DEEP CASA FOR ROBUST PITCH TRACKING AND SPEAKER SEPARATION

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Yuzhou Liu, M.S.

Graduate Program in Computer Science and Engineering

The Ohio State University

2019

Dissertation Committee:

Professor DeLiang Wang, Advisor

Professor Eric Fosler-Lussier

Professor Alan Ritter

Abstract

Speech is the most important means of human communication. In real environments, speech is often corrupted by acoustic inference, including noise, reverberation and competing speakers. Such interference leads to adverse effects on audition, and degrades the performance of speech applications. Inspired by the principles of human auditory scene analysis (ASA), computational auditory scene analysis (CASA) addresses speech separation in two main steps: segmentation and grouping. With noisy speech decomposed into a matrix of time-frequency (T-F) units, segmentation organizes T-F units into segments, each of which corresponds to a contiguous T-F region and is supposed to originate from the same source. Two types of grouping are then performed. Simultaneous grouping aggregates segments overlapping in time to simultaneous streams. In sequential grouping, simultaneous streams are grouped across time into distinct sources. As a traditional speech separation approach, CASA has been successfully applied in various speech-related tasks. In this dissertation, we revisit conventional CASA methods, and perform related tasks from a deep learning perspective.

As an intrinsic characteristic of speech, pitch serves as a primary cue in many CASA systems. A reliable estimate of pitch is important not only for extracting harmonic patterns at a frame level, but also for streaming voiced speech in sequential

grouping. Based on the types of interference, we can divide pitch tracking in two categories: single pitch tracking in noise and multi-pitch tracking.

Pitch tracking in noise is challenging as the harmonic structure of speech can be severely contaminated. To recover the missing harmonic patterns, we propose to use long short-term memory (LSTM) recurrent neural networks (RNNs) to model sequential dynamics. Two architectures are investigated. The first one is conventional LSTM that utilizes recurrent connections to model temporal dynamics. The second one is two-level time-frequency LSTM, with the first level scanning frequency bands and the second level connecting the first level through time. Systematic evaluations show that both proposed models outperform a deep neural network (DNN) based model in various noisy conditions.

Multi-pitch tracking aims to extract concurrent pitch contours of different speakers. Accurate pitch estimation and correct speaker assignments need to be achieved at the same time. We use DNNs to model the probabilistic pitch states of two simultaneous speakers. Speaker-dependent (SD) training is adopted for a more accurate assignment of pitch states. A factorial hidden Markov model (FHMM) then integrates pitch probabilities and generates the most likely pitch tracks. Evaluations show that the proposed SD DNN-FHMM framework outperforms other speaker-independent (SI) and SD multi-pitch trackers on two-speaker mixtures.

Speaker-independent multi-pitch tracking has been a long-standing difficulty. We extend the DNN-FHMM framework, and use an utterance-level permutation invariant training (uPIT) criterion to train the system with speaker-independent data. A

speaker separation front end is further added to improve pitch estimation. The proposed SI approach substantially outperforms all other SI multi-pitch trackers, and largely closes the gap with SD methods.

Besides exploring deep learning based pitch tracking as cues for CASA, we directly address talker-independent monaural speaker separation from the perspectives of CASA and deep learning, resulting in what we call a deep CASA approach. Simultaneous grouping is first performed for frame-level separation of the two speakers with a permutation-invariantly trained neural network. Sequential grouping then assigns the frame-level separated spectra to distinct speakers with a clustering network. Compared to a uPIT system which conducts frame-level separation and speaker tracking in one stage, our deep CASA framework achieves better performance for both objectives. Evaluation results on the benchmark WSJ two-speaker mixture database demonstrate that deep CASA significantly outperforms other spectral-domain approaches.

In talker-independent speaker separation, generalization to an unknown number of speakers and causal processing are two important considerations for real-world deployment. We propose a multi-speaker extension to deep CASA for C concurrent speakers ($C \geq 2$), which works well for speech mixtures with up to C speakers even without the prior knowledge about the speaker number. We also propose extensive revisions to the connections, normalization and clustering algorithm in deep CASA to make a causal system. Experimental results on the WSJ0-2mix and WSJ0-3mix databases show that both extensions achieve the state-of-the-art performance. The development of the deep CASA approach in this dissertation represents a major step towards solving the cocktail party problem.

This work is dedicated to my family.

Acknowledgments

First and foremost, I want to give my deepest thanks to my advisor Prof. DeLiang Wang. He leads by example, and has taught me the key qualities to become a researcher: curiosity, patience and persistence. I have faced numerous challenges throughout my seven-year Ph.D. study, not only from an academic perspective, but also healthwise. Prof. Wang always stands firmly by me, and treats me with patience and kindness. This dissertation would not have been completed without his guidance and support.

I want to thank Prof. Eric Fosler-Lussier and Prof. Alan Ritter for their time and effort in serving on my dissertation committee. I enjoyed two of Prof. Fosler-Lussier's classes, and gained an initial understanding of speech recognition and natural language processing. Prof. Ritter has served on my candidacy examination committee. His insightful comments made me think deeper about my future research. I would like to thank Prof. Brian Kulis. He introduced me to the world of machine learning, and sparked my interest in this field. Without him I probably would have left 5 years ago with a master's degree.

I am fortunate to spend two wonderful summers in Amazon Lab126. I thank Dr. Trausti Kristjansson for hosting me in the Echo audio team. I benefit a lot from his industrial experience, and always enjoy our discussions on research ideas. I thank Krishna Kamath and Dr. Balaji Thoshkahna for serving as my mentors.

Their support made my internship go smoothly. I thank Dr. Anshuman Ganguly for his excellent work in our collaboration. I learned a lot from his expertise in array processing.

I would like to sincerely thank my lab mates. Dr. Xiaojiao Zhao helped me start my research on robust speaker identification. Dr. Kun Han gave me constructive suggestions on robust pitch tracking, which later led to my first journal paper. Dr. Arun Narayanan patiently answered many of my questions on robust ASR. I enjoyed going to lunch with Dr. Jitong Chen and Dr. Yuxuan Wang in the first couple years of my Ph.D study. Our discussions greatly broadened my knowledge of deep learning. I also benefited a lot from Dr. Donald Williamson's expertise in NMF and speech quality. Lastly, I want to thank my junior lab mates, including Masood Delfarah, Zhong-Qiu Wang, Yan Zhao, Ke Tan, Peidong Wang, Hao Zhang, Ashutosh Pandey and Hassan Taherian.

I am grateful to my friends in the U.S., Europe and China. They make my life meaningful and delightful.

I owe my greatest gratitude to my parents, my father Suyan Liu and my mother Yunxiang Zhou, for their unconditional love and utmost support throughout my life.

Vita

June 29, 1991 ..... Born in Xi'an, Shaanxi, China

2012 ..... B.E. in Automation, Xi'an Jiaotong University, Shaanxi, China

2017 ..... M.S. in Computer Science and Engineering, The Ohio State University

Publications

Research Publications

Y. Liu and D. L. Wang. "Divide and conquer: A deep CASA approach to talker-independent monaural speaker separation," *arXiv preprint arXiv:1904.11148*, 2019. *IEEE/ACM Trans. Audio, Speech, and Lang. Process.*, revision under review.

Y. Liu and D. L. Wang. "Permutation invariant training for speaker-independent multi-pitch tracking," In *Proc. ICASSP*, pp. 5594-5598, 2018.

Y. Liu and D. L. Wang. "A CASA approach to deep learning based speaker-independent co-channel speech separation," In *Proc. ICASSP*, pp. 5399-5403, 2018.

Y. Liu and D. L. Wang. "Speaker-dependent multipitch tracking using deep neural networks," *J. Acoust. Soc. Amer.*, vol. 141, pp. 710-721, 2017.

Y. Liu and D. L. Wang. "Time and frequency domain long short-term memory for noise robust pitch tracking," In *Proc. ICASSP*, pp. 5600-5604, 2017.

Y. Liu and D. L. Wang. "Robust pitch tracking in noisy speech using speaker-dependent deep neural networks," In *Proc. ICASSP*, pp. 5255-5259, 2016.

Y. Liu and D. L. Wang. "Speaker-dependent multipitch tracking using deep neural networks," In *Proc. Interspeech*, pp. 3279-3283, 2015.

Fields of Study

Major Field: Computer Science and Engineering

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Motivation

Audition is one of the most important senses of human perception. Without the capability of hearing, we would miss so much information, as well as the delight and chroma of the world. As a kind of the auditory signal, human speech contains rich information: not only phonetics, syntax and semantics which correspond to the key components of a language, but also loudness and articulation, through which we can tell the identity, emotion, and even health of a speaker [82]. Speech is an inseparable part of our daily communication. It is even of greater importance in the early stage of human history. Before the invention of writing, which can be traced back several millennia, speech is the sole media by which human culture is recorded and inherited. Without speech and audition, we would never be who we are today.

The speech sound reaches our ears usually not in a clean forms. In real acoustic environments, speech is contaminated by various types of interference, such as traffic noise, industrial noise, music, and competing speakers. Through millions years of evolution, the human vocal tract has developed to produce vowels like [i] and [u], which are more robust to environmental noise than consonants [72] [87] . On the other hand, a normal-hearing listener excels at attending to the target speech in the

presence of interference. Cherry uses the term "cocktail party effect" to describe the phenomenon of the human listener extracting a single sound source while filtering out other sources [15]. Bregman attributes this effect to auditory scene analysis (ASA) [9], and summarizes the ASA process of the human auditory system into two stages: segmentation and grouping. In segmentation, the input sound is decomposed into auditory segments, each originating from the same sound source. In the grouping stage, segments likely from the same source are organized into a stream. Following the principles of ASA, computational auditory scene analysis (CASA) [107] aims to realize speech separation with machines. The segmentation stage of CASA forms time-frequency (T-F) segments of an input signal, each corresponding to a contiguous region of T-F units originating from the same sound source. The grouping stage of CASA has two subprocesses: simultaneous grouping and sequential grouping. Simultaneous grouping aggregates T-F segments overlapping in time to simultaneous streams. In sequential grouping, simultaneous streams are grouped across time into auditory streams, each corresponding to a distinct source. A main computational goal in CASA is an ideal binary mask (IBM), which is a binary T-F matrix where each T-F unit is either labeled as 1 if it is dominated by the target speech, or 0 if dominated by interference [106]. The CASA principles have been successfully applied in a variety of speech applications, including pitch estimation [51], speech recognition [85], signal to noise ratio (SNR) estimation [86] and speaker identification [124].

Many CASA methods make use of auditory features like pitch to perform binary mask estimation. For example, Hu and Wang [50] utilize pitch for frame-level simultaneous grouping, and expand the target speech along time using the continuity of pitch. In [51], a tandem algorithm is further developed to perform pitch estimation

and mask estimation in an iterative fashion. The tandem algorithm first estimates pitch on T-F segments with high cross-channel correlation values, and expands pitch using temporal continuity. The estimated pitch is then used to predict the associated binary masks. The masks are used in turn to refine the pitch contours. Pitch and mask estimation operate jointly and iteratively to produce a set of consistent pitch-mask pairs. In [52], the outputs of the tandem algorithm are clustered into two sources for talker-independent speaker separation. The clustering is performed by a search to maximize the ratio of between- and within-group speaker distances while penalizing within-group concurrent pitch contours. Substantial SNR gain has been achieved across a range of input SNRs.

Due to paramount importance of pitch in CASA, and its broad utilization in other speech applications, such as automatic speech recognition [12] and emotion recognition [65], robust pitch estimation of human speech is considered a fundamental problem in speech processing. Many algorithms have been designed for robust pitch estimation over the last few decades [8] [16] [101]. They achieve good performance under clean or modestly noisy conditions, but fail to produce consistent results when speech is severely interfered by noise or other speakers.

Recently, a resurgence of research in artificial neural networks (NNs), also known as "deep learning", marks a major milestone in machine learning. Countless neural networks, including feedforward deep neural networks (DNNs), recurrent neural networks (RNNs) and convolutional neural networks (CNNs), have been explored over the past decade for supervised classification/regression [22] [43], reinforcement learning [84] and unsupervised generation [32]. These models are able to handle inputs

of various types, such as one-dimensional features, two-dimensional images and sequential data, and produce excellent results in many speech applications, e.g., pitch tracking [35], speech separation [108], speech recognition [42], speaker recognition [97], speech synthesis [89], and voice conversion [47]. In [35], spectral features are fed to a DNN and an RNN to predict frame-level probabilities of pitch states, which are then connected by a hidden Markov model (HMM) to generate continuous pitch contours. The DNN-HMM framework outperforms other models in a variety of noise conditions. These successes motivate us to explore how the latest progress of deep learning benefits pitch tracking in the presence of different types of interference.

On the other hand, with the rapid development of deep learning, speech separation has been formulated as a supervised learning problem [106]. Typically, a DNN is used to map noisy features to some ideal T-F masks, e.g., the IBM and the ideal ratio mask (IRM), that can separate the target speech from the mixture. If the acoustic features, training targets, training data and learning machines are well chosen, supervised speech separation can generate high-quality results in both matched and unmatched test conditions [14]. In [110], a DNN-based system is trained for monaural speech enhancement. It outperforms the pitch-based tandem algorithm [51] by a large margin. Based on these observations, we ask two questions. Is pitch still important to deep learning based speech separation? Is the CASA framework outdated in the era of deep learning?

For the first question, the answer might be no. In [13], Chen et al. utilize pitch as part of a complementary feature set for deep learning based speech separation. No improvement can be made if the additional pitch based feature is derived using a pitch tracking algorithm PEFAC [31]. Such a result is due to the fact that PEFAC

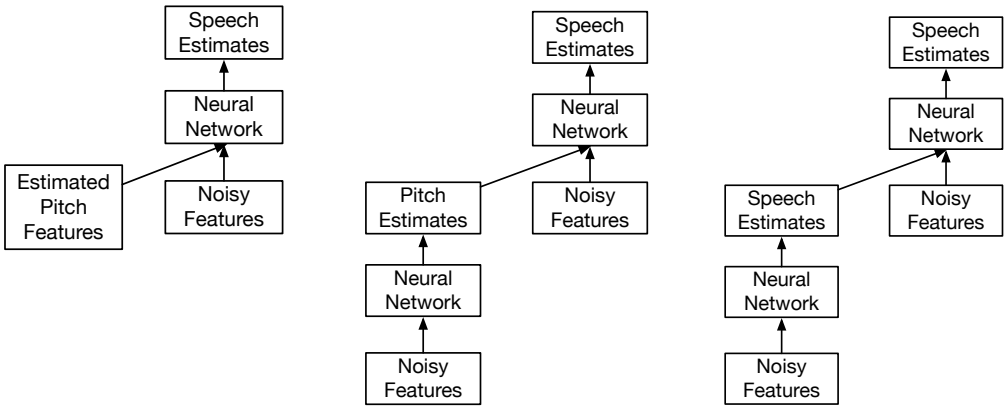


Figure 1.1: Incorporating pitch to deep learning based speech separation. (a) Pitch based features estimated by signal processing based models. (b) Pitch based features estimated by deep learning models. (c) Two-stage speech separation.

makes many mistakes in low SNR conditions, which in turn degrades the performance of deep separation models. This system is illustrated in Fig. 1.1a. To improve the performance of pitch estimation, we can train a DNN based pitch tracker, and use the derived pitch estimates as additional features for DNN based separation, as shown in Fig. 1.1b. The two DNNs can be optimized jointly. Our preliminary results indicate such a model introduces only a marginal improvement. On the other hand, clean speech contains much richer information than clean pitch. If clean speech is given, one can almost perfectly obtain the corresponding clean pitch. However, if clean pitch and noisy speech are given, it is difficult to get a perfect reconstruction of clean speech. Based on these facts, we train an alternative model, as shown in Fig. 1.1c. In this system, an initial estimate of clean speech is used as additional features, and these features are expected to contain richer information than pitch based features. This is also known as two-stage speech separation [126]. Our preliminary results indicate the

two-stage system consistently outperforms DNNs with pitch based features. Although pitch may not be critical for speech separation, it can be used for many other audio processing tasks like emotion recognition [65] and musical note detection.

Is CASA outdated? We think not. For speech enhancement, namely single-speaker separation in noisy conditions, conventional CASA methods are no longer competitive. With data-driven large scale training, deep learning based speech enhancement outperforms cue-based CASA methods in a variety of conditions [110]. Simultaneous grouping and sequential grouping in CASA can be effectively achieved within one stage using sequence models like CNNs and RNNs. However, in the case of multi-speaker separation, where the goal is to separate several concurrent speakers, the concept of CASA is still valuable. Many deep learning based speaker separation models utilize multiple output layers in the network, each corresponding to one speaker source. These studies assume that the output-speaker pairing does not change between training and testing. It has been shown that such talker-dependent training leads to significant intelligibility improvement for hearing impaired listeners [38]. However, talker-dependent training does not generalize to untrained speakers. Talker-independent speaker separation has to address the permutation problem [40] [64], i.e., how the output layers are tied to the underlying speakers. To tackle this problem, a data-driven output alignment algorithm is proposed: permutation invariant training (PIT) [64]. PIT examines all possible label permutations for each utterance during training, and uses the one with the lowest utterance-level loss to train the separation network, which leads to significant improvement for talk-independent separation. However, the one-stage PIT system optimizes simultaneous grouping and sequential grouping at the same time, which seems too difficult for multi-speaker separation.

Even with PIT's well designed output-target alignment, the outputs still switch between different speakers during inference for challenging mixtures. On the other hand, the divide-and-conquer strategy of CASA is well suited for this task. In CASA, simultaneous grouping and sequential grouping are performed in turn, so sequential grouping benefits from the short-term separation results, and better speaker tracking can be achieved. It would be interesting to exploit CASA principles for speaker separation in the deep learning paradigm.

Although microphone-array approaches are widely used for denoising and multi-source tracking, monaural (single-microphone) solutions are more flexible in terms of deployment and may help array-based techniques [108]. All systems described in this dissertation operate on monaural recordings.

1.2 Objectives

This dissertation is concerned with CASA related tasks using deep learning techniques. There are two main objectives: robust pitch estimation and talker-independent speaker separation, which are presented in turn as follows:

- *Robust single pitch tracking.* Pitch is an important cue for conventional CASA methods and many other applications. Based on the type of interference, we divide pitch tracking tasks into two categories: noise-robust single-pitch tracking and multi-pitch tracking. Noise-robust single-pitch tracking deals with single-speaker utterances mixed with non-speech noises. Spectral-, temporal- and spectrotemporal-domains approaches have been proposed for this task, but they do not perform well in the presence of strong nonstationary noise, as harmonic

patterns can be severely corrupted. Exploiting long spectral and temporal context is a key to complete the missing information buried in noise. We investigate deep sequence models along both the time and frequency dimension for pitch probability estimation.

- *Speaker-dependent multi-pitch tracking.* Multi-pitch tracking is concerned with mixtures of concurrent speakers. Compared to single-pitch tracking, this poses new challenges as both pitch estimation and speaker assignment need to be addressed. If speaker-dependent (SD) data is given, models can be trained speaker-dependently to differentiate the target speaker from competing speakers in a mixture. We explore deep learning based speaker-dependent models for multi-pitch tracking. We also investigate the generalization of the models by exposing them to speaker-independent data and various SNR conditions.

- *Speaker-independent multi-pitch tracking.* SD models are not practical for many real applications, as both the SD training data and identities of speakers need to be provided. We extend the SD multi-pitch tracking framework, and investigate PIT techniques for speaker-independent training. Speech separation and pitch tracking are considered a chicken-and-egg problem in the speech community. We explore how speech separation can help pitch tracking in the multi-speaker setting.

- *Deep CASA for talker-independent speaker separation.* As mentioned earlier, although CASA methods are not competitive for speech enhancement in the era of deep learning, the concept of simultaneous and sequential grouping is

still valuable for speaker separation. We redesign the CASA method for talker-independent speaker separation using deep learning techniques. Various models and training targets are explored for the two grouping stages. To better understand the difference between one-stage speaker separation and deep CASA, we analyze their frame-level separation and speaker tracking performance in details.

- *Generalization of deep CASA.* Deep CASA is designed for two-speaker mixtures and needs to be extended to more complex acoustic environments. We explore a multi-speaker extension to deep CASA, and test the model without the prior knowledge about the speaker number. We also explore causal-separation extensions to deep CASA for real-time processing.

1.3 Background

In this section, we review existing pitch tracking and speaker separation approaches. First, we discuss single- and multi-pitch tracking. Then, we introduce recently proposed deep learning based talker-independent speaker separation.

1.3.1 Noise-robust single-pitch tracking

Although many algorithms have been proposed for single pitch tracking [8] [16] [101], they do not produce consistent results when speech is severely interfered by noise. The difficulty of pitch tracking in noise stems from the fact that both temporal continuities and harmonic patterns are corrupted. Recently, many studies try to address the noise-robustness issue for pitch tracking, and most of them consist of two stages. In the fist stage, pitch candidates or pitch probabilities are estimated for each

time frame of speech using temporal, spectral, or spectrotemporal domain information [107]. Temporal-domain methods analyze the periodic cue of speech. For example RAPT [101] captures peaks in normalized autocorrelation functions (ACFs). YIN [16] proposes a number of modifications to the autocorrelation method to improve the accuracy of pitch estimation. Spectral-domain methods are based on harmonic modeling. For instance, PEFAC [31] uses non-linear amplitude compression and a comb-filter to suppress noise in the spectrogram, and selectes pitch candidates from harmonic peaks. Han and Wang [35] feed spectral features to a DNN and an RNN to predict frame-level probabilities of pitch states. Spectrotemporal methods first decompose the signal into a series of sub-bands, and then perform temporal analysis on each frequency channel. For example, Lee and Ellis [67] apply principle component analysis on subband autocorrelation functions, and feed the derived features into a multilayer perceptron for pitch score estimation. Wang and Hansen [105] decompose speech into overlapped time-frequency segments, and select pitch candidates using likelihood scores for each segment. After the estimation of pitch candidates and probabilities, the second stage integrates local pitch clues into continuous pitch tracks using dynamic programming or HMMs.

1.3.2 Multi-pitch tracking

A number of studies have investigated the problem of monaural multi-pitch tracking. Wu et al. [119] propose a probabilistic representation of pitch and track continuous pitch contours with an HMM. Sha and Saul [95] model the instantaneous frequency spectrogram with nonnegative matrix factorization (NMF) and use the inferred weight coefficients to determine pitch candidates. Bach and Jordan [4] propose

direct probabilistic modeling of the spectrogram and track several pitches with a factorial HMM (FHMM). Christensen and Jakobsson [18] describe statistical, filtering and subspace methods for both single- and multi-pitch estimation. Hu and Wang [51] propose a tandem algorithm that performs pitch estimation and voiced speech segregation jointly, producing a set of pitch contours and their associated binary masks. Jin and Wang [60] improve [119] by designing new techniques for channel selection and pitch score estimation in the context of reverberant and noisy signals. The abovementioned studies build a general system without modeling the characteristics of any specific speaker, and can thus be denoted by speaker-independent models.

Although most speaker-independent models perform well for estimating pitch periods, they can not assign pitch estimates to the underlying speakers for multi-pitch tracking. To alleviate this problem, Hu and Wang [52] build their system on the tandem algorithm [51] and group simultaneous pitch contours into two speakers using a constrained clustering algorithm. Similarly, Duan et al. [25] take the pitch estimates of speaker-independent multi-pitch trackers as input and stream pitch points by clustering. However, both approaches achieve limited improvement as individual pitch contours and points are usually too short to contain enough speaker information for clustering.

On the other hand, speaker-dependent models have been investigated. Wohlmayr et al.[117] model the probability of pitch periods using speaker-dependent Gaussian mixture models (GMMs), and then use a speaker-dependent FHMM to track pitches of two simultaneous speakers. They have shown significant improvement over a speaker-independent approach [119].

1.3.3 Talker-dependent speaker separation

The goal of monaural speaker separation is to estimate C independent speech signals $x_c(n)$, $c = 1, ..., C$, from a single-channel recording of speech mixture $y(n)$, where $y(n) = \sum_{c=1}^{C} x_c(n)$ and n indexes time. Many studies focus on the co-channel situation where $C = 2$.

A lot of deep learning based speaker separation systems [24] [54] [123] address this problem in the T-F domain, where short-time Fourier transform (STFT) is calculated using an analysis window $w(n)$ with fast Fourier transform (FFT) length N and frame shift R:

$$Y(t, f) = \sum_{n=-\infty}^{\infty} w(n - tR)y(n)e^{-j2\pi fn/N} \tag{1.1}$$

$$X_c(t, f) = \sum_{n=-\infty}^{\infty} w(n - tR)x_c(n)e^{-j2\pi fn/N} \tag{1.2}$$

$$Y(t, f) = X_1(t, f) + X_2(t, f) \tag{1.3}$$

where t and f denote the frame and frequency, respectively. The magnitude STFT of the mixture signal $|Y(t, f)|$, together with other spectral features, are fed into a neural network to predict a T-F mask $M_c(t, f)$ for each speaker c. The masks are multiplied by the mixture to estimate the original sources:

$$|\tilde{X}_c(t, f)| = M_c(t, f) \odot |Y(t, f)| \tag{1.4}$$

Here $\odot$ denotes element-wise multiplication, and $|\tilde{X}_c(t, f)|$ denotes the estimated magnitude STFT of speaker c. An estimate of complex STFT $\hat{X}_c(t, f)$ can be obtained by coupling $|\tilde{X}_c(t, f)|$ with noisy phase. In the end, separated waveforms are resynthesized using inverse STFT (iSTFT):

$$\hat{x}_c(n) = \frac{\sum_{t=-\infty}^{\infty} w(n - tR) \frac{1}{N} \sum_{f=0}^{N-1} \hat{X}_c(t, f)e^{j2\pi fn/N}}{\sum_{t=-\infty}^{\infty} w^2(n - tR)} \tag{1.5}$$

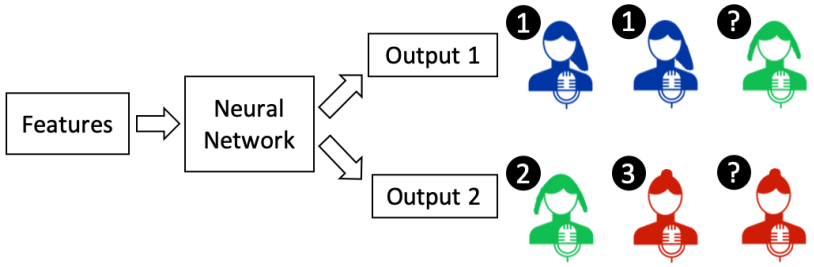


Figure 1.2: Illustration of the permutation problem for talker-independent speaker separation.

Various training targets of $|\tilde{X}_c(t,f)|$ have been explored for supervised speech separation [109]. Phase-sensitive approximation (PSA) is found to be effective as it accounts for errors introduced by the noisy phase [26] [64]. In PSA, the desired reconstructed signal is defined as: $|X_c(t,f)| \odot cos(\phi_c(t,f))$, where $\phi_c(t,f)$ is the phase difference between $Y(t,f)$ and $X_c(t,f)$. Overall, the training loss at each frame is computed as:

$$J_t^{PSA} = \sum_{f=1}^{F} \sum_{c=1}^{2} ||M_c(t,f) \odot |Y(t,f)| - |X_c(t,f)| \odot cos(\phi_c(t,f))|| \tag{1.6}$$

where $||\cdot||$ denotes the l_1 norm.

The above formulation works well only when each output layer is tied to a training target signal with similar characteristics. For instance, we may tie each output to a specific speaker, leading to talker-dependent training. We may also tie two outputs with male and female speakers respectively, leading to gender-dependent training. However, for talker-independent training data, how to select output-speaker pairing becomes a nontrivial problem. Think of a training set consisting of three female speakers, as illustrated in Fig. 1.2. For the mixture of speakers 1 and 2, we can tie Output 1 to speaker 1, and Output 2 to speaker 2. For the mixture of speakers 1

and 3, again Output 1 can be tied to speaker 1, and Output 2 tied to speaker 3. However, it is hard to decide the pairing for the mixture of speakers 2 and 3. If output-speaker pairing is not arranged properly, conflicting gradients may be generated during training, preventing the neural network from converging. This is referred to as the permutation problem [40] [64].

1.3.4 Permutation invariant training

Frame-level permutation invariant training (denoted by tPIT) [64] overcomes the permutation problem in talker-independent speaker separation by providing target speakers as a set instead of an ordered list, and output-speaker pairing for a given frame t, is defined as the pairing that minimizes the loss function over all possible speaker permutations P. For tPIT, the frame-level training loss in Eq. 1.6 is rewritten as:

$$J_t^{tPIT-PSA} = \min_{\theta(t)\in P} \sum_{f,c} ||M_c \odot |Y| - |X_{\theta_c(t)}| \odot cos(\phi_{\theta_c(t)})|| \tag{1.7}$$

We omit (t, f) in M, Y, X, and ϕ for brevity. In (1.7), $\theta_c(t)$ indexes the speaker paired with output c at frame t. $\theta(t)$ includes all C output-speaker pairings at frame t, and corresponds to one speaker permutation. The tPIT objective scans all P permutations, and utilizes the permutation with the minimum frame-level loss. A diagram of the tPIT objective is given in Fig. 1.3.

tPIT does a good job in separating two speakers at the frame level [64] [77]. However, due to its locally optimized training objective, an output layer may be tied to different speakers at different frames, and the correct speaker assignment may swap frequently. If we reassign the outputs with respect to the minimum loss for each

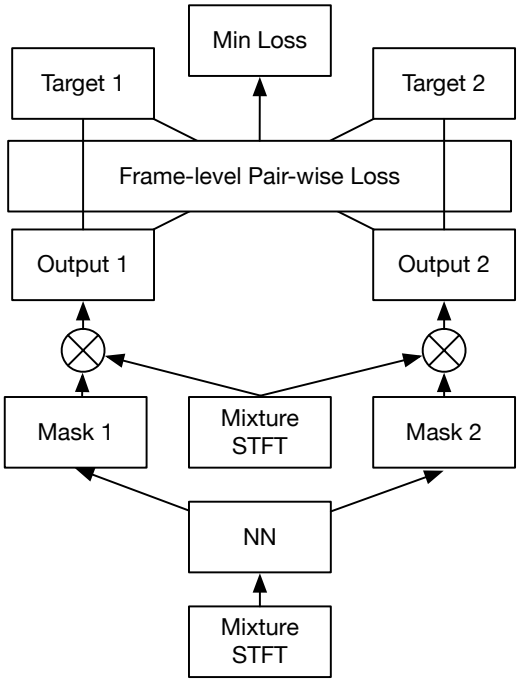


Figure 1.3: Diagram of the tPIT technique.

speaker, tPIT can almost perfectly reconstruct both speakers [77]. One such example is given in Fig. 5.4.

Optimal speaker assignments are not obtainable in practice as the targets are not given beforehand. To address this issue, an utterance-level PIT (uPIT) fixes output-speaker pairing $c \leftrightarrow \theta_c(t)$ for a whole utterance, which corresponds to the pairing that provides the minimum utterance-level loss over all possible permutations. Recent uPIT improvements include new network structure [68] [120] and new training objectives [68]. Conv-TasNet [81] extends uPIT to the waveform domain using a convolutional encoder-decoder structure. FurcaNeXt [96] integrates gated activations and ensemble learning into Conv-TasNet, and reports very high performance.

1.3.5 Deep clustering

Deep clustering (DC) [40] looks at the permutation problem in talker-independent speaker separation from a different perspective. In DC, an RNN with bi-directional long short-term memory (BLSTM) is trained to estimate an embedding vector $\mathbf{V}_i \in \mathbb{R}^{1\times D}$ for the i-th T-F unit of the mixture $Y_i = Y(t, f)$, where i corresponds to T-F indices t and f. Similarly, $\mathbf{A}_i \in \mathbb{R}^{1\times C}$ is a one-hot label vector representing which source in a mixture dominates the i-th T-F unit. Vertically stacking the N T-F bins, the embedding matrix $\mathbf{V} \in \mathbb{R}^{N\times D}$ and the label matrix $\mathbf{A} \in \mathbb{R}^{N\times C}$ are formed. The Frobenius norm between the affinity matrix of embedding vectors ($\mathbf{V}\mathbf{V}^T$) and the affinity matrix of the label vectors ($\mathbf{A}\mathbf{A}^T$) is used as the training objective in DC:

$$J = ||\mathbf{V}\mathbf{V}^T - \mathbf{A}\mathbf{A}^T||_F^2 \tag{1.8}$$

where $||\cdot||_F$ is the Frobenius norm. DC avoids the permutation problem due to the permutation-invariant property of affinity matrices. As training unfolds, embedding vectors of T-F units dominated by the same source are drawn closer together, and embeddings of those units dominated by different sources become farther apart. Clustering these embedding vectors using the K-means algorithm assigns each T-F unit to one of the speakers in the mixture, which can be viewed as binary masking for speech separation.

The original DC system is proposed in [40]. Several upgrades, including deeper network, recurrent dropout and end-to-end training are proposed in [56]. In [80], a concept of attractors is introduced to DC to enable ratio masking and real-time processing. Alternative training objectives, together with a chimera network which simultaneously estimates DC embeddings and uPIT outputs, are proposed in [111]. In

[112], iterative phase reconstruction is integrated into the chimera network to alleviate phase distortions. In [113], a phase prediction network is further added to [112] to estimate the clean phase of each speaker source.

1.4 Organization of dissertation

The rest of this dissertation is organized as follows.

In Chapter 2, we investigate the performance of deep sequence models for noise-robust single-pitch tracking. Long short-term memory (LSTM) RNNs are trained to capture long-term dynamics of speech harmonics along time and frequency. The Viterbi algorithm is then applied to connect probabilistic outputs from LSTM RNNs, and generates the final pitch estimate. The models are trained using speaker-dependent data. We evaluate the pitch tracking performance with detection rate and voicing detection errors [67], and compare the proposed models with speaker-independent and speaker-dependent DNNs.

Chapter 3 presents DNNs for multi-pitch estimation. Speaker-dependent and speaker-pair-dependent DNNs are proposed to model the probabilistic pitch states of two concurrent speakers in a close data set. To relax the constraints, several extensions, including gender-dependent models, speaker adaptation and gain adaptation are introduced. Estimated pitch states are fed into an FHMM to infer the most likely pitch tracks. We compare our methods with other speaker-independent and speaker-dependent multi-pitch trackers on two-speaker mixtures. The total pitch error [117] and overall multi-pitch accuracy [25] are reported. We also investigate the generalization of the proposed methods using noisy two-speaker mixtures and three-speaker mixtures.

Chapter 4 presents speaker-independent extensions to the DNN-FHMM multi-pitch tracking framework. To improve speaker tracking, BLSTM RNNs are used as the learning machine. We adopt the uPIT technique to address the label permutation problem during training. A speaker-separation BLSTM RNN is further added to the system as front-end processing for multi-pitch tracking. The models are trained and evaluated on different sets of speakers.

In Chapter 5, a deep learning based CASA approach is proposed for talker-independent speaker separation, denoted by deep CASA. Similar to conventional CASA methods, simultaneous grouping and sequential grouping are performed serially in deep CASA. Simultaneous grouping separates speakers at the frame level with a permutation-invariantly trained neural network. In sequential grouping, the frame-level outputs are assigned to different speakers with a sequence model. We compare deep CASA with uPIT and DC on the public WSJ0-2mix database [40] using four objective metrics.

Chapter 6 presents multi-speaker and causal-separation extensions to deep CASA. In the multi-speaker extension, we update the sequential grouping module in deep CASA to tackle the factorial increase of label permutations. The multi-speaker extension can be deployed without the prior knowledge about the speaker number. For causal processing, temporal connections, normalization, and clustering algorithms are extensively revised. The proposed extensions are compared with state-of-the-art approaches on the benchmark WSJ0-2mix and WS0-3mix databases [40].

Chapter 7 summarizes the contributions of this dissertation and discusses future directions.

Chapter 2: Time and Frequency Domain Long Short-term Memory for Noise Robust Pitch Tracking

Pitch tracking in noisy speech is a challenging task as temporal and spectral patterns of the speech signal are both corrupted. This chapter presents LSTM based models for pitch probability estimation. Two architectures are investigated. The first one is conventional LSTM that utilizes recurrent connections to model temporal dynamics. The second one is two-level time-frequency LSTM, with the first level scanning frequency bands and the second level connecting the first level through time. The Viterbi algorithm then takes the probabilistic output from LSTM to generate continuous pitch contours. Experiments show that both proposed models outperform a DNN based model in most conditions. Time-frequency LSTM achieves the best performance at negative SNRs. The work presented in this chapter has been published in [76].

2.1 Introduction

Pitch of human speech refers to the fundamental frequency of vocal fold vibrations. A reliable estimate of pitch is useful for various applications, including automatic speech recognition [12], speech separation [107] and emotion recognition [65].

Given that speech has long-term dependency in the time domain, it is natural to exploit temporal dynamics for pitch tracking. However, most pitch tracking algorithms only analyze speech signals within short-time windows when predicting frame-level pitch probabilities/candidates, resulting in inaccurate pitch estimates at noise-dominant frames. To address this problem, Han and Wang [35] propose to use a standard RNN to estimate pitch probabilities over time. Such RNNs are designed to model sequential data, but they suffer from the vanishing and exploding gradient problem [7], and can not propagate information over a long span. LSTM RNNs [45] use gates to stabilize gradient propagation, and are shown to be good at modeling long-term dependencies in many applications such as automatic speech recognition [34] [69] and machine translation [98].

In this study, we extend the RNN based pitch tracking framework, and propose to use LSTM to model the posterior probability that a frequency bin (pitch state) is pitched given frame-level log-spectrogram features. To our knowledge, this is the first study that uses LSTM for pitch tracking in noisy speech. Another important characteristic of voiced speech is that its harmonics are evenly spaced in frequency. When some frequency bands are contaminated by noise, we can still estimate the fundamental frequency from other reliable bands. To leverage this observation, we further propose a two-level LSTM structure. The first level is frequency-domain LSTM (F-LSTM) that scans segments of log-spectrogram along the frequency axis to detect harmonic patterns. The second level is time domain LSTM (T-LSTM), which takes the output of F-LSTM, and models pitch probabilities through time. The overall structure is denoted by time-frequency LSTM (TF-LSTM) in this study. Recently, a similar TF-LSTM network has been shown to outperform conventional LSTM in an

automatic speech recognition task [69]. Once all frame-level pitch probabilities are derived, we use the Viterbi algorithm [27] to generate continuous pitch contours.

The rest of the chapter is organized as follows. The proposed system is described in the next section. In Section 2.3, we present experimental results and comparisons. A conclusion is given in Section 2.4.

2.2 System description

The proposed pitch tracking algorithm consists of two stages: pitch probability estimation and Viterbi decoding.

In the first stage, we extract the log-spectrogram feature $\mathbf{y}_t$ from a noisy utterance sampled at 16 kHz, where t denotes the frame index. Neural networks then use $\mathbf{y}_t$ as input to estimate the posterior probability of pitch states $p(x_t|\mathbf{y}_t)$, where x_t denotes the pitch state at frame t. We quantize the frequency range 60 to 404 Hz into 67 bins $(s^1, s^2, ..., s^{67})$ using 24 bins per octave on a logarithmic scale. Each s^i corresponds to a state in x_t. The logarithmic division of pitch states is due to the fact that resolved pitch frequencies follow a quasi-logarithmic scale in the range [60, 404] Hz approximately [107]. A more accurate representation would be to divide the low-frequency pitch states on a linear scale, and the high-frequency pitch states on a logarithmic scale, like the cochleagram [107]. We adopt the logarithmic scale used in [35] for simplicity. This frequency resolution provides two bins per semitone, and gives less than 3% of relative frequency difference between adjacent pitch states. A non-pitched state s^0 is incorporated into x_t to represent unvoiced speech or silence. $p(x_t = s^i|\mathbf{y}_t)$ equals 1 if the groundtruth pitch is in the frequency bin of s^i, and 0 otherwise. We introduce a DNN as a baseline model in Section 2.2.2 . LSTM and

TF-LSTM are described in Section 2.2.3 and Section 2.2.4. In the second stage, we use the Viterbi algorithm to connect frame-level probabilities and track pitch through time.

2.2.1 Feature extraction

The feature used in study is based on log-spectrogram. To get this feature, the signal is first divided into 32 ms frames with a 10 ms frame shift. We then apply a Hamming window to each frame and derive the spectrogram using 1024-point FFT. Lastly, we compute the logarithm of the amplitude spectrum, and pick bins 2 to 129 (corresponding to a frequency range up to 2000 Hz) as the 128-dimensional feature for each frame. We do not pick all bins in the spectrogram as the energy of high frequency harmonics is relatively low, and the frequency range up to 2000 Hz covers at least 5 harmonics of human speech, enough for continuous pitch tracking.

Since neighboring frames contain useful information, we splice a 15-frame window of features as the DNN's input. For LSTM and TF-LSTM, although the history of input is stored in their memory cells, it is still helpful to apply a context window so that they can receive richer input information at each time step. Taking the model size and computational cost into consideration, we splice a 7-frame window for the input of LSTM and TF-LSTM.

2.2.2 DNN based pitch probability estimation

We first utilize a DNN as a baseline model to estimate the posterior probability of pitch states when the frame-level feature vector is given, i. e., $p(x_t|\mathbf{y}_t)$. The DNN has four hidden layers, each with 1600 rectified linear units [30]. The output layer contains 68 soft-max units, corresponding to the number of pitch states. A cross-entropy cost

function, mini-batch gradient descent, Adam optimization algorithm [62] and dropout regularization [44] are used during training. The initial learning rate is 0.001, and a learning rate decay of 0.7 each epoch is used. The training stops after 30 epochs.

2.2.3 LSTM based pitch probability estimation

To better encode the temporal patterns of human speech, we use LSTM for pitch probability estimation in this subsection. LSTM is composed of a series of recurrently connected memory blocks [45]. Each memory block has a memory cell which stores the temporal state of the network, an input gate which controls the amount of input activation added to the memory cell, a forget gate which adaptively resets the memory cell and an output gate which controls the amount of information passed from the memory cell to the output. In this work, we followed the LSTM architecture in [122]:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \tag{2.1}$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \tag{2.2}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{2.3}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \tag{2.4}$$

$$h_t = o_t \odot \tanh(c_t) \tag{2.5}$$

where i_t, f_t, c_t and o_t denote the input gate, forget gate, memory cell and output gate. x_t and h_t denote the input and output of the memory block. W terms and b terms denote different weight matrices and biases. σ is the logistic sigmoid function. $\odot$ represents element-wise multiplication.

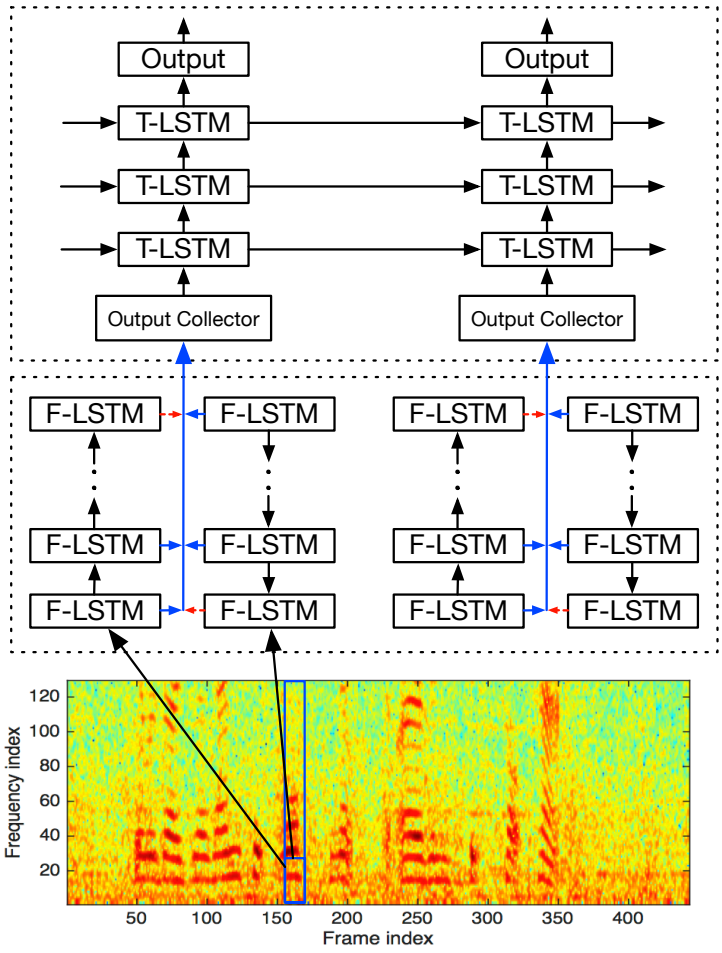


Figure 2.1: Diagram of TF-LSTM.

Our LSTM has four hidden layers, each with 512 hidden units. All layer only contain uni-directional recurrent connections. The output layer is a soft-max layer with 68 units. The number of parameters in LSTM is similar to that in the baseline DNN. To train LSTM, we use a backpropagation through time (BPTT) step of 100. The learning rate decay is set to 0.45 per epoch. Other training recipes follow the baseline DNN.

2.2.4 TF-LSTM based pitch probability estimation

In this subsection, we introduce TF-LSTM which models both temporal and spectral dynamics of speech. A diagram of proposed TF-LSTM is shown in Fig. 2.1. The intention of this architecture is to first use F-LSTM to scan different frequency bands of the log-spectrogram, where useful pitch information can be extracted from relatively clean bands, and propagates along the frequency axis to further affect subsequent noisy bands. The output of F-LSTM is then collected and fed to T-LSTM to track pitch through time. A similar network has been applied to automatic speech recognition tasks and shown to outperform DNN based and conventional LSTM based neural networks [69].

To implement TF-LSTM, we first divide the 128×7-dimensional feature $\mathbf{y}_t$ along the frequency axis into overlapped frequency segments. Each frequency segment contains 24×7 units, and has 16×7 overlapped units with each neighboring frequency segment. In other words, the stride of frequency segments is 8. F-LSTM, which is one layer bidirectional LSTM with 256 units per direction, takes all frequency segments in a frame as input. Therefore F-LSTM is unrolled on the frequency axis $(128-24)/8+1 = 14$ times at each frame. All parameters in F-LSTM are chosen from a development set. The output of F-LSTM is then fed into T-LSTM. Because we have 256×2×14 output units from F-LSTM at each frame, it is inefficient to feed all of them to T-LSTM. We explore two methods to address this problem. The first method only keeps the last outputs in the F-LSTM sequence (red dashed arrows in Fig. 2.1) for T-LSTM, denoted by TF-LSTM-L (TF-LSTM last). Here the last outputs can be viewed as embedding vectors of the log-spectrogram. The second method concatenates all output units in F-LSTM and uses a 512-unit linear transformation layer to reduce

its dimensionality, denoted by TF-LSTM-C (TF-LSTM concatenation). We compare the two methods in Section 2.3. T-LSTM has three 512-unit hidden layers and a 68-unit soft-max output layer. Other details and training recipes follow conventional LSTM.

2.2.5 Viterbi decoding

After the estimation of $p(x_t|\mathbf{y}_t)$, we use the Viterbi algorithm [27] to connect all probabilities along time. The hidden state in the Viterbi algorithm corresponds to x_t, and the observation corresponds to $\mathbf{y}_t$. We use the training data to compute prior probabilities $p(x_t = s_i)$ and transition matrices. Emission probabilities can be computed using the estimated posterior probabilities divided by the prior $p(x_t = s_i)$. The Viterbi algorithm generates a sequence of the most likely pitch states, which is then converted to mean frequencies of the corresponding frequency bins. In the end, a three-frame moving average window is applied to smooth pitch estimates.

2.3 Evaluation results and comparisons

We use the Mocha-TIMIT database [118] for experimental comparisons. This database consists of 460 utterances from both a male and a female speaker. Because the male speaker is less challenging for pitch tracking tasks, we use the female speaker in the following experiments for speaker-dependent learning. The training set is created by mixing 400 utterances of the female speaker with 10,000 noises from a sound-effect library (available at http://www.sound-ideas.com). Each clean utterance is mixed 100 times with a random segment of a random noise at a random SNR from -5 to 5 dB. The total duration of the training set is 44 hours. The test set includes 20 untrained utterances from the Mocha-TIMIT female speaker. Six noises, i. e., babble

noise [48], factory noise [102], speech shape noise (SSN), cocktail-party noise [49], crowd playground noise [49] and crowd music noise [49], are used for test, and all of them are untrained. Each test utterance is mixed with the six test noises at -10, -5, 0, 5 and 10 dB, resulting in a total of 600 test mixtures. The groundtruth pitch is derived by applying the RAPT [101] algorithm on laryngograph signals. We manually remove erroneous pitch in unvoiced regions to improve the quality of the groundtruth pitch.

We use two metrics to evaluate pitch estimates: detection rate (DR) and voicing decision error (VDE) [67]. DR indicates the percentage of correctly estimated voiced frames. VDE computes the percentage of frames that are misclassified in terms of pitched and unpitched decision:

$$\mathbf{DR} = \frac{N_{0.05}}{N_p}, \quad \mathbf{VDE} = \frac{N_{n\to p} + N_{p\to n}}{N} \tag{2.6}$$

Here $N_{0.05}$ is the number of frames whose estimated pitch deviates less than 5% from the groundtruth pitch. $N_{n\to p}$ and $N_{p\to n}$ are the number of frames misclassified as pitched and unpitched respectively. N_p is the number of pitched frames, and N is the total number of frames. Higher DR and lower VDE indicate better pitch estimates.

We compare our methods with two state-of-the-art pitch tracking algorithms: PEFAC [31] and Han and Wang [35]. PEFAC is a representative unsupervised approach that performs relatively well in low SNR conditions. Han and Wang's approach used the same DNN/RNN-HMM framework as ours, and was trained on a speaker-independent dataset. Two of Han and Wang's training noises are seen in our test set.

Table 2.1 and Table 2.2 list the DR and VDE of different approaches, where all values are averaged across six noise types. As shown in the tables, all supervised

Table 2.1: Comparison of approaches in terms of DR.

SNR (dB)	-10	-5	0	5	10
PEFAC	0.373	0.555	0.657	0.696	0.714
Han and Wang DNN	0.434	0.635	0.728	0.755	0.756
Han and Wang RNN	0.406	0.633	0.727	0.755	0.763
Proposed DNN	0.664	0.861	0.934	0.953	0.958
Proposed LSTM	0.706	0.876	**0.938**	**0.956**	**0.959**
Proposed TF-LSTM-L	**0.714**	**0.880**	0.937	0.954	0.958
Proposed TF-LSTM-C	0.711	0.878	**0.938**	0.954	0.957

Table 2.2: Comparison of approaches in terms of VDE.

SNR (dB)	-10	-5	0	5	10
PEFAC	0.337	0.262	0.192	0.142	0.112
Han and Wang DNN	0.295	0.221	0.149	0.103	0.095
Han and Wang RNN	0.301	0.226	0.165	0.120	0.108
Proposed DNN	0.247	0.131	0.062	0.047	**0.041**
Proposed LSTM	0.228	0.119	0.063	0.048	0.043
Proposed TF-LSTM-L	0.221	0.116	**0.059**	**0.046**	0.042
Proposed TF-LSTM-C	**0.204**	**0.112**	0.061	0.047	0.042

learning approaches produce better results than PEFAC across all SNRs. A standard RNN was used by Han and Wang to model temporal dynamics, but it does not outperform their DNN based approach in most cases, which is due to the fact that such RNNs are more difficult to train and can not model long-term effects. By virtue of the large training set, speaker-dependent training [74] and better training recipes, the proposed methods show significant improvements over Han and Wang's approach. When the SNR is positive, all proposed methods generate exceptionally

accurate pitch estimates while making few voicing decision mistakes. When it comes to negative SNRs, the LSTM based method produces clearly higher DRs and lower VDEs than the DNN based method, indicating that the capacity of sequence modeling makes LSTM better at processing very noisy speech. TF-LSTM-L and TF-LSTM-C both outperform LSTM at negative SNRs. They yield comparable results in terms of DR, and TF-LSTM-C achieves a VDE of 0.204 at -10 dB, significantly lower than all other approaches. Such improvement is attributed to the frequency scanning module in TF-LSTM, which makes the model better at distinguishing unpitched signals from voiced speech.

Fig. 2.2 compares the performance of the proposed DNN, LSTM and TF-LSTM-C in different noises, which is consistent with the results in Table 2.1 and 2.2. A major improvement of LSTM and TF-LSTM-C comes from negative SNRs. LSTM outperforms the DNN in most negative SNR conditions. TF-LSTM-C generates similar DR as LSTM, but consistently lower VDE in all cases.

2.4 Conclusion

In this chapter, we have introduced LSTM for robust pitch tracking in noisy speech. Both conventional LSTM and two-level TF-LSTM are utilized to estimate probabilistic pitch states. TF-LSTM first uses F-LSTM to scan harmonic patterns, and then uses T-LSTM to connect frequency-domain activations. Thanks to the power of sequence modeling, both LSTM based models outperform a DNN based model. TF-LSTM further reduces the VDE of conventional LSTM by 10% at -10 dB. In the future, we will incorporate sub-band features into TF-LSTM. We will also

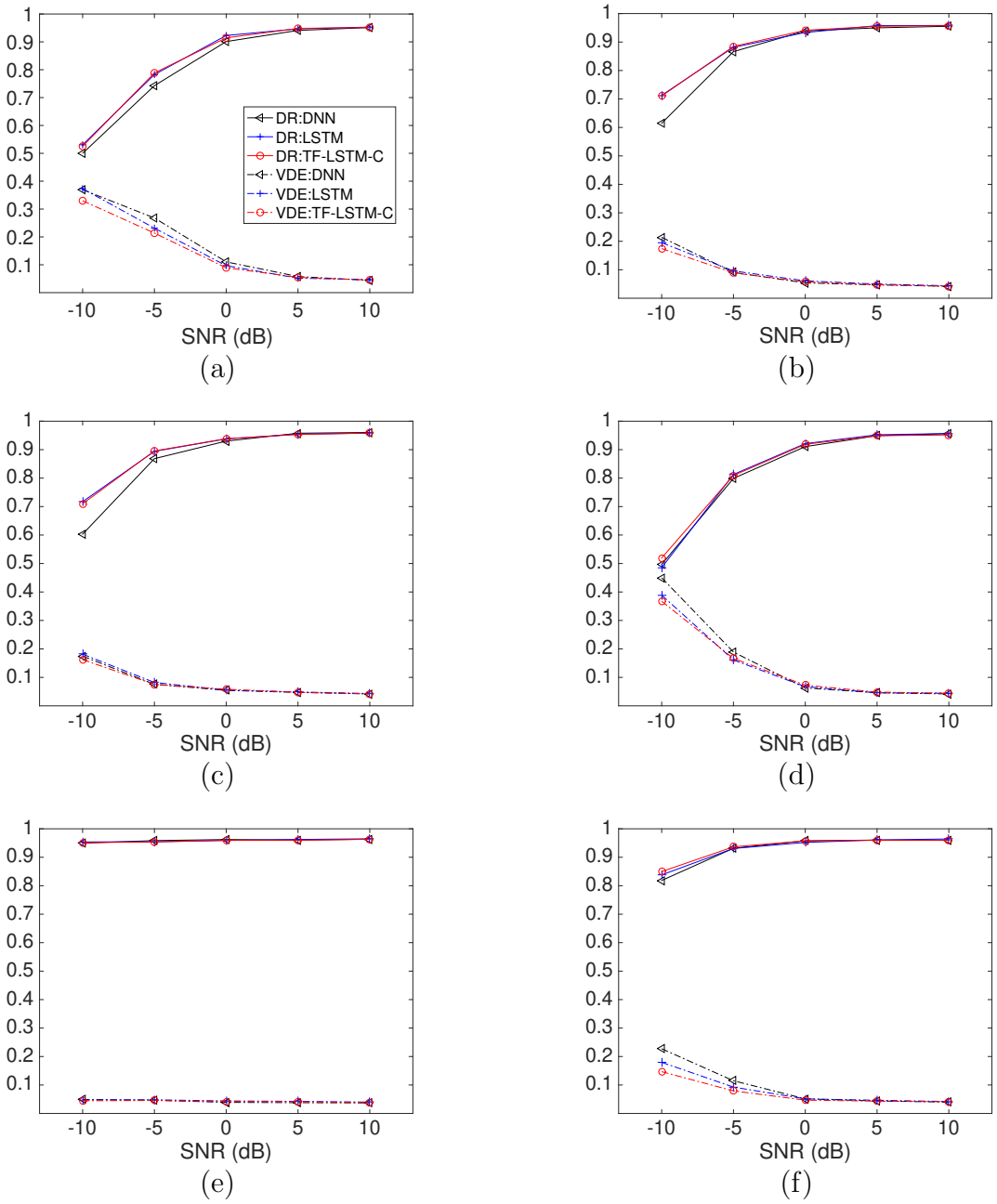


Figure 2.2: DR and VDE for (a) babble noise, (b) factory noise, (c) SSN, (d) cocktail-party noise, (e) crowd playground noise, (f) crowd music noise.

perform speaker-independent training to evaluate how well TF-LSTM generalizes to untrained speakers.

Chapter 3: Speaker-dependent Multi-pitch Tracking using Deep Neural Networks

Multi-pitch tracking is important for speech and signal processing. However, it is challenging to design an algorithm that achieves accurate pitch estimation and correct speaker assignment at the same time. In this chapter, DNNs are used to model the probabilistic pitch states of two simultaneous speakers. To capture speaker-dependent information, two types of DNN with different training strategies are proposed. The first is trained for each speaker enrolled in the system (speaker-dependent DNN), and the second is trained for each speaker pair (speaker-pair-dependent DNN). Several extensions, including gender-pair-dependent DNNs, speaker adaptation of gender-pair-dependent DNNs and training with multiple energy ratios, are introduced later to relax constraints. An FHMM then integrates pitch probabilities and generates the most likely pitch tracks with a junction tree algorithm. Experiments show that the proposed methods substantially outperform other speaker-independent and speaker-dependent multi-pitch trackers on two-speaker mixtures. With multi-ratio training, the proposed methods achieve consistent performance at various energies ratios of the two speakers in a mixture. The work presented in this chapter has been published in [73] [75].

3.1 Introduction

This chapter is concerned with multi-pitch tracking when two speakers are talking simultaneously in a monaural recording. We propose a speaker-dependent and discriminative technique to model the pitch probability at each time frame. Specifically, we use DNNs to model the posterior probability that a pair of frequency bins (pitch states) is pitched given frame-level observations. A DNN is a feedforward neural network that contains more than one hidden layer [43]. Recently, Han and Wang [35] use DNNs to model the posterior probability of pitch states for single-pitch tracking in noisy conditions, which motivates the use of DNNs for multi-pitch tracking in this study. To leverage individual speaker characteristics, we train a DNN for each speaker enrolled in the system, denoted by speaker-dependent DNNs or SD-DNNs. We also train DNNs for different pairs of speakers, denoted by speaker-pair-dependent DNNs or SPD-DNNs. We then extend the DNN based models to relax practical constraints. To deal with untrained speakers, we train three gender-pair-dependent DNNs (male-male, male-female and female-female, denoted by GPD-DNNs) as a generalization of SPD-DNNs. GPD-DNNs only require gender information during testing. With insufficient training data, direct training of SD-DNNs or SPD-DNNs may result in overfitting. To examine this issue, we conduct a fast adaptation of GPD-DNNs for each speaker pair with limited training data. Also, the utterances of the two speakers in a mixture usually have different energy ratios, leading to a ratio mismatch between training and test. We address this problem by including various speaker energy ratios in training, denoted by the multi-ratio training.

After estimating the posterior probability of pitch states, we use an FHMM for pitch tracking. Under the framework of the FHMM, the pitch state of each speaker

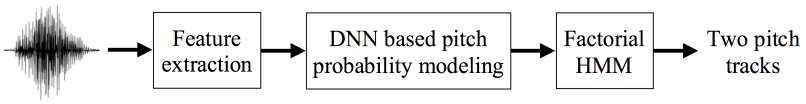


Figure 3.1: Diagram of the proposed multi-pitch tracker.

evolves within its own Markov chain, while the emission probability is derived using the posterior probability estimated by DNNs. We then use the junction tree algorithm [61] to infer the most likely pitch tracks.

The rest of the chapter is organized as follows. The next section gives an overview of the system architecture. Feature extraction is discussed in Section 3.3. The details of DNN based posterior probability estimation are introduced in Section 3.4. Section 3.5 describes the FHMM for multi-pitch tracking. Experimental results and comparisons are presented in Section 3.6. Finally, we conclude the chapter and discuss related issues in Section 3.7.

3.2 System overview

A diagram of the proposed multi-pitch tracker is illustrated in Fig. 3.1. The input to the system is a speech mixture v_t sampled at 16 KHz:

$$v_t = u_t^1 + u_t^2 \tag{3.1}$$

where u_t^1 and u_t^2 are utterances of two speakers. Given the mixture, our system first extracts frame-level features $\mathbf{y}_m$ with a frame shift of 10 ms, which corresponds to the first module in the diagram.

In the second stage, features are fed into DNNs to derive the posterior probability of pitches at frame m, i. e., $p(x_m^1, x_m^2|\mathbf{y}_m)$, where x_m^1 and x_m^2 denote pitch states of

two speakers at frame m. Both x_m^1 and x_m^2 have 68 states $(s^1, s^2, s^3, \ldots, s^{68})$, where s^1 refers to an unvoiced or silent state, and s^2 to s^{68} encode different pitch frequencies ranging from 60 to 404 Hz [35]. Specifically, we quantize the pitch frequency range 60 to 404 Hz using 24 bins per octave on a logarithmic scale, resulting in a total of 67 bins. This frequency resolution provides two bins per semitone, and gives less than 3% of relative frequency difference between adjacent pitch states, adequate for continuous pitch tracking. $p(x_m^1 = s^i, x_m^2 = s^j|\mathbf{y}_m)$ equals one if groundtruth pitches fall into the i^{th} and j^{th} frequency bins respectively. We propose two types of DNN to estimate the posterior probability, which are the speaker-dependent DNNs and the speaker-pair-dependent DNNs. We also explore several extensions. The detailed settings of DNNs can be found in Section 3.4.

The final module converts the posterior probability $p(x_m^1, x_m^2|\mathbf{y}_m)$ to the emission probability of an FHMM $p(\mathbf{y}_m|x_m^1, x_m^2)$. The junction tree algorithm is then applied to infer the most likely pitch tracks. Note that in the following sections, a pitch contour refers to a continuous pitch trajectory from the same speaker, and a pitch track refers to a set of pitch contours from the same speaker.

3.3 Feature extraction

Features should encode the information of pitch and speaker identity at the same time. We compare three features: cochleagram, log spectrogram and mel-frequency cepstral coefficients in our study. Cochleagram and log spectrogram are signal representations shown to be effective for speech separation, automatic speech recognition and speaker recognition. Unique harmonic structure of each speaker is reflected in both cochleagram and log spectrogram. Mel-frequency cepstral coefficients (MFCCs)

are widely used in speech processing, and they are investigated here as a representative cepstral feature found to be useful for pitch estimation long ago [88].

3.3.1 Cochleagram

To get the cochleagram feature, we first decompose the input signal in time-frequency domain by using a bank of 64 gammatone filters whose center frequencies range from 50 Hz to 8000 Hz. Gammatone filters model the impulse responses of auditory filters and are widely used [46]. We divide each subband signal into 20 ms frames with a 10 ms frame shift. The cochleagram is derived by computing the energy of each subband signal at each frame. We then loudness compress the cochleagram with a cubic root operation to get the final 64-dimensional cochleagram feature (for a Matlab implementation see http://web.cse.ohio-state.edu/pnl/shareware/cochleagram/).

3.3.2 Log spectrogram

To get the spectrogram feature, the signal is first divided into 32 ms frames with a 10 ms frame shift. The frame length of log spectrogram is longer than that of the other two features in order to produce a finer resolution of the frequency axis. We then apply a Hamming window to each frame and derive the spectrogram using 1024-point FFT. Lastly, we compute the logarithm of the amplitude spectrum, and pick bins 2-65 (corresponding to a frequency range up to 1000 Hz) as the frame-level feature vector. The dimensionality of this feature is 64, and it is proposed by Wohlmayr *et al.* [117] in their GMM-FHMM based multi-pitch tracker.

3.3.3 Mel-frequency cepstral coefficients

To compute MFCCs, we divide the input signal into 20 ms frames with a 10 ms frame shift. The power spectrogram is derived using short-time Fourier transform filtered by a Hamming window. Next we use a bank of 64 mel scale filters to convert the power spectrogram into mel scale. Lastly, logarithm compression and discrete cosine transform are applied to compute 31-dimensional MFCCs [10].

3.3.4 Incorporating temporal context

To make use of the temporal context, we concatenate neighboring frames into one feature vector. Denoting the feature vector extracted within frame m as $\hat{\mathbf{y}}_m$, we have:

$$\mathbf{y}_m = [\hat{\mathbf{y}}_{m-d}, \ldots, \hat{\mathbf{y}}_m, \ldots, \hat{\mathbf{y}}_{m+d}] \tag{3.2}$$

where d is chosen to be 5 (see Section 3.6.2).

3.4 DNN based pitch probability modeling

DNNs have been successfully applied in various speech processing applications. In this section, we first introduce two types of DNN for posterior probability estimation. Next we extend the models to relax practical constraints.

3.4.1 Speaker-dependent DNNs

The goal of DNNs is to model the posterior probability that a pair of pitch states occurs at frame m, i. e., $p(x_m^1, x_m^2|\mathbf{y}_m)$. However, this would be difficult without the prior knowledge of the underlying speakers. We first focus on training speaker-dependent DNNs to model the posterior probability.

According to the chain rule in probability theory:

$$p(x_m^1, x_m^2|\mathbf{y}_m) = p(x_m^1|\mathbf{y}_m)p(x_m^2|x_m^1, \mathbf{y}_m) \tag{3.3}$$

we can estimate $p(x_m^1|\mathbf{y}_m)$ and $p(x_m^2|x_m^1, \mathbf{y}_m)$ in turn to get $p(x_m^1, x_m^2|\mathbf{y}_m)$. In this study, we estimate the pitch-state probability of speaker one $p(x_m^1|\mathbf{y}_m)$ by training a DNN. The input layer of the DNN corresponds to the frame-level feature vector of the mixture. There are four hidden layers in the DNN, and each hidden layer has 1024 rectified linear units (ReLU) [30]. The reason we choose ReLU instead of sigmoid is that it alleviates the overfitting problem, leading to faster and more effective training/adaptation. The output layer has 68 softmax output units, denoted by $(O_1^1, O_1^2, \ldots, O_1^{68})$, where O_1^j estimates $p(x_m^1 = s^j|\mathbf{y}_m)$. Hence there are 67 '0's and a '1' in the desired output. The value '1' corresponds to the frequency bin of the groundtruth pitch. We use cross-entropy as the cost function. The standard back-propagation algorithm and dropout regularization [44] are used to train the network, with no pretraining. We adopt mini-batch stochastic gradient descent along with a momentum term (0.9) for the optimization. The choice of DNN parameters is justified in Section 3.6.2. The training data contain mixtures of speaker one and a set of interfering speakers.

Fig. 3.2 compares the groundtruth and estimated pitch-state probabilities of speaker one in a female-female test mixture. As shown in the figure, the DNN rather accurately models the conditional probability of x_m^1, even without knowing x_m^2. Therefore the same type of DNN can be used to model $p(x_m^1|x_m^2, \mathbf{y}_m)$ or $p(x_m^2|x_m^1, \mathbf{y}_m)$.

In the next step, we train another DNN to model $p(x_m^2|x_m^1, \mathbf{y}_m)$ using exactly the same structure and training methodology as for the first DNN. The output of the DNN is denoted by $(O_2^1, \ldots, O_2^{68})$. The original posterior probability $p(x_m^1, x_m^2|\mathbf{y}_m)$

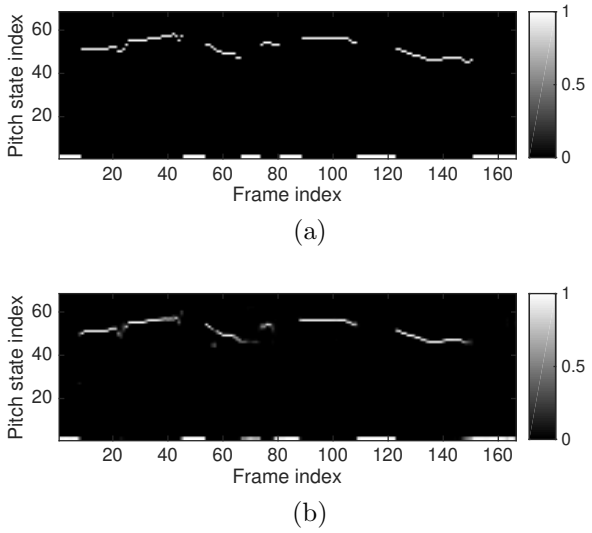


Figure 3.2: Pitch probability modeling of the first speaker in a female-female mixture at 0 dB. (a) Groundtruth probabilities of pitch states. (b) Probabilities of pitch states estimated by a DNN.

can then be obtained by:

$$p(x_m^1 = s^i, x_m^2 = s^j | \mathbf{y}_m) = O_1^i O_2^j \tag{3.4}$$

Because we train a DNN for each enrolled speaker, we denote this model as the speaker-dependent DNN (SD-DNN).

3.4.2 Speaker-pair-dependent DNNs

A speaker-pair-dependent DNN (SPD-DNN) is a DNN trained on a specific pair of speakers. The structure of an SPD-DNN is quite similar to that of an SD-DNN. The input layer corresponds to the frame-level feature vector. There are four hidden layers with 1024 ReLU units. Instead of estimating the probability for only one speaker, we concatenate the pitch-state probabilities of the other speaker into the DNN output. The resulting output layer has 136 units, denoted by $(O_1^1, \ldots, O_1^{68}, O_2^1, \ldots, O_2^{68})$. To

correctly model the probability distribution, the activation function of the output layer is a softmax function. Assuming that output units before applying the activation function have values $(v_1^1, \ldots, v_1^{68}, v_2^1, \ldots, v_2^{68})$, we have:

$$O_i^j = \frac{exp(v_i^j)}{\sum_{k=1}^{68} exp(v_i^k)}, \quad \text{for } i = 1 \text{ or } 2,\ 1 \leq j \leq 68 \tag{3.5}$$

Other training details exactly follow SD-DNNs. The posterior probability of pitch states is estimated by:

$$p(x_m^1 = s^i, x_m^2 = s^j | \mathbf{y}_m) = O_1^i O_2^j \tag{3.6}$$

Because SPD-DNNs are trained on speaker pairs, they should accurately capture the underlying speaker information. On the other hand, for a system with N speakers enrolled, we need to train N SD-DNNs, but $\frac{N(N-1)}{2}$ SPD-DNNs.

3.4.3 Extensions

SD-DNNs and SPD-DNNs utilize detailed speaker information to estimate the posterior probability of pitch states. In this section, we introduce extensions to relax their practical constraints.

Gender-pair-dependent DNN

SD-DNNs and SPD-DNNs are not applicable to untrained speakers. To deal with this constraint, we extend our speaker-dependent models to gender-dependent ones. In this way, only the genders of the two underlying speakers are needed during testing.

A straightforward way to design a gender-dependent model is to follow the structure of SD-DNNs and train two DNNs for male and female speakers, respectively. This idea works well for male-female mixtures, but can not distinguish the two speakers of

the same gender. Therefore we build our gender-dependent model by extending SPD-DNNs to gender-pair-dependent DNNs or GPD-DNNs. We train three GPD-DNNs for different gender pairs: male-female, male-male and female-female. The structure of a GPD-DNN is chosen to be the same as an SPD-DNN for simplicity. For the male-female GPD-DNN, the pitch-state probabilities of the male speaker correspond to the first 68 output units, and the female speaker the remaining output units. For same-gender GPD-DNNs, the first 68 output units correspond to the speaker with lower average pitch, and the other output units correspond to the speaker with higher average pitch. Although this layout may lead to incorrect speaker assignment at some frames, it provides a reasonable way to distinguish two speakers with the little information available. Other training aspects exactly follow SPD-DNNs.

Adaptation of GPD-DNNs with limited training data

SD-DNNs and SPD-DNNs would overfit if we could not collect enough training data. One way to address this problem is to perform speaker adaptation of GPD-DNNs with limited data. Speaker adaptation of DNNs has been studied in automatic speech recognition. Two typical approaches include incorporating speaker-dependent information into DNN's input [1, 93] and regularized retraining [71, 121]. In the first approach, speaker dependent information, like i-vectors and speaker codes, is incorporated into the input of DNNs and the original features are projected into a speaker-normalized space. In regularized retraining, the weights of DNNs are modified using the adaptation data. To ensure that the adapted model does not deviate too much from the original model, a regularization term is added to the cost function. Both approaches substantially improve the performance of unadapted DNNs.

We use regularized retraining to perform speaker adaptation. For each new speaker pair, we retrain all the weights of the corresponding GPD-DNN on limited adaptation data with a relatively small learning rate (0.001) and a weight decay (L_2 regularization) of 0.0001. Other training aspects follow those for training SPD-DNNs.

Multi-ratio training

Utterances of the two speakers in a mixture usually have different energy ratios. A ratio mismatch between training and test may result in performance degradation for supervised algorithms. Under the framework of GMM-FHMM, Wohlmayr and Pernkopf [115] alleviate this problem by adding a gain parameter to the mean vectors of each GMM. An expectation-maximization (EM) based algorithm is then performed to estimate the gains for each test mixture. They further extend the EM-like framework to adapt model parameters to untrained acoustic environments and speakers [116]. However, it is unclear how to apply these techniques to DNNs.

Generally speaking, the performance of supervised learning is sensitive to the information contained in the training set. Therefore a simple and effective way for improving generalization is to enlarge the training set by including various acoustic conditions [14]. In this study, we perform multi-condition training by creating mixtures at different speaker energy ratios, denoted by multi-ratio training. The resulting DNNs are denoted by ratio-adapted DNNs. The details of multi-ratio training are given in Section 3.6.

3.5 Factorial HMM inference

Once all posterior probabilities are estimated by DNNs, we use a factorial HMM to infer the most likely pitch tracks. A factorial HMM is a graphical model that

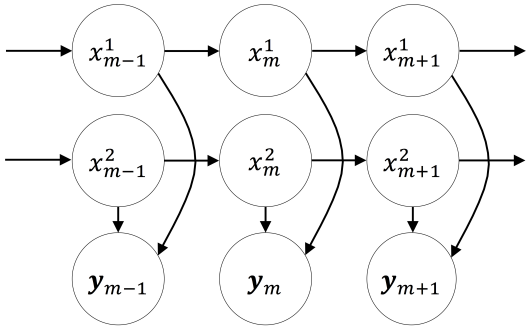


Figure 3.3: A factorial HMM with two Markov chains.

contains several Markov chains [29]. In this study, we only discuss the case of two Markov chains, as shown in Fig. 3.3.

The hidden variables (x^1_m, x^2_m) are the pitch states of two speakers, and the observation variable is the feature vector $\mathbf{y}_m$. The Markov assumption implies that $\mathbf{y}_m$ is independent of all variables given x^1_m and x^2_m. Assuming the total number of frames is M, we denote the sequence of variables in boldface: $\mathbf{X} = \bigcup_{m=1}^{M}\{x^1_m, x^2_m\}$, $\mathbf{Y} = \bigcup_{m=1}^{M}\{\mathbf{y}_m\}$. The overall joint probability of the model is given by:

$$p(\mathbf{X}, \mathbf{Y}) = p(x^1_1)p(x^2_1)p(\mathbf{y}_1|x^1_1, x^2_1)\prod_{m=2}^{M} p(x^1_m|x^1_{m-1})p(x^2_m|x^2_{m-1})p(\mathbf{y}_m|x^1_m, x^2_m) \quad (3.7)$$

Prior probabilities and transition matrices of the hidden variables are computed from single-speaker recordings in the training set either speaker-dependently (for SD-DNNs and SPD-DNNs) or gender-dependently (for GPD-DNNs). To avoid a probability of zero, Laplace smoothing is applied during the computation, where we add one to each possible observation. The emission probability can be computed using the estimated posterior probability and Bayes rule:

$$p(\mathbf{y}_m|x^1_m, x^2_m) = \frac{p(x^1_m, x^2_m|\mathbf{y}_m)p(\mathbf{y}_m)}{p(x^1_m)p(x^2_m)} \quad (3.8)$$

where $p(\mathbf{y}_m)$ is a constant for all feature vectors.

Once all probabilities are derived, we apply the junction tree algorithm to infer the most likely sequence of pitch states. The first step of this algorithm is to convert the directed graphical model to an undirected graphical model. In the next step, the nodes in the undirected graph are arranged to form a junction tree, where belief propagation is performed. For more details on the junction tree algorithm, we refer the interested reader to [61] and [117]. The time complexity of the junction tree algorithm is $O(2 \times 68^3 \times M)$ in our study. We then convert derived pitch states to the mean frequencies of the corresponding frequency bins. Because the resulting frequencies correspond to a rough sampling of possible pitch frequencies, we use a moving average window of length three to smooth frequencies and get final pitch estimates.

3.6 Evaluations and comparisons

3.6.1 Corpus and error measurement

For evaluations, we first use the GRID database [20], which is also used in [117] hence facilitating our comparisons. The corpus consists of 1000 sentences spoken by each of 34 speakers (18 male, 16 female). Two male and two female speakers (No. 1, 2, 18, 20, same as [117]), denoted by MA1, MA2, FE1 and FE2, are selected to train and test the proposed methods, except for GPD-DNNs which are tested on the same four speakers but trained on another set of speakers. We denote these four speakers as Set One. For each speaker in Set One, 950 sentences are selected for training, 40 sentences are used for choosing the best DNN weights during training, and the remaining ten sentences are used for testing. Note that all test sentences used in [117]

are also included in our test set. Another ten male and nine female speakers (No. 3, 5, 6, 9, 10, 12, 13, 14, 17, 19; 4, 7, 11, 15, 16, 21, 22, 23, 24)[1] are used in the training of SD-DNNs and GPD-DNNs, where again for each speaker we select 950 sentences for training, and 40 sentences for selecting the best DNN weights. We denote these nineteen speakers as Set Two. Reference pitches are extracted from single-speaker sentences using RAPT [101], which outperforms other pitch trackers on clean speech signals [23]. Although RAPT makes minor mistakes like pitch halving and doubling, these errors are not severe. Since the main challenge in multi-pitch tracking is the interference of another pitched sound, we treat thus derived pitch as groundtruth.

To mix two sentences u_t^1 and u_t^2, we first select a speaker ratio R in dB, and amplify one of the speakers by R dB. A mixture with a speaker ratio of R dB is created by combining the resulting sentences using: $v_t = 10^{R/20}u_t^1 + u_t^2$ or $v_t = u_t^1 + 10^{R/20}u_t^2$. Note that if we choose a speaker ratio of 0 dB, the two equations to derive v_t are the same. For comparison reasons, we use a matched speaker ratio of 0 dB in the training and test of SD-DNNs, SPD-DNNs, GPD-DNNs and adaptation of GPD-DNNs. Unmatched speaker ratios are used to test multi-ratio training. The details of the training and test set are as follows:

- SD-DNNs: training mixtures are created by mixing each sentence of the target speaker in Set One with 60 random sentences in Set Two at 0 dB. Thus there are 57000 training mixtures created for every target speaker. The test is conducted within Set One. We exhaustively mix test sentences for each speaker pair in Set One at 0 dB, resulting in a total of $10\times10\times6 = 600$ test mixtures.

[1]This list follows the gender-dependent training set in [117]. However, since speaker No. 8 is wrongly marked as a female speaker in their training set, we eliminate this speaker in our study. Results show that the elimination leads to little performance change.

- SPD-DNNs: for each speaker pair in Set One, we build the training set by mixing sentences of the two speakers at 0 dB. We make sure that each sentence of one speaker is randomly mixed with 60 sentences of the other speaker. Therefore 57000 mixtures are created to train each speaker pair. We use the same test set as for SD-DNNs.

- GPD-DNNs: the training is conducted within Set Two. For the male-female case, we randomly create 57000 mixtures at 0 dB. For the same-gender case, we divide the training speakers into two groups, with the first group having higher average pitch. We then create 57000 training mixtures by randomly mixing the utterances in the first group and those in the second group at 0 dB. The same test set is used as for SD-DNNs.

- Adaptation of GPD-DNNs: For each speaker pair in Set One, we randomly select 100 mixtures from the SPD-DNN's training set as the adaptation data. The same test set is used as for SD-DNNs.

- Multi-ratio training: the training is conducted for both SD-DNNs and SPD-DNNs. Mixtures are no longer created at 0 dB in this experiment. Instead, we randomly amplify one of the two speakers with a random ratio out of R = {-12,6,0,6,12} dB for each training mixture. As for the test set, we alternately amplified one of the two sentences with a ratio out of R = {-15,-12,-9,-6,−3,0,3,6,9,12,15} dB, which gives 10×10×6×2 = 1200 mixtures at each speaker energy ratio, and 13200 mixtures in total; note that each mixture at 0 dB appears twice in test.

In addition, we test our proposed methods using the FDA database [5] where the groundtruth pitches are derived from laryngograph data.

We evaluate pitch tracking results using the error measure proposed in [117], which jointly evaluates the performance in terms of pitch accuracy and speaker assignment. Assuming that the ground truth pitch tracks are F_m^1 and F_m^2, we globally assign each estimated pitch track to a groundtruth pitch track based on the minimum mean square error and denote the assigned estimated pitch tracks as f_m^1 and f_m^2. The pitch frequency deviation of speaker i, $i \in \{1, 2\}$, is:

$$\Delta f_m^i = \frac{|f_m^i - F_m^i|}{F_m^i} \tag{3.9}$$

The voicing decision error E_{ij}, $i \neq j$, denotes the percentage of time frames where i pitch points are wrongly detected as j pitch points. For each speaker i, the permutation error E_{Perm}^i is set to one at time frames where the voicing decision for both estimates is correct, but Δf_m^i exceeds 20%, and f_m^i is within the 20% error bound of the other reference pitch, i. e., the error is due to incorrect speaker assignment. The overall permutation error E_{Perm} is the percentage of time frames where either E_{Perm}^1 or E_{Perm}^2 is one. Next, for each speaker i, the gross error E_{Gross}^i is set to one at time frames where the voicing decision for both estimates is correct, but Δf_m^i exceeds 20% with no permutation error. The overall gross error E_{Gross} is the percentage of time frames where either E_{Gross}^1 or E_{Gross}^2 is one. The fine detection error E_{Fine}^i is defined as the average of Δf_m^i in percent at time frames where Δf_m^i is smaller than 20%. $E_{Fine} = E_{Fine}^1 + E_{Fine}^2$. The total error is used as the overall performance measure:

$$E_{total} = E_{01} + E_{02} + E_{10} + E_{12} + E_{20} + E_{21} + E_{Perm} + E_{Gross} + E_{Fine} \tag{3.10}$$

3.6.2 Parameter selection

Because all proposed DNNs have similar structure, we conduct parameter selection for SPD-DNNs only. The best performing parameters are used in other models. We use a new pair of male speakers (No. 26 and 28 in the GRID corpus) as the development set. For each speaker, 950 sentences are used for training, 40 sentences are used for choosing the best DNN weights during training and 10 sentences are used for test. Besides the matched 0 dB training and test condition, we also train the SPD-DNN with multi-ratio training. The details of the training and test set follow Section 3.6.1. The results of multi-ratio training are averaged across all speaker ratios.

The size of the training set has strong impact on DNN's performance. We create five training sets by randomly mixing each sentence of one speaker with 5, 20, 40, 60 and 80 sentences of the other speaker, resulting in 4750, 19000, 38000, 57000, 76000 mixtures. An SPD-DNN is trained for each training set. The results are given in Fig. 3.4a. In general, the total error decreases with the increase of the training size, and the improvement becomes small when the training size reaches 57000. Taking the computational cost into consideration, we choose 57000 training mixtures in the subsequent experiments.

Features are important to the system. As shown in Fig. 3.4b, we compare three features: cochleagram, log spectrogram and MFCCs. We adopt the cochleagram feature in the subsequent experiments as it outperforms other two features.

To incorporate temporal dynamics, a context window is applied to the input feature. We have explored three values of the window size d (see Section 3.3.4). In Fig. 3.4c, the total error substantially decreases when d is increased from 3 to 5, and

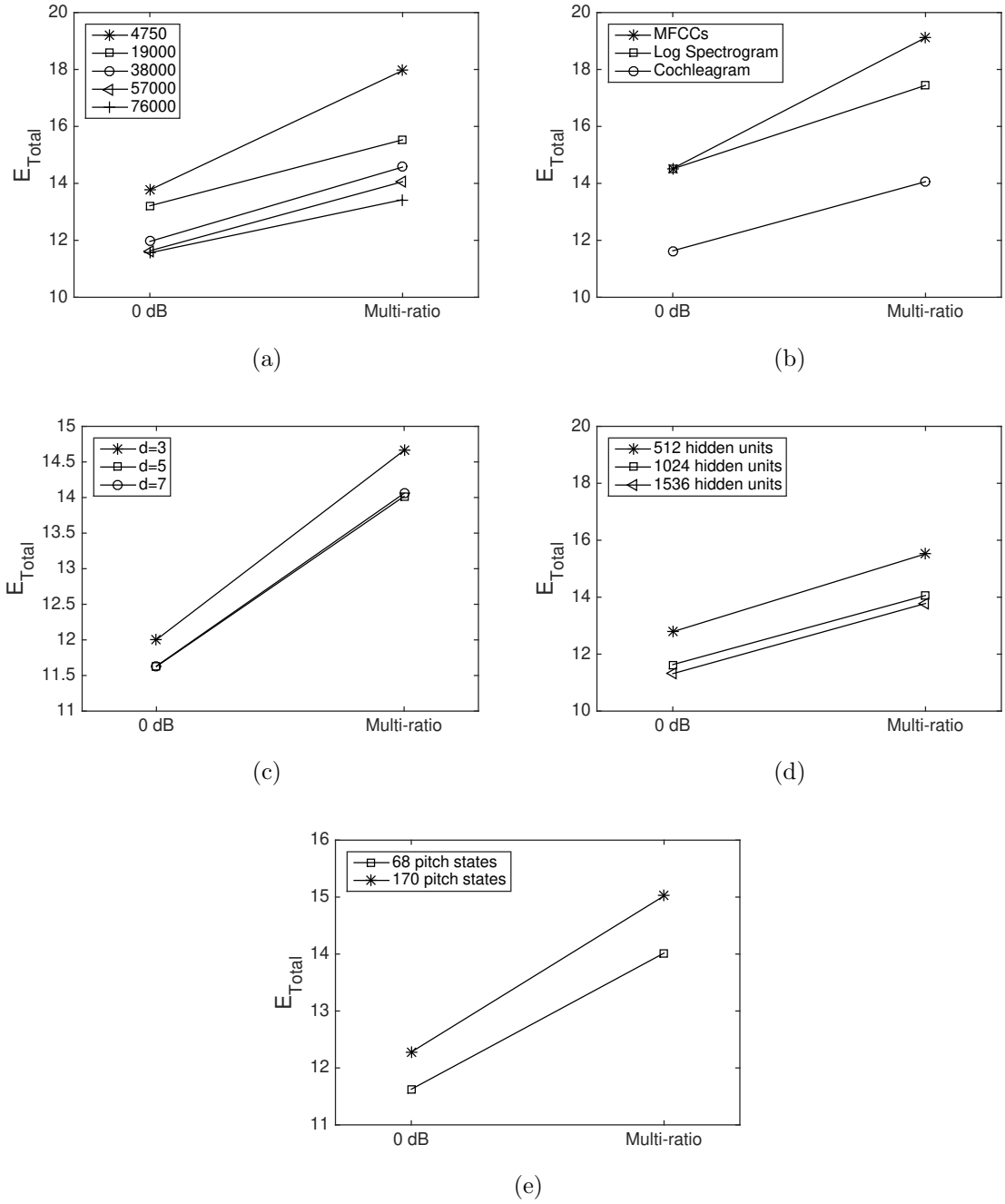


Figure 3.4: Average E_{total} of SPD-DNNs with different (a) sizes of training set, (b) features, (c) sizes of context window, (d) numbers of hidden units, (e) numbers of pitch states.

remains the same when d reaches 7. Therefore we choose $d = 5$ for the cochleagram feature.

Next, we investigate the number of hidden units used in SPD-DNNs. Three numbers are compared: 512, 1024 and 1536. As shown in Fig. 3.4d, the total error is reduced by more than 1.1% when the number is increased from 512 to 1024. However, further increasing the number of hidden units does not significantly boost the performance.

As described in Section II, we follow [35] to use 68 pitch states to quantize the frequency range from 60 to 404 Hz. Another speaker-dependent multi-pitch tracking algorithm [117] quantizes the frequency range from 80 to 500 Hz into 170 pitch states. We compare the two pitch quantizations using SPD-DNNs. The results are given in Fig. 3.4e. Basically, more pitch states do not lead to better performance, probably because the frequency resolution with 170 pitch states is too fine for DNNs to make accurate probability estimates.

Other parameters, including the type of activation functions, the number of hidden layers, learning rate and mini-batch, are also chosen from the same development set.

3.6.3 Results and comparisons

We present our results, and compare with two state-of-the-art multi-pitch trackers: Jin and Wang [60] and Wohlmayr *et al.* [117]. Jin and Wang's approach is designed for noisy and reverberant signals. They use correlogram to select reliable channels and track continuous pitch contours with an HMM. Wohlmayr *et al.* model

speakers with GMMs, and use a mixture maximization model to obtain a probabilistic representation of pitch states. An FHMM is then applied to track pitch over time. The GMM-FHMM structure could also be extended to be gender-dependent. We denote Wohlmayr *et al.*'s speaker-dependent and gender-dependent models as Wohlmayr *et al.* SD and Wohlmayr *et al.* GD, respectively. Wohlmayr *et al.* train their models on the GRID database with groudtruth pitches obtained also by RAPT. The test mixtures used in their study are included in our test set, and we directly adopt the trained GMM/FHMM models posted on their website for comparison.

Table 3.1: E_{Total} for different multi-pitch trackers on 600 test mixtures of the GRID Corpus.

		E_{01}	E_{02}	E_{10}	E_{12}	E_{20}	E_{21}	E_{Gross}	E_{Fine}	E_{Perm}	E_{Total}
Jin and Wang	Mean	4.54	1.25	6.97	5.51	1.94	**12.81**	4.80	6.93	6.47	**51.21**
	Std	2.34	1.38	3.55	3.33	2.16	5.54	4.65	3.17	5.34	11.71
Wohlmayr *et al.* SD	Mean	1.81	0.06	5.89	2.68	1.39	**10.81**	0.93	2.79	0.37	**26.73**
	Std	1.64	0.26	3.42	2.18	2.06	5.26	1.14	0.73	0.79	9.49
SD-DNN	Mean	1.98	0.13	2.01	5.70	0.07	**2.74**	0.72	2.32	1.01	**16.69**
	Std	1.61	0.40	2.02	5.57	0.27	2.06	1.25	0.84	2.23	7.90
SPD-DNN	Mean	1.69	0.07	1.59	3.19	0.05	**2.55**	0.52	1.95	0.15	**11.77**
	Std	1.42	0.26	1.54	2.09	0.24	1.94	0.91	0.33	0.54	3.29

We first evaluate the SD-DNN and SPD-DNN based methods. Table 3.1 compares the SD-DNN and SPD-DNN based methods with the other multi-pitch trackers on 600 test mixtures. Speaker-dependent approaches perform substantially better than the speaker-independent approach, and our SD-DNN and SPD-DNN based methods cut E_{Total} by more than 10% compared to Wohlmayer *et al.* SD. The major improvement in E_{Total} comes from E_{21}, which implies that our methods estimate pitch more accurately when the two speakers are both voiced. The SPD-DNN method

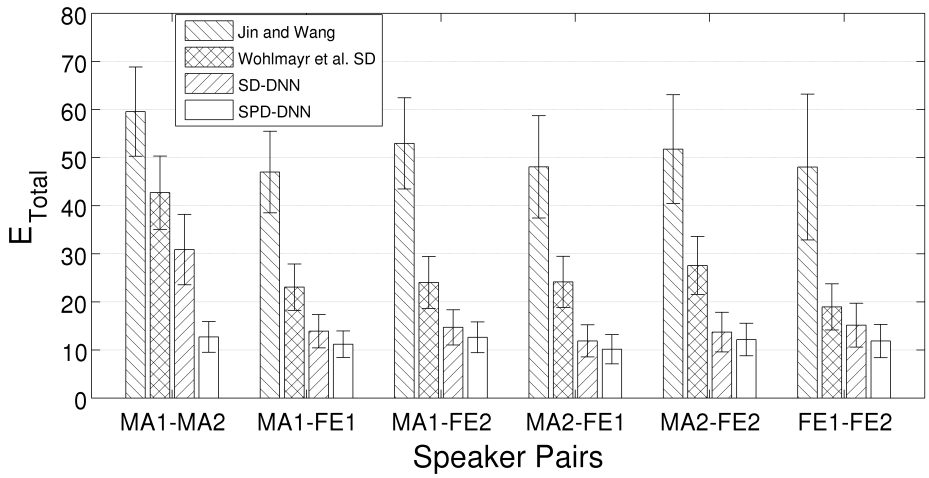


Figure 3.5: E_{total} of different approaches tested on six pairs of speakers. Error bars depict the mean and standard deviation of a method on the test mixtures of a given speaker pair.

performs better than the SD-DNN method, which is not surprising as SPD-DNNs are trained on individual speaker pairs. We further illustrate E_{Total} for each of the six speaker pairs in Fig. 3.5. As shown in the figure, our methods have lower errors across all pairs. SD-DNNs and SPD-DNNs perform comparably on five speaker pairs, and the latter achieve significantly lower E_{Total} on the most difficult pair of MA1-MA2. Fig. 3.6 illustrates pitch tracking results on a test mixture of MA1-MA2. Jin and Wang's approach fails to assign pitches to the underlying speakers. Wohlmayr *et al.*'s approach works better in terms of speaker assignment, but performs poorly when two pitch tracks are close to each other. Moreover, their resulting pitch contours lack continuity. The SD-DNN produces much smoother pitch contours. However, it still has incorrect speaker assignment at a few frames. The SPD-DNN generates very good pitch tracks in both pitch accuracy and speaker assignment.

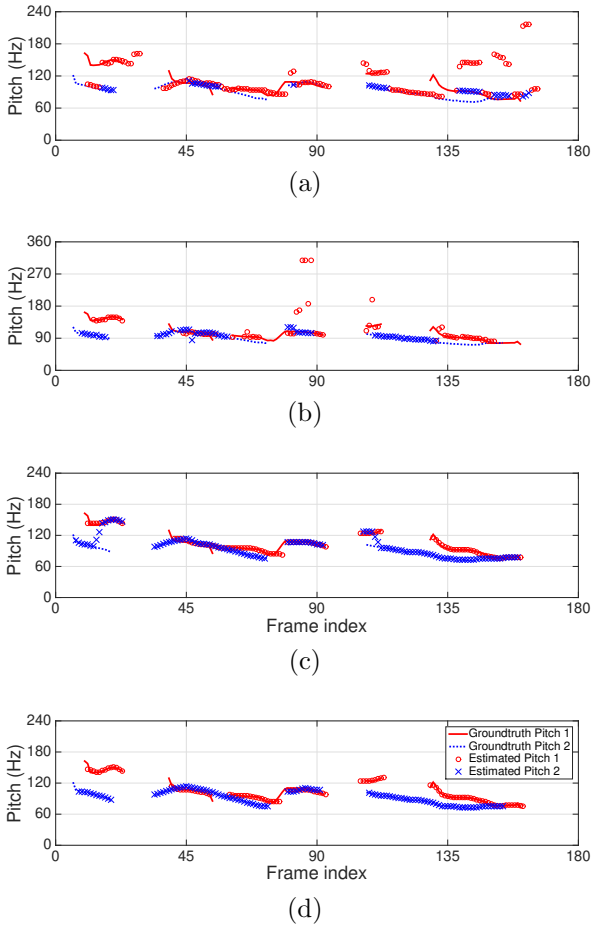


Figure 3.6: Multi-pitch tracking results on a test mixture (pbbv6n and priv3n) of the MA1-MA2 speaker pair. (a) Groundtruth pitch (lines and dotted lines) and estimated pitch (circles and crosses) by Jin and Wang. (b) By Wohlmayr *et al.* SD. (c) By SD-DNN. (d) By SPD-DNN.

To further analyze the above-mentioned improvement achieved by our proposed methods, we compare our SD-DNN based method with Wohlmayr *et al.* SD using the same feature, namely the log spectrogram feature described in Section 3.3.2, and the same training data, i. e., 497 training utterances per speaker. Specifically, we train the SD-DNN based method using three settings: (1) 497 training utterances per speaker with log spectrogram feature, (2) 497 training utterances per speaker with

Table 3.2: Average E_{Total} for SD-DNN and Wohlmayr *et al.* SD on 600 test mixtures of the GRID Corpus.

Training utterances per speaker	497		950
Feature type	Log spectrogram	Cochleagram	Cochleagram
SD-DNN	19.02	17.22	**16.69**
Wohlmayr *et al.* SD	26.73	-	-

cochleagram feature, and (3) 950 training utterances per speaker with cochleagram feature (the proposed training setting). Results on the 600 test mixtures are shown in Table 3.2. When using exactly the same feature and training data, the SD-DNN based method significantly outperforms Wohlmayr *et al.* SD. If we replace SD-DNN's input feature with cochleagram, the total error further decreases. Lastly, increasing the training size slightly boosts SD-DNN's performance. In conclusion, although features and training sizes have an effect, the use of DNNs makes the most contribution to performance gains.

Next, we evaluate three extensions to the previous models. Fig. 3.7 shows the performance of the GPD-DNN based method. It significantly outperforms Wohlmayer *et al.*'s gender dependent model on all speaker pairs. The average E_{Total} of GPD-DNN is 15.89% lower than Wohlmayr *et al.*'s gender-dependent model, and even 5.46% lower than Wohlmayr *et al.*'s speaker-dependent model. However, the performance gap between GPD-DNNs and SD-DNNs/SPD-DNNs is larger than 4.5%. Therefore one should use SD-DNN/SPD-DNN based methods when speaker-dependent information is available.

Fig. 3.8 shows the performance of GPD-DNN adaptation. Four models are compared across all speaker pairs: (1) GPD-DNNs, (2) SPD-DNNs directly trained

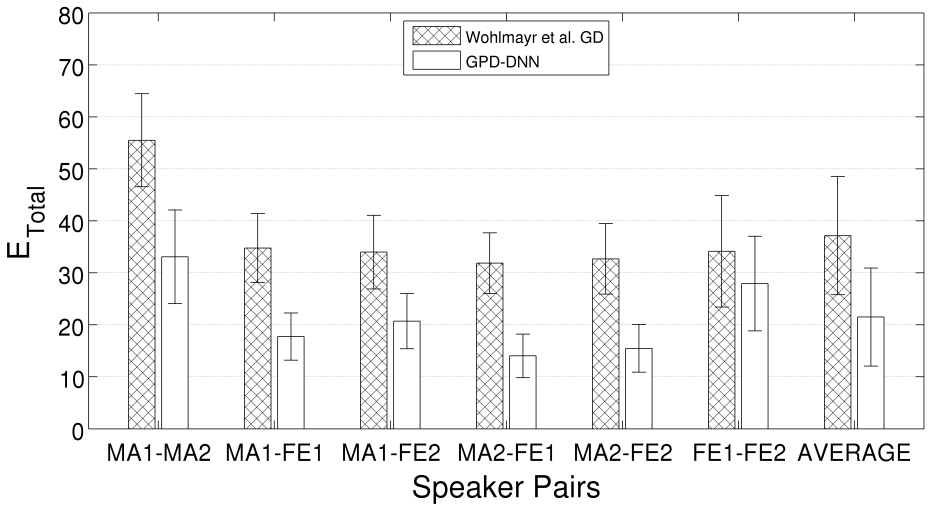


Figure 3.7: E_{total} of gender-dependent approaches. Error bars depict the mean and standard deviation of a method on the test mixtures of a speaker pair.

with 100 mixtures per speaker pair, (3) GPD-DNNs adapted with 100 mixtures per speaker pair, (4) SPD-DNNs trained with 57000 mixtures per speaker pair. As shown in the figure, SPD-DNNs trained with limited data perform better than GPD-DNNs on same-gender mixtures, but worse than GPD-DNNs on different-gender mixtures. GPD-DNN adaptation consistently outperforms the first two methods, resulting in 5% reduction in average E_{Total}. The results indicate the power of GPD-DNN adaptation for small training sizes.

Generalization to different speaker energy ratios is crucial to supervised multi-pitch trackers. Fig. 3.9 shows the performance of SD-DNN, SPD-DNN, and Wohlmayr *et al.*'s speaker-dependent models at various speaker ratios. All models are trained at 0 dB, and results are averaged across all speaker pairs at each speaker ratio. As shown in the figure, the total error increases significantly when the speaker ratio deviates from 0 dB. Errors are not symmetric with respect to 0 dB, as we only scale the level

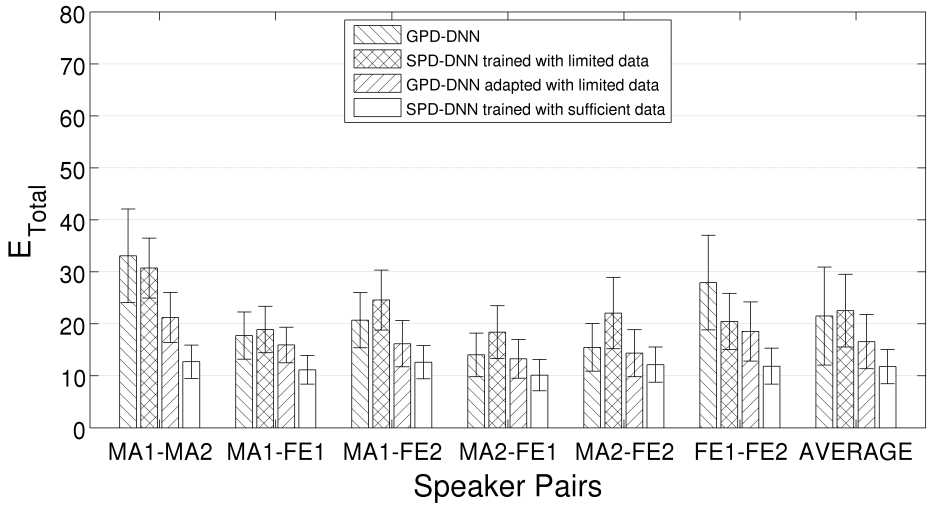


Figure 3.8: Performance of GPD-DNN adaptation. Error bars depict the mean and standard deviation of a method on the test mixtures of a speaker pair.

of one speaker in order to create a specified ratio. For Wohlmayr *et al.*'s speaker-dependent model, when the speaker ratio is positive, the mixture becomes dominated by the amplified speaker, misleading the GMM of the weak speaker. For the SD-DNN and SPD-DNN based methods, it is hard for DNNs to recognize the weak speaker when the speaker ratio is too low. We then apply multi-ratio training for SD-DNNs and SPD-DNNs, and compare them with Jin and Wang's unsupervised multi-pitch tracker as well as the gain-adapted version of Wohlmayr *et al.*'s speaker-dependent models [115]. Note that, unlike multi-ratio training, gain adaptation in [115] uses an expectation-maximization based framework to estimate gains in test mixtures, thus no additional training is needed. The results are given in Fig. 3.10. The performance of multi-ratio trained DNNs remains high across all speaker ratios. At 0 dB, multi-ratio trained SD-DNNs and SPD-DNNs produce only 0.03% and 0.34% higher errors

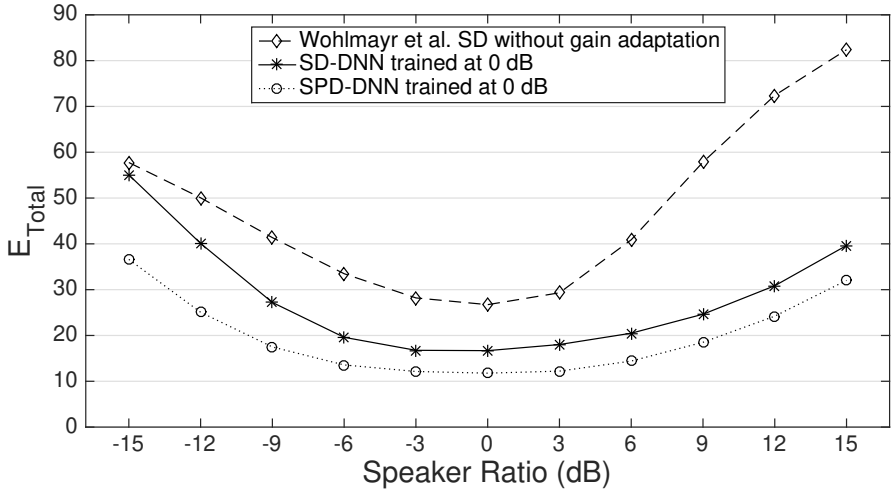


Figure 3.9: Results of different approaches tested on eleven speaker ratios. Each data point represents E_{total} averaged across 1200 test mixtures.

than SD-DNNs and SPD-DNNs trained in the matched 0 dB condition, indicating their strong generalization ability.

Noise robustness is also an important issue in multi-pitch estimation. We evaluate Jin and Wang's model, Wohlmayr *et al.*'s speaker dependent model, the SD-DNN based model and the SPD-DNN based model, when a speech shape noise (SSN) and a babble noise are mixed with two-speaker utterances. SSN is a stationary noise with no pitch, and babble noise is nonstationary with pitched portions. Specifically, we generate 100 test mixtures of MA1-MA2 at the speaker ratio of 0 dB. The test mixtures are then mixed with SSN and babble noise at the SNR of 5, 10, 20 and Inf dB. Here the SNR refers to the ratio of two-speaker-mixture power to the noise power, and Inf dB corresponds to the noise-free condition. Importantly, no retraining is performed for any system. The multi-pitch tracking results in background noise

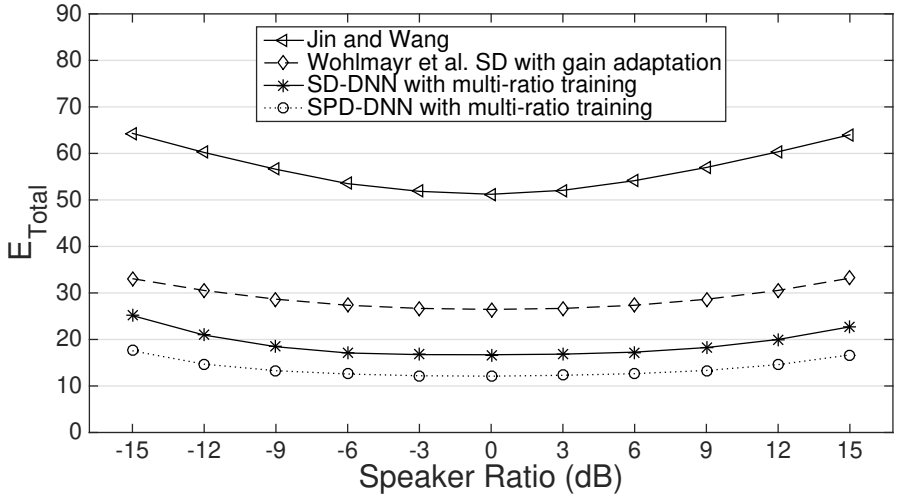
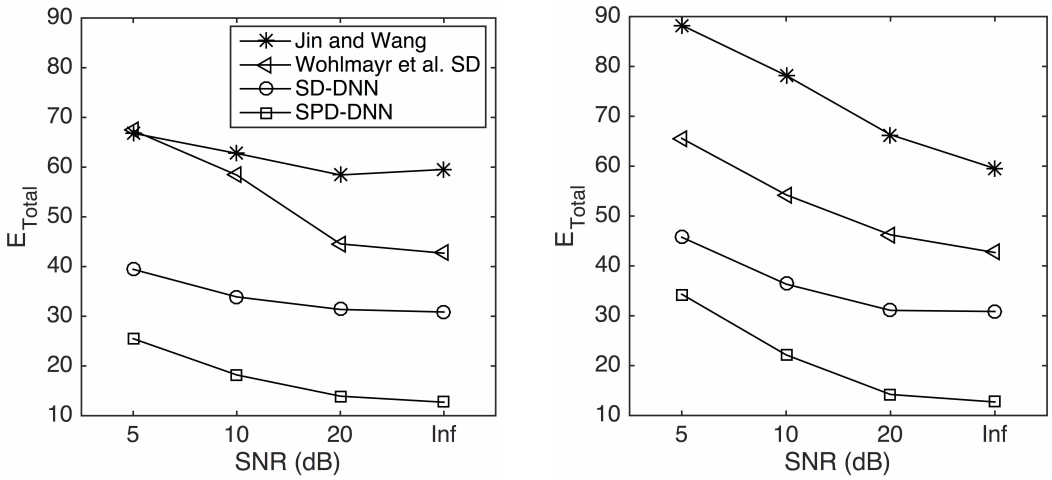


Figure 3.10: Results of different approaches tested on eleven speaker ratios. Each data point represents E_{total} averaged across 1200 test mixtures.

are given in Fig. 3.11. As shown in the figure, our methods remain robust to both kinds of noise, and outperform the comparison models.

In the above experiments, we use RAPT to extract the groundtruth pitch from single speaker recordings, which is not error-free as mentioned previously. We now evaluate our methods on the FDA database [5], where the groundtruth pitch is directly given by laryngograph data. The corpus consists of recordings of 50 sentences by each of two speakers (a male and a female). For each speaker, we choose 40 sentences for testing and 40 test mixtures are created by mixing the test sentences at 0 dB. Because the dataset is not large enough for training SD-DNNs and SPD-DNNs, we conduct experiments with GPD-DNN and GPD-DNN adaptation. A ratio-adapted GPD-DNN trained on the GRID database is used for pitch-state probability estimation. We also perform speaker adaptation of the GPD-DNN with 10 adaptation sentences per speaker, i. e., 10×10 adaptation mixtures. We compare the two methods with



(a) (b)

Figure 3.11: E_{Total} of different approaches tested on the MA1-MA2 speaker pair mixed with (a) speech shape noise, (b) babble noise.

Jin and Wang's speaker-independent model as well as the gain-adapted version of Wohlmayr *et al.*'s gender-dependent model. E_{Total} of different approaches is shown in Fig. 3.12. Results indicate that our GPD-DNN based method outperforms other approaches. The adaptation of the GPD-DNN further reduces the average total error by 8.69%.

In addition to E_{Total}, we use another metric to compare the performance in this experiment: overall multi-pitch accuracy used by [25]. To compute this accuracy, we first assign each estimated pitch track to a groundtruth pitch track. For each estimated pitch track, we call a pitch estimate at a frame correct if it deviates less than 10% from its corresponding groundtruth pitch. The overall multi-pitch accuracy is defined as:

$$Accuracy = \frac{TP}{TP + FP + FN} \tag{3.11}$$

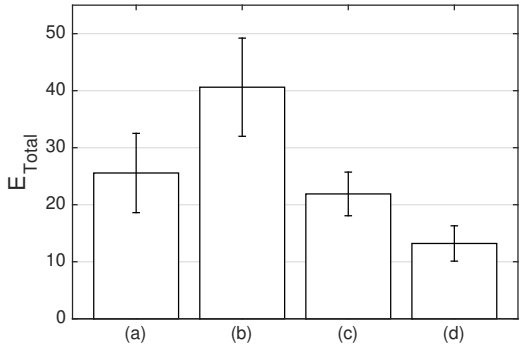


Figure 3.12: Results of different approaches tested on the FDA corpus. (a) Jin and Wang. (b) Wohlmayr *et al.* GD with gain adaptation. (c) Ratio-adapted GPD-DNN. (d) Speaker adaptation of GPD-DNN. Error bars depict the mean and standard deviation of a method on the test mixtures.

where TP (true positive) is the total number of correctly estimated pitches, FP (false positive) is the total number of pitches that appear in some estimated pitch track but do not belong to the corresponding groundtruth pitch track, and FN (false negative) denotes the total number of pitches that appear in some groundtruth pitch track but do not belong to the corresponding estimated pitch track. Different assignments of estimated pitch tracks give us different accuracies, and we choose the highest value to represent the overall accuracy. Similar to Fig. 3.12, the GPD-DNN and GPD-DNN adaptation achieve accuracies of 70.02% and 82.61%. The other two approaches have accuracies lower than 50%.

Table 3.3: Running time comparion for different approaches.

	Jin-Wang	Wohlmayr *et al.* SD	SD-DNN	SPD-DNN
Time (s)	7.77	20.12	0.60	0.43

Lastly, we compare the computational complexity of different approaches. We directly use Jin and Wang's [60] and Wohlmayer *et al.*'s [117] program on their authors' websites. Jin and Wang's [60] program is implemented in Java, and Wohlmayer *et al.*'s [117] program is implemented in Matlab. Our program is a mixture of Matlab and Caffe (a deep learning framework). One hundred mixtures with the total length of 179.7 s are created for this evaluation. The test is performed on a machine with an Intel i7-4770k CPU (3.5 GHz) and 32 GB memory. All computations are performed on the CPU within a single thread. Table 3.3 shows the average processing time per one second mixture. Results indicate that our methods are a lot more efficient. There are two main reasons why Wohlmayr *et al.* SD is slower. First, the number of pitch states used in Wohlmayr *et al.*'s SD is 170, while in our study it is 68. Second, the mixmax interaction model in Wohlmayr *et al.*'s SD occupies 85% of total running time. In our study the corresponding module is DNN, and it only takes less than 0.4 second for one second mixture.

In addition to the above comparisons, we have compared with [52], where a clustering algorithm is used to group short pitch contours into two speakers. We found that this method performs better than Jin and Wang's method, but worse than Wohlmayr *et al.*'s speaker-dependent method. NMF based approaches have been used in multi-pitch tracking [91, 95]. Since a gain-adapted GMM-FHMM based approach has been shown to match the performance of an NMF-FHMM based approach at various speaker energy ratios [91, 115], we do not directly compare our methods with NMF based algorithms.

3.7 Concluding remarks

We have proposed speaker-dependent and speaker-pair-dependent DNNs to estimate the posterior probabilities of pitch states for two simultaneous speakers. Taking advantage of discriminative modeling and speaker-dependent information, our approach produces good pitch estimation in terms of both accuracy and speaker assignment, and significantly outperforms other state-of-the-art multi-pitch trackers. The SPD-DNN based method performs especially well when the two speakers have close pitch tracks. In order to relax constraints, we have introduced three extensions to SD-DNNs and SPD-DNNs. Gender-pair-dependent DNNs are designed for untrained speakers during testing, and they perform substantially better than other speaker-independent and gender-dependent approaches on both GRID and FDA databases. Given limited speaker-dependent training data, speaker adaptation is effective for reducing tracking errors. Lastly, multi-ratio trained SD-DNNs and SPD-DNNs produce consistent results across various speaker ratios.

To apply our speaker-dependent models requires that the identities of the two speakers be known beforehand. Recently, Zhao *et al.* [125] propose a DNN-based cochannel speaker identification algorithm, which can reliably identify the speakers in two-speaker mixtures. Such an algorithm could be used to first identify the two speakers in an input mixture, thus helping select trained SD-DNNs or SPD-DNNs for pitch estimation. When the speakers in a mixture are not enrolled, we can use a similar cochannel gender pair detection algorithm as a front end for gender-pair-dependent multi-pitch tracking. Our experiments show that the accuracy of such a gender pair detector is perfect.

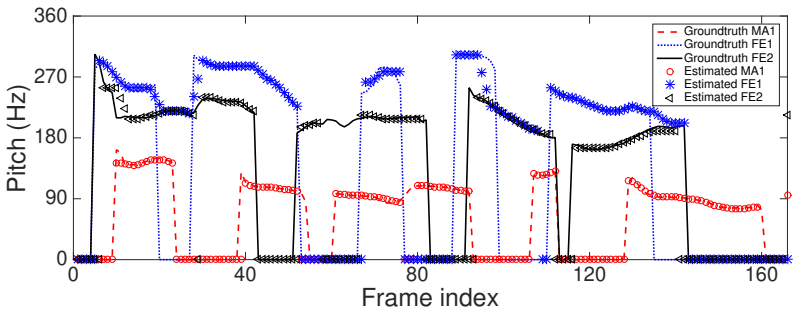


Figure 3.13: Multi-pitch tracking results by SD-DNNs on a three-speaker test sample by mixing MA1, FE1 and FE2 at equal sound levels.

Although the proposed models are designed for two-speaker mixtures, they can be extended to mixtures with more than two speakers. To illustrate this extension, Fig. 3.13 shows an example when three speakers, i. e., MA1, FE1 and FE2 in the GRID database, are mixed in one test sample with equal energy ratio between every pair of speakers. We first use three SD-DNNs trained on the GRID database to estimate pitch-state probabilities for the three speakers. An FHMM with three Markov chains is then employed to connect all probabilities. No retraining is performed for this experiment. As shown in the figure, our algorithm does a decent job tracking three pitch tracks simultaneously. Extensions to more speakers can be achieved in a similar manner. It is worth noting that this relatively straightforward extension is an advantage of our speaker-dependent modeling and our use of FHMM that is not shared by the HMM based model in [60]. Many multi-pitch trackers deal with interfering speakers and additive noise at the same time [60, 119]. We have illustrated the noise-robustness of our models without retraining. Better results are expected if we further include noise corrupted mixtures in the training data set.

To make use of the temporal context, we concatenate neighboring frames into a feature vector. Such a method can only capture temporal dynamics in a limited span. On the other hand, RNNs have self connections through time. Studies have shown that RNNs are good at modeling sequential data like handwriting [33] and speech [104]. We explore RNNs for multi-pitch tracking in the next chapter.

Chapter 4: Permutation Invariant Training for Speaker-independent Multi-pitch Tracking

Speaker-independent multi-pitch tracking has been a long-standing problem in speech processing. In this study, we extend the NN-FHMM framework in the previous chapter, and use the utterance-level permutation invariant training (uPIT) criterion for multi-pitch tracking. Separated speech and label permutations from a speaker separation uPIT-RNN have been further incorporated to improve pitch tracking. We evaluate our methods on the GRID database. Results indicate that the proposed joint speaker separation - pitch tracking system with matched uPIT label permutations outperforms all other gender-dependent and speaker-independent multi-pitch trackers. The improvement is more significant for challenging same-gender mixtures. The work presented in this chapter has been published in [78].

4.1 Introduction

In this chapter, we apply the uPIT method to train a speaker-independent (SI) BLSTM-RNN for multi-pitch tracking. We follow the framework in the previous chapter, and compare an SI uPIT-BLSTM with SPD-BLSTM and GPD-BLSTM for pitch state estimation. Later, we find that by performing multi-pitch tracking alone, uPIT does not lead to good results for same-gender speaker pairs. We extend our

system by using speaker separation (SS) uPIT-BLSTM as a front end. Three new structures are explored. For the first structure, we directly apply the RAPT [101] single-pitch tracking algorithm after uPIT based speaker separation. The second structure combines SS with multi-pitch tracking, by using the outputs of uPIT-SS as additional features for uPIT based multi-pitch tracking. Lastly, we modify the second structure by using the label permutation in uPIT-SS for the multi-pitch tracking network. Consistent improvements are achieved with the third structure.

In the remainder of this chapter, after giving an overview in Section 4.2, the uPIT based systems are introduced in Section 4.3. Section 4.4 describes the FHMM for multi-pitch tracking. Experimental results and comparisons are presented in Section 4.5. A conclusion is given in Section 4.6.

4.2 Speaker-pair and gender-pair dependent pitch probability estimation

4.2.1 Overview

The pitch tracking algorithm in the previous chapter consists of two stages: pitch probability estimation and FHMM. We review the first stage in this section. The input to the system is a speech mixture v_t: $v_t = u_t^1 + u_t^2$, where u_t^1 and u_t^2 are utterances of two speakers. Given the mixture, our system first extracts frame-level log magnitude short-time discrete Fourier transform (STFT) features $\mathbf{y}_m$. We then feed $\mathbf{y}_m$ into neural networks to estimate the posterior pitch probabilities at frame m, i. e., $p(x_m^1, x_m^2|\mathbf{y}_m)$. x_m^1 and x_m^2 denote pitch states of the two speakers at frame m. We quantize the frequency range 60 to 404 Hz into 67 bins using 24 bins per octave on a logarithmic scale. Each bin corresponds to one pitch state. An additional pitch state represents silence or unvoiced speech. $p(x_m^1(s_1), x_m^2(s_2)|\mathbf{y}_m)$ equals one if groundtruth

pitches fall into the s_1^{th} and s_2^{th} frequency bins respectively. Since BLSTM-RNNs [45] make better use of the temporal context, we use BLSTM-RNNs instead of DNNs in this study.

An SPD-BLSTM is a BLSTM-RNN trained on a specific pair of speakers. Apart from the network structure, an SPD-BLSTM is the same as the SPD-DNN in the previous chapter. There are two 68-unit softmax output layers in SPD-BLSTM, with each one estimating the pitch state of the i^{th} speaker $p(x_m^i|\mathbf{y}_m)$. Denoting the i^{th} output layer at frame m by O_m^i, we can write the frame-level cross-entropy loss as:

$$J_m = -\sum_{i=1}^{2}\sum_{s=1}^{68} p(x_m^i(s)|\mathbf{y}_m)\log(O_m^i(s)) \tag{4.1}$$

where s indices 68 pitch states. The final frame-level pitch probability is estimated by:

$$p(x_m^1, x_m^2|\mathbf{y}_m) = O_m^1 O_m^2 \tag{4.2}$$

The SPD-BLSTM multi-pitch model is denoted by SPD-PITCH in this chapter.

SPD-PITCH is not applicable to untrained speakers, we thus introduce GPD BLSTM-RNNs (similar to the GPD-DNNs in the previous chapter) to relax this constraint, denoted by GPD-PITCH.

4.3 uPIT for SI pitch probability estimation

4.3.1 uPIT based SI multi-pitch tracking

The output-speaker pairing in GPD-PITCH provides a reasonable way to differentiate two speakers with little information available. It leads to good results for different-gender mixtures since speech of male and female is intrinsically different in terms of pitch range, timbre, etc. However, for same-gender mixtures, GPD-PITCH's

output-speaker pairing is suboptimal. The order of average pitch of two same-gender speakers usually varies across utterances. Moreover, other important characteristics of speech, including timbre, unvoiced speech, etc., can not be reflected by average pitch.

Utterance-level permutation invariant training (uPIT) [64] has been proposed to replace rule-based label arrangements. In uPIT, the two training labels are provided as a whole set instead of an ordered list, and the output-label pairing $i \leftrightarrow \theta^i$, for a given utterance, is defined as the pairing that minimizes the utterance-level training loss over all possible speaker permutations P. Taking the cross-entropy loss as an example, the optimal permutation is presented as:

$$\theta_* = \underset{\theta \in P}{\operatorname{argmin}} - \sum_m \sum_{i=1}^{2} \sum_{s=1}^{68} p(x_m^{\theta^i}(s)|\mathbf{y}_m) \log(O_m^i(s)) \tag{4.3}$$

θ_* is used for all frames within a training utterance. The frame-level loss can then be calculated as:

$$J_m^{uPIT} = - \sum_{i=1}^{2} \sum_{s=1}^{68} p(x_m^{\theta_*^i}(s)|\mathbf{y}_m) \log(O_m^i(s)) \tag{4.4}$$

In this chapter, we train an SI-BLSTM-RNN with uPIT to predict the pitch states of two speakers, denoted by uPIT-PITCH. As the training unfolds, we expect uPIT-PITCH to learn the correct output permutation for both different-gender and same-gender mixtures.

4.3.2 uPIT based speaker separation followed by single pitch tracking

uPIT is originally proposed for monaural speaker separation (SS). Therefore, an alternative way to apply uPIT for multi-pitch tracking is to first perform uPIT based

SS, and then track the separated signals of the two speakers using conventional single pitch tracking algorithms.

In this study, we follow the uPIT based SS framework in [64], and train a BLSTM-RNN to predict the spectra of two speakers. Magnitude STFT of the mixture is used as the input feature. Two T-F masks are then predicted and multiplied with the mixture STFT, to generate the phase sensitive approximation (PSA) [64] of the two speakers' spectra. The uPIT criterion is used during training. During inference, the estimated outputs are coupled with the noisy phase of the mixture to resynthesize two time-domain signals. In the end, the RAPT [101] algorithm is applied to the resulting signals for pitch tracking. This approach is denoted by uPIT-SS-RAPT.

4.3.3 uPIT based speaker separation followed by uPIT based multi-pitch tracking

uPIT-SS-RAPT gives excellent performance if the speaker separation module works properly. However, for some challenging same-gender mixtures, where uPIT-SS struggles with, directly applying RAPT on top of uPIT-SS is error-prone. There are two main reasons. First, speaker assignment errors in uPIT-SS directly affect the pitch assignment in RAPT. Second, the masked signals contain some artifacts which may degrade the performance of RAPT.

To overcome these issues, we concatenate the two outputs form uPIT based speaker separation with the magnitude STFT of the mixture to form a new input to uPIT-PITCH. A diagram of the network is shown in Fig 4.1. The left module corresponds to uPIT-SS, which is trained first as the base. The right module corresponds to uPIT-PITCH. We do not feed the two outputs of uPIT-SS to separate networks for single-pitch tracking, because that may compound the assignment errors generated

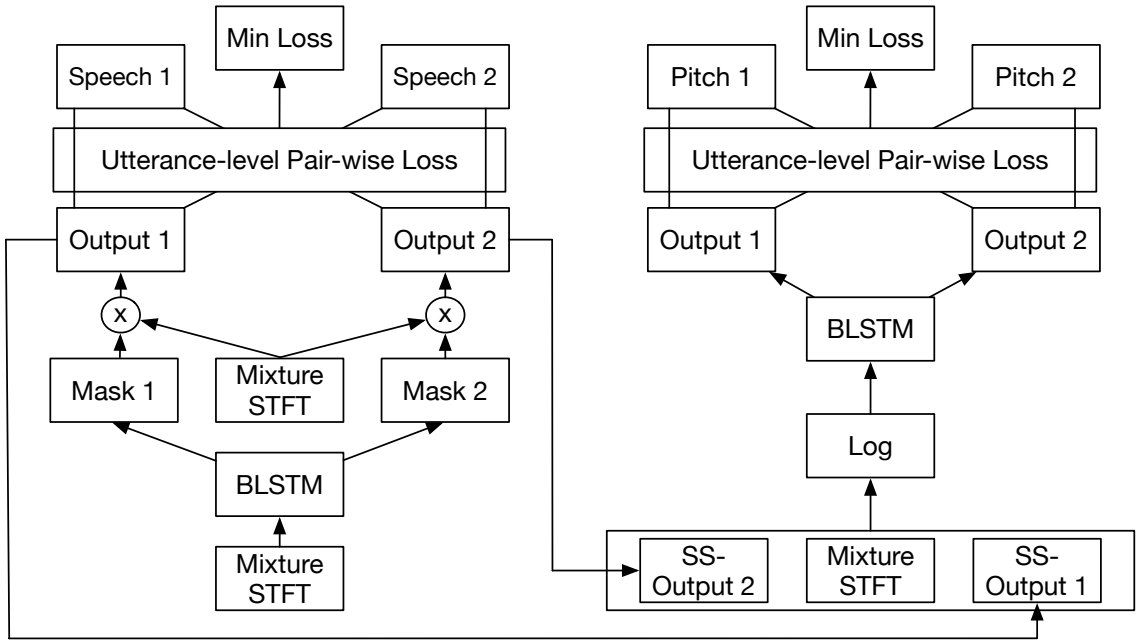


Figure 4.1: Diagram of uPIT-SS-PITCH.

by uPIT-SS. Input features in the right module are pre-processed using logarithmic compression before feeding into the BLSTM-RNN. Lastly, it should be noted that the label permutations of the two modules are optimized independently. We denote the network by uPIT-SS-PITCH in this study.

4.3.4 uPIT based speaker separation followed by multi-pitch tracking with matched label permutation

One observation in the inference of uPIT-PITCH is that for some same-gender speaker pairs, the speaker assignment of pitch estimates swaps very often across time. One possible reason is that the pitch label itself is not informative enough to correctly assign same-gender speakers to different labels. On the other hand, much

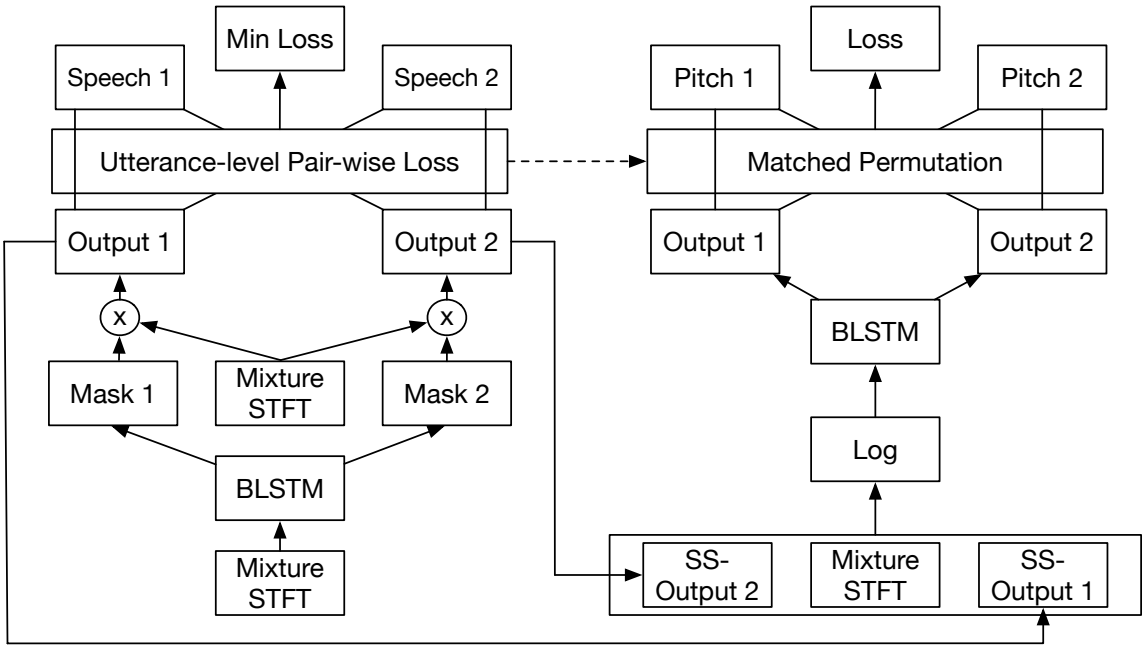


Figure 4.2: Diagram of uPIT-SS-PERM-PITCH.

richer information (unvoiced speech, timbre) is contained in SS's training targets, which may lead to better optimized label permutation during training.

To take advantage uPIT-SS's label permutation, we modify uPIT-SS-PITCH, and use the label permutation in the SS module for that in the pitch module, which is denoted by uPIT-SS-PERM-PITCH. A diagram of the system is shown in Fig 4.2. There is a dashed line connecting the label permutations of the two modules, meaning that the permutations are matched. Better utterance-level label permutation for pitch tracking might be achieved with this structure.

4.4 FHMM inference

After BLSTM based pitch probability estimation, we use a factorial HMM to infer the most likely pitch tracks. The hidden variables are the pitch states of two speakers (x_m^1, x_m^2), and the observation variable is the feature vector $\mathbf{y}_m$. Prior probabilities and transition matrices of the hidden variables are computed from single-speaker training data either speaker-dependently for SPD-PITCH, or speaker-independently for all other models. Laplace smoothing is applied during the computation. We then compute the emission probability of the FHMM from the estimated posterior probability, and apply the junction tree algorithm to infer the most likely sequence of pitch states. In the end, frame-level pitch states are converted back to the centers of frequency bins, and smoothing is applied to get a continuous pitch track.

4.5 Experimental results and comparisons

4.5.1 Experimental setup

We conduct experiments on the GRID database [20], which consists of 1,000 sentences spoken by each of 34 speakers. Two male and two female speakers (No. 1, 2, 18, 20) are selected for testing, denoted by Set One. For each speaker in Set One, we randomly select 10 utterances, and mix them with every other speaker in Set One at -9, -6, -3, 0, 3, 6, and 9 dB. In total, $10\times10\times7$ test mixtures are generated for each of the 6 speaker pairs. We report results with respect to different gender combinations, and absolute energy differences between two test speakers. SPD-PITCH is trained within Set One, where 60,000 training mixtures are generated for each speaker pair by randomly mixing 900 training utterances of both speakers at a random energy ratio between -5 and 5 dB. Set Two is used to train all speaker-independent models,

where another 10 male and 10 female speakers (No. 3, 5, 6, 9, 10, 12, 13, 14, 17, 19; 4, 7, 11, 15, 16, 21, 22, 23, 24, 27) with 900 training utterances each, are selected. For GPD-PITCH, the 60,000-mixture male-female training set is generated by randomly mixing male utterances with female utterances in Set Two at between -5 and 5 dB. We then divide same-gender speakers in Set Two into two groups based on their average pitch. For each same-gender pair, 60,000 mixtures are produced by mixing utterances from different groups at between -5 and 5 dB. Lastly, an SI training set for all uPIT based models is generated by randomly mixing different-speaker utterances within Set Two at between -5 and 5 dB.

Reference pitch is extracted from single-speaker utterances using the RAPT algorithm [101]. All mixtures are sampled at 16 kHz. We extract STFT features using a frame length of 32ms, a frame shift of 10 ms, and the square root of Hanning window.

Results are reported using the error measure E_{Total} proposed in [117], which jointly evaluates the performance in terms of pitch accuracy and speaker assignment. E_{Total} combines the percentile representation of voicing decision errors, permutation errors, gross errors and fine errors. The lower, the better.

4.5.2 Models

All pitch estimation BLSTM-RNNs in this study share the same structure. There are three 500-unit (per direction) BLSTM layers in the model. Two output layers with the softmax activation function are then used to predict pitch states. The networks are trained with the Adam optimization algorithm [62] and dropout regularization [44]. The initial learning rate is set to 0.001, and we decrease the learning rate by a

Table 4.1: E_{Total} (%) of different multi-pitch tracking approaches with respect to different gender combinations, and absolute energy differences.

Methods	Same-Gender				Different-Gender			
	0 dB	3 dB	6 dB	9 dB	0 dB	3 dB	6 dB	9 dB
Wohlmayr et al. SD	31.0	31.5	32.6	34.1	26.0	26.6	27.1	28.3
SPD-PITCH	12.4	12.4	12.8	13.9	11.5	11.6	11.8	12.5
GPD-PITCH	25.7	26.0	27.3	29.6	**14.3**	**14.6**	**15.2**	**16.3**
uPIT-PITCH	**25.1**	**24.9**	**24.4**	**25.2**	14.8	14.8	15.4	16.7

ratio of 0.8 when the validation loss stops decreasing for over 4 epochs. The maximum number of epochs is 30 for SPD-PITCH, and 100 for all other models.

The uPIT based speaker separation network contains 3 BLSTM layers, each with 896 units per direction. Two 257-unit ReLU [30] output layers are used to predict phase sensitive masks. The initial learning rate is set to 0.0002. All other training recipes follow the pitch BLSTM.

We compare all our methods with Wohlmayr et al.'s speaker-dependent GMM-FHMM model with gain adaptation [91, 117], which represents the state-of-the-art for SD multi-pitch tracking. The SD models in [91, 117] are trained within Set One, with the same RAPT based reference pitch. We would like to thank M. Wohlmayr, M. Stark, and F. Pernkopf for providing their pitch tracking code to us.

4.5.3 Results and comparisons

Multi-pitch trackers without speaker separation modules are compared in Table 4.1. All proposed SPD/GPD/SI systems significantly outperform Wohlmayr et al.'s SD models, which reflects the excellent modeling capacity of neural networks. Due to the usage of speaker-dependent information, SPD-PITCH achieves the best results among all systems. GPD-PITCH yields slightly worse E_{Total} than SPD-PITCH

Table 4.2: E_{Total} (%) of joint speaker separation - multi-pitch tracking systems with respect to different gender combinations, and absolute energy differences.

Methods	Same-Gender				Different-Gender			
	0 dB	3 dB	6 dB	9 dB	0 dB	3 dB	6 dB	9 dB
uPIT-SS-RAPT	25.4	25.6	26.0	28.8	12.4	12.7	13.4	15.4
uPIT-SS-PITCH	24.6	24.6	24.2	26.2	14.5	14.6	15.0	16.3
uPIT-SS-PERM-PITCH	**23.8**	**23.4**	**23.5**	**25.3**	14.3	14.5	15.1	16.5

on different-gender mixtures, and far worse E_{Total} on same-gender mixtures. This result is expected since same-gender mixtures are a lot more challenging for speaker-independent approaches, and the heuristic label assignments in GPD-PITCH exacerbate this problem. uPIT-PITCH matches GPD-PITCH's performance on different-gender mixtures, and outperforms GPD-PITCH on same-gender mixtures by a small margin, which shows that the label permutations optimized by uPIT lead to better generalization. However, the improvement is still relative small, thus we further introduce speaker separation to help multi-pitch tracking.

Table 4.2 reports the results of all joint SS-PITCH systems. uPIT-SS-RAPT achieves very good results on different-gender mixtures, primarily due to the fact that the same single pitch tracking algorithm, RAPT, is shared between uPIT-SS-RAPT and the reference pitch. To be more specific, when the separation module works well, uPIT-SS-RAPT tends to generate exactly the same pitch as the reference pitch, which also includes consistent pitch errors, and voicing decision errors in the reference pitch. The errors made by RAPT are regularized by BLSTM-RNNs during training, and thus are regarded as incorrect estimates for BLSTM based pitch trackers. However, for uPIT-SS-RAPT, since the errors are consistent with the reference

pitch, it would be recognized as correct estimation. On the other hand, uPIT-SS-RAPT works relatively poorly on same-gender mixtures, which implies that inaccurate results of speaker separation have an adverse effect on pitch tracking. With the help of the additional input feature, uPIT-SS-PITCH consistently outperforms uPIT-PITCH. There is only one exception, which is for same-gender mixtures at the level difference of 9 dB. The reason is that severe mismatch happens in this condition, so that the additional input may be too noisy to provide useful information. Lastly, with matched label permutation, uPIT-SS-PERM-PITCH generates the best results among speaker-independent models for same-gender mixtures. For different-gender mixtures, uPIT-SS-PERM-PITCH matches the male-female GPD-PITCH, which is specifically trained for male-female mixtures, and has optimally assigned label permutations.

4.6 Conclusion

In this study, we have introduced utterance-level permutation invariant training for multi-pitch tracking. BLSTM-RNNs with two probabilistic pitch outputs are used as the base model. SPD, GPD and uPIT-SI training are compared. For uPIT based pitch estimation, several extensions have been proposed, including incorporating outputs and label permutations in uPIT based speaker separation. Experimental results show that our final model, uPIT-SS-PERM-PITCH, achieves the best results among all GPD and SI models, especially for same-gender speaker pairs. In the future, we will explore multi-target training and joint optimization for speaker separation and multi-pitch tracking.

Chapter 5: A Deep CASA Approach to Talker-independent Monaural Speaker Separation

We address talker-independent monaural speaker separation from the perspectives of deep learning and CASA. Specifically, we decompose the speaker separation task into the stages of simultaneous grouping and sequential grouping. Simultaneous grouping is first performed in each time frame by separating the spectra of different speakers with a permutation-invariantly trained neural network. In the second stage, the frame-level separated spectra are sequentially grouped to different speakers by a clustering network. The proposed deep CASA approach optimizes frame-level separation and speaker tracking in turn, and produces excellent results for both objectives. Experimental results on the benchmark WSJ0-2mix database show that the new approach achieves the state-of-the-art results with a modest model size. The work presented in this chapter has been published in [77] [79] .

5.1 Introduction

Speech usually occurs simultaneously with interference in real acoustic environments. Interference suppression is needed in a wide variety of speech applications, including automatic speech recognition, speaker identification, and hearing aids. One particular kind of interference is the speech signal from competing speakers. Although

human listeners excel at attending to a target speaker even without any spatial cues [11] [36], speech separation remains a challenge for machines despite decades of research. In this chapter, we address monaural speaker separation in the case of two concurrent speakers, which is also known as co-channel speech separation.

DC and PIT represent major approaches to talker-independent speaker separation. There are, however, limitations. As indicated in [64] [77], uPIT sacrifices frame-level performance for the sake of utterance-level assignments. The speaker tracking mechanism in uPIT works poorly for same-gender mixtures. On the other hand, DC is better at speaker tracking, but its frame-level separation is not as good as ratio masking used in tPIT.

Inspired by CASA, PIT and DC, we proposed a deep learning based two-stage method in our preliminary study [77] to perform talker-independent speaker separation. The method consists of two stages, a simultaneous grouping stage and a sequential grouping stage. In the first stage, a tPIT-BLSTM is trained to predict the spectra of the two speakers at each frame without speaker assignment. This stage separates spectral components of the two speakers at the same frame, corresponding to simultaneous grouping in CASA. In the sequential grouping stage, frame-level separated spectra and the mixture spectrogram are fed to another BLSTM to predict embedding vectors for the estimated spectra, such that the embedding vectors corresponding to the same speaker are close together, and those corresponding to different speakers are far apart. A constrained K-means algorithm is then employed to group the two spectral estimates at the same frame across time to different speakers. This stage corresponds to sequential grouping in CASA.

In this chapter, we adopt the same divide-and-conquer strategy but improve its realization in major ways, resulting in what we call a deep CASA approach. In the simultaneous grouping stage, we utilize a UNet [92] convolutional neural network (CNN) with densely-connected layers [53] to improve the performance of frame-level separation. A frequency mapping layer is added to deal with inconsistencies between different frequency bands. To overcome the effects of noisy phase in inverse STFT, we explore complex STFT objectives and time-domain objectives as the training targets. In the sequential grouping stage, we introduce a new embedding representation and weighted objective function. In addition, we leverage the latest development in temporal convolutional networks (TCNs) [6] [66] [81] [90], and use a TCN for sequential grouping, which greatly improves speaker tracking. A new dropout scheme is proposed for TCNs to overcome the overfitting problem. The evaluation results and comparisons demonstrate the resulting system achieves better frame-level separation and speaker tracking at the same time compared to uPIT and [77].

The rest of the chapter is organized as follows. We introduce the proposed algorithm, including the simultaneous and sequential grouping stages, in Section 5.2. Section 5.3 presents experimental results, comparisons and analysis. Conclusion and related issues are discussed in Section 5.4.

5.2 Deep CASA approach to monaural speaker separation

As reported in [64] [77], uPIT considerably improves the separation performance with a default output assignment. But it has the following shortcomings. First, uPIT's output-speaker pairing is fixed throughout a whole utterance, which prevents frame-level loss to be optimized as in tPIT. As a result, uPIT always underperforms

tPIT if their outputs are optimally reassigned. Second, uPIT addresses separation and speaker tracking simultaneously and due to limited modeling capacity of a neural network, uPIT does not work well for speaker tracking, especially for same-gender mixtures.

We employ a divide and conquer idea to break down monaural speaker separation into two stages. In the simultaneous grouping stage, a tPIT based neural network separates spectral components of different speakers at the frame-level. The sequential grouping stage then streams frame-level estimates belonging to the same speaker. Unlike uPIT, separation and tracking are optimized in turn in the deep CASA framework. The two stages are detailed in the following subsections. Notations used in this chapter follow those in Section 1.3.

5.2.1 Simultaneous grouping stage

Baseline system

We adopt the tPIT framework described in [77] as the baseline simultaneous grouping system. The magnitude STFT of the mixture is used as the input. BLSTM is employed as the learning machine. The system is trained using the loss function in Eq. 1.7. In the end, frame-level spectral estimates are passed to the second stage for sequential grouping.

Alternative training targets for tPIT

As mentioned, the PSA training target partially accounts for STFT phase, unlike the ideal binary mask (IBM) and ideal ratio mask (IRM).

$$\mathrm{IBM}_i(t,f) = \begin{cases} 1, & \text{if } |X_i(t,f)| > |X_{j\neq i}(t,f)| \\ 0, & \text{otherwise} \end{cases} \tag{5.1}$$

$$\mathrm{IRM}_i(t,f) = \frac{|X_i(t,f)|}{\sum_{j=1}^{2}|X_j(t,f)|} \tag{5.2}$$

However, PSA cannot completely restore the phase information in clean sources, because it uses noisy phase during iSTFT. Recently, complex ratio masking [114] (cRM) attempts to restore clean phase. The complex ideal ratio mask (cIRM) is defined in the complex STFT domain, with real and imaginary parts. When applied to the complex STFT of the mixture, it perfectly reconstructs clean sources:

$$X_c(t,f) = \mathrm{cIRM}_c(t,f) \otimes Y(t,f) \tag{5.3}$$

where $\otimes$ denotes point-wise complex multiplication.

We propose complex ratio masking to perform monaural speaker separation. Instead of directly using the cIRM as the training target, we first multiply the complex mixture by the estimated complex mask cRM_c to perform complex domain reconstruction:

$$\hat{X}_c(t,f) = \mathrm{cRM}_c(t,f) \otimes Y(t,f) \tag{5.4}$$

The reconstructed sources are then compared with clean sources to form the training objective:

$$J_t^{tPIT-CA} = \min_{\theta(t)\in P} \sum_{f,c} [\ |\mathrm{Re}(\hat{X}_c - X_{\theta_c(t)})| + |\mathrm{Im}(\hat{X}_c - X_{\theta_c(t)})|\ ] \tag{5.5}$$

where the l_1 norm is applied to both the real and imaginary parts of the loss. We call this training objective complex approximation (CA).

We also consider a training objective based on time-domain signal-to-noise ratio (SNR). The proposed framework consists of two steps: First, we organize all frame-level complex estimates $\hat{X}_c$ with respect to the minimum frame-level loss, so that each organized output $\hat{X}_{\Theta_c(t)}$ corresponds to a single speaker. The frame-level loss

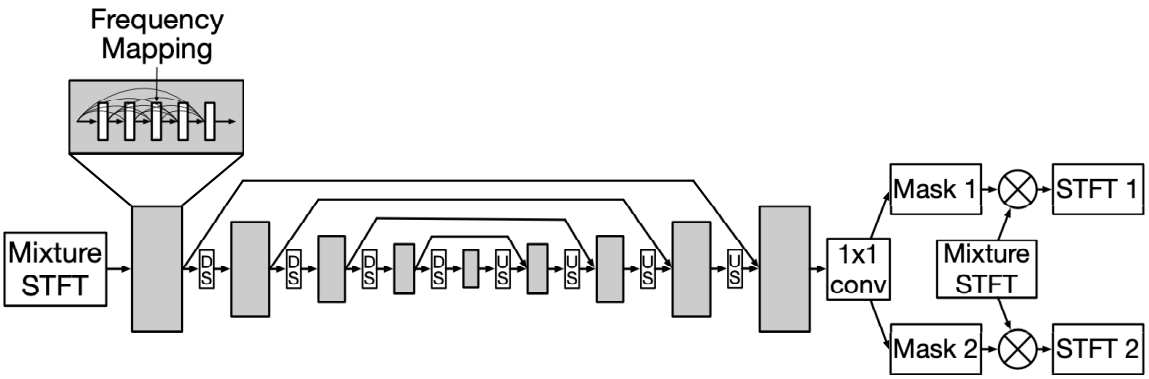


Figure 5.1: Diagram of the Dense-UNet used in simultaneous grouping. Gray blocks denote dense CNN layers. DS blocks denote downsampling layers and US blocks denote upsampling layers. Skip connections are added to connect layers at the same level. The inputs, masks and outputs can be defined in either magnitude or complex STFT domain.

for organization can be defined in three domains: the complex STFT, magnitude STFT and time domain. In each domain, we compare the estimates and ground-truth targets, and calculate the l_1 norm of the difference as the loss. We find that the complex STFT loss leads to slightly better separation performance. Second, we apply iSTFT (Eq. 1.5) to $\hat{X}_{\Theta_{c(t)}}(t,f)$, and compute utterance-level SNR for the final time-domain estimates $\hat{x}_{\Theta_{c(t)}}(n)$:

$$J^{tPIT-SNR} = \sum_{c=1}^{2} 10 \log \frac{\sum_n x_c(n)^2}{\sum_n (x_c(n) - \hat{x}_{\Theta_{c(t)}}(n))^2} \tag{5.6}$$

Convolutional neural networks for simultaneous grouping

Partly motivated by the recent success of DenseNet [53] and UNet [92] in music source separation [58] [99], we propose a Dense-UNet structure for simultaneous grouping. UNet is a natural choice for spectral-domain source separation. With the

"hourglass" architecture and skip connections, UNet models global patterns and preserves fine-grained details in the spectrogram. The use of DenseNet is validated by our preliminary experiments where the proposed Dense-UNet significantly outperforms a standard UNet for speaker separation.

The proposed Dense-UNet is shown in Fig. 5.1, and it is based on a UNet architecture [92]. It consists of a series of convolutional layers, downsampling layers and upsampling layers. The first half of the network encodes utterance-level STFT feature maps into a higher level of abstraction. Convolutional layers and downsampling layers are alternated in this half, allowing the network to model large T-F contexts. Convolutional layers and upsampling layers are alternated in the second half to project the encoded features back to its original resolution. In this study, we use strided 2×2 depthwise convolutional layers [17] as downsampling layers. Strided transpose convolutional layers are used as upsampling layers. Skip connections are added between the layers at the same hierarchical level in the encoder and decoder, and they are important for UNet. As the model goes deeper, the feature maps are projected to more and more abstract representations of the mixture at different resolutions. If skip connections are removed, the network can still produce coarse masks, lacking fine-grain details. Such a phenomenon is discussed in [58].

Next, we replace convolutional layers in the original UNet with densely-connected CNN blocks (DenseNet) [53]. The basic idea of DenseNet is to decompose one convolutional layer with many channels into a sequence of densely connected convolutional layers with fewer channels, where each layer is connected to every other layer in a feedforward fashion:

$$z_l = H_l([z_{l-1}, z_{l-2}, ..., z_0]) \tag{5.7}$$

where z_0 denotes the input feature map, z_l the output of the l^{th} layer, [...] concatenation, and H_l the l^{th} convolutional layer followed by ELU (exponential linear unit) activation [19] and layer normalization [3]. The DenseNet structure has shown excellent performance in image classification [53] and music source separation [99]. In this study, all output layers z_l in a dense block have the same number of channels, denoted by K. The total number of layers in each dense block is denoted by L. As shown in Fig. 5.1, we alternate 9 dense blocks with 4 downsampling layers and 4 upsampling layers. After the last dense block, we use a 1×1 CNN layer to reorganize the feature map, and then output two masks.

In CNNs, convolutional kernels are usually applied across the entire input field. This is reasonable in the case of visual processing, where similar patterns can appear anywhere in the visual field with translation and rotation. However, in the auditory representation of speech, patterns that occur in different frequency bands are usually different. A generic CNN kernel may result in inconsistent outputs at different frequencies. To address this problem, Takahashi and Mitsufuji [99] split the spectral input into several subbands, and train band-dependent CNNs, leading to a substantial rise in model size.

We propose a frequency mapping layer which effectively alleviates this problem with a significant reduction of parameters. The basic idea is to project inconsistent frequency outputs to an organized space using a fully-connected layer. We replace one CNN layer in each dense block with a frequency mapping layer. The input to a frequency mapping layer is a concatenation of CNN layers $z_l^0 = [z_{l-1}, z_{l-2}, ..., z_0] \in \mathbb{R}^{T \times F \times K'}$, where T and F denote time and frequency respectively, K' the number of channels in the input. z_l^0 is passed to a 1×1 convolutional layer, followed by

ELU activation and layer normalization, to reduce the number of channels to K. The resulting output is denoted by $z_l^1 \in \mathbb{R}^{T\times F\times K}$. We then transpose the F and K dimension of z_l^1 to get $z_l^2 \in \mathbb{R}^{T\times K\times F}$. Next, z_l^2 is fed to a 1×1 convolutional layer, followed by ELU activation and layer normalization, to output $z_l^3 \in \mathbb{R}^{T\times K\times F}$. This layer can also be viewed as a frequency-wise fully connected layer, which takes all frequency estimates as the input and reorganize them in a different space. Finally, z_l^3 is transposed back, and the output of the frequency mapping layer $z_l \in \mathbb{R}^{T\times F\times K}$ is generated.

5.2.2 Sequential grouping stage

Baseline system

In this stage, we group frame-level spectral estimates across time using a clustering network, which corresponds to sequential grouping in CASA. In deep clustering based speaker separation, T-F level embedding vectors estimated by BLSTM are clustered into different speakers. We extend this framework to frame-level speaker tracking.

Fig. 5.2 illustrates our sequential grouping. We first stack the mixture spectrogram and two spectral estimates (including real, imaginary and magnitude STFT) as the input to the system. A neural network then projects frame-level inputs to a D-dimensional embedding vector $\mathbf{V}(t) \in \mathbb{R}^{1\times D}$. The target label is a two-dimensional indicator vector, denoted by $\mathbf{A}(t)$. During the training of tPIT, if the minimum loss is achieved when $\hat{\mathbf{X}}_1(t)$ is paired with speaker 1, and $\hat{\mathbf{X}}_2(t)$ is paired with speaker 2, we set $\mathbf{A}(t)$ to [1 0]. Otherwise, $\mathbf{A}(t)$ is set to [0 1]. In other words, $\mathbf{A}(t)$ indicates the optimal output assignment of each frame. $\mathbf{V}(t)$ and $\mathbf{A}(t)$ can be reshaped into a

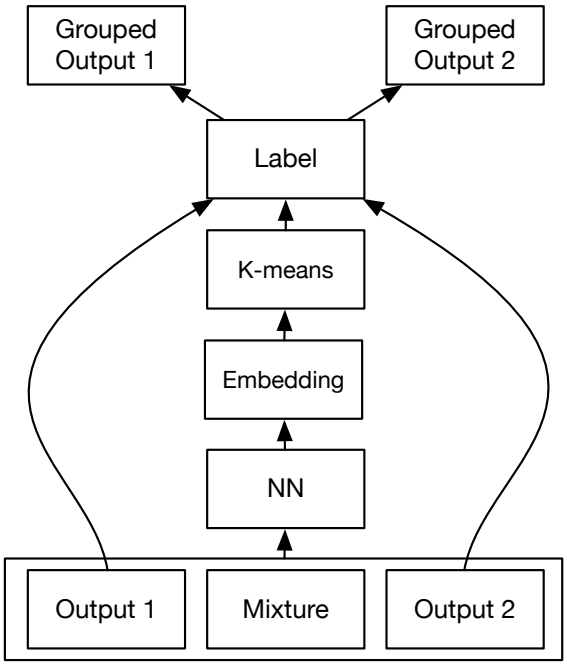


Figure 5.2: Diagram of the sequential grouping stage. We use BLSTM or TCN as the neural network in this stage.

$T \times D$ matrix $\mathbf{V}$, and a $T \times 2$ matrix $\mathbf{A}$, respectively. A permutation independent objective function [40] is:

$$J^{DC} = ||\mathbf{V}\mathbf{V}^T - \mathbf{A}\mathbf{A}^T||_F^2 \tag{5.8}$$

where $||\cdot||_F$ is the Frobenius norm. Optimizing J^{DC} forces $\mathbf{V}(t)$ corresponding to the same optimal assignment to get closer during training, and otherwise to become farther apart.

Because we care more about the speaker assignment of frames where the two outputs are substantially different, a weight $w(t) = \frac{|LD(t)|}{\sum_t |LD(t)|}$ is used during training where $LD(t)$ represents the frame-level loss difference (LD) between the two possible speaker assignments. $LD(t)$ is large if two conditions are both satisfied: 1) the

frame-level energy of the mixture is high; 2) the two frame-level outputs, $\hat{X}_1(t,f)$ and $\hat{X}_2(t,f)$, are quite different, so that the losses with respect to different speaker assignments are significantly different. $w(t)$ can be used to construct a diagonal matrix $\mathbf{W} = diag(w(t))$. The final weighted objective function is:

$$J^{DC-W} = ||\mathbf{W}^{1/2}(\mathbf{V}\mathbf{V}^T - \mathbf{A}\mathbf{A}^T)\mathbf{W}^{1/2}||_F^2 \tag{5.9}$$

This objective function emphasizes frames where the speaker assignment plays an important role.

During inference, the K-means algorithm is first applied to cluster $\mathbf{V}(t)$ into two groups. We then organize frame-level outputs according to their K-means labels. Finally, iSTFT is employed to convert complex outputs to the time domain.

Temporal convolutional networks for sequential grouping

Temporal convolutional networks (TCNs) have been used as a replacement for RNNs, and have shown comparable or better performance in various tasks [6] [66] [81] [90]. In TCNs, a series of dilated convolutional layers are stacked to form a deep network, which enables very long memory. In this study, we adopt a TCN similar to Conv-TasNet [81] for sequential grouping, as illustrated in Fig. 5.3.

In the proposed TCN, input features are first passed to a 2-D dense CNN block, a 1×1 convolutional layer and a layer normalization module, to perform frame-level feature preprocessing. The 1×1 convolutional layer here refers to a 1-D CNN layer with a kernel size of 1. The preprocessed features are then passed to a series of dilated convolutional blocks, with an exponentially increasing dilation factor (2^0, 2^1, ..., 2^{M-1}) to exploit large temporal contexts. Next, the M stacked dilated convolutional blocks

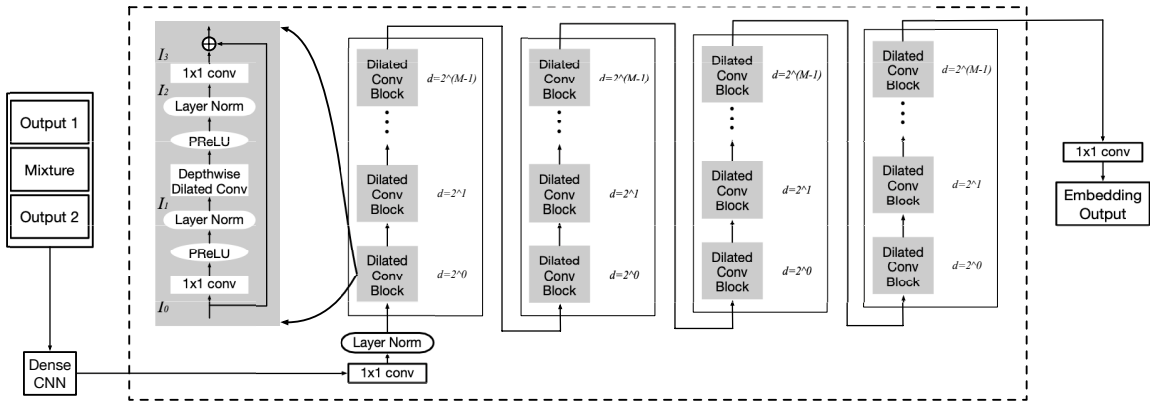


Figure 5.3: Diagram of the TCN used in sequential grouping. Outputs from the previous stage are fed into a series of dilated convolutional blocks to predict frame-level embedding vectors. The dilation factor of each block is marked on the right. The detailed structure of a dilated convolutional block is illustrated in the large gray box. The network within the dashed box can be also used for uPIT based speaker separation.

are repeated 3 times to further increase the receptive field. Lastly, the outputs are fed into a 1×1 convolutional layer for embedding estimation.

In each dilated convolutional block, a bottleneck input with B channels $I_0 \in \mathbb{R}^{T \times B}$ is first passed to a 1×1 convolutional layer, followed by PReLU (parametric rectified linear unit) activation [37] and layer normalization, to extend the number of channels to H, with output denoted by $I_1 \in \mathbb{R}^{T \times H}$. A depthwise dilated convolutional layer [17] with kernel $S \in \mathbb{R}^{3 \times H}$, followed by PReLU activation and layer normalization, is then employed to capture the temporal context. The number 3 here indicates the size of the temporal filter in each channel, and there are H depthwise separable filters in the kernel. We adopt non-causal filters to exploit both past and future information, with a dilation factor from 2^0,... 2^{M-1}, as in [81]. The output of this part is denoted by $I_2 \in \mathbb{R}^{T \times H}$, which is then passed to a 1×1 convolutional layer

to project the number of channels back to B, denoted by $I_3 \in \mathbb{R}^{T \times B}$. In the end, an identity residual connection combines I_3 and I_0 and forms the final output.

Overfitting is a major concern in sequence models. If not regularized properly, sequence models tend to memorize the patterns in the training data, and get trapped in local minima. To address this issue, various dropout techniques [28] [83] [94] have been proposed for RNNs. Consistent improvement has been achieved if dropout is applied to recurrent connections [83]. Meanwhile, a simple dropout scheme for TCNs is used in [6], i.e., dropping I_3 in each dilated convolutional block, but it does not yield satisfactory performance in our experience. Based on these findings, we design a new dropout scheme for the TCN model, denoted by dropDilation. In dropDilation, the dilated connections in depthwise dilated convolutional layers are dropped with a probability of $(1 - p)$, where p denotes the keep rate. To be more specific, a binary mask, $\mathbf{m} = [m_{-d} \quad 1 \quad m_d]^T \in \mathbb{R}^{3 \times 1}$, is multiplied with each depthwise dilated convolutional kernel $S \in \mathbb{R}^{3 \times H}$ during training, with m_{-d} and m_d drawn independently from a Bernoulli distribution $Bernoulli(p)$. In dropDilation, we only drop the dilated connections while keeping the direct connections to preserve local information.

5.3 Evaluation and comparison

5.3.1 Experimental setup

We use the WSJ0-2mix dataset, a monaural two-talker speaker separation dataset introduced in [40], for evaluations. WSJ0-2mix has a 30-hour training set and a 10-hour validation set generated by selecting random speaker pairs in the Wall Street Journal (WSJ0) training set si_tr_s, and mixing them at various SNRs between 0 dB and 5 dB. Evaluation is conducted on the 5-hour open-condition (OC) test set, which

is similarly generated using 16 untrained speakers from the WSJ0 development set si_dt_05 and si_et_05. All mixtures are sampled at 8 kHz. STFT with a frame length of 32ms, a frame shift of 8 ms, and a square root Hanning window is taken for the whole system.

We report results in terms of signal-to-distortion ratio improvement (ΔSDR) [103], perceptual evaluation of speech quality (PESQ) [57], and extended short-time objective intelligibility (ESTOI) [59], to measure source separation performance, speech quality and speech intelligibility, respectively. We also report the final result in terms of scale-invariant signal-to-noise ratio improvement (ΔSI-SNR) [81] for a systematical comparison with other competitive systems.

5.3.2 Models

Simultaneous grouping models

Two models are evaluated for simultaneous grouping: BLSTM and Dense-UNet.

The baseline BLSTM contains 3 BLSTM layers, with 896×2 units in each layer. In each dense block of Dense-UNet, the number of channels K is set to 64, the total number of dense layers L is set to 5, and all CNN layers have a kernel size of 3×3 and a stride of 1×1. The middle layer in each dense block is replaced with a frequency mapping layer. We use valid padding (a term in CNN literature referring to no input padding) for the last CNN layer in each dense block, and same padding (padding the input with zeros so that the output has the same dimension as the original input) for all other layers. The input STFT is zero-padded accordingly.

For both models, when trained with $J_t^{tPIT-PSA}$, the magnitude STFT of the mixture is adopted as the input, and ELU activation is applied to output layers for

phase-sensitive mask estimation. If $J_t^{tPIT-CA}$ or $J^{tPIT-SNR}$ is used for training, a stack of real and imaginary STFT is used as the input, and linear output layers are used to predict the real and imaginary parts of complex ratio masks separately.

Both networks are trained with the Adam optimization algorithm [62] and dropout regularization [44]. The initial learning rate is set to 0.0002 for BLSTM, and 0.0001 for Dense-UNet. Learning rate adjustment and early stopping are employed based on the loss on the validation set.

Sequential grouping models

Two models are evaluated for sequential grouping: BLSTM and TCN. Both models are trained on top of a well-tuned simultaneous grouping model.

The baseline BLSTM contains 4 BLSTM layers, with 300×2 units in each layer. In TCN, the maximum dilation factor is set to $2^6 = 64$, to reach a theoretical receptive field of 8.128s. The number of bottleneck units B is selected as 256. The number of units in depthwise dilated convolutional layers H is set to 512. Same padding is employed in all CNN layers. DropDilation with $p = 0.7$ is applied during training.

A 2-D dense CNN block is used in both models for frame-level feature preprocessing, with $K = 16$, $L = 4$, a kernel size of 1×3 ($T \times F$) and a stride of 1×1. The dimensionality of embedding vectors D is set to 40. Both networks are trained with the Adam optimization algorithm, with an initial learning rate of 0.001 for BLSTM, and 0.00025 for TCN. Learning rate adjustment and early stopping are again adopted.

Table 5.1: Average ΔSDR, PESQ and ESTOI for simultaneous grouping models with optimal output assignment on WSJ0-2mix OC.

	Objective	# of param.	ΔSDR (dB)	PESQ	ESTOI (%)
Mixture	-	-	0.0	2.02	56.1
tPIT BLSTM	PSA	46.3M	13.0	3.13	86.7
tPIT Dense-UNet	PSA	4.7M	14.7	3.41	90.5
tPIT Dense-UNet	CA	4.7M	18.6	3.57	93.8
tPIT Dense-UNet	SNR	4.7M	19.1	3.63	94.3

One stage uPIT models

To systematically evaluate the proposed methods, we train a Dense-UNet and a TCN with SNR objectives and uPIT training criterion, i.e., $J^{uPIT-SNR}$. Other training recipes follow those in the previous subsections.

5.3.3 Results and comparisons

We first evaluate the simultaneous grouping stage. Table 5.1 summarizes the performance of tPIT models with respect to different network structures and training objectives. For all models, outputs are organized with the optimal speaker assignment before evaluation. Scores of mixtures are presented in the first row. Compared to BLSTM, Dense-UNet drastically reduces the number of trainable parameters to 4.7 million, and introduces significant performance gain. The frequency mapping layers in our Dense-UNet introduce a 0.3 dB increment in ΔSDR, 0.1 increment in PESQ, 0.8% increment in ESTOI and a parameter reduction of 0.9 million. Next, we switch from magnitude STFT to complex STFT, and change the training objective to $J_t^{tPIT-CA}$. This change leads to large improvement, revealing the importance of phase

Table 5.2: Average ΔSDR, PESQ and ESTOI for tPIT and uPIT based Dense-UNet trained with SNR objectives.

	Output Assign.	ΔSDR (dB)	PESQ	ESTOI (%)
tPIT Dense-UNet	Optimal	19.1	3.63	94.3
	Default	0.0	1.99	55.8
uPIT Dense-UNet	Optimal	17.0	3.40	91.6
	Default	15.2	3.24	88.9

information for source separation. The SNR objective further outperforms the CA objective. We thus adopt tPIT Dense-UNet trained with $J^{tPIT-SNR}$ for simultaneous grouping in the following evaluations.

Table 5.2 compares tPIT and uPIT based Dense-UNet in terms of both optimal and default output assignments. Both models are trained with SNR objectives. Thanks to the utterance-level output-speaker pairing, uPIT's default assignment is improved by a large margin over tPIT. However, since frame-level loss is not optimized in uPIT, there is a significant gap between uPIT and tPIT with optimal assignment.

Fig. 5.4 illustrates the differences between tPIT and uPIT based Dense-UNet in more details. Because SNR objectives lead to less structured outputs in the T-F domain, the models illustrated in the figure are trained with CA objectives. Speaker assignment swaps frequently in the default outputs of tPIT. However, if we organize the outputs with the optimal assignment, the outputs almost perfectly match the clean sources, as shown in the fourth row. On the other hand, the default outputs of uPIT are much closer to the clean sources compared to tPIT. However, for this same-gender mixture, uPIT makes several assignment mistakes in the default outputs, e.g., from 2s to 2.5s, and from 5s to 5.2s. If we optimally organize uPIT's outputs, as in the

last row, we can see uPIT exhibits much worse frame-level performance than tPIT. In some frames, e.g., around 4.9s, the predicted frequency patterns are totally mixed up. These observations reveal uPIT's limitations in both frame-level separation and speaker tracking for challenging speaker pairs.

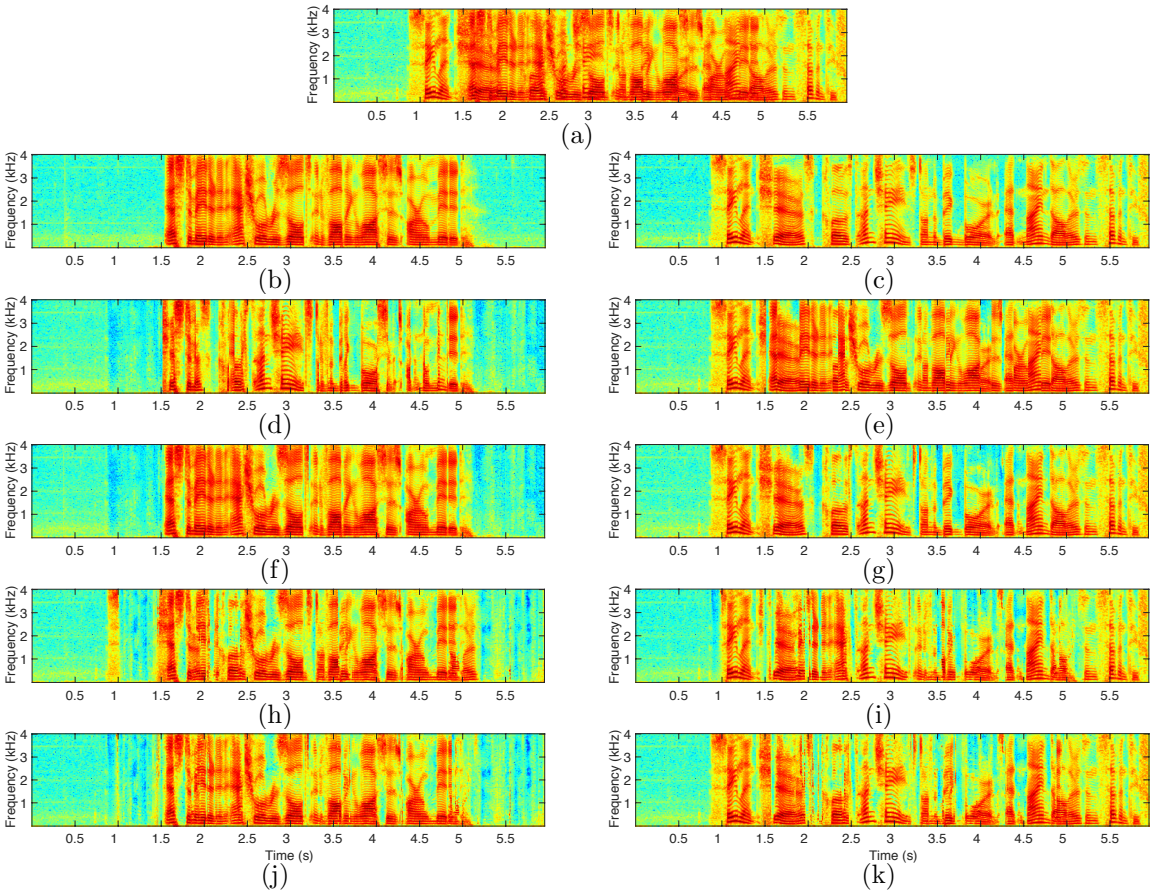


Figure 5.4: Speaker separation results of PIT based models in log-scale magnitude STFT. Two models, tPIT Dense-UNet and uPIT Dense-UNet, are trained with CA objectives. The complex outputs from the models are converted to log magnitude STFT for visualization. (a) A male-male test mixture. (b) Speaker 1 in the mixture. (c) Speaker 2 in the mixture. (d) tPIT's output 1 with default assignment. (e) tPIT's output 2 with default assignment. (f) tPIT's output 1 with optimal assignment. (g) tPIT's output 2 with optimal assignment. (h) uPIT's output 1 with default assignment. (i) uPIT's output 2 with default assignment. (j) uPIT's output 1 with optimal assignment. (k) uPIT's output 2 with optimal assignment.

Table 5.3: Comparison of different sequential grouping methods on WSJ0-2mix OC.

Simul. Group.	Seq. Group.	ΔSDR (dB)	PESQ	ESTOI (%)
tPIT Dense-UNet	BLSTM	16.4	3.31	90.8
tPIT Dense-UNet	TCN	17.9	3.49	92.9
uPIT Dense-UNet	-	15.2	3.25	89.0
uPIT Dense-UNet	Optimal	17.0	3.40	91.6
uPIT TCN	-	13.5	3.06	85.9
uPIT TCN	Optimal	14.9	3.19	88.1

Next, we evaluate different sequential grouping models in Table 5.3. The first two models are trained on top of the tPIT Dense-UNet with the SNR objective. As shown in the table, TCN substantially outperforms BLSTM, both having around 8 million parameters. In BLSTM only neighboring frames are recurrently connected. On the other hand, in TCN, each frame is linked to neighboring and distant frames, facilitating utterance-level speaker tracking. The dropDilation technique in our TCN introduces 0.5 dB ΔSDR gain compared to conventional dropout [6].

In the last four rows of Table 5.3, we report the results of uPIT models. The first uPIT model is trained using Dense-UNet, and it significantly underperforms both deep CASA systems. Even if the outputs are optimally reassigned, uPIT Dense-UNet still systematically underperforms deep CASA (tPIT Dense-UNet + TCN), due to its frame-level separation errors. We also train a TCN model with uPIT objectives, and it yields much worse results than uPIT Dense-UNet.

To further analyze the differences between deep CASA and uPIT, we present frame assignment error (FAE) for the best performing deep CASA system and the

Table 5.4: Frame assignment errors for different methods for frames with significant energy (at least -20 dB relative to maximum frame-level energy).

Simul. Group.	Seq. Group.	Frame Assign. Errors (%)
tPIT Dense-UNet	TCN	1.38
uPIT Dense-UNet	-	3.43
uPIT TCN	-	3.07

two uPIT based models in Table 5.4. FAE is defined as the percentage of incorrectly assigned frames in terms of the minimum frame-level loss. As shown in the table, uPIT Dense-UNet generates the highest FAE, because the network is not specifically designed for sequence modeling. uPIT TCN slightly outperforms uPIT Dense-UNet due to its long receptive field. However, because uPIT TCN does not handle frequency patterns as well, its overall separation performance is worse than uPIT Dense-UNet. Deep CASA cuts FAE by half compared to uPIT models. Such results demonstrate the benefits of the proposed divide-and-conquer strategy, which optimizes frame-level separation and speaker tracking in turn, and achieves better performance in both objectives.

Fig. 5.5 displays the scatter-plots of ΔSDR for deep CASA, uPIT Dense-UNet and uPIT TCN, where color indicates density. Generally speaking, ΔSDR is higher when mixture SDR is lower. Compared to the two uPIT based models, deep CASA not only improves the average results, but also reduces outlier cases, i.e., test samples with ΔSDR far from the dense central region. Such an observation is also reflected by standard deviations, which are 4.2 dB ΔSDR, 0.35 PESQ, and 5.9% ESTOI for deep CASA, 5.5 dB ΔSDR, 0.49 PESQ, and 9.8% ESTOI for uPIT Dense-UNet, and 4.7 dB ΔSDR, 0.45 PESQ, and 9.3% ESTOI for uPIT TCN.

Table 5.5: Average ΔSDR, PESQ and ESTOI for deep CASA and uPIT with respect to different gender combinations.

Model	Gender Comb.	ΔSDR (dB)	PESQ	ESTOI (%)
tPIT Dense-UNet + TCN Assign.	Female-Male	18.9	3.57	93.9
	Female-Female	15.7	3.32	90.5
	Male-Male	17.2	3.45	92.5
uPIT Dense-UNet	Female-Male	17.4	3.43	92.0
	Female-Female	11.0	2.90	83.6
	Male-Male	13.7	3.12	86.8
tPIT Dense-UNet + Opt. Assign.	Female-Male	19.4	3.64	94.4
	Female-Female	18.8	3.61	93.9
	Male-Male	18.7	3.62	94.3

Table 5.5 compares deep CASA and uPIT systems with respect to different gender combinations. Both systems achieve better results on male-female combinations than same gender conditions. The performance gap is larger for female-female mixtures, consistent with the observation in [70]. This might be due to the unbalanced gender distribution in WSJ0-2mix OC, which contains 1086 male-male mixtures, but only 394 female-female mixtures. On the other hand, the performance gap between different gender combinations is much smaller in deep CASA than in uPIT, likely because deep CASA is better at speaker tracking.

Fig. 5.6 illustrates the results of deep CASA. As shown in the second row, tPIT Dense-UNet trained with SNR objectives generates entirely different default outputs compared to the same model trained with CA (cf. Fig. 5.4). The optimal assignments alternate almost every frame, leading to striped patterns. To study the phenomenon, we analyze the overall training process of tPIT Dense-UNet trained with $J^{tPIT-SNR}$. At the beginning, the SNR objective leads to similar outputs as the CA objective.

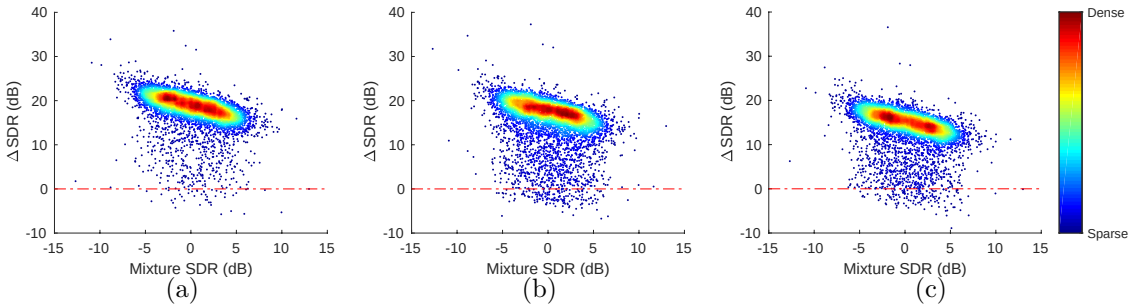


Figure 5.5: Scatter-plots of ΔSDR for different methods on WSJ0-2mix OC. (a) Deep CASA (tPIT Dense-UNet + TCN Assign. without joint optimization). (b) uPIT Dense-UNet. (c) uPIT TCN.

However, because there is 75% overlap between neighboring frames in the proposed STFT, models trained with SNR only need to make accurate predictions every other frame, with frames in between left blank. Such patterns start to occur after a few hundred training steps. The competing speaker then gradually fills in the blanks, and the striped patterns are thus formed. As shown in Fig. 5.6(f), the K-means labels predicted by the sequential grouping system almost perfectly match the optimal labels in speech-dominant frames. However, organizing the default outputs with respect to the K-means labels leads to magnitude STFT that is quite different from the clean sources. Residual patterns from the interfering speaker still exist in some frames. If we convert the complex outputs in Fig. 5.6(g) and (h) to the time-domain, these residual patterns will be cancelled by the overlap-and-add operation in iSTFT due to their opposite phases. In the last row, we apply iSTFT and STFT in turn to the organized complex outputs, and the new results can almost perfectly match the clean sources.

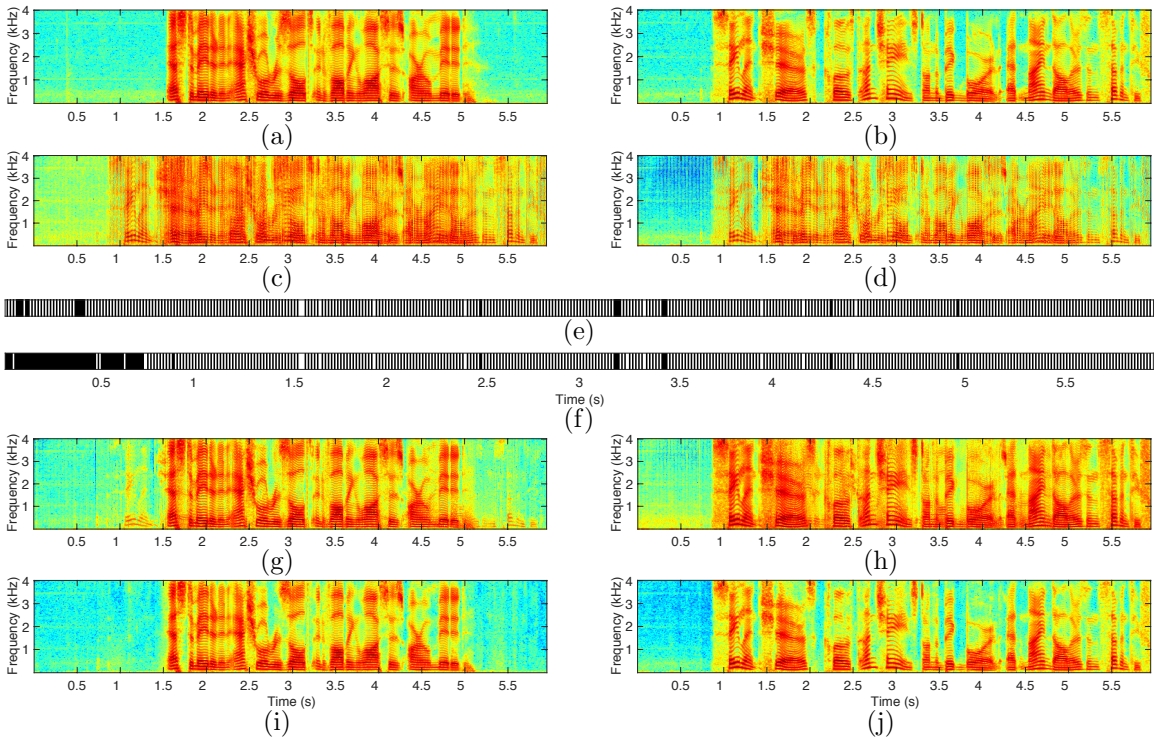


Figure 5.6: Speaker separation results of the deep CASA system, with tPIT Dense-UNet trained with SNR objectives for simultaneous grouping and TCN for sequential grouping. The same test mixture is used as in Fig. 5.4. The complex outputs from the models are converted to log magnitude STFT for visualization. (a). Speaker 1 in the mixture. (b) Speaker 2 in the mixture. (c) tPIT's output 1 with default assignment. (d) tPIT's output 2 with default assignment. (e) Optimal assignment (black and white bars represent two different assignments). (f) K-means assignment. (g) tPIT's output 1 with K-means assignment. (h) tPIT's output 2 with K-means assignment. (i) tPIT's output 1 with K-means assignment after iSTFT and STFT. (j) tPIT's output 2 with K-means assignment after iSTFT and STFT.

Simultaneous and sequential grouping are optimized in turn in the above deep CASA systems. We now consider joint optimization, where the two stages are trained together with small learning rates (1/8 of the initial learning rates) for 40 epochs. For the simultaneous grouping module, we organize the outputs using estimated K-means labels, and compare them with the clean sources to form an SNR objective.

Table 5.6: Number of parameters, average ΔSDR, ΔSI-SNR, PESQ and ESTOI for various state-of-the-art systems evaluated on WSJ0-2mix OC.

	# of param.	ΔSDR (dB)	ΔSI-SNR (dB)	PESQ	ESTOI (%)
Mixture	-	0.0	0.0	2.02	56.1
uPIT [64]	92.7M	10.0	-	2.84	-
Conv-TasNet [81]	5.1M	15.6	15.3	3.24	-
Wang et al. [113]	56.6M	15.4	15.2	3.45	-
FurcaNeXt [96]	51.4M	**18.4**	-	-	-
Deep CASA	12.8M	18.0	**17.7**	**3.51**	**93.2**
IBM	-	13.8	13.4	3.28	89.1
IRM	-	13.0	12.7	3.68	92.9
PSM	-	16.7	16.4	3.98	96.0

Meanwhile, the sequential grouping module is trained using the weighted objective in Eq. 5.9. As joint training unfolds, we observe smoother outputs. Joint optimization introduces slight but consistent improvement in all three metrics (on average by 0.1 dB ΔSDR, 0.02 PESQ, and 0.3% ESTOI, and a reduction of standard deviation by 0.2 dB ΔSDR, 0.02 PESQ, and 0.4% ESTOI).

Finally, Table 5.6 compares the deep CASA system with joint optimization and other state-of-the-art talker-independent methods on WSJ0-2mix OC. For all methods, we list the best reported results, and leave unreported fields blank. The numbers of parameters in different methods are estimated according to their papers. The uPIT system [64] is the basis of this study. Conv-TasNet [81] extends uPIT to the waveform domain, where a TCN is utilized for separation. We have also trained a similar uPIT TCN in this work. However, due to the different domains of signal representation, our uPIT TCN yields slightly worse results than Conv-TasNet, which suggests that better performance may be achieved by extending the deep CASA framework to the time

domain. In [113], a phase prediction network is trained on top of a DC network. It yields high PESQ. FurcaNeXt [96] produces very high ΔSDR. The deep CASA system generates slightly lower ΔSDR results, but has much fewer parameters. In addition, deep CASA yields the best results in terms of ΔSI-SNR, PESQ and ESTOI. The last three rows present the results of the IBM, IRM and ideal phase-sensitive mask (PSM) with the STFT configuration in Section 5.3.1. Deep CASA systematically outperforms the ideal masks in terms of SDR and SI-SNR. However, there is still room for improvement in terms of PESQ.

5.4 Concluding remarks

We have proposed a deep CASA approach to talker-independent monaural speaker separation. Simultaneous grouping is first conducted to separate two speakers at the frame level. Sequential grouping is then employed to stream separated frame-level spectra into two sources. The deep CASA algorithm optimizes frame-level separation and speaker tracking in turn in the two-stage framework, leading to much better performance than DC and PIT. Our contributions also include novel techniques such as complex ratio masking, SNR objectives, Dense-UNet with frequency mapping layers and TCN with dropDilation. Experimental results on the benchmark WSJ0-2mix dataset show that the proposed algorithm produces the state-of-the-art results, with a modest model size.

A major difference between our sequential grouping stage and deep clustering is that embedding operates at the T-F unit level in DC, and at the frame level in deep CASA. There are several advantages to our approach. First, DC excels at speaker tracking due to clustering, but it is not better than ratio masking for frame-level

separation. Therefore, divide and conquer is a natural choice. Second, deep CASA is more flexible. Almost all DC based algorithms are built on time-frequency processing. Our sequential grouping works on frame-level outputs, which can be produced by estimating magnitude STFT, complex masks, or even time-domain signals. In addition, we reduce the computational complexity of clustering from $O(FT)$ in DC to $O(T)$ in deep CASA.

Chapter 6: Multi-speaker and Causal-separation Extensions to Deep CASA

In the previous chapter, we have proposed a deep CASA approach to talker-independent monaural speaker separation. Although deep CASA achieves state-of-the-art results on the benchmark WSJ0-2mix dataset, it has limitations. First, deep CASA is designed for only two concurrent speakers, and can not be straightforwardly extended to three-speaker mixtures or more. Second, to achieve better performance in frame-level separation and speaker tracking, non-causal components have been adopted extensively in deep CASA, making it unsuitable for real-time speech applications, e.g., telecommunication. To address the first limitation, we propose a multi-speaker extension to deep CASA for C concurrent speakers ($C \geq 2$), which works well for speech mixtures with up to C speakers without the prior knowledge about the speaker number. To achieve causal processing, we revise the connections, normalization and clustering algorithms in the system. Experimental results on the benchmark WSJ0 -2mix and -3mix databases show that both extensions achieve excellent results.

6.1 Introduction

Interference from competing speakers is considered a big challenge for automatic speech processing systems and hearing-impaired listeners. Inspired by human ASA mechanisms, deep CASA breaks down the speaker separation task into two stages, simultaneous grouping and sequential grouping. Compared to one stage systems which optimize the two objectives at the same time, deep CASA substantially mitigates the incorrect assignment of speakers, and leads to significant improvements in speaker separation. Although deep CASA represents a step towards solving the cocktail party problem, it has limitations from the viewpoint of real-world deployment.

First, deep CASA is designed for only two concurrent speakers. The frame-level embedding vector in sequential grouping encodes only two possible speaker assignments. In real acoustic environments, the number of concurrent speakers may go beyond two. As the number of speakers increases, the number of possible speaker assignments, each corresponding to a unique embedding pattern, grows factorially, which poses a problem for both the training and inference of sequential grouping. uPIT based algorithms [64], e.g., Conv-TasNet [81], solve this problem naturally by adding output layers for additional speakers. Clustering based methods, e.g., DC [40] and deep attractor networks (DAN) [80], address this problem by adjusting the number of sources in clustering. However, if the number of speakers is not given beforehand, DC and DAN fail to operate properly as the speaker number is needed for clustering. To tackle this problem, Higuchi et al. [41] perform source counting by computing the rank of the covariance matrix of the embedding vectors. An accuracy of 67.3% is achieved for counting 2- and 3-speaker mixtures, far from practical usage. On the other hand, a C-output uPIT model can be directly applied to speech

mixtures with up to C speakers, without the prior knowledge about the speaker number, as some of the outputs can be trained to generate silence as a placeholder. Recently, more DC based methods [111] [112] [113] start to incorporate uPIT as a parallel training target, and use the spectral outputs from uPIT during inference, to address the problem of unknown speaker number for DC. Another direction for speaker-number-independent separation is to recursively remove one speaker at a time from the mixture [63] [100]. In [100], a one-and-rest permutation invariant training (OR-PIT) algorithm is proposed to train such a network. A binary classifier is trained to produce the stopping signal for the system. Satisfactory results have been achieved on 2- and 3-speaker mixtures.

On the other hand, causal processing is a major concern in many real-time applications, including telecommunication and hearing aids. Telecommunication, such as mobile communication, involves real-time interaction, and is sensitive to processing delay. For hearing prosthesis, a processing delay longer than 10 ms may create artifacts caused by the misalignment between real and processed signals [39]. Deep CASA utilizes both past and future information for separation and speaker tracking. Therefore, it is important to extend deep CASA to causal processing. uPIT based methods address this problem using causal temporal connections. Clustering based methods like DC [40] and DAN [80] struggle to operate causally, as centroids/attractors are hard to estimate in an online fashion. One solution is, again, incorporating uPIT targets as additional outputs. Recently, a low-latency multi-headed DC-uPIT method [2] has been proposed, which is not fully causal, but uses limited future frames for separation. This paper shows the importance of future information for speaker tracking.

In this chapter, we propose multi-speaker and causal-processing extensions to deep CASA. In the multi-speaker extension, a new embedding system is used to address the factorial growth of label permutations. Like uPIT [64] based methods, the multi-speaker extension trained on C speakers produces excellent results for speech mixtures with up to C speakers, with no knowledge about the speaker number. In the causal-separation extension, we replace all non-causal connections and normalization in deep CASA. We also propose two causal clustering algorithms for the sequential grouping stage, both matching the performance of non-causal clustering.

The rest of the chapter is organized as follows. Section 6.2 presents the multi-speaker extension to deep CASA. Causal-separation extensions are introduced in Section 6.3. Section 6.4 presents experimental results, comparisons and analysis. Conclusions and related issues are discussed in Section 6.5.

6.2 Multi-talker extension to deep CASA

The goal of monaural speaker separation is to separate C speakers $x_c(n)$, $c = 1, ..., C$, from a single-channel recording of speech mixture $y(n)$. Deep CASA has been proposed in the previous chapter, which functions well for the co-channel situation where $C = 2$, but can not be readily extended to more speakers. In this section, we present a multi-speaker extension to deep CASA for C concurrent speakers ($C \geq 2$). The extension is presented in two parts: simultaneous grouping and sequential grouping. All symbols and notations follow those in Chapter 5.

6.2.1 Simultaneous grouping

The simultaneous grouping stage separates spectral components of the C speakers at the frame level. The same Dense-UNet is adopted as in Chapter 5. There are

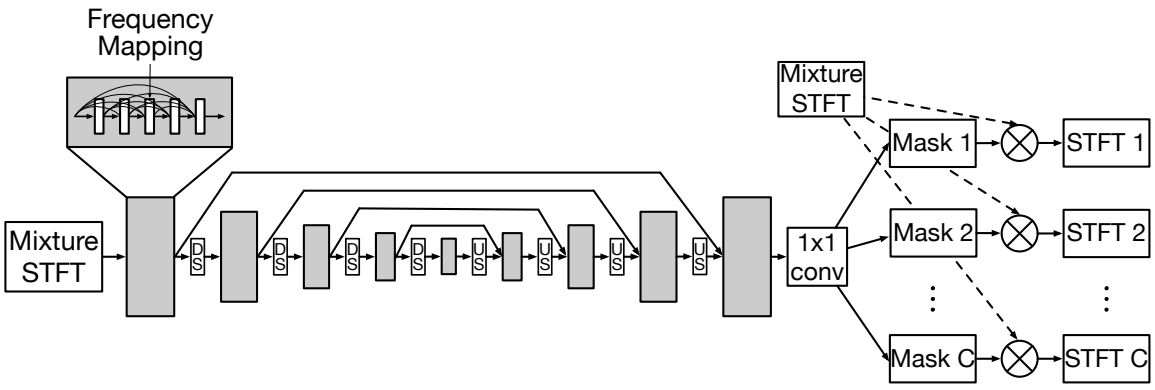


Figure 6.1: Diagram of the Dense-UNet used in multi-speaker simultaneous grouping. Gray blocks denote dense CNN layers. DS blocks denote downsampling layers and US blocks denote upsampling layers. Skip connections are added to connect layers at the same level. The inputs, masks and outputs are defined in the complex STFT domain.

C output layers in the model, matching the total number of speakers, as shown in Fig. 6.1. The time-domain SNR objective $J^{tPIT-SNR}$ is used to tune the network:

$$J^{tPIT-SNR} = \sum_{c=1}^{C} 10 \log \frac{\sum_n x_c(n)^2}{\sum_n (x_c(n) - \hat{x}_{\Theta_c(t)}(n))^2} \tag{6.1}$$

where $\hat{x}_{\Theta_c(t)}(n)$ denotes the organized waveform estimate of speaker c using minimum frame-level loss. Other details, including the number of layers, downsampling/upsampling, and frequency mapping, follow those in Section 5.2.1.

6.2.2 Sequential grouping

The sequential grouping stage tracks all frame-level spectral estimates, and assigns them to the C speakers. A diagram of the multi-speaker extension of sequential grouping is illustrated in Fig. 6.2. Mixture spectrogram and C spectral estimates (including real, imaginary and magnitude STFT) are stacked to form the input to the network.

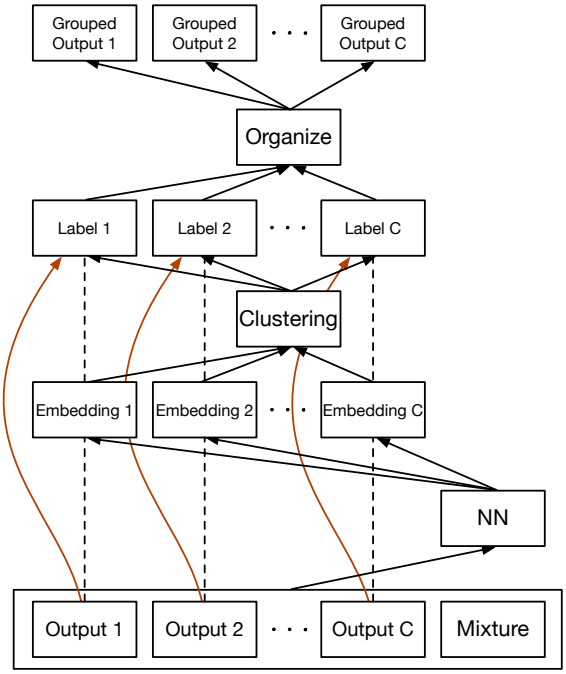


Figure 6.2: Diagram of multi-speaker sequential grouping.

In Chapter 5, a TCN projects frame-level inputs to a D-dimensional embedding vector $\mathbf{V}(t) \in \mathbb{R}^{1\times D}$, which indicates the optimal output assignment of each frame. In the case of two concurrent speakers, there are only two possible output assignments, i.e., swap or no swap. The trained $\mathbf{V}(t)$ exhibits two unique patterns accordingly. However, when the number of speakers C increases, the number of possible assignments is $C! = 1 \times 2 \times ... \times C$, and it becomes intractable to use one vector $\mathbf{V}(t)$ to represent all the assignments. Even if $\mathbf{V}(t)$ can be trained to convey $C!$ patterns, it is difficult to figure out the pattern-assignment pairing during inference.

To avoid these problems, we use a TCN to predict C embedding vectors at each frame $\mathbf{V}_c(t) \in \mathbb{R}^{1\times D}$, each corresponding to one output $\hat{X}_c(t)$ of the Dense-UNet, as shown in Fig. 6.2. The target label for $\mathbf{V}_c(t)$ is a C-dimensional indicator vector,

denoted by $\mathbf{A}_c(t)$. During the training of tPIT, if the minimum loss is achieved when $\hat{X}_c(t)$ is paired with speaker c', the c'th element of $\mathbf{A}_c(t)$ is set to 1, and all other elements are set to 0. In other words, $\mathbf{A}_c(t)$ indicates the optimal speaker assignment of $\hat{X}_c(t)$. Similar to Section 5.2.2, a weight $w_c(t) = \frac{|LD(t)|}{\sum_t |LD(t)|}$ is used during training to emphasize frames where the speaker assignment plays an important role. $LD(t)$ denotes the frame-level loss difference (LD) between the minimum and maximum loss. $w_c(t)$ can be used to construct a $CT \times CT$ diagonal weight matrix $\mathbf{W} = diag(w_c(t))$. $\mathbf{V}_c(t)$ and $\mathbf{A}_c(t)$ can be reshaped into a $CT \times D$ matrix $\mathbf{V}$ and a $CT \times C$ matrix $\mathbf{A}$, respectively. The final weighted objective function between $\mathbf{V}$ and $\mathbf{A}$ is:

$$J^{DC-W} = ||\mathbf{W}^{1/2}(\mathbf{V}\mathbf{V}^T - \mathbf{A}\mathbf{A}^T)\mathbf{W}^{1/2}||_F^2 \tag{6.2}$$

where $||\cdot||_F$ is the Frobenius norm, and $\mathbf{W}^{1/2}$ denotes the element-wise square root of $\mathbf{W}$. Optimizing J^{DC-W} forces $\mathbf{V}_c(t)$ corresponding to the same speaker to get closer during training, and $\mathbf{V}_c(t)$ corresponding to different speakers to become farther apart. The trained $\mathbf{V}_c(t)$ exhibits C unique patterns, each corresponding to one speaker.

During inference, the K-means algorithm is first applied to cluster $\mathbf{V}_c(t)$ into C groups. However, if no post-processing is conducted, several embeddings at one frame may be assigned to the same speaker. We thus design a constrained clustering algorithm to force the frame-level embeddings to different labels, as given in Algorithm 1. The input to the algorithm includes C centroids calculated using the K-means algorithm. In each frame, the resulting permutation $\Theta(t)$ corresponds to the assignment that maximizes the sum of similarities between embeddings and centroids. After the constrained clustering algorithm, frame-level outputs are organized according to their labels, and resynthesized to the time domain.

Algorithm 1 Constrained clustering

Input: Embedding vectors $\mathbf{V}_c(t)$, K-means centroids $\boldsymbol{\mu}_c$

Output: Frame-level labels of all outputs $\Theta(t)$ (resulting permutation)

1: **for** `t in {1,...,T}` **do**

2: $\quad \Theta(t) \leftarrow \underset{\theta(t)\in P}{\operatorname{argmax}} \sum_{c=1}^{C} \mathbf{V}_{\theta_c(t)}(t)\boldsymbol{\mu}_c^T$

3: **end for**

Although multi-speaker deep CASA is designed for C concurrent ($C \geq 2$) speakers, if trained properly, a C-speaker system can generate good results for speech mixtures with less than C speakers, without prior knowledge about the exact speaker number. In such cases, some of the channels produce significantly weaker outputs than other channels, corresponding to silence. The details are presented in Section 6.4.3.

6.3 Causal-separation extension to deep CASA

In this section, we present an extension to achieve causal processing of deep CASA. We analyze the extension from three aspects: temporal convolutions, normalization and clustering.

6.3.1 Temporal convolutions

Dense-UNet and TCN consist of a series of temporal convolutional layers, which are non-causal in the original deep CASA system. The left part of Fig 6.3a illustrates a non-causal temporal convolutional layer in TCN. To generate the output of frame T, future information from frame $T + 1$ is used, making the layer non-causal. In the causal-processing extension, we directly change non-causal convolutions to their causal versions when the temporal resolution stays the same in the input and output, as shown in the right part of Fig 6.3a.

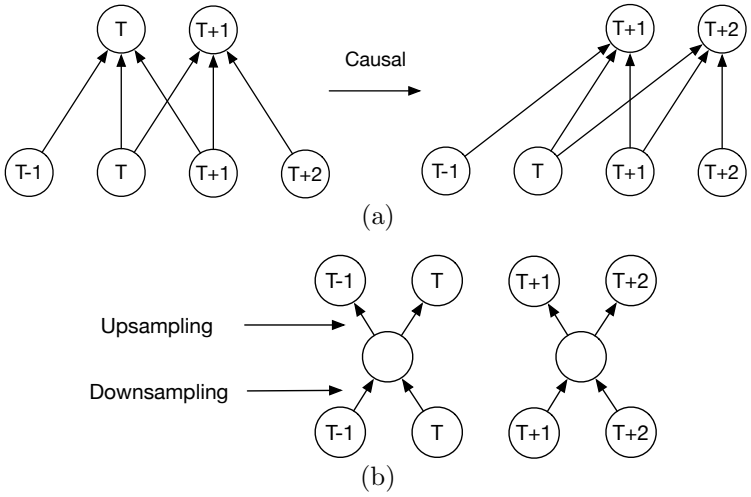


Figure 6.3: Temporal convolutions in deep CASA.

There are two special types of temporal convolutional layers in Dense-UNet, downsampling and upsampling layers. Temporal downsampling is achieved using strided convolutional layers of size 2. Upsampling layers are transpose convolutional layers of size 2. Fig. 6.3b illustrates one pass of temporal downsampling and upsampling. During the downsampling process, inputs from every two frames are encoded into one single unit, which halves the temporal resolution. The upsampling layer then projects the encodings to the original resolution. As a result of encoding, the output at frame $T-1$ requires inputs at both frame $T-1$ and T, making the layers non-causal. Since there is no solution to fix the non-causality of such layers, we remove all frame-wise downsamplings and upsamplings in Dense-UNet, but keep the frequency-wise downsamplings and upsamplings.

6.3.2 Normalization

Normalization is utilized extensively in deep CASA to accelerate training and stabilize neuron activations. Empirical results indicate that the choice of normalization significantly impacts the performance of speaker separation [81]. In non-causal deep CASA, standard layer normalization (LN) [3] is adopted, where the features are normalized over all but the batch dimension. Take Dense-UNet as an example. Feature maps in Dense-UNet have 4 dimensions: $\mathbf{z} \in \mathbb{R}^{B \times T \times F \times K}$, where B, T, F, K denote batch, time, frequency and channel, respectively. A global mean and variance are calculated for each training sample in a batch, and are then utilized to normalize the feature map:

$$E[\mathbf{z}] = \frac{1}{TFK} \sum_{t,f,k} \mathbf{z}(b,t,f,k) \tag{6.3}$$

$$Var[\mathbf{z}] = \frac{1}{TFK} \sum_{t,f,k} (\mathbf{z}(b,t,f,k) - E[\mathbf{z}])^2 \tag{6.4}$$

$$LN(\mathbf{z}) = \frac{\mathbf{z} - E[\mathbf{z}]}{\sqrt{Var[\mathbf{z}] + \epsilon}} \odot \boldsymbol{\gamma} + \boldsymbol{\beta} \tag{6.5}$$

where $\boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^{1 \times 1 \times 1 \times K}$ are trainable gain and bias, and ϵ is a small constant added to variance to avoid dividing by zero. The means and variances are calculated on a whole utterance in both training and inference, which makes layer normalization not applicable to a causal setup.

In this study, we explore three causal normalization techniques as substitutes for layer normalization. In standard batch normalization (BN) [55], features are normalized over all but the channel dimension during training:

$$E[\mathbf{z}] = \frac{1}{BTF} \sum_{b,t,f} \mathbf{z}(b,t,f,k) \tag{6.6}$$

$$Var[\mathbf{z}] = \frac{1}{BTF} \sum_{b,t,f} (\mathbf{z}(b,t,f,k) - E[\mathbf{z}])^2 \tag{6.7}$$

$$BN(\mathbf{z}) = \frac{\mathbf{z} - E[\mathbf{z}]}{\sqrt{Var[\mathbf{z}] + \epsilon}} \odot \boldsymbol{\gamma} + \boldsymbol{\beta} \tag{6.8}$$

where $\boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^{1\times1\times1\times K}$ are trainable gain and bias. Mean and variance gathered in the training phase are utilized for all test utterances. Since recalculation of statistics is not needed, batch normalization is causal during inference.

Because of the complexity of Dense-UNet/TCN, a small batch size is used (4 or 8) during training. Channel-dependent mean and variance in BN may fluctuate severely across mini-batches. We propose a channel-independent version of batch normalization (ciBN) to overcome this issue. In ciBN, features are normalized over all dimensions during training:

$$E[\mathbf{z}] = \frac{1}{BTFK} \sum_{b,t,f,k} \mathbf{z}(b,t,f,k) \tag{6.9}$$

$$Var[\mathbf{z}] = \frac{1}{BTFK} \sum_{b,t,f,k} (\mathbf{z}(b,t,f,k) - E[\mathbf{z}])^2 \tag{6.10}$$

$$ciBN(\mathbf{z}) = \frac{\mathbf{z} - E[\mathbf{z}]}{\sqrt{Var[\mathbf{z}] + \epsilon}} \odot \boldsymbol{\gamma} + \boldsymbol{\beta} \tag{6.11}$$

Mean and variance gathered in training are used for inference.

We also consider a causal version of layer normalization (cLN), where the features are normalized in a causal fashion.

$$E[\mathbf{z}(t=\tau)] = \frac{1}{\tau FK} \sum_{t\leq\tau,f,k} \mathbf{z}(b,t,f,k) \tag{6.12}$$

$$Var[\mathbf{z}(t=\tau)] = \frac{1}{\tau FK} \sum_{t\leq\tau,f,k} (\mathbf{z}(b,t,f,k) - E[\mathbf{z}(t=\tau)])^2 \tag{6.13}$$

$$cLN(\mathbf{z}(t=\tau)) = \frac{\mathbf{z}(t=\tau) - E[\mathbf{z}(t=\tau)]}{\sqrt{Var[\mathbf{z}(t=\tau)] + \epsilon}} \odot \boldsymbol{\gamma} + \boldsymbol{\beta} \tag{6.14}$$

where $\mathbf{z}(t=\tau)$ denotes the τth frame of the feature map. In cLN, normalization is conducted frame by frame, with frame-dependent mean and variance calculated using

all previous frames. A similar normalization technique was used in the causal version of Conv-TasNet [81].

The three normalization techniques can also be applied to the TCN in the sequential grouping stage. All operations stay the same, but the frequency dimension is neglected.

6.3.3 Clustering

Once embedding vectors are generated, a clustering step is needed to assign them to different speakers. Most clustering based speaker separation algorithms, e.g., deep clustering and deep CASA, perform this step in an offline fashion. In deep clustering, the K-means algorithm iteratively generates centroids of clusters using all embedding vectors in the whole utterance. It is difficult to make a causal extension to K-means for deep clustering, as embedding vectors corresponding to some clusters may not be present in the beginning part of an utterance. Therefore, the number of clusters is unclear for causal processing.

On the other hand, in the setting of multi-talker deep CASA, there are C embedding vectors in each frame, each belonging to a unique cluster. The design of causal clustering becomes much easier. The details are given in Algorithm 2.

Algorithm 2 Causal clustering for multi-speaker deep CASA

Input: Embedding vectors $\mathbf{V}_c(t)$, frame-level energy of the mixture $E(t)$, energy threshold α, maximal queue size S_{max}

Output: Frame-level labels of all outputs $\Theta(t)$

for `c in {1,...,C}` **do**
  $Q_c \leftarrow$ `NEW_FIFO_QUEUE()`
  Q_c.`enqueue(`$\mathbf{V}_c(1)$`)`
  $\boldsymbol{\mu}_c \leftarrow Q_c$.`mean()`
  $\Theta_c(1) \leftarrow c$
end for
$E_{avg} \leftarrow E(1)$, $t \leftarrow 2$
while `t` $\geq$ `2` **do**
  $\Theta(t) \leftarrow \underset{\theta(t) \in P}{\operatorname{argmax}} \sum_{c=1}^{C} \mathbf{V}_{\theta_c(t)}(t) \boldsymbol{\mu}_c^T$
  for `c in {1,...,C}` **do**
    if $E(t) > \alpha E_{avg}$ **then**
      Q_c.`enqueue(`$\mathbf{V}_{\Theta_c(t)}(t)$`)`
      if Q_c.`size()` $\geq S_{max}$ **then**
        Q_c.`dequeue()`
      end if
      $\boldsymbol{\mu}_c \leftarrow Q_c$.`mean()`
    end if
  end for
  $E_{avg} \leftarrow ((t-1)E_{avg} + E(t))/t$
  $t \leftarrow t + 1$
end while

At the start of the algorithm, C first-in-first-out (FIFO) queues are created to store embedding vectors belonging to the clusters. Each embedding vector in the first frame is pushed to one of the queues to form the initial data. Centroids of the clusters are calculated as mean values of the queues. Starting from frame 2, each embedding vector is assigned to a unique cluster using the assignment that maximizes the sum of similarities between embeddings and centroids. If the energy of the current frame is insignificant, we move to the next frame. Otherwise, we push the embedding vectors to their corresponding queues, and update the centroids. In order to keep the centroids

relatively local to the current frame, we remove the oldest item in the queue when the maximal queue size S_{max} is reached. To decide whether a frame has significant energy, we keep track of the average frame energy E_{avg}. Frames weaker than αE_{avg} is considered uninformative, and would not be used for centroids calculation. The frame-level assignment continues until all frames are processed. The two parameters α and S_{max} are set to 0.3 and 20 in our study.

Algorithm 3 Causal clustering for two-speaker deep CASA

Input: Embedding vectors $\mathbf{V}(t)$, frame-level energy of the mixture $E(t)$, energy threshold α, similarity threshold ρ, maximal queue size S_{max}

Output: Frame-level label $\Theta(t)$

for `c in {1,2}` **do**
  $Q_c \leftarrow$ `NEW_FIFO_QUEUE()`
end for
Q_1`.enqueue(`$\mathbf{V}(1)$`)`
$\boldsymbol{\mu}_1 \leftarrow \mathbf{V}(1)$, $E_{avg} \leftarrow E(1)$, $\Theta(1) \leftarrow 1$, $t \leftarrow 2$
while `t` $\geq$ `2` **do**
  if Q_2`.empty()` **then**
    if $\mathbf{V}(t-1)\mathbf{V}(t)^T < \rho$ **then**
      $\Theta(t) \leftarrow 2$
    else
      $\Theta(t) \leftarrow 1$
    end if
  else
    $\Theta(t) \leftarrow \underset{c\in\{1,2\}}{\operatorname{argmax}} \mathbf{V}(t)\boldsymbol{\mu}_c^T$
  end if
  if $E(t) > \alpha E_{avg}$ `or` (Q_2`.empty()` `and` $\Theta(t) == 2$) **then**
    $Q_{\Theta(t)}$`.enqueue(`$\mathbf{V}(t)$`)`
    if $Q_{\Theta(t)}$`.size()` $\geq S_{max}$ **then**
      $Q_{\Theta(t)}$`.dequeue()`
    end if
    $\boldsymbol{\mu}_{\Theta(t)} \leftarrow Q_{\Theta(t)}$`.mean()`
  end if
  $E_{avg} \leftarrow ((t-1)E_{avg} + E(t))/t$
  $t \leftarrow t+1$
end while

We also design a causal clustering algorithm for the original two-speaker deep CASA system, as shown in Algorithm 3. In two-speaker deep CASA, each frame only has one embedding vector, indicating the frame-level optimal assignment. At the first frame, we create 2 FIFO queues to store embedding vectors. The first embedding vector is pushed to the first queue. Starting from frame two, if the second queue is empty, we check the similarity of embedding vectors between the current frame and the previous frame. If the similarity is lower than ρ, we set the current frame to cluster 2, and push the embedding vector to the second queue. Otherwise the current frame is set to cluster 1, and the checking continues. Once the second queue loads the first item, the algorithm starts to follow the same process as in Algorithm 2. The energy threshold α, similarity threshold ρ, and S_{max}, are set to 0.3, 0.5 and 20, respectively. Both Algorithm 2 and 3 are easy to implement and fast during inference.

6.4 Evaluation and comparison

6.4.1 Experimental setup

We evaluate our system on two-speaker and three-speaker separation datasets, WSJ0-2mix and WSJ0-3mix [40]. Both datasets have a 30-hour training set and a 10-hour validation set generated by selecting random speakers in the Wall Street Journal (WSJ0) training set, and mixing them at various SNRs between 0 dB and 5 dB. Evaluation is conducted on the 5-hour open-condition (OC) test set, which is similarly generated using 16 untrained speakers from the WSJ0 development set. All mixtures are sampled at 8 kHz. STFT with a frame length of 32ms, a frame shift of 8 ms, and a square root Hanning window is calculated for the whole system.

Performance is evaluated in terms of signal-to-distortion ratio improvement (ΔSDR) [103], perceptual evaluation of speech quality (PESQ) [57], and extended short-time objective intelligibility (ESTOI) [59]. We also report results in terms of scale-invariant signal-to-noise ratio improvement (ΔSI-SNR) [81] for a systematical comparison with other systems.

6.4.2 Models

Both multi-speaker and causal-separation extensions adopt the basic structure of Dense-UNet and TCN as in Chapter 5.

In Dense-UNet, the number of channels K is set to 64, the total number of dense layers L is set to 5, and all CNN layers have a kernel size of 3×3 and a stride of 1×1. The middle layer in each dense block is replaced with a frequency mapping layer. The network is optimized with respect to $J^{tPIT-SNR}$.

In TCN, the maximum dilation factor is set to $2^6 = 64$. The number of bottleneck units B is selected as 256. The number of units in depthwise dilated convolutional layers H is set to 512. DropDilation with $p = 0.7$ is applied during training.

Both networks are trained with the Adam optimization algorithm [62]. The initial learning rate is set to 0.0001 for Dense-UNet, and 0.00025 for TCN. Learning rate adjustment and early stopping are employed based on the loss on the validation set.

For multi-speaker deep CASA with C speakers, we change the number of output layers in Dense-UNet to C, and number of embedding vectors in TCN to C, with 40 dimensions for each embedding vector. For causal deep CASA, temporal connections, normalization and clustering are modified as described in Section 6.3.

Table 6.1: Average ΔSDR, PESQ, ESTOI and frame assignment errors for deep CASA evaluated on WSJ0-2mix OC.

	# of param.	ΔSDR (dB)	PESQ	ESTOI (%)	Frame Assign. Errors
Mixture	-	0.0	2.02	56.1	-
Two-speaker deep CASA	12.8M	17.9	3.49	92.9	1.38%
Multi-speaker deep CASA ($C = 2$)	12.8M	17.7	3.47	92.6	1.69%
Deep CASA with optimal spk. assign.	4.7M	19.1	3.63	94.3	0.00%

6.4.3 Results and comparisons

We first evaluate the multi-speaker extension to deep CASA. Unlike two-speaker deep CASA in Chapter 5, the multi-speaker extension is more flexible and can be applied to C-speaker mixtures, $C \in \{2, 3, ...\}$. Table 6.1 compares two-speaker deep CASA with multi-speaker deep CASA when $C = 2$. The two models share the same simultaneous grouping module, but have different embedding configurations and clustering strategies in sequential grouping. Both systems are trained and evaluated on the WSJ0-2mix dataset, without joint optimization. As shown in the table, the two-speaker system slightly outperforms the multi-speaker extension in terms of all four metrics. The results reflect the principle of Occam's razor. When the number of concurrent speakers is fixed to 2, one embedding vector per frame is enough to indicate the optimal output assignment. The extra embedding vectors in multi-speaker deep CASA do not convey much information, and lead to worse performance during testing.

We then evaluate multi-speaker deep CASA for three-speaker mixtures. A multi-speaker ($C = 3$) model is trained and tested on WSJ0-3mix. Table 6.2 presents the results of simultaneous grouping with the optimal output assignment. Table 6.3 summarizes the final results of multi-speaker deep CASA with TCN sequential grouping, and compares it with other state-of-the-art methods on WSJ0-3mix. For all methods,

Table 6.2: Average ΔSDR, PESQ and ESTOI for simultaneous grouping with the optimal output assignment on WSJ0-3mix OC.

	ΔSDR (dB)	PESQ	ESTOI (%)
Mixture	0.0	1.66	38.5
tPIT Dense-UNet	17.5	3.16	86.7

Table 6.3: Number of parameters, average ΔSDR, ΔSI-SNR, PESQ and ESTOI for various state-of-the-art systems evaluated on WSJ0-3mix OC.

	# of param.	ΔSDR (dB)	ΔSI-SNR (dB)	PESQ	ESTOI (%)
Mixture	-	0.0	0.0	1.66	38.5
uPIT [64]	92.7M	7.7	-	-	-
ADANet [80]	9.1M	9.4	9.1	2.16	-
Conv-TasNet [81]	5.1M	13.1	12.7	2.61	-
Wang et al. [113]	56.6M	12.5	12.1	2.77	-
Multi-speaker deep CASA	12.8M	**14.6**	**14.3**	**2.77**	**80.8**
IBM	-	13.6	13.3	2.86	82.1
IRM	-	13.0	12.6	3.44	88.6
PSM	-	16.8	16.4	3.80	93.7

we list the best reported results, and leave unreported fields blank. The numbers of parameters in different methods are estimated according to their papers. Two important observations can be made out of the two tables. First, for multi-speaker deep CASA, the performance gap between the optimal and estimated speaker assignment rises from 1.4 dB ΔSDR for two speakers to 3.9 dB for three speakers. This is due to the fact that as the number of speakers increases, the number of possible output permutations rises factorially. Our multi-speaker model keeps the assignment errors to a fairly low level despite the difficulty of the problem. Second, our three-speaker model outperforms all existing systems, including the initial uPIT [64], anchored deep attractor network [80], Conv-TasNet [81], and phase reconstruction network [113] in

Table 6.4: Average ΔSDR, PESQ and ESTOI for simultaneous grouping with the optimal output assignment on WSJ0-2mix OC and WSJ0-3mix OC .

# of output	Training set	WSJ0-2mix OC			WSJ0-3mix OC		
		ΔSDR (dB)	PESQ	ESTOI (%)	ΔSDR (dB)	PESQ	ESTOI (%)
2	WSJ0-2mix	19.1	3.63	94.3	-	-	-
3	WSJ0-3mix	18.6	3.54	93.5	17.5	3.16	86.7
3	WSJ0-2mix, WSJ0-3mix	19.1	3.58	94.1	17.5	3.13	86.5

all metrics. The improvements in ΔSDR and ΔSI-SNR are quite significant. Our PESQ score matches Wang et al.'s result [113], possibly because phase reconstruction is optimized jointly in the time and spectral domain in [113], and PESQ is more relevant to spectral patterns. Despite the success of multi-speaker deep CASA, there is a speech quality (PESQ) gap from ideal masks.

Although multi-speaker deep CASA is designed for C concurrent speakers, if trained properly, it can be used to separate speech mixtures with up to C speakers, without prior knowledge of the speaker number. To illustrate the flexibility of multi-speaker deep CASA, we train and test a three-speaker model on both the WSJ0-2mix and WSJ0-3mix datasets, i.e., on both two- and three-speaker mixtures. To be able to train the three-speaker models with WSJ0-2mix, we extend WSJ0-2mix with a third "silent" channel, which consists of white Gaussian noise with an energy level 40 dB below that of the mixture. During evaluation, we select the outputs with a significant energy as active speakers.

Table 6.4 compares simultaneous grouping of multi-speaker deep CASA trained with different numbers of speakers. For all models, outputs are organized with the optimal speaker assignment before evaluation. The three-speaker Dense-UNet trained on WSJ0-2mix and WSJ0-3mix slightly underperforms the models with matched

Table 6.5: Average ΔSDR, ΔSDR, PESQ and ESTOI for various speaker-number-independent systems evaluated on WSJ0-2mix OC and WSJ0-3mix OC.

	WSJ0-2mix OC				WSJ0-3mix OC			
	ΔSDR (dB)	ΔSI-SNR (dB)	PESQ	ESTOI (%)	ΔSDR (dB)	ΔSI-SNR (dB)	PESQ	ESTOI (%)
uPIT [64]	10.1	-	-	-	7.8	-	-	-
OR-PIT [100]	15.0	14.8	3.12	-	12.9	12.6	2.60	-
Multi-speaker deep CASA	**17.9**	**17.7**	**3.43**	**92.5**	**14.5**	**14.3**	**2.75**	**80.6**

training/test speaker numbers on respective datasets, but outperforms the three-speaker Dense-UNet trained on WSJ0-3mix in terms of two-speaker separation.

Table 6.5 illustrates the final results of three-speaker deep CASA trained on WSJ0-2mix and WSJ0-3mix, and compares it with other state-of-the-art speaker-number-independent approaches trained on WSJ0-2mix and WSJ0-3mix. All comparison approaches are uPIT based, as deep clustering based methods do not perform well when the number of speakers is unknown. The results are reported for both WSJ0-2mix OC and WSJ0-3mix OC. The proposed system substantially outperforms the other two approaches in terms of all four metrics. A slight performance drop is observed when we switch from speaker-number-dependent training to speaker-number-independent training.

Next, we evaluate the causal-separation extension to deep CASA for two-speaker mixtures. Different simultaneous grouping models are compared in Table 6.6. Outputs are organized with the optimal speaker assignment before evaluation. The first row corresponds to Dense-UNet with non-causal connections and normalization. A modest performance drop is observed if we switch to the causal version. Two normalization techniques are evaluated. BN leads to negligibly better results than ciBN.

Table 6.6: Average ΔSDR, PESQ and ESTOI for simultaneous grouping models with the optimal output assignment on WSJ0-2mix OC.

	Normalization	Causal	ΔSDR (dB)	PESQ	ESTOI (%)
Non-causal Dense-UNet	LN	✗	19.1	3.63	94.3
Causal Dense-UNet	BN	✓	18.0	3.52	93.2
Causal Dense-UNet	ciBN	✓	17.8	3.52	93.0

Table 6.7: Average ΔSDR, PESQ and ESTOI for sequential grouping models on WSJ0-2mix OC.

Seq. Group.	Normalization	Clustering	Causal	ΔSDR (dB)	PESQ	ESTOI (%)
Two-speaker TCN	BN	Causal	✓	13.9	3.02	87.0
Two-speaker TCN	ciBN	Causal	✓	14.6	3.12	88.5
Two-speaker TCN	cLN	Causal	✓	15.1	3.19	89.5
Multi-speaker TCN	cLN	Causal	✓	14.8	3.15	89.0

Due to slow training, we did not use cLN for causal Dense-UNet, and leave it as future work.

Table 6.7 compares different sequential grouping models for causal deep CASA. All TCNs are causal, and trained on top of the causal Dense-UNet with BN. All clustering algorithms in this table are also causal. The first three rows compare three normalization techniques under the two-speaker Deep CASA setup. Thanks to the matched calculation of mean and variance in the training and test, cLN significantly outperforms the other two techniques. We also train a causal TCN with cLN under the multi-speaker setup, which performs slightly worse than that under the two-speaker setup, consistent with the results in Table 6.1.

Table 6.8: Average ΔSDR, PESQ and ESTOI for causal TCN on WSJ0-2mix OC.

Seq. Group.	Normalization	Clustering	Causal	ΔSDR (dB)	PESQ	ESTOI (%)
Two-speaker TCN	cLN	Offline	✗	15.2	3.19	89.5
Multi-speaker TCN	cLN	Offline	✗	14.8	3.16	89.0

Table 6.8 reports results for causal TCN with non-causal clustering. All settings in the table follow the last two rows of Table 6.7 except for the clustering algorithms. The causal clustering algorithms in Table 6.7 yield almost the same results as non-causal offline clustering, demonstrating the effectiveness of the proposed algorithm.

Finally, Table 6.9 compares causal deep CASA and other state-of-the-art talker-independent causal methods on WSJ0-2mix OC. The Listen and Group system [70] estimates frame-level spectral outputs in an autoregressive fashion. It consists of two stages. In the first stage, the frame-level mixture and source estimates from the previous frame are transformed into mid-level representations. The second stage groups mid-level representations to two sources. We present the fully causal version of Listen and Group, which has no look-aheads for phase reconstruction. Other models include causal versions of uPIT, LSTM-TasNet and Conv-TasNet. As demonstrated in the table, our causal deep CASA system outperforms all existing methods by a large margin.

6.5 Concluding remarks

We have proposed multi-speaker and causal-separation extensions to deep CASA for talker-independent speaker separation. In the multi-speaker extension, we redesign

Table 6.9: Number of parameters, average ΔSDR, ΔSI-SNR, PESQ and ESTOI for various state-of-the-art causal systems evaluated on WSJ0-2mix OC.

	# of param.	Causal	ΔSDR (dB)	ΔSI-SNR (dB)	PESQ	ESTOI (%)
Mixture	-	-	0.0	0.0	2.02	56.1
uPIT [64]	46.3M	✓	7.0	-	-	-
Conv-TasNet [81]	5.1M	✓	11.0	10.6	-	-
LSTM-TasNet [81]	5.1M	✓	11.2	10.8	-	-
Listen and Group [70]	8.2M	✓	11.0	-	-	-
Causal deep CASA	12.8M	✓	**15.1**	**14.9**	**3.19**	**89.5**

the sequential grouping stage to deal with the situation where the number of concurrent speakers is greater than 2. The extension works well even if the speaker number is not given beforehand. In the causal-separation extension, we extensively revise temporal connections and normalization, and propose two causal clustering algorithms. Experimental results on the benchmark WSJ0-2mix and WSJ0-3mix datasets show that the proposed extensions outperform all published results for speaker separation under the multi-speaker and causal setup. This study represents a major step towards speaker separation in real-world applications.

Future work includes examining the performance of causal deep CASA with an unknown number of speakers beyond three. Noise- and reverberation-robust extensions also need to be explored to deal with more realistic acoustic environments [21].

Chapter 7: Conclusions and future work

7.1 Contributions

As a traditional separation approach inspired by human auditory scene analysis principles, CASA has been proven effective in a variety of speech applications [51] [85] [86] [124]. In this dissertation, we have investigated CASA from a deep learning perspective. Specifically, we have explored deep learning based pitch tracking in the presence of noise and competing speakers, with speaker-dependent and speaker-independent training. We have also proposed a deep CASA framework for talker-independent speaker separation, which realizes CASA mechanisms in a deep learning model. Extensions to deep CASA in Chapter 6 have been shown to generalize well to an unknown number of speakers and causal processing, representing a big stride in solving the cocktail party problem.

In Chapter 2, we have introduced LSTM for noise-robust single-pitch tracking. Both conventional LSTM and TF-LSTM are utilized for pitch probability estimation. We find that larger training sets and speaker-dependent training greatly benefit the performance of deep learning. Thanks to the power of sequence modeling, both LSTM based models outperform a feedforward DNN model. With the frequency

scanning mechanism, TF-LSTM achieves better performance in low SNR conditions, and reduces the voicing decision errors of conventional LSTM by 10% at -10 dB.

Chapter 3 presents SD-DNNs and SPD-DNNs for multi-pitch estimation. Thanks to discriminative modeling and speaker-dependent information, SD-DNNs and SPD-DNNs produce good pitch estimates in terms of both accuracy and speaker assignments, and significantly outperform other SD and SI models. SPD-DNNs perform almost the same on same-gender and different-gender speech mixtures, demonstrating the power of SD modeling. During the training of GPD-DNNs, the output-speaker pairing is based on heuristics, which does a reasonable job but slightly underperforms uPIT based methods, as pointed out in Chapter 4. Given limited speaker-dependent training data, speaker adaptation of GPD-DNNs is very effective for reducing pitch errors, especially for same-gender speaker pairs, mostly due to the fact that the suboptimal label assignments in GPD-DNNs are corrected during adaptation. Multi-ratio trained SD-DNNs and SPD-DNNs produce consistent results across various speaker ratios, which shows the generalization of DNNs when exposed to multi-condition training data.

Chapter 4 presents speaker-independent extensions to the NN-FHMM multi-pitch tracking framework. The uPIT technique is adopted to address the label permutation problem during training, which leads to better performance than heuristic GPD models. The best results are achieved by the uPIT-SS-PERM-PITCH model, which indicates that speaker separation benefits multi-pitch tracking, not only with the separated speech, but also with its label permutation during training.

In Chapter 5, deep CASA is proposed to address talker-independent speaker separation. The deep CASA approach follows CASA principles, and optimizes frame-level separation and speaker tracking in the two-stage framework, leading to much better performance than DC and PIT. Various deep learning techniques have been proposed in this study, including frequency mapping layers and drop dilation, leading to cumulative improvements in the final results.

Chapter 6 presents multi-speaker and causal-separation extensions to deep CASA. Sequential grouping is redesigned in the multi-speaker extension to address the factorial increase of label permutations. The multi-speaker extension substantially outperforms the competing systems in terms of ΔSDR and ΔSNR. But the PESQ improvement is still limited. When tested with an unknown number of speakers, the multi-speaker extension generalizes well, and only slightly underperforms the matched-speaker-number models. In the causal-separation extension, temporal connections, normalization, and clustering algorithms are extensively modified for real-time processing. Causal deep CASA produces the best results on WSJ0-2mix OC. However, detailed results indicate that the performance on same-gender mixtures are much worse than different-gender mixtures. Better sequence models are needed to address this issue.

7.2 Future work

In this dissertation, we have proposed several deep learning based pitch tracking algorithms in the presence of noise and competing speakers. Although pitch may not be critical to speech separation in the era of deep learning, it is still useful for other audio processing tasks. Singing pitch detection and note detection are important

real-world applications of pitch tracking. If enough training data are collected, we can extend our speaker-independent multi-pitch tracking framework to address these two tasks. Attention needs to be given to real-time implementation. One should explore more causal sequence models. Model compression is also needed for efficient deployment.

Deep CASA represents a big step towards solving the cocktail party problem. To further extend the deep CASA framework, the following aspects need to be considered:

- *Extension to time-domain processing.* As mentioned in Chapter 5, the time-domain Conv-TasNet [81] yields much better results than a spectral-domain counterpart of Conv-TasNet. It would be interesting to see how deep CASA performs with processing in the time domain [90].

- *Extension to noisy mixtures.* The most straightforward noise-robust extension to deep CASA is to treat noise as another source, and use an additional layer for noise in the output. However, since noise has different characteristics from speech, this may not be a good solution. A two-stage system, with the first stage for denoising and the second stage for speaker separation, may be better.

- *Extension to room reverberation.* A two-stage deep CASA system has been readily extended to perform 2-talker speaker separation and dereverberation [21]. Future extension is needed to incorporate more speakers and non-speech noise.

- *Iterative processing of simultaneous grouping and sequential grouping.* In the tandem algorithm [51], an iterative process of pitch estimation and mask estimation is performed, which produces a set of consistent pitch-mask pairs.

However, in our deep CASA framework, simultaneous grouping is performed first without any information from sequential grouping. It would be interesting to explore how the iterative processing in [51] can be applied to deep CASA.

Bibliography

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

[2] R. Aihara, T. Hanazawa, Y. Okato, G. Wichern, and J. Le Roux. Teacher-student deep clustering for low-delay single channel speech separation. In *Proc. ICASSP*, pages 690–694, 2019.

[3] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[4] F. Bach and M. Jordan. Discriminative training of hidden Markov models for multiple pitch tracking. In *Proc. ICASSP*, pages 489–492, 2005.

[5] P. C. Bagshaw, S. M. Hiller, and M. A. Jack. Enhanced pitch tracking and the processing of F0 contours for computer aided intonation teaching. In *Proc. Eurospeech*, pages 1003–1006, 1993.

[6] S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[7] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5:157–166, 1994.

[8] P. Boersma and D. Weenink. Praat, a system for doing phonetics by computer. In *Glot Int.*, volume 5, pages 341–345, 2001.

[9] A.S. Bregman. *Auditory scene analysis*. Cambridge MA: MIT Press, 1990.

[10] B. Brookes. Voicebox: Speech processing toolbox for matlab.

[11] D. S. Brungart. Informational and energetic masking effects in the perception of two simultaneous talkers. *J. Acoust. Soc. Amer.*, 109:1101–1109, 2001.

[12] C. Chen, R. Gopinath, M. Monkowski, M. Picheny, and K. Shen. New methods in continuous mandarin speech recognition. In *Proc. Eurospeech*, pages 1543–1546, 1997.

[13] J. Chen, Y. Wang, and D. L. Wang. A feature study for classification-based speech separation at low signal-to-noise ratios. *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, 22:1993–2002, 2014.

[14] J. Chen, Y. Wang, S. E. Yoho, D. L. Wang, and E. W. Healy. Large-scale training to increase speech intelligibility for hearing-impaired listeners in novel noises. *J. Acoust. Soc. Amer.*, 139:2604–2612, 2016.

[15] E. C. Cherry. Some experiments on the recognition of speech, with one and with two ears. *J. Acoust. Soc. Amer.*, 25:975–979, 1953.

[16] A. D. Cheveigné and H. Kawahara. YIN, a fundamental frequency estimator for speech and music. *J. Acoust. Soc. Amer.*, 111:1917–1930, 2002.

[17] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proc. CVPR*, pages 1251–1258, 2017.

[18] M. G. Christensen and A. Jakobsson. *Multi-Pitch Estimation*. Morgan & Claypool, San Rafael, CA, USA, 2009.

[19] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). In *Proc. ICLR*, 2016.

[20] M. Cooke, J. Barker, S. Cunningham, and X. Shao. An audio-visual corpus for speech perception and automatic speech recognition. *J. Acoust. Soc. Amer.*, 120:2421–2424, 2006.

[21] M. Delfarah. *Deep learning methods for speaker separation in reverberant conditions*. PhD thesis, Department of Computer Science and Engineering, The Ohio State University, 2019.

[22] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Trans. Patt. Mach. Intell.*, 38:295–307, 2015.

[23] T. Drugman and A. Alwan. Joint robust voicing detection and pitch estimation based on residual harmonics. In *Proc. Interspeech*, pages 1973–1976, 2011.

[24] J. Du, Y. Tu, Y. Xu, L. R. Dai, and C. H. Lee. Speech separation of a target speaker based on deep neural networks. In *Proc. ICSP*, pages 473–477, 2014.

[25] Z. Duan, J. Han, and B. Pardo. Multi-pitch streaming of harmonic sound mixtures. *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, 22:138–150, 2013.

[26] H. Erdogan, J. R. Hershey, and S. Watanabe. Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks. In *Proc. ICASSP*, pages 708–712, 2015.

[27] G. D. Forney Jr. The viterbi algorithm. In *Proc. IEEE*, volume 61, pages 268–278, 1973.

[28] Y. Gal and Z. Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Proc. NIPS*, pages 1019–1027, 2016.

[29] Z. Ghahramani and M. Jordan. Factorial hidden Markov models. In *Proc. NIPS*, pages 472–478, 1996.

[30] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Proc. AISTATS*, pages 315–323, 2011.

[31] S. Gonzalez and M. Brookes. PEFAC-A pitch estimation algorithm robust to high levels of noise. *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, 22:518–530, 2014.

[32] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proc. NIPS*, pages 2672–2680, 2014.

[33] A. Graves, M. Liwicki, H. Bunke, J. Schmidhuber, and S. Fernández. Unconstrained on-line handwriting recognition with recurrent neural networks. In *Proc. NIPS*, pages 577–584, 2008.

[34] A. Graves, A. Mohamed, and G. E. Hinton. Speech recognition with deep recurrent neural networks. In *Proc. ICASSP*, pages 6645–6649, 2013.

[35] K. Han and D. L. Wang. Neural network based pitch tracking in very noisy speech. *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, 22:2158–2168, 2014.

[36] M. L. Hawley, R. Y. Litovsky, and J. F. Culling. The benefit of binaural hearing in a cocktail party: Effect of location and type of interferer. *J. Acoust. Soc. Amer.*, 115:833–834, 2004.

[37] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. ICCV*, pages 1026–1034, 2015.

[38] E. W. Healy, M. Delfarah, J. L. Vasko, B. L. Carter, and D. L. Wang. An algorithm to increase intelligibility for hearing impaired listeners in the presence of a competing talker. *J. Acoust. Soc. Amer.*, 141:4230–4239, 2017.

[39] R. Herbig and J. Chalupper. Acceptable processing delay in digital hearing aids. *Hearing Review*, 17:28–31, 2010.

[40] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe. Deep clustering: Discriminative embeddings for segmentation and separation. In *Proc. ICASSP*, pages 31–35, 2016.

[41] T. Higuchi, K. Kinoshita, M. Delcroix, K. Zmolíková, and T. Nakatani. Deep clustering-based beamforming for separation with unknown number of sources. In *Proc. Interspeech*, pages 1183–1187, 2017.

[42] G. E. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition. *IEEE SIGNAL PROC MAG*, 29:82–97, 2012.

[43] G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

[44] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[45] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[46] J. Holdsworth, I. Nimmo-Smith, R. Patterson, and P. Rice. Implementing a gammatone filter bank. Technical report, MRC Applied Psychology Unit, Cambridge, 1988.

[47] C. C. Hsu, H. T. Hwang, Y. C. Wu, Y. Tsao, and H. M. Wang. Voice conversion from unaligned corpora using variational autoencoding wasserstein generative adversarial networks. *arXiv preprint arXiv:1704.00849*, 2017.

[48] G. Hu. 100 nonspeech sounds, 2006.

[49] G. Hu. *Monaural speech organization and segregation*. PhD thesis, Department of Computer Science and Engineering, The Ohio State University, 2006.

[50] G. Hu and D. L. Wang. An auditory scene analysis approach to monaural speech segregation. In E. Hänsler and G. Schmidt, editors, *Topics in Acoustic Echo and Noise Control*, chapter 12, pages 485–515. Springer, Heidelberg, 2006.

[51] G. Hu and D. L. Wang. A tandem algorithm for pitch estimation and voiced speech segregation. *IEEE Trans. Audio, Speech, Lang. Process.*, 18:2067–2079, 2010.

[52] K. Hu and D. L. Wang. An unsupervised approach to cochannel speech separation. *IEEE Trans. Audio, Speech, Lang. Process.*, 21:122–131, 2013.

[53] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proc. CVPR*, pages 4700–4708, 2017.

[54] P. S. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis. Deep learning for monaural speech separation. In *Proc. ICASSP*, pages 1562–1566, 2014.

[55] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML*, pages 448–456, 2015.

[56] Y. Isik, J. Le Roux, Z. Chen, S. Watanabe, and J. R. Hershey. Single-channel multi-speaker separation using deep clustering. In *Proc. Interspeech*, pages 545–549, 2016.

[57] ITU-R. Perceptual evaluation of speech quality (PESQ) An objective method for end-to-end speech quality assessment of narrowband telephone networks and speech codecs. *Recommendation P.862*, 2001.

[58] A. Jansson, E. Humphrey, N. Montecchio, R. Bittner, A. Kumar, and T. Weyde. Singing voice separation with deep U-Net convolutional networks. In *Proc. ISMIR*, 2017.

[59] J. Jensen and C. H. Taal. An algorithm for predicting the intelligibility of speech masked by modulated noise maskers. *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, 24:2009–2022, 2016.

[60] Z. Jin and D. L. Wang. HMM-based multipitch tracking for noisy and reverberant speech. *IEEE Trans. Audio, Speech, Lang. Process.*, 19:1091–1102, 2011.

[61] M. Jordan, Z. Ghahramani, T. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 1999.

[62] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2015.

[63] K. Kinoshita, L. Drude, M. Delcroix, and T. Nakatani1. Listening to each speaker one by one with recurrent selective hearing networks. In *Proc. ICASSP*, pages 5064–5068, 2018.

[64] M. Kolbæk, D. Yu, Z. H. Tan, and J. Jensen. Multitalker speech separation with utterance-level permutation invariant training of deep recurrent neural networks. *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, 25:1901–1913, 2017.

[65] S. G. Koolagudi and K. S. Rao. Emotion recognition from speech: a review. *International Journal of Speech Technology*, 15:99–117, 2012.

[66] C. Lea, R. Vidal, A. Reiter, and G. D. Hager. Temporal convolutional networks: A unified approach to action segmentation. In *Proc. ECCV*, pages 47–54, 2016.

[67] B. S. Lee and D. P. W. Ellis. Noise robust pitch tracking by subband autocorrelation classification. In *Proc. Interspeech*, 2012.

[68] C. Li, L. Zhu, S. Xu, P. Gao, and B. Xu. CBLDNN-based speaker-independent speech separation via generative adversarial training. In *Proc. ICASSP*, pages 711–715, 2018.

[69] J. Li, A. Mohamed, G. Zweig, and Y. Gong. LSTM time and frequency recurrence for automatic speech recognition. In *Proc. ASRU*, pages 187–191, 2015.

[70] Z.-X. Li, Y. Song, L.-R. Dai, and I. McLoughlin. Listening and grouping: an online autoregressive approach for monaural speech separation. *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, 27:692–703, 2019.

[71] H. Liao. Speaker adaptation of context dependent deep neural networks. In *Proc. ICASSP*, pages 7947–7951, 2013.

[72] P. Lieberman, E. S. Crelin, and D. H. Klatt. Phonetic ability and related anatomy of the newborn and adult human, Neanderthal man, and the chimpanzee. *American Anthropologist*, 74:287–307, 1972.

[73] Y. Liu and D. L. Wang. Speaker-dependent multipitch tracking using deep neural networks. In *Proc. Interspeech*, pages 3279–3283, 2015.

[74] Y. Liu and D. L. Wang. Robust pitch tracking in noisy speech using speaker-dependent deep neural networks. In *Proc. ICASSP*, pages 5255–5259, 2016.

[75] Y. Liu and D. L. Wang. Speaker-dependent multipitch tracking using deep neural networks. *J. Acoust. Soc. Amer.*, 141:710–721, 2017.

[76] Y. Liu and D. L. Wang. Time and frequency domain long short-term memory for noise robust pitch tracking. In *Proc. ICASSP*, pages 5600–5604, 2017.

[77] Y. Liu and D. L. Wang. A CASA approach to deep learning based speaker-independent co-channel speech separation. In *Proc. ICASSP*, pages 5399–5403, 2018.

[78] Y. Liu and D. L. Wang. Permutation invariant training for speaker-independent multi-pitch tracking. In *Proc. ICASSP*, pages 5594–5598, 2018.

[79] Y. Liu and D. L. Wang. Divide and conquer: A deep CASA approach to talker-independent monaural speaker separation. *arXiv preprint arXiv:1904.11148,* 2019. *IEEE/ACM Trans. Audio, Speech, and Lang. Process.*, revision under review.

[80] Y. Luo, Z. Chen, and N. Mesgarani. Speaker-independent speech separation with deep attractor network. *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, 26:787–796, 2018.

[81] Y. Luo and N. Mesgarani. Conv-TasNet: Surpassing ideal time-frequency magnitude masking for speech separation. *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, 27:1256–1266, 2019.

[82] F. Martínez-Sánchez, J. J. G. Meilán, J. Carro, and O. Ivanova. A prototype for the voice analysis diagnosis of Alzheimer's disease. *Journal of Alzheimer's Disease*, 64:473–481, 2018.

[83] S. Merity, N. S. Keskar, and R. Socher. Regularizing and optimizing LSTM language models. *arXiv preprint arXiv:1708.02182*, 2017.

[84] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[85] A. Narayanan and D. L. Wang. Robust speech recognition from binary masks. *J. Acoust. Soc. Amer.*, 128:EL217–222, 2010.

[86] A. Narayanan and D. L. Wang. A CASA based system for long-term SNR estimation. *IEEE Trans. Audio, Speech, Lang. Process.*, 20:2518–2527, 2012.

[87] T. Nearey. *Phonetic Feature Systems for Vowels*. Indiana University Linguistics Club, 1978.

[88] A. Noll. Cepstrum pitch determination. *J. Acoust. Soc. Amer.*, 41:293–309, 1967.

[89] A. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[90] A. Pandey and D. L. Wang. A new framework for CNN-based speech enhancement in the time domain. *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, 27:1179–1188, 2019.

[91] R. Peharz, M. Wohlmayr, and F. Pernkopf. Gain-robust multi-pitch tracking using sparse nonnegative matrix factorization. In *Proc. ICASSP*, pages 5416–5419, 2011.

[92] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Proc. MICCAI*, pages 234–241, 2015.

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

[94] S. Semeniuta, A. Severyn, and E. Barth. Recurrent dropout without memory loss. *arXiv preprint arXiv:1603.05118*, 2016.

[95] F. Sha and L. K. Saul. Real-time pitch determination of one or more voices by nonnegative matrix factorization. In *Proc. NIPS*, pages 1233–1240, 2005.

[96] Z. Shi, H. Lin, L. Liu, R. Liu, and J. Han. FurcaNeXt: End-to-end monaural speech separation with dynamic gated dilated temporal convolutional networks. *arXiv preprint arXiv:1902.04891*, 2019.

[97] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur. X-vectors: Robust dnn embeddings for speaker recognition. In *Proc. ICASSP*, pages 5329–5333, 2018.

[98] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Proc. NIPS*, pages 3104–3112, 2014.

[99] N. Takahashi and Y. Mitsufuji. Multi-scale multi-band DenseNets for audio source separation. In *Proc. WASPAA*, pages 21–25, 2017.

[100] N. Takahashi, S. Parthasaarathy, N. Goswami, and Y. Mitsufuji. Recursive speech separation for unknown number of speakers. *arXiv preprint arXiv:1904.03065*, 2019.

[101] D. Talkin. A robust algorithm for pitch tracking (RAPT). In W. B. Kleijn and K. K. Palatal, editors, *Speech Coding and Synthesis*, pages 497–518. Elsevier Science Inc., 1995.

[102] A. Varga and H. J. M. Steeneken. Assessment for automatic speech recognition: II. NOISEX-92: A database and an experiment to study the effect of additive noise on speech recognition systems. *Speech Communication*, 12:247–251, 1993.

[103] E. Vincent, R. Gribonval, and C. Févotte. Performance measurement in blind audio source separation. *IEEE Trans. Audio, Speech, Lang. Process.*, 14:1462–1469, 2006.

[104] O. Vinyals, S. V. Ravuri, and D. Povey. Revisiting recurrent neural networks for robust ASR. In *Proc. ICASSP*, pages 4085–4088, 2012.

[105] D. Wang and J. H. L. Hansen. F0 estimation for noisy speech by exploring temporal harmonic structures in local time frequency spectrum segment. In *Proc. ICASSP*, pages 6510–6514, 2016.

[106] D. L. Wang. On ideal binary mask as the computational goal of auditory scene analysis. In P. Divenyi, editor, *Speech Separation by Humans and Machines*, chapter 12, pages 181–197. Kluwer Academic, Norwell MA, 2005.

[107] D. L. Wang and G. Brown, editors. *Computational Auditory Scene Analysis: Principles, Algorithms and Applications*. Wiley-IEEE Press, 2006.

[108] D. L. Wang and J. Chen. Supervised speech separation based on deep learning: An overview. *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, 26:1702–1726, 2018.

[109] Y. Wang, A. Narayanan, and D. L. Wang. On training targets for supervised speech separation. *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, 22:1849–1858, 2014.

[110] Y. Wang and D. L. Wang. Towards scaling up classification-based speech separation. *IEEE Trans. Audio, Speech, Lang. Process.*, 21:1381–1390, 2013.

[111] Z.-Q. Wang, J. Le Roux, and J. R. Hershey. Alternative objective functions for deep clustering. In *Proc. ICASSP*, pages 686–690, 2018.

[112] Z.-Q. Wang, J. Le Roux, D. L. Wang, and J. R. Hershey. End-to-end speech separation with unfolded iterative phase reconstruction. In *Proc. Interspeech*, pages 2708–2712, 2018.

[113] Z.-Q. Wang, K. Tan, and D. L. Wang. Deep learning based phase reconstruction for speaker separation: A trigonometric perspective. In *Proc. ICASSP*, pages 71–75, 2019.

[114] D. S. Williamson, Y. Wang, and D. L. Wang. Complex ratio masking for monaural speech separation. *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, 24:483–492, 2016.

[115] M. Wohlmayr and F. Pernkopf. EM-based gain adaptation for probabilistic multipitch tracking. In *Proc. Interspeech*, pages 1969–1972, 2011.

[116] M. Wohlmayr and F. Pernkopf. Model-based multiple pitch tracking using factorial HMMs: model adaptation and inference. *IEEE Trans. Audio, Speech, Lang. Process.*, 21:1742–1754, 2013.

[117] M. Wohlmayr, M. Stark, and F. Pernkopf. A probabilistic interaction model for multipitch tracking with factorial hidden Markov models. *IEEE Trans. Audio, Speech, Lang. Process.*, 19:799–810, 2011.

[118] A. Wrench. A multichannel/multispeaker articulatory database for continuous speech recognition research. *Phonus*, 5:1–13, 2000.

[119] M. Wu, D. L. Wang, and G. Brown. A multipitch tracking algorithm for noisy speech. *IEEE Trans. Speech Audio Process.*, 11:229–241, 2003.

[120] C. Xu, W. Rao, X. Xiao, E. S. Chng, and H. Li. Single channel speech separation with constrained utterance level permutation invariant training using grid LSTM. In *Proc. ICASSP*, pages 6–10, 2018.

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

[122] W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

[123] X. L. Zhang and D. L. Wang. A deep ensemble learning method for monaural speech separation. *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, 24:967–977, 2016.

[124] X Zhao, Y Shao, and D. L. Wang. CASA-based robust speaker identification. *IEEE Trans. Audio, Speech, Lang. Process.*, 20:1608–1616, 2012.

[125] X. Zhao, Y. Wang, and D. L. Wang. Cochannel speaker identification in anechoic and reverberant conditions. *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, 23:1727–1736, 2015.

[126] Y. Zhao, Z.-Q. Wang, and D. L. Wang. Two-stage deep learning for noisy-reverberant speech enhancement. *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, 27:53–62, 2019.