Voice Conversion Based on Deep Learning

Doctoral Dissertation

Voice Conversion Based on Deep Learning

TORU NAKASHIKA

Graduate School of System Informatics

Kobe University

July, 2014

Doctoral Dissertation
Voice Conversion Based on Deep Learning

TORU NAKASHIKA

Voice conversion (VC) is a technique to modify source speaker's speech as if it was spoken by a target speaker. Specific information in the speech of a source speaker is transformed into that of a target speaker while retaining linguistic information. VC techniques have recently attracted much attention in speech-signal processing, since they have been applied to various tasks such as speech enhancement and helping for articlulation disorders.

In recent years, many approaches for VC were proposed while the results were not ideal. Gaussian mixture models (GMMs) are most widely used for VC, and a number of improvements have been proposed. However, GMM-based approaches were not sufficient for capturing complex information in speech data because the conversion function was based on linear transformation.

The thesis proposes VC methods that rely on non-linear transformation functions using deep learning approaches. Deep learning has now become a hot topic in machine learning and signal processing with its great success. The thesis includes four approaches to realize VC based on deep learning. The first approach uses a joint density model of a restricted Boltzmann machine (RBM), which is a basic probabilistic model used in deep learning, for capturing joint distribution of source speaker's speech and target speaker's speech. Secondly, speaker dependent models of RBMs are used. Thirdly, conditional RBMs (CRBMs) are alternatively used, considering time series data. Finally, recurrent temporal RBMs (RTRBMs) are used for capturing latent temporal dependencies. Every approach uses RBMs to represent latent features, leading to non-linear stacked (deep) conversion.

The experimental results show that every proposed methods provided much better performance than the conventional GMM-based approach in terms of subjective and objective criteria.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Voice conversion (VC) techniques, by which specific information in the speech of a source speaker is transformed into that of a target speaker while retaining linguistic information, have recently attracted much attention in speech-signal processing. VC allows us to convert the speech signal uttered by a source speaker to as the target speaker. VC techniques have been applied to various tasks. The most commonly used case is speech synthesis from a text, where personalized voices are created from existing person's speech using VC. VC techniques are also used for speech enhancement [2], emotion conversion [3], speaking assistance [4], and other applications [5, 6].

However, VC is a challenging task due to the fact that there is no unique correct answer. Every time a source speaker utters the same sentence, the observed spectrum is different. Furthermore, the perception of the conversion quality is subjective. Therefore, listening tests that take much time must be used for evaluation of the VC systems. Some objective measures such as mel-cepstral distortion (MCD) or spectral distortion improvement ratio (SDIR) are also used to complement the subjective evaluation [7].

In general, speech includes linguistic and nonlinguistic infomation. It is considered that some hints of speaker identity exist in the nonlinguistic information more than in the linguistic information. The nonlinguistic factors can be categorized into two parts: sociological and physiological factors. Sociological factors, including the place of birth, the social class, and the age of the speaker, mainly affect prosodic features (pitch contour, duration, rhythm, etc). On the other hand, physiological factors (the shape of the vocal tract) directly affect the spectral information and determine the individuality. It is reported that the most important acoustic features to identify the speaker include the third formant, the fourth formant, the fundamental frequency, and the closing phase of the glottal

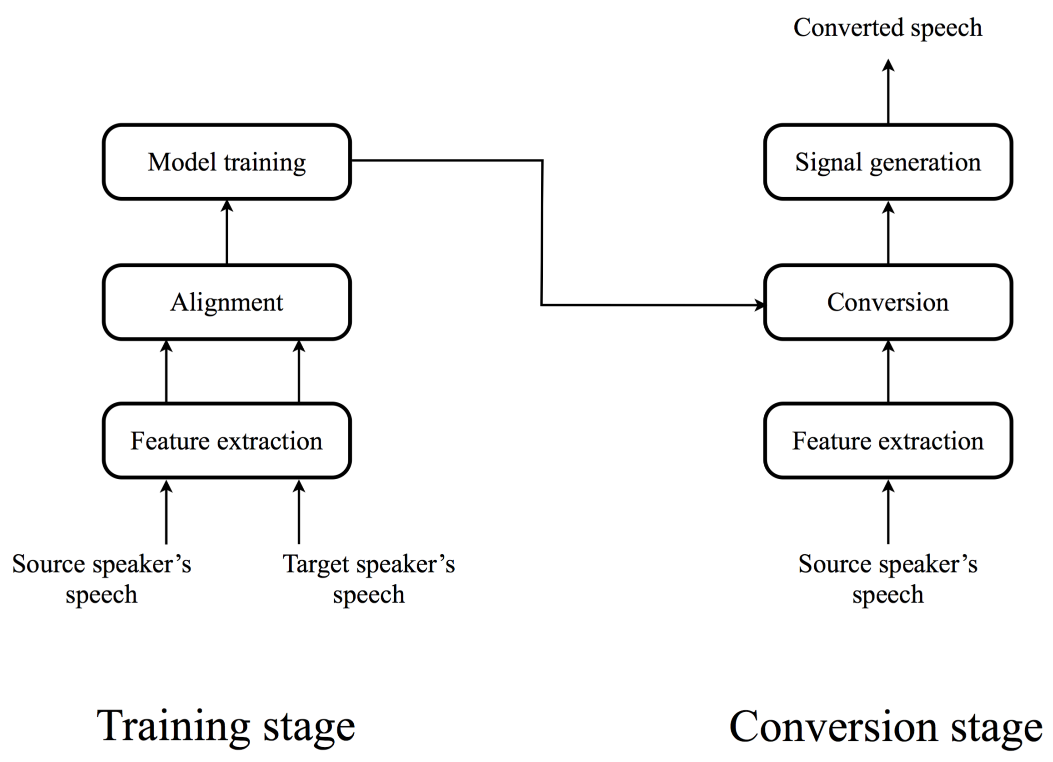


Figure 1.1: Flowchart of a typical VC system.

wave [8].

The most existing VC systems deal with the conversion of spectral features, and that will be focused on in this thesis. However, prosodic features, such as fundamental frequency (F_0), can also be seen as importnat factors of speaker identity. Helander *et al.* showed that when the true prosody features are used, we can recognize the person who is familiar to us [9]. Nevertheless, we obtain good results from a simple stastical mean and variance scaling methods for F_0 conversion. More advanced F_0 conversion methods have also been proposed [10, 11, 12]

1.2 Approaches

Most VC systems have a system flow as shown in Figure 1.1. The system can broadly be divided into two stages: training and conversion stage. Both stages begin with feature extraction. In this process, acoustic features, such as such as mel-frequency cepstral coefficients (MFCC), cepstral coefficients (CC), or linear prediction coefficients (LPC), are extracted from the speech signals. In the training stage, the extracted time-series features are aligned to adjust the time positions of the source speaker's features and those of the target speaker's features (such alined data is called parallel data). Therefore, typical VC systems require the pairs of speech signals uttering the same sentence by a source and a target speakers. Dynamic programming (DP) [13] is often used for the alignment process. Any stastical models or nonstastical models are trained using the obtained parallel data. In conversion stage, the acoustic features are extracted from source speaker's speech and feed to the model, resulting in the acoustic features that is supposed to be the target speakers' one. Finally, the features are back-projected into a speech signal. In such way, we obtain converted speech.

For the models, various statistical approaches have been studied, including those discussed in [14, 15]. Among these approaches, mapping methods based on the Gaussian mixture model (GMM) [16] are widely used, and a number of improvements have been proposed. Some of the approaches do not require parallel data, because they use a GMM-adaptation technique [17], eigen-voice GMM [18, 19] or probabilistic integration model [20].

However, GMM-based approaches rely on "shallow" voice conversion—methods are based on piecewise-linear transformation. Because the shape of the vocal tract is generally non-linear, non-linear voice conversion is more compatible with human speech. To capture the characteristics of speech more precisely, a deeper non-linear architecture with more hidden layers is required. One example of deeper VC methods was proposed by Desai *et al.* [21] based on neural networks (NN). In GMM-based approaches, the conversion is achieved so as to maximize the conditional probability calculated from a joint probability of source and target speech, where the joint model is trained beforehand. In contrast, NN-based approaches directly train the conditional probability, which converts the feature vector of a source speaker to that of a target speaker. These discriminative approaches have been shown to perform better than generative approaches, such as GMM-based approaches, in speech recognition and synthesis, as well as in VC [22, 23]. For these reasons, NN-based approaches achieve relatively high performance if the training samples are carefully prepared [21].

These approaches often suffer from over-smoothing or over-fitting problems. GMM-based approaches represent acoustic features using multiple Gaussian distributions, which are estimated by averaging observations with similar context descriptions in the training. Therefore, the outputs of a GMM distribute near the modes (means) of the Gaussians, which leads to problems with over-smoothing. Furthermore, over-fitting problems arise when we give more Gaussian mixtures due to precise estimation of the observed distribution. In NN-based approaches, the model is often over-fitted due to its complexity. The model exaggerates small fluctuations in the unknown data if the amount of training data is not sufficient for the number of parameters.

Some methods have been proposed for alleviating the over-smoothing of GMMs, such as the global variance model [24], a minimizing-divergence model [25], and post-filtering [26]. Other approaches that reduce over-smoothing have also been proposed, such as canonical correlation analysis (CCA) [27] and an exemplar-based VC system using NMF (non-negative matrix factorization) [28, 29]. In our earlier work [30], we proposed a new VC technique that copes with over-fitting problems in NN-based approaches using a combination of speaker-dependent restricted Boltzmann machines (RBMs) [31] (or deep belief nets; DBN [32]), which capture high-order features in an unsupervised manner, and concatenating NNs. These graphical models are better than GMMs at representing the distribution of high-dimensional observations with cross-dimension correlations in speech synthesis [33] and in speech recognition [34]. Since Hinton et al. introduced an effective training algorithm for the DBN in 2006 [32], the use of deep learning has rapidly spread in the field of signal processing, as well as in speech signal processing. RBMs (or DBNs) have been used, for example, for recognition of handwritten characters [32], recognition of 3-D objects [35], and machine transliteration [36].

1.3 Purpose

The objective of this work is to get an overview of the current VC methods and to develop our own solution. In this paper, we extend our earlier work in [30] to systematically capture time information as well as latent (deep) relationships between source-speaker and target-speaker features in a single network. We do this by combining speaker-dependent recurrent temporal restricted Boltzmann machines (RTRBMs [37]) and a concatenating NN. An RTRBM, which is an extension of an RBM, is a non-linear probabilistic model used to capture temporal dependencies in time-series data. Despite its simplicity, this model does a good job of describing meaningful sequences such as video [37] and music [38]. In our approach, we first train two RTRBMs: one exclusively for the source and one exclusively for the target speakers. We train them independently using segmented training data prepared for each speaker. Then we train an NN using the projected features. Finally we fine-tune the networks as a single recurrent NN. Because the training data for the source speaker RTRBM includes various phonemes particular to the speaker, the speaker-dependent network tries to capture the abstractions to maximally express the training data with abundant speaker-specific information and less phonological information. Furthermore, the network receives a collection of time-series feature vectors with the conditional models, enabling it to discover temporal correlations in the high-order space. Therefore, we expect that the features of these time-dependent, high-order spaces can be converted much more easily than can the voice features of the original cepstrum-based space.

1.4 Related works

Similar research can be found in [1] and [39]. Wu et al. employed a conditional restricted Boltzmann machine (CRBM), another model for representing time-series data that was proposed by Taylor *et al.* [40] for capturing linear and non-linear relationships between source and target features [1]. Chen et al. also used an RBM to model the joint spectral distribution instead of using a conventional joint-density GMM [39]. Unlike these approaches, which are based on joint models, our method trains two exclusive RTRBMs, one for each speaker.

1.5 Outline

Starting in the Chapter 2, we list some basic ideas and technologies related to voice conversion. Chapters 3 describes three conventional methods on voice conversion: GMM-based approach, NN-based approach, and NMF-based approach. In Chapter 4, the algorithm for voice conversion using a JD-RBM is described in details. Details about the voice conversion using speaker-dependent models in deep-learning-based framework are provided in Chapter 5, which will be the basic method in this thesis. The extendend models considering time-related information are described in Chapter 6 using speaker-dependent CRBMs, and in Chapter 7 using speaker-dependent RTRBMs. Finally, Chapter 8 concludes the thesis.

Chapter 2

Acoustic Features and Basics

In this chapter, we review common acousitc features used in voice conversion or other speech applications. First, cepstrum analysis, which is the basic idea to extract important information from a speech signal, is presented. Secondly, we will talk about MFCC, which is well-known features in speech signal processing. Finally, analyzing method of speech using STRAIGHT is presented.

2.1 Cepstrum

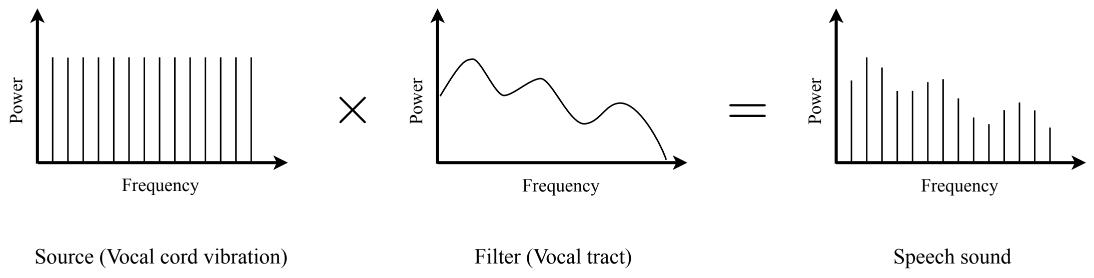


Figure 2.1: Source filter model: speech sound can be generated by multiplying spectra of the vocal cord vibration (source) and spectra of the vocal tract (filter).

Speech signal is generated from vibration of the vocal cord. The vibration passes a vocal tract of the speaker, and arrives at the listner's ears or microphones. When we vibrate our vocal cords and change the shapes of our mouthes and vocal tracts, the sounds of various phonemes such as /a/ or /i/ can be generated. This speech generating process is modeled as a source filter model (Figure 2.1). The filter associated with the vocal tract is called formant, and the fundamental frequency from the vibration is called pitch. In speech signal processing, the information related to the formant that determines phonemes is fairly important. Therefore, when the system recognizes a given speech, we can obtain better results with the formant information than with the original spectrum.

One well-known approach for extracting formants is cepstrum analysis [41]. The cepstrum is obtained as follows: 1) execute Fourier transform to the given speech, 2) take absolute and logarithm, and 3) execute inverse Fourier transform. Letting $G(\omega)$ and $H(\omega)$ be spectrum of vocal cord and spectrum of vocal tract, respectively, the spectrum of the speech $S(\omega)$ is represented as:

$$S(\omega) = G(\omega) \cdot H(\omega). \tag{2.1}$$

When we apply logarithm and inverse Fourier transform to Eq. (2.1), we obtain

$$\log |S(\omega)| = \log |G(\omega)| + \log |H(\omega)| \tag{2.2}$$

and

$$\begin{align} S_{cep}(d) &= DFT^{-1}\{\log |S(\omega)|\} \tag{2.3} \\ &= DFT^{-1}\{\log |G(\omega)|\} + DFT^{-1}\{\log |H(\omega)|\}, \tag{2.4} \end{align}$$

where $S_{cep}(d)$ indicates the dth cepstrum (d is quefrency axis).

When we regard each spectra in Figure 2.1 as a signal, we notice that the vocal cord $G(\omega)$ is a signal that rapidly changes, and the vocal tract $H(\omega)$ is a signal that slowly changes, in constrast. By applying inverse Fourier transform to these signals, $DFT^{-1}\{\log |G(\omega)|\}$ appears in high quefrency and $DFT^{-1}\{\log |H(\omega)|\}$ appears in low quefrency. Furthermore, as Eq. (2.4) shows, a speech signal is represented as a sum of the vocal cord information and the vocal tract information. Therefore, the vocal tract information $DFT^{-1}\{\log |H(\omega)|\}$ can be easily obtained by simple substruction. In other words, the formant information is extracted by liftering (low-pass filtering in quefrency) as shown in Figure 2.2.

2.2 MFCC

The cepstrum features introduced in the previous section was obtained as linear log spectrum (therefore, it is also called linear frequency cepstral coefficient; LFCC). On the other hand, there is another formant extraction method called mel-frequency cepstrum coefficients (MFCC) [42]. MFCC is extracted from the transformation on the mel-scale, which approximates the human-auditory-sensitive scale to pitches.

When we, human-beings, hear something, the auditory sensitivity gets poorer as the pitch is higher. That means our frequency resolution is very low in high

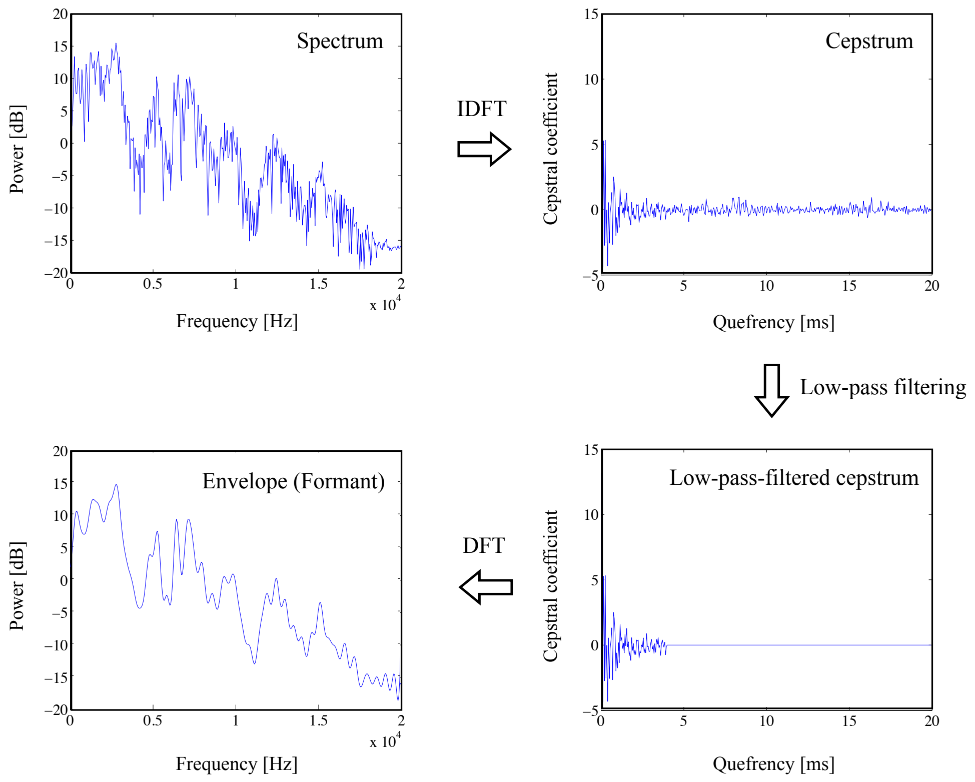


Figure 2.2: Formant extraction using cepstrum analysis. This example uses a speech signal where "a" is uttered.

frequency and high in low frequency. The relationship is not linear; but non-linear as approximated by

$$f' = 1127.01048 \log(1 + \frac{f}{700}), \tag{2.5}$$

where f' is called mel frequency. Typically, we use filter-bank representation instead of using the coefficients themselves. In the MFCC approach, mel-filter-banks as shown in Figure 2.3 are used. Each filter has a triangle shape, and outputs the sum value of the multiplication. The power spectrum on the mel-scale frequency $M(i)$ is obtained by

$$M(i) = \sum_{f=f'_{i-1}}^{f'_{i+1}} W_i(f) \cdot |X(f)|^2. \tag{2.6}$$

From Eq. (2.6), the MFCC can be calculated as follows:

$$M_{cep}(d) = DFT^{-1}\{\log M(i)\}. \tag{2.7}$$

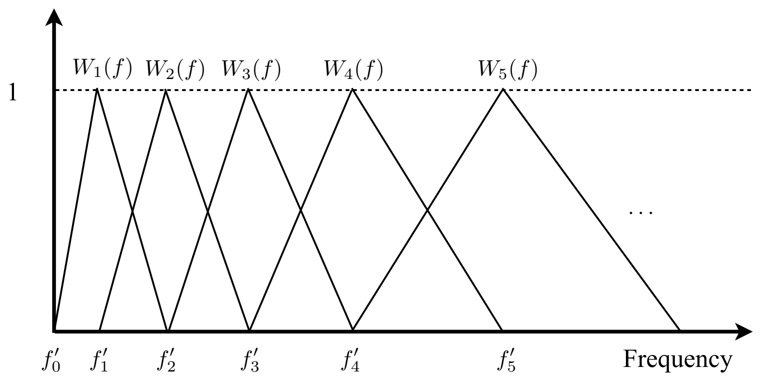


Figure 2.3: Mel-scale filter banks.

MFCC contains some useful information to represent the speech with small numbbers of dimensions; therefore, most works in speech signal processing, specifically in speech recognition, deal with the features. However, as the MFCC is based on filter-bank calculation, it is in general difficult to reconstruct the speech signal from the obtained MFCC (because the values in a filter-bank are summed up with the weights.) Milner and Shao proposed an approximation approach for the speech reconstruction from MFCC using a source-filter model [43]. Other approaches such as an approach using z-transform [44, 45] and an approach using a mel log spectral approximation (MLSA) filter [46] have also been proposed.

2.3 STRAIGHT

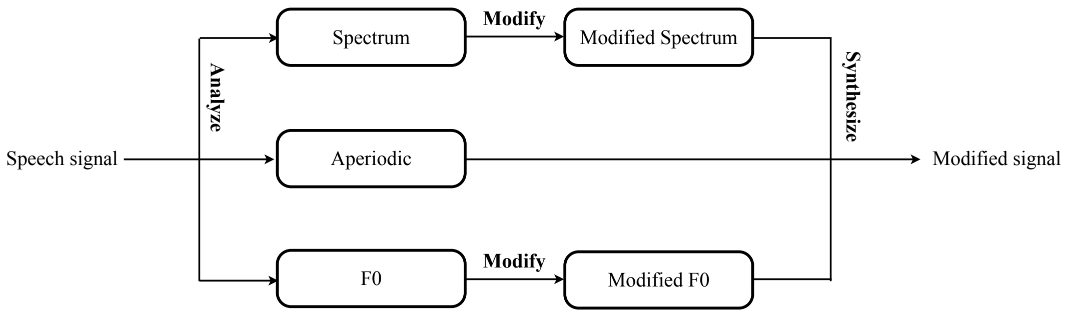


Figure 2.4: Modifying a speech signal using STRAIGHT.

STRAIGHT [47, 48] is a package tool provided by Kawahara that analyzes, modify, and synthesizes speech, written as MATLAB codes. The tool extracts three components from a speech signal: spectrum parameters, fundamental frequencies and aperiodic parameters. The analysis and synthesis quality is fairly high, so many researchers in speech signal processing loves to use it. Figure 2.4 depicts the common way to use the STRAIGHT for modifying a speech signal. The spectrum features extracted using STRAIGHT resembles formant information in cepstrum analysis. Therefore, in voice conversion, we usually modify the spectrum features and the F_0 to as the desired ones (we do not change the aperiodic features in the synthesis stage). The spectrum features are also reffered as STRAIGHT spectrum, STRAIGHT parameters, or just spectrum. In the remaining of this thesis, we refer to spectrum as the spectrum parameters extracted from a speech signal using STRAIGHT.

We can also extract MFCC features from the STRAIGHT spectrum. As discussed in the previous section, the (approximate) MFCC can further be back-projected into the spectrum space. The raw spectrum obtained using STRAIGHT tends to be high-dimensional. If we focus on modifying acoustic features in MFCC space after executing STRAIGHT, we can reduce the dimensional sizes of the interest features, which leads to high performance in estimation of a model, and obtain high-quality speech sound.

Chapter 3

Conventional VC methods

This chapter presents conventional voice conversion (VC) methods—approaches based on Gaussian mixture models (GMMs), approaches based on neural networks (NN), and approaches based on non-negative matrix factorization (NMF). These approaches will be compared with the proposed methods in the remaining chapters. The problems of each method are also declared in this chapter.

As discussed in section 1.2, in order to estimate the models, most existing VC approaches need sets of speech signals where the same sentences are uttered by a souce speaker and a target speaker. Furthermore, the training data must be aligned at the frame level. The aligned data is called parallel data, and is often obtained using dynamic programming (DP), or saying dynamic time warping (DTW) [13, 16, 49]. Such feature vectors are used for training each model. Let us refer the D-dimensional feature vectors at each frame of the source speaker and the target speaker as $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively. Assuming that the parallel data includes T frames, training data consists of the source speaker's set $\mathbf{X} \ni \{\boldsymbol{x}_t\}_{t=1}^{T}$ and the target speaker's set $\mathbf{Y} \ni \{\boldsymbol{y}_t\}_{t=1}^{T}$.

3.1 GMM-based VC

3.1.1 Gaussian mixture model (GMM)

A Gaussian mixture model (GMM) is a statistical probabilistic model for representing the observed data that can be categorized into sub-components. Each component is represented as a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with parameters of a D-dimensional mean vector $\boldsymbol{\mu}$ and a variance matrix $\boldsymbol{\Sigma}$, which is defined as

$$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})\right), \tag{3.1}$$

where $\boldsymbol{x}$, $\boldsymbol{\mu}$, and $\boldsymbol{\Sigma}$ have the elements as follows:

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix}, \quad \boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_D \end{bmatrix}, \quad \boldsymbol{\Sigma} = \begin{bmatrix} \sigma_{11}^2 & \sigma_{12}^2 & \cdots & \sigma_{1D}^2 \\ \sigma_{21}^2 & \sigma_{22}^2 & & \vdots \\ \vdots & & \ddots & \vdots \\ \sigma_{D1}^2 & \cdots & \cdots & \sigma_{DD}^2 \end{bmatrix}, \tag{3.2}$$

and $(\cdot)^T$ and $|\cdot|$ indicate transpose and determinant of a matrix, respectively.

GMM represents the overall distribution of the data using weighted sum of the components. The probability density function (pdf) of GMM is defined as

$$p(\boldsymbol{x}|\boldsymbol{\Theta}) = \sum_{m=1}^{M} \alpha_m \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m), \tag{3.3}$$

where M indicates the number of mixtures. $\boldsymbol{\Theta}$ is a set of parameters of GMM, which contains α_m, $\boldsymbol{\mu}_m$, and $\boldsymbol{\Sigma}_m$ for all m.

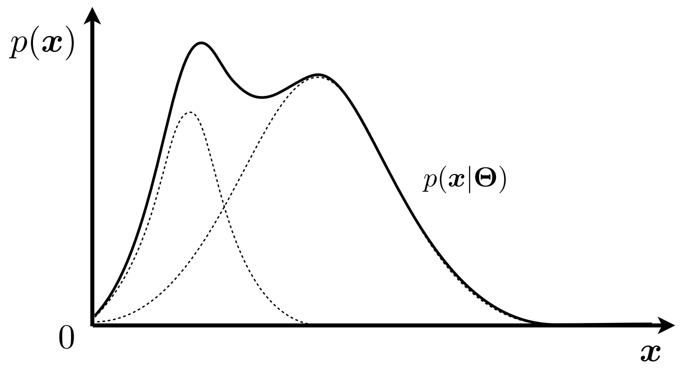


Figure 3.1: Example of a Gaussian mixture model ($M = 2$).

Figure 3.1 shows an example of a one-dimensional GMM that has 2 components with a solid line. The GMM was obtained by weighted sum of two Gaussian distributions depicted with the dotted lines.

The GMM parameters can be estimated using the expectation-maximization (EM) algorithm [50]. The algorithm repeats E-step (expectation) and M-step (maxmization) by turns. First, all parametes are randomly initialized. In the E-step, we calculate a Q-function (expectation of log-likelihood) defined as

$$\begin{aligned} Q(\hat{\boldsymbol{\Theta}}|\boldsymbol{\Theta}) &= E[\log p(\boldsymbol{x}, m|\hat{\boldsymbol{\Theta}})]_{p(m|\boldsymbol{x},\boldsymbol{\Theta})} \\ &= \sum_{m=1}^{M} p(m|\boldsymbol{x}, \boldsymbol{\Theta}) \log p(\boldsymbol{x}, m|\hat{\boldsymbol{\Theta}}), \end{aligned} \tag{3.4}$$

where

$$p(\boldsymbol{x}, m|\hat{\boldsymbol{\Theta}}) = \prod_{n=1}^{N} p(\boldsymbol{x}_n, m|\hat{\boldsymbol{\Theta}})$$
$$= \prod_{n=1}^{N} \hat{\alpha}_m \mathcal{N}(\boldsymbol{x}_n; \hat{\boldsymbol{\mu}}_m, \hat{\boldsymbol{\Sigma}}_m). \quad (3.5)$$

Therefore, Eq. (3.4) becomes

$$Q(\hat{\boldsymbol{\Theta}}|\boldsymbol{\Theta}) = \sum_k \sum_n p(m = k|\boldsymbol{x}_n, \boldsymbol{\Theta}) \log \hat{\alpha}_k$$
$$+ \sum_k \sum_n p(m = k|\boldsymbol{x}_n, \boldsymbol{\Theta}) \log \mathcal{N}(\boldsymbol{x}_n; \hat{\boldsymbol{\mu}}_k, \hat{\boldsymbol{\Sigma}}_k). \quad (3.6)$$

In the M-step, update rules for each parameter are derived so as to maximize the Q-function (Eq. (3.6)). The derived update rules are

$$\hat{\alpha}_k = \frac{1}{N} \sum_{n=1}^{N} p(k|\boldsymbol{x}_n, \boldsymbol{\Theta}), \quad (3.7)$$

$$\hat{\boldsymbol{\mu}}_k = \frac{\sum_n p(k|\boldsymbol{x}_n, \boldsymbol{\Theta}) \boldsymbol{x}_n}{\sum_n p(k|\boldsymbol{x}_n, \boldsymbol{\Theta})}, \quad (3.8)$$

$$\hat{\boldsymbol{\Sigma}}_k = \frac{\sum_n p(k|\boldsymbol{x}_n, \boldsymbol{\Theta})(\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_k)(\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_k)^T}{\sum_n p(k|\boldsymbol{x}_n, \boldsymbol{\Theta})}, \quad (3.9)$$

where $p(k|\boldsymbol{x}_n, \boldsymbol{\Theta})$ is a probability that $\boldsymbol{x}_n$ is sampled from the kth component, which is calculated by

$$p(k|\boldsymbol{x}_n, \boldsymbol{\Theta}) = \frac{\alpha_k \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_k \alpha_k \mathcal{N}(\boldsymbol{x}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}. \quad (3.10)$$

3.1.2 Methodology

When it comes to voice conversion based on GMM, we model a joint probability of a source speaker and a target speaker using GMM. Therefore, this model is called joint density GMM (JD-GMM). In the training stage of the JD-GMM, we use a joint vector $\boldsymbol{z}$ that concatenates a source speaker's vector $\boldsymbol{x}$ and a target speaker's vector $\boldsymbol{y}$ (i.e. $\boldsymbol{z} = [\boldsymbol{x}^T \boldsymbol{y}^T]^T$). The probability $p(\boldsymbol{z})$ is modeled using GMM as follows:

$$p(\boldsymbol{z}|\boldsymbol{\Theta}^{(z)}) = \sum_{m=1}^{M} \alpha_m \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}_m^{(z)}, \boldsymbol{\Sigma}_m^{(z)}), \quad (3.11)$$

where $\boldsymbol{\mu}_m^{(z)}$ and $\boldsymbol{\Sigma}_m^{(z)}$ consist of

$$\boldsymbol{\mu}_m^{(z)} = \begin{bmatrix} \boldsymbol{\mu}_m^{(x)} \\ \boldsymbol{\mu}_m^{(y)} \end{bmatrix}, \quad \boldsymbol{\Sigma}_m^{(z)} = \begin{bmatrix} \boldsymbol{\Sigma}_m^{(xx)} & \boldsymbol{\Sigma}_m^{(xy)} \\ \boldsymbol{\Sigma}_m^{(yx)} & \boldsymbol{\Sigma}_m^{(yy)} \end{bmatrix}. \tag{3.12}$$

The parameters $\boldsymbol{\mu}_m^{(x)}$ and $\boldsymbol{\Sigma}_m^{(xx)}$, and the parameters $\boldsymbol{\mu}_m^{(y)}$ and $\boldsymbol{\Sigma}_m^{(yy)}$ correspond to the source speaker's Gaussian distribution and the target speaker's Gaussian distribution, respectively. The parameter $\boldsymbol{\Sigma}_m^{(xy)}(= \boldsymbol{\Sigma}_m^{(yx)^T})$ indicates a covariance matrix between the observed data $\boldsymbol{x}$ and $\boldsymbol{y}$. In voice conversion, we usually use a diagonal matrix for $\boldsymbol{\Sigma}_m^{(xx)}$, $\boldsymbol{\Sigma}_m^{(xy)}$, and $\boldsymbol{\Sigma}_m^{(xx)}$ because full-covariance matrices bring a lot of parameters to estimate.

On the conversion stage (assuming that the parameters $\boldsymbol{\Theta}^{(z)}$ have already been estimated), we consider the probability of $\boldsymbol{y}$ given an input $\boldsymbol{x}$. That is

$$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\Theta}^{(z)}) = \sum_{m=1}^{M} p(m; \boldsymbol{x}, \boldsymbol{\Theta}^{(z)}) p(\boldsymbol{y}; \boldsymbol{x}, m, \boldsymbol{\Theta}^{(z)}). \tag{3.13}$$

The probabilities on the right side in Eq. (3.13) are represented as

$$\begin{aligned} p(m; \boldsymbol{x}, \boldsymbol{\Theta}^{(z)}) &= \frac{\alpha_m \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_m^{(x)}, \boldsymbol{\Sigma}_m^{(xx)})}{\sum_{m=1}^{M} \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_m^{(x)}, \boldsymbol{\Sigma}_m^{(xx)})} &(3.14)\\ p(\boldsymbol{y}; \boldsymbol{x}, m, \boldsymbol{\Theta}^{(z)}) &= \mathcal{N}(\boldsymbol{y}; \mathbf{E}_m^{(y|x)}, \mathbf{D}_m^{(y|x)}) &(3.15)\\ \mathbf{E}_m^{(y|x)} &= \boldsymbol{\mu}_m^{(y)} + \boldsymbol{\Sigma}_m^{(yx)} (\boldsymbol{\Sigma}_m^{(xx)})^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_m^{(x)}) &(3.16)\\ \mathbf{D}_m^{(y|x)} &= \boldsymbol{\Sigma}_m^{(yy)} - \boldsymbol{\Sigma}_m^{(yx)} (\boldsymbol{\Sigma}_m^{(xx)})^{-1} \boldsymbol{\Sigma}_m^{(xy)}. &(3.17) \end{aligned}$$

Using the probability in Eq. (3.13), we can estimate the most probable $\boldsymbol{y}$ as follows:

$$\begin{aligned} \hat{\boldsymbol{y}} &= \underset{\boldsymbol{y}}{\operatorname{argmax}}\, p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\Theta}^{(z)}) \\ &= \sum_{m=1}^{M} p(m; \boldsymbol{x}, \boldsymbol{\Theta}^{(z)})\, \underset{\boldsymbol{y}}{\operatorname{argmax}}\, p(\boldsymbol{y}; \boldsymbol{x}, m, \boldsymbol{\Theta}^{(z)}) \\ &= \sum_{m=1}^{M} p(m; \boldsymbol{x}, \boldsymbol{\Theta}^{(z)}) \mathbf{E}_m^{(y|x)}. &(3.18) \end{aligned}$$

As Eq. (3.18) shows, the conversion function using JD-GMM is based on the linear transformation (the weighted sum).

3.1.3 Problems

It is often reported that the GMM-based voice conversion includes over-smoothing problems and over-fitting problems. The over-smoothing arises because the parameters of multiple Gaussian components are estimated by averaging the observations with similar context descriptions. As a result, the outputs distribute near the modes of each component. The over-fitting problems come from its complexity. When we give more Gaussian components, the model is over-fitted to the training data.

3.2 NN-based VC

3.2.1 Neural network (NN)

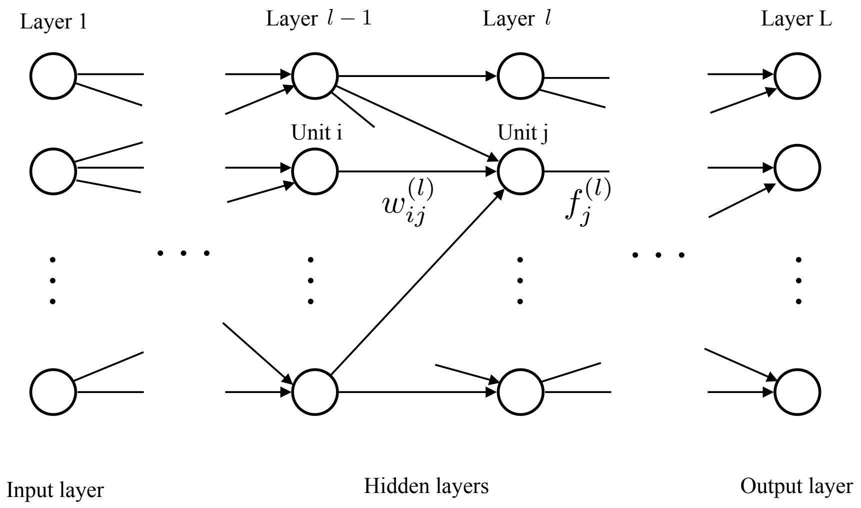


Figure 3.2: A feedforward neural network (NN).

A neural network (NN) is a mathematical model that imitates biological nervous neurons. Various types of NN have been proposed, and most of them aim to train and extract latent patterns in the given data. In this thesis, we focus on a multi-layered feedforward type of NN (it is called a multi-layer perceptron; MLP [51]) and refer this type as NN. This model is used for recognizing patterns, extracting rules, predicting sequence data, and mining data.

An NN that feeds D-dimensional inputs and produces M-dimensional outputs has a hierarchical structure that stacks multiple layers (L layers) as shown in Figure 3.2. The first layer, the last layer, and the middle layers are called an input layer, an output layer, and hidden layers, respectively.

Assume that the lth layer has L_l units. In the feedforward NN, each unit in the lth recieves information from the units in the previous layer ($l-1$th layer) and sends the output information to the units in the following layer ($l+1$th layer). The connection between two units has the unique weight. Let us denote $w_{ij}^{(l)}$ is the weight from the ith unit in the $l-1$th layer to the jth unit in the lth layer, where $2 < l \leq L,\ 1 \leq i \leq L_{l-1},\ 1 \leq j \leq L_l$. Given outputs of each unit in the $l-1$th layer $\boldsymbol{v}^{(l-1)} = [v_1^{(l-1)}, \cdots, v_{L_{l-1}}^{(l-1)}] \in \mathbb{R}^{L_{l-1}}$, the output value of the jth unit in the lth layer $v_j^{(l)}$ can be calculated as

$$\begin{aligned} v_j^{(l)} &= f(u_j^{(l)}) && (3.19)\\ &= f(\sum_{i=1}^{L_{l-1}} w_{ij}^{(l)} v_i^{(l-1)} + b_j^{(l)}), && (3.20)\end{aligned}$$

where $f(\cdot)$ denotes an activate function and $b_j^{(l)}$ indicates a bias parameter for $v_j^{(l)}$. There are a number of common activate functions such as an identity function (Eq. (3.21), Figure 3.3(a)), a step function (Eq. (3.22), Figure 3.3(b)), a sigmoid function (Eq. (3.23), Figure 3.3(c)), a hyperbolic function (Eq. (3.24), Figure 3.3(d)), and a softmax function (Eq. (3.25)), each of them defined as follows.

$$f(x) = x \tag{3.21}$$

$$f(x) = \begin{cases} 0 & (x < 0) \\ 1 & (x \geq 0) \end{cases} \tag{3.22}$$

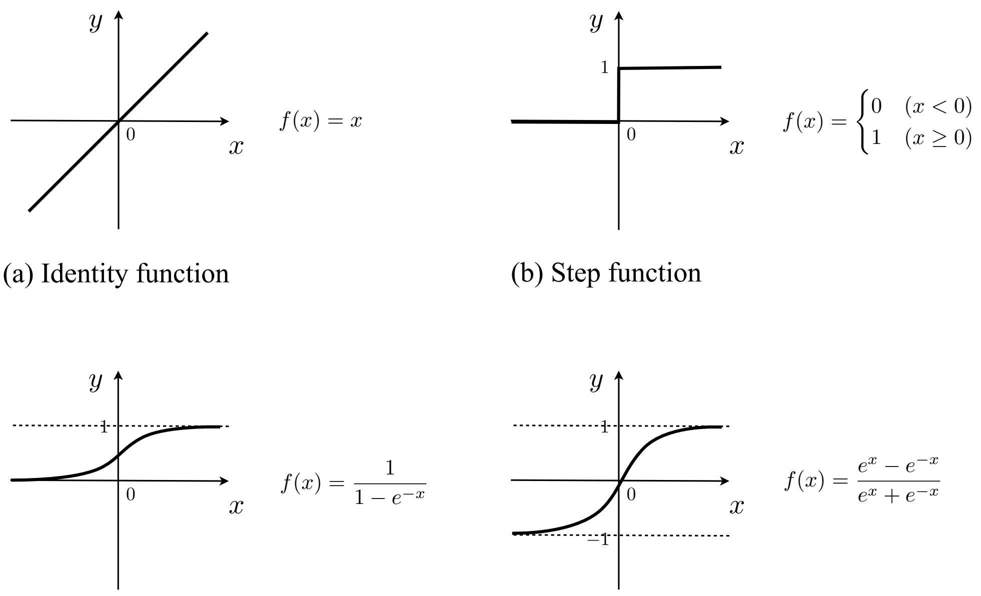
$$f(x) = \frac{1}{1 - e^{-x}} \tag{3.23}$$

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.24}$$

$$f(x) = \frac{e^x}{\sum_i e^{x_i}} \tag{3.25}$$

The most often-used activation function is a sigmoid function. One reason is that it is differentiable. This charastaristics is helpful on parameter estimation. Furthermore, it can limit the range of outputs from zero to one; it behaves like a step function. On the other hand, an identity function is often used for regression when we want real-valued outputs.

Sometimes it would be easier to understand when vector representation is used. The output vector in the lth layer $\boldsymbol{v}^{(l)} = [v_1^{(l)}, \cdots, v_{L_l}^{(l)}] \in \mathbb{R}^{L_l}$ is represented as in Eq. (3.26) rather than in Eq. (3.19).



(a) Identity function (b) Step function

(c) Sigmoid function (d) Hyperbolic function

Figure 3.3: Activate functions in use with an NN.

$$\boldsymbol{v}^{(l)} = f(\boldsymbol{u}^{(l)}) \tag{3.26}$$

$$= f(\boldsymbol{W}^{(l)}\boldsymbol{v}^{(l-1)} + \boldsymbol{b}^{(l)}) \tag{3.27}$$

Here, $\boldsymbol{W}^{(l)} = [w_{ij}^{(l)}] \in \mathbb{R}^{L_{l-1} \times L_l}$ and $\boldsymbol{b}^{(l)} \in \mathbb{R}^{L_l}$ are weight parameters and bias parameters, respectively. Now $f(\cdot)$ becomes an element-wise version of the activate functions defined before. For example, a sigmoid function is rewritten as

$$f(\boldsymbol{x}) = \mathcal{S}(\boldsymbol{x}) = \frac{1}{1 - e^{-\boldsymbol{x}}}. \tag{3.28}$$

For estimating the parameters of an NN ($\boldsymbol{W}^{(l)}$ and $\boldsymbol{b}^{(l)}$), backpropagation algorithm [52, 53] is often used. Given an N number of training data $\{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^{N}$, the algorithm tries to reduce the errors between the target $\boldsymbol{y}_n$ and the output of the network $\boldsymbol{v}_n^{(L)}$, which is defined as follows:

$$E = \sum_n e_n \tag{3.29}$$

$$= \frac{1}{2} \sum_n |\boldsymbol{y}_n - \boldsymbol{v}_n^{(L)}|^2. \tag{3.30}$$

In general, it is difficult to estimate the parameters that minimize the total error E globally. Typically, a steepest descent method is used for finding local minima. The method starts from random initialization for each element (w) in $\boldsymbol{W}$, and performs iterative updates as follows:

$$w \leftarrow w - \mu \frac{\partial E}{\partial w}, \tag{3.31}$$

where μ indicates a learning rate (typically, $0 < \mu \leq 1$) and determines the speed of convergence.

In backpropagation algorithm, the errors on the higher layer are propagated to the lower layer. Using a sigmoid activate function, the partial gradient to weights on the top layer $\frac{\partial E}{\partial w_{ij}^{(L)}}$, for instance, is calculated as

$$\frac{\partial E}{\partial w_{ij}^{(L)}} = \sum_n (y_{j_n} - v_{j_n}) \frac{\partial v_{j_n}^{(L)}}{\partial w_{ij}^{(L)}}. \tag{3.32}$$

From Eq. (3.19),

$$\begin{aligned} \frac{\partial v_j^{(L)}}{\partial w_{ij}^{(L)}} &= \frac{\partial v_j^{(L)}}{\partial u_j^{(L)}} \frac{\partial u_j^{(L)}}{\partial w_{ij}^{(L)}} && (3.33) \\ &= \mathcal{S}(u_j^{(L)})(1 - \mathcal{S}(u_j^{(L)}))v_i^{(L-1)}, && (3.34) \end{aligned}$$

hence, Eq. (3.32) becomes

$$\begin{aligned} \frac{\partial E}{\partial w_{ij}^{(L)}} &= \sum_n -(y_{j_n} - v_{j_n})\mathcal{S}(u_{j_n}^{(L)})(1 - \mathcal{S}(u_{j_n}^{(L)}))v_{in}^{(L-1)} && (3.35) \\ &= \sum_n \delta_{j_n}^{(L)} v_{in}^{(L-1)}, && (3.36) \end{aligned}$$

where $\delta_j^{(L)} = -(y_j - v_j)\mathcal{S}(u_j^{(L)})(1 - \mathcal{S}(u_j^{(L)}))$ behaves like errors and is backpropagated as discussed in the following.

For the weight parameters in the $L-1$th layer, the partial gradient can be calculated as

$$\begin{aligned} \frac{\partial E}{\partial w_{jk}^{(L-1)}} &= \sum_n \sum_{j'} \delta_{j'_n}^{(L)} \frac{\partial v_{j'_n}^{(L-1)}}{\partial w_{jk}^{(L-1)}} && (3.37) \\ &= \sum_n \sum_{j'} \delta_{j'_n}^{(L)} w_{j'k}^{(L-1)} \mathcal{S}(u_{kn}^{(L-1)})(1 - \mathcal{S}(u_{kn}^{(L-1)}))v_{j_n}^{(L-2)} && (3.38) \\ &= \sum_n \delta_{kn}^{(L-1)} v_{j_n}^{(L-2)}, && (3.39) \end{aligned}$$

where $\delta_k^{(L-1)} = (\sum_{j'} \delta_{j'}^{(L)} w_{j'k}^{(L-1)}) \mathcal{S}(u_k^{(L-1)})(1-\mathcal{S}(u_k^{(L-1)}))$, which is calculated using the errors propagated from the higher layer ($\delta_{j'}^{(L)}$).

To generalize, we can derive the following partial gradients for any layers.

$$\frac{\partial E}{\partial w_{ij}^{(l)}} = \sum_n \delta_{j_n}^{(l)} v_{i_n}^{(l-1)} \tag{3.40}$$

$$\delta_j^{(l)} = \begin{cases} -(y_j - v_j)\mathcal{S}(u_j^{(l)})(1 - \mathcal{S}(u_j^{(l)})) & (l = L) \\ (\sum_{j'} \delta_{j'}^{(l+1)} w_{ij'}^{(l)})\mathcal{S}(u_j^{(l)})(1 - \mathcal{S}(u_j^{(l)})) & (l = 2, \cdots, L-1) \end{cases} \tag{3.41}$$

3.2.2 Methodology

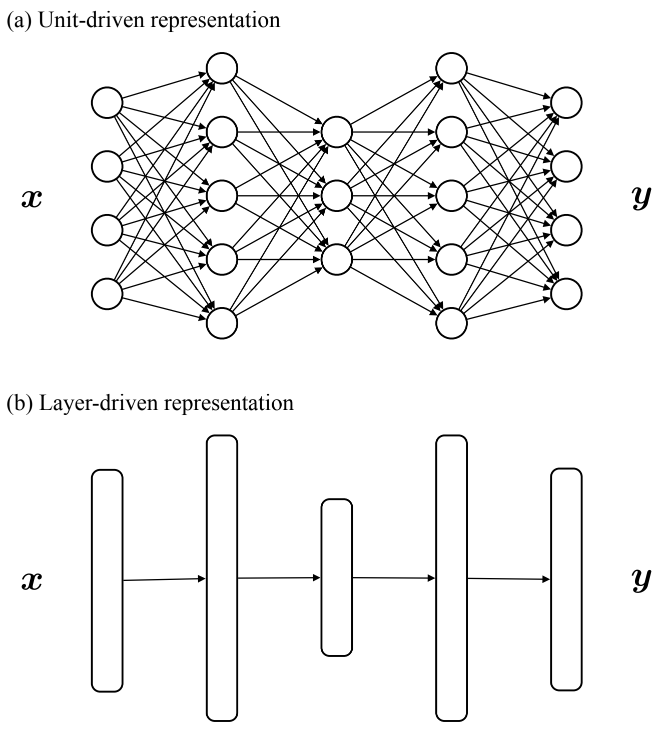


Figure 3.4: Voice conversion using a neural network.

It is straightforward to adapt an NN for voice conversion. A feedforward NN is used to obtain the mapping function from the source speaker's vector to the target speaker's vector [21, 54].

Figure 3.4 shows a five-layered NN that inputs the source vector $\boldsymbol{x}$ and outputs the target vector $\boldsymbol{y}$. Figure 3.4(b) simplifies the representation of Figure 3.4(a), and both of them show the same structure of the NN (the arrows in Figure 3.4(b) indicate full-connected relations). In general, the output $\boldsymbol{y}$ is a real-valued vector;

therefore, an identity function is used for the activate function in the highest layer of the NN. For the other layers (hidden layers), non-linear differentiable functions such as a sigmoid function or a hyperbolic function are used. When we choose a sigmoid activate function, the convert function using the five-layered NN forms

$$\boldsymbol{y} = \boldsymbol{W}^{(4)}\mathcal{S}(\boldsymbol{W}^{(3)}\mathcal{S}(\boldsymbol{W}^{(2)}\mathcal{S}(\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}) + \boldsymbol{b}^{(2)}) + \boldsymbol{b}^{(3)}) + \boldsymbol{b}^{(4)}, \tag{3.42}$$

where $\{\boldsymbol{W}^{(l)}, \boldsymbol{b}^{(l)}\}$ are the parameters of the lth layer. The parameters can be estimated using backpropagation algorithm discussed in the previous section.

To simplify the notation in Eq. 3.42, we define a symbol $\bigodot_{n=1}^{N}$ that denotes composition of N functions. Using the symbol, Eq. 3.42 can be rewritten as

$$\boldsymbol{y} = \boldsymbol{W}^{(4)}(\bigodot_{l=1}^{3}\mathcal{S}(\boldsymbol{W}^{(l)}\boldsymbol{x} + \boldsymbol{b}^{(l)})) + \boldsymbol{b}^{(4)}. \tag{3.43}$$

Now we derive a general formulation for voice conversion using an NN with L layers as follows:

$$\boldsymbol{y} = \bigodot_{l=1}^{L-1} f^{(l)}(\boldsymbol{W}^{(l)}\boldsymbol{x} + \boldsymbol{b}^{(l)}), \tag{3.44}$$

where $f^{(l)}(\cdot)$ denotes an arbitrary element-wise activate function for the lth layer.

3.2.3 Problems

In the similar manner to the problems on GMM-based voice conversion, the NN-based voice conversion is often suffered from over-fitting. If the amount of training data is not enough to the number of parameters, the trained NN is influenced by small fluctuations in the unknown data. Furthermore, over-smoothing problems also occur due to the forced-alignment using dynamic time warping. This is easy to imagine when we consider that after forced-alignment, we have different target vectors to the same source vector. As a result, the parameters are estimated so that the output of the network gets close to the average vector of them (the over-smoothed vector).

The most significant problem of NN-based voice conversion is that it is difficult to estimate the weight parameters correctly when the network has a number of hidden layers (a deep architecture). As Eqs. (3.40)(3.41) indicate, the errors at the higher layer are propagated into the lower layers. Specifically, the lowest layer accumulates all the errors propagated from the following layer, which also conveys the erros from the next layer, etc. Therefore, they may reach poor local minima that are close to the randomly-initialized values and far from the global minima.

3.3 NMF-based VC

3.3.1 Non-negative matrix factorization (NMF)

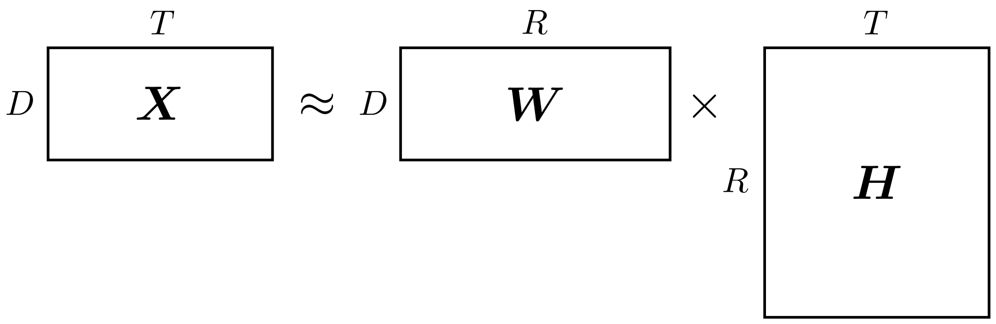


Figure 3.5: Non-negative matrix factorization.

Non-negative matrix factorization (NMF) [55] is a matrix decomposition method under non-negative constranints. In audio signal processing, NMF is widly used for source separation [56, 57], speech enhancement [58, 59], speech recognition [60], automatic music transcription [61, 62, 63, 64], and voice conversion [28, 29, 65, 66, 67, 68]. Common use of NMF in these approaches is that the observed signal is represented as a linear combination of atoms (also called exemplars or dictionaries), which are trained beforehand. Thanks to the non-negative constraints, a small number of atoms are selected and combined. Therefore, NMF can also be seen as a tool of sparse coding [69, 70], though there exist NMF algorithms that contains explicit sparse constraints [71, 72, 73].

The basic idea to decompose a matrix $\boldsymbol{X} \in \mathbb{R}^{D\times T}$ is to find two matrices $\boldsymbol{W} \in \mathbb{R}^{D\times R}$ and $\boldsymbol{H} \in \mathbb{R}^{R\times T}$ that minimize the distance between $\boldsymbol{X}$ and $\boldsymbol{WH}$ under non-negative constraints. Figure 3.5 depicts the decomposition process of NMF. In NMF, $\boldsymbol{W}$ is called a basis matrix, and contains R number of atoms (dictionaries, or bases) in columns. $\boldsymbol{H}$ is called an activity matrix, and indicates the activity of each atom in time. The decomposition process is formulated as

$$\boldsymbol{W}, \boldsymbol{H} = \operatorname*{argmin}_{\boldsymbol{W},\boldsymbol{H}} D(\boldsymbol{X}|\boldsymbol{WH}) \tag{3.45}$$

$$s.t.\ \boldsymbol{W}, \boldsymbol{H} \geq 0, \tag{3.46}$$

where $\cdot \geq 0$ indicates that all elements in the matrix are greater than or equal to zero. $D(\boldsymbol{A}|\boldsymbol{B})$ denotes the distance between the two matrices; typically, the Euclidian distance $D_{EUC}(\boldsymbol{A}|\boldsymbol{B})$ or the Kullback-Leibler (KL) divergence $D_{KL}(\boldsymbol{A}|\boldsymbol{B})$

is used. They are, respectively, defined as

$$D_{EUC}(\boldsymbol{A}|\boldsymbol{B}) = ||\boldsymbol{A} - \boldsymbol{B}||^2 = \sum_{ij}(A_{ij} - B_{ij})^2 \tag{3.47}$$

$$D_{KL}(\boldsymbol{A}|\boldsymbol{B}) = \sum_{ij} A_{ij} \log \frac{A_{ij}}{B_{ij}} - A_{ij} + B_{ij}, \tag{3.48}$$

where A_{ij} denotes the ith-row jth-column element in the matrix $\boldsymbol{A}$.

Now let us consider the case where the Euclidian distance is used. Using Eq. (3.47), the objective function of NMF becomes

$$\begin{aligned} D_{EUC}(\boldsymbol{X}|\boldsymbol{WH}) &= \sum_{dt}(X_{dt} - (\boldsymbol{WH})_{dt})^2 \\ &= \sum_{dt}(X_{dt}^2 - 2X_{dt}(\boldsymbol{WH})_{dt} + (\boldsymbol{WH})_{dt}^2) \\ &= \sum_{dt}(X_{dt}^2 - 2X_{dt}\sum_r W_{dr}H_{rt} + (\sum_r W_{dr}H_{rt})^2) \end{aligned} \tag{3.49}$$

The third term in Eq. (3.49) $(\sum_r W_{dr}H_{rt})^2$ is difficult to derive partial differential. Therefore, an auxiliary function G_{EUC} that satisfies $G_{EUC}(\boldsymbol{W}, \boldsymbol{H}, \boldsymbol{\lambda}) \geq D_{EUC}(\boldsymbol{X}|\boldsymbol{WH})$ is used, which is defined as follows

$$G_{EUC}(\boldsymbol{W}, \boldsymbol{H}, \boldsymbol{\lambda}) \triangleq \sum_{dt}(X_{dt}^2 - 2X_{dt}\sum_r W_{dr}H_{rt} + \sum_r \frac{W_{dr}^2 H_{rt}^2}{\lambda_{drt}}) \tag{3.50}$$

$$s.t.\ \forall\lambda_{drt} \geq 0,\ \sum_{d,r,t}\lambda_{drt} = 1. \tag{3.51}$$

Applying Lagrange multipliers, we obtain the following update rules:

$$\lambda_{drt} \leftarrow \operatorname*{argmin}_{\lambda_{drt}} G_{EUC} = \frac{W_{dr}H_{rt}}{\sum_{r'} W_{dr'}H_{r't}} \tag{3.52}$$

$$W_{dr} \leftarrow \operatorname*{argmin}_{W_{dr}} G_{EUC} = \frac{\sum_t X_{dt}H_{rt}}{\sum_t \frac{H_{rt}^2}{\lambda_{drt}}} \tag{3.53}$$

$$H_{rt} \leftarrow \operatorname*{argmin}_{H_{rt}} G_{EUC} = \frac{\sum_d X_{dt}W_{dr}}{\sum_d \frac{W_{dr}^2}{\lambda_{drt}}}. \tag{3.54}$$

Replacing λ_{drt} in Eqs. (3.53)(3.54) with Eq. (3.52), we finally obtain the common update rules for Euclidian-distance-based NMF as follows:

$$W_{dr} \leftarrow W_{dr}\frac{\sum_t X_{dt}H_{rt}}{\sum_t H_{rt}\sum_{r'} W_{dr'}H_{r't}} \tag{3.55}$$

$$H_{rt} \leftarrow H_{rt}\frac{\sum_d X_{dt}W_{dr}}{\sum_d W_{dr}\sum_{r'} W_{dr'}H_{r't}} \tag{3.56}$$

Similarly, we can derive update rules for KL-based NMF using an auxiliary function G_{KL}, that is

$$\begin{aligned} & G_{KL}(\boldsymbol{W}, \boldsymbol{H}, \boldsymbol{\lambda}) \\ & \triangleq \sum_{dt}(X_{dt} \log X_{dt} - X_{dt} \sum_{r} \lambda_{drt} \log \frac{W_{dr}H_{rt}}{\lambda_{drt}} - X_{dt} + \sum_{r} W_{dr}H_{rt}) \qquad (3.57) \\ & s.t.\ \forall \lambda_{drt} \geq 0,\ \sum_{d,r,t} \lambda_{drt} = 1. \qquad (3.58) \end{aligned}$$

The derived update rules for the KL-based NMF are:

$$W_{dr} \leftarrow W_{dr} \frac{\sum_t \frac{X_{dt}H_{rt}}{\sum_{r'} W_{dr'}H_{r't}}}{\sum_t H_{rt}} \qquad (3.59)$$

$$H_{rt} \leftarrow H_{rt} \frac{\sum_d \frac{X_{dt}W_{dr}}{\sum_{r'} W_{dr'}H_{r't}}}{\sum_d W_{dr}}. \qquad (3.60)$$

Sometimes the update rules in matrix-wise representation is useful and easy to see. They are:

$$\boldsymbol{W} \leftarrow \boldsymbol{W} \odot \frac{\frac{\boldsymbol{X}}{\boldsymbol{WH}}\boldsymbol{H}^{\mathrm{T}}}{\mathbf{1}^{D\times T}\boldsymbol{H}^{\mathrm{T}}} \qquad (3.61)$$

$$\boldsymbol{H} \leftarrow \boldsymbol{H} \odot \frac{\boldsymbol{W}^{\mathrm{T}}\frac{\boldsymbol{X}}{\boldsymbol{WH}}}{\boldsymbol{W}^{\mathrm{T}}\mathbf{1}^{D\times T}}, \qquad (3.62)$$

where $\odot$ and $\dot{-}$ denote element-wise multiplication and division, respectively, and $\mathbf{1}^{N\times M}$ indicates a matrix with the size of $N \times M$ whose elements are all one.

3.3.2 Exemplar-based approach

Figure 3.6 shows the system flow of voice conversion using NMF [28, 65, 66]. The symbols in Figure 3.6: $\boldsymbol{X}(\in \mathbb{R}^{D\times T})$, $\boldsymbol{Y}(\in \mathbb{R}^{D\times T})$, $\boldsymbol{W}^s(\in \mathbb{R}^{D\times R})$, $\boldsymbol{W}^t(\in \mathbb{R}^{D\times R})$, $\boldsymbol{H}^s(\in \mathbb{R}^{R\times T})$, and $\boldsymbol{H}^t(\in \mathbb{R}^{R\times T})$ indicate a source feature matrix (spectrogram), a target spectrogram, source dictionaries, target dictionaries, a source activity matrix, and a target activity matrix, respectively. A voice conversion system using exemplar-based NMF simply uses raw parallel (aligned) training data for dictionaries $\{\boldsymbol{W}^s, \boldsymbol{W}^t\}$. Given a source spectrogram $\boldsymbol{X}$, the process of voice conversion begins with obtaining an activity matrix $\boldsymbol{H}^s$ using the source dictionaries $\boldsymbol{W}^s$. Because a large number of dictionaries are used ($R \gg T$) in the common exemplar-based approaches, it is better to make the activity matrix sparse. Therefore, sparse NMF (SNMF [72, 73]) is used for estimating an activity matrix in voice converion, rather than using Eq. (3.62). Using SNMF, the

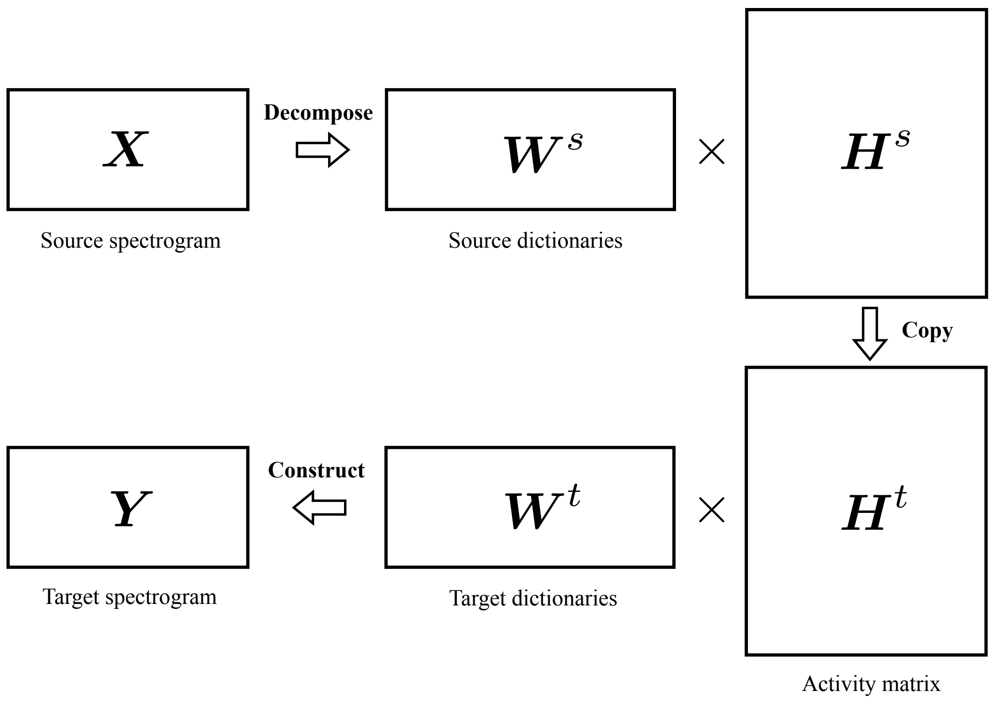


Figure 3.6: Flow of voice conversion based on NMF.

activity matrix $\boldsymbol{H}^s$ is obtained so as to minimize the following cost function:

$$D(\boldsymbol{X}|\boldsymbol{W}^s\boldsymbol{H}^s) + \kappa||\boldsymbol{H}^s||_1 \quad s.t.\ \boldsymbol{H}^s \geq 0, \tag{3.63}$$

where $||\cdot||_1$ indicates $l1$ norm, and $\kappa(\geq 0)$ determines the sparsity. When using KL-divergence for $D(\boldsymbol{X}|\boldsymbol{W}^s\boldsymbol{H}^s)$, we can derive the similar update rule to Eq. (3.62) as follows:

$$\boldsymbol{H}^s \leftarrow \boldsymbol{H}^s \odot \frac{\boldsymbol{W}^{s\mathrm{T}}\frac{\boldsymbol{X}}{\boldsymbol{W}^s\boldsymbol{H}^s}}{\boldsymbol{W}^{s\mathrm{T}}\boldsymbol{1}^{D\times T} + \kappa\boldsymbol{1}^{R\times T}}. \tag{3.64}$$

Iteratively updating $\boldsymbol{H}^s$ using Eq. (3.64) until it converges (remaining the dictionaries $\boldsymbol{W}^s$ unchanged), we obtain the optimum activity matrix $\hat{\boldsymbol{H}}^s$. In the exemplar-based approach, the obtained matrix $\hat{\boldsymbol{H}}^s$ is used as an target speakers's activity matrix without any change; i.e., $\boldsymbol{H}^t = \hat{\boldsymbol{H}}^s$. Finally, we obtain the target features (spectrogram) by multiplying the target dictionaries and the activity matrix as follows:

$$\boldsymbol{Y} = \boldsymbol{W}^t\hat{\boldsymbol{H}}^s. \tag{3.65}$$

The interesting point of voice conversion using the exemplar-based NMF is that it assumes the activity matrix obtained from the source features must be the same as the activity matrix for the target features (the two activity matrices are different from each other actually, though). The idea comes from the fact that the two matrices share with the same prosodic information via parallel dictionaries.

3.3.3 Spectral-mapping approach

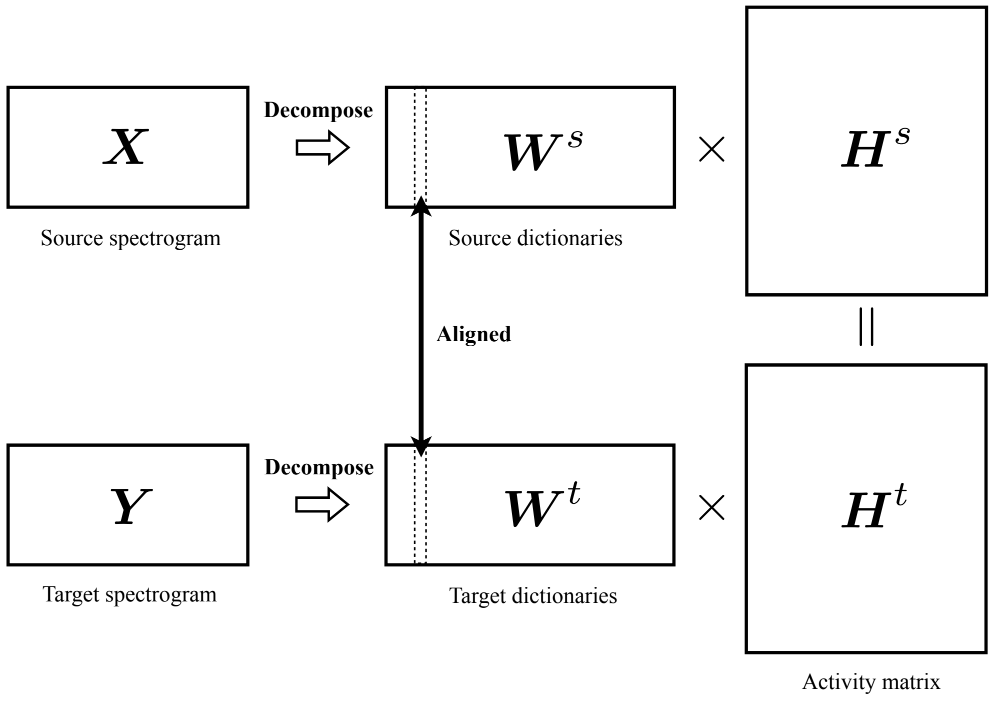


Figure 3.7: Estimating parallel dictionaries in the spectral-mapping-based NMF framework.

In the exemplar-based NMF approach, parallel dictionaries $\{\boldsymbol{W}^s, \boldsymbol{W}^t\}$ were picked from the aligned training data without any change. Here, in the spectral-mapping-based approach [67, 68], the dictionaries are obtained from the training based on iterative updates. Figure 3.7 shows the training process of the parallel dictionaries.

This approach assumes that the activity matrix of the source speaker $\boldsymbol{H}^s$ is completely equivalent to the activity matrix of the target speaker $\boldsymbol{H}^t$ ($\boldsymbol{H}^s = \boldsymbol{H}^t = \boldsymbol{H}$). And then, given parallel training data $\{\boldsymbol{X}, \boldsymbol{Y}\}$, we estimate and train the dictionaries $\{\boldsymbol{W}^s, \boldsymbol{W}^t\}$ with updating $\boldsymbol{H}$ as well. The dictionaries and the activity matrix are estimated so that they minimize the cost function defined as

$$D(\boldsymbol{X}|\boldsymbol{W}^s\boldsymbol{H}) + D(\boldsymbol{Y}|\boldsymbol{W}^t\boldsymbol{H}) + \kappa||\boldsymbol{H}^s||_1 \;\; s.t.\; \boldsymbol{W}^s, \boldsymbol{W}^t, \boldsymbol{H} \geq 0. \tag{3.66}$$

Again, κ is a sparsity term and experimentally determined. If we use KL-divergence for $D(\boldsymbol{X}|\boldsymbol{W}^s\boldsymbol{H})$ and $D(\boldsymbol{Y}|\boldsymbol{W}^t\boldsymbol{H})$, we obtain

$$\begin{aligned}
&D_{KL}(\boldsymbol{X}|\boldsymbol{W}^s\boldsymbol{H}) + D_{KL}(\boldsymbol{Y}|\boldsymbol{W}^t\boldsymbol{H}) \\
&= \sum_{d=1}^{D}\sum_{t=1}^{T}\left(X_{dt}\log\frac{X_{dt}}{(\boldsymbol{W}^s\boldsymbol{H})_{dt}} - X_{dt} + (\boldsymbol{W}^s\boldsymbol{H})_{dt}\right) \\
&\quad + \sum_{d=1}^{D}\sum_{t=1}^{T}\left(Y_{dt}\log\frac{Y_{dt}}{(\boldsymbol{W}^t\boldsymbol{H})_{dt}} - Y_{dt} + (\boldsymbol{W}^t\boldsymbol{H})_{dt}\right) \\
&= \sum_{d=1}^{2D}\sum_{t=1}^{T}\left(\begin{bmatrix}\boldsymbol{X}\\ \boldsymbol{Y}\end{bmatrix}_{dt}\log\frac{\begin{bmatrix}\boldsymbol{X}\\ \boldsymbol{Y}\end{bmatrix}_{dt}}{\begin{bmatrix}\boldsymbol{W}^s\boldsymbol{H}\\ \boldsymbol{W}^t\boldsymbol{H}\end{bmatrix}_{dt}} - \begin{bmatrix}\boldsymbol{X}\\ \boldsymbol{Y}\end{bmatrix}_{dt} + \begin{bmatrix}\boldsymbol{W}^s\boldsymbol{H}\\ \boldsymbol{W}^t\boldsymbol{H}\end{bmatrix}_{dt}\right) \\
&= \sum_{d=1}^{2D}\sum_{t=1}^{T}\left(\begin{bmatrix}\boldsymbol{X}\\ \boldsymbol{Y}\end{bmatrix}_{dt}\log\frac{\begin{bmatrix}\boldsymbol{X}\\ \boldsymbol{Y}\end{bmatrix}_{dt}}{\left(\begin{bmatrix}\boldsymbol{W}^s\\ \boldsymbol{W}^t\end{bmatrix}\boldsymbol{H}\right)_{dt}} - \begin{bmatrix}\boldsymbol{X}\\ \boldsymbol{Y}\end{bmatrix}_{dt} + \left(\begin{bmatrix}\boldsymbol{W}^s\\ \boldsymbol{W}^t\end{bmatrix}\boldsymbol{H}\right)_{dt}\right) \\
&= \sum_{d=1}^{2D}\sum_{t=1}^{T}\left(Z_{dt}\log\frac{Z_{dt}}{(\boldsymbol{W}\boldsymbol{H})_{dt}} - Z_{dt} + (\boldsymbol{W}\boldsymbol{H})_{dt}\right), \qquad (3.67)
\end{aligned}$$

where $\boldsymbol{Z} = \begin{bmatrix}\boldsymbol{X}\\ \boldsymbol{Y}\end{bmatrix}$ and $\boldsymbol{W} = \begin{bmatrix}\boldsymbol{W}^s\\ \boldsymbol{W}^t\end{bmatrix}$ are parallel training data and parallel dictionaries, respectively. As Eq. (3.67) indicates, the optimization problem in Eq. (3.66) results in finding the optimum element of $\boldsymbol{W}$ and $\boldsymbol{H}$ given a matrix $\boldsymbol{Z}$. Therefore, using Eqs. (3.61)(3.64), iterative update rules for $\boldsymbol{W}$ ($\boldsymbol{W}^s$ and $\boldsymbol{W}^t$) and $\boldsymbol{H}$ become:

$$\boldsymbol{W}^s \leftarrow \boldsymbol{W}^s \odot \frac{\frac{\boldsymbol{X}}{\boldsymbol{W}^s\boldsymbol{H}}\boldsymbol{H}^{\mathrm{T}}}{\boldsymbol{1}^{D\times T}\boldsymbol{H}^{\mathrm{T}}} \qquad (3.68)$$

$$\boldsymbol{W}^t \leftarrow \boldsymbol{W}^t \odot \frac{\frac{\boldsymbol{Y}}{\boldsymbol{W}^t\boldsymbol{H}}\boldsymbol{H}^{\mathrm{T}}}{\boldsymbol{1}^{D\times T}\boldsymbol{H}^{\mathrm{T}}} \qquad (3.69)$$

$$\begin{aligned}
\boldsymbol{H} &\leftarrow \boldsymbol{H} \odot \frac{\boldsymbol{W}^{\mathrm{T}}\frac{\boldsymbol{Z}}{\boldsymbol{W}\boldsymbol{H}}}{\boldsymbol{W}^{\mathrm{T}}\boldsymbol{1}^{2D\times T} + \kappa\boldsymbol{1}^{R\times T}} \qquad (3.70) \\
&= \boldsymbol{H} \odot \frac{\boldsymbol{W}^{s\mathrm{T}}\frac{\boldsymbol{X}}{\boldsymbol{W}^s\boldsymbol{H}} + \boldsymbol{W}^{t\mathrm{T}}\frac{\boldsymbol{Y}}{\boldsymbol{W}^t\boldsymbol{H}}}{\boldsymbol{W}^{s\mathrm{T}}\boldsymbol{1}^{D\times T} + \boldsymbol{W}^{t\mathrm{T}}\boldsymbol{1}^{D\times T} + \kappa\boldsymbol{1}^{R\times T}}. \qquad (3.71)
\end{aligned}$$

The conversion process is the same as that of the exemplar-based approach. Given the test data (a source spectrogram), we first compute an activity matrix using the source dictionaries, and then construct a target spectrogram using the target dictionaries.

3.3.4 Problems

The NMF-based voice conversion in the exemplar-based approach does not use any statistical training. Therefore, an over-smoothing problem does not occur unlike GMM-based or NN-based approaches. However, this approach tends to be time-consuming and difficult to execute in real-time, because thousands of dictionaries are used in the conversion process. On the other hand, a spectral-mapping-based approach uses much less dictionaries in conversion. In this approach, however, over-smoothing arises due to the stastical training.

The other serious problem is that NMF-based approaches can not treat with negative-valued data. For instance, MFCCs, which allows negative values, can not be fed as input features in these approaches. In [28, 29, 65, 66, 67, 68], they used magunitude (power) spectum or STRAIGHT spectrum for the acoustic features.

Chapter 4

VC Using a Joint Density RBM

In this chapter, a voice conversion method using a joint density restricted Boltzmann machine (JD-RBM) is described. The method is viewed as a sparse-representation-based approach, which is related to the NMF-based voice conversion discussed in Chapter 3.

4.1 Sparse-representation-based VC

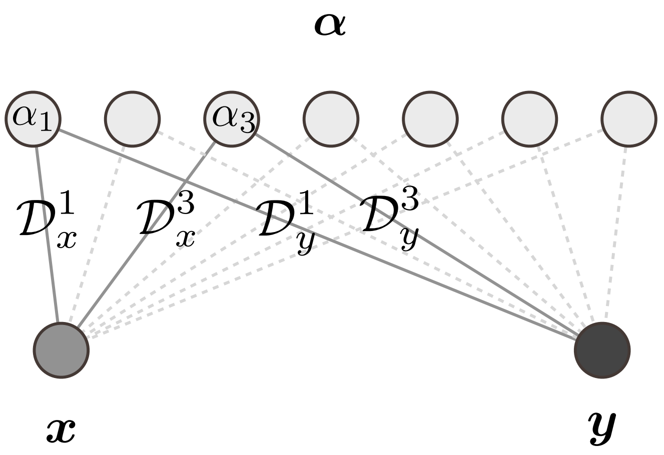


Figure 4.1: Voice conversion based on sparse representation.

In voice conversion, the system converts an acoustic vector of the source speaker $\boldsymbol{x} \in \mathbb{R}^D$ into the target speaker's vector $\boldsymbol{y} \in \mathbb{R}^D$. In sparse-representation-based voice conversion, the target vector $\boldsymbol{y}$ is obtained using a sparse vector $\boldsymbol{\alpha} \in \mathbb{R}^K, \|\boldsymbol{\alpha}\|_0 \ll K$ that is calculated from the source vector $\boldsymbol{x}$, instead of directly estimating the target vector from the source vector.

In this approach, K pairs of the source speaker's dictionaries $\mathcal{D}_x \in \mathbb{R}^{D \times K}$ and the target speaker's dictionaries $\mathcal{D}_y \in \mathbb{R}^{D \times K}$ (parallel dictionaries) are trained beforehand. Given an input vector of the source speaker, the converted target speaker's vector $\boldsymbol{y}$ is obtained using the trained parallel dictionaries as shown in Figure 4.1. In the conversion stage, we first calculate a sparse vector $\boldsymbol{\alpha}$ that satisfies

$$\boldsymbol{x} \approx f(\mathcal{D}_x \boldsymbol{\alpha}), \tag{4.1}$$

and then we obtain the target speaker's vector as follows:

$$\boldsymbol{y} \approx f(\mathcal{D}_y \boldsymbol{\alpha}), \tag{4.2}$$

where $f(\cdot)$ indicates an arbitrary function.

Most sparse-representation-based approaches use training exemplars without changes for the parallel dictionaries $\mathcal{D}_x, \mathcal{D}_y$ [28, 74, 75]. For the calculation of the sparse vector $\boldsymbol{\alpha}$, there are various approaches that can be used, such as L1 normalization [74], K-nearest neighbors algorithm [75], and sparse non-negative matrix factorization (NMF) [28, 29, 65, 66]. As discussed in Chapter 3, another approach based on sparse NMF has also been proposed that uses *trained* parallel dictionaries so that the sparse vectors for the source and the target are the same [67, 68]. The above-mentioned sparse-representation-based voice conversion methods are based on linear function ($f(\cdot)$ are linear functions in Eqs. (4.1)(4.2)).

4.2 Restricted Boltzmann Machine (RBM)

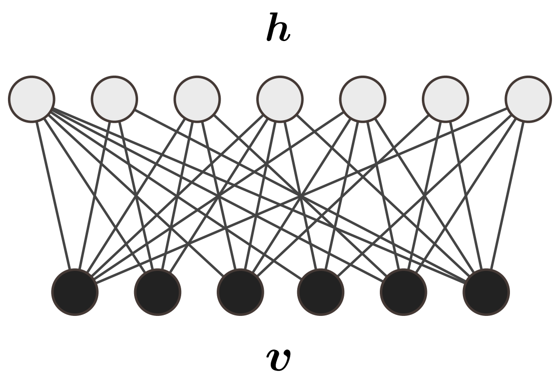


Figure 4.2: Graphical representation of an RBM.

A restricted Boltzmann machine (RBM) is an undirected graphical model that defines the distribution of visible variables with binary hidden (latent) variables as shown in Figure 4.2. There are two layers: a visible layer and a hidden layer, and each unit is fully connected to each other, except for the units in the same layer[1]. An RBM was originally introduced as a method of representing binary-valued data (Bernoulli-Bernoulli RBM; BB-RBM) [31, 78], and it later came to be used to deal with real-valued data (such as acoustic features) known as a Gaussian-Bernoulli RBM (GB-RBM) [79]. In this section, we first introduce the basic BB-RBM, and then explain the GB-RBM, which is more applicable to the real applications.

4.2.1 BB-RBM

4.2.1.1 Principle

BB-RBMs treat with binary inputs and hidden states. The joint probability of visible states $\boldsymbol{v} = [v_1, \cdots, v_I]^{\mathrm{T}}, v_i \in \{0,1\}$ and hidden states $\boldsymbol{h} = [h_1, \cdots, h_I]^{\mathrm{T}}, v_i \in \{0,1\}$ is defined with an energy function E_{BB} as follows:

$$p(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\Theta}) = \frac{1}{Z(\boldsymbol{\Theta})} e^{-E_{BB}(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\Theta})} \tag{4.3}$$

$$E_{BB}(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\Theta}) = -\boldsymbol{b}^{\mathrm{T}}\boldsymbol{v} - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{h} - \boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{h}, \tag{4.4}$$

where $Z(\boldsymbol{\Theta}) = \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E_{BB}(\boldsymbol{v}, \boldsymbol{h}; \boldsymbol{\Theta})}$ is the normalization constant. The parameters of the BB-RBM ($\boldsymbol{\Theta}$) include $\boldsymbol{b} \in \mathbb{R}^I$, $\boldsymbol{c} \in \mathbb{R}^J$, and $\boldsymbol{W} \in \mathbb{R}^{I \times J}$, which denote the bias vector of the visible units, the bias vector of the hidden units, and the weight parameters, respectively. The weight $W_{ij} \ni \boldsymbol{W}$ indicates the strength of the connection between the ith visible unit and the jth hidden unit.

The most helpful characteristic of RBMs, which has no connections between units in the same layer, is that the integral over all possible states of each layer can be representation as a product of one-dimensional integrals. This enables us to calculate the conditional probabilities $p(\boldsymbol{v}|\boldsymbol{h})$ and $p(\boldsymbol{v}|\boldsymbol{h})$ easily. That is:

$$p(\boldsymbol{h}|\boldsymbol{v}) = \prod_j p(h_j|\boldsymbol{v}) \tag{4.5}$$

and

$$p(\boldsymbol{v}|\boldsymbol{h}) = \prod_i p(v_i|\boldsymbol{h}). \tag{4.6}$$

Each conditional probability $p(h_j|\boldsymbol{v})$ and $p(v_i|\boldsymbol{h})$ forms Bernoulli distribution as follows:

$$p(h_j|\boldsymbol{v}) = \mathcal{B}(h_j; \mathcal{S}(c_j + \boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j})) \tag{4.7}$$

[1]The model that allows the connection between the units in the same layer has also been proposed, named Boltzmann machine (BM) [76, 77].

and

$$p(v_i|\boldsymbol{h}) = \mathcal{B}(v_i; \mathcal{S}(b_i + \boldsymbol{W}_{i:}\boldsymbol{h})), \tag{4.8}$$

respectively, where $\mathcal{B}(\cdot; p)$ denotes the Bernoulli distribution with success probability p. $\boldsymbol{W}_{i:}$ and $\boldsymbol{W}_{:j}$ denote the ith row vector and the jth column vector of the matrix $\boldsymbol{W}$, respectively. For more details on Eqs. (4.5)(4.6)(4.7)(4.8), see Appendix.

It would be useful to define multivariant Bernoulli distribution (independent variables, though). Using notation $\mathcal{B}(\boldsymbol{x}; \boldsymbol{p})$ as multivariant Bernoulli distribution with the success probabilities $\boldsymbol{p} \in \mathbb{R}^D$, each of which corresponds to the variable in $\boldsymbol{x} \in \mathbb{R}^D$, we represent the conditional probability distribution as follows:

$$\begin{align} p(\boldsymbol{h}|\boldsymbol{v}) &= \mathcal{B}(\boldsymbol{h}; \mathcal{S}(\boldsymbol{c} + \boldsymbol{W}^{\mathrm{T}}\boldsymbol{v})) \tag{4.9} \\ p(\boldsymbol{v}|\boldsymbol{h}) &= \mathcal{B}(\boldsymbol{v}; \mathcal{S}(\boldsymbol{b} + \boldsymbol{W}\boldsymbol{h})). \tag{4.10} \end{align}$$

As shown above, it is fairly easy to compute the expectation value $\mathbb{E}\{\boldsymbol{h}|\boldsymbol{v}\}$ and the conditionaly probablity that the hidden variables all takes values of one given visible units $p(\boldsymbol{h} = \boldsymbol{1}|\boldsymbol{v})$ as follows:

$$\begin{align} \mathbb{E}\{\boldsymbol{h}|\boldsymbol{v}\} &= \prod_j E\{h_j|\boldsymbol{v}\} \tag{4.11} \\ &= \prod_j 0 \cdot p(h_j = 0|\boldsymbol{v}) + 1 \cdot p(h_j = 1|\boldsymbol{v}) \tag{4.12} \\ &= \prod_j p(h_j = 1|\boldsymbol{v}) \tag{4.13} \\ &= p(\boldsymbol{h} = \boldsymbol{1}|\boldsymbol{v}) \tag{4.14} \\ &= \mathcal{S}(\boldsymbol{c} + \boldsymbol{W}^{\mathrm{T}}\boldsymbol{v}). \tag{4.15} \end{align}$$

Similarly, the conditional expectation value of $\boldsymbol{v}$ and probability that all visible units set to one given hidden units become simple equations:

$$\begin{align} \mathbb{E}\{\boldsymbol{v}|\boldsymbol{h}\} &= p(\boldsymbol{v} = \boldsymbol{1}|\boldsymbol{h}) \tag{4.16} \\ &= \mathcal{S}(\boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}). \tag{4.17} \end{align}$$

4.2.1.2 Training the parameters

Various training methods for RBMs have been proposed. The most fundamental approach is standard maximum likelihood estimation (MLE). Given a training data set $\{\boldsymbol{v}_n\}_{n=1}^N$, the log-likelihood of the BB-RBM is calculated as

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\Theta}) &= \log p(\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_N; \boldsymbol{\Theta}) && (4.18)\\
&= \log \prod_{n=1}^{N} p(\boldsymbol{v}_n; \boldsymbol{\Theta}) && (4.19)\\
&= \sum_{n=1}^{N} \log p(\boldsymbol{v}_n; \boldsymbol{\Theta}) && (4.20)\\
&= \sum_{n=1}^{N} \log \sum_{\boldsymbol{h}} p(\boldsymbol{v}_n, \boldsymbol{h}; \boldsymbol{\Theta}) && (4.21)\\
&= \sum_{n=1}^{N} \log \sum_{\boldsymbol{h}} \frac{1}{Z(\boldsymbol{\Theta})} e^{-E_{BB}(\boldsymbol{v}_n, \boldsymbol{h}; \boldsymbol{\Theta})} && (4.22)\\
&= \sum_{n=1}^{N} \log \sum_{\boldsymbol{h}} e^{-E_{BB}(\boldsymbol{v}_n, \boldsymbol{h}; \boldsymbol{\Theta})} - \sum_{n=1}^{N} \log Z(\boldsymbol{\Theta}) && (4.23)\\
&= \sum_{n=1}^{N} \log \sum_{\boldsymbol{h}} e^{-E_{BB}(\boldsymbol{v}_n, \boldsymbol{h}; \boldsymbol{\Theta})} - N \log \sum_{\boldsymbol{v}', \boldsymbol{h}'} e^{-E_{BB}(\boldsymbol{v}', \boldsymbol{h}'; \boldsymbol{\Theta})}. && (4.24)
\end{aligned}$$

In MLE, the parameters are usually optimized using gradient descent-based methods. The gradient of the likelihood $\mathcal{L}(\boldsymbol{\Theta})$ with respect to the parameter $\theta \in \boldsymbol{\Theta}$ is given by

$$\frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial \theta} = \frac{\partial}{\partial \theta}\left(\sum_{n=1}^{N} \log \sum_{\boldsymbol{h}} e^{-E_{BB}(\boldsymbol{v}_n,\boldsymbol{h};\boldsymbol{\Theta})}\right) - \frac{\partial}{\partial \theta}\left(N \log \sum_{\boldsymbol{v}',\boldsymbol{h}'} e^{-E_{BB}(\boldsymbol{v}',\boldsymbol{h}';\boldsymbol{\Theta})}\right) \tag{4.25}$$

$$= \sum_{n=1}^{N} \frac{\sum_{\boldsymbol{h}} \frac{\partial}{\partial \theta} e^{-E_{BB}(\boldsymbol{v}_n,\boldsymbol{h};\boldsymbol{\Theta})}}{\sum_{\boldsymbol{h}} e^{-E_{BB}(\boldsymbol{v}_n,\boldsymbol{h};\boldsymbol{\Theta})}} - N \frac{\sum_{\boldsymbol{v}',\boldsymbol{h}'} \frac{\partial}{\partial \theta} e^{-E_{BB}(\boldsymbol{v}',\boldsymbol{h}';\boldsymbol{\Theta})}}{\sum_{\boldsymbol{v}',\boldsymbol{h}'} e^{-E_{BB}(\boldsymbol{v}',\boldsymbol{h}';\boldsymbol{\Theta})}} \tag{4.26}$$

$$= -\sum_{n=1}^{N} \frac{\sum_{\boldsymbol{h}} e^{-E_{BB}(\boldsymbol{v}_n,\boldsymbol{h};\boldsymbol{\Theta})} \frac{\partial E_{BB}(\boldsymbol{v}_n,\boldsymbol{h};\boldsymbol{\Theta})}{\partial \theta}}{\sum_{\boldsymbol{h}} e^{-E_{BB}(\boldsymbol{v}_n,\boldsymbol{h};\boldsymbol{\Theta})}} + N \frac{\sum_{\boldsymbol{v}',\boldsymbol{h}'} e^{-E_{BB}(\boldsymbol{v}',\boldsymbol{h}';\boldsymbol{\Theta})} \frac{\partial E_{BB}(\boldsymbol{v}',\boldsymbol{h}';\boldsymbol{\Theta})}{\partial \theta}}{\sum_{\boldsymbol{v}',\boldsymbol{h}'} e^{-E_{BB}(\boldsymbol{v}',\boldsymbol{h}';\boldsymbol{\Theta})}} \tag{4.27}$$

$$= -\sum_{n=1}^{N} \frac{\sum_{\boldsymbol{h}} Z(\boldsymbol{\Theta}) p(\boldsymbol{v}_n,\boldsymbol{h};\boldsymbol{\Theta}) \frac{\partial E_{BB}(\boldsymbol{v}_n,\boldsymbol{h};\boldsymbol{\Theta})}{\partial \theta}}{\sum_{\boldsymbol{h}} Z(\boldsymbol{\Theta}) p(\boldsymbol{v}_n,\boldsymbol{h};\boldsymbol{\Theta})} + N \frac{\sum_{\boldsymbol{v}',\boldsymbol{h}'} Z(\boldsymbol{\Theta}) p(\boldsymbol{v}',\boldsymbol{h}';\boldsymbol{\Theta}) \frac{\partial E_{BB}(\boldsymbol{v}',\boldsymbol{h}';\boldsymbol{\Theta})}{\partial \theta}}{Z(\boldsymbol{\Theta})} \tag{4.28}$$

$$= -\sum_{n=1}^{N} \frac{\sum_{\boldsymbol{h}} p(\boldsymbol{v}_n,\boldsymbol{h};\boldsymbol{\Theta}) \frac{\partial E_{BB}(\boldsymbol{v}_n,\boldsymbol{h};\boldsymbol{\Theta})}{\partial \theta}}{p(\boldsymbol{v}_n;\boldsymbol{\Theta})} + N \sum_{\boldsymbol{v}',\boldsymbol{h}'} p(\boldsymbol{v}',\boldsymbol{h}';\boldsymbol{\Theta}) \frac{\partial E_{BB}(\boldsymbol{v}',\boldsymbol{h}';\boldsymbol{\Theta})}{\partial \theta} \tag{4.29}$$

$$= -\sum_{n=1}^{N} \sum_{\boldsymbol{h}} \frac{p(\boldsymbol{v}_n,\boldsymbol{h};\boldsymbol{\Theta})}{p(\boldsymbol{v}_n;\boldsymbol{\Theta})} \frac{\partial E_{BB}(\boldsymbol{v}_n,\boldsymbol{h};\boldsymbol{\Theta})}{\partial \theta} + N \sum_{\boldsymbol{v}',\boldsymbol{h}'} p(\boldsymbol{v}',\boldsymbol{h}';\boldsymbol{\Theta}) \frac{\partial E_{BB}(\boldsymbol{v}',\boldsymbol{h}';\boldsymbol{\Theta})}{\partial \theta} \tag{4.30}$$

$$= \sum_{n=1}^{N} \left(-\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}_n;\boldsymbol{\Theta}) \frac{\partial E_{BB}(\boldsymbol{v}_n,\boldsymbol{h};\boldsymbol{\Theta})}{\partial \theta}\right) - \sum_{n=1}^{N} \left(-\sum_{\boldsymbol{v}',\boldsymbol{h}'} p(\boldsymbol{v}',\boldsymbol{h}';\boldsymbol{\Theta}) \frac{\partial E_{BB}(\boldsymbol{v}',\boldsymbol{h}';\boldsymbol{\Theta})}{\partial \theta}\right) \tag{4.31}$$

$$= \sum_{n=1}^{N} \left(-\mathbb{E}\{\frac{\partial E_{BB}(\boldsymbol{v}_n,\boldsymbol{h};\boldsymbol{\Theta})}{\partial \theta}|\boldsymbol{v}_n\}\right) - \sum_{n=1}^{N} \left(-\mathbb{E}\{\frac{\partial E_{BB}(\boldsymbol{v}',\boldsymbol{h}';\boldsymbol{\Theta})}{\partial \theta}\}\right). \tag{4.32}$$

Therefore, the gradient for the normalized likelihood $\mathcal{L}'(\boldsymbol{\Theta}) = \frac{\mathcal{L}(\boldsymbol{\Theta})}{N}$ is given by

$$\frac{\partial \mathcal{L}'(\boldsymbol{\Theta})}{\partial \theta} = \frac{1}{N} \cdot \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial \theta} \tag{4.33}$$

$$= \frac{1}{N}\sum_{n=1}^{N} \left(-\mathbb{E}\{\frac{\partial E_{BB}(\boldsymbol{v}_n,\boldsymbol{h};\boldsymbol{\Theta})}{\partial \theta}|\boldsymbol{v}_n\}\right) - \frac{1}{N}\sum_{n=1}^{N} \left(-\mathbb{E}\{\frac{\partial E_{BB}(\boldsymbol{v}',\boldsymbol{h}';\boldsymbol{\Theta})}{\partial \theta}\}\right) \tag{4.34}$$

$$= \langle -\frac{\partial E_{BB}(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\Theta})}{\partial \theta}\rangle_{data} - \langle -\frac{\partial E_{BB}(\boldsymbol{v}',\boldsymbol{h}';\boldsymbol{\Theta})}{\partial \theta}\rangle_{model}, \tag{4.35}$$

where $\langle\cdot\rangle_{data}$ and $\langle\cdot\rangle_{model}$ indicate expectations of input data and the inner model, respectively.

However, it is usually difficult to compute the second term in Eq. (4.35), because it contains exponentially-expanded terms (it requires caluculation for all combinations of $p(\boldsymbol{v}', \boldsymbol{h}'; \boldsymbol{\Theta})\frac{\partial E_{BB}(\boldsymbol{v},\boldsymbol{h}';\boldsymbol{\Theta})}{\partial\theta}$ in terms of $\boldsymbol{v}'$ and $\boldsymbol{h}'$). One popular and effective solution is to use contrastive divergence (CD) learning algorithm [80] proposed by Hinton. In this approach, by repeating random sampling of $\boldsymbol{h}' \sim p(\boldsymbol{h}|\boldsymbol{v}')$ and $\boldsymbol{v}' \sim p(\boldsymbol{v}|\boldsymbol{h}')$ one after the other starting from the observation $\boldsymbol{v}_n^0 = \boldsymbol{v}_n$ (Gibbs sampling), we approximate the value of $\langle -\frac{\partial E_{BB}(\boldsymbol{v}',\boldsymbol{h}';\boldsymbol{\Theta})}{\partial\theta}\rangle_{model}$ using the samples (as shown in Figure 4.3, we obtain a sample $\boldsymbol{h}_n^0 \sim p(\boldsymbol{h}|\boldsymbol{v}_n)$ firstly, a sample $\boldsymbol{v}_n^1 \sim p(\boldsymbol{v}|\boldsymbol{h}_n^0)$ secondly, a sample $\boldsymbol{h}_n^1 \sim p(\boldsymbol{h}|\boldsymbol{v}_n^1)$ thirdly, etc).

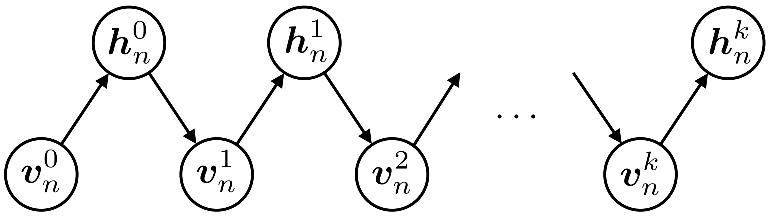


Figure 4.3: Visualizatin of CD learning method.

In CD learning, we run k steps of the Gibbs sampling and approximate the second term in Eq. (4.35) using the kth samples of $\boldsymbol{v}$ and $\boldsymbol{h}$ (the approximation of the second term is represented as $\langle -\frac{\partial E_{BB}(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\Theta})}{\partial\theta}\rangle_{recon.} \triangleq \frac{1}{N}\sum_{n=1}^{N}\left(-\mathbb{E}\{\frac{\partial E_{BB}(\boldsymbol{v}_n^k,\boldsymbol{h}_n^k;\boldsymbol{\Theta})}{\partial\theta}\}\right)$). The gradient for the objective function $\mathcal{L}_{CD}$ in CD learning with respective to θ is written as

$$\frac{\partial\mathcal{L}_{CD}(\boldsymbol{\Theta})}{\partial\theta} = \langle -\frac{\partial E_{BB}(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\Theta})}{\partial\theta}\rangle_{data} - \langle -\frac{\partial E_{BB}(\boldsymbol{v},\boldsymbol{h};\boldsymbol{\Theta})}{\partial\theta}\rangle_{recon.}. \tag{4.36}$$

The important thing in CD learning is that the objective function $\mathcal{L}_{CD}$ actually does not represent the likelihood anymore, but the gap between two KL-divergencies (called contrastive divergence) as follows:

$$\mathcal{L}_{CD}(\boldsymbol{\Theta}) \propto D_{KL}(q(\{\boldsymbol{v}_n\})|p(\boldsymbol{v};\boldsymbol{\Theta})) - D_{KL}(p(\{\boldsymbol{v}_n^k\};\boldsymbol{\Theta})|p(\boldsymbol{v};\boldsymbol{\Theta})), \tag{4.37}$$

where $q(\{\boldsymbol{v}_n\})$ indicates empirical distribution of the observations $\{\boldsymbol{v}_n\}$.

The CD method is based on Gibbs sampling. Hence, it is guaranteed that the closer samples to the true ones we can obtain, the more k is increased $(p(\{\boldsymbol{v}_n^{k\to\infty}\};\boldsymbol{\Theta}) \to p(\boldsymbol{v};\boldsymbol{\Theta})))$. However, it is known that even if we run only one step of sampling ($k=1$), it performs quite well [80]. In this thesis as well, $k=1$ is used. Other approaches, such as the method using Markov chain Monte Carlo (MCMC), persistent contrastive divergence (PCD) [81] as a variant of CD learning, parallel tempering [82], annealed importance sampling [83], maximum pseudo-likelihood [84], ratio matching [85], and neural autoregressive distribution estimator (NADE) [86], have been also proposed to estimate the parameters effectively or overcome the difficulties in RBMs.

The negative gradient of E_{BB} with respect to each parameter, which appears twice in the gradient of the objective function as in Eq. (4.36), can be easily derived as follows:

$$-\frac{\partial E_{BB}(\boldsymbol{v},\boldsymbol{h})}{\partial \boldsymbol{W}} = \boldsymbol{v}\boldsymbol{h}^{\mathrm{T}} \tag{4.38}$$

$$-\frac{\partial E_{BB}(\boldsymbol{v},\boldsymbol{h})}{\partial \boldsymbol{b}} = \boldsymbol{v} \tag{4.39}$$

$$-\frac{\partial E_{BB}(\boldsymbol{v},\boldsymbol{h})}{\partial \boldsymbol{c}} = \boldsymbol{h}. \tag{4.40}$$

In general, a gradient-ascent-based approach uses the following update rule to find the local maxima:

$$\theta \leftarrow \theta + \eta_\theta \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial \theta}, \tag{4.41}$$

where $\eta_\theta > 0$ indicates a learning rate with respect to the parameter θ that determines the speed of convergence.

Summarizing the above equations (Eqs. (4.36)(4.38)(4.39)(4.40)(4.41)), the update rules for each parameter of a BB-RBM using the CD learning method are:

$$\boldsymbol{W} \leftarrow \boldsymbol{W} + \eta_W(\langle \boldsymbol{v}\boldsymbol{h}^{\mathrm{T}}\rangle_{data} - \langle \boldsymbol{v}\boldsymbol{h}^{\mathrm{T}}\rangle_{recon.}) \tag{4.42}$$

$$\boldsymbol{b} \leftarrow \boldsymbol{b} + \eta_b(\langle \boldsymbol{v}\rangle_{data} - \langle \boldsymbol{v}\rangle_{recon.}) \tag{4.43}$$

$$\boldsymbol{c} \leftarrow \boldsymbol{c} + \eta_c(\langle \boldsymbol{h}\rangle_{data} - \langle \boldsymbol{h}\rangle_{recon.}). \tag{4.44}$$

4.2.2 GB-RBM

BB-RBMs discussed in the previous section modeled no more than binary-valued observations; actual data such as natural images and speech signals are not binary-valued but real-valued, though. Therefore, a variant of the BB-RBMs, a Gaussian-Bernoulli RBM (GB-RBM) has been proposed [79], which treats with real-valued data.

However, it has been reported that the original GB-RBM had some difficulties because the training of the parameters was unstable [32, 87, 88]. Later, an improved learning method for GB-RBM was proposed by Cho *et al.* [89] to overcome the difficulties.

In literature dealing with the improved GB-RBM [89], the joint probability $p(\boldsymbol{v}, \boldsymbol{h})$ of real-valued visible units $\boldsymbol{v} = [v_1, \cdots, v_I]^{\mathrm{T}}, v_i \in \mathbb{R}$ and binary-valued hidden units $\boldsymbol{h} = [h_1, \cdots, h_J]^{\mathrm{T}}, h_j \in \{0, 1\}$ is defined as follows:

$$p(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} e^{-E(\boldsymbol{v}, \boldsymbol{h})} \tag{4.45}$$

$$E(\boldsymbol{v}, \boldsymbol{h}) = \left\| \frac{\boldsymbol{v} - \boldsymbol{b}}{2\boldsymbol{\sigma}} \right\|^2 - \boldsymbol{c}^{\mathrm{T}} \boldsymbol{h} - \left(\frac{\boldsymbol{v}}{\boldsymbol{\sigma}^2} \right)^{\mathrm{T}} \boldsymbol{W} \boldsymbol{h} \tag{4.46}$$

$$Z = \sum_{\boldsymbol{v}, \boldsymbol{h}} e^{-E(\boldsymbol{v}, \boldsymbol{h})}, \tag{4.47}$$

where $\| \cdot \|^2$ denotes L2 norm. $\boldsymbol{W} \in \mathbb{R}^{I \times J}$, $\boldsymbol{\sigma} \in \mathbb{R}^{I \times 1}$, $\boldsymbol{b} \in \mathbb{R}^{I \times 1}$, and $\boldsymbol{c} \in \mathbb{R}^{J \times 1}$ are model parameters of the GB-RBM, indicating the weight matrix between visible units and hidden units, the standard deviations associated with Gaussian visible units, a bias vector of the visible units, and a bias vector of hidden units, respectively. The fraction bar in Eq. (4.46) denotes the element-wise division.

Because there are no connections between visible units or between hidden units, the conditional probabilities $p(\boldsymbol{h}|\boldsymbol{v})$ and $p(\boldsymbol{v}|\boldsymbol{h})$ form simple equations as follows:

$$p(h_j = 1 | \boldsymbol{v}) = \mathcal{S} \left(c_j + \left(\frac{\boldsymbol{v}}{\boldsymbol{\sigma}^2} \right)^{\mathrm{T}} \boldsymbol{W}_{:j} \right) \tag{4.48}$$

$$p(v_i = v | \boldsymbol{h}) = \mathcal{N} \left(v \mid b_i + \boldsymbol{W}_{i:} \boldsymbol{h}, \sigma_i^2 \right), \tag{4.49}$$

where $\mathcal{S}(\cdot)$ and $\mathcal{N}(\cdot|\mu, \sigma^2)$ indicate an element-wise sigmoid function and Gaussian probability density function with the mean μ and variance σ^2.

For parameter estimation, the log-likelihood of a collection of visible units $\mathcal{L} = \log \prod_n p(\boldsymbol{v}_n)$ is used as an evaluation function. Differentiating partially with respect to each parameter, we obtain:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_{ij}} = \langle \frac{v_i h_j}{\sigma_i^2} \rangle_{data} - \langle \frac{v_i h_j}{\sigma_i^2} \rangle_{model} \tag{4.50}$$

$$\frac{\partial \mathcal{L}}{\partial b_i} = \langle \frac{v_i}{\sigma_i^2} \rangle_{data} - \langle \frac{v_i}{\sigma_i^2} \rangle_{model} \tag{4.51}$$

$$\frac{\partial \mathcal{L}}{\partial c_j} = \langle h_j \rangle_{data} - \langle h_j \rangle_{model}, \tag{4.52}$$

where $\langle \cdot \rangle_{data}$ and $\langle \cdot \rangle_{model}$ indicate expectations of input data and the inner model, respectively. However, it is generally difficult to compute the second term, so, as discussed in the previous section, the expected reconstructed data $\langle \cdot \rangle_{recon.}$ is used instead (CD learning algorithm).

In the improved GB-RBM, the variance parameter σ_i^2 is replaced as $\sigma_i^2 = e^{z_i}$ so as to constrain the variance to a non-zero value and provide stability in training the parameters. Under this modification, the gradient with respect to z_i becomes

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial z_i} &= e^{-z_i}\langle \frac{(v_i - b_i)^2}{2} - v_i \boldsymbol{W}_{i:}\boldsymbol{h}\rangle_{data} \\ &\quad - e^{-z_i}\langle \frac{(v_i - b_i)^2}{2} - v_i \boldsymbol{W}_{i:}\boldsymbol{h}\rangle_{model}. \end{aligned} \tag{4.53}$$

Using Eqs. (4.50), (4.51), (4.52), and (4.53), each parameter can be updated by stochastic gradient descent with a fixed learning rate starting from initial values just as in the case of BB-RBMs.

4.3 A JD-RBM for parallel dictionary learning and iterative estimation algorithm

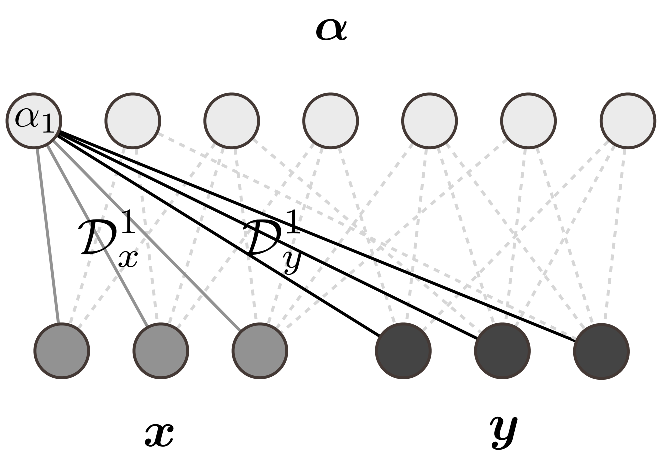


Figure 4.4: A joint density RBM for voice conversion. Each hidden unit connects with the dictionaries of the source and the target speaker.

Our voice conversion system uses a joint density RBM to train parallel dictionaries $\mathcal{D}_x$, $\mathcal{D}_y$ and estimate sparse vectors $\boldsymbol{\alpha}$ at the same time as shown in Figure 4.4. As discussed in the previous section, an RBM is a two-layer network that consists of a visible layer and a hidden layer, characterized in that bi-directional connections exist only between visible and hidden units. As shown in Figure 4.4, the RBM that feeds a concatenated vector of source speaker's features $\boldsymbol{x}$ and target speaker's features $\boldsymbol{y}$ can be regarded as a network where a dictionary-selection weight $\boldsymbol{\alpha}_i$ connects to both $\boldsymbol{x}$ and $\boldsymbol{y}$ with weights of ith dictionaries $\mathcal{D}_x^i$ and $\mathcal{D}_y^i$, respectively.

Given parallel training data $(\boldsymbol{x}, \boldsymbol{y})$, we define a joint probability of $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\alpha}$ as follows:

$$p(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\alpha}; \mathcal{D}_x, \mathcal{D}_y) = \frac{1}{Z} e^{-E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\alpha}; \mathcal{D}_x, \mathcal{D}_y)} \tag{4.54}$$

$$E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\alpha}; \mathcal{D}_x, \mathcal{D}_y) = \left\| \frac{\boldsymbol{x} - \boldsymbol{b}_x}{2\boldsymbol{\sigma}_x} \right\|^2 + \left\| \frac{\boldsymbol{y} - \boldsymbol{b}_y}{2\boldsymbol{\sigma}_y} \right\|^2 \tag{4.55}$$

$$-\boldsymbol{c}^{\mathrm{T}} \boldsymbol{\alpha} - \left(\frac{\boldsymbol{x}}{\boldsymbol{\sigma}_x^2} \right)^{\mathrm{T}} \mathcal{D}_x \boldsymbol{\alpha} - \left(\frac{\boldsymbol{y}}{\boldsymbol{\sigma}_y^2} \right)^{\mathrm{T}} \mathcal{D}_y \boldsymbol{\alpha}$$

where $Z = \sum_{\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\alpha}} e^{-E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\alpha})}$ indicates a normalization term, and $\boldsymbol{c}$ is a bias parameter vector of a dictionary-selection weights. $\boldsymbol{b}_x$ and $\boldsymbol{\sigma}_x$ indicate bias and deviation parameters of the source speaker's acoustic features, respectively, and $\boldsymbol{b}_y$ and $\boldsymbol{\sigma}_y$ indicate bias and deviation parameters of the target speaker's features, respectively. The dictionaries $\mathcal{D}_x, \mathcal{D}_y$ (and the other parameters) can be estimated by maximizing a likelihood $\mathcal{L} = p(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\boldsymbol{\alpha}} p(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\alpha})$ from Eq. (4.54). By partially differentiating the likelihood with respect to each dictionary, we obtain:

$$\frac{\partial \mathcal{L}}{\partial \mathcal{D}_{xdk}} = \langle \frac{x_d \alpha_k}{\sigma_{xd}^2} \rangle_{data} - \langle \frac{x_d \alpha_k}{\sigma_{xd}^2} \rangle_{model} \tag{4.56}$$

$$\frac{\partial \mathcal{L}}{\partial \mathcal{D}_{ydk}} = \langle \frac{y_d \alpha_k}{\sigma_{yd}^2} \rangle_{data} - \langle \frac{y_d \alpha_k}{\sigma_{yd}^2} \rangle_{model}. \tag{4.57}$$

As discussed before, we make practical use of an approximation method (CD learning) to calculate the second terms of Eqs. (4.56)(4.57). For the other parameters, we can also derive the gradients the same as in Eqs. (4.51)(4.52)(4.53).

When it comes to conversion, we estimate the target speaker's vector $\hat{\boldsymbol{y}}$ by repeating forward inference and backward inference of an RBM as shown in Figure **??**. Given an initial vector $\boldsymbol{y}_0$, we first calculate the expectation values of dictionary-selection weights $\boldsymbol{\alpha}$ using the probability that each dictionary is selected:

$$p(\boldsymbol{\alpha} = \mathbf{1} | \boldsymbol{x}, \boldsymbol{y}) = \mathcal{S}(\mathcal{D}_x^{\mathrm{T}} (\frac{\boldsymbol{x}}{\boldsymbol{\sigma}_x^2}) + \mathcal{D}_y^{\mathrm{T}} (\frac{\boldsymbol{y}}{\boldsymbol{\sigma}_y^2}) + \boldsymbol{c}). \tag{4.58}$$

Secondly, the expectation values of $\boldsymbol{y}$ are calculated using $\hat{\boldsymbol{\alpha}}$. The conditional probability of $\boldsymbol{y}$ is given from backward inference of an RBM as follows:

$$p(\boldsymbol{y} | \boldsymbol{\alpha}) = \mathcal{N}(\boldsymbol{y} | \mathcal{D}_y \boldsymbol{\alpha} + \boldsymbol{b}_y, \boldsymbol{\sigma}_y^2). \tag{4.59}$$

Repeating the above-mentioned procedures (estimation of $\boldsymbol{\alpha}$ and $\boldsymbol{y}$) R times, we iteratively obtain the converted vector $\hat{\boldsymbol{y}}$. Although several approaches for

```
Input: Dictionaries $\mathcal{D}_x$, $\mathcal{D}_y$, a source speaker's vector $\boldsymbol{x}$ and an initial target vector $\boldsymbol{y}_0$
Output: Estimated target speaker's vector $\hat{\boldsymbol{y}}$
Initialize: Set the initial values as $\hat{\boldsymbol{y}} = \boldsymbol{y}_0$.

Repeat the following updates $R$ times:

    1. $\hat{\boldsymbol{\alpha}} \triangleq E\{\boldsymbol{\alpha}\}_{p(\boldsymbol{\alpha}|\boldsymbol{x},\hat{\boldsymbol{y}})} = \mathcal{S}(\mathcal{D}_x^{\mathrm{T}}(\frac{\boldsymbol{x}}{\boldsymbol{\sigma}_x^2}) + \mathcal{D}_y^{\mathrm{T}}(\frac{\hat{\boldsymbol{y}}}{\boldsymbol{\sigma}_y^2}) + \boldsymbol{c})$

    2. $\hat{\boldsymbol{y}} \triangleq E\{\boldsymbol{\alpha}\}_{p(\boldsymbol{y}|\hat{\boldsymbol{\alpha}})} = \mathcal{D}_y\hat{\boldsymbol{\alpha}} + \boldsymbol{b}_y$
```

Figure 4.5: Iterative estimation algorithm of the target vector using a joint density RBM.

determining the initial vector $\boldsymbol{y}_0$ can be considered (e.g., an approach that uses the source feature vector $\boldsymbol{x}$, or estimated values obtained from another methods such as GMM), we use a zero vector $\mathbf{0}$ in this paper.

Similar to SMNMF (spectral mapping NMF [68]), our voice conversion method optimizes the likelihood of the training data as well as the likelihood of the dictionary-selection vector as shown in Eq. (4.54). The most obvious difference is that our approach uses a non-linear function for estimating a dictionary-selection vector as in Eq. (4.58), while SMNMF still uses a linear function. Furthermore, while SMNMF is restricted to input non-negative values, our approach can feed real values without constraints. In particular, MFCC, which tends to distribute monomodally, will go together with our approach that assumes Gaussian-distributed inputs.

4.4 Experiments

4.4.1 Conditions

VC experiments were conducted using the ATR Japanese speech database [90], comparing our method (joint density restricted Boltzmann machines, or "JDRBM") with the conventional sparse-representation-based voice conversion that uses exemplar-based NMF [28] ("NN"), and spectral-mapping-based NMF [68] ("SMNMF") and, for a reference, the well-known GMM-based approach (64 mixtures). From this database, a male speaker (identified with "MMY") and a female speaker ("FTK") were selected and used for the source and target speakers, respectively. As an acoustic feature vector for the proposed method and

Table 4.1: Performance of each method.

Method	Joint density RBM (Proposed) JDRBM-96	JDRBM-192	JDRBM-384	Spectral mapping (NMF) SMNMF-1000	SMNMF-2500	Exemplar-based (NMF) EXNMF-1000	EXNMF-58426	GMM GMM-64
SDIR (dB)	5.21	**5.32**	4.45	5.14	4.68	4.91	5.23	4.11

GMM, 24-dimensional MFCC features were calculated from STRAIGHT spectra [48] using filter-theory [43] to decode the MFCC back to STRAIGHT spectra in the synthesis stage. For NMF-based approaches, 513-dimensional vectors of STRAIGHT spectra were used. The parallel data of the source/target speakers processed by Dynamic Programming were created from 216 word utterances (58,426 frames) in the dataset, and were used for the training of each method. For the objective test, 25 sentences (about 100 sec. long) that were not included in the training data were arbitrarily selected from the database. The RBM was trained using gradient descent (ascent) with a learning rate of 0.01 and momentum of 0.9 for all parameters, with the number of epochs being 400. We changed the number of hidden units as 96, 192, and 384, and evaluated their performance (referred to as "JDRBM-96", "JDRBM-192", and "JDRBM-384", respectively). The converted vector was obtained after $R = 5$ iterations (already converged). For spectral-mapping-based NMF, we changed the number of bases as 1,000 ("SMNMF-1000") and 2,500 ("SMNMF-2500"). For exemplar-based NMF, we compared the case where all training frames were used ("EXNMF-58426") and the case where 1,000 frames were arbitrarily used from the training data ("EXNMF-1000").

For the objective evaluation, SDIR (spectral distortion improvement ratio) was used to measure how the converted vector is improved to resemble the original source vector. The SDIR is defined as follows:

$$SDIR[dB] = 10 \log_{10} \frac{\sum_d |\boldsymbol{X}^t(d) - \boldsymbol{X}^s(d)|^2}{\sum_d |\boldsymbol{X}^t(d) - \hat{\boldsymbol{X}}^t(d)|^2}, \qquad (4.60)$$

where $\boldsymbol{X}^t(d)$, $\boldsymbol{X}^s(d)$ and $\hat{\boldsymbol{X}}^t(d)$ denote the dth original target spectra, the source spectra and the converted spectra (spectra obtained from the converted MFCC), respectively. The larger the value of SDIR is, the greater the improvement in the converted spectra. We calculated the SDIR for each frame in the training data, and averaged the SDIR values for the final evaluation.

4.4.2 Results and discussion

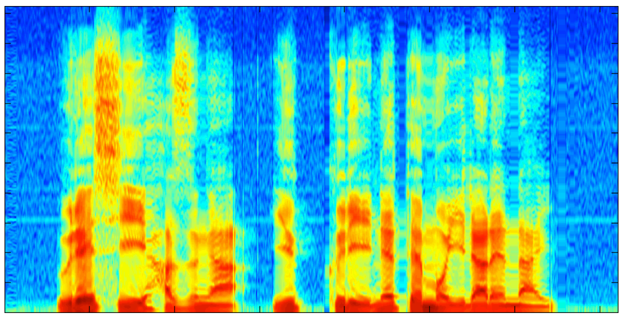
We summarize the experimental results of each method in Table 4.1. As shown in Table 4.1, the proposed approach ("JDRBM-192") outperformed the other methods. The differences between the proposed approach and SMNMF are the types of input features and the gate functions in Eqs. (4.1)(4.2). The reason for the improvement is attributed to the fact that the proposed approach, which inputs real-valued data and uses non-linear gate functions, is able to represent input features better than SMNMF. When we compare within the proposed methods, we can see that the performance degrades when the number of hidden units is too large ("JDRBM-384") or too small ("JDRBM-96"). This is because the JDRBM where the number of hidden units is too small could not represent the input data sufficiently, and the JDRBM where the number of hidden units is too large was over-fitted.

Figure 4.6 shows an example of the spectrogram converted using the proposed method "JDRBM-192", along with the spectrograms of the source speech and the target speech, and expected values of the dictionary-selection vectors. In particular, as shown in Figure 4.6(d), even though the proposed approach does not explicitly use any constraints related to sparsity, the estimated $\boldsymbol{\alpha}$ seems very sparse (almost all of the values in $\boldsymbol{\alpha}$ are zero). This is due to the fact that an RBM characteristically naturally makes the hidden units sparse in the process where the model parameters are estimated so that the hidden units do not capture redundant information between each other.

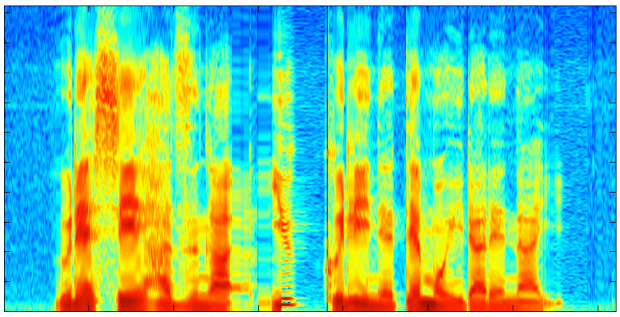
4.5 Summary

This chapter presented a voice conversion method using a joint density RBM as an alternative tool of a sparse-representation-based approach where only a few dictionaries are used for the converted-voice generation. The proposed method demonstrated better performance compared with the conventional sparse-representation-based approach (spectral-mapping NMF and exemplar-based NMF) and the well-known GMM-based approach. In the future, we will study a method that automatically decides the appropriate number of hidden units since the number seems important. Furthermore, we will extend the proposed method to have a deeper architecture using a deep Boltzmann machine [91] or such, so that it captures more complex information in the data.

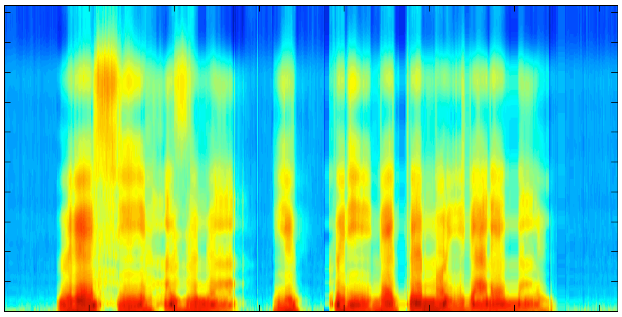
(a)



(b)



(c)



(d)

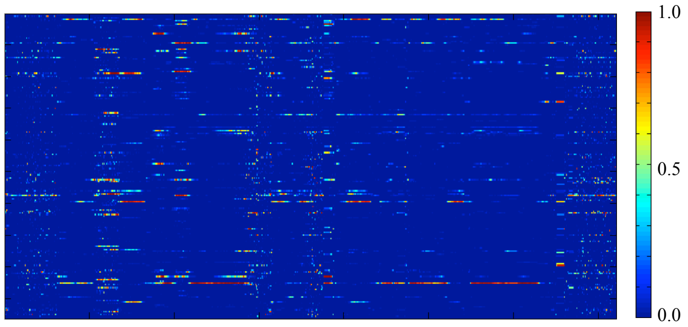


Figure 4.6: Examples of voice conversion using an utterance "ureshii hazuga yukkuri netemo irarenai". (a) Spectrogram of the male source speaker, (b) spectrogram of the female target speaker, (c) converted spectrogram obtained by JRBM-192, and (d) expectation values of the estimated $\boldsymbol{\alpha}$ (vertical and horizontal axes indicate the index and the time, respectively).

Chapter 5

VC Using Speaker-Dependent RBMs

In this chapter, a VC method based on capturing speaker-dependent high-order features that is created using RBMs is presented. In this approach, a source speech is converted to the target one through the high-order space, expecting that such high-order representation makes the conversion easier than the original spectrum space. This chapter investigates the VC approach using a single stack of RBMs or their stacked version (deep belief networks; DBNs [32]), for capturing the latent representation. First we will give some explanation the VC technique using speaker-dependent RBMs, and then VC with speaker-dependent DBNs as its extension.

5.1 Methodology

In the proposed approach, we first train two exclusive RBMs (or DBNs) prepared either for the source or for the target speakers to obtain the high-level spaces that capture abstractions for each speech. It can be considered that because the training data for the source speaker RBMs, for instance, includes various phonemes of the speaker, the networks try to capture the abstractions to maximally express the training data that have abundant speaker individuality information and less phonological information. Therefore, we can expect that it is easier to convert the feature vectors in these speaker-individuality-emphasized high-order spaces than the original spectrum-based space, alleviating the over-fitting or over-smoothing effect.

Fig. 5.1 shows a flow of the proposed method, where an input vector (spectrum or MFCC) of the source speaker (s) is converted to that of a target speaker (t) in the high-order space using RBMs. We prepare different RBMs for source speech

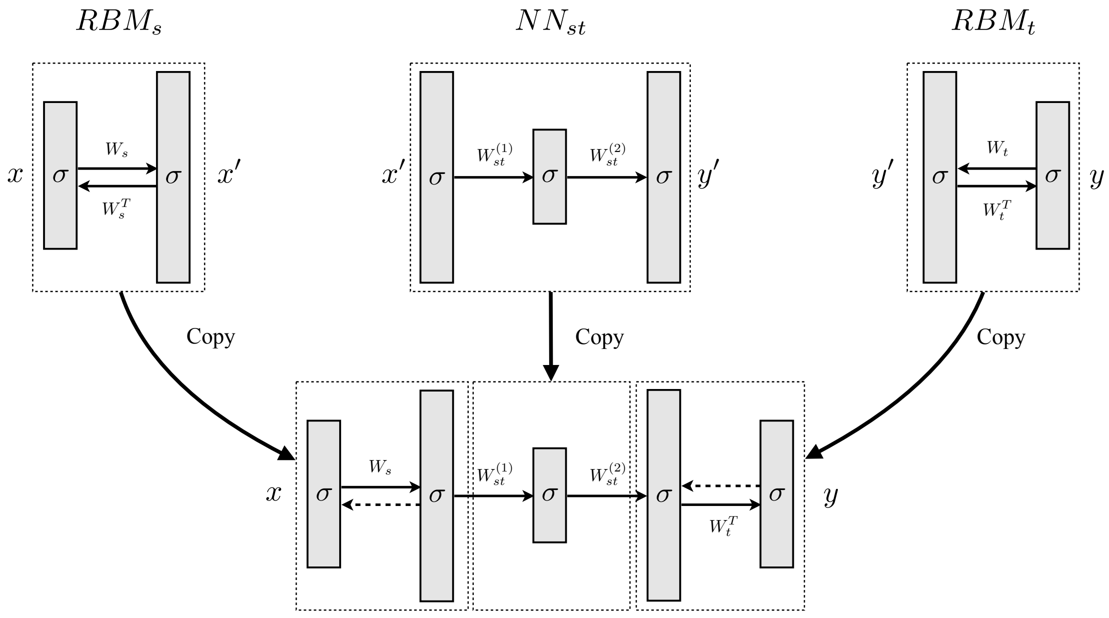


Figure 5.1: The proposed voice conversion architecture, combined with two different RBMs and a concatenating NN. A source feature vector x is fed to RBM_s, NN_{st}, and RBM_t in order, and then converted to a target vector y. This figure shows an example of architectures with one hidden layer in the NN. σ indicates a standard sigmoid function.

and target speech (RBM_s and RBM_t, respectively) so as to capture the speaker-individuality information. All the RBMs are trained using only the corresponding speaker's training data. As discussed in the previous chapter, RBMs share the weights (tied-weights) between bottom-up and top-down. Given weight parameter matrices $\boldsymbol{W}_s$ and $\boldsymbol{W}_t$ for RBM_s and RBM_t, respectively, each bottom-up conversion (encoding) functions $\zeta_s(\boldsymbol{z})$ and $\zeta_t(\boldsymbol{z})$ can be represented by:

$$\zeta_i(\boldsymbol{z}) = \mathcal{S}(\boldsymbol{W}_i\boldsymbol{z}), \quad i \in \{s, t\}. \tag{5.1}$$

Similarly, given a high-order feature vector at the hidden layer, a top-down conversion (decoding) function $\zeta_i^{-1}(\boldsymbol{z})$ that brings it back to the original (spectrum) space is given by:

$$\zeta_i^{-1}(\boldsymbol{z}) = \mathcal{S}(\boldsymbol{W}_i^{\mathrm{T}}\boldsymbol{z}). \tag{5.2}$$

In the proposed approach, the compact-represented input vector calculated using Eq. (5.1) is converted into the high-order target space using $(L+1)$ layers perceptron NN_{st} (L is a small number; less than about 2). Once the weight parameters $\boldsymbol{W}_{st}^{(l)}(l = 1, 2, \ldots, L)$ of NN_{st} are estimated beforehand, an input vector can be converted to:

$$\eta_{st}(\boldsymbol{z}) = \bigodot_{l=1}^{L} \eta_{st}^{(l)}(\boldsymbol{z}) \tag{5.3}$$

$$\eta_{st}^{(l)}(\boldsymbol{z}) = \mathcal{S}(\boldsymbol{W}_{st}^{(l)}\boldsymbol{z}) \tag{5.4}$$

where $\bigodot_{l=1}^{L}$ denotes composition of L functions in this thesis. For instance, $\bigodot_{l=1}^{2} \eta_{st}^{(l)}(\boldsymbol{z}) = \mathcal{S}(\boldsymbol{W}_{st}^{(2)}\mathcal{S}(\boldsymbol{W}_{st}^{(1)}\boldsymbol{z}))$ if we give 2 hidden layers.

Summarizing the above, a conversion function of the proposed method from a source speech $\boldsymbol{x}$ to a target speech $\boldsymbol{y}$ is given by:

$$\boldsymbol{y} = \zeta_t^{-1}(\eta_{st}(\zeta_s(\boldsymbol{x}))) \tag{5.5}$$

$$= \bigodot_{l=1}^{L+2} \mathcal{S}(\boldsymbol{W}^{(l)}\boldsymbol{x}) \tag{5.6}$$

where $\boldsymbol{W}^{(l)}$ denotes an element of a set of weight parameters $\boldsymbol{W}$:

$$\boldsymbol{W} = \{\boldsymbol{W}^{(l)}\}_{l=1}^{L+2} \tag{5.7}$$

$$= \{\boldsymbol{W}_s, \boldsymbol{W}_{st}^{(1)}, \cdots, \boldsymbol{W}_{st}^{(L)}, \boldsymbol{W}_t{}^T\}. \tag{5.8}$$

As Eq. (5.6) indicates, the proposed conversion method is based on the composite function of multiple different non-linear functions. On the other hand, a conventional GMM-based approach converts the source features $\boldsymbol{x}$ as in Eq. (3.18), which shows it to be an additive model of non-linear functions. Therefore, it is expected that the proposed composite model has a richer expression than the conventional GMM-based method.

5.1.1 Training the networks

In order to train the whole VC network as shown in Fig. 5.1, we carry out three-step training. First, we train RBMs for each speaker independently using non-parallel training data. Then, parallel data ($\boldsymbol{x}$ for the source speaker and $\boldsymbol{y}$ for the target speaker in Fig. 5.1) are fed to each of the RBMs and the obtained high-order features ($\boldsymbol{x}'$ and $\boldsymbol{y}'$) are used for the training of the concatenating NN (NN_{st}). Finally, the parameters of the network are fine-tuned using the acousitical parallel data ($\boldsymbol{x}$ and $\boldsymbol{y}$).

After training two RBMs for source and target speakers, we train a converting-in-high-order-space NN_{st} using parallel speeches $\{\boldsymbol{x}_n, \boldsymbol{y}_n\}_{n=1}^{N}$ of source/target speakers. Weight parameters of NN_{st} are estimated so as to minimize the error between the output $\eta_{st}(\zeta_s(\boldsymbol{x}_n))$ and the target vector $\zeta_t(\boldsymbol{y}_n)$ in a common way of

NN. Finally, each parameter of each whole network (RBM_s, RBM_t and NN_{st}) is fine-tuned by back-propagation using the raw parallel data as described above.

The above-mentioned training technique can be applied to binary-valued training data, though acoustic features, such as MFCC, are generally real-valued. One approach to feeding real-valued data to the network is to use GB-RBMs [32] as discussed in the previous chapter. However, we could not achieve high performance with this approach in the experiments (Section 5.2.2) because the GB-RBM was not well suited to the task due to its limited representation ability. Therefore, in the proposed approach, we first use a sigmoid function to give soft-binary' values to the training data, then train the BB-RBMs, and finally de-binarize the outputs using an inverse of a sigmoid function defined as

$$\sigma^{-1}(\boldsymbol{z}) = -\log\left(\frac{1}{\boldsymbol{z}} - 1\right). \tag{5.9}$$

5.1.2 Extension for deeper network

We can easily extend the VC network to be deeper by alternating the speaker-dependent RBMs with DBNs [32]. A DBN stacks multiple RBMs layer-by-layer as shown in Figure 5.2(b). As shown in Figure 5.2(a), after estimating the parameters for the first RBM $\boldsymbol{W}_1$, we estimate the parameters for the following RBM $\boldsymbol{W}_2$, regarding the inferred hidden units in the first RBM ($\mathbb{E}(\boldsymbol{h}|\boldsymbol{v})$) as the visible units in the second RBM. This procedure is repeated layer-by-layer until the highest layer is reached. Stacking such RBMs many times, it can represent more complex features because the higher-layer RBM detects abstractions from the information propagated from the lower-layer RBM. When estimating $\boldsymbol{W}_2$, we do not go back to the first RBM and not modify $\boldsymbol{W}_1$. Therefore, as shown in Figure 5.2(b), the parameters on the top RBM ($\boldsymbol{W}_3$) is bi-directional, while the other parameters ($\boldsymbol{W}_1$ and $\boldsymbol{W}_2$) are undirectional.

As for the deeper architecture of the proposed model, speaker-dependent DBNs are used instead of speaker-dependent RBMs. By replacing the encoding function $\zeta_i(\boldsymbol{z})$ in Eq. (5.1) and the decoding function $\zeta_i^{(-1)}(\boldsymbol{z})$ in Eq. (5.2) with

$$\zeta_i(\boldsymbol{z}) = \bigodot_{l=1}^{L'} \mathcal{S}(\boldsymbol{W}_i^{(l)}\boldsymbol{z}) \tag{5.10}$$

$$\zeta_i^{-1}(\boldsymbol{z}) = \bigodot_{l=1}^{L'} \mathcal{S}(\boldsymbol{W}_i^{(L'-l+1)^T}\boldsymbol{z}), \tag{5.11}$$

repectively, where L' is the number of layers for each DBN.

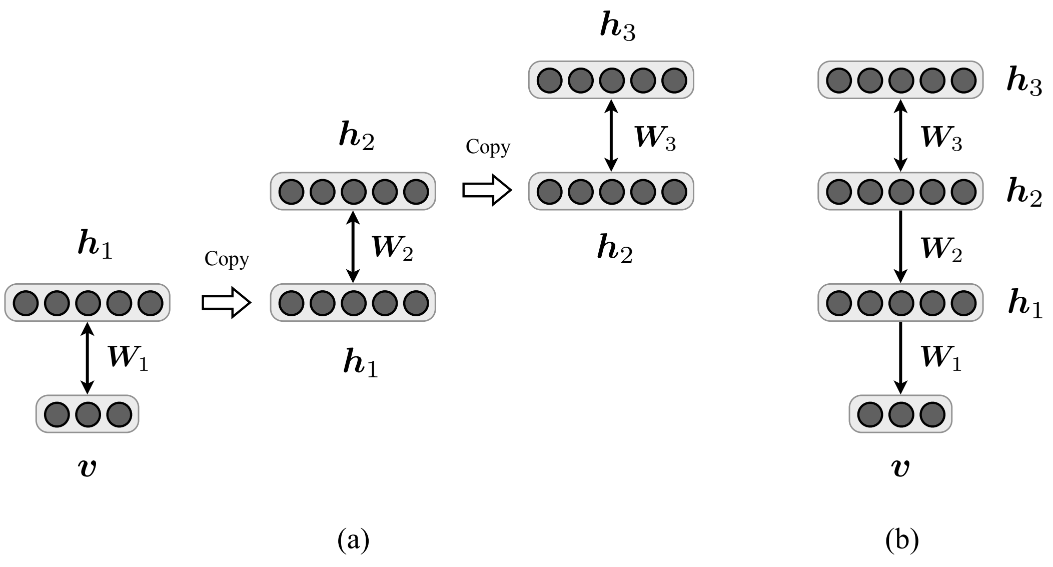


Figure 5.2: The process of estimating the parameters $\boldsymbol{W}_1$, $\boldsymbol{W}_2$, and $\boldsymbol{W}_3$ (a), and the visualization of a DBN (b).

5.2 Experiments

5.2.1 Setup

Voice conversion experiments were conducted using the ATR Japanese speech database [90], comparing the proposed method with the well-known GMM-based approach and conventional NN-based voice conversion. In order to evaluate the proposed method under various conditions, we tested male-to-female (the speakers are identified with MMY and FTK, respectively), female-to-female (FKN and FTK), and male-to-male (MMY and MHT) patterns.

As an input vector, 24-dimensional MFCC features were calculated from STRAIGHT spectra [48] using filter-theory [43] so as to decode the MFCC back to STRAIGHT spectra in the synthesis stage. For the GMM-based approach (64 mixtures), delta (24 dimension) and delta-delta (24 dimension) features were further calculated in a common way of GMM-based VC, and concatenated them as a super vector (totally 72 dimension).

The parallel data of the source/target speakers processed by dynamic programming were created from 216 word utterances in the dataset, and used for the training. Note that the parallel data were prepared for NN and GMM, and two speaker-wise RBMs were trained independently[1] using the separated data

[1]However, in the proposed approach, the parallel data were used in the fine-tuning step.

of the parallel data for each speaker (the amount of parallel training data and the training data for the RBMs is the same, respectively). The learning rate and the number of epochs in the gradient decent-based training of RBMs were 0.05 and 100, respectively. Various architectures of RBMs and NN were tested in subsection 5.2.3. For the objective test, 20 sentences (about 70 sec. long) were arbitrarily selected from the database. These sentences were not included in the training data.

For the objective evaluation, MCD was used, which measures how the converted vector is close to the target vector in the mel-cepstral space. The MCD is defined as below:

$$MCD[dB] = \frac{10}{\ln 10}\sqrt{2\sum_{d=1}^{24}(\boldsymbol{c}_d - \boldsymbol{c}'_d)^2} \tag{5.12}$$

where $\boldsymbol{c}_d$ and $\boldsymbol{c}'_d$ denote the dth original target MFCC and the converted MFCC, respectively. The smaller the value of MCD is, the closer the converted spectra are to the target spectra. The MCD was calculated for each frame in the training data, and averaged them for the final evaluation.

5.2.2 BB-RBM vs. GB-RBM

In this section, the performances of a Bernoulli-Bernoulli RBM (BB-RBM) and a Gaussian-Bernoulli RBM (GB-RBM) are compared on the VC task just for the reference. For the GB-RBM, the hyper-parameters did not change as in the case of BB-RBM except that the learning rate became 0.001. The partial gradient for the variance $\boldsymbol{\sigma}$ can be derived, as well as the other parameters; however the variance parameters are constrained to be positive and difficult to estimate in a normal way. Therefore, we take the common approach where the training data are first normalized to have a zero mean and unit variance and those parameters are fixed as in many existing works [87, 92, 93]. In this preliminary experiment, "arc. 2" were used, which is shown in Table 5.1 for both architectures and 10k training data.

The averaged MCD obtained from the BB-RBM and the GB-RBM were 4.88 and 5.31, respectively, which means that the BB-RBM performed better than the GB-RBM. The joint probability density of a GB-RBM can be rewritten as a linear superposition of Gaussians and its representation is quite limited in that it only captures a parallelepiped distribution, as discussed in [94]. The acoustic features, MFCC, distribute with a non-parallelepiped shape and may not be captured by the GB-RBM. On the other hand, the soft-binarized training data using a sigmoid function distribute like a Bernoulli distribution. Therefore, we obtained a better

Table 5.1: Various architectures for the preliminary experiment.

Arcs.	NN	Proposed	Layers	Params.
arc. 1	[24-24-24-24]	[24:24-24:24]	4	1,800
arc. 2	[24-48-48-24]	[24:48-48:24]	4	4,728
arc. 3	[24-24-24-24-24-24]	[24:24:24-24:24:24]	6	3,000
arc. 4	[24-48-24-48-24]	[24:48-24-48:24]	5	4,752
arc. 5	[24-72-72-24]	[24:72-72:24]	4	8,808
arc. 6	[24-72-24-24-72-24]	[24:72:24-24:72:24]	6	7,704
arc. 7	[24-24-72-72-24-24]	[24:24:72-72:24:24]	6	10,008

result when using the BB-RBM than the GB-RBM in the experiment. BB-RBMs will be used in the following experiments.

5.2.3 Changing the architectures

5.2.3.1 NN-based methods

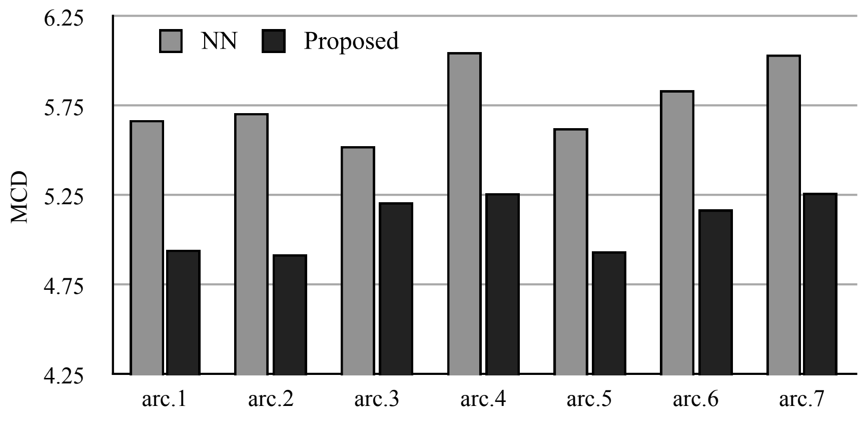


Figure 5.3: Averaged mel-cepstral distortion of various methods with changing network architectures in NN-based VC and the proposed approach (N=10,000).

First we investigated how the proposed approach works as the architecture of the VC network changes, comparing to the conventional NN-based VC with the same architecture.

In this preliminary experiment, we compared 7 type of architectures and change the number of layers and the number of nodes in each layer as listed in Table 5.1. The values in Table 1 indicate the number of nodes from the source layer to the target layer, providing with the number of hidden layers and the number of parameters for each architecture. For the proposed method in particular, the numbers are described like *[RBM (or DBN) for source - Concatenating NN - RBM (or DBN) for target]*.

Fig. 5.3 compares averaged MCD obtained from each architecture. When we look at the relationship between the conversion performance and the number of parameters, the architectures with a large number of parameters do not necessarily perform well. For example, "arc. 1", which has the smallest number of parameters, outperformed "arc. 7", which has the largest number of parameters, for NN and the proposed method. The number of hidden layers seems to be a more important factor than the parameters; i.e., whether the architecture is deep or shallow. Although we expected that the deeper architecture would perform better, as shown in Fig. 5.3, the deeper architecture (such as "arc. 3" or "arc. 6") does not always provide better results than shallower architectures in the proposed method, while the 6-layer architecture "arc. 3" performed best in the conventional NN approach. One reason for this is that the deeper architectures depend on DBNs that estimate the parameters as an undirected model only in the highest layer and the parameters as directed models in the other layers. Hence, some errors might occur when using the transposed weight matrix (as the undirected-model parameters in all layers) to the target speaker's acoustic features in the decoding stage. The best-performed architecture in the proposed method was "arc. 2"; hence for all the remaining experiments reported in this paper, the four layer architecture "arc. 2" is used.

5.2.3.2 GMM-based method

We further investigated the GMM-based voice conversion to confirm the performance when we change the number of mixtures (8, 16, 32, 64 and 128 mixtures). Just as we did in the preliminary experiment of the network-based methods, we compared the performances of these methods fixing the number of training data to 10,000.

Fig. 5.4 shows the voice conversion accuracies when using the GMM with various mixtures. As shown in the figure, the GMM with 64 mixtures performed best of all. Therefore, we used mixtures of 64 for GMM-based voice conversion in the evaluation experiments described in Section 5.2.4.

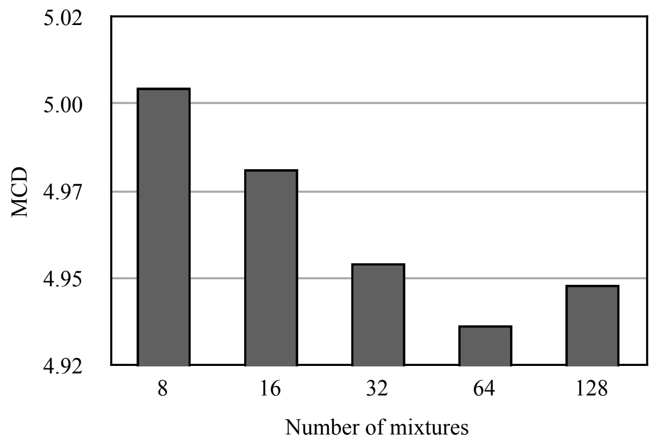


Figure 5.4: Averaged mel-cepstral distortion of GMM-based VC when changing the number of mixtures (N=10,000).

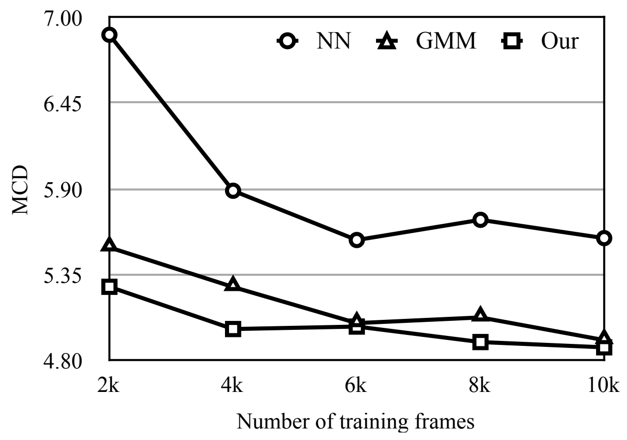


Figure 5.5: Averaged mel-cepstral distortion of male-to-female voice conversion for each method.

5.2.4 Voice conversion evaluation

In this section, we evaluate the proposed method comparing it with conventional GMM and NN using objective and subjective criteria for each pair of speakers, by changing the number of training frames.

Fig. 5.5, Fig. 5.6 and Fig. 5.7 summarize the experimental results for the test data, comparing each method with respect to objective criteria for male-

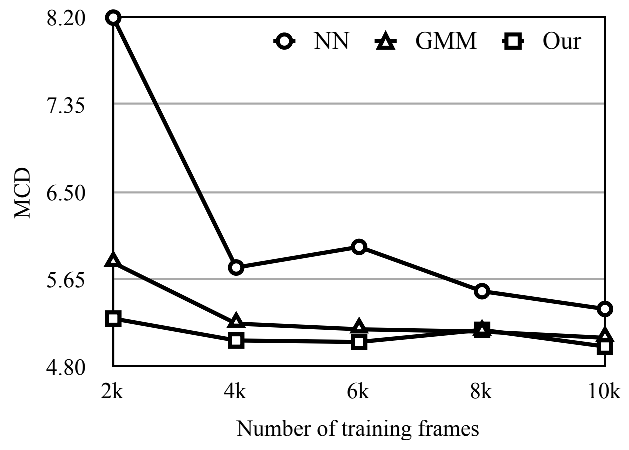


Figure 5.6: Averaged mel-cepstral distortion of female-to-female voice conversion for each method.

to-female, female-to-female, and male-to-male voice conversion, respectively. As shown in these figures, the proposed approach ("Proposed") outperformed both the conventional methods (GMM and NN) in every case. The reason for the improvement can be considered to be the result of the fact that the high-order conversion system (which makes use of RBMs) could capture and convert the abstractions of speaker individualities better than other methods. Especially the proposed method works well when the number of the training data is small compared with the NN approach; the conventional method suffered from the results of over-fitting caused by a shortage of training data, whereas the proposed model did not. The most interesting point is that in the case of male-to-male VC (Fig. 5.7), both GMM and NN approaches are strongly influenced by the training data, and were over-fitted to particular frames in the training data, providing worse performance at $N = 6k$ than at the smaller number of training frames. However, the proposed model, which is based on the feature extraction in high-order latent spaces, obtained very stable result in that case (at $N = 6k$).

For the subjective evaluation, MOS (mean opinion score) listening tests were conducted, where 6 subjects listened to pairs of an original target speech (generated from analysis-by-synthesis) and the converted speeches for each method (the proposed method, GMM and NN), and then selected how close the converted speech sounded to the original one in terms of speaker individuality on a 5-point scale (5: excellent, 4: good, 3: fair, 2: poor, 1: bad). Since we want to compare the accuracy of spectral conversion, we processed Dynamic Programming on each test sentence, converted only spectrum features, and synthesized converted

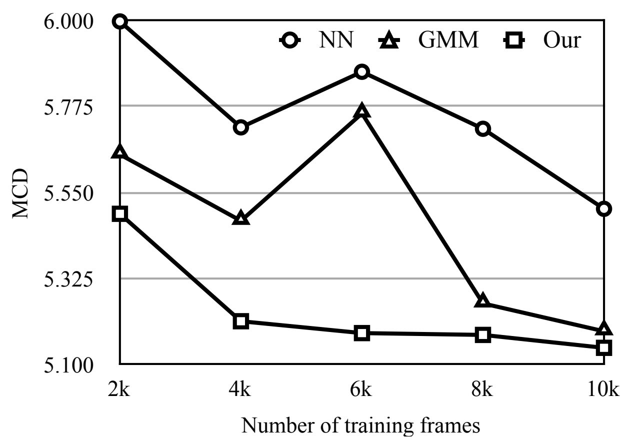


Figure 5.7: Averaged mel-cepstral distortion of male-to-male voice conversion for each method.

speeches with the same pitches as the target speech. That would be helpful for the listeners to concentrate on the differences in the spectra. The results of the preference tests are shown in Fig. 5.8. As shown in Fig. 5.8, the proposed approach performed better than the conventional GMM and NN methods in the subjective criterion as well. However, we could not find any significance between the proposed method and GMM in terms of auditory sense when we look at the confidence intervals, though the figures show significant improvements to the conventional NN approach. This is considered to be due to discontinuous sound caused by the lack of dynamic features in the proposed method.

5.3 Summary

In this chapter, we presented a voice conversion technique using speaker-dependent RBMs to generate the high-order eigen space of the speaker, where it captures the abstractions of speaker individuality. The experimental results showed the efficacy of the proposed method, in comparison to a conventional GMM-based and NN-based approach. Since a DBN does not contain all bidirectional weights, it might produce some erros in the decoding stage. In the future, we will alternately use speaker-dependent deep Boltzmann machines (DBMs) instead of using DBNs, which have bidirectional weights.

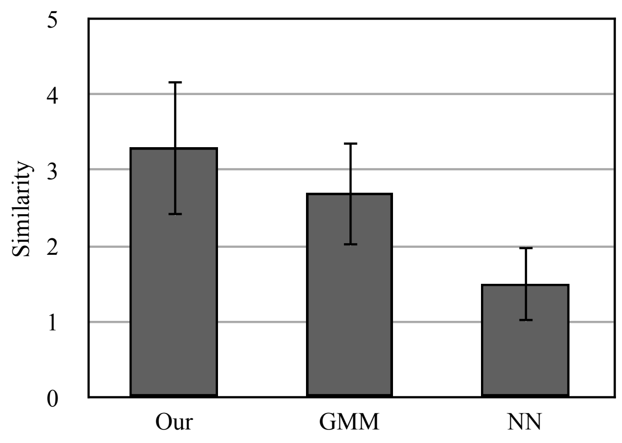


Figure 5.8: MOS scores on male-to-female voice conversion. The error bars show 95% confidence intervals.

Chapter 6

VC Using Speaker-Dependent CRBMs

In this chapter, an extension of the VC method described in Chapter 5 is presented. The proposed method here extends to systematically capture time information as well as latent (deep) relationships between a source speaker's and a target speaker's features in a single network. This is accomplished by combining speaker-dependent conditional restricted Boltzmann machines (CRBMs) and a concatenating NN. This capture starts from introduction of CRBMs, explains the proposed approach, and ends with the experimental evaluation.

6.1 Conditional Restricted Boltzmann Machine (CRBM)

A CRBM is an extended version of RBM proposed by Taylor *et al.* [40], and is suitable for the representation of time series data. As shown in Figure 6.1, in addition to the use of an undirected model as in RBM, CRBM also employs directed models from a collection of previous visible units $\mathcal{V}'^{(t)} = \{\boldsymbol{v}^{(p)}\}_{p=t-P}^{t-1}$, $\boldsymbol{v}^{(p)} = [v_1^{(p)}, \cdots, v_I^{(p)}]^T, v_i^{(p)} \in \mathbb{R}$ to binary hidden units $\boldsymbol{h}^{(t)} = [h_1^{(t)}, \cdots, h_J^{(t)}]^T, h_j^{(t)} \in \{0,1\}$ and to the current visible units $\boldsymbol{v}^{(t)} = [v_1^{(t)}, \cdots, v_I^{(t)}]^T, v_i^{(t)} \in \mathbb{R}$ at the current frame t, where P is the number of previous frames from the current frame taken into account. In this model, there are three types of parameters to be estimated: $\boldsymbol{W}_{v'_p v} \in \mathbb{R}^{I \times I}$ (a directed weight matrix from $\boldsymbol{v}^{(t-p)}$ to $\boldsymbol{v}^{(t)}$), $\boldsymbol{W}_{v'_p h} \in \mathbb{R}^{I \times J}$ (a directed weight matrix from $\boldsymbol{v}^{(t-p)}$ to $\boldsymbol{h}^{(t)}$), and $\boldsymbol{W}_{vh} \in \mathbb{R}^{I \times J}$ (an undirected weight matrix between $\boldsymbol{v}^{(t)}$ and $\boldsymbol{h}^{(t)}$). These weights are estimated using contrastive divergence in a similar manner to an RBM by maximizing the likelihood

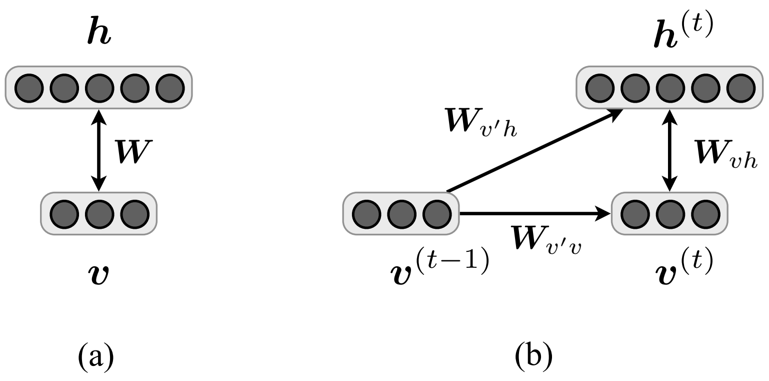


Figure 6.1: Graphical representation of a standard RBM (a) and a CRBM with $P = 1$ (b).

$\mathcal{L} = \log \prod_t p(\boldsymbol{v}^{(t)}|\mathcal{V}'^{(t)})$, where

$$p(\boldsymbol{v}^{(t)}|\mathcal{V}'^{(t)}) = \frac{1}{Z}\sum_{\boldsymbol{h}^{(t)}} e^{-E(\boldsymbol{v}^{(t)},\boldsymbol{h}^{(t)}|\mathcal{V}'^{(t)})} \tag{6.1}$$

$$Z = \sum_{\boldsymbol{v}^{(t)},\boldsymbol{h}^{(t)}} e^{-E(\boldsymbol{v}^{(t)},\boldsymbol{h}^{(t)}|\mathcal{V}'^{(t)})}. \tag{6.2}$$

Inspired by the improvement learning method of a GBRBM, the energy function E is defined as follows:

$$E(\boldsymbol{v}^{(t)},\boldsymbol{h}^{(t)}|\mathcal{V}'^{(t)}) \tag{6.3}$$
$$= \left\| \frac{\boldsymbol{v}^{(t)} - \boldsymbol{b}'^{(t)}}{2\boldsymbol{\sigma}} \right\|^2 - \boldsymbol{c}'^{(t)T}\boldsymbol{h}^{(t)} - \left(\frac{\boldsymbol{v}^{(t)}}{\boldsymbol{\sigma}^2}\right)^T \boldsymbol{W}_{vh}\boldsymbol{h}^{(t)}$$

$$\boldsymbol{b}'^{(t)} = \boldsymbol{b} + \sum_p \boldsymbol{W}_{v'_p v}^T \boldsymbol{v}^{(t-p)} \tag{6.4}$$

$$\boldsymbol{c}'^{(t)} = \boldsymbol{c} + \sum_p \boldsymbol{W}_{v'_p h}^T \boldsymbol{v}^{(t-p)}. \tag{6.5}$$

We obtain the following partial differential equations to the log-likelihood $\mathcal{L}$:

$$\frac{\partial \mathcal{L}}{\partial (\boldsymbol{W}_{v'_p v})_{i'i}} = \langle \frac{v_i^{(t)} v_{i'}^{(t-p)}}{\sigma_i^2} \rangle_{data} - \langle \frac{v_i^{(t)} v_{i'}^{(t-p)}}{\sigma_i^2} \rangle_{model} \tag{6.6}$$

$$\frac{\partial \mathcal{L}}{\partial (\boldsymbol{W}_{v'_p h})_{i'j}} = \langle v_{i'}^{(t-p)} h_j^{(t)} \rangle_{data} - \langle v_{i'}^{(t-p)} h_j^{(t)} \rangle_{model}. \tag{6.7}$$

The second terms $\langle\cdot\rangle_{model}$ can be replaced with the reconstructed values $\langle\cdot\rangle_{recon.}$ using CD learning just as RBMs. The other parameters related to the undirected model ($\boldsymbol{W}_{vh}(=\boldsymbol{W})$, $\boldsymbol{b}$, $\boldsymbol{c}$, and $\boldsymbol{\sigma}$ (or $\boldsymbol{z}$)) are also calculated from Eqs. (4.50), (4.51), (4.52), and (4.53) by proper substitution of variables. Once the parameters are estimated, forward inference (the conditional probability of $\boldsymbol{h}^{(t)}$ given $\boldsymbol{v}^{(t)}$ and $\mathcal{V}'^{(t)}$) and backward inference (the conditional probability of $\boldsymbol{v}^{(t)}$ given $\boldsymbol{h}^{(t)}$ and $\mathcal{V}'^{(t)}$) are respectively written as:

$$p(h_j^{(t)}=1|\boldsymbol{v}^{(t)},\mathcal{V}'^{(t)})=\mathcal{S}\left(c_j+\boldsymbol{v}^{(t)^T}\boldsymbol{W}_{vh_{:j}}+\sum_p \boldsymbol{v}^{(t-p)^T}\boldsymbol{W}_{v'_p h_{:j}}\right) \tag{6.8}$$

$$p(v_i^{(t)}=v|\boldsymbol{h}^{(t)},\mathcal{V}'^{(t)})=\mathcal{N}\left(v|b_i+\boldsymbol{h}^{(t)^T}\boldsymbol{W}^T_{vh_{i:}}+\sum_p \boldsymbol{v}^{(t-p)^T}\boldsymbol{W}_{v'_p v_{:j}},\sigma_i^2\right) \tag{6.9}$$

6.2 Methodology

A CRBM is a non-linear probabilistic model used to represent time series data consisting of three factors: (i) an undirected model between binary latent variables and the current visible variables, (ii) a directed model from the previous visible variables to the current visible variables, and (iii) a directed model from the previous visible variables to the latent variables as seen in the previous section. In the proposed approach, we first train two exclusive CRBMs for the source and the target speakers independently using segmented training data prepared for each speaker, then train an NN using the projected features, and finally fine-tune the networks as a single network for VC. Because the training data for the source speaker CRBM include various phonemes particular to the speaker, the speaker-dependent network tries to capture the abstractions to maximally express the training data that have abundant speaker individuality information and less phoneme-related information. Furthermore, the network captures time-series features with the directed models (ii) and (iii), enabling it to discover temporal correlations at the same time. Therefore, it is expected that if feature conversion is conducted in such time-related individuality-emphasized high-order spaces, it is much easier to convert voice features than if the original spectrum-based space is used.

Figure 6.2 shows an overview of the proposed voice conversion system where $P=1$ is used. In the proposed approach, we independently train CRBMs for each speaker beforehand as shown in Figure 6.2(a). Variables $\boldsymbol{x}^{(t)}$ and $\boldsymbol{y}^{(t)}$ ($\boldsymbol{x}^{(t-1)}$ and $\boldsymbol{y}^{(t-1)}$) represent acoustic feature vectors (e.g. visible units in a CRBM), such as MFCC, at frame t (at frame $t-1$) for a source speaker and a target speaker, respectively.

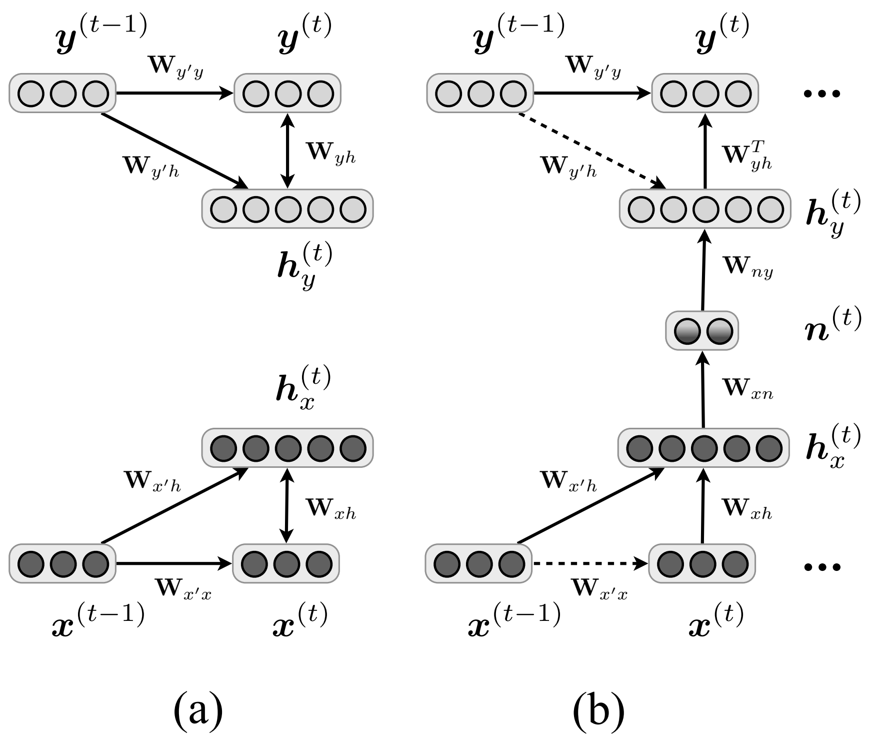


Figure 6.2: A flow chart of the proposed voice conversion system. (a) CRBMs for a source speaker (below) and a target speaker (above), (b) the proposed voice conversion architecture combining the two pre-trained speaker-dependent CRBMs with a concatenating NN.

For the source speaker, for instance, the parameter matrix $\boldsymbol{W}_{xh}$ is, along with $\boldsymbol{W}_{x'h}$ and $\boldsymbol{W}_{x'x}$, estimated so as to maximize the probability of T chained training samples $p(\boldsymbol{x}) = \prod_{t=1}^{T} p(\boldsymbol{x}^{(t)}|\boldsymbol{x}^{(t-1)})$. Using these matrices, an input vector $\boldsymbol{x}^{(t)}$ at frame t given the previous vector $\boldsymbol{x}^{(t-1)}$ is projected into the speaker-dependent latent space that captures speaker-individualities. The latent features $\boldsymbol{h}_x^{(t)}$ can be calculated using mean-field approximation as follows:

$$\boldsymbol{h}_x^{(t)} = \mathcal{S}\left(\boldsymbol{W}_{xh}\boldsymbol{x}^{(t)} + \boldsymbol{W}_{x'h}\boldsymbol{x}^{(t-1)} + \boldsymbol{c}_x\right) \tag{6.10}$$

from Eq. (7.8), where $\boldsymbol{c}_x$ is a bias vector of forward inference for the source speaker. Because each unit in the hidden vector $\boldsymbol{h}_x^{(t)}$ is independent from the others (due to the nature of RBM), it captures the *common* characteristics in the visible units. The training data usually include various phonemes and unvarying speaker-specific features; thus, we expect that the extracted features in $\boldsymbol{h}_x^{(t)}$ represent speaker-individual information. Since we estimate the time-related matrices $\boldsymbol{W}_{x'h}$ and $\boldsymbol{W}_{x'x}$ jointly with the static term $\boldsymbol{W}_{xh}$ as shown in Eq. (6.3) using the training data, they capture time-related information and $\boldsymbol{W}_{xh}$ can focus on capturing other static information. This means that the obtained features in the hidden units $\boldsymbol{h}_x^{(t)}$ also help to capture time-related speaker-individualities. The above discussion applies to the target speaker, and the hidden vector for the target $\boldsymbol{y}^{(t)}$ is obtained in the same manner as in Eq. (6.10):

$$\boldsymbol{h}_y^{(t)} = \mathcal{S}\left(\boldsymbol{W}_{yh}\boldsymbol{y}^{(t)} + \boldsymbol{W}_{y'h}\boldsymbol{y}^{(t-1)} + \boldsymbol{c}_y\right) \tag{6.11}$$

where $\boldsymbol{c}_y$ is a bias vector for the target speaker.

In the proposed approach, we convert such individuality-emphasized features (from $\boldsymbol{h}_x^{(t)}$ to $\boldsymbol{h}_y^{(t)}$) using a neural network (NN) that has $L+2$ layers (L is the number of hidden layers; typically, L is 0 or 1) as shown in Figure 6.2(b). To train the NN, we use the parallel training set $\{\boldsymbol{x}_t, \boldsymbol{y}_t\}_{t=0}^{T'}$ where T' is the number of frames of the parallel data[1]. During the training stage of the NN, the projected vectors of the source speaker's acoustic features $\boldsymbol{h}_x^{(t)}$ are used as inputs, and the projected vectors of the corresponding target speaker's features $\boldsymbol{h}_y^{(t)}$ are used as outputs. Weight parameters of the NN $\{\boldsymbol{W}_l, \boldsymbol{d}_l\}_{l=0}^{L}$ are estimated to minimize the error between the output $\eta(\boldsymbol{h}_x^{(t)})$ and the target vector $\boldsymbol{h}_y^{(t)}$ as is typical for a NN. Once the weight parameters are estimated, an input vector $\boldsymbol{h}_x^{(t)}$ is converted to:

$$\eta(\boldsymbol{h}_x^{(t)}) = \bigodot_{l=0}^{L} \eta_l(\boldsymbol{h}_x^{(t)}) \tag{6.12}$$

$$\eta_l(\boldsymbol{h}_x^{(t)}) = \mathcal{S}(\boldsymbol{W}_l\boldsymbol{h}_x^{(t)} + \boldsymbol{d}_l), \tag{6.13}$$

where $\bigodot_{l=0}^{L}$ denotes the composition of $L+1$ functions as seen in the previous chapter.

To convert the output of the NN to the acoustic features of the target speaker, we simply use backward inference of a CRBM using Eq. (7.9), resulting in:

$$p(\boldsymbol{y}^{(t)}|\boldsymbol{h}_y^{(t)}, \boldsymbol{y}^{(t-1)}) = \mathcal{S}(\boldsymbol{W}_{yh}^{T}\boldsymbol{h}_y^{(t)} + \boldsymbol{W}_{y'y}\boldsymbol{y}^{(t-1)} + \boldsymbol{b}_y) \tag{6.14}$$

where $\boldsymbol{b}_y$ is a bias vector of backward inference for the target speaker.

Generalizing and summarizing the above discussion, a voice conversion function of the proposed method from a source acoustic vector $\boldsymbol{x}^{(t)}$ to a target vector $\boldsymbol{y}^{(t)}$ at frame t, given the previous vectors $\mathcal{X}'^{(t)} = \{\boldsymbol{x}^{(t-p)}\})_{p=1}^{P}$ and $\mathcal{Y}'^{(t)} = \{\boldsymbol{y}^{(t-p)}\}_{p=1}^{P}$, is written as:

$$\boldsymbol{y}^{(t)} = \bigodot_{k=0}^{L+2} \mathcal{S}\big(\boldsymbol{W}_{(k)}\boldsymbol{x}^{(t)} + \boldsymbol{a}_{(k)}(\mathcal{X}'^{(t)}, \mathcal{Y}'^{(t)})\big) \tag{6.15}$$

[1]For sake of simplicity, we used the same parallel data for both training of the CRBMs and the NN in the proposed experiments ($T' = T$).

where $\boldsymbol{W}_{(k)}$ and $\boldsymbol{a}_{(k)}(\mathcal{X}'^{(t)}, \mathcal{Y}'^{(t)})$ denote elements of a set of the proposed model parameters $\Theta = \{\mathcal{W} \cup \mathcal{A}\}$:

$$\begin{aligned}
\mathcal{W} &= \{\boldsymbol{W}_{(k)}\}_{k=0}^{L+2} && (6.16)\\
&= \{\boldsymbol{W}_{xh}, \boldsymbol{W}_0, \cdots, \boldsymbol{W}_L, \boldsymbol{W}_{yh}{}^T\} && (6.17)\\
\mathcal{A} &= \{\boldsymbol{a}_{(k)}(\mathcal{X}'^{(t)}, \mathcal{Y}'^{(t)})\}_{k=0}^{L+2} && (6.18)\\
&= \{\boldsymbol{c}_x + \sum_p \boldsymbol{W}_{x'h}\boldsymbol{x}^{(t-p)}, \boldsymbol{d}_0, && (6.19)\\
&\quad \cdots, \boldsymbol{d}_L, \boldsymbol{b}_y + \sum_p \boldsymbol{W}_{y'y}\boldsymbol{y}^{(t-p)}\}. && (6.20)
\end{aligned}$$

The conversion function shown in Eq. (7.13) implies a dynamic model of a $(L+4)$-layer network with sigmoid activated functions. Therefore, regarding it as a recurrent neural network (RNN), we can fine-tune each parameter of the entire network by back-propagation through time (BPTT) using the acoustic parallel data.

As Eq. (7.13) indicates, we need a current acoustic vector from a source speaker, and previous vectors from both a source speaker and a target speaker to estimate the target speaker's current acoustic vector. However, we never know the correct previous vector of the target speaker, so in practice, we use the last converted (estimated) vectors as the previous target vector iteratively, starting from a zero vector. We confirmed that this approach worked well through the preliminary experiments.

Again, when we compare the proposed method with a conventional GMM-based approach [95], while the GMM method is an additive model of piecewise linear functions, the proposed approach is based on the composite function of multiple different non-linear functions feeding time-series data. Therefore, it is expected that the proposed composite model can represent more complex relationships than the conventional GMM-based method and other static network approaches [21, 30].

6.3 Related work

It is worth noting that we compare the proposed method with the conventional method proposed by Wu et al. in [1], that also employed a conventional restricted Boltzmann machine (CRBM) for VC. Figure 6.3 shows the comparison of graphical models among three methods. Wu's method directly uses CRBM to estimate the target features $\boldsymbol{y}^{(t)}$ from the input $\boldsymbol{x}^{(t)}$ along with the latent features $\boldsymbol{h}^{(t)}$ to capture the linear and non-linear relationship between the source and the target features (Figure 6.3(b)). On the other hand, the proposed method

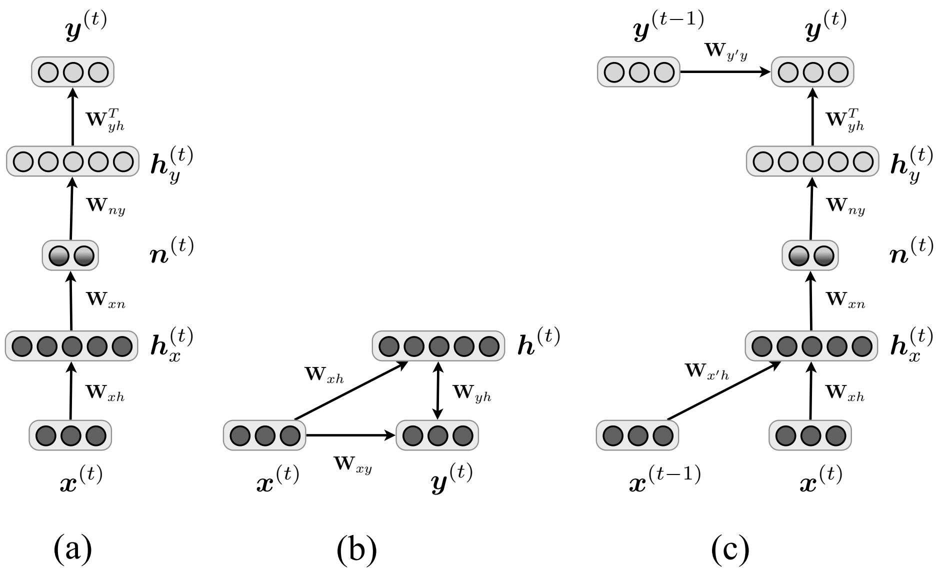


Figure 6.3: Model structures of the related systems. (a) speaker-dependent-RBM model discussed in Chapter 5, (b) CRBM proposed in [1], and (c) the proposed method, speaker-dependent CRBM.

(Figure 6.3(c)) uses two CRBMs for each of the source and the target speakers to obtain their latent features $\boldsymbol{h}_x^{(t)}$ and $\boldsymbol{h}_y^{(t)}$, capturing time-related information (from $t-1$ to t frames). Connecting the latent features using a neural network (NN), the entire conversion network of the proposed method consequently forms a deep architecture. The previous previous approach [30] discussed in Section 5 has a deep network similar to that of the proposed method (Figure 6.3(a)); however, the difference is that it involves time-related relationships in the network.

Since the acoustic signals we are targeting are time-series data, the model that captures time-related information will provide us with the better performance.

6.4 Experiments

6.4.1 Conditions

In the experiments, we conducted voice conversion using the ATR Japanese speech database [90], comparing the proposed method (speaker-dependent restricted Boltzmann machines; say "SD-CRBM") with four methods: the well-known GMM-based approach ("GMM"), conventional NN-based voice conversion [21] ("NN"), the previous approach (speaker-dependent RBM) appeared in

Chapter 5 ("SD-RBM"), and, for a reference, a recurrent neural network with randomly-initialized weights ("RNN"). In order to evaluate the proposed method under various circumstances, we tested male-to-female (the source and the target speakers are identified with MMY and FTK in the database, respectively), female-to-female (FKN and FTK), and male-to-male (MMY and MHT) patterns.

For an input vector, 24-dimensional MFCC features were calculated from STRAIGHT spectra [48] using filter-theory [43] to decode the MFCC back to STRAIGHT spectra in the synthesis stage. Unlike the previous work [30], we processed the obtained MFCC with ZCA (zero component analysis) whitening [88], where we confirmed it worked better than without whitening, especially for "NN". The parallel data of the source/target speakers processed by Dynamic Programming were created from 216 word utterances in the dataset, and were used for the training of each method. (Note that two CRBMs for "SD-CRBM" and two RBMs for "SD-RBM" can be trained without the necessity of using parallel data, although we used the same parallel training data for the CRBMs and the RBMs in this research.)

The network-based approaches ("SD-CRBM", "SD-RBM", "NN" and "RNN") were trained using gradient descent with a learning rate of 0.01 and momentum of 0.9, with the number of epochs being 400. Other configurations, such as the number of hidden units, will be discussed in the following section.

For the objective test, 15 sentences (about 60 sec. long) that were not included in the training data were arbitrarily selected from the database (identified with SDA01$\sim$SDA15). For the objective evaluation, MCD was used to measure how close the converted vector is to the target vector in mel-cepstral space. We calculated the MCD for each frame in the training data, and averaged the MCD values for the final evaluation.

For the subjective evaluation, ABX listening tests were conducted, where nine participants listened to pairs of converted speech signals produced using the proposed approach ("SD-CRBM") and the converted speech signals produced by the other methods ("SD-RBM", "NN", "RNN", and "GMM") along with an original target speech signal (generated from analysis-by-synthesis) They then selected the *better* one in terms of speaker identity (how well they can recognize the speaker from the converted speech) and speech quality (how clear and natural the converted speech is).

6.4.2 Determining appropriate parameters

In this section, we report preliminary experiments in which we tested various models with different hyper parameters to determine the appropriate ones. All models were trained using $N = 20,000$ frames from the male-to-female training data, and evaluated using a development set of 5 sentences (identified with

SDA16∼SDA20 in the database) that were not included in either the training set or the test set.

6.4.2.1 Network-based methods

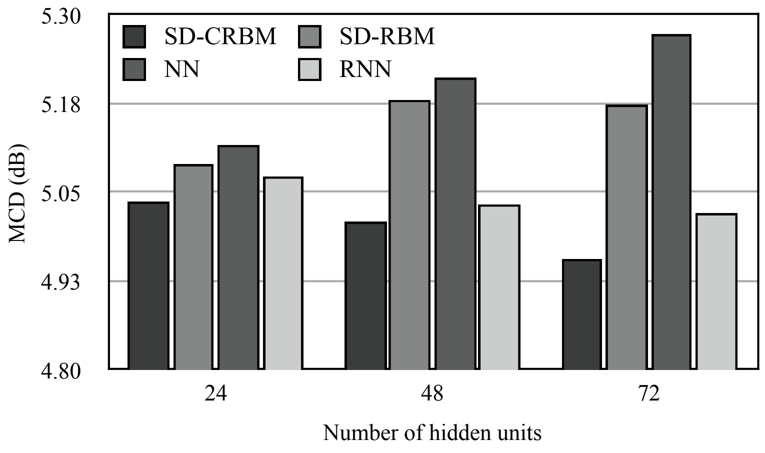


Figure 6.4: Changing hidden units. The values show averaged mel-cepstral distortion with varying numbers of hidden units J for all network-based methods ($N = 20,000$).

Here we will see how the proposed approach works as the number of hidden units J in each hidden layer changes, comparing it to four network-based methods ("SD-CRBM", "SD-RBM", "NN" and "RNN"). In this preliminary experiment, three architectural patterns were tested, where J = 24, 48, and 72. We used $L = 0$, which forms a four-layer network for all methods (for example, when $J = 48$ is used, the numbers of units in "NN" from the input/source layer to the output/target layer become 24, 48, 48, and 24 in order). For "SD-CRBM", $P = 1$ is used (1 delay for "RNN" as well), which means we take into account only one previous frame.

Figure 6.4 compares the averaged MCD obtained for each architecture. As shown in Figure 6.4, the proposed method "SD-CRBM" performed best of all the methods for each case. The interesting point is that the more hidden units the network has, the better performance it provides for "SD-CRBM" and "RNN", while it is the other way around for "SD-RBM" and "NN". This is considered to be due to over-fitting to the training data for "SD-RBM" and "NN" when the number of parameters is large (e.g. $J = 72$), while "SD-CRBM" and "RNN" still required parameters to fit the models that capture time-series data.

For the remaining experiments in this paper, the best architectures for each method were used; i.e., $J = 24$ for "SD-RBM" and "NN", and $J = 72$ for "SD-CRBM" and "RNN".

6.4.2.2 GMM-based method

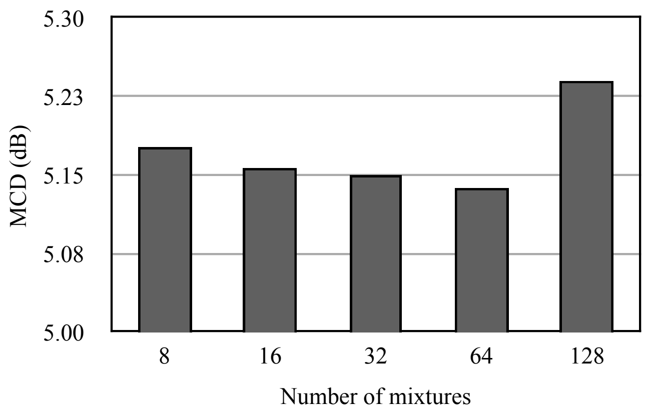


Figure 6.5: Changing mixtures. The values show averaged mel-cepstral distortion with varying numbers of mixtures M for GMM method ($N = 20,000$).

For the GMM-based voice conversion ("GMM"), we tried and evaluated five mixtures (8, 16, 32, 64, 128 mixtures) to determine an appropriate number of mixtures. Figure 6.5 shows the averaged MCDs over the development set when using the GMM with various mixtures. As shown in the figure, the GMM with 64 mixtures performed best of all. Therefore, we used mixtures of 64 for "GMM" in the evaluation experiments described in Section 7.3.3.

6.4.2.3 The number of previous frames

We further investigated the performance of the proposed method "SD-CRBM" with the hidden units of $J = 72$, changing the number of previous frames in the CRBM as $P = 1, 2, 3, 4, 5$. The evaluation results are described in Figure 6.6, showing the averaged MCDs obtained from each case. As shown in Figure 6.6, we could not necessarily obtained a better performance as the number of previous frames increased. One reason is that the neighbor source vectors previous to the current one contained similar information, and only a few source vectors were required to estimate the current target vector. Therefore, the poor performance with the larger number of previous frames (e.g. $P = 4$) was caused because

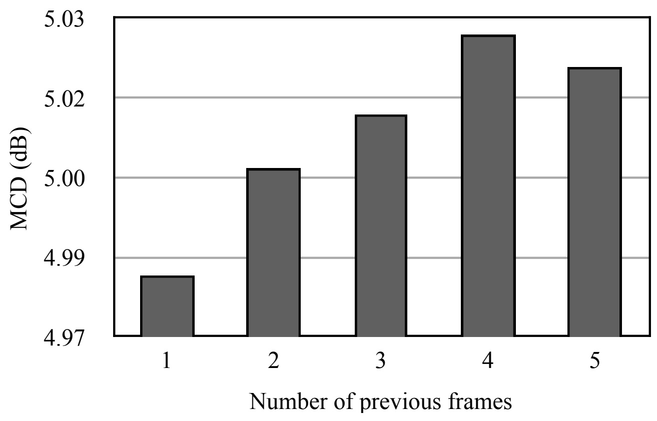


Figure 6.6: Changing previous frames. The values show averaged mel-cepstral distortion with varying numbers of previous frames P to be taken into account for the proposed method ($N = 20,000$).

the parameter estimation became more difficult as the redundant parameters increased.

In the remaining experiments, $P = 1$ was used, which provided the best performance in the preliminary experiment.

6.4.3 Evaluation

In this section, we evaluate the proposed method ("SD-CRBM") comparing it with four methods ("SD-RBM", "NN", "RNN", and "GMM") using objective and subjective criteria for each pair of speakers, by changing the number of training frames as $N = 5,000$, $10,000$, $20,000$, and $40,000$.

6.4.3.1 Results

Figures 6.7, 6.8, and 6.9 summarize the experimental results for the test data, comparing each method with respect to objective criteria for male-to-female, female-to-female, and male-to-male voice conversion, respectively. As shown in these Figures, the MCDs decreased as the number of training data increased in most cases (regardless of the gender or the method). Furthermore, the proposed approach outperformed the other methods in every case, except for the case where $N = 20,000$ in the male-to-male experiment.

Figures 6.10 and 6.11 show the results of subjective evaluation comparing each method in terms of speaker identity and speaker quality, respectively. We also list the p-values produced by pairwise t-testing for each experiment in Tables 6.1 and 6.2, in terms of speaker identity and speech quality, respectively. As shown in Figures 6.10 and 6.11, the proposed method performed better than each opponent method in regard to mean preference score in terms of both speaker identity and speech quality. However, as shown in Tables 6.1 and 6.2, we could not, unfortunately, obtain a significant difference between the proposed method and "NN" w.r.t. speaker identity, and "SD-RBM" and "RNN" w.r.t. speech quality. Significant differences were obtained at a significance level of 0.1 in the other cases.

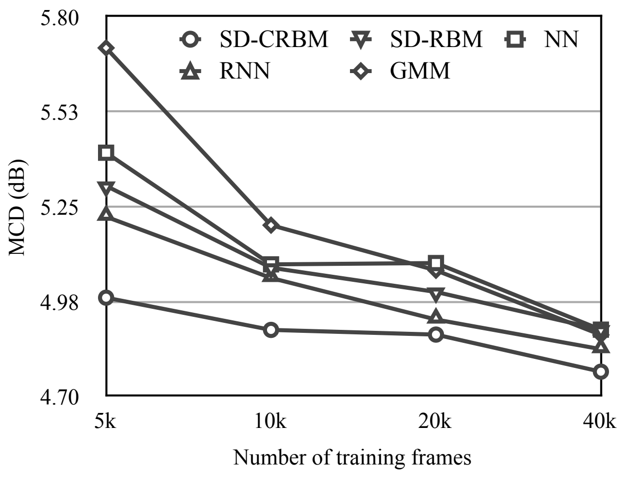


Figure 6.7: Male-to-female voice conversion results. The values show averaged mel-cepstral distortion for each method with varying amounts of training data.

Table 6.1: P-values produced by paired t-test between the proposed method and each method in the subjective evaluation w.r.t. speaker identity (associated with Figure 6.10.) The values that satisfy $p < 0.1$ are in bold.

	SD-RBM	NN	RNN	GMM
p	**0.0913**	0.4417	**0.0913**	**0.0001**

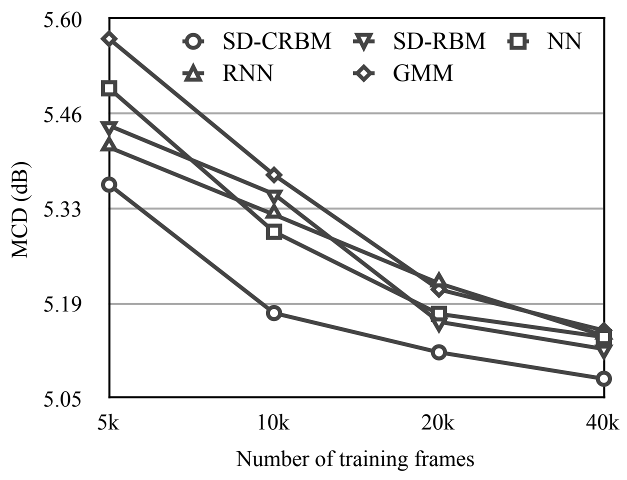


Figure 6.8: Female-to-female voice conversion results. The values show averaged mel-cepstral distortion for each method with varying amounts of training data.

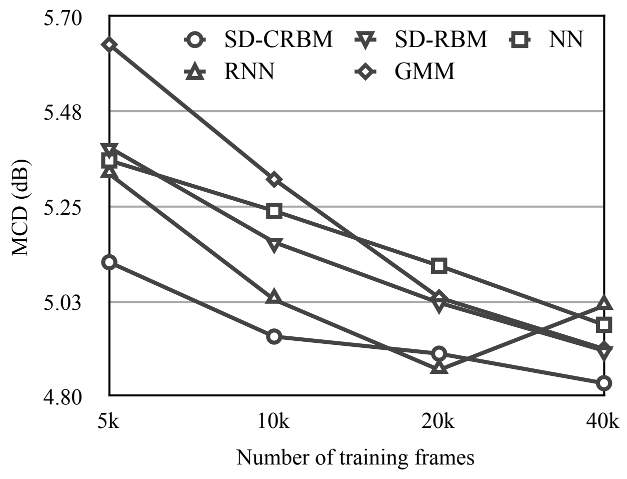


Figure 6.9: Male-to-male voice conversion results. The values show averaged mel-cepstral distortion for each method with varying amounts of training data.

6.4.3.2 Discussion

In objective criteria, the proposed approach ("SD-CRBM") outperformed the other methods, including the popular GMM-based voice conversion method, in most cases. In subjective criteria as well, we obtained significantly better perfor-

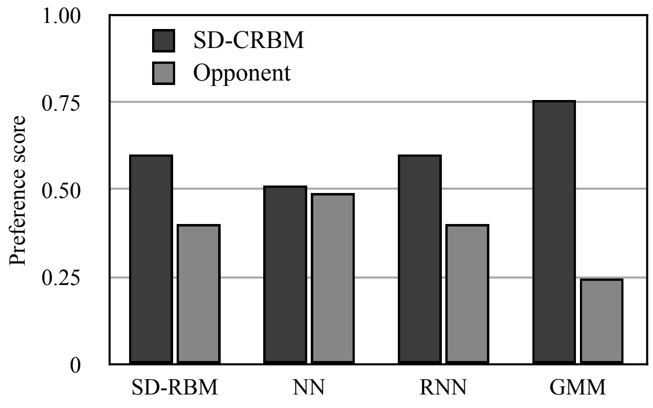


Figure 6.10: Subjective preference scores w.r.t. speaker identity. The proposed method "SD-CRBM" was compared against four methods: "SD-RBM", "NN", "RNN", and "GMM".

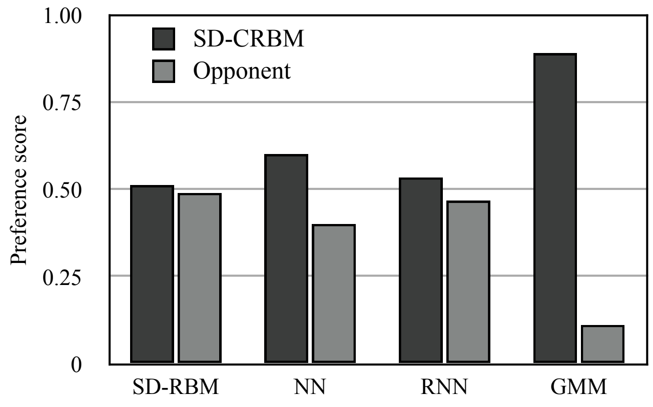


Figure 6.11: Subjective preference scores w.r.t. speech quality. The proposed method "SD-CRBM" was compared against four methods: "SD-RBM", "NN", "RNN", and "GMM".

mance compared with each opponent, in terms of speaker identity and/or speech quality (to be specific, in terms of both speaker identity and speech quality for "GMM", in terms of only speech quality for "NN", in terms of only speaker identity for "SD-RBM" and "RNN"). The reason for the improvement is attributed to the fact that the proposed time-involving, high-order conversion system using

Table 6.2: P-values produced by paired t-test between the proposed method and each method in the subjective evaluation w.r.t. speech quality (associated with Figure 6.11.) The values that satisfy $p < 0.1$ are in bold.

	SD-RBM	NN	RNN	GMM
p	0.4417	**0.0913**	0.3299	**0.0000**

CRBMs is able to capture and convert the abstractions of speaker individualities better than the other methods. In particular, as shown in Figures 6.7, 6.8 and 6.9, the proposed approach achieved high performance in MCD criteria. This is because the CRBMs captured time-series data more appropriately and alleviated estimation errors.

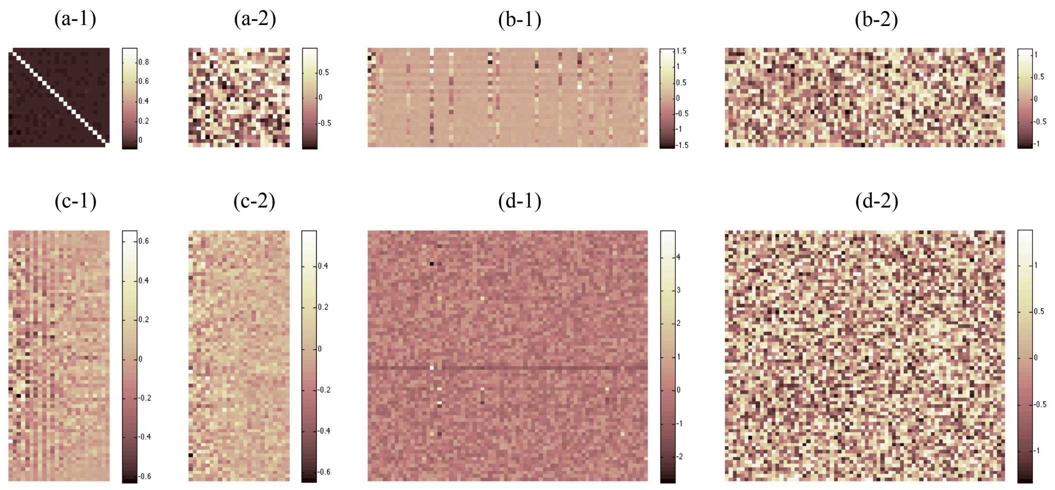


Figure 6.12: Estimated weights of the pre-trained RNN (··1) and the randomly-initialized RNN (··2) after 400 epochs ($N = 40,000, J = 72$, male-to-female). (a-·) The weights from the previous target vector to the current target vector $\boldsymbol{W}_{y'y}$, (b-·) the weights from the second hidden layer to the current target vector $\boldsymbol{W}_{yh}$, (c-·) the weights from the current source vector to the first hidden layer $\boldsymbol{W}_{xh}$, and (d-·) the weights from the first hidden layer to the second hidden layer $\boldsymbol{W}_0$.

One interesting point is that "NN" and "RNN", which were based on random initialization in weight parameters, produced unstable performance (e.g. the

MCD by "NN" increased even as the number of training frames increased from 10,000 to 20,000 in male-to-female conversion, and the MCD by "RNN" also increased as the number of training data changed from 20,000 to 40,000 in male-to-male conversion). This is caused by a fall into local minima starting from the randomly-initialized weights. Figure 6.12 shows some of the converged weights in the network, comparing "RNN" and "SD-CRBM", where the weights were pre-trained using speaker-dependent CRBMs and a concatenating NN followed by fine-tuning using RNN. As shown in Figure 6.12, the weights in "RNN" were almost meaningless and messy; meanwhile, we see that the weights in "SD-CRBM" had a sparse structure and operative bases. In general, an acoustic feature vector at the last previous frame ($\boldsymbol{v}^{(t-1)}$) is very similar to the feature vector at the current frame ($\boldsymbol{v}^{(t)}$), and, therefore, we expect that the conversion matrix from $\boldsymbol{v}^{(t-1)}$ to $\boldsymbol{v}^{(t)}$ may be close to an identity matrix. The recurrent weight obtained by the proposed approach shown in Figure 6.12(a-1) indicates this fact.

6.5 Summary

We presented in this chapter a voice conversion method that combines speaker-dependent conditional restricted Boltzmann machines (CRBMs) and a NN to extract speaker-individual information for speech conversion. Through experiments, we confirmed that the proposed approach is effective, especially in terms of MCD, compared with the well-known conventional Gaussian mixture model (GMM)-based approach, a neural network (NN)-based approach, and the previous work discussed in Chapter 5, speaker-dependent restricted Boltzmann machine (SD-RBM), (and recurrent neural network for a reference), regardless of the gender in conversion.

We also conducted ABX experiments for subjective evaluation. The results showed that the performance of the proposed method was not always significantly different in comparison to NN, RNN, and SD-RBM; however, it did perform significantly better than these methods in terms of either speaker identity or speech quality. In the future, we will work to improve the proposed method so that it obtains better results in regard to the sense of hearing.

Chapter 7

VC Using Speaker-Dependent RTRBMs

This chapter also presents a VC method as an extension of the speaker-dependent-RBM model in Chapter 5 to consider capturing time-related information, similar to the speaker-dependent-CRBM model in Chapter 6. The proposed VC method uses the less phonological and more unique features obtained from speaker-dependent recurrent temporal restricted Boltzmann machines (RTRBMs), a family of RBMs.

7.1 Recurrent Temporal Restricted Boltzmann Machine (RTRBM)

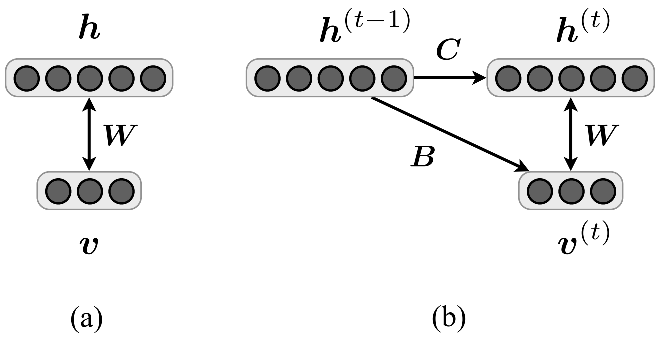


Figure 7.1: Comparison of graphical representation of a standard RBM (a) and an RTRBM (b).

An RTRBM is an extended RBM proposed by Sutskever *et al.* [37], and is suitable for capturing and modeling temporal dependencies in sequence data. Like an RBM, an RTRBM uses an undirected model. As shown in Figure 7.1, an RTRBM also employs directed models from previous hidden units $\boldsymbol{h}^{(t-1)} = [h_1^{(t-1)}, \cdots, h_J^{(t-1)}]^T, h_j^{(t-1)} \in \{0,1\}$ to current hidden units $\boldsymbol{h}^{(t)} = [h_1^{(t)}, \cdots, h_J^{(t)}]^T, h_j^{(t)} \in \{0,1\}$ and current visible units $\boldsymbol{v}^{(t)} = [v_1^{(t)}, \cdots, v_I^{(t)}]^T, v_i^{(t)} \in \mathbb{R}$ at the current frame t. In this model, there are three types of parameters to be estimated: $\boldsymbol{B} \in \mathbb{R}^{I\times J}$ (a directed weight matrix from $\boldsymbol{h}^{(t-1)}$ to $\boldsymbol{v}^{(t)}$), $\boldsymbol{C} \in \mathbb{R}^{J\times J}$ (a directed weight matrix from $\boldsymbol{h}^{(t-1)}$ to $\boldsymbol{h}^{(t)}$), and $\boldsymbol{W} \in \mathbb{R}^{I\times J}$ (an undirected weight matrix between $\boldsymbol{v}^{(t)}$ and $\boldsymbol{h}^{(t)}$). Like with an RBM, the weights are estimated using contrastive divergence by maximizing the log-likelihood $\mathcal{L} = \log \prod_t p(\boldsymbol{v}^{(t)}|\mathcal{A}^{(t)})$ denoted by $\mathcal{A}^{(t)} = \{\boldsymbol{v}^{(\tau)}, \boldsymbol{h}^{(\tau)}|\tau < t\}$, where

$$p(\boldsymbol{v}^{(t)}|\mathcal{A}^{(t)}) = \frac{1}{Z}\sum_{\boldsymbol{h}^{(t)}} e^{-E(\boldsymbol{v}^{(t)},\boldsymbol{h}^{(t)}|\boldsymbol{h}^{(t-1)})}. \tag{7.1}$$

Considering the improvement in the training as in improved GB-RBM, the energy function of the RTRBM E becomes

$$\begin{aligned}
&E(\boldsymbol{v}^{(t)},\boldsymbol{h}^{(t)}|\boldsymbol{h}^{(t-1)}) \\
&= \left\|\frac{\boldsymbol{v}^{(t)}-\boldsymbol{b}^{(t)}}{2\boldsymbol{\sigma}}\right\|^2 - \boldsymbol{c}^{(t)T}\boldsymbol{h}^{(t)} - \left(\frac{\boldsymbol{v}^{(t)}}{\boldsymbol{\sigma}^2}\right)^T \boldsymbol{W}\boldsymbol{h}^{(t)} \qquad (7.2)\\
\boldsymbol{b}^{(t)} &= \boldsymbol{b} + \boldsymbol{B}\boldsymbol{h}^{(t-1)} \qquad (7.3)\\
\boldsymbol{c}^{(t)} &= \boldsymbol{c} + \boldsymbol{C}\boldsymbol{h}^{(t-1)}. \qquad (7.4)
\end{aligned}$$

The previous hidden units $\boldsymbol{h}^{(t-1)}$ in Eqs. (7.3) and (7.4) are replaced with the mean-field values $\hat{\boldsymbol{h}}^{(t-1)}$ as follows:

$$\hat{\boldsymbol{h}}^{(t-1)} = \mathcal{S}(\boldsymbol{c}^{(t-1)} + \boldsymbol{W}^T(\frac{\boldsymbol{v}^{(t-1)}}{\boldsymbol{\sigma}^2})) \tag{7.5}$$

since this approach improves the efficiency of training [37]. In this thesis, for the initial values $\boldsymbol{h}^{(0)}$, a zero vector is used.

We obtain the following partial differential equations for the log-likelihood:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{B}_{ij}} = \langle \frac{v_i^{(t)}\hat{h}_j^{(t-1)}}{\sigma_i^2}\rangle_{data} - \langle \frac{v_i^{(t)}\hat{h}_j^{(t-1)}}{\sigma_i^2}\rangle_{model} \tag{7.6}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{C}_{j'j}} = \langle \hat{h}_{j'}^{(t-1)}\hat{h}_j^{(t)}\rangle_{data} - \langle \hat{h}_{j'}^{(t-1)}\hat{h}_j^{(t)}\rangle_{model}. \tag{7.7}$$

The other parameters related to the undirected model ($\boldsymbol{W}$, $\boldsymbol{b}$ and $\boldsymbol{c}$) are also calculated from Eqs. (4.50), (4.51) and (4.52) by proper substitution of variables.

The second terms in Eqs. (7.6) and (7.7) are computed as the reconstructed values in a way similar to that in an RBM.

After the parameters are estimated, forward inference (the conditional probability of $\boldsymbol{h}^{(t)}$ given $\boldsymbol{v}^{(t)}$ and $\boldsymbol{h}^{(t-1)}$) and backward inference (the conditional probability of $\boldsymbol{v}^{(t)}$ given $\boldsymbol{h}^{(t)}$ and $\boldsymbol{h}^{(t-1)}$) can be written as follows, respectively:

$$p(h_j^{(t)} = 1|\boldsymbol{v}^{(t)}, \boldsymbol{h}^{(t-1)}) = \mathcal{S}(c_j^{(t)} + \boldsymbol{W}_{:j}^{T}(\frac{\boldsymbol{v}^{(t)}}{\boldsymbol{\sigma}^2})) \tag{7.8}$$

$$p(v_i^{(t)} = v|\boldsymbol{h}^{(t)}, \boldsymbol{h}^{(t-1)}) = \mathcal{N}(v|b_i^{(t)} + \boldsymbol{W}_{i:}\boldsymbol{h}^{(t)}, \sigma_i^2). \tag{7.9}$$

7.2 Methodology

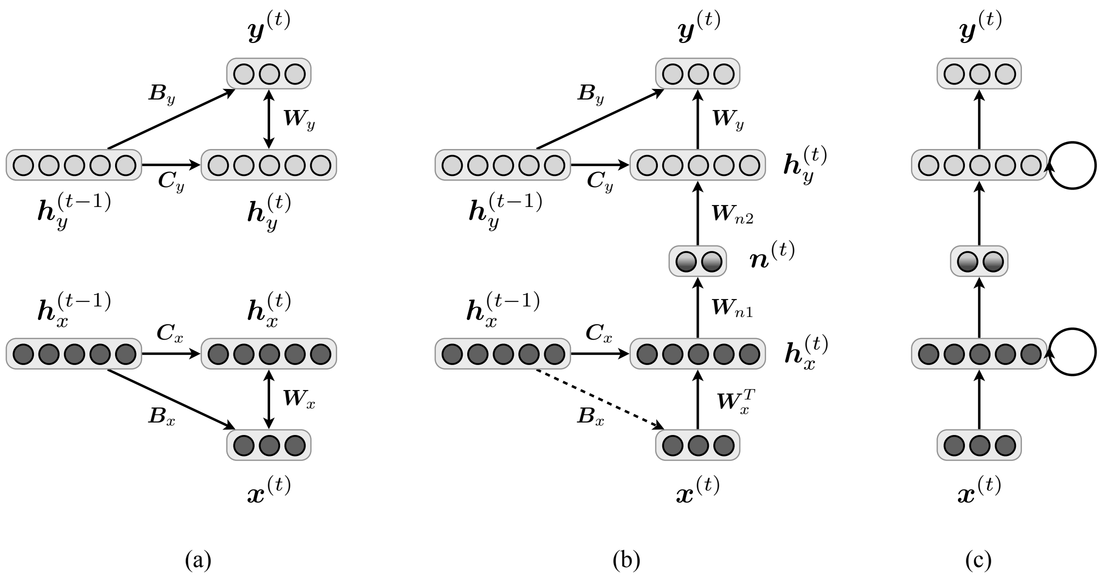


Figure 7.2: (a) RTRBMs for a source speaker (below) and a target speaker (above), (b) the proposed voice conversion architecture, which combines two speaker-dependent RTRBMs with an NN, (c) an alternative representation of (b) that can be regarded as a recurrent NN.

The proposed method extends the previous model discussed in Chapter 5 to systematically capture time information as well as latent (deep) relationships between source-speaker and target-speaker features in a single network. This can be done by combining speaker-dependent recurrent temporal restricted Boltzmann machines (RTRBMs) [37] and a concatenating NN. An RTRBM is a non-linear probabilistic model used to capture temporal dependencies in time-series data as seen in the above section. Despite its simplicity, this model does a good job of describing meaningful sequences such as video [37] and music [38]. In the proposed approach, we first train two RTRBMs: one exclusively for the source and one exclusively for the target speakers. We train them independently using segmented training data prepared for each speaker. Then we train an NN using the projected features. Finally we fine-tune the networks as a single recurrent NN. Because the training data for the source speaker RTRBM includes various phonemes particular to the speaker, the speaker-dependent network tries to capture the abstractions to maximally express the training data with abundant speaker-specific information and less phonological information. Furthermore, the network receives a collection of time-series feature vectors with the conditional models, enabling it to discover temporal correlations in the high-order space. The speaker-dependent CRBM model discussed in Chapter 6 was also introduced to capture time-related information. The most significant difference is that while the CRBM model inputs multiple-frame features of the source speaker, the proposed (RTRBM) model inputs only one frame features.

Figure 7.2 shows an overview of the proposed voice conversion system. Figure 7.2(a) shows how we independently train RTRBMs for each speaker beforehand. Variables $\boldsymbol{x}^{(t)}$ and $\boldsymbol{y}^{(t)}$ ($\boldsymbol{x}^{(t-1)}$ and $\boldsymbol{y}^{(t-1)}$) are acoustic feature vectors (i.e., visible units in RTRBM), such as mel-frequency cepstral coefficients (MFCC), at frame t (at frame $t-1$) for a source speaker and a target speaker, respectively.

For the source speaker, for example, the parameter matrices $\boldsymbol{W}_x$, $\boldsymbol{B}_x$, and $\boldsymbol{C}_x$ are estimated so as to maximize the probability of a T-time sequence $p(\boldsymbol{x}) = \prod_t^T p(\boldsymbol{x}^{(t)}|\mathcal{A}^{(t)})$. Because each unit in the hidden vector $\boldsymbol{h}_x^{(t)}$ is independent from the others, the units capture the *common* characteristics in the visible units. The training data usually include various phonemes and unvarying speaker-specific features; thus, it is expected that the extracted features in $\boldsymbol{h}_x^{(t)}$ emphasize speaker-specific information. Furthermore, because we estimate the time-related matrices $\boldsymbol{B}_x$ and $\boldsymbol{C}_x$ jointly with the static term $\boldsymbol{W}_x$ as shown in Eq. (7.2) using the training data, the matrices capture time-related information. This means that the obtained features in the hidden units $\boldsymbol{h}_x^{(t)}$ also help to capture time-related features that are unique to speakers. An input vector $\boldsymbol{x}^{(t)}$ at frame t is projected into such the speaker-dependent latent space that captures speaker-individualities. In this paper, the latent features $\boldsymbol{h}_x^{(t)}$ are obtained using mean-field approximation as in Eq. (7.8). The above discussion applies to the target speaker, and the hidden vector for the target $\boldsymbol{h}_y^{(t)}$ is obtained in the same manner. In the proposed approach, we convert such speaker-specific-information-emphasized features (from $\boldsymbol{h}_x^{(t)}$ to $\boldsymbol{h}_y^{(t)}$) using an NN with $L+2$ layers (L denotes the number of hidden layers; typically, L is 0 or 1), as shown in Fig. 7.2(b). To train the NN, we use the parallel training set $\{\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}\}_{t=0}^{T'}$ where T' denotes the number of frames of the parallel data.[1] During the training stage of the NN, the projected vectors of the source speaker's acoustic features $\boldsymbol{h}_x^{(t)}$ are the inputs, and the projected vectors of the corresponding target speaker's features $\boldsymbol{h}_y^{(t)}$ are outputs. Weight parameters of the NN $\{\boldsymbol{W}_l, \boldsymbol{d}_l\}_{l=0}^{L}$ are estimated to minimize the error between the output $\eta(\boldsymbol{h}_x^{(t)})$ and the target vector $\boldsymbol{h}_y^{(t)}$ as is typical for a NN. After the weight parameters are estimated, an input vector $\boldsymbol{h}_x^{(t)}$ is converted as follows:

$$\eta(\boldsymbol{h}_x^{(t)}) = \bigodot_{l=0}^{L} \eta_l(\boldsymbol{h}_x^{(t)}) \tag{7.10}$$

$$\eta_l(\boldsymbol{h}_x^{(t)}) = \mathcal{S}(\boldsymbol{W}_l \boldsymbol{h}_x^{(t)} + \boldsymbol{d}_l). \tag{7.11}$$

To convert the output of the NN to the acoustic features of the target speaker, backward inference of an RTRBM is simply used from Eq. (7.9).

Summarizing the above discussion, a voice conversion function of the proposed method from a source acoustic vector $\boldsymbol{x}^{(t)}$ to a target vector $\boldsymbol{y}^{(t)}$ at frame t is written as:

$$\begin{aligned}\boldsymbol{y}^{(t)} &= \operatorname*{argmax}_{\boldsymbol{y}^{(t)}} p(\boldsymbol{y}^{(t)} | \boldsymbol{x}^{(t)}, \boldsymbol{h}_x^{(t-1)}, \boldsymbol{h}_y^{(t-1)}) && (7.12)\\ &= \boldsymbol{a}_{L+2}^{(t)} + \boldsymbol{W}_{L+2} \bigodot_{k=0}^{L+1} \mathcal{S}(\boldsymbol{a}_k^{(t)} + \boldsymbol{W}_k \boldsymbol{x}^{(t)}) && (7.13)\end{aligned}$$

where $\boldsymbol{a}_k^{(t)}$ and $\boldsymbol{W}_k$ denote elements of a set of dynamic parameters $\Theta^{(t)} = \{\boldsymbol{a}^{(t)}, \boldsymbol{W}\}$:

$$\begin{aligned}\boldsymbol{a}^{(t)} &= \{\boldsymbol{a}_k^{(t)}\}_{k=0}^{L+2} = \{\boldsymbol{c}_x^{(t)}, \boldsymbol{d}_0, \cdots, \boldsymbol{d}_L, \boldsymbol{b}_y^{(t)}\} && (7.14)\\ \boldsymbol{W} &= \{\boldsymbol{W}_k\}_{k=0}^{L+2} = \{\boldsymbol{W}_x^T, \boldsymbol{W}_0, \cdots, \boldsymbol{W}_L, \boldsymbol{W}_y\}, && (7.15)\end{aligned}$$

where $\boldsymbol{c}_x^{(t)}$ and $\boldsymbol{b}_y^{(t)}$ denote a forward-inference bias vector in a source speaker's RTRBM and a backward-inference bias vector in the target speaker's RTRBM obtained from Eqs. (7.4) and (7.3), respectively. $\boldsymbol{h}_x^{(0)}$ and $\boldsymbol{h}_y^{(0)}$ are zero vectors. The conversion function shown in Eq. (7.13) implies an $(L+4)$-layer recurrent NN with sigmoid activated functions as shown in Fig. 7.2(c). Therefore, we can fine-tune each parameter of the network of two RTRBMs and the NN by back-propagation through time (BPTT) using acoustic parallel data.

[1]For sake of simplicity, the same parallel data is used for both training of the RTRBMs and the NN in the experiments ($T' = T$).

7.3 Experiments

7.3.1 Conditions

In the VC experiments, we compared the proposed method (SD-RTRBM) with three conventional methods: a well-known GMM-based approach (GMM), an NN-based approach (NN) and the previous model [30] (in Chapter 5), which utilized speaker-dependent RBMs for pre-training of the NN (SD-RBM). Here we used a single-layer DBN (i.e., an RBM) for each speaker and compared these methods. All of the network-based methods (SD-RTRBM, NN, SD-RBM) contained four layers with various numbers of hidden units as discussed in the following section. The network-based methods were trained with a learning rate of 0.01 and momentum of 0.9, with the number of epochs being 400, using acoustic features from the ATR Japanese speech database [90]. The parameters of the proposed method and SD-RBM were fine-tuned after the training of the RTRBMs and RBMs, respectively. In order to evaluate the proposed method under various circumstances, we tested male-to-female (the source and the target speakers are identified with "MMY" and "FTK" in the database, respectively), female-to-female ("FKN" and "FTK"), and male-to-male ("MMY" and "MHT") patterns. 24-dimensional MFCC features were used as an input vector, calculated from STRAIGHT spectra [48] using filter-theory [43] to decode the MFCC back to STRAIGHT spectra in the synthesis stage. Unlike the previous model (speaker-dependent RBM), we processed the obtained MFCC with ZCA (zero component analysis) whitening [88]. Parallel data from source and target speakers processed by dynamic programming was obtained from 216 word utterances in the dataset. This data was used for training. Note that the parallel data was prepared for the NN and GMM methods, and two speaker-wise RTRBMs were trained independently. For the objective test, 15 sentences that were not included in the training data were arbitrarily selected from the database. For objective evaluation, MCD (mel-cepstral distortion) was used to measure how close the converted vector was to the target vector in the mel-cepstral space. We calculated the MCD for each frame in the training data, and averaged them for the final evaluation.

For subjective evaluation, MOS (mean opinion score) listening tests were conducted using five sentences. For the tests, seven participants listened to the original target speech (generated from analysis-by-synthesis) and converted speech for each method, and then selected how close the converted speech sounded to the original speech on a 5-point scale (5: excellent; 4: good; 3: fair; 2: poor; and 1: bad).

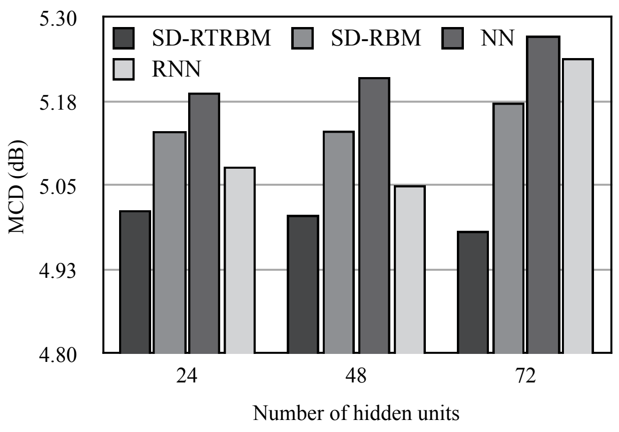


Figure 7.3: Average MCD with changing the number of hidden units for each network-based method.

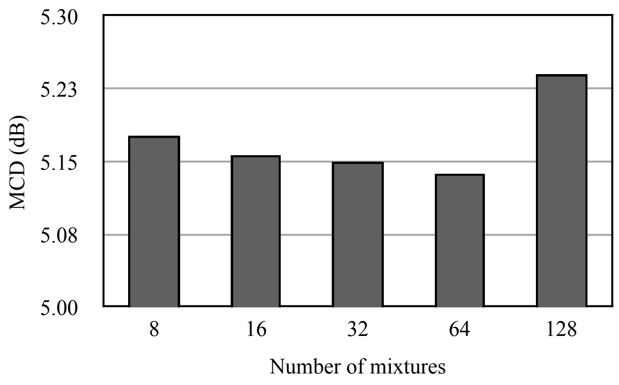


Figure 7.4: Average MCD with varying numbers of mixtures for GMM method.

7.3.2 Determination of Hyper Parameters

Determining the number of hidden units in the network-based approaches and the number of mixtures in the GMM-based approach is important for a fair comparison. For reference, we also compared the proposed method with a recurrent neural network (RNN), whose parameters were randomly initialized with the same architecture as the proposed method. All models were trained using $T = 20,000$ frames from the male-to-female training data, and evaluated using a development set of five sentences (identified with SDA16–20 in the database) that were not included in either the training set or the test set.

7.3.2.1 Network-based methods

In the first experiments, we used 24, 48, and 72 hidden units for the network-based approaches and checked the performance of each method. Each network-based method has a four-layer architecture; for example, the 48-unit NN has 24, 48, 48, and 24 units from the input layer to the output layer. Fig. 7.3 depicts the average MCD obtained from each method, showing that the wider architecture (such as "72") does not always provide better results than narrower architectures except when the proposed method is used. It is expected that this is due to overfitting to the training data for NN and SD-RBM (though SD-RBM did not overfit as much as NN) when the number of hidden units is large (e.g., $J = 72$). RNN performed poorly, because it is based on random-initialization, while the proposed method performed well due to the hierarchical learning algorithm.

For the remaining experiments in this paper, the best number of hidden units for each method were used; i.e., $J = 24$ for "SD-RBM" and "NN," $J = 48$ for "RNN," and $J = 72$ for "SD-RTRBM."

7.3.2.2 GMM-based method

For GMM-based voice conversion, we tried and evaluated five mixtures (8, 16, 32, 64, 128 mixtures) to determine an appropriate number of mixtures. Fig. 7.4 shows the average MCDs over the development set when using the GMM with various mixtures. As shown in the figure, the GMM with 64 mixtures performed the best. Therefore, we used mixtures of 64 for GMM in the evaluation experiments described in section 7.3.3.

7.3.3 Results and Discussion

Table 7.1: Average MOS w.r.t. similarity for each method.

SD-RTRBM	SD-RBM	NN	GMM
2.86	2.80	2.77	2.14

Figs. 7.5, 7.6, and 7.7 summarize the experimental results for the test set, comparing objective criteria for male-to-female, male-to-male, and female-to-female voice conversion, respectively, for $F = 5,000$, $10,000$, $20,000$, and $40,000$ training frames. These figures also include the RNN results for reference. Table 7.1 shows the experimental results with respect to subjective criteria.

As shown in these figures and the table, the proposed approach (SD-RTRBM) outperformed the other methods in every case (regardless of the gender). The

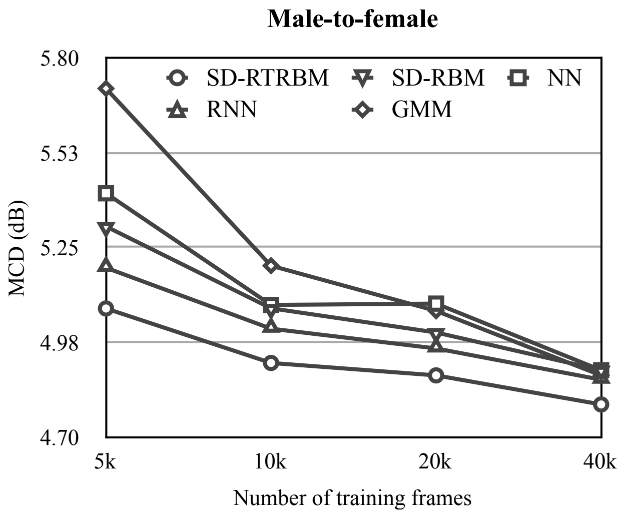


Figure 7.5: Male-to-female voice conversion results. The values show average MCD for each method with varying amounts of training data.

reason for the improvement is attributable to the fact that the time-dependent high-order conversion system using RTRBMs is able to capture and convert the abstractions of unique speaker characteristics better than the other methods. In particular, as shown in Figs. 7.5, 7.6, and 7.7, the proposed approach achieved high performance in MCD criteria. This is because the RTRBMs modeled and captured sequence data more appropriately than the other methods and reduced estimation errors.

One interesting point is that most of the MCDs shown in Figs. 7.5, 7.6, and 7.7 decreased as the amount of training data increased, but the MCD of NN at $F = 20,000$ in the male-to-female experiment and the MCD of RNN at $F = 10,000$ in the female-to-female experiment increased with increases in training data. This is caused by a fall into local minima starting from the randomly-initialized weights. The proposed method provided more stable performance than the randomly-initialized methods.

7.4 Summary

In this chapter, we presented a voice conversion method that combines speaker-dependent RTRBMs and a NN to extract time-dependent unique speaker information from sequence data. Using experiments, we confirmed that the proposed approach is more effective, regardless of gender and especially in terms of MCD, than a well-known approach using a Gaussian mixture model (GMM), an ap-

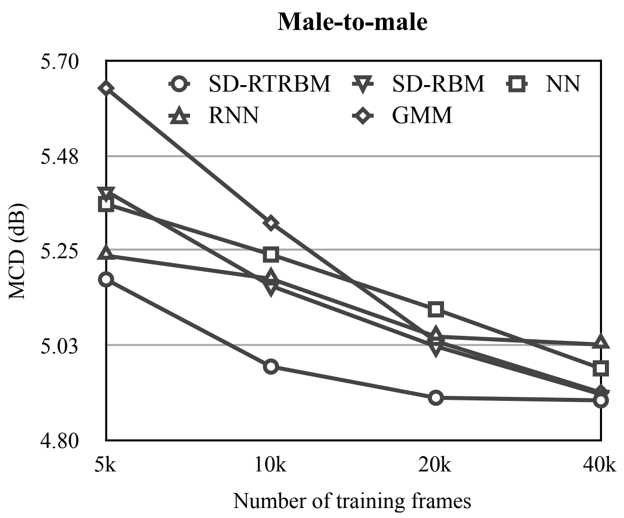


Figure 7.6: Male-to-male voice conversion results. The values show average MCD for each method with varying amounts of training data.

proach based on a neural network (NN), and the previous work on speaker-dependent restricted Boltzmann machines (SD-RBMs). The proposed method can be seen as a special way of generative initialization (training) of a recurrent neual network (RNN). Experimental results that compared the proposed method and a random-initialized RNN indicated that the initial values of the parameter are fairly important for the improvement of accuracy and for stable training. In the future, we will further investigate these relationships and improve the proposed method.

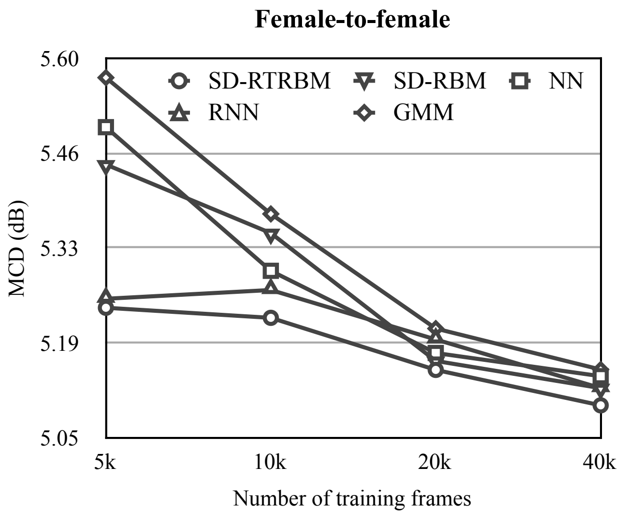


Figure 7.7: Female-to-female voice conversion results. The values show average MCD for each method with varying amounts of training data.

Chapter 8

Conclusion

The thesis describes a new framework for voice conversion based on the recently-fashionable deep learning methods.

In Chapter 4, a voice conversion method using a joint density RBM was presented, as an alternative tool of a sparse-representation-based approach where only a few dictionaries are used for the converted-voice generation. The proposed method demonstrated better performance compared with the conventional sparse-representation-based approach (spectral-mapping NMF and exemplar-based NMF) and the well-known GMM-based approach.

In Chapter 5, we presented a voice conversion technique using restricted Boltzmann machine (RBM) to generate the high-order eigen space of the speaker, where it captures the abstractions of speaker individuality. The experimental results showed the efficacy of the proposed method, in comparison to a conventional GMM-based and NN-based approach.

In Chapter 6, we presented in this chapter a voice conversion method that combines speaker-dependent conditional restricted Boltzmann machines (CRBMs) and a NN to extract speaker-individual information for speech conversion. Through experiments, we confirmed that the proposed approach is effective, especially in terms of MCD, compared with the well-known conventional Gaussian mixture model (GMM)-based approach, a neural network (NN)-based approach, and the previous work, speaker-dependent restricted Boltzmann machine (SD-RBM), (and recurrent neural network for a reference), regardless of the gender in conversion. We also conducted ABX experiments for subjective evaluation. The results showed that the performance of the proposed method was not always significantly different in comparison to NN, RNN, and SD-RBM; however, it did perform significantly better than these methods in terms of either speaker identity or speech quality.

In Chapter 7, we presented a voice conversion method that combines speaker-dependent RTRBMs and a NN to extract time-dependent unique speaker information from sequence data. Using experiments, we confirmed that the proposed approach is more effective, regardless of gender and especially in terms of MCD, than a well-known approach using a Gaussian mixture model (GMM), an approach based on a neural network (NN), and the previous work on speaker-dependent restricted Boltzmann machines (SD-RBMs). The proposed method can be seen as a special way of generative initialization (training) of a recurrent neual network (RNN). Experimental results that compared the proposed method and a random-initialized RNN indicated that the initial values of the parameter are fairly important for the improvement of accuracy and for stable training.

Appendix

Proof of Eqs. (4.5)(4.6)(4.7)(4.8)

In BB-RBM, the joint probability of $\boldsymbol{v}$ and $\boldsymbol{h}$ is represented as

$$
\begin{aligned}
p(\boldsymbol{v}, \boldsymbol{h}) &= \frac{1}{Z} e^{-E_{BB}(\boldsymbol{v},\boldsymbol{h})} && (1)\\
&= \frac{1}{Z} e^{\boldsymbol{b}^{\mathrm{T}}\boldsymbol{v}+\boldsymbol{c}^{\mathrm{T}}\boldsymbol{h}+\boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{h}}. && (2)
\end{aligned}
$$

On the other hand, the probability of $\boldsymbol{v}$ forms:

$$
\begin{aligned}
p(\boldsymbol{v}) &= \sum_{\boldsymbol{h}'} p(\boldsymbol{v}, \boldsymbol{h}') && (3)\\
&= \frac{1}{Z} \sum_{\boldsymbol{h}'} e^{-E_{BB}(\boldsymbol{v},\boldsymbol{h}')} && (4)\\
&= \frac{1}{Z} \sum_{\boldsymbol{h}'} e^{\boldsymbol{b}^{\mathrm{T}}\boldsymbol{v}+\boldsymbol{c}^{\mathrm{T}}\boldsymbol{h}'+\boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{h}'} && (5)\\
&= \frac{1}{Z} e^{\boldsymbol{b}^{\mathrm{T}}\boldsymbol{v}} \sum_{\boldsymbol{h}'} e^{(\boldsymbol{c}^{T}+\boldsymbol{v}^{\mathrm{T}}\boldsymbol{W})\boldsymbol{h}'} && (6)\\
&= \frac{1}{Z} e^{\boldsymbol{b}^{\mathrm{T}}\boldsymbol{v}} \sum_{\boldsymbol{h}'} \prod_{j} e^{(c_j+\boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j})h'_j} && (7)\\
&= \frac{1}{Z} e^{\boldsymbol{b}^{\mathrm{T}}\boldsymbol{v}} \sum_{\boldsymbol{h}'} e^{(c_1+\boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:1})h'_1} \times e^{(c_2+\boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:2})h'_2} \times \cdots \times e^{(c_J+\boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:J})h'_J} && (8)\\
&= \frac{1}{Z} e^{\boldsymbol{b}^{\mathrm{T}}\boldsymbol{v}} \sum_{h'_1} e^{(c_1+\boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:1})h'_1} \times \sum_{h'_2} e^{(c_2+\boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:2})h'_2} \times \cdots \times \sum_{h'_J} e^{(c_J+\boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:J})h'_J} && (9)
\end{aligned}
$$

$$= \frac{1}{Z} e^{\boldsymbol{b}^{\mathrm{T}}\boldsymbol{v}} \prod_j \sum_{h'_j} e^{(c_j + \boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j})h'_j}. \tag{10}$$

Using Eqs. (2)(10), we obtain the conditional probability $p(\boldsymbol{h}|\boldsymbol{v})$ as follows:

$$\begin{aligned}
p(\boldsymbol{h}|\boldsymbol{v}) &= \frac{p(\boldsymbol{v},\boldsymbol{h})}{p(\boldsymbol{v})} & (11)\\
&= \frac{\frac{1}{Z} e^{\boldsymbol{b}^{\mathrm{T}}\boldsymbol{v} + \boldsymbol{c}^{\mathrm{T}}\boldsymbol{h} + \boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}\boldsymbol{h}}}{\frac{1}{Z} e^{\boldsymbol{b}^{\mathrm{T}}\boldsymbol{v}} \prod_j \sum_{h'_j} e^{(c_j + \boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j})h'_j}} & (12)\\
&= \frac{e^{\boldsymbol{b}^{\mathrm{T}}\boldsymbol{v}} \prod_j e^{(c_j + \boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j})h_j}}{e^{\boldsymbol{b}^{\mathrm{T}}\boldsymbol{v}} \prod_j \sum_{h'_j} e^{(c_j + \boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j})h'_j}} & (13)\\
&= \prod_j \frac{e^{(c_j + \boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j})h_j}}{\sum_{h'_j} e^{(c_j + \boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j})h'_j}} & (14)\\
&= \prod_j \frac{e^{(c_j + \boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j})h_j}}{e^{(c_j + \boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j})\cdot 0} + e^{(c_j + \boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j})\cdot 1}} & (15)\\
&= \prod_j \frac{e^{(c_j + \boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j})h_j}}{1 + e^{c_j + \boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j}}} & (16)\\
&= \prod_j p(h_j|\boldsymbol{v}), & (17)
\end{aligned}$$

where $p(h_j|\boldsymbol{v}) = \frac{e^{(c_j + \boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j})h_j}}{1 + e^{c_j + \boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j}}}$ indicates the conditional probability of the jth hidden unit. As Eq. (17) shows, hidden units are conditionally-independent to each other.

Deforming the probability $p(h_j|\boldsymbol{v})$, we obtain

$$\begin{aligned}
p(h_j|\boldsymbol{v}) &= \frac{e^{(c_j + \boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j})h_j}}{1 + e^{c_j + \boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j}}} & (18)\\
&= \left(\frac{e^{c_j + \boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j}}}{1 + e^{c_j + \boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j}}}\right)^{h_j} \cdot \left(\frac{1}{1 + e^{c_j + \boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j}}}\right)^{1-h_j} & (19)\\
&= \left(\frac{1}{1 + e^{-(c_j + \boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j})}}\right)^{h_j} \cdot \left(\frac{e^{-(c_j + \boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j})}}{1 + e^{-(c_j + \boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j})}}\right)^{1-h_j} & (20)
\end{aligned}$$

$$= \left(\frac{1}{1+e^{-(c_j+\boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j})}}\right)^{h_j} \cdot \left(1-\frac{1}{1+e^{-(c_j+\boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j})}}\right)^{1-h_j} \tag{21}$$

$$= \big(\mathcal{S}(c_j+\boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j})\big)^{h_j} \cdot \big(1-\mathcal{S}(c_j+\boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j})\big)^{1-h_j} \tag{22}$$

$$= \mathcal{B}(h_j;\mathcal{S}(c_j+\boldsymbol{v}^{\mathrm{T}}\boldsymbol{W}_{:j})), \tag{23}$$

where $\mathcal{B}(\cdot;p)$ denotes Bernoulli distribution with success probability p.

It is straightward to derive the conditional probability distribution $p(\boldsymbol{v}|\boldsymbol{h})$, because the variables $\boldsymbol{v}$ and $\boldsymbol{h}$ are exchangeable with each other. That is:

$$p(\boldsymbol{v}|\boldsymbol{h}) = \frac{p(\boldsymbol{v},\boldsymbol{h})}{p(\boldsymbol{h})} \tag{24}$$

$$= \frac{\frac{1}{Z}e^{\boldsymbol{b}^{\mathrm{T}}\boldsymbol{v}+\boldsymbol{c}^{\mathrm{T}}\boldsymbol{h}+\boldsymbol{v}\boldsymbol{W}\boldsymbol{h}}}{\frac{1}{Z}e^{\boldsymbol{c}^{\mathrm{T}}\boldsymbol{h}}\prod_i\sum_{v_i'}e^{(b_i+\boldsymbol{W}_{i:}\boldsymbol{h})v_i'}} \tag{25}$$

$$= \prod_i \frac{e^{(b_i+\boldsymbol{W}_{i:}\boldsymbol{h})v_i}}{1+e^{b_i+\boldsymbol{W}_{i:}\boldsymbol{h}}} \tag{26}$$

$$= \prod_i p(v_i|\boldsymbol{h}), \tag{27}$$

where

$$p(v_i|\boldsymbol{h}) = \frac{e^{(b_i+\boldsymbol{W}_{i:}\boldsymbol{h})v_i}}{1+e^{b_i+\boldsymbol{W}_{i:}\boldsymbol{h}}} \tag{28}$$

$$= \mathcal{B}(v_i;\mathcal{S}(b_i+\boldsymbol{W}_{i:}\boldsymbol{h})). \tag{29}$$

Abbreviation

BB-RBM	**B**ernoulli-**B**ernoulli **R**estricted **B**oltzmann **M**achine
BM	**B**oltzmann **M**achine
CC	**C**epstral **D**ivergence
CCA	**C**anonical **C**orrelation **A**nalysis
CD	**C**ontrastive **D**ivergence
CRBM	**C**onditional **R**estricted **B**oltzmann **M**achine
DBM	**D**eep **B**oltzmann **M**achine
DBN	**D**eep **B**elief **N**etwork
DP	**D**ynamic **P**rogramming
GB-RBM	**G**aussian-**B**ernoulli **R**estricted **B**oltzmann **M**achine
GMM	**G**aussian **M**ixture **M**odel
JD-GMM	**J**oint **D**ensity **G**aussian **M**ixture **M**odel
JD-RBM	**J**oint **D**ensity **R**estricted **B**oltzmann **M**achine
LPC	**L**inear **P**rediction **C**oefficient
MCD	**M**el-**C**epstral **D**istortion

MFCC	**M**el-**F**requency **C**epstral **C**oefficient
MLE	**M**aximum **L**ikelihood **E**stimation
MLP	**M**ulti-**L**ayer **P**erceptron
MLSA	**M**el **L**og **S**epctral **A**pproximation
NMF	**N**on-negative **M**atrix **F**actorization
NN	**N**eural **N**etwork
RBM	**R**estricted **B**oltzmann **M**achine
RTRBM	**R**ecurrent **T**emporal **R**estricted **B**oltzmann **M**achine
SDIR	**S**pectral **D**istortion **I**mprovement **R**atio
VC	**V**oice **C**onversion

Acknowledgements

First, I would like to thank to my advisors, Professor Yasuo Ariki and Associate Professor Tetsuya Takiguchi at Kobe University, who have given me their constant help, read the manuscript with great care and offered me invaluable advice. Without their enlightening instruction, impressive kindness and patience, I could not have completed this dissertation. Their keen and vigorous academic observation enlightens me not only in this thesis but also in my future study.

My thanks also go to Professor Fumio Kojima and Professor Hisashi Tamaki, who have carefully examined this thesis and given me informative suggestions. All of their sugestions improved this document.

I shall extend my thanks to all the teachers and researchers in speech signal processing at the other associations who have given me valuable advice during conferences, who have helped me enrich and broaden my knowledge.

Finally, my sincere thanks should go to my family and friends for their valuable encouragement and spiritual support over the years.

References

[1] Zhizheng Wu, Eng Siong Chng, and Haizhou Li. Conditional restricted boltzmann machine for voice conversion. In *IEEE China Summit and International Conference on Signal and Information Processing (ChinaSIP)*, 2013. vii, 5, 60, 61

[2] Alexander Kain and Michael W. Macon. Spectral voice conversion for text-to-speech synthesis. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 285–288, 1998. 1

[3] Christophe Veaux and X. Robet. Intonation conversion from neutral to expressive speech. In *Proc. Interspeech*, pages 2765–2768, 2011. 1

[4] 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. 1

[5] Li Deng, Alex Acero, Li Jiang, Jasha Droppo, and Xuedong Huang. High-performance robust speech recognition using stereo training data. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 301–304, 2001. 1

[6] Aki Kunikoshi, Yu Qiao, Nobuaki Minematsu, and Keikichi Hirose. Speech generation from hand gestures based on space mapping. In *Proc. Interspeech*, pages 308–311, 2009. 1

[7] Sebastian Möller. *Assessment and prediction of speech quality in telecommunications*. Springer, 2000. 1

[8] Yizhar Lavner, Judith Rosenhouse, and Isak Gath. The prototype model in speaker identification by human listeners. *International Journal of Speech Technology*, 4(1):63–74, 2001. 2

[9] Elina Helander and Jani Nurminen. On the importance of pure prosody in the perception of speaker identity. In *INTERSPEECH*, pages 2665–2668, 2007. 2

[10] David T Chappell and John HL Hansen. Speaker-specific pitch contour modeling and modification. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 2, pages 885–888. IEEE, 1998. 2

[11] Ben Gillett and Simon King. Transforming f0 contours. 2003. 2

[12] Elina E Helander and Jani Nurminen. A novel method for prosody prediction in voice conversion. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–509. IEEE, 2007. 2

[13] Meinard Müller. *Information retrieval for music and motion*, volume 6. Springer, 2007. 3, 11

[14] Robert Gray. Vector quantization. *IEEE ASSP Magazine*, 1(2):4–29, 1984. 3

[15] H. Valbret, E. Moulines, and Jean-Pierre Tubach. Voice transformation using PSOLA technique. *Speech Communication*, 11(2):175–187, 1992. 3

[16] 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. 3, 11

[17] Chung-Han Lee and Chung-Hsien Wu. Map-based adaptation for speech conversion using adaptation data selection and non-parallel training. In *Proc. Interspeech*, pages 2254–2257, 2006. 3

[18] Tomoki Toda, Yamato Ohtani, and Kiyohiro Shikano. Eigenvoice conversion based on gaussian mixture model. In *Proc. Interspeech*, pages 2446–2449, 2006. 3

[19] Daisuke Saito, Keisuke Yamamoto, Nobuaki Minematsu, and Keikichi Hirose. One-to-many voice conversion based on tensor representation of speaker space. In *Proc. Interspeech*, pages 653–656, 2011. 3

[20] Daisuke Saito, Shinji Watanabe, Atsushi Nakamura, and Nobuaki Minematsu. Probabilistic integration of joint density model and speaker model for voice conversion. In *Proc. Interspeech*, pages 1728–1731, 2010. 3

[21] Srinivas Desai, E. Veera Raghavendra, B. Yegnanarayana, Alan W. Black, and Kishore Prahallad. Voice conversion using artificial neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3893–3896. IEEE, 2009. 3, 19, 60, 61

[22] Yi-Jian Wu, Hisashi Kawai, Jinfu Ni, and Ren-Hua Wang. Minimum segmentation error based discriminative training for speech synthesis application. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages I–629. IEEE, 2004. 3

[23] Erik McDermott, Timothy J. Hazen, Jonathan Le Roux, Atsushi Nakamura, and Shigeru Katagiri. Discriminative training for large-vocabulary speech recognition using minimum classification error. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(1):203–223, 2007. 3

[24] TODA Tomoki and Keiichi Tokuda. A speech parameter generation algorithm considering global variance for hmm-based speech synthesis. *IEICE TRANSACTIONS on Information and Systems*, 90(5):816–824, 2007. 4

[25] Zhen-Hua Ling and Li-Rong Dai. Minimum kullback–leibler divergence parameter generation for hmm-based speech synthesis. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(5):1492–1502, 2012. 4

[26] Zhen-Hua Ling, Yi-Jian Wu, Yu-Ping Wang, Long Qin, and Ren-Hua Wang. Ustc system for blizzard challenge 2006 an improved HMM-based speech synthesis method. In *Blizzard Challenge Workshop*, 2006. 4

[27] Zhihua Jian and Zhen Yang. Voice conversion using canonical correlation analysis based on gaussian mixture model. In *Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007. Eighth ACIS International Conference on*, pages 210–215. IEEE, 2007. 4

[28] Ryoichi Takashima, Tetsuya Takiguchi, and Yasuo Ariki. Exemplar-based voice conversion in noisy environment. In *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pages 313–317. IEEE, 2012. 4, 21, 23, 27, 29, 39

[29] Zhizheng Wu, Tuomas Virtanen, Tomi Kinnunen, Eng Siong Chng, and Haizhou Li. Exemplar-based voice conversion using non-negative spectrogram deconvolution. In *the 8th ISCA Speech Synthesis Workshop*, 2013. 4, 21, 27, 29

[30] Toru Nakashika, Ryoichi Takashima, Tetsuya Takiguchi, and Yasuo Ariki. Voice conversion in high-order eigen space using deep belief nets. In *Proc. Interspeech*, pages 369–372, 2013. 4, 60, 61, 62, 76

[31] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. *Parallel Distributed Processing*, 1, 1986. 4, 30

[32] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006. 4, 35, 43, 46

[33] Z-H Ling, Li Deng, and Dong Yu. Modeling spectral envelopes using restricted boltzmann machines and deep belief networks for statistical parametric speech synthesis. *IEEE Transactions on Audio, Speech, and Language Processing*, (10):2129–2139, 2013. 4

[34] Abdel-rahman Mohamed, George E. Dahl, and Geoffrey Hinton. Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):14–22, 2012. 4

[35] Vinod Nair and Geoffrey Hinton. 3-d object recognition with deep belief nets. *Advances in Neural Information Processing Systems*, 22:1339–1347, 2009. 4

[36] Thomas Deselaers, Saša Hasan, Oliver Bender, and Hermann Ney. A deep learning approach to machine transliteration. In *Fourth Workshop on Statistical Machine Translation*, pages 233–241. Association for Computational Linguistics, 2009. 4

[37] Ilya Sutskever, Geoffrey Hinton, and Graham Taylor. The recurrent temporal restricted boltzmann machine. In *NIPS*, volume 19, pages 1601–1608, 2008. 4, 72, 73, 74

[38] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *International Conference on Machine Learning*, 2012. 4, 74

[39] Ling-Hui Chen, Zhen-Hua Ling, Yan Song, and Li-Rong Dai. Joint spectral distribution modeling using restricted boltzmann machines for voice conversion. In *Proc. Interspeech*, pages 3052–3056, 2013. 5

[40] Graham W. Taylor, Geoffrey E. Hinton, and Sam T. Roweis. Modeling human motion using binary latent variables. In *Advances in neural information processing systems*, pages 1345–1352, 2006. 5, 55

[41] Bruce P Bogert, Michael JR Healy, and John W Tukey. The quefrency alanysis of time series for echoes: Cepstrum, pseudo-autocovariance, cross-cepstrum and saphe cracking. In *Proceedings of the symposium on time series analysis*, volume 15, pages 209–243. chapter, 1963. 7

[42] Paul Mermelstein. Distance measures for speech recognition, psychological and instrumental. *Pattern recognition and artificial intelligence*, 116:374–388, 1976. 7

[43] Ben Milner and Xu Shao. Speech reconstruction from mel-frequency cepstral coefficients using a source-filter model. In *Proc. Interspeech*, pages 2421–2424, 2002. 9, 40, 47, 62, 76

[44] Zbynek Tychtl and Josef Psutka. Speech production based on the mel-frequency cepstral coefficients. In *EUROSPEECH*, volume 99, pages 2335–2338, 1999. 9

[45] Tenkasi Ramabadran, Jeff Meunier, Mark Jasiuk, and Bill Kushner. Enhancing distributed speech recognition with back-end speech reconstruction. In *INTERSPEECH*, pages 1859–1862, 2001. 9

[46] Satoshi Imai. Cepstral analysis synthesis on the mel frequency scale. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'83.*, volume 8, pages 93–96. IEEE, 1983. 9

[47] 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. 9

[48] 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 *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3933–3936. IEEE, 2008. 9, 40, 47, 62, 76

[49] Arthur R Toth and Alan W Black. Using articulatory position data in voice transformation. *ISCA SSW6*, pages 182–187, 2007. 11

[50] Arthur P Dempster, Nan M Laird, Donald B Rubin, et al. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal statistical Society*, 39(1):1–38, 1977. 12

[51] Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006. 15

[52] Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, DTIC Document, 1961. 17

[53] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985. 17

[54] Srinivas Desai, Alan W Black, B Yegnanarayana, and Kishore Prahallad. Spectral mapping using artificial neural networks for voice conversion. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18(5):954–964, 2010. 19

[55] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2000. 21

[56] Mikkel Schmidt and Rasmus Olsson. Single-channel speech separation using sparse non-negative matrix factorization. In *Interspeech*, 2006. 21

[57] Alexey Ozerov and Cédric Févotte. Multichannel nonnegative matrix factorization in convolutive mixtures for audio source separation. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18(3):550–563, 2010. 21

[58] Tuomas Virtanen. Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(3):1066–1074, 2007. 21

[59] Nasser Mohammadiha, Jalal Taghia, and Arne Leijon. Single channel speech enhancement using bayesian nmf with recursive temporal updates of prior distributions. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 4561–4564. IEEE, 2012. 21

[60] Jort F Gemmeke, Tuomas Virtanen, and Antti Hurmalainen. Exemplar-based sparse representations for noise robust automatic speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 19(7):2067–2080, 2011. 21

[61] Paris Smaragdis and Judith C Brown. Non-negative matrix factorization for polyphonic music transcription. In *Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on.*, pages 177–180. IEEE, 2003. 21

[62] Arshia Cont, Shlomo Dubnov, David Wessel, et al. Realtime multiple-pitch and multiple-instrument recognition for music signals using sparse non-negative constraints. In *Proceedings of Digital Audio Effects Conference (DAFx)*, 2007. 21

[63] Toru Nakashika, Tetsuya Takiguchi, and Yasuo Ariki. Probabilistic spectrum envelope: Categorized audio-features representation for nmf-based sound decomposition. In *INTERSPEECH*, pages 1765–1768, 2011. 21

[64] Toru Nakashika, Tetsuya Takiguchi, and Yasuo Ariki. Constrained spectrum generation using a probabilistic spectrum envelope for mixed music analysis. In *ISMIR*, pages 181–184, 2011. 21

[65] Ryo Aihara, Ryoichi Takashima, Tetsuya Takiguchi, and Yasuo Ariki. Exemplar-based individuality-preserving voice conversion for articulation disorders in noisy environments. In *Interspeech*, pages 3638–3641, 2013. 21, 23, 27, 29

[66] Ryo Aihara, Ryoichi Takashima, Tetsuya Takiguchi, and Yasuo Ariki. A preliminary demonstration of exemplar-based voice conversion for articulation disorders using an individuality-preserving dictionary. *EURASIP Journal on Audio, Speech, and Music Processing*, 2014(1):5, 2014. 21, 23, 27, 29

[67] Ryo Aihara, Ryoichi Takashima, Tetsuya Takiguchi, and Yasuo Ariki. Noise-robust voice conversion based on sparse spectral mapping using non-negative matrix factorization. *IEICE TRANSACTIONS on Information and Systems*, 2014. 21, 25, 27, 29

[68] Ryoichi Takashima, Ryo Aihara, Tetsuya Takiguchi, and Yasuo Ariki. Noise-robust voice conversion based on spectral mapping on sparse space. In *SSW8*, pages 71–75, 2013. 21, 25, 27, 29, 39

[69] Peter Földiak. Forming sparse representations by local anti-hebbian learning. *Biological cybernetics*, 64(2):165–170, 1990. 21

[70] Julian Eggert and Edgar Korner. Sparse coding and nmf. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 4, pages 2529–2533. IEEE, 2004. 21

[71] Patrik O Hoyer. Non-negative matrix factorization with sparseness constraints. *The Journal of Machine Learning Research*, 5:1457–1469, 2004. 21

[72] Patrik O Hoyer. Non-negative sparse coding. In *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on*, pages 557–565. IEEE, 2002. 21, 23

[73] Weixiang Liu, Nanning Zheng, and Xiaofeng Lu. Non-negative matrix factorization for visual coding. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 3, pages III–293. IEEE, 2003. 21, 23

[74] David L Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006. 29

[75] Hong Chang, Dit-Yan Yeung, and Yimin Xiong. Super-resolution through neighbor embedding. In *Computer Vision and Pattern Recognition*, volume 1, pages 275–282. IEEE, 2004. 29

[76] Simon Haykin and Neural Network. A comprehensive foundation. *Neural Networks*, 2(2004), 2004. 30

[77] David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for boltzmann machines*. *Cognitive science*, 9(1):147–169, 1985. 30

[78] Yoav Freund and David Haussler. *Unsupervised learning of distributions of binary vectors using two layer networks.* Computer Research Laboratory, 1994. 30

[79] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. 30, 35

[80] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002. 34

[81] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071. ACM, 2008. 34

[82] Guillaume Desjardins, Aaron C Courville, Yoshua Bengio, Pascal Vincent, and Olivier Delalleau. Tempered markov chain monte carlo for training of restricted boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, pages 145–152, 2010. 35

[83] Radford M Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, 2001. 35

[84] Julian Besag. Statistical analysis of non-lattice data. *The statistician*, pages 179–195, 1975. 35

[85] Aapo Hyvärinen. Some extensions of score matching. *Computational statistics & data analysis*, 51(5):2499–2512, 2007. 35

[86] Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. *Journal of Machine Learning Research*, 15:29–37, 2011. 35

[87] Geoffrey Hinton. A practical guide to training restricted boltzmann machines. In *Tech. Rep. Department of Computer Science, University of Toronto*, 2010. 35, 48

[88] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Computer Science Department, University of Toronto, Tech. Rep*, 2009. 35, 62, 76

[89] KyungHyun Cho, Alexander Ilin, and Tapani Raiko. Improved learning of gaussian-bernoulli restricted boltzmann machines. In *Artificial Neural Networks and Machine Learning*, pages 10–17, 2011. 35, 36

[90] Akira Kurematsu, Kazuya Takeda, Yoshinori Sagisaka, Shigeru Katagiri, Hisao Kuwabara, and Kiyohiro Shikano. ATR japanese speech database as a tool of speech recognition and synthesis. *Speech Communication*, 9(4):357–363, 1990. 39, 47, 61, 76

[91] Ruslan Salakhutdinov and Geoffrey E Hinton. Deep boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, pages 448–455, 2009. 41

[92] Ruslan Salakhutdinov. Learning deep generative models. In *PhD thesis, University of Toronto*, 2009. 48

[93] A-R Mohamed and Geoffrey Hinton. Phone recognition using restricted boltzmann machines. In *in Proceedings of IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages 4354–4357. IEEE, 2010. 48

[94] Nan Wang, Jan Melchior, and Laurenz Wiskott. An analysis of gaussian-binary restricted boltzmann machines for natural images. In *in Proceedings of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, pages 287–292, 2012. 48

[95] 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. 60

Publication List

Journal Articles

[1] Toru Nakashika, Tetsuya Takiguchi, and Yasuo Ariki:"Voice Conversion Based on Speaker-dependent Restricted Boltzmann Machines,"IEICE Transactions on Information and Systems, Vol.E97-D, No.6, pp.1403-1410, June 2014.

[2] Toru Nakashika, Tetsuya Takiguchi, Yasuo Ariki:"Parallel Dictionary Learning Using a Joint Density Restricted Boltzmann Machine for Sparse-Representation-Based Voice Conversion,"Advances in Computer Science and Engineering, Vol.12, No.2, pp.101-117, June 2014.

[3] Toru Nakashika, Tetsuya Takiguchi, Yasuo Ariki:"Voice conversion using speaker-dependent conditional restricted Bolzmann machine,"EURASIP Journal on Audio, Speech, and Music Processing (in review).

[4] Toru Nakashika, Tetsuya Takiguchi, Yasuo Ariki:"Voice Conversion Using Speaker-Dependent Recurrent Temporal Restricted Boltzmann Machines,"IEEE Transactions on Audio, Speech, and Language Processing (in review).

International Conferences

[1] Toru Nakashika, Ryoichi Takashima, Tetsuya Takiguchi, Yasuo Ariki:"Voice Conversion in High-order Eigen Space Using Deep Belief Nets,"INTERSPEECH 2013, pp.369-372, 2013-08.

[2] Toru Nakashika, Tetsuya Takiguchi, Yasuo Ariki:"Voice Conversion in Time-Invariant Speaker-Independent Space,"ICASSP2014, pp.7939-7943, 2014-05.

[3] Toru Nakashika, Tetsuya Takiguchi, Yasuo Ariki:“High-Order Sequence Modeling Using Speaker-Dependent Recurrent Temporal Restricted Boltzmann Machines for Voice Conversion,”INTERSPEECH 2014 (accepted).

Technical Reports

[1] Toru Nakashika, Tetsuya Takiguchi, Yasuo Ariki: “Speaker-dependent conditionl restricted Boltzmann machine for voice conversion,” IEICE Technical Report, vol. 113, no. 366, SP2013-88, pp. 83-88, 2013-12 (2013 IEICE ISS Young Researcher’s Award in Speech Field) (in Japanese).

[2] Toru Nakashika, Tetsuya Takiguchi, Yasuo Ariki: “A joint restricted Boltzmann machine for dictionary learning in sparse-representation-based voice conversion,” IEICE Technical Report, vol. 114, no. 52, IEICE-SP2014-34, pp. 343-348, 2014-05 (in Japanese).

Domestic Conferences

[1] Toru Nakashika, Ryoichi Takashima, Tetsuya Takiguchi, Yasuo Ariki:“Study on voice conversion using Deep Belief Nets,”Acoustical Society of Japan 2013 Spring Meeting, 3-P-46b, pp.517-520, 2013-03 (in Japanese).

[2] Toru Nakashika, Tetsuya Takiguchi, Yasuo Ariki:“Voice conversion based on time-varying deep networks,”Acoustical Society of Japan 2013 Autumn Meeting, 3-7-5, pp.1471-1472, 2013-09 (in Japanese).

[3] Ryo Aihara, Toru Nakashika, Tetsuya Takiguchi, Yasuo Ariki:“Voice Conversion Based on Non-negative Matrix Factorization Using Phoneme Categorized Dictionary,”Acoustical Society of Japan 2013 Autumn Meeting, 3-7-6, pp.1473-1476, 2013-09 (in Japanese).

[4] Toru Nakashika, Tetsuya Takiguchi, Yasuo Ariki:“Parallel-dictionary learning using restricted Boltzmann machine for voice conversion,”Acoustical Society of Japan 2014 Spring Meeting, 1-R5-18, pp.415-416, 2014-03 (in Japanese).

[5] Takao Fujii, Toru Nakashika, Ryo Aihara, Tetsuya Takiguchi, Yasuo Ariki:“Voice Conversion based on NMF using Speaker Adaptation,”Acoustical Society of Japan 2014 Spring Meeting, 1-R5-20, pp.421-424, 2014-03 (in Japanese).