–660, 2011.

[235] Lawrence R Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition*, volume 14. PTR Prentice Hall Englewood Cliffs, 1993.

[236] Miguel Varela Ramos. Voice conversion with deep learning. Master's thesis, Tecnico Lisboa, 2016.

[237] Miguel Varela Ramos, Alan W Black, Ramon Fernandez Astudillo, Isabel Trancoso, and Nuno Fonseca. Segment level voice conversion with recurrent neural networks. *Proc. Interspeech*, 2017.

[238] K Sreenivasa Rao, RH Laskar, and Shashidhar G Koolagudi. Voice transformation by mapping the features at syllable level. In *Pattern Recognition and Machine Intelligence*, pages 479–486. Springer, 2007.

[239] Sushant V Rao, Nirmesh J Shah, and Hemant A Patil. Novel pre-processing using outlier removal in voice conversion. In *Proceedings of the SSW*, 2016.

[240] Dimitrios Rentzos, Saeed Vaseghi Qin, Ching-Hsiang Ho, and Emir Turajlic. Probability models of formant parameters for voice conversion. In *Proceedings of the EUROSPEECH*, 2003.

[241] Douglas A Reynolds and Richard C Rose. Robust text-independent speaker identification using gaussian mixture speaker models. *Speech and Audio Processing, IEEE Transactions on*, 3(1):72–83, 1995.

[242] Ansgar Rinscheid. Voice conversion based on topological feature maps and time-variant filtering. In *Proceedings of the ICSLP*, 1996.

[243] Antony W Rix, John G Beerends, Michael P Hollier, and Andries P Hekstra. Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs. In *Proceedings of the ICASSP*, 2001.

[244] Daisuke Saito, Keisuke Yamamoto, Nobuaki Minematsu, and Keikichi Hirose. One-to-many voice conversion based on tensor representation of speaker space. In *Proceedings of the INTERSPEECH*, 2011.

[245] Daisuke Saito, Shinji Watanabe, Atsushi Nakamura, and Nobuaki Minematsu. Statistical voice conversion based on noisy channel model. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(6):1784–1794, 2012.

[246] Yuki Saito, Shinnosuke Takamichi, and Hiroshi Saruwatari. Voice conversion using input-to-output highway networks. *IEICE Transactions on Information and Systems*, 2017.

[247] Yuki Saito, Yusuke Ijima, Kyosuke Nishida, and Shinnosuke Takamichi. Non-parallel voice conversion using variational autoencoders conditioned by phonetic posteriorgrams and d-vectors. In *Proc. ICASSP*, 2018.

[248] Özgül Salor and Mübeccel Demirekler. Dynamic programming approach to voice transformation. *Speech communication*, 48(10):1262–1272, 2006.

[249] Gerard Sanchez, Hanna Silen, Jani Nurminen, and Moncef Gabbouj. Hierarchical modeling of f0 contours for voice conversion. In *Proceedings of the INTERSPEECH*, 2014.

[250] Yusuke Sekii, Ryohei Orihara, Keisuke Kojima, Yuichi Sei, Yasuyuki Tahara, and Akihiko Ohsuga. Fast many-to-one voice conversion using autoencoders. In *ICAART*, 2017.

[251] K. Shikano, S. Nakamura, and M. Abe. Speaker adaptation and voice conversion by codebook mapping. In *IEEE International Sympoisum on Circuits and Systems*, pages 594–597, 1991.

[252] Zhi-Wei Shuang, Zi-Xiang Wang, Zhen-Hua Ling, and Ren-Hua Wang. A novel voice conversion system based on codebook mapping with phoneme-tied weighting. In *Proceedings of the ICSLP*, 2004.

[253] Zhiwei Shuang, Raimo Bakis, and Yong Qin. Voice conversion based on mapping formants. In *TC-STAR WSST*, pages 219–223, 2006.

[254] Zhiwei Shuang, Fanping Meng, and Yong Qin. Voice conversion by combining frequency warping with unit selection. In *Proceedings of the ICASSP*, 2008.

[255] Berrak Sisman, Grandee Lee, Haizhou Li, and Kay Chen Tan. On the analysis and evaluation of prosody conversion techniques. In *Proc. IALP*, 2017.

[256] Berrak Sisman, Haizhou Li, and Kay Chen Tan. Transformation of prosody in voice conversion. *Proc. APSIPA*, 2017.

[257] P Song, YQ Bao, L Zhao, and CR Zou. Voice conversion using support vector regression. *Electronics letters*, 47(18):1045–1046, 2011.

[258] Alexander Sorin, Slava Shechtman, and Vincent Pollet. Uniform speech parameterization for multi-form segment synthesis. In *Proceedings of the INTERSPEECH*, 2011.

[259] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[260] Ioannis Stylianou. *Harmonic plus noise models for speech, combined with statistical methods, for speech and speaker modification*. PhD thesis, Ecole Nationale Supérieure des Télécommunications, 1996.

[261] Yannis Stylianou. Voice transformation: a survey. In *Proceedings of the ICASSP*, 2009.

[262] Yannis Stylianou, Olivier Cappé, and Eric Moulines. Continuous probabilistic transform for voice conversion. *IEEE Transactions on Speech and Audio Processing*, 6(2):131–142, 1998.

[263] Lifa Sun, Shiyin Kang, Kun Li, and Helen Meng. Voice conversion using deep bidirectional long short-term memory based recurrent neural networks. In *Proceedings of the ICASSP*, 2015.

[264] Lifa Sun, Kun Li, Hao Wang, Shiyin Kang, and Helen Meng. Phonetic posteriorgrams for many-to-one voice conversion without parallel data training. In *Proc. ICME*, 2016.

[265] Lifa Sun, Hao Wang, Shiyin Kang, Kun Li, and Helen M Meng. Personalized, cross-lingual tts using phonetic posteriorgrams. In *Proc. Interspeech*, 2016.

[266] Lifa Sun, Kun Li, Hao Wang, Shiyin Kang, and Mei Ling Helen Meng. Phonetic posteriorgrams for many-to-one voice conversion, 2018. US Patent App. 15/618,980.

[267] D Sündermann, Hermann Ney, and H Hoge. VTLN-based cross-language voice conversion. In *Proceedings of the ASRU*, 2003.

[268] David Sündermann. Voice conversion: State-of-the-art and future work. *Fortschritte der Akustik*, 31(2):735, 2005.

[269] David Sündermann. *Text-independent voice conversion.* PhD thesis, Universitätsbibliothek der Universität der Bundeswehr München, 2008.

[270] David Sündermann and Hermann Ney. An automatic segmentation and mapping approach for voice conversion parameter training. In *Proceedings of the AST*, 2003.

[271] David Sündermann, Antonio Bonafonte, Harald Höge, and Hermann Ney. Voice conversion using exclusively unaligned training data. In *Proceedings of the ACL/SEPLN*, 2004.

[272] David Sündermann, Antonio Bonafonte, Hermann Ney, and Harald Höge. A first step towards text-independent voice conversion. In *Proceedings of the ICSLP*, 2004.

[273] David Sündermann, Antonio Bonafonte, Hermann Ney, and Harald Höge. A study on residual prediction techniques for voice conversion. In *Proceedings of the ICASSP*, 2005.

[274] David Sündermann, Harald Hoge, Antonio Bonafonte, Hermann Ney, Alan Black, and Shri Narayanan. Text-independent voice conversion based on unit selection. In *Proceedings of the ICASSP*, 2006.

[275] David Sündermann, Harald Höge, Antonio Bonafonte, Hermann Ney, and Julia Hirschberg. Text-independent cross-language voice conversion. In *Proceedings of the INTERSPEECH*, 2006.

[276] David Sündermann, Harald Höge, Antonio Bonafonte, Hermann Ney, and Julia Hirschberg. TC-Star: Cross-language voice conversion revisited. In *TC-Star Workshop*. TC-Star Workshop, 2006.

[277] Antti Santeri Suni, Daniel Aalto, Tuomo Raitio, Paavo Alku, Martti Vainio, et al. Wavelets for intonation modeling in hmm speech synthesis. In *Proceedings of the SSW*, 2013.

[278] Shinnosuke Takamichi, Tomoki Toda, Alan W Black, and Satoshi Nakamura. Modulation spectrum-based post-filter for gmm-based voice conversion. In *Proceedings of the APSIPA*, 2014.

[279] Shinnosuke Takamichi, Tomoki Toda, Alan W Black, and Satoshi Nakamura. Modulation spectrum-constrained trajectory training algorithm for gmm-based voice conversion. In *Proceedings of the ICASSP*, 2015.

[280] Ryoichi Takashima, Tetsuya Takiguchi, and Yasuo Ariki. Exemplar-based voice conversion in noisy environment. In *Proceedings of the SLT*, 2012.

[281] Ryoichi Takashima, Ryo Aihara, Tetsuya Takiguchi, and Yasuo Ariki. Noise-robust voice conversion based on spectral mapping on sparse space. In *Proceedings of the SSW*, 2013.

[282] Akira Tamamori, Tomoki Hayashi, Kazuhiro Kobayashi, Kazuya Takeda, and Tomoki Toda. Speaker-dependent wavenet vocoder. In *Proc. Interspeech*, 2017.

[283] Masatsune Tamura, Masahiro Morita, Takehiko Kagoshima, and Masami Akamine. One sentence voice adaptation using gmm-based frequency-warping and shift with a sub-band basis spectrum model. In *Proceedings of the ICASSP*, 2011.

[284] Kou Tanaka, Tomoki Toda, Graham Neubig, Sakriani Sakti, and Satoshi Nakamura. A hybrid approach to electrolaryngeal speech enhancement based on spectral subtraction and statistical voice conversion. In *Proceedings of the INTERSPEECH*, 2013.

[285] Kou Tanaka, Hirokazu Kameoka, Takuhiro Kaneko, and Nobukatsu Hojo. Atts2s-vc: Sequence-to-sequence voice conversion with attention and context preservation mechanisms. *arXiv preprint arXiv:1811.04076*, 2018.

[286] Daisuke Tani, Tomoki Toda, Yamato Ohtani, Hiroshi Saruwatari, and Kiyohiro Shikano. Maximum a posteriori adaptation for many-to-one eigenvoice conversion. In *Proceedings of the INTERSPEECH*, 2008.

[287] Jianhua Tao, Yongguo Kang, and Aijun Li. Prosody conversion from neutral speech to emotional speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(4): 1145–1154, 2006.

[288] Jianhua Tao, Meng Zhang, Jani Nurminen, Jilei Tian, and Xia Wang. Supervisory data alignment for text-independent voice conversion. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(5):932–943, 2010.

[289] Joshua B Tenenbaum and William T Freeman. Separating style and content. In *Advances in neural information processing systems*, pages 662–668, 1997.

[290] Fabio Tesser, Enrico Zovato, Mauro Nicolao, and Piero Cosi. Two vocoder techniques for neutral to emotional timbre conversion. In *Proceedings of the SSW*, 2010.

[291] Xiaohai Tian, Zhizheng Wu, SW Lee, and Eng Siong Chng. Correlation-based frequency warping for voice conversion. In *Proceedings of the ISCSLP*, pages 211–215. IEEE, 2014.

[292] Xiaohai Tian, Zhizheng Wu, Siu Wa Lee, Nguyen Quy Hy, Eng Siong Chng, and Minghui Dong. Sparse representation for frequency warping based voice conversion. In *Proceedings of the ICASSP*, 2015.

[293] Xiaohai Tian, Zhizheng Wu, Siu Wa Lee, Nguyen Quy Hy, Minghui Dong, and Eng Siong Chng. System fusion for high-performance voice conversion. In *Proceedings of the INTERSPEECH*, 2015.

[294] Xiaohai Tian, Siu Wa Lee, Zhizheng Wu, Eng Siong Chng, Haizhou Li, Xiaohai Tian, Siu Wa Lee, Zhizheng Wu, Eng Siong Chng, and Haizhou Li. An exemplar-based approach to frequency warping for voice conversion. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 2017.

[295] Xiaohai Tian, Junchao Wang, Haihua Xu, Eng-Siong Chng, and Haizhou Li. Average modeling approach to voice conversion with non-parallel data. In *Proc. Odyssey*, 2018.

[296] D Michael Titterington, Adrian FM Smith, Udi E Makov, et al. *Statistical analysis of finite mixture distributions*, volume 7. Wiley New York, 1985.

[297] T Toda, K Nakamura, H Saruwatari, K Shikano, et al. Alaryngeal speech enhancement based on one-to-many eigenvoice conversion. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(1):172–183, 2014.

[298] Tomoki Toda and Kiyohiro Shikano. NAM-to-speech conversion with gaussian mixture models. In *Proceedings of the INTERSPEECH*, 2005.

[299] Tomoki Toda, Hiroshi Saruwatari, and Kiyohiro Shikano. Voice conversion algorithm based on gaussian mixture model with dynamic frequency warping of straight spectrum. In *Proceedings of the ICASSP*, 2001.

[300] Tomoki Toda, Alan W Black, and Keiichi Tokuda. Acoustic-to-articulatory inversion mapping with gaussian mixture model. In *Proceedings of the INTERSPEECH*, 2004.

[301] Tomoki Toda, Alan W Black, and Keiichi Tokuda. Spectral conversion based on maximum likelihood estimation considering global variance of converted parameter. In *Proceedings of the ICASSP*, 2005.

[302] Tomoki Toda, Yamato Ohtani, and Kiyohiro Shikano. Eigenvoice conversion based on gaussian mixture model. In *Proceedings of the INTERSPEECH*, 2006.

[303] Tomoki Toda, Alan W Black, and Keiichi Tokuda. Voice conversion based on maximum-likelihood estimation of spectral parameter trajectory. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(8):2222–2235, 2007.

[304] Tomoki Toda, Yamato Ohtani, and Kiyohiro Shikano. One-to-many and many-to-one voice conversion based on eigenvoices. In *Proceedings of the ICASSP*, 2007.

[305] Tomoki Toda, Alan W Black, and Keiichi Tokuda. Statistical mapping between articulatory movements and acoustic spectrum using a gaussian mixture model. *Speech Communication*, 50(3):215–227, 2008.

[306] Tomoki Toda, Takashi Muramatsu, and Hideki Banno. Implementation of computationally efficient real-time voice conversion. In *Proceedings of the INTERSPEECH*, 2012.

[307] Tomoki Toda, Mikihiro Nakagiri, and Kiyohiro Shikano. Statistical voice conversion techniques for body-conducted unvoiced speech enhancement. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(9):2505–2517, 2012.

[308] Tomoki Toda, Daisuke Saito, Fernando Villavicencio, Junichi Yamagishi, Mirjam Wester, Zhizheng Wu, Ling-Hui Chen, et al. The voice conversion challenge 2016. In *Proceedings of the INTERSPEECH*, 2016.

[309] Keiichi Tokuda and Heiga Zen. Directly modeling speech waveforms by neural networks for statistical parametric speech synthesis. In *Proceedings of the ICASSP*, 2015.

[310] Keiichi Tokuda, Takao Kobayashi, and Satoshi Imai. Speech parameter generation from HMM using dynamic features. In *Proceedings of the ICASSP*, 1995.

[311] Viet-Anh Tran, Gérard Bailly, Hélène Lœvenbruck, and Tomoki Toda. Improvement to a nam-captured whisper-to-speech system. *Speech communication*, 52(4):314–326, 2010.

[312] Emir Turajlic, Dimitrios Rentzos, Saeed Vaseghi, and Ching-Hsiang Ho. Evaluation of methods for parameteric formant transformation in voice conversion. In *Proceeding of the ICASSP*, 2003.

[313] Oytun Türk. *Cross-lingual voice conversion*. PhD thesis, Bogaziçi University, 2007.

[314] Oytun Türk and Levent M Arslan. Voice conversion methods for vocal tract and pitch contour modification. In *Proceedings of the INTERSPEECH*, 2003.

[315] Oytun Turk and Levent M Arslan. Donor selection for voice conversion. In *Proceedings of the EUSIPCO*, 2005.

[316] Oytun Turk and Levent M Arslan. Robust processing techniques for voice conversion. *Computer Speech & Language*, 20(4):441–467, 2006.

[317] Oytun Türk and Marc Schröder. A comparison of voice conversion methods for transforming voice quality in emotional speech synthesis. In *Proceedings of the INTERSPEECH*, 2008.

[318] Oytun Turk and Marc Schroder. Evaluation of expressive speech synthesis with voice conversion and copy resynthesis techniques. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(5):965–973, 2010.

[319] Oytun Turk, Osman Buyuk, Ali Haznedaroglu, and Levent Mustafa Arslan. Application of voice conversion for cross-language rap singing transformation. In *Proceedings of the ICASSP*, 2009.

[320] Eiji Uchino, Kazuaki Yano, and Tadahiro Azetsu. A self-organizing map with twin units capable of describing a nonlinear input–output relation applied to speech code vector mapping. *Information Sciences*, 177(21):4634–4644, 2007.

[321] AJ Uriz, PD Aguero, JC Tulli, EL Gonzalez, and A Bonafonte. Voice conversion using frame selection and warping functions. *Proceedings of the RPIC*, 2009.

[322] Alejandro Uriz, Pablo Daniel Agüero, Daniel Erro, and Antonio Bonafonte. Voice conversion using frame selection. *Reporte Interno Laboratorio de Comunicaciones-UNMdP*, 2008.

[323] Alejandro José Uriz, Pablo Daniel Agüero, Antonio Bonafonte, and Juan Carlos Tulli. Voice conversion using k-histograms and frame selection. In *Proceedings of the INTERSPEECH*, 2009.

[324] Yosuke Uto, Yoshihiko Nankaku, Tomoki Toda, Akinobu Lee, and Keiichi Tokuda. Voice conversion based on mixtures of factor analyzers. In *Proceeding of the ICSLP*, 2006.

[325] H. Valbret, E. Moulines, and J. P. Tubach. Voice transformation using PSOLA technique. *Speech Communication*, 11(2):175–187, 1992.

[326] Hélène Valbret, Eric Moulines, and Jean-Pierre Tubach. Voice transformation using PSOLA technique. In *Proceedings of the ICASSP*, 1992.

[327] Cassia Valentini-Botinhao, Zhizheng Wu, and Simon King. Towards minimum perceptual error training for DNN-based speech synthesis. In *Proceedings of the INTERSPEECH*, 2015.

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

[329] Christophe Veaux and Xavier Rodet. Intonation conversion from neutral to expressive speech. In *Proceedings of the INTERSPEECH*, 2011.

[330] Christophe Veaux, Junichi Yamagishi, Kirsten MacDonald, et al. Cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit. 2017.

[331] Ashish Verma and Arun Kumar. Voice fonts for individuality representation and transformation. *ACM Transactions on Speech and Language Processing (TSLP)*, 2(1):4, 2005.

[332] Fernando Villavicencio and Jordi Bonada. Applying voice conversion to concatenative singing-voice synthesis. In *Proceedings of the INTERSPEECH*, 2010.

[333] Fernando Villavicencio, Jordi Bonada, and Yuji Hisaminato. Observation-model error compensation for enhanced spectral envelope transformation in voice conversion. In *Proceedings of the MLSP*, 2015.

[334] Damien Vincent, Olivier Rosec, and Thierry Chonavel. A new method for speech synthesis and transformation based on an arx-lf source-filter decomposition and hnm modeling. In *Proceedings of the ICASSP*, 2007.

[335] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408, 2010.

[336] Wolfgang Wahlster. *Verbmobil: foundations of speech-to-speech translation*. Springer Science & Business Media, 2000.

[337] Jack M Wang, David J Fleet, and Aaron Hertzmann. Multifactor gaussian process models for style-content separation. In *Proceedings of the 24th international conference on Machine learning*, pages 975–982. ACM, 2007.

[338] Miaomiao Wang, Miaomiao Wen, Keikichi Hirose, and Nobuaki Minematsu. Emotional voice conversion for mandarin using tone nucleus model–small corpus and high efficiency. In *Proceedings of the Speech Prosody*, 2012.

[339] Weiran Wang, Raman Arora, Karen Livescu, and Jeff A Bilmes. On deep multi-view representation learning. In *ICML*, pages 1083–1092, 2015.

[340] Zexun Wang and Yibiao Yu. Multi-level prosody and spectrum conversion for emotional speech synthesis. In *Proceedings of the ICSP*, 2014.

[341] Tomomi Watanabe, Takahiro Murakami, Munehiro Namba, Tetsuya Hoya, and Yoshihisa Ishida. Transformation of spectral envelope for voice conversion based on radial basis function networks. In *Proceedings of the ICSLP*, 2002.

[342] Mirjam Wester, Zhizheng Wu, and Junichi Yamagishi. Analysis of the voice conversion challenge 2016 evaluation results. In *Proceedings of the INTERSPEECH*, 2016.

[343] Mirjam Wester, Zhizheng Wu, and Junichi Yamagishi. Multidimensional scaling of systems in the voice conversion challenge 2016. In *Proceedings of the SSW*, 2016.

[344] Alan Wrench. The MOCHA-TIMIT articulatory database. Queen Margaret University College, 1999.

[345] Chung-Hsien Wu, Chi-Chun Hsia, Te-Hsien Liu, and Jhing-Fa Wang. Voice conversion using duration-embedded bi-HMMs for expressive speech synthesis. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(4):1109–1116, 2006.

[346] Jie Wu, Dongyan Huang, Lei Xie, and Haizhou Li. Denoising recurrent neural network for deep bidirectional lstm based voice conversion. *Proc. Interspeech*, 2017.

[347] Yi-Chiao Wu, Hsin-Te Hwang, Chin-Cheng Hsu, Yu Tsao, and Hsin-Min Wang. Locally linear embedding for exemplar-based spectral conversion. In *Proceedings of the INTERSPEECH*, 2016.

[348] Zhizheng Wu and Haizhou Li. Voice conversion versus speaker verification: an overview. *APSIPA Transactions on Signal and Information Processing*, 3:e17, 2014.

[349] Zhizheng Wu, Tomi Kinnunen, Engsiong Chng, and Haizhou Li. Text-independent F0 transformation with non-parallel data for voice conversion. In *Proceedings of the INTERSPEECH*, 2010.

[350] Zhizheng Wu, Tomi Kinnunen, Eng Siong Chng, and Haizhou Li. Mixture of factor analyzers using priors from non-parallel speech for voice conversion. *IEEE Signal Processing Letters*, 19(12):914–917, 2012.

[351] Zhizheng Wu, Eng Siong Chng, and Haizhou Li. Conditional restricted boltzmann machine for voice conversion. In *Proceedings of the ChinaSIP*, 2013.

[352] Zhizheng Wu, Anthony Larcher, Kong-Aik Lee, Engsiong Chng, Tomi Kinnunen, and Haizhou Li. Vulnerability evaluation of speaker verification under voice conversion spoofing: the effect of text constraints. In *Proceedings of the INTERSPEECH*, 2013.

[353] Zhizheng Wu, Tuomas Virtanen, Tomi Kinnunen, Eng Siong Chng, and Haizhou Li. Exemplar-based voice conversion using non-negative spectrogram deconvolution. In *Proceedings of the SSW*, 2013.

[354] Zhizheng Wu, Tuomas Virtanen, Tomi Kinnunen, Engsiong Chng, and Haizhou Li. Exemplar-based unit selection for voice conversion utilizing temporal information. In *Proceedings of the INTERSPEECH*, 2013.

[355] Zhizheng Wu, Eng Siong Chng, and Haizhou Li. Joint nonnegative matrix factorization for exemplar-based voice conversion. In *Proceedings of the INTERSPEECH*, 2014.

[356] Zhizheng Wu, Tuomas Virtanen, Eng Siong Chng, and Haizhou Li. Exemplar-based sparse representation with residual compensation for voice conversion. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 22(10):1506–1521, 2014.

[357] Feng-Long Xie, Yao Qian, Yuchen Fan, Frank K Soong, and Haifeng Li. Sequence error (se) minimization training of neural network for voice conversion. In *Proceedings of the INTERSPEECH*, 2014.

[358] Feng-Long Xie, Yao Qian, Frank K Soong, and Haifeng Li. Pitch transformation in neural network based voice conversion. In *Proceedings of the ISCSLP*, 2014.

[359] Ning Xu, Yibing Tang, Jingyi Bao, Aiming Jiang, Xiaofeng Liu, and Zhen Yang. Voice conversion based on gaussian processes by coherent and asymmetric training with limited training data. *Speech Communication*, 58:124–138, 2014.

[360] Junichi Yamagishi, Christophe Veaux, Simon King, and Steve Renals. Speech synthesis technologies for individuals with vocal disabilities: Voice banking and reconstruction. *Acoustical Science and Technology*, 33(1):1–5, 2012.

[361] Ming-Han Yang, Hung-Shin Lee, Yu-Ding Lu, Kuan-Yu Chen, Yu Tsao, Berlin Chen, and Hsin-Min Wang. Discriminative autoencoders for acoustic modeling. In *Proc. Interspeech*, 2017.

[362] Hui Ye and Steve Young. Perceptually weighted linear transformations for voice conversion. In *Proceedings of the INTERSPEECH*, 2003.

[363] Hui Ye and Steve Young. Voice conversion for unknown speakers. In *Proceedings of the INTERSPEECH*, 2004.

[364] Hui Ye and Steve Young. Quality-enhanced voice morphing using maximum likelihood transformations. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(4): 1301–1312, 2006.

[365] Zhenjun Yue, Xiang Zou, Yongxing Jia, and Hao Wang. Voice conversion using HMM combined with GMM. In *Proceedings of the CISP*, 2008.

[366] Kaori Yutani, Yosuke Uto, Yoshihiko Nankaku, Akinobu Lee, and Keiichi Tokuda. Voice conversion based on simultaneous modelling of spectrum and f0. In *Proceedings of the ICASSP*, 2009.

[367] Neil Zeghidour, Gabriel Synnaeve, Nicolas Usunier, and Emmanuel Dupoux. Joint learning of speaker and phonetic similarities with siamese networks. In *Proceedings of the INTERSPEECH*, 2016.

[368] Neil Zeghidour, Gabriel Synnaeve, Maarten Versteegh, and Emmanuel Dupoux. A deep scattering spectrum-deep siamese network pipeline for unsupervised acoustic modeling. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 4965–4969. IEEE, 2016.

[369] Heiga Zen, Yoshihiko Nankaku, and Keiichi Tokuda. Continuous stochastic feature mapping based on trajectory hmms. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(2):417–430, 2011.

[370] Jian Zhang, Jun Sun, and Beiqian Dai. Voice conversion based on weighted least squares estimation criterion and residual prediction from pitch contour. In *Affective Computing and Intelligent Interaction*, pages 326–333. Springer, 2005.

[371] Meng Zhang, Jianhua Tao, Jilei Tian, and Xia Wang. Text-independent voice conversion based on state mapped codebook. In *Proceedings of the ICASSP*, 2008.

[372] Meng Zhang, Jiaohua Tao, Jani Nurminen, Jilei Tian, and Xia Wang. Phoneme cluster based state mapping for text-independent voice conversion. In *Proceedings of the ICASSP*, 2009.

[373] Guanlong Zhao and Ricardo Gutierrez-Osuna. Exemplar selection methods in voice conversion. In *Proc. ICASSP*, 2017.

[374] Yingbo Zhou, Devansh Arpit, Ifeoma Nwogu, and Venu Govindaraju. Is joint training better for deep auto-encoders? *arXiv preprint arXiv:1405.1380*, 2014.

[375] Parham Zolfaghari and Tony Robinson. A formant vocoder based on mixtures of gaussians. In *Proceedings of the ICASSP*, 1997.

[376] Tudor-Cătălin Zorilă, Daniel Erro, and Inma Hernaez. Improving the quality of standard gmm-based voice conversion systems by considering physically motivated linear transformations. In *Advances in Speech and Language Technologies for Iberian Languages*, pages 30–39. Springer, 2012.