Deep Learning Algorithms for High-performance Automatic Speech Recognition

Tsubasa Ochiai

ID 4G151105

November 2017

Graduate School of Science and Engineering
Department of Information and Computer Science
Doshisha University
Kyoto, JAPAN

Thesis Committee:
Shigeru Katagiri, Chair
Xugang Lu
Tsuneo Kato
Masashi Okubo
Miho Ohsaki

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Acknowledgments

First, I express my deepest gratitude to Dr. Shigeru Katagiri, my dissertation advisor, for his continuous research guidance and valuable advice. Under his guidance, I have learned so many critical aspects about being a researcher and how to conduct basic research. Without his selfless support and excellent guidance, I could not have accomplished this work. Throughout the rest of my life, he will be my role model.

Next, I thank Dr. Shigeki Matsuda, Dr. Hideyuki Watanabe, and Dr. Xugang Lu for their valuable guidance, especially on the first part of my dissertation, as well as Dr. Shinji Watanabe, Dr. Takaaki Hori, and Dr. John R. Hershey who provided valuable guidance, especially for its latter part. I could not have achieved this work without their supervision. I learned a great deal about research (e.g., handling mathematical formulas, developing large-scale experiments, and creating attractive papers and presentations) through daily research with them.

Next, I appreciate the contributions of my supervisory committee members who provided multi-disciplinary coaching that broadened my research horizons: Dr. Miho Ohsaki, Dr. Tsuneo Kato, Dr. Seiichi Yamamoto, and Dr. Ivan Tanev.

I also thank Dr. Yutaka Kidawara, Dr. Hisashi Kawai, and Dr. Chiori Hori from the National Institute of Information and Communications Technology. In my role as cooperation researcher, they provided an excellent research environment for me. In addition, I received financial support from the Japan Society for the Promotion of Science as a Research Fellow (DC1). Their environmental and financial support greatly allowed me to concentrate on my research.

I also thank my friends and research colleagues whose contributions enhanced my

university experience.

Finally, I reserve my deepest appreciation for my family, especially my parents, who provided selfless support and enormous encouragement in all aspects of my life. My long research journey would not have been possible without their love and understanding.

November 24, 2017

Abstract

Speech is one of the most natural communication modalities for humans. Since the advent of computers, the development of natural speech communication channels between humans and computers has been one of the greatest goals in computer science research fields. For such natural communication channels, automatic speech recognition (ASR) technology, which converts speech into text by computer programs, has been scrutinized and comprises the core of such intelligent and user-friendly applications as voice dictation, voice search, voice command, and spoken dialogue systems.

Over the past several decades, machine learning (ML)-based approaches have been a center pillar of ASR research, based on the advancement of such key ML methodologies as (artificial) neural networks and probabilistic modeling. With the recent advent of deep learning (DL) techniques, ASR performances have significantly improved in the past five years or so. However, such DL-based ASR technologies remain insufficient for appropriately coping with the variability of speakers and speaking environments; there are obvious needs for further improving ASR technologies.

The most fundamental way for coping with the variability is to fully represent it in training stages for ASR systems. However, it is basically unrealistic to predict the entire variability. Accordingly, incorporating some effective compensation techniques with DL-based ASR systems is a reasonable solution to the variability problem. Motivated by this understanding, in this dissertation, we investigate novel compensation techniques for deep neural network (DNN)-based ASR systems by introducing an internal structure to DNN, which enables ASR systems to evoke the aptitude of DNN for dealing with speaker and/or environment variability.

In this dissertation, we separately study two kinds of variability: 1) variability in speakers, and 2) variability in speaking environments (changes in background noise and reverberation). For the former issue, we propose a novel speaker adaptation algorithm that incorporates the speaker adaptive training (SAT) concept into the training of DNN in the framework of a hybrid DNN and Hidden Markov Model (HMM) speech recognizer. Our proposed SAT-based speaker adaptation scheme introduces modularity, more precisely, localizing a speaker dependent (SD) module, in the DNN part of the hybrid system and optimizes the DNN part, assuming that the SD module is adapted in the adaptation stage. For the latter environment-related issue, we focus on a recently proposed end-to-end ASR architecture, which is completely composed of neural networks, and propose a novel multichannel end-to-end (ME2E) ASR architecture that integrates speech enhancement and speech recognition components into a single neural network-based architecture. Our proposed architecture allows the overall procedure of multichannel speech recognition (i.e., from speech enhancement to speech recognition) to be optimized only under a recognition-oriented training (learning) criterion using multichannel noisy speech samples and their corresponding transcriptions.

We conducted several experimental evaluations of our proposed methods and successfully demonstrated their effectiveness in further increasing the performances of DNN-based ASR systems. The proposed SAT-based speaker adaptation methods successfully increased the adaptability of DNN-HMM hybrid systems and reduced the size of the adaptable parameters. The proposed ME2E ASR architecture successfully learned a noise suppression function through end-to-end recognition-oriented optimization and improved ASR performances in various noisy environments.

Contents

List of Figures

List of Tables

1 Introduction

1.1 General Background

Speech is one of the most natural mediums through which humans communicate. For such natural communication between humans and computers, automatic speech recognition (ASR) technology, which converts speech into text by computer programs, has been continuously and vigorously investigated in the computer science research field. Moreover, based on the recent dramatic advancement of computer and telecommunication technologies, computers are now being widely used in daily life. The expectations for the development of high-quality ASR technology continue to increase.

Due to the progress of machine learning (ML) and probabilistic modeling technologies, ML-based approaches have become the mainstream of recent ASR research and enable ASR systems to automatically learn mapping functions that transfer speech to text, based on training sample pairs, each of which consists of a speech sample and its corresponding transcription. Over the last decade, deep learning (DL) techniques [1] have achieved remarkable successes in such tasks as computer vision [2, 3, 4], machine translation [5, 6, 7], and ASR [8, 9, 10]. However, regardless of the large success of DL-based approaches, the performances of DL-based ASR systems remain insufficient, mainly because they have failed to cope with the high variability of speakers and/or speaking environments (changes in noise and reverberation).

1.2 Contributions

The objective of this dissertation is to develop methods for further increasing DL-based ASR systems. With the recent advent of DL techniques, we focus on two types of DL-based ASR frameworks: 1) a hidden Markov model (HMM)-based hybrid framework and 2) an end-to-end framework. We also investigate these two frameworks by focusing on two major factors in decreasing ASR performances: speaker and speaking environment variability. Our main idea for solving the variability problems is to incorporate modularity into the DNN structure, which embeds prior knowledge to the ASR architecture and evokes the aptitude of DNN for dealing with such variabilities.

1.2.1 Speaker Variability: Speaker Adaptive Training for Deep Neural Networks

To achieve high recognition performances, ASR systems must effectively cope with the speaker variability derived from differences in speaker characteristics. The differences of such speaker characteristics substantially change the speech waveforms whose linguistic content is identical (such as words) and generally degrades ASR performances. In real world applications, ASR systems must deal with the speech inputs spoken by myriad unknown speakers, which are not included in training speakers. Therefore, for higher ASR performances, the speaker variability problem must be solved. As one solution to it, speaker adaptation techniques have been studied because they compensate the mismatch between the training speakers used for training ASR systems and unknown testing speakers by adapting ASR systems using the target speaker's speech data.

In parallel with the advancement of speaker adaptation techniques, speech recognizers, which have long been constructed by Gaussian Mixture Models (GMMs) and HMMs [11, 12], are welcoming a new hybrid structure of Deep Neural Networks (DNNs) and HMMs [8, 9, 10]. However, despite the high utility demonstrated by DNN in various tasks, the hybrid DNN-HMM

has not yet completely solved the sample finiteness problem in speaker adaptation frameworks, i.e., insufficient adaptation to unseen speakers.

In light of the above backgrounds, our dissertation proposes a novel speaker adaptation algorithm that incorporates the speaker adaptive training (SAT) concept into DNN training. The proposed SAT-based speaker adaptation scheme introduces modularity (more precisely, localizing a speaker dependent (SD) module) in the DNN and optimizes the DNN on the premise that the SD module is adapted in the adaptation stage, which increases the DNN's adaptability.

1.2.2 Environment Variability: Multichannel End-to-end Speech Recognition

To achieve high recognition performance, we must also cope with various types of environment variability, including noise and reverberation. In real world applications, speech inputs for ASR systems are generally contaminated by background noise and reverberation. The existence of such interferences changes the speech waveforms and substantially degrades the ASR performance. Therefore, to compensate the speech contamination, a speech enhancement (noise suppression) mechanism must be incorporated in ASR architecture.

Over the last decade with the advent of DL techniques, a hybrid DNN-HMM framework [8, 9, 10] has become a standard approach for ASR tasks. In conjunction, significant interest exists in developing fully end-to-end DL architectures, such as attention-based encoder-decoder networks [13, 14] and connectionist temporal classification (CTC) systems [15, 16]. The benefits of such approaches include 1) structural simplicity (the entire procedure from input to output consists of a monolithic neural network-based architecture) and 2) optimization consistency (the entire system is optimized with a single ASR-level objective). However, previous research on end-to-end frameworks mainly focused on the ASR problem in a single-channel setup without speech enhancement. Considering real world applications, studying such frameworks in a multichannel setup with speech enhancement is a critical research direction.

In light of the above trend, this dissertation focuses on recently proposed end-to-end ASR frameworks and proposes a novel multichannel end-to-end (ME2E) ASR architecture that integrates the speech enhancement and speech recognition components into a single neural network-based architecture. Our proposed architecture optimizes the overall inference in multichannel speech recognition (i.e., from speech enhancement to speech recognition) based on the end-to-end ASR objective only using multichannel noisy speech samples and their corresponding transcription and leads to an end-to-end framework that works well in the presence of strong background noise.

1.3 Outline

The rest of this dissertation is organized as follows:

■ Chapter 2–Preliminaries

This chapter describes the basic preliminaries of the DL and ASR technologies. After reviewing the structure of DNNs and their training procedure, we describe an overview of recent ASR frameworks that utilize DL techniques.

■ Chapter 3–Speaker Adaptive Training for Deep Neural Networks

This chapter deals with a speaker adaptation problem for the hybrid DNN-HMM framework. We propose a SAT-based speaker adaptation scheme for the hybrid DNN-HMM speech recognizer. The effectiveness of the proposed methods was experimentally validated. Part of this research was presented in the following publications [17, 18, 19, 20]:

- **Tsubasa Ochiai**, Shigeki Matsuda, Hideyuki Watanabe, Xugang Lu, Chiori Hori, Hisashi Kawai, and Shigeru Katagiri, "Speaker Adaptive Training Localizing Speaker Modules in DNN for Hybrid DNN-HMM Speech Recognizers," *IEICE Transactions on Information and Systems*, vol. E99-D, no. 10, pp. 2431-2443, 2016.
- **Tsubasa Ochiai**, Shigeki Matsuda, Xugang Lu, Chiori Hori, and Shigeru Katagiri, "Speaker adaptive training using deep neural networks," *IEEE International Conference on Acous-*

tics, Speech and Signal Processing (ICASSP), pp. 6349-6353, 2014.

– **Tsubasa Ochiai**, Shigeki Matsuda, Hideyuki Watanabe, Xugang Lu, Chiori Hori, and Shigeru Katagiri, "Speaker adaptive training using deep neural networks embedding linear transformation networks," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4605-4609, 2015.

– **Tsubasa Ochiai**, Shigeki Matsuda, Hideyuki Watanabe, Xugang Lu, Hisashi Kawai, and Shigeru Katagiri, "Bottleneck linear transformation network adaptation for speaker adaptation for speaker adaptive training-based hybrid DNN-HMM speech recognizer," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5015-5019, 2016.

■ Chapter 4–Multichannel End-to-end Architecture for Automatic Speech Recognition

This chapter focuses on a noisy ASR problem for the end-to-end ASR framework. We extended an existing attention-based encoder-decoder framework by integrating a neural beamformer and proposed a unified architecture of an ME2E ASR. The effectiveness of the proposed architecture was also experimentally validated. Part of this research was presented in the following publications [21, 22, 23]:

– **Tsubasa Ochiai**, Shinji Watanabe, Takaaki Hori, John R. Hershey, and Xiong Xiao, "A Unified Architecture for Multichannel End-to-End Speech Recognition with Neural Beamforming," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1274-1288, 2017.

– **Tsubasa Ochiai**, Shinji Watanabe, Takaaki Hori, and John R. Hershey, "Multichannel End-to-end Speech Recognition," *International Conference on Machine Learning (ICML)*, pp. 2632-2641, 2017.

– **Tsubasa Ochiai**, Shinji Watanabe, and Shigeru Katagiri, "Does Speech Enhancement Work with End-To-End ASR Objectives?: Experimental Analysis Of Multichannel End-To-End ASR," *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2017.

■ **Chapter 5–Conclusion**

This chapter summarizes the main contributions of this dissertation and discusses future research directions.

2 Preliminaries

2.1 Deep Leaning Techniques

2.1.1 Deep Neural Networks

DNN is a multilayer perceptron (MLP) with multiple (deep) intermediate layers, which can be used as a general function approximator. Given input feature vector $\boldsymbol{x}$, it repeats the feature transformation in the intermediate layers and finally produces the network output vector $\boldsymbol{y}$. The multilayer structure enables the network to learn multilevel feature representations, which provides high feature representation capability to DNN. Although DNN can be used for both classification and regression tasks, this section denotes the DNN's formalization focusing on the classification task.

Figure 2.1 illustrate the overview of the DNN structure. The adopted DNN is a standard MLP network whose node has trainable connection weights and a trainable bias. We denote the weight matrix and the bias vector between network layers L_{l-1} and L_l as $\boldsymbol{W}_l$ and $\boldsymbol{b}_l$. We also denote the pair of $\boldsymbol{W}_l$ and $\boldsymbol{b}_l$ as $\boldsymbol{\lambda}_l = \{\boldsymbol{W}_l, \boldsymbol{b}_l\}$. The example DNN in the figure has seven layers $\{\mathrm{L}_0, \mathrm{L}_1, \cdots, \mathrm{L}_6\}$. For simplicity, we omit the biases in the figure.

Let $\boldsymbol{y}$ be the network output vector defined as $\boldsymbol{y} = \{y_k; k = 1, 2, \cdots, K\}$, where k corresponds to k-th class label and K is the number of classes. Generally, the computation procedure of DNN

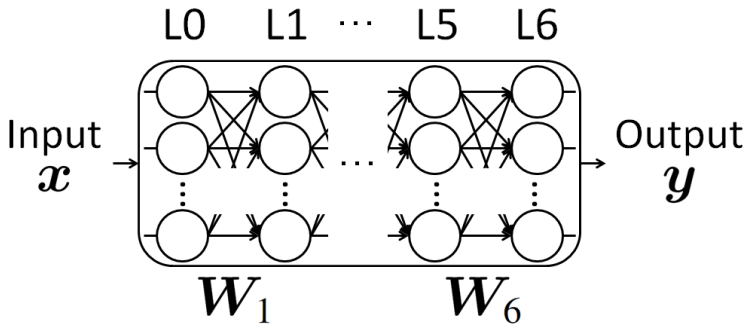


Figure 2.1: DNN structure.

are formalized as follows:

$$\boldsymbol{h}_1 = \phi^h(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1), \tag{2.1}$$

$$\boldsymbol{h}_l = \phi^h(\boldsymbol{W}_l\boldsymbol{h}_{l-1} + \boldsymbol{b}_l) \quad (l = 2, 3, \cdots, L-1), \tag{2.2}$$

$$\boldsymbol{y} = \phi^o(\boldsymbol{W}_L\boldsymbol{h}_{L-1} + \boldsymbol{b}_L), \tag{2.3}$$

where $\boldsymbol{h}_l$ is the outputs from l-th intermediate layer, L is the number of transformation layers, ϕ^h is an activation function for intermediate layers (e.g. sigmoid function), and ϕ^o is an activation function for output layer (e.g. softmax function). When we use softmax function for the activation function, the posterior probability $P(\boldsymbol{y}|\boldsymbol{x})$ can be estimated as the network output.

2.1.2 Training Procedure

We assume that training samples $\mathcal{X} = \{\{\boldsymbol{x}^n, \boldsymbol{t}^n\}; i = 1, \cdots, N\}$ are available to train the DNN, where $\boldsymbol{x}^n$ is the n-th training feature vector, $\boldsymbol{t}^n = \{t_k^n; k = 1, 2, \cdots, K\}$ is its corresponding target label, and N is the number of such samples. Given training feature vector $\boldsymbol{x}^n$ to input layer, DNN emits network outputs $\boldsymbol{y}^n = \{y_k^n; k = 1, 2, \cdots, K\}$ at output layer.

Let $\boldsymbol{\Lambda} = \{\boldsymbol{\lambda}_l; l = 1, \cdots, L\}$ be a total set of trainable model parameters. When we assume that Cross-Entropy (CE) error, which is a standard loss function for DNN-based classifier, is adopted

for the objective function, DNN training is formalized as the following minimization problem:

$$\overline{\mathbf{\Lambda}} = \arg\min_{\mathbf{\Lambda}} E_{\mathrm{CE}}(\mathbf{\Lambda}; \mathcal{X}), \tag{2.4}$$

where E_{CE} is the accumulated CE error defined as:

$$E_{\mathrm{CE}}(\mathbf{\Lambda}; \mathcal{X}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_k^n \log y_k^n. \tag{2.5}$$

Here, t_k^n is the teaching signal for y_k^n that indicates 1 when $\boldsymbol{x}^n$ belongs to class k but indicates 0 otherwise.

Because the computation procedure of DNN is fully differentiable, the above minimization problem can be solved using the arbitrary gradient-based optimization algorithms. Especially, Error Back Propagation (EBP) method [24] is commonly adopted to optimize the objective, where the optimization is conducted by the following iterative parameter update:

$$\mathbf{\Lambda} \leftarrow \mathbf{\Lambda} - \epsilon \frac{\partial E_{\mathrm{CE}}(\mathbf{\Lambda}; \mathcal{X})}{\partial \mathbf{\Lambda}}, \tag{2.6}$$

where ϵ is the positive scalar training rate.

In practice, *mini-batch* mode minimization are commonly adopted for DNN's optimization, which was a mix of the batch and sequential modes that repeated the error calculation and parameter updates over every set of some selected training samples.

2.2 Automatic Speech Recognition

2.2.1 Fundamentals

ASR is a sequence-to-sequence mapping problem, which maps acoustic signal sequence $\boldsymbol{X}$ to a linguistic symbol sequence $\mathcal{W}$. Mathematically, ASR is formalized as a task to search the

optimal word (character) sequence $\overline{\mathcal{W}}$ that maximizes $P(\mathcal{W}|\boldsymbol{X})$, i.e., the posterior probability of the word sequence $\mathcal{W}$ given the speech observation $\boldsymbol{X}$, as follows:

$$\overline{\mathcal{W}} = \arg\max_{\mathcal{W}} P(\mathcal{W}|\boldsymbol{X}). \tag{2.7}$$

In the recent ASR field, there exist two types of major ASR frameworks: 1) HMM-based hybrid framework and 2) end-to-end framework. The main difference between them is how to model the posterior probability $P(\mathcal{W}|\boldsymbol{X})$. The former HMM-based hybrid framework models $P(\mathcal{W}|\boldsymbol{X})$ using several separate modules based on the Bayes rule. On the other hand, the latter end-to-end framework directly models $P(\mathcal{W}|\boldsymbol{X})$ using the fully neural network-based architecture.

In the following subsection, we describe the overview of the HMM-based hybrid framework. Since the end-to-end framework is integrated into our proposed ME2E ASR framework (Chapter 4), we minutely describe it in Section 4.5.

2.2.2 HMM-based Hybrid Frameworks

Overview

In the HMM-based hybrid framework, based on the Bayes rule, we further derive the posterior probability $P(\mathcal{W}|\boldsymbol{X})$ as follows:

$$P(\mathcal{W}|\boldsymbol{X}) = \frac{P(\boldsymbol{X}|\mathcal{W})P(\mathcal{W})}{P(\boldsymbol{X})}. \tag{2.8}$$

Given the speech observation $\boldsymbol{X}$, the prior probability $P(\boldsymbol{X})$ is constant with respect to the word sequence $\mathcal{W}$. Therefore, the ASR task as in Eq. (2.7) is re-formalized as follows:

$$\overline{\mathcal{W}} = \arg\max_{\mathcal{W}} P(\mathcal{W}|\boldsymbol{X}) = \arg\max_{\mathcal{W}} P(\boldsymbol{X}|\mathcal{W})P(\mathcal{W}). \tag{2.9}$$

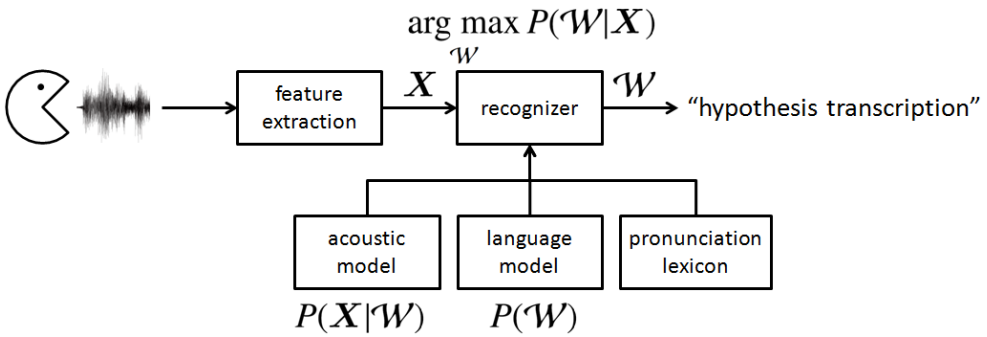


Figure 2.2: Overview of hybrid DNN-HMM framework.

The HMM-based hybrid framework separately models both probabilities $P(\boldsymbol{X}|\mathcal{W})$ and $P(\mathcal{W})$. Here, $P(\boldsymbol{X}|\mathcal{W})$ represents the likelihood of the speech observation $\boldsymbol{X}$ given the word sequence $\mathcal{W}$, which is estimated by an acoustic model. On the other hand, $P(\mathcal{W})$ represents the prior probability of the word sequence $\mathcal{W}$, which is estimated by such language models as N-gram [25, 26] and Recurrent Neural Network (RNN) language model [27, 28]. Figure 2.2 illustrates the overview of the hybrid DNN-HMM framework, where a pronunciation lexicon represents the mapping from the words to the phoneme sequence and is used to link the acoustic and language models.

Since Chapter 3 studies the speaker adaptation techniques for the DNN in the acoustic model, we focus on the acoustic modeling part of the hybrid DNN-HMM framework in the remaining of this section.

DNN-based acoustic model

Figure 2.3 illustrates the structure of the acoustic model part in the hybrid DNN-HMM framework. In this hybrid structure, the GMM part of the conventional GMM-HMM framework, which is used to estimate state output probabilities in the HMM, is replaced by DNN.

Given a speech input, the system first converts it to a sequence of acoustic feature vectors $\boldsymbol{X} = \{\boldsymbol{x}_\tau; \tau = 1, \cdots, \mathcal{T}\}$, where $\boldsymbol{x}_\tau$ is the τ-th acoustic feature vector and $\mathcal{T}$ is the sequence length. Next, the system estimates posterior probability $p(\mathcal{W}|\boldsymbol{X})$ for the possible word sequences $\mathcal{W}$

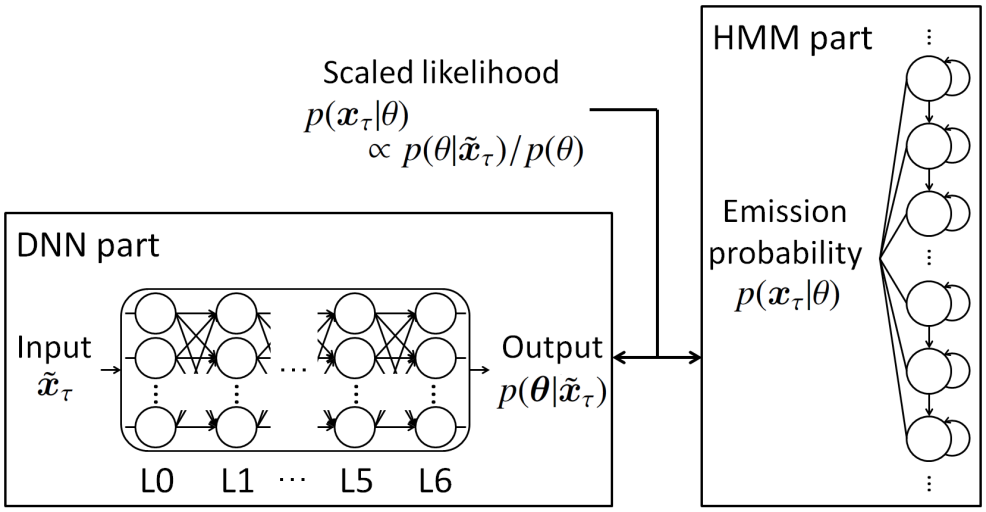


Figure 2.3: Structure of acoustic model part in hybrid DNN-HMM framework.

and classifies $\boldsymbol{X}$ to the word sequence with the largest posterior probability value $\overline{\mathcal{W}}$. Clearly, the system's classification accuracy depends on the estimation quality for $p(\mathcal{W}|\boldsymbol{X})$.

In the DNN-HMM framework, $p(\boldsymbol{X}|\mathcal{W})$ is estimated by $p(\boldsymbol{X}|\mathcal{W})_{\{\boldsymbol{\Lambda}_{\text{DNN}},\boldsymbol{\Lambda}_{\text{HMM}}\}}$, which is a function of $\boldsymbol{\Lambda}_{\text{DNN}}$ (trainable parameters of DNN) and $\boldsymbol{\Lambda}_{\text{HMM}}$ (trainable parameters of HMM). Accordingly, the training seeks the state of $p(\boldsymbol{X}|\mathcal{W})_{\{\boldsymbol{\Lambda}_{\text{DNN}},\boldsymbol{\Lambda}_{\text{HMM}}\}}$ that achieves the highest possible classification accuracies on testing speech data by updating $\boldsymbol{\Lambda}_{\text{DNN}}$ and $\boldsymbol{\Lambda}_{\text{HMM}}$.

$p(\boldsymbol{X}|\mathcal{W})_{\{\boldsymbol{\Lambda}_{\text{DNN}},\boldsymbol{\Lambda}_{\text{HMM}}\}}$ is calculated using such HMM probabilities as state output probability estimates $\{p(\boldsymbol{x}_\tau|\theta)_{\{\boldsymbol{\Lambda}_{\text{DNN}},\boldsymbol{\Lambda}_{\text{HMM}}\}}; \tau = 1, \cdots, \mathcal{T}$, and $\theta = 1, \cdots, \Theta\}$, where Θ is the number of possible states of HMM. Here, again based on the Bayes theorem, $p(\boldsymbol{x}_\tau|\theta)_{\{\boldsymbol{\Lambda}_{\text{DNN}},\boldsymbol{\Lambda}_{\text{HMM}}\}}$ is replaced by scaled likelihood $p(\theta|\boldsymbol{x}_\tau)_{\boldsymbol{\Lambda}_{\text{DNN}}}/p(\theta)_{\boldsymbol{\Lambda}_{\text{HMM}}}$. State posterior probability $p(\theta|\boldsymbol{x}_\tau)_{\boldsymbol{\Lambda}_{\text{DNN}}}$ is estimated by DNN, and state prior probability $p(\theta)_{\boldsymbol{\Lambda}_{\text{HMM}}}$ is estimated based on the frequency of the state assignment produced by HMM's forced alignment. Because $p(\theta|\boldsymbol{x}_\tau)_{\boldsymbol{\Lambda}_{\text{DNN}}}$ must maintain the nature of the probability function, the DNN part uses the softmax activation function at its output nodes.

Recent hybrid DNN-HMM speech recognizers usually adopt context dependent acoustic models. In this situation, the number of HMM states is too large to appropriately estimate state posterior probability $p(\theta|\boldsymbol{x}_\tau)_{\boldsymbol{\Lambda}_{\text{DNN}}}$. To circumvent this problem, the HMM states are often clustered into

several thousands of sub-phonetic units, i.e., *senones*, each representing the HMM tied-state [29]. Following this strategy, in the DNN-HMM framework, the estimate of state posterior probability $p(\theta|\boldsymbol{x}_\tau)_{\boldsymbol{\Lambda}_{\mathrm{DNN}}}$ is replaced with the estimate of senone posterior probability $p(k|\boldsymbol{x}_\tau)_{\boldsymbol{\Lambda}_{\mathrm{DNN}}}$, where k is the senone class index ($k = 1, \cdots, K$), assuming that K is the number of senones.

In most cases of the recent hybrid DNN-HMM speech recognizer, the concatenation of several acoustic feature vectors $\tilde{\boldsymbol{x}}_\tau = \{\boldsymbol{x}_{\tau-\tau_c}, \cdots, \boldsymbol{x}_\tau, \cdots, \boldsymbol{x}_{\tau+\tau_c}\}$ is used in $p(\theta|\boldsymbol{x}_\tau)_{\boldsymbol{\Lambda}_{\mathrm{DNN}}}$ instead of $\boldsymbol{x}_\tau$, where τ_c is a small natural number. Accordingly, $p(k|\boldsymbol{x}_\tau)_{\boldsymbol{\Lambda}_{\mathrm{DNN}}}$ is further replaced by $p(k|\tilde{\boldsymbol{x}}_\tau)_{\boldsymbol{\Lambda}_{\mathrm{DNN}}}$. DNN has enough potential to deal with such large-dimensional features, which includes rich information within a relatively long context window. This point is one of the most important factors that the hybrid DNN-HMM framework performs better than the conventional GMM-HMM framework [30].

Training procedure

The training procedure is twofold: one for the HMM part and another for the DNN part. The HMM part is first trained within the training for the GMM-HMM system. The DNN part is subsequently trained using the senone labels produced by the forced alignment with the baseline GMM-HMM system. These labels are used as teaching signals to train the acoustic feature vector inputs. Using these labels, such an objective function as CE error is defined, and the DNN parameters are optimized under a condition that minimizes the defined objective function, as described in Section 2.1.2.

3 Speaker Adaptive Training for Deep Neural Networks

3.1 Introduction

Unavoidably, the development of pattern recognizers has to cope with the training sample finiteness problem. The recognizers are trained using a finite amount of training samples in hand but must accurately work over (practically infinite) unseen testing samples. In the speech recognition field, this finiteness problem has been particularly investigated in the speaker adaptation framework [31, 32, 33]. Assuming the problem's existence, speech recognizers are often trained in a speaker independent (SI) mode and adapted to the unseen testing samples of new speakers.

In many speaker adaptation scenarios, only a limited amount of speech samples are available. Since the limitation of training samples makes it difficult to adapt the entire recognizer, usually just some part of it is adapted. In this partial adaptation scheme, SI recognizers are not necessarily the best choice for the initial status for the adaptation. SI training does not assume that part of the trained recognizer will be replaced in the later adaptation stage. As one solution to this inadequacy, SAT was proposed [34, 35, 36] for the conventional GMM-HMM framework. If we assume that some part of the recognizer will be replaced later, the remainder should be trained from the start, based on the assumption of such a replacement. Following this understanding, SAT jointly trains the speaker-oriented part of the recognizer and its remainder on the premise that the speaker-oriented part will be replaced in the adaptation stage.

In parallel with the advancement of speaker adaptation technologies, a hybrid DNN-HMM is

consolidating its position as a new standard for speech recognizer structure [8, 9, 10]. However, despite the high utility demonstrated by DNN-HMM recognizers, it is unrealistic to collect all of the possible speech data, which are spoken by different speakers, and develop an almighty recognizer that can recognize speech data correctly in a fully SI mode. Accordingly, adaptation technology is still a key research issue even for the powerful hybrid DNN-HMM recognizers.

Motivated by the high discriminative power of the hybrid DNN-HMM and the advantages of the SAT concept, we propose a new SAT-based speaker adaptation scheme for the hybrid DNN-HMM speech recognizer, which is characterized by introducing modularity, more precisely, localizing an SD module, in the DNN part.

In Section 3.2, we describe the basic formalization of the proposed SAT-based speaker adaptation scheme. The network weight matrix and bias vector of one layer in the DNN is treated as the *SD module* and the DNN remainder is treated as the *SI module*. Based on the SAT concept, multiple SD modules for training speakers and the SI module are jointly trained over the training speech data of many speakers, the trained SD modules are replaced by a new SD module for a target speaker, and only the new SD module is adapted using the speech data of the target speaker.

In Section 3.3, we extend the preceding SAT-based speaker adaptation scheme by embedding Linear Transformation Network (LTN). The linearity introduced by the LTN SD module enables to perform the SAT and speaker adaptation steps using a more appropriate anchorage state of network parameters in the regularization.

In Section 3.4, we theoretically analyze the relation between the number of adaptable parameters and their feature-representation capability in the LTN SD module, and further extend the preceding SAT-based speaker adaptation scheme by introducing a bottleneck structure, which enables to reduce the size of LTN SD module while maintaining its feature-representation power.

The notations used in this chapter are summarized in Appendix A.1.

3.2 Speaker Adaptive Training Localizing Speaker Modules in DNN for Hybrid DNN-HMM Speech Recognizers

3.2.1 Overview

The SAT-based adaptation scheme consists of the following three stages: 1) initialization, 2) SAT, and 3) speaker adaptation. All of the training procedures in these stages adopt the criterion of minimizing the CE error function between the DNN outputs and the senone labels that are produced by the forced alignment with the baseline GMM-HMM speech recognizer. In the initialization stage, we train the whole DNN part in the standard SI discriminative training fashion. In the SAT stage, we localize the SD modules in the DNN part and train the entire DNN while switching the SD modules along with the speaker change in the training data set. Finally, in the speaker adaptation stage, we train only a new SD module for a target speaker using his/her speech data.

3.2.2 Training Procedures

Initialization Stage

We assume that the training samples $\mathcal{X} = \{\{\boldsymbol{X}^n, \boldsymbol{T}^n\}; n = 1, \cdots, N\}$ are available to train the DNN part, where $\boldsymbol{X}^n = \{\boldsymbol{x}_\tau^n; \tau = 1, \cdots, \mathcal{T}\}$ is the n-th training sample, $\boldsymbol{T}_n = \{\boldsymbol{t}_\tau^n; \tau = 1, \cdots, \mathcal{T}\}$ is its corresponding target label, and N is the number of such samples. The speech samples of $\mathcal{X}$ are spoken by many speakers.

In Figure 3.1, we illustrate our initialization procedure for a seven-layer example of a DNN part. For simplicity, we omit the biases in all of the illustrations in this chapter. Given input feature vector $\boldsymbol{x}_\tau^n$ of training sample $\boldsymbol{X}^n$ to input layer L_0, the DNN part emits network outputs $\boldsymbol{y}_\tau^n = \{y_{\tau k}^n; k = 1, 2, \cdots, K\}$ at output layer L_6. The largest output represents the senone classification decision, which is evaluated using the correct class information determined by the forced

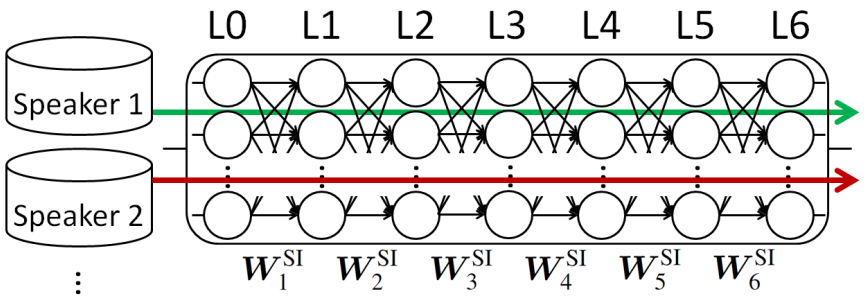


Figure 3.1: DNN structure and SI training procedure for initialization stage.

alignment with the baseline GMM-HMM speech recognizer.

For discussion generality, we consider the initialization of L-layer DNN parameters $\mathbf{\Lambda}_{\rm SI} = \{\boldsymbol{\lambda}_l^{\rm SI}; l = 1, 2, \cdots, L\}$, where $\boldsymbol{\lambda}_l^{\rm SI} = \{\boldsymbol{W}_l^{\rm SI}, \boldsymbol{b}_l^{\rm SI}\}$. To accelerate the initialization training, we preliminarily train $\mathbf{\Lambda}_{\rm SI}$ using the Restricted Boltzmann Machine (RBM) [37] in the greedy layer-wise manner [38]. For later discussions, we denote the RBM-trained state of $\mathbf{\Lambda}_{\rm SI}$ as $\mathbf{\Lambda}_{\rm RBM}$. Next, we conduct the following regularization-incorporated CE error minimization:

$$\overline{\mathbf{\Lambda}}_{\rm SI} = \arg\min_{\mathbf{\Lambda}_{\rm SI}} \left\{ E_{\rm CE}(\mathbf{\Lambda}_{\rm SI}; \mathcal{X}) + \frac{\rho_{\rm SI}}{2} R(\mathbf{\Lambda}_{\rm SI}) \right\}, \tag{3.1}$$

where $E_{\rm CE}$ is the accumulated CE error defined as

$$E_{\rm CE}(\mathbf{\Lambda}_{\rm SI}; \mathcal{X}) = -\sum_{n=1}^{N} \sum_{\tau=1}^{\mathcal{T}} \sum_{k=1}^{K} t_{\tau k}^n \log y_{\tau k}^n. \tag{3.2}$$

Here, $t_{\tau k}^n$ is the teaching signal for $y_{\tau k}^n$ that indicates 1 when $\boldsymbol{x}_\tau^n$ belongs to senone class k but indicates 0 otherwise, R is a regularization term, and $\rho_{\rm SI}$ is a non-negative constant regularization coefficient. In this minimization, $\mathbf{\Lambda}_{\rm RBM}$ works as the initial status of $\mathbf{\Lambda}_{\rm SI}$.

We adopt the regularization to avoid the over-training problem that is often caused by large-size DNNs. The regularization procedure will be described in Section 3.2.2. The minimization in Eq. (3.1) is conducted based on the EBP parameter update rule.

We use the above estimated parameters $\overline{\mathbf{\Lambda}}_{\rm SI}$ for a baseline SI recognizer and also as an initial status of the subsequent SAT stage.

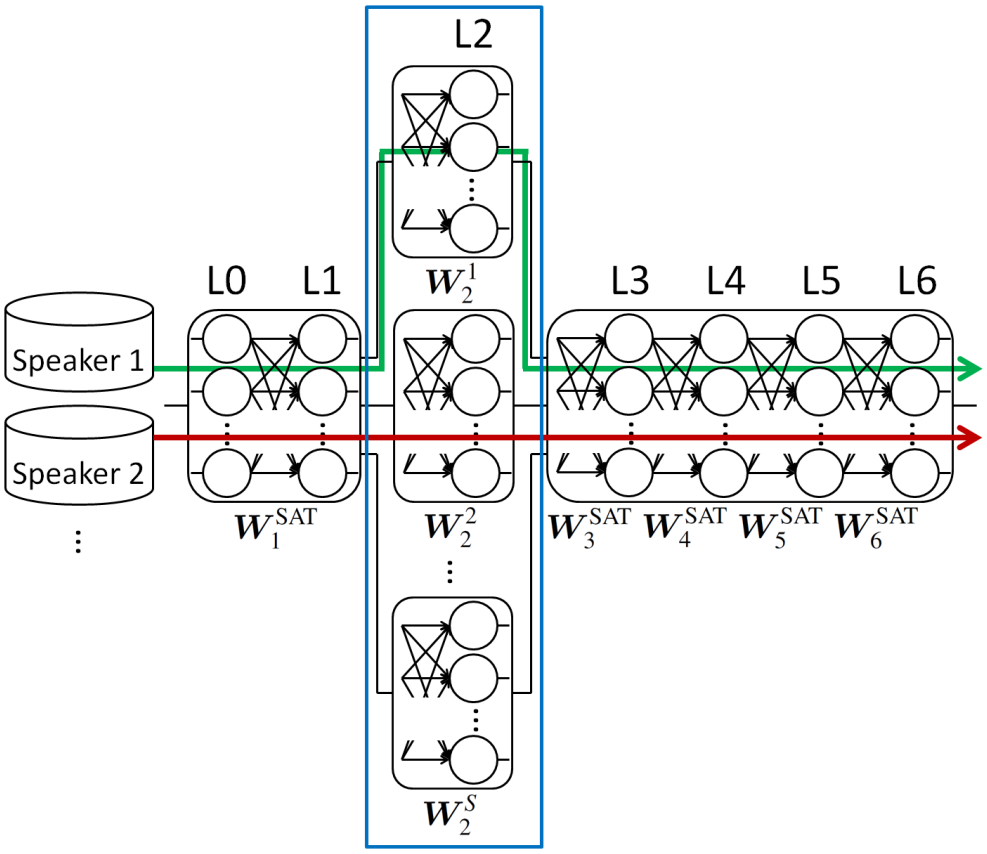


Figure 3.2: DNN structure and SAT training procedure for SAT stage.

Speaker Adaptive Training Stage

In Figure 3.2, we illustrate the procedure of conducting SAT with SD module allocation, where the blue rectangle represents the SD module layer[1]. Because of DNN's multilayer structure, the SD modules can be allocated to any of the intermediate layers. In the figure, as an example, we allocate SD modules $\boldsymbol{G}_2 = \{\boldsymbol{g}_2^s; s = 1, \cdots, S\}$ to L_2, where $\boldsymbol{g}_2^s = \{\boldsymbol{W}_2^s, \boldsymbol{b}_2^s\}$ is the SD module parameters for training speaker s, S is the number of training speakers, and $\boldsymbol{\Lambda}_{\mathrm{SAT}} = \{\boldsymbol{\lambda}_l^{\mathrm{SAT}}; l = 1, 3, \cdots, L\}$ are the parameters of the network layers other than the SD module layer, where $\boldsymbol{\lambda}_l^{\mathrm{SAT}} = \{\boldsymbol{W}_l^{\mathrm{SAT}}, \boldsymbol{b}_l^{\mathrm{SAT}}\}$. For descriptive purposes, we refer to the network layers other than the SD module layer as an SI module layer. Note that the network parameters of the SD module layer are prepared for each training speaker, while those of the SI module layer are shared among all

[1]This is common in all of the illustrations in this chapter.

training speakers.

The figure shows two example cases of the training procedure: one for using the speech data of Speaker 1 ($s = 1$) and another for Speaker 2 ($s = 2$). When using the data of Speaker 1, only the nodes of the SD module for Speaker 1 are connected with the nodes of the adjacent layers; the nodes of the other SD modules are disconnected from the nodes in the adjacent layers. The green line depicts this situation, and the training is executed only along this path. Similarly, the red line depicts the situation when using the data of Speaker 2. Each SD module is trained only using its corresponding speaker's data, but the other part of the network, i.e., $\mathbf{\Lambda}_{\mathrm{SAT}}$, is trained using the data of all of the S speakers.

Again for discussion generality, we consider the case of using SD module layer $\mathrm{L}_{l_{\mathrm{SD}}}$ of the L-layer DNN. The SAT procedure for this general setting is formalized as follows:

$$(\overline{\mathbf{\Lambda}}_{\mathrm{SAT}}, \overline{\boldsymbol{G}}_{l_{\mathrm{SD}}}) = \underset{(\mathbf{\Lambda}_{\mathrm{SAT}}, \boldsymbol{G}_{l_{\mathrm{SD}}})}{\arg\min} \left\{ E_{\text{SAT-CE}}(\mathbf{\Lambda}_{\mathrm{SAT}}, \boldsymbol{G}_{l_{\mathrm{SD}}}; \mathcal{X}) + \frac{\rho_{\mathrm{SAT}}}{2} R(\boldsymbol{G}_{l_{\mathrm{SD}}}) \right\}, \tag{3.3}$$

where

$$E_{\text{SAT-CE}}(\mathbf{\Lambda}_{\mathrm{SAT}}, \boldsymbol{G}_{l_{\mathrm{SD}}}; \mathcal{X}) = \sum_{s=1}^{S} E_{\mathrm{CE}}(\mathbf{\Lambda}_{\mathrm{SAT}}, \boldsymbol{g}_{l_{\mathrm{SD}}}^{s}; \mathcal{X}_s). \tag{3.4}$$

Here, $\mathbf{\Lambda}_{\mathrm{SAT}} = \{\boldsymbol{\lambda}_l^{\mathrm{SAT}}; l = 1, \cdots, l_{\mathrm{SD}} - 1, l_{\mathrm{SD}} + 1, \cdots, L\}$, $\boldsymbol{G}_{l_{\mathrm{SD}}} = \{\boldsymbol{g}_{l_{\mathrm{SD}}}^{s}; s = 1, \cdots, S\}$, $\boldsymbol{g}_{l_{\mathrm{SD}}}^{s}$ is the parameters of the SD module for training speaker s, $\mathcal{X}_s$ is the speech data spoken by training speaker s, and ρ_{SAT} is a non-negative scalar regularization coefficient. The definition of $E_{\mathrm{CE}}(\mathbf{\Lambda}_{\mathrm{SAT}}, \boldsymbol{g}_{l_{\mathrm{SD}}}^{s}; \mathcal{X}_s)$ is basically the same as the accumulated CE error of Eq. (3.2), except that SD module $\boldsymbol{g}_{l_{\mathrm{SD}}}^{s}$ is switched here for every training speaker. The training aims to find the optimal states of both $\mathbf{\Lambda}_{\mathrm{SAT}}$ and $\boldsymbol{G}_{l_{\mathrm{SD}}}$ that correspond to the minimum CE error situation achieved in conjunction with the SD module-based regularization. Here, we define the regularization term only using $\boldsymbol{G}_{l_{\mathrm{SD}}}$, taking into account the limited size of training data for each training speaker. The regularization details will be explained in Section 3.2.2. The minimization in Eq. (3.3) is conducted based on the EBP parameter update rule.

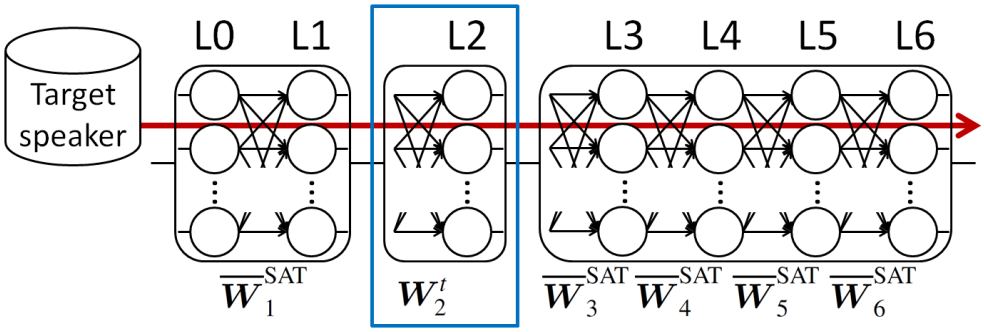


Figure 3.3: DNN structure and adaptation training procedure for speaker adaptation stage.

Speaker Adaptation Stage

In the adaptation stage, all of the adaptation parameter sets in $\boldsymbol{G}_{l_{\mathrm{SD}}}$ are removed and replaced with a new adaptation parameter set $\boldsymbol{g}_{l_{\mathrm{SD}}}^t$ for target speaker t. Then, only $\boldsymbol{g}_{l_{\mathrm{SD}}}^t$ is adapted using his/her speech data.

In Figure 3.3, we illustrate the speaker adaptation procedure, assuming we set the SD modules to L_2 of the DNN trained in the SAT stage. In the figure, $\boldsymbol{g}_2^t = \{\boldsymbol{W}_2^t, \boldsymbol{b}_2^t\}$ represents the SD module parameters for target speaker t, and $\overline{\boldsymbol{\Lambda}}_{\mathrm{SAT}} = \{\overline{\boldsymbol{\lambda}}_l^{\mathrm{SAT}}; l = 1, 3, \cdots, L\}$ represents the SI module parameters optimized in the SAT stage. The inserted SD module $\boldsymbol{g}_2^t$ is adapted to $\overline{\boldsymbol{g}}_2^t$ using the speech data of speaker t. An important point here is that only the inserted SD module is adapted; the remaining DNN part, i.e., the SI module, is fixed.

In the scenario where the SD module layer is $\mathrm{L}_{l_{\mathrm{SD}}}$, the adaptation stage is formalized as follows:

$$\overline{\boldsymbol{g}}_{l_{\mathrm{SD}}}^t = \arg\min_{\boldsymbol{g}_{l_{\mathrm{SD}}}^t} \left\{ E_{\mathrm{CE}}(\overline{\boldsymbol{\Lambda}}_{\mathrm{SAT}}, \boldsymbol{g}_{l_{\mathrm{SD}}}^t; \mathcal{X}_t) + \frac{\rho_{\mathrm{SA}}}{2} R(\boldsymbol{g}_{l_{\mathrm{SD}}}^t) \right\}, \tag{3.5}$$

where $\boldsymbol{g}_{l_{\mathrm{SD}}}^t$ is the SD module parameters for speaker t, $\mathcal{X}_t$ is the speech data spoken by speaker t for adaptation, and ρ_{SA} is a regularization coefficient. The regularized minimization of E_{CE} is conducted only with respect to $\boldsymbol{g}_{l_{\mathrm{SD}}}^t$, just using $\mathcal{X}_t$. Similar to the SAT stage, the minimization in Eq. (3.5) is conducted based on the EBP parameter update rule. The regularization here will be described in Section 3.2.2.

The training procedure of Eq. (3.3) optimizes $\mathbf{\Lambda}_{\mathrm{SAT}}$ on the premise that only the SD module ($\boldsymbol{g}^t_{l_{\mathrm{SD}}}$) is adapted and the SI module ($\overline{\mathbf{\Lambda}}_{\mathrm{SAT}}$) is fixed in the adaptation stage of Eq. (3.5). Based on the consistency between the SAT and adaptation stages, $\overline{\mathbf{\Lambda}}_{\mathrm{SAT}}$ works better than $\overline{\mathbf{\Lambda}}_{\mathrm{SI}}$, which is optimized with no assumption that the SD module is adapted, as an SI module in the adaptation stage.

Regularization

To avoid over-training, we adopt regularization-incorporated error minimization in all of our DNN training procedures. Among various possibilities, we especially use the L^2 norm-based regularization term.

For the initialization stage of the SAT-based adaptation scheme or the standard SI training, we define the regularization term as follows:

$$R(\mathbf{\Lambda}_{\mathrm{SI}}) = \sum_{l=1}^{L} \left(\| \boldsymbol{W}_l^{\mathrm{SI}} \|^2 + \| \boldsymbol{b}_l^{\mathrm{SI}} \|^2 \right), \tag{3.6}$$

which is often referred to as *weight decay* [39, 40] in the neural network research field.

For the SAT stage, we adopt the following regularization term:

$$R(\boldsymbol{G}_{l_{\mathrm{SD}}}) = \sum_{s=1}^{S} \left(\|\boldsymbol{W}^s_{l_{\mathrm{SD}}} - \overline{\boldsymbol{W}}^{\mathrm{SI}}_{l_{\mathrm{SD}}}\|^2 + \|\boldsymbol{b}^s_{l_{\mathrm{SD}}} - \overline{\boldsymbol{b}}^{\mathrm{SI}}_{l_{\mathrm{SD}}}\|^2 \right), \tag{3.7}$$

which was previously called L^2 *prior regularization* [41]. This regularization softly ties $\boldsymbol{G}_{l_{\mathrm{SD}}}$ to $\overline{\boldsymbol{\lambda}}^{\mathrm{SI}}_{l_{\mathrm{SD}}} = \{\overline{\boldsymbol{W}}^{\mathrm{SI}}_{l_{\mathrm{SD}}}, \overline{\boldsymbol{b}}^{\mathrm{SI}}_{l_{\mathrm{SD}}}\}$. In the SAT stage, each SD module is trained using just one speaker's data. On the other hand, the training of the SI module, $\mathbf{\Lambda}_{\mathrm{SAT}}$, can be done using the data of all of the training speakers. Compared to large size of hundreds hours of training data from all speakers, the training data from one speaker is often limited with less than a few tens of minutes. Considering this difference in data size, we define the regularization term only for $\boldsymbol{G}_{l_{\mathrm{SD}}}$.

For the speaker adaptation stage, we also define the L^2 prior-based regularization term for the SD

module of target speaker t, $\boldsymbol{g}^t_{l_{\mathrm{SD}}}$, as follows:

$$R(\boldsymbol{g}^t_{l_{\mathrm{SD}}}) = \|\boldsymbol{W}^t_{l_{\mathrm{SD}}} - \boldsymbol{W}^{\mathrm{anchor}}_{l_{\mathrm{SD}}}\|^2 + \|\boldsymbol{b}^t_{l_{\mathrm{SD}}} - \boldsymbol{b}^{\mathrm{anchor}}_{l_{\mathrm{SD}}}\|^2, \tag{3.8}$$

where $\boldsymbol{g}^t_{l_{\mathrm{SD}}}$ is softly tied to its anchor state, $\boldsymbol{g}^{\mathrm{anchor}}_{l_{\mathrm{SD}}} = \{\boldsymbol{W}^{\mathrm{anchor}}_{l_{\mathrm{SD}}}, \boldsymbol{b}^{\mathrm{anchor}}_{l_{\mathrm{SD}}}\}$, such that it does not over-fit $\mathcal{X}_t$.

$\boldsymbol{g}^{\mathrm{anchor}}_{l_{\mathrm{SD}}}$ can be prepared in several different ways. Simple ways include using a network initialized by small random numbers and using the SD module of the SI-trained network, $\overline{\boldsymbol{\lambda}}^{\mathrm{SI}}_{l_{\mathrm{SD}}}$. Compared to the former, the latter would better fit to the anchor for speaker adaptation (or adaptation-oriented regularization): The latter module is already trained for speech recognition. However, the SI training reflects none of the SAT concept. Obviously, the anchor state should import the SAT concept at least to some extent, since the anchor is used for the adaptation of the SAT-based network. Taking this into account, we adopt the following three-step method: 1) remove $\overline{\boldsymbol{G}}_{l_{\mathrm{SD}}}$ from the DNN part trained by Eq. (3.3); 2) insert $\overline{\boldsymbol{\lambda}}^{\mathrm{SI}}_{l_{\mathrm{SD}}}$ into the SD module layer of the DNN part; 3) re-train (in the SI training sense) the SD module using all the training speech data, while fixing the remainder of the DNN part, i.e., $\overline{\boldsymbol{\Lambda}}_{\mathrm{SAT}}$. The resulting state of the SD module is used as $\boldsymbol{g}^{\mathrm{anchor}}_{l_{\mathrm{SD}}}$.

The anchor state produced in the above way is also used as the initial status of the target speaker's SD module in the adaptation stage. This initialization is expected to be effective for successive adaptation, because the anchor state is trained so as to utilize the optimized SAT-based network.

3.2.3 Relation to Previous Works

With the recent advent of a new HMM-based hybrid approach, various speaker adaptation methods for DNN-HMM recognizers have been extensively studied [41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]. A principal adaptation strategy in these methods is to adapt only the DNN part without changing the pre-trained HMM part. The methods, which also adopt some restriction mechanisms in DNN training to avoid the over-training problem [58], are categorized

into the following two main groups: 1) restricting the network's high feature representation capability using additional small-size adaptable parameters [43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53], and 2) restricting the network's capability by incorporating some regularization terms in the adaptation stage [41, 42]. The first group of methods are further subdivided as follows: 1) adapting only the linear networks inserted into an SI DNN [43, 44, 45, 46, 47], 2) adapting such limited size augmented features as *speaker code* [48, 49] or *i-vector* [50, 51, 59] in SI DNN, and 3) adapting SD parameters embedded in the node activation function of SI DNN [52, 53]. A common maneuver by the second group of methods is to secure a large DNN capability and control it with regularization. Among a number of possibilities of implementing regularization, its effect was studied using the regularization term based on either the L^2 norm of the difference between an initial SI DNN and an adapted DNN [41] or the Kullback-Leibler divergence between the outputs of an initial SI DNN and an adapted DNN [42].

The first group of speaker adaptation methods for DNN-HMM recognizers is probably efficient based on using a limited size of adaptable parameters. However, the intrinsic value of DNN employment is to fully exploit the DNN's high feature representation capability. In addition, the second group of methods simply used SI DNN as an initial condition for adaptation, although it straightforwardly treated DNN's potential. In the light of the SI-based initialization, the first group of methods was also in the same situation as the second group.

In parallel with our proposed scheme, several speaker normalization techniques for DNN-HMM recognizers have been investigated [54, 55, 56]. These techniques adopted canonical DNN modeling of a virtual representative speaker, which is different from a standard SI-training-based DNN. The canonical DNN represented one (even a virtual) speaker in a compact form, and this nature is clearly common to the SAT concept.

The design of the canonical DNN was based in common on the speaker normalization applied to the input features, although the normalization was done differently. It was conducted using GMM-HMM recognizers [54, 55] and additionally adopting a speaker normalization network whose input was i-vector [56]. Compared with these techniques based on input feature level normalization, our scheme, in which the normalization (or adaptation) is embedded in the deep

Table 3.1: Corpus information for TED Talks corpus.

TED Talks	Hours	Speakers
Training	75	300
Development	2.4	10
Evaluation	4	28

layers of DNN, is characterized by a positive use of DNN power.

3.2.4 Experimental Conditions

Data Preparation

We tested our proposed method on the difficult lecture speech data of the TED Talks corpus [2]. We collected the speech data from the TED Talks and prepared three data sets: training, development, and evaluation.

The training data set consisted of the speech data of 300 speakers; the data of each speaker were about 15 minutes long. The total length of the training data was about 75 hours. The development data set consisted of the speech data of ten speakers, each of whom was different from the speakers in the training data set. The total length of the development data was about 2.4 hours. This set was used for finding the optimal values of the hyper-parameters, which produced high recognition accuracies over the set itself, such as the learning rate and the regularization coefficient. The evaluation data set consisted of the speech data of 28 speakers, which was used for the IWSLT2013 evaluation data set [60]. The total length of the evaluation data was about 4 hours. The speakers in this evaluation data were different from those both in the training and development data sets. Each speaker's data ranged from 2.6 to 16.5 minutes, and the average length of the evaluation data was about 8.5 minutes.

[2] `http://www.ted.com`

Table 3.2: Conditions related to feature extraction.

Acoustic feature	MFCC + log-power + Δ + ΔΔ (39-dim)
Input to DNN	11 concatenated acoustic feature (429-dim)
Sampling frequency	16 kHz
Frame length	20 ms
Frame shift	10 ms
Window function	Hamming

Such basic information of the above corpus as the number of hours and speakers is summarized in Table 3.1.

Feature Representation

Conditions related to feature extraction are briefly summarized in Table 3.2. The input speech was first converted to a series of acoustic feature vectors, each of which was calculated through a 20-ms Hamming window that was shifted at 10-ms intervals. The acoustic feature vector consisted of 12 Mel-scale Frequency Cepstrum Coefficients (MFCCs) [61], logarithmic power (log-power), 12 ΔMFCCs, Δlog-power, 12 ΔΔMFCCs, and ΔΔlog-power, where Δ and ΔΔ denote the first and second derivatives. The acoustic feature vectors had 39 dimensions. Next, the 11 adjacent acoustic feature vectors were concatenated as a 429-dimensional input to the DNN part. Each element of the input vectors was normalized so that its mean and variance became 0 and 1.

Evaluated Recognizers

To evaluate our proposed SAT-based adaptation scheme, we compared the following four recognizers:

1. SI: the SI-based DNN-HMM recognizer,

2. SA-SI: the speaker-adapted SI recognizer,
3. SAT: the DNN-HMM recognizer trained in the SAT stage,
4. SA-SAT: the speaker-adapted SAT recognizer.

Here, the SI recognizer works as a baseline case in the experiments, and the SI and SA-SI recognizers are the counterparts to the SAT and SA-SAT recognizers, both of which are based on our proposed scheme.

We adopted a simple seven-layer DNN as the baseline SI recognizer. The whole network was first pre-trained by layer-wise RBM training and successively trained using the CE error minimization over the training data set as in Eq. (3.1).

The SA-SI recognizer was produced by adapting one of the SI recognizer's intermediate network layers, which corresponded to an SD module, using the speech data of an adaptation-target speaker selected from the 28 testing speakers. To avoid the over-training problem due to the limited amount of adaptation data, we applied the regularization term of Eq. (3.8) to the update of the weights and biases of layer l_{SD}, setting $\boldsymbol{g}_{l_{\mathrm{SD}}}^{\mathrm{anchor}}$ to $\overline{\boldsymbol{\lambda}}_{l_{\mathrm{SD}}}^{\mathrm{SI}}$, when adapting the SI recognizer to the SA-SI recognizer.

We built the SAT recognizer by the following procedures, as described in Section 3.2.2. We adopted the baseline SI recognizer as the initial status of the SAT recognizer and inserted 300 SD modules into the baseline recognizer. Here, the number of SD modules is the same as that of the training speakers. We next generated a trained network along the SAT-based optimization course of Eq. (3.3). Finally, we completed the SAT recognizer by replacing the 300 used SD modules with a new SD module, which was the anchor module described in Section 3.2.2. This new SD module worked as the initial status for successive adaptations.

The SA-SAT recognizer was produced by adapting only the SD module of the SAT recognizer in the speaker-by-speaker mode, where an adaptation-target speaker was selected from the 28 testing speakers.

In this experiment, the DNN module in our recognizers had seven layers as described in Fig-

ure 3.1 and used 429 input nodes, 4909 output nodes, and 512 nodes for all of the intermediate layers. A sigmoid activation function was used for the intermediate layer nodes; a softmax activation function was used for the output layer nodes. Here, the number of output nodes was the same as the senone classes.

In all of our recognizers, the HMM part used the context-dependent acoustic model and used the 4-gram language model that was trained over the transcriptions of TED Talks, News Commentary, and English Gigaword [62]. The baseline GMM-HMM recognizer was trained on the Boosted Maximum Mutual Information (BMMI) criterion [63], which was used to obtain the senone alignment labels for the DNN training and the adaptation. During the DNN training, HMM's state transition probability was fixed. In the decoding phase, the DNN-HMM recognizers used the scaled likelihood calculated by DNN in place of the state output probability calculated using the GMM, as described in Section 2.2.2.

As above, from the five intermediate layers, we selected one as an SD module in the adaptation stage of either the SA-SI or SA-SAT recognizer and elaborated the layer selection effect in the speaker adaptation by changing a selected layer from the 1st through the 5th intermediate layers. This decision was motivated by our research interest to reveal the roles of the intermediate layers for (speaker) feature representation.

Evaluation Procedures

In terms of the availability of reference word transcriptions, we evaluated our proposed method in two different experimental procedures: supervised adaptation and unsupervised adaptation. In the supervised adaptation procedure, we adapted the SD modules using the (correct) reference transcriptions of the adaptation speech data. In the unsupervised adaptation procedure, we did the adaptation using transcriptions that were generated by decoding with the SI recognizer in place of the reference transcriptions. Here, we calculated the word confidence measure values based on the confusion networks [64] for the decoded transcriptions. Only the speech segments, whose measure values exceeded a preset threshold, were adopted for the adaptation.

To circumvent the problem of a closed-form evaluation, we adopted the four-fold cross-validation (CV) experiment paradigm where the speech data of every testing speaker were divided into four groups. In this paradigm, the validation of the adaptation result consisted of the SD module adaptation using three of the four data groups and the evaluation of the adapted SD module using the remaining data group. We repeated this validation four times by changing the combination of three groups for adaptation and one group for evaluation. The recognition accuracies, i.e., word error rates (WERs), in later discussions are the averages obtained by performing this CV-based evaluation over the speech data of all the adaptation speakers.

Training and Adaptation

DNN training sometimes requires careful control of the learning rate. Therefore we controlled it at each *training epoch*, where every sample in the training data set was used once for the network parameter updates, using the following rule based on the frame-level recognition accuracies over the development data set. If the recognition accuracy increased over the development data set, the learning rate was kept the same as in the previous epoch. Otherwise, it was halved, and the network parameters, i.e., the weights and biases, were replaced with those that produced the maximum recognition accuracy in the preceding training epochs, and the training for these replaced weights and biases was restarted using the halved learning rate.

Especially in the SAT stage, we used the average frame-level recognition accuracy obtained from the trial adaptation using the development data. At every training epoch in this stage, we first adopted the resultant DNN (the SI module plus the SD modules) of the previous epoch as an initial network status for adaptation. Using the speech data in the development data set, we performed a trial adaptation while switching the SD module and its corresponding speaker data. During the adaptation, the SI module was fixed. After the adaptation, we calculated the frame-level recognition accuracy in the speaker-by-speaker manner. In the same way as the evaluation of the trained recognizers, we repeated the trial adaptation and accuracy calculation in the CV paradigm. The averaged frame-level recognition accuracy we obtained was used to control the learning rate at every epoch.

For the SI recognizer, we set the initial value of the learning rate to 0.004 and repeated 20 epochs, where the learning rate was controlled based on the above update rule. When producing the anchor SD module (for the SAT recognizer) and the speaker adaptive trained network of Eq. (3.3), we adopted the same settings as in the SI recognizer. For the regularization terms, ρ_{SI} in Eq. (3.1) was set to 0.0 based on preliminary experiments: 0.1 for ρ_{SAT} in Eq. (3.3).

In contrast, in the adaptation stage where only the SD module was updated, we simply set the learning rate to a fixed value that was selected based on the frame-level recognition accuracies over the development data set. We selected a learning rate of 0.005 for the adaptation of the SA-SI recognizer and 0.001 for the SA-SAT recognizer. Both of these adaptation procedures were repeated ten times (ten epochs) with a regularization coefficient of 0.1 for ρ_{SA} in Eq. (3.5). , which was selected again using the frame-level recognition accuracies over the development data set.

For each allocation of the SD module layer, the above procedures for setting the hyper-parameters were repeated. Accordingly, regardless of the SD module positioning, all of the training in the initialization, SAT, and adaptation stages was conducted with the tuned hyper-parameters that produced the highest frame-level recognition accuracies over the development data set.

Mini-Batch-Based Error Minimization

To accelerate the experiments with GPU's high computation power, we adopted a mini-batch mode minimization. Especially in the case of using Eq. (3.3) for the SAT recognizer, because we had to feed the training speech data to the network while switching the SD module, the mini-batch of speech data was required to be only composed of the speech data of a single speaker. To meet the requirements of the speech data preparation, we implemented the SAT procedure in the following way. For every training epoch, we first made mini-batches, each of which consisted of the speech data of a single training speaker, over the whole training data. Next we randomized their order to avoid unexpected convergence to poor local optima in the EBP error minimization and conducted error calculation and parameter updates while switching the randomized mini-

Table 3.3: Experimental results (word error rate [%]) in supervised adaptation procedure.

l_{SD}	SI	SA-SI	SAT	SA-SAT
1	26.4	20.0	27.2	**18.9**
2	26.4	19.0	26.9	**18.2**
3	26.4	18.7	27.0	**18.0**
4	26.4	19.0	26.6	**18.4**
5	26.4	19.5	26.5	**19.0**

batches and the SD modules in the speaker-by-speaker manner. Then we repeated the epochs N_{epoch} times, where N_{epoch} was the number of epoch repetitions.

3.2.5 Experimental Results

Results in Supervised Adaptation

Table 3.3 shows the results in the supervised adaptation procedure. It shows the recognition performances of the WER of four evaluated recognizers: the SI recognizer, the SA-SI recognizer, the SAT recognizer, and the SA-SAT recognizer. Each error rate for the SA-SI and SA-SAT recognizers is the average value obtained by the previously described CV paradigm. In the table, l_{SD} is the index of the layer to which the SD module was allocated. Because the baseline SI recognizer did not have an SD module layer, the same error rate value, 26.4%, is shown in all the corresponding columns.

The SA-SI recognizer results show that the conventional way of adapting the SD module in the SI recognizer produced clear improvements. Its error reductions from the rates of the baseline SI recognizer ranged from 6.4 to 7.7 points. However, comparing the SA-SI and SA-SAT recognizers clearly demonstrates the effect of our SAT-based adaptation scheme for DNN-HMM recognizers. Regardless of the allocation of the SD module layer, the SA-SAT recognizer outperformed the SA-SI recognizer. Moreover, the SA-SAT recognizer successfully reduced the

lowest error rate of the SA-SI recognizer, 18.7%, to 18.0%, which was the best among all of the obtained error rates.

To prove the effectiveness of our SAT-based adaptation scheme, we conducted a matched pairs t-test for the difference in WERs between the SA-SAT recognizer and its counterpart SA-SI recognizer. Here, each WER was observed for one of the 28 testing speakers through the CV paradigm. From the test results, we found that the error rate reductions between the SA-SAT recognizer and the SA-SI recognizer were significant with $p < 0.01$ when l_{SD} was set to 1, 2, 3, or 4, and with $p < 0.05$ when l_{SD} was set to 5.

The results of the SAT recognizer were not promising. However, since the SAT scheme aims to increase the recognition accuracy after the adaptation but not to construct a high-performance recognizer without the adaptation, these high error rates are not really a problem.

As shown in Table 3.3, the effects of the recognizer training/adaptation methods are often evaluated using the average accuracies over multiple testing speakers; such evaluation is obviously important. However, at the same time, it is desirable that the methods accurately work for all of the testing speakers or as many testing speakers as possible. Such reliability (or stability) of the methods is also clearly important. From this viewpoint, we compared the accuracy of the SA-SAT and SA-SI recognizers in the speaker-by-speaker manner and found that our proposed SA-SAT recognizer outperformed the SA-SI recognizer for 75% to 93% of the 28 testing speakers[3].

The table also shows another quite interesting finding. The adaptation allocating the SD module to such inner layers as the 2nd or 3rd layer outperformed the allocation of the SD module to the outer layers near the input or output of the network, such as the 1st and 5th layers. This phenomenon was commonly observed in both the SA-SI and SA-SAT recognizers. The DNN part repeated the feature transformation along with the data feed-forwarding from the input layer to the output layer. Allocating the SD modules to the inner layers allowed a complex feature transformation in both the lower and upper layers. Such a well balanced transformation is prob-

[3]The percentage changed according to the selection of the SD module layer.

Table 3.4: Experimental results (word error rate [%]) in unsupervised adaptation procedure.

l_{SD}	SI	SA-SI	SAT	SA-SAT
1	26.4	21.4	27.2	**20.4**
2	26.4	20.6	26.9	**20.0**
3	26.4	20.7	27.0	**20.1**
4	26.4	21.0	26.6	**20.3**
5	26.4	21.5	26.5	**21.0**

ably useful for extracting salient information for recognition/adaptation, although its mechanism remains unclear. Accordingly, we consider the use of DNN more suitable for speaker adaptation (probably also for other types such as speaking environment and transmission channel adaptations) than conventional shallow neural networks or any simple front-end architecture that has no deep layer structure.

Results in Unsupervised Adaptation

Table 3.4 shows the results in the unsupervised adaptation procedure. As in Table 3.3, Table 3.4 shows the recognition performances in the WER of the four evaluated recognizers. Each error rate for the SA-SI and SA-SAT recognizers is also the average value obtained through the CV paradigm. To select the adaptation data used for the unsupervised adaptation, we set the threshold value for the confidence measure to 0.5 based on the preliminary experimental results. In the preliminary experiments, we tested several different values for the confidence measure and found that the confidence measure of 0.5 achieved a reduction in WER of 0.2 (from the rate obtained without the confidence measure) on average for both the SA-SAT recognizer and SA-SI recognizer. The effects of the confidence measure slightly varied according to the selections of the SD module layers and the recognizers.

In the unsupervised adaptation results, we identified a trend similar to the supervised adaptation results. The SA-SAT recognizer outperformed the SA-SI recognizer, regardless of the SD mod-

ule layer allocation. Moreover, the SA-SAT recognizer achieved the lowest error rate, 20.0%, which was 0.6 point lower than that of the SA-SI recognizer. Compared to the supervised adaptation, the accuracy improvement in the unsupervised adaptation was not large: The reference transcription was not used in the adaptation training. However, comparing the SA-SI and SA-SAT recognizers also clearly demonstrates the effect of the SAT-based DNN-HMM recognizer, even in the unsupervised adaptation procedure.

In the unsupervised adaptation case, we again conducted the matched pairs t-test for the differences in the WERs between the SA-SAT recognizer and the SA-SI recognizer. Here, each WER was obtained for one of the 28 testing speakers through the CV paradigm. The test results proved that the error rate reductions between the SA-SAT recognizer and the SA-SI recognizer were significant with $p < 0.01$ when l_{SD} was set to 1 or 4, and with $p < 0.05$ when l_{SD} was set to 2, 3, or 5.

In addition, the adaptation allocating the SD module to the inner layers also outperformed the case of allocating the SD module to the outer layers even in the unsupervised procedure.

Stability Analysis of Adaptation

To deepen our understanding of our SAT-based adaptation, we elaborate the SA-SI and SA-SAT recognizers produced in the supervised adaptation procedure.

In the above experiments, we set the number of adaptation epoch iterations to ten and gained higher performances with our SA-SAT recognizer than with SA-SI recognizer. This setting was done from the viewpoint that fast adaptation was more preferable. However, there is the possibility that the SA-SAT recognizer's advantage was gained by selecting a proper length of the epoch iteration by chance. Actually, as a side effect, a short iteration occasionally increases recognition accuracies. To scrutinize this point, we ran the adaptation by setting the iteration number to 50. Here, except for the number of adaptation epoch iterations, we used the same hyper-parameter settings as described in Section 3.2.4.

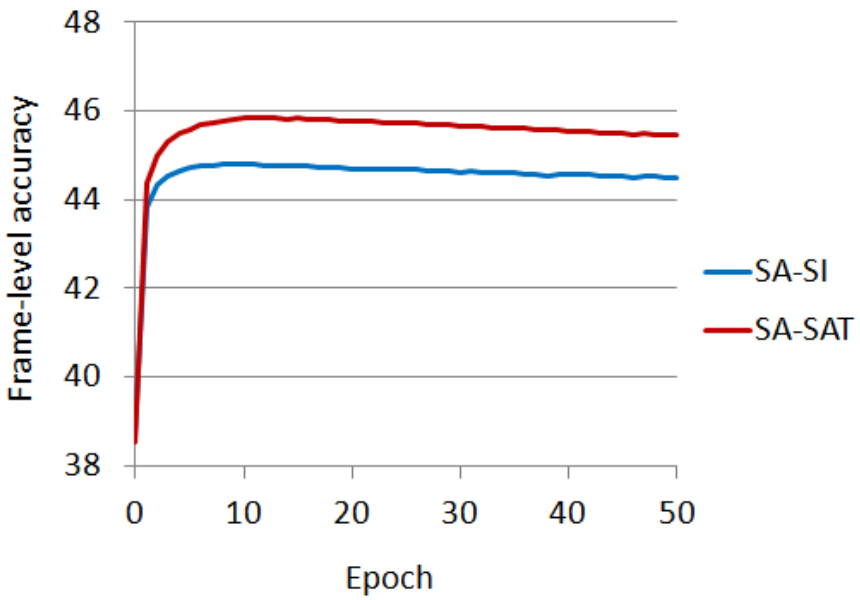


Figure 3.4: Frame-level senone recognition accuracy [%] as a function of supervised adaptation epoch.

Figure 3.4 illustrates the frame-level senone recognition accuracies (in the vertical axis), each of which is a function of the epoch (in the horizontal axis), of the SA-SI and SA-SAT recognizers. This senone recognition accuracy has a close relation with the criterion used in the adaptation stage. Each accuracy curve in the figure was obtained, for its corresponding recognizer, by averaging the accuracies over all the experiment runs conducted by changing the SD module layers in the CV paradigm.

The figure shows that the SA-SAT recognizer stably outperformed the SA-SI recognizer in all the adaptation epochs. The advantage of our SA-SAT recognizer is probably generated by the nature of the SAT-based adaptation mechanism.

Comparison with All Layer Adaptation

In the above experiments, we demonstrated the superiority of our proposed SAT-based adaptation scheme to the SI-based adaptation scheme. In the SI-based adaptation, we adapted only the SD module similarly to the SAT-based adaptation. One may question whether the SAT-based

Table 3.5: Comparisons among module-based adaptation (SA-SI-L3 and SA-SAT-L3 recognizers) and non-module-based adaptation (SA-SI-ALL recognizer).

Recognizer	# Param.	Word error rate [%]
SA-SI-L3	0.26 M	18.7
SA-SI-ALL	2.8 M	18.3
SA-SAT-L3	0.26 M	**18.0**

(module-based) adaptation is more effective than the simple (non-module-based) adaptation of the whole DNN of the SI recognizer. To analyze this point, we compared the following three recognizers in the supervised adaptation procedure: (1) the SA-SAT recognizer allocating the SD module in the third layer (*SA-SAT-L3 recognizer*), (2) the SA-SI recognizer allocating the SD module in the third layer (*SA-SI-L3 recognizer*), and (3) a new Speaker-Adapted SI recognizer in which all of the trainable DNN parameters, i.e., the connection weights and biases in all layers, were used for adaptation (*SA-SI-ALL recognizer*). The SA-SAT-L3 and the SA-SI-L3 recognizers were the best (in terms of SD module layer allocation) SA-SAT and SA-SI recognizers, as described in Table 3.3. Furthermore, the SA-SI-ALL recognizer was constructed, adopting the same training/adaptation procedures (e.g., the use of L^2 prior regularization and the CV paradigm) as those for the recognizers in Table 3.3.

Table 3.5 shows the number of adaptation parameters and the WERs for the above three recognizers. In the table, "M" represents million. Using a larger number of adaptation parameters, the SA-SI-ALL recognizer gained a WER reduction of 0.4 point from that of the SA-SI-L3. However, the rate by the SA-SI-ALL recognizer did not achieve the lowest WER, which was reached by the SA-SAT-L3 recognizer at 18.0%. To analyze the effect of the SAT-based (module-based) adaptation against the non-module-based adaptation, we conducted the matched pairs t-test for the difference in WERs between the SA-SI-ALL recognizer and the SA-SAT-L3 recognizer. The test results proved that improvement of the SA-SAT-L3 recognizer over the SA-SI-ALL recognizer was significant with $p < 0.05$.

Although the t-test proved the statistical difference in WERs between the SA-SAT-L3 and SA-

SI-ALL recognizers, the difference was not so large. A point to note here is that the SA-SAT-L3 recognizer used only 9% (0.26 M) of the adaptation parameters of the SA-SI-ALL recognizer (2.8 M). This leads to a dramatic reduction in the size of adaptation parameters, which must be stored and adapted for each target speaker. For example, let us assume that a speech recognition system runs on some server and tries to increase its discriminative power through speaker adaptation for a huge number of system users (speakers). The system is expected to handle many different speakers' data simultaneously and thus must store on the server many adaptation parameter sets, each for a different speaker. Obviously, a small size of speaker-dependent adaptation parameters is favorable in this common scenario. In addition, it is expected that the use of such a small number of adaptation parameters decreases the risk of the over-training problem, especially in the cases where speech data available for adaptation training are severely limited.

3.3 Extension with Linear Transformation Networks

3.3.1 Motivations

In the previous section (Section 3.2), we described the basic formalization of our proposed SAT-based speaker adaptation scheme, where the SAT scheme was implemented by allocating one SD module for each training speaker to one of the intermediate DNN layers. The method then jointly optimized the SD modules and the SI module, which was shared by all the training speakers, with changing a pair comprised of the SD module and its corresponding training speaker in a speaker-by-speaker manner. The effectiveness of this SAT-based DNN training scheme, which we call the SAT-DNN-ORG method in the following sections, was clearly demonstrated through systematic experiments.

However, the SAT-DNN-ORG method still has room for improvement. For example, it used an SI DNN, which was trained in an SI mode, as an anchorage state in the regularization of the SAT stage. This might have decreased the adaptability of SAT due to a large restriction from the SI DNN. In addition, in the speaker adaptation stage, it just empirically initialized the SD

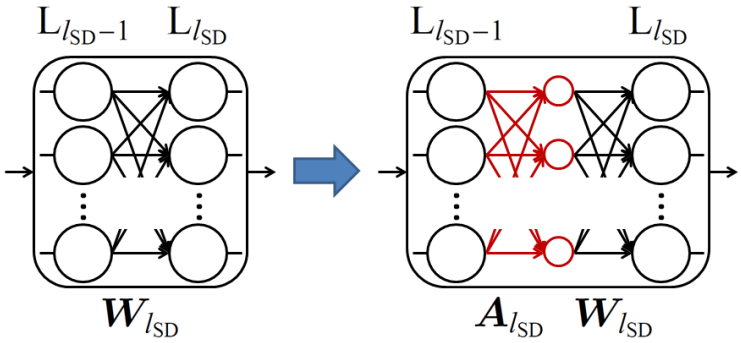


Figure 3.5: Graphical explanation of LTN insertion.

module using a one-layer network that was extracted from the SI DNN and retrained with the SAT-optimized SI network over the speech data of all the training speakers. This initialized SD module represents a certain kind of mean speaker model. Due to the lack of theoretical rationale in such usage, alternatives to SI DNN and the mean speaker model will probably further improve the adaptation results.

Motivated by the above viewpoints, we extend the preceding SAT-DNN-ORG method by embedding LTN [43, 44, 45] in DNN and propose a new SAT-based speaker adaptation scheme, referred to as SAT-DNN-LTN method. In contrast to the SAT-DNN-ORG method, our new method with LTN dynamically changes the anchorage state of the network weight and bias parameters in the regularization along the training progress of SAT and automatically provides, in a natural way, the initial status of an SD module that is used for a new speaker in the speaker adaptation step.

3.3.2 Linear Transformation Network

In our proposed SAT-DNN-LTN method, we insert LTN, which works as an SD module, to one of the DNN layers. In the adaptation stage, we only train the LTN part using target speaker's speech data.

We illustrate the insertion of LTN in Figure 3.5, where based on the fact that the inserted LTN is SD module, we denote the layer having LTN (i.e., SD module layer), its corresponding DNN

parameters, and its corresponding LTN parameters by $\mathrm{L}_{l_{\mathrm{SD}}}$, $\boldsymbol{\lambda}_{l_{\mathrm{SD}}}$, and $\tilde{\boldsymbol{g}}_{l_{\mathrm{SD}}}$, respectively. Note that $\boldsymbol{\lambda}_{l_{\mathrm{SD}}} = \{\boldsymbol{W}_{l_{\mathrm{SD}}}, \boldsymbol{b}_{l_{\mathrm{SD}}}\}$ and $\tilde{\boldsymbol{g}}_{l_{\mathrm{SD}}} = \{\boldsymbol{A}_{l_{\mathrm{SD}}}, \boldsymbol{a}_{l_{\mathrm{SD}}}\}$, where $\boldsymbol{A}_{l_{\mathrm{SD}}}$ and $\boldsymbol{a}_{l_{\mathrm{SD}}}$ are the weight matrix and bias vector of the LTN module, respectively.

Accordingly, a computational procedure for the SD module layer is formulated as follows:

$$\boldsymbol{z}_{l_{\mathrm{SD}}} = \phi(\boldsymbol{W}_{l_{\mathrm{SD}}}(\boldsymbol{A}_{l_{\mathrm{SD}}}\boldsymbol{z}_{l_{\mathrm{SD}}-1} + \boldsymbol{a}_{l_{\mathrm{SD}}}) + \boldsymbol{b}_{l_{\mathrm{SD}}}), \tag{3.9}$$

$$= \phi(\widehat{\boldsymbol{W}}_{l_{\mathrm{SD}}}\boldsymbol{z}_{l_{\mathrm{SD}}-1} + \widehat{\boldsymbol{b}}_{l_{\mathrm{SD}}}), \tag{3.10}$$

$$\widehat{\boldsymbol{W}}_{l_{\mathrm{SD}}} = \boldsymbol{W}_{l_{\mathrm{SD}}}\boldsymbol{A}_{l_{\mathrm{SD}}}, \quad \widehat{\boldsymbol{b}}_{l_{\mathrm{SD}}} = \boldsymbol{W}_{l_{\mathrm{SD}}}\boldsymbol{a}_{l_{\mathrm{SD}}} + \boldsymbol{b}_{l_{\mathrm{SD}}}, \tag{3.11}$$

where $\boldsymbol{z}_l$ is the outputs from L_l, and ϕ is an activation function. Initially, we set LTN parameters $\{\boldsymbol{A}_{l_{\mathrm{SD}}}, \boldsymbol{a}_{l_{\mathrm{SD}}}\}$ to $\{\boldsymbol{I}_{l_{\mathrm{SD}}}, \boldsymbol{0}_{l_{\mathrm{SD}}}\}$, where $\boldsymbol{I}_{l_{\mathrm{SD}}}$ is an identity matrix and $\boldsymbol{0}_{l_{\mathrm{SD}}}$ is a zero vector.

3.3.3 Training Procedures

Except for the difference of the SD modules, the SAT-based adaptation procedures with LTN SD modules are basically same as the preceding SAT-DNN-ORG method as described in Section 3.2.2. Therefore, this subsection focuses on the formalization of the SAT and adaptation stages with LTN SD modules.

Speaker Adaptive Training Stage

In Figure 3.6, we illustrate the procedure of conducting SAT with LTN SD module insertion. As an example, we insert LTN SD modules $\tilde{\boldsymbol{G}}_2 = \{\tilde{\boldsymbol{g}}_2^s; s = 1, \cdots, S\}$ to L_2 in the figure, where $\tilde{\boldsymbol{g}}_2^s = \{\boldsymbol{A}_2^s, \boldsymbol{a}_2^s\}$ is the LTN SD module parameters for training speaker s in the training data set, S is the number of training speakers, $\tilde{\boldsymbol{\Lambda}}_{\mathrm{SAT}} = \{\boldsymbol{\lambda}_l^{\mathrm{SAT}}; l = 1, 2, \cdots, L\}$ are the SI module parameters, and the blue rectangle represents the SD module layer.

When inserting the LTN SD modules into $\mathrm{L}_{l_{\mathrm{SD}}}$, the training procedure of the SAT stage is for-

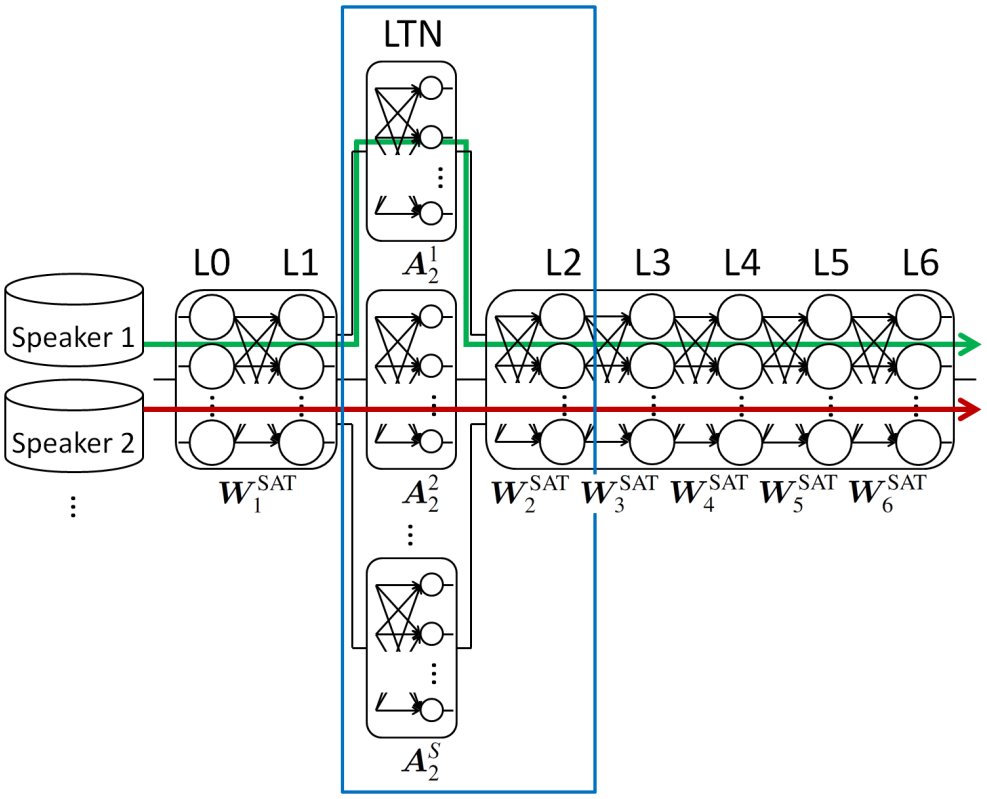


Figure 3.6: DNN structure and SAT training procedure for SAT stage in SAT-DNN-LTN method.

malized as follows:

$$(\overline{\tilde{\Lambda}}_{\text{SAT}}, \overline{\tilde{G}}_{l_{\text{SD}}}) = \underset{(\tilde{\Lambda}_{\text{SAT}}, \tilde{G}_{l_{\text{SD}}})}{\arg\min} \left\{ E_{\text{SAT-CE}}(\tilde{\Lambda}_{\text{SAT}}, \tilde{G}_{l_{\text{SD}}}; \mathcal{X}) + \frac{\rho_{\text{SAT}}}{2} R(\tilde{G}_{l_{\text{SD}}}) \right\}, \tag{3.12}$$

where

$$E_{\text{SAT-CE}}(\tilde{\Lambda}_{\text{SAT}}, \tilde{G}_{l_{\text{SD}}}; \mathcal{X}) = \sum_{s=1}^{S} E_{\text{CE}}(\tilde{\Lambda}_{\text{SAT}}, \tilde{g}^s_{l_{\text{SD}}}; \mathcal{X}_s), \tag{3.13}$$

$$R(\tilde{G}_{l_{\text{SD}}}) = \sum_{s=1}^{S} \left(\|A^s_{l_{\text{SD}}} - I_{l_{\text{SD}}}\|^2 + \|a^s_{l_{\text{SD}}} - 0_{l_{\text{SD}}}\|^2 \right). \tag{3.14}$$

Here, $\tilde{\Lambda}_{\text{SAT}} = \{\lambda_l^{\text{SAT}}; l = 1, \cdots, L\}$, $\tilde{G}_{l_{\text{SD}}} = \{\tilde{g}^s_{l_{\text{SD}}}; s = 1, \cdots, S\}$, $\tilde{g}^s_{l_{\text{SD}}}$ is the parameters of the LTN SD module for training speaker s, $R(\tilde{G}_{l_{\text{SD}}})$ is the regularization term whose anchors are set to initial LTN states $\{I_{l_{\text{SD}}}, 0_{l_{\text{SD}}}\}$, and ρ_{SAT} is a non-negative scalar regularization coefficient.

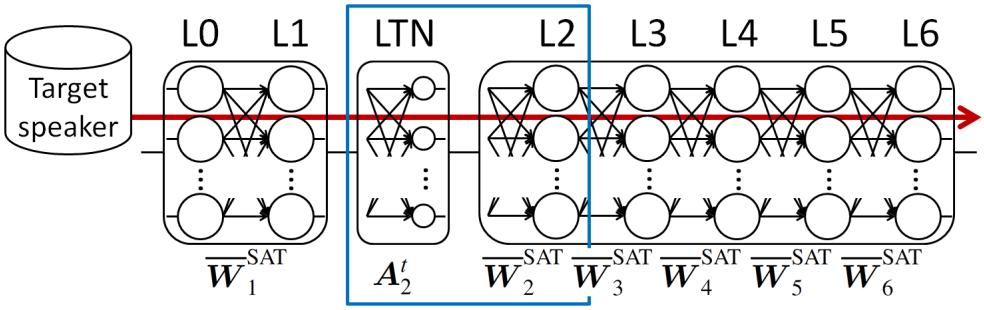


Figure 3.7: DNN structure and adaptation training procedure for speaker adaptation stage in SAT-DNN-LTN method

Speaker Adaptation Stage

In the adaptation stage, we first remove all of the SD modules in $\tilde{\boldsymbol{G}}_{l_{\mathrm{SD}}}$ and insert a new LTN SD module $\tilde{\boldsymbol{g}}_{l_{\mathrm{SD}}}^t = \{\boldsymbol{A}_{l_{\mathrm{SD}}}^t, \boldsymbol{a}_{l_{\mathrm{SD}}}^t\}$ for target speaker t. Then, only $\tilde{\boldsymbol{g}}_{l_{\mathrm{SD}}}^t$ is adapted using his/her speech data. In Figure 3.7, we illustrate the speaker adaptation procedure, assuming we set the SD modules to L_2 of the DNN trained in the SAT stage.

In the scenario where the SD module layer is $\mathrm{L}_{l_{\mathrm{SD}}}$, the adaptation stage is formalized as follows:

$$\overline{\tilde{\boldsymbol{g}}}_{l_{\mathrm{SD}}}^t = \arg\min_{\tilde{\boldsymbol{g}}_{l_{\mathrm{SD}}}^t} \left\{ E_{\mathrm{CE}}(\overline{\tilde{\boldsymbol{\Lambda}}}_{\mathrm{SAT}}, \tilde{\boldsymbol{g}}_{l_{\mathrm{SD}}}^t; \mathcal{X}_t) + \frac{\rho_{\mathrm{SA}}}{2} R(\tilde{\boldsymbol{g}}_{l_{\mathrm{SD}}}^t) \right\}, \tag{3.15}$$

where

$$R(\tilde{\boldsymbol{g}}_{l_{\mathrm{SD}}}^t) = \|\boldsymbol{A}_{l_{\mathrm{SD}}}^t - \boldsymbol{I}_{l_{\mathrm{SD}}}\|^2 + \|\boldsymbol{a}_{l_{\mathrm{SD}}}^t - \boldsymbol{0}_{l_{\mathrm{SD}}}\|^2, \tag{3.16}$$

$R(\tilde{\boldsymbol{g}}_{l_{\mathrm{SD}}}^t)$ is a regularization term whose anchors are $\{\boldsymbol{I}_{l_{\mathrm{SD}}}, \boldsymbol{0}_{l_{\mathrm{SD}}}\}$, and ρ_{SA} is a non-negative regularization coefficient.

3.3.4 Advantage over Preceding SAT-DNN-ORG Method

The preceding SAT-DNN-ORG method adopts such SI-training-based parameters as $\boldsymbol{\lambda}_{l_{\mathrm{SD}}}^{\mathrm{SI}} = \{\boldsymbol{W}_{l_{\mathrm{SD}}}^{\mathrm{SI}}, \boldsymbol{b}_{l_{\mathrm{SD}}}^{\mathrm{SI}}\}$ as the anchors for regularization in the SAT stage, as in Eq. (3.7). However, there is

no rationality to using the SI-based anchors in the SAT-based stages.

On the other hand, as in Eq. (3.14), the SAT-DNN-LTN method adopts the indentity matrix and the zero vector as the anchors for regularization. Based on the linearity of these LTN parameters as in Eqs. (3.9)-(3.11), regularization term $R(\tilde{\boldsymbol{G}}_{l_{\mathrm{SD}}})$ in Eq. (3.14) can be deemed as follows:

$$R(\tilde{\boldsymbol{G}}_{l_{\mathrm{SD}}}) = \sum_{s=1}^{S} \Big(\|\widehat{\boldsymbol{W}}^{s}_{l_{\mathrm{SD}}} - \boldsymbol{W}_{l_{\mathrm{SD}}}\|^2 + \|\widehat{\boldsymbol{b}}^{s}_{l_{\mathrm{SD}}} - \boldsymbol{b}_{l_{\mathrm{SD}}}\|^2 \Big), \tag{3.17}$$

where $\widehat{\boldsymbol{W}}^{s}_{l_{\mathrm{SD}}} = \boldsymbol{W}_{l_{\mathrm{SD}}} \boldsymbol{A}^{s}_{l_{\mathrm{SD}}}$ and $\widehat{\boldsymbol{b}}^{s}_{l_{\mathrm{SD}}} = \boldsymbol{W}_{l_{\mathrm{SD}}} \boldsymbol{a}^{s}_{l_{\mathrm{SD}}} + \boldsymbol{b}_{l_{\mathrm{SD}}}$.

Hereat it turns out that although the anchors in Eq. (3.14) are fixed in form to the identity matrix and the zero vector, they virtually change along the course of SAT stage: The anchors in Eq. (3.17) change along the training progress in the SAT stage. Importantly, the virtual effect of changing the anchors applies to the successive speaker adaptation stage, where the anchors for regularization are fixed in the same way as in Eq. (3.14), frees the initial status of the anchors from the SI-based parameters, and automatically initializes the anchors so that they can be more SAT-oriented. Moreover, the virtual effect would make the training in the SAT stage more effective by softening the regularization. Accordingly, the SAT-DNN-LTN method is expected to further increase the effect of the preceding SAT-DNN-ORG method.

3.3.5 Experimental Conditions

We tested our proposed method on the difficult, lecture speech data of the TED Talks corpus. The experimental conditions basically follow those described in Section 3.2.4 unless otherwise stated.

Evaluated recognizers

To evaluate our SAT-DNN-LTN method, we compared the following four recognizers:

1. SI: the SI-based DNN-HMM recognizer that corresponds to the initial status of SAT-LTN

and SAT-ORG recognizers,

2. SA-SI-ORG: the speaker-adapted SI recognizer developed by adapting the one layer of the DNN,
3. SA-SAT-ORG: the speaker-adapted SAT-ORG recognizer developed by adapting the one layer of the DNN,
4. SA-SAT-LTN: the speaker-adapted SAT-LTN recognizer developed by adapting the LTN,

where SAT-ORG represents the DNN-HMM recognizer trained in the SAT stage of the preceding SAT-DNN-ORG method, and SAT-LTN represents the DNN-HMM recognizer trained in the SAT stage of the SAT-DNN-LTN method. SA-SI-ORG and SA-SAT-ORG recognizers correspond to SA-SI and SA-SAT recognizers in Section 3.2.4, respectively.

Here, the SI recognizer works as a reference case in the experiments, and the SA-SI-ORG and SA-SAT-ORG recognizers are the counterparts to the SA-SAT-LTN recognizer that is based on our proposed scheme in this section.

Training and Adaptation

In the SAT stage, we set the initial value of the learning rate and the number of training epochs to 0.004 and 50, respectively. We also set the regularization coefficient to 0.1 ($\rho_{\mathrm{SAT}} = 0.1$) when inserting LTN into L_1 and to 10.0 ($\rho_{\mathrm{SAT}} = 10.0$) when inserting LTN into L_2 through L_5.

In the speaker adaptation stage, we selected a learning rate of 0.00001 and a regularization coefficient of 0.1 ($\rho_{\mathrm{SA}} = 0.1$) for inserting LTN into L_1. On the other hand, we selected a learning rate of 0.00005 and a regularization coefficient of 10.0 ($\rho_{\mathrm{SA}} = 10.0$) in the case of inserting LTN into L_2 through L_5. In each of these cases, we repeated training epochs ten times.

Table 3.6: Experimental results (word error rate [%]) in supervised adaptation procedure for SAT-DNN-LTN method.

l_{SD}	SI	SA-SI-ORG	SA-SAT-ORG	SA-SAT-LTN
1	26.4	20.0	**18.9**	20.5
2	26.4	19.0	18.2	**17.2**
3	26.4	18.7	18.0	**17.5**
4	26.4	19.0	18.4	**17.5**
5	26.4	19.5	19.0	**18.0**

3.3.6 Experimental Results

Results in Supervised Adaptation

Table 3.6 shows the results in the supervised adaptation procedure. It shows the WERs (recognition performances) for the four evaluated recognizers: the SI recognizer, the SA-SI-ORG recognizer, the SA-SAT-ORG recognizer, and the SA-SAT-LTN recognizer. Each error rate for the SA-SI-ORG, SA-SAT-ORG, and SA-SAT-LTN recognizers is the average value obtained by the CV paradigm as described in Section 3.2.4. In the table, l_{SD} is the index of the layer to which the SD module was allocated. Because the SI recognizer did not have an SD module layer, the same error rate value, 26.4%, is shown in all the corresponding columns.

Comparisons of the SA-SI-ORG and SA-SAT recognizers, i.e., SA-SAT-ORG and SA-SAT-LTN recognizer, clearly demonstrate the effectiveness of the SAT training concept. Regardless of the layer to which the SD module was allocated or inserted, the SA-SAT recognizers outperformed the SA-SI-ORG recognizer, except when the LTN SD module was inserted to L_1 for SA-SAT-LTN recognizer.

In addition, a comparison between the SAT-DNN-LTN and SAT-DNN-ORG methods also shows the effectiveness of the SAT-DNN-LTN method. Our proposed SA-SAT-LTN recognizer stably outperformed its counterpart SA-SAT-ORG recognizer in all cases except when the SD module was allocated or inserted to L_1. Moreover, the SA-SAT-LTN recognizer successfully reduced the

Table 3.7: Experimental results (word error rate [%]) in unsupervised adaptation procedure for SAT-DNN-LTN method.

l_{SD}	SI	SA-SI-ORG	SA-SAT-ORG	SA-SAT-LTN
1	26.4	21.4	**20.4**	21.9
2	26.4	20.6	20.0	**19.1**
3	26.4	20.7	20.1	**19.4**
4	26.4	21.0	20.3	**19.4**
5	26.4	21.5	21.0	**20.2**

lowest error rate from 18.7% to 17.2%, which was the best among all of the obtained error rates, against the SA-SI-ORG recognizer; from 18.0% to 17.2% against the SA-SAT-ORG recognizer.

As described in Section 3.3.4, our proposed SAT-DNN-LTN method dynamically estimates $\boldsymbol{W}_{l_{SD}}$ and $\boldsymbol{b}_{l_{SD}}$, which are anchorage states of network parameters in the regularization of the SAT step, and automatically estimates $\boldsymbol{W}_{l_{SD}}^{\mathrm{anchor}}$ and $\boldsymbol{b}_{l_{SD}}^{\mathrm{anchor}}$, which are the initial status of an SD module in the speaker adaptation stage. Based on these properties, we estimate that our SA-SAT-LTN performed the SAT and speaker adaptation stages using a more appropriate anchorage state of network parameters in the regularization and it led to the better adaptation performance.

In addition, the adaptation allocating the SD module to the inner layers also outperformed the case of allocating the SD module to the outer layers in both the SA-SI-ORG, SA-SAT-ORG and SA-SAT-LTN recognizers.

Results in Unsupervised Adaptation

Table 3.7 shows the results in the unsupervised adaptation procedure. As in Table 3.6, Table 3.7 shows the recognition performances in the WER of the four evaluated recognizers. Each error rate for the SA-SI and SA-SAT recognizers is also the average value obtained through the CV paradigm.

To select the adaptation data used for the unsupervised adaptation, we set the threshold value for

the confidence measure to 0.5 for SA-SI-ORG and SA-SAT-ORG recognizes, 0.6 for SA-SAT-LTN recognizer, based on the preliminary experiment results.

In the unsupervised adaptation results, we identified a trend similar to the supervised adaptation results. The SA-SAT-LTN recognizer achieved the lowest error rate, 19.1%, which was 1.5 points lower than that of the SA-SI-ORG recognizer and 0.9 points lower than that of the SA-SAT-ORG recognizer. It shows the effectiveness of the SAT-DNN-LTN method even in the unsupervised adaptation setups.

3.4 Extension with Bottleneck Linear Transformation Networks

3.4.1 Motivations

In the previous section (Section 3.3), by adopting LTN SD module, our SAT-DNN-ORG method was successfully upgraded to a SAT-based DNN-HMM recognizer whose SD module was defined with LTN, i.e., SAT-DNN-LTN method. Based on the linearity of the LTN SD module, SAT-DNN-LTN method provided two preferable characteristics: 1) dynamically estimate the anchorage states of network parameters in the regularization of the SAT step, and 2) automatically estimate the initial status of an SD module in the speaker adaptation step. The experiments clearly demonstrated the effectiveness of the extended SAT-DNN-LTN method.

In real-world situations where the amount of available data is finite, large-scale networks easily suffer from the over-training problem. Trained networks generally work for the data in hand but often fail for unseen data. This problematic phenomenon becomes more serious in such cases where only a severely limited amount of data is available in the speaker adaptation stage. In our preceding methods, such as the SAT-DNN-LTN method, this problem was effectively controlled using the regularization concept. Nevertheless, the size of the used SD module is still large, and its reduction is obviously desirable because a small SD module further decreases the risk of over-training and reduces computational load as well as storage cost. Motivated by this

understanding, we propose in this section a new technique for reducing the size of the DNN front-end, or more precisely the size of the LTN SD module, in the preceding SAT-DNN-LTN method without degradation of its classification power. Note that an LTN SD module is prepared for every training/target speaker and thus its computational and storage costs are high; furthermore, the size of the SD module should be strictly constrained due to the limited amount of speech data used in the speaker adaptation stage.

Several techniques for alleviating over-training by reducing the adaptable parameter size of LTN have been investigated, focusing on the sharing or restriction of the adaptable parameters [46, 53, 65]. In addition, Singular Value Decomposition (SVD)-based low-rank matrix conversion has also been investigated to reduce the number of adaptation parameters [47, 66, 67]. However, the best choice among these techniques and, more importantly, the relation between the size of adaptable parameters and their feature-representation capability in the LTN have not yet been clarified. Consequently, in this section, we first theoretically analyze this relation and next propose a new SAT-based adaptation scheme, referred to as SAT-DNN-BTN method, for reducing the size of LTN while maintaining its feature-representation power by introducing the bottleneck structure to the SD module layer.

3.4.2 Property Analysis of Linear Transformation Networks

To prepare for matrix/vector manipulation, we re-express the trainable matrices/vectors of DNN and LTN as follows: $\widehat{\mathbf{W}}_{l_{\mathrm{SD}}} = [\widehat{\mathbf{w}}_1 \cdots \widehat{\mathbf{w}}_{N_{l_{\mathrm{SD}}-1}}]$, $\mathbf{W}_{l_{\mathrm{SD}}} = [\mathbf{w}_1 \cdots \mathbf{w}_{N_{l_{\mathrm{SD}}-1}}]$, $\mathbf{A}_{l_{\mathrm{SD}}} = [\boldsymbol{\alpha}_1 \cdots \boldsymbol{\alpha}_{N_{l_{\mathrm{SD}}-1}}]$, $\widehat{\mathbf{w}}_j = [\widehat{w}_{1j} \cdots \widehat{w}_{N_{l_{\mathrm{SD}}}j}]^{\mathrm{T}}$, $\mathbf{w}_j = [w_{1j} \cdots w_{N_{l_{\mathrm{SD}}}j}]^{\mathrm{T}}$, $\boldsymbol{\alpha}_j = [\alpha_{1j} \cdots \alpha_{N_{l_{\mathrm{SD}}-1}j}]^{\mathrm{T}}$, where $N_{l_{\mathrm{SD}}}$ is the number of nodes in $\mathrm{L}_{l_{\mathrm{SD}}}$, T is transpose. Strictly speaking, such column vectors as $\mathbf{w}_j$ should have an index that expresses its corresponding insertion layer, as with $\mathbf{w}_j^{l_{\mathrm{SD}}}$, but for simplicity, we omit it.

Then, through simple matrix/vector manipulations, we reach the following relation:

$$\widehat{\mathbf{w}}_j = \sum_{i=1}^{N_{l_{\mathrm{SD}}-1}} \alpha_{ij} \mathbf{w}_i. \tag{3.18}$$

Here, we can easily find that the relation $\widehat{\mathbf{w}}_j \in S(\mathbf{W}_{l_{\mathrm{SD}}})$ holds, where $S(\mathbf{W}_{l_{\mathrm{SD}}})$ is the column space spanned by the column vectors of $\mathbf{W}_{l_{\mathrm{SD}}}$. Similarly, we can obtain the following bias vector relation: $\mathbf{W}_{l_{\mathrm{SD}}}\mathbf{a}_{l_{\mathrm{SD}}} \in S(\mathbf{W}_{l_{\mathrm{SD}}})$. Accordingly, the potential adaptability of LTN, or in other words, the size of the parameter space searched by LTN, is determined by the dimension of $S(\mathbf{W}_{l_{\mathrm{SD}}})$, that is the real rank of $\mathbf{W}_{l_{\mathrm{SD}}}$. A question arising here concerns the *effective rank* of $\mathbf{W}_{l_{\mathrm{SD}}}$. If it is smaller than the real rank of $\mathbf{W}_{l_{\mathrm{SD}}}$, it is fundamentally possible to construct a low-dimensional parameter space that has the same feature-representation capability as $S(\mathbf{W}_{l_{\mathrm{SD}}})$. Then, because such use of a smaller number of basis vectors reduces the number of adaptable parameters, the new basis vectors, or the new smaller number of parameters, are expected to stably increase the effect of speaker adaptation, especially in cases where the available speech data for adaptation training are severely limited.

3.4.3 Low Rank Approximation of Weight Matrix using Singular Value Decomposition

To find the above (smaller number of) basis vectors, we consider the application of SVD to the weight matrix of the SD module layer. For notation simplicity, omitting SD module layer index l_{SD}, we consider weight matrix $\mathbf{W}$ ($\in R^{\mu\times\nu}$) in this subsection. Then, $\mathbf{W}$ is decomposed as $\mathbf{W} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}$, where $\mathbf{\Sigma}$ ($\in R^{\mu\times\nu}$) is the rectangular diagonal matrix whose diagonal elements are singular values σ_i, and $\mathbf{U}$ ($\in R^{\mu\times\mu}$) and $\mathbf{V}$ ($\in R^{\nu\times\nu}$) are the orthogonal matrices produced by SVD. In the same way, for matrix $\widetilde{\mathbf{W}}$ whose rank is κ ($\leq$ rank $\mathbf{W}$), we obtain $\widetilde{\mathbf{W}} = \widetilde{\mathbf{U}}\widetilde{\mathbf{\Sigma}}\widetilde{\mathbf{V}}^{\mathrm{T}}$, where $\widetilde{\mathbf{\Sigma}}$ ($\in R^{\kappa\times\kappa}$) is the diagonal matrix produced by retaining the κ largest singular value elements (while removing the remainder) in $\mathbf{\Sigma}$, $\widetilde{\mathbf{U}}$ ($\in R^{\mu\times\kappa}$) is the matrix produced by retaining only the κ column vectors of $\mathbf{U}$, each corresponding to the κ largest singular values, and $\widetilde{\mathbf{V}}$ ($\in R^{\nu\times\kappa}$) is the matrix that is similarly produced from $\mathbf{V}$. Finally, using $\widetilde{\mathbf{W}}$, we can obtain the following approximation of $\mathbf{W}$:

$$\widetilde{\mathbf{W}} = \widetilde{\mathbf{U}}\widetilde{\mathbf{\Sigma}}\widetilde{\mathbf{V}}^{\mathrm{T}} \approx \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}} = \mathbf{W}. \tag{3.19}$$

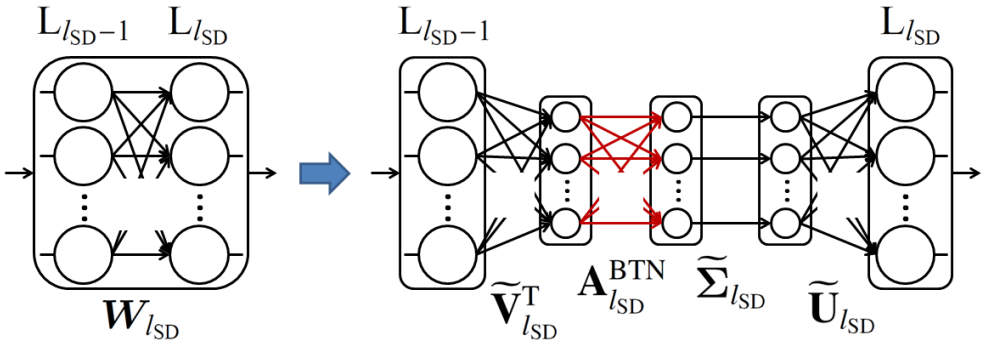


Figure 3.8: Adaptation procedure consisting of SVD-based weight matrix-size reduction and bottleneck LTN insertion.

Eq. (3.19) shows that the space spanned by the column vectors of $\mathbf{W}$ can be approximated by the κ basis vectors of $\widetilde{\mathbf{U}}$.

3.4.4 Adaptation Procedure using Bottleneck Linear Transformation Networks

From the results of Section 3.4.2 and 3.4.3, we propose a new adaptation scheme, which adopts a size-reduced LTN SD module, for improving the SAT-DNN-LTN recognizer. The proposed scheme is illustrated in Figure 3.8 and formalized as follows:

(i) It first conducts the SAT-based training of the entire DNN, based on the procedure as described in Section 3.3.3.

(ii) It next approximates, with the SVD-based matrix decomposition, the weight matrix of SD module as $\overline{\mathbf{W}}_{l_{SD}} \approx \widetilde{\mathbf{U}}_{l_{SD}} \widetilde{\mathbf{\Sigma}}_{l_{SD}} \widetilde{\mathbf{V}}^{T}_{l_{SD}}$. Only the κ basis vectors, which correspond to the κ largest singular values of $\overline{\mathbf{W}}_{l_{SD}}$, are selected in $\widetilde{\mathbf{\Sigma}}_{l_{SD}}$, and therefore $\overline{\mathbf{W}}_{l_{SD}}$ is approximately replaced by the bottleneck network $\widetilde{\mathbf{U}}_{l_{SD}} \widetilde{\mathbf{\Sigma}}_{l_{SD}} \widetilde{\mathbf{V}}^{T}_{l_{SD}}$.

(iii) It inserts a new small-sized LTN into the bottleneck SD module layer whose size (number of nodes) is reduced from ν to κ. This situation is illustrated in the right-side picture in Figure 3.8. To distinguish this small-sized LTN from the LTN originally used in the SAT-DNN-LTN method, we refer to this newly inserted LTN as *bottleneck LTN*, and denote it

by weight matrix $\mathbf{A}_{l_{\mathrm{SD}}}^{\mathrm{BTN}}$ and bias vector $\mathbf{a}_{l_{\mathrm{SD}}}^{\mathrm{BTN}}$. Accordingly, we also refer to the layer into which the bottleneck LTN is inserted as *bottleneck layer*, with *bottleneck size* for κ. Then, the outputs from SD module layer $\mathrm{L}_{l_{\mathrm{SD}}}$ are given as follows:

$$\mathbf{z}_{l_{\mathrm{SD}}} = \phi\Big(\widetilde{\mathbf{U}}_{l_{\mathrm{SD}}}\widetilde{\mathbf{\Sigma}}_{l_{\mathrm{SD}}}(\mathbf{A}_{l_{\mathrm{SD}}}^{\mathrm{BTN}}\widetilde{\mathbf{V}}_{l_{\mathrm{SD}}}^{\mathrm{T}}\mathbf{z}_{l_{\mathrm{SD}}-1} + \mathbf{a}_{l_{\mathrm{SD}}}^{\mathrm{BTN}}) + \mathbf{b}_{l_{\mathrm{SD}}}\Big). \tag{3.20}$$

(iv) It finally adapts only the bottleneck LTN using the speech data of a target speaker.

In the above scheme definition, we inserted the bottleneck LTN between $\widetilde{\mathbf{V}}_{l_{\mathrm{SD}}}^{\mathrm{T}}$ and $\widetilde{\mathbf{\Sigma}}_{l_{\mathrm{SD}}}$ (See Figure 3.8 and Eq. (3.20)). However, in principle, it can also be inserted between $\widetilde{\mathbf{\Sigma}}_{l_{\mathrm{SD}}}$ and $\widetilde{\mathbf{U}}_{l_{\mathrm{SD}}}$. We pre-experimentally investigated both ways of insertion and found that the way of (3.20) worked more stably than its counterpart. Therefore, we adopt it in this section.

3.4.5 Speaker Adaptive Training-based Retraining using Bottleneck Linear Transformation Networks

Based on the definition of bottleneck LTN, it is expected to fundamentally retain the same degree of feature-representation capability as the original weight matrix $\overline{\mathbf{W}}_{l_{\mathrm{SD}}}$ has, provided its size (i.e., bottleneck size) exceeds a certain necessary level. However, removing the basis vectors, which correspond to small singular values, possibly makes the capability of $\widetilde{\mathbf{U}}_{l_{\mathrm{SD}}}\widetilde{\mathbf{\Sigma}}_{l_{\mathrm{SD}}}\mathbf{A}_{l_{\mathrm{SD}}}^{\mathrm{BTN}}\widetilde{\mathbf{V}}_{l_{\mathrm{SD}}}^{\mathrm{T}}$ lower than that of $\overline{\mathbf{W}}_{l_{\mathrm{SD}}}\mathbf{A}_{l_{\mathrm{SD}}}$; this is because if those singular values are not truly zero, their corresponding basis vectors actually contribute to widening the adaptable parameter space. Therefore, the simple size reduction achieved by letting only the large singular value basis vectors remain may not necessarily be sufficient for initializing the bottleneck LTN for the successive adaptation; moreover, if needed, some countermeasures should be added to solve this insufficiency. One possible solution is to re-conduct the SAT-based training of the entire DNN part between (ii) and (iii) in our scheme shown above. The SAT-based optimization procedure here is basically the same as that in the preceding SAT-DNN-LTN method, with the only difference being that the training here involves the small-sized bottleneck LTNs. Note that, similar to the full-sized

LTN in the SAT-DNN-LTN method, the bottleneck LTN is switched along with the speech data selection for every training speaker. This additional SAT procedure will correct the degradation of the SAT concept caused by the bottleneck LTN insertion that deteriorates the SAT-optimized SD module.

3.4.6 Experimental Conditions

We tested our proposed method on the difficult, lecture speech data of the TED Talks corpus. The experimental conditions basically follow those described in Section 3.2.4 unless otherwise stated.

Evaluated Recognizers

For comparison purposes, following our previous training procedures (Section 3.3.3), we first developed the baseline SI recognizer and the SAT-LTN recognizer that had no SVD-based bottleneck layer. Then, using the above SI and SAT-LTN recognizers as baselines, we also developed the following recognizers:

1. SI-BTN: developed by replacing the SD module layer of SI recognizer with an SVD-based low-rank weight matrix.
2. SI-BTN-RET: developed by reapplying the SI training to SI-BTN.
3. SAT-BTN: developed by replacing the SD module layer of SAT-LTN recognizer with an SVD-based low-rank weight matrix.
4. SAT-BTN-RESAT: developed by reapplying the SAT-based training to SAT-BTN, as described in 3.4.5.
5. SA-SI-BTN, SA-SI-BTN-RET[4], SA-SAT-BTN, and SA-SAT-BTN-RESAT: Prefix SA indicates that the corresponding recognizers were speaker-adapted. The adaptation was done

[4]SA-SI-BTN-RET basically corresponds to the recognizer in [47], although there are such small differences as the positioning of the SD modules.

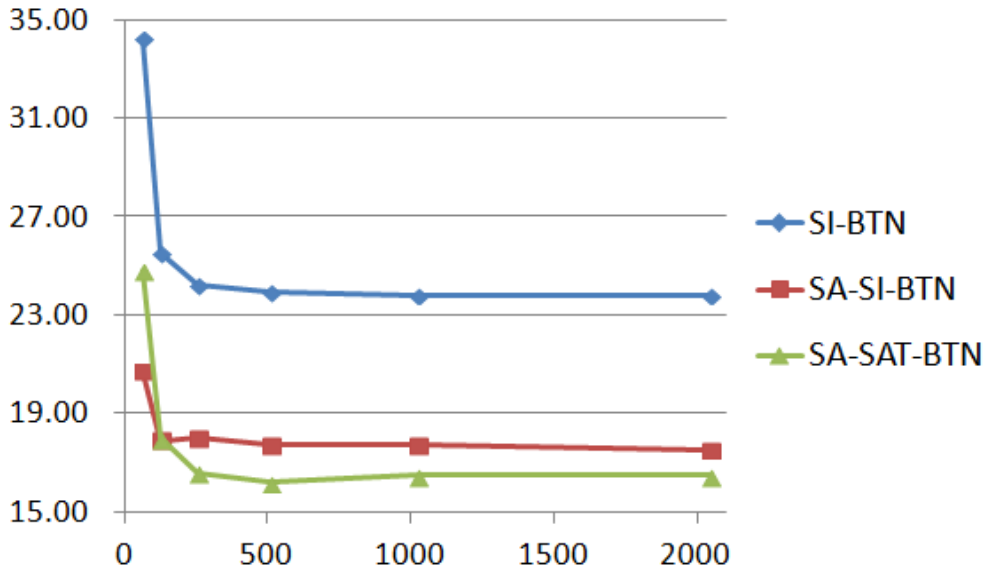


Figure 3.9: Relationship between bottleneck size and recognition performance (word error rate [%]).

using the bottleneck LTN inserted into the SD module layer.

In this experiment, we adopted the larger-scale DNN module, which had seven layers and used 429 input nodes, 4909 output nodes, and 2048 nodes for all of the intermediate layers. In addition, we allocated the SD module only to the layer L_2 ($l_{SD} = 2$), which was shown to be the most effective for the SD module insertion (Section 3.3.6).

3.4.7 Experimental Results

Property Analysis of LTN-based Adaptation

Figure 3.9 shows the WERs of three types of recognizers, i.e., SI-BTN, SA-SI-BTN, and SA-SAT-BTN. The rates were obtained along with the different bottleneck sizes of 64, 128, 256, 512, 1024, and 2048; each of the rates was the average of the results obtained through the CV paradigm over the speech data of all 28 testing speakers. The WERs of SA-SI-BTN and SA-SAT-BTN, at the right-side of the figure, were exactly the same as those of SA-SI-LTN and SA-SAT-

LTN (for these two, the speaker adaptation was done without SVD-based matrix conversion), respectively. Note that in these cases, κ was kept to 2048, although $\overline{\mathbf{W}}_2$ was decomposed based on SVD. As theoretically expected, SVD did not change the feature-representation capability of the matrix when all of the decomposed basis vectors were maintained. Figure 3.9 also shows that the original 2048-dimensional SD module matrix $\overline{\mathbf{W}}_2$ can be sufficiently approximated by such low-rank matrices as the 512-dimensional matrix: The WERs of all three recognizers had little degradation until their bottleneck sizes were reduced to 512. The results clearly support our analysis in Section 3.4.2 that the effective rank of $\overline{\mathbf{W}}_2$ is rather low and the bottleneck LTN having an appropriately selected size can sufficiently retain the adaptation capability of the original high-dimension matrix. For ease of viewing, we omit the results of the other recognizers shown in the list of Section 3.4.6 but they showed the same trend as in Figure 3.9.

Adaptation Using Small Amount of Speech Data

As discussed in Section 3.4.2 and 3.4.5, the use of the bottleneck LTN may have a conflicting two-side effect. Its adaptable parameter reduction possibly increases the adaptability of SD modules, especially over the limited amount of adaptation data, but it also raises the danger of fundamentally decreasing the adaptability. To clarify this effect, we conducted adaptation experiments using two different sizes of bottleneck LTNs: one with $\kappa = 512$ and the other with $\kappa = 2048$. From the above experiment, the bottleneck LTN with $\kappa = 512$ was shown to retain the representation capability of the original 2048-dimensional weight matrix $\overline{\mathbf{W}}_2$, while the bottleneck LTN with $\kappa = 2048$ was also shown to be equivalent to $\overline{\mathbf{W}}_2$. Note that the bottleneck LTN with $\kappa = 2048$ was produced with SVD but actually its structure was not a bottleneck. The experiment was conducted over the speech data, each longer than 6 minutes, of 18 testing speakers; for each testing speaker, his/her first 3 minutes of utterances were reserved for adaptation, and the data of the remaining time were used for testing.

Table 3.8 shows the size conditions and gained WERs of the evaluated recognizers. In the table, for each recognizer, we show the bottleneck size (BTN size), the number of adaptable parameters of the SD module (# Param.), and the WERs for 4 different lengths of adaptation speech data,

Table 3.8: Experimental results (parameter size and word error rate [%]) obtained with supervised adaptation using different amounts of adaptation data.

rRcognizer	BTN size	# Param.	180 s	60 s	30 s	15 s
SI-BTN	2048	4.2 M	24.0	24.0	24.0	24.0
SI-BTN	512	0.26 M	24.1	24.1	24.1	24.1
SA-SI-BTN	2048	4.2 M	18.8	20.3	21.1	21.5
SA-SI-BTN	512	0.26 M	18.9	20.4	21.3	21.7
SA-SAT-BTN	2048	4.2 M	17.8	19.3	20.5	21.3
SA-SAT-BTN	512	0.26 M	18.0	19.5	**20.4**	21.3
SI-BTN-RET	512	0.26 M	24.0	24.0	24.0	24.0
SA-SI-BTN-RET	512	0.26 M	18.9	20.2	21.3	21.7
SA-SAT-BTN-RESAT	512	0.26 M	**17.6**	**18.9**	**20.4**	**20.8**

i.e., 180, 60, 30, and 15 seconds. Note that the length of testing speech data was set to the fixed value described in the above paragraph. Because the SI-BTN and SI-BTN-RET recognizers did not have the speaker adaptation mechanism, the same WERs gained without the adaptation are listed in all of the corresponding columns.

From the table, we can make three observations:

(i) The SAT-based recognizer retrained with SAT (SA-SAT-BTN-RESAT with $\kappa = 512$) achieved the best WERs for all of the adaptation speech data lengths, although the number of the adaptation parameters was only 6.25% of the original 2048-dimensional weight matrix.

(ii) The SAT-based recognizer (SA-SAT-BTN) constantly outperformed its counterpart SI-based recognizer (SA-SI-BTN) for all of the settings of adaptation speech data lengths and bottleneck sizes.

(iii) The SD module size reduction ($\kappa = 2048 \rightarrow \kappa = 512$) basically decreases the adaptation capability of its baseline non-reduced module in both the SI- and SAT-based recognizers.

Similarly to our approach, SVD-based matrix approximation has been studied in recent works with the aim of reducing the adaptation parameter size [47, 66, 67]. However, it was not fully clarified whether adaptation incorporating the parameter-size reduction achieved greater adaptation performance compared to adaptation without such size reduction, especially in cases where the speech data available for adaptation training were severely limited. Our results show that a simple use of SVD-based size reduction is not sufficient for improving this performance, but our novel SAT-based reinitialization procedure for the DNN front-end clearly helps the SAT-based bottleneck LTN to boost its adaptation power. In addition to the SAT-based reinitialization, we actually examined the SI retraining for the SI-BTN recognizer, but this alternative reinitialization was not effective for increasing the adaptation performances of the SA-SI-BTN recognizer. The difference in reinitialization effect probably arises from the training difference that the SAT-based approach reinitializes the bottleneck LTN together with the other DNN part in order to increase the adaptation capability of the LTN module but the SI-based approach gives no consideration to the adaptation.

3.5 Summary

To handle the challenging sample finiteness problem, we studied speaker adaptation techniques for the hybrid DNN-HMM speech recognizer and proposed new SAT-based speaker adaptation algorithms that introduce modularity in the DNN part. By utilizing the SAT concept for the DNN training, the proposed methods allow to increase the adaptability of the DNN, while reducing the size of adaptable parameters. Our experimental results on a difficult TED Talks corpus show that the proposed SAT-based recognizer with small-size SD modules outperformed the baseline SI-based recognizer with large-size SD modules.

4 Multichannel End-to-end Speech Recognition

4.1 Introduction

In the past five years or so, with the advent of DL techniques in ASR tasks, a hybrid DNN-HMM framework has significantly improved ASR performances compared to a conventional GMM-HMM framework [8, 9, 10]. Although such DL-based approaches have replaced several components of the conventional ASR system, current systems continue to adopt a complicated module-based architecture that consists of several separate components, such as acoustic, phonetic, and language models. To build such a complicated architecture, we require wide and deep knowledge about each component, which makes it difficult to develop and tune ASR systems for every applications.

Recently, as an alternative to such complicated architecture, an end-to-end ASR framework has attracted great research interest because it simplifies the above architecture with a single neural network-based architecture [13, 14, 16, 68, 69, 70, 71, 72, 73, 74]. One of promising directions is an attention-based encoder-decoder framework, which integrates all relevant components using RNNs and an attention mechanism [13, 14, 68, 69, 70, 71, 72]. Using the attention mechanism, the framework deals with dynamic time alignment problems within the network and solves the ASR problem as a sequence-to-sequence mapping problem from acoustic feature to word/character label sequences. In addition to the simplified system architecture, another important motivation of the end-to-end framework is that the entire inference procedure can be consis-

tently optimized to improve such final ASR objectives as word/character error rate (WER/CER).

However, previous research on end-to-end frameworks mainly focused on the ASR problem in a single-channel setup without speech enhancement. Considering real world applications, we must also study such frameworks in a multichannel setup with speech enhancement. Actually, recent benchmark studies show that multichannel processing with microphone-array speech enhancement techniques (especially beamforming methods) produces substantial improvements in the presence of strong background noise for conventional DNN-HMM hybrid systems [75, 76].

In light of the above trends, we extend the existing attention-based encoder-decoder framework by integrating multichannel speech enhancement components into the end-to-end framework and propose an ME2E ASR architecture, which directly converts multichannel speech signal to text through speech enhancement. As a speech enhancement component of our ME2E ASR system, we adopt a recently proposed beamforming technique using neural networks, which we call a neural beamformer. Because a neural beamformer can be formalized as a fully differentiable network, the beamforming component can be jointly optimized with the end-to-end ASR component, based on the backpropagated gradients from the final ASR objective.

Recent studies on neural beamformers can be categorized into two types: 1) beamformers with a filter estimation network [77, 78, 79] and 2) those with a mask estimation network [80, 81, 82, 83, 84, 85, 86, 87]. Motivated by the successes of the mask-based beamforming approaches [80, 81, 82, 88] in recent noisy ASR benchmarks (e.g., CHiME 3 and 4 challenges), we mainly focus on the mask-based neural beamformer.

Our mask-based neural beamformer adopts an MVDR formalization given a reference microphone [89] since computing the derivatives is relatively simple. We also propose an additional neural network-based attention mechanism for the reference microphone selection. This allows the entire procedures of the neural beamformer, including the reference selection, to be invariant to microphone geometries including the number of channels, the microphone locations, and the microphone ordering. Therefore, our proposed ME2E ASR architecture can deal with input signals from various microphone geometries without re-configuration and re-training. Of

course, because the channel attention mechanism is also formalized as a differentiable network, the entire procedures of the neural beamformer can be jointly optimized with the end-to-end ASR component based on the backpropagated gradients from the end-to-end ASR objective.

The notations used in this chapter are summarized in Appendix A.2.

4.2 A Unified Architecture for Multichannel End-to-end Speech Recognition with Neural Beamforming

4.2.1 Overview

Figure 4.1 illustrates an overview of the proposed ME2E ASR architecture. The architecture mainly consists of two building blocks: 1) neural beamformers for speech enhancement, and 2) attention-based encoder-decoder networks for speech recognition. Let $X_c = \{\mathbf{x}_{t,c} \in \mathbb{C}^F | t = 1, \cdots, T\}$ be a short-time Fourier transform (STFT) feature sequence recorded at c-th channel, where $\mathbf{x}_{t,c}$ is a F-dimensional STFT feature vector at input time step t, and T is the input sequence length, and let C be the number of channels. Given multichannel input speech sequences $\{X_c\}_{c=1}^C$, the ME2E ASR architecture directly estimates the posterior probabilities for output label sequence $Y = \{y_n \in \mathcal{V} | n = 1, \cdots, N\}$ using the fully neural network-based architecture, where y_n is a label symbol (e.g., character) at output time step n, N is the output sequence length, and $\mathcal{V}$ is a set of labels. First in the neural beamformer stage, the system integrates the multichannel noisy sequences $\{X_c\}_{c=1}^C$ into a single-channel (hopefully) noise-suppressed sequence $\hat{X}$ by linear filtering. Next in the feature extraction stage, it converts the filtered STFT feature sequence $\hat{X}$ that are output from the front-end neural beamformer to a log Mel filterbank feature sequence $\hat{O}$ for inputting to the back-end attention-based encoder-decoder network. Finally in the attention-based encoder-decoder stage, it transforms (decodes) the log Mel filterbank feature sequence $\hat{O}$ to the output label (e.g., character) sequence Y by the estimation of the posterior probabilities $P(Y|\{X_c\}_{c=1}^C)$.

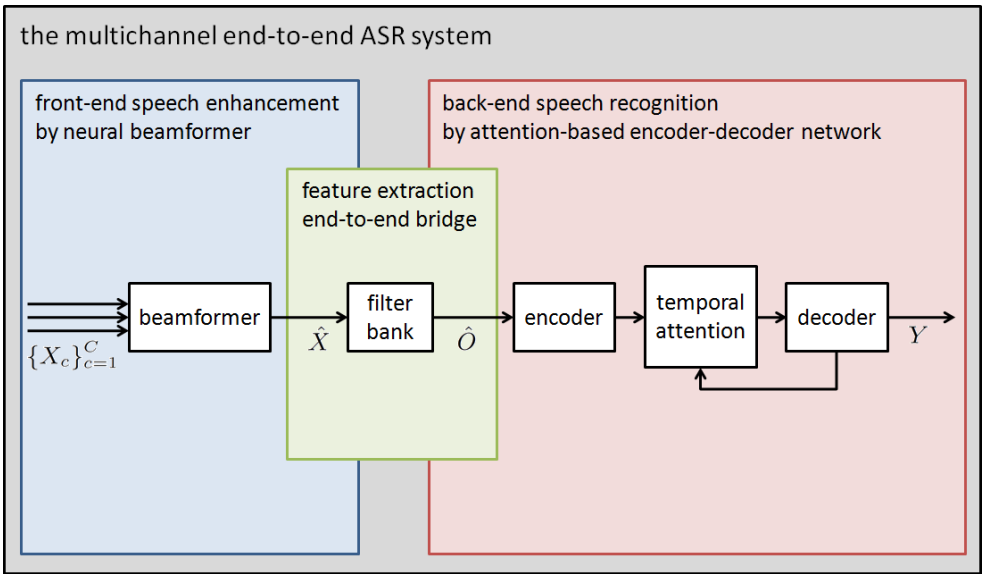


Figure 4.1: Overview of the proposed ME2E ASR architecture. The neural beamformer works as a speech enhancement part and the attention-based encoder-decoder network works as an ASR part, where feature extraction function connects those components.

The entire procedure of the ME2E ASR system can be represented in the following functional forms:

$$\hat{X} = \text{Beamformer}(\{X_c\}_{c=1}^C), \tag{4.1}$$

$$\hat{O} = \text{Feature}(\hat{X}), , \tag{4.2}$$

$$P(Y|\{X_c\}_{c=1}^C) = \text{E2E\_ASR}(\hat{O}), \tag{4.3}$$

where, Beamformer$(\cdot)$ is a speech enhancement function realized by the neural beamformer with the filter estimation network (Section 4.4.2) or the mask estimation network (Section 4.4.3). Feature$(\cdot)$ is a feature extraction function realized by the log Mel filterbank (Section 4.3), which bridges the speech enhancement and end-to-end ASR components. E2E_ASR$(\cdot)$ is an end-to-end speech recognition function realized by the attention-based encoder-decoder networks (Section 4.5). The details of each function are described in the following Sections.

Note that because all of the procedures (i.e., speech enhancement, feature extraction, and end-to-

end speech recognition) are represented as differentiable neural network-based architecture, the entire recognition process can be optimized only using the training sample pairs, each of which consists of multichannel noisy speech samples and its corresponding transcription, to satisfy the end-to-end ASR objective (i.e., generating a correct label sequence) as much as possible.

4.2.2 Training Objective

To learn appropriate time-alignments in the presence of strong background noise, a joint CTC-attention loss [71] are adopted for our end-to-end ASR objective. It is a kind of a multi-task learning objective, which utilized two types of loss function: one for encoder-decoder loss and another for CTC loss. Because CTC loss imposes a left-to-right constraint on the time-alignment, it helps the encoder network and the attention mechanism learn appropriate time-alignments even in the presence of strong background noise.

To define the joint CTC-attention loss, a CTC decoder is added to the encoder's top layer, where the encoder is shared by the attention-based and CTC decoders. Let $P_{\mathrm{ATT}}(Y|\mathbf{X})$ be the posteriors estimated by the attention-based encoder-decoder, $P_{\mathrm{CTC}}(Y|\mathbf{X})$ be the posteriors estimated by the CTC. Then joint CTC-attention loss $E_{\mathrm{JOINT}}(Y|\mathbf{X})$ is formalized as follows:

$$E_{\mathrm{JOINT}}(Y|\mathbf{X}) = \gamma E_{\mathrm{ATT}}(Y|\mathbf{X}) + (1-\gamma)E_{\mathrm{CTC}}(Y|\mathbf{X}), \tag{4.4}$$

where

$$E_{\mathrm{ATT}}(Y|\mathbf{X}) = -\log P_{\mathrm{ATT}}(Y|\mathbf{X}), \tag{4.5}$$

$$E_{\mathrm{CTC}}(Y|\mathbf{X}) = -\log P_{\mathrm{CTC}}(Y|\mathbf{X}), \tag{4.6}$$

$\mathbf{X} = \{X_c\}_{c=1}^{C}$ be a set of multichannel noisy speech inputs, Y is its corresponding target transcription, and $\gamma \in [0, 1]$ is an interpolation weight.

Because the computational processes of $P_{\mathrm{ATT}}(Y|\mathbf{X})$ and $P_{\mathrm{CTC}}(Y|\mathbf{X})$ are fully differentiable, the

optimization process can be conducted based on arbitrary gradient-based optimization algorithms based on the backpropagation through time method [90].

4.3 Feature Extraction Component : Log Mel Filterbank

This section explains our adopted feature extraction function, which bridges the speech enhancement and end-to-end ASR components. Motivated by the success of (log) Mel filterbank-based features in previous studies (e.g., a single-channel end-to-end ASR setup [13, 69], a single-channel joint training setup of speech enhancement and DNN-HMM hybrid system [91], and a multichannel DNN-HMM hybrid setup [77, 84]), a normalized log Mel filterbank are adopted as an inner feature representation for the ME2E ASR architecture.

In other words, Feature(·) transforms the enhanced STFT coefficients that are output from the front-end neural beamformer to the enhanced acoustic feature (i.e., normalized log Mel filterbank) for inputting to the back-end attention-based encoder-decoder network. Concretely, enhanced acoustic feature $\hat{\mathbf{o}}_t \in \mathbb{R}^{D_\mathrm{O}}$ was obtained from enhanced STFT coefficients $\hat{\mathbf{x}}_t \in \mathbb{C}^F$ as follows:

$$\hat{\mathbf{p}}_t = \{\Re(\hat{x}_{t,f})^2 + \Im(\hat{x}_{t,f})^2\}_{f=1}^{F}, \tag{4.7}$$

$$\hat{\mathbf{o}}_t = \mathrm{Norm}(\log(\mathrm{Mel}(\hat{\mathbf{p}}_t))), \tag{4.8}$$

where $\hat{\mathbf{p}}_t \in \mathbb{R}^F$ is a real-valued vector of the power spectrum of the enhanced signal at time step t, and $\hat{x}_{t,f} \in \mathbb{C}$ is an enhanced STFT coefficient at time-frequency bin (t, f). Mel(·) represents the operation of $D_\mathrm{O} \times F$ Mel matrix multiplication, and Norm(·) represents the operation of $D_\mathrm{O} \times D_\mathrm{O}$ global mean and variance normalization so that the mean and variance of each dimension become 0 and 1. Eqs. (4.7)-(4.8) correspond to Feature(·) in Eq. (4.2).

Note that since computation of the normalized log Mel filterbank is fully differentiable, the gradients from the speech recognition part (i.e., the attention-based encoder-decoder net- work) can

be backpropagated to the speech enhancement part (i.e., the neural beamformer)

4.4 Speech Enhancement Component : Neural Beamformers

4.4.1 Overview

This section explains neural beamformer techniques, which are integrated with the encoder-decoder network in the proposed ME2E ASR architecture. In the proposed architecture, frequency-domain beamformers [77] are adopted rather than time-domain ones [78], because the frequency-domain beamformers achieve significant computational complexity reduction in multichannel neural processing [92].

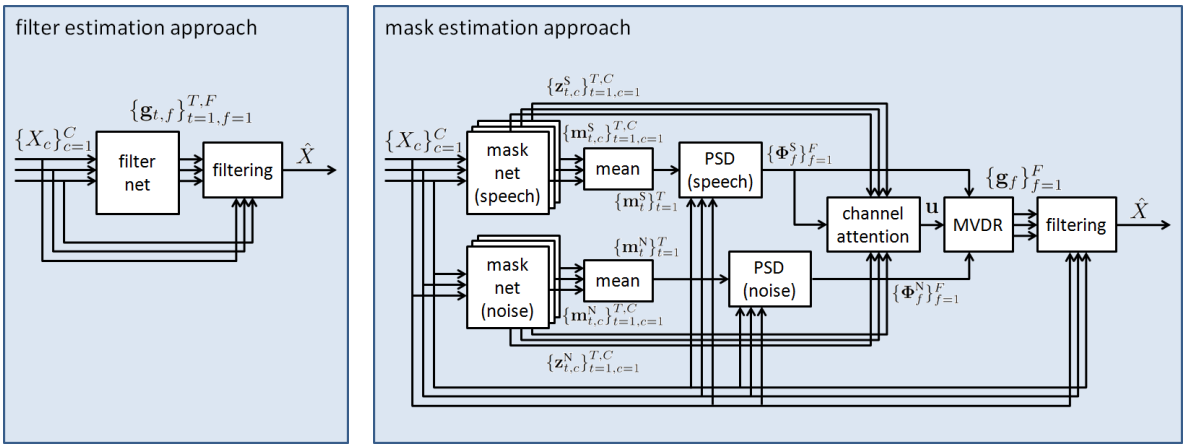
Let $\mathbf{x}_{t,f} = \{x_{t,f,c}\}_{c=1}^{C} \in \mathbb{C}^C$ be the spatial vector of the signals obtained from all the microphones for each time-frequency bin (t, f), where $x_{t,f,c} \in \mathbb{C}$ be an STFT coefficient of c-th channel noisy signal at time-frequency bin (t, f). Moreover, let $\mathbf{g}_{t,f} = \{g_{t,f,c}\}_{c=1}^{C} \in \mathbb{C}^C$ and $\mathbf{g}_f = \{g_{f,c}\}_{c=1}^{C} \in \mathbb{C}^C$ be corresponding *time-variant* and *time-invariant* filter coefficients, respectively. In the frequency domain representation, enhanced STFT coefficient $\hat{x}_{t,f}$ are obtained by linear filtering as follows:

$$\hat{x}_{t,f} = \begin{cases} \mathbf{g}_{t,f}^{\dagger} \mathbf{x}_{t,f} & \text{(time-variant filter)} \\ \mathbf{g}_{f}^{\dagger} \mathbf{x}_{t,f} & \text{(time-invariant filter),} \end{cases} \tag{4.9}$$

where † represents the conjugate transpose.

In the proposed architecture, two types of neural beamformers that basically follow Eq. (4.9) are adopted as the speech enhancement component: 1) with a filter estimation network and 2) with a mask estimation network. Figure 4.2 illustrates an overview of both approaches. The main difference between them is how to estimate the filter coefficients: $\mathbf{g}_{t,f}$ or $\mathbf{g}_f$. The following subsections describe each approach.

Note that the entire beamforming procedures described in the following subsections correspond



(a) Beamforming with filter estimation network

(b) Beamforming with mask estimation network

Figure 4.2: Structures of neural beamformers: (a) filter estimation network approach, which directly estimates the filter coefficients; (b) mask estimation network approach, which estimates time-frequency masks and gets filter coefficients based on MVDR formalization.

to Beamformer(·) in Eq. (4.1).

4.4.2 Neural Beamforming with Filter Estimation Network

A neural beamformer with a filter estimation network directly estimates time-variant filter coefficients $\{\mathbf{g}_{t,f}\}_{t=1,f=1}^{T,F}$ as network outputs, where F is the number of dimensions of the STFT signals.

The following Algorithm 1 summarizes the overall procedures to obtain the enhanced features, and Figure 4.2(a) illustrates an overview of the procedures. The main part of this algorithm is to predict complex-valued filter coefficients $\mathbf{g}_{t,f}$ with a real-valued neural network, Filternet(·), which is described below.

Filter Estimation Network

This approach uses a single real-valued bidirectional long short-term memory (BLSTM) recurrent network [93, 94] to predict the real and imaginary parts of the complex-valued filter coef-

Algorithm 1 Overall procedures of neural beamformer with filter estimation network

Require: multichannel STFT input sequences $\{X_c\}_{c=1}^{C}$

1: $\{\mathbf{g}_{t,f}\}_{t=1,f=1}^{T,F} = \text{Filternet}(\{X_c\}_{c=1}^{C})$ ▷ Eqs. (4.10)-(4.12)
2: **for** $t = 1$ to T **do**
3: **for** $f = 1$ to F **do**
4: $\hat{x}_{t,f} = \mathbf{g}_{t,f}^{\dagger}\mathbf{x}_{t,f}$ ▷ Eq. (4.9)
5: **end for**
6: **end for**
7: return $\hat{X} = \{\hat{x}_{t,f}\}_{t=1,f=1}^{T,F}$

ficients $\{\mathbf{g}_{t,f}\}_{f=1}^{F}$ at every time step. We introduce $2FC$-dimensional output layers to separately compute the real and imaginary parts of the filter coefficients.

Let $\tilde{\mathbf{x}}_t = \{\Re(\mathbf{x}_{t,f}), \Im(\mathbf{x}_{t,f})\}_{f=1}^{F} \in \mathbb{R}^{2FC}$ be an input feature of a $2FC$-dimensional real-valued vector for the BLSTM network, which is obtained by concatenating the real and imaginary parts of all STFT coefficients in all channels at time step t. Given input sequence $\tilde{X} = \{\tilde{\mathbf{x}}_t | t = 1, \cdots, T\}$, the network outputs time-variant filter coefficients $\mathbf{g}_{t,f}$ as follows:

$$Z = \text{BLSTM}(\tilde{X}), \tag{4.10}$$

$$\Re(\mathbf{g}_{t,f}) = \tanh(\mathbf{W}_f^{\Re}\mathbf{z}_t + \mathbf{b}_f^{\Re}), \tag{4.11}$$

$$\Im(\mathbf{g}_{t,f}) = \tanh(\mathbf{W}_f^{\Im}\mathbf{z}_t + \mathbf{b}_f^{\Im}), \tag{4.12}$$

where $Z = \{\mathbf{z}_t \in \mathbb{R}^{D_Z} | t = 1, \cdots, T\}$ is a sequence of the D_Z-dimensional output vectors of the BLSTM network. $\Re(\mathbf{g}_{t,f})$ and $\Im(\mathbf{g}_{t,f})$ are the real and imaginary parts of the filter coefficients. $\mathbf{W}_f^{\Re} \in \mathbb{R}^{C \times D_Z}$ and $\mathbf{W}_f^{\Im} \in \mathbb{R}^{C \times D_Z}$ are the weight matrices that output real and imaginary part of the filter coefficients for frequency f, and $\mathbf{b}_f^{\Re} \in \mathbb{R}^{C}$ and $\mathbf{b}_f^{\Im} \in \mathbb{R}^{C}$ are their corresponding bias vectors. Eqs. (4.10)-(4.12) correspond to Filternet($\cdot$) in Algorithm 1. Using estimated filters $\{\mathbf{g}_{t,f}\}_{t=1,f=1}^{T,F}$, the enhanced STFT coefficients $\{\hat{x}_{t,f}\}_{t=1,f=1}^{T,F}$ are obtained based on Eq. (4.9).

Remarks

This approach has several possible issues due to its formalization. The first issue is the high flexibility of estimated filters $\{\mathbf{g}_{t,f}\}_{t=1,f=1}^{T,F}$, which are composed of a large number of unconstrained variables ($2TFC$) estimated from few observations. This causes problems, such as training difficulties and over-fitting. The second is that the network structure depends on the number and order of the channels. Therefore, a new filter estimation network has to be trained when we change microphone configurations.

4.4.3 Neural Beamforming with Mask Estimation Network

The neural beamformer with a mask estimation network first estimates the time-frequency masks. Then cross-channel power spectral density (PSD) matrices (also known as spatial covariance matrices) are predicted based on the estimated masks. Finally, they are used to compute the time-invariant filter coefficients $\{\mathbf{g}_f\}_{f=1}^{F}$ based on the well-studied MVDR formalization.

The key point of the mask-based neural beamformer is that it constrains the estimated filters based on well-founded array signal processing principles, which can solve/suppress the issues described in Section 4.4.2. This point is the main difference between the mask estimation network approach and the filter estimation network approach described in Section 4.4.2. Also, mask-based beamforming approaches have achieved great performance in recent noisy ASR benchmarks [80, 81, 82, 88]. Motivated by this background, we focus on a neural beamformer with a mask estimation network more than a filter estimation network.

Algorithm 2 summarizes the overall procedures to obtain the enhanced features, and Figure 4.2(b) illustrates an overview of the procedures[1]. Each procedure is described below.

[1]Due to space limitations, the procedures corresponding to State_Feat(·) and Spatial_Feat(·) in Algorithm 2 are not shown in Figure 4.2(b).

Algorithm 2 Overall procedures of neural beamformer with mask estimation network

Require: multichannel STFT input sequences $\{X_c\}_{c=1}^{C}$

for $c = 1$ to C **do**
  $\{\mathbf{m}_{t,c}^{\mathrm{S}}\}_{t=1}^{T}, \{\mathbf{z}_{t,c}^{\mathrm{S}}\}_{t=1}^{T} = \mathrm{Masknet}^{\mathrm{S}}(X_c)$ ▷ Eqs. (4.16)-(4.17)
  $\{\mathbf{m}_{t,c}^{\mathrm{N}}\}_{t=1}^{T}, \{\mathbf{z}_{t,c}^{\mathrm{N}}\}_{t=1}^{T} = \mathrm{Masknet}^{\mathrm{N}}(X_c)$ ▷ Eqs. (4.18)-(4.19)
end for
for $t = 1$ to T **do**
  $\mathbf{m}_t^{\mathrm{S}} = \mathrm{Mean}(\{\mathbf{m}_{t,c}^{\mathrm{S}}\}_{c=1}^{C})$ ▷ Eq. (4.20)
  $\mathbf{m}_t^{\mathrm{N}} = \mathrm{Mean}(\{\mathbf{m}_{t,c}^{\mathrm{N}}\}_{c=1}^{C})$ ▷ Eq. (4.21)
end for
for $f = 1$ to F **do**
  $\mathbf{\Phi}_f^{\mathrm{S}} = \mathrm{PSD}(\{\mathbf{m}_t^{\mathrm{S}}\}_{t=1}^{T}, \{\mathbf{x}_{t,f}\}_{t=1}^{T})$ ▷ Eq. (4.14)
  $\mathbf{\Phi}_f^{\mathrm{N}} = \mathrm{PSD}(\{\mathbf{m}_t^{\mathrm{N}}\}_{t=1}^{T}, \{\mathbf{x}_{t,f}\}_{t=1}^{T})$ ▷ Eq. (4.15)
end for
for $c = 1$ to C **do**
  $\mathbf{q}_c = \mathrm{State\_Feat}(\{\mathbf{z}_{t,c}^{\mathrm{S}})\}_{t=1}^{T}, \{\mathbf{z}_{t,c}^{\mathrm{N}})\}_{t=1}^{T})$ ▷ Eq. (4.24)
  $\mathbf{r}_c = \mathrm{Spatial\_Feat}(\{\phi_{f,c,c'}^{S}\}_{f=1,c'=1}^{F,C})$ ▷ Eq. (4.25)
end for
$\mathbf{u} = \mathrm{Attend\_Channel}(\{\mathbf{q}_c\}_{c=1}^{C}, \{\mathbf{r}_c\}_{c=1}^{C})$ ▷ Eqs. (4.22)-(4.23)
for $f = 1$ to F **do**
  $\mathbf{g}_f = \mathrm{MVDR}(\mathbf{\Phi}_f^{\mathrm{S}}, \mathbf{\Phi}_f^{\mathrm{N}}, \mathbf{u})$ ▷ Eq. (4.13)
end for
for $t = 1$ to T **do**
  for $f = 1$ to F **do**
    $\hat{x}_{t,f} = \mathbf{g}_f^{\dagger}\mathbf{x}_{t,f}$ ▷ Eq. (4.9)
  end for
end for
return $\hat{X} = \{\hat{x}_{t,f}\}_{t=1,f=1}^{T,F}$

MVDR formalization given reference microphones

The time-invariant filter coefficients $\mathbf{g}_f$ in Eq. (4.9) are computed based on the MVDR formalization given the reference microphones [89] as follows:

$$\mathbf{g}_f = \frac{(\mathbf{\Phi}_f^{\mathrm{N}})^{-1}\mathbf{\Phi}_f^{\mathrm{S}}}{\mathrm{Tr}((\mathbf{\Phi}_f^{\mathrm{N}})^{-1}\mathbf{\Phi}_f^{\mathrm{S}})}\mathbf{u}, \tag{4.13}$$

where $\mathbf{\Phi}_f^{\mathrm{S}} \in \mathbb{C}^{C\times C}$ and $\mathbf{\Phi}_f^{\mathrm{N}} \in \mathbb{C}^{C\times C}$ are the PSD matrices for speech and noise signals, respectively. $\mathbf{u} \in \mathbb{R}^C$ is a one-hot vector representing a reference microphone, and $\mathrm{Tr}(\cdot)$ represents the matrix trace operation. Eq. (4.13) corresponds to MVDR$(\cdot)$ in Algorithm 2.

Mask-based estimation for PSD matrices

Let $m_{t,f}^{\mathrm{S}} \in [0, 1]$ and $m_{t,f}^{\mathrm{N}} \in [0, 1]$ respectively be the time-frequency masks for speech and noise signals. Based on previous works [80, 88], the PSD matrices can be robustly estimated using the expectation with respect to time-frequency masks as follows:

$$\mathbf{\Phi}_f^{\mathrm{S}} = \frac{1}{\sum_{t=1}^{T} m_{t,f}^{\mathrm{S}}} \sum_{t=1}^{T} m_{t,f}^{\mathrm{S}} \mathbf{x}_{t,f}\mathbf{x}_{t,f}^{\dagger}, \tag{4.14}$$

$$\mathbf{\Phi}_f^{\mathrm{N}} = \frac{1}{\sum_{t=1}^{T} m_{t,f}^{\mathrm{N}}} \sum_{t=1}^{T} m_{t,f}^{\mathrm{N}} \mathbf{x}_{t,f}\mathbf{x}_{t,f}^{\dagger}. \tag{4.15}$$

Eqs. (4.14) and (4.15) correspond to PSD$(\cdot)$ in Algorithm 2.

Mask estimation network

To estimate the time-frequency masks for every c-th channel ($\mathbf{m}_{t,c}^{\mathrm{S}} = \{m_{t,f,c}^{\mathrm{S}}\}_{f=1}^{F}$ and $\mathbf{m}_{t,c}^{\mathrm{N}} = \{m_{t,f,c}^{\mathrm{N}}\}_{f=1}^{F}$), we use two real-valued BLSTM networks: one for a speech mask and another for a noise mask. Unlike the filter estimation network, because the masks are estimated separately for each channel, $2F$-dimensional output layers are used to separately compute the real and

imaginary parts of the time-frequency masks.

Let $\tilde{\mathbf{x}}_{t,c} = \{\Re(x_{t,f,c}), \Im(x_{t,f,c})\}_{f=1}^{F} \in \mathbb{R}^{2F}$ be the $2F$-dimensional real-valued input features for the BLSTM networks, which is obtained by concatenating the real and imaginary parts of all the STFT features at c-th channel. Given input sequence $\tilde{X}_c = \{\tilde{\mathbf{x}}_{t,c} | t = 1, \cdots, T\}$, each network outputs the time-frequency masks separately for each channel as follows:

$$Z_c^{\mathrm{S}} = \mathrm{BLSTM}^{\mathrm{S}}(\tilde{X}_c), \tag{4.16}$$

$$\mathbf{m}_{t,c}^{\mathrm{S}} = \mathrm{sigmoid}(\mathbf{W}^{\mathrm{S}}\mathbf{z}_{t,c}^{\mathrm{S}} + \mathbf{b}^{\mathrm{S}}), \tag{4.17}$$

$$Z_c^{\mathrm{N}} = \mathrm{BLSTM}^{\mathrm{N}}(\tilde{X}_c), \tag{4.18}$$

$$\mathbf{m}_{t,c}^{\mathrm{N}} = \mathrm{sigmoid}(\mathbf{W}^{\mathrm{N}}\mathbf{z}_{t,c}^{\mathrm{N}} + \mathbf{b}^{\mathrm{N}}), \tag{4.19}$$

where $Z_c^{\mathrm{S}} = \{\mathbf{z}_{t,c}^{\mathrm{S}} \in \mathbb{R}^{D_{\mathrm{Z}}} | t = 1, \cdots, T\}$ is a sequence of D_{Z}-dimensional output vectors of the BLSTM network for a speech mask over c-th channel's input sequence $\tilde{X}_c$. Z_c^{N} is the BLSTM output sequence for a noise mask. $\mathbf{W}^{\mathrm{S}} \in \mathbb{R}^{F \times D_{\mathrm{Z}}}$ and $\mathbf{W}^{\mathrm{N}} \in \mathbb{R}^{F \times D_{\mathrm{Z}}}$ are the weight matrices that output speech and noise masks. $\mathbf{b}^{\mathrm{S}} \in \mathbb{R}^{F}$ and $\mathbf{b}^{\mathrm{N}} \in \mathbb{R}^{F}$ are their corresponding bias vectors. Eqs. (4.16) and (4.17) correspond to $\mathrm{Masknet}^{\mathrm{S}}(\cdot)$, while Eqs. (4.18) and (4.19) correspond to $\mathrm{Masknet}^{\mathrm{N}}(\cdot)$ in Algorithm 2.

After computing the speech and noise masks for each channel, mean masks $\mathbf{m}_t = \{m_{t,f}\}_{f=1}^{F}$ are obtained as follows:

$$\mathbf{m}_t^{\mathrm{S}} = \frac{1}{C}\sum_{c=1}^{C} \mathbf{m}_{t,c}^{\mathrm{S}}, \tag{4.20}$$

$$\mathbf{m}_t^{\mathrm{N}} = \frac{1}{C}\sum_{c=1}^{C} \mathbf{m}_{t,c}^{\mathrm{N}}. \tag{4.21}$$

Eqs. (4.20) and (4.21) correspond to $\mathrm{Mean}(\cdot)$ in Algorithm 2. These mean masks are used to predict PSD matrices ($\mathbf{\Phi}_f^{\mathrm{S}}$ and $\mathbf{\Phi}_f^{\mathrm{N}}$) in Eqs. (4.14) and (4.15).

The mask-based MVDR neural beamformer, given reference microphones, was previously proposed in [81, 82], but our neural beamformer further extends it with attention-based reference

selection, which is described in the next subsection.

Attention-based selection for reference microphones

To incorporate the reference microphone selection in the neural beamformer framework, we apply the idea of an attention mechanism to estimate reference microphone vector $\mathbf{u}$ in Eq. (4.13). Based on an attention mechanism's characteristics, this allows the reference selection to work for arbitrary numbers and orders of channels.

To formalize the attention mechanism, we adopt two types of time-invariant and channel-dependent features: 1) time-averaged state feature $\mathbf{q}_c \in \mathbf{R}^{2D_{\mathrm{Z}}}$ and 2) PSD-based spatial feature $\mathbf{r}_c \in \mathbb{R}^{2F}$. With these feature vectors, reference microphone vector $\mathbf{u}$ is estimated as follows:

$$k_c = \mathbf{w}^{\mathrm{T}} \tanh(\mathbf{V}^{\mathrm{Q}} \mathbf{q}_c + \mathbf{V}^{\mathrm{R}} \mathbf{r}_c + \mathbf{b}), \tag{4.22}$$

$$u_c = \frac{\exp(\alpha k_c)}{\sum_{c=1}^{C} \exp(\alpha k_c)}, \tag{4.23}$$

where $\mathbf{w} \in \mathbb{R}^{1 \times D_{\mathrm{W}}}$, $\mathbf{V}^{\mathrm{Q}} \in \mathbb{R}^{D_{\mathrm{W}} \times 2D_{\mathrm{Z}}}$, and $\mathbf{V}^{\mathrm{R}} \in \mathbb{R}^{D_{\mathrm{W}} \times 2F}$ are trainable weight parameters, and $\mathbf{b} \in \mathbb{R}^{D_{\mathrm{W}}}$ is a trainable bias vector. α is the sharpening factor. Eqs. (4.22) and (4.23) correspond to Attend_Channel(·) in Algorithm 2.

Time-averaged state feature $\mathbf{q}_c$ is extracted from the BLSTM networks for the speech and noise masks in Eqs. (4.16) and (4.18) as follows:

$$\mathbf{q}_c = \frac{1}{T} \sum_{t=1}^{T} \{\mathbf{z}_{t,c}^{\mathrm{S}}, \mathbf{z}_{t,c}^{\mathrm{N}}\}, \tag{4.24}$$

Eq. (4.24) corresponds to State_Feat(·) in Algorithm 2.

PSD-based spatial feature $\mathbf{r}_c$, which incorporates the spatial information into the attention mech-

anism, is extracted from speech PSD matrix $\mathbf{\Phi}_f^{\mathrm{S}}$ in Eq. (4.14) as follows:

$$\mathbf{r}_c = \frac{1}{C-1} \sum_{c'=1, c' \neq c}^{C} \{\Re(\phi_{f,c,c'}^{\mathrm{S}}), \Im(\phi_{f,c,c'}^{\mathrm{S}})\}_{f=1}^{F}, \tag{4.25}$$

where $\phi_{f,c,c'}^{S} \in \mathbb{C}$ is the entry in the c-th row and the c'-th column of speech PSD matrix $\mathbf{\Phi}_f^{\mathrm{S}}$. To select a reference microphone, since the spatial correlation related to speech signals is more informative, we only use speech PSD matrix $\mathbf{\Phi}_f^{\mathrm{S}}$ as a feature. Eq. (4.25) corresponds to Spatial_Feat(·) in Algorithm 2.

Note that, in this mask-based MVDR neural beamformer, the masks for each channel are computed separately using the same BLSTM network, and thus the mask estimation network is independent of the channels. Similarly, the reference selection network is also independent of the channels. Therefore, the neural beamformer deals with input signals with arbitrary numbers and orders of channels without network re-training or re-configuration.

4.5 Speech Recognition Component : Attention-based Encoder-decoder Networks

4.5.1 Overview

This section explains a conventional attention-based encoder-decoder framework, which directly deals with variable length input and output sequences. The framework consists of two RNNs, an encoder and a decoder, both of which are connected by an attention mechanism. Figure 4.3 illustrates the overall architecture of the framework.

Based on these components, given a T-length sequence of input features $O = \{\mathbf{o}_t \in \mathbb{R}^{D_{\mathrm{O}}} | t = 1, \cdots, T\}$, the framework directly estimates the posteriors for N-length sequence of output labels

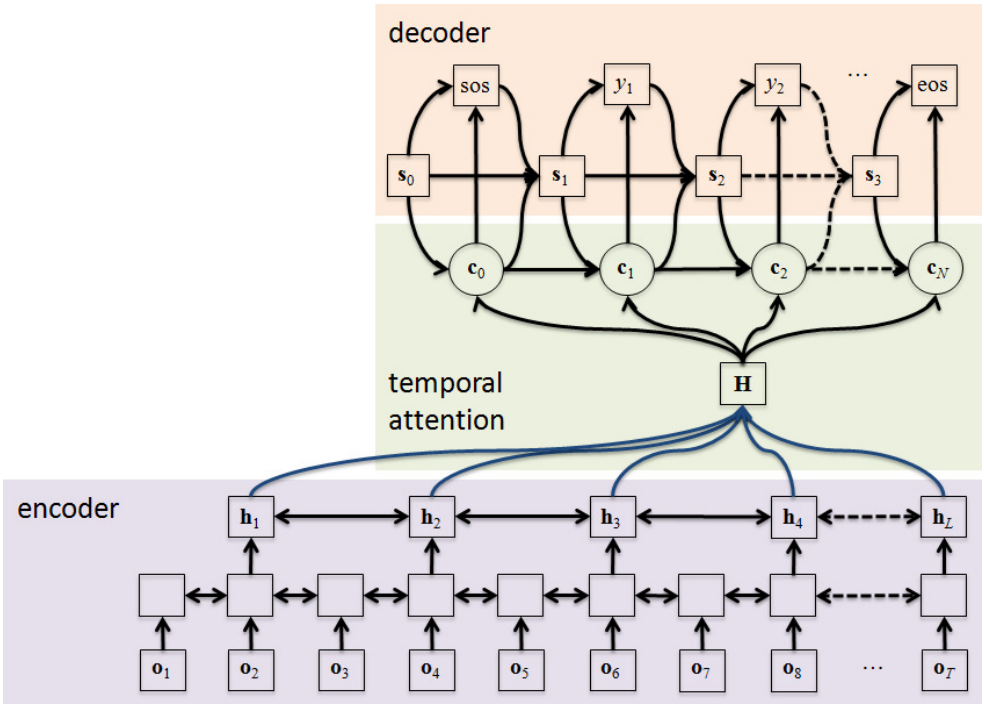


Figure 4.3: Structure of attention-based encoder-decoder network.

Y as follows:

$$P(Y|O) = \prod_n P(y_n|O, y_{1:n-1}), \tag{4.26}$$

$$H = \text{Encoder}(O), \tag{4.27}$$

$$\mathbf{c}_n = \text{Attention}(\mathbf{a}_{n-1}, \mathbf{s}_n, H), \tag{4.28}$$

$$P(y_n|O, y_{1:n-1}) = \text{Decoder}(\mathbf{c}_n, \mathbf{s}_{n-1}, y_{1:n-1}). \tag{4.29}$$

where $\mathbf{o}_t$ is a D_O-dimensional acoustic feature vector, and $y_{1:n-1}$ is a label sequence that consists of y_1 through y_{n-1}. Eqs. (4.26)-(4.29) correspond to E2E_ASR($\cdot$) in Eq. (4.3).

In this framework, the entire network including the encoder, attention, and decoder, is modeled by fully neural network-based architecture, and thus it can be consistently optimized to generate a correct label sequence. This consistent optimization of all the relevant procedures is the main motivation of the end-to-end framework. The following subsections describe each component.

4.5.2 Encoder Network

Given input sequence O, the encoder network transforms it to L-length high-level feature sequence $H = \{\mathbf{h}_l \in \mathbb{R}^{D_\mathrm{H}} | l = 1, \cdots, L\}$, where $\mathbf{h}_l$ is a D_H-dimensional state vector of the encoder's top layer at subsampled time step l. In this work, the encoder network is composed of BLSTM-based recurrent networks. To reduce the length of the input sequence, we apply a subsampling technique [14] to several layers. l represents a time step that was subsampled from t, and L is no more than T. This BLSTM network corresponds to Encoder$(\cdot)$ in Eq. (4.27).

4.5.3 Attention Mechanism

The attention mechanism integrates all encoder outputs H into a D_H-dimensional context vector $\mathbf{c}_n \in \mathbb{R}^{D_\mathrm{H}}$ based on L-dimensional attention weight vector $\mathbf{a}_n \in [0, 1]^L$, which represents a soft alignment of encoder outputs at output time step n. In this work, we adopt a location-based attention mechanism [13], and thus $\mathbf{a}_n$ and $\mathbf{c}_n$ are estimated as follows:

$$\{\mathbf{f}_{n,l}\}_{l=1}^{L} = \mathbf{F} * \mathbf{a}_{n-1}, \tag{4.30}$$

$$k_{n,l} = \tilde{\mathbf{w}}^\mathrm{T} \tanh(\mathbf{V}^\mathrm{S}\mathbf{s}_n + \mathbf{V}^\mathrm{H}\mathbf{h}_l + \mathbf{V}^\mathrm{F}\mathbf{f}_{n,l} + \tilde{\mathbf{b}}), \tag{4.31}$$

$$a_{n,l} = \frac{\exp(\beta k_{n,l})}{\sum_{l=1}^{L} \exp(\beta k_{n,l})}, \tag{4.32}$$

$$\mathbf{c}_n = \sum_{l=1}^{L} a_{n,l}\mathbf{h}_l, \tag{4.33}$$

where $\tilde{\mathbf{w}} \in \mathbb{R}^{1 \times D_\mathrm{V}}$, $\mathbf{V}^\mathrm{H} \in \mathbb{R}^{D_\mathrm{V} \times D_\mathrm{H}}$, $\mathbf{V}^\mathrm{S} \in \mathbb{R}^{D_\mathrm{V} \times D_\mathrm{S}}$, and $\mathbf{V}^\mathrm{F} \in \mathbb{R}^{D_\mathrm{V} \times D_\mathrm{F}}$ are trainable weight matrices. $\tilde{\mathbf{b}} \in \mathbb{R}^{D_\mathrm{V}}$ is a trainable bias vector. $\mathbf{F} \in \mathbb{R}^{D_\mathrm{F} \times D_\mathrm{f}}$ is a trainable convolution filter. $\mathbf{s}_n \in \mathbb{R}^{D_\mathrm{S}}$ is a D_S-dimensional hidden state vector obtained from an upper decoder network at output time step n. β is a sharpening factor [13], and $*$ represents the convolution operation. Eqs. (4.30)-(4.33) correspond to Attention$(\cdot)$ in Eq. (4.28).

The convolution operation is performed with a stride of 1 along the time axis, and the filter $\mathbf{F}$ produce D_F-dimensional feature vector $\mathbf{f}_{n,l}$ at each time step l, where we adopt the zero-padding

technique for the edge region.

4.5.4 Decoder Network

The decoder network recursively updates the hidden state $\mathbf{s}_n$ and estimates the posterior probability for output label y_n as follows:

$$\mathbf{s}_n = \text{Update}(\mathbf{s}_{n-1}, \mathbf{c}_{n-1}, y_{n-1}), \tag{4.34}$$

$$P(y_n|O, y_{1:n-1}) = \text{Posterior}(\mathbf{s}_n, \mathbf{c}_n), \tag{4.35}$$

where the Update$(\cdot)$ and Posterior$(\cdot)$ functions are respectively composed of an unidirectional LSTM-based recurrent network and a feed-forward network. Eqs. (4.34) and (4.35) correspond to Decoder$(\cdot)$ in Eq. (4.29).

Here, special tokens for start-of-sentence (sos) and end-of-sentence (eos) are added to label set $\mathcal{V}$. The decoder network starts the recurrent computation with the sos label and continues to estimate the posterior probability $P(y_n|O, y_{1:n-1})$ until the eos label is emitted, based on the RNN recursiveness.

4.6 Relation to Previous Works

Several related studies exist on neural beamformers based on filter estimation [77, 78, 79] and mask estimation [80, 81, 82, 83, 84, 85, 86, 87]. The main difference is that these previous studies used a component-level training objective within conventional hybrid frameworks, while our work focuses on the entire end-to-end ASR objective. For example, some previous work [80, 82, 83, 86] used a signal-level objective (binary mask classification or regression) to train a network given parallel clean and noisy speech data. On the other hand, other works [77, 78, 79, 81, 84, 85] used ASR objectives (HMM state classification or sequence-discriminative training), but they remain based on the hybrid approach. Speech recognition with

raw multichannel waveforms [95, 96] is also classified into neural beamformers, where the filter coefficients are represented as the network parameters of convolutional neural networks (CNNs), but again these methods are still based on the hybrid approach. Note that the above learning based beamforming approaches can be viewed as an extension of likelihood-maximizing (LIMA) beamformer [97], where beamforming filter coefficients are optimized with HMM/GMM acoustic models based on a maximum likelihood criterion.

If we focus on the network architecture design of the beamforming part aside from the end-to-end framework, our beamformer is based on an MVDR formalization given a reference microphone, which was also previously used in [81, 82]. The difference of our beamformer from those approaches is that it can automatically select a reference microphone within a neural network framework. In [81, 82], one channel is fixedly used as a reference microphone for all utterances by considering microphone geometries. However, our method introduces attention-based reference microphone selection, which allows the beamformer to choose appropriate reference microphones automatically in terms of the entire end-to-end ASR objective without any prior information of microphone geometries.

Similarly, if we only focus on the above automatic reference selection function aside from our entire framework, there exist prior studies [98, 99], which have a function to select dominant channels for a multichannel ASR. [98] uses an attention mechanism to perform channel selection from the pool of multichannel feature candidates in the filterbank domain, while [99] hardly selects dominant features with a max-pooling layer in the hidden state domain. These approaches mainly differ from ours in a sense that they do not hold the beamforming function. This is because they perform their enhancement process in the filterbank or hidden state domain rather than in the STFT domain, and cannot perform beamforming in principle due to the lack of spatial information.

Regarding end-to-end speech recognition, all existing studies are based on a single channel setup. For example, most focus on a standard clean ASR setup without speech enhancement [13, 14, 16, 68, 70, 71, 72, 73]. Several research discussed end-to-end ASR in a noisy environment [69, 74], but these methods deal with noise robustness by preparing various types of simulated

Table 4.1: Corpus information for CHiME-4 and AMI corpora.

CHiME-4	Hours	Speakers
Training	3 (real) + 15 (simu)	4 (real) + 83 (simu)
Development	2.9 (real) + 2.9 (simu)	4 (real) + 4 (simu)
Evaluation	2.2 (real) + 2.2 (simu)	4 (real) + 4 (simu)
AMI	Hours	Speakers
Training	78 (real)	135 (real)
Development	9 (real)	18 (real)
Evaluation	9 (real)	16 (real)

noisy speech for training data without incorporating multichannel speech enhancement in their end-to-end frameworks.

4.7 Experimental Conditions

4.7.1 Data Corpora

We compared the effectiveness of the proposed ME2E system to a baseline end-to-end system with noisy or beamformed speech signals. Even though we had two multichannel ASR benchmarks, CHiME-4 [100] and AMI [101], we mainly used the CHiME-4 corpus to demonstrate our experiments.

CHiME-4 is an ASR task in public noisy environments, consisting of speech recorded using a tablet device with 6-channel microphones in four environments: cafe (CAF), street junction (STR), public transportation (BUS), and pedestrian area (PED). It consists of real and simulated data. The training set consists of 3 hours of real speech data uttered by 4 speakers and 15 hours of simulation speech data uttered by 83 speakers. The development set consists of 2.9 hours of real and simulation speech data uttered by 4 speakers, respectively. The evaluation set consists of 2.2

Table 4.2: Conditions related to feature extraction.

Input for neural beamformer	STFT + DC offset (257-dim)
Input for encoder-decoder	Log Mel filterbank (40-dim)
Sampling frequency	16 kHz
Frame length	25 ms
Frame shift	10 ms
Window function	Hamming

hours of real and simulation speech data uttered by 4 speakers, respectively. From the 6-channel microphones, we excluded the second channel signals, which were captured by a microphone under the tablet, and used the rest five channels for the following multichannel experiments ($C = 5$).

AMI is an ASR task in meetings, consisting of speech recorded using 8-channel circular microphones ($C = 8$). It consists of only real data. The training set consists of about 78 hours of speech data uttered by 135 speakers. The development and evaluation sets consist of about 9 hours of speech data uttered by 18 and 16 speakers, respectively.

ChiME-4 consisted of read speech spoken by native English speakers. while AMI consisted of highly spontaneous speech spoken by mostly non-native English speakers. Such basic information of the above corpora as the number of hours and speakers is summarized in Table 4.1.

4.7.2 Feature Representation

Conditions related to feature extraction are briefly summarized in Table 4.2. The input speech was first converted to a series of STFT feature vectors, each of which was calculated through a 25-ms Hamming window that was shifted at 10-ms intervals. We used 257-dimensional STFT-based features (256 STFT coefficients and 1 DC offset) as an input feature vector for the neural beamformer ($F = 257$) and 40-dimensional log Mel filterbank coefficients as an input feature

Table 4.3: Summary of evaluated systems: FILTER_NET and MASK_NET correspond to proposed method.

System	Training objective	Joint optimization	Use neural network	Use parallel speech
BEAMFORMIT [102]	signal-level	No	No	No
FILTER_NET [3]	ASR-level	Yes	Yes	No
MASK_NET [3]	ASR-level	Yes	Yes	No
ERDOGAN's_MVDR [82]	signal-level	No	Yes	Yes
HEYMANN's_GEV [103]	signal-level	No	Yes	Yes

vector for the encoder-decoder network ($D_{\mathrm{O}} = 40$).

4.7.3 Evaluated Systems

We compared the following seven ASR systems: 1) NOISY, 2) BEAMFORMIT, 3) FILTER_NET, 4) MASK_NET (FIX), 5) MASK_NET (ATT), 6) ERDOGAN's_MVDR, and 7) HEYMANN's_GEV.

NOISY and BEAMFORMIT are the baseline single-channel end-to-end systems that did not include the speech enhancement part in the training phase of their frameworks. Their end-to-end networks were trained only with noisy speech data by following a conventional multi-condition training strategy [100]. During decoding, NOISY used single-channel noisy speech data from 'isolated 1ch track' in CHiME-4 as input, while BEAMFORMIT used the enhanced speech data obtained from the 5-channel signals with BeamformIt [102], which is a well-known weighted delay-and-sum beamformer, as input.

FILTER_NET, MASK_NET (FIX), and MASK_NET (ATT) are the proposed ME2E systems described in Section 4.2. To evaluate the validity of the reference channel selection, we prepared

[3]FILTER_NET and MASK_NET basically follow the formalization in [79] and [82]. However, based on our ME2E ASR concept, they are jointly optimized with the end-to-end ASR back-end based on the end-to-end ASR objective.

MASK_NET (ATT) based on a mask-based beamformer with an attention-based reference selection described in Section 4.4.3, and MASK_NET (FIX) with the 5-th channel as a fixed reference microphone, located on the front in the middle of the tablet device. During training, we adopted a multi-condition training strategy; in addition to optimization with the enhanced features through the neural beamformers, we also used the noisy multichannel speech data as input of the encoder-decoder networks without passing through the neural beamformers to improve the robustness of the encoder-decoder networks.

In addition to the comparison with a conventional delay-and-sum beamformer (BEAMFORMIT), we compared our approach with other state-of-the-art neural beamformer implementations [82, 103], which achieved great ASR performances for conventional hybrid frameworks in the recent CHiME-4 challenge. ERDOGAN's_MVDR and HEYMANN's_GEV also used the same baseline system as well as NOISY and BEAMFORMIT. During decoding, the enhanced speech data produced by the state-of-the-art neural beamformers are used as input to the baseline system.

ERDOGAN's_MVDR adopted the MVDR formalization, similar to our approach, but it always used the 5-th channel as the reference microphone. Therefore, it closely resembles our MASK_NET (FIX). The main difference between them is the training objective. ERDOGAN's_MVDR are separately optimized based on the signal-level objective independent of the ASR component using parallel clean and noisy speech data. On the other hand, MASK_NET (FIX) is jointly optimized based on the end-to-end ASR objective with the ASR component only using noisy speech data. In addition, the structure of mask estimation network is also different from our setting [82].

Different from our approach, HEYMANN's_GEV adopted GEV formalization, which requires the estimation of a steering vector based on eigenvalue decomposition instead of estimating the reference microphone vector. In recent studies on neural beamformers, such a GEV-based neural beamformer is a popular alternative to the MVDR-based neural beamformer. To obtain the enhanced signals, we utilized the software tools provided in the GitHub repository [1].

[1] `https://github.com/fgnt/nn-gev`

Table 4.4: Network configurations.

Model	Layer	Units	Type	Activation
Encoder	L1 - L4	320	BLSTM + Projection	tanh
Decoder	L1	320	LSTM	tanh
	L2	48	Linear	softmax
Filter_net	L1 - L3	320	BLSTM + Projection	tanh
	L4	2570	Linear	tanh
Mask_net	L1 - L3	320	BLSTM + Projection	tanh
	L4	514	Linear	sigmoid

Table 4.3 briefly summarizes the main differences among each evaluated system. "Training objective" indicates that the beamformer was trained based on the ASR-level or the signal-level objective, and "Joint optimization" indicates whether the beamformer was jointly optimized with the end-to-end ASR back-end. "Use neural network" indicates whether the beamformer used the neural network-based architecture, and "Use parallel speech" indicates whether clean speech was used to train the beamformer.

Note that all the evaluated systems basically used the same network structure, which is described in Section 4.7.4. In addition, the hyperparameters for the training and decoding conditions, which are described in Section 4.7.5, were set based on the development accuracy of the NOISY system and shared among all the evaluated systems.

4.7.4 Network Configurations

Network configurations, except the attention mechanisms, are briefly summarized in Table 4.4. All of the above networks were implemented using Chainer [104].

Neural beamformers

For the neural beamformers, we used a 3-layer BLSTM with 320 cells ($D_{\mathrm{Z}} = 320$). After every BLSTM layer, we used a linear projection layer with 320 units to combine the forward and backward LSTM outputs. For the attention mechanism of the reference selection, we set the attention inner product dimension to 320 ($D_{\mathrm{W}} = 320$) and the sharpening factor to 2 ($\alpha = 2$).

Attention-based Encoder-decoder networks

For the attention-based encoder-decoder networks, we used a 4-layer BLSTM with 320 cells in the encoder ($D_{\mathrm{H}} = 320$) and a 1-layer LSTM with 320 cells in the decoder ($D_{\mathrm{S}} = 320$). In the encoder, we subsampled the hidden states of the first and second layers and used every second hidden state for the subsequent layer's inputs. Therefore, the number of hidden states at the encoder's output layer was reduced to $L = T/4$. After every BLSTM layer, we used a linear projection layer with 320 units. For the attention mechanism of the time-alignment, we adopted a location-based attention mechanism, where 10 centered convolution filters of width 100 were used to extract the location-based features. We also set the attention inner product dimension to 320 ($D_{\mathrm{V}} = 320$) and the sharpening factor to 2 ($\beta = 2$).

4.7.5 Training and Decoding

The conditions related to training and decoding are briefly summarized in Table 4.5.

In the training stage, all the parameters were initialized with range [-0.1, 0.1] of a uniform distribution. We used the AdaDelta algorithm [105] with gradient clipping [106] for optimization and initialized AdaDelta hyperparameters $\rho = 0.95$ and $\epsilon = 1^{-8}$. Once the loss over the validation set was degraded, we decreased AdaDelta hyperparameter ϵ by multiplying it by 0.01 at each subsequent epoch. To boost the optimization in a noisy environment, we adopted a joint CTC-attention multi-task loss function [71], as described in Section 4.2.2. We set the interpolation

Table 4.5: Conditions related to training and decoding.

Parameter initialization	Uniform distribution ([-0.1, 0.1])
Optimization technique	AdaDelta + gradient clipping
Training objective	Joint CTC-attention loss ($\gamma = 0.9$)
Training epoch	15
Beam size	20
Length penalty	0.3
Allowed hypothesis length	$0.3 \times L \sim 0.75 \times L$ (CHiME-4)

weight to 0.9 ($\gamma = 0.9$). The training procedure was stopped after 15 epochs.

For decoding, we used a beam search algorithm [5] with a beam size of 20 at each output step to reduce the computation cost. CTC scores were also used to re-score the hypotheses. We adopted a length penalty term [13] to the decoding objective and set the penalty weight to 0.3. In the CHiME-4 experiments, we only allowed hypotheses whose lengths were within $0.3 \times L$ and $0.75 \times L$ during the decoding, while the hypothesis lengths in the AMI experiments were automatically determined based on the above scores. Note that we pursued a pure end-to-end setup without external lexicon or language models and used CER as an evaluation metric.

4.8 Experimental Results

4.8.1 Evaluation in ASR-level Measures (Character Error Rate)

CHiME-4

Table 4.6 shows the recognition performance of CHiME-4 with six systems. The result shows that BEAMFORMIT, FILTER_NET, MASK_NET (FIX), and MASK_NET (ATT) outperformed NOISY, confirming the effectiveness of combining speech enhancement with the attention-based

Table 4.6: Experimental results (character error rate [%]) for CHiME-4 corpus.

Model	dev simu	dev real	eval simu	eval real
NOISY	25.0	24.5	34.7	35.8
BEAMFORMIT	21.5	19.3	31.2	28.2
FILTER_NET	19.1	20.3	28.2	32.7
MASK_NET (FIX)	15.5	18.6	23.7	28.8
MASK_NET (ATT)	**15.3**	**18.2**	**23.7**	**26.8**
ERDOGAN's_MVDR	16.2	**18.2**	24.3	**26.7**

encoder-decoder framework. The comparison of MASK_NET (FIX) and MASK_NET (ATT) validates the using of the attention mechanism for reference channel selection. FILTER_NET also improved the performance more than NOISY, but not as much as MASK_NET (ATT). This is because optimizing the filter estimation network is difficult due to a lack of restrictions to estimate filter coefficients, and it needs optimization, as suggested by a previous work [77]. Finally, MASK_NET (ATT) achieved better recognition performance than BEAMFORMIT, proving the effectiveness of our unified architecture rather than a pipe-line combination of speech enhancement and (end-to-end) speech recognition.

Table 4.6 also shows that the performance of MASK_NET (ATT) is comparable to ERDOGAN's_MVDR, which is a state-of-the-art neural beamformer implementation. Note that MASK _NET (ATT) successfully achieved a good performance without requiring parallel clean and noisy speech data. This result suggests that we can eliminate the requirement of parallel speech data for training by the end-to-end optimization of the ASR system.

We also evaluated HEYMANN's_GEV, but the performance is quite poor[4]. We assume that this result was caused by the speech distortions produced by the GEV-based beamformer. Although the MVDR-based beamformer suppressed the speech distortions, the GEV-based beamformer ignored the speech distortions and only focused on the noise reduction. Such speech distortions sometimes degrade ASR performance, when we input the beamformed signals to existing ASR

[4]For example, CER for eval_real is 71.6.

Table 4.7: Experimental results (character error rate [%]) for AMI corpus.

Model	dev	eval
NOISY	41.8	45.3
BEAMFORMIT	44.9	51.3
MASK_NET (ATT)	**35.7**	**39.0**

systems.

AMI

To further investigate the effectiveness of our proposed ME2E framework, we also experimented with the AMI corpus. Table 4.7 compares the recognition performance of three systems: NOISY, BEAMFORMIT, and MASK_NET (ATT). In NOISY, we used noisy speech data from the 1st channel in AMI as input to the system. Table 4.7 shows that, even in the AMI, our proposed MASK_NET (ATT) achieved better recognition performance than the baseline systems (NOISY and BEAMFORMIT), confirming the effectiveness of our proposed ME2E framework. BEAMFORMIT was worse than NOISY even with the enhanced signals. This phenomenon is sometimes observed in noisy speech recognition where the distortion caused by the sole speech enhancement degrades the performance without re-training. Since our end-to-end system jointly optimized the speech enhancement part with the ASR objective, it can avoid such degradations.

4.8.2 Influence on Number and Order of Channels

As we discussed in Section 4.4.3, one unique characteristic of our proposed MASK_NET (ATT) is its robustness/invariance against the number and order of channels without re-training. Table 4.8 shows the influence of the CHiME-4 validation accuracies on the number and order of the channels. The validation accuracy was computed conditioned on ground truth labels $y^*_{1:n-1}$ in Eq. (4.29) during the decoder's recursive label prediction, which has a strong correlation with

Table 4.8: CHiME-4 validation accuracies [%] for MASK_NET (ATT) with different numbers and orders of channels.

Model	channel	dev
NOISY	isolated_1ch_track	87.9
MASK_NET (ATT)	5_6_4_3_1	91.2
	3_4_1_5_6	91.2
	5_6_4_1	91.1
	6_4_3_1	90.4
	5_6_4	90.9
	6_4_1	90.1

CER. The second column of the table represents the channel indices, which were used as input of the same MASK_NET (ATT) network.

Comparison of 5_6_4_3_1 and 3_4_1_5_6 shows that the order of the channels did not affect the recognition performance of MASK_NET (ATT) at all, as we expected. In addition, even when we used fewer than three or four channels as input, MASK_NET (ATT) still outperformed NOISY (single channel). These results confirm that our proposed ME2E system can deal with input signals with an arbitrary number and order of channels without any re-configuration and re-training.

In addition to the above analyses, comparing the setups using the same number of channels, 5_6_4_1 and 5_6_4 outperformed 6_4_3_1 and 6_4_1, respectively. The observation is due to the fact that the 5-th channel is the single best channel in the real dev and eval sets of CHiME-4 task.

4.8.3 Histogram of Selected Reference Microphone

To analyze the behavior of our proposed attention mechanism for reference microphone selection, Figure 4.4 illustrates a histogram of the selected reference microphone for the development

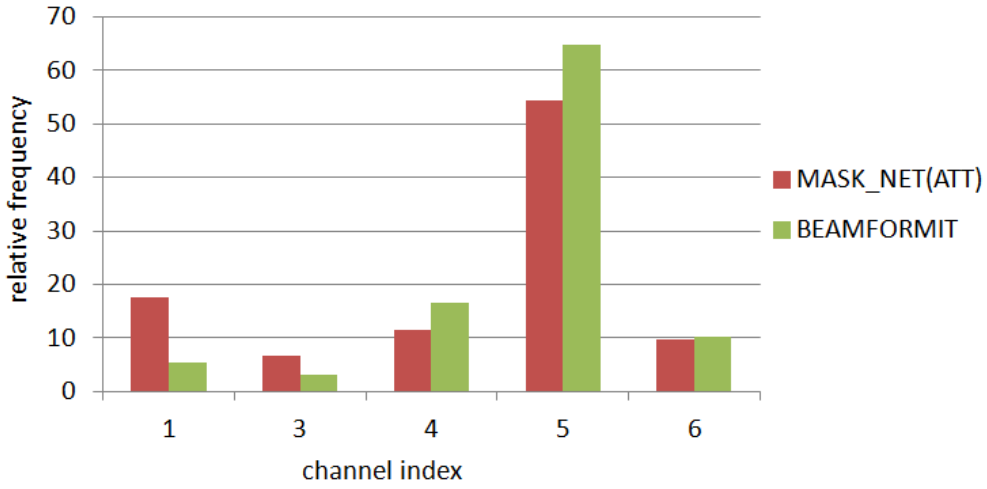


Figure 4.4: Histogram of reference microphone selected by BEAMFORMIT and MASK_NET (ATT).

set with two systems: BEAMFORMIT and MASK_NET (ATT). As described in Eqs. (4.22) and (4.23), our proposed reference selection mechanism is formalized in a probabilistic way, but in this figure, the frequency is counted assuming that the channel index with the highest probability is selected. BEAMFORMIT selected a reference microphone using a metric based on the signal-level criterion, i.e., pairwise cross-correlation in time domain [102].

Figure 4.4 shows that both BEAMFORMIT and MASK_NET (ATT) selected the 5-th channel most frequently. That result seems plausible from the viewpoint of microphone geometries, because the 5-th channel is located on the front and the center of the tablet device, and therefore, it is expected to capture relatively clean speech signals. Our preliminary result also shows that the 5-th channel is the single best performing channel in the array. One interesting finding is that the trends in the selected reference seem similar, although MASK_NET (ATT) only learned the reference selection mechanism to improve the end-to-end ASR objective.

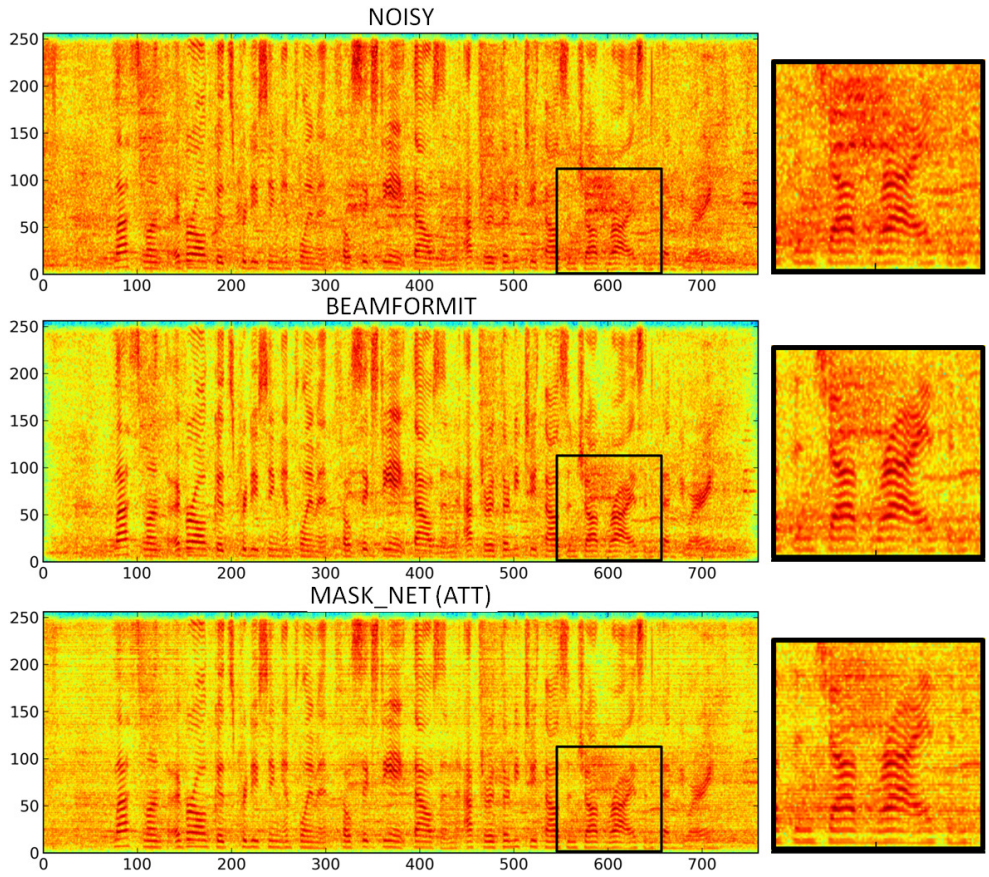


Figure 4.5: Comparison of log-magnitude spectrograms of CHiME-4 utterance with the 5-th channel noisy signal, enhanced signal with BeamformIt, and enhanced signal with our proposed MASK_NET (ATT).

4.8.4 Visualization of Beamformed Signals

To analyze the behavior of our developed speech enhancement component with a neural beamformer, Figure 4.5 visualizes the spectrograms of the same CHiME-4 utterance for three signals: 1) the 5-th channel noisy signal, 2) an enhanced signal with BEAMFORMIT, and 3) an enhanced signal with MASK_NET (ATT). We confirmed that BEAMFORMIT, and MASK_NET (ATT) successfully suppressed noise compared to the 5-th channel signal by eliminating the blurred red areas overall. In addition, by focusing on the black boxes, the harmonic structure, which was corrupted in the 5-th channel signal, was recovered in BEAMFORMIT, and MASK_NET (ATT).

This result suggests that our proposed MASK_NET (ATT) successfully learned a noise suppression function that resembles the conventional beamformer, although it is optimized based on the end-to-end ASR objective, without explicitly using clean data as a target.

4.8.5 Evaluation in Signal-level Measures (Signal-to-Distortion Ratio and Perceptual Evaluation of Speech Quality)

To further analyze the behavior of our developed speech enhancement component with a neural beamformer, we evaluated the speech enhancement quality of the outputs of the beamformer modules based on two signal-level criteria: 1) a signal-to-distortion ratio (SDR) [107] and 2) a perceptual evaluation of speech quality (PESQ) [108]. In both criteria, a higher score indicates that its corresponding estimated signal obtained higher quality.

The score calculation for these two criteria needs a pair of estimated enhanced speech signals and its corresponding clean speech signals. Therefore, we used the development set of the simulation data in the CHiME-4 corpus for this evaluation.

Figs. 4.6 and 4.7 respectively show the SDR and PESQ scores for the simulation data in the CHiME-4 development set for five signals: 1) the noisy signal from 'isolated 1ch track', 2) an enhanced signal with BEAMFORMIT, 3) an enhanced signal with MASK_NET (ATT), 4) an enhanced signal with ERDOGAN's_MVDR, and 5) an enhanced signal with HEYMANN's_GEV. Because the SDR and PESQ scores are computed for each utterance, we overlaid the standard deviation value (thin line) on the mean value (blue bar) in the figures. The results in both of the scores show that MASK_NET (ATT) achieved reasonable speech enhancement and its enhancement quality is competitive to or better than the cases of BEAMFORMIT and HEYMANN's_GEV, suggesting that the neural beamformer developed in our ME2E ASR framework successfully learns speech enhancement (noise suppression) ability, although it was optimized under the end-to-end ASR-oriented criterion.

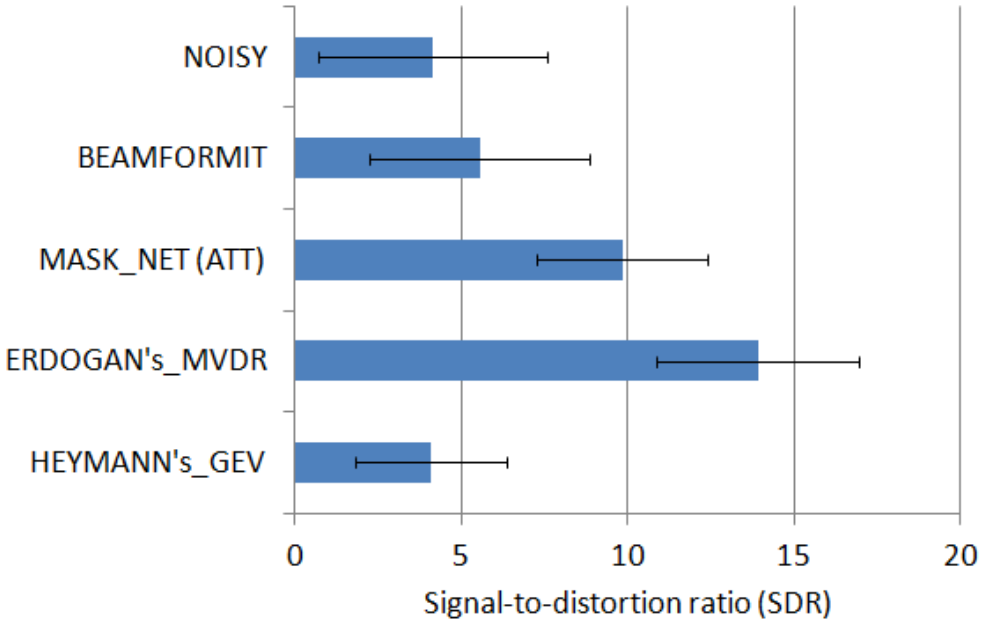


Figure 4.6: Signal-to-distortion ratio (SDR) for CHiME-4 simulation data in development set. Each bar indicates the mean in terms of utterances; a thin line indicates the standard deviation.

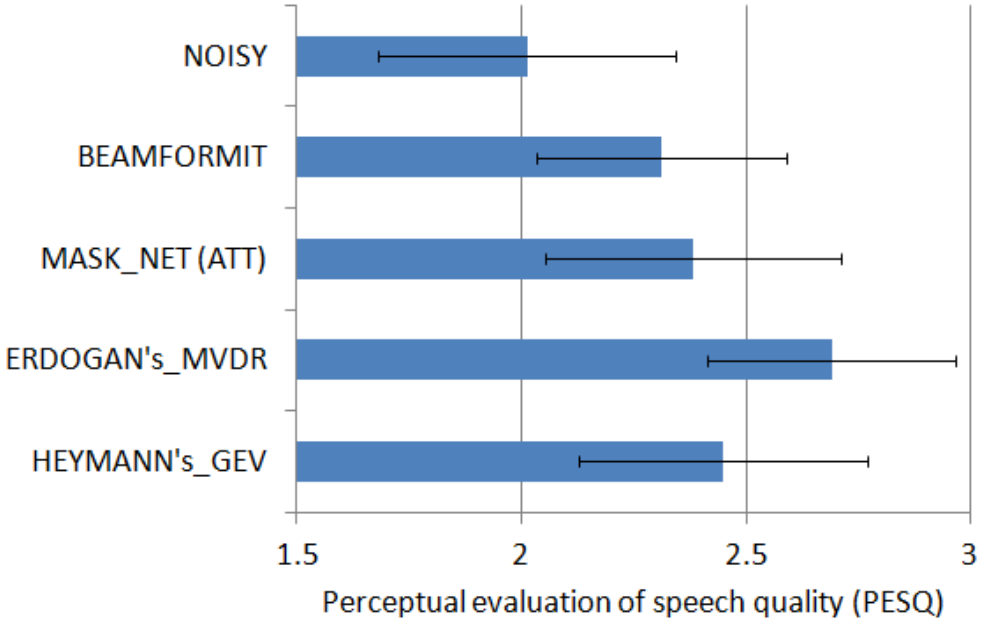


Figure 4.7: Perceptual evaluation of speech quality (PESQ) for CHiME-4 simulation data in development set. Each bar indicates the mean in terms of utterances; a thin line indicates the standard deviation.

4.9 Summary

To handle the challenging noisy ASR tasks, we extended an existing attention-based encoder-decoder framework by integrating a neural beamformer and proposed a unified architecture of ME2E ASR systems. This architecture allows the overall inference in multichannel speech recognition (i.e., from speech enhancement to speech recognition) to be optimized based on the end-to-end ASR objective, and leads to an end-to-end framework that works well in the presence of strong background noise. In addition, because it is formalized independent of microphone geometries, it can deal with input signals with an arbitrary number and order of channels without any re-configuration and re-training. Our experimental results on challenging noisy ASR benchmarks (CHiME-4 and AMI) show that the proposed framework outperformed the end-to-end baseline with noisy and delay-and-sum beamformed inputs. In addition, evaluation of the speech enhancement quality for beamformed features shows that our neural beamformer successfully learned a noise suppression function, although it is optimized based on the end-to-end ASR objective, without using parallel clean and noisy speech data.

5 Conclusion

5.1 Summary of Dissertation

In this dissertation, we studied two frameworks for improving DL-based ASR frameworks, i.e., the hybrid DNN-HMM framework and the end-to-end DNN-based framework, and focused on the issues of speaker and speaking environment variability. Our approaches to the issues were mainly based on the incorporation of modularity, or in other words, internal structure into the DNN part of ASR systems.

In Chapter 3, we addressed the problem of mismatch between training and testing speakers. To solve this problem, we proposed a novel speaker adaptation scheme that incorporated the SAT concept in the DNN training of the hybrid DNN-HMM framework. The proposed SAT-based speaker adaptation scheme introduced modularity, more precisely, SD module localization, in the DNN part and optimized the DNN assuming that the SD module was adapted in the adaptation stage. The effect of our proposed SAT-based training scheme was experimentally proved in ASR tasks using a difficult lecture speech corpus, i.e., TED Talks, and it successfully reduced the number of adaptable parameters and improved the adaptability of the whole hybrid ASR system.

In Chapter 4, we addressed the recognition difficulty caused by the variability of severe background noises. To solve this problem, we focused on the recent architecture of end-to-end ASR systems and proposed a novel ME2E ASR architecture that integrated the speech enhancement and speech recognition components into a single neural network-based ASR system structure. Incorporating the speech enhancement component into the end-to-end ASR architecture made it possible to directly develop a mapping function that transferred the multichannel noisy signals

(system inputs) to transcriptions (system outputs) with the noise suppression function. Note that these functions were gained only through recognition-oriented optimization (training) using multichannel speech samples and their corresponding transcriptions. We experimentally proved the effect of the proposed ME2E architecture in challenging noisy ASR benchmark tasks: CHiME-4 and AMI.

5.2 Future Works

Based on our experiment results, we revealed the fundamental effectiveness of embedding modularity into a DNN for coping with the problems of speaker and environment variability. However, the following remaining issues must be further investigated:

1. In Chapter 3, we proposed novel speaker adaptation schemes that incorporated the SAT concept in DNN training and showed their effectiveness for increasing the adaptability of DNN-based ASR systems while reducing the size of adaptable parameters. However, as a trade-off with the adaptability improvement, our proposed method requires multiple optimization steps. For example, the SAT-based optimization procedure with a bottleneck LTN SD module adopts the following four-step optimization procedures to estimate the seed model for adaptation: 1) layer-wise RBM pre-training, 2) SI training, 3) SAT-based training, and 4) SAT-based retraining. Therefore, the simplification (unification) of these stepwise optimization procedures is critical for increasing the usability of the proposed method.
2. In Chapter 4, we proposed a novel ME2E ASR architecture that integrated the speech enhancement and speech recognition components. The experimental results showed that it simultaneously gained the functions of suppressing noise in input speech signals and transferring input speech signals to output transcriptions only through recognition-oriented optimization. In our experiment, although we introduced a mechanism for compensating the (noise) environment variability into the end-to-end framework, we did not study a mechanism for compensating the speaker variability. This point is an important research

topic. Moreover, a speaker adaptation mechanism must be investigated in ME2E ASR architecture that was originally tuned to the speaking environment variability issue.

Bibliography

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. 1

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105. 1

[3] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9. 1

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778. 1

[5] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems (NIPS)*, 2014, pp. 3104–3112. 1, 82

[6] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014. 1

[7] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014. 1

[8] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al., "Deep

neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012. 1, 2, 3, 16, 57

[9] George E Dahl, Dong Yu, Li Deng, and Alex Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on audio, speech, and language processing*, vol. 20, no. 1, pp. 30–42, 2012. 1, 2, 3, 16, 57

[10] Abdel-rahman Mohamed, George E Dahl, and Geoffrey Hinton, "Acoustic modeling using deep belief networks," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 14–22, 2012. 1, 2, 3, 16, 57

[11] Bing-Hwang Juang, Stephene Levinson, and M Sondhi, "Maximum likelihood estimation for multivariate mixture observations of Markov chains," *IEEE Transactions on Information Theory*, vol. 32, no. 2, pp. 307–309, 1986. 2

[12] Lawrence R Rabiner and Biing-Hwang Juang, "Fundamentals of speech recognition," 1993. 2

[13] Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio, "Attention-based models for speech recognition," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 577–585. 3, 57, 62, 73, 75, 82

[14] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio, "End-to-end attention-based large vocabulary speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 4945–4949. 3, 57, 73, 75

[15] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, "Speech recognition with deep recurrent neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 6645–6649. 3

[16] Alex Graves and Navdeep Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *International Conference on Machine Learning (ICML)*, 2014, pp. 1764–1772. 3, 57, 75

[17] Tsubasa Ochiai, Shigeki Matsuda, Hideyuki Watanabe, Xugang Lu, Chiori Hori, Hisashi

Kawai, and Shigeru Katagiri, "Speaker adaptive training localizing speaker modules in DNN for hybrid DNN-HMM speech recognizers," *IEICE Transactions on Information and Systems*, vol. E99-D, no. 10, pp. 2431–2443, 2016. 4

[18] Tsubasa Ochiai, Shigeki Matsuda, Xugang Lu, Chiori Hori, and Shigeru Katagiri, "Speaker adaptive training using deep neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 6349–6353. 4

[19] Tsubasa Ochiai, Shigeki Matsuda, Hideyuki Watanabe, Xugang Lu, Chiori Hori, and Shigeru Katagiri, "Speaker adaptive training using deep neural networks embedding linear transformation networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 4605–4609. 4

[20] Tsubasa Ochiai, Shigeki Matsuda, Hideyuki Watanabe, Xugang Lu, Hisashi Kawai, and Shigeru Katagiri, "Bottleneck linear transformation network adaptation for speaker adaptive training-based hybrid DNN-HMM speech recognizer," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 5015–5019. 4

[21] Tsubasa Ochiai, Shinji Watanabe, Takaaki Hori, John R Hershey, and Xiong Xiao, "A unified architecture for multichannel end-to-end speech recognition with neural beamforming," *IEEE Journal of Selected Topics in Signal Processing*, 2017. 5

[22] Tsubasa Ochiai, Shinji Watanabe, Takaaki Hori, and John R Hershey, "Multichannel end-to-end speech recognition," in *International Conference on Machine Learning (ICML)*, 2017, pp. 2632–2641. 5

[23] Tsubasa Ochiai, Shinji Watanabe, and Shigeru Katagiri, "Does speech enhancement work with end-to-end ASR objectives?: Experimental analysis of multichannel end-to-end ASR," in *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2017. 5

[24] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al., "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, pp. 1, 1988. 9

[25] Stanley F Chen and Joshua Goodman, "An empirical study of smoothing techniques for language modeling," in *Proceedings of the 34th annual meeting on Association for*

Computational Linguistics. Association for Computational Linguistics, 1996, pp. 310–318. 11

[26] Joshua T Goodman, "A bit of progress in language modeling," *Computer Speech & Language*, vol. 15, no. 4, pp. 403–434, 2001. 11

[27] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur, "Recurrent neural network based language model.," in *Interspeech*, 2010, vol. 2, p. 3. 11

[28] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney, "LSTM neural networks for language modeling," in *Interspeech*, 2012. 11

[29] Mei-Yuh Hwang and Xuedong Huang, "Shared-distribution hidden markov models for speech recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 1, no. 4, pp. 414–420, 1993. 13

[30] Jia Pan, Cong Liu, Zhiguo Wang, Yu Hu, and Hui Jiang, "Investigation of deep neural networks (dnn) for large vocabulary continuous speech recognition: Why dnn surpasses gmms in acoustic modeling," in *Chinese Spoken Language Processing (ISCSLP), 2012 8th International Symposium on*. IEEE, 2012, pp. 301–305. 13

[31] J-L Gauvain and Chin-Hui Lee, "Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains," *IEEE transactions on speech and audio processing*, vol. 2, no. 2, pp. 291–298, 1994. 15

[32] Christopher J Leggetter and Philip C Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models," *Computer Speech & Language*, vol. 9, no. 2, pp. 171–185, 1995. 15

[33] Mark JF Gales, "Maximum likelihood linear transformations for HMM-based speech recognition," *Computer speech & language*, vol. 12, no. 2, pp. 75–98, 1998. 15

[34] Tasos Anastasakos, John McDonough, Richard Schwartz, and John Makhoul, "A compact model for speaker-adaptive training," in *IEEE International Conference on Spoken Language Processing (ICSLP)*, 1996, vol. 2, pp. 1137–1140. 15

[35] Tasos Anastasakos, John McDonough, and John Makhoul, "Speaker adaptive training: A

maximum likelihood approach to speaker normalization," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1997, vol. 2, pp. 1043–1046. 15

[36] Naoto Iwahashi, "Training method for pattern classifier based on the performance after adaptation," *IEICE Transactions on Information and Systems*, vol. 83, no. 7, pp. 1560–1566, 2000. 15

[37] Geoffrey E Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Computation*, vol. 14, no. 8, pp. 1771–1800, 2002. 18

[38] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006. 18

[39] Arthur E Hoerl and Robert W Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970. 22

[40] Christopher M Bishop, *Pattern recognition and machine learning*, springer, 2006. 22

[41] Hank Liao, "Speaker adaptation of context dependent deep neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 7947–7951. 22, 23, 24

[42] Dong Yu, Kaisheng Yao, Hang Su, Gang Li, and Frank Seide, "KL-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 7893–7897. 23, 24

[43] Joao Neto, Luís Almeida, Mike Hochberg, Ciro Martins, Luis Nunes, Steve Renals, and Tony Robinson, "Speaker-adaptation for hybrid HMM-ANN continuous speech recognition system," 1995. 23, 24, 38

[44] Roberto Gemello, Franco Mana, Stefano Scanzio, Pietro Laface, and Renato De Mori, "Linear hidden transformations for adaptation of hybrid ANN/HMM models," *Speech Communication*, vol. 49, no. 10, pp. 827–835, 2007. 23, 24, 38

[45] Bo Li and Khe Chai Sim, "Comparison of discriminative input and output transformations

for speaker adaptation in the hybrid NN/HMM systems," in *Interspeech*, 2010. 23, 24, 38

[46] Frank Seide, Gang Li, Xie Chen, and Dong Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2011, pp. 24–29. 23, 24, 47

[47] Jian Xue, Jinyu Li, Dong Yu, Mike Seltzer, and Yifan Gong, "Singular value decomposition based low-footprint speaker adaptation and personalization for deep neural network," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 6359–6363. 23, 24, 47, 51, 55

[48] Shaofei Xue, Ossama Abdel-Hamid, Hui Jiang, and Lirong Dai, "Direct adaptation of hybrid DNN/HMM model for fast speaker adaptation in LVCSR based on speaker code," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 6339–6343. 23, 24

[49] Ossama Abdel-Hamid and Hui Jiang, "Fast speaker adaptation of hybrid NN/HMM model for speech recognition based on discriminative learning of speaker code," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 7942–7946. 23, 24

[50] George Saon, Hagen Soltau, David Nahamoo, and Michael Picheny, "Speaker adaptation of neural network acoustic models using i-vectors.," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2013, pp. 55–59. 23, 24

[51] Vishwa Gupta, Patrick Kenny, Pierre Ouellet, and Themos Stafylakis, "I-vector-based speaker adaptation of deep neural networks for french broadcast audio transcription," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 6334–6338. 23, 24

[52] Sabato Marco Siniscalchi, Jinyu Li, and Chin-Hui Lee, "Hermitian based hidden activation functions for adaptation of hybrid HMM/ANN models," in *Interspeech*, 2012. 23, 24

[53] Pawel Swietojanski and Steve Renals, "Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models," in *Spoken Language Tech-*

nology Workshop (SLT), 2014 IEEE, 2014, pp. 171–176. 23, 24, 47

[54] Shakti P Rath, Daniel Povey, Karel Veselỳ, and Jan Cernockỳ, "Improved feature processing for deep neural networks.," in *Interspeech*, 2013, pp. 109–113. 23, 24

[55] Takuya Yoshioka, Anton Ragni, and Mark JF Gales, "Investigation of unsupervised adaptation of DNN acoustic models with filter bank input," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014, pp. 6344–6348. 23, 24

[56] Yajie Miao, Hao Zhang, and Florian Metze, "Towards speaker adaptive training of deep neural network acoustic models," pp. 2189–2193, 2014. 23, 24

[57] Jan Trmal, Jan Zelinka, and Ludek Muller, "On speaker adaptive training of artificial neural networks," pp. 554–557, 2010. 23

[58] Sun Yuan Kung, *Digital neural networks*, Prentice-Hall, Inc., 1993. 23

[59] Najim Dehak, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet, "Front-end factor analysis for speaker verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, 2011. 24

[60] Mauro Cettolo, Jan Niehues, Sebastian Stuker, Luisa Bentivogli, and Marcello Federico, "Report on the 10th IWSLT evaluation campaign," in *International Workshop on Spoken Language Translation (IWSLT)*, 2013. 25

[61] Steven Davis and Paul Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE transactions on acoustics, speech, and signal processing*, vol. 28, no. 4, pp. 357–366, 1980. 26

[62] Hitoshi Yamamoto, Youzheng Wu, Chien-Lin Huang, Xugang Lu, Paul R Dixon, Shigeki Matsuda, Chiori Hori, and Hideki Kashioka, "The NICT ASR system for IWSLT2012," in *International Workshop on Spoken Language Translation (IWSLT)*, 2012. 28

[63] Daniel Povey, Dimitri Kanevsky, Brian Kingsbury, Bhuvana Ramabhadran, George Saon, and Karthik Visweswariah, "Boosted MMI for model and feature-space discriminative training," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2008, pp. 4057–4060. 28

[64] Lidia Mangu, Eric Brill, and Andreas Stolcke, "Finding consensus in speech recognition: word error minimization and other applications of confusion networks," *Computer Speech & Language*, vol. 14, no. 4, pp. 373–400, 2000. 28

[65] Yong Zhao, Jinyu Li, Jian Xue, and Yifan Gong, "Investigating online low-footprint speaker adaptation using generalized linear regression and click-through data," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 4310–4314. 47

[66] Kshitiz Kumar, Chaojun Liu, Kaisheng Yao, and Yifan Gong, "Intermediate-layer DNN adaptation for offline and session-based iterative speaker adaptation," in *Interspeech*, 2015. 47, 55

[67] Sheng Li, Xugang Lu, Yuya Akita, and Tatsuya Kawahara, "Ensemble speaker modeling using speaker adaptive training deep neural network for speaker adaptation," in *Interspeech*, 2015. 47, 55

[68] Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "End-to-end continuous speech recognition using attention-based recurrent NN: First results," *arXiv preprint arXiv:1412.1602*, 2014. 57, 75

[69] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4960–4964, 2016. 57, 62, 75

[70] Yu Zhang, William Chan, and Navdeep Jaitly, "Very deep convolutional networks for end-to-end speech recognition," *arXiv preprint arXiv:1610.03022*, 2016. 57, 75

[71] Suyoun Kim, Takaaki Hori, and Shinji Watanabe, "Joint CTC-attention based end-to-end speech recognition using multi-task learning," in *Proc. ICASSP*, 2017, pp. 4835–4839. 57, 61, 75, 81

[72] Liang Lu, Xingxing Zhang, and Steve Renals, "On training the recurrent neural network encoder-decoder for large vocabulary end-to-end speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 5060–

5064. 57, 75

[73] Yajie Miao, Mohammad Gowayyed, and Florian Metze, "EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2015, pp. 167–174. 57, 75

[74] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, et al., "Deep speech 2: End-to-end speech recognition in English and Mandarin," *International Conference on Machine Learning (ICML)*, pp. 173–182, 2016. 57, 75

[75] Keisuke Kinoshita, Marc Delcroix, Sharon Gannot, Emanuël AP Habets, Reinhold Haeb-Umbach, Walter Kellermann, Volker Leutnant, Roland Maas, Tomohiro Nakatani, Bhiksha Raj, et al., "A summary of the REVERB challenge: state-of-the-art and remaining challenges in reverberant speech processing research," *EURASIP Journal on Advances in Signal Processing*, vol. 2016, no. 1, pp. 1–19, 2016. 58

[76] Jon Barker, Ricard Marxer, Emmanuel Vincent, and Shinji Watanabe, "The third 'CHiME' speech separation and recognition challenge: Analysis and outcomes," *Computer Speech & Language*, 2016. 58

[77] Xiong Xiao, Shinji Watanabe, Hakan Erdogan, Liang Lu, John R Hershey, Michael L Seltzer, Guoguo Chen, Yu Zhang, Michael Mandel, and Dong Yu, "Deep beamforming networks for multi-channel speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 5745–5749. 58, 62, 63, 74, 83

[78] Bo Li, Tara N Sainath, Ron J Weiss, Kevin W Wilson, and Michiel Bacchiani, "Neural network adaptive beamforming for robust multichannel speech recognition," in *Interspeech*, 2016, pp. 1976–1980. 58, 63, 74

[79] Zhong Meng, Shinji Watanabe, John R Hershey, and Hakan Erdogan, "Deep long short-term memory adaptive beamforming networks for multichannel robust speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 271–275. 58, 74, 78

[80] Jahn Heymann, Lukas Drude, and Reinhold Haeb-Umbach, "Neural network based spectral mask estimation for acoustic beamforming," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 196–200. 58, 66, 68, 74

[81] Xiong Xiao, Chenglin Xu, Zhaofeng Zhang, Shengkui Zhao, Sining Sun, and Shinji Watanabe, "A study of learning based beamforming methods for speech recognition," in *CHiME 2016 workshop*, 2016, pp. 26–31. 58, 66, 69, 74, 75

[82] Hakan Erdogan, Tomoki Hayashi, John R Hershey, Takaaki Hori, Chiori Hori, Wei-Ning Hsu, Suyoun Kim, Jonathan Le Roux, Zhong Meng, and Shinji Watanabe, "Multi-channel speech recognition: LSTMs all the way through," in *CHiME 2016 workshop*, 2016. 58, 66, 69, 74, 75, 78, 79

[83] Hakan Erdogan, John R Hershey, Shinji Watanabe, Michael Mandel, and Jonathan Le Roux, "Improved MVDR beamforming using single-channel mask prediction networks," in *Interspeech*, 2016, pp. 1981–1985. 58, 74

[84] Jahn Heymann, Lukas Drude, Christoph Boeddeker, Patrick Hanebrink, and Reinhold Haeb-Umbach, "BEAMNET: end-to-end training of a beamformer-supported multi-channel ASR system," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 5325–5329. 58, 62, 74

[85] Xiong Xiao, Shengkui Zhao, Douglas Jones, Eng-Siong Chng, and Haizhou Li, "On time-frequency mask estimation for MVDR beamforming with application in robust speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 3246–3250. 58, 74

[86] Tomohiro Nakatani, Nobutaka Ito, Takuya Higuchi, Shoko Araki, and Keisuke Kinoshita, "Integrating DNN-based and spatial clustering-based mask estimation for robust MVDR beamforming," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 286–290, 2017. 58, 74

[87] Lukas Pfeifenberger, Matthias Zohrer, and Franz Pernkopf, "DNN-based speech mask estimation for eigenvector beamforming," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 66–70. 58, 74

[88] Takuya Yoshioka, Nobutaka Ito, Marc Delcroix, Atsunori Ogawa, Keisuke Kinoshita, Masakiyo Fujimoto, Chengzhu Yu, Wojciech J Fabian, Miquel Espi, Takuya Higuchi, et al., "The NTT CHiME-3 system: Advances in speech enhancement and recognition for mobile multi-microphone devices," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2015, pp. 436–443. 58, 66, 68

[89] Mehrez Souden, Jacob Benesty, and Sofiène Affes, "On optimal frequency-domain multichannel linear filtering for noise reduction," *IEEE Transactions on audio, speech, and language processing*, vol. 18, no. 2, pp. 260–276, 2010. 58, 68

[90] Paul J Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990. 62

[91] Zhong-Qiu Wang and DeLiang Wang, "A joint training framework for robust automatic speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 4, pp. 796–806, 2016. 62

[92] Tara N Sainath, Arun Narayanan, Ron J Weiss, Ehsan Variani, Kevin W Wilson, Michiel Bacchiani, and Izhak Shafran, "Reducing the computational complexity of multimicrophone acoustic models with integrated feature extraction," in *Interspeech*, 2016, pp. 1971–1975. 63

[93] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997. 64

[94] Alex Graves and Jürgen Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005. 64

[95] Yedid Hoshen, Ron J Weiss, and Kevin W Wilson, "Speech acoustic modeling from raw multichannel waveforms," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 4624–4628. 75

[96] Tara N Sainath, Ron J Weiss, Andrew Senior, Kevin W Wilson, and Oriol Vinyals, "Learning the speech front-end with raw waveform CLDNNs," in *Interspeech*, 2015, pp. 1–5. 75

[97] Michael L Seltzer, Bhiksha Raj, and Richard M Stern, "Likelihood-maximizing beamforming for robust hands-free speech recognition," *IEEE Transactions on speech and audio processing*, vol. 12, no. 5, pp. 489–498, 2004. 75

[98] Suyoun Kim and Ian Lane, "Recurrent models for auditory attention in multi-microphone distance speech recognition," in *Interspeech*, 2016, pp. 1–9. 75

[99] Pawel Swietojanski, Arnab Ghoshal, and Steve Renals, "Convolutional neural networks for distant speech recognition," *IEEE Signal Processing Letters*, vol. 21, no. 9, pp. 1120–1124, 2014. 75

[100] Emmanuel Vincent, Shinji Watanabe, Aditya Arie Nugraha, Jon Barker, and Ricard Marxer, "An analysis of environment, microphone and data simulation mismatches in robust speech recognition," *Computer Speech & Language*, 2016. 76, 78

[101] Thomas Hain, Lukas Burget, John Dines, Giulia Garau, Vincent Wan, Martin Karafi, Jithendra Vepa, and Mike Lincoln, "The AMI system for the transcription of speech in meetings," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2007, pp. 357–360. 76

[102] Xavier Anguera, Chuck Wooters, and Javier Hernando, "Acoustic beamforming for speaker diarization of meetings," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 7, pp. 2011–2022, 2007. 78, 86

[103] Jahn Heymann, Lukas Drude, and Reinhold Haeb-Umbach, "Wide residual BLSTM network with discriminative speaker adaptation for robust speech recognition," in *CHiME 2016 workshop*, 2016. 78, 79

[104] Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton, "Chainer: a next-generation open source framework for deep learning," in *Proceedings of Workshop on Machine Learning Systems (LearningSys) in NIPS*, 2015. 80

[105] Matthew D Zeiler, "ADADELTA: An adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012. 81

[106] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio, "On the difficulty of training recurrent neural networks," *International Conference on Machine Learning (ICML)*, pp.

1310–1318, 2013. 81

[107] Emmanuel Vincent, Rémi Gribonval, and Cédric Févotte, "Performance measurement in blind audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 4, pp. 1462–1469, 2006. 88

[108] Antony W Rix, John G Beerends, Michael P Hollier, and Andries P Hekstra, "Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs," in *Proc. ICASSP*, 2001, vol. 2, pp. 749–752. 88

A Notation List

A.1 Speaker Adaptive Training for Deep Neural Networks (Section 3)

Basic indices	
n	training sample index
τ	input time step
k	senone class index
l	network layer index
l_{SD}	network layer index, to which the SD module is allocated
s	training speaker index
t	target speaker index
N	number of training samples
$\mathcal{T}$	length of input sequence
K	number of senone classes
L	number of network layers
S	number of training speakers

SAT-DNN-ORG method (Section 3.2)	
$\mathcal{X}$	speech samples spoken by all training speakers
$\mathbf{X}^n$	sequence of acoustic features for n-th training sample
$\mathbf{x}_\tau^n$	acoustic feature vector at τ in n-th training sample
$\mathbf{T}^n$	sequence of target labels for n-th training sample
$\mathbf{t}_\tau^n$	target label vector at τ in n-th training sample

$\mathbf{y}_\tau^n$	network output vector given input feature vector $\mathbf{x}_\tau^n$
Λ	network parameters for entire layers
λ_l	network parameters for l-th layer
$\mathbf{W}_l$	DNN's weight matrix for l-th layer
$\mathbf{b}_l$	DNN's bias vector for l-th layer
E_{CE}	accumulated cross entropy error
R	L^2 norm-based regularization term
ρ_{SI}	regularization coefficient in initialization stage
$\mathbf{G}_{l_{\mathrm{SD}}}$	SD module parameters for all training speakers, assuming SD module layer is allocated to l_{SD}-th layer
$\mathbf{g}_{l_{\mathrm{SD}}}^s$	SD module parameters for s-th training speaker, which consist of DNN's network parameters
$\mathcal{X}^s$	speech samples spoken by training speaker s
$E_{\text{SAT-CE}}$	accumulated cross entropy error in SAT stage
ρ_{SAT}	regularization coefficient in SAT stage
$\mathbf{g}_{l_{\mathrm{SD}}}^t$	SD module parameters for t-th target speaker, which consist of DNN's network parameters
$\mathcal{X}^t$	speech samples spoken by target speaker t
ρ_{SA}	regularization coefficient in speaker adaptation stage
$\mathbf{g}_{l_{\mathrm{SD}}}^{\mathrm{anchor}}$	anchor state of SD module parameters for L^2 prior-based regularization term

SAT-DNN-LTN method (Section 3.3)	
$\mathbf{A}_{l_{\mathrm{SD}}}$	LTN's weight matrix, which is inserted to l_{SD}-th layer
$\mathbf{a}_{l_{\mathrm{SD}}}$	LTN's bias vector, which is inserted to l_{SD}-th layer
$\mathbf{I}_{l_{\mathrm{SD}}}$	identity matrix
$\mathbf{0}_{l_{\mathrm{SD}}}$	zero vector
ϕ	activation function
$\tilde{\mathbf{G}}_{l_{\mathrm{SD}}}$	SD module parameters for all training speakers, assuming SD module layer is allocated to l_{SD}-th layer

$\tilde{\mathbf{g}}^{s}_{l_{\mathrm{SD}}}$	SD module parameters for s-th training speaker, which consist of LTN's network parameters
$\tilde{\mathbf{g}}^{t}_{l_{\mathrm{SD}}}$	SD module parameters for t-th target speaker, which consist of LTN's network parameters

SAT-DNN-BTN method (Section 3.4)	
σ_i	i-th singular values
$\mathbf{\Sigma}$	rectangular diagonal matrix whose diagonal elements are singular values
$\mathbf{U}$	orthogonal matrix produced by SVD
$\mathbf{V}$	orthogonal matrix produced by SVD
$\widetilde{\mathbf{W}}$	low-rank weight matrix approximated by SVD
$\widetilde{\mathbf{\Sigma}}$	diagonal matrix produced by retaining the κ largest singular value elements in $\mathbf{\Sigma}$
$\widetilde{\mathbf{U}}$	orthogonal matrix produced by retaining only the κ column vectors of $\mathbf{U}$
$\widetilde{\mathbf{V}}$	orthogonal matrix produced by retaining only the κ column vectors of $\mathbf{V}$
$\mathbf{A}^{\mathrm{BTN}}_{l_{\mathrm{SD}}}$	bottleneck LTN's weight matrix, which is inserted to l_{SD}-th layer
$\mathbf{a}^{\mathrm{BTN}}_{l_{\mathrm{SD}}}$	bottleneck LTN's bias vector, which is inserted to l_{SD}-th layer
κ	bottleneck size

A.2 Multichannel End-to-end Speech Recognition (Section 4)

Basic indices	
t	input time step
n	output time step
l	subsampled time step
f	frequency index
c	channel index
T	length of input sequence
N	length of output sequence
L	length of subsampled input sequence
F	dimension of STFT feature
C	number of channels

Multichannel end-to-end ASR (Section 4.2 and 4.3)	
X_c	sequence of input STFT feature at c
$\mathbf{x}_{t,c}$	STFT feature vector at (t, c)
Y	sequence of output label symbol
y_n	label symbol at n
$\mathcal{V}$	set of label symbols
$\hat{X}$	sequence of enhanced STFT feature
$\hat{O}$	sequence of enhanced acoustic feature
E_{JOINT}	joint CTC-attention loss
E_{ATT}	attention loss
E_{CTC}	CTC loss
$\mathbf{X}$	set of multichannel STFT feature sequences
$P_{\mathrm{ATT}}(Y\|\mathbf{X})$	posteriors predicted by attention-based encoder-decoder network
$P_{\mathrm{CTC}}(Y\|\mathbf{X})$	posteriors predicted by CTC
γ	interpolation weight
$\hat{\mathbf{p}}_t$	enhanced power spectrum vector at t

$\hat{\mathbf{o}}_t$	enhanced acoustic feature vector at t
D_{O}	dimension of acoustic feature vector

Neural beamformers (Section 4.4)	
$\mathbf{x}_{t,f}$	vector of STFT coefficient at (t, f)
$x_{t,f,c}$	STFT coefficient at (t, f, c)
$\mathbf{g}_{t,f}$	vector of time-variant beamforming filter at (t, f)
$g_{t,f,c}$	time-variant filter coefficient at (t, f, c)
$\mathbf{g}_f$	vector of time-invariant beamforming filter at f
$g_{f,c}$	time-invariant filter coefficient at (t, f, c)
$\hat{x}_{t,f}$	enhanced STFT coefficient at (t, f)

Neural Beamforming with Filter Estimation Network (Section 4.4.2)	
$\tilde{X}$	sequence of BLSTM network's input
$\tilde{\mathbf{x}}_t$	input vector of BLSTM network at t
Z	sequence of output vector of BLSTM network
$\mathbf{z}_t$	output vector of BLSTM network at t
$\mathbf{W}_f^{\Re}$	weight matrix to output real part of filters at f
$\mathbf{b}_f^{\Re}$	bias vector to output real part of filters at f
$\mathbf{W}_f^{\Im}$	weight matrix to output imaginary part of filters at f
$\mathbf{b}_f^{\Im}$	bias vector to output imaginary part of filters at f
D_{Z}	dimension of BLTSM network's output

Neural Beamforming with Mask Estimation Network (Section 4.4.3)	
$\mathbf{\Phi}_f^{\mathrm{S}}$	PSD matrix for speech at f
$\mathbf{\Phi}_f^{\mathrm{N}}$	PSD matrix for noise at f
$\mathbf{u}$	reference microphone vector
$\mathbf{m}_t^{\mathrm{S}}$	vector of mean mask for speech at t
$\mathbf{m}_t^{\mathrm{S}}$	vector of mean mask for noise at t
$m_{t,f}^{\mathrm{S}}$	mean mask for speech at (t, f)

$m^{\mathrm{N}}_{t,f}$	mean mask for noise at (t, f)
$\mathbf{m}^{\mathrm{S}}_{t,c}$	vector of time-frequency mask for speech at (t, c)
$\mathbf{m}^{\mathrm{N}}_{t,c}$	vector of time-frequency mask for noise at (t, c)
$m^{\mathrm{S}}_{t,f,c}$	time-frequency mask for speech at (t, f, c)
$m^{\mathrm{N}}_{t,f,c}$	time-frequency mask for noise at (t, f, c)
$\tilde{X}_c$	sequence of BLSTM network's input at c
$\tilde{\mathbf{x}}_{t,c}$	input vector of BLSTM network at (t, c)
Z^{S}_c	sequence of BLSTM network's output for speech mask
$\mathbf{z}^{\mathrm{S}}_{t,c}$	output vector of BLSTM network for speech mask at (t, c)
Z^{N}_c	sequence of BLSTM network's output for noise mask
$\mathbf{z}^{\mathrm{N}}_{t,c}$	output vector of BLSTM network for noise mask at (t, c)
$\mathbf{W}^{\mathrm{S}}$	weight matrix to output speech mask
$\mathbf{b}^{\mathrm{S}}$	bias vector to output speech mask
$\mathbf{W}^{\mathrm{N}}$	weight matrix to output noise mask
$\mathbf{b}^{\mathrm{N}}$	bias vector to output noise mask
$\mathbf{q}_c$	time-averaged state feature
$\mathbf{r}_c$	PSD-based spatial feature
$\mathbf{w}$	weight vector for attention inner product
$\mathbf{b}$	bias vector for attention mechanism
$\mathbf{V}^{\mathrm{Q}}$	weight matrix for time-averaged state feature $\mathbf{q}_c$
$\mathbf{V}^{\mathrm{R}}$	weight matrix for PSD-based spatial feature $\mathbf{r}_c$
α	sharping factor
$\phi^{S}_{f,c,c'}$	entry in c-th row and c'-th column of PSD matrix for speech $\mathbf{\Phi}^{\mathrm{S}}_f$
D_{W}	dimension of attention inner product

Attention-based encoder-decoder networks (Section 4.5)	
$P(Y\|O)$	posteriors predicted by attention-based encoder-decoder network
O	sequence of input acoustic feature
$\mathbf{o}_t$	acoustic feature vector at t

H	sequence of encoder's output
$\mathbf{h}_l$	state vector of encoder's top layer at l
$\mathbf{c}_n$	context vector at n
$\mathbf{a}_n$	attention weight at n
$\mathbf{s}_n$	state vector of decoder's top layer at n
D_{H}	dimension of encoder's state vector
$\mathbf{f}_{n,l}$	location-based feature at (n, l)
$\mathbf{F}$	convolution filter for attention mechanism
$\tilde{\mathbf{w}}$	weight vector for attention inner product
$\tilde{\mathbf{b}}$	bias vector for attention mechanism
$\mathbf{V}^{\mathrm{S}}$	weight matrix for decoder's state $\mathbf{s}_n$
$\mathbf{V}^{\mathrm{H}}$	weight matrices for encoder's state $\mathbf{h}_l$
$\mathbf{V}^{\mathrm{F}}$	weight matrix for location-based feature $\mathbf{f}_{n,l}$
β	sharping factor
D_{V}	dimension of attention inner product
D_{S}	dimension of decoder's state vector
D_{F}	number of filters for attention mechanism
D_{f}	filter width for attention mechanism

Publication List

Journal Papers

1. **Tsubasa Ochiai**, Shigeki Matsuda, Hideyuki Watanabe, Xugang Lu, Chiori Hori, Hisashi Kawai, and Shigeru Katagiri, "Speaker Adaptive Training Localizing Speaker Modules in DNN for Hybrid DNN-HMM Speech Recognizers," *IEICE Transactions on Information and Systems*, vol. E99-D, no. 10, pp. 2431-2443, 2016.
2. **Tsubasa Ochiai**, Shinji Watanabe, Takaaki Hori, John R. Hershey, and Xiong Xiao, "A Unified Architecture for Multichannel End-to-End Speech Recognition with Neural Beamforming," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1274-1288, 2017.

Peer-Reviewed International Conference Proceedings

1. **Tsubasa Ochiai**, Shigeki Matsuda, Xugang Lu, Chiori Hori, and Shigeru Katagiri, "Speaker adaptive training using deep neural networks," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6349-6353, 2014.
2. **Tsubasa Ochiai**, Shigeki Matsuda, Hideyuki Watanabe, Xugang Lu, Chiori Hori, and Shigeru Katagiri, "Speaker adaptive training using deep neural networks embedding linear transformation networks," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4605-4609, 2015.
3. **Tsubasa Ochiai**, Shigeki Matsuda, Hideyuki Watanabe, Xugang Lu, Hisashi Kawai, and Shigeru Katagiri, "Bottleneck linear transformation network adaptation for speaker adap-

tation for speaker adaptive training-based hybrid DNN-HMM speech recognizer," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5015-5019, 2016.

4. **Tsubasa Ochiai**, Marc Delcroix, Keisuke Kinoshita, Atsunori Ogawa, Taichi Asami, Shigeru Katagiri, and Tomohiro Nakatani, "Cumulative Moving Averaged Bottleneck Speaker Vectors for Online Speaker Adaptation of CNN-based Acoustic Models," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5175-5179, 2017.
5. **Tsubasa Ochiai**, Shigeki Matsuda, Hideyuki Watanabe, and Shigeru Katagiri, "Automatic Node Selection for Deep Neural Networks Using Group Lasso Regularization," *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5485-5489, 2017.
6. **Tsubasa Ochiai**, Shinji Watanabe, Takaaki Hori, and John R. Hershey, "Multichannel End-to-end Speech Recognition," *International Conference on Machine Learning (ICML)*, pp. 2632-2641, 2017.
7. **Tsubasa Ochiai**, Shinji Watanabe, and Shigeru Katagiri, "Does Speech Enhancement Work with End-To-End ASR Objectives?: Experimental Analysis Of Multichannel End-To-End ASR," *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2017.
8. Mikiyo Kitaoka, Tetsuya Hashimoto, **Tsubasa Ochiai**, Shigeru Katagiri, Miho Ohsaki, Hideyuki Watanabe, Xugang Lu, and Hisashi Kawai, "Speech Pattern Classification Using Large Geometric Margin Minimum Classification Error Training," *IEEE Region 10 Conference*, pp. 1-6, 2015.

Non-Reviewed Domestic Workshop Proceedings

1. **Tsubasa Ochiai**, Hideyuki Watanabe, Shigeru Katagiri, Miho Osaki, Shigeki Matsuda, and Chiori Hori, "Experimental Study on Effect of Pre-training in Deep Learning through Visualization of Unit Outputs," *TECHNICAL REPORT OF IEICE*, vol. 113, no. 493,

pp.253-258, 2014 (in Japanese).

2. **Tsubasa Ochiai**, Shigeki Matsuda, Xugang Lu, Chiori Hori, and Shigeru Katagiri, "Unsupervised Speaker Adaptation by Speaker Adaptive Trained Deep Neural Networks," *2014 Spring Meeting of Acoustical Society of Japan (ASJ)*, 1-4-18, 2014 (in Japanese).
3. **Tsubasa Ochiai**, Shigeki Matsuda, Hideyuki Watanabe, Shigeru Katagiri, Miho Osaki, Xugang Lu, and Chiori Hori, "Analysis of DNN 's behavior based on inter-phoneme and inter-speaker variability in features internally represented in network," *2014 Autumn Meeting of Acoustical Society of Japan (ASJ)*, 1-R-6, 2014 (in Japanese).
4. **Tsubasa Ochiai**, Shigeki Matsuda, Hideyuki Watanabe, Xugang Lu, Chiori Hori, and Shigeru Katagiri, "Unsupervised Speaker Adaptation by Speaker Adaptive Trained Deep Neural Networks Embedding Linear Transformation Networks," *2015 Spring Meeting of Acoustical Society of Japan (ASJ)*, 1-P-26, 2015 (in Japanese).
5. **Tsubasa Ochiai**, Shigeki Matsuda, Hideyuki Watanabe, Xugang Lu, Hisashi Kawai, and Shigeru Katagiri, "Behavior Analysis of Speaker Adaptive Training-based DNN embedding Linear Transformation Network based on rank of weight matrix," *2015 Autumn Meeting of Acoustical Society of Japan (ASJ)*, 1-Q-3, 2015 (in Japanese).
6. **Tsubasa Ochiai**, Shigeki Matsuda, Hideyuki Watanabe, Xugang Lu, Hisashi Kawai, and Shigeru Katagiri, "Bottleneck Linear Transformation Network Adaptation for Speaker Adaptive Traind DNN," *2016 Spring Meeting of Acoustical Society of Japan (ASJ)*, 3-P-2, 2016 (in Japanese).
7. Gregoire Satre, **Tsubasa Ochiai**, Shigeru Katagiri, and Miho Ohsaki, "Autoencoders with Deformable Templates for Image Reconstruction," *IEICE General Conference*, ISS-P-117, p. 117, 2016.

Tutorial Papers

1. **Tsubasa Ochiai**, Tatsuya Matsuhiro, Shigeki Matsuda, and Shigeru Katagiri, "Introduction to Deep Learning Toolkit for Speech and Language Processing," *The Journal of the*

Acoustical Society of Japan, vol. 1, no. 37, pp. 63-72, 2017 (in Japanese).